JOURNAL OF COMPUTER AND SYSTEM SCIENCES **54**, 465–474 (1997)
ARTICLE NO. SS971490

On Fixed-Parameter Tractability and Approximability of NP Optimization Problems[1]

Liming Cai[2]

School of Electrical Engineering and Computer Science, Ohio University, Athens, Ohio 45701

and

Jianer Chen[3]

Department of Computer Science, Texas A&M University, College Station, Texas 77843-3112

Received February 24, 1994; revised June 17, 1996

Fixed-parameter tractability of NP optimization problems is studied by relating it to approximability of the problems. It is shown that an NP optimization problem is fixed-parameter tractable if it admits a fully polynomial-time approximation scheme, or if it belongs to the class MAX SNP or to the class MIN $F^+\Pi_1$. This provides strong evidence that no $W[1]$-hard NP optimization problems belong to these optimization classes and includes a very large class of approximable optimization problems into the class of fixed-parameter tractable problems. Evidence is also demonstrated to support the current working hypothesis in the theory of fixed-parameter tractability.

1. INTRODUCTION

Recently, a framework of fixed-parameter tractability has been introduced by Downey and Fellows [11] to study the intractable behavior of computational problems whose input contains a significant numerical parameter. The complexity of such *parameterized problems* is specified in the value of the parameter, as well as the length of the input. Although, this framework has originally been addressed toward general parameterized problems, it is of particular interest to computational optimization problems because usually a decision problem can be formulated from an optimization problem by the parameterization of certain quantitative description (usually the solution size) in the optimization problem [14].

Many parameterized problems studied in [1, 2, 5, 11, 12] are parameterized optimization problems. It has been observed that quite a few optimization problems, although NP-hard in the general sense, have the time complexity that varies with respect to the value of the parameter. For example, consider the problem Vertex Cover (given a graph G and a parameter k, decide whether G has a vertex cover of size k) and the problem Dominating Set (given a graph G and a parameter k, decide whether G has a dominating set of size k). When the value of the parameter k is fixed, the problem Vertex Cover can be solved in time $O(n^c)$, where c is a constant independent of the parameter k; while the problem Dominating Set has the contrasting situation that the best known algorithms are of the time complexity $O(n^{k+1})$. According to Downey and Fellows [11], a parameterized problem is *fixed-parameter tractable* if it can be solved in time $O(n^c)$, where c is a constant independent of the parameter. Many optimization problems have been shown to be fixed-parameter tractable, while many other optimization problems, such as Dominating Set and Independent Set, are not know to be fixed-parameter tractable. Downey and Fellows [12] have introduced the class FPT of fixed-parameter tractable problems and a hierarchy of fixed-parameter intractable problems (the W-hierarchy).

In the present paper, we study fixed-parameter tractability of NP optimization problems by relating it to approximability of the problems. Our main concern here is to investigate what types of approximability of NP optimization problems imply their fixed-parameter tractability. This study serves for two purposes. First, by identifying the type of approximability that implies fixed-parameter tractability, we can include automatically a *class* of optimization problems into the class FPT, instead of developing (maybe fairly involved) fixed-parameter tractable

[1] A preliminary version of this work was reported in "Proc. 2nd Israel Symposium on Theory and Computing Systems (ISTCS'93), 1993," pp. 118–126.

[2] Supported in part by Engineering Excellence Award from Texas A&M University. E-mail: cai@cs.ohiou.edu.

[3] Supported in part by the National Science Foundation under Grant CCR-9110824 and an HTP 863 grant of P. R. China. E-mail: chen@cs.tamu.edu.

0022-0000/97 $25.00

algorithm for each individual problem in the class. Second, under the current working hypothesis in the theory of fixed-parameter tractability [2], we can immediately exclude from the class of optimization problems with the identified approximability those problems that are hard for a variety of levels of the W-hierarchy. Thus, this approach may provide a new and potentially powerful tool in the study of nonapproximability of NP optimization problems.

Our first result is that the class of optimization problems that have fully polynomial-time approximation schemes must be fixed-parameter tractable. This result immediately includes a number of knapsack-like problems and scheduling problems into the class FPT. This result also gives a strong evidence that no optimization problem that is hard for the first level of the W-hierarchy has a fully polynomial-time approximation scheme, thus complementing a well-known result by Garey and Johnson [14] that strongly NP-hard optimization problems have no fully polynomial-time approximation scheme unless $P = NP$. In particular, our result gives a strong evidence that the problem V-C Dimension has no fully polynomial-time approximation scheme.

We then show that all maximization problems in the class MAX SNP introduced by Papadimitriou and Yannakakis [18] and all minimization problems in the class $\text{MIN F}^+\Pi_1$ introduced by Kolaitis and Thakur [16] are fixed-parameter tractable. This includes a very large class of constant-ratio approximable NP optimization problems into the class FPT. In fact, there are very few known NP optimization problems that are constant-ratio approximable but belong to neither MAX SNP nor $\text{MIN F}^+\Pi_1$. This result shows a strong evidence that optimization problems in the W-hierarchy that are hard for the first level of the W-hierarchy should not be constant-ratio approximable in polynomial time.

Our study is based on the current working hypothesis in the theory of fixed-parameter tractability, which claims that no parameterized problem hard for the first level of the W-hierarchy is fixed-parameter tractable. Therefore, it will be nice to prove that the hypothesis really holds. However, attempting a direct proof for the hypothesis may be a bit too ambitious because the hypothesis implies $P \neq NP$ [12]. Thus, it may be more feasible to provide strong evidence supporting the hypothesis instead of a direct proof. In the second part of the current paper, we provide such an evidence by giving a sufficient and necessary condition in terms of classical complexity theory for each level of the W-hierarchy to collapse to the class FPT. Essentially, our characterization says that a level of the W-hierarchy does not collapse to the class FPT unless a deterministic polynomial-time computation can "guess" a string of length $\omega(\log n)$. This strengthens previous results in the study of structural properties of the W-hierarchy, in which only either sufficient or necessary (but not both) conditions were given for the collapsing of the W-hierarchy.

2. PRELIMINARIES

An *NP optimization problem* Q is either a *minimization problem* or a *maximization problem* and is given as a 4-tuple $(I_Q, S_Q, f_Q, \text{opt}_Q)$, where

- I_Q is the set of *input instances*. It is recognizable in polynomial time;
- $S_Q(x)$ is the set of *feasible solutions* for the input $x \in I_Q$ such that there is a polynomial p and a polynomial-time computable predicate π (p and π only depend on Q) such that for all $x \in I_Q$, $S_Q(x)$ can be expressed as $S_Q(x) = \{y : |y| \leqslant p(|x|) \wedge \pi(x, y)\}$;
- $f_Q(x, y) \in N$ is the *objective function*[4] for each $x \in I_Q$ and $y \in S_Q(x)$. The function f_Q is computable in polynomial time;
- $\text{opt}_Q \in \{\max, \min\}$.

