The Parameterized Complexity of Short Computation and Factorization

Liming Cai and Jianer Chen
Department of Computer Science, Texas A & M University
College Station, Texas 77843 U.S.A.

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

Michael R. Fellows
Department of Computer Science, University of Victoria
Victoria, British Columbia V8W 3P6 Canada
contact author for this abstract: mfellows@csr.uvic.ca

Summary

A completeness theory for parameterized computational complexity has been studied in a series of recent papers, and has been shown to have many applications in diverse problem domains including familiar graph-theoretic problems, VLSI layout, games, computational biology, cryptography, and computational learning [ADF,DEF,DF1-7,FH,FHW,FK]. We here study the parameterized complexity of two kinds of problems: (1) problems concerning parameterized computations of Turing machines, such as determining whether a nondeterministic machine can reach an accept state in k steps (the *Short TM Computation Problem*), and (2) problems concerning derivations and factorizations, such as determining whether a word x can be derived in a grammar G in k steps, or whether a permutation has a factorization of length k over a given set of generators. These include a natural parameterized version of the famous Post Correspondence Systems. We show hardness and completeness for these problems for various levels of the W hierarchy. In particular, we show that *Short TM Computation* is complete for $W[1]$. This gives a new and useful characterization of the most important of the apparently intractable parameterized complexity classes. The result could be viewed as one analogue of Cook's theorem and we believe provides strong evidence for the parameterized intractability of $W[1]$.

1. Introduction

The central issues on which the theory of parameterized computational complexity focuses are rooted in the following general observations concerning computational problems.

• Many important natural problems have input that consists of two or more items. For example, the familiar problems *Bandwidth*, *Min Cut Linear Arrangement*, *Vertex Cover* and *Dominating Set* all take as input a graph G and a positive integer k. So it is natural to consider the relative contributions of the different parts of the input to the complexity of the problem.

• There is a large and growing body of results of the following qualitative sort: there is a constant α such for every fixed "parameter" k, the problem Π is solvable in time $O(n^\alpha)$, with α independent of k. For example, *Min Cut Linear Arrangement* and *Vertex Cover* are solvable in time linear in the number of vertices of G for every fixed k. The algorithmic techniques used to prove such *fixed parameter tractability* results are in some cases interesting and distinctive (e.g., well-quasiordering).

• The fact that a problem Π taking as input two items x and k is complete for *NP* (or *PSPACE*, or *EXPTIME*, ...) tells us *nothing* about the fixed-parameter tractability or intractability of Π when k is held fixed.

• For many natural problems, a small range of parameter values may cover important applications.

• Many natural problems involve a "hidden parameter." Elucidating the contribution of this parameter to the complexity of the problem may illuminate why the problem is "easier to solve in practice" than one might expect from traditional complexity analysis. An example of this phenomena is the complexity of *Type Inference in ML* [HM,DF6]. Conversely, engineering practice may introduce (and fix) a parameter, that renders the complexity of the problem easier than one desires (e.g. in cryptography). An example of this phenomena is described in [FK]. The familiar practice of "coping with NP-completeness" by restricting the input can often be conveniently described by a parameterization (for example, by making the parameter the treewidth of the input).

• For many familiar parameterized problems the difference between fixed-parameter tractability and apparent intractability is very much akin to the apparent difference we observe between problems which are in P and problems which are NP-complete. For example, while we can determine whether a graph has a vertex cover of size k or a cutwidth k layout in linear time for each fixed k, the best known algorithms for determining whether a graph has a dominating set of size k, or a bandwidth k layout, require time $O(n^{k+1})$ and are based essentially on a brute force examination of possible solutions. This is strongly reminiscent of the current situation for many NP-complete problems.

Our main results concern the following two problems about resource-bounded Turing machine computations; informally, these study the power of *k-time* and *k-space* for nondeterminsitic machines.

Short TM Computation
Input: A nondeterministic Turing machine M, a word x and a positive integer k.
Parameter: k
Question: Is there an computation of M on input x that reaches an accept state in at most k steps?

Compact TM Computation
Input: A nondeterministic Turing machine M, a word x and a positive integer k.
Parameter: k

Question: Is there an accepting computation of M on input x that visits at most k tape squares?

