Fixed-Parameter Tractability, A Prehistory*,**

A Festschrift Contribution Devoted to Michael R. Fellows on the Occasion of his 60th Birthday

Michael A. Langston

Department of Electrical Engineering and Computer Science
University of Tennessee
Knoxville, TN 37996-2250
USA
langston@eecs.utk.edu

1 Overview

Many of the foundational parameterized tenets discussed in this festschrift actually predate by over a decade the first systematic treatments of fixed-parameter tractability. In this frank, firsthand account I will, to the best of my recollection, describe some of the earliest research avenues Mike Fellows and I pursued that would turn out later to be highly relevant to parameterized complexity. Although we did not know it at the time, these were the origins and formative years of this burgeoning new field. Readers unfamiliar with the history of fixed-parameter tractability may be surprised to learn that its initial motivations arose from, of all things, automation and optimization for integrated circuit design.

2 A Fortuitous Collaboration

I first met Mike Fellows sometime in the spring of 1985, the year he completed his PhD in Computer Science at the University of California, San Diego. I was then chairing the faculty search committee at Washington State University, where we were fortunate enough to interview and hire him. Mike and I hit it off immediately. We had similar research interests in graph theory, combinatorics and optimization. We both had families with small children. And we both had even served in paratrooper assignments with the US military (Mike as an enlisted man in the Air Force, I as an officer in the Army). As luck would have it, I happened to be working at the time on a spectrum of combinatorial problems motivated by

* Prehistory (from the Latin, with *præ* meaning before, and *historia* meaning story) is often defined as the period before a story is recorded. And that is what this tale is all about. It is an account of the genesis of fixed-parameter tractability, before the field had its terminology or even its name.

** This narrative account was made possible in part by the National Science Foundation under grants MIP-8703879 and MIP-8919312, and by the Office of Naval Research under contract N00014-90-J-1855.

H.L. Bodlaender et al. (Eds.): Fellows Festschrift, LNCS 7370, pp. 3–16, 2012.

circuit layout problems in very large scale integration (VLSI) design. Meanwhile, Mike had been reading the recent work of Neil Robertson and Paul Seymour on Wagner's Conjecture and what is now known as the Graph Minor Theorem. It was a perfect confluence of technologies and ideas. As soon as Mike arrived on campus, we put our heads together, compared notes, and decided to look around the VLSI domain to see if we could find any interesting lower ideals in the minor order. There was nothing particularly out of the ordinary with so inauspicious a start. I think we viewed it as a fairly routine academic exercise. We were merely searching for new research horizons. We could scarcely have foreseen where this journey would eventually take us. Thus it all began.

3 Research Atmosphere

I think the algorithmic landscape at that time was relatively complacent. Most problems of interest had already been found either to reside in $\mathcal{P}$ or to be $\mathcal{NP}$-complete. Thus, natural problems were largely viewed under the classic Jack Edmonds style dichotomy as being good or bad, easy or hard, with not much of a middle ground. Most of our colleagues in the theoretical computer science community seemed pretty satisfied with this simple picture. Exhaustive exact techniques and heuristic approximation algorithms remained the stalwart analytical weapons of choice. I confess that I had in fact worked on the worst-case analysis of scheduling heuristics myself as part of my PhD dissertation a few years earlier. Nevertheless, new possibilities beckoned. Mike and I often wondered out loud why all $\mathcal{NP}$-complete problems were generally being tarred with the same brush. It made little sense to us. Sure, there were notions like strong $\mathcal{NP}$-completeness [8]. But where was any systematic sort of focus on the parameter effect? Even well-known $\mathcal{NP}$-complete problems like independent set and graph coloring were manifestly different, starting with parameters set as low as 3. Sorting out differences in problem difficulty based on parameter specifications was a theme that would keep working its way into, and eventually take over, much of our research program.

4 Changes Brewing

Meanwhile, things were quickly evolving in the worlds of extremal graph theory and well-quasi-ordered sets. Mike and I were I think a little awe struck with the possibilities. With mere checks on various forms of graph containment, one could now quickly show that many seemingly intractable problems, some not previously even known to be decidable, were actually in $\mathcal{P}$. Wow!

