ELSEVIER Theoretical Computer Science 150 (1995) 1–55

Theoretical Computer Science

Fundamental Study

Approximate solution of NP optimization problems [☆]

G. Ausiello [a,*], P. Crescenzi [b], M. Protasi [c]

[a] *Dipartimento di Informatica e Sistemistica, Università degli Studi di Roma "La Sapienza", Via Salaria 113, 00198, Roma, Italy*

[b] *Dipartimento di Scienze dell'Informazione, Università degli Studi di Roma "La Sapienza", Via Salaria 113, 00198, Roma, Italy*

[c] *Dipartimento di Matematica, Università degli Studi di Roma "Tor Vergata", Via della Ricerca Scientifica, 00133 Roma, Italy*

Received April 1994; revised December 1994
Communicated by M. Nivat

Abstract

This paper presents the main results obtained in the field of approximation algorithms in a unified framework. Most of these results have been revisited in order to emphasize two basic tools useful for characterizing approximation classes, that is, combinatorial properties of problems and approximation preserving reducibilities. In particular, after reviewing the most important combinatorial characterizations of the classes PTAS and FPTAS, we concentrate on the class APX and, as a concluding result, we show that this class coincides with the class of optimization problems which are reducible to the maximum satisfiability problem with respect to a polynomial-time approximation preserving reducibility.

Contents

[☆] Research partially supported by the MURST project *Efficienza di Algoritmi e Progetto di Strutture Informative*, by CNR grant *Strutture Informative e Teoria degli Algoritmi*, and by Esprit Basic Research Action n. 7141 (ALCOM II).

[*] Corresponding author. Email addresses: ausiello@dis.uniroma1.it, piluc@dsi.uniroma1 and protasi@mat.utovrm.it.

0. Introduction

It is well known that for several important optimization problems, such as the traveling salesman problem or the graph coloring problem, determining an optimal solution is extremely time consuming due to the inherent complexity of such problems. In fact, no algorithm running in polynomial time is known for them and, even if a precise characterization of their complexity has not yet been established, for a large class of problems, usually called NP-hard optimization problems, the existence of polynomial-time algorithms would imply a positive answer to the famous question $\mathrm{P} = \mathrm{NP}$, a fact which is generally considered to be extremely unlikely.

For this reason when we have to solve problems of this kind we must restrict ourselves to compute an approximate solution, especially when we have to deal with large instances of the problems. In particular, we are interested in the so-called ε-approximate solutions, i.e., solutions whose relative error with respect to the optimal solution is guaranteed to be bounded by a constant, independently from the size of the instance of the problem [28].

Unfortunately, for many NP-hard optimization problems even to calculate such approximate solutions is computationally hard. Therefore the issue of determining under what conditions and by means of what methods we can design polynomial-time algorithms that provide ε-approximate solutions for NP-hard optimization problems is widely recognized as being very relevant both from the practical point of view and from the point of view of complexity theory.

Currently, the class of problems that allow a polynomial-time ε-approximation algorithm at least for one value of ε is called APX while the class of problems that allow an ε-approximate solution for every value of ε (otherwise called a polynomial-time approximation scheme) is denoted as PTAS. When an approximation scheme exists which is uniformly polynomial both in the instance size and in the inverse of the relative error the problem is said to allow a fully polynomial-time approximation scheme and the class of such problems is called FPTAS.

Since the late 1970s the effort of providing a precise characterization of APX, PTAS, and FPTAS has been extensively tackled [3, 5, 27, 37, 48]. These first results were mainly concentrated on attempts to establish formal relationships between the combinatorial structure of the problems and their approximability properties and to provide necessary and/or sufficient conditions for membership in APX, PTAS, and FPTAS.

Subsequently natural notions of approximation preserving reducibilities have been introduced [20, 42] with the aim of establishing hardness and completeness results in approximation classes and of deriving proofs of intractability of approximation from them.

At the end of the 1980s a new approach to the study of approximable NP-hard optimization problems has been developed [46], based on the characterization of feasible solutions in logical terms and this approach led to the definition of various classes of maximization and minimization problems [35, 43] whose relationships to APX and PTAS were thoroughly analyzed. In particular, two classes of problems defined in [46] have been extensively studied for their numerous interesting properties: the class MAX NP and the class MAX SNP, properly contained in the former, both contained in APX.

More recently an important breakthrough in the theory of approximability of NP-hard optimization problems has been determined by the application of techniques based on interactive protocols [1, 2, 26]. Along this line, on the basis of a new characterization of NP languages, it has been shown that problems such as the maximum clique and the maximum independent set problems cannot be approximated within any ε (unless $P = NP$). At the same time, by means of similar arguments, the fact that several problems, such as the maximum cut and the maximum satisfiability problems, do not allow polynomial-time approximation schemes and, hence, are not contained in PTAS (unless $P = NP$), has been proved.

This paper presents the main results obtained in this research field in a unified framework. Most of these results have been revisited in order to emphasize two basic tools useful for characterizing approximation classes, i.e. combinatorial properties of problems and approximation preserving reducibilities. In particular, after reviewing the most important combinatorial characterizations of the classes PTAS and FPTAS, we concentrate on the class APX and, as a concluding result, we show that this class coincides with the class of optimization problems which are reducible to the maximum satisfiability problem with respect to a polynomial-time approximation preserving reducibility.

The paper is organized as follows. After presenting an introduction to the complexity of NP optimization (NPO) problems and to the classification of NPO problems with respect to approximability, in Section 2 necessary and/or sufficient conditions for the existence of polynomial-time approximation schemes are discussed. In Section 3 an approximation preserving reducibility is defined, the notion of completeness in approximation classes is introduced, and problems with such property are shown. Section 4 is devoted to the paradigmatic satisfiability problem and to its relationships with the syntactic characterization of NPO problems in terms of logical formulae. In particular, an approximation class, called OPT NP, is defined in terms of approximation preserving reducibility with the aim of letting the maximum satisfiability problem play a similar role as satisfiability plays in NP. Successively, a sufficient condition based on the logical definability of optimization problems is given for membership in OPT NP. Finally, in Section 5 the results based on the concept of probabilistically checkable proof and on their negative consequences in terms of approximability for a large class of NPO problems are presented. Moreover, the APX-completeness of the maximum satisfiability problem is presented.

Throughout this paper we assume the readers to be familiar with the basic concepts of the theory of computational complexity as developed in text books such as [8, 14, 28, 44]. Apart from this, we tried to make the paper as self-contained as possible. Moreover, due to the lack of space, we could not present many other interesting results: a good pointer to further literature in this field, before 1990, is the survey by Bruschi, et al. [15]. Finally, we considered out of the scope of the paper the presentation of positive and negative results concerning specific NPO problems and we limited ourselves to refer to paradigmatic problems which are listed in the appendix. In any case, we are confident that these and future results fit well in the framework that we propose so that this paper can represent an appropriate point of departure for following the upcoming literature.

1. NPO problems: definitions and preliminaries

The basic ingredients of an optimization problem are the set of instances or input objects, the set of feasible solutions or output objects associated to any instance, and the measure defined for any feasible solution. On the analogy of the theory of NP-completeness, we are interested in studying a class of optimization problems whose feasible solutions are short and easy-to-recognize. To this aim, suitable constraints have to be introduced. We thus give the following defiition.

Definition 1. An NP *optimization* (NPO) *problem* A is a fourtuple $(I, sol, m, goal)$ such that

1. I is the set of the *instances* of A and it is recognizable in polynomial time.

2. Given an instance x of I, $sol(x)$ denotes the set of *feasible solutions* of x. A polynomial p exists such that, for any x and for any $y \in sol(x)$, $|y| \leqslant p(|x|)$. Moreover, for any x and for any y such that $|y| \leqslant p(|x|)$, it is decidable in polynomial time whether $y \in sol(x)$.

```
begin
  guess $y \in \{0,1\}^{p(|x|)}$;
  if $y$ is a feasible solution of $x$ then output $m(x,y)$
  else abort
end.
```

Algorithm 1. The nondeterministic algorithm of an NPO problem.

3. Given an instance x and a feasible solution y of x, $m(x,y)$ denotes the positive integer measure of y (often also called the value of y). The function m is computable in polynomial time and is also called the *objective function.*
4. goal $\in \{\max, \min\}$.

The *class* NPO is the set of all NPO problems.

The goal of an NPO problem with respect to an instance x is to find an *optimum solution*, i.e. a feasible solution y such that

$$m(x,y) = goal\{m(x,y')\colon\ y' \in sol(x)\}.$$

In the following sol^* will denote the multi-valued function mapping an instance x to the set of optimum solutions, while m^* will denote the function mapping an instance x to the measure of an optimum solution.

Observe that, according to Definition 1, a nondeterministic Turing machine N can be associated to any NPO problem A that, for any instance x of A, performs Algorithm 1.

An NPO problem is said to be *polynomially bounded* if a polynomial q exists such that, for any instance x and for any solution y of x, $m(x,y) \leqslant q(|x|)$.

In the appendix several examples of NPO problems are given which will be used in the following sections.

1.1. Three Problems in One

The definition of an optimization problem naturally leads to three ways of solving the problem itself.

1. *Constructive problem*: given an instance x, derive an optimum solution, i.e., compute an element of $sol^*(x)$.

2. *Evaluation problem*: given an instance x, compute the measure of an optimum solution, that is, compute $m^*(x)$.

3. *Decision problem*: given an instance x and an integer k, decide whether the measure of an optimum solution is greater than (less than, for minimization problems) or equal to k.

Example 1. Let us consider the problem Max Clique. In this case, we can either derive a maximum clique, or compute the size of a maximum clique, or decide whether the size of a maximum clique is at least k.

In the case of NPO problems, the relationship among these three problems is summarized in Fig. 1. For example, since the value of the measure function is bounded by

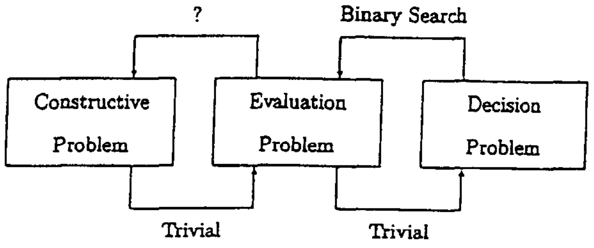

Fig. 1. The relationship among the three problems.

$2^{q(|x|)}$ for a suitable polynomial q, the evaluation problem can be solved by deciding the answer to at most $q(|x|)$ instances of the corresponding decision problem.

As shown in the figure, the relation between computing the measure of an optimum solution and deriving such a solution is not clear. Intuitively, an optimum solution seems to be harder to obtain since it yields additional information.

The next theorem, however, shows that, whenever the decision problem is NP-complete, the constructive and the evaluation problems are computationally equivalent.

Theorem 1 (Crescenzi and Silvestri [21] and Paz and Moran [48]). *Let A be an* NPO *problem whose corresponding decision problem is* NP*-complete. Then the constructive problem associated with A is solvable by a polynomial-time Turing machine with oracle m_A^* (notice that the answer of the oracle to a query x is the value $m_A^*(x)$).*

Proof. Assume A is an NPO maximization problem whose corresponding decision problem is NP-complete (the proof is similar for minimization problems). Since the decision problem associated with A is NP-complete, in order to prove the theorem it suffices to derive an NPO problem B such that the constructive problem associated with A is polynomial-time solvable by a Turing machine with oracle m_B^*.

Such a new problem B is the same as A apart from the measure function m_B which is an "injective version" of m_A. Formally, let p be the polynomial bounding the length of the solutions of A and let $n(y)$ denote the position of solution y in the lexicographic order. Then, for any instance x and for any solution y, we define:

$$m_B(x,y) = 2^{p(|x|)+1} m_A(x,y) + n(y).$$

Clearly, for any instance x and for any two distinct solutions y_1 and y_2 of x, $m_B(x,y_1) \neq m_B(x,y_2)$, i.e., $sol_B^*(x)$ is a singleton. Let $y_B^*(x)$ denote the unique element of $sol_B^*(x)$. Observe that if $m_B(x,y_1) > m_B(x,y_2)$ then $m_A(x,y_1) \geqslant m_A(x,y_2)$, so that $y_B^*(x) \in sol_A^*(x)$. Finally, $y_B^*(x)$ is polynomial-time computable by a Turing machine with oracle m_B^*, since the position of the optimum solution in the lexicographic order can be obtained by computing the remainder of the division between $m_B^*(x)$ and $2^{p(|x|)+1}$. □

It still remains open the question whether an NPO problem exists whose corresponding constructive problem is harder to solve than the corresponding evaluation problem.

The following theorem gives some evidence that the answer to this question may be affirmative.

Theorem 2 (Crescenzi and Silvestri [21]). *If* P $\neq$ NP $\cap$ coNP, *then an* NPO *problem exists such that* $m^* \in FP$ *and it* $sol^* \notin FP$.[1]

Proof. Let $L \in \mathrm{NP} \cap \mathrm{coNP} - \mathrm{P}$. Thus a nondeterministic Turing machine N exists such that, for any x:

1. If $x \in L$, then an accepting computation path of $N(x)$ exists and any computation path either accepts or does not halt.
2. If $x \notin L$, then a rejecting computation path of $N(x)$ exists and any computation path either rejects or does not halt.

Let A be the NP maximization problem such that, for any x, the feasible solutions of x are all halting computation paths of $N(x)$ and they all have measure equal to 1.

It thus follows that $m^*(x) = 1$ for any x so that $m^* \in$ FP. Moreover, if $sol^* \in$ FP, then $x \in L$ can be decided in polynomial time as follows: compute an optimum solution y and accept if and only if y is an accepting computation path. □

1.2. Approximate algorithms and approximation classes

We have already seen that if an NPO problem can be solved in polynomial time, then its corresponding decision problem can also be solved in polynomial time. As a consequence, if P $\neq$ NP, then any NPO problem whose corresponding decision problem is NP-complete is not solvable in polynomial time. In these cases we sacrifice optimality and start looking for approximate solutions computable in polynomial time. Several notions of approximability have been introduced (see, for example, [5, 41]). In this paper, we will use the most widely applied.

Definition 2. Let A be an NPO problem. Given an instance x and a feasible solution y of x, we define:

1. *Relative error of y with respect to x* the ratio

$$E(x,y) = \frac{|m^*(x) - m(x,y)|}{\max\{m^*(x), m(x,y)\}}.$$

2. *Performance ratio of y with respect to x* the ratio

$$R(x,y) = \min\left\{\frac{m(x,y)}{m^*(x)}, \frac{m^*(x)}{m(x,y)}\right\}.$$

[1] The class FP is usually defined as the class of polynomial-time computable single-valued functions. However, we can extend this definition to multi-valued functions as follows: a multi-valued function f belongs to FP if a polynomial-time Turing machine T exists such that, for any input x, $T(x) \in f(x)$.

Of course a strict relationship exists between relative error and performance ratio of a feasible solution. Indeed: $E(x,y) = 1 - R(x,y)$. Both the relative error and the performance ratio range between 0 and 1: the relative error is as close to 1 (i.e., the performance ratio is as close to 0) as the feasible solution is far from the optimum one. In the following we will mainly use the notion of relative error.[2]

Definition 3. Let A be an NPO problem and let T be an algorithm that, for any instance x of A, returns a feasible solution $T(x)$. Given an arbitrary rational $\varepsilon \in (0,1)$, we say that T is an *ε-approximate algorithm for A* if, for any instance x, the relative error of the feasible solution $T(x)$ with respect to x verifies the following inequality:

$$E(x,T(x)) \leqslant \varepsilon$$

(equivalently, T is an ε-approximate algorithm if $R(x,T(x)) \geqslant 1 - \varepsilon$).

For brevity's sake, we will sometimes speak of approximate algorithm instead of ε-approximate algorithm.

Definition 4. An NPO problem A belongs to the class APX if an ε-approximate polynomial-time algorithm T for A exists, for some ε with $0 < \varepsilon < 1$.

Example 2. Let us consider Min Node Cover and let us consider Algorithm 2. Clearly, the subset $V' \subseteq V$ computed by the algorithm is a vertex cover corresponding to a set of disjoint edges whose cardinality is $|V'|/2$ (since for any edge both its endpoints have been added to V'). Since, by definition, any cover must 'touch' all the edges of such set, then it must contain at least $|V'|/2$ nodes. Thus, the cardinality of V' is at most twice the cardinality of a minimum cover, that is, Algorithm 2 $\frac{1}{2}$-approximates Min Node Cover.

Definition 5. Let A be an NPO problem. An algorithm T is said to be an *approximation scheme* for A if, for any instance x of A and for any rational $\varepsilon \in (0,1)$, $T(x,\varepsilon)$ returns a feasible solution whose relative error is at most ε.

Definition 6. An NPO problem A belongs to the class PTAS if it admits a polynomial-time approximation scheme, i.e. an approximation scheme whose time complexity is bounded by $q_\varepsilon(|x|)$, where q_ε is a polynomial.

Observe that the time complexity of an approximation scheme in the above definition may be exponential in the rational ε, i.e. it may be of the type $2^{1/\varepsilon}p(|x|)$ or $|x|^{1/\varepsilon}$, where p is a polynomial. Thus, computations with small ε values may turn out to be practically unfeasible. This leads us to the following definition.

