Fixed-Parameter Tractability and Completeness

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 k-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. We show that for this problem, and for the problem of determining whether a graph has k disjoint cycles, we may take $c = 1$. We also show that if Dominating Set is fixed-parameter tractable, then so are a variety of parameterized problems, such as Independent Set. Some of the main results of a completeness theory for fixed-parameter intractability are surveyed.

1. Introduction

Many natural computational problems have input that consists of a pair of items. 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 embeds on the genus k surface. The problem of Minor Testing is that of determining for an input pair of graphs (G, H) whether H is a minor of G.

One of the reasons for our interest in parameterized problems, is that while many of these problems are NP-complete, it is sometimes the case that only a small range of parameter values are really important in practice, so that the NP-completeness of the general problem may be unduly pessimistic information. For many parameterized problems, we now have encouraging and perhaps useful fixed-parameter tractability results, such as the following.

Theorem 1.1 (Robertson and Seymour) [RS1] For every fixed graph H it can be determined

in time $O(n^3)$ whether a graph G of order n has a minor isomorphic to H.

Theorem 1.2 (Bienstock and Monma) [BM] For every fixed k, it can be determined in time $O(n)$ whether a graph G of order n can be embedded in the plane so that k faces cover all the vertices.

Theorem 1.3 (Bodlaender) [Bod1] For every fixed k, it can be determined in time $O(n)$ whether a graph G of order n has a spanning tree with at least k leaves.

Theorem 1.4 (Lagergren) [La] For every fixed k, it can be determined in time $O(n \log^2 n)$ whether a graph G of order n has treewidth at most u.Q?Ø Qe.w[indent *The7rem 1.5* (Plehn and Voigt) [PV] For every fixed graph H of treewidth w, it can be determined in time $O(n^{w+1})$ whether a graph G of order n has a subgraph isomorphic to H.

Theorem 1.6 (Fellows and Langston) [FL1] For every fixed k, it can be determined in time $O(n)$ whether a graph G of order n has a cycle of length at least k.

For other parameterized problems, such as Minimum Dominating Set (see [GJ] for the definition), we have the contrasting situation where essentially no better algorithm is known than the "trivial" one which just exhaustively tries all possible solutions. For each fixed k, k-Minimum Dominating Set is solvable in this way in time $O(n^{k+1})$.

We make the following definitions in order to frame these complexity issues.

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

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 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.

The difference between the known fixed-parameter complexity of Dominating Set and the problems addressed in Theorems 1.1–1.6, is analogous to the apparent complexity difference between NP-complete problems and problems in P. For most NP-complete problems we essentially know no better algorithm than the "trivial" one requiring exponential time, which tries all possible solutions.

If $P = NP$ then Dominating Set is fixed-parameter tractable. A converse to this statement is not known, and is perhaps unlikely. The reasonable conjecture that Dominating Set is not fixed-parameter tractable is thus stronger than the reasonable conjecture that $P \neq NP$.

In recent years a variety of methods useful for demonstrating fixed-parameter tractability have emerged, such as the well-quasiordering results of Robertson and Seymour [RS1,RS2], and general algorithmic methods for bounded treewidth [AF,Ar,ALS,BLW,Co,WHL].

The reader should note an important detail of the definition of fixed-parameter tractability given above. The results of Theorems 1.1–1.6 (and our Theorem 2.1 below) are clearly uniform; the proofs of these results can be implemented as a single algorithm that works for every fixed parameter value. Consider, contrastingly, the consequence of Theorem 1 and the Graph Minor Theorem [RS2] that for each fixed k, it can be determined in time $O(n^3)$ whether a graph G of order n embeds on the surface of genus k. It is not immediately clear how these (infinitely many) distinct $O(n^3)$ algorithms, each based on a different finite obstruction set, can be combined into a single finite algorithm. This can done, however, by the two different methods of [FL1] and [FL2]. Almost all of the known fixed-parameter tractability results are (or can be made) uniform. We note in passing that a nonuniform framework is also possible and interesting, although we will say no more about it in this survey.

The Graph Minor Theorem has the consequence that for each fixed surface we can decide graph embedability by employing *finitely many* minor tests. Thus the fixed-parameter tractability of Minor Testing leads to the fixed-parameter tractability of the Graph Genus problem. This may be kept in mind as a motivating example for the following definition.

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 $w \in J_y$.

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.