But let me not get ahead of myself. In what follows I shall consider only finite, undirected graphs. H is a *minor* of G if a graph isomorphic to H can be obtained from a subgraph of G by contracting edges. A family F of graphs is said to be *closed* in the minor order if every minor of a graph in F must also reside in F. A graph is an *obstruction* to F if it lies in F's complement and all its proper minors lie in F. Surely the most classic and widely-known example

is planarity, where F denotes the set of planar graphs, which is minor closed, and where the obstructions have long been known to be $K_{3,3}$ and K_5. In the interest of historical accuracy, it should probably be pointed out that planarity was first studied in the *topological* order [12]. (H is topologically contained in G if a graph isomorphic to H can be obtained from a subgraph of G by removing subdivisions.) The obstruction sets for both orders are the same.

The most useful tool to us back then was this:

Theorem 1. [15] *Any family of finite graphs that is closed in the minor order and that excludes a planar graph can be recognized in polynomial time.*

At this point a simple example of a family amenable to this powerful theorem may be helpful. So let's consider vertex cover, which is probably the most widely studied problem in all of parameterized complexity. Let k denote any fixed positive integer. I will leave it as an exercise for the reader to check that (1) if G has a vertex cover of size k, then so does any of its minors and (2) there are planar graphs whose vertex cover size exceeds k. Thus, by Theorem 1, there is a polynomial-time algorithm to decide whether an arbitrary graph has a vertex cover of size k. It is important to note that there will be a different algorithm for each fixed k. Of course vertex cover is solvable by brute force in $O(n^k)$ time anyway, but soon Theorem 1 was superseded by even better tools [16,17,18], limiting the degree of the polynomial to 3. Now things are beginning to sound a lot like what would eventually become fixed-parameter tractability. Incidentally, these improvements also eliminated the need for planar exclusion, and added the *immersion* order to the mix (H is immersed in G if a graph isomorphic to H can be obtained from a subgraph of G by lifting pairs of edges). But I digress; let me return to the main story.

To implement the algorithm ensured by Theorem 1, one needs only to take as input an arbitrary graph and test whether it contains any of the family's obstructions. If it does, then the algorithm is to answer "no." Otherwise, it is to answer "yes." The run time guarantee comes from the facts that containment for each obstruction can be tested in polynomial time, and that only a finite battery of such tests is ever necessary[3].

Of course there are substantial prices to be paid in applying such stunningly abstract and powerful strategies. The specific algorithms provided by these emergent tools impose ginormous and utterly impractical constants of proportionality. Worse yet, they are inherently non-constructive in that each requires complete knowledge of the relevant obstruction set. (I will have more to say on these issues later.) Despite such grave practical drawbacks, however, these dramatic developments helped reinforce our misgivings about any homogeneity among $\mathcal{NP}$-complete problems. Indeed, it seemed more and more plausible to us that

[3] An order is a well-quasi-order if any infinite collection of objects must have a pair of comparable elements under that order. By what is now called the Graph Minor Theorem, we know that finite, simple graphs are well-quasi-ordered under minors. Thus, because obstructions are incomparable (they form an antichain), their number must be finite.

not all $\mathcal{NP}$-complete problems were created equal, and that many might even be sliced into easier sub-problems with the use appropriate parameters. Mike and I frequently discussed the need for some sort of non-uniform notion of problem complexity, usually by keeping a problem constant but applying a distinct algorithm for each parameter value. Of course this is precisely what we do today with fixed-parameter algorithms, but none of that was formalized back then.

5 Shoulders of Giants

In our many interactions with Neil Robertson and Paul Seymour, it became increasingly clear that they had no more than a marginal interest in practicalities, applications and implementations. And why should they? They were hot on the trail of exceedingly profound advances in graph theory. Yet serious practical applications were some of the very topics that Mike and I felt were of central relevance to computation. So off we went, with Neil's and Paul's blessing. We had a clear field in which to plow our furrows. I think it was pretty natural for us to feel rather inadequate when stacked up against their deep and beautiful work (as well as their joint efforts with Robin Thomas and others). In this Mike was always self-effacing and humble, ever quick to point out that by performing algorithm mining on well-quasi-order theory we were standing on the shoulders of giants. One might even argue that Mike held out this attitude with a just bit too much zeal, as he often wound up unfairly downplaying or even dismissing a great deal of his own original and creative work in the process.

