INFORMATION AND COMPUTATION 93, 241–262 (1991)

Completeness in Approximation Classes

PIERLUIGI CRESCENZI*

Dipartimento di Matematica Pura ed Applicata, Università degli Studi dell'Aquila, Coppito, L'Aquila, Italy

AND

ALESSANDRO PANCONESI*

Department of Computer Science, Cornell University, Ithaca, New York 14850

We introduce a formal framework for studying approximation properties of NP optimization (NPO) problems. The classes of approximable problems we consider are those appearing in the literature, namely the class of approximable problems within a constant ε (APX), and the class of problems having a polynomial time approximation scheme (PTAS). We define natural approximation preserving reductions and obtain completeness results in NPO, APX, and PTAS. A complete problem in a class cannot have stronger approximation properties unless $P = NP$. We also show that the degree structure of NPO allows intermediate degrees, that is, if $P \neq NP$, there are problems which are neither complete nor belong to a lower class. © 1991 Academic Press, Inc.

1. Introduction

The widespread belief that NP-complete problems cannot be solved by polynomial-time algorithms made researchers look for strategies other than exact resolution in order to deal with these problems. Since many of the most important NP-complete problems are *the recognition versions* of optimization problems, it is natural to ask the following question: "Can we devise polynomial time algorithms which always find solutions *close* to the optimum?"

Several results are known on the approximability or nonapproximability of the so-called NP optimization problems (optimization problems whose recognition version is in NP) when the quality of the approximation is measured by the relative error. In particular four classes have been identified (Garey and Johnson, 1979; Papadimitriou and Steiglitz, 1982):

* This work was partly done when both authors were at Dipartimento di Informatica e Sistemistica, Via Buonarroti 12, 00185 Roma, Italy.

(i) Problems which are not approximable in polynomial time unless $P = NP$;

(ii) APX: problems which are approximable within some fixed relative error $\varepsilon > 0$;

(iii) PTAS: problems which can be approximated within *any* ε by algorithms having as input an instance x and ε. Such algorithms are called Polynomial-Time Approximation Schemes and their complexity must be polynomial in $|x|$ for each fixed ε.

(iv) FPTAS: problems which can be approximated by Polynomial-Time Approximation Schemes whose running time is polynomial in both the size of the input and $1/\varepsilon$. These algorithms are called Fully Polynomial-Time Approximation Schemes.

In spite of some remarkable attempts the reasons that a problem is approximable or nonapprximable are still not clear (Ausiello, d'Atri, and Protasi, 1980; Ausiello, Marchetti Spaccamela, and Protasi, 1980; Bruschi, Joseph, and Young, 1989; Krentel, 1988; Korte and Schrader, 1981; Orponen and Mannila, 1987; Paz and Moran, 1981). In this paper we are interested in the problem of determining *lower bounds* concerning the approximability of NPO problems; that is, we would like to develop techniques that would allow us to prove statements such as "If $P \neq NP$ then problem F is not in PTAS" or "If $P \neq NP$ then problem F is not in APX" and so on. Generally, results of this kind have been obtained via polynomial reductions mapping an NP-complete problem into the given optimization problem and showing that the approximability of the latter would imply the former to be in P. For the class FPTAS, Garey and Johnson (1979) have developed an alternative approach based on the notion of *strong* NP-*completeness*; an optimization problem whose recognition version is strong NP-complete is not in FPTAS. Roughly speaking, a problem is strong NP-complete if its NP-completeness does not depend on the presence of large weights in the input instance. Here, "large" means nonpolynomial in the length of the instance. In order to derive similar criteria for the other classes it seems natural to try to define suitable concepts of completeness. This approach has been taken by several authors (Ausiello, d'Atri, and Protasi, 1980; Krentel, 1988; Orponen and Mannila, 1987; Paz and Moran, 1981). In (Krentel, 1988), NPO problems are treated as functions of the kind $f\colon \Sigma^* \to N$ and a complextity measure is introduced along with a reducibility that preserves this measure. The complexity of a function f is given by the number of queries necessary for a P^{SAT} machine to compute it. The results are elegant but are not related to approximation. The reason is that MIN TSP and MAX KNAPSACK both turn out to be complete for NPO, and while the first is nonapproximable unless P = NP, the second enjoys a fully polynomial time approximation scheme. In general, all the classes of complete problems introduced in (Krentel, 1988) have both approximable and nonapproximable problems.

In (Orponen and Manilla, 1987), several natural problems are shown to be NPO complete with respect to some kind of approximation preserving reduction. In (Papadimitriou and Yannakakis, 1988), a class of approximable problems is introduced and completeness results are proven.

In the present paper we continue along the same line of research. We introduce natural approximation preserving reductions and show the existence of complete problems both in the clases APX and in the class PTAS.

Perhaps surprisingly, in spite of the fact that $\mathrm{APX} \supseteq \mathrm{PTAS}$ and the reductions we use are very natural, reducibility in PTAS is not a refinement of that in APX.

Besides, one of the most relevant results in the paper shows that in all classes NPO, APX and PTAS there exist intermediate problems which are neither complete nor in the lower class. This answers a question posed in (Orponen and Mannila, 1987) on the existence of incomplete problems in the approximation classes. The significance of this result can be explained in the following way; usually a problem F in a class, say NPO, is proved not to be in a lower class, say APX, by proving a statement such as "if $F \in \mathrm{APX}$ then P = NP". The existence of incomplete problems shows that a proof of nonapproximability does not imply completeness in NPO, and similarly for the other classes.

Thus our notion of completeness 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 or not, complete or incomplete (in our sense).

The paper is organized as follows. In Section 2 we formally define the classes of approximable and nonapproximable optimization problems and we introduce special kinds of reduction and, hence, of completeness. In Sections 3 and 4 we prove completeness results in APX and PTAS; as a by-product we are able to show that reducibility in PTAS is not a refinement of reducibility in APX. In Section 5 we show the existence of incomplete problems with the well known delayed diagonalization technique.

2. A Formal Framework for Optimization Problems

Every known NPO problem F can be characterized by the following objects:

- A set $I_F \subseteq \Sigma^*$ of *input instances*; I_F is recognizable in polynomial time.

• A space $D_F(x)$ of *feasible solutions* on input x. $D_F(x)$ is defined by means of a polynomial predicate $\pi_F(x, y)$ and a polynomial $q_F(n)$:

$$D_F(x) = \{ y \mid x \in I_F \wedge \pi_F(x, y) = \text{true} \wedge |y| \leqslant q_F(|x|) \}.$$

We assume, without loss of generality, that every input instance has some feasible solution; that is, for all $x \in I_F$, $D_F(x) \neq \emptyset$.

