NON-DETERMINISTIC POLYNOMIAL OPTIMIZATION PROBLEMS AND THEIR APPROXIMATION

by

A. Paz* and S. Moran**

Technical Report #108
November 1977

* Technion-Israel Institute of Technology, Dept. of Computer Science
** Technion-Israel Institute of Technology, Dept. of Mathematics
1. INTRODUCTION

NP-problems are considered in this paper as recognition problems over some alphabet \( \Sigma \), i.e. \( A \subseteq \Sigma^* \) is an NP problem if there exists a NDTM (non-deterministic Turing machine) recognizing \( A \) in polynomial time.

It is easy to show that the following theorem holds true.

**Theorem 1:** Let \( A \) be a set in NP. Then there exists a NDTM \( M_A \) which recognizes \( A \) such that \( M_A = M^\mu_A \circ M^\pi_A \circ M^1_A \), where

1) The operation "\( \circ \)" is defined as follows: \( M_1 \circ M_2(x) \) is \( M_1(M_2(x)) \); \( M_1, M_2 \) are Turing machines and \( x \) is an input tape.

2) \( M^1_A \) is a polynomial time deterministic encoding machine. Its task is to encode an input \( a \in A \) in some proper way to be denoted by \( a' \).

3) \( M^\pi_A \) is a NDTM which choses some permutation \( \pi(a') \) out of a possible subgroup of the group of all permutations of the encoded input tape \( a' \) in polynomial time.

4) \( M^\mu_A \) is a polynomial time DTM which computes a number \( \mu(\pi(a')) \).

\[
\begin{align*}
\mu(\pi(a')) & \leq k_a & \text{(min problem)} \\
\mu(\pi(a')) & \geq k_a & \text{(max problem)}
\end{align*}
\]

5) \( a \in A \) iff

\[
\mu(\pi(a')) \leq k_a \quad \text{(min problem)}
\]

where \( k_a \) is a number computed in polynomial time by the machine \( M^1_A \) (\( k_a \) is part of the encoding of \( a \)).

Thus every NP problem can be represented as an optimization problem, and the recognition process can be split into three stages where the non-deterministic stage (the machine \( M^\pi_A \)) is separated from the other stages.
Example: Let $A$ be the following (MAX SAT) problem: $a \in A$ iff $a$ is a string of the form $(C_1, \ldots, C_p, k)$ where the $C_i$ are clauses over a set of variables $\{x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_n, \bar{x}_n\}$; $k$ is an integer ($k \leq p$) and, there exists a truth assignment to the variables that satisfies at least $k$ clauses (if $k = p$ then the problem is SAT). $M_A$, a machine that recognizes $A$, can be constructed as: $M_A = M_\mu A \circ M_\pi A \circ M_1 A$ where: $M_\mu A$ checks if the input is well formed; if so it does not change it. If not it stops in a rejecting state. $M_\pi A$ induces a permutation on the clauses and then induces another permutation on the literals within some of the clauses. $M_1 A$ performs the following algorithm:

1. Set $\mu \leftarrow 0$;
2. If all clauses are marked, halt and return $\mu$;
3. Let $C_1$ be the first unmarked clause. If $C_1$ is empty then mark $C_1$ and go to (2);
4. Let $\sigma$ be the first variable appearing in $C_1$. Set $\mu + \mu + 1$ mark $C_1$;
5. For each $C_j$ if $\sigma \in C_j$ then mark $C_j$. Set $\mu + \mu + 1$;
6. For each unmarked $C_j$, $C_j \leftarrow C_j \setminus \{\sigma\}$;
7. Go to (2).

It is easy to see that the above algorithm is polynomial. After the algorithm is completed, the machine checks if $\mu \geq k$ and stops in an accepting state if the inequality holds true, and in a rejecting state otherwise. It is easy to see that $M_A$ recognizes the set $A$ as required.

Proof of Theorem 1 follows from the fact that every problem in NP is polynomially reducible to SAT, which is, as mentioned above, a special case of MAX SAT. The reduction is then incorporated into $M_1 A$ of the above example.
2. NP OPTIMIZATION PROBLEMS

The conjecture that P ≠ NP is widely believed to be true. This conjecture prompted many researchers to develop and study polynomial approximations for problems in NP, when considered as optimization problems. See e.g. [Jo 73] or [Sa 76].

The previous section points toward the possibility of a new approach to the study of NP problems and NP optimization problems. In what follows, an attempt is made to develop that new approach. The results achieved so far are promising. These results provide some new insight into recently proved approximation results and it is hoped that they will serve as a basis for a more extensive theory of combinatorial approximations.

Definition 1. An NP optimization problem (NPOP) is a subscripted 4-tuple \((A, F, t, \mu)_{\text{EXT}}\) where:

\[
\begin{align*}
\text{EXT} &= \text{MIN or EXT = MAX}.
\end{align*}
\]

\(A \subset \Sigma^*\) is a polynomial time recognizable recursive set over a finite alphabet \(\Sigma\) (\(A\) is the set of all well formed encodings of some given combinatorial entity e.g. graph, family of sets, logical sentence in CNF, etc.). It is assumed that \(\lambda \in A\) where \(\lambda\) denotes the empty word.

\(F\) is a function \(F\colon A \to P_0(A)\) (the set of all finite subsets of \(A\)), where for all \(a \in A\), \(F(a)\) is a subgroup of the group of all permutations of \(a\), to be called "the set of proper permutation of \(a\)". An element in \(F(a)\) will be denoted by \(\pi(a)\). It is also assumed that the many valued function \(a \to F(a)\) is computed in polynomial time by a NDTM ("permutation machine").

\(t\) is a function \(t\colon A \to P_0(\mathbb{Z} \cup \{\infty\})\). \(t\) is a 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

\[ \text{op}(a) = \text{optimum}(k: k \in t(a)) \]

optimum is "max" if \( \text{EXT} = \text{MAX} \) and it is "min" if \( \text{EXT} = \text{MIN} \). We shall
use the value \(-\infty\) in connection with MAX problems and the value \(+\infty\) in
connection with MIN problems.

It is also assumed that \( F \) is compatible with \( t \), that is: \( a' \in F(a) \)
implies that \( t(a') = t(a) \), that \( t(\emptyset) = \{0\} \), and that \( t(a) \neq \emptyset \) for all \( a \in A \).

\( \mu \) is a polynomial time function (the measure function) \( \mu : \Sigma^* \to \mathbb{Z} \cup \{\pm\infty\} \cup \{\} \)
\( \{a\} \) (\( a \in \mathbb{Z} \)) satisfying the following properties:

1. \( \mu(w) = a \) iff \( w \notin A \);
2. \( (\mu(a) = k) \to k \in t(a) \);
3. \( (\forall a \in A)(\exists \pi^*(a) \in F(a)) \ (\mu(\pi^*(a)) = \text{op}(a)) \).

It should be noticed that the combinatorial properties (and the
complexity) of a given NPOP are determined by \( A, t \) and the subscript
\( \text{EXT} \). We shall therefore abbreviate our notation and use the notation \( (A,t)_{\text{EXT}} \)
or \( (A,t,\mu)_{\text{EXT}} \) whenever the other parameters are not relevant to the context.

Examples: (1) The problem mentioned before MAX SAT can be described in
the form \((A,t)_{\text{MAX}}\) where \( A \) is the set of all CNF formulas and for \( a \in A \),
k \( \notin t(a) \) iff there is a truth assignment to the variables occurring in a
which satisfies exactly \( k \) clauses.