An *optimal solution* for an input instance $x \in I_Q$ is a feasible solution $y \in S_Q(x)$ such that $f_Q(x, y) = \text{opt}_Q\{f_Q(x, z) \mid z \in S_Q(x)\}$. To simplify the expressions, the value $\text{opt}_Q\{f_Q(x, z) \mid z \in S_Q(x)\}$ is denoted $\text{opt}_Q(x)$.

An algorithm A is an *approximation algorithm* for Q if, given any input instance x in I_Q, A finds a feasible solution $y_A(x)$ in $S_Q(x)$. A maximization problem (resp. minimization problem) Q is *polynomial-time approximable to a ratio* $r(n)$ if there is a polynomial-time approximation algorithm A for Q such that the *relative error*

$$\Re_A(x) = \frac{\text{opt}_Q(x) - f_Q(x, y_A(x))}{f_Q(x, y_A(x))}$$

$$\left(\text{resp. } \Re_A(x) = \frac{f_Q(x, y_A(x)) - \text{opt}_Q(x)}{\text{opt}_Q(x)}\right)$$

is bounded by $r(|x|) - 1$ for all input instances $x \in I_Q$. An optimization problem Q has a *polynomial-time approximation scheme* if there is an algorithm A that takes a pair $x \in I_Q$ and $\varepsilon > 0$ as input and outputs a feasible solution in $S_Q(x)$ such that the relative error is bounded by ε for all input instances $x \in I_Q$ and the running time of A is bounded by a polynomial of $|x|$ for each fixed ε. We further say that the optimization problem Q has a *fully polynomial-time approximation scheme* if the running time of the algorithm A is bounded by a polynomial of $|x|$ and $1/\varepsilon$.

DEFINITION [11]. A *parameterized problem* Q is a subset of $\Omega^* \times N$, where Ω is a fixed alphabet. Therefore, each

[4] N denotes the set of all natural numbers.

instance of the parameterized problem Q is a pair $\langle x, k \rangle$, where the second component k is called *the parameter*.

The complexity of a parameterized problem can be specified in terms of the two components of its instances.

DEFINITION [11]. A parameterized problem Q is (*strongly*) *fixed-parameter tractable* if there is an algorithm to decide whether $\langle x, k \rangle$ is a member of Q in time $f(k)\,|x|^c$, where $f(k)$ is a recursive function and c is a constant independent of the parameter k. Let *FPT* denote the class of fixed-parameter tractable problems.

A reduction has been introduced [11] that preserves the fixed-parameter tractability of parameterized problems.

DEFINITION [11]. Let Q and Q' be two parameterized problems. Q is *uniformly reducible* to Q' if there is an algorithm M that transforms $\langle x, k \rangle$ into $\langle x', g(k) \rangle$ in time $f(k)\,|x|^c$, where f and g are recursive functions and c is a fixed constant, such that $\langle x, k \rangle \in Q$ if and only if $\langle x', g(k) \rangle \in Q'$.

We say that a circuit α is a Π_h-circuit if α is of unbounded fan-in and of depth at most h with an $\wedge$-gate at the output. A circuit β is a Π_h^c-circuit if β is a Π_{h+1}-circuit in which gates at input level have fan-in at most c. The *weight* of a Boolean string is the number of 1's in the string. Let h and c be two integers, we define a parameterized problem as follows:

$$\text{WCS}(h, c) = \{\langle \alpha, k \rangle \mid \text{The } \Pi_h^c\text{-circuit } \alpha \text{ accepts an input of weight } k\}.$$

DEFINITION [11]. For each integer $h \geqslant 1$, the class $W[h]$ consists of all parameterized problems that are uniformly reducible to the parameterized problem WCS(h, c) for some constant c.

The above leads to an interesting hierarchy (called the *W-hierarchy* [11])

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[h] \subseteq \cdots$$

for which a wide variety of natural problems are now known to be complete or hard for various levels (under the uniform reduction) [1, 11, 12]. It is easy to see that if a problem Q hard for the class $W[h]$ is fixed-parameter tractable, then all problems in the class $W[h]$ are fixed-parameter tractable.

Current Working Hypothesis [2]. The class FPT is a proper subclass of the class $W[1]$. Thus, no parameterized problem that is hard for $W[1]$ is fixed-parameter tractable.

3. FPT AND APPROXIMABILITY

In this section, we demonstrate interesting relationship between fixed-parameter tractability and approximability of NP optimization problems. In particular, we will show the fixed-parameter tractability for some well-known NP optimization classes and investigate the approximability of NP optimization problems based on their fixed-parameter tractability. We will see that in many cases, fixed-parameter intractability of an NP optimization problem implies non-approximability of the problem. Thus, the study of fixed-parameter intractability may provide a new and potentially powerful approach to proving nonapproximability of NP optimization problems.

There are many ways to parameterize an NP optimization problem. We adopt the most natural and reasonable parameterization scheme that associates the parameter with the value of the objective function of the problem.

DEFINITION. Let $Q = (I_Q, S_Q, f_Q, \text{opt}_Q)$ be an NP optimization problem, the *corresponding parameterized problem* $Q_=$ is defined by

$$Q_= = \{\langle x, k \rangle \mid x \in I_Q \text{ and } k = \text{opt}_Q(x)\}.$$

The other two parameterized problems $Q_\geqslant$ and $Q_\leqslant$ can be defined similarly based on the NP optimization problem Q by replacing the relation $k = \text{opt}_Q(x)$ in the definition by $k \geqslant \text{opt}_Q(x)$ and $k \leqslant \text{opt}_Q(x)$, respectively.

We say that the optimization problem Q is *fixed-parameter tractable* if its corresponding parameterized problem $Q_=$ is fixed-parameter tractable.

The complexity of the parameterized problems $Q_=$, $Q_\leqslant$, and $Q_\geqslant$ have been studied in the literature. For example, Leggett and Moore [17] showed that for many NP optimization problems Q, the problems $Q_\leqslant$ and $Q_\geqslant$ are in NP $\cup$ coNP, while the problem $Q_=$ is not in NP $\cup$ coNP unless NP = coNP. On the other hand, from the viewpoint of fixed-parameter tractability, the problems $Q_=$, $Q_\leqslant$, and $Q_\geqslant$ have the same complexity, as shown in the following lemma.

