UNIVERSITE PARIS DAUPHINE

Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision

CNRS UMR 7024

CAHIER DU LAMSADE

271

Janvier 2008

Efficient approximation by "low-complexity" exponential algorithms

Nicolas Bourgeois, Bruno Escoffier, Vangelis Th. Paschos

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Efficient approximation by "low-complexity" exponential algorithms

N. Bourgeois B. Escoffier V. Th. Paschos

LAMSADE, CNRS UMR 7024 and Université Paris-Dauphine, France
{bourgeois,escoffier,paschos}@lamsade.dauphine.fr

December 30, 2007

Abstract

This paper proposes a way to bring together two seemingly "foreign" domains that are the polynomial approximation and the exact computation for **NP**-hard problems. We show how one can match ideas from both areas in order to design approximation algorithms achieving ratios unachievable in polynomial time (unless a very unlikely complexity conjecture is confirmed) with worst-case complexity much lower (though super-polynomial) than that of an exact computation. We then apply these ideas to two famous combinatorial optimization problems, namely, the MAX INDEPENDENT SET and the MIN VERTEX COVER, as well as to some other problems mainly linked to MAX INDEPENDENT SET by simple approximation preserving reductions.

1 Introduction

The two most known paradigms to come up with **NP**-hard problems are either the exact computation (i.e., the computation of optimal solutions for them), or the heuristic resolution, i.e., the development of fast algorithms that compute hopefully near-optimal solutions. Notable part of the heuristic paradigm is the so-called *polynomial approximation* where one tries to devise polynomial algorithms for **NP**-hard problems computing feasible solutions that are close to optimal under an a priori criterion called *approximation ratio*.

Both exact computation and polynomial optimization are very active areas in theoretical computer science and combinatorial optimization. Dealing with the former, very active research has been conducted around the development of optimal algorithms with non-trivial worst-case complexity. As an example, let us consider MAX INDEPENDENT SET. Any optimal algorithm can solve it with complexity $O^*(2^n)$ (where $O^*(\cdot)$ is as $O(\cdot)$ ignoring polynomial factors), where n is the order of G (i.e, the cardinality of V) by exhaustively examining all the subsets in 2^V and by taking the largest among them that forms an independent set; hence, an interesting question is if we can compute a maximum independent set with complexity $O^*(\gamma^n)$, for $\gamma < 2$. More about such issues for several combinatorial problems can be found in the seminal paper by [22]. This area has known a renew of the researcher's interest due to numerous pessimistic results in polynomial approximation, but also due to the fantastic increase of the computational power of modern computers.

On the other hand, dealing with polynomial approximation, very intensive research since the beginnings of 70's has lead to numerous results exhibiting possibilities but also limits to the approximability of **NP**-hard problems. Such limits are expressed as statements that a given problem cannot be approximated within a certain approximation level (for instance, within a constant approximation ratio) unless a very unlikely complexity condition (e.g., $\mathbf{P} = \mathbf{NP}$)

holds. A very rich landscape of the polynomial approximation area can be found in [2, 15, 21]. Since the beginning of 90's, and using the celebrated PCP theorem ([1]), numerous natural hard optimization problems are proved to admit more or less pessimistic inapproximability results. For instance, MAX INDEPENDENT SET is inapproximable within approximation ratio better than $n^{\epsilon-1}$, unless $\mathbf{P} = \mathbf{NP}$ ([14]).

These two areas has remained foreign until now. Researchers in any of them produced results fitting the corresponding paradigm and without links with the other one. Staring point of this work is the idea that both of them can be linked by mutual exchanges of tools and concepts, in order that new results and ideas are established handling solution mechanisms of **NP**-hard problems. For instance, it is interesting to efficiently approximate such problems by devising algorithms that achieve approximation ratios that cannot be achieved in polynomial time, with a worst-case complexity that is significantly lower (though super-polynomial) than the complexity of a exact computation. This issue is called *efficient approximation* in what follows.

2 Preliminaries

An optimization problem Π is in **NPO** if the decision version of Π is in **NP**. Formally, an **NP** optimization problem Π is defined as a four-tuple $(\mathcal{I}, \mathrm{sol}, m, \mathrm{opt})$ such that: $\mathcal{I}$ is the set of instances of Π and it can be recognized in polynomial time; given $x \in \mathcal{I}$, $\mathrm{sol}(x)$ denotes the set of feasible solutions of x; for every $y \in \mathrm{sol}(x)$, $|y|$ is polynomial in $|x|$; given any x and any y whose length is polynomial in $|x|$, one can decide in polynomial time if $y \in \mathrm{sol}(x)$; given $x \in \mathcal{I}$ and $y \in \mathrm{sol}(x)$, $m(x, y)$ denotes the value of y for x; m is polynomially computable and is commonly called feasible value, or objective value; finally, $\mathrm{opt} \in \{\max, \min\}$ denotes the optimization goal for Π. The set of **NP** optimization problems forms the class **NPO**. Given an instance x of an **NPO** problem $\Pi = (\mathcal{I}, \mathrm{sol}, m, \mathrm{opt})$, and a feasible solution y for x, we denote by $\mathrm{opt}(x)$ the value of an optimal solution of x.

For an approximation algorithm $\mathtt{A}$ computing a feasible solution y for x with value $m_{\mathtt{A}}(x, y)$, its approximation ratio on y is defined as $\rho^{\mathtt{A}}_{\Pi}(x, y) = m_{\mathtt{A}}(x, y)/\mathrm{opt}(x)$. The approximation ratio $\rho^{\mathtt{A}}_{\Pi}$ of $\mathtt{A}$ is then defined as the worst[1] over any instance $x \in \mathcal{I}$, of $\rho^{\mathtt{A}}_{\Pi}(x, y)$. In what follows, whenever it is understood, references to problem Π and/or $\mathtt{A}$ will be dropped.

Polynomial approximation is a very active area since the beginning of the 70's. The celebrated paper by [16] is considered as the startpoint of this research programme that has dominated a large part of the research conducted in complexity theory. On the other hand, exact solution of combinatorial problems is a natural requirement for these problems that remains in the heart of the research in combinatorial optimization and in operational research more generally. These two approaches are complementary in the sense that, informally, the former gives priority to fast computation of feasible solutions against optimality, while, for the latter, priority is given to solutions' optimality against speed of such computation.

If the area of exact computation is in the heart of combinatorial optimization since the beginnings of this domain, the main concerns of a large majority of its researchers were rather about the design of clever solution algorithms (mainly based upon tree-search procedures, dynamic programming, etc.) than the precise estimation of their running-time. Before the middle of 90's, when a broad research programme around such concerns has been built, fairly little research has been dedicated to this issue. On the other hand, numerous open questions, posed since the beginnings of polynomial approximation, as for example, the approximation of MAX INDEPENDENT SET[2] within constant ratio, have received strongly negative answers at the beginnings

[1] The min if Π is a maximization problem, the max, otherwise.

[2] Given a graph $G(V, E)$, MAX INDEPENDENT SET consists of finding a set $S \subseteq V$ of maximum size such that for any $(u, v) \in S \times S$, $(u, v) \notin E$.

of 90's with the proof of the famous PCP theorem (carrying over a novel and fine characterization of **NP**, [1]). Similar answers, known as *inapproximability* or *negative results* in polynomial approximation theory, have been provided for numerous other paradigmatic optimization problems, as MIN SET COVER, MIN VERTEX COVER[3], MIN COLORING, etc.

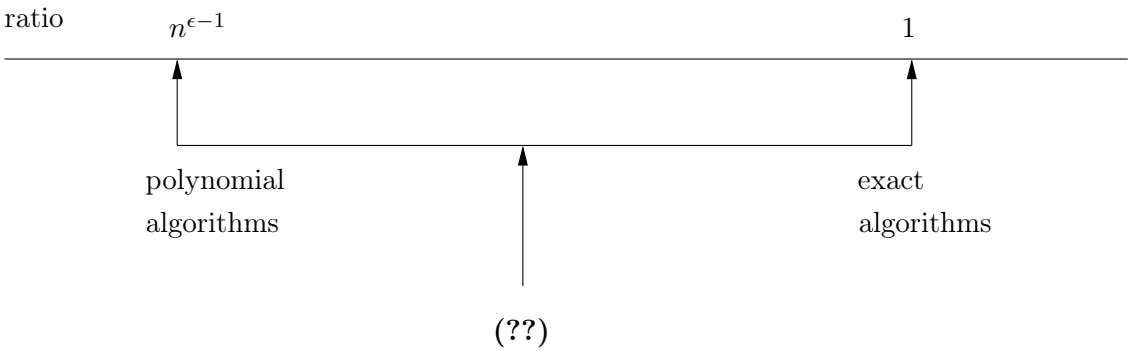

Figure 1: The approximability gap for MAX INDEPENDENT SET.

These results exhibit large gaps between what it is possible to do in polynomial time and what becomes possible in exponential time. Let us take, once again, the case of MAX INDEPENDENT SET. It is proved in [14] that this problem is inapproximable within ratio better than $O(n^{\epsilon-1})$, unless $\mathbf{P} = \mathbf{NP}$ (note that any approximation algorithm trivially achieves approximation ratio $O(n)$ in polynomial time). We so are faced with a huge gap impossible to be bridged in polynomial time (Figure 1).

Hence, a natural question is how much time takes the computation of an r-approximate solution, for $r \in [n^{\epsilon-1}, 1[$? Of course, we have a lower bound to this time (any polynomial to the size of the instance) and also an upper bound (the running time of exact computation). But:

- can we devise, for some ratio r, an r-approximate algorithm with an improved running time located somewhere between these bounds?
- is this possible for *any* ratio r, i.e., can we specify a global relationship between running time and approximation ratio?

In this paper, we try to bring answers to these questions by matching ideas and results from exact computation (mainly dealing with improved worst-case complexities of optimal algorithms for the problems handled) and from polynomial approximation (mainly around approximation preserving reductions). This issue has been also handled very recently by [6, 9, 10], though in a different setting and with different objectives oriented towards development of fixed-parameter algorithms.

Note finally that tradeoffs between approximation ratio and running time have already been studied for polynomially solvable problems (but with practically long running times) such as maximum matching.

In what follows, in Section 3, we give some easy examples of approximation algorithms for MAX INDEPENDENT SET that is the pivot problem of this paper. The algorithms given can achieve ratios impossible to be polynomially achieved with non-trivial super-polynomial or exponential worst-case complexity.

[3]Given a graph $G(V, E)$, MIN VERTEX COVER consists of finding a set $C \subseteq V$ of minimum size such that, for every $(u, v) \in E$, either u, or v belongs to C.

In Section 4, we give efficient approximation results of a broad class of maximization graph-problems where their solutions are subgraphs of the input-graph that satisfy some non-trivial hereditary property[4].

In Section 5 we develop efficient non-polynomial approximation algorithms for another paradigmatic problem in combinatorial optimization, the MIN VERTEX COVER. Results in these two sections are obtained by a basic technique consisting of optimally solving the problems handled in subgraphs of the input graph obtained by partition of its vertices.

In Section 6, we propose randomized approaches that improve complexity results obtained in Sections 4 (in particular for the case of MAX INDEPENDENT SET) and 5.

In Section 7, we consider specific classes of graphs where MAX INDEPENDENT SET is polynomially approximable. We show there how approximation algorithms for MAX INDEPENDENT SET can be improved in order to guarantee any approximation ratio with low exponential complexity. The method presented in this section is somewhat complementary to the one in Sections 4 and 5. Here, we partition the input-graph into two subgraphs and we apply an exhaustive search to one of them and an approximation algorithm in the other one.

