Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 165 (2005) 555–568
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Invited Tutorial

Polynomial approximation algorithms with performance guarantees: An introduction-by-example

Marc Demange [a], Vangelis Th. Paschos [b,*]

[a] *ESSEC, France*
[b] *LAMSADE, Université Paris-Dauphine, Place Marechal de Lattre de Tassigny, Paris Cedex 1675775, France*

Received 30 December 2003; accepted 30 March 2004
Available online 19 June 2004

Abstract

We present a short overview on polynomial approximation of **NP**-hard problems. We present the main approximability classes together with examples of problems belonging to them. We also describe the general concept of approximability preserving reductions together with a discussion about their capacities and their limits. Finally, we present a quick description of what it is commonly called inapproximability results. Such results provide limits on the approximability of the problems tackled.

Keywords: Combinatorial optimization; Computing science; Complexity theory; Traveling salesman

1. Preliminaries

An **NP** optimization (NPO) problem Π is defined as a four-tuple $(\mathscr{I}, \mathrm{Sol}, m, \mathrm{opt})$ such that

- $\mathscr{I}$ is the set of instances of Π and it can be recognized in polynomial time;
- given $I \in \mathscr{I}$, $\mathrm{Sol}(I)$ denotes the set of feasible solutions of I; for every $S \in \mathrm{Sol}(I)$, $|S|$ (the size of S) is polynomial in $|I|$ (the size of I); given any I and any S polynomial in $|I|$, one can decide in polynomial time if $S \in \mathrm{Sol}(I)$;
- given $I \in \mathscr{I}$ and $S \in \mathrm{Sol}(I)$, $m(I,S)$ denotes the value of S; m is polynomially computable and is commonly called objective function;
- $\mathrm{opt} \in \{\max, \min\}$.

The class of **NP** optimization problems is commonly denoted by **NPO**. The most interesting subclass of **NPO** is the class of the **NP**-hard optimization problems, known to be unsolvable in polynomial time unless $\mathbf{P} = \mathbf{NP}$. A problem $\Pi \in \mathbf{NPO}$ is **NP**-hard if its decision version, denoted by Π_D, is **NP**-complete, i.e., if any problem $\Pi' \in \mathbf{NP}$ reduces to Π_D by a Karp reduction (see [5,16,25] for more details about this reduction).

It is widely believed that no polynomial time algorithm can ever be devised for optimally solving **NP**-hard problems. This belief motivates both

* Corresponding author. Tel.: +33-1-4405-4582; fax: +33-1-4405-4091.
E-mail addresses: demange@essec.fr (M. Demange), paschos@lamsade.dauphine.fr (V.Th. Paschos).

0377-2217/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2004.03.021

researchers and practitioners in trying to find a trade-off between computational time and solution's quality for such problems; this is what is commonly called *heuristically solving* **NP**-*hard problems*. Heuristic computation consists *of trying to find in reasonable time, not the best solution but one solution which is "near" the optimal one*.

Among classes of heuristic methods approximately solving **NP**-hard problems, the so-called *polynomial approximation algorithms* aim in solving a given **NP**-hard problem *in polynomial time* by computing *feasible solutions* being, under some predefined criterion, *as near as possible to the optimal ones*. The polynomial approximation theory deals with the study of such algorithms.

We now briefly present some basic notions of polynomial approximation theory. More details can be found in [5,18,32]. Given an instance I of a combinatorial maximization (resp., minimization) problem $\Pi = (\mathscr{I}, \mathrm{Sol}, m, \mathrm{opt})$, $\omega(I)$, $m_{\mathtt{A}}(I,S)$ and $\mathrm{opt}(I)$ will denote the values of the worst solution of I (in the sense of the objective function), the approximated one, S (provided by some polynomial time approximation algorithm $\mathtt{A}$ supposed to feasibly solve problem Π), and the optimal one, respectively. The worst solution of I is the optimal solution for I with respect to the **NPO** problem $\Pi' = (\mathscr{I}, \mathrm{Sol}, m, \mathrm{opt}')$ where $\mathrm{opt}' = \max$, if $\mathrm{opt} = \min$ and $\mathrm{opt}' = \min$, if $\mathrm{opt} = \max$. There exist mainly two frameworks dealing with polynomial approximation.

Standard approximation. Here, the quality of an approximation algorithm $\mathtt{A}$ is expressed by the ratio $\rho_{\mathtt{A}}(I) = m_{\mathtt{A}}(I,S)/\mathrm{opt}(I)$. Note that approximation ratio for minimization problems is in $[1,\infty)$, while for maximization ones this ratio is in [0,1].

Differential approximation. Here, the quality of an approximation algorithm $\mathtt{A}$ is expressed by the ratio $\delta_{\mathtt{A}}(I) = |\omega(I) - m_{\mathtt{A}}(I,S)|/|\omega(I) - \beta(I)|$. The value of the differential ratio is always in [0,1], independently of the nature (maximization or minimization) of the problem.

In this paper, we deal with the standard approximation paradigm. Let us note that results obtained adopting the one or the other of the two paradigms are very often different even for the same problem.

A particularly interesting case of polynomial time approximation algorithm (representing "ideal" approximation behavior), is the one of the *polynomial time approximation scheme*. A polynomial time approximation scheme is a sequence $\mathtt{A}_\epsilon$ of polynomial time approximation algorithms guaranteeing, for every $\epsilon > 0$, approximation ratio $\rho_{\mathtt{A}_\epsilon} = 1 - \epsilon$, if the problem studied is a maximization problem, or $1 + \epsilon$ if the problem dealt is a minimization one, with complexity $O(n^k)$, where k is a constant not depending on n but eventually depending on $1/\epsilon$. If, furthermore, the complexity of $\mathtt{A}_\epsilon$ is $O((1/\epsilon)^{k_1} n^{k_2})$, where k_1 and k_2 are constants depending neither on n nor on $1/\epsilon$, then we speak about *fully polynomial time approximation scheme*.

In what follows, we present in a first time approximation algorithms for some classical examples of **NP**-hard problems. In a second time, we show how classical notions and tools of complexity theory can be matched with polynomial approximation in order to devise structural results for **NPO**. For graph-theoretic notions, definitions and standard notations, one can refer to [7]. Dealing with some non-standard notations, for a vertex v of a graph $G(V,E)$ (defined on a vertex set V with edge set E), we denote by $\Gamma(v)$ the set of neighbors of v, i.e., the set of vertices of V linked to v with some edge; for $V' \subseteq V$, we denote by $G[V']$ the subgraph of G induced by V'; finally, we denote by $\Delta(G)$ the maximum degree of G.

2. Constant approximation ratios and the class APX

One of the most interesting **NPO** problem families is the one whose members can be solved by polynomial algorithms achieving approximation ratios that are fixed constants (i.e., independent on any instance parameter). This family is called **APX**. We describe in this section constant-ratio approximation algorithms for three paradigmatic problems, the MIN VERTEX COVER, BIN PACKING and MIN METRIC TSP.

2.1. Min vertex cover

Given a graph $G(V,E)$, a matching is a subset $E' \subseteq E$ such that the partial graph $G'(V,E')$ has maximum degree 1. A matching E' is called maximal (for the inclusion) if E' is a matching and E' plus

a new edge is no more a matching. The following algorithm based upon the computation of a maximal matching E' is the most classical for approximately solving MIN VERTEX COVER. It works as follows, supposing that E' is initialized to $\emptyset$:

1. pick an edge $e \in E$;
2. set $E' = E' \cup \{e\}$;
3. delete from E any edge having a common endpoint with e;
4. repeat steps 1–3 until $E = \emptyset$;
5. output V', the set of the endpoints of edges in E'.

