アルゴリズム 18－3
（1990．11．22）

最適解を求める複雑さについて

陳 致中　戸田 誠之助
電気通信大学情報工学科

解の値が入力の長さの多項式で抑えられる NP 最適化問題 ($NPCOP$) に対し, 最適解を求める計算量について述べる。任意の $NPCOP$ Π に対して, Π の実例を入力として与えられたら, NP オラックルへ一組のパラレル質問をし, その答えを評価して一つの最適解を非常に高い確率で出力する確率アルゴリズムが存在することを示す。更に, 幾つかの自然な $NPCOP$'s に対し, 最適解を計算するどの関数も $\mathrm{PF}_{\mathrm{tt}}^{\mathrm{NP}}$ にある関数と少なくとも同じ難しさを持つことを示す。

ON THE COMPLEXITY OF COMPUTING OPTIMAL SOLUTIONS

Zhi-Zhong Chen　Seinosuke Toda

Department of Computer Science and Information Mathematics
University of Electro-Communications

Chofugaoka 1-5-1, Chofu-shi, Tokyo 182, Japan

We study the complexity of computing optimal solutions for NP optimization problems whose solution costs are bounded above by a polynomial in the length of their instances (we call such an NP optimization problem an $NPCOP$). We first show that for any $NPCOP$ Π, there exists a polynomial-time bounded randomized algorithm which, given an instance of Π, uses one free evaluation of parallel queries to an NP oracle set and outputs some optimal solution with very high probability. We then show that for several natural $NPCOP$'s, any function giving those optimal solutions is at least as hard as all functions in $\mathrm{PF}_{\mathrm{tt}}^{\mathrm{NP}}$.

1 Preliminaries

We assume that the reader is familiar with the basic concepts from the theories of optimization problems and computational complexity. We use $\Sigma = \{0,1\}$ as our alphabet. By a *language* or a *set*, we mean a subset of Σ^*. We denote by $|x|$ the length of a string x. The empty string is denoted by λ. For any finite set A, $||A||$ denotes the number of elements of A and χ_A denotes the characteristic function of A. Let $A^{\leq n}$ and $A^{=n}$ denote the sets $\{x \in A : |x| \leq n\}$ and $\{x \in A : |x| = n\}$, respectively. For any sets A and B, $A \oplus B$ denotes the marked union of A and B; that is, $A \oplus B = \{0x : x \in A\} \cup \{1y : y \in B\}$. The symbol $\oplus$ is also used to denote the exclusive-or operation of Boolean values. We write N for the set of non-negative integers. Let $bin(n)$ be a standard binary representation of non-negative integer n over Σ, and write $\log(n)$ to mean the base 2 logarithm of n for $n > 0$. We assume a standard one-to-one pairing function from $\Sigma^* \times \Sigma^*$ to Σ^* that is polynomial-time computable and polynomial-time invertible. For strings x and y, we denote the output of the pairing function by $\langle x, y \rangle$; this notation is extended to denote any k-tuples for $k > 2$ in a usual manner.

An *optimization problem* Π is a quintuple (op, D, S, R, c), where

(1) $op \in \{max, min\}$ is the *underlying operation* (i.e., maximization or minimization),
(2) D is the set of *instances*,
(3) S is the set of *feasible solutions* ,
(4) $R \subseteq D \times S$ is the *instance-solution relation*, and
(5) $c : D \times S \rightarrow$N is the *solution cost function* (for simplicity, we consider only the case that all costs are non-negative integers).

We call Π a *maximization problem* if $op = max$ and call it a *minimization problem* otherwise. To each instance $x \in D$, we associate a finite subset $S(x) = \{y \in S : R(x,y) \text{ holds true }\}$, and call it the *solution space* of x. The *optimal cost function* $c^* : D \rightarrow$N of Π is defined by

$$c^*(x) = op\{c(x,y) : y \in S(x)\}$$

and the set of optimal solutions for an instance $x \in D$, $\text{optsol}_\Pi(x)$, is defined by

$$\text{optsol}_\Pi(x) = \{y \in S(x) : c(x,y) = c^*(x)\}.$$

The objective in solving a given optimization problem Π is to compute an optimal solution for any instance of Π. We particularly note that examining the complexity of computing optimal solutions but not just giving optimal costs is the essence of this paper.

