Parametric Complexity for the Skeptic

Rod Downey

Victoria University

Wellington

New Zealand

What's in this for me?

- A mathematical idealization is to identify "Feasible" with P. (I won't even bother looking at the problems with this.)
- With this assumption, the theory of NP-hardness is an excellent vehicle for mapping an *outer* boundary of intractability, for all practical purposes.
- Indeed, assuming the reasonable current working assumption that NTM acceptance is $\Omega(2^n)$, NP-hardness allows for practical lower bound for exact solution for

problems.

- A very difficult practical and theoretical problem is "How can we deal with P?".
- More importantly how can we deal with $P-FEASIBLE$, and map a further boundary of intractability.
- *Lower bounds in P* are really hard to come by. But this theory will alow you establish infeasability for problems in P, under a reasonable complexity hypothesis.
- Also it will indicate to you how to attack the problem if it looks bad.
- It is thus both a positive and negative tool kit.

I'm Dubious; Example?

- Below is *one* CCC application that points at why the *completeness* theory might interest you.
- The great PCP Theorem of Arora et. al. allows us to show that things don't have PTAS's on the assumption that P$\neq$NP.
- Some things actually *do* have PTAS's. Lets look at a couple taken from recent major conferences: STOC, FOCS, SODA etc.

- Arora 1996 gave a $\mathcal{O}(n^{\frac{3000}{\epsilon}})$ PTAS for Euclidean Tsp

- Chekuri and Khanna 2000 gave a $\mathcal{O}(n^{12(\log(1/\epsilon)/\epsilon^8)})$ PTAS for Multiple Knapsack

- Shamir and Tsur 1998 gave a $\mathcal{O}(n^{2^{2^{\frac{1}{\epsilon}}}-1)})$ PTAS for Maximum Subforest

- Chen and Miranda 1999 gave a $\mathcal{O}(n^{(3mm!)^{\frac{m}{\epsilon}+1}})$ PTAS for General Multiprocessor Job Scheduling

- Erlebach *et al.* 2001 gave a $\mathcal{O}(n^{\frac{4}{\pi}(\frac{1}{\epsilon^2}+1)^2(\frac{1}{\epsilon^2}+2)^2})$ PTAS for Maximum Independent Set for geometric graphs.

- Deng, Feng, Zhang and Zhu (2001) gave a $O(n^{5\log_{1+\epsilon}(1+(1/\epsilon))})$ PTAS for UNBOUNDED BATCH SCHEDULING.
- Shachnai and Tamir (2000) gave a $O(n^{64/\epsilon+(log(1/\epsilon)/\epsilon^8)})$ PTAS for CLASS-CONSTRAINED PACKING PROBLEM (3 cols).

Reference	Running Time for a 20% Error
Arora (Ar96)	$\mathcal{O}(n^{15000})$
Chekuri and Khanna (CK00)	$\mathcal{O}(n^{9,375,000})$
Shamir and Tsur (ST98)	$\mathcal{O}(n^{958,267,391})$
Chen and Miranda (CM99)	$> \mathcal{O}(n^{10^{60}})$ (4 Processors)
Erlebach *et al.* (EJS01)	$\mathcal{O}(n^{523,804})$
Deng et. al (DFZZ01)	$\mathcal{O}(n^{50})$
Shachnai and Tamir (ST00)	$\mathcal{O}(n^{1021570})$

Table 1: The Running Times for Some Recent PTAS's with 20% Error.

What is the problem here?

- Arora (1997) gave a PTAS running in nearly linear time for EUCLIDIAN TSP. What is the difference between this and the PTAS's in the table. Can't we simply argue that with more effort all of these will eventually have truly feasible PTAS's.
- The principal problem with the baddies is that these algorithms have a factor of $\frac{1}{\epsilon}$ (or worse) in their exponents.
- By analogy with the situation of NP completeness, we have some problem that has an exponential algorithm.

Can't we argue that with more effort, we'll find a much better algorithm? As in Garey and Johnson's famous cartoon, we cannot seem to prove a better algorithm. BUT we prove that it is NP hard.

- Then assuming the *working hypothesis* that there is basically *no way to figure out if a NTM has an accepting path of length n except trying all possibilities* there is no hope for an exact solution with runnig time significantly better than 2^n. (Or at least no polynomial time algorithm.)
- Moreover, if the PCP theorem applies,then using this basic

hypothesis, there is also no PTAS.

- In the situation of the *bad* PTAS's the algorithms *are* polynomial. Polynomial lower bound are hard to come by.
- It is difficult to apply classical complexity since the classes are not very sensitive to things in P.
- Our idea in this case is to follow the NP analogy but work within P.