Algorithm above is linear in $|E|$ and set E' computed along steps 1–4 is a maximal matching for E'. Moreover, it is easy to see that V' is a vertex covering for G since any edge in E is covered by some vertex in V'. Indeed, any edge e selected during step 1 is covered by its endpoints; these vertices cover also edges deleted because of e during step 3.

Obviously, $|V'| = m(G, V') = 2|E'|$. On the other hand, any of edges in E' is covered by a proper vertex in any feasible vertex covering of G; hence $\text{opt}(G) \leqslant |E'|$. Consequently, the approximation ratio $\rho(G) = m(G, V')/\text{opt}(G) \leqslant 2$.

Note that ratio 2 is tight for algorithm above. In fact, consider a star G on k vertices $\{v_1, v_2, \ldots, v_k\}$, where v_1 is the star-center. Then, a maximal matching in G consists of taking any edge (v_1, v_i), $i = 2, \ldots, k$. This will produce a set V' of cardinality 2, while the optimal vertex covering of G consists of taking only the star-center v_1 that covers any edge of G. Hence, the approximation ratio achieved here is equal to 2.

Ratio 2 is the best known approximation ratio for MIN VERTEX COVER and its improvement is a very interesting problem remaining open since three decades. Actually, the best approximation result known for this problem is only a second-order improvement of the result presented here; it is bounded above by $2 - O(\log\log n/\log n)$ [6,24].

2.2. Bin packing

BIN PACKING has been tackled by numerous approximation algorithms guaranteeing more or less good approximation ratios. One of the most natural among these algorithms is the so-called *first fit* algorithm working as follows (denoting by $L = \{u_1, \ldots, u_n\}$ a generic instance of BIN PACKING and by $|u_i|$ the size of item u_i):

- consider a sequence $B_1, \ldots, B_n$ of n empty unit-capacity bins;
- for $i = 1$ to n place item u_i in the lowest-indexed bin for which the total size of the items already contained in it is smaller than, or equal to, $1 - |u_i|$;
- output the set B of non-empty bins.

In fact, one of the features of first fit is that it never starts a new bin until a new item can be put into a bin already containing some lower-indexed items. Moreover, there exists at least one bin for which the total size of the items contained is less than, or equal to, 1/2. In fact, if there exist more than two such bins, the contents of the highest-indexed of them would be placed by the algorithm into the lowest-indexed one. Hence, $m(L, B) \leqslant \lceil 2\sum_{i=1}^{n} |u_i| \rceil$. On the other hand, one can easily see that $\text{opt}(L) \geqslant \lceil \sum_{i=1}^{n} |u_i| \rceil$. Consequently, the approximation ratio of first fit is bounded above by 2.

A finer analysis of the approximation of first fit produces $m(L, B)/\text{opt}(L) \leqslant ((17/10)) + (2/\text{opt}(L))$ and this analysis is asymptotically tight [20]. Note finally, that the best approximation result known for BIN PACKING is an asymptotic polynomial time approximation scheme, i.e., a sequence of algorithms achieving approximation ratio $1 + \epsilon + (1/\text{opt}(L))$, for any L and for any $\epsilon > 0$. On the other hand, unless $\mathbf{P} = \mathbf{NP}$, BIN PACKING cannot be solved by a polynomial time approximation scheme. In fact, such a scheme would be used to solve the PARTITION problem, that is strongly **NP**-complete [16], in polynomial time.

2.3. Metric min tsp

We consider here instances of MIN TSP on a metric space, i.e., with edge distances satisfying the triangle inequality; in other words, for any three edges (v_i, v_j), (v_j, v_k) and (v_i, v_k), the sum of the distances of two of them is at least equal to the distance of the third one. This problem can be

approximately solved by the following famous algorithm known as Christofides' algorithm by the name of its author [9]:

1. compute a minimum spanning tree T in K_n;
2. determine the set of vertices having odd degree in T and construct the subgraph G' of K_n induced by them;
3. compute a perfect minimum-cost matching M of G';
4. compute an Eulerian cycle C in the partial subgraph G'' of K_n induced by the edges of T and M;
5. starting from C compute a Hamiltonian cycle H of K_n by visiting vertices in the order of their first occurrence in C.

Note that the order of G' constructed in step 2 is always even (or nul). The computation of M in step 3 can be done in polynomial time by flow techniques [22]; the computation of C in step 4 is polynomial also (simply remark that G'' is connected and the degrees of all of its vertices are even; this is a necessary and sufficient condition in order that it admits an Eulerian cycle). Finally, for step 5 remark that, on the one hand, the vertex set of G'' coincides with the vertex set of K_n and, on the other hand, an Eulerian cycle, since it passes on any edge of G'', it also visits (many times) all of its vertices. Hence, if we consider any of them only once, then the sequence obtained corresponds to Hamiltonian cycle since K_n is complete.

Assume that H, C and T are represented by the set of their edges and set: $d(H) = \sum_{e\in H} d(e)$, the total distance of the Hamiltonian cycle H returned by the algorithm in step 5, $d(C) = \sum_{e\in C} d(e)$, the total distance of the Eulerian cycle C computed in step 4, $v(M) = \sum_{e\in M} d(e)$ and $v(T) = \sum_{e\in T} d(e)$, the values of the perfect matching M and of the spanning tree T, computed in steps 3 and 1, respectively.

Since we deal with triangle inequalities, $m(K_n, H) = d(H) \leqslant d(C) = v(T) + v(M)$. On the other hand, $\mathrm{opt}(K_n) \geqslant v(T)$ (deleting an edge from a Hamiltonian cycle one obtains a Hamiltonian path that is a spanning tree of K_n). Moreover $\mathrm{opt}(K_n) \geqslant 2v(M)$. In fact, consider an optimal tour of K_n. Starting from it, one can easily construct, thanks to triangle inequalities, a Hamiltonian tour H' of G' of total distance $d(H') = \mathrm{opt}(K_n)$. Since H' is a Hamiltonian cycle of G', it can be seen as the union of two disjoint perfect matchings of G' each one of cardinality at least $v(M)$; in all, $\mathrm{opt}(K_n) \geqslant d(H') \geqslant 2v(M)$. We so get $m(K_n, H) \leqslant v(T) + v(M) \leqslant \mathrm{opt}(K_n) + (\mathrm{opt}(K_n)/2) \leqslant 3\mathrm{opt}(K_n)/2$. Hence, Christofides algorithm achieves approximation ratio bounded above by 3/2. Moreover, this bound is tight [11] and constitutes the best approximation result known for METRIC MIN TSP.

A particular class of METRIC MIN TSP problems, the ones where edge distances are either 1 or 2 denoted by MIN TSP12, is approximable within approximation ratio 7/6 [28]. Note that this version has an undoubted historical value since the **NP**-completeness of MIN TSP has in fact been established for MIN TSP12. METRIC MIN TSP cannot be solved by a polynomial time approximation scheme unless $\mathbf{P} = \mathbf{NP}$ [28]. This result has been refined in [26] where it is shown that METRIC MIN TSP cannot be approximated within $129/128 - \epsilon$, $\forall \epsilon > 0$, unless $\mathbf{P} = \mathbf{NP}$. Finally, the best known inapproximability bound for MIN TSP12 is $743/742 - \epsilon$, for any $\epsilon > 0$ [15]. If we restrict ourselves on the plane, the MIN TSP version obtained, called EUCLIDEAN MIN TSP is solvable by a polynomial time approximation scheme [1]. As we shall see in Section 6, the general version of MIN TSP is inapproximable within any ratio strictly better than $d_{\max}/(nd_{\min})$, unless $\mathbf{P} = \mathbf{NP}$, where $d_{\max}$ and $d_{\min}$ denote, respectively, the maximum and minimum edge distances in MIN TSP instance.