Proof. Let $f(|y|)q(|x|)$ be the bound on the running time of the reduction from L to L', and suppose L'_w is decidable in time $g(|w|) \cdot n^\alpha$. Let $y \in \Sigma^*$ and let $J_y \subseteq \Sigma^*$ be the associated finite subset of Σ^* for the reduction. Then we can determine if $(x, y) \in L$ in time $f(|y|)g(m)|x|^\alpha q(|x|)$ where $m = \max\{|w| : w \in J_y\}$. □

The plan of this paper is as follows. In Section 2 we illustrate some of the methods for proving fixed-parameter tractability results, showing that for every fixed k the problems k-Feedback Vertex Set and k-Disjoint Cycles can be solved in time linear in the number of vertices of an input graph. In Section 3 we show that a variety of parameterized problems reduce to Dominating Set. In Section 4 we survey the main facts presently known concerning a completeness theory for fixed-parameter tractability. Section 5 concludes with a discussion

of some of the many open problems in this subject.

2. Fixed-Parameter Tractability for Problems About Cycles in Graphs.

In this section we consider the following parameterized problems. For each of these problems, and for each fixed k the problem can be addressed by the Graph Minor Theorem, since for each fixed k the set of yes–instances forms a lower ideal. Thus our results only improve upon previous fixed-parameter tractability results available by those methods. In fact, linear-time algorithms for these problems based on graph minor theory and graph rewriting (but using $O(n^{O(k)})$ space!) can be obtained by the methods of [ACPS]. A similar approach to ours was found independently by Bodlaender [Bod2].

Feedback Vertex Set
Instance: An undirected graph $G = (V, E)$ and a positive integer k.
Question: Is there a set of V' of at most k vertices of G such that every cycle in G passes through some vertex $u \in V'$?

Disjoint Cycles
Instance: An undirected graph $G = (V, E)$ and a positive integer k.
Question: Is there a set of k vertex disjoint cycles in G?

Lemma 2.1 For all fixed k, uniformly in time $O(n)$ we can accomplish one of the following for a graph $G = (V, E)$ of order n.
(1) Find a tree decomposition of G of width at most $16k^4 + 16k^3 + 8k^2 + 1$.
(2) Find k+1 vertex disjoint cycles in G.
(3) Find a vertex u of G with the property that if $V' \subseteq V$ is any k-element feedback vertex set in G, then $u \in V'$.

Proof. We use a slightly modified form of the linear-time algorithm due to Bodlaender (Theorem 3.12 of [Bod1]) based on the depth-first search minor testing technique introduced in [FL1]. This algorithm yeilds either (1) or a minor of G isomorphic to the $2 \times (2k + 2)$ grid (which gives (2)), or a minor of G isomorphic to a circus graph of size $2k^2 + 1$. The last case immediately gives a minor of G isomorphic to a bouquet of size $k^2 + 1$. (A *bouquet* of size b consists of b loops on a single vertex.) Note that if G has any vertex v with a loop, then we may take $u = v$ to accomplish (3). Thus the subgraph H of G identified in the remaining case (which is contractible to the size k^2+1 boquet) consists of a tree T together with k^2+1 disjoint cycles, each attached to the tree at a single vertex. If more than k vertices of T have attached cycles, then we have achieved (2). Otherwise, some vertex u of T has at least $k + 1$ disjoint cycles attached. If u does not belong to V', then V' cannot be a k-element set covering all cycles in G. □

A vertex as in (3) of the lemma above we will refer to as *essential*. The next lemma gives a useful general result. To state the lemma, we review briefly some of the main results concerning the finite-state tree automata point of view on bounded treewidth graph properties. For greater detail the reader should consult the papers [ACPS,AF,Co,FL2,La].

Definition. A t-boundaried graph $G = (V, E, B, f)$ is an ordinary graph $G = (V, E)$ to gether with (1) a distinguished subset (the *boundary*) of the vertex set $B \subseteq V$ with $|B| = t$, and (2) a bijection $f : B \to \{1, ..., t\}$.

Definition. If $G = (V, E, B, f)$ and $G' = (V', E', B', f')$ are t-boundaried graphs, then $G \oplus G'$ denotes the ordinary graph obtained from the disjoint union of the graphs $G = (V, E)$ and $G' = (V', E')$ by identifying each vertex $u \in B$ with the vertex $v \in B'$ for which $f(u) = f'(v)$.

If F is an arbitrary family of (ordinary) graphs, we define the following canonical equivalence relation induced by F on the set of t-boundaried graphs.