• An *objective function* $f_F: I_F \times D_F(x) \to Q^+ \cup \{0\}$; $f_F(x, y)$ is computable in polynomial time.

The *optimum on input* x (max or min) of the optimization problem F is

$$\text{opt}_F(x) = \text{opt}\{ f_F(x, y) \mid y \in D_F(x) \}.$$

As an example to explain the previous definition, consider MAX CLIQUE: the set of input instances I_{CLQ} is the set of all (strings encoding) undirected graphs $G = (V, E)$. The polynomial predicate $\pi_{\text{CLQ}}(G, G')$ is "Is G a graph and G' a clique of G?". Thus $D_{\text{CLQ}}(G)$ is the set of all cliques contained in G. Finally, the o.f. is $f_{\text{CLQ}}(G, G') = \|V'\|$.

Without any loss of generality we can assume that the predicate π_F incorporates the test "$x \in I_F$?" so we can define

DEFINITION 1. An NPO-*problem* F is a triple $F = (q_F, \pi_F, f_F)$ where: (i) $q_F(n)$ is a polynomial; (ii) $\pi_F(x, y)$ is a polynomial-time decidable predicate; (iii) the set $\{x \mid \text{there is a } y \text{ such that } \pi_F(x, y)\}$ is in P; (iv) $f_F: I_F \times D_F(x) \to Q^+ \cup \{0\}$ is a polynomial-time computable function.

Definition 1 allows us to associate to any NPO problem F a nondeterministic Turing machine (NTM) N_F defined as follows:

```
**guess** $y \in \{0, 1\}^{q_F(|x|)}$
**if** $\pi_F(x, y) = \text{FALSE}$ **then abort**
  **else** output $f_F(x, y)$
```

The above machine has the property that, for every x, $y \in D_F(x)$ iff there is a computation path with output $f_F(x, y)$. This characterization will turn out to be useful for proving completeness. Our characterization of NPO in terms of NDTM's is different from that of (Krentel, 1988). There, NPO problems are considered to be functions of the kind $f: \Sigma^* \to N$; a function f is in NPO if there exists a NDTM N_f such that, for all x,

$$f(x) = \max\{ y \mid y \text{ is the output of a computation path of } N_f \text{ on input } x \}.$$

In this characterization there is no clear notion of the set $D_F(x)$ of feasible solutions and of the objective function f_F.

Going back to our example, the NDTM for MAX CLIQUE is the following:

```
guess a subgraph $G' = (V', E')$ of $G$
if $G'$ is not a clique then abort
  else output $\|V'\|$
```

In the sequel we consider only maximization problems. The same results hold in the case of minimization problems, provided we do minor modifications to our definitions and proofs.

In order to define classes of approximable NPO problems we need the notion of relative error; the following is a widely used definition restated by means of our notation (Johnson, 1974; Garey and Johnson, 1979; Papadimitriou and Steiglitz, 1982).

Definition 2. Let F be an NPO problem. Given $x \in I_F$, for any $y \in D_F(x)$ the *error* of y with respect to F is

$$\mathscr{E}(F(x), y) = \begin{cases} 0 & \text{if } \operatorname{opt}_F(x) = 0 \\ \dfrac{\operatorname{opt}_F(x) - f_F(x, y)}{\operatorname{opt}_F(x)} & \text{otherwise.} \end{cases}$$

This definitions is suited for the maximization case. It should be modified for the minimization problems.

We can now define classes of approximable problems. With $(0, 1)_Q$ we indicate the set of rationals in $(0, 1)$.

Definition 3. An NPO problem F is in APX if there exist an $\varepsilon \in (0, 1)_Q$ and a polynomial time DTM T such that, for any $x \in I_F$, (i) $T(x) \in D_F(x)$ and (ii) $\mathscr{E}(F(x), T(x)) \leqslant \varepsilon$.

Definition 4. T is a *Polynomial-Time Approximation Scheme* (ptas) for F if, for every input (x, ε) such that $x \in I_F$, (i) $T(x, \varepsilon) \in D_F(x)$; (ii) $\mathscr{E}(F(x), T(x, \varepsilon)) \leqslant \varepsilon$; (iii) T's complexity is $h_T(x, \varepsilon)$, where h_T is polynomial in $|x|$ and arbitrary in $1/\varepsilon$.

The definition of h_T allows to have complexities such as $|x|^{1/\varepsilon}$ or $k^{1/\varepsilon}|x|$, which arise in practice (see for example (Hochbaum and Shmoys, 1987; Papadimitriou and Steiglitz, 1988).

Definition 5. An NPO problem F is in PTAS if there exists a ptas T_F for it.

DEFINITION 6. An NPO problem F is in FPTAS if there exists a ptas T_F for it whose complexity is $h_T(x, \varepsilon) = p(x, 1/\varepsilon)$ where p is a polynomial in both $|x|$ and $1/\varepsilon$.

Clearly the following inclusions hold: NPO $\supseteq$ APX $\supseteq$ PTAS $\supseteq$ FPTAS; it is also well known that these inclusions are strict given the hypothesis P $\neq$ NP.

In order to introduce the notion of completeness for our classes, we now define three kinds of reduction between problems. All of them are refinements of the following

DEFINITION 7. Given $F, G \in$ NPO a reduction from F to G is a triple (t_1, t_2, c), where

— $t_1 : I_F \to I_G$ is polynomially computable;

— $t_2(x, y)$ is a polynomially computable function such that if $y \in D_G(t_1(x))$ then $t_2(x, y) \in D_F(x)$;

— $c: (0, 1)_Q \to (0, 1)_Q$.

Roughly speaking, if we want to map F into G we use t_1 to map instances of F into instances of G, and t_2 to map back approximated solutions of G into approximated solutions of F. The role of c is that of preserving the quality of the approximation.

The first reduction, called A-reduction, is needed to introduce completeness in NPO.

DEFINITION 8. Let F, G be two NPO problems; F is said to be A-*reducible* to G, in symbols $F \leqslant_A G$, if there exists a reduction from F to G such that, for any $y \in D_G(t_1(x))$:

$$\mathscr{E}(G(t_1(x)), y) \leqslant \varepsilon \Rightarrow \mathscr{E}(F(x), t_2(x, y)) \leqslant c(\varepsilon).$$

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

PROPOSITION 1. *If F is* A-*reducible to G and $G \in$* APX, *then $F \in$* APX.

The above definition allows us to define the notion of NPO-completeness.

DEFINITION 9. A problem $G \in$ NPO is NPO-complete if, for any $F \in$ NPO, $F \leqslant_A G$.

