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]

reçu : septembre 1994.

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

Contents

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 α 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 ρ-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 borne inférieure sur le rapport de tout algorithme polynomial approché pour la couverture de sommets. Nous prouvons aussi un résultat similaire pour un problème de programmation convexe (maximisation) incluant la programmation quadratique comme sous-problème.
Finalement, nous montrons que l'algorithme glouton naturel pour le problème du stable maximum sur les graphes admettant un couplage parfait admet un rapport d'approximation strictement plus grand que $2/\Delta$ où Δ est le degré maximum des sommets du graphe.

Mots-clés: Problème NP-complet, algorithme polynomial approché, transversal, stable.

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 α its stability number and by m the cardinality of a maximum matching of G.

Next, we prove that the existence of a polynomial time ρ-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 (maximization) 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 Δ 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_κ the following problem: "given a graph G admitting a maximum independent set of cardinality greater than or equal to n/κ, 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-Egervary (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 covering and matching, are, the maximization of an independent set, the minimization of a vertex covering and the maximization of a matching, respectively, (in what follows, we shall denote the first one by S and the second by VC. Problems S and VC are famous NP-complete problems ([6,7]), while the maximum matching problem admits an $O(n^{2.5})$ algorithm ([7]).

For VC, the maximum matching algorithm constitutes a polynomial time algorithm solving approximately VC (the endpoints of the edges of a maximum matching cover all of the edges of a graph) with approximation ratio smaller than 2. There are two interesting open problems associated with this result, (i) to design an algorithm with a better ratio or, (ii) to prove a lower bound for the ratios of any algorithm supposed to solve approximately VC.
On the other hand, the approximability of S has constituted an interesting open problem for many years. Recently ([1]), it has been proved that unless P = NP, S, even for bounded degree graphs, cannot be approximated within a constant error and also that VC does not admit a polynomial time approximation schema.

In section 6, we examine the way the behaviour of S_3 influences VC. We prove some conditional results that show how the existence of an approximation algorithm for S_3 would lead to the design of an 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.

Part I

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

$$\mathrm{S} = \begin{cases} \max & \vec{1}_n \cdot \vec{x} \\ & A \cdot \vec{x} \leq \vec{1}_{|E|} \\ & \vec{x} \in \{0,1\}^n \end{cases}$$

where A is the edge-vertex incidence matrix of G and $\vec{1}_D$ ($\vec{0}_D$) is the one-column vector of $\mathbb{R}^D$ ($D \in \mathbb{N}$), all of its coordinates being equal to 1 (0).
Let us denote by SR the following relaxed version of S:

$$\mathrm{SR} = \begin{cases} \max & \vec{1}_n \cdot \vec{x} \\ & A \cdot \vec{x} \leq \vec{1}_{|E|} \\ & \vec{x} \geq \vec{0}_n \end{cases}$$

The dual of SR denoted by ECR is

$$\mathrm{ECR} = \begin{cases} \min & \vec{1}_{|E|} \cdot \vec{x} \\ & A^T \cdot \vec{x} \geq \vec{1}_n \\ & \vec{x} \geq \vec{0}_{|E|} \end{cases}$$

where, this problem is denoted by ECR in order to indicate that it is the relaxed version of the following problem EC known as the *minimum edge covering*, which is polynomial ([3]):

$$\mathrm{EC} = \begin{cases} \min & \vec{1}_{|E|} \cdot \vec{x} \\ & A^T \cdot \vec{x} \geq \vec{1}_n \\ & \vec{x} \in \{0,1\}^{|E|} \end{cases}$$

Let us denote by $v(I)$ the optimal value (of the objective function) of an instance I of a problem. Remark that $\vec{0}_n$ and $\vec{1}_{|E|}$ are feasible for SR and ECR, respectively. As these dual instances

[1] In what follows, for purpose of clarity when no confusion arises, we use the same notation to denote a problem and its instance; but we have to remark that all instances of the problems mentioned below correspond to the same matrix A.

have their respective constraint sets non empty, they have the same optimal value.
So, the following inequalities hold: $v(\mathrm{S}) \leq v(\mathrm{SR}) = v(\mathrm{ECR}) \leq v(\mathrm{EC})$.

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

$$\mathrm{VC} = \left\{ \begin{array}{ll} \min & \vec{1}_n \cdot \vec{x} \\ & A \cdot (\vec{1}_n - \vec{x}) \leq \vec{1}_{|E|} \\ & \vec{x} \in \{0,1\}^n \end{array} \right. \iff \mathrm{VC} = \left\{ \begin{array}{ll} \min & \vec{1}_n \cdot \vec{x} \\ & A \cdot \vec{x} \geq A \cdot \vec{1}_n - \vec{1}_{|E|} = \vec{1}_{|E|} \\ & \vec{x} \in \{0,1\}^n \end{array} \right.$$

where the constraints of the lefthand side of the equivalence mean that the complement of a vertex covering is an independent set. The equality $A \cdot \vec{1}_n - \vec{1}_{|E|} = \vec{1}_{|E|}$ is due to the form of the matrix A.
We consider now the relaxed version VCR of VC:

$$\mathrm{VCR} = \left\{ \begin{array}{ll} \min & \vec{1}_n \cdot \vec{x} \\ & A \cdot \vec{x} \geq \vec{1}_{|E|} \\ & \vec{x} \geq \vec{0}_n \end{array} \right.$$

Its dual, denoted by MR, is written as:

$$\mathrm{MR} = \left\{ \begin{array}{ll} \max & \vec{1}_{|E|} \cdot \vec{x} \\ & A^T \cdot \vec{x} \leq \vec{1}_n \\ & \vec{x} \geq \vec{0}_{|E|} \end{array} \right.$$

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

$$\mathrm{M} = \left\{ \begin{array}{ll} \max & \vec{1}_{|E|} \cdot \vec{x} \\ & A^T \cdot \vec{x} \leq \vec{1}_n \\ & \vec{x} \in \{0,1\}^{|E|} \end{array} \right.$$

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

Note that given a graph G of order n, there is a well-known relation between $v(\mathrm{M})$ and $v(\mathrm{EC})$: $v(\mathrm{M}) = n - v(\mathrm{EC})$. On the other hand, a similar relation holds between S and VC: $v(\mathrm{S}) = n - v(\mathrm{VC})$.
So, the following schema summarizes the above discussion:

$$\begin{array}{ccccccc} v(\mathrm{S}) & \leq & v(\mathrm{SR}) & = & v(\mathrm{ECR}) & \leq & v(\mathrm{EC}) \\ \| & & & & & & \| \\ n - v(\mathrm{VC}) & \leq & n - v(\mathrm{VCR}) & = & n - v(\mathrm{MR}) & \leq & n - v(\mathrm{M}) \end{array} \quad (1)$$

Recall that the KE graphs are defined as the graphs where $\alpha(G) = v(\mathrm{S}) = n - v(\mathrm{M})$; using the above relations, we can give alternative expressions like $\alpha(G) = v(\mathrm{S}) = v(\mathrm{EC})$, or $\tau(G) = v(\mathrm{VC}) = v(\mathrm{M})$. In any case, we can interpret these graphs as the ones where there is no "discrete duality gap" between S and EC.