[2] Another measure of approximation that has often appeared in the literature is the inverse of the performance ratio: this quantity ranges between 1 and ∞ and is as close to 1 as the feasible solution is close to the optimum one.

```
begin
  $V' := \emptyset$;
  $E' := E$;
  while $E' \neq \emptyset$ do
  begin
      pick any edge $(u, v) \in E'$;
      $E' := E' - \{(u, v)\}$;
      if $u \notin V' \wedge v \notin V'$ then
      $V' := V' \cup \{u, v\}$;
  end;
end.
```

Algorithm 2. A $\frac{1}{2}$-approximate algorithm for MIN NODE. COVER

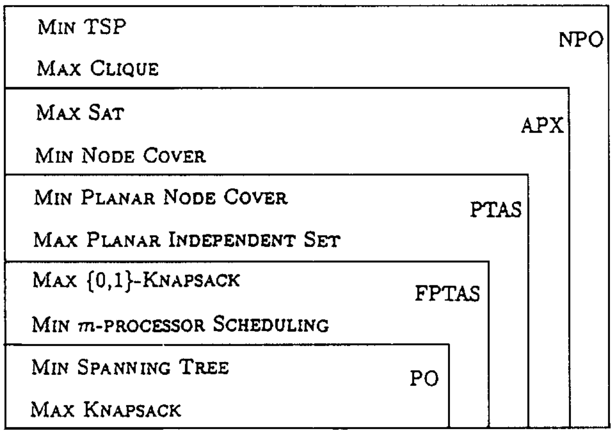

Fig. 2. The NPO world.

Definition 7. An NPO problem A belongs to the class FPTAS if it admits a *fully* polynomial-time approximation scheme, i.e. an approximation scheme whose time complexity is bounded by $q(|x|, 1/\varepsilon)$, where q is a polynomial.

We shall see examples of problems in PTAS and in FPTAS in the next section. Clearly, the following inclusions hold:

$$\text{FPTAS} \subseteq \text{PTAS} \subseteq \text{APX} \subseteq \text{NPO}.$$

It is also easy to see that these inclusions are strict if and only if P $\neq$ NP. In Fig. 2 we present the world of NPO problems with a few well-known examples of members of the above defined approximation classes (in the figure PO denotes the class of NPO problems that are solvable in polynomial time).

2. Necessary and sufficient conditions for PTAS and FPTAS

The investigation of the approximability properties of NPO problems can be carried on in two directions. On the one hand, we try to find approximation algorithms that guarantee good performances with respect to approximability showing that a problem belongs either to APX, to PTAS, or to FPTAS. On the other hand, when a problem does seem difficult to approximate, we are interested in formally finding negative results.

In few cases it is possible to show that a problem does not belong to PTAS using the so-called "gap technique". Let us consider Min Graph Coloring. It is easy to show that no polynomial-time approximation algorithm can exist furnishing a performance ratio greater than $\frac{3}{4}$ unless P = NP. Actually an algorithm with such a performance would be able to color any 3-colorable graph with no more than 3 colors. So we would be able to solve the NP-complete 3-colorability problem in polynomial time.

More generally, the gap technique can be expressed in the following way. Assume that a reduction from an NP-complete problem (say, Sat) to a minimization problem A exists creating a gap in the measure function, i.e. the optimum solution has measure c if the Boolean formula is satisfiable, otherwise it has measure at least $c/(1-g)$ for some gap g $(0 < g < 1)$. Then, unless P = NP, no polynomial-time approximation algorithm can exist with performance ratio greater than $1 - g$. In fact, in case such an algorithm exists, we would be able to solve Sat in polynomial time. In conclusion, we can rule out the existence of a polynomial-time approximation scheme for problem A.

Of course a similar condition holds for maximization problems. Unfortunately, this technique has not been applied to many problems because it is very difficult to find such gaps. In Section 5, we will see how to use this technique in connection with probabilistic checkable proofs.

In this section we present some necessary and sufficient conditions that guarantee that an NPO problem belongs either to the class PTAS or to the class FPTAS. Since finding approximation schemes is a difficult task, we investigate whether it is possible to prove that a problem has a nice approximability performance by using structural properties that do not immediately rely on a computational approach. In particular, we will show that combinatorial properties of the instances of the problems exist that assure the existence of either a polynomial-time approximation scheme or a fully polynomial-time approximation scheme.

2.1. History

The study of general properties that explain the good or bad performance of NPO problems with respect to approximability started at the end of the 1970's and at the beginning of the 1980's.

Korte and Schrader [37] proved that, under very general assumptions, every polynomial-time approximation scheme (and every fully polynomial-time scheme) can be reduced to the same kind of algorithmic procedure so showing that we can design a sort of prototypal algorithm for every problem in PTAS and FPTAS. Further generalizations were shown by Marchetti-Spaccamela and Romano [40].

Another important line of research regarded the study of combinatorial properties that characterize the membership of a problem in PTAS or in FPTAS. In this setting the approach consisted in studying particular subsets of the set of instances of an NPO problem. In one case, a sequence $\{I_j\}$ of subsets of this set was introduced in such a way that, for any j, $I_j \subseteq I_{j+1}$. A set I_j includes any instance whose optimum value is bounded by j. It happens that, for many NPO problems, every I_j is decidable in polynomial time while, for some NPO problems, there are I_j's that are NP-complete. Paz and Moran [48] introduced such a kind of characterization. Garey and Johnson [27], instead, divided the set of instances according to another criterion. Namely, they put some bound on the size of the integers occurring in the input of the problem. The equivalence, under some general conditions, of these two approaches was proved in Ausiello et al. [5].

2.2. *General algorithmic procedures for PTAS and FPTAS*

In this subsection, we present a general algorithmic procedure that, when applicable, guarantees that a problem belongs to PTAS. Moreover, we will see that, for a very large class of problems, every ε-approximate algorithm can be always replaced by such procedure, still preserving the same level of approximability.

Let us start from a paradigmatic problem, that is, MAX $\{0,1\}$-KNAPSACK. Actually, it is well known that this problem belongs to FPTAS. However, we will show a weaker property, namely, that it belongs to PTAS since the approximation scheme we introduce allows to capture the typical technique for proving that a problem is in PTAS. Indeed, after considering this problem, the approximation procedure we will present is just a generalization of the algorithm applied to this example.

An obvious way of solving MAX $\{0,1\}$-KNAPSACK consists in applying a total enumeration algorithm that examines all possible subsets of the instance. In this case we get the optimum solution but, at the same time, the algorithm takes exponential time. On the other hand, if we use a greedy algorithm, we have a very efficient algorithm but unfortunately instances exist for which the approximate solution we find has a value arbitrarily far from the optimum one. This greedy approach with some modifications can be improved but even in this case the approximation ratio is bounded by $\frac{1}{2}$. An approximation scheme for this problem has then been devised by putting together the efficiency of the greedy approach and the correctness of the enumeration technique.

Let us consider an arbitrary instance, that is a finite set $I = \{1,\ldots,n\}$ (for each $i \in I$ we have an integer size a_i and an integer profit p_i) and an integer b, and let J be a subset of I. Let us denote by *greedy*(J) the solution found by the greedy algorithm applied to the subinstance corresponding to $I - J$ with bound $b - \sum_{i\in J} a_i$.

We now define a sequence of algorithms that provides an approximation scheme: the kth approximation algorithm in the sequence works in the following way. We consider all sets J such that $|J| \leqslant k$ and $\sum_{i\in J} a_i \leqslant b$, and, for each of these sets J, we

```
{the feasible solution and its measure are stored in SMAX and PMAX, respectively}
  begin
    order the items by nonincreasing profit densities $p_i/a_i$;
    SMAX := $\emptyset$;
    PMAX := 0;
    for all $J \subseteq \{1,\dots,n\}$ such that $|J| \leqslant k$ and $\sum_{i \in J} a_i \leqslant b$ do
    begin
        $S := J \cup greedy(J)$;
        $P := \sum_{i \in S} p_i$;
        if PMAX < P then
        begin
          SMAX := S;
          PMAX := P
        end
    end
  end.
```

Algorithm 3. A polynomial-time approximation scheme for MAX{0,1}-KNAPSACK.

get a feasible solution by combining J and the solution *greedy*(J). The value of the approximate solution is then found looking for the maximum over all these sets J.

More formally, for any k, let us consider Algorithm 3.

Observe that, since the cycle in the algorithm is repeated $O(kn^k)$ times and the computation of *greedy*(J) requires $O(n)$ time, globally the algorithm has $O(kn^{k+1})$-time complexity.

Theorem 3 (Sahni [49]). MAX {0,1}-KNAPSACK *belongs to PTAS.*

Proof. The theorem is proved by showing that, for any $\varepsilon > 0$, Algorithm 3 with $k = 1/\varepsilon - 1$ is ε-approximate.

Let us assume that the optimum solution is given by $\{i_1,\dots,i_j\}$. If $j \leqslant k$, the algorithm finds the optimum solution because every set of size at most k (and therefore also the optimum solution) is tried. So we make the hypothesis $j > k$.

Assume that the $p_{i_1}, p_{i_2}, \dots, p_{i_k}$ are the k largest profits in the optimum solution and that the remaining $j - k$ items are ordered by nonincreasing profit densities. Thus,

$$p_{i_{k+t}} \leqslant \frac{m^*(x)}{k+1} \quad \text{for } 1 \leqslant t \leqslant j - k. \tag{1}$$

Let us then consider the step in the execution of the algorithm in which $J = \{i_1,\dots,i_k\}$ and let S_J be the corresponding solution. Let i_m be the first index in the optimum solution which is not included in S_J. If no such profit exists we obtain the optimum solution and we are done. Otherwise, we have that S_J must contain s indices

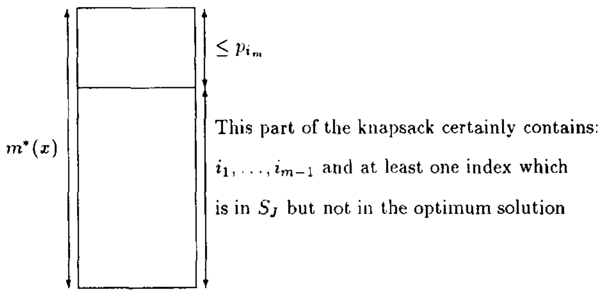
$\leq p_{i_m}$
$m^*(x)$
This part of the knapsack certainly contains:
$i_1, \ldots, i_{m-1}$ and at least one index which
is in S_J but not in the optimum solution

Fig. 3. Comparison among the measures of the approximate and optimum solution.

$\hat{i}_1, \ldots, \hat{i}_s$ with $s \geqslant 1$ that are not in the optimum solution and such that:

$$\frac{p_{\hat{i}_1}}{a_{\hat{i}_1}} \geqslant \cdots \geqslant \frac{p_{\hat{i}_s}}{a_{\hat{i}_s}} \geqslant \frac{p_{i_m}}{a_{i_m}} \geqslant \frac{p_{i_{m+1}}}{a_{i_{m+1}}} \geqslant \cdots \geqslant \frac{p_{i_j}}{a_{i_j}}$$

and

$$\sum_{t=1}^{s} a_{\hat{i}_t} > \sum_{t=m+1}^{j} a_{i_t}.$$

Thus,

$$\sum_{t=m+1}^{j} p_{i_t} \leqslant \frac{p_{\hat{i}_s}}{a_{\hat{i}_s}} \sum_{t=m+1}^{j} a_{i_t} < \frac{p_{\hat{i}_s}}{a_{\hat{i}_s}} \sum_{t=1}^{s} a_{\hat{i}_t} \leqslant \sum_{t=1}^{s} p_{\hat{i}_t}.$$

This, in turn, implies that (see Fig. 3)

$$m^*(x) = \sum_{t=1}^{m} p_{i_t} + \sum_{t=m+1}^{j} p_{i_t} < \sum_{t=1}^{m} p_{i_t} + \sum_{t=1}^{s} p_{\hat{i}_t} = \sum_{t=1}^{m-1} p_{i_t} + \sum_{t=1}^{s} p_{\hat{i}_t} + p_{i_m} \leqslant m(x, S_J) + p_{i_m}.$$

From (1), it finally follows that

$$m^*(x) - m(x, S_J) < \frac{m^*(x)}{k+1},$$

i.e. the relative error of the algorithm is at most $1/(k+1) = \varepsilon$. □

The same approach can be generalized to a large class of problems, thus giving a first necessary and sufficient approximation criterion. In the following, we refer to the family of *maximum independent subset* problems. For each of these problems, an instance is formed by a finite set $I = \{1, \ldots, n\}$ and, for each $i \in I$, an integer profit p_i. For any instance x, the set $sol(x)$ of feasible solutions is an independence system, i.e. $sol(x)$ is a set of subsets of I such that, for any $S \in sol(x)$, every subset of S is also a feasible solution. The goal is to find a feasible solution S which maximizes the value $\sum_{i \in S} p_i$. Observe that, according to the above definition, we are not guaranteed that all members of this family of problems belong to NPO. Indeed, the definition of feasible solution does not assure that we can decide in polynomial time whether a subset of I belongs to $sol(x)$. In the following, however, we restrict ourselves to consider only those problems which are in NPO even though the results can be easily extended to any maximum independent subset problem [37].

Recall now that, in the case of MAX {0,1}-KNAPSACK, we have used a partial enumeration algorithm plus a greedy procedure in order to develop a polynomial-time approximation scheme. The basic idea behind the generalization of this approach is to combine an enumeration technique with a completion procedure. While the enumeration part is similar to what we did for the knapsack problem, the key step consists in finding a general filling-up procedure which is able to guarantee the approximation.

Definition 8. Let x be an instance of a maximum independent subset problem and let S be a feasible solution of x with $|S| = k$. A feasible solution $S' \supseteq S$ is said to be a *k-completion of S* if

$$m(x,S') \geqslant \frac{k-1}{k} m(x,S^*) - \frac{1}{k} m^*(x) - p_{\max}(S,S^*),$$

where $S^* \supseteq S$ is any feasible solution such that

$$m(x,S^*) = \max\{m(x,S') : S \subseteq S' \text{ and } S' \text{ is a feasible solution}\}$$

and $p_{\max}(S,S^*) = \max\{p_i : i \in S^* - S\}$.

A *polynomial-time k-completion algorithm* for a maximum independent subset problem is an algorithm T such that, for any instance x and for any S,

$$T(x,S,k) = \begin{cases} \text{a } k\text{-completion } S' \text{ of } S & \text{if } S \text{ is a feasible solution and } |S| = k, \\ S & \text{if } S \text{ is a feasible solution and } |S| < k \text{ (in this case the } k\text{-completion of } S \text{ is } S \text{ itself)}, \\ \text{undefined} & \text{otherwise (in this case it makes no sense to find a } k\text{-completion)}. \end{cases}$$

Moreover, for any fixed k, the running time of T is polynomial with respect to $|x|$.

Theorem 4 (Korte and Schrader [37]). *A maximum independent subset problem is in* PTAS *if and only if, for any k, it admits a polynomial-time k-completion algorithm.*

Proof. Let A be a maximum independent subset problem that admits a polynomial-time k-completion algorithm T and, for any k, let us consider Algorithm 4.

We now prove that the output of this algorithm is a feasible solution $SMAX$ such that

$$m(x,SMAX) \geqslant \frac{k-3}{k} m^*(x).$$

In this way we get a polynomial-time approximation scheme for A.

Let $S_{\text{opt}} \in sol^*(x)$. Since, in the algorithm, we find the k-completion for every set S with $|S| \leqslant k$, if $|S_{\text{opt}}| \leqslant k$, then we obtain the optimum solution. Therefore, we can

```
{the feasible solution and its measure are stored in SMAX and PMAX, respectively}
begin
  $SMAX := \emptyset$;
  $PMAX := 0$;
  for all feasible solution $S$ with $|S| \leqslant k$ do
  begin
      $S' := T(x, S, k)$;
      if $PMAX < m(x, S')$ then
      begin
          $SMAX := S'$;
          $PMAX := m(x, S')$
      end
  end
end.
```

Algorithm 4. A polynomial-time approximation scheme for the maximum independent subset problem.

restrict ourselves to the case $|S_{\rm opt}| > k$. Among all subsets of $S_{\rm opt}$ of cardinality k, a set S exists such that $p_i \geqslant p_j$ for every $i \in S$ and $j \in S_{\rm opt} - S$. This implies that

$$p_{\max}(S, S_{\rm opt}) \leqslant \frac{m^*(x)}{k+1}. \tag{2}$$

By applying T to x, S, and k, by definition, we obtain

$$m(x, T(x, S, k)) \geqslant \frac{k-1}{k} m(x, S_{\rm opt}) - \frac{1}{k} m^*(x) - p_{\max}(S, S_{\rm opt}).$$

Since $m(x, S_{\rm opt}) = m^*(x)$ and because of (2), we have that

$$m(x, SMAX) \geqslant \frac{k-1}{k} m^*(x) - \frac{1}{k} m^*(x) - \frac{1}{k+1} m^*(x) > \frac{k-3}{k} m^*(x).$$

Conversely, let A be a maximum independent subset problem having a polynomial-time approximation scheme T. By using T, we show how to build a polynomial-time k-completion algorithm T' for A.