Analogously, we define reductions for the classes APX and PTAS. The only difference between the reduction in NPO and that in APX concerns the role of the mapping c:

DEFINITION 10. Let F, G be two NPO problems; F is said to be P-*reducible* to G, in symbols $F \leqslant_{\mathrm{P}} G$, if there exists a reduction from F to G such that, for any $y \in D_G(t_1(x))$,

$$\mathscr{E}(G(t_1(x)), y) \leqslant c(\varepsilon) \Rightarrow \mathscr{E}(F(x), t_2(x, y)) \leqslant \varepsilon.$$

This definition is more general than that used in (Papadimitriou and Yannakis, 1980), but it serves the same purpose; namely, it preserves membership in PTAS.

PROPOSITION 2. *If $F \leqslant_{\mathrm{P}} G$ and $G \in$ PTAS then $F \in$ PTAS.*

DEFINITION 11. A problem $G \in$ APX is APX-*complete* if, for any $F \in$ APX, $F \leqslant_{\mathrm{P}} G$.

In (Orponen and Mannila, 1987), reductions equivalent to $\leqslant_{\mathrm{A}}$ and $\leqslant_{\mathrm{P}}$ are defined but completeness with respect to the latter is not proved.

To define completeness in PTAS we need to modify the function c substantially

DEFINITION 12. Let F, G be two NPO problems; F is said to be F-*reducible* to G, in symbols $F \leqslant_{\mathrm{F}} G$, if there exist three functions t_1, t_2, c such that

(i) t_1, t_2 are as in Definition 7,

(ii) $c: (0, 1)_Q \times I_F \rightarrow (0, 1)_Q$

(iii) for any $y \in D_G(t_1(x))$ if $\mathscr{E}(G(t_1(x), y) \leqslant c(\varepsilon, x)$ then $r\mathscr{E}(F(x), t_2(x, y)) \leqslant \varepsilon$,

(iv) the time complexity of c is $p(1/\varepsilon, |x|)$, where p is a polynomial, and

(v) the value of c is $1/q(1/\varepsilon, |x|)$, where q is a polynomial.

DEFINITION 14. A problem $G \in$ PTAS is PTAS-complete if, for any $F \in$ PTAS, $F \leqslant_{\mathrm{F}} G$.

The definition above satisfy the following proposition.

PROPOSITION 3. *If $F \leqslant_{\mathrm{F}} G$ and $G \in$ FPTAS then $F \in$ FPTAS.*

The following can also be easily proved.

PROPOSITION 4. *The defined reductions are reflexive and transitive.*

The reductions we defined are quite natural, nevertheless they are related in a strange way. As APX $\supseteq$ PTAS it would seem reasonable to assert

"$F \leqslant_{\mathrm{F}} G$ implies $F \leqslant_{\mathrm{P}} G$"; in other words, the F-reduction should be definable as a P-reduction with some additional constraint. But this is not the case; in Section 4 we will show that, surprisingly, *any* APX problem is F-reducible to a PTAS-complete problem.

3. APX Completeness

The aim of this section is to show the APX-completeness of the following ptoblem, Bounded SAT (BSAT):

Bounded SAT (BSAT).

Instance. A boolean formula φ with variables $x_1, ..., x_n$ of weights $w_1, ..., w_n$ and a separate weight W. The weights must satisfy

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

Problem. Maximize the following function, defined on the assignments of φ:

$$f_{\mathrm{BSAT}}(\varphi, \tau) = \begin{cases} W & \text{if } \varphi(\tau(x_1), ..., \tau(x_n)) = \text{FALSE} \\ \sum_{i=1}^{n} w_i \tau(x_i) & \text{otherwise.} \end{cases}$$

Observe that BSAT can be approximated trivially with error $\frac{1}{2}$ (the assignment $x_i = 1$, $1 \leqslant i \leqslant n$, has value either W or $\sum_{i=1}^{n} w_i$), and that approximating it with lower error is NP-hard. The unbounded version of BSAT, Max Weighted VAR, has been proven NPO-complete with respect to approximation preserving reductions by (Orponen and Manilla, 1987; Paz and Moran, 1981).

In order to understand the result concerning BSAT, we first consider the case of NPO-completeness. For a class that has a machine representation, a common way of showing completeness is to define a universal machine for the class. This method also works for NPO. If we consider tuples such as $X = (x, N_F, 0^k)$, where $N_F = (q_F, \pi_F, f_F)$, we can define a "universal" problem U_n, where $\pi_U(X, y)$ is "simulate $\pi_F(x, y)$ for k steps; if k is too little, reject," and $f_U(X, y)$ is "simulate $f_F(x, y)$ for k steps; if k is too little, output 0." Clearly, if k is big enough to simulate nondeterministically every branch of N_F we have that U_N is *exactly the same* as F. As a consequence, if we define $t_1(x) = (x, N_F, 0^k)$, $k = p_F(|x|)$, where p_F is N_F's polynomial time bound, and t_2 as the identity function we can easily show that $\mathscr{E}(U_N(t_1(x)), y) = \mathscr{E}(F(x), t_2(y))$; NPO-completeness follows.

However, much more can be proved; by modifying Cook's proof of the NP-completeness of SAT slightly, it is possible to prove NPO-completeness for Max Weighted Var (Krentel, 1988; Paz and Moran, 1981). By modifying other NP-completeness proofs, natural problems can be proved NPO-complete. For example, the following theorem is from Orponen and Mannila (1987).

THEOREM 1. TSP *and* 01-*Integer Programming are* NPO-*complete.*

In APX the situation is considerably different because we do not have a machine model to simulate. How can we enumerate the class? $F \in$ APX if there are a NDTM for it *and* a polynomial time T approximating F. The right idea seems be to to consider tuples such as $(x, \varepsilon, T, N_F, 0^k)$, where T is an ε-approximating algorithm for N_F. In doing so we have to face two kinds of problems: (i) we cannot know in advance whether T approximates N_F; (ii) since a problem in APX can be approximated by any $\varepsilon \in (0, 1)$ we have to "map" all the ε's into one fixed ε_0.

We start by defining our "universal" problem U_A by means of a nondeterminsitic algorithm; from its definition it will be clear that U_A is a NPO problem.

The inputs for the algorithm are of the form $X = (x, \varepsilon, T, N_F, 0^k)$, where

— T is a polynomial-time deterministic TM,

— N_F is a NPO problem, that is, a triple $N_F = (q_F, \pi_F, f_F)$,

— 0^k is a padding of k 0's, ε is a rational in $(0, 1)$, and x is an input for T and N_F.

On input $X = (x, \varepsilon, T, N_F, 0^k)$ the machine for U_A performs the following nondeterministic algorithm. The algorithm is divided in two parts: the first, the Trunk, is deterministic while the second, the Branches, is nondeterministic.

DEFINITION OF U_A.

