CAHIER DU LAMSADE

Laboratoire d’Analyse et Modélisation de Systèmes pour l’Aide à la Décision
(Université Paris-Dauphine)
Unité de Recherche Associée au CNRS n° 825

EXACT AND APPROXIMATION RESULTS ON MAXIMUM INDEPENDENT SET AND MINIMUM VERTEX COVERING – GRAPHS WITH GREAT STABILITY NUMBER

CAHIER N° 128
juin 1995

Marc DEMANGE ¹
Vangelis PASCHOS ¹


¹ LAMSADE, Université Paris-Dauphine, Place du Maréchal De Lattre de Tassigny, 75775 Paris Cedex 16, France.
Contents

I A polynomial algorithm for the maximum independent set problem on a class of graphs including König-Egerváry graphs 2

1 From independent set to matching 2

2 A characterization of König-Egerváry graphs in terms of linear programming 3

3 A generalization of the König-Egerváry graphs 6

4 König-Egerváry graphs revisited 11

5 Some independent set approximation and exact polynomial results “inspired” from König-Egerváry graphs 11

II The approximability behaviour of some combinatorial problems with
Résultats exacts et approchés sur les problèmes du stable maximum et de la couverture minimum de sommets - graphes à grand nombre de stabilité

Résumé

Dans la première partie de cet article, nous présentons un algorithme exact polynomial pour le problème du stable maximum dans la classe des graphes incluant les graphes de König-Egervary. Tout graphe $G$ d’ordre $n$ appartenant à la classe des graphes de König-Egervary est tel que $\alpha(G) = n - m$ où $\alpha$ est son nombre de stabilité et $m$ la cardinalité d’un couplage maximum de $G$.

Nous prouvons ensuite que l’existence d’un algorithme polynomial $\rho$-approché (où $\rho < 1$ est une constante fixée) pour une classe de problèmes du stable maximum conduit à un algorithme polynomial approché avec un rapport d’approximation strictement plus petit que 2 pour la couverture de sommets alors que la non-existence d’un tel algorithme induit une
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-Egerváry graphs. The class of König-Egerváry 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 on graphs admitting a perfect matching achieves an approximation ratio strictly greater than $2/\Delta$, where $\Delta$ is the maximum degree of the vertices of the graph.

Keywords: NP-complete problem, polynomial time approximation algorithm, vertex covering, independent set.
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 each edge \( uv \in E \), at least one of \( u, v \) belongs to \( C \). In a graph \( G \) a (minimum) vertex covering is the complement, with respect to the vertex-set of \( G \), of a (maximum) independent set; in what follows, we shall denote by \( \alpha(G) \) and \( \tau(G) \) the cardinalities of a maximum independent set and a minimum vertex covering, respectively; so, \( \alpha(G) = n - \tau(G) \).

A matching is a subgraph of \( G \), all of its vertices having degree equal to one, and a maximum matching is the maximum order such subgraph. The size (number of edges) of any matching of a graph is at most the size of any vertex cover of a graph. A matching is called perfect if its cardinality equals the half of the order of \( G \).

A graph of order \( n \) is called König-Edgervay (KE) if \( \alpha(G) = n - m \), where \( m \) is the cardinality of a maximum matching of \( G \). This condition is, obviously, equivalent to the condition \( \tau(G) = m \).

The underlying optimization problems, with respect to the notions of independent set, vertex
Part I
A polynomial algorithm for the maximum
have their respective constraint sets non empty, they have the same optimal value.
So, the following inequalities hold: \( v(S) \leq v(SR) = v(ECR) \leq v(EC) \).

One can construct an identical schema concerning VC expressed in terms of a (0,1) linear program as follows:

\[
\begin{align*}
\text{VC} = \left\{ \begin{array}{l}
\min \quad \vec{1}_n \cdot \vec{x} \\
A \cdot (\vec{1}_n - \vec{x}) \leq |E| \\
\vec{x} \in \{0,1\}^n
\end{array} \right. \iff \text{VC} = \left\{ \begin{array}{l}
\min \quad \vec{1}_n \cdot \vec{x} \\
A \cdot \vec{x} \geq A \cdot \vec{1}_n - |E| = |E| \\
\vec{x} \in \{0,1\}^n
\end{array} \right.
\]

where the constraints of the lefthand side of the equivalence mean that the complement of a vertex covering is an independent set. The equality \( A \cdot \vec{1}_n - |E| = |E| \) is due to the form of the matrix \( A \).

We consider now the relaxed version VCR of VC:

\[
\text{VCR} = \left\{ \begin{array}{l}
\min \quad \vec{1}_n \cdot \vec{x} \\
A \cdot \vec{x} \geq |E|
\end{array} \right. \]

Its dual, denoted by MR, is written as:

\[
\text{MR} = \left\{ \begin{array}{l}
\max \quad |E| \cdot \vec{x} \\
A^T \cdot \vec{x} \leq \vec{1}_n \quad \vec{x} \geq \vec{0}_n
\end{array} \right.
\]

MR is the relaxed version of the maximum matching problem M defined as follows:

\[
\text{M} = \left\{ \begin{array}{l}
\max \quad |E| \cdot \vec{x} \\
A^T \cdot \vec{x} \leq \vec{1}_n \\
\vec{x} \in \{0,1\}^{|E|}
\end{array} \right.
\]

As previously, we have \( v(M) \leq v(MR) = v(VCR) \leq v(VC) \).

Note that given a graph \( G \) of order \( n \), there is a well-known relation between \( v(M) \) and \( v(EC) \): \( v(M) = n - v(EC) \). On the other hand, a similar relation holds between \( S \) and VC: \( v(S) = n - v(VC) \).

