Basic Parametric Complexity I: Positive Techniques

Rod Downey
Victoria University
Wellington
Isaac Newton Institute, Cambridge

LATA, March 2012

This lecture:

- Basic Definitions
- Classical Motivations
- Basic Positive Techniques

What's in this for me?

- A mathematical idealization is to identify "Feasible" with P (polynomial time). (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 infeasibility for problems in P, under a reasonable complexity hypothesis.
- Also it will indicate to you how to attack the problem if it looks bad.
- As we soon see, sensitizing the run times to parameters allows the development of a distinctive and often useful toolkit.
- In particular, focusing on the paramaters as a standard attack method for practice can be systematized, and sometimes even automated. More later this lecture.
- The theory equips us with both a positive and negative tool kit.

I'M DUBIOUS; EXAMPLE?

- Below is one (negative) 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 $O(n^{\frac{3000}{\epsilon}})$ PTAS for EUCLIDEAN TSP
- Chekuri and Khanna 2000 gave a $O(n^{12(\log(1/\epsilon)/\epsilon^8)})$ PTAS for MULTIPLE KNAPSACK
- Shamir and Tsur 1998 gave a $O(n^{2^{2^{\frac{1}{\epsilon}}}-1)})$ PTAS for MAXIMUM SUBFOREST
- Chen and Miranda 1999 gave a $O(n^{(3mm!)^{\frac{m}{\epsilon}+1}})$ PTAS for GENERAL MULTIPROCESSOR JOB SCHEDULING
- Erlebach et al. 2001 gave a $O(n^{\frac{4}{\pi}(\frac{1}{\epsilon^2}+1)^2(\frac{1}{\epsilon^2}+2)^2})$ 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)	$O(n^{15000})$
CHEKURI AND KHANNA (CK00)	$O(n^{9,375,000})$
SHAMIR AND TSUR (ST98)	$O(n^{958,267,391})$
CHEN AND MIRANDA (CM99)	$> O(n^{10^{60}})$ (4 PROCESSORS)
ERLEBACH ET AL. (EJS01)	$O(n^{523,804})$
DENG ET. AL (DFZZ01)	$O(n^{50})$
SHACHNAI AND TAMIR (ST00)	$O(n^{1021570})$

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 running 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 concentrating on the distinctive positive techniques which have been developed.
- In this and the next talk, 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 the hardness theory include the following
- (Alekhnovich and Razborov (AR01)) Neither resolution not tree-like resolution is automizable unless $W[P]$ is randomized FPT by a randomized algorithm with one-sided error. (More on the hypothesis later)
- Frick and Grohe showed that towers of twos obtained from general tractability 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 $1.28^k k^2 + c|G|$, 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 $\Omega(|G|^k)$.

- In the blow I will mostly talk for convenience about graphs.
- I could just as easily be talking about many other areas.
- In the Computer Journal alone, there is biological, artificial intelligence, constraint satisfaction, geometric problems, scheduling, cognitive science, voting, combinatorial optimzation, phylogeny. Model check is the basis of Flum-Grohe.

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,

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

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

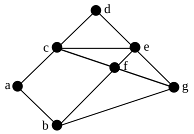
- E.g. VERTEX COVER has $C = 1$. Vertex Cover has been implemented and shown to be practical for a class of problems arizing from computational biology for k up to about 7000 and n large.
- One example: Langston et. al. 2008 Innovative computational methods for transcriptomic data analysis: A case study in the use of FPT for practical algorithm design and implementation. in *The Computer Journal*, 51(1):26–38, 2008

- 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

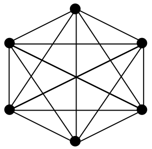
$$L_k^{(>g(k))} \in DTIME(n^c).$$

- e.g. For VERTEX COVER, g is about 2^k.
- Can do this with other classes, such as LOGSPACE, etc.

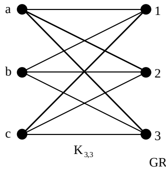
Other possible versions

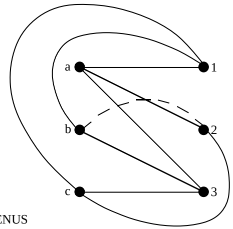


VERTEX COVER : Vertices cover edges.
Example: {c, f, b, e, h}.