LEMMA 3.1. *Let Q be an arbitrary NP optimization problem. Then the three parameterized problems $Q_=$, $Q_\leqslant$, $Q_\geqslant$ are either all fixed-parameter tractable, or all fixed-parameter intractable.*

Proof. Suppose that the parameterized problem $Q_\leqslant$ is fixed-parameter tractable. Thus, there is an algorithm $A_\leqslant$ that solves the problem $Q_\leqslant$ in time $O(f(k)\,n^c)$, where f is a recursive function, and c is a constant independent of the parameter k. To show that the parameterized problem $Q_=$ is fixed-parameter tractable, note that the condition $k = \text{opt}_Q(x)$ is equivalent to the condition $(k \leqslant \text{opt}_Q(x))\&((k+1) > \text{opt}_Q(x))$. Therefore, given an instance $\langle x, k \rangle$ for

the parameterized problem $Q_=$, we can run the algorithm $A_\leqslant$ on the input $\langle x, k\rangle$ and on the input $\langle x, k+1\rangle$ to decide whether $k = \mathrm{opt}_Q(x)$. The total running time is obviously bounded by $O((f(k)+f(k+1))\,n^c)$. Thus, the problem $Q_=$ is fixed-parameter tractable.

Conversely, suppose that the problem $Q_=$ is fixed-parameter tractable. Then there is an algorithm $A_=$ that solves the problem $Q_=$ in time $O(f'(k)\,n^{c'})$, where f' is a recursive function and c' is a constant independent of the parameter k. Now given an instance $\langle x, k\rangle$ of the parameterized problem $Q_\leqslant$, we run the algorithm $A_=$ on k inputs $\langle x, 0\rangle$, $\langle x, 1\rangle$, ..., $\langle x, k-1\rangle$. It is easy to see that $\langle x, k\rangle$ is a no-instance of the parameterized problem $Q_\leqslant$ if and only if the algorithm $A_=$ concludes yes for at least one of those k inputs. Note that the running time of the above process is bounded by $O(kf'(k)\,n^{c'})$. Thus, the problem $Q_\leqslant$ is fixed-parameter tractable.

Therefore, the parameterized problem $Q_=$ is fixed-parameter tractable if and only if the parameterized problem $Q_\leqslant$ is fixed-parameter tractable. In a similar way, we can show that the parameterized problem $Q_\geqslant$ is fixed-parameter tractable if and only if the parameterized problem $Q_=$ is fixed-parameter tractable. ∎

Fixed-parameter tractability provides a necessary condition for an NP optimization problem to have a very good polynomial-time approximation algorithm.

THEOREM 3.2. *If an NP optimization problem has a fully polynomial-time approximation scheme, then it is fixed-parameter tractable.*

Proof. Suppose that an NP optimization problem $Q = (I_Q, S_Q, f_Q, \mathrm{opt}_Q)$ has a fully polynomial-time approximation scheme A. Then, the algorithm A takes as input both an instance $x \in I_Q$ of Q and an accuracy requirement $\varepsilon > 0$, and then outputs a feasible solution in time $O(p(1/\varepsilon, |x|))$ such that the relative error $\Re_A(x)$ is bounded by ε, where p is a two-variable polynomial.

First we assume that Q is a maximization problem. By Lemma 3.1, we only need to show that the problem $Q_\leqslant$ is fixed-parameter tractable. Given an instance $\langle x, k\rangle$ for the parameterized problem $Q_\leqslant$, we run the algorithm A on input x and $1/2k$. In time $O(p(2k, |x|))$, the algorithm A produces a feasible solution $y \in S_Q(x)$ such that

$$\Re_A(x) = \frac{\mathrm{opt}_Q(x) - f_Q(x, y)}{f_Q(x, y)} \leqslant \frac{1}{2k} < \frac{1}{k}.$$

If $k \leqslant f_Q(x, y)$, then certainly $k \leqslant \mathrm{opt}_Q(x)$ since Q is a maximization problem. On the other hand, if $k > f_Q(x, y)$, then $k - 1 \geqslant f_Q(x, y)$. Combining this with the inequality $(\mathrm{opt}_Q(x) - f_Q(x, y))/f_Q(x, t) < 1/k$, we get immediately $k > \mathrm{opt}_Q(x)$. Therefore, $k \leqslant \mathrm{opt}_Q(x)$ if and only if $k \leqslant f_Q(x, y)$. Since the feasible solution y can be constructed by the algorithm A in time $O(p(2k, |x|))$, we conclude that the NP optimization problem Q is fixed-parameter tractable.

The case when Q is a minimization problem can be proved similarly by showing that the problem $Q_\geqslant$ is fixed-parameter tractable. ∎

Theorem 3.2 immediately includes a number of knapsack-like problems and scheduling problems [15, 20] into the class FPT. Under our current working hypothesis that FPT is a proper subclass of $W[1]$, Theorem 3.2 exhibits a strong evidence that many NP optimization problems do *not* have fully polynomial-time approximation scheme, as stated in the following corollary.

COROLLARY 3.3. *The NP optimization problems that are $W[1]$-hard under the uniform reduction have no fully polynomial-time approximation scheme unless $W[1] = FPT$.*

Corollary 3.3 complements a classical result of Garey and Johnson (Theorem 6.8 in [14, p. 140]) that a strongly *NP*-hard optimization problem Q with $\mathrm{opt}_Q(x) < p(|x|, \mathrm{num}(x))$ for some two-variable polynomial p has no fully polynomial-time approximation scheme unless P = NP, where $\mathrm{num}(x)$ is the largest integer appearing in the input instance x. The consequence $W[1] = \mathrm{FPT}$ in Corollary 3.3 is weaker than P = NP but the assumption in Corollary 3.3 requires neither strong NP-hardness nor the condition $\mathrm{opt}_Q(x) < p(|x|, \mathrm{num}(x))$.

We show an application of Theorem 3.2. The V-C Dimension problem has been studied recently in the literature. It has been proved that the V-C dimension of a family $\mathscr{C}$ of subsets of some finite set is a reasonable precise estimate of the complexity of learning $\mathscr{C}$ [4]. Papadimitriou and Yannakakis [19] have shown strong evidence that the problem V-C Dimension is in NP but neither in P or NP-complete. Therefore, the problem V-C Dimension does not seem to be strongly NP-hard, and Theorem 6.8 of Garey and Johnson [14] is not applicable. On the other hand, Downey and Fellows [13] have recently shown that the corresponding parameterized V-C Dimension problem is $W[1]$-hard under the uniform reduction. Thus, it is unlikely to be fixed-parameter tractable. By Theorem 3.2, the problem V-C Dimension is unlikely to have fully polynomial-time approximation scheme even though it seems easier than NP-complete problems.