3. Approximation schemes and the classes PTAS and FPTAS

3.1. Polynomial time approximation scheme and the class PTAS

As for class **APX**, **PTAS** is the class of the **NPO** problems solved by polynomial time approximation scheme. This approximation level represents a very favorable scenario for the approximability of the problems dealt, since ratios attained can be as close to 1 as one wishes. Let us note that wide

classes of (provably) **NP**-hard problems, like the whole class of dense **MAX-SNP** (see Section 6 for definition of this class) can be solved by polynomial time approximation scheme [2,21].

We give in this section a simple example of how one can obtain a polynomial time approximation scheme. Consider the following scheduling problem called MIN MAKESPAN. Assume $\ell \geqslant 2$ a fixed constant and consider n tasks $t_1, \ldots, t_n$ with respective lengths $d_1, \ldots, d_n$ that can indifferently be executed on any of ℓ identical machines (there is no precedence constraints; the tasks are supposed unsplittable). We ask for determining a scheduling that minimizes the utilization time of the most busy processor.

If the number of tasks is fixed, then we can solve this problem by exhaustive search in constant time. Hence, we can assume $n > \ell$. Then, for any $k \in \mathbb{N}$, run the following algorithm, denoted by $\mathtt{A}_k$:

- range tasks in decreasing length order;
- determine an optimal scheduling of the first k tasks (by exhaustive search);
- greedily complete the scheduling obtained above by affecting the first non-affected task to the less charged processor;
- output S, the scheduling computed.

Algorithm $\mathtt{A}_k$ is polynomial since k is assumed fixed. Consider an instance I of size $n > \max\{k, \ell\}$ and denote by $opt_k(I)$ the optimal value of I when restricted to the k longest (first) tasks. Obviously, $m_{\mathtt{A}_k}(I,S) \geqslant \mathrm{opt}(I) \geqslant \mathrm{opt}_k(I)$; furthermore, if $m_{\mathtt{A}_k}(I,S) = \mathrm{opt}_k(I)$, then MIN MAKESPAN is polynomially solved by $\mathtt{A}_k$ in I. So, we can assume $m_{\mathtt{A}_k}(I,S) > \mathrm{opt}_k(I)$. In this case, there exists a task t_j, $j > k$, completed at time $m_{\mathtt{A}_k}(I,S)$. Since task t_j has been greedily affected to a processor, we can deduce that any machine is saturated until instant $m_{\mathtt{A}_k}(I,S) - d_j$. Setting $D = \sum_{i=1}^{n} d_i$, we get: $D \geqslant \ell(m_{\mathtt{A}_k}(I,S) - d_j) + d_j$; since $d_j \leqslant d_{k+1}$, we obtain $D \geqslant \ell m_{\mathtt{A}_k}(I,S) - (\ell-1)d_{k+1}$. Moreover, $\mathrm{opt}(I) \geqslant D/\ell$. We so have: $m_{\mathtt{A}_k}(I,S) \leqslant \mathrm{opt}(I) + (1-(1/\ell))d_{k+1}$. On the other hand, considering the location of the k longest tasks in an optimal scheduling, we get $\mathrm{opt}(I) \geqslant (k+1/\ell)d_{k+1}$ which finally implies $m_{\mathtt{A}_k}(I,S)/\mathrm{opt}(I) \leqslant (k+\ell)/(k+1) = 1 + ((\ell-1)/(k+1))$.

Hence, one can fix an $\epsilon > 0$ and choose $k = k_\epsilon \geqslant ((\ell-1)/\epsilon) - 1$; then algorithm $\mathtt{A}_{k_\epsilon}$ achieves approximation ratio bounded above by $1+\epsilon$. Since ℓ has been assumed fixed, k_ϵ is independent on n. In other words, *setting* $k_\epsilon = \lceil((\ell-1)/\epsilon) - 1\rceil$ *the family of algorithms* $\mathtt{A}_{k_\epsilon}$ *constitutes a polynomial time approximation scheme.*

3.2. Fully polynomial time approximation schemes and the class FPTAS

By the way a polynomial time approximation scheme is defined in the literature, if one demands that ϵ is a function of $|I|$, then the complexity of the scheme can be exponential in n. This is quite obvious in Section 3.1 where for ϵ depending on n, k_ϵ does so, and consequently, the exhaustive search in the first step of algorithm $\mathtt{A}_{k_\epsilon}$ is no more polynomial.

A fully polynomial time approximation scheme is a polynomial time approximation scheme with time complexity that is polynomial in both $|I|$ and $1/\epsilon$. Hence, ϵs are allowed to be functions of $|I|$. So, a fully polynomial time approximation scheme represents an ideal scenario for polynomially computing solutions of **NP**-hard problems, since the approximation ratios attained by such schemes can be bounded above or below by values of the form $1 \pm (1/|I|)^k$, for any fixed constant $k > 0$. **FPTAS** denotes the class of **NPO** problems admitting fully polynomial time approximation schemes.

We present here a classical fully polynomial time approximation scheme for KNAPSACK. An instance of KNAPSACK can be written as an integer linear program in the following way:

$$I = \begin{cases} \max & \sum_{i=1}^{n} a_i x_i \\ & \sum_{i=1}^{n} c_i x_i \leqslant b \end{cases}$$

We set

$$a_{\max} = \max_{i=1,\ldots,n} \{a_i\} \quad \text{and} \quad c_{\max} = \max_{i=1,\ldots,n} \{c_i\}.$$

The decision version KNAPSACK is **NP**-complete in the weak sense; its optimization version can be solved to optimality by dynamic programming with time-complexity $O(n^2 a_{\max} \log c_{\max})$ [16]. This

algorithm can be turned into a fully polynomial time approximation scheme in the following way:

- fix an $\epsilon > 0$;
- construct $I' = ((a'_i, c_i)_{i=1,\dots,n}, b)$ with $a'_i = \lfloor a_i n/(a_{\max}\epsilon)\rfloor$;
- output as solution for I, solution S computed by dynamic programming on I'.

Clearly, algorithm above works in time $O(n^2 a'_{\max}\log c_{\max}) = O((n^3 \log c_{\max})/\epsilon)$ which is polynomial in n. Let S^* be an optimal solution for I; obviously, it is feasible for I'. Setting $t = a_{\max}\epsilon/n$, we get, taking into account that $|S^*| \leqslant n$:

$$\begin{aligned}\operatorname{opt}(I') &\geqslant \sum_{i\in S^*} a'_i \geqslant \sum_{i\in S^*}((a_i/t) - 1)\\ &\geqslant (\operatorname{opt}(I)/t) - |S^*| \geqslant (\operatorname{opt}(I)/t) - n.\end{aligned}$$

This implies: $t\operatorname{opt}(I') \geqslant \operatorname{opt}(I) - nt$. On the other hand, one can assume that, $\forall i = 1,\dots,n$, $c_{\max} \leqslant b$ (if not, we can drop item i out of I). Consequently, solution consisting in taking only $i_0 = \arg\max_{i=1,\dots,n}\{a_i\}$ is feasible for I; so, $\operatorname{opt}(I) = \sum_{i\in S^*} a_i \geqslant a_{\max}$.

