Fixed-Parameter Tractability and Completeness II: On Completeness for W[1]

Rod G. Downey
Mathematics Department
Victoria University
Wellington, New Zealand

Michael R. Fellows
Computer Science Department
University of Victoria
Victoria, B.C. Canada

Abstract

For many fixed-parameter problems that are trivially solvable in polynomial-time, such as k-DOMINATING SET, essentially no better algorithm is presently known than the one which tries all possible solutions. Other problems, such as FEEDBACK VERTEX SET, exhibit *fixed-parameter tractability*: for each fixed k the problem is solvable in time bounded by a polynomial of degree c, where c is a constant independent of k. In a previous paper, the W Hierarchy of parameterized problems was defined, and complete problems were identified for the classes $W[t]$ for $t \geq 2$. Our main result shows that INDEPENDENT SET is complete for $W[1]$.

1. Introduction

Many natural computational problems have input that consists of a pair of items. For practical applications, it is often the case that only a small range of parameter values are significant.

We now have encouraging fixed-parameter tractability results for many problems. For example, for each fixed parameter value k, it can be determined whether a graph G on n vertices has k disjoint cycles in time $O(n)$ [Bo,DF1]. MINOR TESTING and the GRAPH GENUS problem can be solved in $O(n^3)$ time for each fixed parameter value by the deep results of Robertson and Seymour [RS1,RS2].

There are many other parameterized problems, such as DOMINATING SET, for which essentially no better algorithm is presently known than the trivial brute-force algorithm that checks all sets of k vertices.

The following definitions provide a framework for the study of fixed-parameter complexity.

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

Definition. A parameterized problem L is (*uniformly*) *fixed-parameter tractable* if there exists a constant α and an algorithm to determine if (x, y) is in L in time $f(|y|) \cdot |x|^\alpha$, where $f : N \to N$ is an arbitrary function. We will denote the class of fixed-parameter tractable problems by FPT.

In a previous paper we defined a hierarchy of parameterized problem classes

$$FPT \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \cdots \subseteq W[SAT] \subseteq W[P]$$

and exhibited problems complete for $W[t]$ for $t \geq 2$. For example, DOMINATING SET is complete for $W[2]$. Our main result in the present paper shows that several natural problems (including INDEPENDENT SET) are complete or hard for $W[1]$. We remark that $W[1]$ is currently the most important of the parameterized classes that we beleive to be intractable. This is because it is our current "minimally intractable" class in the sense that we believe it to be intractable and if we wish to prove a problem to be fixed parameter intractable we will establish this by showing hardness for $W[1]$. The reasons that we believe that $W[1]$ is fixed parameter intractable are that many problems have been shown to be $W[1]$ complete here and elsewhere (e.g. [CCDF]) and the following generic problem is $W[1]$ complete (in [CCDF]):

SHORT TURING MACHINE COMPUTATION

Input: A Nondeterministic Turing Machine M and a string x.
Parameter: k
Question: Does M have a computation path that accepts x in at most k steps?

We believe that the $W[1]$ completeness of this problem establishes a miniaturized Cook-Levin theorem and provides very strong evidence that $W[1]$ really is fixed parameter intractable.

For a parameterized problem L and $y \in \Sigma^*$ we write L_y to denote the associated fixed-parameter problem (y is the parameter) $L_y = \{x | (x, y) \in L\}$.

Definition. A (*uniform*) *reduction* of a parameterized problem L to a parameterized problem L' is an oracle algorithm A that on input (x, y) determines whether $x \in L_y$ and satisfies
(1) There is an arbitrary function $f : N \to N$ and a polynomial q such that the running time of A is bounded by $f(|y|)q(|x|)$.
(2) For each $y \in \Sigma^*$ there is a finite subset $J_y \subseteq \Sigma^*$ such that A consults oracles only for fixed-parameter decision problems L'_w where $w \in J_y$.

If, additionally the functions f and $y \to J_y$ are both recursive we say that the reduction is *strongly* uniform. (All of the reductions in this paper are strongly uniform.)

A motivating example for the above definition is the reduction of the Graph Genus problem to the problem of MINOR TESTING. By the deep results of Robertson and Seymour [RS1,RS2] the GRAPH GENUS problem for each fixed parameter value k reduces to finitely many minor tests; the reduction can be made uniform by the techniques of [FL1,FL2]. The following is easily verified.

Lemma 1.1 If the parameterized problem L reduces to the parameterized problem L', and if L' is f.p. tractable, then L is f.p. tractable. □

In the Section 2 we review the definition of the W hierarchy. In Section 3 we prove our main result, that INDEPENDENT SET is complete for $W[1]$. In Section 4 we discuss a number of natural problems that are hard for $W[1]$, including a parameterized variant of SUBSET SUM. Section 5 concludes with a discussion of open problems. We remark that the results of this paper have been used in many $W[1]$ hardness proofs, as well as applied to Computational Learning Theory([DEF]). Moreover, as we montioned earlier, since the writing of the present paper, many other $W[1]$ hardness and completeness results have been found. We make some further remarks towards this in an Addendum in §6.

2. The W Hierarchy of Parameterized Problems

The complexity classes $W[t]$ of parameterized problems intuitively reflect the difficulty of checking a solution. 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.

(3) *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), not counting *not* gates, on an input-output path in C. The *weft* 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 weft* if there is constant t such that every circuit in the family F has weft at most t. 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 $L_F = \{(C, k) : C \in F$ and C accepts an input vector of weight $k\}$.

Definition. A parameterized problem L belongs to $W[t]$ (*monotone* $W[t]$) if L uniformly reduces to the parameterized circuit problem $L_{F(t,h)}$ for the family $F(t, h)$ of mixed type (monotone) decision circuits of weft at most t, and depth at most h, for some constant h.

Thus we have the containments

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

and we conjecture that each of these containments is proper. We term the union of these classes the *W hierarchy.* If we place no bound on the depth or weft of the circuits we similarly get the class W.

By definition, a parameterized problem $L \in W[1]$ reduces to $L_{F(1,h)}$ for the family $F(1, h)$ of weft 1 circuits of depth bounded by h, for some h. The following argument shows that we may assume the circuits in F to have depth 2 and a particularly simple form, consisting of a single output *And* gate which receives arguments from *or* gates having fan-in bounded by a constant h'. Thus each such circuit is isomorphically represented by a boolean expression in conjunctive normal form having clauses with at most h' literals. We will say that a family of circuits having this form is *normalized.* With this in mind we have the following definition.

