Parameterized Approximation Problems

Rodney G. Downey[1], Michael R. Fellows[2], and Catherine McCartin[3]

[1] Victoria University, Wellington, New Zealand,
Rod.Downey@mcs.vuw.ac.nz
[2] The University of Newcastle, Calaghan, Australia,
mfellows@cs.newcastle.edu.au
[3] Massey University, Palmerston North, New Zealand,
C.M.McCartin@massey.ac.nz

Abstract. Parameterized complexity is fast becoming accepted as an important strand in the mainstream of algorithm design and analysis. Up until now, most of the work in the area has focussed on exact algorithms for decision problems. The goal of the present paper is to apply parameterized ideas to approximation. We begin exploration of *parameterized approximation problems*, where the problem in question is a parameterized decision problem, and the required approximation factor is treated as a second parameter for the problem.

1 Introduction

Parameterized complexity is fast becoming accepted as an important strand in the mainstream of algorithm design and analysis, alongside approximation, randomization, and the like. It is fair to say that most of the work in the area has focussed on exact algorithms for decision problems. On the other hand it is clear that parameterized ideas have applications to many other questions of algorithmic design. For example, in [6] and [8] the ideas have been applied to counting problems and in [5], [8] the ideas were applied to online problems.

The goal of the present paper is to apply the ideas to approximation. Already we have seen that there are close ties between classical approximation and the theory of parameterized complexity.

For example, the following is now well-known. We can define a classical optimization problem to have an *efficient P-time approximation scheme* (EPTAS) if it can be approximated to a goodness of $(1+\epsilon)$ of optimal in time $f(1/\epsilon)n^c$ where c is a constant. If we set $k = 1/\epsilon$ as the parameter, and then produce a reduction to the PTAS from some parametrically hard problem, we can, in essence, demonstrate that no such EPTAS exists [1], [3].

In this paper, we begin exploration of *parameterized approximation problems*, where the problem in question is a parameterized decision problem, and the required approximation factor is treated as a second parameter for the problem. Consider the following 'classic' parameterized problem:

k-Independent Set
Input: A graph $G = (V, E)$

Parameter: k, a positive integer
Output: An independent set $V' \subseteq V$ for G of size at least k, or 'NO' if none such exists.

How might we define a problem that provides an 'approximate' solution to this problem? Here are two possibilities which we will consider in this paper. In both cases we relax our requirements by introducing a 'gap' between YES and NO solutions to the problem. In the first case the gap size is additive in the approximation parameter , in the second case the gap size is multiplicative in the approximation parameter.

Add-Approx k-Independent Set
Input: A graph $G = (V, E)$
Parameters: k, c, positive integers
Output: 'NO', asserting that no independent set $V' \subseteq V$ of size $\geq k$ for G exists, or an independent set $V' \subseteq V$ for G of size at least $k - c$.

Mult-Approx k-Independent Set
Input: A graph $G = (V, E)$
Parameters: k, c, positive integers
Output: 'NO', asserting that no independent set $V' \subseteq V$ of size $\geq k$ for G exists, or an independent set $V' \subseteq V$ for G of size at least k/c.

The first parameterized approximation question is the the parameterized version of absolute approximability. The question is, are there parameterized algorithms to solve the above questions, in spite of our belief that there is no such algorithm for the exact problem? More generally, we will be considering the following class of questions. Our setting will be languages $L \subseteq \Sigma^* \times \Sigma^*$ We state the following for maximization problems, the analogous definition would work for minimization.

$g(k)$-Approximation
Input $\langle x, k\rangle$
Parameter k, g
Output 'NO' asserting $\langle x, k\rangle \notin L$ or $\langle x, k'\rangle \in L$ for some $k' \leq g(k)$.

As stated, the approximation problem above is for the version where we ask for the certificate $\langle x, k'\rangle$. There could also be a version where we simply ask for the 'YES' asserting that some such certificate exists. Since all the practical examples are self-reducible, we will get the certificates from the problem for free.

Notice that we can take a arbitrarily 'bad' language $L = \{\langle x, 2k\rangle : k \in \mathbb{N}\}$ and consider $L' = L \cup \{\langle x, 2k + 1\rangle : x \in \Sigma^* \wedge k \in \mathbb{N}\}$. Then, in spite of the fact that $\langle x, m\rangle \in L'$ is as bad as you like, we can always have an approximation with $g(m) = m + 1$. The problem is that the odd parameters give *no* information. On the other hand, we believe that some natural problems are sufficiently well-structured so that, for certain functions g, the approximation schemes should give enough information so as to be able to solve the original problem.

