Parameterized Computational Feasibility

RODNEY G. DOWNEY *and MICHAEL R. FELLOWS †

24 November 1992, revised 5 May 1994

Abstract

Many natural computational problems have input consisting of two or more parts. For example, the input might consist of a graph and a positive integer. For many natural problems we may view one of the inputs as a parameter and study how the complexity of the problem varies if the parameter is held fixed. For many applications of computational problems involving such a parameter, only a small range of parameter values is of practical significance, so that fixed-parameter complexity is a natural concern. In studying the complexity of such problems, it is therefore important to have a framework in which we can make qualitative distinctions about the contribution of the parameter to the complexity of the problem. In this paper we survey one such framework for investigating parameterized computational complexity and present a number of new results for this theory.

Introduction

Many natural computational problems have input that consists of two or more objects. For example, the Graph Genus problem is that of determining for an input pair \((G, k)\), where \(G\) is a graph and \(k\) is a positive integer, whether the graph \(G\) has genus at most \(k\). The problem of Minor Testing is that of determining for an input pair of graphs \((G, H)\) whether the graph \(H\) is a minor of the graph \(G\). For every fixed graph \(H\) the latter problem can be solved in time \(O(n^3)\) [RS3].

There are also many natural reducibilities between parameterized problems, such as the reduction of the Graph Genus problem to the problem of Minor Testing established by the Graph Minor Theorem [RS3]. This particular reduction has the striking consequence that for every fixed \(k\) the Graph Genus problem can be solved in time \(O(n^3)\).

---

*Research supported in part by a grant from the Victoria University IGC, by the United States / New Zealand Cooperative Science Foundation under grant INT 90-20558, and by the Mathematical Sciences Institute at Cornell and Cornell University. Email address: downey@math.unsw.edu.au.

†Research supported in part by the National Science and Engineering Research Council of Canada, and by the United States National Science Foundation under grant MIP-8819312. Email address: mrfellows@cs.uvic.ca.
For many other problems involving more than one input we have a sharply contrasting situation (much like the apparent difference between \( P \) and \( NP \)). For example, the best known algorithm for the Minimum Dominating-Set problem [GJ] which takes as input a graph \( G \) and a positive integer \( k \) and seeks to determine whether \( G \) has a set of \( k \) vertices that "covers" the vertex set of \( G \), involves checking all of the \( k \)-element sets of vertices and requires time \( O(n^k) \).

In addition to our structural-theoretic interest in how parameters contribute to the complexity of problems, there are several practical motivations for our interest in parameterized complexity. We give three examples.

**Example 1.** Graph width metrics: VLSI, computational biology and natural language processing.

There are a number of different width metrics for graph and hypergraph linear layout problems, for example, pathwidth [RS1], cutwidth [GJ], gate matrix layout [DKL], vertex separation number [Le], and bandwidth [GJ]. For a number of these problems, good algorithms for finding layouts of width less than or equal to \( k \), for fixed values of \( k \leq 10 \) would have useful applications in VLSI design [DKL]. For the Perfect Phylogeny problem of computational biology [Gu], the number of characters used in constructing the phylogenetic tree corresponds to treewidth (another graph width metric). Phylogenies are routinely computed for data sets based on a small number of characters [BFW]. It has been proposed that the syntactic structure of sentences of natural languages be modeled by dependency graphs of pathwidth no more than 6 (corresponding in some sense to the "bandwidth" of human attention) [Mc].

Thus for many natural parameterized problems, a small range of parameter values captures many important applications, and we are therefore keenly interested in whether efficient algorithms for fixed-parameter versions of the problems can be devised, or whether, by completeness demonstrations, they may be unlikely to exist.

For all of the width metrics \( w \) mentioned above, determining whether an input graph \( G \) satisfies \( w(G) \leq k \) is \( NP \)-complete, yet we can distinguish important qualitative differences in the way the parameter contributes to the complexity of the problem. For example, for every fixed value of \( k \) it can be determined in linear time whether a graph has cutwidth at most \( k \), while the best known algorithm for the \( k \)-bandwidth problem has running time \( O(n^k) \).

**Example 2.** Logic programming.

Type inference is a problem of importance to implementations of programming languages such as ML that are based on polymorphic typed \( \lambda \)-calculus. In [HM] it is shown that the problem is complete for deterministic exponential time, yet it has been widely noted that in practice the problem is solvable quickly. One explanation for this discrepancy between theory and practice comes from noting that the logic formulas that occur in natural programs tend to have small bounded depth of \( \text{let}'s \). For
a parameter $k$ bounding this depth, it can be shown that the problem is fixed-parameter tractable [Ab].

Thus the study of parameterized complexity can shed new light on the observed complexity of some well-known problems.

**Example 3.** Hardware implementations of public-key cryptosystems.

Some proposals for implementations of public key cryptosystems have considered limiting the size or Hamming weight of keys in order to obtain faster processing times. A cautionary note is sounded by the result [FK] that for every fixed $k$, with high probability it can be determined in time $f(k)n^k$ whether an $n$-bit positive integer has a prime divisor less than $n^k$. If a similar result holds for the Discrete Logarithm problem for exponents of bounded Hamming weight, then the security of cryptographic implementations such as proposed in [AMOV] may be compromised. (Both problems are trivially solvable in time $O(n^{k+c})$, where $c$ is a small constant.)

The perspective provided by a theory of parameterized complexity encourages us to perceive and address problems such as the above.

The formal framework for our study is established as follows.

**Definition.** A parameterized problem is a set $L \subseteq \Sigma^* \times \Sigma^*$ where $\Sigma$ is a fixed alphabet.

In the interests of readability, and with no effect on the theory, we consider that a parameterized problem $L$ is a subset of $L \subseteq \Sigma^* \times N$. For a parameterized problem $L$ and $k \in N$ we write $L_k$ to denote the associated fixed-parameter problem ($k$ is the parameter) $L_k = \{ (x,k) \in L \}$.

There are natural examples (some of which are discussed in the next section) of the following three flavours of fixed-parameter tractability.

**Definition.** We say that a parameterized problem $L$ is:

1. **nonuniformly fixed-parameter tractable** if there is a constant $\alpha$ and a sequence of algorithms $\Phi_x$ such that, for each $x \in N$, $\Phi_x$ computes $L_x$ in time $O(n^\alpha)$;

2. **uniformly fixed-parameter tractable** if there is a constant $\alpha$ and an algorithm $\Phi$ such that $\Phi$ decides if $(x,k) \in L$ in time $f(k)|x|^{\alpha}$ where $f : N \to N$ is an arbitrary function;

3. **strongly uniformly fixed-parameter tractable** if $L$ is uniformly fixed-parameter tractable with the function $f$ recursive.

The reader familiar with classical recursion theory will notice the analogy with the classical notion of piecewise recursive recursively enumerable sets. We define three corresponding flavours of reducibility.

**Definition.** Let $A, B$ be parameterized problems. We say that $A$ is **uniformly $P$-reducible** to $B$ if there is an oracle algorithm $\Phi$, a constant $\alpha$, and an arbitrary function $f : N \to N$ such that

(a) the running time of $\Phi(B; (x,k))$ is at most $f(k)|x|^\alpha$,
(b) on input \( \langle x, k \rangle \), \( \Phi \) only asks oracle questions of \( B^{f(k)} \) where

\[
B^{f(k)} = \bigcup_{j \leq f(k)} B_j = \{ \langle x, j \rangle : j \leq f(k) \land \langle x, j \rangle \in B \}
\]

(c) \( \Phi(B) = A \).