Combining this expression with the fact that $nt = a_{\max}\epsilon$, we get: $nt = a_{\max}\epsilon \leqslant \epsilon\operatorname{opt}(I)$. Since $a'_i \leqslant a_i/t$, $m(I,S) = \sum_{i\in S} a_i \geqslant t\sum_{i\in S} a'_i = t\operatorname{opt}(I')$. In all, by what has been discussed, $m(I,S) \geqslant \operatorname{opt}(I) - nt \geqslant \operatorname{opt}(I) - \epsilon\operatorname{opt}(I) \geqslant (1-\epsilon)\operatorname{opt}(I)$, q.e.d.

4. Beyond APX

There exist problems for which, either no constant-ratio approximation algorithms are known, or such algorithms cannot exist under a strong complexity theoretic hypothesis (as $\mathbf{P} \neq \mathbf{NP}$). Approximation ratios achieved in such cases, even by the most efficient algorithms known, are rather pessimistic since they are functions of instance parameters that tend either to 0 (dealing with maximization problems) or to ∞ (dealing with minimization problems) whenever the instance size tends to ∞. The most common parameters used for analyzing approximation ratios are, for example, the order n or the maximum degree $\Delta(G)$ of an input G, when dealing with graph problems, the size or the maximum cardinality set, when dealing with set systems, etc. Ratios functions of such parameters, exhibit new approximability classes beyond **APX**; the most notorious ones are, for the case of minimization problems:

Poly-APX: The class of problems approximable within ratios of the form $O(|I|^k)$ for some constant $k \geqslant 0$;
Log-APX and Polylog-APX: The classes of problems approximable within ratios of $O(\log|I|)$ (for the first case), or of the form $O(\log^k|I|)$ for any $k \geqslant 0$ (for the second case);
Exp-APX: The class of problems approximable within ratios of the form $O(2^{p(|I|)})$ for some polynomial p.

Obviously, the definition of these classes for maximization problems are analogous up to considering exponent $k < 0$.

We present in what follows two approximation algorithms for two famous optimization problems: MIN SET COVER and MAX INDEPENDENT SET.

4.1. The greedy algorithm for unweighted MIN SET COVER

Let $I(\mathscr{S}, C)$ be an instance of MIN SET COVER. Denote by Δ the quantity $\max_{S_i\in\mathscr{S}}\{|S_i|\}$, by $\mathscr{S}'$ a feasible set cover and consider the following natural greedy algorithm for MIN SET COVER originally studied in the seminal paper of [19]:

1. compute $S \in \arg\max_{S_i\in\mathscr{S}}\{|S_i|\}$;
2. set: $\mathscr{S}' = \mathscr{S}' \cup \{S\}$ ($\mathscr{S}'$ is considered to be empty at the beginning of the algorithm);
3. update I setting: $\mathscr{S} = \mathscr{S} \setminus \{S\}$, $C = C \setminus S$ and, for $S_j \in \mathscr{S}$, $S_j = S_j \setminus S$;
4. repeat steps 1 and 3 until $C = \emptyset$.

It is very easy to see that algorithm above is polynomial. We present now an elegant analysis of the greedy algorithm as it is proposed in [23]. We prove that its approximation ratio is bounded above by $1 + \ln\Delta$. Hence, MIN SET COVER $\in \mathbf{Log - APX}$.

Denote by $I_i(\mathscr{S}_i, C_i)$ the surviving instance at the first moment residual cardinalities of sets in $\mathscr{S}$

are at most i; denote by $m(I_i, \mathscr{S}')$ the number of sets of residual cardinality i placed in $\mathscr{S}'$; note that $m(I_\Delta, \mathscr{S}') = m(I, \mathscr{S}')$.

For $i = 1, \ldots, \Delta$, we have the following Δ-line equation-system: $|C_i| = \sum_{k=1}^{i} k \times m(I_k, \mathscr{S}')$. Any such equation is simply the expression of the fact that, anytime a set S is chosen to be part of $\mathscr{S}'$, algorithm removes from C the elements of S. Multiply the Δth line of the above system by $1/\Delta$ and, for the other lines, multiply the ith line by $1/i(i+1)$. Then, taking into account that $C_\Delta = C$ and $m(I, \mathscr{S}') = \sum_{i=1}^{\Delta} m(I_i, \mathscr{S}')$, we get:

$$\left(\sum_{i=1}^{\Delta-1} |C_i|/i(i+1)\right) + |C|/\Delta = \sum_{i=1}^{\Delta} m(I_i, \mathscr{S}') = m(I, \mathscr{S}'),$$

where we have used that $1/(i \times (i+1)) = ((1/i) - (1/(i+1)))$.

Denote by $\mathscr{S}^*$ an optimal solution for I and by $\mathscr{S}_i^*$ an optimal solution for I_i, $i = 1, \ldots, \Delta$. Elements of $\mathscr{S}^*$ covering C_i are always present in I_i and form a feasible solution for it. Therefore, $\mathrm{opt}(I_i) = |\mathscr{S}_i^*| \leqslant |\mathscr{S}^*| = \mathrm{opt}(I)$ and $|C_i| \leqslant i \times \mathrm{opt}(I_i)$. Combination of the three last expressions derives:

$$m(I, \mathscr{S}') \leqslant \mathrm{opt}(I) \times \sum_{i=1}^{\Delta} (1/i) \leqslant \mathrm{opt}(I)(1 + \ln \Delta),$$

q.e.d.

We now show that this ratio is asymptotically tight; the instance we are going to describe is taken from [19]. Fix a k and consider the instance of Fig. 1, designed for $k = 4$. We are given a set C of 3×2^k elements: $C = \{1, 2, \ldots, 3 \times 2^k\}$ and a family $\mathscr{S} = \{S_1, S_2, \ldots, S_{k+4}\}$ of $k+4$ subsets of C. For simplicity, consider that elements of C are entries of a $3 \times (k+1)$ matrix. Then, sets in $\mathscr{S}$ are as follows:

- three disjoint sets: S_1, S_2 et S_3, each one containing the 2^k elements of each of the three lines of the matrix; they form a partition on C, i.e., $C = S_1 \cup S_2 \cup S_3$ and $S_i \cap S_j = \emptyset$, $i, j = 1, 2, 3$;
- $k+1$ sets $S_4, \ldots, S_{k+4}$, of respective sizes: $3 \times 2^{k-1}, 3 \times 2^{k-2}, \ldots, 3 \times 2^1, 3 \times 2^0$ and 3 contain, respectively, the points of the 2^{k-1}, $2^{k-2}, \ldots, 2^1, 2^0$ and 1 columns.

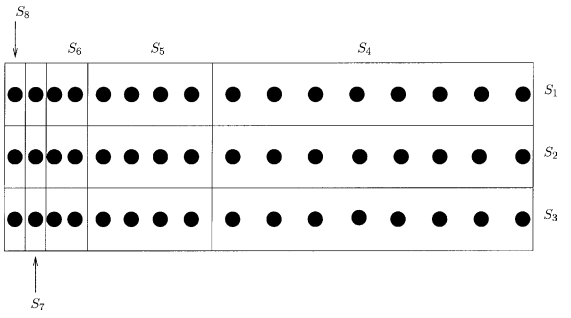

Fig. 1. On the tightness of the greedy MIN SET COVER-algorithm with $k = 4$.

