The Parameterized Complexity of Certain Fundamental Problems * in the Theory of Linear Codes and Integer Lattices

Rod G. Downey
Victoria University,
Wellington, New Zealand
& **Michael R. Fellows**
University of Victoria
Victoria, B.C. Canada
and Victoria University,
New Zealand
& **Alexander Vardy**
University of Illinois
Urbana, IL 61801, USA
& **Geoff Whittle**
Victoria University,
Wellington, New Zealand

November 1, 1999

Two Basic Examples

- VERTEX COVER

  Input: A Graph G.

  Parameter : A positive integer k.

  Question: Does G have a size k vertex cover? (Vertices cover edges.)

- DOMINATING SET

  Input: A Graph G.

  Parameter : A positive integer k.

  Question: Does G have a size k dominating set? (Vertices cover vertices.)

- VERTEX COVER is solvable by an algorithm $\mathfrak{O}$ in time $f(k)|G|$, a behaviour we call *fixed parameter tractability*, (Specifically $c + 1.2^k|G|$, with c a small absolute constant independent of k.) This is now implemented in several applications and found to be practical for $k \leq 300$. Theoretically practical for $k \leq 80$.
- Whereas the only known algorithm for DOMINATING SET is complete search of the possible k-subsets, which takes time $\Omega(|G|^k)$.

Basic Definition(s)

- Setting : Languages $L \subseteq \Sigma^* \times \Sigma^*$.
- Example (Graph, Parameter).
- We say that a language L is *fixed parameter tractable* if there is a algorithm M, a constant C and a function f such that for all x, k,

$$(x, k) \in L \text{ iff } M(x) = 1 \text{ and}$$

the running time of $M(x)$ is$f(k)|x|^C$.

- E.g. VERTEX COVER has $C = 1$.

Why is this interesting 1

- The table below illustrates why this might be interesting.

- Note that we are using arbitarily $f(k) = 2^k$, and sometimes we can do better.
- In fact the current VERTEX COVER algorithm is feasible for $k = 70$, and large input.
- *So the FPT is interesting since it works better than complete search for problems where we might be interested in small parameters but large input size.*

Why is this interesting 2

- Lots of applications, and potential applications.
- Most computational problems involve several pieces of input, one or more of which may be a relevant parameter for various applications.
- For instance, VERTEX COVER can be often used to represent the *conflict graph* of some computational problem. Some Interesting parameters below.
- *Graph linear layout width metrics* are of interest in VLSI layout and

routing problems and have important applications for width values of $k \leq 10$. Interval graphs of pathwidth $k \leq 10$ have applications in DNA sequence reconstruction problems and in other evolutionary problems such as language evolution and phylogenic tree reconstruction.

- *Logic and database problems* frequently are defined as having input consisting a formula (which may be small and relatively invariant), and some other structure (such as a database) which is typically quite large and changeable. Formula size, or other aspects of formula structure may be a relevant parameter(See e.g. Yannakakis STOC). The number of degrees of freedom in a robot motion-planning problem is commonly in the range $k \leq 10$. The number of wiring layers in VLSI chip manufacture is typically bounded by $k \leq 30$.

- *Network problems* may be naturally concerned with optimally locating a small number of facilities.

  These few examples are only suggestive and by no means exhaustive. There are myriad ways in which numbers that are small or moderately large (e.g., $k \leq 40$) arise naturally in problem specifications.

Whis is it interesting 3

- How Parameters arise in practice.
- In classical complexity a decision problem is specified by two items of information:
  (1) The input to the problem.
  (2) The question to be answered.
- In parameterized complexity there are three parts of a problem specification:
  (1) The input to the problem.
  (2) The aspects of the input that constitute the parameter.
  (3) The question.

- We have already seen *Graph Width Metrics*. These can often be *hidden* parameters in some computational problem. Examples include TREEWIDTH, CUTWIDTH, BANDWIDTH and PATHWIDTH.
- We have already seen *Implicit Parameters in Logic Programming, Compiler Design and Database Applications.*
- *Possible explanation of unpredicted tractability.* Problems can be much easier in practice than one would expect from theoretical analyses. E. g. the problem of type inference arises in implementations of programming languages such as ML

that are based on the polymorphic typed λ-calculus. (Henglein and Mairson) The problem is complete for deterministic exponential time, yet it has been widely noted that in practice the problem is *efficiently* solved by an algorithm that is known to be exponential in the worst case. An explanation for this discrepancy comes from noting that the logic formulas that occur in natural programs tend to have small bounded depth of *let*'s. For a parameter k bounding this depth, the problem is linear fixed-parameter tractable.