Definition. $G \sim_F H$ if and only if, for every t-boundaried graph K, $G \oplus K \in F \iff H \oplus K \in F$.

We say that a graph family F is *t-cutset regular* if $\sim_F$ has a finite number of equivalence classes on the set of t-boundaried graphs. We say that F is *cutset regular* if it is t-cutset regular for every t. The beauty of this notion, which the reader may note is similar in spirit to the classic Myhill-Nerode theorem of formal languages, is that it supports a similar necessary and sufficient condition for finite-state recognition from structural parse trees for graphs of bounded treewidth [AF] (see also [Co]). We have the following.

Lemma 2.2 If F is t-cutset regular, then F is recognizable in time $O(n)$ for graphs of treewidth at most t given with a tree-decomposition. □

Definition. The *cycle cover number* $\gamma(G)$ of a graph G is the minimum number of vertices in a feedback vertex set for G.

Lemma 2.3 Suppose F is a cutset regular family of graphs of bounded cycle cover number. Then F is recognizable in time $O(n)$.

Proof. Let k be the bound on the cycle cover number of graphs in F. Run the algorithm of Lemma 2.1. If we discover $k + 1$ disjoint cycles then we know that $G \notin F$. If we obtain a bounded width tree decomposition, then we can finish the decision in time $O(n)$ by the cutset regularity of F and Lemma 2.2. Otherwise, we have located an essential vertex u of G. In this case we run the algorithm of Lemma 2.1 again on $G - u$. (We may repeat this up to k times.) If we ever obtain a bounded width tree decomposition, then by including the (at most k) deleted essential vertices into each set of the decomposition, we obtain a bounded width tree decomposition (with the bound increased by at most k) and can finish the decision in linear time as above. If more than k essential vertices are identified, then we

know that G is not in F. □

Theorem 2.1 For each fixed k, Feedback Vertex Set and Disjoint Cycles can be solved in time $O(n)$ where n is the number of vertices of the input graph G.

Proof. By the theorem of Erdos and Posa [EP], if $\gamma(G) \geq C \cdot k \log k$ then G has more than k disjoint cycles. Cutset regularity can be demonstrated either directly or through expressibility in second-order monadic logic [Co]. Thus the hypotheses of Lemma 2.3 are satisfied for each fixed k. □

All of the algorithms involved in Theorem 6 can be made uniform in k, so that these two problems are shown to be uniformly fixed-parameter tractable in the sense of our definition in Section 1. Lemma 2.3 has the further interesting consequence that every lower ideal in the topological order having bounded cycle cover number is recognizable in time $O(n)$, since for every k the family of graphs not containing k disjoint cycles is well-quasiordered by topological containment and topological order tests are expressible in second order monadic logic.

3. A Key Combinatorial Reduction

Neither of the well-known computational problems of (1) determining whether a graph G has a dominating set of size k (Dominating Set), and (2) determining whether a graph G has an independent set of size k (Independent Set) is know to be f.p. tractable, and it is perhaps a reasonable conjecture that they are not. The reader skeptical of this conjecture and willing to challenge it, will be advised by the results of this section to begin by working on Independent Set, since a consequence of our Theorem 7 is that Independent Set reduces to Dominating Set (and so the latter is "harder" with respect to f.p. tractability). Presently the best known results for these problems are the trivial $O(n^{k+1})$ algorithm for Dominating Set, and a nontrivial algorithm for Independent Set due to Nesetril and Poljak [NP], requiring time $O(n^{k(2+\epsilon)/3})$ where $2+\epsilon$ represents the best known exponent for fast matrix multiplication.

We show that Weighted Satisfiability (defined below) reduces to Dominating Set. By the *weight* of a truth assignment to a set of boolean variables, we mean the number of variables assigned the value *true*. Since Independent Set (and many other parameterized problems) easily reduce to this problem, we have the consequence claimed above. For example, a graph $G = (V, E)$ has a k-element independent set if and only if the expression $\prod_{uv \in E}(\bar{u} + \bar{v})$ has a weight k truth assignment. The notion of reduction that we use is defined in Section 1.

Weighted Satisfiability
Instance: A boolean expression X in conjuctive normal form, and a positive integer k.
Question: Is there a truth assignment of weight k that satisfies X?

Theorem 3.1 Weighted Satisfiability uniformly reduces to Dominating Set.