A *combinatorial optimization problem* is an optimization problem $\Pi = (op, D, S, R, c)$ for which there exists a polynomial p such that for all $x \in D$ and all $y \in S(x)$, $c(x,y) \leq p(|x|)$. We call a combinatorial optimization problem $\Pi = (op, D, S, R, c)$ an *NP combinatorial optimization problem* (*NPCOP* for short) if it satisfies the following additional conditions:

(1) D, S, and R are polynomial-time decidable,
(2) c is polynomial-time computable, and
(3) for some polynomial p, all $x \in D$, and all $y \in S$, $y \in S(x)$ implies $|y| \leq p(|x|)$.

When Π is a maximization (resp., minimization) problem, we see from these conditions that

the problem of deciding whether, given an instance $x \in D$ and a natural number k, the optimal cost $c^*(x)$ is at least (resp., at most) k is decidable in NP. This is the reason why Π is called an NP combinatorial optimization problem. Throughout this paper, we will deal with only NP combinatorial optimization problems.

When discussing the upper bound of the complexity of computing optimal solutions for a given *NPCOP* Π, we may only require an algorithm solving Π to compute *some* optimal solution for any given instance of Π. In particular, when we consider a randomized algorithm solving the *NPCOP*, the algorithm may produce some different optimal solutions depending on random bits used; we will never require the algorithm to compute a single optimal solution independent of random bits used. Only a requirement to randomized algorithms is that for each instance, they must compute some optimal solution *with very high probability.*

On the other hand, when discussing the lower bound of the complexity of computing optimal solutions for a given *NPCOP*, we will examine the relative complexity between solving the *NPCOP* and the other classes of problems, as in the theory of NP-completeness. Intuitively speaking, we want to show that solving the *NPCOP* is at least as hard as a problem whose computational complexity appears to be settled. As mentioned in the introduction, we regard the problem of solving an *NPCOP* as a class of functions giving an optimal solution for any given instance of the *NPCOP*, and the opponents compared with those functions are functions in the class $\mathrm{PF}_{tt}^{\mathrm{NP}}$. Thus, with each *NPCOP* $\Pi = (op, D, S, R, c)$, we associate a class OPTSOL_Π of functions defined by

$$\mathrm{OPTSOL}_\Pi = \{F : D \to S \ : \ (\forall x \in D)[F(x) \in \mathrm{optsol}_\Pi(x)] \ \}.$$

Then, the reducibility notion below will capture the above purpose.

Definition 1.1 Let $\mathbf{F}$ and $\mathbf{H}$ be two classes of functions, and let H be a function. Then, H is *uniformly polynomial-time 1-Turing reducible to* $\mathbf{F}$, in symbols $H \leq_{1\text{-T}}^{\text{uniform-PF}} \mathbf{F}$, if there exist polynomial-time computable functions f and g such that for every function $F \in \mathbf{F}$ and every $x \in \Sigma^*$, $H(x) = g(x, F(f(x)))$. We call the pair $\langle f, g \rangle$ a $\leq_{1\text{-T}}^{\text{uniform-PF}}$-*reduction* of H to $\mathbf{F}$ (note that the reduction $\langle f, g \rangle$ of H to $\mathbf{F}$ must be the same for all functions chosen from $\mathbf{F}$). $\mathbf{F}$ is $\leq_{1\text{-T}}^{\text{uniform-PF}}$-*hard* for $\mathbf{H}$ if every function in $\mathbf{H}$ is $\leq_{1\text{-T}}^{\text{uniform-PF}}$-reducible to $\mathbf{F}$.

If we can show that some OPTSOL_Π is $\leq_{1\text{-T}}^{\text{uniform-PF}}$-hard for $\mathrm{PF}_{tt}^{\mathrm{NP}}$, then we see that computing optimal solutions for the *NPCOP* Π is at least as computationally hard as computing the hardest functions in $\mathrm{PF}_{tt}^{\mathrm{NP}}$.