GRAPH LINKING NUMBER :
K_6 has linking number 1.





GRAPH GENUS

$K_{3,3}$ has genus 1 by
putting all lines except
$< b, 2 >$ on a sphere
and $< b, 2 >$ on a handle.

FIGURE: Examples of FPT problems

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: 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.

Positive techniques

- Elementary ones
- Logical metatheorems
- Limits

KERNELIZATION

- I believe that the most important practical technique is called kernelization.
- pre-processing, or reducing

Karsten Weihe's Train Problem

- Train Covering by Stations
  Instance: A bipartite graph $G = (V_S \cup V_T, E)$, where the set of vertices V_S represents railway stations and the set of vertices V_T represents trains. E contains an edge $(s, t), s \in V_s, t \in V_T$, iff the train t stops at the station s.
  Problem: Find a minimum set $V' \subseteq V_S$ such that V' covers V_T, that is, for every vertex $t \in V_T$, there is some $s \in V'$ such that $(s, t) \in E$.

Weihe's Solution

- Reduction Rule TCS1:
  Let $N(t)$ denote the neighbours of t in V_S. If $N(t) \subseteq N(t')$ then remove t' and all adjacent edges of t' from G. If there is a station that covers t, then this station also covers t'.
- Reduction Rule TCS2:
  Let $N(s)$ denote the neighbours of s in V_T. If $N(s) \subseteq N(s')$ then remove s and all adjacent edges of s from G. If there is a train covered by s, then this train is also covered by s'.

- European train schedule, gave a graph consisting of around $1.6 \cdot 10^5$ vertices and $1.6 \cdot 10^6$ edges.
- Solved in minutes.
- This has also been applied in practice as a subroutine in practical heuristical algorithms.

The Idea

- Reduce the parameterized problem to a kernel whose size depends solely on the parameter
- As compared to the classical case where this process is a central heuristic we get a provable performance guarantee.
- We remark that often the performance is much better than we should expect especially when elementary methods are used.

Vertex Cover

- Reduction Rule VC1:
  Remove all isolated vertices.
- Reduction Rule VC2:
  For any degree one vertex v, add its single neighbour u to the solution set and remove u and all of its incident edges from the graph.
- Note $(G, k) \to (G', k - 1)$.
- (S. Buss) Reduction Rule VC3:
  If there is a vertex v of degree at least $k + 1$, add v to the solution set and remove v and all of its incident edges from the graph.
- The result is a graph with no vertices of degree $> k$ and this can have a VC of size k only if it has $< k^2$ many edges.

KERNELIZATION

DEFINITION (KERNELIZATION)

Let $L \subseteq \Sigma^* \times \Sigma^*$ be a parameterized language. Let $\mathcal{L}$ be the corresponding pa rameterized problem, that is, $\mathcal{L}$ consists of input pairs (I, k), where I is the main part of the input and k is the parameter. A reduction to a problem kernel, or kernelization, comprises replacing an instance (I, k) by a reduced instance (I', k'), called a problem kernel, such that

(i) $k' \leq k$,
(ii) $|I'| \leq g(k)$, *for some function* g *depending only on* k, *and*
(iii) $(I, k) \in L$ *if and only if* $(I', k') \in L$.

The reduction from (I, k) to (I', k') must be computable in time polynomial in $|I|$.

A Useless Theorem

Theorem (Cai, Chen, Downey and Fellows)

$L \in FPT$ iff L is kernelizable.