- Such considerations give a different possible explanation from those currently offered such as probabilistic behaviour.

- *Problems considered* in practice *come from "people" situations and hence can often have very small parameteres asscoiated with them because of our "brain" limitations.*

Why is this interesting 4

- A Distinctive Positive Toolkit. Lots of general techniques, begging for refinement. For example.
- Well-Quasi-Ordering. E. g. Robertson-Seymour.
- *Bounded Treewidth and Pathwidth,* Currently approaching practicality, Bodlaender: $f(k) = 2^{2^k}$.
- Related here are *automata theoretical methods*, practical depending on the presentation of the objects.
- *Color-Coding (Hashing)* and derandomization. (Alon, Yuster and

Zwick) The method works quite well for certain kinds of problems, such as finding small subgraphs in a graph (such as paths, cycles and matchings). Practical: $f(k) = 2^k$ and the resulting algorithms offer significant practical parameter ranges and no sources of large hidden constants.

- *Elementary Methods.* *Search trees* and *reductions to a problem kernel* An interesting aspect of these approaches is that while they are simple algorithmic strategies, they are in some sense "new ideas" that might not immediately come to mind, *because they involve costs that* *are exponential in the parameter.*

Why is this intersting 5

- A new paradigm for seeking tractability.

Why is this interesting 6

- Natural Measures of *fixed parameter INtractability.* They tell you when this approach with this particular parameter will fail.

- Natural basic hardness class: $W[1]$. Does not matter what it is, save to say that the analog of Cook's Theorem is

  SHORT NONDETERMINISTIC TURING MACHINE ACCEPTANCE
  Instance: A nondeterministic Turing Machine M and a positive integer k.
  Parameter: k.

Question: Does M have a computation path accepting the empty string in at most k steps?

- If one believes the philosophical argument that Cook's Theorem provides compelling evidence that SAT is intractible, then one surely must believe the same for the parametric intractability of SHORT NONDETERMINISTIC TURING MACHINE ACCEPTANCE.

Why is this interesting 7

- You can use it to prove unparameterized problems to not be feasible even though they are probably not NP-complete or are unknown. E.g. VC-DIMENSION, PRECEDENCE CONSTRAINED k-PROCESSOR SCHEDULING.
- Deep connections with approximation. $W[1]$-hardness means no good PTAS (Cai-Chen, Bagzan).
- Connections with logic. E.g. if in Max-SNP then FPT.
- Finally it's mathematically rich! and fun!

Our Case Study: Linear Codes

Problem: MAXIMUM-LIKELIHOOD DECODING

Instance: A binary $m \times n$ matrix H, a target vector $s \in \mathcal{F}^m$, and an integer $k > 0$.

Question: Is there a set of at most k columns of H that sum to s ?

Parameter: k

Problem: WEIGHT DISTRIBUTION

Instance: A binary $m \times n$ matrix H, and an integer $k > 0$.

Question: Is there a set of k columns of H that sum to the all-zero vector?

Parameter: k

Problem: Minimum Distance

Instance: A binary $m \times n$ matrix H, and an integer $k > 0$.

Question: Is there a set ($\neq \varnothing$) of at most k columns of H that sum to the all-zero vector?

Parameter: k

- Classically, all 3 are known NP-complete.
- **Theorem [DFVW]** WEIGHT DISTRIBUTION and MAXIMUM-LIKELIHOOD DECODING are hard for $W[1]$ (and in $W[2]$).

A Corollary

- The following Corollary is a archetype of the use of the parametric analysis to address questions no usually addressable by classical analysis.

- Maximum-likelihood decoding is a nearest neighbor search in the vector space $\mathcal{F}^n$ endowed with the Hamming distance $d(x, y) =$ number of positions where $x \in \mathcal{F}^n$ and $y \in \mathcal{F}^n$ differ.