So, the following schema summarizes the above discussion:

\[
\begin{align*}
v(S) & \leq v(SR) = v(ECR) \leq v(EC) \\
n - v(VC) & \leq n - v(VCR) = n - v(MR) \leq n - v(M)
\end{align*}
\]

Recall that the KE graphs are defined as the graphs where \( \alpha(G) = v(S) = n - v(M) \); using the above relations, we can give alternative expressions like \( \alpha(G) = v(S) = v(EC) \), or \( \tau(G) = v(M) \).
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 \((\tilde{x}, \tilde{y}) \in \mathbb{R}^n \times \mathbb{R}^{\lvert E \rvert}\); then, the fact that \( \tilde{x} \) is a solution of SR and, simultaneously, that \( \tilde{y} \) is a solution of ECR, is equivalent to the following conditions:

\[
\begin{align*}
A \cdot \tilde{x} & \leq \bar{1}_{\lvert E \rvert} \quad \text{(i)} \\
\tilde{x} & \geq \bar{0}_n \quad \text{(ii)} \\
AJ \cdot \tilde{y} & \geq \bar{1}_n \quad \text{(iii)} \\
\tilde{y} & \geq \bar{0}_{\lvert E \rvert} \quad \text{(iv)} \\
\tilde{x}_i > 0 \implies \sum_{j \in \text{adj}(i)} \tilde{y}_j = 1 \quad \text{(v)} \\
\tilde{y}_j > 0 \implies \sum_{i \in \text{extr}(j)} \tilde{x}_i = 1 \quad \text{(vi)}
\end{align*}
\]

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 \( \hat{S} \) 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 \( B(\hat{S}) = (\hat{S}, V \setminus \hat{S}, E') \), with \( e = xy \in E' \) if and only if \( x \in \hat{S} \) \((y \notin \hat{S})\), or \( x \in \hat{S} \) and \( e \in E \). The edge-vertex incidence matrix \( A' \) of \( B(\hat{S}) \) is obtained from \( A \) by deleting the rows of \( A \) corresponding to edges linking vertices of \( V \setminus \hat{S} \). It is trivial to verify that \( \hat{S} \) remains an independent set in \( B(\hat{S}) \), and since it "touches" all the edges of \( E \) having in \( \tilde{y} \) non-zero values, these edges are, by definition, contained also in \( E' \); so, the projection \( \tilde{y'} \) of \( \tilde{y} \) in \( \mathbb{R}^{\lvert E \rvert} \) (we retain the components corresponding to an edge of \( E' \)) constitutes a feasible solution of ECR in \( B(\hat{S}) \) and, moreover, the pair \((\tilde{x}, \tilde{y'})\) satisfies the optimality conditions \((i) \div (vi)\) in \( B(\hat{S}) \); so, \( \hat{S} \) is a maximum independent set for \( B(\hat{S}) \). Also, every bipartite graph (consequently \( B(S) \)) is KE ([3]); hence, we deduce that \( v(S_B(\hat{S})) = v(\text{EC}_B(\hat{S})) = v(S_G) \), where we use indices to denote the graphs on which the instances of the problems are defined. On the other hand, \( v(\text{EC}_B(\hat{S})) \geq v(\text{EC}_G) \) because every feasible solution for EC in \( B(\hat{S}) \) implies a feasible solution for EC in \( G \) by considering the same set of edges as solution in both instances; so, \( v(S_G) \geq v(\text{EC}_G) \); moreover, the opposite inequality is always satisfied as we have already seen above; so \( v(S_G) = v(\text{EC}_G) \) and \( G \) is a KE graph.

Let us notice here that the equality \( v(\text{EC}) = v(\text{ECR}) \) is not always true. In fact, the difference \( v(\text{EC}) - v(\text{ECR}) \) can be arbitrarily large. Let us consider the following sequence \( G_p, p \in \mathbb{N}^* \), of graphs: \( p \) triangles and a vertex \( x_p \) linked to a (arbitrary) vertex of every triangle by an edge. The graph \( G_p \) is of order \( 3p + 1 \); the value (maximum cardinality) of a matching is \( p + 1 \) (we form such a matching by taking an edge per triangle and an edge incident to \( x_p \)). We have then \( v(\text{EC}) = 2p \); moreover, by assigning the value \( 1/2 \) to every edge of the triangles and the value \( 1 \) to all edges incident to \( x_p \), we construct a feasible solution, of value \((3p/2) + 1\), for ECR; so, \( v(\text{EC}) - v(\text{ECR}) \geq (p/2) - 1 \rightarrow p \rightarrow +\infty \).

On the other hand, let us consider a \( K_{3p-1} \in \mathbb{N}^* \). The size of a maximum matching...
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:

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

where \( A \) is the vertex-vertex incidence matrix of the graph-instance \( G \) of HC and \( \bar{z}_i, i = 1, \ldots, n \), 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 \( \bar{z}_i, i = 1, \ldots, n \), represents one or zero vertices; constraint (i) signifies that if \( x_{i+1} \) and \( x_i \) 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 right-hand 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 \((\bar{z}_1, \ldots, \bar{z}_n)\) is of the form \((\bar{z}_1, \ldots, \bar{z}_k, \bar{I}_n, \ldots, \bar{I}_n)\), where \( k \in \{0, \ldots, n\} \) and \( \bar{z}_i, i \leq k \), represents a vertex; so, in the case where \( k \neq 0 \), the sequence \((\bar{z}_1, \ldots, \bar{z}_n)\) 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.

Let us now relax constraint (v) by transforming it into \( \bar{z}_i \geq \bar{I}_n \), \( i = 1, \ldots, n \). The resulting program has now an optimal value less than or equal to \( n \) because of constraint (iii); in fact, its optimal value is equal to \( n \) since the solution obtained by setting \( \bar{z}_i = (1/n)\bar{I}_n, i = 1, \ldots, n \), is feasible or, equivalently, \( G \) is Hamiltonian and program HC has the same optimal value than its relaxed version.