(2) Colorability: \((G,t)_{\text{MIN}}\) where \( G \) is the set of all graphs and for
\( G \in G, k \in t(G) \) iff \( G \) is \( k \)-colorable.
(3) Let \( A \) be any set in NP. One can show that \( A \) can be defined by the NPOP \((\Sigma^*, t_A)_{\text{MAX}}\) where \( t_A(w) \subset \{0,1\} \) for all \( w \in \Sigma^* \), and \( 1 \notin t_A(a) \) for all \( a \in A \). It follows that \( A = \{w \mid \text{op}(w) = 1\} \).

Remark: When considering NP problems as recognition problems a distinction should be made between "polynomially constructive" solutions and "polynomially nonconstructive" solutions.

Consider e.g. colorability: it is clear 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). This suggests the necessity of distinguishing "constructive" and "nonconstructive" solutions of NPOP's.

Let \((A,F,t,l)_{\text{EXT}}\) be a NPOP. An "algorithm that solves \((A,t)_{\text{EXT}}\)" is a recursive function \( f: \Sigma^* \rightarrow \mathbb{Z} \cup \{\pm \infty\} \cup \{\alpha\} \) \((\alpha \in \mathbb{Z})\) satisfying the following:

1) \( f(w) = \alpha \leftrightarrow w \notin A \),
2) \((\forall a \in A) \ (f(a) = \text{op}(a))\).

An "algorithm that solves \((A,t)_{\text{EXT}}\) constructively" is a recursive function \( f: \Sigma^* \rightarrow \Sigma^* \) satisfying the following:

1) \( f(w) = \beta \leftrightarrow w \notin A \) \((\beta \text{ is a string not in } A)\),
2) \((\forall a \in A) \ (f(a) = \pi(a))\).

Definition 2. \((A,t)_{\text{EXT}}\) is "(constructively) polynomially solvable" if there exists a polynomial time algorithm that solves \((A,t)_{\text{EXT}}\) (constructively), and such an algorithm is a "(constructive) polynomial solution" of \((A,t)_{\text{EXT}}\).
We shall show now that the two notions of solvability are equivalent to each other in some global sense.

**Lemma 2.** \((A, F, t, u)_{\text{EXT}}\) is constructively polynomially solvable implies that \((A, t)_{\text{EXT}}\) is polynomially solvable.

**Proof.** Let \(f\) be a constructive polynomial solution of \((A, t)_{\text{EXT}}\).
Define \(f'\) by:
\[
f'(w) = \begin{cases} u(f(w)) & w \in A \\ a & w \notin A \end{cases}
\]

It follows directly from the definitions that \(f'\) is a polynomial algorithm that solves \((A, t)_{\text{EXT}}\).

**Theorem 3.**
(a) If all NPOP's are polynomially solvable then \(P = NP\).
(b) If \(P = NP\) then all NPOP's are constructively polynomially solvable.

**Proof.** Part (a) follows easily from Theorem 1. Part (b) will be proved in 2 stages. We shall show first that if \(P = NP\) then all NPOP's are polynomially solvable, and then we shall show that for each NPOP \((A, t)_{\text{EXT}}\) there exists an NPOP \((A, t')_{\text{EXT}}\) such that a polynomial solution of the second provides a constructive polynomial solution of the first.

To prove the first part, we note that a polynomial time measure machine can compute, on input \(a\), integers that are not larger than \(2^P(1(a))\) for some polynomial \(P\). This implies that \(\text{op}(a) \leq 2^P(1(a))\). Moreover, for each \(k \leq 2^P(1(a))\), the set \(\{a \in A \mid \text{op}(a) \leq k\}\) can be recognized by a nondeterministic polynomial time Turing machine. (This follows directly
from the definition of NPOP.) Therefore, using binary search, no more than $P(l(a))$ NP recognition problems of the form "is \(\text{op}(a) \leq k?\)" have to be solved in polynomial time, and therefore \(\text{op}(a)\) can be found in polynomial time.

To prove the second part, we assume W.L.G. that \(\Sigma = \{0, 1\}\). Let \(n(a)\) be the binary number represented by a string \(a\) (\(a \in \Sigma^+\)), e.g. \(n(0010) = 10\). Let \((A, F, t, \mu)_{\text{EXT}}\) be a given NPOP and assume the output of the measure function \(\mu\) to be given in binary numbers. Define a new measure function \(\mu'\) as follows: \(\mu'(a) = 2^{l(a)}\mu(a) + n(a)\) (e.g. if \(a = 0010\) and \(\mu(a) = 101\) then \(\mu'(a) = 1010010\)). Finally let \(t'(a)\) be defined as \(t'(a) = \{k \mid (a' \in F(a)) \ (\mu'(a') = k)\}\). One verifies easily that:

1) \(\mu(a) < \mu'(a') < t(a)\);  
2) If \(\pi^*(a)\) is an optimal permutation for \((A, t')_{\text{EXT}}\) then it is also an optimal permutation for \((A, t)_{\text{EXT}}\).  
3) If \((A, t)_{\text{EXT}}\) is an NPOP then so is \((A, t')_{\text{EXT}}\).  
4) For any \(a \in A\), \(\text{op}'(a) = \text{op}(a)\) according to \(t'\) is a number such that its last \(l(a)\) digits represent \(\pi^*(a)\), and the other digits are (the binary representation of) \(\text{op}(a)\).

From the above discussion it is evident that any solution for \((A, t')_{\text{EXT}}\) provides a constructive solution for \((A, t)_{\text{EXT}}\), since knowing \(\text{op}'(a)\) is equivalent to the knowing of \(\pi^*(a)\) and \(\text{op}(a)\) at the same time.

Q.E.D.

Combining Lemma 2 with Theorem 3 we have the following:

**Corollary 1.** The following three conditions are equivalent:

1. \(P = \text{NP}\)
2. All NPOP are polynomially solvable.
3. All NPOP are constructively polynomially solvable.
Proof. By Theorem 3 (2) implies (1) implies (3). By Lemma 2 (3) implies 2. Q.E.D.

Another interesting consequence of Lemma 2 and Theorem 3 is reflected in the following definition and corollary:

**Definition 3.** An NPOP is NPOP complete if the existence of a polynomial solution to it implies that $P = NP$.

**Corollary 2.** 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.

**Remark 1.** It is clear that every NP complete problem when viewed as an NPOP, and every NPOP problem such that a polynomial solution to it would imply that some NP complete problem is polynomially solvable, are NPOP complete.

**Remark 2.** Corollary 2 justifies Definition 3 which is shown, in this corollary, to be in accordance with the definition of NP complete problems adopted by Aho-Hopcroft's Ulman [AHU 74, p.373].
3. REDUCTIONS BETWEEN NPOP's

On the basis of the previous definitions we are able to define and study reducibility and in particular polynomial reducibility between NPOP's.

Definition: Let \((A_1, t_1)^{\text{EXT}_1}\) and \((A_2, t_2)^{\text{EXT}_2}\) be two NPOP's. Then
g: \(\Sigma^* \rightarrow \Sigma^\star\) is a (polynomial) reduction of the first NPOP into the second
iff g is a polynomial function which satisfies the following conditions:

1. \(a_1 \in A_1\) iff \(g(a_1) \in A_2\);

2. There exists a (polynomial time) function \(\delta: A_1 \times Z \rightarrow Z\) such that:
\(\forall a_1 \in A_1, \, \delta(a_1, \text{op}(g(a_1))) = \text{op}(a_1)\) (that is, one can compute \(\text{op}(a_1)\)
if \(\text{op}(g(a_1))\) is known). For abbreviation we shall use the following
notation: given \((A, E, t, \mu)^{\text{EXT}}, \) then for \(a \in A, \, k_1, k_2 \in t(a)\):

\[ k_1 < k_2 \iff \delta(k_1) = \text{MIN} \]
\[ k_1 > k_2 \iff \delta(k_1) = \text{MAX} \]

(that is if \(k_1\) is a better approximation to \(\text{op}(a)\) than \(k_2\)).