6 Armchair Polynomial Time

It wasn't long before Mike and I started being invited around the country to speak on our work. We took the Mike and Mike show on the road with alacrity. At about this time I went on sabbatical leave at the University of Illinois, but I wound up away on travel at least as much as I was in Champaign. Nonconstructive algorithm design tools were a new thing, and many wanted to find out how they worked. On one jaunt, Mike was invited to give a talk at Princeton, where he spoke to a computer science audience. There he humorously christened our earliest effort "armchair polynomial time." As he rightly pointed out, it was sometimes embarrassingly easy to apply Theorem 1 and its extensions. To reinforce the point, he ended the presentation of each proof with a tiny glyph of an armchair. I liked it!

As previously mentioned, fixed k vertex cover is a handy example of an exceedingly straightforward application of Theorem 1. With much subsequent work, however, vertex cover has now migrated from $O(n^k)$ to $O(n^3)$ to $O(n \log n)$ and of course now all the way down to $O(n)$. See, for example, [1]. Many other problems have followed this trajectory. Probably a better example, therefore, and one to showcase the truly astonishing power of this general approach, is knotlessness [5]. Here we are asked whether a graph can be embedded in three dimensional space so that none of its cycles are knotted. This sounds quite difficult. Given

an arbitrary graph, no method is known just to bound the number of its embeddings that must be tested. Thus, without recourse to Theorem 1, knotlessness is in no obvious way even known to be decidable. With the use of minor closure, however, it is not at all difficult to show:

Theorem 2. [5] *Knotlessness can be decided in polynomial time.*

Notorious problems like knotlessness and others of its ilk (e.g., linklessness [5]) are further distinguished from the rest of the problems I will discuss by the fact that they have no obvious associated parameter(s). Thus, while a problem like vertex cover has a distinct finite obstruction set, and hence a distinct algorithm, for every fixed cover size, knotlessness has but a single forbidden set. (Of course one could probably parameterize by, say, the number of knots or the dimensionality of the embedding space, but as far as I know no one has looked into that.)

7 Circuit Layout Applications

While Mike and I found applications across many domains, we noticed early on that the field of circuit design abounds with combinatorial problems amenable to this general approach. One of the first problems we studied, and the only one I will discuss in any detail here, is gate matrix layout. This style was introduced in [14] for CMOS circuits. Solving the problem at its heart was, and is, a central step in circuit synthesis. It was known at the time that gate matrix layout was equivalent to various problems encountered in multiple PLA folding and the use of Weinberger and one-dimensional logic arrays [4]. Gate matrix layout was later shown also to be equivalent to pathwidth [6], vertex separation number [10] and several other graph metrics.

The problem can be stated as follows. We are given a Boolean matrix M and a positive integer k. We are asked whether the columns of M can be permuted so that, if in each row every 0 lying between the leftmost and rightmost 1 is changed to $*$, no column contains more than k 1's and $*$'s. In this formulation, rows denote electrical circuits, columns denote gates, and a $*$ represents the fact that all gates within a circuit must be physically connected. Circuits are not permitted to overlap within a track. Minimizing the maximum number of 1's and $*$'s in any column, over all column permutations, therefore corresponds to minimizing the number of tracks and hence the area utilized in circuit realization.

Gate matrix layout is $\mathcal{NP}$-complete [9]. Despite many years of study, however, no algorithm is known to approximate it to within a multiplicative constant. Nor can it be approximated to within an additive constant, unless $\mathcal{P} = \mathcal{NP}$ [2]. In order to apply Theorem 1, our first task is to transform an arbitrary instance into a graph. To accomplish this, we expand a given matrix in the following manner. We replace any column with more than two 1's with a set of columns, each with only two 1's, representing all the possible ways to choose two 1's from that column. Next, we derive from this expanded matrix a finite simple graph, where we treat rows as vertices and columns as edges. Proofs for the next three results are rather tedious.