Finally, in Section 8, we present approximation results for other combinatorial problems linked to MAX INDEPENDENT SET by simple approximation-preserving reductions. In particular, for one of the problems handled in this section that is MAX CLIQUE, we also produce an exact computation parametric result that is interesting per se.

Note that the algorithms devised in Sections 4 to 8 use as subroutines exact algorithms for computing optimal solutions of hard problems. In this sense, the major part of these results can be seen as reductions from approximate computation to exact computation on "small" instances. In particular, any improvement of the running times of exact algorithms for the problems handled, would immediately result in improvements of running times of our approximation algorithms. Besides, this is one of the reasons for which we study sometimes several algorithms whose complexity depends on several parameters.

Before closing this section we give some notations that will be used later. Let $T(\cdot)$ be a super-polynomial and $p(\cdot)$ be a polynomial, both on integers. In what follows, using notations in [22], for an integer n, we express running-time bounds of the form $p(n) \cdot T(n)$ as $O^*(T(n))$ by ignoring, for simplicity, polynomial factors. We denote by $T(n)$ the worst-case time required to solve the considered combinatorial optimization problem with n variables. We recall (see, for instance, [11]) that, if it is possible to bound above $T(n)$ by a recurrence expression of the type $T(n) \leqslant \sum T(n - r_i) + O(p(n))$, we have $\sum T(n - r_i) + O(p(n)) = O^*(\alpha(r_1, r_2, \ldots)^n)$ where $\alpha(r_1, r_2, \ldots)$ is the largest zero of the function $f(x) = 1 - \sum x^{-r_i}$.

Given a graph $G(V, E)$, we denote by n the size of V, by $\alpha(G)$ the size of a maximum independent set of G and by $\tau(G)$ the size of a minimum vertex cover of G. Also, we denote by $\Delta(G)$ the maximum degree of G. Given a subset V' of V, $G[V']$ denotes the subgraph of G induced by V'. Sometimes, for a graph G, we denote by $V(G)$ its vertex-set.

3 Simple efficient approximation results for MAX INDEPENDENT SET

3.1 Generating a "small" number of candidate solutions

Consider a graph $G(V, E)$ of order n and run the following algorithm:

- generate all the $\sqrt{n}$-subsets (subsets of cardinality $\sqrt{n}$) of V;

[4] A graph G is said to satisfy a hereditary property π if every subgraph of G satisfies π whenever G satisfies π. Furthermore, π is non-trivial if it is satisfied for infinitely many graphs and it is false for infinitely many graphs; for instance, properties "independent set", "clique", "planar graph", "k-colorable graph", etc., are non-trivial hereditary properties.

- if one of them is independent, then output it;
- otherwise output a vertex at random.

It is easy to see that the approximation ratio of this algorithm is $n^{-1/2}$. Indeed, if algorithm's output is done at the second item, i.e., an independent set of size $\sqrt{n}$ is discovered, then, since $\alpha(G) \leqslant n$, the approximation ratio achieved is at least $\sqrt{n}/n = n^{-1/2}$. On the other hand, if no independent set is found at the second step, then $\alpha(G) \leqslant \sqrt{n}$ and the approximation ratio guaranteed in third step is at least $1/\sqrt{n} = n^{-1/2}$, impossible for polynomial algorithms according to [14].

The complexity of the algorithm above is roughly bounded above by $O^*(\binom{n}{\sqrt{n}}) = O^*(2^{\sqrt{n}\log n})$, much lower than the best known exact complexity for MAX INDEPENDENT SET that is $O^*(1.18^n)$ due to [19].

3.2 Approximating by pruning the search tree

The most common tool used to devise exponential algorithm with non-trivial worst case complexity consists of pruning the search tree ([22]). We show in this section that pruning can be properly adapted to devise approximation algorithms with improved worst-case complexity. The running times that we obtain via this method are worse than the ones obtained in the next sections. But our goal here is to illustrate on a simple example how to approximate by pruning.

Consider a simple search tree-based algorithm for solving MAX INDEPENDENT SET, which consists of recursively applying the following rule (see for instance [22]):

1. if $\Delta(G) \leqslant 2$, then output a maximum independent set;
2. else, branch on a vertex v with degree at least 3 as follows:
   (a) either take v and solve MAX INDEPENDENT SET in the subgraph surviving after the removal of v and its neighbors;
   (b) or do not take v, and solve MAX INDEPENDENT SET in the subgraph surviving after the removal of v.

Step 1 can be done in polynomial time. On the other hand, when branching, we have to solve a subproblem of size either $n - \Delta(v) - 1 \leqslant n - 4$, or $n - 1$. This leads to a running time $T(n) \leqslant T(n-1)+T(n-4)+p(n)$, for some polynomial p, which comes up to $T(n) \leqslant O^*(1.381^n)$.

We now explain how one can get a $1/2$-approximation algorithm based on the above algorithm, with running time much better than $O^*(1.381^n)$. The idea is that, when a branching occurs, in case 2a both our algorithm and an optimum solution take v. In this case, if we only seek a $1/2$-approximate solution, then roughly speaking, the algorithm can make an error on another vertex (not taking it in the solution while an optimal solution takes it). Indeed, vertex v taken in both solutions compensates this error. So, when applying the branching, in case 2a we can remove any other vertex of the graph. We then get a subproblem of size $n - 5$ instead of $n-4$. More generally, consider an edge (v_i, v_j) in the surviving graph (or even a clique K). Since an optimal solution can take at most one vertex of a clique, then when branching in case 2a, we can remove vertices v_i and v_j (resp., the whole clique K).

A second improvement deals with step 1. Indeed, we do not need to deal with cases where the optimum can be polynomially reached, but with cases where a $1/2$-approximate solution can be found in polynomial time. For instance, MAX INDEPENDENT SET can be approximately solved in polynomial time within approximation ratio $(\Delta(G)+3)/5$ [5]. Hence, if $\Delta(G) \leqslant 7$, then MAX INDEPENDENT SET is $1/2$-approximable in G. This leads to the following algorithm:

1. if $\Delta(G) \leqslant 7$, then run the algorithm by [5];
2. else, branch on a vertex v with degree at least 8 as follows:
   (a) either take v, and solve MAX INDEPENDENT SET in the subgraph surviving after the removal of v, of its neighbors and and of two other adjacent vertices v_i, v_j;
   (b) or do not take v, and solve the problem in the subgraph surviving after the removal of v.

It is easy to recursively verify that the algorithm above guarantees an approximation ratio $1/2$. Concerning its running time, during step 2a we remove 11 vertices (note that if there is no edge (v_i, v_j) to be removed, the surviving graph is an independent set per se); hence, $T(n) \leqslant T(n-1) + T(n-11) + p(n)$. This leads to $T(n) = O^*(1.185^n)$.

Note that the above algorithm can be generalized to find a $1/k$-approximation algorithm (for any integer k) in time $T(n) \leqslant T(n-1)+T(n-7k+3)+p(n)$. Obviously, improved running times would follow from considering, for example, either modifications of algorithms more sophisticated than the one presented in this section, or a more efficient counting technique such as the one presented in [12]. However, up to now, the techniques presented in next sections give better results.

4 Maximum induced subgraph problems with property π

We handle in this section a large class of graph-problems that is defined as follows: given a graph $G(V, E)$ and some hereditary property π, find a subset $V' \subseteq V$, of maximum size, such that the subgraph of G induced by V' satisfies the property π. For a fixed property π we denote by MAX HEREDITARY-π the particular **NPO** problem resulting when considering π. For instance, if π is "independent set", then MAX HEREDITARY-"independent set" is exactly MAX INDEPENDENT SET.

The idea of the method proposed consists in splitting the instance into several subinstances (of much smaller size) and in solving the problem on these subinstances using an exact algorithm. The ratio obtained is directly related to the size of the subinstances, hence to the global running time of the algorithm.

Proposition 1. *Fix a hereditary property π and assume that there exists an exact algorithm $\mathcal{A}$ for* MAX HEREDITARY-π *with worst-case complexity $O^*(\gamma^n)$ for some $\gamma \in \mathbb{R}$, where n is the order of the input-graph, for* MAX HEREDITARY-π. *Then for any $\rho \in \mathbb{Q}$, $\rho \leqslant 1$, there exists a ρ-approximation algorithm for* MAX HEREDITARY-π *that runs in time $O^*(\gamma^{\rho n})$.*

Proof. Consider a graph G of order n and fix a rational $\rho \leqslant 1$. Since $\rho \in \mathbb{Q}$, it can be written as $\rho = p/q$, $p, q \in \mathbb{N}$, $p \leqslant q$.

Consider now the following algorithm, called with parameters G and ρ:

1. arbitrarily partition G into q induced subgraphs $G_1, \ldots, G_q$ of order (except eventually for G_q) n/q;
2. build the q subgraphs $G'_1, \ldots G'_q$ that are unions of p consecutive subgraphs $G_{i+1}, \ldots, G_{i+p}$, $i = 1, \ldots, q$ (where of course $G_{q+1} = G_1$);
3. optimally solve MAX HEREDITARY-π in every G'_i, $i = 1, \ldots, q$;
4. output the best of the solutions computed in step 3.

Denote by S the solution output by the algorithm and fix an optimal solution S^* of G (following notations in Section 2, $|S^*| = \mathrm{opt}(G)$). Then, $|S| \geqslant (p/q)\,\mathrm{opt}(G) = \rho\,\mathrm{opt}(G)$.

Indeed, let $S_i^* = S^* \cap G_i$. Then, by heredity, $|S_{i+1}^*| + |S_{i+2}^*| + \ldots + |S_{i+p}^*| \leqslant \mathrm{opt}(G_i') \leqslant |S|$. Summing up for $i = 1, 2, \ldots, q$, we get: $p|S^*| = p\sum_{i=1}^{q} |S_i^*| \leqslant q|S|$, that proves the approximation ratio claimed.

It is easy to see that the above algorithm involves q executions of `A` (the exact algorithm for MAX HEREDITARY-π) on graphs of order roughly $pn/q = \rho n$. Hence, its complexity is of $O^*(\gamma^{\rho n})$, that proves the running time claimed and the proposition. ∎

Obviously, Proposition 1 holds for several hereditary properties as "independent set", "clique", "planar graph", "bipartite graph", etc.

Let us now focus on the most notorious among these properties that is "independent set". Denote by `IS` the instantiation of the algorithm above to "independent set" and assume that it is parameterized by two parameters: the input-graph G and the ratio ρ to be achieved. For the rest of the paper assume that there exists an exact algorithm for MAX INDEPENDENT SET with worst-case running time $O^*(\gamma^n)$ (to the best of our knowledge, the best γ currently known for general graphs is 1.18 due to [19]). We then have the following immediate corollary.

Corollary 1. *For any $\rho \leqslant 1$, Algorithm `IS`$(\mathtt{G}, \rho)$ computes a ρ-approximation of* MAX INDEPENDENT SET *with running time $O^*(\gamma^{\rho n})$.*

From Proposition 1 we can note the following two interesting facts:

1. the algorithm of Proposition 1 can be implemented to use *polynomial space* provided that the exact algorithms used do so;

2. any improvement to the basis γ of the exponential for the running time of the exact algorithm for MAX HEREDITARY-π is immediately transferred to Proposition 1.

Note also that the result of Corollary 1 slightly improves the result in Section 3.1 for $\rho = \sqrt{n}$, as well as the result in Section 3.2 (dealing with pruning the search tree) for $\rho = 1/2$.