The reduction is order preserving if the above function satisfies the following additional conditions:

Let \(a_1 \in A_1\) and \(a_2 = g(a_1) \in A_2, \) then

1. \(\forall k \in t_2(a_2), \, \delta(a_1,k) \in t_1(a_1)\); 

2. \(\forall k_1, k_2 \in t_2(a_2)\) it is true that

\[ k_1 < k_2 \iff \delta(k_1) < \delta(k_2) \]

An example of order preserving reduction is given by ([Ka 72]): Let

\(\text{MAX}. \text{CLIQUE}\) be the following NPOP:

\((G, t_{MC})^{\text{MAX}}\) where \(G\) is the set of all graphs, and for \(G \in G, \)
\(t_{MC}(G) = \{k \mid G \text{ contains a complete subgraph with } k \text{ nodes}\}.)
Let NODE COVER be the following (G, t^NC, min) problem: G is the set of all graphs, as before, and for G ∈ G, t^NC(G) = {k | there exist k nodes in G which are incident to all arcs of G}.

The following reduction g: G(N, A) → G'(N, A) where A = {(i, j) | (i, j) ∈ A} is an order preserving reduction of MAX CLIQUE to NODE COVER and vice versa, with the following g: g(G(N, A), k) = |N| - k.

An order preserving reduction is measure preserving if the function g satisfies also the property:

\[(3.3) \ (\forall a_1 ∈ A_1) \ (\forall k ∈ Z), \ g(a_1, k) = k.\]

The measure preserving reductions have the property that any measure μ_2 on (A_2, t_2)_EXT induces a measure μ_1 on (A_1, t_1)_EXT such that μ_1(a_1) = μ_2(a_2) (a_1 ∈ A_1 and a_2 = g(a_1) ∈ A_2). It is easy to show that measure preserving reductions can exist only between NPOPs such that EXT_1 = EXT_2. The importance of measure preserving reductions will be illustrated in Section 4, Lemma 6.

The notation "(A, t_1)_EXT \leq (B, t_2)_EXT" will be used to denote measure preserving reducibility, where "\leq" denotes polynomial reducibility and "\leq_p" denotes polynomial reducibility with corresponding function g. The relation \leq is reflexive and transitive.

**Constructive reductions**

**Definition 4.** For any given NPOP (A, F, T, μ)_EXT for all a ∈ A and every pair a_1, a_2 ∈ F(a)

\[a_1 \prec a_2 ⇔ μ(a_1) < μ(a_2).\]

Thus \(π^*(a) \leq π(a) \in F(a)\) for all \(π(a) \in F(a)\). The relation \leq induces a partial order on F(a) for all a ∈ A.
In many cases we would like to find for a given NPOP and \( a \in A \), a \( \pi(a) \) such that \( \pi(a) \ll a \) rather than having only a value \( k \) having the property that \( k \ll \mu(a) \). Thus if the NPOP is MAX CLIQUE we would like to get as big a clique as possible in a given graph rather than ascertaining that the graph \( \Gamma \) has a clique containing \( k \) nodes. This motivates the following:

**Definition 4.** Let \( (A_1,F_1,t_1,\mu_1)_{EXT_1} \) and \( (A_2,F_2,t_2,\mu_2)_{EXT_2} \) be two NPOP's. \( g: \mathbb{L}^* \rightarrow \mathbb{L}^* \) is a constructive (polynomial) reduction of the first NPOP to the second iff \( g \) is a (polynomial) function satisfying condition (1) of Definition 4, together with the following condition:

\[ (2') \text{ There exists a (polynomial time) function } f^c: A_1 \times A_2 \rightarrow A_1 \text{ such that:} \]
\[ (\forall a_1 \in A_1) \quad \delta^c(a_1, \pi^*(g(a_1))) = \pi^*(a_1) \quad \text{(that is: given } a_1 \text{, one can compute } \pi^*(a_1) \text{ if } \pi^*(g(a_1)) \text{ is known). A constructive reduction is order preserving if the above function } \delta^c \text{ satisfies also the additional conditions:} \]

Let \( a_1 \in A_1 \) and \( \mu_2 = g(a_1) \in A_2 \), then:

\[ (4.1) \quad (\forall a_2 \in F_2(a_2)) \quad \delta^c(a_1,a_2) \in F_1(a_1) . \]
\[ (4.2) \quad (\forall a_2^t,a_2^m \in F_2(a_2)) \quad a_2^t \ll a_2^m \rightarrow \delta^c(a_2^t,a_1) \ll \delta^c(a_2^m,a_1) . \]

A constructive order preserving reduction is measure preserving if the function \( \delta^c \) satisfies the following:

\[ (4.3) \quad (\forall a_1 \in A_1) (\forall a_2^t \in F(a_2)) \quad \nu_2(a_2^t) = \nu_1(\delta^c(a_1,a_2^t)) . \]

It can be shown that the existence of constructive reductions of one of the three types above implies the existence of nonconstructive reductions of the same type, the proof being similar to the proof of Lemma 2 (page 6).
Some Examples. We give now, without proof, some measure preserving reduction between NPOP's. All of the following reductions can be shown to be constructive reductions. Part of the reductions are Karp's reductions ([Ka 72]) when adjusted to NPOP's. For definitions of the NPOP's see appendix.

(i) \( g_1 : \) COLOROBILITY + CLIQUE COVER: a graph \( G(N,A) \) is reduced to the complemented graph \( G'(N,\bar{A}) \).

(ii) \( g_1^{-1} : \) CLIQUE COVER + COLOROBILITY, 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 \( S_1 \cup \cdots \cup S_n = S = \{ x_1, \ldots, x_m \} \) is reduced to a graph \( G(N,A) \), where:

\[
N = \{1,2,\ldots,n,x_1,x_2,\ldots,x_m\}
\]
\[
A = \{(i,j) \mid 1 \leq i < j \leq n\} \cup \{(i,x_t) \mid x_t \in S_i\}.
\]

(iv) \( g_3 : \) DOMINATING SET + SET COVER: An input to DOMINATING SET of the form \( G(N,A) \) is reduced to a family of sets \( \phi \) 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) \( g_4 : \) 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) \( g_5 : \) 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:

\[
N = \{ V_{\sigma} \mid \sigma \text{ is a literal, } \sigma \in C_i \}
\]
\[
A = \{(V_{\sigma}, V_{t,j}) \mid t \neq \overline{\sigma}, i \neq j\}.
\]
(vii) $g_5$ : NODE COVER $\rightarrow$ SET COVER: An input to NODE COVER of the form $G(N,A)$ is reduced to $\phi = \{S_i\}_{i \in \mathbb{N}}$, 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 $\rightarrow$ FEEDBACK NODE SET: A graph $G(N,A)$ is reduced to a digraph $D(V,E)$ where

$V = N$

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

(ix) $g_1$ : NODE COVER $\rightarrow$ FEEDBACK ARC SET: A graph $G(N,A)$ is reduced to a digraph $D(V,E)$ where $V = \bigcup_{i \in \mathbb{N}} \{i_1,i_2\}$

$E = \bigcup \{(i_1+i_2), (i_2+j_1), (j_1+j_2), (j_2+i_1) \mid (i,j) \in E\}$

The following diagram illustrates the above reductions:
It will be shown now that the class of NPOP's can be divided into two subclasses, such that no problem in one class can be reduced by a measure preserving reduction to a problem in the second class (unless $P = NP$).