This completes the proof of proposition 1.\(^{11}\)

\(^2\)To determine the stability number in a KE graph, one has only to solve SR by a real-linear programming method (in \( O(n^3) \)) or, better, to determine a maximum matching (of cardinality \( m \)) in the graph and to compute the quantity \( n - m \) (because of the equality \( v(S) = v(EC) \) and the fact that EC is polynomial (\( [3] \))).

\(^3\)In [6], HC is defined as follows: "given a (non-directed) graph \( G \) of order \( n \), does \( G \) contain a Hamiltonian circuit, that is, an ordering \((v_1, v_2, \ldots, v_n)\) of the vertices of \( G \), such that \( v_i v_{i+1} \in E \) and \( v_i v_{i+1} \in E \) for all \( i, 1 \leq i \leq n \)?"
3 A generalization of the König-Egerváry 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 $S_\bar{\alpha}$ a general instance of the weighted version of $S$ represented by a graph $G = (V, E)$ of order $n$ and by a vector $\bar{\alpha} \in (\mathbb{R}^{n+})$, the components of which being the weights of $V$. So, $S_\bar{\alpha}$ can be described by the following integer-linear program:

$$S_\bar{\alpha} = \left\{ \begin{array}{ll}
\max & \bar{\alpha} \cdot \bar{x} \\
& A \cdot \bar{x} \leq \mathbf{1}_{|E|} \\
& \bar{x} \in \{0, 1\}^n
\end{array} \right.$$

where $A$ is the edge-vertex incidence matrix of $G$.

The relaxed version $SR_\bar{\alpha}$ of $S_\bar{\alpha}$ is defined by

$$SR_\bar{\alpha} = \left\{ \begin{array}{ll}
\max & \bar{\alpha} \cdot \bar{x} \\
& A \cdot \bar{x} \leq \mathbf{1}_{|E|} \\
& \bar{x} \geq \mathbf{0}_n
\end{array} \right.$$

and the dual $ECR_\bar{\alpha}$ of $SR_\bar{\alpha}$ is finally defined as follows:

$$ECR_\bar{\alpha} = \left\{ \begin{array}{ll}
\min & \mathbf{1}_{|E|} \cdot \bar{y} \\
& A^T \cdot \bar{y} \geq \bar{\alpha} \\
& \bar{y} \geq \mathbf{0}_{|E|}
\end{array} \right.$$

In what follows, we treat the class $G_\bar{\alpha}$ of graphs for which $v(S_\bar{\alpha}) = v(SR_\bar{\alpha})$; it is easy to see that, by theorem 1, the class $G_{\mathbf{1}_n}$ is exactly the class of the KE graphs.

We present a polynomial time algorithm (algorithm 1), optimally solving the maximum independent set problem in the class $G_\bar{\alpha}$. We suppose that the input of the algorithm is a graph $G = (V, E)$ known to be in $G_\bar{\alpha}$; later, we shall show how the same algorithm can be applied to every graph in order to decide if its input belongs to $G_\bar{\alpha}$.

A first step of this algorithm is the construction of a solution $\bar{y}$ of $ECR_\bar{\alpha}$; we do not detail on this step since $ECR_\bar{\alpha}$ is a linear problem on real numbers which can be solved by a linear programming method in $O(n^3)$ (see for example [2])4. Let us denote by $C$ the set $\{e \in E : \tilde{y}_e > 0\}$; also, given a set $P \subset E$, we denote, $\forall v \in V, v \in P \Rightarrow \Gamma_P(v)$ the set $\{u \in V : uv \in P\}$; of course, $\Gamma_E(v) = \Gamma(v)$ (the neighbour set of $v$ in $G$). Finally, given a vertex-set $V' \subset V$, we denote by $\Gamma(V')$ the set of neighbours of the vertices of $V'$. Algorithm 1 calls the recursive procedure $LABEL$ and the function $TEST$. Procedure $LABEL$ has as arguments the "current graph" $G_r = (V_r, E_r)$ (a sub-graph of $G$), some vertices of which being labelled by $s$ and some other ones by $c$ (we denote by $LV_r$ the set of the labelled vertices of $G_r$) and the set $C$; $LABEL$ completes, while this is possible, the vertex labelling by respecting the following two rules: (i) if a vertex is labelled by $s$, then $LABEL$ labels its non-labelled neighbours by $c$, and (ii) if a vertex $v$ is labelled by $c$, then $LABEL$ labels the non-labelled members of $\Gamma_C(v)$ by $s$. Function $TEST$ has as arguments the ones of $LABEL$ plus a solution $\tilde{y}_r$ of the problem $ECR_\bar{\alpha}$, instance of which is the graph $G_r$; this function tests if the set of vertices labelled by $s$ forms an independent set of value $\tilde{d}_{|E|} \cdot \tilde{y}_r$.

Theorem 2. Algorithm 1 is an $O(n^3)$ exact maximum-weight independent set algorithm for the class $G_\bar{\alpha}$, $\bar{\alpha} > \mathbf{0}_n$.

4In the case where $\bar{\alpha} = \mathbf{1}_n$, since $ECR$ and $EC$ have the same optimal value, following the discussion in section 2, one can determine a solution $\bar{y} \in \{0, 1\}^{|E|}$ starting from a maximum matching on $G$. 

6
begin
  \( S \leftarrow \emptyset; \)
  \( G_r \leftarrow G; \)
  find a solution \( \tilde{y} \) of \( \text{ECR}_g \) 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;
    \( \ell(x_0) \leftarrow c; \)
    \( LV_r \leftarrow \{x_0\}; \)
    \( \text{LABEL}(G_r, C, LV_r); \)
    if \( \neg \text{TEST}(G_r, C, LV_r, \tilde{y}_r) \) then
      begin
        \( \ell(x_0) \leftarrow s; \)
        \( LV_r \leftarrow \{x_0\}; \)
        \( \text{LABEL}(G_r, C, LV_r) \)
      end;
    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 \( \nu \rightarrow \{c, s\} \).