5 Efficient approximations for MIN VERTEX COVER

There exists a very close and well-known relation between a vertex cover and an independent set in a graph $G(V, E)$ ([4]): if S is an independent set of G, then the set $V \setminus S$ is a vertex cover of G. The same complementarity relation holds obviously for a maximum independent set S^* and the set $C^* = V \setminus S^*$ that is a minimum vertex cover of G.

MIN VERTEX COVER is approximable within approximation ratio 2 and one of the most known open problems in polynomial approximation is either to improve this ratio, or to prove that such an improvement is impossible until a strong unlikely complexity condition (e.g., $\mathbf{P} = \mathbf{NP}$) holds. A recent result by [17] gives a strong evidence that the latter alternative might be true.

On the other hand, from an exact computation point of view, the relation between MIN VERTEX COVER and MAX INDEPENDENT SET has as immediate corollary that an optimal vertex cover can be determined in $O^*(\gamma^n)$. Furthermore, the following parameterized complexity result is proved by [7].

Theorem 1. *([7]) For any $k \leqslant n$, there exists an algorithm that determines if a graph G contains a vertex cover of size k or not and, if yes, it computes it with complexity $O^*(\delta^k)$. The currently best δ known for general graphs is equal to 1.28.*

We now recall the seminal result by [18] characterizing the polytope of MAX INDEPENDENT SET (or, equivalently, of MIN VERTEX COVER). Before, for readability, let us recall the integer linear

program of MAX INDEPENDENT SET (denoted also by is) as well as the mathematical program of its linear programming relaxation (LP-relaxation), denoted by MAX INDEPENDENT SET-R. Given a graph G, denoting by A its incidence matrix:

$$\text{MAX INDEPENDENT SET} = \begin{cases} \max & \vec{1} \cdot \vec{x} \\ & A\vec{x} \leqslant \vec{1} \\ & \vec{x} \in \{0,1\}^n \end{cases}$$

$$\text{MAX INDEPENDENT SET-R} = \begin{cases} \max & \vec{1} \cdot \vec{x} \\ & A\vec{x} \leqslant \vec{1} \\ & \vec{x} \in (\mathbb{Q}^n)^+ \end{cases}$$

Obviously, solution of MAX INDEPENDENT SET-R can be done in polynomial time.

Theorem 2. *([18]) The basic optimal solution of the LP-relaxation of* MAX INDEPENDENT SET *is semi-integral, i.e., it assigns to the variables values from* $\{0, 1, 1/2\}$. *Let* V_0, V_1 *and* $V_{1/2}$ *be the subsets of* V *associated with 0, 1 et 1/2, respectively. There exists a maximum independent set* S^* *such that* $V_1 \subseteq S^*$ *and* $V_0 \subseteq C^* = V \setminus S^*$.

Obviously, Theorem 2 identically holds for MIN VERTEX COVER also.

Corollary 2. $\alpha(G[V_{1/2}]) \leqslant |V_{1/2}|/2$, $\tau(G[V_{1/2}]) \geqslant |V_{1/2}|/2$. *Also, denoting by* S' *and* C' *an independent set and a vertex cover of* $G[V_{1/2}]$, $S = V_1 \cup S'$ *is an independent set of* G *and* $C = V \setminus S = V_0 \cup C'$ *is a vertex cover of* G.

The following lemma that links approximabilities of MAX INDEPENDENT SET and MIN VERTEX COVER will be extensively used in what follows.

Lemma 1. *If* MAX INDEPENDENT SET *is approximable within approximation ratio* ρ, *then* MIN VERTEX COVER *is approximable within ratio* $2 - \rho$.

Proof. Let $\mathtt{A}$ be an MAX INDEPENDENT SET-algorithm computing an independent set S guaranteeing $|S| \geqslant \rho\alpha(G)$ for some $\rho < 1$. Run the following algorithm for MIN VERTEX COVER:

1. solve the LP-relaxation of MAX INDEPENDENT SET in G to produce sets V_0, V_1 and $V_{1/2}$;
2. run $\mathtt{A}$ in $G[V_{1/2}]$;
3. return $C = V_0 \cup (V_{1/2} \setminus \mathtt{A}(\mathtt{G}[\mathtt{V}_{1/2}])$.

By Corollary 2, $V_1 \cup \mathtt{A}(\mathtt{G}[\mathtt{V}_{1/2}])$ is an independent set of G and $C = V \setminus (V_1 \cup \mathtt{A}(\mathtt{G}[\mathtt{V}_{1/2}])) = V_0 \cup (V_{1/2} \setminus \mathtt{A}(\mathtt{G}[\mathtt{V}_{1/2}]))$ is a vertex cover. Then, the approximation ratio of C is:

$$\frac{|C|}{\tau(G)} = \frac{|V_0| + \left(V_{1/2} \setminus \mathtt{A}\left(\mathtt{G}\left[\mathtt{V}_{1/2}\right]\right)\right)}{|V_0| + \tau\left(G\left[V_{1/2}\right]\right)} \leqslant \frac{\left(V_{1/2} \setminus \mathtt{A}\left(\mathtt{G}\left[\mathtt{V}_{1/2}\right]\right)\right)}{\tau\left(G\left[V_{1/2}\right]\right)} \quad (1)$$

$$\leqslant \frac{\left|V_{1/2}\right| - \left|\mathtt{A}\left(\mathtt{G}\left[\mathtt{V}_{1/2}\right]\right)\right|}{\left|V_{1/2}\right| - \alpha\left(G\left[V_{1/2}\right]\right)} \leqslant \frac{\left|V_{1/2}\right| - \rho\alpha\left(G\left[V_{1/2}\right]\right)}{\left|V_{1/2}\right| - \alpha\left(G\left[V_{1/2}\right]\right)} = 1 + \frac{1-\rho}{\frac{|V_{1/2}|}{\alpha(G[V_{1/2}])} - 1} \quad (2)$$

According to Corollary 2, $|V_{1/2}|/\alpha(G[V_{1/2}]) \geqslant 2$. Putting this together with (2), we get, after some easy algebra: $|C|/\tau(G) \leqslant 2 - \rho$, q.e.d. ∎

Let us note that, as it can be immediately seen from (1), when tackling approximation of MAX INDEPENDENT SET and of MIN VERTEX COVER, we can restrict ourselves to subgraph $G[V_{1/2}]$, instead of the whole G. By Corollary 2, $\alpha(G[V_{1/2}]) \leqslant |V_{1/2}|/2$ and $\tau(G[V_{1/2}]) \geqslant |V_{1/2}|/2$.

According to Corollary 2 and Lemma 1, MIN VERTEX COVER can be approximately solved by the following algorithm called `VC1`:

1. solve the LP-relaxation of MAX INDEPENDENT SET to obtain sets V_1, V_0 and $V_{1/2}$ (this step runs in polynomial time);
2. set $G = G[V_{1/2}]$ and run $\texttt{IS}(\texttt{G}, \rho)$;
3. output $V \setminus (V_1 \cup \texttt{IS}(\texttt{G}, \rho))$.

Combination of Corollary 1 and Lemma 1, immediately derives the following result.

Theorem 3. *For any $\rho \leqslant 1$, Algorithm $\texttt{VC1}(\rho, \texttt{G})$ computes a $2-\rho$-approximation of* MIN VERTEX COVER *with running time $O^*(\gamma^{\rho n})$.*

In other words, any approximation ratio $r \in [1, 2[$ for MIN VERTEX COVER can be attained by Algorithm VC1, with complexity $O^*(\gamma^{(2-r)n})$.

In what follows in this section, we first improve the result of Theorem 3, by showing that ratio $r > 1$ for MIN VERTEX COVER can be achieved with lower worst case complexity (function of n). Next, we give a parameterized approximation result analogous to the one of Theorem 1.

5.1 Improving running time for MIN VERTEX COVER's approximation

Our goal in this section is to improve Theorem 3, i.e., to show how we can get approximation ratio ρ, for every $\rho > 1$, in time smaller than $O^*(\gamma^{(2-\rho)n})$.

For this, we propose a method based upon a tradeoff between the exact algorithm in $O^*(\gamma^n)$ for MAX INDEPENDENT SET and the fixed-parameter algorithm in $O^*(\delta^k)$ for MIN VERTEX COVER. Indeed, if $\tau(G)$ is small, then of course the latter algorithm is fast (see Lemma 3). On the other hand, if $\tau(G)$ is large, then the result of Theorem 3 can be further improved (see Lemma 2).

Recall that, in the sequel, we suppose that we work on graph $G[V_{1/2}]$. For simplicity we use V instead of $V_{1/2}$ and G instead of $G[V_{1/2}]$.

Lemma 2. *If, for some $\lambda < 1/2$, $\alpha(G) \leqslant \lambda n$, then a ρ-approximation for* MIN VERTEX COVER *can be found in $O^*(\gamma^{(\rho-(\rho-1)/\lambda)n})$.*

Proof. Note first that, if $\lambda < 1/2$, then $\rho - ((\rho-1)/\lambda) < 2-\rho$.

Fix a ratio ρ to be achieved for MIN VERTEX COVER and denote by $\alpha'(G)$ the cardinality of some independent set of G. From (2), setting $r = \alpha'(G)/\alpha(G)$, $\tau'(G)/\tau(G) \leqslant (1-r\lambda)/(1-\lambda)$. So, a ratio ρ for MIN VERTEX COVER can be achieved for some r verifying:

$$\rho = \frac{1-r\lambda}{1-\lambda} \Longrightarrow r = \rho - \frac{\rho-1}{\lambda} \overset{\lambda<1/2}{<} 2-\rho \tag{3}$$

We distinguish two cases depending on the sign of r, namely $r \leqslant 0$ and $r > 0$.

If $r \leqslant 0$, then $\lambda \leqslant (\rho-1)/\rho$. In this case, the whole vertex-set of the input-graph is a vertex cover per se guaranteeing approximation ratio ρ. Indeed, apply (2) with $r = \alpha'(G)/\alpha(G) = 0$ and remark that it is increasing with $\lambda = \alpha(G)/n$. Some very easy algebra shows that V guarantees by itself an approximation ratio ρ for MIN VERTEX COVER.

Assume now that $r \geqslant 0$. Take $V \setminus \texttt{IS}(\texttt{G}, \texttt{r})$, with $r = \rho - ((\rho-1)/\lambda)$, as MIN VERTEX COVER-solution. By Corollary 1, this can be done with complexity $O^*(\gamma^{rn}) = O^*(\gamma^{(\rho-(\rho-1/\lambda))n}) < O^*(\gamma^{(2-\rho)n})$ and, from (3), it guarantees ratio ρ. ∎

Lemma 3. *If, for some $\lambda < 1/2$, $\alpha(G) \geqslant \lambda n$, then a ρ-approximation of* MIN VERTEX COVER *can be found in $O^*(\delta^{(2-\rho)(1-\lambda)n})$.*

Proof. Fix a $\rho > 1$, set $p/q = 2-\rho$, denote by $\texttt{OPT\_VC}(\texttt{G}, \texttt{k})$ the algorithm of Theorem 1 and run the following algorithm denoted by $\texttt{PROCEDURE\_VC}(\texttt{G}, \rho)$:

1. partition $G(V,E)$ into q induced subgraphs $G_1,\ldots,G_q$ of order n/q (except eventually for G_q);

2. build subgraphs $G'_1,\ldots G'_q$ that are the unions of p consecutive subgraphs $G_{i+1},\ldots,G_{i+p}$;

3. for $i=1,\ldots,q$, run $\texttt{OPT\_VC}(G'_i,(1-\lambda)(2-\rho)n)$ and store the best (say C'_{i^*}) among the covers satisfying $\tau(G'_i)\leqslant(1-\lambda)(2-\rho)n$ (if any);

4. if such a cover C'_{i^*} has been computed in Step 3 for G'_{i^*}, then output $C=C'_{i^*}\cup(V\setminus V(G'_{i^*}))$, else exit.