The condition of having a fully polynomial-time approximation scheme in Theorem 3.2 is not necessary. In the following, we show that for a very large class of important optimization problems that are polynomial-time approximable to a constant ratio, the corresponding parameterized problems are fixed-parameter tractable.

The first class of optimization problems we will study is the class MAX SNP introduced by Papadimitriou and Yannakakis [18]. It has been shown that several natural and well-studied NP optimization problems belong to this

class and that all NP optimization problems in this class can be approximated in polynomial time to a constant ratio [18]. Formally, a maximization problem $Q = (I_Q, S_Q, f_Q, \text{opt}_Q)$ is in the class MAX SNP if its optimum $\text{opt}_Q(X)$, $X \in I_Q$, can be expressed as

$$\text{opt}_Q(X) = \max_S |\{v: \psi(v, X, S)\}|,$$

where the input instance $X = (U, P^1, ..., P^h)$ is described by a finite structure over the finite universe U and P^i is a predicate of arity r_i for some integer $r_i \geq 1$, S is also a finite structure over the universe U, v is a vector of fixed arity of elements in U, and ψ is a quantifier-free formula.

Arora *et al.* [3] have recently made a breakthrough and shown that MAX SNP-complete optimization problems have no polynomial-time approximation scheme unless P = NP. Therefore, Theorem 3.2 is not applicable. We show in the following theorem the fixed-parameter tractability of problems in MAX SNP.

THEOREM 3.4. *All maximization problems in the class MAX SNP are fixed-parameter tractable.*

Proof. Let c be a fixed constant. Consider the following MAX c-SAT problem:

> MAX c-SAT. Given a set of Boolean formulas $\phi_1, \phi_2, ..., \phi_m$, each has at most c variables, find a truth assignment that satisfies the maximum number of the formulas.

We first show that the corresponding parameterized problem $(\text{MAX } c\text{-SAT})_=$ is fixed-parameter tractable for every constant c. By Papadimitriou and Yannakakis [18], for any input instance $x = \{\phi_1, ..., \phi_m\}$ of the problem MAX c-SAT, the optimal value of x is at least $m/2^c$. Moreover, since each formula in x has at most c variables, the total number of variables appearing in x is bounded by cm. Therefore, the following algorithm demonstrates the fixed-parameter tractability of the parameterized problem $(\text{MAX } c\text{-SAT})_=$: given a pair $\langle x, k \rangle$, where $x = \{\phi_1, ..., \phi_m\}$ is an instance of the problem MAX c-SAT and k is an integer, we first compare the values k and $m/2^c$. If $k < m/2^c$, then $\langle x, k \rangle$ is obviously a no-instance for the parameterized problem $(\text{MAX } c\text{-SAT})_=$. If $k \geq m/2^c$, then we check all assignments to the variables in x to verify that whether the maximum number of formulas in x that can be satisfied by a single assignment is k. Since the total number of variables in x is bounded by cm, there are at most 2^{cm} different assignments to the variables in x. Therefore, the above algorithm runs in time $O(2^{cm}m)$, which is bounded by $O(2^{O(k)}m)$ since $m = O(k)$. This shows that the parameterized problem $(\text{MAX } c\text{-SAT})_=$ is fixed-parameter tractable.

According to Papadimitriou and Yannakakis [18], every MAX SNP problem Q can be reduced to the problem MAX c-SAT for some constant c in the following sense: there is a polynomial-time algorithm that, given an input instance $x \in I_Q$ of Q, constructs an input instance x' of MAX c-SAT such that the optimal value for the instance x of the problem Q is equal to the optimal value for the instance x' of the problem MAX c-SAT. Therefore, the membership of the input instance $\langle x, k \rangle$ for the parameterized problem $Q_=$ is identical to the membership of the input instance $\langle x', k \rangle$ for the parameterized problem $(\text{MAX } c\text{-SAT})_=$. Since x' can be constructed from x in polynomial time and the parameterized problem $(\text{MAX } c\text{-SAT})_=$ is fixed-parameter tractable, we conclude that the optimization problem Q in MAX SNP is also fixed-parameter tractable. Since Q is an arbitrary problem in the class MAX SNP, the theorem follows. ∎

We can also show that an important class of minimization problems introduced by Kolaitis and Thakur [16] are fixed-parameter tractable.

DEFINITION. Define MIN $\text{F}^+\Pi_1(h)$, $h \geq 2$, to be the class of all minimization problems Q whose optimum can be expressed as

$$\text{opt}_Q(X) = \min_S \{|S|: \forall v \psi(v, S, X)\},$$

where $S = (U, P_0)$ and P_0 is a predicate of arity 1 over the finite universe U, $|S|$ denotes the weight of the predicate P_0, i.e., the number of elements of the universe U on which the predicate P_0 has truth value, and $\psi(v, S, X)$ is a quantifier-free CNF formula in which all occurrences of P_0 are positive, and P_0 occurs at most h times in each clause. Finally, let MIN $\text{F}^+\Pi_1$ be the union $\bigcup_{h \geq 2}$ MIN $\text{F}^+\Pi_1(h)$.

The class MIN $\text{F}^+\Pi_1$ contains a number of well-studied minimization problems, including Vertex Cover and a large number of *vertex-deletion* and *edge-deletion* graph problems [16]. It is known that all minimization problems in MIN $\text{F}^+\Pi_1$ are polynomial-time approximable to a constant ratio [16]. Theorem 3.2 is not applicable to the class MIN $\text{F}^+\Pi_1$ since the problem Vertex Cover is MAX SNP-hard [18], so it does not have a fully polynomial-time approximation scheme unless P = NP [3]. In the following, we derive a direct proof for the fixed-parameter tractability of MIN $\text{F}^+\Pi_1$.

THEOREM 3.5. *All minimization problems in the class MIN $\text{F}^+\Pi_1$ are fixed-parameter tractable.*

Proof. Let Q be a minimization problem in the class MIN $\text{F}^+\Pi_1(h)$, where $h \geq 2$ is a fixed integer. Then the optimum $\text{opt}_Q(X)$ for any input instance X of Q can be expressed as

$$\text{opt}_Q(X) = \min_S \{|S|: \forall v \psi(v, S, X)\},$$

Algorithm WEIGHTEDCNF(F, k)

INPUT: A CNF formula $F = \{C_1, \cdots, C_s\}$ and an integer k, where each C_i is a clause in which there are at most h unknowns, all of the form $P_0(u)$, $u \in U$, and appearing positively.

