# Advice Classes of Parameterized Tractability [*]

Liming Cai and Jianer Chen [†]
Department of Computer Science
Texas A & M University
College Station, TX 77843-3112

Rodney G. Downey [‡]
Department of Mathematics
Victoria University
Wellington, New Zealand

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

20 July 1993

**Abstract**

Many natural computational problems have input consisting of two or more parts, one of which may be considered a *parameter*. For example, there are many problems for which the input consists of a graph and a positive integer. A number of results are presented concerning parameterized problems that can be solved (uniformly with respect to the parameter) in complexity classes below $P$, given a single word of advice for each parameter value. Different ways in which the word of advice can be employed are considered, and it is shown that the class $FPT$ of *tractable* parameterized problems (the parameterized analog of $P$) has interesting and natural internal structure.

[*]The authors thank Ashish Karkare for pointing out an error in an earlier version of the paper.

[†]Cai is supported by an Engineering Excellence Award from Texas A & M University, and Chen by NSF Grant CCR-9110824.

[‡]Research supported in part by a grant from the Victoria University IGC, by the United States / New Zealand Cooperative Science Foundation under grant INT 90-20558, and by the Mathematical Sciences Institute at Cornell and Cornell University. Email address: downey@math.vuw.ac.nz.

[§]Research supported in part by the National Science and Engineering Research Council of Canada, and by the United States National Science Foundation under grant MIP-8919312. Email address: mfellows@csr.uvic.ca.

## 1 Introduction

In a series of earlier papers, the second two authors have developed a theoretical framework for studying parameterized computational complexity [DF1-4]. The theory is motivated by the many concrete applications of parameterized problems, and by the fact that for many of these problems useful applications may involve only a small range of parameter values. The following are some examples of well-known parameterized problems.

VERTEX COVER
*Instance:* A graph $G = (V, E)$.
*Parameter:* A positive integer $k$.
*Question:* Is there a set of vertices $V' \subseteq V$ of cardinality at most $k$, such that for every edge $uv \in E$, either $u \in V'$ or $v \in V'$?

FEEDBACK VERTEX SET
*Instance:* A graph $G = (V, E)$.
*Parameter:* A positive integer $k$.
*Question:* Is there a set of vertices $V' \subseteq V$ of cardinality at most $k$ such that $G - V'$ is acyclic?

$k$-LEAF SPANNING TREE
*Instance:* A graph $G = (V, E)$.
*Partameter:* A positive integer $k$.
*Question:* Is there a spanning tree of $G$ with at least $k$ leaves?

MINOR TESTING
*Instance: A graph $G$*
*Parameter: A graph $H$*
*Question: Is $G \geq_m H$?*

DOMINATING SET
*Instance:* A graph $G = (V, E)$.
*Parameter:* A positive integer $k$.
*Question:* Is there a set of vertices $V' \subseteq V$ of cardinality at most $k$ such that for every vertex $u \in V$, there is an edge $uv \in E$ for some vertex $v \in V'$?

All of these problems are $NP$-complete in general (which tells us nothing about their fixed-parameter complexity), and all of these except DOMINATING SET can be solved in time $f(k)n$, that is, in linear time for each fixed parameter value.

For DOMINATING SET the simple brute-force algorithm that examines all vertex sets of size $k$ in time $O(n^{k+1})$ has not been improved upon. The best known algorithm for the well-known problem INDEPENDENT SET (or equivalently, CLIQUE, see [GJ]) is only slightly better [NT]. Thus, the difference between fixed-parameter tractability and apparent intractability appears to qualitatively resemble the difference in complexity behavior that we commonly observe when contrasting problems in $P$ and problems which are $NP$-complete.

**Definition.** A *parameterized problem* is a set $L \subseteq \Sigma^* \times \omega$ where $\Sigma$ is a fixed alphabet. We define the $k$th *slice* of a parameterized problem $L$ to be $L_k = \{x : \langle x, k\rangle \in L\}$.

From this definition we may frame our notion of fixed parameter tractability at various levels of uniformity

**Definition.** (i) A parameterized problem $L$ is *fixed-parameter tractable* if there is an algorithm to decide whether $\langle x, k\rangle$ is a member of $L$ in time $f(k)n^\alpha$ for some function $f : \omega \to \omega$, and for a constant $\alpha$ that is independent of the parameter $k$. $FPT$ denotes the class of fixed-parameter tractable problems.

(ii) If $f$ in (i) above is recursive then we say that $L$ is *strongly* FPT.

(iii) Finally we say that $L$ is $nonuniformly\ FPT$ if there is a sequence of algorithms $A_k$ for $k \in \omega$ such that for each $k$, $A_k$ decided membership of $L_k$ and runs in time $O(|x|^\alpha)$ on input $\langle x, k\rangle$.

We remark that it is easy to demonstrate by diagonalization that the flavours of $FPT$ are all distinct. Furthermore, there are a number of known natural examples of problems that seem to lie in each class. Obviously (ii) is the ideal class for practical problems and is the class we concentrate upon in this paper

In the previous papers of this series, we have introduced a completeness theory with which to address the apparent fixed-parameter intractability of problems such as DOMINATING SET and INDEPENDENT SET, and have studied the structure of parameterized complexity classes and reducibilities [DF1-6,DS]. For applications of this theory to concrete problem domains such as cryptography and computational biology, see [DF6,FHW,FK]. The issues raised in the study of parameterized complexity seem to have very wide-ranging practical and theoretical significance.

Although it is possible (and interesting from a structural viewpoint) to define several different notions of reducibility [DF5], the following definition serves for most applications of the theory.

**Definition.** A parameterized problem $L_1$ *reduces* to a parameterized problem $L_2$ if there is an algorithm transforming $\langle x, k\rangle$ into $\langle x', g(k)\rangle$ in time $f(k)|x|^\alpha$, so that $\langle x, k\rangle \in L_1$ if and only if $\langle x', g(k)\rangle \in L_2$, where $f$ and $g$ are arbitrary functions $f, g : \omega \to \omega$ and $\alpha$ is a constant independent of $k$. We say that the reduction is *strong* if additionally the functions $f$ and $g$ are recursive.

We remark that all known concrete reductions are strong ones.

In [DF1,2] the $W$ hierarchy

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

is defined and studied. This hierarchy is based on the logical depth needed to describe parameterized problems in terms of circuits. It is shown, for example, that INDEPENDENT SET (equivalently, CLIQUE) is complete for $W[1]$ [DF3], and that DOMINATING SET is complete for $W[2]$ [DF2]. (For a compendium of the many known completeness and hardness results see [DF2].)

We conjecture that the above hierarchy of parameterized problem classes is proper. If $P = NP$ then $FPT = W[P]$. Conversely, if $FPT = W[P]$ then a *quantitative* version of the $P \neq NP$ conjecture fails [ADF].

