Theoretical Computer Science 15 (1981) 251–277
North-Holland Publishing Company

NON DETERMINISTIC POLYNOMIAL OPTIMIZATION PROBLEMS AND THEIR APPROXIMATIONS*

A. PAZ
Department of Computer Science, University of California, Berkeley, CA 94720, U.S.A.

S. MORAN
Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, U.S.A.

Communicated by M. Harrison
Received December 1979
Revised May 1980

Abstract. A unified and general framework for the study of nondeterministic polynomial optimization problems (NPOP) is presented and some properties of NPOP's are investigated. A characterization of NPOP's with regard to their approximability properties is given by proving necessary and sufficient conditions for two approximability schemes. Known approximability results are shown to fit within the general frame developed in the paper. Finally NPOP's are classified and studied with regard to the possibility or impossibility of 'reducing' certain types of NPOP's to other types in a sense specified in the text.

1. Introduction

Non deterministic polynomial optimization problems (NPOP) as introduced in this paper are intended to provide a basis for a natural generalization of the theory of NP sets. Intuitively an NPOP is a set whose elements are encoded according to some 'reasonable' scheme. Each element in the set has a set of nonnegative integers associated with it assumed to represent a certain combinatorial property of the element, a property we are interested in (e.g. if the element is a graph the set of numbers associated with it could be the magnitudes of the different cliques in that graph). The elements of the set or rather their encodings are assumed to be polynomially recognizable and the sets of numbers associated with those elements are assumed to be computable by a nondeterministic Turing machine in polynomial time. An algorithm that solves a given NPOP is understood to be an algorithm which provides, for any element in the set, the maximal (or minimal according to the

* The contribution of the first author was supported in part by the following grants: National Science Foundation Grant DCR74-15091, Department of Systems Sciences, University of California at Los Angeles (the first version of this paper) and National Science Foundation Grant MCS79-03767, Division of Computer Science, Department of Electrical Engineering and Computer Sciences and the Electronics Research Laboratory, University of California at Berkeley.

problem at hand) number out of the set of numbers associated with it. Clearly, if every element in the set of an NPOP has only two numbers associated with it, 0 or 1, then such an NPOP is equivalent to the recognition problem for an NP set. The conjecture that $P \neq NP$ is widely believed to be true and NP complete problems are generally believed to be intractable. This prompted many researchers to develop and study polynomial approximation schemes for NP-problems. Such schemes are more natural and easier to define and study in the context of optimization problems where both the output and its approximation are numerical.

In this paper we suggest a formal definition of NPOP and propose a framework for their study. The definition is similar to other definitions which appeared in the literature both in explicit and implicit form (see references at the end of the paper) and has the property that almost all of the known results concerning NPOP in their various forms, can be stated within its framework. In addition some new and important results are stated and proved, results providing some new insight concerning the nature of the various approximation schemes for NPOP investigated by several authors (see e.g. [1, 3, 5, 6, 9, 10, 13, 15, 16, 21–23]) and providing a characterization of two different approximation schemes studied in the literature. We believe that these results are important in several ways: A unified framework for the study of NPOP, their approximation, and their reductions of different types is set. (As in the NP case reductions may be useful for getting new approximation algorithms out of known such algorithms.) A characterization of various approximation schemes may be useful for the finding of proper approximation schemes to fit new or existing NPOP. Finally such a characterization may also lead in the future to a universal algorithm which will fit automatically an approximation algorithm to an NPOP provided such an algorithm exists and provided that the NPOP in case is known to have certain properties.

The paper is divided into five sections including this introductory first section. Section 2 provides the basic definitions. In the third section some general properties of NPOP and their relation to NP sets are studied. Section 4, which is the core of the paper, deals with the approximability properties of NPOP and provides a full characterization of two known approximation schemes. The last section deals with reducibility properties of NPOP. The paper has two appendices; the first one provides formal definitions for the different NPOP quoted in the paper and the second one contains an NPOP version of Cook's reducibility result for NP.

2. NP optimization problems

Definition 2.1. An *optimization problem* is a subscripted triple $(A, S, \bar{t})_{\text{Ext}}$ where

(1) Ext = Min or Ext = Max.

(2) A is a recursive set. (A is the set of all well formed encodings of some given combinatorial entity, e.g. graphs, families of sets, integer sequences, etc. over a given alphabet Σ.)

(3) S is a function $S: A \to P_0(\Sigma^*)$. ($P_0(\Sigma^*)$ denotes the set of all finite subsets of Σ^*.) For each $a \in A$, $S(a)$ denotes the set of feasible solutions of a. $S(A)$ denotes the set $\bigcup_{a \in A} S(a)$.

(4) A function $\bar{t}$ is assumed to be given, $\bar{t}: A \times S(A) \to Z^+ \cup \{\pm\infty\}$, such that for each pair (a, b), where $a \in A$, $b \in S(a)$, $\bar{t}(a, b)$ denotes the numerical value of b as a solution of a.[1]

(5) For each $a \in A$, $t(a)$ is defined by:

$$t(a) = \{k \mid k = \bar{t}(a, b) \text{ for some } b \in S(a)\}.$$

(If $S(a) = \emptyset$, then $t(a)$ is defined by $t(a) = \{\pm\infty\}$.) t is a set function intended to specify the property of the elements of A we want to study, e.g. the number of clauses which are satisfiable in a given CNF formula, the number of nodes that are pairwise adjacent in a given graph, etc. With regard to the function t we shall use the following notation:

$$\text{op}(a) = \text{optimum}\{k \mid k \in t(a)\}$$

where optimum is 'max' if Ext = Max, and it is 'min' if Ext = Min. The value '$\pm\infty$' will be regarded as '$+\infty$' if Ext = Min and as '$-\infty$' if Ext = Max. For a given $a \in A$, $S^*(a)$ is defined by:

$$S^*(a) = \{b \mid b \in S(a) \text{ and } \bar{t}(a, b) = \text{op}(a)\}.$$

An optimization problem is an NP optimization problem (NPOP) if the following additional conditions hold true:

(6) The set A is recognizable in polynomial time.

(7) For each $a \in A$, the function $S: A \to S(A)$ has the following properties:

(7.1) There is a polynomial p such that, if $a \in A$ and $b \in S(a)$, then $l(b) \leqslant p(l(a))$.[2]

(7.2) The set $\{(a, b) \mid a \in A, b \in S(a)\}$ is recognizable in deterministic polynomial time.

It is easy to see that the above conditions imply that the function S can be computed by a nondeterministic polynomial Turing machine.

(8) The function $\bar{t}$ is of polynomial time complexity.

In many cases, the combinatorial properties (and hence the complexity) of a given NPOP are determined by A and t alone. We shall therefore abbreviate our notation and use the notation $(A, t)_{\text{Ext}}$ whenever the remaining parameter is not relevant to the context. With each optimization problem and each $k \in Z^+$, one can associate a set defined in a natural way by the following definition:

Definition 2.2. Let $(A, t)_{\text{Ext}}$ be an optimization problem. Then

$$(A, t)_{\text{Ext},k} = \{a \in A \text{ op}(a) \leqslant k\}.$$

[1] All the results proved in this paper will remain true if t is allowed to assume negative as well as positive integral values provided that some minor changes are introduced in the various definitions and proofs.

[2] $l(x)$ denotes the length of the input word x.

Given an NPOP $(A, t)_{\text{Ext}}$, if Ext = Min, then $(A, t)_{\text{Ext},k}$ is in NP, and if Ext = Max, then $(A, t)_{\text{Ext},k}$ is in co-NP. The problem: 'given $a \in A$, $k \in Z^+$, is $a \in (A, t)_{\text{Min},k}$?' is the recognition problem which corresponds to the optimization problem $(A, t)_{\text{Min}}$, and the problem: 'given $a \in A$, $k \in Z^+$, is $a \in A$ $(A, t)_{\text{Max},k}$?' is the recognition problem which corresponds to $(A, t)_{\text{Max}}$.

Example 2.3. The subset sum problem is the following NPOP $(\text{IS}, S_{SS}, \bar{t}_{\text{SS}})_{\text{Max}}$ where:

(1) IS is the set of all finite sequences of positive integers.

(2) For $a = (a_1, \ldots, a_n, b) \in \text{IS}$, $S_{\text{SS}}(a)$ is defined by:

$$S_{\text{ss}}(a) = \{(\varepsilon_1, \ldots, \varepsilon_n) \mid \varepsilon_i \in \{0, 1\} \text{ and } (\varepsilon_1, \ldots, \varepsilon_n) \neq (0, \ldots, 0)\}$$

$$\bar{t}_{\text{ss}}(a, (\varepsilon_1, \ldots, \varepsilon_n)) = \begin{cases} \sum \varepsilon_i a_i & \text{if } \sum \varepsilon_i a_i \leqslant b, \\ -\infty & \text{if } \sum \varepsilon_i a_i > b. \end{cases}$$

(It follows that $\text{op}(a) = \max\{k \mid k = \sum \varepsilon_i a_i,\ k \leqslant b\}$.)

3. Solutions of optimization problems: constructive and nonconstructive

When considering optimization problems a distinction should be made between 'constructive' solutions and 'nonconstructive' solutions: Consider e.g. the colorability problem: It is reasonable to assume that the problem of ascertaining whether a given graph is k-colorable is different from the problem of actually finding a k coloration (provided that it exists).

Let $(A, S, \bar{t})_{\text{Ext}}$ be a given NPOP. An algorithm that solves $(A, S, t)_{\text{Ext}}$ is a recursive function $f: \Sigma^* \to Z^+ \cup \{\pm\infty\} \cup \{\alpha\}$ (α is a special symbol not in Z), satisfying the following conditions:

(a) $f(w) = \alpha \Leftrightarrow w \notin A$,

(2) $(\forall a \in A)(f(a) = \text{op}(a))$.

An 'algorithm that solves $(A, t)_{\text{Ext}}$ constructively' is a recursive function $f: \Sigma^* \to \Sigma^* \cup \{\beta\}$ satisfying the following:

(1) $f(w) = \beta \Leftrightarrow w \notin A$ (β is a new symbol not in Σ),

(2) $(\forall a \in A)(f(a) \in S^*(a))$.

Definition 3.1. $(A, S, \bar{t})_{\text{Ext}}$ is (*constructively*) *polynomially solvable* if there exists a polynomial time algorithm that solves $(A, S, t)_{\text{Ext}}$ (constructively). Such an algorithm is a (*constructive*) *polynomial solution* of $(A, S, \bar{t})_{\text{Ext}}$.