Definition. The family of parameterized problems $W[1, s]$ is defined to be those parameterized problems in $W[1]$ reducible to $L_{F(s)}$ for the family $F(s)$ of depth 2, weft 1 normalized circuits, with the *or* gates on level 1 having fan-in bounded by s.

Lemma 2.1. Let F be a family of weft 1 circuits of depth bounded by a constant h. Then L_F is reducible to $L_{F(s)}$ for $s = 2^h + 1$, and hence $L_F \in W[1, s]$.

Proof. Let $C \in F$ and let k be a positive integer. We describe how to produce a circuit $C' \in F(s)$ and an integer k' such that C accepts a weight k input if and only if C' accepts an input of weight k'.

Step 1. The reduction to tree circuits.

The first step is to transform C into an equivalent weft 1 *tree circuit* C' of depth at most h. In a tree circuit every logic gate has fan-out one, and thus the circuit can be viewed as equivalent to a Boolean formula. The transformation can be accomplished by replicating the portion of the circuit above a gate as many times as the fan-out of the gate, beginning with the top level of logic gates, and proceeding downward level by level. The creation of C' from C may require time $O(|C|^{O(h)})$ and involve a similar blow-up in the size of the circuit.

This is permitted since h is a fixed constant independent of k and $|C|$.

Step 2. Moving the *not* gates to the top of the circuit.

Let C denote the circuit we receive from the previous step (we will use this notational convention throughout the proof). Transform C into an equivalent circuit C' by commuting the *not* gates to the top, using DeMorgan's laws. This may increase the size of the circuit by at most a constant factor. The tree circuit C' thus consists (from the top) of the input nodes, with *not* gates on some of the lines fanning out from the inputs. In counting levels we consider all of this as level 0.

Step 3. A preliminary depth 4 normalization.

The goal of this step is to produce a tree circuit C' of depth 4 that corresponds to a Boolean expression E in the following form. (For convenience we use product notation to denote logical $\wedge$ and sum notation to denote logical $\vee$.)

$$E = \prod_{i=1}^{m} \sum_{j=1}^{m_i} E_{ij}$$

where:
(1) m is bounded by a function of h
(2) for all i, m_i is bounded by a function of h
(3) for all i, j, E_{ij} is either:

$$E_{ij} = \prod_{k=1}^{m_{ij}} \sum_{l=1}^{m_{ijk}} x[i,j,k,l]$$

or

$$E_{ij} = \sum_{k=1}^{m_{ij}} \prod_{l=1}^{m_{ijk}} x[i,j,k,l]$$

where the $x[i,j,k,l]$ are literals (i.e., input Boolean variables or their negations) and for all i, j, k, m_{ijk} is bounded by a function of h. The family of circuits corresponding to these expressions has weft 1, with the large gates corresponding to the E_{ij}. (In particular, the m_{ij} are *not* bounded by a function of h.)

To achieve this form, let g denote a large gate in C. An input to g is computed by a subcircuit of depth bounded by h consisting only of small gates, and so is a function of at most 2^h literals. This subcircuit can thus be replaced, at constant cost, by either a product-of-sums expression (if g is a large $\wedge$ gate), or a sum-of-products expression (if g is a large $\vee$ gate). In the first case, the product of these replacements over all inputs to g yields the subexpression E_{ij} corresponding to g. In the second case, the sum of these replacements yields the corresponding E_{ij}.

The output of C is a function of the outputs of at most 2^h large gates. This function can be expressed as a product-of-sums expression of size at most 2^{2^h}. At the cost of possibly duplicating some of the large gate subcircuits at most 2^{2^h} times, we can achieve the desired normal form with the bounds: $m \leq 2^{2^h}$, $m_i \leq 2^h$ and $m_{ijk} \leq 2^h$.

Step 4. Employing additional nondeterminism.

Let C denote the normalized depth 4 circuit received from Step 3 and corresponding to the Boolean expression E described above. For convenience, assume that the E_{ij} for $j = 1, ..., m'_i$ are sums-of-products and the E_{ij} for $j = m'_i + 1, ..., m_i$ are products-of-sums. Let $V_0 = \{x_1, ..., x_n\}$ denote the variables of E.

In this step we produce an expression E' in product-of-sums form with the size of the sums bounded by $2^h + 1$ that has a satisfying truth assignment of weight

$$k' = 2k + k(1 + 2^h)2^{2^h} + m + \sum_{i=1}^{m} m'_i$$

if and only if C has a satisfying truth assignment of weight k. The main idea is to use additional (bounded weight) nondeterminism to guess both: (1) a weight k input x for C, and (2) additional information that will allow us to check that $C(x) = 1$ with a $W[1, s]$ circuit, $s = 2^h + 1$.

The set V of variables of E' is $V = V_0 \cup V_1 \cup V_2 \cup V_3$ where

$$V_1 = \{x[i, j] :\ 1 \leq i \leq n,\ 0 \leq j \leq (1 + 2^h)2^{2^h}\}$$

$$V_2 = \{u[i, j] :\ 1 \leq i \leq m,\ 1 \leq j \leq m_i\}$$

$$V_3 = \{w[i, j, k] :\ 1 \leq i \leq m,\ 1 \leq j \leq m'_i,\ 0 \leq k \leq m_{ij}\}$$

The expression E' is a product of subexpressions $E' = E_1 \wedge \cdots \wedge E_8$ described:

$$E_1 = \prod_{i=1}^{n} (\neg x_i + x[i, 0])(\neg x[i, 0] + x_i)$$

$$E_2 = \prod_{i=1}^{n} \prod_{j=0}^{2^{2^h+1}} (\neg x[i, j] + x[i, j + 1 \pmod{r}]) \qquad r = 1 + (1 + 2^h)2^{2^h}$$

$$E_3 = \prod_{i=1}^{m} \sum_{j=1}^{m_i} u[i, j]$$

$$E_4 = \prod_{i=1}^{m} \prod_{j=1}^{m_i - 1} \prod_{j'=j+1}^{m_i} (\neg u[i, j] + \neg u[i, j'])$$

$$E_5 = \prod_{i=1}^{m} \prod_{j=1}^{m'_i} \prod_{k=0}^{m_{ij}-1} \prod_{k'=k+1}^{m_{ij}} (\neg w[i,j,k] + \neg w[i,j,k'])$$

$$E_6 = \prod_{i=1}^{m} \prod_{j=1}^{m'_i} (\neg u[i,j] + \neg w[i,j,0])$$

$$E_7 = \prod_{i=1}^{m} \prod_{j=1}^{m'_i} \prod_{k=1}^{m_{ij}} \prod_{l=1}^{m_{ijk}} (\neg w[i,j,k] + x[i,j,k,l])$$

$$E_8 = \prod_{i=1}^{m} \prod_{j=m'_i+1}^{m_i} \prod_{k=1}^{m_{ij}} \left(\neg u[i,j] + \sum_{l=1}^{m_{ijk}} x[i,j,k,l] \right)$$

