What have I been thinking about?

Rod Downey
Victoria University
Wellington
New Zealand

Shonan, May 2013

The New Book!

- First and foremost Mike and I have been writing Fundamentals of Parameterized Complexity
- It is nearly done; I have brought a draft.
- End of May.
- I need exercises. Any (helpful) comments needed.
- Here is the TOC.
- download and comments: http://homepages.ecs.vuw.ac.nz/~downey/maindoc.pdf

Some other things

- Metatheme-Goal is truly useful FPT.
- Dimitrious pointed out that the right parameter is not always obvious, but also
- Asking the right question is not always obvious For example: Mike Langston and Mike Fellows suggested the following as a correct formulation:
  Input: I and a current solution S
  Parameter: k
  Question: Is there a better solution with k steps of S?
- Most things are hard with this set up. Perhaps first observed by Joing Guo that they work by taking a carefully designed I and a silly S and making sure that a better solution must be no more than k steps of S, to code in, say, Clique.
- e.g. Dominating Set

Permissive Local Search

- First noted by Marx;
  Input: I and a current solution S
  Parameter: k
  Question: Can we find a better solution S' possibly an arbitrary distance from S, or, determine that there is no better solution within k steps of S?
- Obvously related to parameterized approximation.
- And related to the following work which is
- joint with Judith Egan, Mike, Fran and Peter Shaw.
- Solution repair:

Solution repair

- Long ago Mike, Catherine McCartin and I tried
  Input: I and a defective current solution S
  Parameter: k
  Question: Is there a solution S' within k steps of S?
- Obvious candidate DOMINATING SET
- Alas $S = \emptyset$. But,...
  Input: I with a solution S and a modified instance I' with $d_1(I, I') \leq k$.
  Parameters: k, r
  Question: Can we find a solution S' for I' with $d_2(I, I') \leq r$.
- Rhetoric: Who knows how we got S, and we were happy. We don't even remember the algorithm. Perhaps very expensive. Perhaps costly to get S and far too costly to start again. e.g. cell phone stations, hamburgers franchises, scheduling, DNA.

Theorem

Edge Dominating Set Repair *is FPT (I.e. d_1 is number of edges deleted, d_2 is vertices to be added.*

- We can think of only deletions. The failure of S to dominate G' involves $2k$ vertices of G.
- let H denote the subgraph induced by these $2k$ affected vertices.
- There's no point in deleting things from S, so we restrict ourselves to $G \setminus (H \cup S)$
- There are only 2^{2k} different types (by their neighbourhoods in the affected H), and we would add to S at most one for each type.
- This gives a expnential kernel.

No poly kernel

- It has no poly kernel unless co-NP$\subseteq$ NP/Poly.
- Usual methods, in particular using Dorn, Lokshtanok, Saurabh colours and ID's.

Hand in Hand with Heuristics

- Hartung and Niedermeier had a very good idea of using FPT algorithms in a subroutine for incremental computation.
  Incremental k-List Colouring
  Input: $G = (V, E)$ and k-list colour for $G[v \setminus \{x\}]$ and $c \in \mathbb{N}$.
  Parameters: k
  Question: Is there a k-list colouring f' of G such that

  $$|\{v \in V \setminus \{x\} : f(v) \neq f'(v)\}| \leq c.$$

  Here a list colouring must choose from $L(v) \subset \{1, \ldots, k\}$. A k colouring is simply $L(v) = [k]$.
- This is NP-complete and even $W[1]$-hard for bounded treewidth. This is true either for the parameter t, or c, but
- for (t, c) this is FPT $O(k(k-1)^c|V|)$, and no poly kernel unless collapse.

- This is the good news. A standard benchmark implementation of graph colouring uses iterated greedy.
- The observation is that usually the conservation value c is small. using this as a subroutine resulted in significant improvement of performance of heuristics.
- Experimentally, average 11% imporvement, for $k \leq 117$ $c \leq 8$.
- There are so many heuristics around that perhaps this is a general methodology.

Hand in Hand with Heuristics

- Hartung and Niedermeier's work leads to a new program.
- (Mike's name) Turbo-Charging Greedy Heuristics With Appropriate Incremental FPT Subroutines
- The recipe is: start with a concrete greedy heuristic, and from that articulate an **inductive route** such as one might use for iterative compression, except that here we don't care about keeping the solution small. Then define a hopefully FPT parameterized problem that can be used when you hit a snag to try to keep the cost of the solution from increasing.
- These three things are all very similar in the end:
  – iterative compression
  – turbo-charging greedy heuristics with FPT subroutines
  – the heuristic scheme proposed by Karp in his article on algorithms in computational biology (which looked a lot like iterative compression)

- The program implicit in Hartung-Niedermeier: backwards — start with the "classical" greedy heuristic. In the case of GRAPH COLORING, the champion heuristic that they are *FPT-turbocharging* works along the lines: Order the vertices, according to descending vertex degree (so things get easier later in the ordering).
- So then you move along (think of *iterative compression*, but preferably think of this technique abstractly — it is really just any sort of **inductive route** that leads eventually to the input you were asked to deal with) from one step to the next, carrying along a solution to be, hopefully, efficiently modified (in the case of *iterative compression*: compressed, *exactly*).