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

Procedure 1. LABEL.
begin
TEST ← true
if \( \sum_{v : \ell(v) = s} a_v \neq \tilde{a}_s' \cdot \tilde{y}_s \) 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 \( |\Gamma_{E_r}(v)| \neq 0 \) then TEST ← false
end;

Function 1. TEST.

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

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

Lemma 1. The graph \( G^k_r \) is a subgraph of \( G \) belonging to \( \mathcal{G}_d \); moreover, for \( k < K \), a maximum-weight independent set \( S^k_r \) of \( G^k_r \) can be obtained by setting \( S^k_r = \{ x \in LV^k_r : \ell(x) = s \} \cup S^{k+1}_r \), where \( S^{k+1}_r \) is a maximum-weight independent set of \( G^{k+1}_r \) and \( \ell \) is the mapping \( V^k_r \to \{ e, s \} \) computed by algorithm 1.

Proof: (lemma 1.) Let us prove by induction that \( G^k_r \in \mathcal{G}_d \).
First, we shall prove that, \( \forall k, G^k_r \) satisfies the property: \( \Gamma_C(v) \subseteq V^k_r, \forall v \in V^k_r \). For \( k = 1 \), since \( G^1_r = G \) and \( C \subseteq E \), the property is obvious. Let us suppose the truth of the property for \( k < K \) (we recall that algorithm stops after the \( K \)th iteration). The set \( LV^k_r \) 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 \( v \in V^k_r \setminus LV^k_r \), then \( \Gamma_C(v) \subseteq V^k_r \setminus LV_r = V^{k+1}_r \) 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 \( G^k_r \) is in \( \mathcal{G}_d \), we shall prove the following lemma 2.

Lemma 2. Let \( G = (V, E) \) be a graph of the class \( \mathcal{G}_d \), \( \tilde{y} \) be an optimal solution of \( \text{ECR}_d \) 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, \( \tilde{G} \in \mathcal{G}_d \), moreover, in this case, the tracks of \( \tilde{y} \) and \( \tilde{x} \) on \( \tilde{V} \) are solutions of \( \text{ECR}_{\tilde{G}} \) and \( \text{SR}_d \) on \( \tilde{G} \), respectively.

Proof: (lemma 2.) Let \( \tilde{x} \) be an optimal solution of \( \text{SR}_d \) and, consequently, of \( \text{SR}_d \) (\( G \in \mathcal{G}_d \)); let us rewrite the primal-dual optimality conditions for the dual linear programs \( \text{SR}_d \) and \( \text{ECR}_d \): let \( (\tilde{x}, \tilde{y}) \in \mathbb{R}^n \times \mathbb{R}^{|E|} \); then, the fact that \( \tilde{x} \) is a solution of \( \text{SR}_d \) and \( \tilde{y} \) is a solution of \( \text{ECR}_d \) is
equivalent to the following conditions:

$$
\begin{align*}
\begin{cases}
A \cdot \tilde{x} & \leq \mathbb{1}_{|E|} \quad \text{(i)} \\
\tilde{x} & \leq \mathbb{0}_{n} \quad \text{(ii)} \\
AT \cdot \tilde{y} & \geq \tilde{a} \quad \text{(iii)} \\
\tilde{y} & \geq \mathbb{0}_{|E|} \quad \text{(iv)} \\
x_{i} > 0 & \Rightarrow \sum_{j \in \text{adj}(i)} \tilde{y}_{j} = a_{i} \quad \text{(v)} \\
\tilde{y}_{j} > 0 & \Rightarrow \sum_{i \in \text{extr}(j)} \tilde{x}_{i} = 1 \quad \text{(vi)}
\end{cases}
\end{align*}
$$

where $\text{adj}(i)$, $\text{extr}(j)$ 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 $\tilde{x} \in \mathbb{R}^{|\tilde{V}|}$ such that, $\forall v \in \tilde{V}$, $\tilde{x}_{v} = \tilde{x}_{v}$ and $\tilde{y} \in \mathbb{R}^{|\tilde{E}|}$, such that $\forall e \in \tilde{E}$, $\tilde{y}_{e} = \tilde{y}_{e}$ the projections of $\tilde{x}$ and $\tilde{y}$ on the characteristic spaces of $\tilde{V}$ and $\tilde{E}$, respectively; we also define $\tilde{a} \in \mathbb{R}^{|\tilde{V}|}$ such that, $\forall v \in \tilde{V}$, $\tilde{a}_{v} = a_{v}$, the projection of vector $\tilde{a}$ (it is easy to see that $\tilde{x} \in \{0,1\}^{|\tilde{V}|}$).

We shall show that the pair $(\tilde{x}, \tilde{y})$ 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 $A$.