We finally mention some complexity classes that we deal with in the present paper. Throughout this paper, we mean by an NTM a nondeterministic Turing machine. Let NP be the class of sets accepted by polynomial-time bounded NTM's. A language A is NP-*hard* if for any language B in NP, there exists a polynomial-time computable function f such that for every $x \in \Sigma^*$, $x \in A$ if and only if $f(x) \in B$. $\mathrm{PF}_{tt}^{\mathrm{NP}}$ is the class of all functions F for which there exist a set A in NP and two polynomial-time computable functions g, e such that for all strings x, $F(x) = e(x, \chi_A(y_1), \cdots, \chi_A(y_m))$, where $g(x) = \langle y_1, \cdots, y_m \rangle$. More intuitively, a function F

is in $\mathrm{PF}_{tt}^{\mathrm{NP}}$ if there exist a polynomial-time bounded deterministic oracle transducer (DOTM for short) N and a set $A \in$NP such that N^A computes F and N^A on all inputs prepares all query strings before asking them to the oracle set A. In Section 3, we will use this intuitive definition of $\mathrm{PF}_{tt}^{\mathrm{NP}}$, to describe an algorithm computing a function in $\mathrm{PF}_{tt}^{\mathrm{NP}}$.

2 An upper bound of computing optimal solutions

For any strings x and y in $\{0,1\}^n$, let $x \cdot y$ denotes $(x_1 \wedge y_1) \oplus \cdots \oplus (x_n \wedge y_n)$, where x_i (resp., y_i) denotes the i-th bit of x (resp., y) and $\oplus$ denotes the exclusive-or. Let X be a finite set of strings and Q be a predicate over strings. We denote by $Prob\{w_1, \cdots, w_k \in X : Q(w_1, \cdots, w_k)\}$ the probability that $Q(w_1, \cdots, w_k)$ holds for strings w_1, $\cdots$, w_k chosen randomly from X under uniform distribution. Then, Valiant and Vazirani showed the following result:

Theorem 2.1 [8] Let n be a positive integer. Then, for any subset S of $\{0,1\}^n$,

$$Prob\{\, w_1, \cdots, w_n \in \{0,1\}^n : (\exists j, 0 \le j \le n)[\, \|\{\, y \in S : (\forall i, 1 \le i \le j)[y \cdot w_i = 0]\,\}\| = 1\,]\,\} \ge \tfrac{1}{4}.$$

Theorem 2.2 Let $\Pi = (op, D, S, R, c)$ be an $NPCOP$ and let e be any polynomial. Then, there exist a function $G \in \mathrm{PF}_{tt}^{\mathrm{NP}}$ and a polynomial r such that for all $x \in D$ with $|x| = n$,

$$Prob\{w \in \{0,1\}^{r(n)} : G(x,w) \in \mathrm{optsol}_\Pi(x)\} \ge 1 - 2^{-e(n)}.$$

Proof We consider only the case that Π is a maximization problem. The other case is quite similar. For simplicity, we assume that there exists a polynomial q such that for all $x \in D$ and all $y \in S$, $y \in S(x)$ implies $|y| = q(|x|)$. We lose no generality under this assumption. Let p be a polynomial such that for all $x \in D$ and all $y \in S(x)$, $c(x,y) \le p(|x|)$. Then we first define two sets A and B as follows:

$$\begin{aligned} A = \{&\langle x, i\rangle : i \ge 0, x \text{ has a solution with cost } i\} \\ B = \{&\langle x, i, j\rangle : x \in D,\ 0 \le i \le p(|x|),\ 1 \le j \le q(|x|), \text{ and} \\ &\quad \text{there exists a } y \in S(x) \text{ with cost } i \text{ such that the } j\text{-th symbol of } y \text{ is } 1\} \\ \cup\{&\langle x, i, j, w_1, \cdots, w_k\rangle : x \in D,\ 0 \le i \le p(|x|),\ 1 \le j \le q(|x|),\ 1 \le k \le q(|x|), \\ &\quad w_1, \cdots, w_k \in \{0,1\}^{q(|x|)}, \text{ and there exists a } y \in S(x) \text{ with cost } i \text{ such} \\ &\quad \text{that the } j\text{-th symbol of } y \text{ is 1 and } y \cdot w_1 = \cdots = y \cdot w_k = 0\}. \end{aligned}$$

Obviously, A and B are in NP. Thus, we have $A \oplus B \in$NP. We also define the polynomial r by $r(n) = 3 \cdot e(n) \cdot q(n)^2$.