2 A characterization of König-Egervary graphs in terms of linear programming

In this section, we try to interpret of the KE graphs using arguments of linear programming. We have already mentioned that, for these graphs, the relation $v(\mathrm{S}) = v(\mathrm{EC})$ holds. In the following theorem 1, we prove that this relation is equivalent to the relation $v(\mathrm{S}) = v(\mathrm{SR})$.

Theorem 1. *In a graph G, $v(S) = v(EC) \iff v(S) = v(SR)$.*

Proof: From schema (1), the implication $v(S) = v(EC) \Longrightarrow v(S) = v(SR)$ 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 $(\vec{\hat{x}}, \vec{\hat{y}}) \in I\!R^n \times I\!R^{|E|}$; then, the fact that $\vec{\hat{x}}$ is a solution of SR and, simultaneously, that $\vec{\hat{y}}$ is a solution of ECR, is equivalent to the following conditions:

$$\begin{cases} A \cdot \hat{x} & \leq & \vec{1}_{|E|} & \text{(i)} \\ \hat{x} & \geq & \vec{0}_n & \text{(ii)} \\ A^T \cdot \hat{y} & \geq & \vec{1}_n & \text{(iii)} \\ \hat{y} & \geq & \vec{0}_{|E|} & \text{(iv)} \\ \hat{x}_i > 0 & \Rightarrow & \sum_{j \in \text{adj}(i)} \hat{y}_j = 1 & \text{(v)} \\ \hat{y}_j > 0 & \Rightarrow & \sum_{i \in \text{extr}(j)} \hat{x}_i = 1 & \text{(vi)} \end{cases}$$

where adj(i) is the set of the edges adjacent to vertex i and extr(j) is the set of the endpoints of edge j.
If we suppose that G admits the property $v(\text{S}) = v(\text{SR})$, then there exists a solution $\vec{\hat{x}}$ of SR having only $(0, 1)$ coefficients ($\vec{\hat{x}}$ is the characteristic vector of a maximum independent set $\hat{S}$ of G). Let us now consider a solution $\vec{\hat{y}}$ of ECR ($\vec{\hat{y}}$ having real coefficients). The pair $(\vec{\hat{x}}, \vec{\hat{y}})$ satisfies the optimality conditions (i) $\div$ (vi). Let us notice that condition (vi) means that $\hat{S}$ "covers" ("touches") all edges having a non-zero value in $\vec{\hat{y}}$.
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 $\vec{\hat{y}}$ non-zero values, these edges are, by definition, contained also in E'; so, the projection $\vec{\hat{y}}'$ of $\vec{\hat{y}}$ in $I\!R^{|E'|}$ (we retain the components corresponding to an edge of E') constitutes a feasible solution of ECR in $B(\hat{S})$ and, moreover, the pair $(\vec{\hat{x}}, \vec{\hat{y}}')$ 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 $B(\hat{S})$) is KE ([3]); hence, we deduce that $v(\text{S}_{B(\hat{S})}) = v(\text{EC}_{B(\hat{S})}) = v(\text{S}_G)$, 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, $v(\text{S}_G) \geq v(\text{EC}_G)$; moreover, the opposite inequality is always satisfied as we have already seen above; so $v(\text{S}_G) = v(\text{EC}_G)$ and G is a KE graph. ∎

Let us notice here that the equality $v(\text{ECR}) = v(\text{EC})$ 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, $p \in I\!N^*$, 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, $v(\text{EC}) - v(\text{ECR}) \geq (p/2) - 1 \rightarrow_{p \to \infty} \infty$.

On the other hand, let us consider a K_{2p}, $p \in I\!N^*$. The size of a maximum matching in K_{2p} is equal to p; so, $v(\text{EC}) = p$; moreover, the optimal value of SR in K_{2p} is greater than p, value corresponding to the assignment of the value $1/2$ on every vertex of the graph; so, $v(\text{SR}) = v(\text{ECR}) = v(\text{EC}) = p$; but $\alpha(K_{2p}) = 1$; hence, a result analogous to the one of theorem 1 cannot be hoped for the pair (EC,ECR).

For the KE graphs where $v(\mathrm{S}) = v(\mathrm{EC}) = v(\mathrm{SR})$, 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:

$$\mathrm{HC} = \left\{ \begin{array}{lll} \max & \sum_{i=1}^{n} \vec{1}_n \cdot \vec{x_i} & \\ & A \cdot \vec{x_i} \geq \vec{x_{i+1}} \quad i \in \{1, \ldots n\} & \text{(i)} \\ & A \cdot \vec{x_1} \geq \vec{x_n} & \text{(ii)} \\ & \vec{1}_n \cdot \vec{x_i} \leq n \quad i \in \{1, \ldots n\} & \text{(iii)} \\ & \sum_{i=1}^{n} \cdot \vec{x_i} \leq \vec{1}_n \quad i \in \{1, \ldots n\} & \text{(iv)} \\ & \vec{x_i} \in \{0,1\}^n \quad i \in \{1, \ldots n\} & \text{(v)} \end{array} \right.$$

where A is the vertex-vertex incidence matrix of the graph-instance G of HC and $\vec{x_i}$, $i = 1, \ldots, n$, 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 $\vec{x_i}$, $i = 1, \ldots, n$, represents one or zero vertices; constraint (i) signifies that if $\vec{x_{i+1}}$ and $\vec{x_i}$ 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 $(\vec{x_1}, \ldots, \vec{x_n})$ is of the form $(\vec{x_1}, \ldots \vec{x_k}, \vec{0}_n, \ldots, \vec{0}_n)$, where $k \in \{0, \ldots, n\}$ and $\vec{x_i}$, $i \leq k$, represents a vertex; so, in the case where $k \neq 0$, the sequence $(\vec{x_1}, \ldots, \vec{x_n})$ 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.
Let us now relax constraint (v) by transforming it into $\vec{x_i} \geq \vec{0}^n$, $i = 1, \ldots, n$. 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 $\vec{x_i} = (1/n)\vec{1}_n$, $i = 1, \ldots, n$, 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. ∎

[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 $v(\mathrm{S}) = v(\mathrm{EC})$ 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 $\langle v_1, v_2, \ldots, v_n \rangle$ of the vertices of G, such that $v_n v_1 \in E$ and $v_i v_{i+1} \in E$ for all i, $1 \leq i \leq n$?".