Given an instance $x = (I, p_1, \ldots, p_n)$ and given a subset S of I, let us distinguish the following three cases:

1. $|S| > k$: in this case $T'(x, S, k)$ is undefined.
2. $|S| < k$: in this case T' has just to decide whether S is a feasible solution. Since A is in NPO, this can be done in polynomial time.
3. $|S| = k$: in this case, T' first decides whether S is a feasible solution. Successively it has to build a k-completion of S (without loss of generality, we can assume that $k < n - 1$).

In order to do this, we apply the scheme T to x and $\varepsilon = (2k-1)/4k$ obtaining an approximate solution whose measure d verifies the following inequality:

$$m^*(x) \geqslant d \geqslant \frac{2k+1}{4k} m^*(x).$$

Let us define a new instance x' of the problem in which the profits are changed as follows:

$$p_i' = \begin{cases} 2\left\lfloor \dfrac{n-k}{b} d \right\rfloor & \text{if } i \in S, \\ \left\lfloor \dfrac{n-k}{b} p_i \right\rfloor & \text{otherwise,} \end{cases}$$

where $b = \min\{p_i : i \notin S \text{ and } S \cup \{i\} \text{ is a feasible solution}\}$ (observe that if b does not exist then the k-completion of S is S itself).

Again we apply the scheme T to x' and $\varepsilon' = 1/k(2k+1)$ obtaining an approximate solution S' which is the k-completion of S. Indeed, $S \subseteq S'$ since otherwise:

$$\begin{aligned} m(x',S') &\leqslant 2(k-1)\left\lfloor \frac{n-k}{b} d \right\rfloor + \left\lfloor \frac{n-k}{b} m^*(x) \right\rfloor \\ &\leqslant 2(k-1)\left\lfloor \frac{n-k}{b} d \right\rfloor + \left\lfloor \frac{n-k}{b} \frac{4k}{2k+1} d \right\rfloor \\ &\leqslant 2(k-1)\left\lfloor \frac{n-k}{b} d \right\rfloor + \frac{4k}{2k+1}\left\lfloor \frac{n-k}{b} d \right\rfloor + 1 \\ &= (1-\varepsilon')2k\left\lfloor \frac{n-k}{b} d \right\rfloor + 1 \end{aligned}$$

in contradiction to the fact that the polynomial-time approximation scheme guarantees:

$$\begin{aligned} m(x',S') &\geqslant (1-\varepsilon')m^*(x') \geqslant (1-\varepsilon')2k\left\lfloor \frac{n-k}{b} d \right\rfloor + (1-\varepsilon')(n-k) \\ &> (1-\varepsilon')2k\left\lfloor \frac{n-k}{b} d \right\rfloor + 1 \end{aligned}$$

where the last inequality is due to the fact that, for any $k < n-1$, $(1-\varepsilon')(n-k) > 1$.

To prove that

$$m(x,S') \geqslant \frac{k-1}{k} m(x,S^*) - \frac{1}{k} m^*(x) - p_{\max}(S,S^*),$$

we first observe that the following inequality holds:

$$\begin{aligned} m(x',S'-S) &= m(x',S') - m(x',S) \geqslant (1-\varepsilon')m^*(x') - m(x',S) \\ &\geqslant (1-\varepsilon')m(x',S^*) - m(x',S) \\ &= (1-\varepsilon')m(x',S^*-S) + (1-\varepsilon')m(x',S) - m(x',S) \\ &= (1-\varepsilon')m(x',S^*-S) - \varepsilon' m(x',S). \end{aligned}$$

We then have

$$\begin{aligned}
m(x,S'-S) &\geqslant \frac{b}{n-k}\sum_{i\in S'-S}\left\lfloor \frac{n-k}{b}p_i \right\rfloor = \frac{b}{n-k}m(x',S'-S)\\
&\geqslant \frac{b}{n-k}(1-\varepsilon')m(x',S^*-S) - \frac{b}{n-k}\varepsilon' m(x',S)\\
&\geqslant \frac{b}{n-k}(1-\varepsilon')m(x',S^*-S) - \frac{b}{n-k}\varepsilon' 2k\frac{n-k}{b}d\\
&= \frac{b}{n-k}(1-\varepsilon')m(x',S^*-S) - \varepsilon' 2kd\\
&\geqslant (1-\varepsilon')m(x,S^*-S) - (1-\varepsilon')b - \varepsilon' 2kd\\
&\geqslant \frac{k-1}{k}m(x,S^*-S) - b - \frac{2}{2k+1}d\\
&\geqslant \frac{k-1}{k}m(x,S^*-S) - \frac{1}{k}m^*(x) - p_{\max}(S,S^*)
\end{aligned}$$

that implies the desired inequality.

From the above three cases we have that T' is a polynomial-time k-completion algorithm for A and the theorem thus follows. □

A similar approach for providing a characterization for the class FPTAS can be applied, again generalizing the properties that show that Max $\{0,1\}$-Knapsack has a fully polynomial-time approximation scheme. In this case the algorithm for this latter problem is based on a dynamic programming approach together with a dominance rule. The dominance rule is exploited to get a better solution, once a feasible one has been achieved. In this general setting we are able to find a necessary and sufficient condition for a problem being in FPTAS introducing the so-called ε-dominance test. It is possible to show that a maximum independent subset problem admits a fully polynomial-time approximation scheme if and only if an ε-dominance test exists running in time bounded by a polynomial in the lenght of the input and $1/\varepsilon$.

We have presented necessary and sufficient conditions for maximum independent subset problems. Actually the results still hold for a larger set of problems. In fact, instead of trying to maximize the sum of the profits, we can consider instances in which we are interested in maximizing the product of the profits. Also in this case the theorems still hold [40]. This generalization allows, for instance, to show that the product version of Max $\{0,1\}$-Knapsack belongs to FPTAS.

2.3. Simplicity and approximation

Another important approach tries to characterize the problems in PTAS and FPTAS from a combinatorial point of view. In this case, the aim is to find structural properties that assure the approximability of a problem not in terms of the existence of an algorithmic procedure but exploiting the combinatorial and computational aspects of several decision problems associated to an optimization problem. The approach introduced by Paz and Moran [48] is based on the idea of considering the complexity of subsets of instances of the problems obtained by putting some bounds on the objective function. Once we consider suitable subsets, it is possible to divide the class of NPO problems in various subclasses that share different approximability properties.

We start by defining a general and abstract way of dividing all the instances of an NPO problem in subsets.

Definition 9. Let A be an NPO problem and g be an arbitrary function. The set A_g is defined as follows:

$$A_g = \{x : m^*(x) \leqslant g(|x|)\}.$$

By applying this general definition to particular functions g it is possible to introduce important distinctions inside the class of NPO problems.

Definition 10. An NPO problem A is *simple* if, for every positive integer k, A_k is decidable in polynomial time.

Example 3. An example of simple problem is Max Clique. In order to decide if the optimum value is at most k, it is sufficient to consider all the sets of $k+1$ nodes and to verify whether at least one of them is complete. This can be done in time $O(n^{k+1})$, where n is the number of nodes.

Instead Min Graph Coloring is an example of problem that is not simple. If we consider the set A_3, we obtain an NP-complete problem so contradicting the definition of simplicity (if $P \neq NP$).

The notion of simple problem introduces an interesting subdivision inside the class of NPO problems. From a combinatorial point of view we can consider a simple problem "easier" to solve than another one which is not. The concept of simple problem can be immediately related to the approximability properties.

Theorem 5 (Paz and Moran [48]). *If an NPO problem belongs to the class* PTAS, *then it is simple.*

Proof. Let A be an NP maximization problem in PTAS (the other case is similar). By definition, a polynomial-time approximation scheme T exists such that, for any instance x and for any integer k,

$$E(x, m(x,y)) \leqslant \frac{1}{k+2},$$

where $y = T(x, 1/(k+2))$.

Since A is a maximization problem, $m(x,y) > k$ implies $m^*(x) > k$. On the other hand, from the above inequality it follows that

$$\frac{m(x,y)}{m^*(x)} \geqslant \frac{k+1}{k+2}$$

and, for $m(x,y) \leqslant k$, this inequality is possible only if $m^*(x) \leqslant k$. It follows that $m^*(x) \leqslant k$ if and only if $m(x,y) \leqslant k$. In other words, the set A_k is polynomial-time decidable. □

We note that simplicity is a necessary but not sufficient condition for a problem to be in PTAS. For instance, the problem Max Clique is simple but not only it does not belongs to PTAS but it has a very bad approximation behavior (see the last section). We can conclude that the definition of simplicity is shared by a very large (too large!) subclass of NPO problems. On the other hand, the concept of simplicity is quite interesting because together with another condition it allows to characterize the class PTAS in a complete way.

Definition 11. An NPO problem A satisfies the *boundedness condition* if an algorithm T_b and a positive integer constant E exist such that the following hold:

1. For every instance x of A and for every positive integer c, $T_b(x,c)$ is a solution y of x such that

$$m^*(x) \leqslant m(x,y) + cE \quad \text{if } A \text{ is a maximization problem,}$$
$$m^*(x) \geqslant m(x,y) - cE \quad \text{otherwise.}$$

2. The time complexity of $T_b(x,c)$ is a polynomial in $|x|$ whose degree depends only on the value $m(x, T_b(x,c))/c$.

Theorem 6 (Paz and Moran [48]). *An* NPO *problem A admits a* PTAS *if and only if it is simple and satisfies the boundedness condition.*

Proof. The proof will be given for maximization problems and it is similar in the case of minimization ones.

Let A be a problem that admits a polynomial-time approximation scheme T. From the proof of Theorem 5 it follows that A is simple. We now prove the boundedness condition. For any instance x of A, let $t = m(x, T(x, \frac{1}{2}))$. This means that

$$t \leqslant m^*(x) \leqslant 2t.$$

For any integer c, if $c \geqslant t/2$ then we define $T_b(x,c) = T(x, \frac{1}{2})$, otherwise we define $T_b(x,c) = T(x, \varepsilon/(1+\varepsilon))$ where $\varepsilon = c/t$. In the first case, we have that

$$m^*(x) \leqslant 2t \leqslant m(x, T_b(x,c)) + 2c,$$

while, in the second case,

$$m^*(x) \leqslant m(x, T_b(x,c))(1+\varepsilon) = m(x, T_b(x,c)) + c\frac{m(x, T_b(x,c))}{t}$$
$$\leqslant m(x, T_b(x,c)) + 2c\frac{m(x, T_b(x,c))}{m^*(x)} \leqslant m(x, T_b(x,c)) + 2c.$$

By choosing $E = 2$, in both cases we then have

$$m^*(x) \leqslant m(x, T_b(x,c)) + cE.$$

For what concerns the time complexity of $T_b(x,c)$, observe that, in the first case, this complexity is a polynomial in $|x|$ with constant degree, while, in the second case, it is a polynomial in $|x|$ whose degree may depend on

$$\frac{1+\varepsilon}{\varepsilon} = \frac{t+c}{c} < \frac{3}{2}\frac{t}{c} < \frac{9}{4}\frac{m(x, T_b(x,c))}{c}$$

where the last inequality is due to the fact that $m(x, T_b(x,c)) > \frac{2}{3}t$.

Conversely, let A be a simple NPO problem satisfying the boundedness condition. Given any input x and any $\varepsilon > 0$, we will find an ε-approximate solution for x. This solution will be achieved in time polynomial in $|x|$ and it will be equal to $T_b(x, 2^k)$ for a suitable k.

First of all we note that, because of the boundedness condition, a suitable constant E exists such that

$$m^*(x) \leqslant m(x, T_b(x, 2^k)) + 2^k E$$

for any integer k. This implies that

$$\frac{m^*(x) - m(x, T_b(x, 2^k))}{m^*(x)} \leqslant \frac{m^*(x) - m(x, T_b(x, 2^k))}{m(x, T_b(x, 2^k))} \leqslant \frac{2^k E}{m(x, T_b(x, 2^k))}.$$

Therefore, it suffices to find a value of k such that

$$\frac{m(x, T_b(x, 2^k))}{2^k} \geqslant \frac{E}{\varepsilon} > 1.$$

Since the problem is simple, we can decide whether $m^*(x) \leqslant 2E/\varepsilon$ in time bounded by a polynomial in $|x|$. If this is the case, we can exactly compute $m^*(x)$ again in time bounded by a polynomial in $|x|$ and we are done. Therefore, in the following, we assume that $m^*(x) > 2E/\varepsilon$.

Since $m^*(x) < 2^{r(|x|)}$ for some polynomial r, we have that

$$\frac{m(x, T_b(x, 2^{r(|x|)}))}{2^{r(|x|)}} \leqslant \frac{m^*(x)}{2^{r(|x|)}} < 1.$$

On the other hand, by exploiting the boundedness condition and the assumption on $m^*(x)$, we obtain $m(x, T_b(x, 1)) > E/\varepsilon$ (note that computing $T_b(x, 1)$ may require a time greater than a polynomial in $|x|$). Taking into account these two conditions we can deduce that an integer k exists with $0 \leqslant k < r(|x|)$ such that:

$$\frac{m(x, T_b(x, 2^{k+1}))}{2^{k+1}} \leqslant \frac{E}{\varepsilon} \leqslant \frac{m(x, T_b(x, 2^k))}{2^k}.$$

Such an integer k can be found in the following way: for each $i = r(|x|), \ldots, 1$, we compute $T_b(x, 2^i)$ and we verify whether $E/\varepsilon \leqslant m(x, T_b(x, 2^i))/2^i$. The first time this happens we can stop. In order to verify that the time complexity of the procedure is polynomial in $|x|$, it suffices to observe that, for any integer k satisfying the above condition, the following inequalities hold:

$$\frac{m(x, T_b(x, 2^k))}{2^k} \leqslant 2\frac{m^*(x)}{2^{k+1}} \leqslant 2\left(\frac{m(x, T_b(x, 2^{k+1}))}{2^{k+1}} + E\right) \leqslant 2(1+\varepsilon)\frac{E}{\varepsilon}$$

and since $\varepsilon < 1$, $m(x, T_b(x, 2^k))/2^k \leqslant 4E/\varepsilon$.

From the second property of the boundedness condition it thus follows that the time complexity of the procedure is polynomial in $|x|$ and this concludes the proof of the theorem. □

Using a similar approach we can find a necessary and sufficient condition for a problem to admit a fully polynomial-time approximation scheme. In this case, we modify the two conditions on simplicity and boundedness. For what concerns the first definition, we use the concept of *p-simplicity* in which, for every positive integer k, A_k has to be decidable in time bounded by a polynomial in the lenght of the instance and in k. For what regards the boundedness condition the constant E is substituted by a suitable polynomial, thus obtaining a *polynomial boundedness condition*.

The study of structural properties that characterize the approximability of NPO problems is worthwhile for two reasons. First of all, we are interested in discovering what kind of combinatorial structure influences approximation and computational properties of this class of problems. Secondly, since finding and exhibiting a polynomial-time approximation scheme or proving that a problem does not belong to PTAS are difficult tasks, a combinatorial characterization could be very useful in answering such questions. In fact, instead of exhibiting a polynomial-time approximation scheme, we could produce the equivalent conditions of simplicity and boundedness. So we could hope, in such a way, to enlarge the list of problem which are well-approximable or, from a negative point of view, to add new items to the list of problems which are not efficiently approximable. Unfortunately, Theorem 6 is not very useful in achieving such an aim. In fact, generally speaking, proving that an NPO problem satisfies the conditions of the theorem does not appear easier than directly proving that the problem either has or has not a polynomial-time approximation scheme. In particular, to find an algorithm T_b that satisfies the boundedness condition, is rather similar to find an algorithmic scheme that ensures the good approximability of the problem.

2.4. Strong NP-completeness and pseudo-polynomiality

The considerations developed at the end of the last section push to find other results, still based on the combinatorial structure of the problems, but more directly linked to the problem itself. To reach this aim, we present another approach due to Garey and Johnson [27] that exploits some properties of the inputs in a direct way and surely permits a more intuitive treatment. However, in order to have simpler characterizations, a price is paid: in fact, the new approach is less powerful and we will not be able to provide necessary and sufficient conditions for the approximability of NPO problems but only partial answers.

In this case we start from another way of considering subsets of instances of a problem. In particular, we are now interested in studying the different ways in which numbers play a role in an NPO problem.

Let us consider, for example, Max Cut. This problem is NP-complete and remains NP-complete even if we restrict the instances to graphs having unitary weights. This fact suggests that Max Cut is computationally hard independently on the size of the numbers which occur in the instances. Therefore we have a strong hint that it is the combinatorial structure, i.e. the property that the graph has to satisfy, that makes the problem hard.