If \( A \) is uniformly \( P \)-reducible to \( B \) we write \( A \leq^P_B \) \( B \). Where appropriate we may say that \( A \leq^P_B \) \( B \) via \( f \). If the reduction is many:1 (an \( m \)-reduction), we will write \( A \leq^m_B \) \( B \).

**Definition.** Let \( A, B \) be parameterized problems. We say that \( A \) is strongly uniformly \( P \)-reducible to \( B \) if \( A \leq^P_B \) \( B \) via \( f \) where \( f \) is recursive. We write \( A \leq^P_B \) \( B \) in this case.

**Definition.** Let \( A, B \) be parameterized problems. We say that \( A \) is nonuniformly \( P \)-reducible to \( B \) there is a constant \( \alpha \), a function \( f : \mathbb{N} \rightarrow \mathbb{N} \), and a collection of procedures \( \{ \Phi_k : k \in \mathbb{N} \} \) such that \( \Phi_k(B^{f(k)}) = A_k \) for each \( k \in \mathbb{N} \), and the running time of \( \Phi_k \) is \( f(k) \alpha \). Here we write \( A \leq^P_B \) \( B \).

Note that the above are good definitions, since whenever \( A < B \) with \( < \) any of the reducibilities, if \( B \) is fixed-parameter tractable so too is \( A \).

Note that if \( P = NP \) then problems such as Minimum Dominating Set are fixed-parameter tractable. Thus, a completeness program to address the apparent fixed-parameter intractability of this and other problems is reasonable.

A variety of methods are now known for demonstrating the several flavours of fixed-parameter tractability. It Section 2 we describe some examples of these results and techniques. Some of the methods are straightforward and elementary, and some depend on very deep results in combinatorics.

In Section 3 we describe the basic framework and results of the completeness theory for fixed-parameter tractability.

In Section 4 we discuss some new results that serve to illustrate how the basic reducibilities in fixed-parameter complexity theory differ from the reducibilities in \( NP \)-completeness theory. In particular, we prove that the problem of determining whether a tournament has dominating set of cardinality \( k \) is \( W[2] \)-complete (the general problem is unlikely to be \( NP \)-complete), and we discuss some applications of fixed-parameter reducibilities in computational learning theory.

Section 5 concludes with a discussion of some of the many open problems in this subject.

1 Fixed-Parameter Tractability: Flavors and Techniques

In §1 we defined three different forms of fixed-parameter tractability. There are important natural examples of all three of these, and there are identifi-
able general methods for obtaining such results. We believe it would be fair to say that the toolkit of algorithm design techniques for fixed-parameter tractability is both rich, and somewhat distinctive from the usual toolkit of techniques for demonstrating polynomial-time complexities. The distinctive nature of some of these methods reflects various approaches to shifting the complexity burden onto the parameter.

1.1 Non-uniform Fixed-Parameter Tractability

One of the most striking recent developments in combinatorial mathematics has been the theory of graph minors (and immersions) pioneered by Robertson and Seymour. Their deep results in the area of well-quasi-ordering theory give very powerful and easy to use methods for establishing non-uniform fixed-parameter tractability. For background on these methods and various applications, see [RS1] and [FL1]. It seems likely that the basic theory of well-quasi-ordering will continue to develop, and to support applications to decision problems for many different kinds of combinatorial objects.

We are concerned here with explaining how these methods, which apply to a great variety of natural parameterized problems (see, for example [FL1] and [FL4]), relate to the forms of fixed-parameter tractability defined in §1. The complexity of the following three parameterized problems can be addressed by means of the Graph Minor Theorem (stated below).

**Graph Linking Number**

**Instance:** A graph $G$.

**Parameter:** A positive integer $k$.

**Question:** Can $G$ be embedded in 3-space in such a way that no set of $k$ or more vertex disjoint cycles in $G$ is topologically linked?

**Diameter Improvement for Planar Graphs**

**Instance:** A planar graph $G$.

**Parameter:** A positive integer $k$.

**Question:** Can $G$ be augmented with additional edges in such a way that the resulting graph $G'$ remains planar, and so that the diameter of $G'$ is at most $k$?

**Planarity Edit Distance**

**Instance:** A graph $G = (V, E)$.

**Parameter:** A positive integer $k$.

**Question:** Is there a set of at most $k$ vertices $V' \subseteq V$ such that $G - V'$ is planar?

A graph $H$ is a *minor* of a graph $G$, written $H \leq_m G$ if a graph isomorphic to $H$ can be obtained from $G$ by a sequence of the operations: (1) taking a subgraph, and (2) contracting an edge. (In the contraction of an edge, the the endpoints of the edge become identified as the edge is “shrunk” to nothing.)

**The Graph Minor Theorem.** (Robertson and Seymour [RS4]) If $\mathcal{F}$ is a fam-
ily of finite graphs that is closed under the minor order \((G \in \mathcal{F} \text{ and } H \leq_m G \implies H \in \mathcal{F})\), then there is a finite set of graphs \(O_\mathcal{F} = \{H_1, \ldots, H_t\}\) such that \(G \not\in \mathcal{F}\) if and only if \(G \geq_m H_i\) for some \(H_i \in \mathcal{F}\).

A family of graphs \(\mathcal{F}\) as in the statement of the Graph Minor Theorem is termed a minor order lower ideal, and the set of graphs \(O_\mathcal{F}\) is termed the obstruction set for \(\mathcal{F}\). A classical example of an obstruction set is given by (the minor order version of) Kuratowski's theorem: \(O_{\text{planar}} = \{K_3,3, K_5\}\). It is easy to verify that for each fixed parameter value \(k\) the set of yes-instances for the above problems are minor order lower ideals.

The reader can readily verify that the Graph Minor Theorem provides a nonuniform fixed-parameter Turing reduction of each of the above problems to the problem of Minor Testing.

**Minor Testing**

**Instance:** A graph \(G\)

**Parameter:** A graph \(H\)

**Question:** Is \(G \geq_m H\)?

Minor Testing has been shown by Robertson and Seymour [RS03] to be (strongly uniformly) fixed-parameter tractable in cubic time. Consequently, each of the above problems is nonuniformly fixed-parameter tractable. The Graph Minor Theorem alone does not yield any stronger form of fixed-parameter tractability, because we know only that a finite obstruction set exists for each parameter value \(k\). No information is given by either the theorem or its proof on how many obstructions there are in \(O_k\), how large they are, or how they might be determined.

At the present time, we know only that Graph Linking Number is nonuniformly fixed-parameter tractable, by the above considerations. For the other two problems we can apply general techniques (described below) to show stronger forms of fixed-parameter tractability.

The following theorem shows that the Turing reducibilities provided by the Graph Minor Theorem can be made many-to-one.

**Theorem 2.1** Given a set of graphs \(G, H_1, \ldots, H_t\) we can compute in polynomial time graphs \(G'\) and \(H'\) such that \(G' \geq_m H'\) if and only if \(G \geq_m H_i\) for some \(i, 1 \leq i \leq t\).