Since for each fixed parameter value, each of the problems in the class $W[P]$ is soluable in polynomial time, this theory addresses in some sense (i.e., with parameters fixed) complexity issues *inside of* $P$. Alternatively, one may view the larger issue as regarding limited amounts of nondeterminism. For related studies addressing these issues see [BG,CC,PY,Re].

In this paper, we focus on some natural issues concerning the structure of FPT. A closer inspection of the above examples reveals an apparent qualitative distinction about the manner in which these problems are fixed-parameter tractable. Consider the following results.

**(1.1) Theorem.**
(i) (S. Buss [BG]) VERTEX COVER *is soluable in time* $O(n + k^k)$.
(ii) (Downey and Fellows [DF7]) FEEDBACK VERTEX SET *is soluable in time* $O((2k + 1)^k \cdot n^2)$.
(iii) (Robertson and Seymour [RS1]) MINOR TESTING *can be solved in time* $O(f(k)n^3)$ *where* $f(k)$ *is "approximately"* $500^{k^2}$ *and* $k = |H|$.

We were motivated towards the present study by the apparent distinction between (i) of (1.1), and (ii) and (iii). The "additive" form of (i) results from a general method of parameterized algorithm design that can be viewed as based on ($k$-uniform) polynomial-time recognizability with the help of a single word of *advice* for each parameter value $k$ (where the advice is computable, but not necessarily efficiently). There are strong analogies between this study in parameterized complexity and the advice classes introduced by Karp and Lipton in the classical setting [KL].

The classical advice class $P/poly$, for example, is concerned with languages that can be recognized in polynomial time when we are provided, for input of size $n$, a word of advice

$w_n$, where the length of $w_n$ is bounded by some polynomial $q(n)$. This is a reasonable computational extension of $P$, because we may be able to *pre-compute* (perhaps expensively, but only once) the words of advice $w_n$ for the input sizes $n$ of interest. The reasonable "feasibility" requirement imposed in the definition of $P/poly$ is that the words of advice be polynomially succinct.

Computational feasibility is addressed in parameterized complexity by a focus of the fact that for many parameterized computational problems, a small range of parameter values may have important applications. Thus, for example, for $k \leq 10$ a running time of $2^k n$ would be quite acceptable, as would a word of advice of size $2^k$. This point of view concerning parameterized feasibility, just as with the feasibility claims of $P$, is, of course, essentially qualitative. That is, a parameterized complexity time bound of $2^{2^{2^k}} n$ would be about as "useful" as a polynomial running time bound of $n^{1000}$.

In our parameterized analog of $P/poly$, which is motivated by the natural examples of *additive* fixed-parameter tractability (such as VERTEX COVER), we have the situation where the advice $w_k$ allows us to solve the parameterized problem uniformly (i.e., by a single algorithm that accesses this advice), for any parameter $k$, and for *all* input sizes, in time bounded by a polynomial (independent of $k$). (c.f. The Main Definition below where $g(|x|)$ represents the cost independent of $k$ and $w(k)$ is the advice.) The only requirement we have on the advice is that it is finite for each parameter value $k$.

As in the motivating natural examples for this study, we consider that in solving any given instance, it is not necessary to consider *all* of the advice. For this reason, we model the advice formally as a *finite language.* Furthermore there is nothing special here about $P$, we can apply the advice idea to any complexity class. In discussing space-bounded complexity classes we employ the standard model of a Turing machine with a read only two way input tape, a write only output tape and a work tape (where the space complexity of a computation is measured).

**Definition.** (i) Let $\mathbf{C}$ be a class of functions representing time (space) resource bounds. We say that a (parameterized) language $L$ is $\mathbf{C} +$ *advice* if there is a function $g \in \mathbf{C}$, an oracle Turing machine $\Gamma$, and a function $w : \omega \to \Sigma^*$ (the *advice* function) such that

$$\langle x, k \rangle \in L \text{ iff } \Gamma^{w(k)}(\langle x, k \rangle) \text{ accepts,}$$

and futhermore for all $k$, $x$, the running time (space) of $\Gamma^{w(k)}(\langle x, k \rangle)$ is $\leq g(|x|)$.

(ii) If $w$ is recursive then we say that $L$ is in $uniform$ $\mathbf{C} +$ *advice.*

The advice $w(k)$ often is a table of hard instances for the parameter $k$. The following lemma illustrates this nicely.

**(1.2) Lemma.** (i) *Suppose that* $g \geq \log |x|$ *with* $g \in \mathbf{C}$*, and* $h$ *are functions, and* $\Phi$ *is a*

*Turing machine with $L$ a language so that for all $x,\ k,$*

$$\langle x,k\rangle \in L \text{ iff } \Phi \text{ accepts } \langle x,k\rangle,$$

*and $\Phi(\langle x,k\rangle)$ runs in time (space) $g(|x|)+h(k)$. Then $L$ is in $\mathbf{C}+advice$.* If $k$ is recursive then $L$ is in *uniform* $\mathbf{C}+advice$.
(ii) *Furthermore the converse holds for space bounds provided the witness function is recursive. That is, if $\mathbf{C}$ is a class of space bounds and $L$ is in $\mathbf{C}+advice$ with $w(k)$ computable from $k$, then there exist $\Phi,\ g,$ and $h$ as in (i).*

**Proof.** (i) Observe that $2g(|x|)>g(|x|)+h(k)$ for almost all $x$, and $2g(|x|)$ is in $\mathbf{C}$ as $\mathbf{C}$ represents a complexity class. It follows that we can emulate $\Phi$ by a an oracle machine $\Gamma$ running in time (space) $2g(|x|)$ with an oracle consisting of a lookup table for the values $x$ with $g(|x|)+h(k)\geq 2g(|x|)$, together with the value of $x$ where $2g(n)>g(n)+h(k)$ for all $n\geq |x|$. Note that if $k$ is recursive then this can all be done effectively from $g$.
(ii) In the space bound case, we can write in the memory all the oracle answers corresponding to the advice $w(k)$. Then we can emulate the machine $\Gamma^{w(k)}$ by accessing this precomputed advice whenever an oracle question would be asked in $\Gamma$. This only costs a finite additional cost of $h(k)$ since the advice now written on the tape can be addressed as many times as we like, because we are dealing with space. $\square$.

For illustration, note that (1.1) shows that VERTEX COVER is in $P+advice$ whereas one the face of it, FEEDBACK VERTEX SET and MINOR TESTING are only in $FPT$. In the same spirit as the previous definition we could define the class of $SLICEWISE$ $\mathbf{C}$ as those languages $L$ accepted in time (space) $f(k)g(|x|)$. As with the other definitions we append the adjective "uniform" if additionally, $f$ is recursive. Under this definition, $FPT$ is $SLICEWISE\ P$.

Surprisingly, for some complexity classes $\mathbf{C}$ there is no distinction.