On the other hand, if we consider other problems the situation is rather different. Let us consider, again, Max $\{0,1\}$-Knapsack. Using a dynamic programming algorithm, we can solve this problem in time $O(n^2b)$ but b is an integer contained in the instance and thus this algorithm is not a polynomial-time one. In fact the complexity of the algorithm has to be evaluated with respect to the size of the instance. The integers a_i and p_i in the input are codified with $O(\log a_i)$ and $O(\log p_i)$ bits, respectively, and the same holds for b. This means that the size of the instance is $O(n \log b)$ and this implies that no polynomial function of this quantity exists bounding $O(n^2b)$. Equivalently, we can say that the algorithm is not polynomial-time computable because the bound b can assume values which are exponential in the size of the instance. However, if we restrict ourselves to instances in which the numbers a_i, p_i, and b have values bounded by a polynomial in the length of the instance, we obtain a polynomial-time algorithm. In this case, therefore, the fact that Max $\{0,1\}$-Knapsack is NP-complete is strongly related to the presence of very large numbers in the input.

These considerations can be generalized and formalized in the following way.

Definition 12. For any NPO problem A and for any instance x of A, $max(x)$ denotes the value of the largest number occurring in x.

Example 4. In the case of Max Cut, given a weighted graph G, $max(G)$ is the value of the maximum edge weight. Instead, for Max $\{0,1\}$-Knapsack, given an instance x, $max(x) = \max\{a_1, \ldots, a_n, p_1, \ldots, p_n, b\}$.

We note that, from a formal point of view, the function *max* depends on the encoding of the instance. However, we can repeat for the function *max* the same kind of considerations that are usually made when considering the computational complexity of a problem assuming the length of the instance as the main parameter. In fact, if we choose two different functions *max* and *max'* for the same problem, the results we are going to present do not change in the case that these two functions are polynomially related, i.e. two polynomials p and q exist such that, for any instance x, both $max(x) \leqslant p(max'(x))$ and $max'(x) \leqslant q(max(x))$ hold.

For the NPO problems we are interested in, all the intuitive *max* functions we can think of are polynomially related. Thus, the concept of *max* is sufficiently flexible to be used in practice without any limitation.

Definition 13. An NPO problem A is *pseudo-polynomial* if it can be solved by a pseudo-polynomial algorithm, i.e. an algorithm that, on any instance x, runs in time bounded by a polynomial in $|x|$ and in $max(x)$.

Example 5. The dynamic programming algorithm for MAX {0,1}-KNAPSACK proves that this problem is pseudo-polynomial.

Definition 14. Let A be an NPO problem and let $A^{max,p}$ denote the problem obtained by restricting A to only those instances x for which $max(x) \leqslant p(|x|)$ where p is a polynomial. A is said to be *strongly* NP-*hard* if a polynomial p exists such that the decision problem associated with $A^{max,p}$ is NP-complete.

Example 6. MAX CUT is an example of strongly NP-hard problem. Indeed, it is sufficient to consider the polynomial $p(n) = 1$.

Theorem 7 (Garey and Johnson [27]). *A strongly* NP-*hard problem cannot be pseudo-polynomial unless* P = NP.

Proof. Let us assume that A is a pseudo-polynomial problem. This means that an algorithm exists that solves A in time $q(|x|, max(x))$ for a suitable polynomial q. Then, for any polynomial p, $A^{max,p}$ could be solved in time $q(|x|, p(|x|))$ and therefore in polynomial time contrary to the hypothesis that a polynomial p exists such that the decision problem associated with $A^{max,p}$ is NP-complete. □

Example 7. Unless P = NP, MAX CUT is not solvable in pseudo-polynomial time.

It is possible to prove an interesting relationship between the concepts of strong NP-hardness and full approximability. In order to do this, we first need to prove the following result.

Theorem 8 (Garey and Johnson [27] and Papadimitriou and Steiglitz [45]). *Let A be an* NPO *problem in* FPTAS. *If a polynomial p exists such that, for every input x, $m^*(x) \leqslant p(|x|, max(x))$, then A is a pseudo-polynomial problem.*

Proof. Let T be a fully polynomial-time scheme for A. We shall exhibit a pseudo-polynomial algorithm T' that solves A. This algorithm is simply defined as

$$T'(x) = T\left(x, \frac{1}{p(|x|, max(x)) + 1}\right).$$

Since the optimum measure is bounded by $p(|x|, max(x))$, only an optimum solution of x can be approximate, and hence T' solves A. To show that T' is pseudo-polynomial, recall that T operates within time $O(q(|x|, 1/\varepsilon))$ for some polynomial q. Therefore, T' operates in time $O(q(|x|, p(|x|, max(x)) + 1))$ that is a polynomial in both $|x|$ and $max(x)$. □

Corollary 1 (Garey and Johnson [27]). *Let A be a strongly* NP-*hard problem that admits a polynomial p such that $m^*(x) \leqslant p(|x|, max(x))$ for every input x. Then A does not belong to* FPTAS *unless* P = NP.

Proof. The proof derives from Theorems 7 and 8. □

We can immediately derive an interesting consequence from this result. Since a polynomially bounded NPO problem verifies the hypothesis of the theorem, we have that such a kind of problem does not belong to FPTAS.

The concepts of pseudo-polynomiality and strong NP-hardness allow to classify NPO problems in different classes. Once we have shown that an NPO problem is pseudo-polynomial, we can think that it is computationally easier than a problem that is strongly NP-hard. On the other hand, Corollary 1 allows to capture some connections of these concepts with the approximability properties. Also from this point of view, even if we only have partial relationships, it is clear that pseudo-polynomiality is linked to well-approximable problems while strong NP-hardness seems to be one of the characteristics of problems which have a bad behaviour with respect to approximability.

Considering the global framework studied until now in this section, it is quite natural to try to compare all the introduced concepts. From an intuitive point of view the notions of simplicity and p-simplicity on the one hand and the notions of pseudo-polynomiality and strong NP-hardness on the other appear to be possibly related because in both approaches there is the common idea of bounding subsets of instances of a problem. Actually strict relationships between the two different approaches can be formally proved.

Theorem 9 (Ausiello et al. [5]). *Given an* NPO *problem A whose corresponding decision problem is* NP-*complete, if m^* and max are polynomially related, then A is p-simple if and only if A is a pseudo-polynomial problem.*

Proof. Let A be a pseudo-polynomial problem and let q be a polynomial such that, for every instance x, $max(x) \leqslant q(m^*(x))$. Since A is pseudo-polynomial, $m^*(x)$ is computable within time $p(|x|, max(x)) \leqslant p(|x|, q(m^*(x)))$, where p is a polynomial. Given an integer k, to decide in polynomial time whether an instance x belongs to A_k we apply the pseudo-polynomial algorithm for $p(|x|, q(k))$ steps. If the algorithm halts, then we have computed $m^*(x)$ and we thus can decide whether $x \in A_k$. Otherwise, we can conclude that $m^*(x) > k$, i.e. $x \notin A_k$.

Conversely, let A be a p-simple problem and let q be a polynomial such that, for every instance x, $m^*(x) \leqslant q(max(x))$. For every x and for every k, since A is p-simple we can decide whether $x \in A_k$ in time $p(|x|, k)$, where p is a polynomial. In order to compute $m^*(x)$ we can then apply a binary search technique between the values 1 and $q(max(x))$ requiring at most $\log(q(max(x)))p(|x|, q(max(x)))$ time. Since the decision problem corresponding to A is NP-complete, the pseudo-polynomiality of A follows from Theorem 1. □

The above theorem guarantees that whenever there are polynomial relationships between m^* and max, the two approaches coincide. Since this happens for most problems (for instance, Max {0,1}-Knapsack), p-simplicity and pseudo-polynomiality can be indifferently used in various settings. However, there are problems for which the hypothesis of the theorem are not satisfied.

Example 8. Let us consider a particular version of Max {0,1}-Knapsack in which the constraint b has to be exactly reached. It is possible to prove that this problem is pseudo-polynomial but not p-simple (unless P = NP). We also note that there are instances for which max and m^* are not polynomially related. Since this problem is not p-simple we can derive that it is not fully approximable, while this property cannot directly be inferred using the concept of pseudo-polynomiality.

The concept of strong NP-hardness can also be compared with a different kind of simplicity, as shown in [5].

Finally we note that, for maximum independent subset problems, the inheritance and the existence of a k-completion algorithm are equivalent to the properties of simplicity and boundedness because in both cases we have a necessary and sufficient condition to be in the class PTAS. A similar consideration also holds for the class FPTAS, i.e. the inheritance and the existence of an ε-dominance test are equivalent to the conditions of p-simplicity and polynomial boundedness.

3. Completeness in approximation classes

The major open problem in the theory of approximation complexity is whether the inclusions among the four classes NPO, APX, PTAS, and FPTAS are strict. We have already observed that this is the case if and only if P $\neq$ NP so that answering this question seems to be very hard.

Parallelizing the development of the theory of NP-completeness, we can then look for the "hardest" problems in the above classes, i.e. problems which cannot have stronger approximation properties unless P = NP.

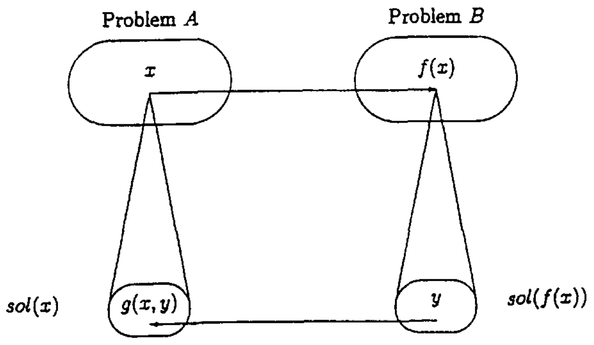

Fig. 4. Reducibility between optimization problems.

In complexity theory, saying that a problem is the hardest one in a class is equivalent to saying that it is complete for that class with respect to a suitable reducibility. Clearly, the many-to-one polynomial-time reducibility is inadequate to study the approximability properties of optimization problems. Indeed, if we want to map an optimization problem A into an optimization problem B then we need not only a function mapping instances of A into instances of B but also a function mapping back solutions of B into solutions of A (see Fig. 4).

In this section we define a natural approximation preserving reducibility and show the existence of complete problems both in NPO and in APX.

3.1. History

In order to derive lower-bounds concerning the approximability of NPO problems, various concepts of completeness have been introduced (see, for example, [3,4,11, 38,42,46,48,50]) In particular, in Orponen and Mannila several problems are shown to be complete with respect to a given kind of approximation preserving reducibility. In this section we mainly present the results obtained in Crescenzi and Panconesi [20] and in Crescenzi and Trevisan [24] along this line of research.

3.2. Approximation preserving reducibilities

In order to introduce the notion of completeness for our classes we define the following reducibility between optimization problems which serves two purposes: it preserves membership both in APX and in PTAS.

Definition 15. Let A and B be two NPO problems. A is said to be PTAS-reducible to B, in symbols $A \leqslant_{\mathrm{PTAS}} B$, if three functions f, g, and c exist such that:
1. For any $x \in I_A$, $f(x) \in I_B$ is computable in polynomial time.

2. For any $x \in I_A$, for any $y \in sol_B(f(x))$, and for any $\varepsilon \in (0,1)_Q$,[3] $g(x,y,\varepsilon) \in sol_A(x)$ is computable in time polynomial with respect to both $|x|$ and $|y|$.
3. $c: (0,1)_Q \to (0,1)_Q$ is computable and surjective.
4. For any $x \in I_A$, for any $y \in sol_B(f(x))$, and for any $\varepsilon \in (0,1)_Q$,

$$E_B(f(x),y) \leqslant c(\varepsilon) \text{ implies } E_A(x, g(x,y,\varepsilon)) \leqslant \varepsilon.$$

The triple (f,g,c) is said to be a PTAS-reduction from A to B.

Remark 1. The previous definition is a slight but relevant modification of the definition of PTAS-reducibility given in [20]. Such modification essentially consists in introducing the dependency of g on ε. Even though such dependency has been seldom used in the literature, it turns out to be necessary for proving completeness results that will be discussed in the following.

It is easy to show that the previous definition satisfies the following fact.

Proposition 1. *The* PTAS-*reducibility is reflexive and transitive.*

The following two results show that the PTAS-reducibility indeed preserves membership both in APX and in PTAS and thus allows us to define the notion of both NPO-completeness and APX-completeness.

Proposition 2. *If* $A \leqslant_{\mathrm{PTAS}} B$ *and* $B \in$ APX, *then* $A \in$ APX.

Proof. Let T_B be a δ-approximate algorithm for B and let (f,g,c) be a PTAS-reduction from A to B. Since c is surjective, an ε exists such that $c(\varepsilon) = \delta$. Then

$$T_A(x) = g(x, T_B(f(x)), \varepsilon),$$

is an ε-approximate algorithm for A. □

Proposition 3. *If* $A \leqslant_{\mathrm{PTAS}} B$ *and* $B \in$ PTAS, *then* $A \in$ PTAS.

Proof. Let T_B be a polynomial-time approximation scheme for B and let (f,g,c) be a PTAS-reduction from A to B. Then

$$T_A(x,\varepsilon) = g(x, T_B(f(x), c(\varepsilon)), \varepsilon)$$

is a polynomial-time approximation scheme for A. □

Definition 16. A problem $A \in$ NPO (respectively, $A \in$ APX) is NPO-*complete* (respectively, APX-*complete*) if, for any $B \in$ NPO (respectively, $B \in APX$), $B \leqslant_{\mathrm{PTAS}} A$.

[3] $(0,1)_Q$ denotes the set of positive rational numbers smaller than 1.

Remark 2. Papadimitriou and Yannakakis [46] defined a different kind of reducibility between optimization problems which is similar to the PTAS-reducibility and is called L-reducibility. Indeed, an L-reduction turns out to be essentially a PTAS-reduction with $c(\varepsilon) = \varepsilon/\alpha\beta$, where α and β are constants. Observe that in this case the function c is not surjective so that the L-reducibility is not assured to preserve membership in APX. However, whenever A is L-reducible to B and $A \in$ APX, it is possible to prove that $A \leqslant_{\mathrm{PTAS}} B$ [23]. More recently, Khanna et al. [34], proposed a variant of the L-reducibility, called E-reducibility. An E-reduction turns out to be a PTAS-reduction with $c(\varepsilon) = \varepsilon/[\varepsilon + \beta(1-\varepsilon)]$, where β is a constant. In this case the function c is surjective.

In order to simplify the proof of most of the completeness results of the following sections, we will make use of a technical lemma that has been more or less explicitly hinted in [34]. Indeed, this lemma formally states that an E-reduction is a PTAS-reduction.

Lemma 1. *Let A and B be two* NPO *problems, let x_A (respectively, x_B) be an instance of A, and let y_A (respectively, y_B) be a solution of x_A (respectively, x_B). If, for any rational $\varepsilon > 0$,*

$$d_B(x_B, y_B) = \frac{E_B(x_B, y_B)}{1 - E_B(x_B, y_B)} = \frac{|m_B^*(x_B) - m_B(x_B, y_B)|}{\min\{m_B^*(x_B), m_B(x_B, y_B)\}} \leqslant \varepsilon$$

implies

$$d_A(x_A, y_A) = \frac{E_A(x_A, y_A)}{1 - E_A(x_A, y_A)} = \frac{|m_A^*(x_A) - m_A(x_A, y_A)|}{\min\{m_A^*(x_A), m_A(x_A, y_A)\}} \leqslant \beta\varepsilon,$$

where β is a constant, then, for any $\varepsilon \in (0,1)_Q$,

$$E_B(x_B, y_B) \leqslant c(\varepsilon) \textit{ implies } E_A(x_A, y_A) \leqslant \varepsilon.$$

where $c(\varepsilon) = \varepsilon/[\varepsilon + \beta(1-\varepsilon)]$ and hence is surjective.

Proof. Given $\varepsilon \in (0,1)_Q$, let us assume that $E_B(x_B, y_B) \leqslant \varepsilon/[\varepsilon + \beta(1-\varepsilon)]$. Then

$$d_B(x_B, y_B) \leqslant \frac{\varepsilon}{\beta(1-\varepsilon)}.$$

From the hypothesis of the lemma, it follows that

$$d_A(x_A, y_A) \leqslant \frac{\varepsilon}{1-\varepsilon},$$

which in turn implies that $E_A(x_A, y_A) \leqslant \varepsilon$. □

Remark 3. Observe that the function d defined in the previous lemma measures the distance of the inverse of the performance ratio from 1. Formally, for any instance x and for any solution y of x, $d(x, y) = 1/R(x, y) - 1$.

Remark 4. In [20] Crescenzi and Panconesi define also a reducibility for the class PTAS, called FPTAS-reducibility. In the following, however, we preferred to focus our attention on the class APX whose properties have been, until recently, less "understood" than those of the classes PTAS and FPTAS (see the previous section).

3.3. NPO-completeness

The aim of this section is to show completeness results in the class NPO. In the following MAX NPO and MIN NPO will denote the class of NPO maximization problems and the class of NPO minimization problems, respectively.

Theorem 10 (Ausiello et al. [4] and Orponen and Mannila [42]). MAX WEIGHTED SAT *is* MAX NPO-*complete.*

Proof. Let A be a maximization problem in NPO and let us consider the corresponding nondeterministic Turing machine N (see Algorithm 1). From Cook's Theorem we can derive, for any x, a Boolean formula φ_x whose satisfying truth-assignments are in a one-to-one correspondence with the halting computation paths of $N(x)$. Let $y_1, \ldots, y_r$ be the Boolean variables describing the solution y and let $m_1, \ldots, m_s$ be the Boolean variables which correspond to the tape cells on which N prints the value $m_A(x, y)$. We then assign a zero weight to all variables excluding the m_i's which instead receive the weight 2^{s-i}.