Proof. Let X be a Boolean expression in conjuctive normal form consisting 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 $G = (V, E)$ that has a dominating set of size $2k$ if and only if X is satisfied by a truth assignment of weight k.

An example of the construction is shown in Figure 1.

Figure 1. An example of the reduction.

The vertex set V of G is the union of the following sets of vertices:
$V_1 = \{a[r, s] : 0 \leq r \leq k-1, 0 \leq s \leq n-1\}$
$V_2 = \{b[r, s, t] : 0 \leq r \leq k-1, 0 \leq s \leq n-1, 1 \leq t \leq n-k+1\}_3 = \{c[j] : 1 \leq j \leq m\}$
$V_4 = \{a'[r, u] : 0 \leq r \leq k-1, 1 \leq u \leq 2k+1\}$
$V_5 = \{b'[r, u] : 0 \leq r \leq k-1, 1 \leq u \leq 2k+1\}$
$V_6 = \{d[r, s] : 0 \leq r \leq k-1, 0 \leq s \leq n-1\}$

For convenience, we introduce the following notation for important subsets of some of the vertex sets above. Let
$A(r) = \{a[r, s] : 0 \leq s \leq n-1\}$
$B(r) = \{b[r, s, t] : 0 \leq s \leq n-1, 1 \leq t \leq n-k+1\}$
$B(r, s) = \{b[r, s, t] : 1 \leq t \leq n-k+1\}$

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.
$E_1 = \{c[j]a[r, s] : x_s \in C_j\}$
$E_2 = \{a[r, s]a[r, s'] : s \neq s'\}$
$E_3 = \{b[r, s, t]b[r, s, t'] : t \neq t'\}$
$E_4 = \{a[r, s]b[r, s', t] : s \neq s'\}$
$E_5 = \{b[r, s, t]d[r, s'] : s' \neq s + t \pmod{n}\}$

$E_6 = \{a[r,s]a'[r,u]\}$
$E_7 = \{b[r,s,t]b'[r,u]\}$
$E_8 = \{c[j]b[r,s,t] : \exists i\ \overline{x_i} \in C_j, s < i < s+t\}$
$E_9 = \{d[r,s]a[r',s] : r' = r+1 \pmod n\}$

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

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

is a dominating set in G.

Conversely, suppose D is a dominating set of $2k$ vertices in G. The closed neighborhoods of the $2k$ vertices $a'[0,1], ..., a'[k-1,1], b'[0,1], ..., b'[k-1,1]$ are disjoint, so D must consist of exactly $2k$ vertices, one in each of these closed neighborhoods. Also, none of the vertices of $V_4 \cup V_5$ are in D, since if $a'[r,u] \in D$ then necessarily $a'[r,u'] \in D$ for $1 < u' < 2k+1$ (otherwise D fails to be dominating), which contradicts that D contains exactly $2k$ vertices. It follows that D contains exactly one vertex from each of the sets $A(r)$ and $B(r)$ for $0 \leq r \leq k-1$.

The possibilities for D are further constrained by the edges of E_4, E_5 and E_9. The vertices of D in V_1 represent the variables set to *true* in a satisfying truth assignment for X, and the vertices of D in V_2 represent intervals of variables set to *false*. Since there are k variables to be set to *true* there are, considering the indices of the variables mod n, also k intervals of variables to be set to *false*.

The edges of E_4, E_5 and E_9 enforce that the $2k$ vertices in D must represent such a choice consistently. To see how this enforcement works, suppose $a[3,4] \in D$. This represents that the third of k distinct choices of variables to be given the value *true* is the variable x_4. The edges of E_4 force the unique vertex of D in the set $B(3)$ to belong to the subset $B(3,4)$. The index of the vertex of D in the subset $B(3,4)$ represents the difference (mod n) between the indices of the third and fourth choices of a variable to receive the value *true*, and thus the vertex represents a range of variables to receive the value *false*. The edges of E_5 and E_9 enforce that the index t of the vertex of D in the subset $B(3,4)$ represents the "distance" to the next variable to be set *true*, as it is represented by the unique vertex of D in the set $A(4)$.

It remains only to check that the fact that D is a dominating set insures that the truth assigment represented by D satisfies X. This follows by the definition of the edge sets E_1 and E_8. □

Because Dominating Set can be easily reduced to Weighted Satisfiability with no negated literals, the above theorem shows the surprising fact that Weighted Satisfiability reduces to Monotone Weighted Satisfiability. Interpreted in terms of circuits, this combinatorial

reduction plays a crucial role in the fundamental completeness results surveyed in the next section.