**(1.3) Theorem.** (i) *strong $FPT=uniform\ P+advice(=uniform\ SLICEWISE\ P)$.*
(ii) *$nonuniform\ FPT=P+advice$.*

**Proof.** (i) Let $L\in strong\ FPT$. Then there is a procedure $M$, a constant $c$ and a recursive function $f$ such that $\langle x,k\rangle\in L$ iff $M(\langle x,k\rangle)$ accepts, and $M$ runs in time $f(k)|x|^c$. Consider the following new machine $N$ accepting $L$. This machine runs in time $|x|^{c+1}+h(k)$, where $h$ is a function to be described. The point is that we know that $f(k)|x|^c$ is dominated by $|x|^{c+1}$ and so we can compute a string $x=0^n$ for some $n$ by which $|x|^{c+1}>2f(k)|x|^c$. Once this $n$ is known we can write all the values for $L(\langle x,k\rangle)$ for $|x|\leq n$ in a table $T$. (This is the advice.) The algorithm $N$ first computes $n$ and then constructs the table $T$. It then looks at the length of $x$ in the input $\langle x,k\rangle$. If the length exceeds $n$ then the machine emulates $M(\langle x,k\rangle)$. Otherwise, the machine $N$ uses table lookup to give the desired answer. Clearly from $f(k)$, $c$, and $M$ one can compute a value $h(k)$ so that the running time for this procedure is bounded by $|x|^{c+1}+h(k)$. Now we can use Lemma (1.2).

Conversely, if $L \in uniform\ P + advice$, then there is an oracle machine $M$ running in time $O(n^c)$ with recursive advice $w_k$ for each $k$. Now simply let $f(k)$ be any function sufficiently large that we can compute all the values of the oracle $w_k$. Clearly $M$ can be emulated by a procedure with no oracle but running in time $f(k)|x|^c$.
Now (ii) is quite similar. $P + advice$ is clearly in $nonuniform\ FPT$ by the reasoning above, since we need only construct $f$ from $w_k$. Conversely, if $L \in nonuniform\ FPT$ then the advice we need will consist of the information of (i), as well as the index of the procedure $A_k$ used to compute $L_k$. □

The proof of Theorem (1.3) depends little on $\mathbf{C} = P$; only on the appropriate closure properties of $P$. For instance, the following can also be easily shown.

**(1.4) Theorem.**
(i) (*uniform*) *SLICEWISE POLYLOGSPACE* = (*uniform*) *POLYLOGSPACE* + *advice*.
(ii) (*uniform*) *SLICEWISE PSPACE* = (*uniform*) *PSPACE* + *advice*.

Theorem (1.3) essentially says that, assuming that $P \neq NP$, for $NP$ complete problems whose parameterized versions are in $FPT$, hard instances can only be found when the size of the paramameter is close to that of the object under consideration, and therefore being $FPT$ yields real qualitative insight into the distribution of hard instances.

In the present paper, we shall study a couple of natural subclasses of $FPT$ for which the collapse of (1.3) not occur. In particular, in §2 we will look at the class $LOGSPACE$+*advice* and to a lesser extent, related classes such as $NLOGSPACE + advice$. Several examples are given covering certain basic techniques commonly used to demonstrate membership of $FPT$. In §3 we shall briefly look at some structural aspects of the notions introduced and finally in §4 we will suggest some conclusions and mention some open questions.

## 2 *LOGSPACE + advice* and Related Classes

In this section we (mainly) look at some problems that are in the class *uniform LOGSPACE*+ *advice*. The simplest method of all for showing that a language is in (*uniform*) *LOGSPACE*+ *advice* is to show that the problem is "essentially finite". We give one example. This example uses the "dynamic programming" approach used, for instance, in showing that problems have pseudopolynomial time algorithms (see e.g. [GJ,§4.2, for PARTITION), and are therefore $FPT$.

**(2.1) Theorem.** *The following problem is in uniform LOGSPACE + advice*
FIXED SUBSET SUM (*for positive integers*)
*Instance: A set B*= $\{a_1, ..., a_n\}$ *with "sizes"* $s(a_i) \in \omega$

*Parameter: k*
*Question: Is there a subset $B'$ of $B$ with $\sum_{a_i \in B'} s(a_i) = k$?*

**Proof.** Reading the input make a table of the sizes $s \leq k$ occurring in $B$ together with their frequency $f(s)$ until $f(s) \times s$ exceeds $k$. Check all possibilities see if $k$ is attainable. This method actually shows that the problem is in *constant space + advice*. $\Box$.

A similar argument shows that BOUNDED SUBSET SUM is in $uniform\ LOGSPACE+$ *advice*. For this problem, we ask if there is a set of elements of a set $B$, the sum of whose (positive) sizes equals a given one $S$ given that $s(a) \leq k$ for all $a \in B$. The parameter here is $k$. A variation on this problem is to allow the sizes $s(a_i) \in Z$, that is possibly negative. In that case a the parameterized problem becomes $NP$-complete. (Reduction from SUBSET SUM. Take an instance of SUBSET SUM, $\{a_1, ..., a_n\}$ which we need to sum to $S$. Let $s(b_i) = s(a_i) - S + k$, and consider the problem that asks if there is a subset $A$ of $\{1, ..., n\}$ with $\sum_{i \in A} s(b_i) = k$.) Another related problem is the following:

POST CORRESPONDENCE
*Input:* Sets of words $A = \{w_1, ...w_n\}$ and $B = \{u_1, ..., u_m\}$.
*Parameter: k*
*Question:* Do there exist $k$ indices $\{i_1, ..., i_k\}$ such that the corresponding elements of $A$ and $B$ have equal concatenations. That is, $w_{i_1}\hat{}...\hat{}w_{i_k} = u_{i_1}\hat{}...\hat{}u_{i_k}$?

**(2.2) Theorem.** *(i)* POST CORRESPONDENCE *is* $W[1]$- *complete.*
*(ii) Over a unary alphabet,* POST CORRESPONDENCE *is in* $LOGSPACE$.

The proof of (i) relies on gadget design and will appear elsewhere ([CCDF]). The $LOGSPACE$ tractability of (ii) follows by the realization that one only needs to check to see if the lenghts line up. This can be checked by lexicographic order on the lengths and then systematic search.