To see that Step 4 works correctly, suppose τ is a weight k truth assignment to V_0 that satisfies E. We describe how to extend τ to weight k' truth assignment τ' to the variables V that satisfies E' as follows:
(1) For each i such that $\tau(x_i) = 1$ and for $j = 0, ..., (1+2^h)2^{2^h}$ set $\tau'(x[i,j]) = 1$.
(2) For each $i = 1, ..., m$ choose choose an index j_i such that E_{i,j_i} evaluates to 1 (this is possible, since τ satisfies E) and set $\tau'(u[i,j_i]) = 1$.
(3) If in (2) E_{i,j_i} is a sum-of-products, then choose an index k_i such that

$$\prod_{l=1}^{m_{i,j,k_i}} x[i,j,k,l]$$

evaluates to 1, and correspondingly set $\tau'(w[i,j,k_i]) = 1$.
(4) For $i = 1, ..., m$ and $j = 1, ..., m'_i$ such that $j \neq j_i$, set $\tau'(w[i,j,0]) = 1$.

It is straightforward to check that the above described weight k' extension τ' satisfies E'.

Conversely, suppose υ' is a weight k' truth assignment to the variables of V that satisfies E'. We argue that the restriction υ of υ' to V_0 is a weight k truth assignment that satisfies E.

Claim 1. υ sets at most k variables of V_0 to 1.

If this were not so, then the clauses in E_1 and E_2 would together force at least $(k+1)(2+(1+2^h)2^{2^h})$ variables to be 1 in order for υ' to satisfy E', a contradiction as this is more than k'.

Claim 2. υ sets at least k variables of V_0 to 1.

The clauses of E_4 insure that υ' sets at most m variables of V_2 to 1. The clauses of E_5 insure that υ' sets at most $\sum_{i=1}^{m} m'_i$ variables of V_3 to 1. If Claim 2 were false then for υ' to have weight k' there must be more than k indices j for which some variable $x[i,j]$ of V_1 has the value 1, a contradiction in consideration of E_1 and E_2.

The clauses of E_3 and the arguments above show that v' necessarily has the following restricited form:
(1) Exactly k variables of V_0 are set to 1.
(2) For each of the k in (1) the corresponding $(1+2^h)2^{2^h}+1$ variables of V_1 are set to 1. (3) For each $i = 1, ..., m$ there is exactly one j_i for which $u[i, j_i] \in V_2$ is set to 1.
(4) For each $i = 1, ..., m$ and $j = 1, ..., m'_i$ there is exactly one k_i for which $w[i, j, k_i] \in V_3$ is set to 1.

To argue that v satisfies E it suffices to argue that v satisfies every E_{i,j_i} for $i = 1, ..., m$.

The clauses of E_6 insure that if $v'(u[i,j]) = 1$ then $k_i \neq 0$. This being the case, the clauses of E_7 force the literals in a subexpression of E_{i,j_i} to evaluate in a way that shows E_{i,j_i} to evaluate to 1. The clauses of E_8 enforce that E_{i,j_i} evaluates to 1 for $j_i > m'_i$. □

Thus we have the following stratification of $W[1]$ that will be useful to our arguments.

$$W[1] = \bigcup_{s=1}^{\infty} W[1, s]$$

Our main result shows that $W[1]$ collapses to $W[1, 2]$.

3. Antimonotonicity

A family of circuits F is termed *monotone* if the circuits in F do not have any *not* gates. Equivalently, the circuits in F correspond to boolean expressions having only positive literals. If we restrict the definition of $W[t]$ and $W[1, s]$ to monotone circuit families we obtain the family of parameterized problems *monotone* $W[t]$ (*monotone* $W[1, s]$).

We say that a family of circuits F is *antimonotone* if the boolean expressions corresponding to the circuits in F have only negative literals. In an antimonotone circuit each fan-out line from an input node goes to a *not* gate (and in the remainder of the circuit there are no other *not* gates). The restriction to antimonotone circuit families yields the classes of parameterized problems *antimonotone* $W[t]$ (*antimonotone* $W[1, s]$).

Theorem 3.1 of [DF2] employed as a change-of-variables as in the proof of theorem 4.1 of that paper shows the following relationship.

Proposition 3.1. $W[t] =$ monotone $W[t]$ for t even and $t \geq 2$. □

We prove the following complementary result.

Proposition 3.2. $W[t] =$ antimonotone $W[t]$ for t odd, $t \geq 1$.

We first prove the following lemma.

Lemma 3.1 $W[1,s] =$ antimonotone $W[1,s]$ for all $s \geq 2$.

Proof. The plan of our argument is to identify a problem (RED/BLUE NONBLOCKER) that belongs to antimonotone $W[1,s]$, and then show that the problem is hard for $W[1,s]$. RED/BLUE NONBLOCKER is the parameterized problem which takes as input a graph $G = (V, E)$ where V is partitioned into two color classes $V = V_{\text{red}} \cup V_{\text{blue}}$, and a positive integer k. The problem is to determine if there is a set of red vertices $V' \subseteq V_{\text{red}}$ of cardinality k such that every blue vertex has at least one neighbor that does not belong to V'.

The *closed neighborhood* of a vertex $u \in V$ is the set of vertices $N[u] = \{x : x \in V \text{ and } x = u \text{ or } xu \in E\}$.

It is easy to see that the restriction of RED/BLUE NONBLOCKER to graphs G of maximum degree s belongs to antimonotone $W[1,s]$ since the product-of-sums boolean expression

$$\prod_{u \in V_{\text{blue}}} \sum_{x_i \in N[u] \cap V_{\text{red}}} \neg x_i$$

has a weight k truth assignment if and only if G has size k nonblocking set. By the *weight* of a truth assignment to a set of boolean variables, we mean the number of variables assigned the value *true*.

Such an expression corresponds directly to a circuit meeting the defining conditions for antimonotone $W[1,s]$. We will refer to the restriction of RED/BLUE NONBLOCKER to graphs of maximum degree bounded by s as s-RED/BLUE NONBLOCKER. We next argue that s-RED/BLUE NONBLOCKER is complete for $W[1,s]$.