It is easy to see that the greedy algorithm presented above will choose the $k+1$ sets $S_4, \ldots, S_{k+4}$, in this order, returning a set cover of size $k+1$, while optimal set cover is family $\{S_1, S_2, S_3\}$ of size 3. Note finally that, for the instance dealt $\Delta = 3 \times 2^{k-1}$. Consequently, approximation ratio achieved here is $(k+1)/3 = O(\ln(3 \times 2^{k-1}))$.

A more recent elegant analysis of the greedy algorithm given in [31] provides a tight ratio of $O(\log |C|)$. On the other hand, as proved in [29], it is impossible to approximate MIN SET COVER within better than $O(\log |C|) - O(\log \log |C|)$, unless a highly unlikely complexity classes relationship is true.

Note finally that for weighted MIN SET COVER one can slightly modify the greedy algorithm presented above by choosing in step 1 a set $S = \arg\max_{S_i \in \mathscr{S}} \{|S_i|/w(S_i)\}$, where $w(S_i)$ denotes the weight of set $S_i \in \mathscr{S}$. In this case also, the approximation ratio achieved is bounded above by $1 + \ln \Delta$ [10].

4.2. Max independent set and local search

We present in this section a local-search-based approximation algorithm for MAX INDEPENDENT SET. Given a graph $G(V, E)$, an independent set S, and a constant $a \in \mathbb{N}$, an *a-improvement* consists of removing $a' < a$ vertices from S and of adding $a'+1$ new vertices of $V \setminus S$ in such a way that the so-obtained new set S remains an independent set.

When no 1-improvement is possible, the resulting independent set is maximal for the inclusion. A 2-improvement consists, either of an 1-improvement ($a' = 0$), or of removing one vertex u from S and of adding two new vertices v_1 and v_2, in such a way that $S \cup \{v_1, v_2\} \setminus \{u\}$ remains independent.

In what follows, we assume that an initial independent set is computed by application of the greedy MAX INDEPENDENT SET-algorithm. It consists of iteratively taking the current maximum-degree vertex of G and of removing its neighbors until V becomes empty. Consider now the following algorithm:

1. apply the greedy algorithm for computing an initial independent set S (remark that even setting $S = \emptyset$ suffices to initialize the algorithm);
2. while a 2-improvement (v_1, v_2, u) of S is possible, set $S = S \cup \{v_1, v_2\} \setminus \{u\}$;
3. output S.

This algorithm works in $O(n^4)$. We show that it achieves approximation ratio at least $2/(\Delta(G)+1)$, where $\Delta(G)$ denotes the maximum vertex degree of G.

Let S^* be a maximum independent set and n be the order of G. Without loss of generality, we can restrict ourselves to the case $S^* \cap S = \emptyset$. Indeed, it suffices to remark that the approximation ratio for solution S is

$$(|S \setminus S^*| + |S \cap S^*|)/(|S^* \setminus S| + |S \cap S^*|) \geqslant |S \setminus S^*|/|S^* \setminus S|.$$

Decompose now S^* into $S_1^* \cup S_2^*$ where, denoting by $\Gamma(s)$ the set of neighbors of $s \in V$, $S_1^* = \{s \in S^* : |\Gamma(s) \cap S| = 1\}$ (vertices of S^* having only one neighbor in S). Remark that a vertex of S cannot have more than one neighbor in S_1^*, since, in the opposite, step 2 of algorithm above should be executed one more time; hence $|S| \geqslant |S_1^*|$. Set S being maximal for inclusion, the number of edges incident to the vertices of S is at least equal to $|S_1^*| + 2|S_2^*| \geqslant 2|S^*| - |S|$. A simple edge-counting deduces: $\Delta(G) \times |S| \geqslant 2|S^*| - |S|$ and approximation ratio $2/(\Delta(G)+1)$ comes immediately from this expression.

Since $\Delta(G) \leqslant n$, the ratio just provided is bounded below by $2/n$; so, one can immediately conclude that MAX INDEPENDENT SET $\in$ **Poly-APX**.

The best ratios for MAX INDEPENDENT SET known up to date are: $O(\log^2 n/n)$, when dealing with ratios functions of the order n of the input graph [8], $k/\Delta(G) - o(1/\Delta(G))$, $\forall k \in \mathbb{N}$, when dealing with ratios functions of the maximum degree $\Delta(G)$ of the input graph [13], and $O(\log n/(\Delta(G)\log\log n))$, when dealing with ratios functions of both n and $\Delta(G)$ [14].

For weighted MAX INDEPENDENT SET the best approximation ratios known are: $O(\log^2 n/n)$, when dealing with ratios functions of n [17] and $O(\log n/(\Delta(G)\log\log n))$, when dealing with ratios functions of both n and $\Delta(G)$ [14].

5. Approximation preserving reductions

A conclusion of what has been discussed previously is that, following the approximation ratio of the best algorithm known, problems in **NPO** can be seen as belonging to approximability classes as **APX**, **PTAS**, etc. The most notorious such classes have already been mentioned above and are shown in Fig. 2 (dealing with standard approximation ratio; the landscape in the case of differential approximation is similar, but with a different

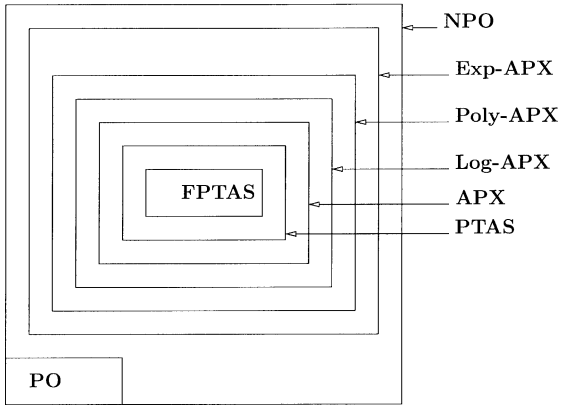

Fig. 2. The most notorious approximability classes.

distribution of the problems in these classes), where **PO** denotes the class of optimization problems optimally solved in polynomial time. Classes inclusions in the figure are strict.

This way of building **NPO** hierarchy is somewhat "absolute" in the sense that problems are classified following algorithms solving them. However, this classification does not allow comparisons between approximability properties of problems. For instance, if we restrict ourselves in such classification, we cannot answer, or we can only partially answer, to questions like

- how can one compare problems with respect to their approximability properties and independently on their respective approximation levels?
- how one can compare approximability of different versions of the same problem (for example, weighted version vs. unweighted one)?
- how one can link different types of approximation for a same problem (for instance, do there exist tranferts of approximability results between standard and differential approximation for a given problem)?
- how to apprehend the role of parameters in the study of approximability (for example, we have seen that the function describing approximation ratio for MAX INDEPENDENT SET is different when dealing with n or when dealing with $\Delta(G)$)?
- can we transfer approximation results from a problem to another one?
- can we refine structure of the approximation classes given above (by providing, for example, complete problems for these classes)?

Approximation people tries to provide answers to these questions (and to many other ones) by borrowing tools from complexity theory; these tools are mainly carefully defined reductions called *approximation preserving reductions*. Any of the existing ones imposes particular conditions on the way optimal solutions, or approximation ratios, or ..., are transformed from one problem to the other one. The scope of the paper is not to present them but rather to give an idea of how these reductions are used in approximation. For more details, the interested reader can be referred in [5,12,18,32].