4. A Completeness Theory for Fixed-Parameter Intractability

So far we have seen that many parameterized problems are fixed-parameter tractable, by means of a variety of algorithm design techniques. We have also seen that there are a number of interesting reductions between natural problems. 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.

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) 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 *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 $W[t]$ (*monotone* $W[t]$) if L uniformly

reduces to the parameterized circuit problem L_F for some family F of bounded depth, mixed type (monotone) decision circuits of weft at most t.

Definition. We designate the class of fixed-parameter tractable problems the *Easy* class and denote it E.

Thus we have the containments

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

and we conjecture that each of these containments is proper. We term the union of these classes the W Hierarchy, and denote it WH.

Lemma 4.1 If $P = NP$ then $WH \subseteq E$. □

Our main result shows that Weighted Satisfiability is complete for $W[2]$ and that similar problems are complete for each level of the W Hierarchy of parameterized problem classes. Theorem 4.1 plays a role in our theory analogous to Cook's theorem for NP-completeness. It is interesting that the combinatorial reduction of Theorem 3.1 plays a key role (as a "change of variables") in our proof of Theorem 4.1. Thus the entire argument that Dominating Set is complete for $W[2]$ actually uses this combinatorial reduction *twice*. A variation of Weighted Satisfiability based on a normal form for boolean expressions supplies the problems we identify in Theorem 4.1 as complete.

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 4.1 For $t \geq 2$ Weighted t-Normalized Satisfiability is complete for $W[t]$.

Proof. Let $L \in W[t]$. Let F be the family of circuits of depth bounded by h and weft bounded by t to which L reduces. It suffices to reduce L_F to Weighted t-Normalized Satisfiability. An instance of the latter problem may be viewed as a pair consisting of a positive integer k and a circuit having t alternating layers of *And* and *Or* gates corresponding to the t-normalized expression structure P-o-S-o-P-..., and having a single output *And* gate. Thus the argument essentially shows how to "normalize" the circuits in F.

Let (C, k) be an instance of L_F. We show how to determine whether C accepts a weight k input vector, by consulting an oracle for Weighted t-Normalized Satisfiability (viewed as

a problem about circuits) for finitely many weights k'. The algorithm for this determination will be uniform in k, and run in time $f(k)n^{\alpha}$ where n is the size of the circuit C. The exponent α will be a (possibly exponential) function of h and t. This is permissible, since every circuit in F observes these bounds on depth and weft.

Step 1. The reduction to tree circuits.

The first step is to transform C into a *tree circuit* C' of depth and weft bounded by h and t, respectively. In a tree circuit every logic gate has fan-out one. (The input nodes may have large fan-out.) The transformation is 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 $O(n^{O(h)})$ and involve a similar blow-up in the size of the circuit. The tree circuit C' accepts a weight k input vector if and only if the original circuit C accepts a weight k input vector.

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). Trpns3orvP`a` 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, and may refer to negated fan-out lines from the input nodes as negated inputs. Next, there are levels consisting only of large and small *and* and *or* gates, with a single output gate (which may be of either principal logical denomination at this point).

Step 3. Homogenizing the layers.

We transform our current circuit C (received from the previous step) by repeating the following operations (with operation (1) of higher priority) until no further operation of either kind is possible.
(1) Merge two gates of the same logical character if the (single) output line from one gate is an input line to the other gate. (This produces an equivalent circuit.)
(2) Commute a small gate upward past the two gates (necessarily of complementary character) that produce its inputs. For example, a small *and* gate is commuted past two large *or* gates that supply the two arguments in accordance with the distribution:

$$(a_1 + a_2 + ... + a_r)(b_1 + b_2 + ... + b_s) = (a_1 + b_1)(a_1 + b_2)(a_2 + b_1)...(a_r + b_s)$$

This step may require time (and increase the size of the circuit) by the function $n \mapsto n^{2^{h^2}}$. The circuit C' produced has large gates only in the bottommost t layers.

Step 4. Removing a bottommost *Or* gate.

Since our reductions are Turing reductions, we can determine whether a tree circuit giving output from an *Or* gate accepts weight k input vector, by simply making the same determination for each of the input branches to the gate.

Step 5. Organizing the small gates.