- Proof Let $L \in FPT$ via a algorithm running in time $n^c.f(k)$. Then run the algorithm which in time $O(n^{c+1})$, which eventually dominates $f(k)n^c$, either computes the solution or understands that it is in the first $g(k)$ many exceptional cases. ("Eventually polynomial time")

STRATEGIES FOR IMPROVING I: BOUNDED SEARCH TREES

- Buss's algorithm gives crudely a $2n + k^{k^2}$ algorithm for k-VC.
- Here is another algorithm: (DF) Take any edge $e = v_1 v_2$. either v_1 or v_2 is in any VC. Begin a tree T with first children v_1 and v_2. At each child delete all edges covered by the v_i.
- repeat to depth k.
- Gives a $O(2^k \cdot n)$ algorithm.
- Now combine the two: Gives a $2n + 2^k k^2$ algorithm.

- It is worth remarking that there are problems notably FPT by bounded search tree (type checking in ML) that are not known to have polynomial size kernels, and some “provably” don’t.
- Another easy example for bounded search trees is PLANAR INDEPENDENT SET. (Start with a degree 5 vertex, branching rule of size 6)

Pruning trees and clever reduction rules

- If G has paths of degree 2, then there are simple reduction rules to deal with them first. Thus we consider that G is of min degree 3.
  BRANCHING RULE VC2:
  If there is a degree two vertex v in G, with neighbours w_1 and w_2, then either both w_1 and w_2 are in a minimum size cover, or v together with all other neighbours of w_1 and w_2 are in a minimum size cover.
- Now when considering the kernel, for each vertex considered either v is included or all of its neighbours (at least) $\{p, q\}$ are included.
- Now the tree looks different. The first child nodes are labelled v or $\{p, q\}$, and on the right branch the parameter drops by 2 instead of 1. or similarly with the w_i case.

- Now the size of the search tree and hence the time complexity is determined by some recurrence relation.
- many, many versions of this idea with increasingly sophisticated reduction rules.
- This method has a 2005 (Fomin, Grandoni, Kratsch) incarnation called measure and conquer where the branching rules are given *rational valued* weights, and decisions as to what to do are figured out by optimization.
- For example the best exact algorithm for SET COVER and DOMINATING SET use this. (van Rooij-Bodlaender point out that this can be used for algoritm design as well.)
- Jianer Chen and others use this in many FPT algorithms such as the state of the art for FEEDBACK VERTEX SET and VERTEX COVER.

Shrink the Kernel

Theorem (Nemhauser and Trotter (1975))

For an n-vertex graph $G = (V, E)$ with m edges, we can compute two disjoint sets $C' \subseteq V$ and $V' \subseteq V$, in $O(\sqrt{n} \cdot m)$ time, such that the following three properties hold:

(i) There is a minimum size vertex cover of G that contains C'.
(ii) A minimum vertex cover for the induced subgraph $G[V']$ has size at least $|V'|/2$.
(iii) If $D \subseteq V'$ is a vertex cover of the induced subgraph $G[V']$, then $C = D \cup C'$ is a vertex cover of G.

Theorem (Chen et al. (2001))

Let $(G = (V, E), k)$ be an instance of k-Vertex Cover. *In $O(k \cdot |V| + k^3)$ time we can reduce this instance to a problem kernel $(G = (V', E'), k')$ with $|V'| \leq 2k$.*

- The current champion using this approach is a $O^*(1.286^k)$ (Chen01) The best is $O^*(1.2745^k)$(Chen10 using this, iterative compression, struction, measure and conquer, and other methods).
- Here the useful O^* notation only looks at the exponential part of the algorithm.

$2k$ can be gotten from IP relaxation

- For each $v \in V$ we have a variable x_v with values $\{0, 1\}$. Vertex Cover is then :

  (i) Minimize $\sum_{v \in V} x_v$ *subject to*
  ii) $x_u + v_v \geq 1$ *for each* $uv \in E$.

- The classical relaxation of this is to make it a linear programming problem by having x_v rational valued with $x_v \in [0, 1]$. Asking that the values be in $\{0, \frac{1}{2}, 1\}$.

$$S_1 = \{v \in V \mid x_v > \frac{1}{2}\}$$

$$S_{\frac{1}{2}} = \{v \in V \mid x_v = \frac{1}{2}\}$$

$$S_0 = \{v \in V \mid x_v < \frac{1}{2}\}.$$

- There is a minimal sized (integral) VERTEX COVER S with $S_1 \subseteq S \subseteq S_1 \sqcup S_{\frac{1}{2}}$, $S \cap S_0 = \emptyset$ and there $(G[S_{\frac{1}{2}}], k - |S_1|)$ is a kernel of size at most $2k$.