A common method of demonstrating that a parameterized problem $L$ is fixed-parameter tractable is to give a method for "reducing", in time bounded by some polynomial $q(|x|, k)$, a problem instance $I = \langle x, k\rangle$ to an equivalent "small" instance $I' = \langle x', k'\rangle$, where by "small" we mean that both $|x'|$ and $k'$ are bounded by some recursive function $f(k)$ of the parameter $k$. This algorithm design strategy is termed *the method of reduction to a problem kernel* and several examples are given in [DF6] and [DF7]. This method immediately yields that $L \in$ $uniform\ P$+*advice*, with the advice $w_k$ consisting of an exhaustive analysis of all instances of size at most $f(k)$. If the equivalent small instance $\langle x', k'\rangle$ can be constructed in small space, then we may be able to further show that $L$ belongs to ($uniform$) $LOGSPACE$ + *advice* or ($uniform$) $SLICEWISE\ LOGSPACE$. (The latter may be the best we can do; for instance for $I$ and $I'$ graphs, in the reduction of $I$ to $I'$ we often need $I'$ to be described by something like $f(k)$ many vertices of $I$ and thus may require $f(k)log(|I|)$ space to describe

$I'$). A good example of this method comes from analysing proofs that VERTEX COVER is in $FPT$.

**(2.3) Theorem.** VERTEX COVER *is in uniform $LOGSPACE + ADVICE$*

**Proof (1).** Our proof uses a modification of an algorithm of Sam Buss. Fist observe that any vertex of degree $> k$ must belong to any vertex cover of a given $G$. So, firstly count the number of vertices in $G$ of degree $> k$, until we stop with a value $p \leq k$ or we exceed $k$ in which case we declare that there is no $k$ element vertex cover. In the former case, let $m = k - p$. Now try to compute the number $q$ of vertices that are not covered by a vertex of degree $> k$. We do this until either we find that there are $> m(k+1)$ such vertices, in which case we reject, or we discover that there are $\leq m(k+1)$ vertices corresponding to edges with no high degree vertex covering them. (The bound here is generated by the fact that a simple graph with a $m$-element vertex cover and all degrees bounded by $k$ has no more than $m(k+1)$ vertices.)

Finally, if we are in the case that there are at most $m(k+1)$ vertices not covered by high degree vertices, we need to do a complete check. We need to do this in very small space. What we do is build an isomorphic copy of the induced subgraph $H$ of $G$ generated by the vertices of edges not covered by those of high degree. The idea is to use the natural "intrinsically definable" lexicographic order on the uncovered vertices to induce the isomorphism. Thus we generate to incidence array of our copy $H'$ of $H$ by first locating the lexixographically least uncovered vertex. Now to generate the first row of the $q \times q$ incidence array, we look at the vertices of $G$ in lexicographic order, and put an entry if the one presently being considered is uncovered. The entry is of course dependent upon whether the first uncovered vertex is connected to the current one. We then increment our counter to the next entry of the first row, until we fill in all $q$ entries. We continue this process inductively, until we determine the whole $q \times q$ matrix. Now in constant space we can check if $H'$ has a $m$ element vertex cover. If not then $G$ does not have a $k$ element one. If $H'$ has such a vertex cover then $G$ has a $k$-element one generated by the $p$ elements of degree $> k$ and the one induced by lifting the $m$ element one in $H'$. □

Before we continue with further examples, we mention that another method of proving fixed parameter tractability is to use what Downey and Fellows[DF7] call the *method of bounded search trees.* This method relies on the fact that for some problems classical intractability seems to be derived from the fact that witnesses can be very large and for a fixed $k$ we can use a pruning process to bound the search. To demonstrate how this commonly used procedure can sometimes be carried out in small space we again reanalyse the VERTEX COVER problem.

**Proof (2).** We base this on the proof from [DF1,2]. Again this uses the intrinsic definability of the construction via lexicographic ordering. Recall that the the proof of [DF1,2] went as follows. Build a tree of $2^k$ possible vertex covers by taking an edge $\langle v_{i_1}, w_{j_1} \rangle$ and beginning a tree of possibilities by placing $v_{i_1}$ on one node and $w_{j_1}$ on the other. Inductively for a node $\sigma$ of the bounded search tree, take an edge not covered by the vertices defined by, $\sigma = v_{q_1}\hat{}...\hat{}v_{q_m}$ and use this to make the children of $\sigma$. The tree has depth $k$ and then we check order if everyone is covered.

We claim that this proof can actually be accomplished in small space. Consider the standard tree $T = 2^k$. We can regard a node $\eta$ of $T$ as "intrinsically" coding a potential vertex cover as follows. Let $\eta = i_1\hat{}...\hat{}i_k$. Then the vertex of $G$ coded by $i_1$ is the lexicographically least vertex of the lexicographically least edge of $G$ if $i_1 = 0$, and the other vertex of the lexicographically least edge of $G$ if $i_1 = 1$. This description can be extended inductively, the children of the $\gamma$ being associated with the lexicographically least edge not covered by the vertices intrinsically defined by $\gamma$. We can easily show by induction there is a $LOGSPACE$ procedure $\Phi(\gamma, n)$ which computes the vertex of $G$ associated with the $\gamma$ at level $n$, by cycling through candidates in lexicographic order. Now to see if all edges are covered by $\eta$ we can again cycle through all the edges of $G$ and for each edge, invoke $\Phi(\eta, j)$ sequentially for each $i \leq k$ until either the edge is found to be covered by a vertex associated with $\eta$, or we reject the potential vertex cover coded by $\eta$. $\square$

The technique of the second proof is rather widely applicable since it describes a more or less canonical method of converting a problem solved by the method of search trees into one soluable in $uniform\ LOGSPACE + advice$. Notice also that it shows the problem LOG VERTEX COVER which asks for a vertex cover of size log of the input size or less is not only soluable in time $O(n^2)$ by the proof of [DF1,2], it is soluable in $LOGSPACE$, since the search tree has height $k = \log(|G|)$. As further illustrations of these ideas, we consider some further problems below.

RESTRICTED ALTERNATING HITTING SET

*Instance:* A collection $C$ of subsets of a set $B$ with $|S| \leq k_1$ for all $S \in C$.
*Parameter:* $\langle k_1, k_2 \rangle$.
*Question:* Does player I have a win in $\leq k_2$ moves in the following game? Players play alternatively and choose unchosen elements, until, for each $S \in C$ some member of $S$ has been chosen. The player whose choice this happens to be wins.

Let $\varphi$ be a formula of propositional logic. We say that $\varphi$ is in $q$-CNF form if $\varphi$ consists of a conjunction of clauses all bounded by $q$ in size. Our standard hard parameterized problem is WEIGHTED 3-CNF SATISFIABILITY, which asks if a given 3-CNF formula $\varphi$ has a weight $k$ (i.e. exactly $k$ literals true) satisfying assignment. A variation on this problem is WEIGHT $\leq k$ 3-CNF SATISFIABILITY which asks for a satisfying assignment of weight $\leq k$ for the given parameter $k$. When there is no a priori bound on the clause size, the second two authors have proven that the problem of determining if a CNF formula has a weight $\leq k$ satisfying assignment is of the same f.p. complexity as determining if a formula has a weight exactly $k$ satisfying assignment. However a bounded search tree argument shows that this is apparently not true for bounded CNF formulae. We have the following.