Trunk. For k steps do the following: if $\pi_F(x, T(x)) = \text{TRUE}$ then set $t = f_F(x, T(x))$. If k is too small abort computation (X is not a valid input instance).

Branches. Simulate *nondeterministically* k steps of N_F. If k is too small output the value $A(X) + t$; otherwise output $A(X) + \min\{t/(1-\varepsilon), f_F(x, y)\}$. Note that in the latter case $y \in D_F(x)$. The value $A(X)$ is a nonnegative, polynomially computable function whose value will be specified later.

Note that if we perform the Trunk, U_A outputs at least the value $A(X) + t$; moreover $\text{opt}_{U_A}(X)$, U_A's optimum value, is bounded in terms of

t. In the next lemma we show how $A(X)$ and t are used to show APX membership of U_A.

LEMMA 1. *U_A is in* APX.

Proof. In this lemma we specify the value of $A(X)$, which appeared in the above definition of U_A. We want to show that the following is a $\frac{1}{2}$-approximated algorithm for U_A:

if k is big enough to perform the Trunk, output $T(x)$.

What we have to show is that $\mathscr{E}(U_A(X), T(x)) \leqslant \frac{1}{2}$.

Let $m(X) = A(X) + t$ and $M(X) = A(X) + t/(1-\varepsilon)$. By definition of U_A we have $m(X) \leqslant \mathrm{opt}_{U_A}(X) \leqslant M(X)$ and hence

$$\mathscr{E}(U_A(X), T(x)) \leqslant \frac{M(X) - m(X)}{M(X)} = t\frac{\varepsilon}{A(X)(1-\varepsilon) + t} \leqslant \varepsilon.$$

In order to define $A(X)$ we distinguish two cases:

(i) $[\varepsilon \leqslant \frac{1}{2}]$. We set $A(X) = 0$, we have $\mathscr{E}(U_A(X), T(x)) \leqslant \varepsilon \leqslant \frac{1}{2}$.

(ii) $[\varepsilon > \frac{1}{2}]$. In this case $A(X)$ is "responsible" for the ratio $(M(X) - m(X))/M(X)$ being less than or equal to $\frac{1}{2}$. We define $A(X)$ in such a way that

$$\frac{M(X) - m(X)}{M(X)} = t\frac{\varepsilon}{A(X)(1-\varepsilon) + t} = \frac{1}{2},$$

that is,

$$A(X) = t\frac{2\varepsilon - 1}{1 - \varepsilon}.$$

Thus $A(X)$ is polynomially computable in $|X|$ and we know it is always the case that $\mathscr{E}(U_A(X), T(x)) \leqslant \frac{1}{2}$. ∎

LEMMA 2. *U_A is* APX-*complete.*

Proof. Given any $F \in \mathrm{APX}$ there must be a pair T, δ such that T δ-approximates F. Let p_T, p_F be the polynomial bounds of T and N_F. We have to exhibit a P-reduction $F \leqslant_{\mathrm{P}} U_A$. Given x define $k = \max\{p_F(|x|), p_T(|x|)\}$, and

$$t_1(x) = (x, \delta, T, N_F, 0^k) \overset{\mathrm{def}}{=} X.$$

The size of k is big enough to simulate N_F, hence $D_{U_A}(X) = D_F(x)$; as a consequence we can define t_2 to be the identity $t_2(x, y) = y$.

By our choice of k and since T δ-approximates F, we have $\operatorname{opt}_{U_A}(X) \leqslant A(X) + \operatorname{opt}_F(x)$ and hence

$$\mathscr{E}(U_A(X), y) = \frac{\operatorname{opt}_F(x) - f_F(x, y)}{A(X) + \operatorname{opt}_F(x)}.$$

From the definition of P-reduction we have to find a function $c(\varepsilon)$ that for all ε's satisfies

$$\mathscr{E}(U_A(X), y) \leqslant c(\varepsilon) \Rightarrow \mathscr{E}(F(x), y) \leqslant \varepsilon.$$

Suppose then that

$$\mathscr{E}(U_A(X), y) = \frac{\operatorname{opt}_F(x) - f_F(x, y)}{A(X) + \operatorname{opt}_F(x)} \leqslant c(\varepsilon).$$

We want to define $c(\varepsilon)$ in order to satisfy the above implication.

From the definition of $A(X)$ we consider two cases. If $A(X) = 0$ it is enough to set $c(\varepsilon) = \varepsilon$; otherwise (we simplify the last inequality)

$$\begin{aligned}\mathscr{E}(F(x), y) &\leqslant c(\varepsilon)\left(\frac{A(X)}{\operatorname{opt}_F(x)} + 1\right)\\ &= c(\varepsilon)\left(\frac{t}{\operatorname{opt}_F(x)}\frac{2\delta - 1}{1 - \delta} + 1\right)\\ &\leqslant c(\varepsilon)\frac{\delta}{1 - \delta}.\end{aligned}$$

Then, in order to have $\mathscr{E}(F(x), y) \leqslant \varepsilon$ we set

$$c(\varepsilon) = \frac{1 - \delta}{\delta}\,\varepsilon.$$

Since $(1 - \delta)/\delta$ is a constant of the reduction process the above is a valid P-reduction. ▌

We can now state

THEOREM 2. BSAT *is* APX-*complete.*

Proof. We have already observed that BSAT is in APX. We reduce U_A to BSAT via a P-reduction $R = (t_1, t_2, c)$.

Let $X=(\delta, x, T, N, 0^k)$ be the given instance of U_A. Recall that, by definition of U_A, $m(X)=A(X)+t \leqslant \mathrm{opt}_{U_A}(X) \leqslant A(X)+t/(1-\delta)=M(X)$, and that, by definition of $A(X)$, $M(X) \leqslant 2m(X)$.

We know that, for all feasible solutions $y \in D_{U_A}(X)$, $f_{U_A}(X, y) \leqslant M(X)$. This implies that to output any value $f_{U_A}(X, y)$ we need at most $p+1$ tape cells $v_0, ..., v_p$, where $p=\lfloor \log M(X) \rfloor$.

By Cook's construction, given X and the NDTM for U_A, we can construct in polynomial time a boolean formula ϕ_X such that

$$y \in D_{U_A}(X) \wedge f_{U_A}(X, y)=v \Leftrightarrow \phi_X(\tau_y(y', v'))=\mathrm{TRUE},$$

where τ_y is an assignment to the set of variables y' and $v'=\{v'_0, ..., v'_p\}$. From the construction, it also follows that it is possible to recover y from τ_y in polynomial time.