Below, we define a deterministic oracle transducer N which uses $A \oplus B$ as an oracle set. Given an instance $x \in D$ with $|x| = n$ and a string $w \in \{0,1\}^{r(n)}$, N operates as follows:

Step 1. N computes the optimal cost $c^*(x)$. This is done by asking the queries $\langle x, 0\rangle$, $\langle x, 1\rangle$, $\cdots$, $\langle x, p(n)\rangle$ to the oracle set A and computing the largest integer k $(= c^*(x))$ such that $\langle x, k\rangle$ is in A.

Step 2. Let w_1, w_2, $\cdots$, $w_{3e(n)}$ be the strings in $\{0,1\}^{q(n)^2}$ such that $w_1 w_2 \cdots w_{3e(n)} = w$, and for every l, $1 \leq l \leq 3e(n)$, let $w_{l,1}$, $\cdots$, $w_{l,q(n)}$ be the strings in $\{0,1\}^{q(n)}$ such that $w_{l,1} \cdots w_{l,q(n)} = w_l$. Then, by asking queries to the oracle set B, N computes strings $s_{i,l,m}$ as follows: for all i, l, m such that $0 \leq i \leq p(n)$, $1 \leq l \leq 3e(n)$, and $0 \leq m \leq q(n)$,

(a) $s_{i,l,0} = \chi_B(\langle x,i,1\rangle)\chi_B(\langle x,i,2\rangle)\cdots\chi_B(\langle x,i,q(n)\rangle)$ and

(b) $s_{i,l,m} = \chi_B(\langle x,i,1,w_{l,1},\cdots,w_{l,m}\rangle)\cdots\chi_B(\langle x,i,q(n),w_{l,1},\cdots,w_{l,m}\rangle)$ for $1 \leq m \leq q(n)$.

Step 3. Let $k = c^*(x)$. If $s_{k,l,m} \in S(x)$ for some l and m, then N outputs the string $s_{k,l,m}$; otherwise, N outputs nothing (in this case, the function computed here is supposed to be "undefined" on x and w).

Let G denote the function computed by $\mathrm{N}^{A\oplus B}$. We can easily see that N is polynomial-time bounded and each query string is prepared independently of the other query strings; hence, the query strings made by N on input $\langle x,w\rangle$ can be realized as parallel queries to the oracle set $A \oplus B$. Thus, G is in $\mathrm{PF}_{\mathrm{tt}}^{\mathrm{NP}}$. To show that G satisfies the theorem, we first show the following claim:

Claim Suppose that for some l and m, $||\{y \in \mathrm{optsol}_\Pi(x) \ : \ y \cdot w_{l,1} = y \cdot w_{l,2} = \cdots = w_{l,m} = 0\}|| = 1$. Then $s_{k,l,m}$ is an optimal solution of x, where $k = c^*(x)$.

Proof of Claim. Let y be the unique optimal solution of x in the set $\{y \in \mathrm{optsol}_\Pi(x) : y \cdot w_{l,1} = \cdots = y \cdot w_{l,m} = 0\}$. Then we see, from the definition of B, that for all $1 \leq j \leq q(|x|)$, $\chi_B(\langle x,k,j,w_{l,1},\cdots,w_{l,m}\rangle) = 1$ iff the j-th bit of y is 1. Thus, we have $s_{k,l,m} = y$. ∎