3 A generalization of the König-Egervary 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 $S_{\vec{a}}$ a general instance of the weighted version of S represented by a graph $G = (V, E)$ of order n and by a vector $\vec{a} \in (\mathbb{R}^{*+})^n$, the components of which being the weights of V. So, $S_{\vec{a}}$ can be described by the following integer-linear program:

$$S_{\vec{a}} = \begin{cases} \max & \vec{a} \cdot \vec{x} \\ & A \cdot \vec{x} \leq \vec{1}_{|E|} \\ & \vec{x} \in \{0,1\}^n \end{cases}$$

where A is the edge-vertex incidence matrix of G.
The relaxed version $SR_{\vec{a}}$ of $S_{\vec{a}}$ is defined by

$$SR_{\vec{a}} = \begin{cases} \max & \vec{a} \cdot \vec{x} \\ & A \cdot \vec{x} \leq \vec{1}_{|E|} \\ & \vec{x} \geq \vec{0}_n \end{cases}$$

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

$$ECR_{\vec{a}} = \begin{cases} \min & \vec{1}_{|E|} \cdot \vec{y} \\ & A^T \cdot \vec{y} \geq \vec{a} \\ & \vec{y} \geq \vec{0}_{|E|} \end{cases}$$

In what follows, we treat the class $\mathcal{G}_{\vec{a}}$ of graphs for which $v(S_{\vec{a}}) = v(SR_{\vec{a}})$; it is easy to see that, by theorem 1, the class $\mathcal{G}_{\vec{1}_n}$ 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 $\mathcal{G}_{\vec{a}}$. We suppose that the input of the algorithm is a graph $G = (V, E)$ known to be in $\mathcal{G}_{\vec{a}}$; later, we shall show how the same algorithm can be applied to every graph in order to decide if its input belongs to $\mathcal{G}_{\vec{a}}$.
A first step of this algorithm is the construction of a solution $\vec{y}$ of $ECR_{\vec{a}}$; we do not detail on this step since $ECR_{\vec{a}}$ 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 $\{e \in E : \hat{y}_e > 0\}$; also, given a set $P \subset E$, we denote, $\forall v \in V$, by $\Gamma_P(v)$ 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 $V' \subseteq V$, 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 $\vec{\hat{y}_r}$ of the problem $ECR^r_{\vec{a}}$, 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 $\vec{a}_{|E_r|} \cdot \vec{\hat{y}_r}$.

Theorem 2. *Algorithm 1 is an $O(n^3)$ exact maximum-weight independent set algorithm for the class $\mathcal{G}_{\vec{a}}$, $\vec{a} > \vec{0}_n$.*

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

```
begin
    S ← ∅;
    G_r ← G;
    find a solution ȳ of ECR_ā in G
    while V_r ≠ ∅ do
        determine ȳ_r by projection of ȳ on the space corresponding to E_r;
        select an x_0 ∈ V_r, such that x_0 is not isolated;
        ℓ(x_0) ← c;
        LV_r ← {x_0};
        LABEL(G_r, C, LV_r);
        if ¬TEST(G_r, C, LV_r, ȳ_r) then
          begin
              ℓ(x_0) ← s;
              LV_r ← {x_0};
              LABEL(G_r, C, LV_r)
          end;
        V_r ← V_r \ LV_r;
        E_r ← E_r \ Γ(LV_r);
        S ← S ∪ {x_0 ∈ LV_p : ℓ(x_0) = s}
    od
end.
```

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

```
begin
    E_test ← LV_r;
    for all x ∈ LV_r do
        if ℓ(x) = c and Γ_{C∩E_r}(x) ⊄ LV_r then
          for all y ∈ Γ_{C∩E_r}(x) such that y ∉ LV_r do
              ℓ(y) ← s;
              LV_r ← LV_r ∪ {y}
          od
        fi
        if ℓ(x) = s and Γ_{E_r}(x) ⊄ LV_r then
          for all y ∈ Γ_{E_r}(x) ∧ y ∉ LV_r do
              ℓ(y) ← c;
              LV_r ← LV_r ∪ {y}
          od
        fi
    od
    if LV_r ≠ E_test then LABEL(G_r, C, LV_r) fi
end;
```

Procedure 1. LABEL.

begin
TEST $\leftarrow$ **true**
if $\sum_{v,\ell(v)=\mathbf{s}} a_v \neq \vec{a_r} \cdot \vec{y_r}$ **then** TEST $\leftarrow$ **false**
else
delete all the edges of G_r incident to at most one vertex x such that $\ell(x) = \mathbf{s}$;
update all the degrees in V_r;
if $\exists v \in V_r$ with $|\Gamma_{E_r}(v)| \neq 0$ **then** TEST $\leftarrow$ **false**
fi
end;

Function 1. TEST.

Proof: Consider a graph $G = (V, E)$ of order n belonging to $\mathcal{G}_{\vec{a}}$; 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 $\mathcal{G}_{\vec{a}}$; moreover, for $k < K$, a maximum-weight independent set S_r^k of G_r^k can be obtained by setting $S_r^k = \{x \in LV_r^k : \ell(x) = \mathbf{s}\} \cup S_r^{k+1}$, where S_r^{k+1} is a maximum-weight independent set of G_r^{k+1} and ℓ is the mapping $V_r^k \mapsto \{\mathbf{c}, \mathbf{s}\}$ computed by algorithm 1.*

Proof: (lemma 1.) Let us prove by induction that $G_r^k \in \mathcal{G}_{\vec{a}}$.
First, we shall prove that, $\forall k$, G_r^k satisfies the property: $\Gamma_C(v) \subset V_r^k$, $\forall v \in V_r^k$. For $k = 1$, since $G_r^1 = G$ and $C \subset E$, the property is obvious. Let us suppose the truth of the property for $k < K$ (we recall that algorithm stops after the Kth 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 $\Gamma_C(v) \subset V_r^k \setminus LV_r = V_r^{k+1}$ 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 $\mathcal{G}_{\vec{a}}$, we shall prove the following lemma 2.

Lemma 2. *Let $G = (V, E)$ be a graph of the class $\mathcal{G}_{\vec{a}}$, $\vec{\tilde{y}}$ be an optimal solution of* $\mathrm{ECR}_{\vec{a}}$ *on G and C be as defined previously; let $\tilde{G} = (\tilde{V}, \tilde{E})$ be a subgraph of G induced by $\tilde{V} \subset V$ 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, $\tilde{G} \in \mathcal{G}_{\vec{a}}$; moreover, in this case, the tracks of $\vec{\tilde{y}}$ and $\vec{\tilde{x}}$ on $\tilde{V}$ are solutions of* $\mathrm{ECR}_{\vec{a}}$ *and* $\mathrm{S}_{\vec{a}}$ *on $\tilde{G}$, respectively.*

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

equivalent to the following conditions:

$$\begin{cases} A \cdot \vec{\hat{x}} & \leq & \vec{1}_{|E|} & \text{(i)} \\ \vec{\hat{x}} & \geq & \vec{0}_n & \text{(ii)} \\ A^T \cdot \vec{\hat{y}} & \geq & \vec{a} & \text{(iii)} \\ \vec{\hat{y}} & \geq & \vec{0}_{|E|} & \text{(iv)} \\ \hat{x}_i > 0 & \Rightarrow & \sum_{j \in \text{adj}(i)} \hat{y}_j = a_i & \text{(v)} \\ \hat{y}_j > 0 & \Rightarrow & \sum_{i \in \text{extr}(j)} \hat{x}_i = 1 & \text{(vi)} \end{cases}$$