Lemma 1. [4] *Matrix expansion does not affect the cost of a gate matrix layout solution.*

Lemma 2. [4] *For every fixed k, the "yes" family of derived graphs is minor closed.*

Lemma 3. [4] *For every fixed k, the "no" family of derived graphs contains a planar element.*

So we apply the three preceding lemmas, fortify them with Theorem 1, and now bring on the armchair.

Theorem 3. [4] *For every fixed k, gate matrix layout is solvable in polynomial time.*

We went on to find quite an assortment of well-known layout problems for which we could prove analogs of Theorem 3. Just a short list would include disk dimension, minimum cut linear arrangement, topological bandwidth, crossing number, maximum leaf spanning tree, search number and two dimensional grid load factor. See [7].

Thus, "trolling" for applications using this general approach became for us a pretty standard recipe:

- look around for a provably difficult problem,
- fix parameter(s) as required,
- make sure it can't now be solved by brute force or table lookup,
- devise problem transmogrifications as needed, and
- check for minor or immersion closure (an armchair is highly recommended at this step).

This line of work was really quite addictive. When our quest was successful, it usually turned out that the "yes" family was the one that was closed. But there were exceptions. Consider, for example, longest path. For every fixed k, the "no" family is actually minor closed. Moreover, most of the time obstructions were highly elusive, requiring enormous effort for comprehensive analysis (see, for example, [11]). But again there were exceptions. And again consider longest path. For every fixed k, the only obstruction to "no" family membership is a path of length k.

8 What the Hell Is VLSI?

At about this time Mike and I wrote several proposals, some successful and some not. One of our early successes was at the National Science Foundation, where we proposed to study the application of Theorem 1 along with other novel methods to combinatorial problems of relevance to VLSI design (hence our initial work

on gate matrix layout). There the current program director, Bob Grafton[4], was most helpful and quick to sense the potential of these remarkable tools. After describing them to him over the phone, I liked his reactions so much that I more or less paraphrased one of his responses in our proposal abstract. Here is that abstract, from late 1985, with emphasis added to the sentence that I think best reflects Bob's statement to me:

> *In the design and manufacturing of Very-Large-Scale Integrated (VLSI) systems, practical problems are characterized by fixed-parameter instances. For example, a parameter might represent the number of tracks permitted on a chip, the number of memory cells available, the number of processing elements to be employed, or other variables significant to the solution of the problem at hand.* ***In fixing the value of such parameters, we focus on the physically realizable nature of the system rather than on the purely abstract aspects of the model.*** *In this investigation, research efforts are concentrated on this central, practical feature of real VLSI design problems, whose domains span the spectrum from the gate to the systems architecture levels. Powerful and, in many cases, emergent techniques from the fields of complexity, graph and group theory are brought to bear on these fixed-parameter problems so as to yield exact or guaranteed approximate solutions.*

Observe from this that we were employing terms such as "fixed-parameter instances" and "fixed-parameter problems" well over a dozen years before fixed-parameter tractability had been formalized, codified and systematically presented [3] by Mike along with Rod Downey, who is also slated to contribute to this festschrift.

By this time Mike (and I as well, probably) had become much more caught up in the revolutionary algorithmic techniques than in the circuit problems themselves. So much so that by the time I learned that this particular proposal was funded and told Mike, he looked at me excitedly and said[5] "Wow that's really great! But what the hell is VLSI?" This was surely a curious remark from a co-PI on a proposal written to perform research on, drum roll please, VLSI. In fairness, we had probably not discussed that proposal since we had written it several months earlier. And I knew of course what Mike meant by his statement. The way he said it, however, made us both laugh out loud. Mike has always been singularly quotable.

9 Constructive Complexity

From the beginning, Mike and I recognized the need to address the show-stopping shortcomings of applications stemming from remarkable results such as Theorem

[4] We are indebted both to Bob Grafton (NSF) and to Ralph Wachter (ONR), and of course to their anonymous review panels, for their early feedback, understanding and recognition of the long-term potential for what eventually has become the field of parameterized complexity.

[5] I believe this quote is verbatim. It seems like only yesterday.