A connection between the complexity of an optimization problem $(A, t)_{\text{Ext}}$ and the complexity of the sets $(A, t)_{\text{Ext},k}$ induced by it is established in the following lemma. Some more restricted but related results can be found in the literature (see e.g. [20, Theorem 2.5.1] and [23, p. 732]).

Lemma 3.2. *Let* $(A, t)_{\text{Ext}}$ *be an NPOP. The following two conditions are equivalent*:

(1) *There exists a polynomial* p *(not depending on* k*) such that the problem 'is* $a \in (A, t)_{\text{Ext},k}$*?' is decidable in* $p(l(a))$ *time.*

(2) $(A, t)_{\text{Ext}}$ *is polynomially solvable.*

Proof. (2) $\Rightarrow$ (1): By the definition of NPOP, there exists a polynomial p_1 such that $|\text{op}(a)| < 2^{p_1(l(a))}$ for each a in A. Hence, after $\text{op}(a)$ has been computed, one needs at most $\log(\text{op}(a)) < p_1(l(a))$ steps to find whether $\text{op}(a) \leqslant k$.

(1) $\Rightarrow$ (2): Using binary search, at most $p_1(l(a))$ recognition problems (p_1 is defined as above) of the form 'is $a \in (A, t)_{\text{Ext},k}$?' have to be solved in order to find $\text{op}(a)$. Each of those problems can be solved in time $p(l(a))$. The total time needed is at most $p(l(a)) \cdot p_1(l(a))$.

We shall next formalize and generalize known techniques to obtain polynomial time constructive solutions to optimization problems, given polynomial time (nonconstructive) solutions for those problems (see e.g. [20, Theorem 2.3.1]).

Definition 3.3. Let $(A, S, \bar{t})_{\text{Ext}}$ be an optimization problem, and let $x \in \Sigma^*$ be given. For each $a \in A$, $S_{(x)}(a)$ and $t_{(x)}(a)$ and $\text{op}_x(a)$ are defined by:

$$S_{(x)}(a) = S(a) \cap x \cdot \Sigma^* \quad (= \{b \mid b \in S(a) \text{ and } b = xy \text{ for some } y \in \Sigma^*\}),$$

$$t_{(x)}(a) = \{k \mid k = \bar{t}(a, b) \text{ for some } b \in S_{(x)}(a)\},$$

$$\text{op}_x(a) = \text{optimum}\{k \mid k \in t_x(a)\}.$$

(If $S_{(x)}(a) = \emptyset$, then $t_{(x)}(a) = \{\pm\infty\}$.)

Lemma 3.4. *Let* $(A, S, \bar{t})_{\text{Ext}}$ *be an NPOP. If the following 'prefix restricted form' of* $(A, S, \bar{t})_{\text{Ext}}$*: 'on input* (a, x), $a \in A$, $x \in \Sigma^*$, *find* $\text{op}_{(x)}(a)$*' is polynomially solvable, then* $(A, S, t)_{\text{Ext}}$ *is constructively polynomially solvable.*

Proof. Assume that the prefix restricted form of $(A, S, \bar{t})_{\text{Ext}}$ is polynomially solvable. There exists a polynomial p_1 such that for each $a \in A$, $b \in S(a)$ implies that $l(b) < p_1(l(a))$. Thus, if $l(x) > p_1(l(a))$, then there is no $b \in S(a)$ whose prefix equals x and therefore the only solution for such an input (a, x) is $\pm\infty$. Otherwise, if $l(x) \leqslant p_1(l(a))$, then the length of the input is bounded by a polynomial in $l(a)$. In both cases the prefix restricted form of $(A, S, \bar{t})_{\text{Ext}}$ is solvable in time polynomial in $l(a)$ alone.

The following algorithm finds a $b^* \in S^*(a)$ such that $\bar{t}(a, b^*) = \text{op}(a)$ in time which is polynomial in $l(a)$. (Due to the fact that $(A, S, \bar{t})_{\text{Ext}}$ is solvable in NP space, $l(b^*)$ is bounded by a polynomial in $l(a)$.)

1. $x \leftarrow \lambda$; $\|\lambda$ denotes the null string $\|$
2. find $\text{op}(a)$; $\|\text{op}(a) = \text{op}_{(\lambda)}(a)\|$
3. if $x \in S(a)$ and $\bar{t}(a, x) = \text{op}(a)$ then halt and return x. else

‖By the construction of x so far $\mathrm{op}_{(x)}(a) = \mathrm{op}(a)$, and by step 3 above $x \notin S^*(a)$. It follows that there exists some $y \in \Sigma^+$ such that $xy \in S^*(a)$, and hence there is some $\sigma \in \Sigma$ such that $\mathrm{op}_{(x\sigma)}(a) = \mathrm{op}(a)$‖

4. find $\sigma \in \Sigma$ such that $\mathrm{op}_{(x\sigma)}(a) = \mathrm{op}(a)$. $x \leftarrow x\sigma$, go to 3.

It is easy to see that the above algorithm constructs, step by step, a string $x \in \Sigma^*$ such that $x \in S^*(a)$, and that the number of repetitions of lines 3–4 is $l(x)$. The result follows easily from the fact that $|\Sigma|$ is a constant, and $\mathrm{op}_{(x)}(a)$ is computable in time polynomial in $l(a)$ for any x, where the polynomial does not depend on x.

Many problems have the property that the existence of a polynomial time solutions for them implies the existence of a polynomial time constructive solutions.

The following corollary gives a simple characterization of such problems:

Corollary 3.5. *Let $(A, S, \bar{t})_{\mathrm{Ext}}$ be an NPOP. If the prefix restricted form of it is polynomially reducible*[3] *to it, then the problem of solving 'constructively' $(A, S, \bar{t})_{\mathrm{Ext}}$ is polynomially reducible to $(A, S, \bar{t})_{\mathrm{Ext}}$.*

Proof. The conditions of the corollary imply the conditions of Lemma 3.4.

A problem for which the condition of the corollary above is not known to hold is the following optimization version of the 'non prime' problem, $(Z^+, S_p, \bar{t}_p)_{\max}$ where for $a \in Z^+$, $S_p(a) = \{k \mid k \in Z^+, 1 < k < a\}$, and t_p is defined by

$$\bar{t}_p(a, k) = \begin{cases} 2 & \text{if } k \mid a, \\ 1 & \text{otherwise,} \end{cases}$$

$t_p(a) = \{b \mid b = \bar{t}_p(a, k) \text{ for some } 1 < k < a\}$. Thus, for $a \in Z^+$, $\mathrm{op}(a) = 2$ iff a is nonprime. For a given $x \in \Sigma^*$ (assume $\Sigma = \{0, 1\}$), $\mathrm{op}_{(x)}(a) = 2$ iff there is a divisor of a whose (binary) representation is xy for some y. No polynomial time algorithm is known such that for each $x \in \Sigma^*$ reduces the prefix restricted form of $(Z^+, S_p, \bar{t}_p)_{\mathrm{Max}}$ to $(Z^+, S_p, \bar{t}_p)_{\mathrm{Max}}$. Indeed, it is probably much easier to solve the problem itself (i.e., to tell whether a given integer is a prime) than to solve it constructively (i.e., to find a divisor of a given integer, provided it is nonprime); see e.g. [19, 25].

Theorem 3.6. *The following three conditions are equivalent*:

(1) P = NP.

(2) *All NPOP's are polynomially solvable.*

(3) *All NPOP's are constructively polynomially solvable.*

Proof. (1) $\Rightarrow$ (2): Let $(A, t)_{\mathrm{Ext}}$ be an NPOP. It follows from the definitions of an NPOP that the set

$$\{(a, k) \mid a \in A, k \in Z^+, \mathrm{op}(a) \leqslant k\} \qquad (*)$$

[3] I.e., one can solve the prefix restricted form of $(A, S, \bar{t})_{\mathrm{Ext}}$ in polynomial time given that $(A, S, \bar{t})_{\mathrm{Ext}}$ is polynomially solvable.

is in NP or in co-NP. Therefore if P = NP the set (∗) is recognizable in time which is polynomial in the length of the pair (a, k). There is a polynomial p_1 such that for each $a \in A$, $l(\mathrm{op}(a)) \leq p_1(l(a))$. Thus, if $l(k) > p_1(l(a))$, then the pair (a, k) is trivially in the set (∗). Otherwise, if $l(k) \leq p_1(l(a))$, then the length of the pair (a, k) is bounded by a polynomial in $l(a)$. In both cases, if P = NP, then the set (∗) is decidable in time which is polynomial, in $l(a)$ alone. Thus, in order to check whether $a \in (A, t)_{\mathrm{Ext},k}$ it is enough to check whether $(a, k) \in (*)$, and this can be done in time which is polynomial in $l(a)$ alone. It follows that the first condition of Lemma 3.2 is satisfied, this implying by Lemma 3.2. the condition (2) of this lemma.

(2) ⇒ (3): The prefix restricted form of any NPOP is itself an NPOP as one can see easily. Therefore the above implication follows from Lemma 3.4.

(3) ⇒ (1): Trivially.

Another interesting consequence of Lemma 3.4 and Theorem 3.6 is reflected in the following definition and corollary:

Definition 3.7. An NPOP $(A, t)_{\mathrm{Ext}}$ is NPOP complete if for each NPOP $(B, t')_{\mathrm{Ext}}$, there is an algorithm which solves $(B, t')_{\mathrm{Ext}}$ in polynomial time if that algorithm is allowed to use the operation of computing $\mathrm{op}(a)$ for each $a \in A$ in polynomial time.

Corollary 3.8. *If some NPOP complete problem is polynomially solvable then all NPOP's are constructively polynomially solvable; in addition a given NPOP complete problem is polynomially solvable if and only if it is constructively polynomially solvable.*

4. *p*-approximations for NPOP's

The approximability of various optimization problems has been investigated extensively in the literature in the past few years. While some problems have been shown to have good approximation algorithms (e.g. [10, 9, 21, etc]), other problems have been shown not to be approximable (in a sense to be defined) if P ≠ NP (e.g. [22, 6, 7, 15, etc]).

Some of the above mentioned results are formalized (to fit our definitions) and generalized, and some new results are given in this section.

Definition 4.1. A function $h : \Sigma^* \to Z^+ \cup \{\pm\infty\} \cup \{\alpha\}$ is a *p-approximation* for an optimization problem $(A, t)_{\mathrm{Ext}}$ iff h is a polynomial time function satisfying the following properties:

(1) $h(w) = \alpha \Leftrightarrow w \notin A$ (α is a special symbol not in Z),

(2) $h(a) \geq \mathrm{op}(a)$ if Ext = Min and $h(a) \leq \mathrm{op}(a)$ if Ext = Max,

(3) $h(a) = \pm\infty$ if $\mathrm{op}(a) = \pm\infty$.