In this paper we will consider different kinds of functions g. We first consider $g(k) = k + c$, absolute additive approximation. We demonstrate that for many

of the basic $W[1]$-hard problems no such approximation scheme can exist unless $W[1] = FPT$. These problems include k-Independent Set, k-Clique and k-Step Turing Machine Acceptance. We also demonstrate that no such approximation scheme can exist for k-Dominating Set unless $W[2] = FPT$.

Next we consider multiplicative and other values values for g. Notice that for instance, Bin Packing parameterized by the number of bins, has (by First Fit) a natural approximation with $g(k) = 2k$, say. (See Garey and Johnson [7].) Thus there are natural problems with such multiplicative parameterized approximation schemes.

On the other hand we show that there exist problems where there is no approximation scheme for *any* function $g(k)$ unless $W[2] = FPT$. One example of this phenomenon is k-Independent Dominating Set . That is, for any computable function $g(k) \geq k$, there is no algorithm which either asserts that there is no independent dominating set of size $\leq k$ for a given graph G, or otherwise asserts that there is one of size $\leq g(k)$. We call such problems *completely inapproximable.*

Up to this time there is no literature on this kind of approximation. The idea was introduced by Downey and Fellows in [4] for Dominating Set. As we were about to submit this paper, we were sent a copy of a new paper by Cai and Huang [2] also studying the same kind of problems; their results being quite complementary to ours. Interestingly, the fundamental problem we wished to classify, Dominating Set, remains open.

2 Additive parameterized approximation

We show that the following parameterized approximation problems are, in each case, reducible to the corresponding original parameterized problem: Add-Approx k-Independent Set, Add-Approx k-Clique, Add-Approx k-Dominating Set, Add-Approx k-Step Turing Machine Acceptance.

Theorem 1. Add-Approx k-Independent Set *is* $W[1]$*-hard.*

Proof. We transform from k-Independent Set.

Let $G = (V, E)$ be a graph and let k be the parameter. We produce $G' = (V', E')$ such that G' has a c-additive approximate solution for dk-Independent Set, i.e. G' contains an independent set of size at least $dk - c$, iff G contains an independent set of size at least k.

To build G' we begin with the original graph G, and proceed as follows:

1. Find smallest d such that $\lceil \frac{dk-c}{d} \rceil \geq k$
2. G' consists of d separate copies of G

$\Leftarrow$ Suppose that G contains an independent set of size at least k, then, by the construction of G', there must be an independent set of size at least dk in G'.
$\Rightarrow$ Suppose that G' contains an independent set of size at least $dk - c$, then some copy of G in G' must contain an independent set of size at least k, by the choice of d. □

This simple *amplification technique* can be used in parallel fashion to show:

Theorem 2. Add-Approx k-Clique *is* $W[1]$*-hard.*

The case of Add-Approx k-Dominating Set also employs the amplification technique, except now we are looking to minimize, rather than maximize, the solution.

Theorem 3. Add-Approx k-Dominating Set *is* $W[2]$*-hard.*

Proof. We transform from k-Dominating Set.

Let $G = (V, E)$ be a graph and let k be the parameter. We produce $G' = (V', E')$ such that G' has a c-additive approximate solution for dk-Dominating Set, i.e. G' contains a dominating set of size at most $dk + c$, iff G contains a dominating set of size at most k.

To build G' we begin with the original graph G, and proceed as follows:

1. Find smallest d such that $\lfloor \frac{dk+c}{d} \rfloor \leq k$
2. G' consists of d separate copies of G

$\Leftarrow$ Suppose that G contains a dominating set of size at most k, then, by the construction of G', there must be a dominating set of size at most dk in G'.
$\Rightarrow$ Suppose that G' contains a dominating set of size at most $dk + c$, then some copy of G in G' must contain a dominating set of size at most k, by the choice of d. □

We now consider additive approximation for k-Turing Machine Acceptance, the problem of deciding if a nondeterministic Turing machine with arbitrarily large fanout has a k-step accepting path on the empty input string.

Add-Approx k-Turing Machine Acceptance
Input: A Turing machine M
Parameters: k, c, positive integers
Output: 'NO' asserting that no k-step accepting path for M exists, or an accepting path of length at most $k + c$ for M.

Theorem 4. Add-Approx k-Turing Machine Acceptance *is* $W[1]$*-hard.*

Proof. We transform from k-Turing Machine Acceptance.