1. For one thing, the algorithms had staggering constants of proportionality. A little thought experiment may be useful here. Pick your favorite enormous constant such as, let's say, some reasonable upper bound on the number of fish in the sea. As it turns out, that's too small. So pick, say, a bound on the number of grains of sand on all the world's beaches. That's still much too small. All right then, let's try a bound on the number of elementary particle interactions that could possibly have occurred in the lifetime of the known universe. That's still too small! Those sorts of huge numbers are rapidly dwarfed by the "towers of two" constants employed in the algorithms of Robertson and Seymour. In this context, I am often reminded of a relevant statement perhaps dubiously attributed to Disraeli: "There are three kinds of lies: lies, damned lies, and statistics." Around this time I began pointing out to my graduate students that there are in fact at least four kinds of lies: lies, damned lies, statistics, and the big "oh" notation. After all, what does it really mean to claim you have, say, even a linear-time algorithm when its constant of proportionality is so outrageous? Mike and I subsequently worked for quite some time on a variety of solutions for this issue. We were generally able to mitigate constants greatly (plus reduce degree bounds), mainly through the use of graph width metrics.

Another weakness of applications based solely on Theorem 1 was the lack of any techniques for search or optimization. This theorem and its subsequent improvements provide algorithms for decision only. So Mike and I worked on this too. We were generally (but interestingly, not always) able to reduce search and optimization to decision, mainly through self reduction. These results are discussed in detail in [6].

A third shortcoming, an egregious one, and the only algorithmic deficiency I'll address in detail here, was non-constructivity. Theorem 1 and its analogs provide no general means for finding (or even recognizing) the promised algorithms. All we are told is that such algorithms must exist. So Mike and I set out to find ways to remedy this situation. How could inherently non-constructive tools ever be made constructive? It was a puzzler!

After many false starts, the constructivization methods we finally devised operate in a rather counterintuitive, and perhaps even paradoxical, fashion. We were able to prove that we could, in principle, write down an algorithm. We could show that it was correct. We knew it relied on the finiteness of its obstruction set. And we could watch it run as long as we liked. Yet we also could show that we could never know for sure the obstruction set itself and, sometimes, we could not even know an exact bound on the algorithm's running time.

The following is a greatly simplified version of a much more general result from [6], where we deal with arbitrary well-quasi-orders. I have stripped it down and restricted it here; otherwise I would need to introduce several fairly cumbersome definitions.

Theorem 4. [6] *Let F denote a closed family in the minor or immersion order. If the following are available*

1. *a solution checking algorithm that runs in* $O(T_1(n))$ *time,*
2. *order tests that need at most* $O(T_2(n))$ *time, and*
3. *a self-reduction bounded by* $O(T_3(n))$ *time,*

then $O(max\{T_1(n), T_2(n) * T_3(n)\})$ *time is sufficient to solve the decision version of* F*-membership.*

I should probably explicate with an example. So let me select one familiar to most readers, and turn again to vertex cover. As previously observed, the "yes" family for any fixed k is minor closed. First, if a putative solution is proffered, its correctness can be checked in at most quadratic time. All one has to do is delete the vertices in the supposed cover and look to see if the resulting graph is edgeless. Second, we know that this family excludes a planar graph. We can therefore test whether an arbitrary input graph contains a fixed obstruction in at most $O(n \log n)$ time via a bounded treewidth argument [6]. And third, if a decision oracle reports that an input is a "yes" instance, we can self-reduce to a solution in a variety of ways. For example, we can iteratively eliminate each vertex v in turn, and re-invoke the decision oracle for $k - 1$. If and only if this oracle says "yes" do we mark v as a member of a satisfying cover and decrement k by 1. Thus, at most a linear number of self-reduction calls[6] are required. Theorem 4 therefore guarantees that the entire procedure can be accomplished in $O(max\{n^2, (n \log n) * n\}) = O(n^2 \log n)$ time.

The vast majority of problems known to be amenable to Theorem 1 are also amenable to Theorem 4. In fact, Theorem 4 generally gives us not just a constructively known algorithm, but a constructively known polynomial-time algorithm. This is because polynomial-time checking usually comes from membership in $\mathcal{NP}$, fast tests are available for both orders, and self-reduction is most always possible. Only those problems otherwise not even known to be decidable seem resistant to this constructivization. Intriguingly, the algorithm guaranteed by Theorem 4 relies on the correctness and finiteness of F's obstruction set — but we can never use it to learn the set! This is because no finite amount of observation will ever tell us whether the algorithm has found all the set's elements.