In fact, $\tilde{x}$ is the characteristic vector of an independent set of $\tilde{G}$ because $\tilde{x}$ corresponds to an independent set of $G$ and, moreover, $\tilde{G}$ is a sub-graph of $G$; so, $\bar{A} \cdot \tilde{x} \leq \mathbb{1}_{|E|}$ and $\tilde{x} \geq \mathbb{0}_{|\tilde{V}|}$ (conditions (i) and (ii)). It is also easy to obtain $\tilde{y} \geq \mathbb{0}_{|\tilde{E}|}$ (condition (iv)). Moreover, vector $\tilde{y}$ satisfies $\bar{A} \cdot \tilde{y} \geq \tilde{a}$ (condition (iii)). Plainly, let us consider one of these constraints; it corresponds to a vertex $v \in \tilde{V}$; let an edge $e \in E$ 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 $\tilde{G}$ 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 $\tilde{x}_{v} > 0$, 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 $G$...
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 \( z_0 \) and no label to any other vertex (see algorithm 1) is a "good labelling", then the labelling completing this one is also a "good labelling". In this case, the vertices labelled by \( s \) constitute, in \( G^b_r \), the track of a maximum-weight independent set of \( G \), hence a weighted independent set of value \( \bar{d}_{|E|} \cdot L_{\bar{y}} \), this fact being tested by function TEST. Consequently, since \( LG^b_r \in G^b \), 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 \( LG^b_r \). If this is not true, then the initial labelling was not a "good labelling", i.e., \( z_0 \) belongs to every maximum-weight independent set. So, the labelling assigning to \( z_0 \) 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 \( LG^b_r \). 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 \( LG^b_r \) of value \( \bar{d}_{|E|} \cdot L_{\bar{y}} \). On the other hand, if \( k < K \), a maximum-weight independent set of \( G^b_r \) has value \( \bar{d}_{|E|} \cdot L_{\bar{y}} \); so, since the disjoint union of \( LV^b_r \) and \( V^{b+1}_r \) equals \( V^b_r \), the union of these weighted independent sets (in \( G^b_r \) and \( C^b_r \)) has value \( \bar{d}_{|E|} \cdot L_{\bar{y}} + \bar{d}_{|E|} \cdot L_{\bar{y}} \), which is the value of a maximum independent set in \( V^b_r \). So, to prove that this union constitutes a maximum-weight independent set for \( G^b_r \), it suffices to show that this union constitutes an independent set for \( C^b_r \). 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 \( LV^b_r \)), the vertices labelled by \( s \) at the end of iteration \( k \) are not linked, in \( G \), to any vertex of the set \( V^{b+1}_r \); so, the union of an independent set of \( LG^b_r \) with an independent set of \( G^b_r \) constitutes an independent set of \( G^b_r \).

This completes the proof of lemma 1. 

We are well prepared now to conclude the proof of the theorem.

If algorithm 1 does not stop at step \( k \), then, since \( G^b_r \in G^b \), the arguments developed previously remain valid for \( G^{b+1}_r \). Moreover, while \( |V(G_r)| \neq 0 \), a new execution of the while loop is always possible and produces a set \( LV^b_r \) of size greater than 2, since if \( LV^b_r \) contains a vertex \( v \), then it contains \( \Gamma_C(v) \), this set being of size greater than 1, because \( C \) is an edge covering of \( G \); consequently, the number of the total while calls is smaller than \( n/2 \) and the convergence of the algorithm is concluded. At the end of the \( K \)th (last) iteration, \( LV^b_r = V^b_r \); so, vertices of \( V^b_r \) labelled by \( s \) constitute a maximum-weight independent set of \( G^b_r \); lemma 1 and 2 allow then to verify immediately, by means of an easy backward induction, that algorithm 1 produces a maximum-weight independent set of \( G \).

Concerning the complexity of algorithm 1, as we have already mentioned, obtaining a solution \( \bar{y} \) for ECR in \( G \) is performed in \( O(n^3) \); on the other hand, we notice that during an iteration of the while loop, procedure LABEL takes time of \( O(|E|) \) and since \( k \leq n/2 \), this results to a total time (for the while loop) of \( O(n^3) \). The two described operations being independent, the total time complexity of algorithm 1 is of \( O(n^3) \) and this completes the proof of the theorem. 

**Theorem 3.** Algorithm 1 decides in \( O(n^3) \) if a given graph \( G \) belongs to \( G \).

**Proof:** It suffices to apply algorithm 1 on a given graph \( G \); algorithm stops in \( O(n^3) \) steps providing a set of vertices labelled by \( s \); one can verify then if these vertices form an independent set of total weight \( \bar{d}_{|E|} \cdot \bar{y} \), this verification taking time linear to \( |E| \). If this is the case, then \( G \in G^b \), while in the opposite case \( G \notin G^b \), since from theorem 2, if \( G \in G^b \), then algorithm 1 determines a solution of optimal value \( \bar{d}_{|E|} \cdot \bar{y} \).
4 König-Egerváry graphs revisited

In the case where \( d = I_n \), the results of section 3 mean that one can decide, in polynomial time, if a given graph \( G \) is K.E. and, if this is the case, determine always in polynomial time a
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 $\lceil n/(2(\kappa + 1)) \rceil - 2$.

Proof:
(i) The condition $f + g \leq \kappa$ implies that both $f$ and $g$ are bounded above by $\kappa$. So, for all integer $k \in \{0, \ldots, \min\{n - 2m, \kappa - h\}\}$, for all $h$-tuple $H$ of matching edges and for all $k$-tuple $K$ of exposed vertices of $V$ with respect to $M$, we form the sub-graphs of $G$ induced by the vertex set $V \setminus (K \cup T[H])$ where, by $T[H]$, we denote the endpoints of the edges of the set $H$.

We next apply the simplified version of algorithm 1, discussed in section 4, on all of the so-obtained sub-graphs of $G$. Then, for one of the pairs $(H, K)$, the induced subgraph is KE and, moreover, in this graph, a maximum independent set is identical to the maximum independent set of $G$. Hence, the maximum cardinality set between the so-obtained independent sets is a maximum independent set of $G$.

Given that $\kappa$ is a universal constant, there is a polynomial number of pairs $(H, K)$ and, consequently, the whole of the described process remains also polynomial.

(ii) Obviously, $f \leq \kappa$ and $g \leq \kappa$. We arbitrarily partition the edges of $M$ into $\kappa + 1$ sets $M_1, \ldots, M_{\kappa+1}$, where $|M_i| = \lfloor m/(\kappa + 1) \rfloor$, $i = 1, \ldots, \kappa$ and $M_{\kappa+1} = m - \sum_{i=1}^{\kappa} \lfloor m/(\kappa + 1) \rfloor$. We also arbitrarily partition the set $X$ of the exposed vertices of $G$ into $\kappa + 1$ sets $X_1, \ldots, X_{n+1}$ sets, each of size at least $\lfloor (n - 2m)/(\kappa + 1) \rfloor$. We so obtain $(\kappa + 1)^2$ sub-graphs of $G$ ($\kappa \leq m \leq n/2$), each one induced by the vertex-set $X_i \cup T[M_j]$ (where, as previously, by $T[M_i]$, we denote the set of the endpoints of the edges in $M_j$), $(i, j) \in \{1, \ldots, \kappa + 1\}^2$ (Cartesian square). So, we can apply simplified version of algorithm 1, discussed in section 4, on all of the so-obtained $(\kappa + 1)^2$ graphs. Since at least one of these graphs is KE, one of the obtained solutions will be of size at least $(n - 2m)/(\kappa + 1) + (m)/(\kappa + 1) - 2 = (n - m)/(\kappa + 1) - 2$.