**(2.4) Theorem.** *The following problems are in $uniform\ LOGSPACE + advice$:*
(i) RESTRICTED ALTERNATING HITTING SET.
(ii) WEIGHT $\leq k$ $q$-CNF SATISFIABILITY *for any fixed $q$.*
(iii) DOMINATING SET FOR PLANAR GRAPHS.

**Proof.** These are all similar. For (i) consider the argument from [ADF2]: It is simplest to consider $k_1 = 2$, the analogue of the $PSPACE$ complete problem ALTERNATING VERTEX COVER. Take an edge $(x, y)$. All vertex covers must include $x$ or $y$. Try each, generating the tree of possibilities. Terminate a branch and put the cover at the leaf if a branch achieves a vertex cover. This gives a tree with at most $k_1^{k_2} = 2^{k_2}$ leaves (corresponding to posible candidates for vertex covers), at most $k_2 2^{k_2}$ vertices, and all size $\leq k_2$ covers must contain a subset occurring at one of the leaves.

Now we select $k_2$ additional vertices of $G$, not occurring at any of the leaves of the tree (we can assume $V$ is large compared to $k_2$, else the problem is easily done). Consider all possible strategies played on the subgraph induced by these at most $k_2 + k_2 2^{k_2}$ vertices. It is easy to see that player I has a winning strategy in $\leq k_2$ moves in $G$ iff he has one in this set of strategies. It is not difficult to see that the above can be performed in $uniform\ LOGSPACE + advice$ for the same reason as VERTEX COVER can.
(ii) is again similar. Let $\varphi$ be any $q$-CNF formula. Again we build a bounded search tree. To see if $\varphi$ has a weight $\leq k$ satisfying assignment, first see if any clause has only unnegated variables. If there is no such clause, answer that $\varphi$ is (weight 0) satisfiable. Otherwise, pick the first such monotone clause and begin to build a tree with first children being the variables of this monotone clause. Each child represents the possibility that we will set this variable to be *true*. Inductively, for a branch $\sigma$ the nodes will represent a collection of variables that we will need to made true. to determine $\sigma$'s children, find a monotone clause $C$ containing no member of $\sigma$. If none exists then output "yes". Otherwise stop when the length of a branch reaches $k$. Now check if any of the branches represent a satisfying assignment for $\varphi$. Again we can use the VERTEX COVER modification to perform this all in small space. Finally, (iii) follows using similar reasoning applied to [DF7, Theorem 2.4]. $\square$.

Another nice example of the problem kernel method is provided by the $k$-LEAF SPANNING TREE PROBLEM. We consider also the following variation:

$k$-LEAF SPANNING FOREST
*Input:* A graph $G$.
*Parameter:* $k$
*Question:* Does $G$ have a $k$-leaf spanning forest? That is, Does there exist a collection of trees $\{T_i : i \in I\}$ such that collectively the trees have at least $k$ leaves and each $T_i$ is a spanning tree for its component of $G$. Thus if $G$ is connected then a spanning forest is simply a spanning tree.

It is not difficult to see the following.

**(2.5) Theorem.** $k$-LEAF SPANNING TREE $is\ in\ uniform\ SLICEWISE\ NLOGSPACE$.

**Proof.** Nondeterministically guess the spanning tree, check that it is a tree, and then check that it spans the given graph $G$ by sequentially checking all the vertices of $G$. □.

We can do rather better than (2.5). We begin our analysis by giving a linear time algorithm for the $k$-LEAF SPANNING TREE problem. In fact our algorithm is $O(n+(2k)^{4k})$, improving a result of Hans Bodlaender who proved that the problem is in $LIN + advice$, that is he proved the problem to be soluable in linear time but with a multiplicative constant depending upon $k$. ([Bo1,2]) We give the argument in full since details were not supplied in [DF7].

**(2.5) Theorem.** (Downey and Fellows[DF7]) $k$-LEAF SPANNING TREE *is soluable in time* $O(n + (2k)^{4k})$

**Proof.** Note that any graph $G$ that is a *yes* instance must be connected. A vertex $v$ is called *useless* if (i) it has neighbours $u$, $w$ of degree 2 and (ii) $v$ has degree 2. We will argue that any sufficiently large graph without useless vertices of degree 2 is necessarily a *yes* instance. Note also that if $G$ has a vertex of degree at least $k$, then $G$ is a *yes* instance.

Say that a useless vertex $v$ is *resolved* by deleting $v$ from $G$ and adding an edge between $u$ and $w$. Let $G'$ denote the graph obtained from $G$ (in linear time) by resolving all useless vertices.

Our algorithm for $k$-LEAF SPANNING TREE is very simply described:
*Step 1.* Check whether $G$ is connected, and whether there is a vertex of degree $\geq k$.
*Step 2.* If the answer is still undetermined, then compute $G'$. If $G'$ has at least $3k(k+1)$ vertices then the answer is *yes.*
*Step 3.* Otherwise, exhaustively analyze $G'$ and answer accordingly, since $G'$ has a $k$-leaf spanning tree if and only if $G$ does.

Our proof that the algorithm is correct employs the following fact.

*Claim.* If $H$ is connected simple graph of order at least $k(k+1)$ and every vertex of $G$ has degree $\neq 2$, then $G$ has a spanning tree with at least $k$ leaves.

To establish the claim, it can be shown by elementary induction that
(2.6) If a tree $T$ has $i$ internal vertices of degree at least 3, then $T$ has at least $i+2$ leaves.

Suppose $H$ satisfies the hypotheses of the above Claim. If $H$ has any vertex of degree $k$ we are done, and hence w.l.o.g. we may presuppose that $H$ has no vertex of degree $\geq k$. Let

$T$ be a spanning tree of $H$ having a maximum number of leaves $l$, and suppose $l \leq k-1$. Now as $T$ has $k(k+1)$ vertices but only $l$ leaves, it must have at least $k(k+1)-(k-3)-(k-1)$ vertices of degree 2. (By (2.6).) Now as the maximum degree in $H$ of any vertex is $k-1$, and

$$k(k+1)-(k-3)-(k-1)-(k-1)(k-1)=k+5,$$

it follows that there is at least $k+5$ vertices of degree 2 in $T$ that have the property that they are not connected to any leaf of $T$. Let $v$ be such a degree 2 vertex of $T$. Regard $T$ as a rooted tree with root $v$. Let $u_1, ..., u_m$ be the children of $v$. Now as $v$ has degree 2 in $T$, and yet degree 3 or more in $H$ it follows that $v$ is adjacent to some vertex $w$ in $H$ which is not a child of $v$ since $H$ is simple, and furthermore $w$ is not a leaf of $T$ by choice of $v$. But now it is possible to change $T$ into a new tree $T'$ with more leaves as follows. $w$ must occur below one of the children, say $u$ of $v$. We can therefore make $u$ a leaf by deleting the edge $(v,u)$ and add an edge $(v,w)$. The point is that as $w$ is not a leaf, this must increase the net number of leaves. This concludes the proof of the claim.