**Definition 5.** Let $(A,t)_{EXT}$ be a NPOP. Then for each $k \in \mathbb{Z}$,

$$(A,t)_{EXT,k} = \{a | a \in A \text{ and } \text{op}(a) \leq k\}.$$ 

**Definition 6.** $(A,t)_{EXT}$ is a "simple NPOP" iff for all $k \in \mathbb{Z}$, $(A,t)_{EXT,k}$ is a set in $P$. It is a "rigid NPOP" if it is not simple (i.e. for some $k$, $(A,t)_{EXT,k}$ is in $NP \setminus P$, where the notation $NP \setminus P$ stands for the sets which are in $NP$ and are not in $P$ provided that $P \neq NP$).

**Theorem 4.** If $(A,t_1)_{EXT}$ is a rigid NPOP and $(B,t_2)_{EXT}$ is a simple NPOP, then

$$(A,t_1)_{EXT} \not\preccurlyeq_P (B,t_2)_{EXT}.$$ 

**Proof.** Let $k_0 \in \mathbb{Z}$ be such that $(A,t_1)_{EXT,k_0} \in NP \setminus P$. Assume that

$$(A,t_1)_{EXT} \not\leq_P (B,t_2)_{EXT}.$$ 

The following polynomial algorithm will check for each $w \in \Sigma^*$ if $w \in (A,t_1)_{EXT,k_0}$:

(a) check if $w \in A$, if not reject;

(b) reduce $w$ by $g$ to $b \in B$;

(c) check whether $b \in (B,t_2)_{EXT,k_0}$. If so accept else reject.

(Clearly, $b \in (B,t_2)_{EXT,k_0} \iff w \in (A,t_1)_{EXT,k_0}$.)

All three steps of the algorithm are polynomial, so that the algorithm is polynomial as a whole. It follows that $(A,t_1)_{EXT,k_0} \in P$, which is impossible. The theorem is thus proved.
Remark. Theorem 4 will remain true if the definition of measure preserving reductions is generalized as follows:

Replace condition 3.3 by the condition

3.3' \((\forall a_1 \in A_1)(\forall k \in \mathbb{Z})(\delta(a_1,k) = \eta(k))\)

where \(\eta\) is a (polynomial time) function from \(\mathbb{Z}\) to \(\mathbb{Z}\).

Similar generalizations are possible for the other theorems concerning measure preserving reductions given in the sequel.

If \(P \neq NP\), then a set \(A\) is NP complete implies \(A \in NP \setminus P\).

Combining this with the known NP completeness results, all known NPOP's can be shown to be either rigid or simple. Some examples are given below:

**RIGID NPOP's**

(a) Colorability (see [S 73]). Planar colorability is a special type of rigid NPOP as there is only one \(k(=3)\), for which \((A,t)^{EXT,k}\) is NP complete.

(b) Bin Packing = \((IS,t_{BP})_{MIN}\)

\(IS = \{(a_1,\ldots,a_n,a_{n+1}) | \forall i \ a_i \in \mathbb{Z}\}\)

\(t_{BP}((a_1,\ldots,a_n,a_{n+1})) = \{k | \text{the set } \{a_1,\ldots,a_n\} \text{ can be divided into } k \text{ subsets, the sum of numbers in each of them } \leq a_{n+1}\}\).

**SIMPLE NPOP's**

(a) MAX SAT

(b) MAX CLIQUE

(c) SET COVER

(d) DOMINATING SET

(e) NODE COVER

(f) FEEDBACK NODE SET

(g) FEEDBACK ARC SET

(h) MAX SUBSET SUM
A NPOP complete by transformation problem.

In the note of Knuth [Kn 74] a distinction is made between NP complete and NP complete by transformation problems, where the former is a set such that if it is in P then P = NP, and the later is a set such that all NP sets can be reduced to it by a polynomial time reduction.

A similar distinction can be made for NPOP's where a NPOP complete by transformation problem is a problem such that all NPOP's can be reduced to it by a measure preserving (polynomial) reduction. (This kind of reduction is chosen due to its properties with regard to approximation algorithms - see Lemma 6.) For the NPOP case it follows from the preceding sections that if P ≠ NP then the NPOP complete by transformation set is properly included in the NPOP complete set. Such a (proper) inclusion has not been yet proved (or disproved) for the NP case.

Following the same note of Knuth, we define an "NPOP hard problem" as an optimization problem such that a polynomial solution to it would imply that P = NP. A NPOP hard problem does not have to be a NPOP.

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} ≤ (A,t)_{EXT} where EXT = MAX or EXT = MIN. Such an NPOP is 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.
Theorem of Cook

Let $T$ be an NDTM, and let $f: \mathbb{N} \to \mathbb{N}$ be a (polynomial time) function, $f(n) \geq n$. Then there exists a function $g: \Sigma^* \to \Sigma^*$ that satisfies the following conditions:

1. $g(w) \in \text{SAT} \iff w$ is accepted by $T$ within $f(|w|)$ steps.
2. The time complexity of $g$ is $p(f(|w|))$ 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 $(W(CNF), t_{COM})$ be a set and corresponding $t$-function where:

$W(CNF)$ is the set of all logical formulas in Conjunctive Normal Form over some set of variables $X$, combined with a weight function $W: X \to \mathbb{Z}$.

For a given $a \in W(CNF)$, we define $t_{COM}(a)$ as follows:

Let $B_a = \{B_1, B_2, \ldots, B_n\}$ where $B_i: X_a \to \{0,1\}$ is a valuation of the set $X_a$ of the variables appearing in $a$ ($B_i(\sigma) = 1 \iff B_i(\overline{\sigma}) = 0$).

Define a function $M_{COM}: B_a \to \mathbb{Z} \cup \{\infty\}$ as

$M_{COM}(B) = \begin{cases} 
\infty & \text{if } B \text{ does not satisfy the logical formula } a \\
\Sigma_{x \in X_a} W(x)B(x) & \text{else}.
\end{cases}$

then: $t_{COM}(a) = \bigcup_{B \in B_a} \{M_{COM}(B)\}$.

Definition 7. An "NP measure function" is a function

$\mu: \Sigma^* \to \mathbb{P}[\Sigma^* \cup \{\infty\} \cup \mathbb{N}]$ that can be computed by a nondeterministic polynomial time Turing machine. ("NP measure machine".) (The machine $M_A$ of Theorem 1 is an NP measure machine.) Let $(A,F,t,\mu)_{EXT}$ be a NPOP.

By Definition 1 there exists an NP measure machine, $T$, such that for
each $a \in A$, $k \in \mu(F(a)) \iff$ 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 $k = \sigma_1\sigma_2 \ldots \sigma_i$, $\sigma_i = 0$ or $\sigma_i = 1$, then the output on the tape will be $\sigma_i \sigma_{i-1} \ldots \sigma_1$.

**Theorem 5. (Cook theorem for NPOP's):** Let $T$ be a nondeterministic measure machine, and let $\delta : \mathbb{Z} \rightarrow \mathbb{Z}$ be a recursive (polynomial time) function ($\delta(n) \geq n$). Then there exists a recursive function $g: \Sigma^* \rightarrow \Sigma^*$ such that:

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

2. The time complexity of $g$ is $p(\delta(l(w)))$, where $p(n) < \Theta(n^4)$ is some fixed polynomial.

**Proof.** WLG 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 $C(1,1,\delta(l(w)))$ be the variables in Cook's reduction which asserts that the symbol $1$ is written in cell $i$ at time $\delta(l(w))$, $i = 0,1 \ldots, \delta(l(w))$. Now, for all $x \in X$:

$$W(x) = \begin{cases} 2^i & \text{if } x = C(i,1,\delta(l(w))) \\ 0 & \text{else.} \end{cases}$$
We now claim that:

1. For $a \in \Sigma^*$, $g(a) \in \text{SAT} \iff$ on input $a$, $T$ halts in an accepting state within $\delta(l(w))$ steps. (This is, in fact, Cook's Theorem.)
2. For $a \in \Sigma^*$, $k \in \text{M}_{\text{COM}}(g(a)) \iff$ on input $a$, there exists a legal computation of $T$ which terminates in an accepting state within time $\delta(l(a))$, with $k$ written on its tape, in reverse order, in binary digits.