Well that seems rather curious. What about just computing the set directly somehow? Mike and I thought a lot about that. We were only able to show that if one is given an algorithm for minor-closed F-membership in the form of a Turing machine, then there could be no algorithm to find all the relevant obstructions [6]. I'm sure the same is true for the immersion order. But of course that's not very surprising. Precious few things are computable when arguments are reduced to the Turing machine model.

Not long after Theorem 4 became fairly well known within the community, Vijaya Ramachandran kindly pointed us to an intriguing idea generally attributed to Leonid Levin [13]. We were immediately attracted to Levin's strategy, and sought ways to employ it. Soon enough we managed to produce the following

[6] It is important that we consult decision oracles only for non-increasing values of k, and that we do not overly inflate the size of the graph as we modify it during self-reduction. These are known as uniformity and honesty requirements, respectively.

result, which I would probably characterize as illuminating and entertaining, but wildly impractical and unimplementable. Once again this is a simplified version of a more general result from [6].

Theorem 5. [6] *Let F denote a closed family in the minor or immersion order. And let $T_0(n)$ denote the time complexity of any algorithm solving the search version of F-membership. If the following are available*

1. *a solution checking algorithm that runs in $O(T_1(n))$ time,*
2. *order tests that need at most $O(T_2(n))$ time, and*
3. *a self-reduction (its time requirements are irrelevant),*

then $O(max\{T_0(n) + T_1(n) * \log T_0(n), T2(n)\})$ time is sufficient to solve the search version of F-membership.

Under reasonable assumptions about $T_0(n)$, $T_1(n)$ and $T_2(n)$, the algorithm of Theorem 5 is asymptotically optimal — yet we may never know exactly what sort of upper limit optimality provides. This is because we are only ensured a runtime that's bounded above by a multiplicative constant of any satisfying $T_0(n)$. We hugely exploit the fact that each procedure has a constant index in any fixed enumeration of all algorithms. But this gives us no systematic means for learning anything useful about how an algorithm achieving a low order $T_0(n)$ actually works.

I think constructivizations such as these remain a bit unsettling, even to this day. Theorem 4 tells us that we can, usually, know an algorithm. We can write it down. We can trace it as it employs a growing but finite obstruction set. We are assured that it will not loop forever. Nevertheless, no matter how long we watch it, we may never know the entire obstruction set upon which it relies. Similarly, Theorem 5 tells us that we can, in principle, construct an asymptotically fastest algorithm. We know exactly how it works. We can check its results. Yet we will never know its time complexity. What an odd turn of events. All this reminds me again of the fourth type of lie I mentioned earlier. Accordingly, in almost any practical sense, I would not argue against the sentiment that much of what we had done, especially in Theorem 5, was play a bunch of clever but dirty tricks with the big "oh."

And so it went. Every time Mike and I found ways to remove various forms of non-constructivity from one arena, they would quickly seem to pop up in another. It was as if we were playing Whack a Mole at the complexity theory arcade. Despite all this, I think Theorems 4 and 5 (or rather their more general versions) and others that Mike and I devised have been big hits for many years. I am often asked about them. They seem to get at the core of what can and cannot be made constructive using well-quasi order theory. Working on them was a lot of fun for the two of us. They exemplify the sorts of results we obtained on what then were challenging algorithmic paradigms in a completely new research domain.

10 Community Reactions

Early on, I think only a few of our colleagues expressed much interest in what Mike and I were investigating. Fewer still gave us much in the way of encouragement. Notable exceptions include Gene Lawler, Vic Klee, Manny Blum, Steve Mahaney and a handful others (apologies to those whose names I've failed to recall). To them we are eternally grateful. We sorely needed the occasional attaboy, and they came through for us.