- in HN, list of vertices considered on the inductive route tends towards lower degree vertices (which then have longer lists of possible colorings, since fewer are forbidden by the "current" solution).
- Hopefully, there are plenty of colors available, so you just use one of them. But with the non-turbocharged heuristic, maybe at a given step, you are forced to use a "new" color.
- But the program here investigates if the previous coloring, can be fiddled with a bit (the parameter, here called *conservation*), so that you don't have to use a new color.
- This is the essence of the FPT turbocharging.
- Also should be comnined with measure and conquer, perhaps, inductive rules as to what to do.

- It is best to see their incremental problem, as the parameterized incremental problem that arises from an effort to FPT-turbocharge the greedy algorithm.
- Our "forward reading" FPT result about DOMINATING SET.
- But how does the problem we address arise in a "backward reading", starting from a greedy heuristic for DOMINATING SET that we wish to turbocharge in a similar manner?
  First problem: what is a reasonable greedy heuristic for DOMINATING SET?
- There is likely literature on this. But if you think on this a little bit, the following greedy heuristic seems moderately reasonable:

- **Dominating Set: Greedy Plan A**
  (1) Order the vertices of G from small degree to large degree.
  (2) At each step (addition of the next vertex) ... on our way to G (the inductive route), if the new vertex v is not already dominated, then add the highest degree vertex in $N[v]$.
  This seems like the right idea for the *greedy underlying template* because it at least captures the *degree 1 vertex kernelization rule*. (There may be general connections between smart FPT kernelization, and smart greedy heuristics, not yet explored, but possibly programmatic, in a big way.)

- **Dominating Set: Greedy Plan B**
  The goal here is to try to "reverse engineer" our incremental FPT result about DOMINATING SET to be interpretable in the above way — as naturally arising from a greedy algorithm we wish to FPT-turbocharge.
  Our result has d_1 being edge-edit distance, so this needs to be respected in the **inductive route** (like with iterative compression).
- So here the plan is to mirror the "movie" of Plan A, starting with the complete graph on n vertices, and gradually delete edges, eventually to obtain G. In the beginning, the current solution obviously has size 1, and as edges are gradually deleted, it is a compression problem. When the thing gets stuck, call in the incremental FPT algorithm.
  The incremental problem derived in this way is:
  INPUT: G, e, S where e is an edge, and S is a dominating set of G but not $G - e = G'$
  PARAM: k
  QUESTION: Can we find S' such that $d_v(S, S') \leq k$, $|S| = |S'|$ and S' is a dominating set for G'?

- another experiment: FFEDBACK VERTEX SET
- Need to start with a concrete greedy heuristic. The following seems reasonable:
  (a) Order the vertices from high degree to low degree. Use this to make an initial movie M of how to build G, adding to the picture one vertex per step.
  (b) Turn this into an edge-addition movie; start with the empty graph on n vertices, and add edges, as directed by M, to build G. This is the inductive route we will use.
  (c) If adding the edge e creates a unique uncovered cycle, choose the vertex of the cycle of highest degree. If multiple cycles involving e are created, then choose the endpoint of highest degree.
- This leads to the incremental problem:
  INPUT: G, e, S where S is an fvs for G, but not for $G' = G + e$.
  PARAM: k
  QUESTION: Is there an fvs S' for G' with $d_v(S, S') \leq k$ and $|S'| = |S|$?

- Other unformed thoughts
- We know that H a subgraph of G is FPT for fixed H by e.g. Plehn-Voight, but is this a dichotomy? So if $\{H_i \mid i \in \omega\}$ is a family of graphs of uniformly bounded treewidth then determining H_i a subgraph of G with parameter H_i is FPT.
- But is it true that if $\{H_i \mid i \in \omega\}$ is a family of unbounded treewidth, is determining H_i a subgraph of G with parameter H_i $W[1]$-hard.
- Parameterizing by bounds on rationals in various settings?
- Replacing smoothness?
- Also parameterised parity games. (Björklund, Sandberg, Vorobyov)
- Randomization again and an analog of Toda?

Thank You