**Proof.** (Sketch) Let \(N = 1 + \max\{|H_i| : 1 \leq i \leq t\}\). \(G'\) has a central vertex \(u\) of degree \(t\), as does \(H'\). An example of the construction for \(t = 4\) is shown in Figure 1. In this construction, \(G'\) is essentially a tree with leaves attached to copies of the complete graph \(K_N\) and to one copy of \(G\). An attachment to a copy of \(K_N\) consists of a single edge to a vertex of \(K_N\). The attachment to the copy of \(G\) consists of edges from the leaf to each vertex of \(G\), as indicated pictorially in the figure. The attachments of the leaf vertices of \(H'\) to the graphs of the obstruction set and to the copies of \(K_N\) are similar.\(\square\)
1.2 Uniform Fixed-Parameter Tractability

Many of the computational problems to which the Graph Minor Theorem can be applied can be shown constructively to be uniformly fixed-parameter tractable by the method of [FL2] based on polynomial time self-reducibility. We illustrate this method with the problem of Diameter Improvement for Planar Graphs.

Theorem 2.1 Diameter Improvement for Planar Graphs is (constructively) uniformly fixed-parameter tractable.

Proof. For this problem it is easy to describe the following three algorithms which we will use as subroutines. The first of these, $A$, is simply a decision algorithm for the problem (that does not run in polynomial time) based on exhaustively examining all possible embeddings of $G$ and all possible augmentations of these to a triangulation of the plane. The second auxiliary algorithm $B$ is a polynomial-time self-reduction of the naturally associated search problem (finding a diameter improvement scheme, if one exists) to the decision problem. The third algorithm $C$ that we will use as a subroutine, is a polynomial time algorithm to check whether a given improvement scheme is correct. $C$ simply checks the diameter of the improved graph, and checks that the improved graph is planar.

Here is how algorithm $B$ works. Note that this is an oracle algorithm for which we assume that a decision algorithm for the Diameter Improvement problem is available to use as a black box, and that our job is to compute an improvement scheme, if one exists, in time polynomial in the number $n$ of vertices in the graph, and assuming that each consultation of the oracle requires unit time. For each pair of vertices $u, v$ of $G$, we may ask the
black box whether the graph $G + uv$ is $k$-improvable. If $G$ is $k$-improvable
(and does not already have diameter $\leq k$) then at least one such probe will
succeed. We repeat this procedure (at most $O(n^2)$ times) until we have
discovered an improvement scheme.

Now we argue that using the above three algorithms as subroutines, we
have uniform fixed-parameter tractability for the problem. First, we have
an additional procedure $D$ which generates all finite graphs, beginning with
the empty graph. How efficient $D$ is does not matter to our argument.

Suppose we are given $G$ and $k$ as input. We repeatedly use procedure
$D$ together with procedure $A$ to find a “new” obstruction. We do this by
simply generating graphs using $D$ until we find a graph $H$ with the property
(using $A$ to identify this) that $H \not\in \mathcal{F}$ but every proper minor of $H$
belongs to $\mathcal{F}$ (this property characterizes the obstructions for $\mathcal{F}$).
Having found a “new” obstruction, we add it to a list $L$ of known obstructions.
Note that none of the computations here refer to the graph $G$, and that we will repeat
this generative cycle at most a finite number of times for a given $k$, since
$O_k$ is finite. Thus the total amount of computation involved in this part of
the algorithm is simply bounded by some (unknown, and not necessarily
recursive) function of $k$.

Having found a “new” obstruction $H$ we do the following:
(1) We run the algorithm for Minor Testing to see if $G \succeq_m H$. If so,
then we are done, since this shows that $G \not\in \mathcal{F}$. This requires time $O(n^3)$
for $|G| = n$ for each such $H$, of which there are finitely many, since $O_k$ is
finite.

(2) If the above step (1) fails to settle the question negatively, then we
attempt to discover a positive resolution by running procedure $B$ using the
list $L$ as a (possibly faulty) oracle for $\mathcal{F}$-membership (using the algorithm
for Minor Testing for each of the graphs on the list $L$). The procedure $B$
may malfunction (which we can detect) because of the potentially faulty
oracle we are using, or it may produce a purported improvement scheme.
We can check, using procedure $C$, whether any such purported solution
is correct. If so, then we are done, having produced a certificate for the
answer yes.

If neither of (1) or (2) above produces a (certifiably correct) answer,
then we return to the generative cycle to find a new unknown obstruction.
Within a finite number of cycles, since $O_k$ is finite, either (1) or (2) must
produce a correct answer. It is easy to see that entire algorithm runs
in polynomial time; this depends particularly on the fact that our self-
reduction algorithm $B$ runs in polynomial time, as well as procedure $C$
for checking a solution. \qed

This method cannot presently be applied to the problem of Graph Linking
Number, for the primary reason that a polynomial-time self-reduction
is not known for this problem. Note that the method does not yield any
knowledge of the function $f(k)$ in the running time.
1.3 Strongly Uniform Fixed-Parameter Tractability

The Planarity Edit Distance problem can be shown to be strongly uniformly fixed-parameter tractable by the method of [FL3]. This essentially consists of a uniform method for computing the obstruction sets for the problem. Although this method is applicable to many of the natural lower ideals in the minor order, we presently do not know how to apply it to the Diameter Improvement problem. The method is based on a graph-theoretic generalization of the Myhill-Nerode theorem of formal language theory and is developed further in [AF].

For the remainder of our discussion of uniform fixed-parameter tractability we focus on two widely applicable elementary techniques: (1) Search Trees, and (2) Reduction to a Problem Kernel.

1.3.1 The Method of Search Trees

We next describe how the Search Tree technique can be applied to the well-known problems: Vertex Cover, Dominating Set for Planar Graphs, and Feedback Vertex Set [GJ]. We show how it can be used to improve the results of [BM] on the face cover number of plane graphs. The problems that we consider are defined as follows.

**Vertex Cover**

**Instance:** A graph \( G = (V, E) \).

**Parameter:** A positive integer \( k \).

**Question:** Is there a set of vertices \( V' \subseteq V \) of cardinality at most \( k \) such that for every edge \( uv \in E \), either \( u \in V' \) or \( v \in V' \)?

**Feedback Vertex Set**

**Instance:** A graph \( G = (V, E) \).

**Parameter:** A positive integer \( k \).

**Question:** Is there a set of vertices \( V' \subseteq V \) of cardinality at most \( k \) such that \( G - V' \) is acyclic?

**Dominating Set for Planar Graphs**

**Instance:** A planar graph \( G = (V, E) \).

**Parameter:** A positive integer \( k \).

**Question:** Is there a set of vertices \( V' \subseteq V \) of cardinality at most \( k \) such that for every vertex \( u \in V \), there is an edge \( uv \in E \) for some vertex \( v \in V' \)?

**Face Cover Number for Plane Graphs**

**Instance:** A planar graph \( G = (V, E) \) together with an embedding of \( G \) in the plane.

**Parameter:** A positive integer \( k \).

**Question:** Is there a set \( F \) of at most \( k \) faces of the embedding such that every vertex of \( G \) occurs on the boundary of at least one of face \( f \in F \)?

**Theorem 2.1** Vertex Cover can be solved in time \( O(2^k \cdot n) \) where \( n \) is the number of vertices in the graph (and the hidden constant is independent.
of both $n$ and $k$).