Our main result, Theorem 1, shows that *Short TM Computation* is complete for $W[1]$, and will be seen to provide a new and useful characterization of this most important of the parameterized complexity classes. ("Most important," because $W[1]$-hardness is presently the minimum available demonstration of likely fixed-parameter intractability.) Concretely, Theorem 1 shows that *Short TM Computation* is fixed-parameter tractable if and only if the familiar (and apparently resistant) *Clique* problem is fixed parameter tractable.

Theorem 2 shows that the k-space analog, *Compact TM Computation*, is hard for the complexity class $W[P]$.

In the next section we review the basics of parameterized computational complexity. In §3 we sketch the proofs of the main theorems. In §4 we address related problems about grammars, Post systems, and square tiling. In §5 we consider some more distantly related short factorization problems in permutation groups and monoids.

2. Parameterized Computational Complexity

The formal framework for our study is established as follows.

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

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

Definition. We say that a parameterized problem L is (uniformly) *fixed-parameter tractable* if there is a constant α and an algorithm Φ such that Φ decides if $(x,k) \in L$ in time $f(k)|x|^\alpha$ where $f : N \to N$ is an arbitrary function.

Definition. Let A, B be parameterized problems. We say that A is (uniformly many:1) *reducible* to B if there is an algorithm Φ which transforms (x,k) into (x',k') in time $f(k)|x|^\alpha$, where $f : N \to N$ is an arbitrary function and α is a constant independent of k, so that $(x,k) \in A$ if and only if $(x',k') \in B$.

It is easy to see that if A reduces to B and B is fixed parameter tractable then so too is A. Note that if $P = NP$ then problems such as Minimum Dominating Set are fixed-parameter tractable. Thus a completeness program is reasonable.

The class $W[1]$. As with classical complexity theory we need natural distinguished classes such as NP, $PSPACE$, etc. Due to the refined nature of the reductions, it seems that there are many natural intractable classes (i.e. parameterized *degrees*). Our current working "minimal" intractable class is one we call $W[1]$ as it is the bottom class of a hierarchy we call the weft hierarchy as we discuss below. However the reader can think of it as the class generated by the following problem:

Weighted 3SAT.
Input: A formula X in 3CNF form.
Parameter: k.
Question: Does X have a weight k satisfying assignment? (A weight k assignment is one with exactly k literals true.)

The reader should note that we are careful to specify that the formula is in $3CNF$ form. Clearly one could consider the problem of *Weighted SAT* for any formula not just one in $3CNF$ form. However we believe that the problems are really of differing parameterized complexity and form only parts of a hierarchy as we see below. However for the purposes of the present paper, the only problems the reader needs are *weighted* $3SAT$ and the question of whether a circuit (or Turing Machine) accepts a weight k vector. This latter class we call $W[P]$. A more precise characterization of this last class and a context for the classes is given in the digression below. This can be skipped with no effect on readibility of the rest of the paper, and the reader can well skip to §3 if one wishes to proceed directly to our results.

Digression on the W-hierarchy. The classes of parameterized problems that we define below are intuitively based on the complexity of the circuits required to check a solution, or alternatively the "natural logical depth" of the problem. (See also [CC1-2], [KT] and [PY] for differing but complementary views of this issue in terms of alternating logarithmically bounded Turing Machines and Fagin-style logical expressibility frameworks.)

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

Definition. A Boolean circuit is of *mixed type* if it consists of circuits having gates of the following kinds.
(1) *Small gates*: *not* gates, *and* gates and *or* gates with bounded fan-in. We will usually assume that the bound on fan-in is 2 for *and* gates and *or* gates, and 1 for *not* gates.
(2) *Large gates*: *And* gates and *Or* gates with unrestricted fan-in.

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 decision 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. 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 $W[t]$ if L reduces to the parameterized circuit problem $L_{F(t,h)}$ for the family $F(t, h)$ of mixed type decision circuits of weft at most

t, and depth at most h, for some constant h.

Definition. A parameterized problem L belongs to $W[P]$ is L reduces to the circuit problem L_F, where F is the set of all circuits (no restrictions). If we restrict to Boolean circuits we call the class $W[SAT]$.

Definition. We designate the class of fixed-parameter tractable problems FPT.

The above leads to an interesting hierarchy

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

for which a wide variety of natural problems are now known to be complete or hard for various levels (compendiums can be found in [DF2,4,7]). In terms of boolean formulae we have the following characterization from [DF2,4,7]. We say that a formula is in t-normalized form if it is in the form of product of sums of product of ... with t alternations. (So a CNF formula is 2-normalized.) Downey and Fellows prover that weighted satisfiability for t-normalized formulae is complete for $W[t]$.