Definition 4.2. A function $h_c: \Sigma^* \to \Sigma^*$ is a *constructive p-approximation* for an optimization problem $(A, S, \bar{t})_{\text{Ext}}$ if h_c is a polynomial time function satisfying the following properties:

(1) $h_c(w) = \alpha'$ if $w \notin A$ (α' is a symbol not in Σ),

(2) $(\forall a \in A)\ (h_c(a) \in S(a))$.

The following results hold true for both the constructive and the non-constructive case. We shall omit the proofs for the constructive case, those proofs being similar to the proofs for the nonconstructive case.

The performance of a p-approximation h can be defined as follows (see e.g. [10, 21]):

$$(\forall a \in A) P_h(a) = \begin{cases} \dfrac{|h(a) - \text{op}(a)|}{\min\{h(a), \text{op}(a)\}} & \text{if } \min\{h(a), \text{op}(a)\} \neq \pm\infty, \\ \infty & \text{else.} \end{cases}$$

As a function of the length of the input, the performance is defined by:

$$(\forall n \in Z^+) P_h(n) = \max\{p_h(a) \mid l(a) \leq n \text{ and } p_h(a) \neq \infty\}.$$

Definition 4.3. An optimization problem is $f(n)$ *p-approximable* if there is a p-approximation function, h, for $(A, t)_{\text{Ext}}$ such that $p_h(n) \leq f(n)$ for all $n \in Z^+$.

An optimization problem is *approximable* iff for each $\varepsilon > 0$ there is a p-approximation function h for $(A, t)_{\text{Ext}}$ such that $p_h(a) < \varepsilon$ for all $a \in A$. $(A, t)_{\text{Ext}}$ is *fully approximable* if for each $\varepsilon > 0$ there is a p-approximating function h as above with the additional properties that h can be computed in polynomial time Q where $Q = Q(l(a), 1/\varepsilon)$, i.e., Q is polynomial in both the length of a and the value $1/\varepsilon$. (Those problems are defined as problems which have a 'polynomial time approximation scheme' or 'fully polynomial time approximation scheme' respectively in [7].)

It is known from the literature that there are problems which are approximable or fully approximable, and there are other problems which are not approximable according to the above definitions, if $P \neq NP$. It is important therefore to have necessary or sufficient conditions for approximability or full approximability.

After the first version of this paper appeared [17], the authors became aware of a result of Garey and Johnson [7] concerning a necessary condition for full approximability. That result is reproduced here (with due changes to fit our definition) for the sake of comparison and completeness.

Let max be a function from problem instances to Z^+. Problem instances are assumed to have components which are positive integers, and $\max(a)$ denotes the maximal integer appearing in a where $a \in A$ is an instance of a problem.

An algorithm is pseudo polynomial if its time complexity is bounded from above by a polynomial in $l(a)$ and $\max(a)$.

Theorem 4.4 (G–J). *Let* $(A, t)_{\text{Ext}}$ *be an optimization problem such that* $\text{op}(a)$ *is bounded from above by a polynomial in both* $\max(a)$ *and* $l(a)$. *If* $(A, t)_{\text{Ext}}$ *is fully approximable, then it has a pseudo polynomial time solution.*

A deficiency of the above theorem is its restricted nature as reflected in the conditions on $\text{op}(a)$ which do not always hold (e.g. consider the problem: 'on input $(a_1, \ldots, a_n, b_1, \ldots, b_n, b)$ find a subset $I \subset \{1, 2, \ldots, n\}$ such that $\prod_{i \in I} a_i$ is maximal subject to $\sum_{i \in I} b_i \leq b$. Other examples can be found in [16]).

We shall introduce now necessary conditions for approximability and full approximability not having the above deficiency, and stronger in a sense to be described in the sequel.

4.1. Necessary conditions for approximability and full approximability

Definition 4.5. An optimization problem $(A, t)_{\text{Ext}}$ is *simple* iff for each $k \in Z^+$, $(A, t)_{\text{Ext},k}$ is a set in P. $(A, t)_{\text{Ext}}$ is *rigid* if it is not simple.

Notice that no connection is assumed between $l(a)$ and $\text{op}(a)$ in the condition defined above. The condition is therefore applicable to the previous examples.

Definition 4.6. A simple optimization problem $(A, t)_{\text{Ext}}$ is *p-simple* iff there is some polynomial $Q(x, y)$ such that for each $k \in Z^+$, $(A, t)_{\text{Ext},k}$ is recognizable in $Q(l(a), k)$ time.

Remark 4.7. The following properties are easy to prove for a given NPOP $(A, t)_{\text{Ext}}$:

(1) If it is p-simple and, for all $a \in A$, $\text{op}(a)$ is bounded from above by a polynomial in $l(a)$, then it is polynomially solvable.

(2) If it is p-simple and, for all $a \in A$, $\text{op}(a)$ is bounded from above by a polynomial in $\max(a)$ and $l(a)$, then it has a pseudo polynomial solution.

(3) If it has a pseudo polynomial solution and, for all $a \in A$, $\max(a)$ is bounded from above by a polynomial in $l(a)$ and $\text{op}(a)$, then it is p-simple (see e.g. [16]).

Theorem 4.8. *If* $(A, t)_{\text{Ext}}$ *is approximable, then* $(A, t)_{\text{Ext}}$ *is simple.*

Proof. Let $(A, t)_{\text{Ext}}$ be approximable, and let $k \in Z^+$ be given. Then $(A, t)_{\text{Ext},k}$ is in P: by definition, for each $\varepsilon \geq 0$ there is a polynomial (time) function $h_\varepsilon : \Sigma^* \to Z^+ \cup \{\pm\infty\} \cup \{\alpha\}$ such that $\forall a \in A$,

$$\frac{|h_\varepsilon(a) - \text{op}(a)|}{\min\{h_\varepsilon(a), \text{op}(a)\}} < \varepsilon.$$

Let $\text{Ext} = \text{Max}$. (The other case is similar and is omitted.) $h_\varepsilon(a)$ and $\text{op}(a)$ are integers by definition and $h_\varepsilon(a) \leq \text{op}(a)$. Thus, $h_\varepsilon(a) > k$ implies that $\text{op}(a) > k$. On

the other hand, choosing $\varepsilon = 1/k$, the inequality

$$\frac{|h_\varepsilon(a) - \mathrm{op}(a)|}{\min\{h_\varepsilon(a), \mathrm{op}(a)\}} < \frac{1}{k}$$

implies that

$$\frac{\mathrm{op}(a) - h_\varepsilon(a)}{h_\varepsilon(a)} < \frac{1}{k} \quad \text{or} \quad \frac{\mathrm{op}(a)}{h_\varepsilon(a)} - 1 < \frac{1}{k}$$

and for $h_\varepsilon(a) \leq k$ this inequality is impossible unless $\mathrm{op}(a) = h_\varepsilon(a)$. It follows that:

$$[h_{1/k}(a) \leq k] \leftrightarrow [\mathrm{op}(a) \leq k].$$

In other words, $h_{1/k}$ is a polynomial function that recognizes $(A, t)_{\mathrm{Ext},k}$.

It can be shown that the converse of Theorem 4.8 is not true, and that there are some simple NPOP's which are not p-approximable (the TSP[4] problem [18] is an example), assuming $\mathrm{P} \neq \mathrm{NP}$.

Theorem 4.9. *$(A, t)_{\mathrm{Ext}}$ is fully approximable implies that $(A, t)_{\mathrm{Ext}}$ is p-simple.*

Proof. Let $(A, t)_{\mathrm{Ext}}$ be fully approximable and let $k \in Z^+$ be given. Then $(A, t)_{\mathrm{Ext},k}$ is recognizable in $Q(l(a), k)$ time for some polynomial $Q(x, y)$: by definition there is some polynomial $Q'(x, y)$ such that $(A, t)_{\mathrm{Ext}}$ is ε p-approximable in $Q'(l(a), 1/\varepsilon)$ time. Choosing $\varepsilon = 1/k$, $(A, t)_{\mathrm{Ext}}$ is $1/k$ p-approximable in $Q'(l(a), k)$ time, and applying the same argument as in Theorem 4.8 we see that $(A, t)_{\mathrm{Ext},k}$ is recognizable in $Q'(l(a), k)$ time (that is: $Q = Q'$).

Remark 4.10. If $\mathrm{P} \neq \mathrm{NP}$, then p-simplicity is not a sufficient condition for full approximability, as can be shown by the following NPOP 'modified MAX SUBSET SUM': $(\mathrm{IS}, t'_{\mathrm{ss}})_{\mathrm{Max}}$, which is similar to max subset sum[5] with one exception: For an integer sequence $(a_1, a_2, \ldots, a_n, b)$, t'_{ss} is defined by:

$$t'_{\mathrm{ss}}(a_1, \ldots, a_n, b) = \{k \mid k \text{ divides } b \text{ and } k = \sum_{j=1}^{m} a_{i_j} \text{ for some sequence}$$

$$1 \leq i_1 < \cdots < i_m \leq n\}.$$

Given an input $(a_1 \cdots a_n, b)$ for this problem, its optimum is equal to b if and only if that input satisfies the knapsack property. Let h be an ε-approximation to the above problem with $\varepsilon < 1$. If $h(a) > b/2$, then $b/2 < h(a) \leq \mathrm{op}(a) \leq b$ and, as $\mathrm{op}(a) \mid b$ this implies that $\mathrm{op}(a) = b$. If, on the other hand $h(a) \leq b/2$, then $\mathrm{op}(a)/h(a) \leq \varepsilon + 1 < 2$ or $\mathrm{op}(a) < 2h(a) \leq b$. Thus $\mathrm{op}(a) = b$ iff $h(a) > b/2$. This means that any ε-approximation with $\varepsilon < 1$ to the above problem solves the NP-complete knapsack problem.

[4] See Appendix 1.
[5] See Appendix 1.

Hence, if $P \neq NP$, then this problem is not approximable, and of course not fully approximable; but it can be shown that this problem is p-simple (see next section).