Proof. We construct a binary tree of height $k$ as follows. Label the root of the tree with the empty set, and the graph $G$. Choose an edge $uv \in E$. In any vertex cover $V'$ of $G$ we must have either $u \in V'$ or $v \in V'$, so we create children of the root node corresponding to these two possibilities. Thus the first child is labeled with $\{u\}$ and $G - u$, and the second child is labeled with $\{v\}$ and $G - v$. The set of vertices labeling a node represents a "possible" vertex cover, and the graph labeling the node represents what remains to be covered in $G$. In general, for a node labeled with the set of vertices $S$ and the subgraph $H$ of $G$, we choose an edge $uv \in E(H)$ and create the two child nodes labeled, respectively, $S \cup \{u\}$ and $G - u$, and $S \cup \{v\}$ and $G - v$. If we create a node at height at most $k$ in the tree that is labeled with a graph having no edges, then a vertex cover of cardinality at most $k$ has been found. There is no need to explore the tree beyond height $k$. □

Theorem 2.2 Feedback Vertex Set can be solved in time $O((2k+1)^k \cdot n^2)$.

Proof. First note that a graph $G$ has a feedback vertex set of size $k$ if and only if the reduced graph $G'$ has one, where $G'$ is obtained from $G$ by replacing each maximal path in $G$ having internal vertices all of degree 2 with a single edge. Note that the reduced graph $G'$ may have loops and multiple edges, but that if $G'$ is simple then it has minimum degree 3. The reduced graph $G'$ can be computed from $G$ in linear time. Also, in linear time, a $k$-element feedback vertex set that has been identified in $G'$ can be lifted to a $k$-element feedback vertex set in $G$.

As in the proof of Theorem 2.1, we build a search tree where each node is labeled with a set of vertices $S$ representing a possible partial solution. The cardinality of a label corresponds to the height of the node in the tree, and we will therefore explore the tree to a height of no more than $k$. In linear time we can check whether a set $S$ is a solution. If the label set $S$ of a node in the search tree is not a solution and the node has height less than $k$, then we can generate the children of the node, as follows.

Let $H$ denote the graph $G - S$, and let $H'$ be the reduction of $H$ (as described above). If a vertex $v$ of the $H'$ has a self-loop, then $v$ must belong to every feedback vertex set of $H'$. Corresponding to this observation, we create a single child node with label $S \cup \{v\}$.

If the reduced graph $H'$ of the graph $H = G - S$ has multiple edges between a pair of vertices $u, v \in V(H)$, then either $u$ or $v$ must belong to every feedback vertex set of $H'$, and we correspondingly create two child nodes with labels, respectively, $S \cup \{u\}$ and $S \cup \{v\}$.

If the reduced graph $H'$ has no loops or multiple edges, then we can make use of the following.

Claim. If a simple graph $J$ of minimum degree 3 has a $k$-element feedback vertex set, then the girth of $J$ (the length of a shortest cycle) is bounded above by $2k$.

We prove this by induction on $k$. If $J$ is simple then by a standard
result \( J \) must contain a subdivision of \( K_3 \) [Lo], and this implies that a feedback vertex set must contain at least two elements.

For the induction step suppose \( U' \) is a feedback vertex set consisting of \( k + 1 \) vertices of \( J \). Suppose that \( u, v \in U' \) with the distance from \( u \) to \( v, d(u, v) \leq 2 \) in \( J \). Contracting the edges of a shortest path from \( u \) to \( v \) yields a graph \( J' \) of minimum degree 3 that has a feedback vertex set of \( k \) elements. By the induction hypothesis, there is a cycle \( C \) in \( J' \) of length at most \( 2k \). This implies that there is a cycle in \( J \) of length at most \( 2k + 2 \). Otherwise, suppose no two vertices \( u, v \) of \( U' \) have \( d(u, v) \leq 2 \) in \( J \). Then every vertex of \( J - U' \) has degree at least two, and so there is a cycle in \( J \) not containing any vertex of \( U' \), a contradiction. This establishes our claim.

By the above claim, we know that for the node of the search tree that we are processing, either \( H' \) contains a cycle of length at most \( 2l \) where \( l = k - |S| \), or that \( S \) cannot be extended to a \( k \)-element feedback vertex set. An algorithm of Ibarra and Rodeh [IR] can be employed to find in \( H' \) a cycle of length \( 2l \) or \( 2l + 1 \) in time \( O(n^2) \). Thus in time \( O(n^2) \) we can either decide that the node should be a leaf of the search tree (because there is no cycle in \( H' \) of length at most \( 2l + 1 \)) or we can find a short cycle and create at most \( 2l + 1 \) children, observing that at least one vertex of the short cycle that we discover in \( H' \) must belong to any feedback vertex set.

We remark that it is possible to show that Feedback Vertex Set is linear time fixed-parameter tractable [Bo,DF] with running time \( O((17k^4)! \cdot n) \). Whether the directed version of the problem is fixed-parameter tractable is presently unknown.

For the next example of the Tree Search technique, we will make use of the following lemma concerning planar graphs.

**Lemma 2.3** If \( G = (V, E) \) is a simple planar graph with a vertex partition into two sets \( V = V_1 \cup V_2 \) satisfying:

1. the minimum degree of vertices in \( V_1 \) is at least 3, and
2. \( V_1 \) is an independent set in \( G \),

then there is a vertex \( u \in V_2 \) of degree at most 10 in \( G \).

**Proof.** Let \( G \) be a counterexample of minimum possible order having a maximum number of edges, and consider an embedding of \( G \) in the plane. Let \( H \) denote the subgraph of \( G \) induced by \( V_2 \). In any face of the inherited embedding of \( H \), there can be at most one vertex of \( V_1 \), else an edge could be added between two vertices of \( V_2 \) on the boundary of the face, and therefore \( G \) would not have a maximum number of edges as supposed. Let \( u \) be a vertex of degree at most 5 in \( H \). The vertex \( u \) is on the boundary of at most 5 faces of \( H \), and consequently in \( G \), \( u \) has degree at most 10. \( \square \)

**Theorem 2.4** Dominating Set for Planar Graphs can be solved in time \( O(11^k \cdot n) \).

**Proof.** We construct a search tree for which each node has at most 10
children. Each node in the tree is labeled with a set of vertices \( S \) that represents a partially constructed dominating set.

The root node, labeled with the empty set, will have at most 6 children based on the following consideration. Since \( G \) is planar, \( G \) has a vertex \( v \) of degree at most 5 which can be found in linear time. Any dominating set for \( G \) must contain either \( v \) or one of the neighbors of \( v \). We create a child node for each possible choice of a vertex to dominate \( v \).

In general, for a node in the search tree, we first check whether \( S \) is a dominating set. This can be done in linear time. The levels of the tree correspond to the cardinality of the labels \( S \), so the tree will have height at most \( k \). To compute the children of a node, we find a vertex \( u \) in \( G \) not dominated by \( S \) that has degree at most 10, and create a child node for each possible choice \( y \) of a vertex to dominate \( u \) (there can be at most 11 possibilities, including \( u \)). The child node is labeled with \( S \cup \{ y \} \). We must argue that such a vertex \( u \) must be available; this being so, it can easily be found in linear time. We term such a vertex \( u \) a splitter for the node in the search tree.

Let \( U \) denote the set of vertices not dominated by \( S \), and let \( T \subseteq V - S - U \) be the set of vertices not in \( S \) and not in \( U \). Let \( H \) be the subgraph of \( G \) induced by \( V - S \cup T \cup U \), and let \( H' \) be the subgraph of \( H \) obtained by deleting from \( H \) any edges between vertices of \( T \).