- What parametric complexity has to offer:
- Then assume the *working hypothesis* that there is basically *no way to figure out if a NTM has an accepting path of length k except trying all possibilities.* Note that there are $\Omega(n^k)$ possibilities. (Or at least no way to get the "k" out of the exponent or an algorithm deciding k-STEP NTM,)

- One then defines the appropriate reductions from k-Step Turing Machine Halting to the PTAS using $k = \frac{1}{\epsilon}$ as a parameter to *argue that if we can "get rid" of the k from the exponent then it can only be if the working hypothesis is wrong.*

Efficient PTAS's

- Even if you are only interested in "classical" problems you would welcome a methodology that allows for "practical" lower bounds in P, modulo a reasonable complexity assumption.

- An optimization problem Π has an *efficient P-time approximation scheme e* (EPTAS) if it can be approximated to a goodness of $(1+\epsilon)$ of optimal in time $f(k)n^c$ where c is a constant and $k = 1/\epsilon$.

Bagzan, Cai-Chen

- (without even the formal definition) (Bazgan (Baz95), also Cai and Chen (CC97)) Suppose that Π_{opt} is an optimization problem, and that Π_{param} is the corresponding parameterized problem, where the parameter is the value of an optimal solution. Then Π_{param} is fixed-parameter tractable if Π_{opt} has an EPTAS.

- It is unknown if the PTAS's in the table have EPTAS's or not.
- In this talk, I will give a tourist guide through the area or parameterized complexity, making sure to mention a number of applications like the above to "classical" complexity.
- I would also like to highlight the way that parameterized complexity allows for an extended "dialog" with the problem at hand. (More on this soon).

- Others to use this technique include the following
- (Alekhnovich and Razborov (AR01)) Neither resolution not tree-like resolution is axiomatizable unless $W[P]$ is randomized FPT by a randomized algorithm with one-sided error. (More on the hypothesis later)
- Frick and Grohe shoed that towers of twos obtained from general tracatbility results with respect to model checking can't be gotten rid of unless $W[1] = FPT$, again more later.

Parameters

- Without even going into details, think of all the graphs you have given names to and each has a relevant parameter: planar, bounded genus, bounded cutwidth, pathwidth, treewidth, degree, interval, etc, etc.
- Also *nature* is kind in that for many practical problems the input (often designed by *us*) is nicely ordered.

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 $1.4^k k^2 + c|G|$, with c a small absolute constant independent of k.)
- 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 $C = 1$. Vertex Cover has been implemented and shown to be practical for a class of problems arizing from computational biology for k up to about 400 (Stege 2000, Dehne, Rau-Chaplin, Stege, Taillon 2001) and n large.

- Keep in mind:an FPT language is in P "by the slice", *and more*: each k-slice is in the *same* polynomial time class via the *same* machine.

- Let L_k denote the k-th slice of L and $L_k^{(>m)}$ denote $\{\langle x, k\rangle : |x| > m\}$, the part of L_k *from m onwards.*

- (Cai, Chen Downey, Fellows; Flum and Grohe) $L \in$ FPT iff there is an algorithm M, a constant c, and a computable function g such that M witnesses that

$$L_k^{(>g(k))} \in DTIME(n^c).$$

- e.g. For Vertex Cover, g is about 2^k.

- Can do this with other classes, such as LOGSPACE, etc.

Does this matter?

- The table below illustrates why this might be interesting.

	$n = 50$	$n = 100$	$n = 150$
$k = 2$	625	2,500	5,625
$k = 3$	15,625	125,000	421,875
$k = 5$	390,625	6,250,000	31,640,625
$k = 10$	1.9×10^{12}	9.8×10^{14}	3.7×10^{16}
$k = 20$	1.8×10^{26}	9.5×10^{31}	2.1×10^{35}

Table 2: The Ratio $\frac{n^{k+1}}{2^k n}$ for Various Values of n and k

- Note that we are using arbitarily $f(k) = 2^k$, and sometimes we can do better. (Such as the case of Vertex Cover)

- *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.*

- *Question.* That's pretty weird. How can you have this arbitrary f in the definition of *parametric feasibility* when it could be Ackermann's function or worse?

- *Answer.* When it was introduced, polynomial time as a central paradigm for analyzing computational feasibility was somewhat controversial. Edmonds makes a special point of discussing this in [**?**]. The usual criticisms: what about ridiculously large exponents, what about ridiculously large constants etc. The answer is kind of pragmatic. First PTIME has very nice closure properties that make it mathematically sound. Second, at least until the last 10 years, “real” problems that are in P have feasible solutions, by and large.