Remark 4.11. Consider the following problem: On input $(a_1, \ldots, a_n)$ where for all i, $a_i = 3b_i$, $b_i \in Z^+$, find the minimal k, $k > 0$, such that $k = 1 + \sum \varepsilon_i a_i$, where $\varepsilon_i \in \{-1, 1\}$. This problem is rigid. For $k = 1$, the set $(A, t)_{\text{Ext},1}$ (where $(A, t)_{\text{Ext}}$ is the above problem) is easily seen to be equivalent to the partition problem, which is known to be NP complete, [11]. Therefore, as follows from Theorem 4.8 above, it is not approximable, which implies that it is not fully approximable. On the other hand, it does have a pseudo polynomial solution (see [7]). Thus the G–J theorem is not applicable to this problem (and there are other such problems, see e.g. [16]), while Theorem 4.8 here is. In addition, it follows from Remark 4.7(2) that the necessary condition given in the G–J theorem is implied by the necessary condition in Theorem 4.9. In this sense our necessary condition is stronger than the condition of Garey and Johnson, as claimed before.

Remark 4.12. One can show that the following problems are simple but not p-simple: MAX CLIQUE, SET COVER, MAX SAT,[6] etc. Those problems cannot be fully approximable if $P \neq NP$, as follows from Theorem 4.9. On the other hand, there are problems which are fully approximable and therefore p-simple [9, 21, 16, etc].

The following is a 'prototype' of a p-simple problem – the SUBSET SUM[7] problem.

Let $a = (a_1, \ldots, a_n, b) \in (Z^+)^{n+1}$ be an input to 'MAX SUBSET SUM'. Then the following algorithm will solve the problem: 'is $a \in (A, t)_{\text{Ext},k}$?' in $O(l(a) \cdot k)$ time units. The algorithm contains a variable 'T' which is the set of values of all 'feasible solutions', and at the end of the algorithm $T = t_{ss}(a)$.

```
1.   begin;
2.   T ← {0}, i ← 1;
3.   For every c in T do begin;
4.   If k < c + a_i ≤ b then halt and reject (comment op(a) > k);
5.   If c + a_i ≤ k then T ← T ∪ {c + a_i} end;
6.   If i = n then halt and accept;
7.   i ← i + 1 go to 3;
8.   End.
```

The algorithm checks, for a given a, whether $a \in (\text{IS}, t_{ss})_{\text{Max},k}$ and its time complexity is $O(n \cdot k) = O(l(a) \cdot k)$. This follows from the fact that $|T| \leq k$ all through the execution of the algorithm.

[6] See Appendix 1.

[7] See Appendix 1.

4.2 Other necessary conditions for approximability and full approximability

We are now introducing other necessary conditions for approximability and full approximability. It will be shown that the conditions in the previous sections together with the conditions to be introduced here are sufficient for approximability and full approximability respectively.

Definition 4.13. An NPOP $(A, t)_{\text{Ext}}$ satisfies the *boundedness condition* B1 if there is a function $B: A \times Z^+ \to \mathbb{Z}_0^+$[8] (for $a \in A$, $c \in Z^+$; $B(a, c)$ is denoted by $B_{a,c}$), satisfying the following:

B1.1: There is a polynomial $E(n)$, $E(n) \geq 0$ for all positive n, such that for each $a \in A$ and for each $c \in Z^+$:

$$B_{a,c} \leq \frac{\text{op}(a)}{c} \leq B_{a,c} + E(l(a)) \quad \text{if Ext} = \text{Max},$$

$$B_{a,c} \geq \frac{\text{op}(a)}{c} \geq B_{a,c} - E(l(a)) \quad \text{if Ext} = \text{Min},$$

B1.2: The time complexity of $B(a, c)$ is bounded from above by $Q(l(a), B_{a,c})$, where Q is some polynomial in two variables.

Definition 4.14. An NPOP $(A, t)_{\text{Ext}}$ satisfies the *boundedness condition* B2 if there is a function B as above, satisfying:

B2.1: There is some constant $E_0 > 0$ such that for each $a \in A$ and for each $c \in Z^+$:

$$B_{a,c} \leq \frac{\text{op}(a)}{c} \leq B_{a,c} + E_0 \quad \text{if Ext} = \text{Max},$$

$$B_{a,c} \geq \frac{\text{op}(a)}{c} \geq B_{a,c} - E_0 \quad \text{if Ext} = \text{Min}.$$

B2.2: The time complexity of $B(a, c)$ is bounded by $Q_{B_{a,c}}(l(a))$, where $Q_{B_{a,c}}$ is a polynomial which depends on the integer $B_{a,c}$ alone.

Remark 4.15. Notice that Condition B1 (and similarly for B2) could also be stated in the following more natural form.

B1.1: There is a polynomial $E(n)$, $E(n) \geq 0$ for all positive n, such that for each $a \in A$ and for each $c \in Z^+$:

$$0 \leq (\text{op}(a)/c - B_{a,c}) \leq E(l(a)) \quad \text{if Ext} = \text{Max}.$$

B1.2: The time complexity of $B(a, c)$ is polynomial in $l(a)$ and $\text{op}(a)/c$.

Example 4.16. The MAX SUBSET SUM problem satisfies Condition B2, as shown by the following algorithm B:

[8] $\mathbb{Z}_0^+$ denotes the set of nonnegative integers including zero.

B: **Input:** $((a_1, \ldots, a_n, b), c)$. **Output:** $B_{a,c}$

1. Split $\{1, 2, \ldots, n\}$ into $I_G = \{i \mid 1 \leq i \leq n \text{ and } a_i \geq c\}$ and $I_S = \{1, 2, \ldots, n\} \setminus I_G$
   ‖comment: for $I \subset \{1, 2, \ldots, n\}$, S_I is defined by: $S_I = \sum_{i \in I} a_i$‖
2. $S^* \leftarrow \max\{S_I \mid I \subset I_G, S_I \leq b\}$
3. ‖let $I_S = (i_1, \ldots, i_l)$‖ for $j = 1$ to l do
4. if $S^* + a_{i_j} \leq b$ then $S^* \leftarrow S^* + a_{i_j}$
5. $B_{a,c} \leftarrow \left\lfloor \frac{S^*}{c} \right\rfloor$.

Let $\bar{S}^*$ be the value of S^* at the termination of the algorithm. If $\sum_{i=1}^n a_i \leq b$, then $\sum_{i=1}^n a_i = \bar{S}^* = \mathrm{op}(a)$. Otherwise, by lines 3–4 of the algorithm (and by the definition of I_S), $0 \leq b - \bar{S}^* \leq c$. In both cases $c \geq \mathrm{op}(a) - \bar{S}^* \geq 0$, and therefore:

$$0 \leq \frac{\mathrm{op}(a)}{c} - \left\lfloor \frac{\bar{S}^*}{c} \right\rfloor = \frac{\mathrm{op}(a)}{c} - B_{a,c} < 2;$$

hence Condition B2.1 holds for $E_0 = 2$.

The time complexity of lines 1, 3, 4, 5 of the algorithm is $O(n)$. Line 2 can be performed by the following algorithm:

2.1. Sort the set $\{a_i \mid i \in I_G\}$
   ‖let the sorted set be $(\bar{a}_1, \ldots, \bar{a}_t)$, $\bar{a}_i \leq \bar{a}_{i+1}$‖
2.2. $\hat{\imath} \leftarrow \max\{i \mid \sum_{j=1}^{i} \bar{a}_j \leq b\}$
2.3. $S^* \leftarrow \max\{S_I \mid I \subset I_G, |I| \leq \hat{\imath}, S_I \leq b\}$.

The integer $\hat{\imath}$ computed in lines 2.1–2.2 satisfies the following:

(a) if $I \subset I_G$ and $|I| > \hat{\imath}$, then $S_I \geq \sum_{j=1}^{\hat{\imath}+1} \bar{a}_j > b$.

(b) $\hat{\imath} \leq \lfloor \bar{S}^*/c \rfloor = B_{a,c}$, since $\sum_{j=1}^{\hat{\imath}} \bar{a}_j \leq \bar{S}^*$, and for each j, $\bar{a}_j \geq c$. If follows from (a) above that S^*, as defined in line 2.3, satisfies: $S^* = \max\{S_I \mid I \subset I_G, S_I \leq b\}$. The time required to compute $\hat{\imath}$ (lines 2.1 and 2.2) is $O(n \log n)$.[9] The time complexity of line 2.3 is $O(\sum_{j=1}^{\hat{\imath}} \binom{n}{j})$, and by (b) above:

$$O\left(\sum_{j=1}^{\hat{\imath}} \binom{n}{j}\right) = O(n^{\hat{\imath}}) = O(n^{B_{a,c}}).$$

It follows that the time complexity of the algorithm as a whole is $O(n^{B_{a,c}})$, and hence Condition B2.2 holds too.

In the following theorems we shall assume that Ext = Max. The proofs for the cases where Ext = Min are similar.

Theorem 4.17. *If $(A, t)_{\mathrm{Max}}$ is fully approximable, then $(A, t)_{\mathrm{Max}}$ satisfies Condition B1.*

[9] This can be reduced to $O(n)$, by using median finding algorithms instead of sorting, as proposed in [13], Section 4].

Proof. Let $(A, t)_{\text{Max}}$ be fully approximable. By the definition there exists a polynomial $Q(x, y)$ such that for each $\varepsilon > 0$, $(A, t)_{\text{Ext}}$ is ε p-approximable in $Q(l(a), 1/\varepsilon)$ time. Without loss of generality Q is assumed to be nondecreasing in both its variables. For a given $a \in A$, $c \in Z^+$, $B_{a,c}$ is computed by the following algorithm B:

B: **Input:** $(a, c) \in A \times Z^+$. **Output:** $B_{a,c}$

1. Compute an 1-approximation (i.e., an ε-approximation for $\varepsilon = 1$) to $\text{op}(a)$. Denote this approximation by $h_1(a)$. (By definitions, $h_1(a) \leqslant \text{op}(a) \leqslant 2h_1(a)$.)
2. $\varepsilon \leftarrow \dfrac{c}{h_1(a)}$
3. if $\varepsilon \geqslant 1$ then begin $B_{a,c} \leftarrow 0$, halt, end.
4. else compute an ε approximation to $\text{op}(a)$, to be denoted as $h_2(a)$. $(h_2(a) \leqslant \text{op}(a) \leqslant h_2(a)(a+\varepsilon))$
5. $B_{a,c} \leftarrow \left\lfloor \dfrac{h_2(a)}{c} \right\rfloor$, halt

Suppose, first that $\varepsilon < 1$. By the definitions of $B_{a,c}$ and ε in the algorithm we have the following:

$$B_{a,c} \cdot \varepsilon = \left\lfloor \frac{h_2(a)}{c} \right\rfloor \frac{c}{h_1(a)} \leqslant \frac{h_2(a)}{h_1(a)} \leqslant \frac{\text{op}(a)}{h_1(a)} \leqslant 2$$

and

$$B_{a,c} \leqslant \frac{h_2(a)}{c} < B_{a,c} + 1.$$

Hence we have:

$$B_{a,c} \leqslant \frac{h_2(a)}{c} \leqslant \frac{\text{op}(a)}{c} \leqslant \frac{h_2(a)}{c}(1+\varepsilon) < (B_{a,c}+1)(1+\varepsilon)$$

$$= B_{a,c} + \varepsilon \cdot B_{a,c} + 1 + \varepsilon < B_{a,c} + 4.$$

If $\varepsilon \geqslant 1$, then $c/h_1(a) \geqslant 1$, and therefore $c/\text{op}(a) \geqslant \frac{1}{2}$, or $\text{op}(a)/c \leqslant 2$. Hence, in this case,

$$B_{a,c} = 0 \leqslant \frac{\text{op}(a)}{c} \leqslant 2 = 0 + 2 = B_{a,c} + 2.$$

In both cases $B_{a,c}$ satisfies Condition B1.1. (In fact even Condition B2.1, which is stronger.)