Let X be a boolean expression in conjunctive normal form with clauses of size bounded by s. Suppose X consists of m clauses $C_1, ..., C_m$ over the set of n variables $x_0, ..., x_{n-1}$. We show how to produce in polynomial-time by local replacement, a graph $G = (V_{\text{red}}, V_{\text{blue}}, E)$ that has a nonblocking set of size $2k$ if and only if X is satisfied by a truth assignment of weight k.

Before we give any details, we give a brief overview of the construction, whose component design is outlined in Diagram 1. There are $2k$ 'red' components arranged in a circle. These are alternatively gouped as blocks from V_1 and then V_2 sets to be precisely described below. The idea is that V_1 blocks should represent a positive choice (corresponding to a literal being true) and the V_2 blocks corresponding to the 'gap' until the next positive choice. We will ensure that for each pair in a block there will be a blue vertex connected to the pair and nowhere else (these are the sets V_3 and V_5). This device ensures that at most one red vertex from each block can be chosen and since we must choose $2k$ this ensures that we choose *exactly* one red vertex from each block. The reader should think of the V_2 blocks as arranged in k columns. Now if i is chosen fron a V_1 block we will ensure that column i gets to select the next gap. To ensure this we connect a blue degree 2 vertex to i and each vertex not in the i-th column of the next V_2 block. Of course this means that if i is chosen since these blue vertices must have an unchosen red neighbour, we must choose from the i-th column. The final part of the component design is to enforce consistency in the next V_1 block. That is if we choose i and have a gap choice in the next V_2 block, column i, of j then the next chosen variable should be $i + j + 1$ (here we work mod n). Again we can enforce this by using many degree 2 blue vertices to block any other choice (These are the V_6 vertices.) The last part of the construction is to force consistency with the clauses. We do this as follows. For each way a nonblocking set can correspond to making a clause false, we make a blue vertex and join it up to the s relevant vertices. This ensures that thay can't *all* be chosen. (This is the point of the V_7 vertices.) We now turn to the formal details.

The red vertex set V_{red} of G is the union of the following sets of vertices:
$V_1 = \{a[r_1, r_2] : 0 \le r_1 \le k-1, 0 \le r_2 \le n-1\}$
$V_2 = \{b[r_1, r_2, r_3] : 0 \le r_1 \le k-1, 0 \le r_2 \le n-1, 1 \le r_3 \le n-k+1\}$

The blue vertex set V_{blue} of G is the union of the following sets of vertices:
$V_3 = \{c[r_1, r_2, r_2'] : 0 \le r_1 \le k-1, 0 \le r_2 < r_2' \le n-1\}$
$V_4 = \{d[r_1, r_2, r_2', r_3, r_3'] : 0 \le r_1 \le k-1, 0 \le r_2, r_2' \le n-1, 0 \le r_3, r_3' \le n-1$ and either $r_2 \ne r_2'$ or $r_3 \ne r_3'\}$
$V_5 = \{e[r_1, r_2, r_2', r_3] : 0 \le r_1 \le k-1, 0 \le r_2, r_2' \le n-1, r_2 \ne r_2', 1 \le r_3 \le n-k+1\}$
$V_6 = \{f[r_1, r_1', r_2, r_3] : 0 \le r_1, r_1' \le k-1, 0 \le r_2 \le n-1, 1 \le r_3 \le n-k+1, r_1' \ne r_2 + r_3 \bmod n\}$
$V_7 = \{g[j, j'] : 1 \le j \le m, 1 \le j' \le m_j\}$

In the desription of V_7, the integers m_j are bounded by a polynomial in n and k of degree a function of s which will be described below. Note that since s is a fixed constant independent of k, this is allowed by our definition of reduction for parameterized problems.

For convenience we distinguish the following sets of vertices.
$A(r_1) = \{a[r_1, r_2] : 0 \le r_2 \le n-1\}$
$B(r_1) = \{b[r_1, r_2, r_3] : 0 \le r_2 \le n-1, 1 \le r_3 \le n-k+1\}$
$B(r_1, r_2) = \{b[r_1, r_2, r_3] : 1 \le r_3 \le n-k+1\}$

The edge set E of G is the union of the following sets of edges. In these descriptions we implicitly quantify over all possible indices for the vertex sets $V_1, ..., V_7$.
$E_1 = \{a[r_1, q]c[r_1, r_2, r_2'] : q = r_2$ or $q = r_2'\}$
$E_2 = \{b[r_1, q_2, q_3]d[r_1, r_2, r_2', r_3, r_3'] :$ either $(q_2 = r_2$ and $q_3 = r_3)$ or $(q_2 = r_2'$ and $q_3 = r_3')\}$
$E_3 = \{a[r_1, r_2]e[r_1, r_2, q, q']\}$
$E_4 = \{b[r_1, q, q']e[r_1, r_2, q, q']\}$
$E_5 = \{b[r_1, r_2, r_3]f[r_1, r_1', r_2, r_3]\}$
$E_6 = \{a[r_1 + 1 \bmod n, r_1']f[r_1, r_1', r_2, r_3]\}$

We say that a red vertex $a[r_1, r_2]$ *represents the possibility* that the boolean variable x_{r_2} may evaluate to *true* (corresponding to the possibility that $a[r_1, r_2]$ may belong to a $2k$-element nonblocking set V' in G). Similarly, we say that a red vertex $b[r_1, r_2, r_3]$ *represents the possibility* that the boolean variables $x_{r_2+1}, ..., x_{r_2+r_3-1}$ (with indices reduced mod n) may evaluate to *false*.

Suppose C is a clause of X having s literals. There are $O(n^{2s})$ distinct ways of choosing, for each literal $l \in C$, a single vertex representative of the possibility that $l = x_i$ may evaluate to *false*, in the case that l is a positive literal, or in the case that l is a negative literal $l = \neg x_i$, a representative of the possibility that x_i may evaluate to *true*. For each clause C_j of X, $j = 1, ..., m$, let $R(j, 1), R(j, 2), ..., R(j, m_j)$ be an enumeration of the distinct possibilities for such a set of representatives. We have the additional sets of edges for the clause components of G:
$E_7 = \{a[r_1, r_2]g[j, j'] : a[r_1, r_2] \in R(j, j')\}$
$E_8 = \{b[r_1, r_2, r_3]g[j, j'] : b[r_1, r_2, r_3] \in R(j, j')\}$

Suppose X has a satisfying truth assigment τ of weight k, with variables $x_{i_0}, x_{i_1}, ..., x_{i_{k-1}}$ assigned the value *true*. Suppose $i_0 < i_2 < ... < i_{k-1}$. Let $d_r = i_{r+1(\mathrm{mod} k)} - i_r \pmod n$ for $r = 0, ..., k-1$. It is straightforward to verify that the set of $2k$ vertices

$$N = \{a[r, i_r] : 0 \leq r \leq k-1\} \cup \{b[r, i_r, d_r] : 0 \leq r \leq k-1\}$$

is a nonblocking set in G.