A few who shall remain nameless, on the other hand, found our earliest work somewhat blasphemous. It some quarters it took years for us to gain credibility. Depending on whom you asked, we were either the village idiots or the lunatic fringe. The specter of non-constructive algorithms was, I think, viewed as a bit of a threat by some who had staked their careers on the theory of $\mathcal{NP}$-completeness. You see, the usual response to any worry that $\mathcal{P}$ might somehow equal $\mathcal{NP}$ was that all those published and highly cited completeness reductions would still be important. They would simply become useful polynomial-time algorithms. But what if there were a non-constructive proof of, gasp, $\mathcal{P} = \mathcal{NP}$? After all, the standard approaches had long failed to resolve the question. Something radical might be required. In that case, all those swell completeness proofs would be meaningless. You could still map all of $\mathcal{NP}$ to some $\mathcal{NP}$-complete problem Π all right, solvable in polynomial time via Theorem 1. But without constructive knowledge of the polynomial-time algorithm for Π, you're stuck. Of course today, most of us would probably bet that surely $\mathcal{P} \neq \mathcal{NP}$. Probably the same was true back then too, although word on the street was that hopelessly flawed proofs in both directions were running about 50-50. It is fairly remarkable that the question remains open to this day.

Some in our community even dismissed our early work as mere "mathematical curiosities." Perhaps you have heard stories from Mike about the disingenuous "elevator chat" at STOC, or seen the insulting "minor results" article actually produced in print. Although I would prefer not to name any names here, I was there for these and several other early snubs, sometimes poorly disguised as humor. It seemed to us that hubris among the theoretical computer science intelligentsia was never in short supply back then. While all has long been forgiven, it hasn't been forgotten. Perhaps I merely flatter myself, but I would like to think I have a pretty thick skin. So this was to me just part of doing business in computer science. Besides, I'm not a dyed-in-the-wool theorist, nor really very much of a pure theorist at all. But these early affronts by the in-crowd were, I believe, actually quite painful for Mike. They became huge and long-lasting influencing factors on his attitude toward research in general and theoretical computer science in particular. They helped shape his thinking, created in him something of a firebrand for the cause against elitism, and released him from the parochial and highly inbred model of theory as then practiced in the US. Over time, I think they helped steel his resolve to make his work truly world class and now so well respected around the globe. *Per adversas res fortitude!*

11 Shifting Gears

Thus, the early years were absorbed mainly with the positive side of the equation. We focused on concrete problems, speed-ups, self-reductions, constructivity issues and the like. While we brought forward the basic notions of non-uniform complexity and what is now known as fixed-parameter tractability, at that time we could only struggle with a variety of feeble ideas for lower bounds. As time marched on, Mike and I both took positions elsewhere. I stayed in the US, moving to the University of Tennessee and focusing mostly on applications. After stints at the Universities of New Mexico and Idaho, Mike eventually moved outside the US, first to Canada, later to slots in New Zealand and other countries, and finally to Australia, where he now holds a professorship at Charles Darwin University. During all those years and across all those moves, Mike has built up hugely successful collaborations with a coterie of renowned scientists. One of the things that rather quickly came out of all that effort has been the much-heralded negative side of the equation (by that I mean hardness and the $\mathcal{W}$ hierarchy). Other sections of this festschrift will no doubt describe some of those collaborations.

12 Retrospective

When I first met Mike I was working on a wide variety of topics (as I still do today). At that time I was investigating problems in packing and scheduling theory, parallel computation, time-space optimal algorithms and several other areas. Maybe I have trouble committing myself just to one subject. In any case, I think pursuing multiple research interests can make it somewhat easier to find good collaborators, attract funding, and pique the interest of capable students. Plus I find it to be a lot of fun. On the other hand, spreading your time across multiple target areas is not really the best way to build brand recognition for your research program. Instead, it's probably a lot smarter to pick a single focus area and concentrate on being the world's best in that one area. (Young scientists take note!) And this is where Mike has always excelled. I think I worked pretty hard in those early days, but again I had several ongoing research projects. In contrast, Mike worked almost all the time and single-mindedly, with minimal distraction, on what eventually evolved into parameterized complexity. Except for an occasional sideline interest (such as Computer Science Unplugged), he generally still displays that fantastically keen focus today.