It should be emphasized that there is no easy correspondence between membership or hardness in any of these parameterized complexity classes and NP- or *PSPACE*-completeness (etc.) for the corresponding "unparameterized" problems. For example, in [ADF] problems concerning k-move games are studied, all of which are *PSPACE*-complete in unparameterized form; some of these turn out to be fixed parameter tractable, and others to be complete for $W[P]$. In [DEF] it is shown that *Vapnik-Chervonenkis Dimension* is complete for $W[1]$; the unparameterized form of this problem is probably of difficulty intermediate between P and NP (see [PY]).

3. k-Resource Bounded Turing Machine Computations

Theorem 1. *Short TM Computation* is complete for $W[1]$.

Proof. To show hardness for $W[1]$ we reduce from *Clique*, which is shown to be hard for $W[1]$ in [DF3]. Let $G = (V, E)$ be a graph for which we wish to determine whether it contains a k-clique. We shown how to construct a nondeterministic Turing machine M that can reach an accept state in $k' = f(k)$ moves if and only if G contains a k-clique. The Turing machine M is designed so that any accepting computation consists of two phases. In the first phase, M nondeterministically writes k symbols representing vertices of G in the first k tape squares. (There are enough symbols so that each vertex of G is represented by a symbol.) The second phase consists of making $\binom{k}{2}$ scans of the k tape squares, each scan devoted to checking, for a pair of positions i, j, that the vertices represented by the symbols in these positions are adjacent in G. Each such pass can be accomplished by employing $O(|V|)$ states in M dedicated to the ij^{th} scan.

In order to show membership in $W[1]$ it suffices to show how the *Short Computation* problem for a Turing machine $M = (\Sigma, Q, q_0, \delta, F)$ and positive integer k can be translated into one about whether a circuit C accepts a weight k' input vector, where C has depth bounded by some t (independent of k and the Turing machine M), and has only a single large (output) *and* gate, with all other gates small. We arrange the circuit so that the k' inputs to be chosen to be set to 1 in a weight k' input vector represent the various data: (1) the i^{th} transition of M, for $i = 1, ..., k$, (2) the head position at time i, (3) the state of M at time i, and (4) the symbol in square j at time i for $1 \leq i, j \leq k$. Thus we may take $k' = k^2 + 3k$. In order to force exactly one input to be set equal to 1 among a pool of input variables (for representing one of the above choices), we can add to the circuit, for each such pool of input variables, and for each pair of variables x and y in the pool, a small "not both" circuit representing $(\neg x \vee \neg y)$. It might seem that we must also enforce (e.g. with a large *or* gate) the condition, "at least one variable in each such pool is set true" — but this is actually unnecessary, since in the presence of the "not both" conditions on each pair of input variables in each pool, an accepted weight k' input vector *must have* exactly one variable set true in each of the k' pools. Let n denote the total number of input variables in this construction. We have in any case $n = O(k\delta + k^2 + k|Q| + k^2|\Sigma|)$.

The remainder of the circuit encodes various checks on the consistency of the above choices. These consistency checks conjunctively determine whether the choices represent an accepting k-step computation by M, much as in the proof of Cook's theorem. These consistency checks can be implemented so that each involves only a bounded number b of the input variables. For example, we will want to enforce that if five variables are set true indicating particular values of: the tape head position at time $i + 1$, and the the head position, state, scanned symbol and machine transition at time i, then the values are consistent with δ. Thus $O(n^5)$ small "checking" circuits of bounded depth are sufficient to make these consistency checks; in general we will have $O(n^b)$ "checking" circuits for consistency checks involving b values. All of the small "not both" and "checking" circuits

feed into the single large output *and* gate of C. The formal description of all this is laborious but straightforward. □

We remark that Theorem 1 depends crucially on there being no bound on the size of the Turing machine alphabets in the definition of the problem. If we restrict *Short TM Computation* to TM's with $|\Sigma|$ bounded by some constant b, then the number of configurations is bounded by $b^k|Q|k$ and the problem becomes fixed parameter tractable. We suggest that Theorem 1 provides strong evidence that the class $W[1]$ really is intractable because of the generic nature of Turing Machine computations.

Theorem 2. *Compact TM Computation* is hard for $W[P]$.