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 $\vec{\tilde{x}} \in \mathbb{R}^{|\tilde{V}|}$ such that, $\forall v \in \tilde{V}$, $\tilde{x}_v = \hat{x}_v$ and $\vec{\tilde{y}} \in \mathbb{R}^{|\tilde{E}|}$, such that $\forall e \in \tilde{E}$, $\tilde{y}_e = \hat{y}_e$ the projections of $\vec{\hat{x}}$ and $\vec{\hat{y}}$ on the characteristic spaces of $\tilde{V}$ and $\tilde{E}$, respectively; we also define $\vec{\tilde{a}} \in \mathbb{R}^{|\tilde{V}|}$ such that, $\forall v \in \tilde{V}$, $\tilde{a}_v = a_v$, the projection of vector $\vec{a}$ (it is easy to see that $\vec{\tilde{x}} \in \{0,1\}^{|\tilde{V}|}$).
We shall show that the pair $(\vec{\tilde{x}}, \vec{\tilde{y}})$ satisfies the primal-dual optimality conditions (i) $\div$ (vi) (for $\text{SR}_{\vec{a}}$ and $\text{ECR}_{\vec{a}}$, respectively) in the graph $\tilde{G}$ of edge-vertex incidence matrix $\tilde{A}$.
In fact, $\vec{\tilde{x}}$ is the characteristic vector of an independent set of $\tilde{G}$ because $\vec{\tilde{x}}$ corresponds to an independent set of G and, moreover, $\tilde{G}$ is a sub-graph of G; so, $\tilde{A} \cdot \vec{\tilde{x}} \leq \vec{1}_{|\tilde{E}|}$ and $\vec{\tilde{x}} \geq \vec{0}_{|\tilde{V}|}$ (conditions (i) and (ii)). It is also easy to obtain $\vec{\tilde{y}} \geq \vec{0}_{|\tilde{E}|}$ (condition (iv)). Moreover, vector $\tilde{y}$ satisfies $\tilde{A} \cdot \vec{\tilde{y}} \geq \vec{\tilde{a}}$ (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 $\hat{y}_e > 0$, 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 $\hat{y}_e$, 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 $\text{ECR}_{\vec{a}}$ is saturated, so does the corresponding constraint of $\text{ECR}_{\vec{a}}$ 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 $\vec{\tilde{x}}$ and $\vec{\tilde{y}}$ are solutions of $\text{SR}_{\vec{a}}$ and $\text{ECR}_{\vec{a}}$, respectively, defined on $\tilde{G}$; moreover, $\vec{\tilde{x}}$ has (0,1) coordinates; so, $\tilde{G} \in \mathcal{G}_{\vec{a}}$ and this completes the proof of **lemma 2.** ∎

We now continue the proof of lemma 1.
By lemma 2, for a given k, G_r^k is an element of $\mathcal{G}_{\vec{a}}$; so, the first part of the lemma is proved.
Let us notice that, from the proof of lemma 2, we can deduce also that $LG_r^k \in \mathcal{G}_{\vec{a}}$ and, moreover, the intersections of a maximum-weight independent set of G with V_r^k and LV_r^k, constitute maximum-weight independent sets for G_r^k and LG_r^k, respectively; if we denote also by $\vec{\hat{y}}_r^k$ and $L\vec{\hat{y}}_r^k$ the projections of $\vec{\hat{y}}$ on the spaces corresponding to E_r^k and LE_r^k, respectively, then the graphs G_r^k and LG_r^k have weighted-stability numbers equal to $\vec{a}_{|V_r^k|} \cdot \vec{\hat{y}}_r^k$ and $\vec{a}_{|LV_r^k|} \cdot L\vec{\hat{y}}_r^k$, respectively.
Given a labelling $\ell : V_r^k \mapsto \{\mathbf{c}, \mathbf{s}\}$ of G_r^k, we call it a "good labelling" if there exists a maximum-weight independent set of G_r^k 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 $\vec{a}_{|LE_r^k|} \cdot L\vec{\hat{y}}_r^k$, this fact being tested by function TEST. Consequently, since $LG_r^k \in \mathcal{G}_{\vec{a}}$, 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 $\vec{a}_{|LV_r^k|} \cdot L\vec{\hat{y}}_r^k$. On the other hand, if $k < K$, a maximum-weight independent set of G_r^{k+1} has value $\vec{a}_{|V_r^{k+1}|} \cdot \vec{\hat{y}}_r^{k+1}$; 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 $\vec{a}_{|LV_r^k|} \cdot L\vec{\hat{y}}_r^k + \vec{a}_{|V_r^{k+1}|} \cdot \vec{\hat{y}}_r^{k+1} = \vec{a}_{|V_r^k|} \cdot \vec{\hat{y}}_r^k$, 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 set of LG_r^k with an independent set of G_r^{k+1} constitutes an independent set of G_r^k.
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 $G_r^{k+1} \in \mathcal{G}_{\vec{a}}$, the arguments developed previously remain valid for G_r^{k+1}. Moreover, while $|V(G_r)| \neq 0$, a new execution of the **while** loop is always possible and produces a set LV_r of size greater than 2, since if LV_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 Kth (last) iteration, $LV_r^K = V_r^K$; so, vertices of V_r^K labelled by **s** constitute a maximum-weight independent set of G_r^K; lemmas 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 $\vec{\hat{y}}$ for $\mathrm{ECR}_{\vec{a}}$ 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 $O(|E_r^k|)$ 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 $\mathcal{G}_{\vec{a}}$.*

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 $\vec{a}_{|E|} \cdot \vec{\hat{y}}$, this verification taking time linear to $|E|$. If this is the case, then $G \in \mathcal{G}_{\vec{a}}$, while in the opposite case $G \notin \mathcal{G}_{\vec{a}}$, since from theorem 2, if $G \in \mathcal{G}_{\vec{a}}$, then algorithm 1 determines a solution of optimal value $\vec{a}_{|E|} \cdot \vec{\hat{y}}$. ∎

4 König-Egervary graphs revisited

In the case where $\vec{a} = \vec{1}_n$, 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[5] 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[6] (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 of $O(n^{2.5})$ ([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-Egervary 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 to C^* and the other one to S^*. Let us call this edges "dissymmetric" and denote by F the set of these "dissymmetric" edges ($|F| = f$). For M (in the case that it is not perfect), let us denote by X the set of the exposed (non-saturated) vertices of G with respect to M, and by X_C ($|X_C| = g$) and X_S the subsets of X belonging to C^* and S^*, respectively (of course, $X = X_C \cup X_S$). The numbers f and g, consequently the sets F and X_C, depend not only on M but also, for a fixed matching, on the sets C^* (and S^*) considered. However, the sum $f + g$ is a quantity depending only on G ($f + g$ can be considered as the "discrete duality gap"). In fact, for every graph

$$\begin{aligned} |C^*| &= \tau(G) = m + (f + g) \\ |S^*| &= \alpha(G) = n - m - (f + g). \end{aligned} \tag{2}$$

So, a KE graph is exactly a graph where $f + g = 0$.