We shall now show that the time complexity of algorithm B satisfies the Condition B1.2: The time complexity of Step 1 is $Q(l(a), 1) = Q_1(l(a))$. The time complexity of Step 2 is polynomial in $l(a)$ (note that $l(\text{op}(a))$ is bounded by a polynomial in the length of a). The time complexity of Step 3 is a constant. The time complexity of

Step 4 is

$$Q\left(l(a), \frac{1}{\varepsilon}\right) = Q\left(l(a), \frac{h_1(a)}{c}\right) \leqslant Q(l(a), B_{a,c}+4).$$

The time complexity of Step 5 is polynomial in $l(a)$. Altogether, the time complexity of each step in the algorithm is polynomial either in $l(a)$ or in $l(a)$ and $B_{a,c}$. Hence the time complexity of the algorithm as a whole is polynomial in $l(a)$ and $B_{a,c}$, as required.

Theorem 4.18. *If* $(A, t)_{\text{Max}}$ *is approximable, then it satisfies Condition B2.*

Proof. If $(A, t)_{\text{Ext}}$ is approximable then there exists an infinite sequence of polynomials $(Q_1(n), Q_2(n), \ldots)$ such that for each $\varepsilon > 0$, $(A, t)_{\text{Ext}}$ can be ε-approximated in $Q_{\lceil 1/\varepsilon \rceil}(l(a))$ time. Without loss of generality we may assume that $k_2 > k_1$ implies $Q_{k_2}(n) \geqslant Q_{k_1}(n)$ for all n. (If this is not the case, define $Q'_k(n) = \sum_{i=1}^{k} Q_i(n)$ and $Q'_k(n)$ satisfies the assumption.) Bearing this in mind, Theorem 4.18 can be proved along the same lines as Theorem 4.17. The details are omitted.

Remark 4.19. While the necessary conditions given in Section 4.1 are decidability conditions (to decide whether given sets are in P) and as such have been investigated for many particular cases, the boundedness conditions given here are new, numerical and have not been yet investigated. To get some more insight into the nature of those conditions notice the following properties:

(1) As $\text{op}(a) \leqslant 2^{p(l(a))}$ for some polynomial p and all $a \in A$, if $c \geqslant 2^{p(l(a))}$ (or $l(c) \leqslant p^{(l(a))}$), then the conditions will hold true for $B_{a,c} = 0$. Thus the condition must be verified for bounded c only.

(2) Assume that a given NPOP is fully approximable. Then it is p-simple and it satisfies B1. If the problem is not polynomially solvable, then $\text{op}(a)$ is not polynomially bounded by $l(a)$ (see Remark 4.7(1)). Thus, although $|B_{a,1} - \text{op}(a)| \leqslant E(l(a))$ it is not 'practical' to get an approximation to $\text{op}(a)$ by computing $B_{a,1}$ as the complexity of this computation is polynomial in $B_{a,1}$, which is 'polynomially close' to $\text{op}(a)$ which is *not* 'polynomially bounded' by $l(a)$.

4.3. Necessary and sufficient conditions for approximability and full approximability

Theorem 4.20. $(A, t)_{\text{Ext}}$ *is fully approximable* $\Leftrightarrow$ *the following conditions hold*:

(1) $(A, t)_{\text{Ext}}$ *is p-simple,*

(2) $(A, t)_{\text{Ext}}$ *satisfies Condition B1.*

Proof. ($\Rightarrow$) Condition (1) was shown to be necessary in Theorem 4.9 Condition (2) was shown to be necessary in Theorem 4.17.

($\Leftarrow$) We have to show that there is a polynomial $Q(x, y)$ such that for each $\varepsilon > 0$, $(A, t)_{\text{Ext}}$ can be ε-approximated using an algorithm with time complexity

$Q(l(a), 1/\varepsilon)$. Given a and ε, we shall construct an ε-approximation of the form $c \cdot B_{a,c}$. Without loss of generality we assume that $\varepsilon \leqslant 1$:

(a) By Condition B1.1, for any c the following inequalities hold provided that $B_{a,c} > 0$: $B_{a,c} \leqslant \mathrm{op}(a)/c \leqslant B_{a,c} + E(l(a))$. Dividing by $B_{a,c}$ and rearranging terms, we get:

$$0 \leqslant \frac{\mathrm{op}(a) - c \cdot B_{a,c}}{c \cdot B_{a,c}} \leqslant \frac{E(l(a))}{B_{a,c}}.$$

(b) It follows from (a) that $c \cdot B_{a,c}$ is an ε-approximation to $\mathrm{op}(a)$, provided that

$$\frac{E(l(a))}{B_{a,c}} \leqslant \varepsilon \quad \text{or} \quad B_{a,c} \geqslant \frac{E(l(a))}{\varepsilon}.$$

Notice that the above condition implies that $B_{a,c} > 0$.

(c) By the p-simplicity it can be checked in time polynomial in $l(a)$ and $1/\varepsilon$ whether $\mathrm{op}(a) \leqslant 2E(l(a))/\varepsilon$. If this is the case, $\mathrm{op}(a)$ can be computed (using binary search, for instance), in time which is also polynomial in $l(a)$ and $1/\varepsilon$. We shall assume, therefore, that $\mathrm{op}(a) > 2E(l(a))/\varepsilon$.

(d) There exists a polynomial $p(l(a))$ (which can be computed in time polynomial in $l(a)$) such that $\mathrm{op}(a) < 2^{p(l(a))}$. By Condition B1.1:

$$B_{a,2^{p(l(a))}} \leqslant \frac{\mathrm{op}(a)}{2^{p(l(a))}} < 1$$

and hence $B_{a,2^{p(l(a))}} = 0$.

(e) Again, by Condition B1.1 we have that $B_{a,1} \geqslant \mathrm{op}(a) - E(l(a))$. From this, from (c) above and from the fact that $\varepsilon \leqslant 1$, we get:

$$B_{a,1} \geqslant \mathrm{op}(a) - E(l(a)) \geqslant \frac{2E(l(a))}{\varepsilon} - E(l(a)) \geqslant \frac{E(l(a))}{\varepsilon}.$$

Since, by (d) above, $B_{a,2^{p(l(a))}} = 0$, there exists an integer k, $0 < k \leqslant p(l(a))$, such that

$$B_{a,2^k} \leqslant \frac{E(l(a))}{\varepsilon} \leqslant B_{a,2^{k-1}}.$$

(f) Let k be an integer satisfying the conditions of (e) above. By Condition B1.1, the following inequalities hold:

$$B_{a,2^{k-1}} \leqslant \frac{2\mathrm{op}(a)}{2 \cdot 2^{k-1}} = \frac{2\mathrm{op}(a)}{2^k} \leqslant 2(B_{a,2^k} + E(l(a)) \leqslant 2\left(\frac{E(l(a))}{\varepsilon} + E(l(a))\right)$$

$$= \frac{2}{\varepsilon} E(l(a))(1+\varepsilon),$$

and since $\varepsilon \leqslant 1$, $B_{a,2^{k-1}} \leqslant (4/\varepsilon)E(l(a))$.

(g) By Condition B1.2 the time complexity of the function $B : (a, c) \to B_{a,c}$ is bounded by some polynomial in two variables $\hat{Q}(l(a), B_{a,c})$, where $\hat{Q}$ is nondecreasing in both its variables.

(h) An integer k satisfying the conditions of (e) above, and the corresponding $B_{a,2^{k-1}}$, can be computed as follows:

Step 1: $s \leftarrow p(l(a))$; $B_{a,2^s} \leftarrow 0$.
Step 2: while $B_{a,2^s} \leqslant E(l(a))/\varepsilon$ do $s \leftarrow s-1$ compute $B_{a,2^s}$.
Step 3: $k \leftarrow s+1$, $B_{a,2^{k-1}} \leftarrow B_{a,2^s}$, halt.

The time complexity of Step 1 and the number of repetitions of Step 2 are both bounded by $p(l(a))$, by (e) above. By (f) above, at all stages of the algorithm $B_{a,2^s} \leqslant 4E(l(a))/\varepsilon$. Hence, by (g), the time required to compute $B_{a,2^s}$ at each stage is bounded by $\hat{Q}(l(a), 4E(l(a))/\varepsilon)$, which is polynomial in $l(a)$ and $1/\varepsilon$.

Hence, each repetition of Step 2 is of time complexity which is polynomial in $l(a)$ and $1/\varepsilon$. It follows that the time complexity of the evaluation of the integer 2^k and of $B_{a,2^{k-1}}$ is polynomial in $l(a)$ and $1/\varepsilon$.

(i) From (b) and (e) above it follows that $2^{k-1} \cdot B_{a,2^{k-1}}$ is an ε-approximation to $\mathrm{op}(a)$. This completes the proof.

Theorem 4.21. $(A, t)_{\mathrm{Ext}}$ *is approximable* $\Leftrightarrow$ *the following conditions hold*:
(1) $(A, t)_{\mathrm{Ext}}$ *is simple,*
(2) $(A, t)_{\mathrm{Ext}}$ *satisfies condition B2.*

Proof. ($\Rightarrow$) Condition (1) is necessary by Theorem 4.8. Condition (2) is necessary by Theorem 4.18.