The minimum of this function, for $m \in [0, n/2]$, is obtained for $m = n/2$ and its value is, in this case, equal to $n/[2(\kappa + 1)] - 2$.

Corollary 1. Given a fixed positive constant $\kappa$, deciding if a graph $G$ satisfies $0 \leq f + g \leq \kappa$ is polynomial.
Part II
The approximability behaviour of some combinatorial problems with respect to the approximability of a class of maximum independent set problems

6 Minimum vertex covering and maximum independent set

Proposition 3. There exists a $\kappa_0$ such that, for every fixed constant $\rho < 1$, there is no polynomial time approximation algorithm for $S_{\kappa_0}$, guaranteeing approximation ratio greater than or equal to $\rho$.

Unfortunately, up to now, we do not precisely know the value of $\kappa_0$. However, we shall see at the end of this section that $\kappa_0 \geq 2$. In any case, $S_\kappa$ ($\kappa \geq 2$), in particular for small values of $\kappa$, seems to have interesting properties, since it interferes with VC (as well as with some interesting mathematical programming problems, as we shall see in the next section) and its approximability behaviour.

In this section, we examine how the approximability of $S_3$ influences the one of VC. We prove some conditional results which show how the existence of a polynomial time approximation algorithm for $S_3$ would imply the existence of a polynomial time approximation algorithm for VC guaranteeing an approximation ratio strictly smaller than 2, while the non-existence of such an algorithm induces a lower bound for the ratio of VC.

In what follows, given a graph $G = (V,E)$ of order $n$, we denote by $\tau(G)$ (resp. $\tau'(G)$) and $\alpha(G)$ (resp. $\alpha'(G)$) the cardinality of the minimum (resp. approximate) vertex covering and the stability (resp. approximate stability) number of $G$, respectively. Also, given a set of vertices $A$, we denote by $T[A]$ the set of the endpoints of the edges of $A$; given a set $V' \subseteq V$ of vertices of $G$, by $G[V']$ we shall denote the subgraph of $G$ induced by $V'$. Given a maximum matching $M$ and an edge $uv \in M$, vertices $u$ and $v$ are called mates and we will denote vertex $v$ (resp. vertex $u$) by $m(u)$ (resp. $m(v)$; $m$ stands for mate); given a vertex $x$, exposed with respect to $M$, we denote by $M[x]$ the subset of $M$ such that, for any edge $e \in M[x]$, at least one endpoint of $e$ is adjacent to $x$. In the following, we shall denote by $E(G)$ (resp. $V(G)$) the edge (resp. vertex) set of $G$.

6.1 The non-constant-approximability of $S_3$ would induce a lower bound for vertex cover’s approximation ratio

Proposition 4. If $S_3$ is not constant-approximable in polynomial time, then there cannot exist a polynomial time approximation algorithm for VC guaranteeing approximation ratio strictly smaller than $3/2$.

Proof: Let us suppose that VC admits a polynomial time approximation algorithm $A$ with an
\[
\tau'(G) \leq \left(1 - \frac{2}{3} \epsilon \right) n
\]

and hence one can obtain immediately an independent set on \( G \) of cardinality
\[
\alpha'(G) = n - \tau'(G) \geq \frac{2}{3} \epsilon n.
\]

Consequently, \( A \) (provided with a set-difference instruction) constitutes a polynomial algorithm for \( S_2 \), always guaranteeing a ratio \( \alpha'(G)/\alpha(G) \geq \alpha'(G)/n \geq (2/3) \epsilon \), this ratio being a (universal) constant. So, on the hypothesis that \( S_2 \) is not constant-approximable in polynomial time, such a polynomial time approximation algorithm \( A \) cannot exist for VC (unless \( P = NP \)). \]

6.2 The constant-approximability of \( S_2 \) 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 \( A \) for \( S_2 \) with a fixed positive constant approximation ratio\(^7\) \( \rho \). 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 ([7]; 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 \( A \), and then taking the complement of a the solution provided by \( A \).

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 \( S \)

\(^7\)We suppose that whenever \( A \) operates on graphs \( G \) which are not instances of \( S_2 \), it stops in polynomial time, delivering either non-feasible solutions or maximal independent sets of cardinality smaller than \( \rho n/3 \).
begin
apply A on G to obtain an independent set solution S;
C ← V \ S
end;

Procedure 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 \( T[M_1] \cap S = \{ e_1, \ldots, e_m \} \) and \( c_l = m(s_l), i = 1, \ldots, m_1 \); 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 \( E_2 \).

**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) \( v \notin S_1 \cap S \).

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 \( s \in S_1^j \), vertex \( m(s) \in C_1^j \). 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 \( S_1^j \) during the \( k \)th execution of line (25); let us denote \( S_1^j(k) \) and \( C_1^j(k) \), the subsets of \( S_1^j \) and \( C_1^j \), respectively, resulting from the insertion of \( s_k \) in \( S_1^j \). 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 \( S_1^j(0) \) and \( C_1^j(0) \), the non-empty subsets of \( S_1^j \) and \( C_1^j \) resulting from the last execution of lines (19), or (22).