INTERACTIONS

- Now we can ask lots of questions. How small can the kernel be?
- Notice that applying the kernelization to the unbounded problem yields a approximation algorithm.
- Using the PCP theorem we know that no kernel can be smaller that 1.36 k unless P=NP (Dinur and Safra) as no better approximation is possible. Is this tight?
- Assuming the “Unique Games Conjecture” the $2k$ kernel is tight (Khot etc).
- Actually we know that no $O^*(1 + \epsilon)^k)$ algorithm is possible unless ETH fails.
- ETH n-valued 3SAT is not in DTIME$(2^{o(n)})$.

CROWN REDUCTION RULES

DEFINITION

A crown in a graph $G = (V, E)$ consists of an independent set $I \subseteq V$ and a set H containing all vertices in V adjacent to I.

- For example a degree 1 vertex and its neighbour is a crown.
- For a crown $I \cup H$ in G, then we need at least $|H|$ vertices to cover all edges in the crown.
- REDUCTION RULE CR:
  For any crown $I \cup H$ in G, add the set of vertices H to the solution set and remove $I \cup H$ and all of the incide nt edges of $I \cup H$ from G.
- Shrinkage $(G, k) \to (G', k - |H|)$.

How to use Crowns?

Theorem (Chor, Fellows, Juedes (2004))

If a graph $G = (V, E)$ has an independent set $V' \subset V$ such that $|N(V')| < |V'|$, then a crown $I \cup H$ with $I \subseteq V'$ can be found in G in time $O(n + m)$.

- Can get the crown: Take a maximal matching M of G. If $|M| > k$ say no. Else $I = G - M$ is an independent set $(\leq k)$, and then use bipartite matching to match I and its neighbours. Combinatorial arguments show that this has a submatching which is a crown. Delete and repeat.
- Other examples found in SIGACT News Gou-Niedermeier's survey on kernelization.

INTERLEAVING

- (Niedermeier and Rossmanith, 2000) showed that iteratively combining kernelization and bounded search trees often performs much better than either one alone or one followed by the other.
- Begin a search tree, and apply kernelization, then continue etc. Analysing the combinatorics shows a significant reduction in time complexity, which is very effective in practice.

An Example

- (NR) As an example, 3-Hitting Set (Given a collection of subsets of size 3 from a set S find k elements of S which hit the sets.) An instance (I, k) of this problem can be reduced to a kernel of size k^3 in time $O(|I|)$, and the problem can be solved by employing a search tree of size 2.27^k. Compare a running time of $O(2.27^k \cdot k^3 + |I|)$ (without interleaving) with a running time of $O(2.27^k + |I|)$ (with interleaving).
- Interesting and not yet developed generalization due to Abu-Khzam 2007 uses pseudo-kernelization. (TOCS, October 2007)

ITERATIVE COMPRESSION

- Reed, Smith and Vetta 2004. For the problem of “within k of being bipartite” (by deletion of edges).

DEFINITION (COMPRESSION ROUTINE)

A compression routine is an algorithm that, given a problem instance I and a solution of size k, either calculates a smaller solution or proves that the given solution is of minimum size.

An Example, VC again!

- $(G = (V, E), k)$, start with $V' = \emptyset$, and (solution) $C = \emptyset$.
- Add a new vertex v to both V' and C, $V' \leftarrow V' \cup \{v\},\ C \leftarrow C \cup \{v\}$.
- Now call the compression routine on the pair $(G[V'], C)$, where $G[V']$ is the subgraph induced by V' in G, to obtain a new solution C'. If $|C'| > k$ then we output NO, otherwise we set $C \leftarrow C'$.
- If we successfully complete the nth step where $V' = V$, we output C with $|C| \leq k$. Note that C will be an optimal solution for G. (Algo runs in time $O(2^k mn)$.)