We define the boolean formula of the BSAT instance to be $\varphi = z \wedge \phi_X$, where z is a new variable. The only nonzero weights are $w(v'_i)=w_i=2^i$ and $w(z)=2M(X)$. Finally, we set the value W to be equal to $2M(X)$. This ends the construction of $t_1(X)$. Note that we have a (one-to-one and onto) correspondence between feasible solutions y of U_A of cost v and satisfying assignments τ_y for φ of cost $W+v$. What we constructed is a valid BSAT instance, since

$$W \leqslant W+\sum_{i=0}^{m} w_i \leqslant W+2M(X)=2W.$$

We define

$$t_2(X, \tau_y)=\begin{cases} T_A(X) & \text{if} \quad \varphi(\tau_y(z, y', v'))=\mathrm{FALSE} \\ y & \text{otherwise} \end{cases}$$

where T_A is the $\frac{1}{2}$-approximated algorithm for U_A, and where y is the feasible solution of U_A computed from τ_y. Finally, we define, for any $\varepsilon<1$, $c(\varepsilon)=\varepsilon/5$.

To complete the proof we have to show that, for any $\varepsilon<1$ and for any assignment τ of φ,

$$\mathscr{E}(\mathrm{BSAT}(\varphi), \tau) \leqslant c(\varepsilon) \Rightarrow \mathscr{E}(U_A(X), t_2(X, \tau)) \leqslant \varepsilon.$$

If $\varphi(\tau)=\mathrm{FALSE}$ then $\mathscr{E}(\mathrm{BSAT}(\varphi), \tau) \geqslant \frac{1}{5} > c(\varepsilon)$, and hence the implication is trivially satisfied. Otherwise, we have that $f_{\mathrm{BSAT}}(\varphi, \tau_y)= W+f_{U_A}(X, y)=W+f_{U_A}(X, t_2(X, \tau_y))$ and hence

$$\mathscr{E}(\mathrm{BSAT}(\varphi), \tau_y)=\frac{\mathrm{opt}_{U_A}(X)-f_{U_A}(X, t_2(X, \tau_y))}{W+\mathrm{opt}_{U_A}(X)} \leqslant c(\varepsilon),$$

which implies

$$\begin{aligned}\mathscr{E}(U_A(X), t_2(X, \tau_y)) &= \frac{\mathrm{opt}_{U_A}(X) - f_{U_A}(X, t_2(X, \tau_y))}{\mathrm{opt}_{U_A}(X)}\\ &\leqslant \frac{\varepsilon}{5}\left(\frac{W}{\mathrm{opt}_{U_A}(X)} + 1\right)\\ &\leqslant \frac{\varepsilon}{5}\left(\frac{2M(X)}{m(X)} + 1\right)\\ &\leqslant \varepsilon. \quad \blacksquare\end{aligned}$$

4. PTAS Completeness

According to the same line of the preceding section we will show that the following problem, Linear BSAT (LBSAT), is PTAS-complete:

Linear BSAT (LBSAT).

Instance. A boolean formula φ with variables $x_1, ..., x_n$ of weights $w_1, ..., w_n$ and a separate weight W. The weights must satisfy

$$W \leqslant \sum_{i=1}^{n} w_i \leqslant \left(1 + \frac{1}{n-1}\right) W.$$

Problem. Maximize the following function, defined on the assignments of φ:

$$f_{\mathrm{LBSAT}}(\varphi, \tau) = \begin{cases} W & \text{if } \varphi(\tau(x_1), ..., \tau(x_n)) = \mathrm{FALSE} \\ \sum_{i=1}^{n} w_i \tau(x_i) & \text{otherwise.} \end{cases}$$

We can modify the definition of U_A to get a complete problem for PTAS; let us call it U_P. U_P is defined in the same way as U_A except that in the Trunk, on input $X = (x, \delta, T, N_F, 0^k)$, $T(x, \delta)$ is simulated instead of $T(x)$. In order to show membership in PTAS for U_P we have to modify the function $A(X)$; this is done in the next lemma.

Lemma 3. *U_P is in* PTAS.

Proof. We have to show that U_P has a polynomial-time approximation scheme, that is, an algorithm T that given any ε is able to ε-approximate U_P in time $h_T(X, \varepsilon)$, where h_T is polynomial in $|X|$ and arbitrary in $1/\varepsilon$. The way to obtain this is to define $A(X)$ in such a way that if we want to approximate U_P with an error smaller than $1/|X|$ then $1/\varepsilon$ is so large as to allow a deterministic simulation of the NDTM for U_P in polynomial time.

As in Lemma 1, let $M(X) = A(X) + t/(1-\delta)$ and $m(X) = A(X) + t$.

Polynomial Time Approximation Scheme for U_P.

Input. $X = (x, \delta, T, N_F, 0^k)$ and ε' is the *wanted approximation.*

- If k is large enough to perform the Trunk compute $E = (M(X) - m(X))/M(X)$; abort otherwise (X is not a valid input).
- If $E \leqslant \varepsilon'$ then output $T(x, \delta)$; otherwise simulate U_P on input X *deterministically* and print an optimal solution y^*, i.e., $U_P(X, y^*) = \mathrm{opt}_{U_P}(X)$.

We now show that for any input (X, ε') this algorithm approximates U_P within ε' in time polynomial in $|X|$ and exponential in ε'. Note that, similarly to the U_A case, $\mathscr{E}(U_P(X), T(x, \delta)) \leqslant E$. There are two possibilities; the first is when $E \leqslant \varepsilon'$. In this case, since $\mathscr{E}(U_P(X), T(x, \delta)) \leqslant E$, and the complexity of the Trunk is linear, the algorithm works.

The second case is when $E > \varepsilon'$. We shall define $A(X)$ to manage this case properly. A few calculations show that $E = \delta t/(A(X)(1-\delta) + t)$; if we define $A(X)$ in such a way that $E = 1/|X|$, that is,

$$A(X) = \frac{t(\delta\,|X| - 1)}{1-\delta},$$

we have that $E > \varepsilon' \Rightarrow 1/\varepsilon' > |X|$. Clearly, $A(X)$ is polynomially computable.

Note now that in order to simulate U_P deterministically we need $k2^k \leqslant |X|\,2^{|X|} < |X|\,2^{1/\varepsilon'}$ steps. In other words, if $E > \varepsilon'$ we can simulate U_P deterministically in time polynomial in $|X|$ and exponential in $1/\varepsilon'$. Hence our algorithm is a polynomial-time approximation scheme. ∎

THEOREM 3. *U_P is* PTAS-*complete.*

Proof. Given any $F \in \mathrm{PTAS}$ we have to exhibit an F-reduction witnessing $F \leqslant_F U_P$. Let T_F be the polynomial time approximation scheme for F and p_F, h_T the complexities of N_F and T_F. Define $X = t_1(x) = (x, \frac{1}{2}, T_F, N_F, 0^k)$, where $k = \max\{p_F(|x|), h_T(\frac{1}{2}, |x|)\}$. Note the complexity given by h_T; by our choice of $\delta = \frac{1}{2}$, k is a polynomial value in $|x|$ no matter what h_T is (any other fixed rational in (0, 1) would be as good as $\frac{1}{2}$). This choice also implies that $A(X) = t(|X| - 2)$. The other two parts of the F-reduction are defined as

— $t_2(x, y) = y$; recall that by our choice of k we have enough time to simulate both T_F and N_F and hence $D_{U_P}(X) = D_F(x)$;

— $c(x, \varepsilon) = \varepsilon/(|X| - 1)$. This is a valid definition for c since both its complexity and its value are polynomial in $1/\varepsilon$ and $|X|$.