QUESTION: Is there a finite structure $S = (U, P_0)$, $k > |S|$, and S satisfies F?

1. IF $k \leq 0$, then REJECT.
2. IF $F = \phi$, then ACCEPT.
3. Suppose that $C_1 = \{P_0(u_1), P_0(u_2), \cdots, P_0(u_c)\}$, where $u_j \in U$ and $c \leq h$;
4. FOR $j = 1$ to c Do

   Make $P_0(u_j) = 1$;

   Let F' be the CNF formula obtained from the formula F by deleting all clauses that contain the occurrence $P_0(u_j)$;

   Recursively call WEIGHTEDCNF$(F', k-1)$.

FIG. 1. The algorithm WEIGHTEDCNF(F, k).

where $S = (U, P_0)$, and P_0 is a predicate of arity 1 over the finite universe U, $v = (v_1, ..., v_d)$ is a vector over U of arity d for some fixed d, $\psi(v, S, X)$ is a quantifier-free CNF formula in which all occurrences of P_0 are positive, and P_0 occurs at most h times in each clause.

We show below that the parameterized problem $Q_{\leqslant}$ is fixed-parameter tractable.

Fix an input instance X of Q. Consider the formula

$$F(X, S) = \bigwedge_{v \in U^d} \psi(v, S, X).$$

Note that the formula $F(X, S)$ is a quantifier-free CNF formula in which the only unknowns are $P_0(u)$, where $u \in U$. Moreover, according to the definition of ψ, each clause of $F(X, S)$ contains at most h unknowns $P_0(u)$, $u \in U$, and all of them occur positively. Since the number of elements of the universe U is bounded by the length $|X|$ and d is a fixed integer, the formula $F(X, S)$ can be constructed from the input instance X in polynomial time.

Now given an instance $\langle X, k \rangle$ of the parameterized problem $Q_{\leqslant}$, we first construct the formula $F(X, S) = \{C_1, C_2, ..., C_s\}$, then execute the algorithm WEIGHTEDCNF(F, k) described in Fig. 1. It is easy to verify the correctness of the algorithm WEIGHTEDCNF(F, k). If the algorithm WEIGHTEDCNF(F, k) accepts, then there is a finite structure $S_0 = (U, P_0)$, $|S_0| < k$, and $F(X, S_0) = \forall v \psi(v, S_0, X) = 1$. Therefore,

$$\text{opt}_Q(X) = \min_S \{|S| : \forall v \psi(v, S, X)\} \leqslant |S_0| < k.$$

Thus, $\langle X, k \rangle$ is a no-instance of the parameterized problem $Q_{\leqslant}$. Otherwise we should have

$$\text{opt}_Q(X) = \min_S \{|S| : \forall v \psi(v, S, X)\}$$

$$= \min_S \{|S| : F(X, S)\} \geqslant k.$$

Thus, $\langle X, k \rangle$ is a yes-instance of the parameterized problem $Q_{\leqslant}$. Therefore, the problem $Q_{\leqslant}$ can be solved through the algorithm WEIGHTEDCNF.

We claim that the running time of the algorithm WEIGHTEDCNF(F, k) is bounded by $O(h^k |F|)$. This is certainly true when $k \leqslant 1$. If $k \geqslant 2$, then the loop body of Step 4 is executed at most h times, each time involves a recursive call WEIGHTEDCNF$(F', k-1)$, where $|F'| \leqslant |F|$. By the inductive hypothesis, the running time of each call of WEIGHTEDCNF$(F', k-1)$ is bounded by $O(h^{k-1} |F|)$. The conclusion follows.

Note that the length of the formula F is bounded by a polynomial of the length of the input X which is independent of the parameter k, and h is a constant. Consequently the parameterized problem $Q_{\leqslant}$ is fixed-parameter tractable. ∎

Under our current working hypothesis, no $W[1]$-hard parameterized problem is fixed-parameter tractable. Therefore, Theorem 3.2, Theorem 3.4, and Theorem 3.5 immediately exclude any $W[1]$-hard optimization problems from the classes MAX SNP and MIN $F^+\Pi_1$, and from the class of optimization problems with fully polynomial-time

approximation schemes. Note that there have been several dozens of NP optimization problems in the W-hierarchy that are known to be $W[1]$-hard [12], and none of them is known to be polynomial-time approximable with a constant ratio. Therefore, fixed-parameter intractability seems to hint nonapproximability for NP optimization problems. The study of fixed-parameter tractability may provide a new and potentially very powerful approach to the study of nonapproximability of NP optimization problems.

4. FPT AND THE W-HIERARCHY

We have seen from the previous section that for many NP optimization problems, nonapproximability of the problems can be derived from fixed-parameter intractability of the problems, provided our current working hypothesis holds. However, attempting a direct proof for the hypothesis seems a bit ambitious because it would imply $P \neq NP$ [11]. In this section, we show that the collapsing of the W-hierarchy is equivalent to an unlikely fact in classical complexity theory. This connection establishes a strong evidence supporting the current working hypothesis.

Our discussion will be based on the following GC model (for "Guess-then-Check") introduced by Cai and Chen [7].

DEFINITION. Let $s(n)$ be a function and let $\mathscr{C}$ be a complexity class. A language L is in the class $GC(s(n), \mathscr{C})$ if there are a language $A \in \mathscr{C}$ and an integer $c > 0$ such that for all x, $x \in L$ if and only if $\exists y \in \{0, 1\}^*$, $|y| \leqslant c \cdot s(|x|)$, and $(x, y) \in A$.

Intuitively, the first component $s(n)$ in the $GC(s(n), \mathscr{C})$ model specifies the length of the guessed string y, which is the amount of nondeterminism allowed to make in the computation, while the second component $\mathscr{C}$ specifies the verifying power of the computation. A number of restricted forms of the GC model have been studied recently in the literature [6, 7, 10, 19].

We are particularly interested in the classes $GC(s(n)\, \Pi_h^B)$, where $s(n) = \omega(\log n)$, and Π_h^B is the class of languages accepted by $O(\log n)$ times alternating Turing machines that make at most h alternations and always begin with universal states (such alternating Turing machines will be called "Π_h^B-ATMs"). By our current knowledge, a deterministic polynomial-time computation can only "guess" a string of length at most $\Theta(\log n)$ by exhaustively enumerating all strings of length $\Theta(\log n)$. Moreover, it has now become well-known that the class Π_h^B is a proper subclass of the class P [21]. Therefore, the model $GC(s(n), \Pi_h^B)$ has a (presumely) stronger guessing ability but (provably) weaker computational power than deterministic polynomial-time computations. (For more detailed discussion of the model $GC(s(n), \Pi_h^B)$, the reader is referred to our recent papers [7, 8].)