For any truth-assignment which satisfies φ_x, we then recover a solution y for A by simply looking at the values of y_i's variables. It is clear that $m_A(x, y)$ is exactly equal to the sum of the weights of the true variables. Hence we have proved that $A \leqslant_{\text{PTAS}}$ MAX WEIGHTED SAT with $c(\varepsilon) = \varepsilon$ invertible. □

In a similar way, we can prove the following result.

Theorem 11. MIN WEIGHTED SAT *is* MIN NPO-*complete.*

In Orponen and Mannila [42] other MIN NPO-completeness results are proved: for instance, both the traveling salesman problem and the $\{0, 1\}$-integer programming problem are shown to be MIN NPO-complete. Recently, Crescenzi et al. [19] proved that any NPO minimization (respectively, maximization) problem can be PTAS-reduced to any MAX NPO-complete (respectively, MIN NPO-complete) problem. This result implies that both MAX WEIGHTED SAT and MIN WEIGHTED SAT are, indeed, NPO-complete.

It is also unknown whether a polynomially bounded optimization problem exists which is complete for NPO (indeed, a result of Crescenzi et al. [19] gives strong evidence that this is not the case). However, Berman and Schnitger [11] proved the existence of complete problems for the class of polynomially bounded maximization problems and several problems are now known to be complete for this class [30, 31].

```
begin
  guess $y \in \{0,1\}^{p(|x|)}$;
  if $y$ is a feasible solution of $x$ such that $m_A(x,y) \geqslant t(x)$ then
    output $m_A(x,y)$
  else abort
end.
```

Algorithm 5. The nondeterministic algorithm of an APX problem.

Successively, Kann [32] proved analogous results for the class of polynomially bounded minimization problems. In Crescenzi et al. [19], finally, it has been shown that all these problems are indeed complete for the whole class of polynomially bounded optimization problems.

3.4. APX-completeness

Let us consider Max Bounded Weighted Sat which is a variant of Max Weighted Sat such that the weights of the variables satisfy the following constraint:

$$W \leqslant \sum_{i=1}^{n} w_i \leqslant 2W,$$

where W is an integer given in input, any truth assignment is a feasible solution, and the measure function is modified as follows:

$$m_{\mathrm{MBWS}}(\varphi, \tau) = \begin{cases} \max(W, \sum_{i=1}^{n} w_i \tau(x_i)) & \text{if } \tau \text{ satisfies } \varphi, \\ W & \text{otherwise.} \end{cases}$$

Theorem 12 (Crescenzi and Panconesi [20]). Max Bounded Weighted Sat *is* APX-*complete.*

Proof. Observe that Max Bounded Weighted Sat can be trivially approximated with error $\frac{1}{2}$ since the assignment $x_i = 1$, $1 \leqslant i \leqslant n$, has measure either W or $\sum_{i=1}^{n} w_i$.

In order to understand the difficulties in proving the APX-hardness of this problem, notice that, given a problem in APX, no nontrivial bounds can be imposed on the measure of its solutions so that the technique of Theorem 10 cannot be directly applied. However, a problem A belongs to APX if $A \in$ NPO *and* a polynomial-time approximation algorithm exists for A. The right idea is then to use the approximation algorithm in order to consider only those solutions whose measure is bounded in the desired way.

Let A be a maximization problem in APX and let T be the corresponding δ-approximate algorithm. For any input x, let $t(x) = m_A(x, T(x))$ and let us consider the nondeterministic machine N that performs Algorithm 5.

Clearly, for any input x and for any halting computation path of $N(x)$, the output of this computation is a value between $t(x)$ and $kt(x)$, where k is the least integer greater than $1/(1-\delta)$.

Let us now define the PTAS-reduction from A to Max Bounded Weighted Sat in the following way (observe that in this case g does not depend on ε).

1. For any instance x, $f(x) = \varphi_x \wedge z$, where φ_x is the Boolean formula associated to $N(x)$ as in the proof of Theorem 10 and z is a new variable whose weight is equal to $(k-2)t(x)$. Observe that

$$(k-1)t(x) \leqslant \sum_{i=1}^{n} w_i \leqslant kt(x) + (k-2)t(x) = 2(k-1)t(x),$$

so that $f(x)$ is indeed an instance of Max Bounded Weighted Sat.

2. For any instance x and for any truth assignment τ to the variables of $f(x)$,

$$g(x,y) = \begin{cases} y_\tau & \text{if } \tau \text{ satisfies } f(x), \\ T(x) & \text{otherwise,} \end{cases}$$

where y_τ denotes the feasible solution for A recovered from τ as in the proof of Theorem 10.

3. For any ε,

$$c(\varepsilon) = \frac{\varepsilon}{\varepsilon + (k-1)(1-\varepsilon)}.$$

Clearly, c is surjective.

From the above definitions, it follows that, for any truth assignment τ,

$$m_{\mathrm{MBWS}}(f(x),\tau) = m_A(x, g(x,y)) + (k-2)t(x),$$

so that

$$\frac{m_A^*(x) - m_A(x,g(x,y))}{m_A(x,g(x,y))} \leqslant (k-1)\frac{m^*_{\mathrm{MBWS}}(f(x)) - m_{\mathrm{MBWS}}(f(x),\tau)}{m_{\mathrm{MBWS}}(f(x),\tau)}.$$

From Lemma 1, we have that f, g, and c yield a PTAS-reduction from A to Max Bounded Weighted Sat.

In a similar way, we can prove that, for any minimization problem A in APX, $A \leqslant_{\mathrm{PTAS}}$ Max Bounded Weighted Sat. Indeed, we can first PTAS-reduce A to a maximization problem B in APX and, successively, reduce B to Max Bounded Weighted Sat. Let T be the δ-approximate algorithm for A and, for any x, let $t(x) = m_A(x, T(x))$. The problem B is then identical to A apart from the measure function which is defined as follows:

$$m_B(x,y) = \begin{cases} t(x) + k(t(x) - m_A(x,y)) & \text{if } m_A(x,y) \leqslant t(x), \\ t(x) & \text{otherwise,} \end{cases}$$

where k is the least integer greater than $1/(1-\delta)$. The reduction between A and B is defined in the following way (once again g does not depend on ε).

1. For any instance x, $f(x) = x$.
2. For any instance x and for any solution y of $f(x)$,

$$g(x,y) = \begin{cases} y & \text{if } m_A(x,y) \leqslant t(x), \\ T(x) & \text{otherwise.} \end{cases}$$

3. For any ε,

$$c(\varepsilon) = \frac{\varepsilon}{\varepsilon + k(1-\varepsilon)}.$$

Clearly, c is surjective.

By making use of Lemma 1, it is easy to see that the above reduction is a PTAS-reduction so that $A \leqslant_{\text{PTAS}} B \leqslant_{\text{PTAS}}$ MAX BOUNDED WEIGHTED SAT.

Since A was an arbitrary APX problem, it follows that MAX BOUNDED WEIGHTED SAT is APX-complete. □

Remark 5. An interesting result proved by Crescenzi and Panconesi [20] states that, unless P = NP, in both classes NPO and APX "intermediate" problems exist which are neither complete nor in the lower class. Recently, in Crescenzi et al. [19] it has been proved that the bin packing problem is APX intermediate. The significance of this result can be explained in the following way; usually a problem A in a class, say APX, is proved not to be in a lower class, say PTAS, by proving a statement like "if $F \in$ PTAS then P = NP'. The existence of incomplete problems shows that a proof of nonapproximability within any ε does not imply completeness in APX. Thus the notion of completeness introduced in this section captures a deeper level of structure than the notion of NP-completeness. In fact, an NP-complete problem, when considered in its optimization version, can be approximable within any ε or not, complete or incomplete.

At the end of the previous section, we observed that no polynomially bounded NPO-complete problem is known. On the contrary, APX contains polynomially bounded complete problems.

MAX POLYNOMIALLY BOUNDED WEIGHTED SAT is equal to MAX BOUNDED WEIGHTED SAT apart from the measure function which is defined as follows:

$$m_{\text{MPBWS}}(x,\tau) = n + \left\lfloor \frac{n(m_{\text{MBWS}}(x,\tau) - W)}{W} \right\rfloor,$$

where n denotes the number of variables.

Observe that according to the above definition, for any instance x of MAX POLYNOMIALLY BOUNDED WEIGHTED SAT and for any truth-assignment τ, $n \leqslant m_{\text{MPBWS}}(x,\tau) \leqslant 2n$, i.e. this problem is indeed polynomially bounded (even though the values of the numbers that appear in the instance may not be polynomially bounded).

Theorem 13 (Crescenzi and Trevisan [24]). MAX BOUNDED WEIGHTED SAT *is* PTAS-*reducible to* MAX POLYNOMIALLY BOUNDED WEIGHTED SAT.

Proof. Let $x = (\varphi, w_1, \ldots, w_n, W)$ denote an arbitrary instance of Max Bounded Weighted Sat. The reduction is then defined as follows (at last we are using the fact that g may depend on ε!).

1. $f(x) = x$.
2. For any τ and for any $\varepsilon \in (0,1)_Q$,

$$g(x,\tau,\varepsilon) = \begin{cases} \tau & \text{if } \varepsilon > 1/n, \\ \tau^* & \text{otherwise,} \end{cases}$$

where τ^* denotes an optimum solution for Max Bounded Weighted Sat.

3. For any $\varepsilon \in (0,1)_Q$,

$$c(\varepsilon) = \frac{\varepsilon}{\varepsilon + 2(1-\varepsilon)}.$$

Clearly, c is surjective.

Observe that, according to the definition of the PTAS-reducibility, the running time of g can be exponential in $1/\varepsilon$. If $\varepsilon \leqslant 1/n$ then g has enough time to compute τ^* so that, in this case, the fourth condition in the definition of PTAS-reducibility is clearly satisfied.

Assume now that $\varepsilon > 1/n$ and that τ is any truth assignment. Let

$$i_\tau = \left\lfloor \frac{n(m_{\mathrm{MBWS}}(x,\tau) - W)}{W} \right\rfloor.$$

Observe also that

$$m_{\mathrm{MBWS}}(x,\tau) \geqslant W(1 + i_\tau/n).$$

Then

$$\frac{m^*_{\mathrm{MPBWS}}(x) - m_{\mathrm{MPBWS}}(x,\tau)}{m_{\mathrm{MPBWS}}(x,\tau)} = \frac{i_{\tau^*} - i_\tau}{n + i_\tau},$$

while

$$\begin{aligned} \frac{m^*_{\mathrm{MBWS}}(x) - m_{\mathrm{MBWS}}(x,\tau)}{m_{\mathrm{MBWS}}(x,\tau)} &\leqslant \frac{m_{\mathrm{MBWS}}(x,\tau) + (i_{\tau^*} - i_\tau + 1)W/n - m_{\mathrm{MBWS}}(x,\tau)}{m_{\mathrm{MBWS}}(x,\tau)} \\ &= \frac{(i_{\tau^*} - i_\tau + 1)W/n}{m_{\mathrm{MBWS}}(x,\tau)} \\ &\leqslant \frac{(i_{\tau^*} - i_\tau + 1)W/n}{W(1 + i_\tau/n)} \\ &= \frac{i_{\tau^*} - i_\tau + 1}{n + i_\tau}. \end{aligned}$$

If $i_{\tau^*} - i_\tau = 0$ then

$$E_{\mathrm{MBWS}}(x,\tau) \leqslant \frac{m^*_{\mathrm{MBWS}}(x) - m_{\mathrm{MBWS}}(x,\tau)}{m_{\mathrm{MBWS}}(x,\tau)} \leqslant \frac{1}{n + i_\tau} \leqslant \frac{1}{n} < \varepsilon.$$

Otherwise

$$\frac{m^*_{\mathrm{MBWS}}(x) - m_{\mathrm{MBWS}}(x,\tau)}{m_{\mathrm{MBWS}}(x,\tau)} \leqslant \frac{i_{\tau^*} - i_\tau + 1}{n + i_\tau} \leqslant 2\frac{i_{\tau^*} - i_\tau}{n + i_\tau} = 2\frac{m^*_{\mathrm{MPBWS}}(x) - m_{\mathrm{MPBWS}}(x,\tau)}{m_{\mathrm{MPBWS}}(x,\tau)}.$$

From Lemma 1 it thus follows that, in both cases, the fourth condition in the definition of PTAS-reducibility is satisfied and this concludes the proof. □

As a consequence of the previous two theorems, we have the following result.

Corollary 2. Max Polynomially Bounded Weighted Sat *is* APX-*complete.*

In Section 5 we will use this result in order to prove the existence of "natural" APX-complete problems.

4. Reducibilities, satisfiability, and logical definability

In Section 2 the approximation properties of NPO problems have been discussed and their relationship with the combinatorial structure of the problems has been analyzed with the aim of understanding from what intrinsic, structural properties of problems the complexity of achieving approximate solution originates. The previous section, instead, can be seen as an attempt to characterize the approximation classes in terms of hardest problems with respect to suitable reducibilities. Unfortunately, neither of the two approaches turns out to be sufficient in characterizing the class APX. On the one hand, no combinatorial properties have been found so far that are specific for problems in this class, on the other hand the only NPO problems that we have been able to prove APX-complete so far are in a certain sense "artificial".

In this section we shall pursue a different aim. Let us first remember that Cook's theorem allows us to define NP as the class of decision problems that are polynomial-time reducible to the satisfiability problem. Moreover, it is well-known that this is true even if we restrict the instances of this problem to formulae with at most three literals in each clause. If we consider the maximization version of the 3-satisfiability problem, it seems reasonable to define a class analogous to NP, i.e. the class of NPO problems that are reducible to Max 3-Sat. In particular, we first shall prove that Max 3-Sat is $\frac{1}{2}$-approximable and we will then define the class OPT NP as the class of NPO problems that are PTAS-reducible to Max 3-Sat.

Successively, an interesting sufficient condition for NPO problems to be in OPT NP will be given in terms of logical definability. We will finally conclude with several OPT NP-completeness and hardness results.

4.1. History

This section is an elaboration of the line of research that started in Papadimitriou and Yannakakis [46] where the characterization of optimization problems by means

```
begin
  for any variable $u$ in $U$ do
  begin
    $p_j(u)$ := fraction of truth assignments for the remaining variables
                which satisfy the $j$th clause with $u$ true;
    $n_j(u)$ := fraction of truth assignments for the remaining variables
                which satisfy the $j$th clause with $u$ false;
    if $\sum_j p_j(u) \geqslant \sum_j n_j(u)$ then
      assign the value true to $u$
    else assign the value false to $u$;
    update the clauses
  end
end.
```

Algorithm 6. An approximation algorithm for MAX 3-SAT.

of logical formulae has been initially proposed. Building on the work of Papadimitriou and Yannakakis a more systematic approach has been developed in Kolaitis and Thakur [35], leading to the definition of two hierarchies of maximization and minimization problems, respectively. Such hierarchies are indeed shallow and consist of a few coarse levels. Refinements have been proposed in Panconesi and Ranjan [43] and in Behrendt et al. [9]. In Kann [31] more classes based on logical definability are introduced and an extensive survey of results in the area is presented.

Our elaboration mainly consists of focusing our attention on the role played by MAX 3-SAT in order to develop an analogy with the class NP in terms of approximation preserving reducibilities instead of logical definability, and on presenting the logical definability results as a tool for proving approximability properties of optimization problems.

4.2. The class OPT NP

Let us first recall that MAX 3-SAT is the same as MAX SAT with at most three literals per clause. The first result of this section shows that this problem is easily approximable by means of a greedy technique.

Theorem 14 (Johnson [29]). MAX 3-SAT belongs to APX.

Proof. We will prove that Algorithm 6 $\frac{1}{2}$-approximates MAX 3-SAT.

Firstly, we show that at each iteration of the algorithm the value $\sum_j [p_j(u) + n_j(u)]$ cannot decrease. Suppose that the value of variable u must be determined and q variables are still unassigned. Then the number of truth assignments to these q variables that satisfy the jth clause is equal to $2^{q-1}[p_j(u) + n_j(u)]$. Without loss of generality, we can assume that u has been assigned the value true, i.e. $\sum_j p_j(u) \geqslant \sum_j n_j(u)$. Thus, after this assignment, the number of truth assignments to the remaining $q-1$ variables satisfying the jth clause is equal to $2^{q-1}p_j(u)$. Let u' be the variable considered at the next iteration. Then, for any j, $p_j(u')+n_j(u')$ is equal to $2^{q-1}p_j(u)/2^{q-2} = 2p_j(u)$ and

$$\sum_j [p_j(u')+n_j(u')] = 2\sum_j p_j(u) \geqslant \sum_j [p_j(u)+n_j(u)].$$

Secondly, let u_f be the first variable considered by the algorithm. Then, for any j, $p_j(u_f)+n_j(u_f)$ is twice the fraction of truth assignments to all variables satisfying the jth clause which, in turn, is at least $\frac{1}{2}$ (since each clause contains at least one literal, see also Example 17). Thus, $\sum_j [p_j(u_f)+n_j(u_f)]$ is at least m, where m is the number of clauses.