($\Leftarrow$) Most of the details of the proof of this part are omitted, the proof being similar to that of Theorem 4.20. On input (a, ε), we look for an ε-approximation to $\mathrm{op}(a)$ of the form $c \cdot B_{a,c}$:

(a) By Condition B2.1, for any c the following inequalities hold:

$$B_{a,c} \leqslant \frac{\mathrm{op}(a)}{c} \leqslant B_{a,c} + E_0$$

where E_0 is some given constant, or:

$$0 \leqslant \frac{\mathrm{op}(a) - c \cdot B_{a,c}}{c \cdot B_{a,c}} \leqslant \frac{E_0}{B_{a,c}},$$

provided that $B_{a,c} > 0$.

(b) From (a) it follows that $c \cdot B_{a,c}$ is an ε-approximation to $\mathrm{op}(a)$, provided that $E_0/B_{a,c} \leqslant \varepsilon$ (or $B_{a,c} \geqslant E_0/\varepsilon > 0$).

(c) By the simplicity it can be checked whether $\mathrm{op}(a) \leqslant 2E_0/\varepsilon$ in time which is bounded by $Q_{\lceil 2E_0/\varepsilon \rceil}(l(a))$ where $Q_{\lceil 2E_0/\varepsilon \rceil}(n)$ is a polynomial which depends on $\lceil 2E_0/\varepsilon \rceil$ alone, and hence on ε alone. If $\mathrm{op}(a) \leqslant 2E_0/\varepsilon$, then $\mathrm{op}(a)$ can be found in $Q'_{\lceil 2E_0/\varepsilon \rceil}(l(a))$ time, where $Q'_{\lceil 2E_0/\varepsilon \rceil}$ is another polynomial which depends on $\lceil 2E_0/\varepsilon \rceil$ alone.

(d) There exists a polynomial $p(n)$ such that $\mathrm{op}(a) < 2^{p(l(a))}$.

(e) There exists an integer k, $0 < k < p(l(a))$ such that

$$B_{a,2^k} \leqslant \frac{E_0}{\varepsilon} \leqslant B_{a,2^{k-1}}.$$

(f) For an integer k satisfying (e) above, the following inequality holds: $B_{a,2^{k-1}} < 4E_0/\varepsilon$.

(g) By Condition B2.2, for each $i \in Z^+$ there exists a polynomial $Q_i(n)$ such that the function $B : (a, c) \to B_{a,c}$ can be computed in $Q_{B_{a,c}}$ time. Moreover: for each i and j, $i < j$ implies $Q_i(n) < Q_j(n)$ for all n (see Remark 4.19).

(h) An integer k that satisfies the condition of (e), and the corresponding $B_{a,2^{k-1}}$ can be computed as follows:

Step 1: $s \leftarrow p(l(a))$, $B_{a,2^s} \leftarrow 0$,

Step 2: while $B_{a,2^s} \leqslant E_0/\varepsilon$ do $s \leftarrow s - 1$, compute $B_{a,2^s}$.

Step 3: $k \leftarrow s + 1$, $B_{a,2^{k-1}} \leftarrow B_{a,2^s}$, halt.

By an argument similar to that given in part (h) of the proof of Theorem 4.17, it can be shown that the time complexity of the evaluation of 2^k and $B_{a,2^{k-1}}$ is bounded by a polynomial which depends on $4E_0/\varepsilon$ only, and hence on ε only.

(i) $2^{k-1} \cdot B_{a,2^{k-1}}$ is an ε-approximation to $\mathrm{op}(a)$. This concludes the proof.

Remark 4.22. Let us assume that a given problem satisfies the condition: $\mathrm{op}(a) \leqslant p(l(a))$ where p is some polynomial. Such a problem satisfies the B1 condition in a trivial way (set $B_{a,c} = 0$ for all c). On the other hand if such a problem is simple it cannot be p-simple if it is not polynomially solvable (see Remark 4.7). Therefore if such a problem is to be approximable it will have to satisfy the Condition B2 whose first part is stronger than the first part of Condition B1, although the resulting approximability property is weaker than p-approximability. It is therefore not necessarily the case that providing an approximation scheme for a given problem is easier than providing a p-approximation scheme for it. In fact most problems that are known to be approximable are also known to be fully approximable (but see [14] for an exception).

The theorems proved in this section are summarized in Fig. 1: Arrows represent implication. The broken line box includes the G–J theorem. The full line box includes our theorems.

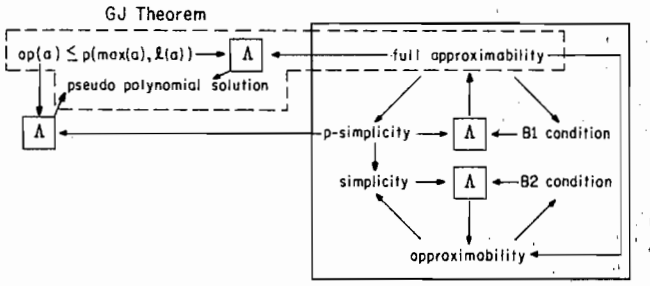

Fig. 1.

5. Reductions which preserve approximability

The concept of 'reduction', and in particular polynomial time reduction, is of crucial importance in the theory of NP recognition problems. There are several definitions of 'polynomial reducibility', due to [11, 4 and 2, p. 373]. According to all the definitions, if a problem A is polynomially reducible to a problem B, then a polynomial time solution of B provides a polynomial time solution to A. In this section we define and study reductions between optimization problems, such that if an NPOP $(A, t_1)_{\text{Ext}}$ is polynomially reducible to $(B, t_2)_{\text{Ext}}$, then a 'good' p-approximation algorithm to $(B, t_2)_{\text{Ext}}$ provides a 'good' p-approximation to $(A, t_1)_{\text{Ext}}$.

Definition 5.1. Let $(A_1, S_1, t_1)_{\text{Ext}}$ and $(A_2, S_2, t_2)_{\text{Ext}}$ be two NPOP's with equal subscripts Ext. Then $g: \Sigma^* \to \Sigma^*$ is a (*polynomial time*) *ratio preserving reduction* of the first NPOP to the second if g is a (polynomial time) function which satisfies the following conditions:

(1) $a_1 \in A_1 \Leftrightarrow g(a_1) \in A_2$,

(2) there exist constants C_1 and C_2, $0 < C_1 \leqslant C_2$, such that for each $a \in A_1$, $C_1 \text{op}(a) \leqslant \text{op}(g(a)) \leqslant C_2 \text{op}(a)$.

If $C_1 = C_2 = 1$, then the reduction is *measure preserving*. The importance of ratio preserving and measure preserving reductions follows from the following:

Lemma 5.2. *Let $f(n)$ be a function which is* $O(\log n)$.[10] *Let $g: (A, t_1)_{\text{Ext}} \to (B, t_2)_{\text{Ext}}$ be a ratio preserving polynomial time reduction. Let $(B, t_2)_{\text{Ext}}$ be an $f(n)$ p-approximable NPOP, where $f(n)$ is as above. Then $(A, t_1)_{\text{Ext}}$ is $O(f(n))$ p-approximable. If g is measure preserving and $(B, t_2)_{\text{Ext}}$ is (fully) approximable, then so is $(A, t_1)_{\text{Ext}}$.*

Proof. Let h be an $f(n)$ p-approximation to $(B, t_2)_{\text{Ext}}$. By definition we have that $C_1 \text{op}(a) \leqslant \text{op}(g(a)) \leqslant C_2 \text{op}(a)$ for some two constants C_1 and C_2 and all $a \in A$. Assume that Ext = Min (the proof for the other case is similar) and consider the following algorithm:

h': **Input** $a \in A$. **Output** $h'(a)$.

1. Reduce a by g to $b = g(a) \in B$
2. Compute $h(b)$
3. $h'(a) \leftarrow \left\lfloor \dfrac{h(b)}{C_1} \right\rfloor$

We claim that h' is an $O(f(n))$ p-approximation to $(A, t)_{\text{Ext}}$. The fact that h and g are of polynomial time complexity implies that h' is also of polynomial time complexity. To show that h' is an $O(f(n))$ approximation to $(A, t)_{\text{Min}}$ we must show that for any $a \in A$ the following inequality holds:

$$0 \leqslant \frac{h'(a) - \text{op}(a)}{\text{op}(a)} \leqslant q(l(a))$$

[10] I.e. $\limsup g(n)/\log(n) < \infty$.

where $q(n)$ is some $O(f(n))$ function. The left-hand side of the above inequality is implied by the following inequalities:

$$\mathrm{op}(a) \leqslant \frac{\mathrm{op}(g(a))}{C_1} \leqslant \frac{h(g(a))}{C_1} = h'(a)$$

(h is a p-approximation to $(B, t_2)_{\mathrm{Min}}$). The right-hand side of the above inequality is equivalent to

$$h'(a) \leqslant \bar{q}(l(a))\mathrm{op}(a)$$

where $\bar{q}(n)$ is $O(f(n))$. (If $\bar{q}(n)$ is $O(f(n))$, then so is $q(n) = \bar{q}(n) - 1$.) But

$$\frac{h(g(a)) - \mathrm{op}(g(a))}{\mathrm{op}(g(a))} \leqslant f(l(g(a))$$

or

$$h(g(a)) \leqslant (f(l(g(a))) + 1)\mathrm{op}(g(a)).$$

Now, as g is polynomial, $l(g(a))$ is polynomial in the length of a and, as $f(n)$ is assumed to be $O(\log(n))$, we have that $f(l(g(a))) \leqslant C_3 f(l(a))$ where C_3 is some constant. Also $\mathrm{op}(g(a)) \leqslant C_2\mathrm{op}(a)$, this following from the definitions. Thus

$$h(g(a)) \leqslant (C_3 f(l(a)) + 1)C_2\mathrm{op}(a)$$

and

$$h'(a) = \frac{h(g(a))}{C_1} = \frac{C_3(f(l(a)) + 1)C_2\mathrm{op}(a)}{C_1} = \bar{q}(l(a))\mathrm{op}(a)$$

where $\bar{q}(l(a)) = C_3(f(l(a)) + 1)C_2/C_1$ is $O(f(l(a)))$ as required.

To prove the second part of the lemma notice that if g is measure preserving then $C_1 = 1$. Thus $h'(a) = h(g(a))$ and $\mathrm{op}(a) = \mathrm{op}(g(a))$ and we have that

$$0 \leqslant \frac{h(g(a)) - \mathrm{op}(g(a))}{\mathrm{op}(g(a))} = \frac{h'(a) - \mathrm{op}(a)}{\mathrm{op}(a)}$$

and

$$\frac{h'(a) - \mathrm{op}(a)}{\mathrm{op}(a)} = \frac{h(g(a)) - \mathrm{op}(g(a))}{\mathrm{op}(g(a))} < \varepsilon$$

so h' is an ε p-approximation if h is.