Let us now show by induction on \( k \in \{0, \ldots, N\} \) that: (a) \( v \in S_1^j(k) \cup C_1^j(k) \), \( m(v) \in \delta_1^j(k) \cup C_1^j(k) \) and (b) for every vertex \( s \in S_1^j(k) \), there exists an alternating path from \( s \) to \( x_l \) starting from a matched edge and exclusively containing vertices of \( S_1^j(k) \cup C_1^j(k) \).

Proof. For \( k = 0 \), (ii) immediately result from the discussion of case (j).
property (b) on range $k + 1$, it suffices to consider the case where $s_{k+1} \notin C_4^1$ (the opposite case being treated in case (i)); then, from line (23) of procedure 4, $m(s_{k+1}) \notin C_4^1(k)$ and, also, there exists an alternating path $\mu$ from $s$ to $x_1$ exclusively containing vertices from $S_1^1(k) \cup C_4^1(k)$; on the other hand, from the inductive hypothesis on property (a), $\lbrace s_{k+1}, m(s_{k+1}) \rbrace \notin S_1^1(k) \cup C_4^1(k)$ and, moreover, $(S_1^1(k) \cup C_4^1(k)) \cap \lbrace s_{k+1}, m(s_{k+1}) \rbrace = \emptyset$; 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 $\lbrace c_3, c_4, c_5, c_6 \rbrace$ 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 $\mu = s_6 - c_6 - s_5 - c_5 - s_4 - c_4 - s_3 - c_3 - s_2 - c_2 - s_1 - x$; 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 $\lbrace x_i, x_j \rbrace \subseteq X, x_i \neq x_j$, with $\lbrace ux_i, vx_j \rbrace \subseteq E$.

**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:** Concerning 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 $n^0$ (where $n$ is the order of $G$).
Consider now the set $C$ created during the execution of the \textbf{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 $uu$, where $u \in \mathcal{V}$. 


begin
    call algorithm 1 on G
    if G is KE then find and store the complement of the provided independent set
    call procedure 2 and store the obtained solution for VC;
    compute a maximum matching $M$ in $G$
    if $M$ is perfect then call procedure 3 and store the obtained solution for VC
    else call procedure 4 and store the obtained solution for VC
fi
    choose, between the two candidate solutions, the smallest one
end.

Algorithm 2.

In 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 VC, guaranteeing approximation ratio smaller than $2 - (p/6) < 2$.

Proof: Let $G = (V, E)$ be a graph of order $n$, instance of 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
(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

$$\frac{\tau'(G)}{\tau(G)} \leq \frac{3}{2}$$

(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 $G$: $\alpha'(G) \geq \rho n/3$, $\tau'(G) = n - \alpha'(G) \leq \lceil 1 - (\rho / 3) \rceil n$, $\tau(G) \geq m = n/2$ and, consequently,

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

(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 $m_2 \geq \rho m/3$ (case (c.2.2))

(c.2.1) $m_1 \leq \rho m/3$.
This case is treated during the execution of the then consequence of the if clause in procedure 4. We have $|C \cup (S \cap T[M_1])| = \tau'(G) = \tau'(G') + m_1 \leq 2m[1 - (\rho / 3)] + \rho m/3$; on the other hand, $\tau(G) \geq m$; consequently,

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

(10)

(c.2.2) $m_2 \geq \rho m/3$.
This case represents the else consequence of the if clause in procedure 4.
Let us give a bound for the cardinality of the solution $C_1$ found by procedure 4 (whenever its else consequence is executed).
First, we shall prove that there are less than $f + g$ exposed vertices added in $C_1$ at line (21) of the procedure. Plainly, given a triangle $x - s - c - x$, then either $x \in X_C$, or $[x \in X_S$ and $sc \in F]$ (this last inclusion is due to the definition of the set $F$). Moreover, the case of the existence of two vertices $x_i, x_j$, both belonging to $X$ (a fortiori to $X_S$), and of a matched edge $sc$ of $M_1$ such that both $x isc x_i$ and $x isc x_j$ are triangles discovered at line (20) of procedure 4 is excluded, because of lemma 4. Consequently, there may be exposed vertices of $X$ forming triangles with, eventually, many edges of $M_1$, but there is no more than one exposed vertex of $X$ (a fortiori

---

We have arbitrarily chosen the constant 3 for the denominator of the fraction; in fact, theorem 5 remains valid for all constant greater than or equal to 2 in this denominator (up to a modification of the VC-ratio's value).
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 \(x_i - s_i - m(s_{i_j}) - m(s_{i_k}) - s_{i_k} - x_i\) and \(x_j - s_i - m(s_{i_k}) - m(s_{i_j}) - s_{i_j} - x_j\), \(i \neq j\) (intersected only on the endpoints of the (matched) edge \(m(s_{i_k})\)), then the path \(x_i - s_{i_k} - m(s_{i_k}) - m(s_{i_j}) - s_{i_j} - x_j\) is augmenting; so, this case is to be excluded. Moreover, for a set of non-disjoint cycles, a unique exposed vertex is introduced in \(C_1\). Consequently, if at line (c) of procedure 4 (for all iterations of the for loop) \(k\) disjoint 5-cycles have been detected, then \(k\) exposed vertices have been introduced in \(C_1\) at line (18) of
fixed positive constant $\epsilon$.

To prove proposition 5, it suffices to replace, in the part (i) of theorem 5, $(3/2) - \epsilon$ by $2 - \epsilon$; then, expression 3 gives $\tau(G) \leq n/2$, expression 4 gives $\tau'(G) \leq [1 - (\epsilon/2)]n$, from expression 5, we get $\alpha'(G) = n - \tau'(G) \geq (2/3)n$. Consequently, the hypothetical algorithm $A$ constitutes a polynomial algorithm for $S_n$ guaranteeing always $\alpha'(G)/\alpha(G) > \alpha'(G)/n > \epsilon/2$, a contradiction.
see that $\hat{S}$ constitutes an independent set for $G$. Plainly, the facts: $A \cdot \hat{f}(f) \leq 1$ and $\hat{S}(f) \geq \hat{\delta}$ imply, on the one hand, that $\hat{x}_i(f) \in [0,1]$, 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; consequently, two adjacent vertices cannot belong to $\hat{S}$ (recall that $\hat{S}$ is constituted by vertices $i$ for which $\hat{x}_i(f) > 1/2$).