Thirdly, let u_ℓ be the last variable considered by the algorithm. Then, for any j,

$$p_j(u_\ell)+n_j(u_\ell) = \begin{cases} 0 & \text{if the } j\text{th clause is already not satisfied,} \\ 1 & \text{if the } j\text{th clause is either } u_\ell \text{ or } \neg u_\ell, \\ 2 & \text{if the } j\text{th clause is already satisfied or it is } u_\ell \vee \neg u_\ell. \end{cases}$$

Thus, $\sum_j [p_j(u_\ell)+n_j(u_\ell)]$ is at most twice the number of satisfied clauses at the end of the algorithm.

In conclusion, the truth assignment computed by the algorithm satisfies at least $m/2$ clauses. Clearly, the relative error of this solution is at most $\frac{1}{2}$. □

Remark 6. The previous theorem clearly holds for MAX SAT as well.

Since the satisfiability decision problem is a paradigmatic example of an NP problem which is difficult to solve in polynomial time and since no polynomial-time approximation scheme is known for MAX 3-SAT, one can hope that this latter problem is a paradigmatic example of an NPO problem which is approximable but not in PTAS. We thus give the following definition.

Definition 17. An NPO problem A belongs to the *class OPT NP* if it is PTAS-reducible to MAX 3-SAT.

In the following section we will show examples of OPT NP problems.

4.3. Logical characterization of NPO problems

In this section, we show a sufficient condition based on the logical definability of optimization problems for membership in OPT NP. In order to understand what logic has to do with optimization, let us observe that, taking into account that given an input instance x, the search space of x (in symbols, *search*(x)) consists of a combinatorial structure (e.g. power set, permutation group, etc.), the set of feasible solutions y may be defined as a subspace of the search space satisfying a suitable property π_1, depending both on the instance x and on the feasible solution y. That is,

$$sol(x) = \{y\colon y \in search(x) \wedge \pi_1(x,y)\}$$

and, consequently,

$$m^*(x) = goal\{m(x,y) \colon y \in search(x) \wedge \pi_1(x,y)\}.$$

Example 9. In the case of Max Clique, we have that, given a graph $G = (V,E)$, the search space consists of all subsets of V and the set of feasible solutions (i.e. the cliques) is

$$sol(G) = \{V' \colon V' \subseteq V \wedge (\forall u,v)[(u \in V' \wedge v \in V') \rightarrow (u = v \vee (u,v) \in E)]\},$$

i.e.

$$\pi_1(G,V') = (\forall u,v)[(u \in V' \wedge v \in V') \rightarrow (u = v \vee (u,v) \in E)].$$

The optimum measure is then equal to:

$$m^*(G) = \max\{|V'| : V' \subseteq V \wedge \pi_1(G,V')\}.$$

In many cases (and the previous example is one), the measure function m consists of the cardinality of a set of elements z satisfying a given property π_2, depending on x, y, and z. That is,

$$m^*(x) = goal_{y \in search(x)} |\{z \colon \pi_1(x,y) \wedge \pi_2(x,y,z)\}|.$$

Example 10. From the previous example we have that, for Max Clique, $\pi_2(G,V',z) = z \in V'$ so that

$$m^*(G) = \max_{V' \subseteq V} |\{z \colon \pi_1(G,V') \wedge \pi_2(G,V',z)\}|.$$

Example 11. Let us consider Max Sat. In this case, an instance may be described by the set of variables U, by the set of clauses C and by two sets $P,N \subseteq U \times C$ such that $(z,c) \in P$ (respectively, $(z,c) \in N$) if the variable z occurs positive (respectively, negative) in clause c. The search space coincides with the set of feasible solutions (i.e. π_1 is always true) and consists of all subsets of U of variables which have been assigned the value true. For any feasible solution (i.e. truth assignment τ), any clause c satisfied by τ must satisfy the following property:

$$\pi_2((U,C,P,N),\tau,c) = c \in C \wedge \exists z[((z,c) \in P \wedge z \in \tau) \vee ((z,c) \in N \wedge z \notin \tau)].$$

Consequently, the optimum measure is

$$m^*(U,C,P,N) = \max_{\tau \subseteq U} |\{c \colon \pi_2((U,C,P,N),\tau,c)\}|.$$

According to this approach a natural way of characterizing optimization problems may be based on the structure of the logical definition of the property $\pi = \pi_1 \wedge \pi_2$. It can immediately be perceived that the structure of the logical definition of this predicate is different in the two above examples, being universally quantified in the first case and existentially quantified in the second.

Since we are interested in the class characterized by Max 3-Sat, let us see how this problem can be expressed in logical terms. First observe that an instance of Max 3-Sat can be described by means of four subsets of the set C of clauses. In particular, for any i with $0 \leqslant i \leqslant 3$, clause c belongs to the set C_i if the first $3-i$ literals of c are positive and the remaining i literals are negative. For any truth assignment τ, any clause c satisfied by τ must satisfy the following property:

$$\begin{aligned}\pi_2((U,C,C_0,\ldots,C_3),\tau,c) = c \in C \;&\wedge\; (c \in C_0 \rightarrow c_1 \in \tau \vee c_2 \in \tau \vee c_3 \in \tau)\\ &\wedge\; (c \in C_1 \rightarrow c_1 \in \tau \vee c_2 \in \tau \vee c_3 \notin \tau)\\ &\wedge\; (c \in C_2 \rightarrow c_1 \in \tau \vee c_2 \notin \tau \vee c_3 \notin \tau)\\ &\wedge\; (c \in C_3 \rightarrow c_1 \notin \tau \vee c_2 \notin \tau \vee c_3 \notin \tau),\end{aligned}$$

where, for every i with $1 \leqslant i \leqslant 3$, c_i is the ith literal of clause c. Consequently, the optimum measure is

$$m^*(U,C,C_0,\ldots,C_3) = \max_{\tau \subseteq U} |\{c\colon \pi_2((U,C,C_0,\ldots,C_3),\tau,c)\}|.$$

We thus have that the logical definition of Max 3-Sat is even simpler than the previous examples. Indeed, property π is expressible as an unquantified first-order formula. This suggests a sufficient condition that guarantees that an NPO problem belongs to OPT NP. In order to formulate this condition let us give the following definitions.

Definition 18. A *finite similarity type* is a pair of finite sequences of nonnegative integers. Given a finite similarity type $T = (n_1,\ldots,n_k; m_1,\ldots,m_h)$, a *finite T-structure* is a $(k+h+1)$-tuple $S = (X; p_1,\ldots,p_k; f_1,\ldots,f_h)$ where X is a nonempty finite set, called the *domain* of S, $p_i \subseteq X^{n_i}$ for $i = 1,\ldots,k$, and $f_i : X^{m_i} \rightarrow X$ for $i = 1,\ldots,h$.

Remark 7. Intuitively a finite T-structure is the interpretation of a finite similarity type T.

Example 12. Given a set V, a graph over V with set E of edges is a finite T-structure $(V;E;)$, where $T = (2;)$. In other words, in this case $k = 1$, $h = 0$, and $n_1 = 2$.

Definition 19. Let a Σ_n-formula (respectively, a Π_n-formula) be a prefix first-order formula with n alternating blocks of quantifiers beginning with $\exists$ (respectively, $\forall$). The *class* MAX-Σ_n (respectively, MAX-Π_n) consists of maximization problems A whose instances and solutions are finite structures I and S, respectively, and the optimum measure on input I is definable by the expression

$$m^*(I) = \max_S |\{x\colon \varphi(x,S,I)\}|,$$

where φ is a Σ_n-formula (respectively, a Π_n-formula) and x is a tuple of fixed dimension whose components range over the domain of I.

Example 13. From the above examples we have that Max 3-Sat belongs to MAX-Σ_0, Max Sat belongs to MAX-Σ_1, and Max Clique belongs to MAX-Π_1.

Remark 8. From the syntactic point of view, the class MAX-Σ_0 and the class MAX-Σ_1 coincide with the classes MAX SNP and MAX NP, respectively, defined by Papadimitriou and Yannakakis [46]. The first one contains problems that may be characterized by unquantified first-order formulae while the second one contains problems that can be characterized by means of existentially quantified formulae.

Beside MAX 3-SAT other interesting problems belong to the class MAX-Σ_0. Here are a few examples.

Example 14. Let us consider MAX CUT with unitary weights. In this case, the optimum measure can be represented in the following way:

$$m^*(G) = \max_{V' \subseteq V} |\{(u,v): (u,v) \in E \wedge ((u \in V' \wedge v \notin V') \vee (v \in V' \wedge u \notin V'))\}|.$$

Example 15. Let us consider MAX INDEPENDENT SET-B, i.e. a restriction of MAX INDEPENDENT SET in which the degree of the graph is bounded by B. Any instance of this problem can be represented as the set V of nodes and the set $A \subseteq V^{B+1}$ where $(u, v_1, \dots, v_B) \in A$ means that $v_1, \dots, v_B$ are adjacent to u. Note that for every u there must be exactly one tuple $v_1, \dots, v_B$ (possibly with dummy nodes if the degree of u is smaller than B) such that $(u, v_1, \dots, v_B) \in A$. The optimum value can then be represented in the following way:

$$m^*(V,A) = \max_{V' \subseteq V} | \{(u, v_1, \dots, v_B): (u, v_1, \dots, v_B) \in A \\ \wedge u \in V' \wedge v_1 \notin V' \wedge \cdots \wedge v_B \notin V'\}|.$$

Analogously, we can show problems that belong to MAX-Σ_1 beside MAX SAT.

Example 16. Let MAX GSAT-B be the generalized satisfiability problem where the formulae are conjunctions of clauses and the clauses are disjunctions of conjunctive subformulae which may be composed of up to B conjuncts (for example, the formula

$$(\neg x \vee \neg y) \wedge (x \vee z) \wedge ((x \wedge z) \vee (y \wedge z))$$

is a satisfiable MAX GSAT-2 formula). Note that MAX SAT coincides with MAX GSAT-1. We leave to the reader the easy task to prove that, for any constant B, MAX GSAT-B belongs to MAX-Σ_1.

Remark 9. On the basis of Definition 19 a hierarchy of classes of optimization problems syntactically defined in terms of logical formulae can be studied whose properties are the following [35]:

$$\text{MAX-}\Sigma_0 \subset \text{MAX-}\Sigma_1 \subset \text{MAX-}\Pi_1 \subset \text{MAX-}\Pi_2.$$

Moreover, as a consequence of a result of Fagin [25], MAX-Π_2 coincides with the class of all NPO polynomially bounded maximization problems.

It is now time to relate the logical definability of an NPO problem with the class OPT NP. To this aim, we will first show that all problems in MAX-Σ_0 have the following property: the average value of the feasible solutions has a guaranteed constant ratio with respect to the value of the optimum solution.

Definition 20. Let A be an NPO problem. The *average measure of an instance* x is defined in the following way:

$$\mathbb{E}(x) = \frac{\sum_{y \in sol(x)} m(x,y)}{|sol(x)|}.$$

Definition 21. A maximization problem A in NPO is said to have *guaranteed average measure* if a positive constant k exists such that, for any instance x,

$$\mathbb{E}(x) \geqslant \frac{m^*(x)}{k}.$$

Interesting problems have guaranteed average measure: in our context, the following example is maybe one of the most interesting.

Example 17. Let us consider once again Max 3-Sat and let x be a formula with n variables and m clauses. Let T be the set of all possible truth assignments (i.e. feasible solutions). Clearly $|T| = 2^n$. If we now compute the average measure we obtain

$$\begin{aligned}\mathbb{E}(x) &= \frac{\sum_{\tau \in T} \sum_{j=1}^{m} \delta(\tau, c_j)}{2^n} \\ &= \frac{\sum_{j=1}^{m} \sum_{\tau \in T} \delta(\tau, c_j)}{2^n} \\ &\geqslant \frac{\sum_{j=1}^{m} (2^n - 2^{n-|c_j|})}{2^n} \geqslant \frac{m}{2} \geqslant \frac{m^*(x)}{2},\end{aligned}$$

where

$$\delta(\tau, c_j) = \begin{cases} 1 & \text{if } \tau \text{ satisfies } c_j, \\ 0 & \text{otherwise.} \end{cases}$$

Hence Max 3-Sat has guaranteed average measure with constant 2.

By a suitable generalization of the previous example we can show the following result.

Theorem 15 (Crescenzi and Silvestri [22] and Papadimitrion and Yannakakis [46]). *Any NPO problem in* MAX-Σ_0 *has guaranteed average measure.*

Proof. The proof is based on the logical characterization of the problems belonging to the class MAX-Σ_0 and mimics the proof of the previous example. The only difference is that the constant is not 2 anymore but depends on the length of formula φ. □

Beside being interesting for its own combinatorial meaning, the above theorem suggests that problems in MAX-Σ_0 are easy to approximate since a randomly chosen feasible solution is a "good" solution with high probability. Indeed, by simply generalizing the approximation algorithm for MAX 3-SAT it is possible to prove the following result.

Theorem 16 (Papadimitriou and Yannakakis [46]). *Every problem in* MAX-Σ_0 *belongs to* APX.

The next theorem shows that any problem in MAX-Σ_0 not only is approximable but it also belongs to OPT NP.

Theorem 17 (Papadimitriou and Yannakakis [46]). *Every problem in* MAX-Σ_0 *is* PTAS*-reducible to* MAX 3-SAT.

Proof. Suppose we are given a problem A in MAX-Σ_0. By definition, its optimum measure satisfies, for every instance x, the following equality:

$$m^*(x) = \max_S |\{z: \varphi(z,x,S)\}|,$$

where φ is a quantifier-free first-order formula, z is a tuple of fixed dimension whose components range over the domain of x, and S is the structure corresponding to a feasible solution. Moreover, from the previous theorem it follows that A admits a δ-approximate algorithm T for some δ.

Let us denote with $\varphi_1, \varphi_2, \ldots, \varphi_m$ the formulae corresponding to the possible values of z. Hence the problem of maximizing the set of values of z which satisfy φ is transformed into the problem of maximizing the set of formulae φ_i that can be simultaneously satisfied by a suitable truth assignment.

We now transform any formula φ_i into a set C_i of clauses with at most three literals each. The transformation is obtained by considering the Boolean circuit corresponding to the formula φ_i (whose inputs are the original variables of φ_i) and by describing all gates by 3-clauses, with the help of auxiliary variables. In particular, for each gate g of the circuit:

1. If g is a NOT gate with input a, C_i includes the clauses

$$g \vee a, \neg g \vee \neg a.$$

2. If g is an AND gate with input a and b, C_i includes the clauses

$$a \vee \neg g, b \vee \neg g, g \vee \neg a \vee \neg b.$$

3. If g is an OR gate with input a and b, C_i includes the clauses

$$\neg a \vee g, \neg b \vee g, \neg g \vee a \vee b.$$

4. If g is the output gate, C_i includes the clause g.

For example, the formula $a \wedge (b \vee \neg c)$ is transformed by means of the auxiliary variables g_1, g_2, and g_3 into the following set of clauses:

$$(c \vee g_1) \wedge (\neg c \vee \neg g_1) \wedge (\neg g_1 \vee g_2) \wedge (\neg b \vee g_2) \wedge (\neg g_2 \vee g_1 \vee b)$$
$$\wedge (g_2 \vee \neg g_3) \wedge (a \vee \neg g_3) \wedge (g_3 \vee \neg g_2 \vee \neg a) \wedge g_3.$$

This transformation guarantees that any truth assignment τ to the variables of φ_i can be extended to the auxiliary variables in such a way that all clauses belonging to C_i are satisfied except, possibly, the variable corresponding to the output gate (in case φ_i is not satisfied by τ).

The overall instance $f(x)$ of Max 3-Sat is then the union of all the sets of clauses $C_1, \ldots, C_m$.

Clearly, any truth assignments τ to the variables of $f(x)$ can be improved in order to satisfy $m_2 = m(k-1) + m_1$ clauses and to identify a solution $g(x, \tau)$ of A whose measure is equal to m_1. In particular, $m^*(f(x)) = m(k-1) + m^*(x)$. Then the following inequality holds:

$$m^*(x) - m(x, g(x, \tau)) \leqslant m^*(f(x)) - m(f(x), \tau).$$

Moreover, we can assume that $m(x, g(x, \tau)) \geqslant m(x, T(x)) \geqslant (1-\delta) m^*(x)$. Finally, from the proof of Theorem 15 it easily follows that a constant h exists such that $m^*(x) \geqslant m/h$, so that

$$\frac{m^*(x) - m(x, g(x, \tau))}{m(x, g(x, \tau))} \leqslant (1 + h(k-1)(1-\delta)) \frac{m^*(f(x)) - m(f(x), \tau)}{m(f(x), \tau)}.$$

From Lemma 1, we then have that the above reduction is a PTAS-reduction if we define:

$$c(\varepsilon) = \frac{\varepsilon}{\varepsilon + \beta(1-\varepsilon)}$$

with $\beta = 1 + h(k-1)(1-\delta)$. In conclusion, A is PTAS-reducible to Max 3-Sat. □

The above two theorems thus state a sufficient condition for an NPO problem to be in OPT NP (and hence to be approximable) based only on its logical definability. Clearly, this condition is not necessary since all problems in MAX-Σ_0 are polynomially bounded while OPT NP contains problems, such as Max {0,1}-Knapsack, which are not polynomially bounded. Moreover, OPT NP contains minimization problems as shown in the following example.