It is now easy to see that the algorithm is correct since the derived graph, $G'$, will satisfy the hypotheses of the claim in step 2 of the algorithm. □

We claim that the above proof can be achieved in *uniform NLOGSPACE + advice*. Certainly, we can check if $G$ is connected in *NLOGSPACE*. Then deterministically, we can count the number of nonuseless vertices, until either that number exceeds $3k(k+1)$ or we get a number $q$ of nonuseless vertices. In the latter case we use "topological" variation of the isomorphism construction we used in the VERTEX COVER (proof (1)) case. In this case we need to generate an $H$ isomorphic to $G'$ in the above, but with a small description. Again we begin by considering the lexicographically least nonuseless vertex $v$. This time we consider its neighbours in $G$ in lexicographic order. Let $u$ be the neighbour under consideration. If $u$ has is nonuseless, put a 1 in position $b$ where $u$ is the lexicographically $b$-th nonuseless vertex. If $u$ is useless traverse to $u'$, $u$'s other neighbour and continue inductively until a nonuseless vertex is found. After all of the neighbours of $v$ have been processed, put 0's in all the entries of the first row that do not already have a 1. This reasoning gives the following result.

**(2.7) Theorem.** (i) $k$-LEAF SPANNING TREE *is in uniform NLOGSPACE + advice.* (ii) $k$-LEAF SPANNING FOREST *is in uniform LOGSPACE + advice.*

**Proof.** (ii) To see this we merely note that the only nondeterministic part of the algorithm outlined in the paragraph preceding (2.6) is the determination of connectivity. □.

We remark that we do not know if Theorem (2.6) (i) can be improved to say that $k$-LEAF SPANNING TREE is in *uniform SLICEWISE LOGSPACE* but we do know any such result is improbable since any such result would imply that CONNECTIVITY for undirected graphs would be in *LOGSPACE*. This is thought to be unlikely. The reader is referred to for instance, Johnson [Jo], page 128.

## 3 The Structure of $FPT$ and Additional Comments

One issue that we have not addressed so far is the structural one generated by our considerations. In the same way that we analyse $P$ under $LOGSPACE$ reductions, we are naturally lead to looking at the structure of $FPT$ under a variety of reductions. Some natural ones that suggest themselves are $(uniform)$ $SLICEWISE$ $LOGSPACE$ and $(uniform)$ $LOGSPACE + advice$ reductions. These are defined as follows.

**Definition.** Let $L_1$ and $L_2$ be parameterized problems. We say that $L_1$ is $SLICEWISE$ $LOGSPACE$ reducible to $L_2$ iff there is a procedure $M$ and a function $f(k)$ such that for all $z \in \Sigma^*$, and all $k \in \omega$,

$$\langle z, k\rangle \in L_1 \text{ iff } M(\langle x, k\rangle) \text{ accepts and runs in space } f(k)\log|z|.$$

If additionally $f$ is recursive then we call the $uniform$.
Simiarly we can define $LOGSPACE + advice$ reductions for $M$ by asking for the existence of an oracle Turing machine $\Gamma$, and an advice function $w$ so that for all $x,\ k$,

$$\langle x, k\rangle \in L_1 \text{ iff } \Gamma^{w(k)\oplus L_2}(\langle x, k\rangle) \text{ accepts},$$

and $\Gamma$ is running in space $\log|x|$.

Note that the reductions are good in the sense that, for instance, if $L$ is in $(uniform)$ $LOGSPACE + advice$ and $L'$ is reducible to $L$ under a $(uniform)$ $LOGSPACE + advice$ reduction, then $L'$ is in $(uniform)$ $LOGSPACE + advice$. (This is easily observed by amalgamating the advice for the reduction with the advice function for membership for $L$.)

Let $A \subseteq \Sigma^*$. Define $N_1(A) = \{\langle z, 0\rangle : z \in A\}$ and $N_2(A) = \{\langle z, k\rangle : z \in A,\ k \in \omega\}$. Note that $N_1(A) \equiv N_2(A)$ for either of the reductions defined above. Also let $B$ be a parameterized problem. We can define the *standardization* of $B$ to be $S(B) = \{\langle\langle z, q\rangle, k\rangle : \langle z, q\rangle \in B \text{ and } q \leq k\}$. Again note that $B$ and $S(B)$ have the same parameterized complexity. Standardization ensures that the structure of a set is smooth in the sense that the $k$'th row codes all the information of the preceding rows. The reader should note that virtually all natural problems are standardized.

**(3.1)Theorem.** *(i) Let $A$ be $P$-complete. Then $N_1(A)$ and $N_2(A)$ are complete for $(strong)$ $FPT$ under $(unifrom)$ $LOGSPACE$+advice reductions. (ii) Furthermore, if $B$ is complete for $FPT$ under $SLICEWISE$ $LOGSPACE$ reductions then for some $k$, for all $k' \geq k$, the $k'$-th row of $S(B)$ is $P$-complete.*

**Proof.** Neither of these are hard. We begin with (i). Let $N = N_1$ and let $A$ be $P$-complete. Let $C \in FPT$. Then there is a procedure $M$ a function $f$, and a constant $c$ such that

$$\langle z, k\rangle \in C \text{ iff } M(\langle z, k\rangle)\text{accepts, and } M \text{ runs in time } f(k)|z|^c.$$

Take a set $W$ accepted in time $|z|^{c+1}$, universal for all sets accepted in time $O|z|^c$. Such a machine can be canonically constructed by waiting to code the $e$-th $O(|z|^c)$ machine untill we are considering those $z$ which are sufficiently long so that $|z|^{c+1}$ exceeds the collective running time for the first $e$ machines. Then we directly code the values for the $e$-th machine into $W$, say via the $e$-th row, putting alll earlier values in a table lookup. Such a reduction is clearly also a $LOGSPACE + advice$ one, and the value of the constant can be computed from an index for the $O(|z|^c)$ problem. Consider each of the rows of $C$ as giving a member of the class of sets accepted in time $O(|z|^c)$. The above construction implies that $C$ is $LOGSPACE + advice$ reducible to $W$. But as $W$ is in $P$, $W$ is $LOGSPACE$-reducible to $A$. It easily follows that $C$ is $LOGSPACE + advice$ reducible to $N(A)$.

(ii) Let $B$ be complete for $FPT$ under $SLICEWISE\ LOGSPACE$ reductions. Let $A$ be $P$-complete. Now $N_2(A)$ is $SLICEWISE\ LOGSPACE$ reducible to $S(B)$. It follows that for some $k$, the $k$-th row of $S(B)$ can, in $O(\log |z|)$ compute $A$. The result follows. □.

From the above it follows that $FPT = SLICEWISE\ LOGSPACE$ iff $P = LOGSPACE$. A number of other structural results concerning the structure of $FPT$ are similarly easily obtainable. We mention a couple below.