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

Q.E.D.

Remark 1. The time required for the above reduction differs from that of Cook's original reduction by at most $O(\delta(l(a))^2)$ steps required to define the weight function $W$.

Remark 2. It can be shown that every NPOP can be reduced by a constructive measure preserving reduction to $(W(CNF), t_{COM}^*)$ by requiring that the NP measure machine will write not only $\Pi^*(a)$ but also $\Pi^*(a)$ at the end of the computation. (The details are similar to those of Theorem 2.) The reader is referred to [HB 76] for related topics.

Other properties of reduction between NPOP's will be discussed in a forthcoming paper.

4. P-APPROXIMATION FOR NPOP

The last section of the paper will deal with the problem of approximating NPOP's in polynomial time.

Definition 8. A function $h: \Sigma^* \rightarrow \mathbb{Z} \cup \{\pm \infty\} \cup \{a\}$ is a p-approximation for an NPOP $(A, t)_{\text{EXT}}$ iff $h$ is a polynomial time function.
function satisfying the following properties:

(1) \( h(w) = \alpha \) \iff \( w \notin A \);

(2) \( h(a) \geq \text{op}(a) \) \iff \( \text{EXT} = \text{MIN} \) and \( h(a) \leq \text{op}(a) \) \iff \( \text{EXT} = \text{MAX} \).

**Definition 8.** A function \( h^C : \mathbb{Z}^* \to \mathbb{Z}^* \) is a constructive p-approximation for an \( \text{NPOP}(A, F, t, \mu)^{\text{EXT}} \) iff \( h^C \) is a polynomial (in the length of \( a \)) time function satisfying the following properties:

(1) \( h^C(w) = \alpha' \) \iff \( w \notin A \) (\( \alpha' \) is a string not in \( A \));

(2) \( (\forall a \in A) \ h^C(a) \in F(a) \).

In what follows, we shall state the results for both constructive and non-constructive cases, but the proof for the constructive case will be omitted, in general.

The performance of a p-approximation \( h \) can be defined as follows (see [Sa 76]):

\[
(\forall a, a \neq \text{Ph}(A, t)^{\text{EXT}}(a)) = \left\{ \frac{|h(a) - \text{op}(a)|}{\min(h(a), \text{op}(a))} \right\}.
\]

And as a function of the length of the input the performance is defined as:

\[
(\forall n \in \mathbb{Z}) \text{Ph}(a, t)^{\text{EXT}}(n) = \max \{ \text{Ph}(a, t)^{\text{EXT}}(a) \mid \|a\| \leq n \}. \quad (\|a\| = \text{length of } a).
\]

**Definition 9.** An \( \text{NPOP}(A, t)^{\text{EXT}} \) is \( \delta(n) \) p-approximable if there is a p-approximation function \( h \) for \( (A, t)^{\text{EXT}} \) such that \( \text{Ph}(A, t)^{\text{EXT}}(n) \leq \delta(n) \) for all \( n \in \mathbb{Z} \).

An \( \text{NPOP}(A, t)^{\text{EXT}} \) is p-approximable iff for any \( \varepsilon > 0 \) there is a p-approximation function \( h \) for \( (A, t)^{\text{EXT}} \) such that \( \text{Ph}(A, t)^{\text{EXT}}(a) \leq \varepsilon \) for all \( a \in A \).

\( (A, t)^{\text{EXT}} \) is fully p-approximable iff for any \( \varepsilon > 0 \) there is a p-approximating function \( h \) as above with the additional property that \( h \) can be computed in polynomial time \( Q \) where \( Q = Q(\|a\|, \frac{1}{\varepsilon}) \), i.e., \( Q \) is polynomial in both the length of \( a \) and the value \( \frac{1}{\varepsilon} \).
(A,t)_{\text{EXT}} \text{ is constructively (fully) p-approximable if the above } h \text{ is a constructive p-approximation.}

The importance of measure preserving reductions follows from the following:

Lemma 6. If \((A,F_1,t_1,\mu_1)_{\text{EXT}} \xrightarrow{P} (B,F_2,t_2,\mu_2)_{\text{EXT}}\) then the following holds true: If \((B,t_2)_{\text{EXT}}\) is (fully) p-approximable then so is \((A,t_1)_{\text{EXT}}\).

If \(g\) is a constructive reduction and \((B,t_2)_{\text{EXT}}\) is constructively (fully) p-approximable then so is \((A,t_1)_{\text{EXT}}\).

Proof. Let the time complexity of \(g\) be \(P_0(n)\), for some polynomial \(P_0\). Then, by definition, for all \(a \in A\), \(1(g(a)) \leq P_0(1(a))\).

Assume that \((B,t_2)_{\text{EXT}}\) is fully p-approximable in \(P(l(a),\frac{1}{\epsilon})\) time for some polynomial \(P\). One can assume that \(P\) is nondecreasing in both its variables, otherwise the negative terms may be omitted. We must show that \((A,t_1)_{\text{EXT}}\) is fully p-approximable in \(P'(l(a),\frac{1}{\epsilon})\) time for some (other) polynomial \(P'\).

Let \(a \in A\) and \(\epsilon > 0\) be given. Then we can find an \(\epsilon\)-approximation to \(\text{op}(a)\) using the following algorithm:

(1a) Reduce \(a\) by \(g\) to \(b \in B\);

(1b) Find an \(\epsilon\)-approximation to \(\text{op}(b) = \text{op}(a)\) (\(g\) is measure preserving)

Due to the fact that for measure preserving reductions \(\text{EXT}_1 = \text{EXT}_2\), every \(\epsilon\)-approximation to \(\text{op}(b)\) is also an \(\epsilon\)-approximation to \(\text{op}(a)\) (both approximations will have the same value and \(\text{op}(a) = \text{op}(b)\)). The time required by Step (1a) of this algorithm is bounded by \(P_0(1(a))\) and the time required by Step (1b) of the algorithm is bounded therefore by
Remark: This proof will fit also the $p$-approximation case with some minor changes which are left to the reader.

The "constructive" part of the lemma follows from the fact that, if $(B,t_2)_{EXT}$ is constructively fully $p$-approximable, then $(lb)$ of the algorithm can be changed to $(lb')$: Find $b' \in F_2(b)$ such that $\mu_2(b')$ is an $\varepsilon$ approximation to $\mu_2(b)$.

$g$ being a constructive reduction, one can find (in a polynomial time) an $a' \in F_1(a)$ such that, $\mu_1(a') = \mu_2(b')$ and since $\mu_1(a) = \mu_2(b)$, $\mu_1(a')$ is an $\varepsilon$-approximation to $\mu_1(a)$.

Remark 1: See remark after Theorem 4.

Remark 2: Lemma 6 can be extended in two ways:

(a) If $(B,t_2)_{EXT}$ is $C_p$-approximable for some constant $C$ then so is $(A,t_1)_{EXT}$.

(b) Let $\phi(n)$ be a function that satisfies the following:

For each $k \in \mathbb{Z}$, there exists a constant $C_k$ such that for all $n \in \mathbb{Z}$, $\phi^k(n) \leq C_k$.

(e.g. $\phi(n) = (\log n)^r$ is such a function with $C_k = k^r$.

Then if $(B,t_2)_{EXT}$ is $\phi(n)$-approximable, then $(A,t_1)_{EXT}$ is $O(\phi(n))$-approximable. The proof is omitted and this paper will deal only with $p$-approximation and fully $p$-approximation.