From this claim and Theorem 2.1, we have that for all $x \in D$ with $|x| = n$,

$$
\begin{aligned}
&Prob\{w \in \{0,1\}^{r(n)} \ : \ G(x,w) \in \mathrm{optsol}_\Pi(x)\} \\
&= Prob\{w \in \{0,1\}^{r(n)} \ : \ (\exists l,m, 1 \leq l \leq 3e(n), 0 \leq m \leq q(n))[s_{c^*(x),l,m} \in \mathrm{optsol}_\Pi(x)]\} \\
&\geq Prob\{w \in \{0,1\}^{r(n)} \ : \ (\exists l,m, 1 \leq l \leq 3e(n), 0 \leq m \leq q(n)) \\
&\qquad\qquad [\ ||\{y \in \mathrm{optsol}_\Pi(x) \ : \ y \cdot w_{l,1} = \cdots = y \cdot w_{l,m} = 0\}|| = 1]\} \\
&\geq 1 - \textstyle\prod_{l=1}^{3e(n)} Prob\{w_{l,1},\cdots,w_{l,q(n)} \in \{0,1\}^{q(n)} \ : \ (\forall m, 0 \leq m \leq q(n)) \\
&\qquad\qquad [\ ||\{y \in \mathrm{optsol}_\Pi(x) \ : \ y \cdot w_{l,1} = \cdots y \cdot w_{l,m} = 0\}|| \neq 1\ ]\} \\
&\geq 1 - (\tfrac{3}{4})^{3e(n)} \ = \ 1 - (\tfrac{27}{64})^{e(n)} \ \geq \ 1 - 2^{-e(n)}.
\end{aligned}
$$

Thus, we have this theorem. ∎

3 Hardness of computing optimal solutions

In this section, we first give a sufficient condition for showing that for a given *NPCOP* Π, OPTSOL_Π is $\leq_{1-\mathrm{T}}^{\mathrm{uniform-PF}}$-hard for $\mathrm{PF}_{\mathrm{tt}}^{\mathrm{NP}}$. We use the following notions in the general result.

Definition 3.1 Let $\Pi = (op, D, S, R, c)$ be an optimization problem. Then, we define the *decision problem* L_Π *associated with* Π as follows:

$$L_\Pi = \{\langle x,k\rangle \ : \ x \in D, \ k \text{ is a non-negative integer, and } c^*(x) \ \theta \ k\},$$

where θ is $\leq$ (less than or equal to) if $op = min$, and θ is $\geq$ (greater than or equal to) otherwise. Π is said to be *linearly paddable* if there exist two polynomial-time computable functions $f_1 : D \times D \to D$ and $f_2 : D \times D \times S \to S \times S$ such that

(a) for all $x_1, x_2 \in D$, $|f_1(x_1, x_2)| = O(|x_1| + |x_2|)$, and

(b) for all $x_1, x_2, x \in D$, and all $H \in \mathrm{OPTSOL}_\Pi$, if $x = f_1(x_1, x_2)$ and $f_2(x_1, x_2, H(x)) = \langle y_1, y_2 \rangle$, then $y_1 \in \mathrm{optsol}_\Pi(x_1)$ and $y_2 \in \mathrm{optsol}_\Pi(x_2)$.

Theorem 3.1 Let $\Pi = (op, D, S, R, c)$ be a linearly paddable *NPCOP* whose associated decision problem L_Π is NP-hard. Then, OPTSOL_Π is $\leq_{1\text{-}\mathrm{T}}^{\mathrm{uniform\text{-}PF}}$-hard for $\mathrm{PF}_{\mathrm{tt}}^{\mathrm{NP}}$.

Proof We define a function Q_Π as follows:

$$Q_\Pi(\langle x_1, k_1 \rangle, \langle x_2, k_2 \rangle, \cdots, \langle x_m, k_m \rangle) = \chi_{L_\Pi}(\langle x_1, k_1 \rangle)\chi_{L_\Pi}(\langle x_2, k_2 \rangle) \cdots \chi_{L_\Pi}(\langle x_m, k_m \rangle).$$

Since L_Π is NP-hard, we easily see that for all $F \in \mathrm{PF}_{\mathrm{tt}}^{\mathrm{NP}}$, there exist two polynomial-time computable functions g_1 and g_2 such that for all $x \in \Sigma^*$, $F(x) = g_2(x, Q_\Pi(g_1(x)))$. Thus it suffices to show that Q_Π is $\leq_{1\text{-}\mathrm{T}}^{\mathrm{uniform\text{-}PF}}$-reducible to OPTSOL_Π.