THEOREM 4.1. *For any nondecreasing function $t(n)$ constructible in polynomial time and for any integer $h > 1$, the following language* $\textsc{Csat}(t, h)$ *is complete for the class $GC(t(n) \log(n), \Pi_h^B)$ under the polynomial-time reduction*:

> $\textsc{Csat}(t, h)$. *The set of all strings of the form $x = \alpha \# w$, where w is an integer not larger than $t(|x|)$, and α is a Π-circuit that accepts an input of weight w.*

Proof. The authors have shown in another paper (Theorem 4.4 and Theorem 4.7 in [7]) that for any nondecreasing function $t(n)$ constructible in polynomial time, the language $\textsc{Csat}(t, h)$ is hard for the class $GC(t(n) \log(n), \Pi_h^B)$ under the polynomial-time reduction for all $h > 1$. Therefore, to prove the theorem, we only have to show that the language $\textsc{Csat}(t, h)$ is in the class $GC(t(n) \log(n), \Pi_h^B)$. Consider the algorithm $\textsc{Csat}(t, h)$-SIMULATOR in Fig. 2. We shall show that the algorithm $\textsc{Csat}(t, h)$-SIMULATOR can be implemented by a Π_h^B-ATM such that for all $x = \alpha \# w$, $x \in \textsc{Csat}(t, h)$ if and only if there is a string $y \in \{0, 1\}^*$, $|y| \leqslant 2t(|x|) \log(|x|)$ such that the algorithm accepts (x, y).

The binary string y of length $2w \log(|x|)$ encodes a weight w input v_y of the circuit α as follows. The first $(w+1) \log(|x|)$ bits of y are interpreted as $w+1$ pairs (p_0, p_1), (p_1, p_2), ..., (p_{w-1}, p_w), (p_w, p_{w+1}), with $0 = p_0 < p_1 < p_2 < \cdots < p_w < p_{w+1} = m+1$, such that the jth bit of v_y is 1 if and only if $j = p_i$ for some i, $0 < i < w+1$. Here without loss of generality, we assume that $\log(|x|) \geqslant 2\,|m|$ so that a pair (p_i, p_{i+1}) of integers with $0 \leqslant p_i, p_{i+1} \leqslant m+1$ can be encoded into $\log(|x|)$ bits. The rest part of the string y will be ignored in the simulation.

It is not very difficult to verify that the step "Checking" of the algorithm can be implemented by a Π_2^B-ATM. Now we analyze the step "Simulation."

Since the output gate of α is an $\wedge$-gate, the first execution of the loop body in Step 2 is a universal branch, which is combined with the starting universal branching of the algorithm to form the first phase of the algorithm.

The loop execution of Step 2 simply simulates the computation of the circuit α. After h executions of the loop body in Step 2, the algorithm is in its hth phase, which is an existential phase if h is even, or a universal phase if h is odd. Therefore, the branching in Step 3 can be combined with the last phase in Step 2 to form the last phase of the algorithm. Therefore, the algorithm $\textsc{Csat}(t, h)$-SIMULATOR makes at most h alternations. Moreover, it is obvious that each computation path of the algorithm takes time $O(\log n)$. This concludes that the algorithm $\textsc{Csat}(t, h)$-SIMULATOR can be implemented by a Π_h^B-ATM.

Recall that v_y is the weight w Boolean string of length m such that the jth bit of v_y is 1 if and only if there is a pair (p_i, p_{i+1}) in y, $i > 0$ and $j = p_i$. We show that the algorithm

Algorithm CSAT(t, h)-SIMULATOR

INPUT: (x, y), where $x = \alpha\#w$ and α is a circuit with input $v_1 \cdots v_m$.

Universally branch into step Checking and step Simulation.

Checking {Syntax Checking}

Reject if any of the following conditions is not satisfied.

- $w \leq m$ and α is a Π_h-circuit;
- $|y| = 2w \log(|x|)$ and the first $(w+1)\log(|x|)$ bits of y encode $w+1$ pairs (p_0, p_1), (p_1, p_2), $\cdots$, (p_w, p_{w+1}), where $0 = p_0 < p_1 < p_2 < \cdots p_w < p_{w+1} = m+1$ and each pair (p_i, p_{i+1}) consists of exactly $\log(|x|)$ bits.

Simulation {Circuit Simulation}

1. Let g be the output gate of α;
2. Repeat the following loop h times:
   if g is an $\wedge$-gate, then universally guess an input g' of g. Let $g = g'$;
   if g is an $\vee$-gate, then existentially guess an input g' of g. Let $g = g'$;
3. { At this point, g is an input node of the circuit α. This step is different for the case h is even and for the case h is odd. }
   [In case h is even]
   Existentially read a pair (p_i, p_{i+1}) from the string y;
   (a) If g is a positive literal v_j, then accept if and only if $p_i = j$;
   (b) If g is a negative literal $\bar{v}_j$, then accept if and only if $p_i < j < p_{i+1}$.
   [In case h is odd]
   Universally read a pair (p_i, p_{i+1}) from the string y;
   (a) If g is a positive literal v_j, then reject if and only if $p_i < j < p_{i+1}$;
   (b) If g is a negative literal $\bar{v}_j$, then reject if and only if $p_i = j$.

FIG. 2. The algorithm CSAT(t, h)-SIMULATOR.

CSAT(t, h)-SIMULATOR on input (x, y) correctly simulates the circuit α on input v_y, where $x = \alpha\#w$. First assume that h is an even number. Let g_0 be a gate at input level of α, which must be an $\vee$-gate. Therefore, the gate g_0 gets value 1 on the input v_y if and only if either an input of g_0 is a positive literal v_j and there is a pair (p_i, p_{i+1}) in y such that $j = p_i$, or an input of g_0 is a negative literal $\bar{v}_j$ and j is not any p_i contained in y; i.e., there is a pair (p_i, p_{i+1}) in y such that $p_i < j < p_{i+1}$. This process is correctly simulated by a last phase of the algorithm CSAT(t, h)-SIMULATOR, which guesses an input of g_0 in the last phase of Step 2 and a pair (p_i, p_{i+1}) in Step 3 and checks the above conditions. The case that $h > 1$ is odd can be verified similarly.