We have now to show that, for every ε,

$$\mathscr{E}(U_P(X), y) \leqslant c(x, \varepsilon) \Rightarrow \mathscr{E}(F(x), t_2(x, y)) \leqslant \varepsilon.$$

Simple calculations yield

$$\mathscr{E}(U_P(X), y) = \frac{\text{opt}_F(x) - f_F(x, y)}{A(X) + \text{opt}_F(x)}.$$

Then saying that

$$\mathscr{E}(U_P(X), y) \leqslant c(x, \varepsilon)$$

is equivalent to saying that

$$\begin{aligned}\mathscr{E}(F(x), t_2(x, y)) &= \frac{\text{opt}_F(x) - f_F(x, y)}{\text{opt}_F(x)} \\ &\leqslant c(x, \varepsilon)\left(\frac{A(X)}{\text{opt}_F(x)} + 1\right) \\ &\leqslant c(x, \varepsilon)\left(\frac{A(X)}{t} + 1\right).\end{aligned}$$

Substituting in the last expression the values of $A(X)$ and $c(x, \varepsilon)$ yields our conclusion, namely $\mathscr{E}(F(x), t_2(x, y)) \leqslant \varepsilon$. ▮

COROLLARY 1. LBSAT *is* PTAS-*complete*

Proof. To show LBSAT $\in$ PTAS is very easy. The definition of LBSAT is such that the value W always ensures an error less than or equal to $1/n$. If an approximation $\varepsilon < 1/n$ is wanted then $2^{1/\varepsilon} > 2^n$ and we can find the optimal assignment deterministically in polynomial-time.

We now reduce U_P to LBSAT via an F-reduction $R = (t_1, t_2, c)$. The proof follows very closely that of the APX-completeness of BSAT.

Let $X = (\delta, x, T, N, 0^k)$ be the given instance of U_P. By definition of U_P, $m(X) = A(X) + t \leqslant \text{opt}_{U_P}(X) \leqslant A(X) + t/(1 - \delta) = M(X)$, and, by definition of $A(X)$,

$$\frac{M(X)}{m(X)} = \frac{\|X\|}{\|X\| - 1}.$$

It follows that $M(X) \leqslant 2m(X)$ whenever $\|X\| \geqslant 2$.

Let $p = \lfloor \log M(X) \rfloor$. As in the proof of Theorem 2, we can construct a boolean formula $\varphi = z \wedge \phi_X(y', v')$ in polynomial time that mimics the behaviour of $U_P(X)$. Let n be the number of variables in φ, n is polynomial in $\|X\|$. The only non zero weights of variables in φ are $w(z) = 2(n-1)\, M(X)$ and $w(v_i') = w_i = 2^i$, $1 \leqslant i \leqslant p$. Finally, we set $W = w(z)$. This defines $t_1(X)$, which is a valid LBSAT instance.

As in Theorem 2, there is a bijection between $y \in D_{U_P}$ of cost $f_{U_P}(X, y) = v$ and satisfying assignments τ_y for φ of cost $W + v$. We then define $c(\varepsilon, X) = \varepsilon/(4n-2)$ and

$$t_2(X, \tau_y) = \begin{cases} T(x) & \text{if } \varphi(\tau_y(z, y', v')) = \text{FALSE} \\ y & \text{otherwise.} \end{cases}$$

In order to show that

$$\mathscr{E}(\text{LBSAT}(\varphi), \tau) \leqslant c(\varepsilon, X) \Rightarrow \mathscr{E}(U_P(X), t_2(X, \tau)) \leqslant \varepsilon$$

we distinguish two cases. If τ_y satisfies φ the implication follows from the fact that $f_{\text{LBSAT}}(\varphi, \tau_y) = W + f_{U_P}(X, y) = W + f_{U_P}(X, t_2(X, \tau_y))$. If τ does not satisfy φ then $\mathscr{E}(\text{LBSAT}(\varphi), \tau) \geqslant 1/(4n-2) > c(\varepsilon)$ and the above implication holds trivially. We can conclude that $U_P \leqslant_{\text{P}} \text{LBSAT}$. ∎

Theorem 3 has another interesting corollary.

Definition 14. The *closure* of a class $\mathscr{C} \subseteq \text{NPO}$ with respect to a reduction $\leqslant_{\text{R}}$ is the set

$$C(\mathscr{C}, \leqslant_{\text{R}}) = \{F \mid \exists G \in \mathscr{C} \text{ such that } F \leqslant_{\text{R}} G\}.$$

This is a restatement in our framework of the notion of closure of a class of languages with respect to a reduction.

Corollary 2. *$C(\text{PTAS}, \leqslant_{\text{F}})$ includes* APX.

Proof. Observe the role of $\delta = \frac{1}{2}$ in the proof of Theorem 3; as already pointed out any other rational in $(0, 1)$ could be used. In fact, if $F \in \text{APX}$ can be approximated within some ε_F the proof of Theorem 3 with $\delta = \varepsilon_F$ shows that F is F-reducible to U_P. ∎

The intuitive reason for this to happen is that the function c in the definition of P-reduction must be independent of $|x|$ in order to respect time bounds. On the other hand, in the F-reduction c can be polynomially dependent on $|x|$. For this very same reason we have that the inclusion of Corollary 2 is proper. To see this, pick some $F \in \text{NPO} - \text{APX}$ whose o.f. has value either 0 or 1 (for example, the characteristic function of SAT) and F-reduce it to any PTAS problem with o.f. bounded in terms of $p(|x|)$, where p is a polynomial (for example a suitable version of U_P); the reduction is possible since the definition of F-reduction allows $c(x, \varepsilon) = 1/p(|x|)$.