Proof. To show that the problem is hard for $W[P]$ we reduce from the problem *Monotone Weight k Circuit Satisfiability* that has been shown to be complete for $W[P]$ in [DFHKW] (See also [ADF2]). Let C be a circuit for which we wish to determine whether there is an input vector of weight k accepted by C. We may assume that each logic gate g of C has two inputs. In time polynomial in $|C|$ we can describe a Turing machine M sketched as follows.

M has an alphabet consisting of one letter for each input to C, and the operation of M consists of two phases. In the first phase, M makes k moves nondeterministically, writing down in the first k tape squares k symbols which represent k inputs to C set to 1. In the second phase (and visiting no other tape squares), M checks whether the the guess made in the first phase represents a vector accepted by the circuit C.

The key point is that we can structure the transition table of M to accomplish this, with the size of the table polynomial in $|C|$. To do this, we make two states q^l_{up} and q^l_{down} for each connection (or *line*) l of the circuit C. Let g be an *and* gate of C, let l be an output line of g and suppose the input lines to the gate g are l_1 and l_2. We include in the transition table for M transitions from q^l_{up} to $q^{l_1}_{up}$, from $q^{l_1}_{down}$ to $q^{l_2}_{up}$, and from $q^{l_2}_{down}$ to q^l_{down}. The significance of being a state q^l_{down} is that this represents a value of 1 for the line l as computed by C on the input guessed in the first phase. The state q^l_{up} might be viewed as a state of *query* about the value of l for the circuit C on the input guessed in the first phase. Note that the three transitions described above for the *and* gate g thus enforce that q^l_{down} can be reached only if $q^{l_1}_{down}$ *and* $q^{l_2}_{down}$ can be reached. The appropriate transitions for an *or* gate will differ in the obvious way, i.e., we arrange that the state q^l_{down} can be reached if either of $q^{l_1}_{down}$ or $q^{l_2}_{down}$ can be reached.

If l is an input line to the the circuit C, then we encode in the state table for M a "check" (involving a scan of the k tape squares) to see if the corresponding input symbol was written during the first phase of computation. The second phase begins in the state $q^{l_{out}}_{up}$ where l_{out} is the output line of C, and the only accept state is $q^{l_{out}}_{down}$. □

4. Grammars, Post Systems and Tiling

In this section we consider the following problems.

Short Derivation (for Unrestricted Grammars)

Input: A phrase-structure grammar G (unrestricted), a word x and a positive integer k.
Parameter: k
Question: Is there a G-derivation of x of length k?

Short Post Correspondence
Input: A Post system Π and a positive integer k.
Parameter: k
Question: Is there a length k solution for Π?

Lemma 1. *Clique* reduces to *Short Derivation*.

Proof. The proof of this is rather intricate, and space limitations here permit only the barest sketch. Let S denote the starting symbol for the grammar. We have the initial production (the only one possible)

$$S \to R[k,0] \cdots R[1,0] \# X[1]X[2] \# X[1]X[3] \# \cdots \# X[k-1]X[k]Z$$

There are productions by which the symbols $X[i]$ may each produce a terminal symbol representing a vertex of the graph; this represents in some sense a "guess" about the vertices of a k-clique. Note that the $X[i]$ symbols are grouped according to "all possible pairs." For such a guess to be consistent, it must be verified that each symbol $X[i]$ produces the same guessed vertex. The R symbols check this by "commuting across" the intermediate string (transforming into other symbols which record information appropriately along the way). These "checking symbols" can only commute past pairs of vertex symbols (the descendants of the $X[i]$'s) that represent adjacent vertices in the graph. □

Lemma 2. *Short Post Correspondence* reduces to *Short TM Computation*.

Proof. Given an instance (Π, k) of the *Short Post Correspondence* problem, we can easily express the question in terms of whether a particular Turing machine has a k-step accepting computation as follows. Consider a machine M that computes in two phases, over an alphabet that includes one symbol for each pair of strings in the Post system, and one symbol for each of the positive integers $1, ..., m$ where m is a bound on the total number of symbols (in the strings of the Post system) for any solution. (Clearly $m \leq k|\Pi|$.) In the first phase, M writes on $3k$ tape squares indicating (nondeterministically) a *guessed solution* including the information: (1) what Post pair is the i^{th} factor for $i = 1, ..., k$, (2) on what symbol positions the i^{th} factors begin (in the two concatenated strings). In the second phase, M conducts a number of "checks", each consisting of: (1) a scan of the *guess*, recording in the resulting state q, e.g., that the i^{th} factor is (x_j, y_j) and begins in the first component in symbol position r, and that the next factor is due to begin in symbol position s. There is a transition out of q in the transition table for M if and only if the information recorded in the state q is "valid", i.e., $s = r + |x_j|$. The number of states required for this check is $O(|\Pi|^3)$. It is not too hard to see that $2k$ checks of this sort are enough to insure that the guessed starting positions of the factors are consistent. It is also necessary to similarly make k^2 checks that each guessed (factor + starting position) in the first component is compatible with each (factor + starting position) in the second component, i.e., that this data does not