Let $\langle x_1, k_1 \rangle, \cdots, \langle x_m, k_m \rangle$ be arbitrary arguments to Q_Π. For simplicity, we assumt that $m = 2^l$ for some $l \geq 0$; otherwise, we may add some dummy arguments to the original ones so that the resulting number of arguments becomes a power of 2. Let f_1 and f_2 be two polynomial-time computable functions witnessing that Π is linearly paddable. That is,

(a) for all $x_1, x_2 \in D$, $|f_1(x_1, x_2)| = O(|x_1| + |x_2|)$, and

(b) for all $x_1, x_2, x \in D$ and all $H \in \mathrm{OPTSOL}_\Pi$, if $x = f_1(x_1, x_2)$ and $f_2(x_1, x_2, H(x)) = \langle y_1, y_2 \rangle$, then $y_1 \in \mathrm{optsol}_\Pi(x_1)$ and $y_2 \in \mathrm{optsol}_\Pi(x_2)$.

Then we inductively define $2m - 1$ instances $I_i^{(j)}$ of Π as follows:

(1) $I_i^{(0)} = x_i$ for all i satisfying $1 \leq i \leq m$, and

(2) $I_i^{(j)} = f_1(I_{2i-1}^{(j-1)}, I_{2i}^{(j-1)})$ for all j and i satisfying $1 \leq j \leq l$ and $1 \leq i \leq 2^{l-j}$.

Furthermore, we define a function g as follows:

$$g(\langle x_1, k_1 \rangle, \cdots, \langle x_m, k_m \rangle) = I_1^{(l)}.$$

Then, we easily see by induction on j that for some constant $d > 0$, all j, and all i, $|I_i^{(j)}| \leq d^j \cdot (\sum_{h=1}^m |x_h|)$. Hence, we have that $|I_1^{(l)}|$ is bounded above by a polynomial in $\sum_{h=1}^m |x_h|$. From this and the polynomial-time computability of f_1, we see that g is polynomial-time computable. Let H be an arbitrary function in OPTSOL_Π and let $H(I_1^{(l)}) = s^{(l)}$. Next, we show that from $x_1, \cdots, x_m$ and $s^{(l)}$, we can compute optimal solutions for the instances $x_1, x_2, \cdots, x_m$ in time polynomial in $\sum_{h=1}^m |x_h|$. By using the function f_2, we can compute optimal solutions $s_1^{(l-1)}$ and $s_2^{(l-1)}$ for $I_1^{(l-1)}$ and $I_2^{(l-1)}$, respectively, from $s^{(l)}$, $I_1^{(l-1)}$, and $I_2^{(l-1)}$. Using f_2 again, we can further compute optimal solutions $s_1^{(l-2)}$, $s_2^{(l-2)}$, $s_3^{(l-2)}$, and $s_4^{(l-2)}$ for $I_1^{(l-2)}$, $I_2^{(l-2)}$, $I_3^{(l-2)}$, and $I_4^{(l-2)}$, respectively, from $s_1^{(l-2)}$, $s_2^{(l-2)}$, $I_1^{(l-2)}$, $\cdots$, $I_4^{(l-2)}$. By using f_2 repeatedly in this way, we can finally obtain optimal solutions $s_1^{(0)}$, $\cdots$, $s_m^{(0)}$ for the instances $x_1, \cdots, x_m$, respectively. Since f_2 is polynomial-time computable and is used $2^l - 1 = m - 1$ times in the whole computations above, we see that the above computations can be done in time polynomial in $\sum_{h=1}^m |x_h|$. To get the value of $Q_\Pi(\langle x_1, k_1 \rangle, \cdots, \langle x_m, k_m \rangle)$, we may compare

each $c(x_i, s_i^{(0)})$ with k_i. Since c is polynomial-time computable, all the comparisons can be done in polynomial-time. This gives us a $\leq_{1\text{-}T}^{\text{uniform-PF}}$-reduction of Q_Π to OPTSOL_Π. ∎

We next show that several well-known $NPCOP$'s are linearly paddable. For the graph-theoretic problems below, we suppose that each graph is encoded by its adjacency matrix, an n by n (0, 1)-matrix whose (i, j)-component is 1 if and only if the i-th vertex is connected to the j-th vertex in the graph, where n is the number of vertices in the graph and all vertices are assumed to be indexed by 1 through n.