Observe that a set of vertices \( W \subseteq V - S \) has the property that \( S \cup W \) is a dominating set in \( G \) if and only if \( W \) is a dominating set in \( H' \). In other words, we may restrict our attention to \( H' \) in searching for a splitter. \( H' \) satisfies condition (2) of Lemma 3.3, but there may be vertices in \( T \) that have degree 2 in \( H' \). Necessarily any such vertex \( r \in T \) of degree 2 has two neighbors \( s, t \in U \). Consider the graph \( H'' \) obtained from \( H' \) by deleting such vertices \( r \) and adding the edges \( st \). Lemma 3.3 applies to \( H'' \), so there is a vertex \( u \in U \) in \( H'' \) of degree at most 10. The splitter vertex \( u \) also has degree at most 10 in \( H' \), \( H \) and \( G \).

We can prove a similar result for the following more general problem.

**Planar Red/Blue Dominating Set**

Instance: A planar bipartite graph \( G = (V, E) \), \( V = V_{\text{red}} \cup V_{\text{blue}} \).

Parameter: A positive integer \( k \).

Question: Is there a set \( V' \subseteq V_{\text{red}} \) of cardinality at most \( k \) such that every vertex of \( V_{\text{blue}} \) is adjacent to at least one vertex of \( V_{\text{red}} \)?

**Theorem 2.5** Planar Red/Blue Dominating Set is solvable in time \( O(12^{k} \cdot n) \).

**Proof.** Let \( G \) be an instance of the problem. We apply the search tree technique essentially as in Theorem 2.4. The central point we must argue is that a node can be expanded to at most 12 children in linear time, without losing the possibility of discovering a solution if one exists.

Let \( S \subseteq V_{\text{red}} \) be the label on a node in the search tree. Let \( B(S) \subseteq V_{\text{blue}} \) denote the vertices in \( V_{\text{blue}} \) dominated by \( S \). Let \( T = V_{\text{red}} - S \) and \( U = V_{\text{blue}} - B(S) \). It suffices to argue that there is a vertex \( u \in U \) of degree at most 10 in the subgraph \( H \) induced by the vertices of \( T \cup U \).
Let \( T_i \subseteq T \) be the vertices of \( T \) in \( H \) of degree \( i \), for \( i = 1, 2 \). Note that any two vertices \( x, y \) of \( T_1 \) adjacent to the same vertex of \( U \) in \( H \) are equivalent, in the sense that there is an extension of \( S \) that is a solution for \( G \) containing \( x \) if and only if there is a solution extension of \( S \) containing \( y \). Thus, without loss of generality, we may assume: (*) each vertex of \( U \) in \( H \) is adjacent to at most one vertex of \( T_1 \).

Let \( H' \) be the same graph as \( H - T_1 \), but considering each vertex of \( T_2 \) as a "virtual edge" between the two vertices of \( U \) to which it is adjacent. \( H' \) satisfies the conditions of Lemma 2.3 and therefore there is a vertex \( u \in U \) of degree at most 10 in \( H' \) and in \( H - T_1 \) as well. Taking (*) into account, it suffices to create at most 12 children in the search tree for the node being processed.

\[ \text{Theorem 2.6} \quad \text{Face Cover Number for Plane Graphs can be solved in time } O(12^k \cdot n). \]

\[ \text{Proof.} \quad \text{Let } G \text{ be a plane graph (a graph together with an embedding in the plane). In linear time we may reduce the problem of finding } k \text{ faces of the embedding which cover all vertices of } G \text{ to an instance of red/blue planar dominating set, by creating one red vertex for each face of the embedding of } G \text{ and connecting it to each (blue) vertex on the boundary of the face.} \]

We remark that Theorem 2.6 is an improvement on the result for this problem in [BM], where a time bound of \( O(2^{8k} \cdot n) \) is obtained. Our method of proof is also considerably simpler.

### 1.3.2 The Method of Reduction to a Problem Kernel

The main idea of this method is to reduce (in polynomial time) a problem instance \( I \) to an "equivalent" instance \( I' \), where the size of \( I' \) is bounded by some function of the parameter \( k \). The instance \( I' \) is then exhaustively analyzed, and a solution for \( I' \) can be lifted to a solution for \( I \), in the case where a solution exists. We illustrate the method with the problems Vertex Cover and Max Leaf Spanning Tree [GJ] (defined below).

\[ \text{Theorem 2.7 (Buss [Bu])} \quad \text{Vertex Cover can be solved in time } O(n + k^k). \]

\[ \text{Proof.} \quad \text{Observe that for a simple graph } G \text{ any vertex of degree greater than } k \text{ must belong to every } k \text{-element vertex cover of } G. \]

**Step 1:** Locate all vertices in \( H \) of degree greater than \( k \); let \( p \) equal the number of such vertices. If \( p > k \), there is no \( k \)-vertex cover, otherwise, let \( k' = k - p \).

**Step 2:** Discard all \( p \) vertices found in step 1 and the edges incident to them. If the resulting graph \( H' \) has more than \( k'(k + 1) \) vertices, reject.

**Step 3:** If \( H' \) has no \( k' \)-vertex cover, reject. Otherwise, any \( k' \)-vertex cover of \( H' \) plus the \( p \) vertices from step 1 comprise a \( k \)-vertex cover of \( H \).
The bound $2k'(k + 1)$ in step 2 is justified by the fact that a simple graph with a $k'$-vertex cover and degrees by bounded by $k$ has no more than $k'(k + 1)$ vertices. For fixed $k$ this makes step 3 a constant time operation, where the constant is $O(k^k)$. □

We can similarly solve the following problem.

**Max Leaf Spanning Tree**

**Instance:** A graph $G = (V, E)$.

**Parameter:** A positive integer $k$.

**Question:** Is there a spanning tree of $G$ with at least $k$ leaves?

**Theorem 2.8** Max Leaf Spanning Tree can be solved in time $O(n + (2k)^{4k})$.

**Proof.** Note that any graph $G$ that is a yes instance must be connected. We will argue that any sufficiently large graph without useless vertices of degree 2 is necessarily a yes instance. Note also that if $G$ has a vertex of degree at least $k$, then $G$ is a yes instance.

A vertex $v$ of degree 2 is termed useless if it has neighbors $u, w$ of degree 2. Say that a useless vertex $v$ is resolved by deleting $v$ from $G$ and adding an edge between $u$ and $w$. Let $G'$ denote the graph obtained from $G$ (in linear time) by resolving all useless vertices.

Our algorithm for Max Leaf Spanning Tree is very simply described:

**Step 1.** Check whether $G$ is connected, and whether there is a vertex of degree $\geq k$.

**Step 2.** If the answer is still undetermined, then compute $G'$. If $G'$ has at least $3k(k + 1)$ vertices then the answer is yes.

**Step 3.** Otherwise, exhaustively analyze $G'$ and answer accordingly, since $G'$ has a $k$-leaf spanning tree if and only if $G$ does.

The argument that the algorithm is correct is elementary; details will be given elsewhere [CCDF, DF6]. □

Theorem 2.8 improves a result of Bodlaender, who showed that Max Leaf Spanning Tree is linear-time fixed-parameter tractable with a multiplicative factor depending on $k$ [Bo1].