Let M be a Turing machine and let k be the parameter. We define M' such that M' has a c-additive approximate solution for $dk + 1$-Turing Machine Acceptance, iff M has an accepting path of length at most k.

Choose $d >> c$. Choose an alphabet for M' sufficiently large such that all d-sets of symbols from the alphabet for M may be represented. On the empty input string M' runs d copies of M in parallel, repeating each step of the computation for M d times, before proceeding to the next. M' will halt and accept immediately that all copies of M have halted and accepted.
$\Leftarrow$ Suppose that M has an accepting path of length at most k, then, by the construction of M', there must be an accepting path of length at most $dk + 1$ for M'.

$\Rightarrow$ Suppose that M' has an accepting path of length at most $dk + 1 + c$, then some copy of M run by M' must have an accepting path of length at most k, by the choice of d. $\square$

3 A completely inapproximable parameterized problem

In this section we show that k-Independent Dominating Set is *completely inapproximable*. Specifically, we show that there is no approximation scheme for k-Independent Dominating Set for *any* function $g(k)$ unless $W[2] = FPT$.

The natural parameterized version of the Dominating Set problem is the following.

k-Dominating Set
Input: A graph G.
Parameter: A positive integer k.
Question: Does G have a dominating set of size k? (A dominating set for G is a set $X \subseteq V(G)$ such that for all $y \in V(G)$, there is an $x \in X$ with $\langle x, y\rangle \in E(G)$.)

In [4] Downey and Fellows show that k-Dominating Set is $W[2]$-hard via a transformation from Weighted CNF Satisfiability.

For X a Boolean expression in conjunctive normal form consisting of m clauses $C_1, ..., C_m$ over the set of n variables $x_0, ..., x_{n-1}$, they 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.

The size $2k$ dominating set in G corresponding to a weight k truth assignment for X, is in fact an independent set as well. Thus the same transformation shows that k-Independent Dominating Set is $W[2]$-hard.

We outline the construction of the graph G used in the reduction here. There are k gadgets arranged in a vertical line. Each of the gadgets has 3 main parts. Taken from top to bottom, these are variable selection, gap selection and gap and order enforcement. The variable selection component $A(r)$ is a clique and the gap selection component $B(r)$ consists of n cliques which are called columns. The first action is to ensure that in *any* dominating set of $2k$ elements, we must pick one vertex from each of these two components. This goal is achieved by $2k$ sets of $2k + 1$ enforcers, vertices from V_4 and V_5. (The names refer to the sets below.) Take the V_4, for instance. For a fixed r, these $2k + 1$ vertices are connected to all of the variable selection vertices in the component $A(r)$, and nowhere else. Thus if they are to be dominated by a $2k$ dominating set, then we must choose *some* element in the set $A(r)$, and similarly we must choose an element in the set $B(r)$ by virtue of the V_5 enforcers. Since we will need exactly $2k$ (or even $\leq 2k$) dominating elements it follows that we must pick *exactly* one from each of the $A(r)$ and $B(r)$ for $r = 1, ..., k$.

Each of the k variable selection components consists of a clique of n vertices labelled $0, ..., n - 1$. The intention being that the vertex labelled i represents a

choice of variable i being made true in the formula X. Correspondingly in the next $B(r)$ we have columns (cliques) $i = 0, ..., n-1$. The intention is that column i corresponds to the choice of variable i in the preceding $A(r)$. We join the vertex $a[r, i]$ corresponding to variable i, in $A(r)$, to all vertices in $B(r)$ *except* those in column i. This means that the choice of i in $A(r)$ will cover all vertices of $B(r)$ except those in this column. It follows that we *must* choose the dominating element from this column and nowhere else. (There are no connections from column to column.) The columns are meant to be the gap selection saying how many 0's there will be till the next positive choice for a variable. We finally need to ensure that (i) if we chose variable i in $A(r)$ and gap j in column i from $B(r)$ then we need to pick $i+j+1$ in $A(r+1)$ and (ii) that the selections are in order. This is the role of the gap and order enforcement component which consists of a set of n vertices (in V_6.)

Thus the above provides a selection gadget that chooses k true variables with the gaps representing false ones. We enforce that the selection is consistent with the clauses of X via the clause variables V_3. These are connected in the obvious ways. One connects a choice in $A[r]$ or $B[r]$ corresponding to making a clause C_q true to the vertex c_q. Then if we dominate all the clause vertices too, we must have either chosen in some $A[r]$ a positive occurrence of a variable in C_q or we must have chosen in $B[r]$ a gap corresponding to a negative occurrence of a variable in C_q, and conversely.