- This was forst successflly applied by Reed, Smith, Vetta to GRAPH BIPARTITIZATION. The algorithm is similar, building a minimal bipartitization at each step and using what we can call acceptable partitions for the search step.
- The best now is $O^*(3.83^k)$, and it works better with algorithm engineering (Gray Codes, tree pruning) with (e.g.) biological data Hüffner 2004.
- it is a crucial step for the best two algorithms for VERTREX COVER (Chen, Kanj, Xia 2010, $O^*(1.2745^k)$ and FEEDBACK VERTEX SET (Can I remove k verteices and get a acyclic graph?) (Cao, Chen, Liu, 2009).

PRACTICE

- I remark that in practice these methods work much better than we might expect.
- Langston's work with irradiated mice, ETH group in Zurich, Karesten Weihe
- See The Computer Journal especially articles by Langston et al.

LESS PRACTICAL ALGORITHMS

- In what follows we look at algorithms that in general seem less practical but can sometimes work in practice.

Colour Coding

- k-Subgraph Isomorphism
  Instance: $G = (V, E)$ and a graph $H = (V^H, E^H)$ with $|V^H| = k$.
  Parameter: A positive integer k (or V^H).
  Question: Is H isomorphic to a subgraph in G?
- Idea: to find the desired set of vertices V' in G, isomorphic to H, we randomly colour all the vertices of G with k colours and expect that there is a colourful solution; all the vertices of V' have different colours.
- G uniformly at random with k colors, a set of k distinct vertices will obtain different colours with probability $(k!)/k^k$. This probability is lower-bounded by e^{-k}, so we need to repeat the process e^k times to have high probability of obtaining the required colouring.

Derandomization

- We need a list of colorings of the vertices in G such that, for each subset $V' \subseteq V$ with $|V'| = k$ there is at least one coloring in the list by which all vertices in V' obtain different colors.

Definition (k-Perfect Hash Functions)

A k-perfect family of hash functions is a family $\mathcal{H}$ of functions from $\{1, 2, ..., n\}$ onto $\{1, 2, ..., k\}$ such that, for each $S \subset \{1, 2, ..., n\}$ with $|S| = k$, there exists an $h \in \mathcal{H}$ such that h is bijective when restricted to S.

THEOREM (ALON ET AL. (1995))

Families of k-perfect hash functions from $\{1, 2, ..., n\}$ onto $\{1, 2, ..., k\}$ can be constructed which consist of $2^{O(k)} \cdot log\ n$ hash functions. For such a hash function, h, the value $h(i)$, $1 \leq i \leq n$, can be computed in linear time.

An example

- *k*-Path
- For each colouring h, we check every ordering $c_1, c_2, \ldots, c_k$ of the k colours to decide whether or not it realizes a k-path. We first construct a directed graph G' as follows:
  For each edge $(u, v) \in E$, if $h(u) = c_i$ and $h(v) = c_{i+1 \pmod k}$ for some i, then replace (u, v) with arc $\langle u, v \rangle$, otherwise delete (u, v).
  In G', for each v with $h(v) = c_1$, we use a breadth first search to check for a path C from v to v of length k.
- $2^{O(k)} \cdot \mathrm{l}og\,|V|$ colourings, and $k!$ orderings. k-path in time $O(k \cdot |V|^2)$.

Bounded Width Metrics

- Graphs constructed inductively. Treewidth, Pathwidth, Branschwidth, Cliquewidth mixed width etc. k-Inductive graphs, plus old favourites such as planarity etc, which can be viewed as local width.
- Example:

Definition

[Tree decomposition and Treewidth] Let $G = (V, E)$ be a graph. A tree decomposition, TD, of G is a pair $(T, \mathcal{X})$ where
1. $T = (I, F)$ is a tree, and
2. $\mathcal{X} = \{X_i \mid i \in I\}$ is a family of subsets of V, one for each node of T, such that

(i) $\bigcup_{i \in I} X_i = V$,
(ii) for every edge $\{v, w\} \in E$, *there is an* $i \in I$ *with* $v \in X_i$ *and* $w \in X_i$, *and*
(iii) for all $i, j, k \in I$, *if* j *is on the path from* i *to* k *in* T, *then* $X_i \cap X_k \subseteq X_j$.