We remark that we do not at present know whether the problems Feedback Vertex Set or Planar Dominating Set can be shown to be linear fixed-parameter tractable by the method of reduction to a problem kernel, or whether they can be solved in time $O(n + C_k)$ by any method.

The exploration and articulation of standard techniques for algorithm design for fixed-parameter problems (with the goal of establishing fixed-parameter tractability) is an interesting area for further research. It appears that demonstrations of fixed-parameter tractability can sometimes be obtained by novel approaches that shift the complexity burden onto the parameter. In some cases, effective strategies for doing this seem to run counter to our established practices and habits of thought in designing polynomial-time algorithms. In the parameterized setting, the parameter can be "sacrificed" in interesting ways.

How much improvement might be possible in Theorems such as 2.4 and 2.8? Because the algorithms are uniform, and assuming $P \neq NP$, we
Parameterized Computational Feasibility

must expect an additive or multiplicative factor that is super-polynomial in \( k \). Yet conceivably running times such as \( O(c^k \cdot n) \) are possible, where \( c \) is a small number greater than 1, and may be practical for a reasonable range of parameter values.

Note that the method of reduction to a problem kernel raises issues similar to those considered in the model of advice classes such as \( P/poly \) [KL]. Here, however, in reasonable time we can answer for instances of arbitrary size, given the help provided by a kernel of advice for the parameter value \( k \). For example, in the case of Theorem 2.8 this could take the form of a circuit to determine for a graph of order at most \( 3k(k+1) \) whether the graph has a \( k \)-leaf spanning tree.

2 Parameterized Complexity Classes

In order to frame a completeness theory to address the apparent fixed-parameter intractability of Dominating Set and other problems, we need to define appropriate classes of parameterized problems. The classes that we define below are intuitively based on the complexity of the circuits required to check a solution, or alternatively the "natural logical depth" of the problem. (See also [CC] for a view of this idea in terms of alternating logarithmically bounded Turing Machines.)

We first define circuits in which some gates have bounded fan-in and some have unrestricted fan-in. It is assumed that fan-out is never restricted.

Definition. A Boolean circuit is of mixed type if it consists of circuits having gates of the following kinds.

(1) Small gates: not gates, and gates and or gates with bounded fan-in. We will usually assume that the bound on fan-in is 2 for and gates and or gates, and 1 for not gates.

(2) Large gates: And gates and Or gates with unrestricted fan-in.

We will use lower case to denote small gates (or gates and and gates), and upper case to denote large gates (Or gates and And gates).

Definition. The depth of a circuit \( C \) is defined to be the maximum number of gates (small or large) on an input-output path in \( C \). The weight of a circuit \( C \) is the maximum number of large gates on an input-output path in \( C \).

Definition. We say that a family of circuits \( F \) has bounded depth if there is a constant \( h \) such that every circuit in the family \( F \) has depth at most \( h \). We say that \( F \) has bounded weight if there is constant \( t \) such that every circuit in the family \( F \) has weight at most \( t \). \( F \) is monotone if the circuits of \( F \) do not have not-gates. \( F \) is a decision circuit family if each circuit has a single output. A decision circuit \( C \) accepts an input vector \( x \) if the single output gate has value 1 on input \( x \). The weight of a boolean vector \( x \) is the number of 1's in the vector.
Definition. Let \( F \) be a family of decision circuits. We allow that \( F \) may have many different circuits with a given number of inputs. To \( F \) we associate the parameterized circuit problem \( L_F = \{(C, k) : C \text{ accepts an input vector of weight } k\} \).

Definition. A parameterized problem \( L \) belongs to \( W[1] \) (monotone \( W[1] \)) if \( L \) uniformly reduces to the parameterized circuit problem \( L_F \) for some family \( F \) of bounded depth, mixed type (monotone) decision circuits of weight at most \( t \).

As an example of problem classification we offer the following. The V-C dimension of a family of sets is an important concept in computational learning theory [BEHW].

**Definition.** The Vapnik-Chervonenkis dimension of a family of sets \( \mathcal{F} \) of a base set \( U \) is defined to be the maximum cardinality of a set \( S \subseteq U \) that is shattered by \( \mathcal{F} \). That is, for each subset \( T \subseteq S \) there is a set \( A \in \mathcal{F} \) such that \( A \cap S = T \).

**Proposition 3.1.** The problem of determining whether the V-C dimension of a family of sets is at least \( k \) is in \( W[1] \).

**Proof.** Let \( \mathcal{F} \subseteq 2^U \) denote the family of sets under consideration. Suppose \( \mathcal{F} = \{X_j : 1 \leq j \leq m\} \) and \( U = \{1, \ldots, n\} \). It suffices to show that in time \( f(k) \cdot (mn)^{k} \) we can produce a product-of-sums Boolean expression \( E \) in which the clauses have size bounded by some constant, and such that \( E \) has a satisfying truth assignment of weight \( k' = g(k) \) if and only if the V-C dimension of \( \mathcal{F} \) is at least \( k \).

The set of variables for \( E \) is \( V = V_1 \cup V_2 \) where:
\[
V_1 = \{a[i,j] : 1 \leq i \leq 2^k, 1 \leq j \leq m\}
\]
\[
V_2 = \{b[r,s] : 1 \leq r \leq k, 1 \leq s \leq n\}
\]

The variables of \( V_1 \) serve to indicate which sets in \( \mathcal{F} \) witness the shattering of the \( k \)-element subset of \( U \) indicated by the variables of \( V_2 \). Let \( \gamma \) be a fixed 1:1 correspondence between the integers \( i \) in the range \( 1 \leq i \leq 2^k \) and the length \( k \) 0-1 vectors, and write \( \gamma_i(l) \in \{0, 1\} \) to indicate the value of the \( l \)-th component of the vector associated to \( i \).

We will take \( k' = k + 2^k \), and \( E = E_1 \cdot E_2 \) as follows.

\( E_1 \) is a product of small clauses that enforces the conditions:

1. For each index value of \( i \) (\( r \)) in the definition of \( V_1 \) (\( V_2 \)), at most one variable of \( V_1 \) (\( V_2 \)) is set to \textit{true}.
2. If \( b[i,j] \) and \( b[i',j'] \) are set to \textit{true}, with \( i < i' \), then \( j < j' \).

This can be accomplished with clauses of size 2.

Note that any satisfying truth assignment to \( E_1 \) of weight \( k + 2^k \) must set \textit{exactly} one variable of \( V_1 \) (\( V_2 \)) \textit{true} for each index value of \( i \) (\( r \)).

\( E_2 \) is the product of clauses expressing the implications: \( a[i,j] \Rightarrow \neg b[r,s] \) for all \textit{incompatible} pairs of indices \((i,j)\) and \((r,s)\), where such a pair is defined to be \textit{incompatible} if and only if either (1) \( \gamma_i(r) = 1 \) and \( s \not\in X_j \) or (2) \( \gamma_i(r) = 0 \) and \( r \not\in X_j \).

The verification that this works correctly is straightforward. \( \square \)

**Definition.** We denote the class of fixed-parameter tractable problems
Thus we have the containments

\[ FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots \]

and we conjecture that each of these containments is proper. We term the union of these classes the \( W \) Hierarchy, and denote it \( WH \). We have the following implication if \( P = NP \).