This concludes that the algorithm CSAT(t, h)-SIMULATOR is a Π_h^B-ATM, which accepts (x, y), where $x = \alpha\#w$, if and only if y is a string of length $2w \log(|x|)$ that encodes a weight w Boolean string v_y of length m as described above, and the circuit α accepts the input v_y.

For any $x = \alpha\#w$, if $x \in$ CSAT(t, h), then $w \leq t(|x|)$ and α accepts an input v of weight w. Let y_v be the binary string of length $2w \log(|x|)$ that encodes v as described above. Then the pair (x, y_v) is accepted by the algorithm CSAT(t, h)-SIMULATOR, and $|y_v| = 2w \log(|x|) \leq 2t(|x|) \log(|x|)$. On the other hand, if $x \notin$ CSAT$(t(n), h)$, then the pair (x, y) is not accepted by the algorithm CSAT(t, h)-SIMULATOR for any y of length at most $2t(|x|) \log(|x|)$ (note that by the algorithm, a string of length at most $2t(|x|) \log(|x|)$ cannot be used to encode an input of α that has weight larger than $t(|x|)$).

Thus, the language CSAT(t, h) is in the class $GC(t(n) \log(n), \Pi_h^B)$ and consequently, is complete for the class $GC(t(n) \log(n), \Pi_h^B)$ under the polynomial-time reduction. ∎

A single Boolean literal is called a *Π_0-normalized Boolean expression* as well as a *Σ_0-normalized Boolean expression.* Inductively, a *Π_h-normalized Boolean expression* is an "$\wedge$" of Σ_{h-1}-normalized Boolean expressions, and a *Σ_h-normalized Boolean expression* is an "$\vee$" of Π_{h-1}-normalized Boolean expressions.

THEOREM 4.2 [11]. *For $h > 1$, the following parameterized problem* ESAT(h) *is complete for the class $W[h]$ under the uniform reduction*:

> ESAT(h). *The set of all pairs $\langle \beta, k \rangle$, where β is a Π_h-normalized Boolean expression that has a satisfying assignment of weight k.*

Now we are ready for our main theorem of this section.

THEOREM 4.3. *For all $h > 1$, $W[h] = FPT$ if and only if there is an unbounded, nondecreasing function $t(n) \leqslant n$ computable in polynomial time such that the class $GC(t(n)\log(n), \Pi_h^B)$ is a subclass of the class P.*

Proof. Suppose that $GC(t(n)\log(n), \Pi_h^B)$ is a subclass of the class P, then by Theorem 4.1, the problem $\text{CSAT}(t, h)$ is in P. An algorithm $\text{ESAT}(h)$-SOLVER solving the parameterized problem $\text{ESAT}(h)$ can be described as follows: given an input $\langle \beta, k \rangle$, first compare the values k and $t(|\beta|)$. If $k > t(|\beta|)$, then the algorithm tries all possible assignments of weight k to the expression β to see if any of these assignments satisfies the expression β. In case $k \leqslant t(|\beta|)$, the algorithm first converts the expression β into a Π_h-circuit α_β; then it solves the problem $\text{CSAT}(t, h)$ on input $x = \alpha_\beta \# k$.

We analyze the above algorithm $\text{ESAT}(h)$-SOLVER. Define

$$\tau(k) = \min\{m \mid \text{for all } n \geqslant m,\ t(n) \geqslant k\}.$$

Since $t(n)$ is unbounded, nondecreasing, and computable in polynomial time, the function $\tau(k)$ is well-defined and recursive. In the case $k > t(|\beta|)$, the size $|\beta|$ of the Boolean expression β is bounded by $\tau(k)$. In particular, the number of variables in the Boolean expression β is bounded by $\tau(k)$. Thus, the number of assignments of β that have weight k is bounded by $(\tau(k))^k$. Therefore, in this case, the algorithm $\text{ESAT}(h)$-SOLVER takes time $O((\tau(k))^k\,|\beta|^2)$ to examine all possible assignments of weight k to the expression β, assuming we are using a straightforward quadratic-time algorithm to evaluate a Boolean expression given an assignment. On the other hand, if $k \leqslant t(|\beta|)$, then the algorithm $\text{ESAT}(h)$-SOLVER takes time $O(|\alpha_\beta|^c) = O(|\beta|)^c$ for some constant c independent of the parameter k because of our assumption that the problem $\text{CSAT}(t, h)$ is in P. In conclusion, the algorithm $\text{ESAT}(h)$-SOLVER runs in time $O((\tau(k))^k\,|\beta|^d)$, where $d \geqslant 2$ is a constant independent of the parameter k. That is, the problem $\text{ESAT}(h)$ is fixed-parameter tractable. By Theorem 4.2, we have $W[h] = FPT$.

Conversely, suppose that $W[h] = FPT$, where $h > 1$. By Theorem 4.2, the parameterized problem $\text{ESAT}(h)$ is fixed-parameter tractable. Suppose that $\text{ESAT}(h)$-SETTLER is an algorithm of running time $O(f(k)\,n^c)$ which solves the problem $\text{ESAT}(h)$, where f can be assumed to be an unbounded nondecreasing recursive function [9] and $c \geqslant 1$ is a constant. Let t be the inverse function of f defined by

$$t(m) = \max\{n \mid f(n) \leqslant m\}.$$

Without loss of generality, we can assume that $t(n) \leqslant n$, and that $t(n)$ is unbounded, nondecreasing, and computable in time polynomial in n (see [9] for a formal proof). Now an algorithm $\text{CSAT}(t, h)$-SETTLER solving the problem $\text{CSAT}(t, h)$ can be described as follows: on input $x = \alpha \# w$, where α is a Π_h-circuit and $w \leqslant t(|x|)$, first convert α into an equivalent Π_h-normalized Boolean expression β_α; then call the algorithm $\text{ESAT}(h)$-SETTLER on input $\langle \beta_\alpha, w \rangle$.