Example 18. Min Node Cover-B, i.e. Min Node Cover restricted to graphs of degree at most B, belongs to OPT NP since it is easy to see that Min Node Cover-B is PTAS-reducible to Max Independent Set-B which is in MAX-Σ_0 (see Example 15).

Remark 10. The relationship between the logical definability of NP minimization problems and their approximation properties has also been studied but it seems to be somewhat different than that in the case of maximization ones (see [36]).

4.4. OPT NP-hardness

The class OPT NP captures the basic idea that was behind its definition, i.e., to characterize a class of problems which are *no more* difficult to approximate than MAX 3-SAT. It is then natural to look for optimization problems which are at least as hard as MAX 3-SAT. Once again, the comparison between pairs of optimization problems is formally defined by means of the notion of reducibility.

Definition 22. An NPO problem A is OPT NP-hard if MAX 3-SAT $\leqslant_{\mathrm{PTAS}} A$.

Example 19. Let us prove that MAX 3-SAT is PTAS-reducible to MAX CLIQUE [8, 17]. Given an instance (U, C) of MAX 3-SAT, we define a graph $G = (V, E)$ such that

$$V = \{(l, c): l \in c\}$$

and

$$E = \{((l_1, c_1), (l_2, c_2)): l_1 \neq \neg l_2 \wedge c_1 \neq c_2\},$$

where l denotes a variable or the negation of a variable in U and c is a clause. Clearly, any clique V' in G correspond to a truth assignment satisfying at least $|V'|$ clauses. Thus, the above reduction is a PTAS-reduction with $c(\varepsilon) = \varepsilon$.

Many other problems can be shown to be OPT NP-hard: MAX SAT, MAX GSAT-B (see Example 16), and MIN NODE COVER [46], MIN Δ-TSP , that is, MIN TSP restricted to distance values satisfying the triangle inequality [47], MIN STEINER TREE [12], MIN SUPERSTRING [13], and MAX PATH [33]. For each of them, the following thus holds: if it belongs to PTAS then OPT NP is contained in PTAS. In the following section we shall see that this latter event is unlikely to happen.

Finally, among the problems which are OPT NP-hard, those who are *equivalent* to MAX 3-SAT, i.e. belong to OPT NP and thus are OPT NP-complete, are of particular interest since they belong to PTAS if and only if OPT NP is contained in PTAS. This is clearly analogous to the concept of APX-complete problem, but this time we are able to find "natural" completeness results.

Indeed, examples of OPT NP-complete problems are provided by Papadimitriou and Yannakakis [46] such as MAX 2-SAT, MAX CUT, MAX INDEPENDENT SET-B, MIN NODE COVER-B.

5. Probabilistically checkable proofs and approximation

We have already observed in Section 2 that a well-known technique for showing negative approximability results is the gap technique and that very few nontrivial gaps were obtained since a couple of years ago. The creation of a gap in the measure of an optimization problem has been recently connected with another kind of gap originating from the probabilistic model of computation. This connection can be intuitively described as follows: a probabilistic algorithm for an NP-complete language requires a large gap in the probability of acceptance of correct and incorrect inputs. Such algorithms can then be used to construct a family of instances of an optimization problem with a large gap in the measure function thus proving its nonapproximability unless P = NP.

Unfortunately, no NP-complete problem is known to admit a "simple" probabilistic algorithm but it has been recently shown that any NP-complete problem admits an "efficient" *probabilistically checkable proof*. Intuitively, such proofs allow to prove whether an instance belongs to a given language by using few random coin tossing and by checking a constant number of bits of the proof!

In this section we see in a very sketched way how this can be obtained and we subsequently show how this per se interesting result can be applied in order to prove that no OPT NP-hard problem admits a polynomial-time approximation scheme. Finally, we conclude by showing that OPT NP indeeed coincides with APX.

5.1. *History*

The theory of probabilistic checking of proofs started with the work by Babai, et al. [7] where the notion of transparent proof was introduced: informally, a transparent proof either is correct or mistakes will appear almost everywhere, thus enabling a probabilistic verifier to check it by a cursory examination. The results of Feige et al. [26] then triggered the successive important results of Arora and Safra [2] and of Arora et al. [1], part of which are summarized in this section. Other results along this line of research have been obtained by Bellare et al. [10], Condon et al. [16], and Lund and Yannakakis [39]. The fact that OPT NP is equal to APX is a consequence of the results obtained by Khanna et al. [34] and Crescenzi and Trevisan [24].

5.2. *Probabilistically checkable proofs*

Even though the notion of a proof is an intuitive one, theorem-proving procedures may substantially differ one from the other. The most natural procedure consists in writing down the proof in a book, while a more general way of communicating a proof is based on the notion of interaction and consists in explaining the proof to some recipients as in the case of a teacher-students environment.

In this section, we shall consider book-writing alike procedures, called probabilistically checkable proofs, in which a prover writes the proof on a book and the verifier checks its correctness so that if the theorem is true, then a proof exists convincing the verifier to accept, otherwise no proof can convince the verifier with better than negligible probability.

In particular, we shall consider such theorem-proving procedures from the following point of view: having assumed that the proof is available somewhere (e.g. as an oracle of a Turing machine), how much of it does the verifier have to know in order to be convinced that the proof is correct? Clearly, if the verifier is deterministic, the proof has to be read entirely. However, if it is probabilistic, then we shall see that a sublinear number of random and query bits is sufficient to characterize NP.

Definition 23. A language L admits a *probabilistically checkable proof* if an oracle probabilistic Turing machine T exists such that

1. For every $x \in L$, an oracle X_x exists such that T^{X_x} accepts x with probability 1.
2. For every $x \notin L$ and for every oracle X, T^X accepts x with probability at most 1/3.

Definition 24. The class $\mathrm{PCP}(r(n), q(n))$ is the set of languages which admit a probabilistically checkable proof such that the corresponding machine T operates in polynomial time and generates at most $\mathrm{O}(r(n))$ random bits and at most $\mathrm{O}(q(n))$ queries.

Theorem 18 (Arora et al. [1]). *$NP \subseteq PCP(\log n, 1)$.*

Proof (*sketch*). Informally the proof can be summarized as follows.

1. Define a restricted version of probabilistically checkable proof (rPCP) in which both the input and the proof are "segmented" into at most $2^{\mathrm{O}(r(n))}$ segments of length at most $\mathrm{O}(q(n))$, the verifier only accesses a constant number of these segments, and its output can be computed by a circuit whose size is polynomial with respect to $q(n)$. Moreover, the verifier uses the random bits simply to determine the addresses of the segments to be read and does this in a nonadaptive way, that is, before receiving any of the answers.
2. *Composition Lemma.* Show that if NP is contained both in $\mathrm{rPCP}(r_1(n), q_1(n))$ and in $\mathrm{rPCP}(r_2(n), q_2(n))$, then NP is contained in $\mathrm{rPCP}(r_1(n)+r_2(q_1^{\mathrm{O}(1)}(n)), q_2(q_1^{\mathrm{O}(1)}(n)))$.
3. Show that NP is contained in $\mathrm{rPCP}(\log n, \log^{\mathrm{O}(1)} n)$. By applying the composition lemma, we then obtain that NP is contained in $\mathrm{rPCP}(\log n, (\log\log n)^{\mathrm{O}(1)})$.
4. Show that NP is contained in $\mathrm{rPCP}(n^{\mathrm{O}(1)}, 1)$.
5. By applying the composition lemma to the previous two results, we obtain that NP is contained in $\mathrm{rPCP}(\log n, 1)$.

While the proof of the composition lemma follows in a quite straightforward way, the last three steps borrow significantly from results on polynomial checking, proof verification, program result checking, and coding theory. Giving the details of these results goes far beyond the scope of this survey and we thus refer the interested reader to the papers that originated them. $\square$

5.3. The hardness of approximation

The areas of proof checking and of approximation seem quite unrelated at a first glance. However, as stated at the beginning of this section, the existence of probabilistically checkable proofs for NP implies that the membership question for any NP language can be converted to an NPO problem which has a gap associated with it.

Theorem 19 (Arora et al. [1]). MAX 3-SAT *does not belong to* PTAS, *unless* P = NP.

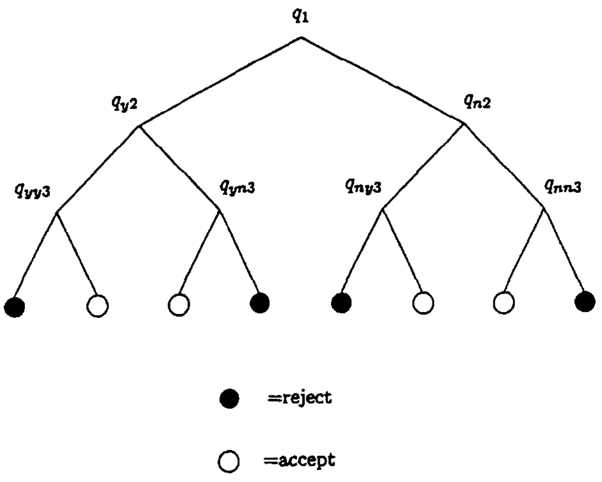

Fig. 5. A query-tree of depth 3.

Proof. Let L be a language in NP. From Theorem 18 it then follows that an oracle probabilistic Turing machine T exists such that

1. For every $x \in L$, an oracle X_x exists such that T^{X_x} accepts x with probability 1.
2. For every $x \notin L$ and for every oracle X, T^X accepts x with probability at most $\frac{1}{3}$.

Moreover, T generates at most $h_1 \log n$ random bits and at most h_2 queries where h_1 and h_2 are constants.

We shall now derive a Boolean formula φ_x such that if $x \in L$ then φ_x is satisfiable, otherwise at most $1-g$ of all clauses of φ_x can be satisfied simultaneously, where g is a suitable constant. This clearly implies that Max 3-Sat does not admit a polynomial-time approximation scheme unless P = NP.

In order to construct φ_x, observe that, for any possible string y of length at most $h_1 \log n$, the computation $T(x)$ when using y as its random string can be represented as a "query-tree" of depth at most h_2 whose nodes correspond to the queries performed by the computation (see Fig. 5). This tree, in turn, can be translated into a constant size Boolean formula φ_{xy} which is satisfiable if and only if $T(x)$ accepts and whose variables are the queried bits (referring to the example shown in Fig. 5, we have

$$\varphi_{xy} = (q_1 \wedge q_{y2} \wedge \neg q_{yy3}) \vee (q_1 \wedge \neg q_{y2} \wedge q_{yn3}) \vee (\neg q_1 \wedge q_{n2} \wedge \neg q_{ny3}) \\ \vee (\neg q_1 \wedge \neg q_{n2} \wedge q_{nn3})).$$

The formula φ_x is then the conjunction of the $m = n^{h_1}$ formulae φ_{xy} (possibly rewritten in conjunctive normal form with at most three literals per clause).

Clearly, if $x \in L$ then a truth assignment exists satisfying φ_x so that $opt(x) = km$ for a suitable constant k. But if $x \notin L$ then any truth assignment will be unable to satisfy more than a constant fraction of all clauses of φ_x. In particular, since T accepts with probability at most $\frac{1}{3}$, then at least two out of three formulae φ_{xy} contain at least one unsatisfied clause. Then, $opt(x) \leqslant km - \frac{2}{3}m = (1 - 2/3k)km$. This concludes the proof. □

Corollary 3. *No* OPT NP*-hard problem belongs to* PTAS, *unless* P = NP.

Corollary 4. MAX CLIQUE *does not belong to APX, unless* P = NP.

Proof. First observe that, for any graph G, it is possible to compute in polynomial time a graph G' such that G contains a clique of size k if and only if G' contains a clique of size k^2. Indeed, G' contains a copy of G for any node of G itself. Given a copy corresponding to a node u, we connect any node of such a copy to each node of the copies corresponding to nodes adjacent to u.

This observation implies that if, for *some* $\varepsilon < 1$, an ε-approximate algorithm for MAX CLIQUE exists, then this problem admits a δ-approximate algorithm, for *any* $\delta < 1$. That is, if MAX CLIQUE belongs to APX then it belongs to PTAS.

The result thus follows from the OPT NP-hardness (see Example 19) and from the above corollary. □

Remark 11. The previous corollary can indeed be strengthened. In fact, it has been proved that, for some ε, MAX CLIQUE cannot be n^{ε}-approximated, unless P = NP [1].

5.4. *OPT NP versus APX*

In this section we show that MAX 3-SAT is APX-complete. This result thus settles the question of the relationship between OPT NP and APX posed in the previous section. We already know that OPT NP is contained in APX. An immediate consequence of the APX-completeness of MAX 3-SAT is that the two classes indeed coincide.

Recall that, for any language L in NP, a polynomial p and a polynomial-time decidable binary relation R exist such that

$$x \in L \Leftrightarrow \exists y[|y| \leqslant p(|x|) \wedge R(x,y)].$$

In order to prove the main result of this section, we first state a stronger version of Theorem 19.

Theorem 20 (Khanna et al. [34]). *Let $L = (p,R)$ be a language in NP. Then a polynomial-time computable function f and a positive rational $\delta < 1$ exist such that, for any x, the following hold.*

1. *$f(x)$ is an instance of* MAX 3-SAT *with c clauses, where c is dependent only on $|x|$.*
2. *$(1-\delta)c$ clauses of $f(x)$ are satisfiable by some truth-assignment.*
3. *If $x \in L$ then $f(x)$ is satisfiable.*
4. *If $x \notin L$ then no truth-assignment satisfies more than $(1-\delta)c$ clauses of $f(x)$.*
5. *Given a truth-assignment which satisfies $f(x)$, a word y such that $|y| \leqslant p(|x|) \wedge R(x,y)$ can be costructed in polynomial time.*
6. *Given a truth-assignment which satisfies more than $(1-\delta)c$ clauses of $f(x)$, a truth-assignment which satisfies $f(x)$ can be constructed in polynomial time.*

Proof (*sketch*). Properties 1, 3, and 4 are immediately obvious from the proof of Theorem 19. Property 6 is based on simple considerations on error-correcting codes while properties 2 and 5 are obtained by reconsidering and slightly modifying the proof of Theorem 18. □

Theorem 21 (Khanna et al. [34]). Max Polynomially Bounded Weighted Sat *is* PTAS *-reducible to* Max 3-Sat.

Proof. Given an instance x of Max Polynomially Bounded Weighted Sat, let $d = 2n$ where n denotes the number of variables. Then, for any truth-assignment τ, $d/2 \leqslant m_{\text{MPBWS}}(x, \tau) \leqslant d$. For any i with $1 \leqslant i \leqslant d$, let $L_i = \{y\colon \exists\tau[m_{\text{MPBWS}}(y, \tau) \geqslant i]\}$. Clearly, L_i is in NP. Let φ_i be the instance of Max 3-Sat which is obtained from Theorem 20 when applied to the instance x of L_i. Consider now the Boolean formula $\varphi = \bigwedge_{i=d/2}^{d} \varphi_i$. Clearly, $m^*(\varphi) = (1-\delta)cd + \delta c m^*_{\text{MPBWS}}(x)$. Moreover, given an assignment τ, let j_τ be the maximum i such that τ satisfies more than $(1-\delta)c$ clauses of φ_i. From properties 5 and 6 of Theorem 20, it follows that a truth-assignment τ' can be constructed in polynomial time such that $m_{\text{MPBWS}}(x, \tau') \geqslant j_\tau$. We thus have that

$$\begin{aligned}
\frac{m^*_{\text{M3S}}(\varphi) - m_{\text{M3S}}(\varphi, \tau)}{m_{\text{M3S}}(\varphi, \tau)} &= \frac{(1-\delta)cd + \delta c m^*_{\text{MPBWS}}(x) - m_{\text{M3S}}(\varphi, \tau)}{m_{\text{M3S}}(\varphi, \tau)} \\
&\geqslant \frac{(1-\delta)cd + \delta c m^*_{\text{MPBWS}}(x) - (1-\delta)cd - \delta c j_\tau}{(1-\delta)cd + \delta c j_\tau} \\
&= \delta \frac{m^*_{\text{MPBWS}}(x) - j_\tau}{(1-\delta)d + \delta j_\tau} \\
&\geqslant \delta \frac{m^*_{\text{MPBWS}}(x) - j_\tau}{(1-\delta)2j_\tau + \delta j_\tau} \\
&= \frac{\delta}{2-\delta} \frac{m^*_{\text{MPBWS}}(x) - j_\tau}{j_\tau} \\
&\geqslant \frac{\delta}{2-\delta} \frac{m^*_{\text{MPBWS}}(x) - m_{\text{MPBWS}}(x, \tau')}{m_{\text{MPBWS}}(x, \tau')}.
\end{aligned}$$

From Lemma 1, it then follows that

$$E_{\text{M3S}}(\varphi, \tau) \leqslant c(\varepsilon) \text{ implies } E_{\text{MPBWS}}(x, \tau') \leqslant \varepsilon.$$

where $c(\varepsilon) = \varepsilon/(\varepsilon + [(2-\delta)/\delta](1-\varepsilon))$. That is, we have obtained a PTAS-reduction from Max Polynomially Bounded Weighted Sat to Max 3-Sat and this concludes the proof. □

An immediate consequence of the above theorem is the following result.