We now prove the following fact.

Fact 1. If $\alpha(G)\geqslant\lambda n$, then there exists a graph G'_{i^*} where a cover C'_{i^*} satisfying $\tau(G'_{i^*})\leqslant(1-\lambda)(2-\rho)n$ has been computed during step 3 of Algorithm PROCEDURE_VC. ∎

Indeed, as shown in the proof of Proposition 1, there exists a subgraph G'_{i^*} (among those built in step 2) for which $\alpha(G'_{i^*})\geqslant(p/q)\alpha(G)\geqslant(2-\rho)\lambda n$. Hence, the following holds for G'_{i^*}: $|C_{i^*}|=\tau(G'_{i^*})\leqslant(2-\rho)n-(2-\rho)\lambda n=(1-\lambda)(2-\rho)n$, that proves Fact 1.

According to Lemma 1, since $S'_{i^*}=V(G'_{i^*})\setminus C'_{i^*}$ is a $2-\rho$ approximation for MAX INDEPENDENT SET, then the cover C returned by PROCEDURE_VC is a ρ-approximation for MIN VERTEX COVER.

Finally, assuming that ρ is constant, the running time of Algorithm $\texttt{PROCEDURE\_VC}(G,\rho)$ is $O^*(\delta^{(2-\rho)(1-\lambda)n})$ as claimed. ∎

Consider now the following algorithm denoted by VC2 in what follows and run with parameters G and ρ:

1. fix $\rho>1$ and determine λ satisfying $\gamma^{(\rho-(\rho-1/\lambda))}=\delta^{(1-\lambda)(2-\rho)}$ (the first increases, while the second decreases with λ);

2. solve the LP-relaxation of MAX INDEPENDENT SET, store V_0 and V_1 and set $G=G[V_{1/2}]$;

3. set $\rho-((\rho-1)/\lambda)=p/q$ and compute $C_0=V_0\cup(V\setminus\texttt{IS}(\texttt{G},2-\rho))$;

4. set $p/q=2-\rho$ and compute $C=V_0\cup\texttt{PROCEDURE\_VC}(\texttt{G},\rho)$;

5. output the best among C_0 and C.

Based upon Lemmata 2 and 3, the following theorem holds and concludes the section.

Theorem 4. *For any $\rho>1$,* MIN VERTEX COVER *can be solved approximately by Algorithm VC2 within ratio ρ and with running time $O^*(\gamma^{(\rho-(\rho-1)/\lambda)n})$.*

Revisit now step 1 of Algorithm VC2 where parameter λ were determined as solution of the equation $\gamma^{\rho-\frac{\rho-1}{\lambda}}=\delta^{(1-\lambda)(2-\rho)}$. This equation is, indeed, a second-degree equation whose solution is given by:

$$\lambda=\frac{r\log\delta-\rho\log\gamma+\sqrt{\rho^2\log^2\gamma+(2-\rho)^2\left(\log^2\delta-2\log\gamma\log\delta\right)}}{2(2-\rho)\log\delta}$$

The improvement obtained with this algorithm is illustrated in Table 1 (at the end of Section 5). To conclude, note that this improvement cannot be transferred to MAX INDEPENDENT SET, since it is based upon Lemma 1 that does not work in both directions.

5.2 Another efficient approximation algorithm for MIN VERTEX COVER

As we have already mentioned (Theorem 1), there exists an exact algorithm for MIN VERTEX COVER (denoted by `OPT_VC`), with worst-case complexity $O^*(\delta^k)$ that decides if a graph G has a vertex cover of size k or not and, if yes, it computes it. In this section, we extend this result to deal with efficient approximation of MIN VERTEX COVER. The technique used is the same as in Proposition 1, up to the facts that the problem is not hereditary and the algorithm upon which it is based is parameterized.

Theorem 5. *For every graph G and for any $r = p/q \in \mathbb{Q}$, if there exists a solution for minimal vertex cover whose size is less than k, it is possible to determine with complexity $O^*(\delta^{rk})$ a $2 - r$-approximation of it.*

Proof. As we have pointed out in Lemma 1, anything we do can be w.l.o.g. with respect to the graph $G[V_{1/2}]$.

Consider the following algorithm, denoted by `VC3` and called with parameters r and G:

1. arbitrarily partition G into q induced subgraphs $G_1, \ldots, G_q$ of order (except eventually for G_q) n/q;
2. form the subgraphs $G'_1, \ldots G'_q$ that are the unions of p consecutive subgraphs $G_{i+1}, \ldots, G_{i+p}$ and set $k = 1$;
3. run `OPT_VC(G,k)` in G'_i, $i = 1, \ldots, q$ in order to compute a minimum vertex cover C'_i of G'_i; if no vertex cover is found, repeat step 3 with $k = k + 1$; otherwise, let i^* be such that a vertex cover C'_{i^*} has been computed in G'_{i^*};
4. output $C = C'_{i^*} \cup (V \setminus V(G'_{i^*}))$.

As seen in the proof of Proposition 1, $\alpha(G'_{i^*}) \geqslant (p/q)\alpha(G)$. Then, the solution returned is such that $|C| = n - \alpha(G'_{i^*}) \leqslant n - (p/q)\alpha(G) = n(1 - (p/q)) + (p/q)\tau(G)$. Since $\tau(G) \geqslant n/2$, we get: $|C|/\tau(G) \leqslant 2(1 - (p/q)) + (p/q) = 2 - (p/q)$.

It is easy to see that the running time of Algorithm `VC3` is of $O^*(\delta^{r\tau(G)})$, that completes the proof of the theorem. ∎

In Table 1, we give running times for Algorithms `VC1`, `VC2` and `VC3`, for some values of ratios achieved and with $\gamma = 1.18$ and $\delta = 1.28$ (the best known values for MAX INDEPENDENT SET and MIN VERTEX COVER (see [19, 7]), respectively. Furthermore, for Algorithm `VC2`, we also give the value of λ for which the corresponding time is get.

6 Randomized algorithms

We give in this section randomized algorithms for MAX INDEPENDENT SET and MIN VERTEX COVER that, with probability $1 - \exp\{-cn\}$ for some constant c, turn to efficient approximation algorithms with running-time lower (though exponential) than the one of the deterministic algorithms seen in Sections 4 and 5.

6.1 MAX INDEPENDENT SET

In the deterministic algorithm for MAX INDEPENDENT SET seen previously, we split the instance into subinstances of size rn to get a r-approximation algorithm. Here, we show that by splitting into subinstances of smaller size βn, with $\beta < r$, we can achieve the same ratio by iterating the splitting a very large (exponential) number of times (this is Algorithm `RIS1`). The tradeoff between the size of the subinstances and the number of times we iterate splitting to get the

Ratio	VC1	VC2		VC3
		Time	λ	
1.9	1.017^n	1.013^n	0.493	1.025^k
1.8	1.034^n	1.026^n	0.486	1.051^k
1.7	1.051^n	1.039^n	0.477	1.077^k
1.6	1.068^n	1.054^n	0.468	1.104^k
1.5	1.086^n	1.069^n	0.457	1.131^k
1.4	1.104^n	1.086^n	0.443	1.160^k
1.3	1.123^n	1.104^n	0.427	1.189^k
1.2	1.142^n	1.124^n	0.407	1.218^k
1.1	1.161^n	1.148^n	0.378	1.249^k

Table 1: Running times of Algorithms VC1, VC2 and VC3 with $\gamma = 1.18$ and $\delta = 1.28$, for some values of ρ.

ratio is given in Theorem 6. Next, we determine the optimal choice for β (for Algorithm RIS1). Finally, we further improve Algorithm RIS1 by combining it with the fixed-parameter Algorithm OPT_VC for MIN VERTEX COVER in order to devise Algorithms RIS2 and RIS3.

Theorem 6. *For any $\rho < 1$ and for any β, $\rho/2 \leqslant \beta \leqslant \rho$, it is possible to find an independent set that is, with probability $1 - \exp\{-cn\}$ (for some constant c), a ρ-approximation for* MAX INDEPENDENT SET*, with running time $O^*(K_n \gamma^{\beta n})$, where:*

$$K_n = \frac{n\binom{n}{n/2}}{\binom{\beta n}{\rho n/2}\binom{n-\beta n}{((1-\rho)n/2)}}$$

Proof. By Corollary 2, we can assume $\alpha(G)/n \leqslant 1/2$. Fix a maximum independent set S^* of G and consider a subgraph B of G (for simplicity, denote also by B the vertex-set of B), whose size is $\beta n \geqslant \rho n/2 \geqslant \rho\alpha(G)$. Probability that B contains $\rho\alpha(G)$ vertices from S^* is given by the following formula:

$$p_{\beta,\alpha} = \Pr\left[|S^* \cap B| = \rho\alpha(G)\right] = \frac{\binom{\beta n}{\rho\alpha(G)}\binom{n-\beta n}{(1-\rho)\alpha(G)}}{\binom{n}{\alpha(G)}} \tag{4}$$

If we take at random K_n such different subgraphs B_i, the probability that $|S^* \cap B|$ is never greater than $\rho\alpha(G)$ is bounded above by:

$$\begin{aligned}
\Pr\left[|S^* \cap B_i| < \rho\alpha(G), \forall i \leqslant K_n\right] &= \left(1 - \Pr\left[|S^* \cap B_i| \geqslant \rho\alpha(G)\right]\right)^{K_n} \\
&\leqslant \left(1 - \Pr\left[|S^* \cap B_i| = \rho\alpha(G)\right]\right)^{K_n} \\
&\leqslant \exp\left\{K_n \log\left(1 - p_{\beta,\alpha}\right)\right\} \leqslant \exp\left\{-K_n p_{\beta,\alpha}\right\} \\
&\leqslant \exp\left\{-\frac{n p_{\beta,\alpha}}{p_{\beta,n/2}}\right\}
\end{aligned}$$

We now study $p_{\beta,\alpha}$ to show that the the previous probability is bounded by $\exp\{-cn\}$. Fix $\lambda = \alpha(G)/n$. From Stirling's formula we get:

$$\begin{aligned}
p_{\beta,\alpha} &= \frac{(\beta n)!(n-\beta n)!(\lambda n)!(n-\lambda n)!}{(\beta n - \rho\lambda n)!(\rho\lambda n)!(\lambda n - \rho\lambda n)!(n - \beta n - \lambda n + \rho\lambda n)!n!} \\
&= \theta\left(q_{\beta,\alpha}/\sqrt{n}\right)
\end{aligned} \tag{5}$$

where:

$$q_{\beta,\alpha} = \left(\frac{\beta^{\beta}(1-\beta)^{1-\beta}\lambda^{\lambda}(1-\lambda)^{1-\lambda}}{(\beta-\rho\lambda)^{\beta-\rho\lambda}(\rho\lambda)^{\rho\lambda}((1-\rho)\lambda)^{(1-\rho)\lambda}(1-\beta-(1-\rho)\lambda)^{1-\beta-(1-\rho)\lambda}} \right)^{n} \tag{6}$$

Consider now function:

$$f(\rho,\lambda,\beta) = \frac{-\log\left(q_{\beta,\lambda n}\right)}{n} \tag{7}$$

Function f is continuously differentiable on $\{(\rho,\lambda,\beta) \in]0,1[\times]0,1/2[^2; \lambda\rho \leqslant \beta \leqslant \rho\}$ and its derivatives are:

$$\begin{aligned} \frac{\partial f}{\partial \lambda} &= \rho\log\rho + (1-\rho)\log(1-\rho) - \rho\log(\beta-\rho\lambda) \\ &\quad - (1-\rho)\log(1-\beta-(1-\rho)\lambda) + \log(1-\lambda) \qquad (8) \\ \frac{\partial^2 f}{\partial\lambda\partial\beta} &= \frac{\beta-\rho}{(\beta-\rho\lambda)(1-\beta-(1-\rho)\lambda)} \leqslant 0 \qquad (9) \end{aligned}$$

where in (9) inequality holds because $\beta-\rho \leqslant 0$, $\beta-\rho\lambda \geqslant 0$ and $1-\beta-(1-\rho)\lambda \geqslant 1-\rho-(1-\rho)\lambda \geqslant 0$.