- This gives the following well-known definition.

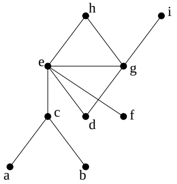
DEFINITION

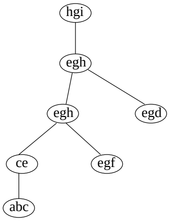
The width of a tree decomposition $((I, F), \{X_i \mid i \in I\})$ is $\max_{i \in I} |X_i| - 1$. The treewidth of a graph G, denoted by $tw(G)$, is the minimum width over all possible tree decompositions of G.

THE CANONICAL METHOD

- The following refers to any of these inductively defined graphs families. Notes that many commercial constructions of, for example chips are inductively defined.
  1. Find a bounded-width tree (path) decomposition of the input graph that exhibits the underlying tree (path) structure.
  2. Perform dynamic programming on this decomposition to solve the problem.

An example for Independent Set





$\emptyset$	a	b	c	ab	ac	bc	abc
0	1	1	1	2	-	-	-

Bodlaender's Theorem

- The following theorem is shows that treewidth is FPT. Improves many earlier results showing this. The constant is about 2^{35k^3}.

Theorem (Bodlaender)

*k-*Treewidth *is linear time FPT*

- Not practical because of large hidden O term.
- Unknown if there is a practical FPT treewidth algorithm
- Nevertheless approximation and algorithms specific to known decomps run well at least sometimes.

Linear Integer Programming

- There have been some (at least theortical) applications on IP with bounded variables.

Theorem (Lenstra)

Integer programming feasibility can be solved with $O(p^{\frac{9p}{2}}L)$ arithmetical operations in integers of $O(p^{\frac{9p}{2}}L)$ bits where p is the number of input variables and L is the number of input bits for the LIP instance.

- I don't know much about this but you can look at Rolf Niedermeier's book (Invitation to Fixed Parameter Algorithms)
- Mostly impractical.

METATHEOREMS

- (First order Logic)
  1. **Atomic formulas:** $x = y$ and $R(x_1, ..., x_k)$, where R is a k-ary relation symbol and $x, y, x_1, ..., x_k$ are individual variables, are FO-formulas.
  2. **Conjunction, Disjunction:** If ϕ and ψ are FO-formulas, then $\phi \wedge \psi$ is an FO-formula and $\phi \vee \psi$ is an FO-formula.
  3. **Negation:** If ϕ is an FO-formula, then $\neg\phi$ is an FO-formula.
  4. **Quantification:** If ϕ is an FO-formula and x is an individual variable, then $\exists x\ \phi$ is an FO-formula and $\forall x\ \phi$ is an FO-formula.
- Eg We can state that a graph has a clique of size k using an FO-formula,

$$\exists x_1 ... x_k \bigwedge_{1 \leq i \leq j \leq k} E(x_i, x_j)$$

Monadic Second Order Logic

- Two sorted structure with variables for sets of objects.
- 1. **Additional atomic formulas:** For all set variables X and individual variables y, Xy is an MSO-formula.
  2. **Set quantification:** If ϕ is an MSO-formula and X is a set variable, then $\exists X\ \phi$ is an MSO -formula, and $\forall X\ \phi$ is an MSO-formula.
- Eg k-col

$$\exists X_1,,,\exists X_k\left(\forall x \bigvee_{i=1}^{k} X_i x \wedge \forall x \forall y\left(E(x,y) \rightarrow \bigwedge_{i=1}^{k} \neg(X_i x \wedge X_i y)\right)\right)$$

Model Checking

- Instance: A structure $\mathcal{A} \in \mathcal{D}$, and a sentence (no free variables) $\phi \in \Phi$.
  Question: Does $\mathcal{A}$ satisfy ϕ?
- PSPACE-complete for FO and MSO.