imply any mismatched symbols in the two solution strings for the Post problem. A successful check will make $f(k)$ moves, where f is an appropriately chosen function of k. □

As a consequence of Lemmas 1 and 2 above, our Theorem 1, and the proof of the undecidability of Post correspondence of, say, Davis [Da, Theorem 4.2] (which gives a (parameterized) reduction from (Short) Derivation to (Short) Post Correspondence) we have:

Theorem 3. *Short Derivation* is complete for $W[1]$.

Theorem 4. *Short Post Correspondence* is complete for $W[1]$.

By a slightly more complicated argument than Lemma 1, we can shown that *Short Derivation* remains $W[1]$ complete for context-sensitive grammars. Using the gadgetry of Lewis [Le], we can show a similar result for the naturally parameterized problem *Square Tiling* defined in Garey and Johnson [GJ].

Theorem 5. *Square Tiling* is complete for $W[1]$.

5. Short Algebraic Factorizations

The following problem clearly belongs to $W[P]$.

Permutation Group Factorization
Instance: A set A of permutations $A \subseteq S_n$, and $x \in S_n$.
Parameter: A positive integer k.
Question: Does x have a factorization of length k over A?

Theorem 6. *Permutation Group Factorization* is hard for $W[1]$.

Proof. We reduce from the parameterized problem *Perfect Code* that is shown to be hard for $W[1]$ in [DF3]. Let $G = (V, E)$ be a graph of order n for which we wish to determine whether there is a perfect code of size k. Let $n' = (k+1)n$. We show how to produce an equivalent factorization problem instance for $S_{n'}$.

View n' as divided into n blocks of size $k+1$, with these blocks in 1:1 correspondence with the vertices of G. Let γ denote a cyclic permutation of the elements of a block. Our set A consists of n permutations, one for each vertex of G. For a vertex u and with $N[u]$ denoting the solid neighborhood of u in G, let a_u denote the element of $S_{n'}$ which acts on the blocks corresponding to $v \in N[u]$ according to γ, and which is the identity map on all other blocks.

The permutation x to be factored consists of the permutation γ on each of the n blocks.

The correctness of the reduction is easily seen. □

We next consider the related problem of finding short factorizations in the monoid H_n of self-maps on a set of n elements.

H_n Factorization
Instance: A set A of self-maps on $[n]$, and a self-map h.

Parameter: A positive integer k.
Question: Is there a factorization of h of length k over A?

Theorem 7. *H_n Factorization* is hard for $W[2]$.

Proof. The reduction is from *Dominating Set*. Let $G = (V, E)$ be a graph for which we are to determine whether there is a k-element dominating set. Let n be the order of G. As in Theorem 6, we construct a set A of self-maps on $[n']$ where we view $[n']$ as consisting of n blocks. Here we have $n' = 2n$ (the blocks have size 2). Let α denote the self-map on $\{1, 2\}$ that maps both elements to 2. For each vertex u of G we construct a map a_u that consists of α in each block corresponding to a vertex $v \in N[u]$, and that is the identity map on all other blocks.

The self-map h to be factored consists of α in each block.

Verification that this construction works correctly is straightforward, noting that $\alpha = \alpha^i$ for any number of compositions of the block map α. □

Since the above arguments do not employ any global aspects of the symmetric group or the monoid of self-maps (i.e., everything interesting occurs in the blocks separately) it seems reasonable to ask whether a more intricate construction could be used to improve these results. For example, it might be that these factorization problems are hard for $W[t]$ for any fixed t. Alternatively, it would be interesting if Permutation Group Factorization turned out to belong to $W[1]$ or $W[2]$. This is a nice example of a widespread situation in our present knowledge of the parameterized complexity of concrete problems: large gaps between membership and hardness results in the W hierarchy.