Since the value $t(|x|)$ can be computed in time bounded by a polynomial in $|x|$, the condition $w \leqslant t(|x|)$ can be checked in polynomial time. The Π_h-normalized Boolean expression β_α equivalent to the Π_h-circuit α has size $|\beta_\alpha|$ bounded by $|\alpha|^h$ because the depth of the circuit α is bounded by h. Moreover, it is easy to see that the Π_h-normalized Boolean expression β_α can be constructed from the Π_h-circuit α in time $O(|\beta_\alpha|^2)$. Now by our assumption, the running time of the algorithm $\text{ESAT}(h)$-SETTLER on input $\langle \beta_\alpha, w \rangle$ is bounded by $O(f(w)\,|\beta_\alpha|^c)$. Since $w \leqslant t(|x|)$ and the function f is nondecreasing, we have $f(w) \leqslant f(t(|x|)) \leqslant |x| = O(|\beta_\alpha|)$, where the second inequality comes from the fact that the function t is the inverse function of the function f. Therefore, the time complexity of the algorithm $\text{CSAT}(t, h)$-SETTLER is bounded by $O(|\beta_\alpha|^2) + O(|\beta_\alpha|^{c+1}) = O(|\alpha|^{h(c+1)}) = O(|x|^{h(c+1)})$.

This shows that the problem $\text{CSAT}(t, h)$ can be solved in polynomial time. By Theorem 4. 1, the language $\text{CSAT}(t, h)$ is complete for the class $GC(t(n)\log(n), \Pi_h^B)$ under the polynomial-time reduction. Therefore, the class $GC(t(n)\log(n), \Pi_h^B)$ is a subclass of the class P. ∎

A deterministic polynomial-time computation can enumerate all strings of length $c \log n$ for a fixed constant c. However, it is unknown whether a deterministic polynomial-time computation can "guess" a string of length larger than $\Theta(\log n)$. Note that for any unbounded nondecreasing function $t(n)$, the function $t'(n) = t(n)\log(n)$ is of order $\omega(\log n)$. Thus, the model $GC(t(n)\log n, \Pi_h^B)$ seems to have a stronger ability of guessing than a deterministic polynomial-time computation. Theorem 4.3 basically says that the W-hierarchy does not collapse unless a deterministic polynomial-time computation can guess a string of length larger than $\Theta(\log n)$, which seems unlikely based on our current understanding of nondeterminism.

We should point out that Theorem 4.3 gives the first sufficient and necessary condition for each level of the W-hierarchy to collapse to FPT. Previous study on the structural properties of the W-hierarchy has also investigated the consequence of collapsing the W-hierarchy but was only able to give either sufficient or necessary conditions, but not both, for the collapsing of the W-hierarchy. Abrahamson, Downey, and Fellows [2] have carefully studied the relationship between the class FPT and the class $W[P]$, which contains the entire W-hierarchy, and were able to derive a sufficient and necessary condition for the class $W[P]$ to collapse to FPT. For each even level $W[2h]$ of the W-hierarchy, they demonstrated that $\text{FPT} = W[2h]$

implies a necessary consequence that seems very unlikely in classical complexity theory.

ACKNOWLEDGMENTS

We thank Rod Downey and Mike Fellows for informing us of the most recent progress in the study of fixed-parameter tractability and providing valuable comments on this work. We are thankful to Mihalis Yannakakis who read an earlier version of this work and provided valuable advice. In particular, part of the proof of Theorem 4.1 was hinted by a personal communication from him. Finally, we would like to thank an anonymous referee, whose comments, suggestions, and criticism have greatly improved the presentation.

REFERENCES

1. K. R. Abrahamson, R. G. Downey, and M. R. Fellows, "Fixed Parameter Intractability, II," Lecture Notes in Computer Science, Vol. 665, pp. 374–385, Springer-Verlag, New York/Berlin, 1993.
2. K. R. Abrahamson, R. G. Downey, and M. R. Fellows, Fixed parameter intractability and completeness IV: On completeness for $W[P]$ and *PSPACE* analogues, *Ann. Pure Appl. Logic* **73** (1995), 235–276.
3. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, Proof verification and intractability of approximation problems, *in* "Proceedings, 33th IEEE Symposium on Foundations of Computer Science, 1992," pp. 14–23.
4. A. Blummer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, Learnability and the Vapnik–Chervonenkis dimension, *J. Assoc. Comput. Mach.* **36** (1989), 929–965.
5. H. L. Bodlaender, M. R. Fellows, and M. T. Hallett, Beyond NP-completeness for problems of bounded width: Hardness for the *W*-hierarchy, *in* "Proc. 26th ACM Symposium on Theory of Computing, 1994," pp. 449–458.
6. J. F. Buss and J. Goldsmith, Nondeterminism within *P*, *SIAM J. Comput.* **22** (1993), 560–572.
7. L. Cai and J. Chen, "On the Amount of Nondeterminism and the Power of Verifying," Lecture Notes in Computer Science, Vol. 711, pp. 311–320, Springer-Verlag, New York/Berlin, 1993; *SIAM J. Comput.*, to appear.
8. L. Cai and J. Chen, On input read-modes of alternating Turing machines, *Theoret. Comput. Sci.* **148** (1995), 33–55.
9. L. Cai, J. Chen, R. G. Downey, and M. R. Fellows, On the structure of parameterized problems in NP, *Inform. and Comput.* **123** (1995), 38–49.
10. J. Díaz and J. Torán, Classes of bounded nondeterminism, *Math. System Theory* **23** (1990), 21–32.
11. R. G. Downey and M. R. Fellows, Fixed-parameter intractability, *in* "Proc. 7th Structure in Complexity Theory Conference, 1992," pp. 36–49.
12. R. G. Downey and M. R. Fellows, Fixed-parameter intractability and completeness I: basic results, *SIAM J. Comput.* **24** (1995), 873–921.
13. M. R. Fellows, private communication, 1992.
14. M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," Freeman, San Francisco, 1979.
15. O. H. Ibarra and C. E. Kim, Fast approximation algorithms for the knapsack and sum of subset problems, *J. Assoc. Comput. Mach.* **22** (1975), 463–468.
16. P. G. Kolaitis and M. N. Thakur, Approximation properties of NP minimization classes, *J. Comput. System Sci.* **50** (1995), 391–411.
17. E. W. Leggett, Jr. and D. J. Moore, Optimization problems and the polynomial hierarchy, *Theoret. Comput. Sci.* **15** (1981), 279–289.
18. C. H. Papadimitriou and M. Yannakakis, Optimization, approximation, and complexity classes, *J. Comput. System Sci.* **43** (1991), 425–440.
19. C. H. Papadimitriou and M. Yannakakis, On limited nondeterminism and the complexity of the V-C dimension *in* "Proc. 8th Structure in Complexity Theory Conference, 1993," pp. 12–18,
20. S. Sahni, Algorithms for scheduling independent tasks, *J. Assoc. Comput. Mach.* **23** (1976), 116–127.
21. A. Yao, Separating the polynomial-time hierarchy by oracles, *in* "Proc. 26th Annual Symposium on Foundations of Computer Science, 1985," pp. 1–10.