In general, given two **NPO** problems $\Pi = (\mathscr{I}, \mathrm{Sol}, m, \mathrm{opt})$ and $\Pi' = (\mathscr{I}', \mathrm{Sol}', m', \mathrm{opt}')$, an approximation preserving reduction **R** from Π to Π' is a triple (f, g, c) of polynomially computable functions such that

- f transforms an instance $I \in \mathscr{I}$ into an instance $f(I) \in \mathscr{I}'$;
- g transforms a solution $S' \in \mathrm{Sol}'(f(I))$ into a solution $g(I, S') \in \mathrm{Sol}(I)$;
- c transforms ratio $\rho'(f(I), S')$ into $\rho(I, g(I, S')) = c(\rho'(f(I), S'))$.

Then, if Π' is approximable within ratio ρ', Π is approximable within ratio $\rho = c(\rho')$. On the other hand, if, under a likely complexity hypothesis, Π is not approximable within ratio ρ, then Π' is not approximable within ratio $\rho' = c^{-1}(\rho)$ (when c^{-1} is definable).

We now give two examples of approximation preserving reductions helping us to exhibit abilities but also limits of these tools.

5.1. Max induced bipartite subgraph and max independent set

We first describe an approximation preserving reduction from MAX INDUCED BIPARTITE SUBGRAPH to MAX INDEPENDENT SET.

Let $G(V, E)$ be an instance of MAX INDUCED BIPARTITE SUBGRAPH of order n. We construct an instance $G'(V', E')$ of MAX INDEPENDENT SET of order $n' = 2n$ by taking two copies G_1 and G_2 of G and by linking by an edge the two copies of the same vertex of G. In other words, the transversal edges form a perfect matching of the form $\{(v_i^1, v_i^2) : v_i \in V\}$. It is easy to see that any independent set of size $|S'|$ in G' corresponds to a bipartite subgraph of G of size $|S| = |S'|$. Consequently, optimal values of both problems coincide. Furthermore, any approximate solution for MAX INDEPENDENT SET on G' (i.e., any solution computed by a polynomial time approximation algorithm for MAX INDEPENDENT SET on G') of a certain size can be immediately transformed into an approximate solution for MAX INDUCED BIPARTITE SUBGRAPH on G of the same order (think that an independent set is a bipartite graph

by itself). Hence, the ratios for the two problems dealt have the same value in the corresponding instances. One concludes that c is the identity function and could derive that $\rho = c(\rho') = \rho'$.

Things are, however, somewhat more complicated. This is absolutely true when ρ' is a fixed constant. Consider now the case where ρ' is function of the order of the input graph. In this case, the value of the ratio derived for MAX INDUCED BIPARTITE SUBGRAPH on G is $\rho'(2n) = \rho(n)$, i.e., considering that ρ decreases with the order of the graph, this ratio is smaller than $\rho(n)$ which would be the value of the ratio if identity of ratios held independently on the parameter with respect to which ratio is analyzed. Also, if ρ' is function of the maximum degree of the input graph, since $\Delta(G') = \Delta(G) + 1$, the ratio guaranteed for MAX INDUCED BIPARTITE SUBGRAPH on G is $\rho'(\Delta(G) + 1) = \rho(\Delta(G))$. As a consequence, magnitudes of ratios expressed as functions of the order of the input graph, or of the maximum graph degree for MAX INDEPENDENT SET, are preserved when they are transferred to MAX INDUCED BIPARTITE SUBGRAPH.

Let us now describe the inverse operation, i.e., an approximation preserving reduction from MAX INDEPENDENT SET to MAX INDUCED BIPARTITE SUBGRAPH. Consider a graph $G(V, E)$ of order n instance of MAX INDEPENDENT SET. The instance $G'(V', E')$ derived for MAX INDUCED BIPARTITE SUBGRAPH consists of two copies $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ of G (with $V_1 = V_2 = V$ and $E_1 = E_2 = E$) and of all the transversal edges between these two copies, i.e., any vertex of V_1 is linked to any vertex of V_2. Given an independent set S of G, the subgraph induced by the copies of S in G_1 and G_2 is bipartite. This is in particular true when dealing with a maximum independent set S^* in G and such a bipartite graph is optimal for MAX INDUCED BIPARTITE SUBGRAPH in G'. Hence, for an optimal solution S'^* of the latter problem, we have an optimal solution S^* of the former one with value the half of the value of S'^*; on the other hand, given any solution S' for MAX INDUCED BIPARTITE SUBGRAPH in G' one can construct an independent set of G of size at least the half of the order of S' by simply retaining the largest of the two color-classes of S'. Consequently, if an algorithm solves MAX INDUCED BIPARTITE SUBGRAPH in G' within approximation ratio ρ', the approximation ratio ρ derived, from the discussion above, for MAX INDEPENDENT SET in G is at least as good as ρ'. Once more, one could say that c is the identity function. Let us see what reality is

- constant ratios for MAX INDUCED BIPARTITE SUBGRAPH are transformed into constant ratios (of the same value) for MAX INDEPENDENT SET;
- ratios ρ' functions of the order of the input graph for MAX INDUCED BIPARTITE SUBGRAPH are transformed into ratios $\rho(n) = \rho'(2n)$ for MAX INDEPENDENT SET;
- ratios ρ' functions of the maximum degree of the input graph for MAX INDUCED BIPARTITE SUBGRAPH are transformed into ratios $\rho'(\Delta(G')) = \rho'(n + \Delta(G)) = \rho(n)$ for MAX INDEPENDENT SET.

5.2. Max independent set and max clique

There exists a very well-known strong reduction (which is considered as a kind of equivalence) between MAX INDEPENDENT SET and MAX CLIQUE. It is expressed by the fact that an independent set in a graph G becomes a clique of the same size in $\overline{G}$, the complement of G and vice versa. But this reduction, even if it preserves constant ratios and also ratios depending on the order of the input graph (the orders of a graph and of its complement are identical), it does not preserve ratios expressed as functions of the maximum graph degree. Indeed, there is no clear relation between $\Delta(G)$ and $\Delta(\overline{G})$.

We describe here another kind of reduction which transforms ratios functions of the order of the input graph for MAX INDEPENDENT SET to ratios of the same form but functions of the maximum graph degree for MAX CLIQUE. Also, this reduction asymptotically preserves ratios functions of the maximum degree. Consider any polynomial time approximation algorithm A for MAX INDEPENDENT SET, assume that A achieves ratio ρ function of the order of the input graph, consider a graph $G(V, E)$ input of MAX CLIQUE and proceed as follows:

1. for any $v \in V$ construct $G_v = G[v \cup \Gamma(v)]$ and $\overline{G}_v$;
2. run A on any $\overline{G}_v$ constructed in step 1;
3. retain the best (greatest) among the independent sets computed in step 2; denote it by $\overline{S}_0$;
4. output the clique K_0 of G associated with $\overline{S}_0$.