The tree circuit C received from the previous step has the properties: (i) the output gate is an *And* gate, (ii) from the bottom, the circuit consists of layers which alternately consist of only *And* gates or only *Or* gates, for up to t layers, and (iii) above this, there are branches B of height $h' = h - t$ consisting only of small gates. Since a small gate branch B has bounded depth, it has at most $2^{h'}$ gates, and thus in constant time (since h is fixed), we can find an equivalent sum-of-products circuit with which to replace B.

In this step, all such small gate branches B of C are replaced in this way. The depth 2 sum-of-products circuits replacing the small gate branches B have a bottomost *or* gate g_B of fan-in at most $2^{2^{h'}}$, and the *and* gates feeding into g_B have fan-in at most $2^{h'}$, so the weft of the circuit has been preserved by this transformation, which may increase the size of C by the constant factor $2^{2^{h'}}$. If the topmost level of large gates (to which the branches B are attached in C) consists of *Or* gates, then the gates g_B can be merged into this level. For the next step we consider two cases, depending on whether the topmost level of large gates consists of *And* gates or *Or* gates. (Essentially, this corresponds to whether t is even or odd.)

Step 6. A monotone change of variables. (Two cases.)

Case 1. The topmost large-gate level consists of *Or* gates.

Let C denote the circuit resulting from the above transformations. We perform a "change of variables" based on the combinatorial reduction of Theorem 3.1. The sequence of transformations of C for this step is shown schematically in Figure 2.

Figure 2. Case 1: A topmost large-gate layer of *Or* gates.

The new circuit inputs correspond to the vertices (other than the clause vertices) of the graph G_X constructed in the proof of that theorem. Each positive input to C (corresponding to a positive literal in the construction of G_X) is represented k times in the construction, and each negated input to C (corresponding to a negative literal) is represented $O(n^2)$ times, where n is the number of inputs to C.

There are two aspects to this change of variables. First, each input line to the top layer of *and* gates in C that arrives directly from an input gate (without a negation) is replaced by an *Or* gate with fan-in k, and each input line to the top layer of small *or* gates that is negated enroute from an input gate is replaced by a (large) *Or* gate. Secondly, we must add to the circuit an *enforcement* of the combinatorics of the Dominating Set problem. This can be expressed as the requirement that the weight $2k$ input must also be accepted by a P-o-S circuit which is the product over the vertex set (the new input set) of the sum of the inputs corresponding to a closed neighbor of a vertex. This enforcement can be merged with the bottommost (output) *And* gate of the circuit.

The result is a circuit C' with no *not* gates. The input weight we are now concerned with is $2k$, and the construction of C' from C may involve quadratic blow-up.

Next, we commute the small *and* gates on the second level upward past the *Or* gates introduced by the change of variables, and then merge the *Or* gates down to the topmost large-gate layer (of *Or* gates). The one remaining step is described below following the treatment of case 2.

Case 2. The topmost large-gate level consists of *And* gates.

The sequence of transformations for this case is depicted schematically in Figure 3. Below each gate of the topmost large-gate layer (of *And* gates), a double negation is introduced (equivalently). One of the *not* gates is commuted to the top of the circuit (by DeMorgan's identities). This is followed by a change of variables based on Theorem 3.1, as in Case 1. The second level *and* gates are commuted upwards, and the *Or* gates of the second and third levels are merged, as in Case 1. Finally, the remaining *not* gates are commuted to the top. We are now in position for the last step.

Figure 3. Case 2: A topmost large-gate level of *And* gates.

Step 7. Eliminating the remaining small gates.

The circuit C that we receive at the beginning of this step is depicted schematically in figure 2(d) or figure 3(d). If we regard the inputs to C as variables, this step consists of another "change of variables." Let k be the relevant weight parameter value supplied by the last transformation. In this step we will produce a circuit C' corresponding directly to a t-normalized boolean expression (that is, consisting only of t alternating layers of *And* and *Or* gates) such that C accepts a weight k input vector if and only if C' accepts a vector of weight $k' = k \cdot 2^{k+1} + 2^k$.

Suppose that C has m remaining small gates. In Case 1, these are *and* gates, and the inputs are all positive. In Case 2, these are *or* gates, and the inputs are all negated. For $i = 1, ..., m$ we define the sets A_i of the inputs to C to be the sets of input variables to these small gates. The central idea for this step is to create new inputs representing the sets A_i of inputs to C.