**Lemma 3.1** If \( P = NP \) then \( WH \subseteq FPT \). \( \square \)

The following theorem plays a role in our theory analogous to Cook's theorem for \( NP \)-completeness. A parameterized variation of Satisfiability based on a normal form for boolean expressions supplies the problems that we identify as complete for the various levels of \( WH \).

**Definition.** A boolean expression \( X \) is termed \( t \)-normalized if:

1. \( t = 2 \) and \( X \) is in product-of-sums (P-o-S) form,
2. \( t = 3 \) and \( X \) is in product-of-sums-of-products (P-o-S-o-P) form,
3. \( t = 4 \) and \( X \) is in P-o-S-o-P-o-S form,

... etc.

**Weighted \( t \)-Normalized Satisfiability**

**Input:** A \( t \)-normalized boolean expression \( X \) and a positive integer \( k \).

**Question:** Does \( X \) have a satisfying truth assignment of weight \( k \)?

**Theorem 3.2 [DF1]** Weighted \( t \)-Normalized Satisfiability is complete for \( W[t] \) for \( t \geq 2 \). \( \square \)

Independent Set is an example of a problem complete for \( W[1] \) (see [DF3]). Dominating Set is shown to be complete for \( W[2] \) in [DF1]. In order to address the issue of fixed-parameter complexity for problems for which solutions can be checked in polynomial time it is natural to define the following complexity class.

**Definition.** A parameterized problem \( L \) belongs to \( W[P] \) (\textit{monotone} \( W[P] \)) if \( L \) uniformly reduces to the parameterized circuit problem \( L_F \) for some family of circuits \( F \).

Note that \( W[t] \) is contained in \( W[P] \) for every \( t \), and that \( W[P] = FPT \) if \( P = NP \). The following problems, for example, are complete for \( W[P] \) (see [ADF2]):

**Monotone Circuit Satisfiability**

**Instance:** A monotone circuit \( C \) and a positive integer \( k \).

**Question:** Does \( C \) accept an input vector of weight \( k \)?

**Degree Three Subgraph Annihilator**

**Instance:** A graph \( G = (V, E) \) and a positive integer \( k \).

**Question:** Is there a set \( X \subseteq V \) of at most \( k \) vertices such that \( G - X \) has no subgraph of minimum degree three.

Density questions concerning parameterized complexity classes and reductions offer significantly more difficult challenges than do corresponding questions in the study of ordinary polynomial-time reducibility. By
rather demanding methods (e.g., $O^*$ priority arguments) we have been able to show the following analogue of Ladner's classical density theorem.

**Theorem 3.3** [DF5] If any of the containments

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

of the $W$ Hierarchy is proper, then there are infinitely many intervening equivalence classes of parameterized problems with respect to strong uniform reductions.

The analogous density question with respect to uniform reductions remains open. A compendium of parameterized problems presently known to be complete for various levels of the $W$ hierarchy can be found in [DF4] and [DF6].

## 3 Fixed-Parameter Reducibilities and Completeness

In this section we describe a number of new results which serve to illustrate how the reductions in the theory of fixed-parameter complexity differ from the familiar reductions of $NP$-completeness.

**Definition.** A tournament is a directed graph $G = (V, A)$ such that for every pair of vertices $u, v \in V$ exactly one of $uv$ and $vu$ belongs to the set of arcs $A$.

It is easy to observe that a tournament of order $n$ must have a dominating set of order $\log n$. (One can be constructed by repeatedly selecting and removing a vertex having outdegree $\geq$ indegree.) Thus, this problem can be solved with a polylogarithmic amount of nondeterminism, and therefore is unlikely to be complete for $NP$. (For a further study of polynomial-time computational power augmented by limited amounts of nondeterminism see [BG], [Re] and [CC].)

**Tournament Dominating Set**

**Instance:** A tournament $T$ and a positive integer parameter $k$.

**Question:** Does $T$ have a dominating set of cardinality at most $k$?

**Theorem 4.1** Tournament Dominating Set is complete for $W[2]$.

**Proof.** As a special case of Dominating Set it is easily seen to be in $W[2]$. To show that it is hard for $W[2]$ we reduce from Dominating Set. Let $G = (V, E)$ be an undirected graph for which we wish to determine whether $G$ has a dominating set of size $k$. We describe how a tournament $T$ can be constructed that has a dominating set of size $k + 1$ if and only if $G$ has a dominating set of size $k$. The size of $T$ is $O(2^k \cdot n)$ where $n$ is the number of vertices of $G$, and $T$ can be constructed in time polynomial in $n$ and $2^k$.

The vertex set of the tournament $T$ is partitioned into three sets: $V_A$, $V_B$ and $V_G$. The vertices in the set $V_A$ are in 1:1 correspondence with the vertices of $G$, and we write $V_A = \{a[u] : u \in V(G)\}$. The set of vertices $V_B$
of $T$ corresponds to $m$ copies of the vertex set of $G$ and we may write this as $V_B = \{b[i, u] : 1 \leq i \leq m, u \in V(G)\}$. (The appropriate cardinality for $m$ is determined below.) $V_C$ consists of just a single vertex which we will denote $c$.

The construction of $T$ must insure that for every pair of vertices $x, y$ of $T$, one of the arcs $xy$ or $yx$ is present. Let $T_0$ be any tournament on $n$ vertices (we will use $T_0$ as "filler"). Include arcs in $T$ to make a copy of $T_0$ between the vertices of each of the $n$-element sets $V_A$ and $V_B(i) = \{b[i, u] : u \in V(G)\}$ for $i = 1, \ldots, m$.

Let $T_1$ be a tournament on $m$ vertices that has no dominating set of size $k + 1$. It can be shown that there are easy constructions of such a tournament, with $m = O(2^{k+1})$. Consider that the vertex set of $T_1$ is $V(T_1) = \{1, \ldots, m\}$. For each arc $ij$ in $T_1$ include in $T$ an arc from each vertex of $V_B(i)$ to each vertex of $V_B(j)$.

The adjacency structure of $G$ is represented in $T$ in the following way: for each vertex $u \in V(G)$ include arcs from the vertex $a[u]$ to the vertices $b[i, u]$ for every $v \in N_G(u)$ and for each $i$, $1 \leq i \leq m$, and from every other vertex in $V_B$ include an arc to $a[u]$. Thus the neighborhood structure of $G$ is represented in arcs from $V_A$ to $V_B$, and otherwise there are arcs from $V_B$ to $V_A$.

Finally, there are arcs in $T$ from $c$ to every vertex in $V_A$ and from every vertex in $V_B$ to $c$.

We now argue that the construction works correctly. If there is a $k$-element dominating set $S$ in $G$, then the corresponding vertices in $V_A$ dominate all of the vertices in $V_B$. Thus together with $c$ we have a dominating set of size $k + 1$ in $T$.

Conversely, suppose $T$ has a dominating set $D$ of size $k + 1$. At least one vertex of $D$ must belong to $V_B$ or $V_C$, else the vertex $c$ is not dominated. Thus there are at most $k$ vertices of $D$ in $V_A$. Let $S_A$ denote the corresponding vertices of $G$. If $S_A$ is not a dominating set in $G$, then let $x \in V_A$ denote some vertex of $G$ that is not dominated. Let $D_A = D \cap V_A$, and let $D_B = D \cap V_B$.