Some results concerning $p$-approximable, and in particular $p$-approximable NPOP's, are represented below.
4.1 Necessary Condition for p-Approximability

Theorem 7. If \((A,t)_{EXT}\) is p-approximable then \((A,t)_{EXT}\) is simple.

Proof. Let \((A,t)_{EXT}\) be p-approximable, and let \(k \in \mathbb{Z}\) be given.

Then \((A,t)_{EXT,k}\) is in P: by definition, for each \(\varepsilon > 0\) there is a polynomial (time) function \(h : \mathbb{E}^* + \varepsilon \cup \{\pm\} \cup \{a\}\) such that \(\forall a \in A,

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

Let \(EXT = \text{MAX}\). (The other case is similar and is omitted.)

\(h_\varepsilon(a)\) and \(op(a)\) are integers by definition and \(h_\varepsilon(a) \leq op(a)\). Thus, \(h_\varepsilon(a) > k\) implies that \(op(a) > k\). On the other hand, choosing \(\varepsilon = \frac{1}{k}\), the inequality

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

implies that \(\frac{op(a) - h_\varepsilon(a)}{h_\varepsilon(a)} < \frac{1}{k}\) or \(\frac{op(a)}{h_\varepsilon(a)} - 1 < \frac{1}{k}\)

and for \(h_\varepsilon(a) \leq k\) this inequality is impossible unless \(op(a) = h_\varepsilon(a)\).

It follows that:

\[
[h_\varepsilon(a) \leq k] \iff [op(a) \leq k].
\]

In other words, \(h_\varepsilon\) is polynomial function that recognizes \((A,t)_{EXT,k}\).

Q.E.D.

It can be shown that the converse of Theorem 7 is not true, and that there are some simple NPOP's which are not p-approximable (the TSP\(^{*}\) problem [PS 76] is an example), assuming \(P \neq NP\).

\(^{*}\) See Appendix.
4.2 Necessary Condition for Fully p-Approximability

Definition 10. \((A,t)_{\text{EXT}}\) is p-simple iff there is some polynomial \(Q(x,y)\) such that \(\forall k \in \mathbb{Z}, (A,t)_{\text{EXT},k}\) is recognizable in \(Q(l(a),k)\) time.

Theorem 8. \((A,t)_{\text{EXT}}\) is fully p-approximable implies that \((A,t)_{\text{EXT}}\) is p-simple.

Proof. Let \((A,t)_{\text{EXT}}\) be fully p-approximable and let \(k \in \mathbb{Z}\) be given. Then \((A,t)_{\text{EXT},k}\) 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 \((A,t)_{\text{EXT}}\) is \(\varepsilon\)-p-approximable in \(Q'(l(a),\frac{1}{\varepsilon})\) time, choosing \(\varepsilon = \frac{1}{k}\), \((A,t)_{\text{EXT}}\) is \(\frac{1}{k}\)-p-approximable in \(Q'(l(a),k)\) time, and applying the same argument as in Theorem 4 we see that \((A,t)_{\text{EXT},k}\) is recognizable in \(Q'(l(a),k)\) time (that is: \(Q = Q'\)).

Q.E.D.

Remark. If \(P \neq \text{NP}\) then p-simplicity is not a sufficient condition for fully p-approximability, as can be shown by the following NPOP "modified MAX SUB SET SUM": \((I S,t'_{ss})_{\text{MAX}}\), which is similar to max subset sum\(^*\) with one exception: For an integer sequence \((a_1,a_2 \ldots , a_n,a_{n+1})\), \(t'_{ss}\) is defined by:

\[
t'_{ss}((a_1,\ldots ,a_{n+1})) = \{k | k \text{ divides } a_{n+1}, \text{ and } k = \sum_{j=1}^{m} a_{i_j} \text{ for some sequence } 1 \leq i_1 < \ldots < i_m \leq n\}.
\]

If \(k\) divides \(a_{n+1}\) and \(k \neq a_{n+1}\), then \(\frac{(a_{n+1})-k}{k} \geq 1\). That means that for \(\varepsilon < 1\) any \(\varepsilon\) approximation for this problem will solve the NP complete

\(^*\) See Appendix.
Knapsack problem ([Ka 72]). Hence, if \( P \neq \text{NP} \) then this problem is not p-approximable, and of course not fully p-approximable; but it can be shown that this problem is p-simple (see next section).

**Definition 11.** Let \( \delta: \mathbb{Z} \rightarrow \mathbb{Z} \) be a (recursive) function and let \((A,t)_{\text{EXT}}\) be a NPOP. Then:

\[
(A,t)_{\text{EXT}}, \delta(n) = \{ a \in A \mid \text{op}(a) \leq \delta(l(a)) \}.
\]

Let \((A,t)_{\text{EXT}}\) be a NPOP, and let \( A' \subseteq A \). Then the NPOP induced by \((A,t)_{\text{EXT}}\) on \( A' \) is the NPOP \((A',t)_{\text{EXT}}\). (It is assumed that \( A' \) is polynomial time recognizable).

**Example.** Let \( G' \) be the set of all planar graphs. Then the colorability problem induced on \( G' \) is the colorability problem for planar graphs.

The following lemma introduces a useful tool for recognizing p-simple NPOP's:

**Lemma 9.** \((A,t)_{\text{EXT}}\) is p-simple implies that for any given polynomial \( p_1(n) \), the NPOP induced by \((A,t)_{\text{EXT}}\) on \((A,t)_{\text{EXT}}, p_1(n)\) (namely the NPOP \( ((A,t)_{\text{EXT}}, p_1(n), t)_{\text{EXT}}\)) is polynomially solvable. (Note that the lemma asserts two things: (a) that the problem induced on \((A,t)_{\text{EXT}}, p_1(n)\) is a NPOP (hence \((A,t)_{\text{EXT}}, p_1(n)\) is a set in \( P \); (b) that this NPOP is polynomial solvable.)

**Proof.** \((A,t)_{\text{EXT}}\) is p-simple implies that there exist a (non-decreasing) polynomial \( Q(x,y) \), such that for all \( k \in \mathbb{Z}, (A,t)_{\text{EXT},k} \) is recognizable in \( Q(l(a),k) \) time. By using binary search (see Theorem 3, pp.6) one can show that the problem induced by \((A,t)_{\text{EXT}}\) on \((A,t)_{\text{EXT},k}\) can be
solved in \( Q'(l(a), k) \) time, where \( Q'(l(a), k) \leq 0 \ [Q(l(a), k) \log_2 k] \). This implies that \( ((A, t)_{\mathrm{EXT}}, P_1(n), t)_{\mathrm{EXT}} \) is solvable in \( Q'(l(a), P_1(l(a)) = P_1(l(a)) \) time, where \( P \) is a polynomial.

Q.E.D.

Corollary. Each of the following simple NPOP's cannot be fully p-approximable if \( P \neq \text{NP} \).

(1) SET COVER
(2) MAX CLIQUE
(3) DOMINATING SET
(4) NODE COVER
(5) FEED BACK ARC SET
(6) FEED BACK NODE SET
(7) MAX SAT
(8) STEINER TREE
(9) MAX CUT

Proof. By Theorem 8, in previous page, it is suffice to show that each of the above problems is not p-simple.

By the preceding Lemma 9, that can be done by showing that for some polynomial \( p(n) \), each of the above problems satisfies the following:

Let \( (A, t)_{\mathrm{EXT}} \) be one of those problems, then the problem induced by \( (A, t)_{\mathrm{EXT}, n} \) is NPOP hard (that is: a polynomial solution of it implies \( P = \text{NP} \)).