From (8) and (9) we get:

$$\frac{\partial f}{\partial \lambda} \geqslant \frac{\partial f}{\partial \lambda}(\beta=\rho) = 0 \tag{10}$$

From (10), f grows with λ and $q_{\beta,\lambda n}$ decreases with λ. Thus, the minimum for $q_{\beta,\alpha}$ is reached for $\lambda = 1/2$, and $p_{\beta,\alpha}/p_{\beta,n/2} > c$, for some constant c. This fact derives that $\Pr[\max_{i\leqslant K_n} |S^* \cap B_i| \geqslant rm^*] \geqslant 1-\exp\{-cn\}$.

Consider now the following straightforward algorithm (denoted by `RIS1` and called with parameters G and ρ), that is the randomized counterpart of Algorithm `IS` of Section 4 and where by `OPT_IS` we denote an exact MAX INDEPENDENT SET-algorithm:

1. solve the LP-relaxation of MAX INDEPENDENT SET, store sets V_0 and V_1 and set $G = G[V_{1/2}]$;
2. for $i = 1$ to K_n, compute $S_{i+1} = \max\{S_i, V_1 \cup \texttt{OPT\_IS}(\texttt{B}_\texttt{i})\}$;
3. output S_{K_n}.

It is easy to see that the running time of Algorithm `RIS1` is $O^*(K_n\gamma^{\beta n})$, while the probability that it returns a ρ-approximation of the optimal is $1-\exp\{-cn\}$. ∎

Let us now try to determine the optimal value for β. Note that if we choose $\beta = \rho$, then $K_n(\beta=\rho)$ is polynomial. In this case the running time of Algorithm `RIS1` is as the running time of Algorithm `IS` (Section 4) multiplied by a polynomial term. We now show that, according to a good choice of β, it is possible to decrease overall running time of Algorithm `RIS1` by an exponential term.

Fix $g(\rho,\beta) = f(\rho,1/2,\beta) + \beta\log(\gamma)$. According to the definition of f (expression (7)) and to (6), running time becomes $O^*(K_n\gamma^{\beta n}) = O^*(\exp\{g(\rho,\beta)\})$. Function g is derivable for any (ρ,β) such that $\rho/2 < \beta < \rho$:

$$\begin{aligned} \frac{\partial g}{\partial \beta} &= \log\left(\beta-\frac{\rho}{2}\right) - \log\left(\frac{1}{2}-\beta+\frac{\rho}{2}\right) - \log\beta + \log(1-\beta) + \log\gamma \\ \frac{\partial g}{\partial \beta} \geqslant &\Leftrightarrow \left(\beta-\frac{\rho}{2}\right)(1-\beta)\gamma - \beta\left(\frac{1}{2}-\beta+\frac{\rho}{2}\right) \geqslant 0 \qquad (11) \end{aligned}$$

Expression (11) is a second degree inequality $P(\beta) \geqslant 0$, so it has at most two zeros. Since $P(\beta=\rho/2) = -\rho/4 < 0$ and $P(\beta=\rho) = (\rho(\rho-1)/2)(\gamma-1) > 0$, it has exactly two solutions, β_+ greater than ρ and $\beta_- \in [\rho/2,\rho]$. Thus, β_- is a global minimum in $]\rho/2,\rho[$.

Ratio	IS	RIS1	
		Time	β_-
0.1	1.017^n	1.016^n	0.088
0.2	1.034^n	1.032^n	0.177
0.3	1.051^n	1.048^n	0.269
0.4	1.068^n	1.065^n	0.363
0.5	1.086^n	1.083^n	0.459
0.6	1.104^n	1.101^n	0.559
0.7	1.123^n	1.119^n	0.662
0.8	1.142^n	1.139^n	0.769
0.9	1.161^n	1.159^n	0.882

Table 2: Running times of Algorithms IS and RIS1 with $\gamma = 1.18$.

In Table 2, we give running times for Algorithms IS, and RIS1, for some values of ratios achieved. For Algorithm RIS1, we also give the value of β_- for which the corresponding time is got.

We now improve Algorithm RIS1 in two different ways, leading to Algorithms RIS2 and RIS3, respectively.

The way the first improvement is obtained (leading to Algorithm RIS2 is somehow analogous to that of Theorem 4. The basic idea is to show that, informally, the smaller the independent set, the higher the probability of finding a good approximation by splitting. In other words, when $\alpha(G)$ is small, we need a smaller running time to get the same approximation ratio with high probability, i.e., Algorithm RIS1 is more efficient. But, on the other hand, when $\alpha(G)$ is large, the fixed-parameter Algorithm OPT_VC runs fast. Then, Algorithm RIS2 combines these two algorithms. Let us note that this result cannot be used to improve the deterministic algorithm given in Proposition 1.

Proposition 2. *For any $\rho < 1$ and any β such that $\rho/2 \leqslant \beta \leqslant \rho$, it is possible to compute an independent set that realizes with probability $1-\exp\{-cn\}$ (for some constant c) a ρ-approximation of the optimum with running time $O^*(\delta^{(1-\lambda)\rho n})$, where λ is defined by:*

$$\gamma^\beta \exp\{f(\rho,\lambda,\beta)\} = \delta^{(1-\lambda)\rho} \tag{12}$$

and f is as in (7).

Proof. As previously, denote by S^* a maximum independent set in G. Assume first that $|S^*| \geqslant \lambda n$. Then, revisiting Lemma 3, Algorithm VC2 is able to find out with with running time $O^*(\delta^{(1-\lambda)\rho n})$ in some subgraph G'_0 of G with size ρn where a a vertex cover C_0 of G'_0, is associated with an independent set S_0 verifying $|S_0| \geqslant \rho|S^*|$.

Consider now the case $|S^*| \leqslant \lambda n$ and set $L_n = n/p_{\beta,\alpha}$. Quantity K_n defined in the proof of Theorem 2 is indeed the restriction of L_n to the case $\lambda = 1/2$. According to (10), $q_{\beta,\lambda n}$ (defined by (6)) decreases with λ; hence, $p_{\beta,\alpha}/p_{\beta,\lambda n} \geqslant c$ (for some constant c) when $\alpha \leqslant \lambda n$. Then:

$$\Pr\left[|S^* \cap B_i| < \rho\alpha(G); \forall i \leqslant L_n\right] \leqslant (1-p_{\beta,\alpha})^{L_n} \leqslant \exp\left\{-n\frac{p_{\beta,\alpha}}{p_{\beta,\lambda n}}\right\} \leqslant \exp\{-cn\}$$

The discussion above forwardly derives the following algorithm, denoted by RIS2:

1. fix ρ and β and determine λ satisfying (12);

2. solve the LP-relaxation of MAX INDEPENDENT SET, store V_0 and V_1 and set $G = G[V_{1/2}]$;

3. store $S_1 = \texttt{RIS1}(\texttt{G}, \rho)$;

4. compute $S_2 = V_1 \cup (V_{1/2} \setminus \texttt{PROCEDURE\_VC}(\texttt{G}, \rho))$;

5. output the best among S_1 and S_2.

Obviously, if $|S^*| \geqslant \lambda n$, S_2 realizes a ρ-approximation of S^* (Lemma 3), otherwise, by Theorem 6, S_1 realizes a ρ-approximation of S^* with probability $1 - \exp\{-cn\}$. ∎

For the case of Proposition 2, β_- has to verify $[\partial(f + \beta\log(\gamma)))/\partial\beta](\beta_-) = 0$, i.e.:

$$(\gamma - 1)\beta_-^2 + (1 - (1 - \rho)\lambda - \gamma(1 + \rho\lambda))\,\beta_- + \gamma\rho\lambda = 0$$

Some easy algebra as in (11) concludes that there always exists one and only one feasible (with respect to the interval $]\rho\lambda, \rho[$) value β_- to this equation, and this value is actually a minimum.

Ratio	IS	RIS2		
		Time	β_-	λ
0.1	1.017^n	1.015^n	0.082	0.394
0.2	1.034^n	1.031^n	0.166	0.390
0.3	1.051^n	1.047^n	0.252	0.386
0.4	1.068^n	1.063^n	0.341	0.381
0.5	1.086^n	1.080^n	0.433	0.375
0.6	1.104^n	1.098^n	0.529	0.369
0.7	1.123^n	1.117^n	0.631	0.362
0.8	1.142^n	1.136^n	0.739	0.353
0.9	1.161^n	1.157^n	0.859	0.343

Table 3: Running times of Algorithms IS and RIS2 with $\gamma = 1.18$ and $\delta = 1.28$.

In Table 3, we perform a comparative study of running times for Algorithms IS, and RIS2, for some ratio values. As one can see from Tables 2 and 3, Algorithm RIS2 dominates Algorithm RIS1 since, in fact, the former is a refinement of the latter.

The second improvement follows a different approach, based upon an exhaustive lookup of all the candidate values for $\alpha(G)$, and using an exact algorithm for MIN VERTEX COVER rather than for MAX INDEPENDENT SET. Informally, the underlying idea for this approach (leading to Algorithm RIS3) is that randomization allows to split the input graph into "small" subgraphs, on which a fixed-parameter algorithm can be efficiently used to reach both a good overall running time and any a priori fixed approximation ratio. Then, Algorithm RIS3 consists of running Algorithm OPT_VC on subgraphs of size $\beta n < rn$ taken at random and for a sufficient number of times, where β is optimally determined as a function of $\alpha(G)$.

Theorem 7. *For any $\rho < 1$, it is possible to compute an independent set that realizes with probability $1 - \exp\{-cn\}$ (for some constant c) a ρ-approximation of the optimum, in time:*

$$O^*\left(\exp\{nf(\rho, \beta, \lambda)\}\,\delta^{(\beta - \lambda\rho)n}\right)$$

where f is as in (7) and β and λ are defined by the following system:

$$(1 - \beta)(\beta + \delta - \delta\beta)\rho^\rho(1 - \rho)^{1-\rho} = \beta^\rho(1 - \beta)^{1-\rho} \quad (13)$$

$$\frac{(\delta - 1)\beta(1 - \beta)}{\rho - \beta + (\delta - 1)\rho(1 - \beta)} = \lambda \quad (14)$$

Proof. Assume that $|S^*| = \alpha(G) = \alpha$, for some $\alpha \leqslant n/2$. Consider any set $B \subset V$ where $\rho\alpha \leqslant |B| \leqslant \rho n$. For convenience, set $b = |B|/n$. Then, according to Theorem 6, probability that B contains exactly $\rho\alpha$ vertices from S^* is $p_{b,\alpha}$, and, if we take at random $L_n = n/p(\beta, k)$ different subsets $(B_{i \leqslant L_n})$ having the same size, the probability that one of them contains at least $\rho\alpha$ vertices from S^* is greater than $1 - \exp\{-cn\}$.

At this step, algorithms we have seen previously use an exact MAX INDEPENDENT SET-algorithm running in $O^*(\gamma^{bn})$. However, if we know that $|S^*| = \alpha$, it is possible to find out a minimal vertex cover of B_i with complexity $O^*(\delta^{bn-\rho\alpha})$.