Courcelle's and Seese's Theorems

Theorem (Courcelle 1990)

The model-checking problem for MSO restricted to graphs of bounded treewidth is linear-time fixed-parameter tractable.

Detleef Seese has proved a converse to Courcelle's theorem.

Theorem (Seese 1991)

Suppose that $\mathcal{F}$ is any family of graphs for which the model-checking problem for MSO is decidable, then there is a number n such that, for all $G \in \mathcal{F}$, the treewidth of G is less than n.

Local Treewidth

- $ltw(G)(r) = \max \{tw(N_r(v)) \mid v \in V(G)\}$ where $N_r(v)$ is the neighbourhood of radius r about v.
- A class of graphs $\mathcal{C} = \{G : G \in D\}$ has bounded local treewidth if there is a function $f : \mathbb{N} \to \mathbb{N}$ such that, for $r \geq 1$, $ltw(G)(r) \leq f(r)$i, for all $G \in \mathcal{C}$.
- Examples Bounded degree, bounded treewidth, bounded genus, excluding a minor

The Frick Grohe Theorem

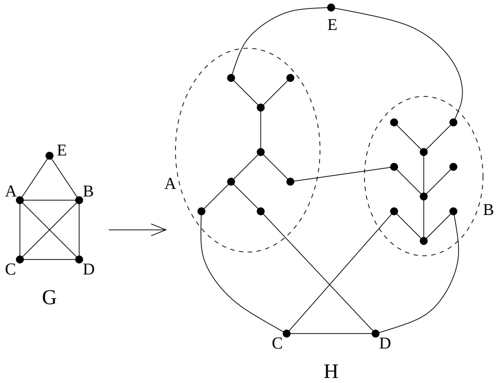
Theorem (Frick and Grohe 1999)

Parameterized problems that can be described as model-checking problems for FO are fixed-parameter tractable on classes of graphs of bounded local treewidth.

For example Dominating set, Independent Set, or Subgraph isomorphism are FPT on planar graphs, or on graphs of bounded degree

More exotic methods

- minor ordering



  - Robertson-Seymour Finite graphs are WQO's under minor ordering. $H \leq_{minor} G$ is $O(|G|^3)$ FPT for a fixed H.
- THEOREM (MINOR-CLOSED MEMBERSHIP)

  If $\mathcal{F}$ is a minor-closed class of graphs then membership of a graph G in $\mathcal{F}$ can be determined in time $O(f(k) \cdot |G|^3)$, where k is the collective size of the graphs in the obstruction set for $\mathcal{F}$.
  - Likely I won't have time to discuss what this means but see DF for more details.

Other work

- There has been a lot of recent work exploring the bad behaviour of the algorithms generated by the metatheorems
- Including work by Grohe and co-authors showing that the iterated exponentials cannot be gottent rid of unless P=NP or FPT=W[1] in the MSO case and the local treewidth case respectively.
- Including work of Bodlaender, Downey, Fellows, and Hermelin showing that unless the polynomial time hierarchy collapses no small kernels for e.g. treewidth, and a wide class of problems.
- Still much to do.

Some questions

- Commercially many things are solved using SAT solvers. Why do they work. What is the reason that the instances arizing from real life behave well?
- How to show no reasonable FPT algorithm uaing some assumption?
- Develop a reasonable randomized version, PCP, etc. This is the “hottest” area in TCS yet not really developed in parameterized complexity. (Moritz Meuller has some nice work here)

Some References

- Parameterized Complexity, springer 1999 DF
- Invitation to Parameterized Algorithms, 2006 Niedermeier, OUP
- Parameterized Complexity Theory, 2006, Springer Flum and Grohe
- Theory of Computing Systems, Vol. 41, October 2007
- Parameterized Complexity for the Skeptic, D, proceedings CCC, Aarhus, (see my homepage)
- The Computer Journal, (ed Downey, Fellows, Langston)
- Confronting intractability via parameters, Downey Thilikos, Computing Reviews
- Fundamentals of Parameterized Complexity, Downey-Fellows, this year.

WHAT SHOULD YOU DO?

- You should buy that wonderful book...(and its friends)
- Thank You