The value $\hat{v}(f)$ corresponding to $\hat{S}$ can be decomposed as follows (recall that $f$ is increasing):

$$\hat{v}(f) = \sum_{i \in \hat{S}} f(\hat{x}_i(f)) + \sum_{i \notin \hat{S}} f(\hat{x}_i(f)) \leq |\hat{S}| f(1) + (n - |\hat{S}|) f(1/2),$$

or

$$|\hat{S}| \geq \frac{\hat{v}(f) - nf(1/2)}{f(1) - f(1/2)}. \quad (14)$$

Let us now consider a $\kappa \geq 2$ satisfying proposition 3, and an instance $G$ of $S_{\kappa}$. From expressions (13) and (14), $\forall f \in F$ and given that $\alpha(G) \leq n$, we have for the cardinality of $|\hat{S}|$:

$$|\hat{S}| \geq [(nf(1/\kappa) - nf(1/2))/(f(1) - f(1/2))],$$

or

$$\frac{|\hat{S}|}{\alpha(G)} \geq \frac{f(1/\kappa) - f(1/2)}{f(1) - f(1/2)}. \quad (15)$$

So, the hypothetical algorithm $A$ for CPM($\kappa$) is used to solve polynomially $S_{\kappa}$ within an approximation factor (15), because this ratio does not depend on $G$. We
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 $\left[\kappa/(\kappa - 1)\right]\sup_{f \in \mathcal{F}}\{f(1/2)/f(1)\}$.

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 $IPm$:

$$IPm = \left\{ \min_{\bar{z} \in \{0, 1\}^n} \sum_{i=1}^{n} f(1 - x_i) \right\}
A \cdot \bar{z} \leq 1
x_i \geq 0, \ i \in \{1, \ldots, n\}$$

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, $\hat{\nu}(f) \leq \rho v^*(f)$.

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

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

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, $\forall f \in \mathcal{F}$,

$$|\hat{S}|/\alpha \geq 2[1 - \rho(1/2)]f(1)/f(1/2))$$
and since we have supposed that $S_\kappa$ is not polynomially constant-approximable we have, $\forall f \in \mathcal{F}$, $1 - \rho(1/2)]f(1)/f(1/2)) \leq 0$, or $\rho \geq f(1/2)/f(1))1/[1 - \rho(1/2)]$.

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

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 $P = NP$.
input: a graph $G$ admitting a perfect matching;
output: a maximal independent set $S'$;

begin

\[ S' \leftarrow \emptyset \]

repeat

\[ u_j \leftarrow \arg \min_{u_i \in V} \{|\Gamma(u_i)|\}; \]

\[ S' \leftarrow S' \cup u_j; \]

\[ V \leftarrow V \setminus (\{u_j\} \cup \Gamma(u_j)); \]

delete from $E$ all edges incident to $\{u_j\} \cup \Gamma(u_j)$;
update the degrees of the vertices in $V$;

until $V = \emptyset$

end.

Algorithm 3. The greedy $S$ algorithm.

Part III

On the approximation ratio of the greedy algorithm of the maximum independent set problem

In what follows, given a connected graph $G = (V, E)$ of order $n$, we denote by $\Gamma(u_i)$, $u_i \in V$, the neighbour-set of vertex $u_i$; we denote by $\Delta$ the quantity $\max_{u_i \in V}\{|\Gamma(u_i)|\}$, i.e., the maximum degree of the vertices of $G$; finally, as previously, we denote by $m$ the size of a maximum matching $M$ of $G$.

The result of this section concerns the approximation ratio of the natural greedy algorithm 3 on graphs admitting a perfect matching $M$ (a matching of cardinality $\lfloor n/2 \rfloor$). In fact, we try to refine the analysis of the approximation performance of the greedy algorithm by taking into account the existence in $M$ of a set $F$ of "dissymmetric" matching edges reducing the size of the stability number $\alpha(G)$.

Theorem 9. Given a graph $G = (V, E)$ of order $n$ with maximum vertex degree $\Delta$, algorithm 3 (having a perfect matching $M$ among its inputs) is an $O(|E|)$ approximation algorithm for $S$, achieving an approximation ratio at least $(2/\Delta) + [2/\Delta(n - 2)]$.

Proof: In repeat loop, the choice of a vertex of minimum degree is performed in $O(n)$ by supposing that $G$ is represented by adjacency lists. Once a vertex $u_i$ is selected to be added in $S'$, the deletion of vertices in $\Gamma(u_i)$ can be made by treating the edges incident to $u_i$. On the other hand, the updating of the vertices of the "survived" graph concerns edges $u_ju_k$, $u_j \in \Gamma(u_i)$ and $u_k$ is a "survived" vertex. Finally, since all edges incident to deleted vertices are also deleted, all edges are treated only once. We see thus that, totally, the execution of the second repeat loop takes time proportional to the sum of degrees that equals twice the number of edges; therefore, we can conclude that the operations of the second repeat loop are performed in $O(|E|)$.

Algorithm 3, during the first selection of an element of $S'$, removes at most $(\delta + 1)$ vertices of the set $V$ (the selected one and its neighbours), where $\delta = \min_{u_i \in V}\{|\Gamma(u_i)|\}$ is the minimum degree of the vertices of $G$. After, for each of the later selections, the algorithm removes at most $\Delta$ vertices (always the selected one and its neighbours). To prove that, it suffices to prove that there will
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 $V_1 \subseteq V$ (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 $\alpha'(G) \geq \lfloor n - (\delta + 1)/\Delta \rfloor + 1$.

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:

$$\alpha'(G) = n - \gamma - (f + g) - m + \gamma (f + n).$$