In the first part of this section, we allow a relaxation of the condition $f + g = 0$ by considering this quantity bounded above, first by a universal constant, next by a number which can be function of the size of the instance, and we give some polynomial results for maximum

[5] We have adopted the terminology of [7] where the non-saturated vertices of a graph with respect to a given matching are called exposed.

[6] A matching saturating all of the vertices of the graph.

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 κ 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 κ among its input parameters) providing an independent set of cardinality at least equal to $\lfloor n/[2(\kappa+1)] - 2 \rfloor$.*

Proof:
(i) The condition $f+g \leq \kappa$ implies that both f and g are bounded above by κ. So, for all integer $h \leq \min\{m, \kappa\}$ (where m is the cardinality of a maximum matching M of G) and 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 κ 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 $|M_i| = \lceil m/(\kappa+1) \rceil$, $i = 1, \ldots, \kappa$ and $M_{\kappa+1} = m - \sum_{i=1}^{\kappa} \lceil m/(\kappa+1) \rceil$. We also arbitrarily partition the set X of the exposed vertices of G into $\kappa+1$ sets $X_1, \ldots, X_{\kappa+1}$ sets, each of size at least $\lceil (n-2m)/(\kappa+1) \rceil$. 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 $T[M_j]$, 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 fuction, 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 κ, 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 κ_0 such that, for every fixed constant $\rho < 1$, there is no polynomial time approximation algorithm for S_{κ_0}, guaranteeing approximation ratio greater than or equal to ρ.*

Unfortunately, up to now, we do not precisely know the value of κ_0. However, we shall see at the end of this section that $\kappa_0 \geq 2$. In any case, S_κ ($\kappa \geq 2$), in particular for small values of κ, 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 edges 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 $\mathcal{A}$ with an approximation ratio less than or equal to $(3/2) - \epsilon$, for a fixed constant $\epsilon > 0$. Let also G be an instance of S_3.
From the hypothesis on the existence of the ρ-approximation algorithm $\mathcal{A}$, we have $\tau'(G)/\tau(G) \leq (3/2) - \epsilon$ and, on the other hand, since G is an instance of S_3, we get

$$\tau(G) = n - \alpha(G) \leq \frac{2}{3}n. \tag{3}$$

begin
compute a maximum matching M in G;
$C \leftarrow T[M]$
end;

Procedure 2.

From the above expressions, we deduce

$$\tau'(G) \leq \left(1 - \frac{2}{3}\epsilon\right) n \tag{4}$$

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

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

Consequently, $\mathcal{A}$ (provided with a set-difference instruction) constitutes a polynomial algorithm for S_3, always guaranteeing a ratio $\alpha'(G)/\alpha(G) \geq \alpha'(G)/n \geq (2/3)\epsilon$, 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[7] ρ. 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 $\mathcal{A}$, and then taking the complement of a the 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

[7] We suppose that whenever $\mathcal{A}$ operates on graphs G which are not instances of S_3, it stops in polynomial time, delivering either non-feasible solutions or maximal independent sets of cardinality smaller than $\rho n/3$.

begin
apply $\mathcal{A}$ on G to obtain an independent set solution S;
$C \leftarrow V \setminus S$
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 $T[M_1] \cap S = \{s_1, \ldots, s_{m_1}\}$ 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 lth iteration of the **for** loop of line **(12)**. We then distinguish two cases: **(i)** $v \in S_1 \cap C$, and **(ii)** $v \in S_1 \cap S$.

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 $s \in S_1^l$, vertex $m(s) \in C_1^l$. 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^l during the kth execution of line **(25)**; let us denote $S_1^l(k)$ and $C_1^l(k)$, the subsets of S_1^l and C_1^l, respectively, resulting from the insertion of s_k in S_1^l. 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 $S_1^l(0)$ and $C_1^l(0)$, the non-empty subsets of S_1^l and C_1^l resulting from the last execution of lines **(19)**, or **(22)**.
Let us now show by induction on $k \in \{0, \ldots, N\}$ that: **(a)** $\forall v \in S_1^l(k) \cup C_1^l(k)$, $m(v) \in S_1^l(k) \cup C_1^l(k)$ and **(b)** for every vertex $s \in S_1^l(k)$, there exists an alternating path from s to x_l starting from a matched edge and exclusively containing vertices of $S_1^l(k) \cup C_1^l(k)$.

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

Inductive step: suppose **(a)** and **(b)** true for $k < N$; the only newly introduced in $S_1^l(k) \cup C_1^l(k)$ 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) $G' \leftarrow G[T[M]]$
(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** $C \leftarrow C \cup (T[M_1] \cap S)$
(6) **else**
(7) $C_2 \leftarrow T[M_2]$;
(8) $C_1 \leftarrow \emptyset$;
(9) $S_1 \leftarrow \emptyset$;
(10) order arbitrarily the elements of X_1;
(11) let $X_1 = \{x_1, \ldots, x_{|X_1|}\}$ be the resulting ordering
(12) **for** $l \leftarrow 1$ **to** $|X_1|$ **do**
(13) $C_1^l \leftarrow \emptyset$;
(14) $S_1^l \leftarrow \emptyset$;
(15) $V_l \leftarrow T[M_1[x_l]] \cup \{x_l\}$;
(16) $E_l \leftarrow E(G[V_l])$;
(17) find all 5-cycles $x_l - s_i - c_i - c_j - s_j - x_l$ such that $\{s_i c_i, c_j s_j\} \subseteq M_1$;
(18) $C_1^l \leftarrow C_1^l \cup \{x_l, c_i, c_j\}$;
(19) $S_1^l \leftarrow S_1^l \cup \{s_i, s_j\}$;
(20) find all triangles $x_l - s_i - c_i - x_l$ such that $\{s_i c_i\} \in M_1$;
(21) $C_1^l \leftarrow C_1^l \cup \{x_l, c_i\}$;
(22) $S_1^l \leftarrow S_1^l \cup \{s_i\}$
(23) **while** $\exists (c_k \in (V_l \setminus C_1^l) \wedge s_i \in S_1^l)$, $c_k s_i \in E_l$, $k \neq i$ **do**
(24) $C_1^l \leftarrow C_1^l \cup \{c_k\}$;
(25) $S_1^l \leftarrow S_1^l \cup \{m(c_k)\}$
(26) **od**
(27) **while** $\exists s_k c_k \in M_1, \{s_k, c_k\} \subseteq V_l, \{s_k, c_k\} \cap C_1^l = \emptyset$ **do**
(28) $C_1^l \leftarrow C_1^l \cup \{s_k\}$;
(29) $S_1^l \leftarrow S_1^l \cup \{c_k\}$
(30) **od**
(31) $C_1 \leftarrow C_1 \cup C_1^l$;
(32) $S_1 \leftarrow S_1 \cup S_1^l$;
(33) **od**
(34) $S_1 \leftarrow S_1 \cup (X \setminus C_1)$;
(35) $C \leftarrow C_1 \cup C_2$
(36) **fi**
end;

Procedure 4.