**(3.2)Theorem.** (i) $(uniform)$ $LOGSPACE$+*advice* $\neq$ $(uniform)$ $SLICEWISE\ LOGSPACE$. (ii) $(uniform)$ *SLICEWISE LOGSPACE has no problem complete under* $(uniform)$ $LOGSPACE + advice$*-reductions.*

**Proof.** Both of these are easy diagonalization arguments, which we very briefly sketch. The recursive, uniform, cases are easier. Thus we will only look at the more intricate nonuniform case. To see (i), we only need to build a language whose $k$-th slice is accepted in space $f(k)\log |x|$, where we get to build $f$. We need to construct $f(k)$ so that we have enough room to diagonalize. We need the following lemma.

**(3.3) Lemma** *Suppose that L is in* $LOGSPACE$+*advice. Then there exists a witness oracle Turing machine* $\Gamma$ *accepting* $L$ *running in space* $O(\log |x|)$ *which has an advice function* $w$ *recursive in* $\mathbf{0}'$, *the Turing degree of the halting problem.*

**Proof.** Take the oracle machine $\Gamma$ accepting $L$. Define a string $\sigma$ to be acceptable if

$$\forall x, k[\langle x, k\rangle \in L \text{ iff } \Gamma^\sigma(\langle x, k\rangle) \text{ accepts.}]$$

As being acceptable is a $\Pi_1$ condition, it is computable by $\mathbf{0}'$. (Alternatively use the Shoenfield limit lemma to finitely approximate the least acceptable $\sigma$ and this will serve as the relevant advice function.) □.

In view of (3.3), it suffices to meet the following requirements.

$R_{e,n}$ : Either $\exists k(\lim_s(\phi_e(k,s)(=_{\text{def}} \phi_e(k))$ does not exist ),
or $\exists x, k(\Phi_e^{\phi_e(k)}(\langle x,k\rangle)$ does not run in space $n.\log(|x|))$
or $\exists x, k(\Phi_e^{\phi_e(k)}(\langle x,k\rangle) \neq L(\langle x,k\rangle)$.

In the above $\langle \phi_e(\ ,\ ), \Phi_e\rangle_{e\in\omega}$ is a simultaneous enumeration of all pairs consisting of an oracle Turing machine and a partial recursive binary function. We devote the $\langle e,n\rangle$'th slice of $L$ to meeting the requirement $R_{e,n}$. We define $f(\langle e,n\rangle) = n+1$. In view of the proof of lemma (3.3), we can assume that if $\phi_e(\langle e,n\rangle, s) \downarrow \neq \phi_e(\langle e,n\rangle,t)\downarrow$ and $t > s$ then for all $u \geq t$, $\phi_e(\langle e,n\rangle,u) \neq \phi_e(\langle e,n\rangle,s)$.

At stage $s$, on row $\langle e, \langle e,n\rangle\rangle$ with $1^{\langle e,n\rangle} \leq s$ we perform the following action. If we see $\phi_{e,s}(\langle e,n\rangle,t)\downarrow$ in $\leq \log(|s|)$ space and $\Phi_e^{\phi_e(\langle e,n\rangle,t)}(\langle 1^s, \langle e,n\rangle\rangle)\downarrow$ in $\leq (n+1)\log(s)$ space, with $t \leq \log(|s|)$, for the largest such $t$, let

$$L(\langle 1^s, \langle e,n\rangle\rangle) = 1 + \Phi_e^{\phi_e(\langle e,n\rangle,t)}(\langle 1^s, \langle e,n\rangle\rangle).$$

This action will succeed in diagonalizing the *possibility* of $\phi_e(\langle e,n\rangle,t)$ as being a value for $\lim_s(\phi_e(\langle e,n\rangle,s))$. Since either we will thus diagonalize a final value for $\phi_e(\langle e,n\rangle)$, or possible values will change infinitely often, or the running space will never be correct, or finally $\phi_e(\langle e,n\rangle)$ will be partial. In any case we succeed in meeeting the $R_{e,n}$, giving the theorem.
We remark that (ii) is essentially similar. For if we had such a problem $L$, accepted in space $f(k)\log(|x|)$ slicewise, then we could build $L'$ not computable by $L$ in space $2^k f(k)\log(|x|)$ since the additional $2^k$ multiplicative factor gives us plenty of extra space to diagonlize and no finite advice could lift $f(k)$ to $2^k f(k)$. $\Box$.

While the above result provides separation, they do not provide natural examples of problems providing such separations. There are a number of natural candidates for this sort of problem. For instance, the work of Lipton and Zalcstein [LZ] provides such a candidate. In [LZ], Lipton and Zalcstein analysed computational complexity of the word problem for groups. Of course, by the famous work of Boone and of Novikov, we know that in general the word problem, which asks one to decide if a word in over a given alphabet of generators is the identity, is undecidable. Nevertheless, the word problem for certain classes of groups such as one relator groups is decidable, so one is naturally lead to consider its computational complexity. Lipton and Zalcstein proved the following interpreted in our setting.

**(3.4) Theorem.** ([LZ]) *The problem* LINEAR GROUP WORD PROBLEM *(below) is in uniform SLICEWISE LOGSPACE.*

LINEAR GROUP WORD PROBLEM
*Input:* A word $w$ from a linear group (or even semigroup) $G$ presented as a finitely generated group of $k \times k$ matrices over a field $F$ of characteristic zero.
*Parameter: G*
*Question:* Does $w = 1$?

**Proof.** The Lipton-Zalcstein argument is to show that the general problem can be rephrased as the following:
*Input:* A sequence of $k \times k$ matrices $A_1, ..., A_n$ over $Z[x_1, ..., x_m]$ with all entries having degree at most $d$ and coefficients bounded in absolute value by $b$.
*Parameter:* $\langle m, k, d, b \rangle$.
*Question:* Does $A_1 ... A_n = I$?

Lipton and Zalcstein then proved that the rephrased problem can be solved in space $O(\log n)$ where the constant recursively depends only upon $\langle m, k, d, b \rangle$. By definition the problem belongs to *uniform SLICEWISE LOGSPACE.* $\square$.

Before we leave the topic of $P$-completeness, we would like to mention a related result concerning the structure of the $W$-hierarchy. The reader should recall that a problem $A$ is complete for the top level of the hierarchy, $W[P]$, iff the problem of deciding if a formula of propositional logic has a weight $k$ satisfying assignment is f.p. reducible to $A$. (For more details see, e.g., [ADF1,2], [DF1,2].) It has been noticed that all the *natural* $W[P]$-complete problems have the property that they are $P$-complete by the slice. This leads to the *false* conjecture that a problem $A$ is $W[P]$-complete iff there is a $k$ such that for each $k' \geq k$, $A_{k'}$ (the $k'$-th slice of $A$) is $P$-complete. The examples above show that the "only if" part fails. The if part cannot be exactly true as we could use many rows to code a piece of information. However one is lead to the following possibility:

$$\text{(3.5)} \qquad \text{If } A \text{ is } W[P]-\text{complete then for some } k,\ S(A)_k \text{ is } P-\text{complete.}$$

We do not believe that (3.5) holds since it seems to imply something along the lines of $P = LOGSPACE$, although the issue is still not totally clear.