- We would argue that the same is true of our mathematical idealization, FPT. There are a number of truly applied FPT algorithms, and they are reasonable. However, there are also some general techniques which give theoretical feasibility but for which there are no known really feasible algorithms, nor proofs that no such algorithms exist subject to any reasonable assumption even (But Frick-Flum-Grohe FTP-FEASIBLE)

- *Question.* How do you know which parameter to use? Is there a canonical choice?
- *Answer.* Often there is no canonical parameter. Sometimes there is at least one obvious one. However, the beauty is, as we will see, we can often look at the type of data we are provided with and seek a parameter that makes the problem tractable. It is *good* that a problem can have *many* parameterizations. This is the idea behind using this technique as a systematic way to address feasibility.

Some Areana of Applications

- 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.

How parameters arize 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.
- Natural obvious arenas such as computational bilogy and

phylogenics. Many actual inplmentations, such as those of the survey due to Warnow et. al and Hallett, M. and Lagergren, J. (2001) Efficient algorithms for detecting horizontal gene transfer, RECOMB 2001, 149-156.
Addario-Berry, L., Hallett, M., Lagergren, J. (2003) Towards identifying lateral gene transfer events. Pacific Symposium on Biocomputing 8:279-290(2003).

- We have already seen *Graph Width Metrics*. These can often be *hidden* parameters in some computational problem. Examples include TREEWIDTH, CUTWIDTH, BANDWIDTH and PATHWIDTH. (For example, it has been proposed that the syntactic structure of sentences of natural languages can be modeled by dependency graphs of pathwidth less than 10 (Kornai and Tuza).)

- 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.

- Another example concerns the

register allocation problem for imperative programming languages. This is a problem that is *NP*-complete in general. It has been shown that for structured programs (with short-circuit evaluation) the control-flow graphs have treewidth less than 10, which allows linear FPT algorithms based on this hidden structural parameter to be developed in, for instance, Alstrup, Lauridsen and Thorup. Also recently done for jAVA programs. (?)

- 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.*

Reductions and Intractability

- 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.
- Moreover, recent work has shown that if SHORT NTM is fpt then n-variable 3SAT is in DTIME$(2^{o(n)})$

- Given two parameterized languages $L, \widehat{L} \subseteq \Sigma^* \times \Sigma^*$, say $L \leq_{FPT} \widehat{L}$ iff there are (computable) $f, x \mapsto x', k \mapsto k'$ and a constant c, such that for all x,

$$(x, k) \in L \text{ iff } (x', k') \in \widehat{L},$$

  in time $f(k)|x|^c$.

- Lots of technical question still open here.

Analog of Cook's Theorem

- Analog of Cook's Theorem: (Downey, Fellows, Cai, Chen) Weighted 3Sat$\equiv_{FTP}$ Short NTM Acceptance.

  Weighted 3Sat
  Input: A 3 CNF formula φ
  Parameter: k
  Question: Does φ has a satisfying assignment of Hamming weigth k, meaning exactly k literals made true.

W-Hierarchy

- Think about the usual poly reduction from SAT to 3SAT. It takes a clause of size p, and turns it into many clauses of size 3. *But* the weight control goes awry. A weight 4 assignment could go to anything.

- We don't think WEIGHTED CNF SAT$\leq_{ftp}$WEIGHTED 3 SAT.

- Gives rise to a heirarchy:

$W[1] \subseteq W[2] \subseteq W[3] \ldots W[SAT] \subseteq W[P] \subseteq XP.$

- XP has k-Cat and Mouse Game and some other games ((DF99a)),
- $W[P]$ has Linear Inequalities, Short Satisfiability, Weighted Circuit Satisfiability ((ADF95)) and Minimum Axiom Set((DFKHW94)).
- Then there are a number of quite im portant problems from combinatorial pattern matching which are $W[t]$ hard for all t: Longest Common Subsequence ($k =$ number of seqs.,$|\Sigma|$-two parameters) ((BDFHW95)), Feasible Register Assignment, Triangulating Colored Graphs, Bandwidth, Proper Interval Graph Completion ((BFH94)), Domino Treewidth ((BE97)) and Bounded Persiste nce Pathwidth ((McC03)).
- $W[2]$ include Weighted $\{0,1\}$ Integer Programming, Dominating Set ((DF95a)), Tournament Dominating Set ((DF95c)) Unit Length Precedence Constrained Scheduling (hard) ((BF95)), Shortest Common Supersequence (k)(hard) ((FHK95)), Maximum Likelihood Decoding (hard), Weight Distribution in Linear Codes (hard), Nearest Vector in Integer Lattices (hard)

((DFVW99)), SHORT PERMUTATION GROUP FACTORIZATION (hard).