Conversely, suppose N is a $2k$-element nonblocking set in G. It is straightforward to check that a truth assignment for X of weight k is described by setting those variables *true* for which a vertex representative of this possibility belongs to N, and by setting all other variables to *false*.

Note that the edges of the sets E_1 (E_2) which connect pairs of distinct vertices of $A(r_1)$ ($B(r_1)$) to blue vertices of degree two, enforce that any $2k$-element nonblocking set must contain exactly one vertex from each of the sets $A(0), B(0), A(1), B(1), ..., A(k-1), B(k-1)$. The edges of E_3 and E_4 enforce (again by connections to blue vertices of degree two) that if a representative of the possibility that x_i evaluates to *true* is selected for a nonblocking set from $A(r_1)$, then a vertex in the i^{th} row of $B(r_1)$ must be selected as well, representing (consistently) the interval of variables set false (by increasing index mod n) until the "next" variable selected to be *true*. The edges of E_5 and E_6 insure consistency between the selection in $A(r_1)$ and the selection in $A(r_1 + 1 \bmod n)$. The edges of E_7 and E_8 insure that a consistent selection can be nonblocking if and only if it does not happen that there is a set of representatives for a clause witnessing that every literal in the clause evaluates to *false*. (There is a blue vertex for every such possible set of representatives.) □

Proof of Prop. 3.2. Let C be a circuit of weft t for t odd, $t \geq 3$. By Theorem 4.1 of [DF2] we may assume that C is represented by a boolean expression E_0 that is in (alternating) product-of-sums-of-products... form (for t alternations). The first level of the circuit below the inputs consists of *And* gates (since t is odd).

Suppose the inputs to C are $x_1, ..., x_n$. Let X_1 be the boolean expression with single-literal clauses $X_1 = (x_1)(x_2) \cdots (x_n)$ and let G be the graph constructed from X_1 by the reduction in the lemma above. Let $y_1, ..., y_z$ be new variables, one for each red vertex in G.

Let E_1 be the boolean expression

$$E_1 = \prod_{u \in (V_{\text{blue}} - V_7)} \sum_{y_i \in N[u]} \neg y_i$$

and let C_1 be a circuit realizing E_1.

We modify C in the following ways:
(1) Each positive fan-out from an input x_i to C is replaced by an *And* gate receiving negated inputs from all of the new input variables y_j for which the corresponding red vertices of G represent the possibility that x_i evaluates to *false*.
(2) Each negated fan-out from an input x_i to C is replaced by an *And* gate receiving negated inputs from all of the new input variables y_j for which the corresponding red vertices of G represent the possibility that x_i evaluates to *true*.
(3) The circuit C_1 is conjunctively combined with C at the bottommost (output) *And* gate.

The circuit C' obtained in this way accepts a weight $2k$ input vector if and only if C accepts a weight k input vector. The argument for correctness is essentially the same as for Lemma 3.1. The circuit C' has weft t after the *And* gates replacing the former inputs are coalesced with the *And* gates of the topmost large gate level (this is feasible, since t is odd). All of the input fan-out lines of C' are negated. □

Lemma 3.1 provides the starting point for demonstrating the following collapse of the $W[1]$ stratification.

Proposition 3.3. $W[1] = W[1, 2]$

Proof. It suffices to argue that for all $s \geq 2$, antimonotone $W[1, s] = W[1, 2]$. The argument here consists of another change of variables. Let C be an antimonotone $W[1, s]$ circuit for which we wish to determine whether a weight k input vector is accepted. We show how to produce a circuit C' corresponding to an expression in conjunctive normal form with clause size two, that accepts an input vector of weight

$$k' = k2^k + \sum_{i=2}^{s} \binom{k}{i}$$

if and only if C accepts an input vector of weight k. (The circuit C' will in general not be antimonotone, but this is immaterial by Lemma 3.1. Actually in [DEF] we use another reduction that only needs $k' = k^{s+1} + \sum_{i=2}^{s} \binom{k}{i}$ and is hence polynomial in k for a fixed s.)

Let $x[j]$ for $j = 1, ...n$ be the input variables to C. The idea is to create new variables representing all possible sets of at most s and at least 2 of the variables $x[j]$. Let $A_1, ..., A_p$ be an enumeration of all such subsets of the input variables $x[j]$ to C. The inputs to each *or* gate g in C (all negated, since C is antimonotone) are precisely the elements of some A_i. The new input corresponding to A_i represents that all of the variables whose negations are inputs to the gate g have the value *true*. Thus in the construction of C', the *or* gate g is replaced by the negation of the corresponding new "collective" input variable.

We introduce new input variables of the following kinds:
(1) One new input variable $v[i]$ for each set A_i for $i = 1, ..., p$, to be used as above.
(2) For each $x[j]$ we introduce 2^k copies $x[j, 0], x[j, 1], x[j, 2], ..., x[j, 2^k - 1]$.

In addition to replacing the *or* gates of C as described above, we add to the circuit additional *or* gates of fan-in 2 that provide an enforcement mechanism for the change of variables. The necessary requirements can be easily expressed in conjunctive normal form with clause size two, and thus can be incorporated into a $W[1, 2]$ circuit.

We require the following implications concerning the new variables:

(1) The $n \cdot 2^k$ implications, for $j = 1, ..., n$ and $r = 0, ..., 2^k - 1$,

$$x[j, r] \Rightarrow x[j, r + 1 \pmod{2^k}]$$

(2) For each containment $A_i \subseteq A_{i'}$, the implication

$$v[i'] \Rightarrow v[i]$$

(3) For each membership $x[j] \in A_i$, the implication

$$v[i] \Rightarrow x[j, 0]$$

It may be seen that this transformation may increase the size of the circuit by a linear factor exponential in k. We make the following argument for the correctness of the transformation.