Choosing \( p(n) = n \) it is easily checked that each of the problems (1)-(7) satisfies \( (A, t)_{\mathrm{EXT}, n} = A \), and therefore the problems induced on \( (A, t)_{\mathrm{EXT}, n} \) are equal to the original problems which are easily shown to be NPOP complete. As for the problems (8)-(9) it is known that both of them remain NPOP complete even if we restrict the weight function \( W \) to be \( W(i, j) = 1 \) for all \( (i, j) \in A \), (for (9) see [GJS 74]), and with this restriction, \( \text{op}(G) \leq 1(G) \) for all \( G \in G \). Moreover, in these cases...
Definition 12: \((A,t)^\text{EXT}\) is condensable iff there exists a polynomial \(E(n)\) and a polynomial time function \(d: A \times Z \rightarrow A\), denoted by \(d(a,c) = a_c\), that satisfies the following:

\[
\begin{align*}
(4.3.1) & \quad \frac{\text{op}(a_c)}{c} \leq \frac{\text{op}(a)}{c} \leq \text{op}(a_c) + E(l(a)) \quad \text{if} \quad \text{EXT} = \text{MAX} \\
(4.3.1') & \quad \frac{\text{op}(a_c)}{c} \geq \frac{\text{op}(a)}{c} \geq \text{op}(a_c) - E(l(a)) \quad \text{if} \quad \text{EXT} = \text{MIN}.
\end{align*}
\]

\(E(n)\) will be referred to as the "error function".

Example. The TSP problem is condensable, where for a given weighted graph \(W(G)\), \(d(W(G),c)\) is the same graph \(G\) with a new weight function \(W_c\), defined by \(W_c(i,j) = \left\lfloor \frac{W(i,j)}{c} \right\rfloor\) \((*)\). \(E(n) = n\) is a proper error function.

Theorem 10. Let \((A,t)^\text{EXT}\) satisfy the following:

1. \((A,t)^\text{EXT}\) is p-simple;
2. \((A,t)^\text{EXT}\) is condensable.

Then \((A,t)^\text{EXT}\) is fully p-approximable.

Proof. We assume \(\text{EXT} = \text{MAX}\), the proof is the same for \(\text{EXT} = \text{MIN}\) with minor changes. \((A,t)^\text{EXT}\) is p-simple + there exists a polynomial \(Q'(x,y)\) (which is nondecreasing in both its variables) such that the NPOP induced by \((A,t)^\text{EXT}\) on \((A,t)^\text{EXT}',k\) is solvable in \(Q'(l(a),k)\) time. In particular, \((A,t)^\text{EXT},k\) is recognizable in \(Q'(l(a),k)\) time.

\((*)\) For a real number \(r\), \([r]\) stands for the least integer not smaller than \(r\), and \(\lfloor r\rfloor\) stands for the greatest integer not greater than \(r\).
the set \((A,t)_{\text{EXT},n}\) is a NP complete set. Thus, \(P \neq \text{NP}\) implies that the problems induced by these NPOP's on \((A,t)_{\text{EXT},n}\) are not necessarily NPOP's, and hence are NPOP hard but not necessarily NPOP complete.

While the above NPOIS are simple but not p-simple, we will prove that "MAX SUBSET SUM" is p-simple. For other examples, the reader is referred to \([Sa 76]\).

Let \(a = (a_1, \ldots, a_n) \in (Z)^{n+1}\) be an input to "MAX SUBSET SUM". Then the following algorithm will solve the problem: "is \(a \in (A,t)_{\text{EXT},k}\)"

in \(O(\ell(a) \cdot k)\) time units. The algorithm contains a variable "\(T\)" which is the set of all "feasible solutions", and at the end of the algorithm

\[ T = t_{ss}(a), \]

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

The algorithm checks, for given \(a\), whether \(a \in (A,t)_{\text{EXT},k}\) for this NPOP and its time complexity is \(O(n \cdot k) = O(1(a) \cdot k)\). This follows from the fact that \(|T| \leq k\) all through the execution of the algorithm.

4.3 Sufficient Conditions for Fully p-Approximability

We now introduce a property of NPOP's, that together with p-simplicity provides a sufficient (but not necessary) conditions for fully p-approximability.
(A,t)_{EXT} \text{ is condensable} \implies \text{there exists a polynomial } E(n) \text{ and a function }
\begin{align*}
d: A \times \mathbb{Z} &\rightarrow A \text{ as in Definition 12.} 
\end{align*}

Let \( \varepsilon \) be given, for simplicity we assume \( \varepsilon = \frac{1}{n} \) for some \( n \geq 1 \)
so that \( \frac{k}{\varepsilon} \) is an integer for every integer \( k \). On input \( a \in A \), we per-
form the following algorithms which contain 3 variables: "c" is an integer
of the form \( 2^n \). "b" is a word, \( b = a_c \), and \( h \) is the \( \varepsilon \) approximation
of \( \text{op}(a) \).

begin

(1) \( c+1 \), \( b+a \), \( \varepsilon' \approx 2 \varepsilon / 2 \); 
(2) if \( b \in (A,t) \) then go to (6) ; 
(3) \( c+2c \); 
(4) \( b+a_c \); 
(5) go to (2) ; 
(6) find \( \text{op}(a_c) \); \( h + c \cdot \text{op}(a_c) \) end.

We have to prove two facts: that \( h \) is an \( \varepsilon \)-approximation to
\( \text{op}(a) \), and that the algorithm requires \( p(1(a), \frac{1}{\varepsilon}) \) time for some polynomial \( p \).

Case (a): \( \text{op}(a) \ll \frac{2}{\varepsilon} E(1(a)) \). In this case the algorithm finds \( h = \text{op}(a) \) \( \text{in} \)
\( Q'(1(a), \frac{2}{\varepsilon} E(1(a))) = p(1(a), \frac{1}{\varepsilon}) \) time.

Case (b): \( \text{op}(a) > \frac{2}{\varepsilon} E(1(a)) \). Let \( c_{\varepsilon'} \) be first the value of the variable
\( c \) when \( b(= a_{c_{\varepsilon'}}) \in (A,t) \).

\begin{align*}
\text{Since } a_{c_{\varepsilon'}} \notin (A,t) &\text{ we have that } \frac{2}{\varepsilon} E(1(a)) < \text{op}(a_{c_{\varepsilon'}}). 
\end{align*}

On the other hand it follows from 4.3.1 that
\[ \text{op}(a_{\epsilon, \epsilon'}) \leq \frac{2 \text{op}(a_{\epsilon, \epsilon'})}{c_{\epsilon, \epsilon'}} \leq 2 \text{op}(a_{\epsilon, \epsilon'}) + 2E(l(a)). \]

Combining those inequalities we get that

\[ \frac{1}{\epsilon'} E(l(a)) \leq \text{op}(a_{\epsilon, \epsilon'}) + E(l(a)) \]
or
\[ \left( \frac{1 - \epsilon'}{\epsilon'} \right) E(l(a)) \leq \text{op}(a_{\epsilon, \epsilon'}) \]
or
\[ \frac{E(l(a))}{\text{op}(a_{\epsilon, \epsilon'})} \leq \frac{\epsilon'}{1 - \epsilon'} \leq 2 \epsilon' = \epsilon. \]

Using again 4.3.1 for \( \epsilon' \) and applying the above inequality we get

\[ 1 \leq \frac{\text{op}(a)}{c_{\epsilon, \epsilon'}, \text{op}(a_{\epsilon, \epsilon'})} \leq 1 + \frac{E(l(a))}{\text{op}(a_{\epsilon, \epsilon'})} < 1 + \epsilon; \]

or
\[ \frac{\text{op}(a) - c_{\epsilon, \epsilon'}, \text{op}(a_{\epsilon, \epsilon'})}{c_{\epsilon, \epsilon'}, \text{op}(a_{\epsilon, \epsilon'})} \leq \epsilon, \]

which shows that \( h \left( c_{\epsilon, \text{op}(a_{\epsilon, \epsilon'})} \right) \) is an \( \epsilon \)-approximation to \( \text{op}(a) \).