## 4 Conclusions and Open Problems

In this paper we have demonstrated that the parameterized view of the world yields insight into the structural and algorithmic complexity of various concrete problems. In particular we have examined how fixed parameter tractability can be viewed as an extension of the notion of advice computations introduced by Karp and Lipton, where a *finite* piece of information is enough to uniformly solve all of the instances of a particular parameter value. This in turn yields qualitative insight into the distribution of hard instances of various intractable problems. We have also shown that standard techniques (such as those surveyed in [DF7]) of demonstrating parameterized tractability can often give much more information regarding the precise structural complexity of the problem at hand. In this paper we have concentrated upon *LOGSPACE* but there is no reason that we could have studied similar ideas for, say, *NC* giving classes such as *NC+advice* and *SLICEWISE NC*. We remark that not even all problems soluable by, say, the search tree method seem to be classified in, e.g., *SLICEWISE*

$LOGSPACE$. A notable example here is FEEDBACK VERTEX SET which is shown to be soluable in time $O((2k+1)^k n^2)$ in Downey-Fellows [DF7] by the search tree method. (In [DF1] and [Bo] using minor testing methods and bounded tree width arguments it is shown to also be soluable in time $O((17k^4)!n)$.) The relevant tree does not seem amenable to the compression outlined above. Another noteworthy open question here is whether MINOR TESTING (where the parameter is a fixed graph) is one of our $LOGSPACE$-advice classes. A method of demonstrating that this is unlikely would be to show that the problem is $P$-complete for some parameter graph $H$. It would also be nice to know the space complexity generated by the well quasi-order methods of Abrahamson-Fellows and the monadic second order logic methods of Courcelle's theorem. (See e.g. [AF].)

Finally, we believe that the analysis above is sufficiently close to "practical complexity" that we could analyse the situation for subpolynomial time bounds. That is, while we know that $FPT = P + advice$, nevertheless it is clear that some of the $FPT$ algorithms have quite a different structure than others. While this does not matter at the level of $P$, it does seem to matter at say "linear time". The trouble is that such classes are not robust. In this case the we could fix the model as, say, a RAM and then it would make emminent sense to look at $DTIME_{RAM}(O(n) + advice)$ and $DTIME_{RAM}(SLICEWISE\ O(n))$ for a whole class of concrete problems.

## References

[ADF1] K. Abrahamson, R. Downey and M. Fellows,“Fixed-parameter Intractability II,” in *Proc. 10th Symposium on Theoretical Aspects of Computer Science*,(1993) 374-385.

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

[AF] K. Abrahamson and M. Fellows,“Finite Automata, Bounded Treewidth, and Well-Quasiordering,” in *Graph Structure Theory* (ed. N. Robertson and P. Seymour) Contemporary Mathematics Vol 147, American Mathematical Society, Providence, RI (1993) 539-564.

[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.

[Bo1] H. L. Bodlaender, “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.

[Bo2] H. L. Bodlaender, “On Disjoint Cycles,” Technical Report RUU-CS-90-29, Dept. of Computer Science, Utrecht University, Utrecht, The Netherlands, August 1990.

[BFW] H. L. Bodlaender, M. R. Fellows and T. Warnow, “Two Strikes Against Perfect Phylogeny,” in: W. Kuich (editor), *Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP'92)*, Springer-Verlag, Berlin, Lecture Notes in Computer Science, volume 623, pp. 273-283.

[BG] J. F. Buss and J. Goldsmith, “Nondeterminism Within $P$,” *SIAM J. Comp.*, Vol. 22, (1993) 560-572.

[CC] L. Cai and J. Chen, “On the Amount of Nondeterminism and the Power of Verifying,”to appear in *Proc. Mathematical Foundations of Computer Science,* (1993) 'Lecture Notes in Computer Science, Springer-Verlag, Vol. 711 (MFCS'93) (1993) 311-320.

[CCDF] L. Cai, J. Chen, R. Downey and M. Fellows,“The Complexity of Short Computation and Factorization,” to appear, *Proceedings Sacks Conference.* (special issue, Archive for Mathematical Logic)

[Co] B. Courcelle,“Graph Rewriting: an Algebraic and Logical Approach,” in [Le1], Ch 5.

[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. Comput.*.

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

[DF4] R. G. Downey and M. R. Fellows,"Fixed Parameter Intractability (Extended Abstract)," *Proc. Structure in Complexity Theory, Seventh Annual Conf.*, (1992) 36-50.

[DF5] R. G. Downey and M. R. Fellows, "Fixed Parameter Tractability and Completeness III: Some Structural Aspects of the $W$-Hierarchy," in *Complexity Theory: Current Research* (Ed. K. Ambos-Spies, S. Homer, and U. Schoning) Cambridge University Press (1993) 166-191.

[DF6] R. G. Downey and M. R. Fellows, "Parameterized Complexity," monograph in preparation.

[DF7] R. G. Downey and M. F. Fellows, "Parameterized Computational Feasibility," in *Feasible Mathematics II* (ed. P. Clote and J. Remmel) Birkhauser, Boston, 1995, 219-244.

[FHW] M. R. Fellows, M. T. Hallett and H. T. Wareham, "DNA Physical Mapping: Three Ways Difficult," in *Proceedings of the First European Symposium on Algorithms* (1993),T. Lengauer (ed.) Lecture Notes in Computer Science no. 726. Springer-Verlag; Berlin. pp. 157-168. .

[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.

[FL2] 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.

[FL3] 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.

[Jo] D. S. Johnson,"A Catalogue of Complexity Classes,"in [Le1], 68-161.

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

[Le1] J. van Leeuwen,"Handbook of Theoretical Computer Science, Vol 1," (ed.) Elsevier Science Publicationers B.V., 1990.

[Le2] J. van Leeuwen,"Graph Algorithms,"in [Le1], 527-632.

[LZ] R. Lipton and Y. Zalcstein,"Word Problems Solvable in Logspace," *J. Assoc. Comput. Mach.* 24 (1977) 522-526.

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

[Re] K. Regan, "Finitary Substructure Languages, With Application to the Theory of $NP$-Completeness," Proc. 4th Structure in Complexity Theory Conf. (1989), 87-96.

[RS3] N. Robertson and P. D. Seymour, "Graph Minors XIII. The Disjoint Paths Problem," to appear.

[RS4] N. Robertson and P. D. Seymour, "Graph Minors XV. Wagner's Conjecture," to appear.