If C accepts a weight k input vector, then setting the corresponding copies $x[i, j]$ among the new input variables accordingly, together with appropriate settings for the the new "collective" variables $v[i]$ yields a vector of weight k' that is accepted by C'.

For the other direction, suppose C' accepts a vector of weight k'. Because of the implications in (1) above, exactly k sets of copies of inputs to C must have value 1 in the accepted input vector (since there are 2^k copies in each set). Because of the implications described in (2) and (3) above, the variables $v[i]$ must have values in the accepted input vector compatible with the values of the sets of copies. By the construction of C', this implies there is a weight k input vector accepted by C. □

We have now done most of the work required to show that the following well- known problems are complete for $W[1]$.

INDEPENDENT SET
Instance: A graph $G = (V, E)$ and a positive integer k.
Question: Is there a set $V' \subseteq V$ of cardinality k, such that $\forall u, v \in V'$, $uv \notin E$?

CLIQUE
Instance: A graph $G = (V, E)$ and a positive integer k.
Question: Is there a set of k vertices $V' \subseteq V$ that forms a complete subgraph of G (that is, a clique of size k)?

Theorem 3.1. INDEPENDENT SET is complete for $W[1]$.

Proof. It is easy to observe that INDEPENDENT SET belongs to $W[1]$. By Lemma 3.1 and Theorem 3.1 it is enough to argue that INDEPENDENT SET is hard for antimonotone $W[1, 2]$. Given a boolean expression X in conjuctive normal form (product of sums) with clause size two and all literals negated, we may form a graph G_X with one vertex for each variable of X, and having an edge between each pair of vertices corresponding to variables in a clause. The graph G_X has an independent set of size k if and only if X has a weight k truth assignment. $\square$

Corollary 3.2 CLIQUE is complete for $W[1]$.

Proof. This follows immediately by considering the complement of a given graph. The complement has an independent set of size k if and only if the graph has a clique of size k. $\square$

4. Problems Hard for $W[1]$

In this section we show that the following problems are hard for $W[1]$. None of them is presently known to belong to $W[1]$. We conjecture that the first two problems, which are shown to be equivalent with respect to uniform reductions, and to belong to $W[2]$, are of difficulty intermediate between $W[1]$ and $W[2]$.

PERFECT CODE
Instance: A graph $G = (V, E)$ and a positive integer k.
Question: Does G have a k-element perfect code? A *perfect code* is a set of vertices $V' \subseteq V$ with the property that for each vertex $v \in V$ there is precisely one vertex in $N[v] \cap V'$.

WEIGHTED EXACT CNF SATISFIABILITY
Instance: A boolean expression E in conjuctive normal form, and a positive integer k.
Question: Is there a truth assignment of weight k to the variables of E that makes exactly

one literal in each clause of E *true*?

SIZED SUBSET SUM
Instance: A list of positive integers $L = (x_1, x_2, ..., x_n)$, a positive integer S and a positive integer k.
Question: Is there a sublist of L of size k that sums to S?

Lemma 4.1. PERFECT CODE $\in W[2]$.

Proof. Let $G = (V, E)$ be a graph for which we wish to determine whether G has a k-element perfect code. It suffices to show how to efficiently construct a boolean expression E_G in product-of-sums form that has a weight k truth assignment if and only if the graph G has a k-element perfect code. Let E_G be the expression $E = E_0 E_1 E_2$ where the variables of E_G are in one-to-one correspondence with vertices of G and

$$E_0 = \prod_{u \in V} \sum_{x \in N[u]}$$

$$E_1 = \prod_{uv \in E} (\neg u + \neg v)$$

$$E_2 = \prod_{uv, vw \in E} (\neg u + \neg w)$$

If G has a k-element perfect code $V' \subseteq V$, then the truth assignment which sets the variables corresponding to the vertices of V' *true* and all others *false* satisfies E_G, since V' is an independent set (so that E_1 is satisfied), and V' contains no vertices at a distance 2 from each other in G (so that E_2 is satisfied), and yet V' is dominating set (so that E_2 is satisfied). Conversely, any satisfying truth assignment for E_G of weight k must satisfy each of of these subproducts, and therefore the vertices corresponding to the variables set to *true* must be a perfect code in G. □

Lemma 4.2. PERFECT CODE reduces to WEIGHTED EXACT CNF SATISFIABILITY.

Proof. A graph G has a k-element perfect code if and only if the expression E_0 constructed as in Lemma 5.1 has a weight k truth assignment that makes exactly one literal in each clause *true*. □

Lemma 4.3. WEIGHTED EXACT CNF SATISFIABILITY reduces to PERFECT CODE.

Proof. The reduction can be demonstrated using the transformation used in the proof of Theorem 3.1 of [DF1] (which is there used to reduce Weighted Satisfiability to DOMINATING SET). □

Lemma 4.4. PERFECT CODE reduces to SIZED SUBSET SUM.

Proof. Let $G = (V, E)$ be a graph for which we wish to determine whether G has a perfect code of size k. Suppose for convenience that the vertex set of the graph $V = \{0, ..., n-1\}$. We can easily compute the list of positive integers $L = (x[i,j] : 1 \leq i \leq k,\ 0 \leq j \leq n-1)$ and the positive integer M, where

$$x[i,j] = (k+1)^{n+k-i} + \sum_{u \in N[j]} (k+1)^u$$

$$M = \sum_{t=0}^{n+k-1} (k+1)^t$$

such that L has a sublist of size k summing to M if and only if G has a k-element perfect code. The correctness of this transformation is easily observed if the numbers of L are represented in base $k+1$, and it is noted that there can be no carries in a sum of k integers from L expressed in this way. $\square$

Theorem 4.1. PERFECT CODE is hard for $W[1]$.