For example, suppose (Case 1) that the output of the small *and* gate g_i in C is the boolean product $(abcd)$ of the inputs a, b, c, d to C. Thus $A_i = \{a, b, c, d\}$. The gate g_i can be eliminated by replacing it with an input line from a new variable $v[i]$ which represents the predicate $a = b = c = d = 1$. (This representation, of course, will need to be enforced by additional circuit structure.) Similarly (Case 2) if g_i computes the value $(\bar{a} + \bar{b} + \bar{c} + \bar{d})$ then g_i can be replaced by a negated input line from $v[i]$.

Let $x[j]$ for $j = 1, ...s$ be the input variables to C. We introduce new input variables of the following kinds:
(1) One new variable $v[i]$ for each set A_i for $i = 1, ..., m$ to be used as indicated above.
(2) For each $x[j]$ we introduce 2^{k+1} copies $x[j, 0], x[j, 1], x[j, 2], ..., x[j, 2^{k+1} - 1]$.
(3) "Padding" consisting of 2^k meaningless variables (inputs not supplying output to any gates) $z[1], ..., z[2^k]$.

We add to the circuit an enforcement mechanism for the change of variables. The necessary requirements can be easily expressed in P-o-S form, and thus can be incorporated into the bottom two levels of the circuit as additional *Or* gates attached to the bottommost (output) *And* gate of the circuit.

We require the following implications concerning the new variables:

(1) The $s \cdot 2^{k+1}$ implications, for $j = 1, ..., s$ and $r = 0, ..., 2^{k+1} - 1$,
$$x[j, r] \Rightarrow x[j, r + 1 \pmod{2^{k+1}}]$$
(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

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

(4) For $i = 1, ..., m$ the implication

$$(\prod_{x[j] \in A_i} x[j,0]) \Rightarrow v[i]$$

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 between $k \cdot 2^{k+1}$ and $k \cdot 2^{k+1} + 2^k$ that is accepted by C'. The reason the weight of this corresponding vector may fall short of $k' = k \cdot 2^{k+1} + 2^k$ is that not all of the subsets of the k input variables to C having value 1 may occur among the sets A_i. An accepted vector of weight exactly k' can be obtained by employing some of the "padding" input variables $z[i]$ to C'

Note that the seemingly simpler strategy of creating a new input variable for each set of at most k inputs to C would not serve our purposes, since it would involve increasing the size n of the circuit to possibly n^k. (We are limited in our computational resources for the reduction to $f(k)n^\alpha$. The constant α can be an arbitrary function of the depth and weft bounds h and t, but not k.)

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. Because of the implications (2)–(4), 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. □

The $W[t]$ hierarchy reflects, in a finely resolved way, the difficulty of "solution checking." What happens if, more bluntly, we simply address fixed-parameter complexity for problems for which solutions can be checked in *polynomial* time? To study this question, it is natural to define the following complexity class.

Definition. A parameterized problem L belongs to $W[P]$ (*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 $W[P] = E$ if $P = NP$. We have been able to show that all of the problems identified in [AEFM] as complete for PGT are uniformly complete for $W[P]$. (We would argue that the present theory offers a better framework for those results.) We also have these further completeness results.

Theorem. The following problems are complete for $W[P]$:

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.

5. Open Problems

In some ways, the study of fixed-parameter tractability and completeness addresses the subject of computational infeasibility inside of P. For related work see Buss and Goldsmith [BG]. Many of the approaches and issues concerning the standard complexity classes have natural analogues in this setting that are so far unexplored.

Consider, for example, the issue of parallel complexity. Trivially, there is a parallel algorithm running in time $O(\log n)$ and using n^k processors to determine if a graph G on n vertices has a dominating set of size k, for each fixed k. For a contrasting result, Lagergren [La] has shown that for each fixed k, it can be determined in time $O(\log^2 n)$ with $O(n^3)$ processors whether a graph has treewidth at most k. This suggests a natural fixed-parameter analogue of NC.

For another example, consider approximation algorithms. One of the fundamental results of Robertson and Seymour (quite apart from their work on graph minors) is that there is an algorithm that in time $f(k) \cdot n^2$ finds for a graph G of order n, either: (1) a tree decomposition of width at most $5k$, or (2) evidence that the treeewidth of G is greater than k. An analogous result for Dominating Set might be an algorithm running in time $f(k) \cdot n^c$ that finds either: (1) a dominating set of size $O(k)$, or (2) evidence that the minimum size of a dominating set for G is greater than k. Such an algorithm is presently unknown. It may even be that the existence of such an algorithm would imply the collapse of the W hierarchy, much as the existence of a P-time relative approximation algorithm for the Travelling Salesman problem would imply $P = NP$ [GJ].