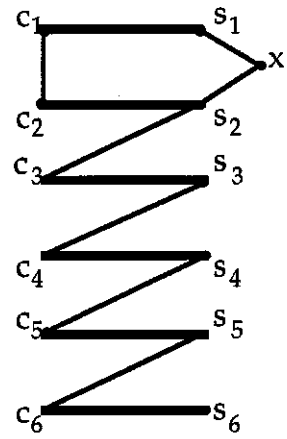

Figure 1. The thick lines represent matched edges.

property **(b)** on range $k+1$, it suffices to consider the case where $s_{k+1} \notin C_1^l$ (the opposite case being treated in case **(i)**); then, from line **(23)** of procedure 4, $m(s_{k+1}) \notin C_1^l(k)$ and, also, $\exists s \in S_1^l(k)$ such that $m(s_{k+1})s \in E_l$; from the inductive hypothesis on property **(b)**, there exists an alternating path μ from s to x_l exclusively containing vertices from $S_1^l(k) \cup C_1^l(k)$; on the other hand, from the inductive hypothesis on property **(a)**, $\{s_{k+1}, m(s_{k+1})\} \not\subset S_1^l(k) \cup C_1^l(k)$ and, moreover, $(S_1^l(k) \cup C_1^l(k)) \cap \{s_{k+1}, m(s_{k+1})\} = \emptyset$; 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 π 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 μ 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: 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^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, $x_l \in X_1$, and **(ii)** edges $v_i v_j$, where $v_i \in V_l$, $v_j \in V_p$, $\{x_l, x_p\} \subseteq X_1$, $l \neq p$.

(i) We first prove that the subset of C_1^l created at the lth iteration of the **for** loop of the procedure (concerning $x_l \in X_1$) covers all edges in E_l.
Let us first point out that $S_1^l \cap V_l = V_l \setminus C_1^l$. In fact, obviously, $C_1^l \cap S_1^l = \emptyset$; line **(34)** of procedure 4 guarantees that $X \subseteq S_1 \cup C_1$; on the other hand, at the lth iteration of the **for** loop, $l = 1, \ldots, |X_1|$, if a vertex of $T[M_1]$ is introduced in C_1^l, its mate is immediately introduced in S_1^l. Finally, the second **while** loop (lines **(27)-(30)**) guarantees that all of the edges of $M_1 \cap E_l$ are covered by C_1^l. Hence, $S_1^l \cup C_1^l \supseteq T[M_1 \cap E_l] \cup X \supseteq V_l$.
Observe that when the execution exits the first **while** loop (lines **(23)-(26)**), $S_1^l \subseteq S_1$, and thus S_1^l is an independent set of G. Now, let $c_k \in S_1^l$; suppose there exists $s_i \in S_i^l$ such that $s_i c_k \in E_l$. Since c_k is inserted in S_1^l during the execution of the second **while** loop (lines **(27)-(30)**), $i \neq k$; but then $c_k \in C_1^l$ (first **while** loop), a contradiction. Suppose now there exists $c_i \in S_1^l$ such that $c_k c_i \in E_l$. Then, c_k and c_i would be inserted in S_1^l during two distinct executions of the second **while** loop and, consequently, there would be a cycle $x_l - s_k - c_k - c_i - s_i - x_l$ detected at line **(17)** of the procedure.
Consequently, $S_1^l \cap V_l$ really constitutes an independent set for $G[V_l]$, set $V_l \setminus S_1^l$ constituting a vertex covering for this graph.

(ii) Now, we will use lemma 3 to prove that set C_1, obtained by procedure 4, covers all edges $v_i v_j$, where $v_i \in V_l$, $v_j \in V_p$, $\{x_l, x_p\} \subseteq X_1$, $l \neq p$. We show this by contradiction; let $v_i \in V_l$, $v_j \in V_p$, $\{x_l, x_p\} \subseteq X_1$ such that $v_i v_j \in E(G)$; let us suppose that $\{v_i, v_j\} \subseteq S_1$. We distinguish two cases, depending on whether or not at least one between v_i, v_j belongs to X_1.
We first study the case where $\{v_i, v_j\} \subseteq T[M_1]$ (neither v_i, nor v_j belong to X_1).
If $\{v_i, v_j\} \subseteq T[M_1]$, then by lemma 3, there exist two alternating paths $v_i - m(v_i) - \ldots - x_l$ and $v_j - m(v_j) - \ldots - x_p$, respectively. These two paths do not have vertices in common because, in the opposite case, if they had a vertex v in common, they would have the edge $vm(v)$ in common too, and an augmenting path $x_l - \ldots - m(v) - v - \ldots - x_p$ would be brought to the fore. Consequently, on the hypothesis of the existence of an edge $v_i v_j$, the path $x_l - \ldots - m(v_i) - v_i - v_j - m(v_j) - \ldots - x_p$ is still augmenting, contradicting the maximality of M.
Let us now study the case where at least one of the v_i, v_j belongs in X; we first note that the case $\{v_i, v_j\} \subseteq X$ is excluded since X is an independent set. So, we suppose that $v_i \in X$ and $v_j \notin X$, $v_j \in V_p$, $x_p \neq v_i$. But then, the application of lemma 3 on v_j brings to the fore an alternating path $v_j - m(v_j) - \ldots - x_p$, and consequently, the path $v_i - v_j - m(v_j) - \ldots - x_p$ is augmenting, contradicting so the maximality of M.
The above discussion concludes the proof that C_1 constitutes a vertex covering for the graph $G[T[M_1] \cup X_1]$.

On the other hand, obviously, set $T[M_2]$ covers all edges in $E(G) \setminus E(G[T[M_1] \cup X])$ (recall that M_2 is maximal for $G[T[M_2] \cup X_2]$); consequently, it covers all cross edges between $T[M_1]$ and $T[M_2]$. In fact, the edges between X_2 and $T[M_1] \cap C_1$ are covered by $C_1 \cup C_2$; on the other

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.

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 $\mathcal{A}$ (directly called by procedure 3) for S_3.

6.2.2 The main result

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

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 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,

$$|C^*| = \tau(G) = m + (f + g) \tag{6}$$

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 VC. As we have seen in part I, in this case VC admits a polynomial time algorithm of approximation ratio equal to 1. So, for case **(a)** we have

$$\frac{\tau'(G)}{\tau(G)} = 1. \tag{7}$$

(b) $\mathbf{f+g \geq m/3}$.
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

$$\frac{\tau'(G)}{\tau(G)} \leq \frac{3}{2} \tag{8}$$

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

(c) $\mathbf{0 < (f+g) \leq m/3}$.
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 ($\mathbf{g = |X_S| = |X_C| = |X| = 0}$).
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 $\mathcal{A}$ 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,

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

(c.2) M is not perfect ($\mathbf{X \neq \emptyset}$).
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)**)[8].