The vertex set V of G is the union of the following sets of vertices:
$V_1 = \{a[r, s] : 0 \leq r \leq k-1, 0 \leq s \leq n-1\}$
$V_2 = \{b[r, s, t] : 0 \leq r \leq k-1, 0 \leq s \leq n-1, 1 \leq t \leq n-k+1\}$
$V_3 = \{c[j] : 1 \leq j \leq m\}$
$V_4 = \{a'[r, u] : 0 \leq r \leq k-1, 1 \leq u \leq 2k+1\}$
$V_5 = \{b'[r, u] : 0 \leq r \leq k-1, 1 \leq u \leq 2k+1\}$
$V_6 = \{d[r, s] : 0 \leq r \leq k-1, 0 \leq s \leq n-1\}$

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

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.
$E_1 = \{c[j]a[r, s] : x_s \in C_j\}$
$E_2 = \{a[r, s]a[r, s'] : s \neq s'\}$
$E_3 = \{b[r, s, t]b[r, s, t'] : t \neq t'\}$
$E_4 = \{a[r, s]b[r, s', t] : s \neq s'\}$
$E_5 = \{b[r, s, t]d[r, s'] : s + t + 1 \leq n \wedge s' \neq s + t\} \cup \{b[k-1, s, t]d[k-1, s'] : s' \neq s + t (\mathrm{mod} n)\}$
$E_6 = \{a[r, s]a'[r, u]\}$
$E_7 = \{b[r, s, t]b'[r, u]\}$

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

Suppose X has a satisfying truth assignment 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

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

is a dominating set in G.

Conversely, suppose D is a dominating set of $2k$ vertices in G. The closed neighbourhoods 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 neighbourhoods. 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 $0 \le r \le k-1$.

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*. Furthermore the set E_5 forces the chosen variables to be chosen so that if $r < r'$ and we choose $a[r,q]$ and $a[r',q']$ then $q < q'$.

The edges of E_4, E_5 and E_9 enforce that the $2k$ vertices in D must represent such a choice consistently. It remains only to check that the fact that D is a dominating set ensures that the truth assignment represented by D satisfies X. This follows by the definition of the edge sets E_1 and E_8.

We modify the construction described above to show that the following parameterized approximation problem is $W[2]$-hard for *any* choice of $g(k)$.

$g(k)$-Approx Independent Dominating Set
Input: $G = (V, E)$
Parameter: k, g
Output: 'NO' asserting that no independent dominating set $V' \subseteq V$ of size $\le k$ for G exists, or an independent dominating set $V' \subseteq V$ for G of size at most $g(k)$.

Theorem 5. $g(k)$-Approx Independent Dominating Set *is* $W[2]$*-hard.*

Proof. We transform from Weighted CNF Satisfiability.

Given X, a Boolean expression in conjunctive normal form we construct a graph G that has a $g(k)$ approximate solution for $2k$-Independent Dominating Set if and only if X is satisfied by a truth assignment of weight k.

To build G we begin with the construction from [4] described in detail above. We single out *one* of the variable selection components $A(q)$ and add to the construction $k' = g(k) - 2k + 1$ new variable selection components, $G_1, G_2, \ldots, G_{k'}$, along with new edges between all vertices in each of these new components and all vertices in $A(q)$. Each of the new components is connected to $B(q)$, to the $(q - 1 \bmod k)$ gap and order enforcement component, and to the clause vertices of $V_3 = \{c[j] : 1 \leq j \leq m\}$ in exactly the same way as is $A(q)$. We modify the edge sets E_1, E_4 and E_9 accordingly.

For each of the new variable selection components except the last, that is G_i, $1 \leq i < k'$, we connect the jth vertex, $0 \leq j \leq n - 1$, in G_i, by an edge to all vertices in G_{i+1} except for the jth vertex in G_{i+1}.

Finally, we blow up the size of the enforcement vertex sets V_4, V_5 and V_6 so that
$V_4 = \{a'[r, u] : 0 \leq r \leq k - 1, 1 \leq u \leq g(k) + 1\}$
$V_5 = \{b'[r, u] : 0 \leq r \leq k - 1, 1 \leq u \leq g(k) + 1\}$
$V_6 = \{d[r, s, t] : 0 \leq r \leq k - 1, 0 \leq s \leq n - 1, 1 \leq t \leq g(k) + 1\}$
and modify the edge sets E_5, E_6, E_7 and E_9 accordingly.

The construction is illustrated in Figure 1.