Many interesting structural questions concerning the W hierarchy remain to be explored. For example, we conjecture that $W[1]$ does not have complete problems, in contrast to $W[t]$ for $t \geq 2$. We also do not know if a problem such as the following belongs to $W[t]$ for any t.

Two-Player Dominating Set
Instance: A graph $G = (V, E)$ and a positive integer k.
Question: Is it true that for every k-element subset $V' \subseteq V$, there is a k-element subset

$V'' \subseteq V$ such that $V' \cup V''$ is a $2k$-element dominating set for G?

It would be interesting to know if the collapse of the W hierarchy would imply any "disaster" concerning the usual complexity classes (such as the collapse of the polynomial hierarchy (see [GJ]) to a finite level). We have been able to prove the following analogue of Ladner's well-known theorem.

Theorem 5.1. If any of the containments

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

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

Lastly, there are many natural parameterized problems that may well be complete for various levels of the W hierarchy. Demonstrations of such completeness would provide an explanation of why, though they are solvable in polynomial time for each fixed parameter value, these problems resist attempts to show fixed-parameter tractability.

Acknowledgements Thanks to Karl Abrahamson for useful early discussions about this work.

References

[ACPS] S. Arnborg, B. Courcelle, A. Proskurowski and D. Seese, "An Algebraic Theory of Graph Reduction," Technical Report 90-02, Laboratoire Bordelais de Recherche en Informatique, Universite de Bordeaux I, January 1990.

[AEFM] K. R. Abrahamson, J. A. Ellis, M. R. Fellows and M. E. Mata, "On the Complexity of Fixed-Parameter Problems." In *30th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press (1989), 210-215.

[AF] K. R. Abrahamson and M. R. Fellows, "Cutset Regularity Beats Well-Quasiordering for Bounded Treewidth," to appear.

[Ar] S. Arnborg, "Efficient Algorithms for Combinatorial Problems on Graphs with Bounded Decomposability — A Survey," *BIT* 25 (1985), 2-23.

[ALS] S. Arnborg, J. Lagergren and D. Seese, "Problems Easy for Tree-Decomposable Graphs (extended abstract)." In T. Lepisto and A. Salomaa, eds., *Proc. 15th Int. Coll. Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 317 (Springer, Berlin, 1988), 38-51.

[BG] J. F. Buss and J. Goldsmith, "Nondeterminism Within P," manuscript, 1990.

[BLW] M. W. Bern, E. L. Lawler and A. L. Wong, "Linear Time Computation of Optimal Subgraphs of Decomposable Graphs," *J. of Algorithms* 8 (1987), 216-235.

[BM] D. Bienstock and C. L. Monma, "On the Complexity of Covering Vertices by Faces in a Planar Graph," *SIAM J. Comp.* 17 (1988), 53-76.

[Bod1] H. L. Bodlaender, "On Linear Time Minor Tests and Depth First Search," In F. Dehne et al., eds., *Proc. 1st Workshop on Algorithms and Data Structures*, Lecture Notes in Computer Science, Vol. 382 (Springer, Berlin, 1989), 577-590.

[Bod2] H. L. Bodlaender, "On Disjoint Cycles," Technical Report RU*-C)¯9¸¯YøVQe"2x:ff of Computer Science, Utrecht University, Utrecht, The Netherlands, August 1990.

[Co] B. Courcelle, "Graph Rewriting: An Algebraic and Logical Approach." In J. van Leeuwen, ed., *Handbook of Theoretical Computer Science, Vol. B* (North-Holland, Amsterdam, 1990), Chapter 5.

[EP] P. Erdős and L. Pósa, "On Independent Circuits Contained in a Graph," *Canad. J. Math.* 17 (1965), 347-352.

[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).

[La] J. Lagergren, "Algorithms and Minimal Forbidden Minors for Tree-Decomposable Graphs," Dissertation, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden, March 1991.

[NP] J. Nesetril and S. Poljak, "On the Complexity of the Subgraph Problem," Commen. Math. Univ. Carol. 26 (1985), 415-419.

[PV] J. Plehn and B. Voigt, "Finding Minimally Weighted Subgraphs," to appear.

[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.

[WHL] T. V. Wimer, S. T. Hedetniemi and R. Laskar, "A Methodology for Constructing Linear Graph Algorithms," *Congressus Numerantium* 50 (1985), 43-60.