A measure preserving reduction $g : (A, t_1)_{\mathrm{Ext}} \to (B, t_2)_{\mathrm{Ext}}$ is *constructive* if there is a polynomial time algorithm $f : A \times B \to A$ such that for all $g(a) \in B$, $g^*(a) \in S^*(B)$ implies that $f(a, g^*(a)) \in S^*(a)$ (i.e., one can find an element in $S^*(a)$ given $g^*(a)$ an element in $S^*(B)$). Many reductions specified in [11] when analysed in the light of our definitions here can be shown to be measure preserving reductions, as specified in the examples below. For the definitions of the NPOP's involved in the following

reductions, see Appendix 1. In addition to ratio preserving and measure preserving reductions one may define and study other types of reductions as well, such as order preserving reductions, etc. but we shall not deal with this subject here.

Examples 5.3. (i) g_1 : COLORABILITY → CLIQUE COVER: a graph $G(N, A)$ is reduced to the complemented graph $G'(N, \bar{A})$.

(ii) g_1^{-1} : CLIQUE COVER → COLORABILITY is also a measure preserving reduction.

(iii) g_2 : SET COVER → DOMINATING SET: An input to SET COVER of the form $\phi = \{S_1, \ldots, S_n\}$, where $\bigcup_{i=1}^{n} S_i = S = \{x_1, \ldots, x_m\}$ is reduced to a graph $G(N, A)$, where

$$N = \{1, 2, \ldots, n, x_1, x_2, \ldots, x_m\},$$

$$A = \{(i, j) \mid 1 \leqslant i < j \leqslant n\} \cup \{(i, x_t) \mid x_t \in S_i\}.$$

(iv) g_2' : DOMINATING SET → SET COVER: An input to DOMINATING SET of the form $G(N, A)$ is reduced to a family of sets ϕ in the following manner: Suppose $N = \{1, 2, \ldots, n\}$, then $\phi = \{S_1, S_2, \ldots, S_n\}$ where $S_i = \{i\} \cup \{j \mid (i, j) \in A\}$.

(v) g_3 : NODE COVER → DOMINATING SET: An input $G(N, A)$ to NODE COVER is transformed to $G'(N', A')$ where

$$N' = N \cup A,$$

$$A' = \{(i, j) \mid i, j \in N\} \cup \{(i, e) \mid i \in N, e \in A, i \text{ incident to } e\}.$$

(vi) g_4 : MAX SAT → MAX CLIQUE: An input to MAX SAT of the form $\{C_1, \ldots, C_p\}$, where each C_i is a clause over a set of variables $\{X_1, \bar{X}_1, \ldots, X_n, \bar{X}_n\}$ is reduced to a graph $G(N, A)$, where

$$N = \{V_{\sigma i} \mid \sigma \text{ is a literal}, \sigma \in C_i\},$$

$$A = \{(V_{\sigma i}, V_{ij}) \mid t \neq \bar{\sigma}, i \neq j\}.$$

(vii) g_5 : NODE COVER → SET COVER: An input to NODE COVER of the form $G(N, A)$ is reduced to $\phi = \{S_i\}_{i \in N}$, where $S_i = \{(i, j) \mid (i, j) \in A\}$ (note that the existence of g_5 follows from the existence of g_3 and g_2').

(viii) g_6 : NODE COVER → FEEDBACK NODE SET: A graph $G(N, A)$ is reduced to a digraph $D(V, E)$ where

$$V = N,$$

$$E = \{(i \to j), (j \to i) \mid (i, j) \in A\}.$$

(ix) g_7 : NODE COVER → FEEDBACK ARC SET: A graph $G(N, A)$ is reduced to a digraph $D(V, E)$ where

$$V = \bigcup_{i \in N} \{i_1, i_2\},$$

$$E = \bigcup_{(i,j) \in E} \{(i_1 \to i_2), (i_2 \to j_1), (j_1 \to j_2), (j_2 \to i_1)\};$$

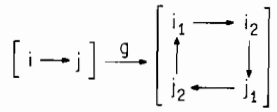

Fig. 2.

Fig. 3 illustrates the above reductions:

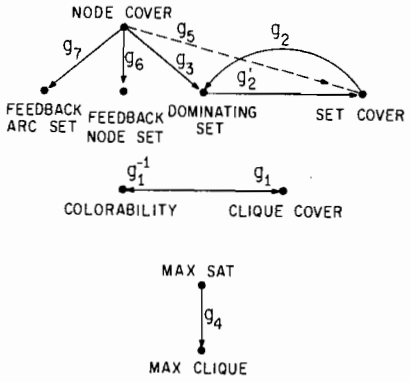

Fig. 3.

The following theorem shows that the classes of NPOP introduced here, rigid, simple and p-simple but not simple, have the property that no problem in the first class can be measure preserving reduced to a problem in the second class, the same being true for the second and third classes.

Theorem 5.4. *Let* $(A, t_1)_{\text{Ext}}$ *be a rigid NPOP,* $(B, t_2)_{\text{Ext}}$ *be a simple but not p-simple NPOP, and* $(C, t_3)_{\text{Ext}}$ *be a p-simple NPOP. Then* $(A, t_1)_{\text{Ext}}$ *is not reducible by a measure preserving reduction to* $(B, t_2)_{\text{Ext}}$, *and* $(B, t_2)_{\text{Ext}}$ *is not reducible by a measure preserving reduction to* $(C, t_3)_{\text{Ext}}$.

Proof. Let k_0 be an integer such that $(A, t_1)_{\text{Ext},k_0}$ is not in P. If $(A, t_1)_{\text{Ext}}$ is polynomially reducible by a measure preserving reduction to $(B, t_2)_{\text{Ext}}$, then the following algorithm will recognize $(A, t_1)_{\text{Ext},k_0}$ in polynomial time: 'given $a \in A$, reduce a to $b \in B$ by a mreaure preserving reduction, and check whether $b \in (B, t_2)_{\text{Ext},k_0}$'. Clearly, $a \in (A, t_1)_{\text{Ext},k_0}$ iff $b \in (B, t_2)_{\text{Ext},k_0}$. Hence, $(A, t_1)_{\text{Ext},k_0}$ is in P, a contradiction. The other part of the theorem may be proved similarly.

We conclude the paper with the following remarks: Let the notation '$(A, t_1)_{\text{Ext}} \leqslant_{\text{p}} (B, t_2)_{\text{Ext}}$' be used to denote polynomial time measure preserving reducibility of $(A, t_1)_{\text{Ext}}$ to $(B, t_2)_{\text{Ext}}$. The relation $\leqslant_{\text{p}}$ is reflexive and transitive. A natural question that may be asked is: "Is there an NPOP $(A_0, t_0)_{\text{Ext}}$ which is maximal with respect to the relation $\leqslant_{\text{p}}$?", i.e., is there an NPOP such that every other NPOP

is polynomially reducible to it by a measure preserving reduction? In Appendix 2 a positive answer is given to this question, by adjusting the technique used in Cook's theorem for recognition problems to optimization problems.

In the note of Knuth [12] a distinction is made between sets which are NP complete and sets which are NP complete by transformation, where the former are sets such that, if one of them is in P, then P = NP, and the latter are sets such that all NP sets can be transformed to any one of them by a polynomial time algorithm.

A similar distinction can be made for NPOP's, where an NPOP complete by transformation problem is an NPOP such that all NPOP's can be reduced to it by a polynomial time measure preserving reduction. By the above discussion, if $P \neq NP$, then the collection of NPOP complete by transformation problems is properly included in the collection of NPOP complete problems. Such a (proper) inclusion has not been yet proved (or disproved) for the NP case.

Appendix 1

We give here a formal definition of several NPOP studied in the literature. Some of the NPOP defined below were mentioned in the paper.

(1) TSP (Travelling Salesman Problem) $:= (W(\mathcal{G}), t_{\text{TSP}})_{\text{MIN}}$, where $W(\mathcal{G})$ is the set of all weighted graphs $W(\mathcal{G})$ (that is, graphs combined with a weight function $W: A \to Z^+$), and for a given weighted graph $W[G(N, A)]$,

$$t_{\text{TSP}}(W[G(N, A)]) = \{k \mid \text{there exists a Hamiltonian cycle in the graph whose weight is } k\} \cup \{\pm\infty\}$$

(we add $\pm\infty$ to $t_{\text{TSP}}(W[G(N, A)])$ to make sure that it is not empty).

(2) MAX CUT $:= (W(\mathcal{G}), t_{\text{CUT}})_{\text{MAX}}$, where $W(\mathcal{G})$ is as above and

$$t_{\text{CUT}}(W[G(N, A)]) = \{k \mid A \text{ contains a cutset of weight } k\}.$$

(3) MAX SUBSET SUM $= (\text{IS}, t_{\text{ss}})_{\text{MAX}}$ where $\text{IS} = \{(a_1, \ldots, a_n, b)\}$ is the set of all finite integer sequences, and

$$t_{\text{ss}}((a_1, \ldots, a_n, b)) = \{k \mid k \leq b \text{ and there are } 1 \leq i_1 < \cdots < i_s \leq n, \sum_{j=1}^{s} a_{i_j} = k\}.$$

(4) JSD (Job Sequencing with Deadlines) $= (\text{IS}^3, t_{\text{JS}})_{\text{MAX}}$ where

$$\text{IS}^3 = \{(T_1, D_1, P_1, \ldots, T_n, D_n, P_n) \mid \{T_i, D_i, P_i\} \subset Z \text{ for } i = 1, \ldots, n\}$$

and

$$t_{\text{JS}}((T_1, D_1, P_1, \ldots, T_n, D_n, P_n)) = \{k \mid \text{there is a permutation } \sigma \text{ of } (1, 2, \ldots, n) \text{ such that } \sum_{i=1}^{n} \delta_{\sigma(i)} P_{\sigma(i)} = k,$$
$$\text{where } \delta_{\sigma(i)} = [\text{if } T_{\sigma(1)} + T_{\sigma(2)} + \cdots + T_{\sigma(i)} > D_{\sigma(i)} \text{ then 0 else 1}]\}$$

(5) SET COVER $= (f, t_{sc})_{MIN}$ where f is the set of all finite families of finite sets, and for $\{S_1, \ldots, S_n\} \in f$,

$$t(\{S_1, \ldots, S_n\}) = \left\{ i \,\middle|\, \text{there exists } 1 \leq j_1 < j_2 < \cdots < j_{i-n} \text{ so that } \bigcup_{r=1}^{i} S_{j_r} = \bigcup_{r=1}^{n} S_r \right\}.$$