Theorem 3.2 The following $NPCOP$'s are linearly paddable:

- MAXIMUM TWO SATISFIABILITY (MAX2SAT for short):
  Instance: A CNF Boolean formula ϕ such that each clause of ϕ contains at most two literals.
  Output: A truth assignment to the variables under which the maximum number of clauses become true.
- MAXIMUM CLIQUE (MAXCLIQUE for short):
  Instance: An undirected graph G.
  Output: A maximum clique of G.
- MINIMUM COLORING (MINCOLOR for short):
  Instance: An undirected graph $G = (V, E)$.
  Output: A partition $\langle U_1, U_2, \cdots, U_k \rangle$ of V such that k is the chromatic number of G and no two vertices u, v of G belong to the same U_i whenever $\{u, v\} \in E$. (Below, we simply call such a partition of V a *k-coloring* of G.)
- LONGEST PATH (LONGPATH for short):
  Instance: An undirected graph G.
  Output: A longest simple path in G.
- 0-1 INTEGER PROGRAMMING (01IP for short):
  Instance: A 3-tuple $\langle A, B, C \rangle$ of a $(0,1)$-matrix and two $(0,1)$-vectors.
  Output: A $(0,1)$-vector X maximizing $C^T X$ subject to $AX \leq B$.
- 0-1 TRAVELING SALESPERSON (01TSP for short):
  Instance: An undirected complete graph G with weights 0 or 1 on the edges.
  Output: A shortest traveling salesperson tour in G.

All decision problems associated with the $NPCOP$'s in Theorem 3.2 are well known to be NP-complete [1, 2, 3, 5, 7] (see [2] for a comprehensive reference). Thus, we have the following corollary.

Corollary 3.3 For any $NPCOP$ Π in Theorem 4.2, OPTSOL_Π is $\leq_{1\text{-}T}^{\text{uniform-PF}}$-hard for PF_{tt}^{NP}.

4 Conclusion

Some interesting questions still remain open. A natural question closely related to this work is whether $\mathrm{PF}_{\mathrm{tt}}^{\mathrm{NP}}$-hardness of $NPCOP$'s as in this paper implies the linear paddability of the $NPCOP$'s. A typical $NPCOP$ for this question is LONGEST CYCLE, the problem of finding a longest cycle of a given undirected graph. It is not so hard to show that LONGEST CYCLE is $\mathrm{PF}_{\mathrm{tt}}^{\mathrm{NP}}$-hard in the sense of this paper, but it seems slightly hard to show that the problem is linearly paddable. A relaxed version of linear paddability can be considered. For instance, we can consider *polynomial paddability* which is defined by removing the condition (a) in the definition of linear paddability, In fact, it is not so hard to show that LONGEST CYCLE is polynomially paddable. However, we have not been able to show that the polynomial paddability of $NPCOP$'s implies $\mathrm{PF}_{\mathrm{tt}}^{\mathrm{NP}}$-hardness of the $NPCOP$'s, as in Theorem 4.1. As mentioned by Krentel [6], there are some $NPCOP$'s which have not been classified yet. Some typical $NPCOP$'s are BIN PACKING and EDGE-COLORING [4].

References

[1] S. Even, *Graph Algorithms*, Computer Science Press, 1979.

[2] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.

[3] M. Garey, D. Johnson, and L. Stockmeyer, *Some Simplified NP-Complete Graph Problems*, Theoret. Comput. Sci., 1 (1976), pp. 237-267.

[4] I. Holyer, *The NP-Completeness of Edge-Coloring*, SIAM J. Comput., 10 (1981), pp. 718-720.

[5] R. Karp, *Reducibility among combinatorial problems*, in R. E. Miller and J. W. Thetcher (eds.), Complexity of Computer Computations (1972), Plenum Press, New York, pp. 85-113.

[6] M. W. Krentel, *The Complexity of Optimization Problems*, J. Comput. System Sci., 36 (1988), pp. 490-509.

[7] C. Papadimitriou and K. Steiglize, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, N.J., 1982.

[8] L. Valiant and V. Vazirani, *NP Is as Easy as Detecting Unique Solutions*, Theoret. Comput. Sci. 47 (1986), pp. 85-93.

[9] O. Watanabe and S. Toda, *Structural Analysis on the Complexity of Inverting Functions*, SIGAL International Symposium on Algorithms, accepted for presentation, 1990.