We now search for a value for $\beta(\lambda)$ that minimizes expression $\exp\{nf(\rho, \beta, \lambda)\}\delta^{(\beta-\lambda\rho)n}$. Fix $\varphi(\beta, \lambda) = f(\rho, \beta, \lambda) + (\beta - \rho\lambda)\log\delta$. Then:

$$\begin{array}{rcl} \dfrac{\partial\varphi}{\partial\beta} & = & \dfrac{\partial f}{\partial\beta} + \log(\delta) \\ \dfrac{\partial\varphi}{\partial\beta} = 0 & \Leftrightarrow & \delta = \dfrac{(1-\beta-\lambda+\rho\lambda)\beta}{(1-\beta)(\beta-\rho\lambda)} \\ & \Leftrightarrow & \dfrac{\beta(1-\beta)(\delta-1)}{\rho-\beta+\rho(1-\beta)(\delta-1)} = \lambda \end{array} \qquad (15)$$

Equation in (15) is a second-degree equation on β that admits one and only one, the smaller, feasible solution in $[\rho\lambda, \rho]$. Checking signs of this interval bounds ensures that $\beta(\lambda)$ is indeed a minimum.

Let us estimate the value of α that maximizes complexity. Since f increases with λ and $\varphi - f$ decreases with λ, we cannot state, as previously, that the worst case is when $\lambda = 1/2$ (and this is not, actually, the case). Since $\beta(\lambda)$ is a local minimum for $\beta \mapsto \varphi(\beta, \lambda)$, then:

$$\begin{array}{rcl} \dfrac{d\varphi(\beta(\lambda),\lambda)}{d\lambda}(\lambda) & = & \dfrac{\partial\varphi(\beta,\lambda)}{\partial\lambda}(\beta(\lambda),\lambda) \\ & = & \log\left(\dfrac{\rho^\rho(1-\rho)^{1-\rho}(1-\beta(\lambda))(\beta(\lambda)+\delta-\beta(\lambda)\delta)}{\beta^\rho(\lambda)(1-\beta(\lambda))^{1-\rho}}\right) \end{array}$$

Thus, the worst case corresponds to a solution of (13).

Consider now the following algorithm denoted by `RIS3`:

- solve the LP-relaxation of MAX INDEPENDENT SET, store V_0 and V_1 and set $G = G[V_{1/2}]$;
- for $\alpha = 1$ to n do:
  1. determine $\beta(\alpha/n)$ satisfying (14);
  2. set $L_n = n/p(\beta, \alpha/n)\}$ and $S_0 = \emptyset$;
  3. for $i = 1$ to L_n, set $S_i = \max\{S_{i-1}, B_i \setminus \texttt{OPT\_VC}(\texttt{B}_\texttt{i}, \alpha)\}$;
  4. set $S^\alpha = S_{L_n}$;
- output $V_1 \cup \text{argmax}\{|S^\alpha|\}$.

It can be immediately seen that Algorithm `RIS3` meets the statement of the theorem and concludes its proof. ∎

Table 4 presents a comparative study of running times for Algorithms `IS`, and `RIS3`. For approximation ratios smaller than 0.75, the latter algorithm dominates all the previous ones. But, as far as ratios greater than 0.75 are dealt, it is dominated by Algorithm `RIS2` and even by `IS`.

Ratio	IS	RIS3		
		Time	$\beta(\lambda)$	λ
0.1	1.017^n	1.013^n	0.059	0.226
0.2	1.034^n	1.027^n	0.121	0.224
0.3	1.051^n	1.042^n	0.183	0.221
0.4	1.068^n	1.057^n	0.250	0.218
0.5	1.086^n	1.075^n	0.319	0.215
0.6	1.104^n	1.093^n	0.393	0.211
0.7	1.123^n	1.115^n	0.472	0.206
0.8	1.142^n	1.139^n	0.561	0.199
0.9	1.161^n	1.169^n	0.664	0.190

Table 4: Running times of Algorithms IS and RIS3 with $\delta = 1.28$.

6.2 MIN VERTEX COVER

Obviously, Lemma 1 still holds for randomized algorithms. Hence, the complements of the solutions provided by Algorithms RIS1, RIS2 and RIS3 are vertex covers for G achieving ratios $2-\rho$ with probability $1-\exp\{-cn\}$. In what follows we propose randomized efficient approximation algorithms for MIN VERTEX COVER with running times better than those get in Section 6.1. Underlying ideas are similar to the previous ones but, taking into account once again Lemma 1, a more involved computation leads to better results.

In Proposition 3 below, we simply mix the randomization technique of Algorithm RIS1 and the fixed-parameter approximate Algorithm VC3.

Proposition 3. *For any $r < 1$ and any β such that $r\lambda < \beta < r$, it is possible to compute with probability $1-\exp\{-cn\}$ (for some constant c) a $(2-r)$-approximation of* MIN VERTEX COVER *in time $O^*(\delta^{(1-\lambda)rn})$, where λ is solution of:*

$$\delta^{(1-\lambda)r} = \gamma^{\beta} \exp\left\{f\left(r'_{\lambda}, \lambda, \beta\right)\right\}$$

and r'_{λ} is a function of λ defined by:

$$r'_{\lambda} = 2 - r - \frac{1-r}{\lambda}$$

Proof. Consider the following algorithm, denoted by RVC1:

- set $C_1 = \texttt{VC3}(\texttt{G}, 2-\texttt{r})$;
- set $C_2 = V \setminus \texttt{RIS1}(\texttt{G}, \texttt{r})$;
- output $C = \operatorname{argmin}\{|C_1|, |C_2|\}$.

Assume first that $|S^*| \geqslant \lambda n$. Then, $C^* \leqslant (1-\lambda)n$. According to Theorem 5, C_1 is a $2-r$ approximation of C^* computed in $O^*(\delta^{(1-\lambda)rn})$.

Assume next that $|S^*| \leqslant \lambda n$. Then, according to Theorem 6, Algorithm RIS1 can compute in time $O^*(\gamma^{\beta}\exp\{f(r'_{\lambda}, \lambda, \beta)\})$ and with probability $1-\exp\{-cn\}$, an r-approximation for MAX INDEPENDENT SET which, according to Lemma 1, turns to a $2-r$-approximation for MIN VERTEX COVER. ∎

We now conclude this section by further improving the result of Proposition 3. The idea is similar to the one for Algorithm RIS3: we use the fixed-parameter algorithm OPT_VC on subinstances of size $\beta n < rn$, where β is optimally determined as a function of $\alpha(G)$.

Proposition 4. *For any $r < 1$, and any β such that $r\lambda < \beta < r$, it is possible to compute with probability $1 - \exp\{-cn\}$ (for some constant c) a $(2-r)$-approximation of* MIN VERTEX COVER *in $O^*(\delta^{(1-\lambda)r'_\lambda n})$, where β and λ are defined by the following system:*

$$\begin{aligned} \frac{\beta(1-\beta)(\delta-1)}{r'_\lambda - \beta + r'_\lambda(1-\beta)(\delta-1)} &= \lambda \\ \frac{d\varphi\left(\beta(\lambda), \lambda, r'_\lambda\right)}{d\lambda} &= 0 \end{aligned} \tag{16}$$

and r'_λ is a function of λ defined by:

$$r'_\lambda = 2 - r - \frac{1-r}{\lambda}$$

Proof. Consider the following algorithm, denoted by `RVC2`, that is devised in the same spirit as Algorithm `RIS3`:

1. solve the LP-relaxation of MAX INDEPENDENT SET, store V_0 and V_1 and set $G = G[V_{1/2}]$;
2. for any $k \leqslant n$ do:
   (a) set $\lambda = k/n$ and determine $\beta(\lambda)$ satisfying (16);
   (b) set $L_n = n \exp\{f(r'_\lambda, k/n, \beta)n\}$ and $S_0 = \emptyset$;
   (c) for $i = 1$ to L_n, compute $S_i = \max\{S_{i-1}, B_i \setminus \texttt{OPT\_VC}(\texttt{B}_\texttt{i}, \texttt{k})\}$;
   (d) set $S^k = S_{L_n}$
3. output $V_0 \cup (V_{1/2} \setminus \text{argmax}\{|S^k|\}$.

According to the analysis of Algorithm `RIS3`, $S^{|S^*|}$ is an independent set whose size is at least $r'_{|S^*|/n}|S^*|$. Then, according to Lemma 1, $V_{1/2} \setminus S^{|S^*|}$ is a vertex cover whose size is at most $(2-r)|C^*|$.

The complexity of Algorithm `RVC2` is determined by the worst value for λ. Computing such a value amounts to study the function $\varphi(\beta(\lambda), \lambda, r'_\lambda)$. But, unfortunately, this does not lead to a simple expression for λ. For this reason analytic computation of λ is omitted. ∎

Ratio	`VC1`	`VC2`	`RVC1`	`RVC2`
1.9	1.017^n	1.013^n	1.013^n	1.010^n
1.8	1.034^n	1.026^n	1.026^n	1.021^n
1.7	1.051^n	1.039^n	1.039^n	1.032^n
1.6	1.068^n	1.054^n	1.053^n	1.043^n
1.5	1.086^n	1.069^n	1.068^n	1.056^n
1.4	1.104^n	1.086^n	1.085^n	1.069^n
1.3	1.123^n	1.104^n	1.102^n	1.083^n
1.2	1.142^n	1.124^n	1.122^n	1.099^n
1.1	1.161^n	1.148^n	1.146^n	1.127^n

Table 5: Running times of Algorithms `VC1`, `VC2`, `RVC1` and `RVC2` with $\gamma = 1.18$ and $\delta = 1.28$.

In Table 5, the running times of Algorithms `VC1`, `VC2`, `RVC1` and `RVC2` are shown for some ratios and for $\gamma = 1.18$ and $\delta = 1.28$.

7 Efficient approximation of MAX INDEPENDENT SET in particular classes of graphs

In this section we consider particular classes of MAX INDEPENDENT SET-instances admitting polynomial approximation algorithms achieving some ratio ρ. For instance, a notable example of such a class is the class of bounded-degree graphs. For these graphs, denoting by $\Delta(G)$ the bound on the degrees, MAX INDEPENDENT SET can be polynomially approximated within ratio $\rho = 5/(\Delta(G)+3)$ ([5]).

Consider some class $\mathcal{C}$ of graphs where MAX INDEPENDENT SET is approximable in polynomial time within approximation ratio ρ by an algorithm called `APIS` in what follows. We show that the graph splitting technique used previously can be efficiently applied to get interesting tradeoffs between running times and approximation ratios (greater than ρ).

Proposition 5. *For any rational $r < 1$, it is possible to compute, for any graph $G \in \mathcal{C}$ a $(r + (1-r)\rho)$-approximation of* MAX INDEPENDENT SET*, with running time $O^*(2^{rn})$.*

Proof. Let $G(V,E)$ be a graph in $\mathcal{C}$ and S^* be a maximum independent set of G. Fix $r = p/q$. Run the following algorithm, denoted by `EIS1`, where $\Gamma(H)$ denotes the set of neighbors of H in $V \setminus H$:

1. arbitrarily partition G into q induced subgraphs $G_1, \ldots, G_q$ of order (except eventually for G_q) n/q;
2. build subgraphs $G'_1, \ldots G'_q$ that are the unions of p consecutive subgraphs $G_{i+1}, \ldots, G_{i+p}$; let V'_i be the vertex set of G'_i, $i = 1, \ldots, q$;
3. for any V'_i and any $H \subseteq V'_i$, if H is independent, then $S = H \cup \texttt{APIS}(\texttt{G}[\texttt{V} \setminus (\texttt{H} \cup \Gamma(\texttt{H}))])$;
4. output the best among S's computed at step 3.