(c.2.1) $\mathbf{m_1 \leq \rho m/3}$.
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,

$$\frac{\tau'(G)}{\tau(G)} \leq 2 - \frac{\rho}{3} < 2. \tag{10}$$

(c.2.2) $\mathbf{m_1 \geq \rho m/3}$.
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 $x_i s c x_i$ and $x_j s c x_j$ 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

[8] 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 $x_i - s_{i_j} - m(s_{i_j}) - m(s_{i_k}) - s_{i_k} - x_i$ and $x_j - s_{i_k} - m(s_{i_k}) - m(s_{i_l}) - s_{i_l} - x_j$, $x_i \neq x_j$ (intersected only on the endpoints of the (matched) edge $m(s_{i_k})s_{i_k}$), then the path $x_i - s_{i_k} - m(s_{i_k}) - m(s_{i_l}) - s_{i_l} - x_j$ 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 the procedure.
Finally, for the m_1 edges of M_1, m_1 vertices have been introduced in C_1 for all iterations of the **for** loop of procedure 4, since once a vertex v of $T[M_1]$ is introduced in C_1, its mate $m(v)$ is introduced in S_1.
So, $|C_1| = \tau'(G[T[M_1] \cup X_1]) \leq (f+g) + k + m_1$; hence, $|C| = \tau'(G) \leq (f+g) + k + m_1 + 2m_2$; on the other hand, $\tau(G) = m + (f+g)$, with $m = m_1 + m_2$; moreover, $m_1 \geq \rho m/3$; finally, $k \leq m_1/2$ and $f + g > 0$. So, we have

$$\frac{\tau'(G)}{\tau(G)} \leq \frac{k + m_1 + 2m_2}{m} = 2 - \frac{m_1 - k}{m} \leq 2 - \frac{\frac{m_1}{2}}{m} \leq 2 - \frac{\rho}{6}. \tag{11}$$

Consequently, from expressions (7), (8), (9), (10) and (11), algorithm 2 constitutes a polynomial time $[2-(\rho/6)]$-approximation algorithm for VC. Since ρ is supposed to be a universal constant, this ratio is always strictly smaller than 2. ∎

The following theorem summarizes the results of sections 6.1 and 6.2.

Theorem 6. *Let $\rho < 1$ be a fixed positive constant. Under the hypothesis $P \neq NP$, (i) the non-existence of a ρ-approximation polynomial time algorithm for S_3 implies that no polynomial time approximation algorithm for VC can guarantee an approximation ratio strictly smaller than $3/2$; (ii) if, on the contrary, such an algorithm exists for S_3, then there exists an algorithm for VC guaranteeing an approximation ratio smaller than, or equal to, $2 - (\rho/6) < 2$.*

As we have already mentioned above, we do not precisely know the infimum κ_0 of the values of κ for which S_κ does not admit a polynomial time approximation algorithm of constant error. However, we can prove that $\kappa_0 \geq 2$. In fact, let us consider an instance G of $S_{2-\epsilon}$, for a positive constant ϵ. Then, for this instance, there exists an $\epsilon' > 0$ for which $\alpha(G) \geq [(1/2) + \epsilon']n$ holds. Let e be the number of the exposed vertices with respect to a maximum matching M of cardinality m. Then, $n = 2m + e$, and moreover, it is easy to see that $\alpha(G) \leq m + e$. If we choose as approximate solution for the instance the set of these exposed vertices, then the combination of the above inequalities leads to $e \geq 2\epsilon' n$. Since $2\epsilon'$ is a constant for $S_{2-\epsilon}$, we can conclude that $\kappa_0 \geq 2$.
In fact, our guess is that S_κ is not constant-approximable for any $\kappa > 2$, unless P = NP. In any case, with arguments similar to the ones used for the proof of statement (i) of theorem 5, one can easily prove the following proposition 5.

Proposition 5. *If S_2 is non-constant approximable in polynomial time, then no polynomial time approximation algorithm for VC can guarantee an approximation ratio $\rho < 2 - \epsilon$, for a*

fixed positive constant ϵ.

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 $\alpha'(G) = n - \tau'(G) \geq (2/3)\epsilon n$. Consequently, the hypothetical algorithm $\mathcal{A}$ constitutes a polynomial algorithm for S_2 guaranteeing always $\alpha'(G)/\alpha(G) \geq \alpha'(G)/n \geq \epsilon/2$, a contradiction, since $\epsilon/2$ 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:

$$\text{CPM}(\kappa) \quad = \quad \begin{cases} \max & \sum_{i \in \{1,\ldots,n\}} f(x_i) \\ & x \in \mathcal{P} \end{cases}$$

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

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

The following theorem is the main result of this section.

Theorem 7. *Let* $\kappa \geq 2$ *be such that* S_κ *does not admit a polynomial time algorithm guaranteeing a maximal independent set greater than* ρn *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(*κ*) guaranteeing an approximation ratio greater than* $\kappa \inf_{f \in \mathcal{F}} \{f(1/2)/f(1)\}$.

Proof: Let us suppose the existence of an approximation algorithm $\mathcal{A}$ for CPM(κ), with a constant ratio $\rho < 1$. Given an instance of CPM(κ), let us denote by $\hat{v}$ the value of the solution $\vec{\hat{x}}$ found by $\mathcal{A}$, and by v^* the value of the optimal solution $\vec{x^*}$; 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(κ) for which $\mathcal{P} = \{\vec{x} \in I\!R^n : A \cdot \vec{x} \leq \vec{1}, \vec{x} \geq \vec{0}\}$. We denote this family by IPM(κ) and we express it in terms of a nonlinear program with linear constraints as follows:

$$\text{IPM}(\kappa) \quad = \quad \begin{cases} \max & \sum_{i \in \{1,\ldots,n\}} f(x_i) \\ & A \cdot \vec{x} \leq \vec{1} \\ & \vec{x} \geq \vec{0} \end{cases}$$

Of course, IPM(κ) being a particular instance of CPM(κ), it can be solved approximately within a ratio ρ. Let us now consider an instance of IPM(κ) and let us denote by $v^*(f)$ its optimal value, and by $\hat{v}(f)$ the value of the approximate solution found by $\mathcal{A}$. Let also $\alpha(G)$ be the stability number of G, let S^* be a maximum independent set of G, and $\vec{x}(S^*)$ the corresponding vector.
Since, $\forall f \in \mathcal{F}$, $\vec{x}(S^*)$ is feasible for IPM and since the objective value of this vector is $f(1)\alpha(G)$, we have, $\forall f \in \mathcal{F}$,

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

On the other hand, let us consider the solution $\vec{\hat{x}}(f)$ of IPM(κ). We then consider the set $\hat{S} = \{i \in \{1,\ldots,n\}, \hat{x}_i(f) > 1/2\}$, where x_i is the ith component of vector $\vec{x}$. It is easy to

see that $\hat{S}$ constitutes an independent set for G. Plainly, the facts: $A \cdot \vec{\hat{x}}(f) \leq \vec{1}$ and $\vec{\hat{x}}(f) \geq \vec{0}$ 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