Any independent set of $\overline{G}_v$, $v \in V$, corresponds to a clique of G_v on the same vertex set. Furthermore, any maximum-size clique of G is contained in one of the G_vs constructed in step 1 (a clique is a vertex and a part of its neighborhood). Fix a maximum clique K^* of G and assume that K^* is included in G_{v_0} (one of the graphs constructed in step 1). Denote also by $\overline{S}^*$ the maximum independent set of $\overline{G}$ associated with K^* and denote by $\overline{S}_{v_0}$ the independent set computed by A in $\overline{G}_{v_0}$ during step 2. Obviously, $|\overline{S}_0| \geqslant |\overline{S}_{v_0}|$. Therefore, by the hypothesis on A, $|\overline{S}_0|/|\overline{S}^*| = |S_0|/|K^*| \geqslant \rho(n_{v_0})$, where n_{v_0} denotes the order of $\overline{G}_{v_0}$ and G_{v_0}. It suffices now to remark that $n_{v_0} \leqslant \Delta(G) + 1$ in order to conclude about the first claim, i.e., that an approximation ratio function of the order of the input for MAX INDEPENDENT SET is transformed, by the reduction just described, into a ratio of the same form function of the maximum graph degree for MAX CLIQUE. The second claim is concluded analogously. It suffices to remark that, $\forall v \in V$, $\Delta(\overline{G}_v) \leqslant n_v \leqslant \Delta(G) + 1$.

This reduction allows us, based upon the $O(\log^2 n/n)$-approximation ratio of [8] for MAX INDEPENDENT SET, to derive an approximation ratio of

$$O\big(\log^2(\Delta(G)+1)/(\Delta(G)+1)\big) = O\big(\log^2 \Delta(G)/\Delta(G)\big)$$

for MAX CLIQUE. Hence, it allows us to conclude that parameters n and $\Delta(G)$ are "equivalent" for MAX CLIQUE, in the sense that ratios functions of the one can be transformed into ratios functions of the other, all these functions being of the same form. This is not true for MAX INDEPENDENT SET (simply think that this problem is **NP**-hard even in graphs with maximum degree bounded by a fixed constant greater than, or equal to, 3).

6. Inapproximability

We take in this section a very quick glance to another field of polynomial approximation (somewhat far from the classical operations research problematic), trying to produce tools and methods in order to provide limits to the approximability capacities of **NPO** problems.

Let us introduce the initial idea for obtaining inapproximability results by strengthening a result [30] on general MIN TSP. We will show that, *unless* $\mathbf{P} = \mathbf{NP}$ *no polynomial time algorithm can achieve approximation ratio strictly smaller than* $d_{\max}/(nd_{\min})$ *for* MIN TSP. A contrario, suppose that there exists a polynomial time algorithm A achieving such ratio. We will show how, based upon it, one can decide if a graph G is Hamiltonian, or not. One can transform G (suppose it of order n), instance of HAMILTONIAN CYCLE, into an instance $(K_n, \vec{d})$ of MIN TSP in the following way:

- add the missing edges of G in order to produce K_n;
- for any $(v_i, v_j) \in E(K_n)$, set $d(i,j) = 1$, if $(v_i, v_j) \in E(G)$, $d(i,j) = W$, otherwise; then, $d_{\max} = W$ and $d_{\min} = 1$.

If G is Hamiltonian, then $\mathrm{opt}(K_n) = n$ and algorithm A will return a tour T with value $m(K_n, T) < n(d_{\max}/(n \times d_{\min})) = d_{\max}/d_{\min}$. If, on the other hand, G is not Hamiltonian, then $\mathrm{opt}(K_n) > W = d_{\max}/d_{\min}$ and, obviously, the tour returned by A will have value $m(K_n, T) > \mathrm{opt}(K_n) > d_{\max}/d_{\min}$. Consequently, it suffices to read the value of the solution computed by A in order to decide (in polynomial time, following the hypothesis on the complexity of A) if G is Hamiltonian, or not. Since HAMILTONIAN CYCLE is **NP**-complete, this is not possible unless $\mathbf{P} = \mathbf{NP}$.

Proof above draws a general strategy, called *gap technique*, for proving inapproximability results by reductions. Let Π be an **NP**-complete decision problem and Π' an **NP**-hard minimization problem of **NPO** (the case of maximization is completely analogous). If one devises a reduction from Π to Π' (transforming an instance I of Π into an instance I' of Π') such that, for some constants k and r:

- if the answer for Π in I is *yes*, then $\mathrm{opt}(I') \leqslant k$;
- if, on the other hand, the answer for Π in I is *no*, then $\mathrm{opt}(I') > rk$,

then Π' is not approximable within ratio r (or, there exists an approximation gap r for Π'). In fact, provided that we have devised such a reduction, if a polynomial algorithm A achieving approximation ratio r existed for Π', then, on the hypothesis that answer in I is *yes*, $\mathrm{opt}(I') \leqslant k$ and the value of the solution returned by A would be at most rk. On the other hand, if answer in I is *no*, then the value of the solution returned by A is, obviously greater than $\mathrm{opt}(I') > rk$.

Suppose now that there exists an approximability preserving reduction $\mathsf{R} = (f, g, c)$ from Π' to Π''. Then it is easy to see that, unless $\mathbf{P} = \mathbf{NP}$, Π'' is not approximable within ratio $c^{-1}(r)$. A great number of the inapproximability results known are produced by such combinations. In other words, one tries to devise approximability preserving reductions from a problem for which an inapproximability result is already known to another problem for which one tries to deduce such result. For example, since MAX INDEPENDENT SET is not approximable within approximation ratio $O(n^{\epsilon-1})$, $\forall \epsilon > 0$, unless problems in **NP** can be solved by slightly super-polynomial algorithms, reduction of Section 5.1, allows us to conclude that the same holds for MAX INDUCED BIPARTITE SUBGRAPH.

The gap technique described above has been strengthened during last years by a very novel characterization of **NP** using the concept of probabilistic checkable proofs (known as **PCP** system). This concept is out of the scope of this paper, but we will give some words about it.

Shortly, dealing with an instance I of a decision problem Π, an interactive proof system is a kind of particular conversation between a prover (P) sending a candidate solution S for I, and a verifier (V) accepting or rejecting it; this system decides I if, (i) for any S (sent by P), V always accepts it if it constitutes a feasible solution for I, and (ii) for any non-feasible S, neither P, nor any imposter substituting P, can make V accept S (as feasible solution) with probability greater than p, for some $p \in \mathbb{Q}$.

Interactive proofs have produced fine characterizations for **NP** [4], highly contributing to the development of very sophisticated gap-techniques for obtaining new non-trivial inapproximability results. The most important among these corollaries, that has constituted a kind of break-through for the achievement of negative answers for numerous open problems in polynomial approximation, is the one of [4] that *no* **Max-SNP**-*hard problem admits a polynomial time approximation scheme*, unless $\mathbf{P} = \mathbf{NP}$ (as we have mentioned in Section 3.1, dense instances of problems in **Max-SNP** are solvable by polynomial time approximation scheme). The class **Max-SNP** is introduced in [27] and, informally, it is a subclass of maximization problems of **APX** the optima of which can be syntactically defined as $\max_S |\{x \in U^k : \varphi(I, S, x)\}|$ where $(U, \mathscr{I})$ and $(U, \mathscr{S})$ are finite structures over alphabet U, $I \in \mathscr{I}$, $S \in \mathscr{S}$, φ is a quantifier-free first-order logical formula and k is a constant. The completeness of a particular problem in **Max-SNP**, holding under a special kind of approximation preserving reduction, called L-reduction in [27], means that if this problem admits a polynomial time approximation scheme, then so do all the problems in **Max-SNP**. For more about the **PCP** system and inapproximability results in general, the interested reader can be referred to [5,32] as well as to [3].