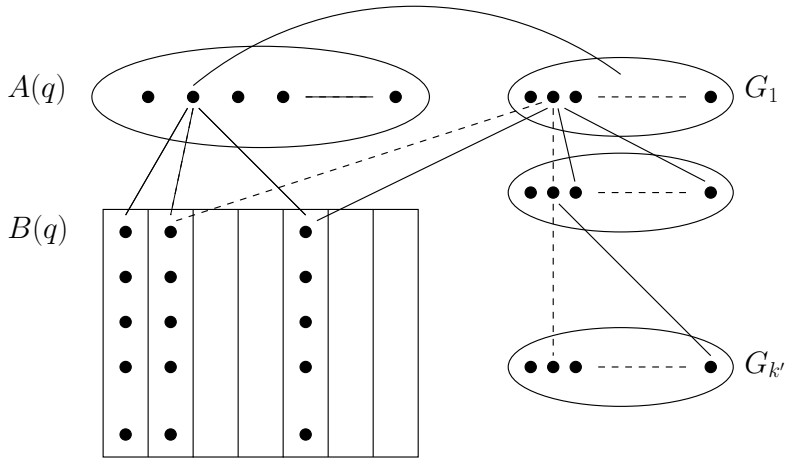

Fig. 1. Gadget for $g(k)$-Approx Independent Dominating Set

$\Leftarrow$ Suppose X has a satisfying truth assignment 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 an independent dominating set in G.

$\Rightarrow$ Suppose that G contains an independent dominating set D of size at most $g(k)$.

There are two possibilities here. In the first case, D has size $2k$ and contains exactly one vertex from each of the sets $A(r)$ and $B(r)$ for $0 \leq r \leq k-1$. The closed neighbourhoods 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 at least $2k$ vertices, one in each of these closed neighbourhoods. Since D is independent we can choose at most one from each of the $A(r)$, $0 \leq r \leq k-1$, and at most one from each of the $B(r)$, $0 \leq r \leq k-1$ (in the correct non-dominated column.) 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 $1 < u' < g(k)+1$ (otherwise D fails to be dominating), which contradicts that D contains at most $g(k)$ vertices. None of the vertices in V_6 are in D by a similar argument. Finally, if D contains a vertex from $A(q)$ then, since G is independent, none of the vertices in the new variable selection components, G_i, $1 \leq i \leq k'$ are in D. It follows that D contains exactly one vertex from each of the sets $A(r)$ and $B(r)$ for $0 \leq r \leq k-1$.

In the second case, D has size $g(k)$ and contains exactly one vertex from each of the sets $A(r)$, $0 \leq r \leq k-1, r \neq q$, and $B(r)$, $0 \leq r \leq k-1$, and exactly one vertex from each of the sets $G_i, 1 \leq i \leq k'$. In the case of the G_i sets, if D contains the jth vertex of G_1, then D contains the jth vertex of each of the other G_i, $2 \leq i \leq k'$. Note that, as in the first case, 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 $1 < u' < g(k)+1$ (otherwise D fails to be dominating), which contradicts that D contains at most $g(k)$ vertices. None of the vertices in V_6 are in D by a similar argument. Finally, if D contains a vertex from any of the G_i then, since G is independent, none of the vertices in $A(q)$ can be in D.

In either case, the truth assignment represented by D satisfies X. This follows by the definition of the edge sets E_1 (modified) and E_8. □

References

1. C. Bazgan: *Schemas d'approximation et complexite parametree.* Rapport de DEA, Universite Paris Sud, 1995.
2. L. Cai, X. Huang: *Fixed parameter Approximation: Conceptual Framework and Approximability Results.* Manuscript.
3. M. Cesati, L. Trevisan: *On the efficiency of polynomial approximation schemes.* Information Processing Letters, 64(4), pp 165-171, 1997.
4. R. G. Downey, M. R. Fellows: *Parameterized Complexity* Springer-Verlag, 1999.
5. R. G. Downey, C. M. McCartin: *Online Problems, Pathwidth, and Persistence.* Proceedings of IWPEC 2004, Springer-Verlag LNCS 3162, pp 13-24, 2004.
6. J. Flum, M. Grohe: *The Parameterized Complexity of Counting Problems.* SIAM Journal on Computing, 33(4), pp 892-922, 2004.
7. M. R. Garey and D. S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-completeness* Freeman, New York, 1979.
8. C. M. McCartin: *Contributions to Parameterized Complexity* Ph.D. Thesis, Victoria University, Wellington, 2003.