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

Let us now consider a $\kappa \geq 2$ satisfying proposition 3, and an instance G of S_κ. From expressions (13) and (14), $\forall f \in \mathcal{F}$ and given that $\alpha(G) \leq n$, we have for the cardinality of $|\hat{S}|$: $|\hat{S}| \geq [(nf(1)\rho/\kappa) - nf(1/2)]/[f(1) - f(1/2)]$, or

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

So, the hypothetical algorithm $\mathcal{A}$ for CPM(κ) is used to solve polynomially S_κ within an approximation ratio given by expression (15) and moreover, this ratio does not depend on G. We can thus define a ratio for S_κ for every function f.
Because of the hypothesis on the non-constant approximability of S_κ, all these ratios are zero or negative, or, $\forall f \in \mathcal{F}$, $(f(1)\rho/\kappa) - f(1/2) \leq 0$; consequently, $\rho \leq \kappa f(1/2)/f(1)$ and, by using the property given by expression (12), we obtain $\rho \leq \kappa \inf_{f \in \mathcal{F}}\{f(1/2)/f(1)\}$. ∎

If we take the singleton $\mathcal{F} = \{(x \mapsto x^2)\}$, then CPM($\kappa$) becomes: *find a point of maximum norm $\|\cdot\|_2$ in a polytope*, which is a special case of the quadratic programming problem, and more particularly, of the case of quadratic programming known to be NP-complete ([6]); in this case, $\ell = 1/4$, and the following corollary holds.

Corollary 2. *If S_3 does not admit a polynomial time approximation algorithm of (universally) constant ratio, then the quadratic programming problem does not admit a polynomial time approximation algorithm guaranteeing an approximation ratio greater than or equal to 3/4, unless $P = NP$.*

On the other hand, let us consider the family of all the convex increasing functions f with $f(0) = 0$; then, $\ell = 0$ and we get the following negative result.

Corollary 3. *The problem of maximizing a convex function in a polytope does not admit a constant-ratio polynomial time approximation algorithm, unless $P = NP$.*

7.2 Concave programming and maximum independent set

Let us, once again, consider a $\kappa > 2$ satisfying proposition 3 and the following minimization problem:

$$\text{CPm}(\kappa) \quad = \quad \left\{ \begin{array}{ll} \min & \sum_{i \in \{1,\ldots,n\}} f(1 - x_i) \\ & \vec{x} \in \mathcal{P} \end{array} \right.$$

where f belongs to the family $\mathcal{F}$ of functions increasing in [0,1] with $f(0) = 0$, satisfying the property

$$\sup_{f \in \mathcal{F}} \left\{ \frac{f(\frac{1}{2})}{f(1)} \right\} \in \left] \frac{\kappa - 1}{\kappa}, 1 \right]. \tag{16}$$

Theorem 8. *Let $\kappa \geq 2$ such that S_κ 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 $\mathcal{A}$ with a fixed constant approximation ratio ρ for CPm. Then, to every instance of the maximum independent set problem, we associate the following family of instances of CPm:

$$\text{IPm} = \begin{cases} \min & \sum_{i\in\{1,\cdots,n\}} f(1-x_i) \\ & A\cdot\vec{x} \leq \vec{1} \\ & x_i \geq 0,\ i \in \{1,\ldots,n\} \end{cases}$$

where, as previously, A is the edge-vertex matrix of a graph G.
The approximation algorithm for CPm(κ) solves also the instances of IPm; consequently, $\hat{v}(f) \leq \rho v^*(f)$.
As for theorem 7, given a maximum independent set S^* of G, $\vec{x}(S^*)$ is feasible for IPm and we have, $\forall f \in \mathcal{F}$,

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

On the other hand, let us consider the solution value $\hat{v}(f)$ (given by $\mathcal{A}$) 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, $\forall f \in \mathcal{F}$, $\hat{v}(f) = \sum_{i\in\hat{S}} f(1-\hat{x}_i(f)) + \sum_{i\notin\hat{S}} f(1-\hat{x}_i(f)) \geq |\hat{S}|f(0) + (n-|\hat{S}|)f(1/2)$, or

$$|\hat{S}| \geq n - \frac{\hat{v}(f)}{f\left(\frac{1}{2}\right)}. \tag{18}$$

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

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 $P = NP$.*

input: a graph G admitting a perfect matching;
output: a maximal independent set S';

begin
  $S' \leftarrow \emptyset$
  repeat
    $v_j \leftarrow \text{argmin}_{v_i \in V}\{|\Gamma(v_i)|\}$;
    $S' \leftarrow S' \cup v_j$;
    $V \leftarrow V \setminus (\{v_j\} \cup \Gamma(v_j))$;
    delete from E all edges incident to $\{v_i\} \cup \Gamma(v_i)$;
    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 $\Gamma(v_i)$, $v_i \in V$, the neighbour-set of vertex v_i; we denote by Δ the quantity $\max_{v_i \in V}\{\Gamma(v_i)\}$, 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 $\lceil n/2 \rceil$). 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 Δ, 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 v_i is selected to be added in S', the deletion of vertices in $\Gamma(v_i)$ can be made by treating the edges incident to v_i. On the other hand, the updating of the vertices of the "survived" graph concerns edges $v_j v_k$, $v_j \in \Gamma(v_i)$ and v_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 $\delta = \min_{v_i \in V}\{\Gamma(v_i)\}$ is the minimum degree of the vertices of G. After, for each of the later selections, the algorithm removes at most Δ 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 $\alpha'(G) \geq [[n - (\delta + 1)]/\Delta] + 1$. Since $\delta \leq \Delta$, the above expression results to

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

From expression (2), we get:

$$\alpha(G) = n - m - (f + g) = m + e - (f + g) \tag{20}$$

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):

$$\begin{aligned} m &\leq \frac{n}{2} \\ e - g &\leq 1 \\ f &= \frac{n}{x}. \end{aligned} \tag{21}$$

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

$$\frac{\alpha'(G)}{\alpha(G)} \geq \frac{\frac{n-1}{\Delta}}{\frac{n}{2} + 1 - \frac{n}{x}}. \tag{22}$$

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

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, *Proof verification and intractability of approximation problems*, Proc. FOCS'92, pp. 14-23, 1992.

[2] B. Aspvall and R. E. Stone, *Khachiyan's linear programming algorithm*, J. Algorithms 1, pp. 1-13, 1980.

[3] C. Berge, *Graphs and hypergraphs*, North Holland, Amsterdam, 1973.

[4] J. - M. Bourjolly, P. L. Hammer and B. Simeone, *Node-weighted graphs having the König-Egervary property*, Math. Prog. Study 22, pp. 44-63, 1984.

[5] R. W. Deming, *Independence numbers of graphs - an extension of the König-Egervary theorem*, Disc. Maths 27, pp. 23-33, 1979.

[6] M. R. Garey and D. S. Johnson, *Computers and intractability. A guide to the theory of NP-completeness*, W. H. Freeman and Company, San Francisco, 1979.

[7] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*, Prentice Hall, New Jersey, 1981.