References

[ADF] K. Abrahamson, R. Downey and M. Fellows, "Fixed-Parameter Intractability II," *Proc. 10th Symposium on Theoretical Aspects of Computer Science (STACS)* (1993), 374–385, Springer-Verlag, Berlin, Lecture Notes in Computer Science.

[ADF2] K. Abrahamson, R. Downey and M. Fellows,"Fixed-Paramater Tractability and Completeness IV: On Completeness for $W[P]$ and $PSPACE$ Analogues,"Annals Pure and Applied Logic (submitted).

[CC1] L. Cai and J. Chen, "On the Amount of Nondeterminism and the Power of Verifying," in *Proc. International Conference on the Mathematical Foundations of Computer Science (MFCS)*, Lecture Notes in Computer Science Vol 711, (1993) 311-320.

[CC2] L. Cai and J. Chen, "On Fixed-Parameter Tractability and Approximability of NP-hard Optimization Problems," in *Proc. 2nd Israel Symposium on Theory and Computing Systems (ISTCS)*, (1993) 118-126.

[CCDF] L. Cai, J. Chen, R. G. Downey and M. R. Fellows, "Advice Classes of Parameterized Tractability," submitted to *Proc. Asian Logic Conference, 1993* (Annals Pure and Applied Logic, Special Issue).

[Da] M. Davis, "Unsolvable Problems," in the *Handbook of Mathematical Logic*, J. Barwise

(ed.), Elsevier, 1977, p. 580.

[DEF] R. G. Downey, P. A. Evans and M. R. Fellows, "Parameterized Learning Complexity," in *Proc. Sixth ACM Workshop on Computational Learning Theory (COLT)*, (1993) 51-57.

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

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

[DF3] R. G. Downey and M. R. Fellows, "Fixed Parameter Tractability and Completeness II: On Completeness for $W[1]$," to appear Theor. Comput. Sci.

[DF4] R. G. Downey and M. R. Fellows, "Fixed Parameter Intractability (Extended Abstract)," *Proceedings of the Seventh Annual IEEE Conference on Structure in Complexity Theory* (1992), 36-49.

[DF5] R. G. Downey and M. R. Fellows, "Fixed Parameter Tractability and Completeness III: Some Structural Aspects of the W-Hierarchy," to appear in *Complexity Theory* (ed. Ambos-Spies, Homer, Schoning) (Cambridge University Press).

[DF6] R. G. Downey and M. R. Fellows, "Parameterized Computational Feasibility," to appear in *Feasible Mathematics II* (ed. Clote and Remmel) (Birkhauser, Boston).

[DF7] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, monograph in preparation.

[DFHKW] R. G. Downey, M. Fellows, M. Hallett, B. Kapron, T. Wareham,"The Paramaterized Complexity of Some Problems in Logic and Linguistics," to appeat in *Procs. Vancouver Conf. on Theoretical Computer Science* (ed. Marek et. al.)

[FH] M. R. Fellows and M. T. Hallett, "Bandwidth is Hard for $W[1]$," in preparation.

[FHW] M. R. Fellows, M. T. Hallett and H. T. Wareham, "DNA Physical Mapping: Three Ways Difficult," in *Proc. First European Symposium on Algorithms*, 1993.

[FK] M. R. Fellows and N. Koblitz, "Fixed-Parameter Complexity and Cryptography," in *Proceedings of the Tenth International Conference on Algebraic Algorithms and Error-Correcting Codes (AAECC 10)*, Springer-Verlag, Lecture Notes in Computer Science, 1993.

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

[HM] F. Henglein and H. G. Mairson, "The Complexity of Type Inference for Higher-Order Typed Lambda Calculi." In *Proc. Symp. on Principles of Programming Languages (POPL)* (1991), 119-130.

[KT] P. G. Kolaitis and M. N. Thakur, "Approximation Properties of NP Minimization Classes," *Proc. 6th Structure in Complexity Theory Conference* (1991), 353-366.

[Le] H. R. Lewis, "Complexity Results for Classes of Quantificational Formulas," *J. Computer and Systems Sciences* 21 (1980), 317-353.

[PY] C. H. Papadimitriou and M. Yannakakis, "On the Complexity of Computing the V-C Dimension," *Proccedings of the 1993 IEEE Conf. on Structure in Complexity Theory*, (1993).