According to Proposition 1, one of the graphs G'_i $i = 1, \ldots q$, built at step 2 contains a set $S_0 \subseteq S^*$ with at least $r|S^*|$ vertices. Since $\Gamma(S_0) \cap S^* = \emptyset$, the set $S^* \setminus S_0$ is contained in $V \setminus (S_0 \cup \Gamma(S_0))$. Algorithm `APIS` is also called by Algorithm `EIS1` on the subgraph induced by $V \setminus (S_0 \cup \Gamma(S_0))$ and, in this case, the independent set computed has size is at least $\rho(|S^*| - |S_0|) + |S_0|$ and the same holds for the largest of the so-computed sets S returned by step 4. Hence, the approximation ratio finally achieved is at least:

$$\frac{|S_0| + \rho\,(|S^*| - |S_0|)}{|S^*|} \geqslant r + (1-r)\rho$$

Obviously, Algorithm `EIS1` runs $q2^{pn/q}$ times a polynomial algorithm. Hence, its complexity is $O^*(2^{rn})$. ∎

Let us note that the algorithm is only interesting if its ratio is better than that of `IS` that has the same running time. For this the following must hold:

$$r + (1-r)\rho \geqslant \frac{\log 2}{\log \gamma} r \iff r \leqslant \frac{\rho \log \gamma}{\log 2 - (1-\rho)\log \gamma}$$

For instance, for graphs whose degree is bounded above by 3, this means that:

$$r \leqslant \frac{5 \log \gamma}{6 \log 2 - \log \gamma} \Leftrightarrow r + (1-r)\rho \leqslant \frac{5 \log 2}{6 \log 2 - \log \gamma} \approx 0.870$$

Now, we show how to improve this result. It is easy to see that in the analysis of Proposition 5 if, roughly speaking, S^* is not "uniformly" distributed over the G'_i's, then the ratio improves. In the following proposition, we deal with the problematic case of a uniform distribution and show that, informally, generating only "small" subsets of G'_i's is sufficient.

Proposition 6. *For any $r < 1$, it is possible to compute on any graph $G \in \mathcal{C}$ a $(r + (1-r)\rho)$-approximation of* MAX INDEPENDENT SET*, with running time $O^*(2^{\beta rn})$, where $\beta < 1$ is a solution of:*

$$2^{\beta} = \frac{\beta}{\lambda^{\frac{\lambda}{\beta}}(\beta - \lambda)^{1-\frac{\lambda}{\beta}}} \tag{17}$$

and λ is such that it verifies:

$$2^{\beta r} = \delta^{(1-\lambda)(r+(1-r)\rho)} \tag{18}$$

if some solution of this equation is less than $1/2$, $\lambda = 1/2$ otherwise.

Proof. Fix a rational $r < 1$ as well as $\beta < 1$, and $\lambda < 1/2$. Assume first that $|S^*| \geqslant \lambda n$. Then, according to Theorem 4 (see, in particular Fact 1), Algorithm $\texttt{VC2}(\texttt{G}, 2 - \texttt{r} - (1 - \texttt{r})\rho)$ returns in time $O^*(\delta^{(1-\lambda)(r+(1-r)\rho)})$ a $(r + (1-r)\rho)$-approximation for MAX INDEPENDENT SET.

Consider now the case $|S^*| \leqslant \lambda n$. Set $p/q = r$ and arbitrarily partition G into q induced subgraphs $G_1, \ldots, G_q$ of order (except eventually for G_q) n/q. Build the q subgraphs $G'_1, \ldots G'_q$ that are unions of p consecutive subgraphs $G_{i+1}, \ldots, G_{i+p}$, $i = 1, \ldots, q$. Denote by V'_i the vertex set of G'_i, S^* a maximum independent set of G and assume that there exists some $i_0 \in \{1, \ldots, q\}$, such that:

$$\widehat{\alpha} = \left|S^* \cap V'_{i_0}\right| \geqslant \frac{r\,|S^*|}{\beta}$$

Then, fix $p'/q' = \beta$, arbitrarily partition G'_{i_0} into q' induced subgraphs of the same order and build the q' subgraphs $G''_1, \ldots G''_{q'}$ that are unions of p' consecutive subgraphs $G''_{i+1}, \ldots, G''_{i+p'}$, $i = 1, \ldots, q'$. According to Proposition 1, there exists one of these graphs that contains at least $\beta\widehat{\alpha} = r|S^*|$ vertices from $S^* \cap V'_{i_0}$. Computing any subset of any combination of a specific partition of any V'_{i_0} has cost:

$$q' \times 2^{\frac{\beta pn}{q}} \times q = O^*\left(2^{\beta rn}\right)$$

It remains now to handle the case where for any $i \in \{1, \ldots, q\}$:

$$\widehat{\alpha} \leqslant \frac{r\,|S^*|}{\beta} \leqslant \frac{r\lambda n}{\beta}$$

Then, Algorithm `EIS1` achieves the claimed ratio up to the fact that it does only consider subsets smaller than $r\lambda n/\beta$. Thus, in this case its running time is only:

$$\binom{rn}{\lambda rn/\beta} = O^*\left(\frac{\beta}{\lambda^{\frac{\lambda}{\beta}}(\beta - \lambda)^{1-\frac{\lambda}{\beta}}}\right)^{rn}$$

The discussion above directly leads to the following algorithm, denoted by `EIS2`:

1. solve the LP-relaxation of MAX INDEPENDENT SET, store V_0 and V_1 and set $G = G[V_{1/2}]$;
2. fix $r = p/q$ and compute λ and β according to (17) and (18), respectively; fix $\beta = p'/q'$;
3. if $\lambda < 1/2$, then set $S' = V_{1/2} \setminus \texttt{PROCEDURE\_VC}(\texttt{G}, 2 - \texttt{r} - \rho(1 - \texttt{r}))$;
4. if $\lambda \geqslant 1/2$, replace λ by $1/2$ and β by 1; set $S' = \emptyset$;
5. arbitrarily partition G into q induced subgraphs $G_1, \ldots, G_q$ of order (except eventually for G_q) n/q; build the q subgraphs $G'_1, \ldots G'_q$ that are unions of p consecutive subgraphs $G_{i+1}, \ldots, G_{i+p}$, $i = 1, \ldots, q$; let V'_i be the vertex set of G'_i, $i = 1, \ldots, q$;

6. for any V_i' and any $H \subseteq V_i'$ such that $|H| \leqslant \lambda rn/\beta$, if H is an independent set, then set $S' = \max\{H \cup \texttt{APIS}(\texttt{V} \setminus (\texttt{H} \cup \Gamma(\texttt{H}))), S'\}$;

7. for any V_i':

   (a) arbitrarily partition V_i' into q' induced subgraphs of the same order and build the q' subgraphs $G_1'', \ldots G_{q'}''$ that are unions of p' consecutive subgraphs $G_{i+1}'', \ldots, G_{i+p'}''$, $i = 1, \ldots, q'$;

   (b) for any G_i'' and any $H \subseteq V(G_i'')$, if H is an independent set, then set $S' = \max\{H \cup \texttt{APIS}(\texttt{V} \setminus (\texttt{H} \cup \Gamma(\texttt{H}))), S'\}$;

8. output $S = V_1 \cup S'$.

It is easy to see that Algorithm EIS2 perfectly fulfils the statement of the proposition. ∎

We conclude this section by a comparative study of the running times of Algorithms EIS1 and EIS2 with respect to Algorithm $\texttt{IS}(\texttt{G}, \texttt{r} + \rho(\texttt{1} - \texttt{r}))$, called with $\rho = 5/6$, $\gamma = 1.112$ ([13]), $\delta = 1.194$ ([8]), for $\Delta(G) = 3$, $\rho = 5/7$, $\gamma = 1.17$ ([3]), $\delta = 1.28$, for $\Delta(G) = 4$, and $\rho = 0.5$, $\gamma = 1.18$, $\delta = 1.28$, for $\Delta(G) = 7$. Results are shown in Table 6.

8 MAX SET PACKING, MAX CLIQUE and MAX BIPARTITE SUBGRAPH

We show in this section how the results obtained in previous sections can be applied also to efficiently approximate other combinatorial problems. Let us first note that all these problems are hereditary problems, hence Proposition 1 is applied to each of them with a parameter γ, the basis of the exponential, depended on the particular problem. But in what follows, we show that for the problems handled in this section, namely, MAX SET PACKING, MAX CLIQUE and MAX BIPARTITE SUBGRAPH, any of the results of the former sections identically apply with parameters γ' and δ' that depend on those of MAX INDEPENDENT SET and MIN VERTEX COVER, respectively.

8.1 MAX SET PACKING

Let us first handle the case of MAX SET PACKING[5] that is quite simple. Given an instance $I(\mathcal{S}, C)$ of MAX SET PACKING with $\mathcal{S} = \{S_1, \ldots, S_m\}$, $S_i \subseteq C$, $i = 1, \ldots, m$, and $C = \{c_1, \ldots, c_n\}$, we construct the graph $G(V, E)$ as follows:

- $V = \{v_1, \ldots, v_m\}$, in other words $|V| = |\mathcal{S}|$;
- $E = \{(v_i, v_j) : S_i \cap S_j \neq \emptyset\}$.

In other words, if two sets in $\mathcal{S}$ have non-empty intersection then the corresponding vertices in G are adjacent. It is easy to see that any set packing in I transforms into an independent set of the same size in G and vice-versa. So the following result holds directly.

Proposition 7. MAX SET PACKING *is as efficiently approximable as* MAX INDEPENDENT SET. *Parameters* γ *and* δ *of* MAX SET PACKING *are the same as* MAX INDEPENDENT SET *and* MIN VERTEX COVER*, respectively. The exponent for* MAX SET PACKING *is the cardinality* m *of the set-family* $\mathcal{S}$.

[5] Given a ground set C and a family $\mathcal{S}$ of n subsets of C (i.e., $\mathcal{S} \subseteq 2^C$), MAX SET PACKING consists of determining a maximum-size collection $\mathcal{S}' \subseteq \mathcal{S}$ such that sets in $\mathcal{S}'$ are pairwise disjoint.

	r	$r+\rho(1-r)$	IS	EIS1	EIS2		
					Time	β	λ
$\rho = 5/6$	0	5/6	1.092^n	polynomial	polynomial	1	0.5
	0.04	0.84	1.093^n	1.028^n	1.028^n	1	0.5
	0.1	0.85	1.094^n	1.072^n	1.072^n	1	0.5
	0.16	0.86	1.096^n	1.117^n	1.109^n	0.931	0.323
$\rho = 5/7$	0	5/7	1.117^n	polynomial	polynomial	1	0.5
	0.02	0.72	1.12^n	1.014^n	1.014^n	1	0.5
	0.055	0.73	1.121^n	1.039^n	1.039^n	1	0.5
	0.09	0.74	1.123^n	1.064^n	1.064^n	1	0.5
	0.125	0.75	1.125^n	1.091^n	1.091^n	1	0.5
	0.16	0.76	1.127^n	1.117^n	1.115^n	0.984	0.418
	0.195	0.77	1.129^n	1.145^n	1.135^n	0.938	0.333
$\rho = 1/2$	0	0.5	1.086^n	polynomial	polynomial	1	0.5
	0.02	0.51	1.088^n	1.014^n	1.014^n	1	0.5
	0.04	0.52	1.090^n	1.028^n	1.028^n	1	0.5
	0.06	0.53	1.092^n	1.042^n	1.042^n	1	0.5
	0.08	0.54	1.093^n	1.057^n	1.057^n	1	0.5
	0.1	0.55	1.095^n	1.072^n	1.072^n	1	0.489
	0.12	0.56	1.097^n	1.087^n	1.085^n	0.981	0.410
	0.14	0.57	1.099^n	1.102^n	1.096^n	0.947	0.347
	0.16	0.58	1.101^n	1.117^n	1.106^n	0.91	0.295

Table 6: Running times of Algorithms IS, EIS1, and EIS2 with: $\rho = 5/6$, $\gamma = 1.112$, $\delta = 1.28$ for $\Delta(G) = 3$, $\rho = 5/7$, $\gamma = 1.17$, $\delta = 1.28$ for $\Delta(G) = 4$ and $\rho = 0.5$, $\gamma = 1.18$, $\delta = 1.28$ for $\Delta(G) = 7$.