Proof. We reduce from INDEPENDENT SET. Let $G = (V, E)$ be a graph. We show how to produce a graph $H = (V', E')$ that has a perfect code of size $k' = \binom{k}{2} + k + 1$ if and only if G has a k-element independent set. The vertex set V' of H is the union of the sets of vertices:

$V_1 = \{a[s] : 0 \leq s \leq 2\}$
$V_2 = \{b[i] : 1 \leq i \leq k\}$
$V_3 = \{c[i] : 1 \leq i \leq k\}$
$V_4 = \{d[i,u] : 1 \leq i \leq k, u \in V\}$
$V_5 = \{e[i,j,u] : 1 \leq i < j \leq k, u \in V\}$
$V_6 = \{f[i,j,u,v] : 1 \leq i < j \leq k,\ u, v \in V\}$

The edge set E' of H is the union of the sets of edges:

$E_1 = \{a[0]a[i] : i = 1, 2\}$
$E_2 = \{a[0]b[i] : 1 \leq i \leq k\}$
$E_3 = \{b[i]c[i] : 1 \leq i \leq k\}$
$E_4 = \{c[i]d[i,u] : 1 \leq i \leq k, u \in V\}$
$E_5 = \{d[i,u]d[i,v] : 1 \leq i \leq k,\ u, v \in V\}$
$E_6 = \{d[i,u]e[i,j,u] : 1 \leq i < j \leq k, u \in V\}$
$E_7 = \{d[j,v]e[i,j,u] : 1 \leq i < j \leq k, v \in N[u]\}$
$E_8 = \{e[i,j,x]f[i,j,u,v] : 1 \leq i < j \leq k, x \neq u, x \notin N[v]\}$
$E_9 = \{f[i,j,u,v]f[i,j,x,y] : 1 \leq i < j \leq k, u \neq x \text{ or } v \neq y\}$

An overview of this construction is given in Diagram 2 and a (partial) example is given in Diagram 3. Suppose C is a perfect code of size k' in H. Since $a[1]$ and $a[2]$ are pendant vertices attached to $a[0]$, neither vertex belongs to C because both cannot belong to C, and if only one belongs to C, then C fails to be a dominating set. It follows that $a[0] \in C$. This implies that none of the vertices in V_2 and V_3 belong to C (V_3 would kill V_2), and it implies also that exactly one vertex in each of the cliques formed by the edges of E_5 belongs to C (to cover V_3). Note that each of these k cliques has n vertices indexed by V, the vertex set of G (this is the *selection* gadget). Let I be the set of vertices of G corresponding to the elements of C in these cliques. We argue that I is an independent set of order k in G.

Suppose $u, v \in I$ and that $uv \in E$. Then there are indices $i < j$ between 1 and k such that (without losss of generality) $d[i,u] \in C$ and $d[j,v] \in C$. By the definition of E_6 and E_7 each of these vertices is adjacent to $e[i,j,u]$, which contradicts that C is a perfect code in H. Thus I is an independent set in G.

Conversely, we argue that if $J = \{u_1, ..., u_k\}$ is a k-element independent set in G, then H has a perfect code C_J of size k'. We may take C_J to be the following set of vertices:

$$C_J = \{a[0]\} \cup \{d[i,u_i] : 1 \leq i \leq k\} \cup \{f[i,j,u_i,u_j] : 1 \leq i,j \leq k\}$$

That C_J is a perfect code can be verified directly from the definition of H. □

By Lemmas 4.2, 4.4 and the above theorem we have the following hardness results as well.

Theorem 4.2 WEIGHTED EXACT CNF SATISFIABILITY is hard for $W[1]$. □

Theorem 4.3 SIZED SUBSET SUM is hard for $W[1]$. □

One problem that we are quite interested in is the natural analogue of Travelling Salesperson:

SHORT CHEAP TOUR
Instance: A weighted graph and positive integers S and k.
Question: Is there a tour through at least k vertices of cost at most S?

The precise difficulty of this problem is at present open but a variation is hard for $W[1]$. Let SHORT EXACT TOUR be the same as SHORT CHEAP TOUR except that we ask that the tour costs *exactly* S.

Theorem 4.4 SHORT EXACT TOUR is hard for $W[1]$.

Proof. Let (L, S, k) be an instance of sized subset sum,with $L = \{x_1, ..., x_n\}$. Construct a graph G as follows: For each x_i we have two vertices y_i and z_i . Join y_i to z_i with an edge of weight x_i. Let d exceed $x_1 + ... + x_n$. For i not equal to j join y_i to z_j. Give all such edges weight d. Now ask if G has a $2k$ vertex tour of weight $S + kd$? □

The reader should note that the natural analogue of HAMILTON CIRCUIT which asks if there is a cycle through k or more vertices is strongly uniformly fixed parameter tractable (Bodlaender), but it is unknown if the problem of determining if there is a cycle of size *exactly* k is also tractable. (See [JvL,section 2.4.3]).

As a final example, we remark that the reduction of [DF2] can be used to show that the following problem is also hard for $W[1]$.

WEIGHTED EXACT BINARY INTEGER PROGRAMMING
Instance: A binary vector $\mathbf{b}$, a binary matrix A and an integer k.
Question: Is there a binary vector $\mathbf{x}$ of weight k such that $A\mathbf{x}$ equals $\mathbf{b}$?

5. Open Problems

The study of fixed-parameter tractability and completeness can be viewed as addressing aspects of the the general subject of computational infeasibility inside of P. For related work examining limited ammounts of nondeterminism see [BG]. Many familiar issues in complexity theory have unexplored analogues in the fixed-parameter setting (such as parallel and randomized complexity, one-way functions, and approximation). A number of basic structural questions concerning the W hierarchy have yet to be resolved. For example, while it is known a collapse of the W hierarchy implies a collapse involving more familiar unparameterized complexity classes ([ADF2]), the exact relationship is unknown.

A wide variety of natural parameterized problems may well be complete for various levels of the W hierarchy. Well-known natural problems for which neither fixed-parameter tractability nor $W[t]$ hardness is presently known include: DIRECTED FEEDBACK VERTEX SET, GRAPH TOPLOGICAL CONTAINMENT and IMMERSION ORDERING (the parameters in the last two problems being a fixed Graph.) (for the definitions, see [GJ]).

6. Addendum 7 Feb 1994

Since the original writing of this paper, there has benn quite a bit of activity regarding $W[1]$ and it is clear that this is probably the most important class one can use to establish fixed parameter *in*tractability along the lines of establishing intractability via NP completeness. Particularly strong evidence for the intractability of $W[1]$ is given in Cai, Chen, Downey, and Fellows [CCDF] where it is established that the following very generic problem is $W[1]$ complete:

SHORT TURING MACHINE COMPUTATION
Input: A Nondeterministic Turing Machine M and a string x.
Parameter: k.
Question: Does M have a length k computation path accepting x?

This problem is particularly signifigant as it proves a sort of Cook's theorem in a parameterized setting. Many other problems have been shown to be $W[1]$ complete. We quote a couple. In [CCDF] it is also proven that the following are $W[1]$ complete:

SHORT DERIVATION (for unrestricted grammars)
Input: A phrase-structure grammar G and a word x.
Parameter: k
Question: Is there a G derivation of x of length k?