(6) DOMINATING SET $= (\mathcal{G}, t_{DS})_{MIN}$, where for $G \in \mathcal{G}$,

$$t_{DS}(G) = \{k \mid \text{there are } k \text{ nodes in } G \text{ that are adjacent to all other nodes of } G\}.$$

(7) CLIQUE COVER $= (\mathcal{G}, t_{cc})_{MIN}$, where for $G(V, E) \in \mathcal{G}$,

$$t_{cc}(G) = \{k \mid \text{there exist } k \text{ cliques in } G \text{ whose union is } V\}.$$

(8) FEEDBACK ARC SET $= (\mathcal{D}, t_{FBA})_{MIN}$, where $\mathcal{D}$ is the set of all directed graphs, and for $D(V, E) \in \mathcal{D}$ ($V =$ the set of vertices, $E =$ the set of edges)

$$t_{FBA}(D) = \{k \mid \text{there exists } k \text{ edges in } A \text{ that each (directed) cycle in } D \text{ contains at least one of them}\}.$$

(9) FEEDBACK NODE SET $= (\mathcal{D}, t_{FBN})_{MIN}$, where for $D(V, E) \in \mathcal{D}$,

$$t_{FBN}(D) = \{k \mid \text{there exists } k \text{ vertices in } N \text{ such that each (directed) cycle contains at least one of them}\}.$$

(10) STEINER TREE $= ((W(\mathcal{G}), S), t_{STR})_{MIN}$, where $(W(\mathcal{G}), S)$ is the set of all weighted graphs together with a given subset of the nodes of the graphs. For a given element $(W(G), S)$ of this set,

$$t_{STR}((W(G), S)) = \{k \mid \text{there exists a subtree of } G \text{ that contains } S, \text{ whose weight is } k\}.$$

Appendix 2

We present here a set A and corresponding function t such that both $(A, t)_{MIN}$ and $(A, t)_{MAX}$ are NPOP's and, for each NPOP $(B, t)_{EXT}$, $(B, t)_{EXT} \leq_p (A, t)_{EXT}$ where EXT = MAX or EXT = MIN. Such an NPOP is an NPOP complete by transformation problem. Our example will, therefore, provide an analogue to Cook's theorem (for recognition problems) for NPOP's. As a matter of fact, the example we are going to present is an extension of the example of Cook made to fit our definitions. We first restate Cook's theorem (without proof) in a slightly different form, suitable for our purpose.

Cook's Theorem. *Let T be an NDTM and let $f: Z \to Z$ be a (polynomial time computable) function, $f(n) \geq n$. Then there exists a function $g: \Sigma^* \to \Sigma^*$ that satisfies the following conditions:*

(1) $g(w) \in \text{SAT} \Leftrightarrow w$ *is accepted by* T *within* $f(l(w))$ *steps.*

(2) *The time complexity of* g *is* $p(f(l(w)))$ *for some fixed polynomial* $p(n)$ $(p(n) < O(n^4))$. (*In the original theorem of Cook* f *is the polynomial representing the time complexity of* T.)

Let $(W(\text{CNE}), t_{\text{COM}})$ be a set and corresponding t-function where $W(\text{CNF})$ is the set of all logical formulas in Conjunctive Normal Form over some set of variables X, combined with a weight function $W: X \to Z$. For a given $a \in W(\text{CNF})$, we define $t_{\text{COM}}(a)$ as follows: Let $\mathcal{B}_a = \{B \mid B: X_a \to \{0, 1\}$ is a valuation of the set X_a of the variables appearing in $a\}$ $(B(\sigma) = 1 \Leftrightarrow B(\bar{\sigma}) = 0)$.

Define a function $M_{\text{COM}}: \mathcal{B}_a \to Z \cup \{\pm\infty\}$ as

$$(\forall B \in \mathcal{B}_a) M_{\text{COM}}(B) = \begin{cases} \pm\infty & \text{if } B \text{ does not satisfy the logical formula } a, \\ \sum\limits_{x \in X_a} W(x)B(x) & \text{else.} \end{cases}$$

Then

$$t_{\text{COM}}(a) = \bigcup_{B \in \mathcal{B}_a} \{M_{\text{COM}}(B)\}.$$

Definition A.1. An *NP measure function* is a function $\mu: \Sigma^* \to P_0[Z \cup \{\pm\infty\} \cup \alpha]$ $(\alpha \notin Z)$ that can be computed by a nondeterministic polynomial time Turing machine (*NP measure machine*). Let $(A, S, t)_{\text{EXT}}$ be an NPOP. By Definition 2.1 there exists an NP measure machine, T, such that for each $a \in A$, $k \in t(a) \Leftrightarrow$ there exists a legal computation of T which terminates within $p(l(a))$ steps, in an accepting state, with k written on its tape. Moreover, we may assume that k is printed in binary digits in reverse order, i.e., if $k = \sigma_1\sigma_2 \cdots \sigma_r$, $\sigma_i = 0$ or $\sigma_i = 1$, then the output on the tape will be $\sigma_r\sigma_{r-1} \cdots \sigma_1$.

Theorem A.2 (Cook's theorem for NPOP's). *Let* T *be a nondeterministic measure machine, and let* $f: Z \to Z$ *be a recursive (polynomial time) function* $(f(n) \geq n)$. *Then there exists a recursive function* $g: \Sigma^* \to \Sigma^*$ *such that*

(1) $g(w) \in W(\text{CNF})$ *and* $k \in t_{COM}(g(w)) \Leftrightarrow$ *there exists a legal computation of* T *which terminates within* $f(l(w))$ *steps in an accepting state with* k *written on its tape.*

(2) *The time complexity of* g *is* $p(f(l(w)))$, *where* $p(n)$ *is some fixed polynomial* $(p(n) < O(n^4))$.

Proof. Without loss of generality we may assume that T has the properties described above (i.e., prints the output in reverse order). For a given $w \in \Sigma^*$, we define the reduction g as follows:

(1) Perform the usual reduction of Cook for w. As a result one gets a logical formula in CNF, over some set of variables X.

(2) Define a weight function on X in the following way: Let $C(i, l, f(l(w)))$ be the variables in Cook's reduction which *asserts* that the symbol l is written in cell i at time $f(l(w))$, $i = 0, 1, \ldots, f(l(w))$. Now for all $x \in X$:

$$W(x) = \begin{cases} 2^i & \text{if } x = C(i, 1, f(l(w))), \\ 0 & \text{else.} \end{cases}$$

We now claim that:

(1) for $a \in \Sigma^*$, $g(a) \in \text{SAT} \Leftrightarrow$ on input a T halts in an accepting state within $f(l(w))$ steps. (This is, in fact, Cook's Theorem.)

(2) for $a \in \Sigma^*$, $k \in M_{\text{COM}}(g(a)) \Leftrightarrow$ on input a there exists a legal computation of T which terminates in an accepting state within time $f(l(a))$, with k written on its tape, in reverse order, in binary digits.

(2) follows from (1) and from the definition of the weight function W.

Remark A.3. The time required for the above reduction differs from that of Cook's original reduction by at most $O(f(l(a))^2)$ steps required to define the weight function W.

References

[1] G. Ausiello, A. D'Atri and M. Protasi, On the structure of combinatorial problem and structure preserving reduction, *Proc. 4th ICALP*, pp. 45–60.

[2] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, MA, 1974).

[3] G. Ausiello, On the structure and properties of NP complete problems and their associated optimization problems, *Proc. 6th International Symposium on MFCS* (1977).

[4] S.A. Cook, The complexity of theorem proving procedures, *Proc. 3rd STOC* (1971) 151–158.

[5] M.R. Garey and D.S. Johnson, Approximation algorithm for combinatorial problems: An annotated bibliography, in: J. Traub, Ed., *Algorithm and Complexity* (1976), 41–52.

[6] M.R. Garey and D.S. Johnson, The complexity of near optimal graph coloring, *J. ACM* **23** (1976) 43–49.

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

[8] M.R. Garey, D.S. Johnson and L. Stockmeyer, Some simplified NP-complete problems, *Proc. 6th STOC* (1974) 47–63.

[9] O.H. Ibarra and C.E. Kim, Fast approximation algorithms for the knapsack and sum of subsets problems, *J. ACM* **22** (1975) 463–468.

[10] D.S. Johnson, Approximation algorithms for combinatorial problems, *Proc. 5th STOC* (1973) 38–49.

[11] R. M. Karp, Reducibility among combinatorial problems, in: R. E. Miller and J. W. Thatcher, Eds., *Complexity of Computer Computations* (Plenum Press, New York, 1972) 85–104.

[12] D.E. Knuth, Postscript about NP hard problems, *SIGACT News* **23** (1974) 15–16.

[13] E.L. Lawler, Fast approximation algorithms for knapsack problems, *Proc. 18th IEEE* (1977) 206–213.

[14] R.J. Lipton and R.E. Tarjan, Applications of a planar separator theorem, *Proc. 18th Annual Symposium on Foundations of Computer Science* (IEEE Computer Society, Long Beach, CA) 162–170.

[15] S. Moran, The complexity of approximation algorithms for a generalized clique problem, to appear.
[16] S. Moran, General approximation algorithms for some arithmetical combinatorial problems, *Theoret. Comput. Sci.* **14** (1981) 289–303.
[17] A. Paz and S. Moran, Nondeterministic polynomial optimization problems and their approximation, *Proc. 4th International Colloquium on Automata, Languages and Programming*, Turku, Finland (1977).
[18] C.H. Papadimitriou and K. Steiglitz, Some complexity results for the TSP, *Proc. 8th STOC* (1976) 1–9.
[19] M.O. Rabin, Probabilistic algorithms.
[20] S. Sahni, Computationally related problems, *SICOMPS* **3** (1974) 262–279.
[21] S. Sahni, General techniques for combinatorial approximation, TR 76-6, Department of Computer Science, University of Minnesota (1976).
[22] S. Sahni and T. Gonzales, P-complete approximation problems, *J. ACM* **23** (1976) 555–565.
[23] S. Sahni and E. Horowitz, Combinatorial problems: Reducibility and approximation, *Operations Res.* **26** (1978) 718–759.
[24] W.J. Savitch, Relationship between nondeterministic and deterministic tape complexities, *J. Comput. System Sci.* **42** (1970) 177–192.
[25] R. Solovay and V. Strassen, A fast Monte Carlo test for primality, *SIGOPS* **6** (1977) 28–32.