- Let H be an $m \times n$ binary matrix, and let $\mathcal{C}$ be the binary linear code defined by H, that is

$\mathcal{C} = \{x \in \mathcal{F}^n \ : \ Hx^t = \mathbf{0}\}$. Then for all $y \in \mathcal{F}^n$, a *maximum-likelihood decoder* for $\mathcal{C}$ always finds the closest codeword, that is $x \in \mathcal{C}$ such that $d(x, y)$ is the minimum possible. If $s = Hy^t$, this is equivalent to finding the smallest set of columns of H that sums to s; hence the corresponding decision problem is precisely MLD.

- Almost nothing is presently known regarding the complexity of bounded-distance decoding, even though most of the decoders used in practice are bounded-distance decoders.

- A decoder is said to be *bounded-distance* if there exists a constant $t > 0$ such that for all $y \in \mathcal{F}^n$, the decoder always finds the closest codeword $x \in \mathcal{C}$, provided $d(x, y) \le t$

- **Problem:** $\mathrm{BDD}(t)$

  Instance: A binary $m \times n$ matrix H, a target vector $s \in \mathcal{F}^m$, and an integer $k \le t$.

  Question: Is there a set of at most k columns of H that sum to s ?

- Notice that $\mathrm{BDD}(t)$ is trivially in P for all t, since it can be solved in time $O(n^t)$ by simply computing Hx^t for every vector x in a Hamming sphere of radius t.

- **Thorem [DFVW]** Assume $W[1] \neq FPT$. Then for any positive constant c, there exists an integer t_0, such that BDD(t) is not solvable in time $O(n^c)$ for all $t \geq t_0$, even if the multiplicative constant in $O(n^c)$ may depend on t.

Coding Problems from Integer Lattices

Problem: NEAREST VECTOR

Instance: A basis $X = \{x_1, x_2, \ldots, x_n\} \subset \mathcal{Z}^n$ for a lattice Ł, a target vector $s \in \mathcal{Z}^n$, and an integer $k > 0$.

Question: Is there a vector $x \in$ Ł, such that $\|x - s\|^2 \le k$?

Parameter: k

Problem: THETA SERIES

Instance: A basis $X = \{x_1, x_2, \ldots, x_n\} \subset \mathcal{Z}^n$ for a lattice Ł, and an integer $k > 0$.

Question: Is there a vector $x \in Ł$, such that $\|x\|^2 = k$?

Parameter: k

Problem: SHORTEST VECTOR

Instance: A basis $X = \{x_1, x_2, \ldots, x_n\} \subset \mathcal{Z}^n$ for a lattice Ł, and an integer $k > 0$.

Question: Is there a non-zero vector $x \in Ł$, such that $\|x\|^2 \leq k$?

Parameter: k

- Problem 1 is classically known to be NP-complete. Problems 2 and 3 were open.

- **Theorem** (DFVW) THETA SERIES *is NP complete and both* THETA SERIES *and* NEAREST VECTOR *are* $W[1]$ *hard.*

The Proofs, and Red/Blue Graphs

- All of the proofs work through graphs.
- Let $G = (\mathcal{R}, \mathcal{B}, E)$ be a bipartite graph with the partition of the vertices into the red set $\mathcal{R}$ and the blue set $\mathcal{B}$.
- $R \subseteq \mathcal{R}$ is called a
  - a ***dominating set*** if every vertex in $\mathcal{B}$ has at least one neighbor in R
  - a ***perfect code*** if every vertex in $\mathcal{B}$ has a unique neighbor in R
  - an ***odd set*** if every vertex in $\mathcal{B}$ has an odd number of neighbors in R

– an ***even set*** if every vertex in $\mathcal{B}$ has an even number of neighbors in R.

More Problems

Problem: Even Set

Instance: A red/blue graph $G = (\mathcal{R}, \mathcal{B}, E)$ and an integer $k > 0$.

Question: Is there a non-empty set of at most k vertices $R \subseteq \mathcal{R}$ that is an even set in G ?

Parameter: k

Problem: Exact Even Set

Instance: A red/blue graph $G = (\mathcal{R}, \mathcal{B}, E)$ and an integer $k > 0$.