SHORT POST CORRESPONDENCE
Input: A Post Correspondence System Π.
Parameter: k
Question: Is there a length k solution for Π?

Downey, Fellows, Kapron, Hallett, and Wareham [DFKHW] proved that the following problem is $W[1]$ complete:

SHORT CSL DERIVATION
Input: A context sensitive grammar G and a word $x \in \Sigma^*$.
Parameter: k
Question: Is there a G derivation of x of length at most k?

Downey and Fellows proved that some parameterized versions of embedding questions turn out to be $W[1]$ complete. For instance from [DF7] we have the following being $W[1]$ complete.

SEMIGROUP EMBEDDING
Input: A semigroup G.
Parameter: A semigroup H.
Question: Is H embeddable into G?

SEMILATTICE EMBEDDING
Input: A semialttice S.
Parameter: A semialttice L.
Question: Is L embeddable into S?

BIPARTITE GRAPH EMBEDDING
Input: A bipartite graph G.
Parameter: A bipatrite graph H.
Question: Is H embeddable into G?

Another area that has found $W[1]$ complete problems is that of Computational Learning Theory. Consider the following problem which is the most important parameter in learning theory.

VAPNIK CHERVONENKIS DIMENSION
Input: A family F of subsets of a base set X.
Parameter: k.
Question: Is the VC-dimension of F at least k?

In [DEF], the authors together with P. Evans proved that VAPNIK-CHERVONENKIS DIMENSION is hard for $W[1]$, and hence combined with membership of $W[1]$ which is proven in Downey-Fellows [DF5], we see that this problem is $W[1]$ complete. We remark that this is very interesting since the unparameterized version is highly unlikely to be NP-complete unless NP is very small. See, Papadimitriou and Yannakakis [PY].

Finally we mention some problems that are $W[1]$ complete arizing from molecular biology, which is a particularly fertile area of applications for this theory in view of the fact that many problems have small parameters (such as the number of strands of DNA) yet large problem size.

LONGEST COMMON SUBSEQUENCE
Input: A set of k strings $X_1, ..., X_k$ over Σ^*.
Parameter[1]: k.
Parameter[2]: m.
Parameter[3]: m, k.
Question: Is there a string $X \in \Sigma^*$ that has at least m symbols that is a subsequence of $X_1, ..., X_k$?

In Bodlaender, Downey, Fellows, and Wareham [BDFW] it is shown that all of the variations LCS[i] (with the obvious meanings) are $W[1]$ hard, and that LCS[3] is $W[1]$ complete.

We conclude by remarking that there have beeen very many other problems which have been proven to be $W[1]$ hard and even complete for other levels of the W-hierarchy. Partial lists can be found in [DF2] and [DF4].

References

[ADF1] K. A. Abrahamson, R.G. Downey and M. F. Fellows," Fixed-Parameter Intractability II," In STACS'93 (1993) Springer-Verlag Lecture Notes in Computer Science, 374-385.

[ADF2] K. A. Abrahamson, R. G. Downey, and M. R. Fellows, "Fixed-Parameter Tractability and Completeness IV: On Completeness for $W[P]$ and $PSPACE$ Analogues", to appear, Annals of Pure and Applied Logic.

[BG] J. F. Buss and J. Goldsmith, "Nondeterminism Within P," SIAM J of Comput. **22** (1993) 560-572.

[Bo] H. L. Bodlaender, "On Disjoint Cycles," Technical Report RUU-CS-90-29, Dept. of Computer Science, Utrecht University, Utrecht, The Netherlands, August 1990.

[BDFW] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and H. Todd Wareham,"The Parameterized Complexity of Sequence Alignment and Concensus," In *Proceedings Combinatorial Pattern Matching, Asilomar, 1994,* to appear.

[CCDF] L. Cai, J. Chen, R. G. Downey and M. F. Fellows,"The Parameterized Complexity of Short Computation and Factorization," submitted.

[DEF] R. G. Downey, P. Evans, and M. F. Fellows,"Parameterized Learning Complexity,"in COLT'93 (6 th Annual Conference on Learning Theory, 1993) IEEE publ. 51-57.

[DF1] R. G. Downey and M. R. Fellows, "Fixed-Parameter Tractability and Completeness, Cong. Numer. **87** (1992) 161-187.

[DF2] R. G. Downey and M. R. Fellows, "Fixed-Parameter Tractability and Completeness I: Basic Results," to appear SIAM J. of Comput.

[DF3] R. G. Downey and M. R. Fellows, "Fixed-Parameter Tractability and Completeness III: Some Structural Aspects of the W-Hierarchy," in *Complexity Theory,* (ed. K. Ambos-Spies, S. Homer, and U. Sch"/oing), Cambridge University Press, (1993) 166-191.

[DF4] R. G. Downey and M. F. Fellows,"Fixed-Parameter Intractability," Proceedings 7th Annual Structure in Complexity (1992) 36-49.

[DF5] R. G. Downey and M. F. Fellows,"Parameterized Computational Feasibility,"in *Feasible Mathematics II,* (ed. P. Clote and J. Remmel) Birkhauser, Boston, to appear.

[DF6] R. G. Downey and M. F. Fellows,*Parameterized Complexity* Monograph in preparation.

[DF7] R. G. Downey and M. F. Fellows,"Fixed-Parameter Tractability and Intractability-a

Survey," in preparation.

[FL1] M. R. Fellows and M. A. Langston, "On Search, Decision and the Efficiency of Polynomial-Time Algorithms." In *Proc. Symp. on Theory of Computing (STOC)* (1989), 501-512.

[FL2] M. R. Fellows and M. A. Langston, "An Analogue of the Myhill-Nerode Theorem and Its Use in Computing Finite Basis Characterizations." In *Proc. Symp. Foundations of Comp. Sci. (FOCS)* (1989), 520-525.

[GJ] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979).

[PY] C. Papadimitriou and M. Yannakakis,"On limited Nondeterminism and the Complexity of the VC-Dimension,"in *Proceedings 8th Annual Structure in Complexity Conference,* IEEE Publ. (1993) 12-18.

[RS1] N. Robertson and P. D. Seymour, "Graph Minors XIII. The Disjoint Paths Problem," to appear.

[RS2] N. Robertson and P. D. Seymour, "Graph Minors XV. Wagner's Conjecture," to appear.

[JvL] J. van Leeuwen, "Graph Algorithms." in *Handbook of Theoretical Computer Science, Volume A* (ed J. van Leeuwen) Elsevier Science Publishers B.V.,The Netherlands (1990), 525-632.