So I've watched Mike grow from being the voice of "What the hell is VLSI?" to his now being the voice for a whole new computational discipline. From the day his house self destructed when its pipes froze then re-thawed, to his now being in such international demand that he needs no permanent house at all. Mike and I have worked, played and laughed together for over a quarter century. Friends and colleagues like that don't grow on trees.

Let me therefore say congratulations, Mike, old friend, on the completion of your 60th year on Planet Earth. I wish you continued good health, much happiness, and many many more fruitful and productive years to come!

Acknowledgments and Disclaimers. I would like to take this opportunity to thank Rod Downey and Daniel Marx for their helpful suggestions in improving the presentation of this collection of reflections on Mike's career. In fact I would like to thank all the festschrift organizers for inviting me to prepare this contribution. At their behest, I have done my dead level best to describe the background and genesis of fixed-parameter tractability. Thanks also go to Fran Rosamond. Her enthusiasm, spontaneity and consistent good humor are positively contagious.

The opinions expressed here are mine alone, and should be foisted upon neither Mike nor the organizers of this project. I have tried to be open and forthright, yet politic, calling events honestly as I recall them. I have pulled only a few punches, and those mostly in the uneasy early relationship Mike and I had with the computer science theory establishment. I would also like to thank profusely the many who helped inspire Mike and me along the way, and apologize again to those whose names I've inadvertently failed to mention.

References

1. Chen, J., Kanj, I.A., Xia, G.: Simplicity is beauty: Improved upper bounds for vertex cover. Technical Report TR05-008, DePaul University, Chicago, Illinois (2005)
2. Deo, N., Krishnamoorthy, M.S., Langston, M.A.: Exact and approximate solutions for the gate matrix layout problem. IEEE Transactions on Computer Aided Design 6, 79–84 (1987)
3. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer (1999)
4. Fellows, M.R., Langston, M.A.: Nonconstructive advances in polynomial-time complexity. Information Processing Letters 26, 157–162 (1987)
5. Fellows, M.R., Langston, M.A.: Nonconstructive tools for proving polynomial-time decidability. Journal of the ACM 35, 727–739 (1988)
6. Fellows, M.R., Langston, M.A.: On search, decision and the efficiency of polynomial-time algorithms. In: Proceedings of ACM Symposium on Theory of Computing, pp. 501–512 (1989)
7. Fellows, M.R., Langston, M.A.: On well-partial-order theory and its application to combinatorial problems of VLSI design. SIAM Journal on Discrete Mathematics 5, 117–126 (1992)
8. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness. W.H. Freeman and Company, New York (1979)
9. Kashiwabara, T., Fujisawa, T.: An $\mathcal{NP}$-complete problem on interval graphs. In: Proceedings of IEEE Symposium on Circuits and Systems, pp. 657–660 (1979)
10. Kinnersley, N.G.: The vertex separation of a graph equals its path-width. Information Processing Letters 42, 345–350 (1992)
11. Kinnersley, N.G., Langston, M.A.: Obstruction set isolation for the gate matrix layout problem. Discrete Applied Mathematics 54, 169–213 (1994)
12. Kuratowski, K.: Sur le problème des courbes gaushes en topologie. Fundamenta Mathematicae (French) 15, 271–283 (1930)

13. Levin, L.A.: Universal enumeration problems. Problemic Peredaci Informacii (Russian) 3, 115–116 (1972)
14. Lopez, A.D., Law, H.-F.S.: A dense gate matrix layout method for MOS VLSI. IEEE Transactions on Electron Devices 27, 1671–1675 (1980)
15. Robertson, N., Seymour, P.D.: Disjoint paths - a survey. Journal of Algebraic and Discrete Methods 6, 300–305 (1985)
16. Robertson, N., Seymour, P.D.: Graph minors IV. Tree-width and well-quasi-ordering. Journal of Combinatorial Theory, Series B 48, 227–254 (1990)
17. Robertson, N., Seymour, P.D.: Graph minors XIII. The disjoint paths problem. Journal of Combinatorial Theory, Series B 63, 65–110 (1995)
18. Robertson, N., Seymour, P.D.: Graph minors XX. Wagner's conjecture. Journal of Combinatorial Theory, Series B 92, 325–357 (2004)