Question: Is there a set of k vertices $R \subseteq \mathcal{R}$ that is an even set in G ?

Parameter: k

Problem: Odd Set

Instance: A red/blue graph $G = (\mathcal{R}, \mathcal{B}, E)$ and an integer $k > 0$.

Question: Is there a set of at most k vertices $R \subseteq \mathcal{R}$ that is an odd set in G ?

Parameter: k

Problem: Exact Odd Set

Instance: A red/blue graph $G = (\mathcal{R}, \mathcal{B}, E)$ and an integer $k > 0$.

Question: Is there a set of k vertices $R \subseteq \mathcal{R}$ that is an odd set in G ?

Parameter: k

- **Theorem [DFVW]** All of the above are hard for $W[1]$.
- The main combinatorial engine is the following.
- **Theorem [DFVW]** Let G be a graph on n vertices, and let k be a positive integer. In time polynomial in n and k we can produce a red/blue bipartite graph G' and a positive integer k', such that:

  P1. Every dominating set in G' has size at least k'.

  P2. Every dominating set in G' of size k' is a perfect code in G'.

P3. There is a perfect code of size k in G if and only if there is a perfect code of size k' in G'.

A simple application

- Odd Set and Exact Odd Set are hard for $W[1]$ and NP-complete. Theorem 1 immediately implies a polynomial-time parameterized transformation from Perfect Code, which is a $W[1]$-hard and NP-complete problem, to Odd Set and to Exact Odd Set. Given an instance G and k of Perfect Code, we construct G' and k' as in Theorem 1. It follows from properties **P1**–**P3** that G' contains an odd set of size at most k' if and only if G has a k-element perfect code, and the size of this odd set is, in fact, exactly k'.

- (Using ODD SET) MAXIMUM-LIKELIHOOD DECODING is hard for $W[1]$. Given an instance $G = (\mathcal{R}, \mathcal{B}, E)$ and k of ODD SET, we construct an instance of MAXIMUM-LIKELIHOOD DECODING as follows. The binary $m \times n$ matrix $H = [h_{ij}]$ is the red/blue adjacency matrix of G, whose columns are indicators of the neighborhoods of vertices in $\mathcal{R}$. That is, w.l.o.g. we let $\mathcal{R} = \{1, 2, \ldots, n\}$ and $\mathcal{B} = \{1, 2, \ldots, m\}$, and then set $h_{ij} = 1$ if and only if $i \in \mathcal{B}$ is adjacent to $j \in \mathcal{R}$ in G. Thus G is a Tanner graph for the binary linear code having H as its parity-check matrix (Tanner,81). We set the target vector s equal to the all-one vector $\mathbf{1}$ of length $m = |\mathcal{B}|$. It is easy to see that a set of k columns of H sums to $s = \mathbf{1}$ if and only if the corresponding k vertices of $\mathcal{R}$ constitute an odd set in G.

An Open Question

- Aside from the question left open in the above (Is MINIMUM DISTANCE $W[1]$ hard, what about SHORTEST VECTOR?), the following might be of interest to the present audience.

- (W[1] Hard?) BOUNDED HAMMING WEIGHT DISCRETE LOGARITHM
  Instance: An n-bit prime, a generator g of F_p^*, an element $a \in F_p^*$, a positive integer k.
  Parameter: A positive integer k.
  Question: Is there a positive integer x whose binary representation has at

most k 1's (that is, x has a Hamming weight of k) such that $a = g$?

- Note the related problem below in randiomized FPT (Fellows and Koblitz)

BOUNDED FACTOR FACTORIZATION
Instance: An n-bit positive integer N, a positive integer k.
Parameter: A positive integer k.
Question: Is there a prime factor p of N such that $p < n^k$?

Your Strategy

- Buy that very reasonably priced book *"Parameterized Complexity"* (It is an excellent Present!)
- Some of the papers can be found in `www.mcs.vuw.ac.nz/research /math-pubs.shtml`
- Especially "Parameterized computational feasibility" and "Parameterized Complexity Analysis"
- my e-mail `rod.downey@vuw.ac.nz`

- Thank you very much for your attention.