Corollary 5. Max 3-Sat *is* APX*-complete.*

Combining this corollary with the results presented in Section 4 yields the following results.

Corollary 6. OPT NP *coincides with* APX.

Corollary 7. *The following* NPO *problems are* APX-*complete.*
1. Max Sat,
2. Max 2-Sat,
3. Max GSat-B,
4. Min Node Cover,
5. Max Independent Set-B,
6. Max Cut,
7. Min Δ-TSP,
8. Min Steiner Tree,
9. Min Superstring.

We have thus shown that Max Sat indeed plays the same role with respect to NPO problems in APX as the satisfiability decision problem plays with respect to NP problems. This positively answers the question that motivated the definition of OPT NP.

As a concluding remark, we observe that any optimization problem in APX which has been or will be shown to be hard for OPT NP is indeed APX-complete.

6. Conclusion

In this paper we presented in a unified and updated approach the main results concerning the characterization of approximation classes of optimization problems. In particular, we reviewed the algorithmic and combinatorial characterizations of classes PTAS and FPTAS and, successively, we concentrated on the study of the properties of class APX. With respect to this latter issue, the most successful tools have been approximation preserving reducibilities and probabilistically checkable proofs. Indeed, on the ground of several recent results, we have been able to show that APX coincides with the class of optimization problems which are PTAS-reducible to Max Sat. As a consequence of this result, many well-known problems that are known to be approximable turn out to be APX-complete and hence do not allow a polynomial-time approximation scheme unless P = NP. Our presentation of these APX-completeness results is essentially based on two steps. In the first step we have introduced the class OPT NP as an analog of the class NP in the world of optimization problems while in the second step we have shown that OPT NP coincides with APX. This kind of presentation allows the reader to easily understand the historical path that led to the final results. However, the equivalence between OPT NP and APX implies that, in the future, it will suffice to use the notion of APX-completeness: indeed, *any result of* OPT NP-*hardness or, equivalently,* MAX SNP-*hardness will turn out to be a statement of* APX-*hardness.*

As a consequence of these recent developments, we expect that in the near future it will be possible to prove that many approximable problems do not belong to PTAS unless P = NP. Indeed, this can be done by making use of techniques similar to those applied in the world of decision problems.

Moreover, many technical problems are still open. Among them we believe that the following are the most interesting.

1. To characterize the class APX in terms of algorithmic procedures and/or combinatorial properties in analogy with the results of Section 2. Indeed, in [6, 34, 46] it is shown that the "core" of APX can be characterized either by a greedy strategy or by a local search strategy.
2. To compare different notions of approximation preserving reducibilities. For example, it is easy to see that the L-reducibility is strictly stronger than the PTAS-reducibility unless P = NP.
3. To study classes of problems which are approximable within a nonconstant ratio. In particular, given a family F of functions, let F-APX denote the class of NPO problems which are $(g(n) - 1)/g(n)$-approximable for a given $g \in F$. It is then worth looking for completeness results with respect to F-APX: indeed, in [34] such results have been obtained for the classes of NPO problems which have polynomial-time algorithms with performance ratio bounded either polynomially or logarithmically.
4. To provide a proof of the APX-completeness of MAX 3-SAT without making use of the notion of probabilistically checkable proof.

Acknowledgements

Many thanks to Luca Trevisan for his helpful comments and suggestions. We also thank Andrea Clementi, Miriam Duella, and Viggo Kann for a careful reading of a preliminary version of the paper. Finally, thanks to Rajeev Motwani and Madhu Sudan for giving us a preliminary version of the results contained in [34].

sectionAppendix A

Appendix A. A list of NPO problems

In this section we give the formal definition of all NPO problems that have been used throughout the survey (a more extensive list containing approximately 150 problems along with their approximability properties is presented in [18]). For the sake of simplicity, we avoid to specify the goal of a problem since it can be immediately derived from the name of the problem itself.

Max Clique

Instance: Graph $G=(V,E)$.

Solution: Subset $V' \subseteq V$ such that every two nodes in V' are joined by an edge in E.

Measure: $|V'|$.

Max Cut

Instance: Graph $G=(V,E)$ and weight $w(e)$ for each $e \in E$.

Solution: Subset $V' \subseteq V$.

Measure: Sum of the weights of the edges from E that have one endpoint in V' and one endpoint in $V - V'$.

Max Independent Set

Instance: Graph $G=(V,E)$.

Solution: Subset $V' \subseteq V$ such that no two nodes in V' are joined by an edge in E.

Measure: $|V'|$.

Max Knapsack

Instance: Finite set $I = \{1,\dots,n\}$, for each $i \in I$ an integer size a_i and an integer profit p_i, and integer b.

Solution: Function $f : I \to [0,1]$ such that $\sum_{i \in I} f(i)a_i \leqslant b$.

Measure: $\sum_{i \in I} f(i)p_i$.

Max Planar Independent Set

Instance: Planar graph $G=(V,E)$.

Solution: Subset $V' \subseteq V$ such that no two nodes in V' are joined by an edge in E.

Measure: $|V'|$.

Max Path

Instance: Graph $G=(V,E)$.

Solution: Simple path in G.

Measure: Length of the path.

Max Sat

Instance: Set U of variables and collection C of clauses over U.

Solution: Truth assignment to the variables in U.

Measure: Number of satisfied clauses.

Max Weighted Sat

Instance: Boolean formula φ with variables $x_1,\dots, x_n$ of weights $w_1,\dots,w_n$.

Solution: Truth assignment τ to the variables that satisfies φ.

Measure: $\max(1, \sum_{i=1}^{n} w_i \tau(x_i))$.

Max {0,1}-Knapsack

Instance: Finite set $I = \{1,\dots,n\}$, for each $i \in I$ an integer size a_i and an integer profit p_i, and integer b.

Solution: Subset $S \subseteq I$ such that $\sum_{i \in S} a_i \leqslant b$.
Measure: $\sum_{i \in S} p_i$.

MIN GRAPH COLORING
Instance: Graph $G = (V, E)$.
Solution: Total function $f : V \rightarrow N$ such that, for each edge $(u, v) \in E$, $f(u) \neq f(v)$.
Measure: Cardinality of the range of f.

MIN m-PROCESSOR SCHEDULING
Instance: A set of task T and m execution-time functions $l_i : T \rightarrow N$, one for each of m available processors.
Solution: A task-assignment function $f : T \rightarrow \{1, \ldots, m\}$ that specifies to which processor a given task is assigned.
Measure: $\max_{i \in \{1, \ldots, m\}} \sum_{t \in T : f(t) = i} l_i(t)$.

MIN NODE COVER
Instance: Graph $G = (V, E)$.
Solution: Subset $V' \subseteq V$ such that, for each edge $(u, v) \in E$, at least one of u and v belongs to V'.
Measure: $|V'|$.

MIN PLANAR NODE COVER
Instance: Planar graph $G = (V, E)$.
Solution: Subset $V' \subseteq V$ such that, for each edge $(u, v) \in E$, at least one of u and v belongs to V'.
Measure: $|V'|$.

MIN SPANNING TREE
Instance: Weighted graph $G = (V, E)$.
Solution: A spanning tree for G, i.e., a subgraph $G' = (V, E')$ of G that is connected and has no cycles.
Measure: $\sum_{e \in E'} w(e)$ where $w(e)$ denotes the weight of edge e.

MIN STEINER TREE
Instance: Weighted complete graph $G = (V, E)$ and a subset $S \subseteq V$ of required nodes.
Solution: A subtree T of G that includes all the nodes in S.
Measure: $\sum_{e \in T} w(e)$ where $w(e)$ denotes the weight of edge e.

MIN SUPERSTRING
Instance: A finite set S of words over an alphabet Σ.
Solution: A word $y \in \Sigma^*$ such that each $s \in S$ is a substring of y.
Measure: $|y|$.

Min TSP

Instance: Complete weighted graph $G = (V, E)$.

Solution: A permutation π of V.

Measure: $\sum_{i=1}^{|V|-1} w(v_{\pi(i)}, v_{\pi(i+1)}) + w(v_{\pi(|V|)}, v_{\pi(1)})$ where $w(e)$ is the weight of edge e.

Min Weighted Sat

Instance: Boolean formula φ with variables $x_1, \dots, x_n$ of weights $w_1, \dots, w_n$.

Solution: Truth assignment τ to the variables that satisfies φ.

Measure: $\max(1, \sum_{i=1}^{n} w_i \tau(x_i))$.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, Proof verification and hardness of approximation problems; in: *Proc. 33-rd Ann. IEEE Symp. on Foundations of Computer Science* (1992) 14–23.

[2] S. Arora and S. Safra, Probabilistic checking of proofs: a new characterization of NP, in: *Proc. 33rd Ann. IEEE Symp. on Foundations of Computer Science* (1992) 2–13.

[3] G. Ausiello, A. D'Atri and M. Protasi, Structure preserving reductions among convex optimization problems, *J. Comput. Systems Sci.* **21** (1980) 136–153.

[4] G. Ausiello, A. D'Atri and M. Protasi, Lattice theoretic properties of NP-complete problems, *Fundamenta Informaticae* **4** (1981) 83–94.

[5] G. Ausiello, A. Marchetti-Spaccamela and M. Protasi, Toward a unified approach for the classification of NP-complete optimization problems, *Theoret. Comput. Sci.* **12** (1980) 83–96.

[6] G. Ausiello and M. Protasi, Local search, reducibility, and approximability of NP optimization problems, *Inform. Process. Lett.* **54** (1995) 73–79.

[7] L. Babai, L. Fortnow, L.A. Levin and M. Szegedy, Checking computations in polylogarithmic time, in: *Proc. 23-rd Ann. ACM Symp. on Theory of Computing* (1991) 21–31.

[8] J.L. Balcazar, J. Diaz and J. Gabarro, *Structural Complexity I* (Springer, Berlin, 1988).

[9] T. Behrendt, K. Compton and E. Graedel, Optimization problems: expressibility, approximation properties and expected asymptotic growth of optimal solutions, in: *Proc. 6th Workshop on Computer Science Logic* (1993) 43–60.

[10] M. Bellare, S. Goldwasser, C. Lund and A. Russell, Efficient probabilistically checkable proofs and applications to approximation. in: *Proc. 25th Ann. ACM Symp. on Theory of Computing* (1993) 294–304.

[11] P. Berman and G. Schnitger, On the complexity of approximating the independent set problem, *Information and Computation* **96** (1992) 77–94.

[12] M. Bern and P. Plassmann, The Steiner problem with edge lengths 1 and 2, *Inform Process. Lett.* **32** (1989) 171–176.

[13] A. Blum, T. Jiang, M. Li, J. Tromp and M. Yannakakis, Linear approximation of shortest superstrings, in: *Proc. 23-rd Ann. ACM Symp. on Theory of Computing* (1991) 328–336.

[14] D.P. Bovet and P. Crescenzi, *Introduction to the Theory of Complexity* (Prentice-Hall, Englewood Cliffs, NJ, 1993).

[15] D. Bruschi, D. Joseph and P. Young, A structural overview of NP optimization problems, *Algorithms Rev.* **2** (1991) 1–26.

[16] A. Condon, J. Feigenbaum, C. Lund and P. Shor, Probabilistically checkable debate systems and approximation algorithms for PSPACE-hard functions, in: *Proc. 25th Ann. ACM Symp. on Theory of Computing* (1993) 305–314.

[17] P. Crescenzi, C. Fiorini and R. Silvestri, A note on the approximation of the MAX CLIQUE problem, *Inform. Process. Lett.* **40** (1991) 1–5.

[18] P. Crescenzi and V. Kann, A compendium of NP optimization problems, Manuscript, 1994.

[19] P. Crescenzi, V. Kann and L. Trevisan, Natural complete and intermediate problems in approximation classes, Manuscript, 1994.

[20] P. Crescenzi and A. Panconesi, Completeness in approximation classes. *Inform. and Comput.* **93** (1991) 241–262.

[21] P. Crescenzi and R. Silvestri, Relative complexity of evaluating the optimum cost and constructing the optimum for maximization problems, *Inform. Process. Lett.* **33** (1990) 221–226.

[22] P. Crescenzi and R. Silvestri, Average measure, descriptive complexity and approximation of maximization problems, *Int. J. Foundations Comput. Sci.* **4** (1993) 15–30.

[23] P. Crescenzi, R. Silvestri and L. Trevisan, On the structure of complete sets in approximation classes, Manuscript, 1994.

[24] P. Crescenzi and L. Trevisan, On approximation scheme preserving reducibility and its applications, *Proc. 14th Conference on Foundations of Software Technology and Theoretical Computer Science*, Lecture Notes in Computer Science, Vol. 880 (Springer, Berlin, 1994) 330–341.

[25] R. Fagin, Generalized first-order spectra and polynomial-time recognizable sets, in: *SIAM-AMS Proc.* (1974) 43–73.

[26] U. Feige, S. Goldwasser, L. Lovasz, S. Safra and M. Szegedy, Approximating clique is almost NP-complete, in: *Proc. 32nd Ann. IEEE Symp. on Foundations of Computer Science* (1991) 2–12.

[27] M.R. Garey and D.S. Johnson, Strong NP-completeness results: motivation, examples, and implications. *J. ACM* **25** (1978) 499–508.

[28] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness* (Freeman, San Fransico, CA, 1979).

[29] D.S. Johnson, Approximation algorithms for combinatorial problems, *J. Comput. and Systems Sci.* **9** (1974) 256–278.

[30] V. Kann, On the approximability of the maximum common subgraph problem, in: *Proc. 9-th Ann. Symp. on Theoretical Aspects of Computer Science* (1992) 377–388.

[31] V. Kann, On the approximability of NP-complete optimization problems, Ph.D. Thesis, Department of Numerical Analysis and Computing Science, Royal Institute of technology, Stockolm, 1992.

[32] V. Kann, Polynomially bounded minimization problems which are hard to approximate, in: *Proc. 20th Internat. Colloq. on Automata, Languages and Programming* (1993) 52–63.

[33] D. Karger, R. Motwani and G.D.S. Ramkumar, On approximating the longest path in a graph, in: *Proc. 3rd Workshop on Algorithms and Data Structures* (1993) 421–432.

[34] S. Khanna, R. Motwani, M. Sudan and U. Vazirani, On syntactic versus computational views of approximability, in *Proc. 35th Ann. IEEE Symp. on Foundations of Computer Science* (1994) 819–836.

[35] P.G. Kolaitis and M.N. Thakur, Logical definability of NP optimization problems, Tech. Report CRL 90-48, University of California, Santa Cruz, 1990.

[36] P.G. Kolaitis and M.N. Thakur, Approximation properties of NP minimization classes. in: *Proc. 6th Structure in Complexity Theory* (1991) 353–366.

[37] B. Korte and R. Schrader, On the existence of fast approximation schemes, in: *Nonlinear Programming* (Academic Press, New York, 1981) 415–437.

[38] M.W. Krentel, The complexity of optimization problems, *J. Comput. Systems Sci.* **36** (1988) 490–509.

[39] C. Lund and M. Yannakakis, On the hardness of approximating minimization problems, in: *Proc. 25th Ann. ACM Symp. on Theory of Computing* (1993) 286–293.

[40] A. Marchetti-Spaccamela and S. Romano, On different approximation criteria for subset product problems, *Inform. Process. Lett.* **21** (1985).

[41] R. Motwani, Lecture notes on approximation algorithms, Tech. Report STAN-CS-92-1435, Department of Computer Science, Stanford University, 1992.

[42] P. Orponen and H. Mannila, On approximation preserving reductions: complete problems and robust measures, Tech. Reprot C-1987-28, Department of Computer Science, University of Helsinki, 1987.

[43] A. Panconesi and D. Ranjan, Quantifiers and approximation, in: *Proc. 22nd Ann. ACM Symp. on Theory of Computing* (1990) 446–456.

[44] C.H. Papadimitriou, *Computational complexity* (Addison-Wesley, Reading, MA, 1993).

[45] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity* (Prentice-Hall, Englewood Cliffs, NJ. 1982).

[46] C.H. Papadimitriou and M. Yannakakis, Optimization, approximation, and complexity classes, *J. Comput. Systems Sci.* **43** (1991) 425–440.

[47] C.H. Papadimitriou and M. Yannakakis, The traveling salesman problem with distances one and two, *Math. Oper. Res.* 1992.

[48] A. Paz and S. Moran, Non deterministic polynomial optimization problems and their approximations. *Theoret. Comput. Sci.* **15** (1981) 251–277.

[49] S. Sahni, Approximate algorithms for the 0/1 knapsack problem. *J. ACM* **22** (1975) 115–124.

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