8.2 MAX CLIQUE

We handle in this section another famous combinatorial optimization problem that is MAX CLIQUE[6]. It is very well known that an independent set in a graph G becomes a clique of the same size in the complement $\bar{G}$ of G where we keep vertices we delete E and we add an edge (v_i, v_j) if and only if $i \neq j$ and $(v_i, v_j) \notin E$. So, results of previous sections for independent set trivially apply to MAX CLIQUE. In what follows, we improve these results replacing exponent n, the order of the input graph G for MAX CLIQUE, by $\Delta(G)$, the maximum degree of G.

Consider the following reduction from MAX CLIQUE to MAX INDEPENDENT SET. Let $G(V, E)$ be the input graph of MAX CLIQUE, $V = \{v_1, \ldots, v_n\}$ and, for $i = 1, \ldots, n$, denote by $\Gamma(v_i)$ the neighbors of v_i. Build the n graphs $G_i = G[\{v_i\} \cup \Gamma(v_i)]$. Since in any clique of G any vertex is a

[6]Given a graph $G(V, E)$, MAX CLIQUE consists of determining a maximum-size subset $V' \subseteq V$ such that $G[V']$ is a complete graph.

neighbor of any other vertex of a clique, any of these cliques are subsets of the neighborhood of each of their vertices. So, a maximum clique is a subset of the neighborhood of some graph G_i just built. For every G_i, build its complement $\bar{G}_i$ and solve MAX INDEPENDENT SET in $\bar{G}_i$. Let S_i, $i = 1, \ldots, n$, the independent sets so computed. These sets are cliques of G_i. Then, take the largest of these sets as solutions.

Obviously, if an exact algorithm for MAX INDEPENDENT SET is used, then the largest among the sets S_i is a maximum clique in G. By taking into account that the order of any of the graphs G_i is bounded above by $\Delta(G) + 1$, we immediately deduce that computing a maximum clique in a graph G takes time $O(n\gamma^{\Delta(G)+1}) = O^*(\gamma^{\Delta(G)})$, where γ is the basis of the exponential of MAX INDEPENDENT SET.

Discussion just above derives, at very first, the following parametric complexity result for the exact computation of MAX CLIQUE, interesting per se.

Theorem 8. MAX CLIQUE *can be exactly solved in* $O^*(\gamma^{\Delta(G)})$*, where* $\Delta(G)$ *is the maximum degree of the input graph.*

Also, any of the results dealing with MAX INDEPENDENT SET seen in the previous sections, identically applies to MAX CLIQUE also. So; the following theorem holds and concludes this section.

Theorem 9. *For the efficient approximation of* MAX CLIQUE*, parameters* γ *and* δ *are the same as* MAX INDEPENDENT SET *and* MIN VERTEX COVER*, respectively. The exponent for* MAX CLIQUE *is the maximum degree* $\Delta(G)$ *of the input-graph.*

8.3 MAX BIPARTITE SUBGRAPH

Consider the following reduction from MAX BIPARTITE SUBGRAPH[7] to MAX INDEPENDENT SET ([20]). Let $G(V, E)$ be an instance of MAX BIPARTITE SUBGRAPH of order n. Construct a graph $G'(V', E')$ for MAX INDEPENDENT SET by taking two distinct copies of G (denote them by G_1 and G_2, respectively) and adding the following edges: a vertex v_{i_1} of copy G_1 is linked with a vertex v_{j_2} of G_2, if and only if $(v_i, v_j) \notin E$. In other words, we link any vertex v of one copy, to the vertices of the other one with which v is not linked in G. It is easy to see that the so-constructed graph G' is regular with vertex-degree n.

Consider an independent set S of G' and denote by S_i the subset of S that belong to G_i and by V_i the corresponding set of V, $i = 1, 2$. Then, V_1 and V_2 are two independent sets in G that are completely linked one to the other, i.e., they form a complete bipartite graph of the same size as S. Conversely, if V_1 and V_2 are independent sets that form a complete bipartite graph in G then their copies V_{1_1} and V_{2_2} form a whole independent set (of size $|V_{1_1} \cup V_{2_2}|$) in G'. So, any solution for MAX INDEPENDENT SET in G' can be transformed into a solution, of the same size, for MAX BIPARTITE SUBGRAPH in G.

Observe finally that according to the reduction just above, an instance of size n for MAX BIPARTITE SUBGRAPH transforms into an instance of size $2n$ for MAX INDEPENDENT SET. So the following result can be forwardly derived.

Proposition 8. *For the efficient approximation of* MAX BIPARTITE SUBGRAPH*, parameters* γ *and* δ *of* MAX INDEPENDENT SET *and* MIN VERTEX COVER*, are transformed into* γ^2 *and* δ^2 *respectively. The exponent for* MAX BIPARTITE SUBGRAPH *is the size* n *of the input-graph.*

In other words, considering $\gamma = 1.18$ and $\delta = 1.28$, the corresponding bases for MAX BIPARTITE SUBGRAPH become 1.39 and 1.64, respectively.

[7] Given a graph $G(V, E)$, MAX BIPARTITE SUBGRAPH consists of finding a maximum-size subset $V' \subseteq V$ such that the graph $G[V']$ is a complete bipartite subgraph.

9 Conclusion

We have proposed an approach matching polynomial approximation and exact computation, that can be seen as a further tool for handling intractability. This approach allows, as we have seen, to approximately solve hard combinatorial optimization problems within approximation ratios unachievable in polynomial time and with non-trivial running times faster than those of exact computation.

It can be easily seen that all the algorithms proposed in this paper use polynomial space. Also, as the results are parameterized by the bases of the best worst-case complexity known for the problems handled, any improvement of these bases is immediately transferred to them.

The most of the results presented is based upon appropriate splittings of the initial instance into smaller ones in such a way that solution of the latter allow recovering of a solution of the former. This method is quite general and works for other problems such as MAX SAT problems.

The issue considered is this paper deserves to our opinion further research. A main direction could be to devise other efficient methods, either for improving our results or for handling other paradigmatic problems such as MIN TSP. These methods could be, for instance, inspired by exact algorithms (for example by a sharp pruning of the search tree), or could be direct approximation algorithms based upon some non-polynomial computations rather than exact computations on subinstances.

In another order of ideas, revisit for a while Section 8.3. The result is obtained there via an approximation preserving reduction. The important parameters of this reduction are not only the ratio's expansion (this is the case in the polynomial approximation framework), but also the instance size amplification (that is crucial for efficient approximation). Defining appropriate notions of approximation preserving reductions involving, for instance, small linear instance size amplifications, may be also an interesting issue that deserves further investigations.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. *J. Assoc. Comput. Mach.*, 45(3):501–555, 1998.

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and approximation. Combinatorial optimization problems and their approximability properties.* Springer-Verlag, Berlin, 1999.

[3] R. Beigel. Finding maximum independent sets in sparse and general graphs. In *Proc. Symposium on Discrete Algorithms, SODA'99*, pages 856–857, 1999.

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

[5] P. Berman and T. Fujito. On the approximation properties of independent set problem in degree 3 graphs. In *Proc. International Workshop on Algorithms and Data Structures, WADS'95*, volume 955 of *Lecture Notes in Computer Science*, pages 449–460. Springer-Verlag, 1995.

[6] L. Cai and X. Huang. Fixed-parameter approximation: conceptual framework and approximability results. In H. L. Bodlaender and M. A. Langston, editors, *Proc. International Workshop on Parameterized and Exact Computation, IWPEC'06*, volume 4169 of *Lecture Notes in Computer Science*, pages 96–108. Springer-Verlag, 2006.

[7] J. Chen, I. A. Kanj, and W. Jia. Vertex cover: further observations and further improvements. *J. Algorithms*, 41:280–301, 2001.

[8] J. Chen, I. A. Kanj, and G. Xia. Labeled search trees and amortized analysis: improved upper bounds for np-hard problems. In T. Ibaraki, N. Katoh, and H. Ono, editors, *Proc.International Symposium on Algorithms and Computation, ISAAC'03*, volume 2906 of *Lecture Notes in Computer Science*, pages 148–157. Springer-Verlag, 2003.

[9] Y. Chen, M. Grohe, and M. Grüber. On parameterized approximability. In H. L. Bodlaender and M. A. Langston, editors, *Proc. International Workshop on Parameterized and Exact Computation, IWPEC'06*, volume 4169 of *Lecture Notes in Computer Science*, pages 109–120. Springer-Verlag, 2006.

[10] R. G. Downey, M. R. Fellows, and C. McCartin. Parameterized approximation problems. In H. L. Bodlaender and M. A. Langston, editors, *Proc. International Workshop on Parameterized and Exact Computation, IWPEC'06*, volume 4169 of *Lecture Notes in Computer Science*, pages 121–129. Springer-Verlag, 2006.

[11] D. Eppstein. Improved algorithms for 3-coloring, 3-edge-coloring, and constraint satisfaction. In *Proc. Symposium on Discrete Algorithms, SODA'01*, pages 329–337, 2001.

[12] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: domination – a case study. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Proc. ICALP'05*, volume 3580 of *Lecture Notes in Computer Science*, pages 191–203. Springer-Verlag, 2005.

[13] M. Fürer. A faster algorithm for finding maximum independent sets in sparse graphs. In J. R. Corea, A. Hevia, and M. Kiwi, editors, *Proc. Latin American Symposium on Theoretical Informatics, LATIN'06*, volume 3887 of *Lecture Notes in Computer Science*, pages 491–501. Springer-Verlag, 2006.

[14] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.

[15] D. S. Hochbaum, editor. *Approximation algorithms for NP-hard problems.* PWS, Boston, 1997.

[16] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256–278, 1974.

[17] S. Khot and O. Regev. Vertex cover might be hard to approximate to within $2-\varepsilon$. In *Proc. Annual Conference on Computational Complexity, CCC'03*, pages 379–386, 2003.

[18] G. L. Nemhauser and L. E. Trotter. Vertex packings: structural properties and algorithms. *Math. Programming*, 8:232–248, 1975.

[19] J. M. Robson. Finding a maximum independent set in time $O(2^{n/4})$. Tecnical Report 1251-01, LaBRI, Université de Bordeaux I, 2001.

[20] H. U. Simon. On approximate solutions for combinatorial optimization problems. *SIAM J. Disc. Math.*, 3(2):294–310, 1990.

[21] V. Vazirani. *Approximation algorithms.* Springer, Berlin, 2001.

[22] G. J. Wœginger. Exact algorithms for NP-hard problems: a survey. In M. Juenger, G. Reinelt, and G. Rinaldi, editors, *Combinatorial Optimization - Eureka! You shrink!*, volume 2570 of *Lecture Notes in Computer Science*, pages 185–207. Springer-Verlag, 2003.