As for the timing: since there exists a polynomial \( P_2(n) \) such that for all \( a \in A, \text{op}(a) \leq P_2(l(a)) \), the loop (2)-(5) should be repeated no more than \( P_2(l(a)) \) time. The execution of lines (2) and (6) require no more than \( Q'(\ell(a), \frac{2}{\epsilon} E(\ell(a))) = p_1(l(a), \frac{1}{\epsilon}) \) time each.

The execution of line (4) requires \( p_3(l(a)) \) time for some polynomial \( p_3 \). Hence, the total time required for the execution of the algorithm is

\[ 0(p_2(l(a))[p_1(l(a), \frac{1}{\epsilon}) + p_3(l(a))]) = p(l(a), \frac{1}{\epsilon}) \]

where \( p \) is the required polynomial. Q.E.D.
Remark. In practice the above algorithm can be greatly improved, since for most problems one can easily get a good estimation to $\text{op}(a)$, instead of exhaustingly repeating the loop (2)-(5). (In "MAX SUBSET SUM", for instance, it can be shown that for an input $a = (a_1 \ldots a_n, a_{n+1})$, except for some trivial cases, $\frac{a_{n+1}}{2} < \text{op}(a) \leq a_{n+1}$.

Although the $p$-approximation technique introduced in Theorem 10 is not the most general one (see Remark, p.32) it holds for most of the natural NPOP's which have been shown to be fully $p$-approximable. Using Theorem 8 (for necessary conditions) and 10 (for sufficient conditions) one can show that for most of the natural NPOP's: Either the problem cannot be fully $p$-approximable (if $P \neq \text{NP}$) or it is fully $p$-approximable. There is at least one exception, as far as we know - which is the $k$-Chinese Postman problem ([FHK 76]): $\min \{W(G) \times Z, t_{cp}^k\}$: where for a weighted graph $W(G)$ and an integer $k$, $t_{cp}^k(W(G), k) = \{c \mid \text{there exists } k \text{ cycles } c_1, c_2, \ldots, c_k \text{ that cover all edges of } G, \text{ and } \max_{1 \leq j \leq k} W(c_j) = c \}$. This problem is not known to be fully $p$-approximable, and so far we cannot prove that it cannot be fully $p$-approximable.

It can be shown that this problem is fully $p$-approximable $\iff$ the NPOP induced by this problem of the class of weighted graphs with unit weights ($W(e) = 1 \forall e \in E$), is polynomially solvable.

We now give an analogue of Theorem 10 for $p$-approximation:

**Definition 13.** $(A, t)_{\text{EXT}}$ is strongly condensable iff it is condensable with a constant error function (that is: $E(n) = E_0$ for some constant $E_0$).
Theorem 11. Let \((A,t)_{\text{EXT}}\) satisfy the following:

1. \((A,t)_{\text{EXT}}\) is simple;
2. \((A,t)_{\text{EXT}}\) is strongly condensable.

Then \((A,t)_{\text{EXT}}\) is \(p\)-approximable. The proof is similar to that of Theorem 11, and is omitted.

Although the condition (1) of Theorem 11 is weaker than the corresponding condition of Theorem 10, which would be expected for getting a weaker result, still, the second condition of this theorem is much stronger than the corresponding condition of Theorem 10. This provides perhaps, some intuitive explanation to the fact that all natural NPOP's that are known to be \(p\)-approximable are in fact fully \(p\)-approximable.

Remark. Farther results concerning (fully) \(p\)-approximability, such as:

(a) A necessary and sufficient condition for (fully) \(p\)-approximability;
(b) Some fully \(p\)-approximable NPOP's, that require approximation technique which differ from that of Theorem 10,

will appear in a later paper.
BIBLIOGRAPHY


APPENDIX

The following NPOP's were mentioned in the paper, but were not formally defined:

(1) TSP (Travelling Salesman Problem): $= (W(G), t_{TSP})_{\text{MIN}}$, where $W(G)$ is the set of all weighted graphs $W(G)$, that is graphs combined with a weight function $W: A \rightarrow \mathbb{Z}_+$, and for a given weighted graph $W[G(N,A)]$, $t_{TSP}(W[G(N,A)]) = \{k \mid \text{there exists a Hamiltonian cycle in the graph whose weight is } k\} \cup \{\infty\}$ (we add $\infty$ to $t_{TSP}(W[G(N,A)])$ to make sure that it is not empty).

(2) MAX CUT: $= (W(G), t_{\text{CUT}})_{\text{MAX}}$, where $W(G)$ 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 = $(IS, t_{\text{ss}})_{\text{MAX}}$, where $IS = \{(a_1, \ldots, a_n, a_{n+1})\}$ is the set of all finite integer sequences, and $t_{\text{ss}}((a_1, \ldots, a_n, a_{n+1})) = \{k \mid k < a_{n+1} \text{ and there are } 1 \leq i_1 < \cdots < i_s \leq n, \sum_{j=1}^{s} a_{i_j} = k\}$.

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

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

where

$\delta_{\sigma(i)} = \left\{\begin{array}{ll}
1 & \text{iff } T_{\sigma(i)} + T_{\sigma(i+1)} + \ldots + T_{\sigma(n)} \leq D_{\sigma(i)} \leq D_{\sigma(i+1)} \\
0 & \text{else}
\end{array}\right.$

Technion - Computer Science Department - Technical Report CS0108 - 1977
(5) SET COVER = \( (f, t_{\text{sc}})_{\text{MIN}} \) where: \( f \) is the set of all finite families of finite sets, and for \( \{S_1, \ldots, S_n\} \in f \)
\[
t(\{S_1, \ldots, S_n\}) = \{ i \mid \text{there exists } 1 \leq j_1 < j_2 < \ldots < j_i \leq n \text{ so that } \bigcup_{r=1}^{j_i} S_r = \bigcup_{r=1}^{n} S_r \}.
\]

(6) DOMINATING SET = \( (G, t_{\text{DS}})_{\text{MIN}} \) where for \( G \in G \),
\[
t_{\text{DS}}(G) = \{ k \mid \text{there are } k \text{ nodes in } G \text{ that are adjacent to all other nodes of } G \}.
\]

(7) CLIQUE COVER: \( (G, t_{\text{cc}})_{\text{MIN}} \) where for \( G(V,E) \in G \):
\[
t_{\text{cc}}(G) = \{ k \mid \text{there exists } k \text{ cliques in } G \text{ whose union is } V \}.
\]

(8) FEEDBACK ARC SET \( (D, t_{\text{FBA}})_{\text{MIN}} \) where \( D \) is the set of all directed graphs, and for \( D(V,E) \in D \), \( (V = \text{the set of vertices, } E = \text{the set of edges}) \)
\[
t_{\text{FBA}}(D) = \{ k \mid \text{there exists } k \text{ edges in } A \text{ such that each (directed) cycle in } D \text{ contains at least one of them} \}.
\]

(9) FEEDBACK NODE SET: \( (D, t_{\text{FBN}})_{\text{MIN}} \) where for \( D(V,E) \in D \):
\[
t_{\text{FBN}}(D) = \{ k \mid \text{there exists } k \text{ vertices in } N \text{ such that each (directed) cycle contains at least one of them} \}.
\]

(10) STEINER TREE: \( ((W(G),S), t_{\text{STR}})_{\text{MIN}} \) where \( (W(G),S) \) is the set of all weighted graphs together with a given subset of the nodes of the graph. For a given element \( (W(G),S) \) of this set,
\[
t_{\text{STR}}((W(G),S)) = \{ k \mid \text{there exists a subtree of } G \text{ that contains } S, \text{ whose weight is } k \}.
\]