Appendix A. The NPO problems considered

Max independent set. Given a graph $G(V, E)$, an *independent set* is a subset $V' \subseteq V$ such that whenever $\{v_i, v_j\} \subseteq V'$, $v_i v_j \notin E$, and MAX INDEPENDENT SET consists in finding an independent set of maximum size. In weighted MAX INDEPENDENT SET we consider that vertices are provided with positive weights and the objective becomes to determine an independent set of maximum total weight.

Max clique. Consider a graph $G(V, E)$. A *clique* of G is a subset $V' \subseteq V$ such that every pair of vertices of V' are linked by an edge in E, and MAX CLIQUE consists in finding a maximum size set V' inducing a clique in G (a maximum-size clique).

Min vertex cover. Given a graph $G(V, E)$, a *vertex cover* is a subset $V' \subseteq V$ such that, $\forall uv \in E$,

either $u \in V'$, or $v \in V'$, and MIN VERTEX COVER consists of determining a minimum-size vertex cover.

Min set cover. Given a collection $\mathcal{S}$ of subsets of a finite set C, a *set cover* is a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ such that $\cup_{S_i \in \mathcal{S}'} S_i = C$, and MIN SET COVER consists of finding a cover of minimum size. In weighted MIN SET COVER, members of $\mathcal{S}$ are provided with positive weights and the objective becomes to determine a set cover of minimum total weight.

Min makespan. Assume $\ell \geqslant 2$ a fixed constant and consider n tasks $t_1, \ldots, t_n$ with respective lengths $d_1, \ldots, d_n$ that can indifferently be executed on any of ℓ identical machines (there is no precedence constraints; the tasks are supposed unsplittable). The objective is to determine a scheduling that minimizes the utilization time of the most busy processor.

Knapsack. Given two integer n-vectors $\vec{a}$ and $\vec{c}$ and an integer b, the objective is to determine a vector $\vec{x} \in \{0,1\}^n$ which, under the constraint $\vec{c} \cdot \vec{x} \leqslant b$, maximizes the scalar $\vec{a} \cdot \vec{x}$.

Min tsp. Given a complete graph on n vertices, denoted by K_n, with positive costs on its edges, MIN TSP consists of minimizing the total cost of a Hamiltonian cycle, this cost being the sum of the costs of its edges.

Bin packing. Given a list $L = \{u_1, \ldots, u_n\}$ of n items, each of them having size $|u_i| \leqslant 1$, and n bins, each of capacity 1, we search to arrange all items of L in these bins in such a way that we use a minimum number of them and, moreover, the sum of capacities of the items placed in any bin does not exceed 1.

Max induced bipartite subgraph. Given a graph $G(V,E)$, we search to determine the largest subset $V' \subseteq V$ such that the graph $G[V']$ is bipartite.

References

[1] S. Arora, Polynomial time approximation schemes for Euclidean TSP and other geometric problems, in: Proc. FOCS'96, 1996, pp. 2–11.

[2] S. Arora, D. Karger, M. Karpinski, Polynomial time approximation schemes for dense instances of NP-hard problems, Journal of Computer and System Sciences 58 (1999) 193–210.

[3] S. Arora, C. Lund, Hardness of approximation, in: D.S. Hochbaum (Ed.), Approximation Algorithms for NP-hard Problems, PWS, Boston, 1997, pp. 399–446, Chapter 10.

[4] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, Proof verification and intractability of approximation problems, Journal of the Association for Computing Machinery 45 (3) (1998) 501–555.

[5] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, Complexity and Approximation. Combinatorial Optimization Problems and their Approximability Properties, Springer, Berlin, 1999.

[6] R. Bar-Yehuda, S. Even, A local-ratio theorem for approximating the weighted vertex cover problem, Annals of Discrete Applied Mathematics 25 (1985) 27–46.

[7] C. Berge, Graphs and Hypergraphs, North Holland, Amsterdam, 1973.

[8] B.B. Boppana, M.M. Halldórsson, Approximating maximum independent sets by excluding subgraphs, BIT 32 (2) (1992) 180–196.

[9] N. Christofides, Worst-case analysis of a new heuristic for the traveling salesman problem, Technical Report 388, Grad. School of Industrial Administration, CMU, 1976.

[10] V. Chvátal, A greedy-heuristic for the set covering problem, Mathematics of Operations Research 4 (1979) 233–235.

[11] G. Cornuejols, G.L. Nemhauser, Tight bounds for Christofides' traveling salesman heuristic, Mathematical Programming 14 (1978) 116–121.

[12] P. Crescenzi, A short guide to approximation preserving reductions, in: Proceedings of the Conference on Computational Complexity, 1997, pp. 262–273.

[13] M. Demange, V.Th. Paschos, Improved approximations for maximum independent set via approximation chains, Applied Mathematics Letters 10 (3) (1997) 105–110.

[14] M. Demange, V.Th. Paschos, Towards a general formal framework for polynomial approximation, Cahier du LAMSADE 177, LAMSADE, Université Paris-Dauphine, 2001.

[15] L. Engebretsen, M. Karpinski, Approximation hardness of TSP with bounded metrics, in: Proc. ICALP'01, Lecture Notes in Computer Science, vol. 2076, Springer, 2001, pp. 201–212.

[16] M.R. Garey, D.S. Johnson, Computers and Intractability. A Guide to the Theory of NP-completeness, W.H. Freeman, San Francisco, 1979.

[17] M.M. Halldórsson, Approximations of weighted independent set and hereditary subset problems, Journal of Graph Algorithms and Applications 4 (1) (2000) 1–16.

[18] D.S. Hochbaum (Ed.), Approximation Algorithms for NP-hard Problems, PWS, Boston, 1997.

[19] D.S. Johnson, Approximation algorithms for combinatorial problems, Journal of Computer and System Sciences 9 (1974) 256–278.

[20] D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey, R.L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms, SIAM Journal of Computing 3 (4) (1974) 298–325.

[21] M. Karpinski, Polynomial time approximation schemes for some dense instances of NP-hard problems, Algorithmica 30 (2001) 386–397.
[22] E.L. Lawler, Combinatorial Optimization: Networks and Matroids, Holt, Rinehart and Winston, New York, 1976.
[23] L. Lovász, On the ratio of optimal integral and fractional covers, Discrete Mathematics 13 (1975) 383–390.
[24] B. Monien, E. Speckenmeyer, Ramsey numbers and an approximation algorithm for the vertex cover problem, Acta Informatica 22 (1985) 115–123.
[25] C.H. Papadimitriou, K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Prentice Hall, New Jersey, 1981.
[26] C.H. Papadimitriou, S. Vempala, On the approximability of the traveling salesman problem, in: Proc. STOC'00, 2000, pp. 126–133.
[27] C.H. Papadimitriou, M. Yannakakis, Optimization, approximation and complexity classes, Journal of Computer and System Sciences 43 (1991) 425–440.
[28] C.H. Papadimitriou, M. Yannakakis, The traveling salesman problem with distances one and two, Mathematics of Operation Research 18 (1993) 1–11.
[29] R. Raz, S. Safra, A sub-constant error probability low-degree test and a sub-constant error probability PCP characterization of NP, in: Proc. STOC'97, 1997, pp. 475–484.
[30] S. Sahni, T. Gonzalez, P-complete approximation problems, Journal of the Association for Computing Machinery 23 (1976) 555–565.
[31] P. Slavík, A tight analysis of the greedy algorithm for set cover, in: Proc. STOC'96, 1996, pp. 435–441.
[32] V. Vazirani, Approximation Algorithms, Springer, Berlin, 2001.