- $W[1]$ we have a collection including k-STEP DERIVATION FOR CONTEXT SENSITIVE GRAMMARS, SHORT NTM COMPUTATION, SHORT POST CORRESPONDENCE, SQUARE TILING ((CCDF96)), WEIGHTED q–CNF SATISFIABILITY ((DF95b)), VAPNIK–CHERVONENKIS DIMENSION ((DEF93)) LONGEST COMMON SUBSEQUENCE (k, m = LENGTH OF COMMON SUBSEQ.) ((BDFW95)), CLIQUE, INDEPENDENT SET ((DF95b)), and MONOTONE DATA COMPLEXITY FOR RELATIONAL DATABASES

Parameterized vs Unparameterized

- 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.
- As we have seen: connections with approximation. $W[1]$-hardness means no good PTAS (Cai-Chen, Bagzan).
- Connections with logic. E.g. if in Max-SNP then FPT. (esp Grohe)

A Case Study: Databases

- (Chandra and Merlin, 77) the complexity of query languages in the study of database theory.
- Vardi 1982 notes that classical complexity seemed wrong: suggested evaluation of a query when the size of the query was fixed as a function of the size of the database

- (Standard sort of problem) *Input:* A boolean query φ and a database instance I.
  Parameter: Some parameter of φ, such as the size of φ. (or its complexity etc)
  Problem: Evaluate φ in I.

- Yananakakis 1995 suggested that parameterized complexity good framework to address this.

- (Downey-Fellows-Taylor, 95 Papadimitriou-Yannakakis 97) The morning edition of the news is bad.

- Papadimitriou and Yannakakis systematically also looked at other parameters such as bounding the number of variables following ideas of Vardi (Va95). They looked at positive queries, conjunctive queries, first order theories and datalog ones and found them to be all $W[1]$ hard and at various levels of the W-hierarch y.

- Other analyses look at other parametric aspects and give even more bad news. (e.g. Demri, Laroussinie and Schnoebelen (DLS02).)
- You might well ask now, with "good" news like this provided by parameterized complexity, what use is it? You could argue that once we knew these problems were NP-hard

and likely PSPACE complete. Now we know that even when you bound the obvious parameters then they are *still* hard!

- One interpretation is that we should learn to live with this by searching for new coping strategies
- The parametric point of view is to continue the dialog with the problem. To cope by finding new, and maybe more appropriate parameters.
- (Frick and Grohe (FrG02), Flum and Grohe (FG02a)) Let C be a class of relational database instances such that underlying graph of instances in C are any of the following forms:

bounded degree, bounded treewidth, bounded local treewidth, planar or have an excluded minor. Then the query evaluation problem for the relational calculus on C is FPT.

- Frick and Grohe also looked at things beyong query languages, such as XML and temporal logics such as LTL and CTL* are used for specification languages for automated verification, proving sometimes practical FPT results.

- Notice that there are at least two ways to parameterize: Parameterize the part of the problem you want to look at and to parameterize the problem itself.
- This point of view makes this sometime a promise problem. Input something, promise it is parameterized, and ask questions about it.
- Some recent work “lowers the hardness barrier”; perhaps giving better inapproximability results.

The Minimob

- INPUT A parametrically minature problem QUESTION Is it in the class

  e.g. INPUT a graph G of size k log n Does it have a vertex cover of size d?

- Get minivertex cover, mini Dominating set, Minisat etc.

- Core problem: minicircuitsat.

- (Downey and Fellows) we don't believe that M[1]=W[1]

- Fellows conjectures that PCP like techniques will show M[1]=W[1] using randomized reductions

Fascinating Positive Toolkit

- Strongly recommend reading Niedermeier's Habilitationschrift
- Robertson-Seymour
- Color Coding
- Elementary methods.
- Logical methods: e.g. (Courcelle (Co87)) Suppose that φ is any sentence in monadic second order logic, and $\mathcal{C}_k$ is the class of graphs of bounded treewidth k. Then there is a linear time algorithm deciding if a given graph in $\mathcal{C}_k$ satisfies φ. That is, the problem is FPT.

- Hlineny (2002,2003) applications of the same methods to representable matroids.
- (Frick and Grohe) for first order via "local" treewidth.
- BUT: If $NP \neq P$, then the model checking problem for finite trees (in fact words) is not solvable in time $f(k)p(n)$, where n is the size of the word and k the sentence, for any polynomial p and *elementary* computable function f.
  If $W[1] \neq FPT$, then the model checking for first order logic on structures of degree 2 and bounded de gree $d \geq 3$ is not solvable in time $2^{2^{o(k)}}p(n)$ (degree 2) and $2^{2^{2^{o(k)}}}p(n)$

for any polynomial p

- It would take another lecture to get to how the many techniques can be applied. I refer to recent surveys of Raman, Fellows, Stege, Niedermeier, etc for details. An enormous amout of problems to map out usefulness and feasbility here. You are invited.

- Thank You