5. The Existence of Intermediate Degrees

A natural theoretical quastion to ask in our framework is whether, assuming $P \neq NP$, incomplete problems can exist; by using delayed diagonalization (see (Ladner, 1975; Homer, 1986)) we are able to show two different versions of this fact, namely

Theorem 4. *If* $P \neq NP$, *there is a* NPO *problem* INT *such that*

— $\mathrm{INT} \notin \mathrm{APX}$,

— INT *is not* NPO-*complete, and*

— INT *is* APX-*hard via* P-*reductions.*

Theorem 5. *If* $P \neq NP$, *there is a* NPO *problem* INT′ *such that*:

— $\mathrm{INT}' \notin \mathrm{APX}$,

— INT′ *is not* NPO-*complete, and*

— INT′ *is not* APX-*hard via* P-*reductions.*

These and the followng theorems are interesting because they explain the difference between showing that a given problem is not approximable (or, more in general, that is does not have some approximation property) and showing that it is complete. In particular they show that NPO-completeness is not a consequence of the proof of statements such as "if $F \in \mathrm{APX}$ then $P = NP$" usually used to show nonapproximability of problems. Moreover, Theorem 5 shows that there are problems with a quite counterintuitive characteristic; even if they are strong enough to be non-approximable they are not able to represent "weaker" problems, namely those in APX.

Proof of Theorem 4. Let T_i, ε_j, A_k be enumerations of polynomials time deterministic Turing machines, rationals in $(0, 1)$, and A-reductions, respectively. Then let U_N be the NPO-complete problem informally introduced at the beginning of Section 4; since it is possible to reduce every $F \in \mathrm{NPO}$ to U_N in such a way that $\mathscr{E}(U_N(t_1(x)), y) = \mathscr{E}(F(x), t_2(x, y))$ we have that U_N is also APX-hard with respect to P-reductions. Finally, let U_A and T_A be the APX-complete problem of Section 4 with its $\frac{1}{2}$-approximated algorithm.

The theorem relies on the possibility of defining the new problem INT partly as U_N and partly as U_A. INT is defined as U_N in order to be nonapproximable, and as U_A in order to be incomplete.

We are going to define INT in terms of a countably infinite sequence of problems. Let $z_1 < z_2 < \cdots < z_i < \cdots$ be a countably infinite sequence of strings in Σ^*. We start off by defining a sequence of NPO problems:

$$\mathrm{INT}(1, x) = U_A(x)$$

$$\mathrm{INT}(2k, x) = \begin{cases} \mathrm{INT}(2k-1, x) & \text{if } x < z_{2k-1} \\ U_N(x) & \text{if } z_{2k-1} \leqslant x \end{cases}$$

$$\mathrm{INT}(2k+1, x) = \begin{cases} \mathrm{INT}(2k, x) & \text{if } x < z_{2k} \\ U_A(x) & \text{if } z_{2k} \leqslant x. \end{cases}$$

Then, we define INT as follows:

$$\mathrm{INT}(x) = \begin{cases} \mathrm{INT}(2k, x) & \text{if } z_{2k-1} \leqslant x < z_{2k} \\ \mathrm{INT}(2k+1, x) & \text{if } z_{2k} \leqslant x < z_{2k+1}. \end{cases}$$

This definition of INT is equivalent to

$$\mathrm{INT}(x) = \begin{cases} U_N(x) & \text{if } z_{2k-1} \leqslant x < z_{2k} \\ U_A(x) & \text{if } z_{2k} \leqslant x < z_{2k+1}. \end{cases}$$

In order to show that INT satisfies the claim of the theorem we introduce the following "even" predicates:

$C_{2k}(x) = \mathrm{TRUE}$ iff $k = (i, j)$ and $T_i(x)$ does not approximate $\mathrm{INT}(2k, x)$ within ε_j; i.e., $\mathscr{E}(\mathrm{INT}(2k, x), T_i(x)) > \varepsilon_j$.

We know that $C_{2k}(x)$ is true infinitely often. The reason is that U_N is not approximable (if $\mathrm{P} \neq \mathrm{NP}$) and therefore T_i fails to approximate U_N within ε_j for infinitely many x's. Note now that $\mathrm{INT}(2k, x)$ is the same as U_N except on an initial segment of Σ^*.

Similarly, consider the "odd" predicates:

$C_{2k+1}(x) = \mathrm{TRUE}$ iff $A_k = (t_{1k}, t_{2k}, c_k)$ *fails* at x. That is, if $w = t_{1k}(x)$ then $\mathscr{E}(\mathrm{INT}(2k+1, w), T_A(w)) \leqslant \frac{1}{2}$, and $\mathscr{E}(U_N(x), t_{2k}(x, T_A(w))) > c_k(\frac{1}{2})$.

Again, we know that $C_{2k+1}(x)$ is true infinitely often. To see this, note first that $\mathrm{INT}(2k+1, x)$ is the same as $U_A(x)$ almost always, and hence it is approximable within $\frac{1}{2}$ with a polynomial time algorithm that uses T_A and a finite table. If $C_{2k+1}(x)$ were false almost always, we could approximate U_N within $c_k(\frac{1}{2})$ with the algorithm $t_{2k}(x, T_A(t_{1k}(x)))$ and a finite table.

In order to define INT we need a polynomial time TM D such that

- range$(D) = 1^*$,
- $\forall k\, \exists x: C_{2k}(x) = \text{TRUE} \wedge D(x) = 1^{2k}$, and
- $\forall k\, \exists x: C_{2k+1}(x) = \text{TRUE} \wedge D(x) = 1^{2k+1}$.

The proof that such a machine exists is in Lemma 4 below. If we now define

$$z_k = \min\{x \mid D(x) = 1^{k+1}\}$$

we have that

- $\forall k: z_{2k-1} \leqslant x < z_{2k} \Rightarrow D(x) = 1^{2k-1}$, and
- $\forall k: z_{2k} \leqslant x < z_{2k+1} \Rightarrow D(x) = 1^{2k}$.

As a consequence, the following is an equivalent definition of INT:

$$\text{INT}(x) = \begin{cases} U_N(x) & \text{if } |D(x)| \text{ is even} \\ U_A(x) & \text{if } |D(x)| \text{ is odd.} \end{cases}$$

Since D is a polynomial-time DTM, INT is an NPO problem. INT is the problem we were looking for. From the definition of D and the predicates C_k, it follows that INT is neither NPO-complete nor in APX.

Finally, INT is APX-hard via P-reductions because both U_N and U_A are APX-hard, and we can tell which of the two INT(x) is in polynomial time by running $D(x)$. ∎

We now prove the existence of the machine D.

LEMMA 4. *There is a polynomial-time computable function $D(x)$ such that*

- range$(D) = 1^*$,
- $\forall n\, \exists x: C_n(x) = \text{TRUE} \wedge D(x) = 1^n$.

Proof. We inductively define points x_k and z_k:

(i) $x_1 = \min\{x \mid C_1(x) = \text{TRUE}\}$. Set $z_1 = 0^{t_1}$, where t_1 is the time (i.e., number of steps) needed to find x_1 by following this procedure: test $C_1(x)$ for increasingly larger x's.