The vertices $b[i, x]$ of $V_B$ for $1 \leq i \leq m$ are not dominated in $T$ by the vertices of $D_A$. The vertices of $V_B$ can be viewed as belonging to $m$ copies of $V(G)$ for which we have introduced the notation $V_B(i)$, $1 \leq i \leq m$. Say that a copy $V_B(i)$ is occupied if $V_B(i) \cap D_B \neq \emptyset$. There are at most $k + 1$ occupied copies of $V(G)$ in $V_B$. Because of the structure of $T_1$ used in the construction of $T$, these occupied copies do not dominate all of the copies of $V(G)$ in $V_B$. Let $j$ be the index of an undominated copy. But then the vertex $b[j, x]$ of $T$ is not dominated by $D$, a contradiction. Thus $S_A$ must be a dominating set of cardinality at most $k$ in $G$.

The above reduction would not suffice to demonstrate that the problem is $NP$-complete, for the reason that the reduction is, "exponential in $k". This is one fundamental way in which parameterized problem reducibilities may differ from the "usual" polynomial-time reducibilities. The fact that
the blow-up is "only" exponential in \( k \) yields as a Corollary the result of
[PY] that Tournament Dominating Set is complete for the complexity class
LOGSNP. We remark, however, that most of our earlier reductions in, for
instance, [DF1-5], are in fact polynomial in both \( n \) and \( k \), and yield com-
pleteness results simultaneously for all intermediate classes such as LOGNP etc.
as well as for \( NP \). Since determination of the \( V-C \) dimension is
LOGNP complete ([PY]), it is a very interesting question to ask if this
problem is, for instance, \( W[1] \)-complete.

The reduction in Theorem 4.1 also has an interesting property that we
will term blindness, informally defined as follows.

**The Blindness of the Reduction.** The vertex set of the tournament \( T(G) \)
that is constructed depends only on the size of \( G \), and furthermore, the
correspondence between the solutions to the two problems is such that for
a pair of corresponding problem instances \((G, T(G))\):

1. A solution for \( T(G) \) can be computed from a solution for \( G \), even if \( G \)
is invisible, that is, knowing only the size of \( G \).
2. A solution for \( G \) can be similarly computed from a solution for \( T(G) \),
even if \( T(G) \) is invisible.

A moment's reflection on familiar reductions in the theory of \( NP \)-
completeness will reveal that the reductions there almost never have this
property. Reductions in fixed-parameter complexity theory often do possess
this property, which provides for interesting applications in computational
learning theory. We briefly describe how these applications come about.

In the model of exact learning by membership and (extended) equi-
valence queries we study the situation where the Teacher possesses (for ex-
ample) a graph on \( n \) vertices that is hidden from the Learner. We assume
that the Learner initially knows nothing about the graph, not even its size.
The goal of the Learner is to produce "knowledge" of, for example, the
dominating sets in the graph. Such knowledge can be represented by an
algorithm (e.g., a circuit) which decides whether a given set of vertices is
dominating set in the graph being taught.

There are two ways in which the Learner may interact with the Teacher:

1. By presenting a knowledge representation to the Teacher and asking
   whether it is correct. The Teacher responds either yes (in which case
   the Learner is done), or provides a counterexample to the correctness of
   the knowledge representation. This interaction is termed an (extended)
   equivalence query.
2. By asking whether a particular set of vertices is a dominating set in
   the graph being taught. The Teacher responds either yes or no. This
   interaction is termed a membership query.

The central question that is studied is whether there is an algorithm
that the Learner can execute in polynomial time (in the size of the graph
to be learned) which will yield a complete and correct knowledge rep-
resentation in interaction with any Teacher who responds correctly (but not
necessarily judiciously) to the queries.
The blindness of the reduction of Theorem 4.1 allows us to prove the following.

**Theorem 4.2** If the $k$-element dominating sets of a tournament can be learned in fixed-parameter polynomial time, then so can the $k$-element dominating sets in an arbitrary graph.

*Proof.* (Sketch) The Learner is interacting with a Teacher of $k$-element dominating sets in a graph of unknown size. By diagonalization, however, the Learner can be presumed to have knowledge of the size of the graph. The Learner forms a mental model of a tournament as described by the reduction of Theorem 4.1. Since the graph is not visible to the Learner, this model is not complete. The Learner pretends to learn the dominating sets in the tournament. Because of the blindness of the reduction, queries that the Learner wishes to make about the tournament can be translated into queries about dominating sets in the graph that the Teacher actually present is teaching, and similarly, counterexamples provided by the Teacher can be translated into counterexamples concerning the (imaginary) tournament. When the Learner has a correct knowledge representation of the dominating sets in the tournament, this provides (by composing with the blind translation of solutions between the problems) a correct representation of the dominating sets in the graph. □

By a similar but more intricate argument using the fixed-parameter reduction of Weighted Satisfiability to Monotone Weighted Satisfiability we have been able to prove the following strengthening of a theorem of Valiant [Val]. (Details will appear elsewhere. [DEF])

**Theorem 4.3** Arbitrary DNF formulas can be exactly learned in polynomial time by extended equivalence queries (only) if and only if monotone DNF formulas can be so learned. □

The main point of this discussion is that fixed-parameter reductions differ in a number of interesting and non-trivial ways from the reductions usually found in the theory of $NP$-completeness. We also note that in many cases they seem to be somewhat more difficult to devise, although many problems have been identified as complete or hard for various levels of the $W$ hierarchy. (A published compendium can be found in [DF4] and the authors maintain a current list available electronically).

## 4 Open Problems

There are a number of concrete problems for which a demonstration of either fixed-parameter tractability or completeness (or hardness) for some level of the $W$ hierarchy would be interesting, and in the case of tractability, of possible practical significance. The current “most wanted” list includes the problems: **Directed Feedback Vertex Set [GJ]**, **Perfect Phylogeny for $k$ Characters [BFW]**, **Graph Bandwidth [GJ]**, **$V$-C Dimension [BEHW]**, **Traveling Salesman Tour of at least $k$ Cities [GJ]**, and **$k$-Small Steiner**
Trees in Graphs [GJ].
An important issue that has so far been little explored is that of fixed-parameter approximation algorithms. One of the cornerstones of the Robertson-Seymour results concerning graph minors is an algorithm that in time $f(k)\cdot n^2$ either determines that an input graph $G$ has treewidth greater than $k$, or produces a tree decomposition of width at most $5k$. (This has since been improved by a number of authors, with the best result presently due to Bodlaender [Bo2].) What are the prospects for similar approximation algorithms for Dominating Set and other parameterized problems? (Note that this is different from the question of the existence of a relative approximation algorithm in the usual sense.)

Perhaps the foremost structural question regarding the $W$ hierarchy is whether collapse at the $k$-th level ($W[k-1] = W[k]$) propagates either downward (so that $FPT = W[k]$) or upward (so that $W[k-1] = Wh$).

In [ADF2] the following connection is made between the issue of the proper nature of the $W$ hierarchy and a “quantitative” version of $P = NP$ question.

Theorem. $FPT = W[P]$ if and only if Satisfiability can be solved in time $p(n) \cdot 2^{o(n)}$ where $p$ is a polynomial, $n$ is the size of the boolean expression, and $v$ is the number of variables.

References


Conjecture,” to appear.

Rodney Downey
Mathematics Department
Victoria University
P.O. Box 600, Wellington
New Zealand

Michael Fellows
Department of Computer Science
University of Victoria
Victoria, British Columbia, V8W 3PY
Canada