(ii) $x_k = \min\{x \mid C_k(x) \wedge x \geqslant z_{k-1}\}$. Set $z_k = 0^{t_k}$, where t_k is the time needed to find x_k by following this procedure: test $C_k(x)$ for increasingly larger x's provided that $x \geqslant z_{k-1}$.

Our machine D acts as follows: on input x, for $|x|$ steps look for $z_0, z_1, ...$; let z_k be the largest, output 1^{k+1}.

By definition, D is a polynomial-time Turing machine such that $D(x) = 1^{k+1}$ whenever $z_k \leqslant x < z_{k+1}$. Moreover, for each k we have $z_k \leqslant x_{k+1} < z_{k+1}$. From the discussion of Theorem 4 it follows that there are infinitely many z_k's, hence the claim. ∎

Proof of Theorem 5. In this proof we have to diagonalize over three, rather than two conditions. We need the following ingredients: G, an NPO-complete problem; A, an APX problem which is approximable within $\frac{1}{2}$ by T_A but *not* approximable within $\frac{1}{3}$ (since it does not have to be complete it is easy to construct such a problem); P, a PTAS problem with its polynomial-time approximation scheme T_P; T_i, A_i, P_i, ε_j enumerations of polynomial time deterministic Turing machines, A-reductions, P-reductions, and rationals in (0, 1), respectively. Finally, we need three kinds of predicate; C'_{2k} and C'_{2k+1} are as C_{2k} and C_{2k+1} in Theorem 4. The new predicates C'_{2k+2} ensure that INT′ is not APX-hard:

> $C'_{2k+2}(x) = \text{TRUE}$ iff $P_k = (t_{1k}, t_{2k}, c_k)$ *fails* to reduce A to INT′ at point x. That is, if $w = t_{1k}(x)$ then there exists a $y \in D_{\text{INT}'}(w)$ such that $\mathscr{E}(\text{INT}'(w), y) \leqslant c_k(\frac{1}{3})$ and $\mathscr{E}(A(x), y) > \frac{1}{3}$.

INT′ is defined in the following way:

> On input x, INT′ is the same as G if $|D'(x)| \equiv 0 \pmod 3$, as A if $|D'(x)| \equiv 1 \pmod 3$, and as P if $|D'(x)| \equiv 2 \pmod 3$,

where D' is a polynomial Turing machine defined analogously to the D of Theorem 4, and whose construction is hence omitted. ∎

By similar arguments we can show this general theorem:

THEOREM 6. *Let $\mathscr{X}, \mathscr{Y}$ be two classes among* NPO, APX, PTAS, FPTAS *with $\mathscr{Y} \subseteq \mathscr{X}$. Given $G \in \mathscr{Y}$ and H $\mathscr{X}$-complete it is possible to build F which is not $\mathscr{X}$-complete but belongs to $\mathscr{X} - \mathscr{Y}$. Moreover it is possible to make it either $\mathscr{Y}$-hard or not.*

The proof is omitted because it can be obtained by the same techniques as those of the previous lemma and theorems.

6. Concluding Remarks and Open Problems

We defined natural approximation preserving reductions and proved completeness in the classes APX and PTAS for weighted versions of SAT.

We also illustrated a certain "pathology" of the F-reduction with respect to the A-reduction: the closure of PTAS with respect to the F-reduction strictly includes APX.

Then we proved the existence of incomplete problems. This result illustrates how rich is the structure of NPO: problems whch are NP-complete in their recognition version behave differently in their original optimization form with respect to both approximation properties and completeness in the approximation classes.

Some interesting questions can now be posed. Are there natural complete problems in APX and PTAS? Similarly, are there natural incomplete problems? In this latter case "natural" has a somewhat wider interpretation than in the former question: we ask if it is possible to show, under some complexity theoretic assumptions, the existence of incomplete problems without making use of diagonalization, in particular whether some weighted version of SAT can be proved incomplete.

Acknowledgments

We thank Giorgio Ausiello and Pekka Orponen for many helpful discussions and suggestions. We also thank the unknown referees for suggesting several improvements, both in the contents of the paper and in the style of our presentation. Finally, we are indebted to Phokion Kolaitis for pointing out a mistake in Theorem 2.

Received September 19, 1988; final manuscript received May 30, 1990

References

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

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

Bruschi, D., Joseph, D., and Young, P. (1989), "A Structural Overview of NP Optimization Problems," Computer Science Technical Report 861, University of Wisconsin–Madison.

Cook, S. A. (1971), The complexity of theorem-proving procedures, *in* "Proceedings, 3rd ACM Symposium on Theory of Computation," pp. 151–158.

Garey, M. R. and Johnson, D. (1979), "Computers and Intractability: A Guide to the Theory of NP-Completeness," Freeman, San Francisco.

Homer, S. (1986), On simple and creative sets in NP, (1986), *Theoret. Comput. Sci.* **47**, 169–180.

Hochbaum, D. and Shmoys, D. (1987), Using dual approximation algorithms for scheduling problems: Theoretical and practical results, *J. Assoc. Comput. Mach.* **34**, 144–162.

Ibarra, O. H., Kim, C. E., (1975), Fast approximation for the knapsack and sum of subset problems, *J. Assoc. Comput. Mach.* **22**, 463–468.

Johnson, D. (1974), Approximation algorithms for combinatorial problems, *J. Comput. System Sci.* **9**, 256–278.

KRENTEL, M. W. (1988), The complexity of optimization problems, *J. Comput. System Sci.* **36**, 490–509.

KORTE, B. AND SCHRADER, R. (1981), On the existence of fast approximation schemes, *in* "Nonlinear Programming 4" (O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, Eds.), Academic Press, New York.

LADNER, R. (1975), On the structure of polynomial time reducibility, *J. Assoc. Comput. Mach.* **22**, 155–171.

ORPONEN, P. AND MANNILA, H. (1987), "On Approximation Preserving Reductions: Complete Problems and Robust Measures" Technical Report, University of Helsinki.

PAZ, A. AND MORAN, S. (1981), NP-optimization problems and their approximation, *Theoret. Comput. Sci.* **15** 251–277.

PAPADIMITRIOU, C. AND STEIGLITZ, K. (1982), "Combinatorial Optimization: Algorithms and Complexity," Prentice-Hall, Englewood Cliffs, NJ.

PAPADIMITRIOU, C. AND YANNAKAKIS, M. (1988), Optimization, approximation, and complexity classes, *in* "Proceedings, 20th ACM Symposium, Theory of Computation," pp. 229–234.

SAHNI, S. K. (1976), Algorithms for scheduling independent tasks, *J. Assoc. Comput. Mach.* **23**, 116–127.