Why is $\mathcal{P}$ Not Equal to $\mathcal{NP}$? *

Michael Fellows[1,2] and Frances Rosamond[1]

[1] University of Newcastle, Callaghan NSW 2308, Australia
{michael.fellows,frances.rosamond}@newcastle.edu.au
[2] Durham University, Institute of Advanced Study,
Durham DH1 3RL, United Kingdom

Abstract. The question of whether the complexity classes $\mathcal{P}$ and $\mathcal{NP}$ are equal remains one of the most important open problems in the foundations of mathematics and computer science. We discuss two conjectures that seem to be reasonably accessible to intuition, and that imply $\mathcal{P} \neq \mathcal{NP}$. One of these is a novel conjecture about the existence of *intrinsically hard puzzles*. We propose a way to explore this conjecture empirically in a way that may have a number of scientific payoffs.

1 Introduction

This paper is primarily concerned with issues of mathematical aesthetics and speculation. Some well-known researchers in computational complexity have predicted that the "$\mathcal{P}$ *versus* $\mathcal{NP}$" question may well be with us for some time. Ogihara, for example, has suggested that it might not be resolved before the year 3000, so difficult and lacking of programmatic access does the question appear [Hem02]. Aaronson has argued that we may never have a proof one way or the other [Aar06b].

The question is so fundamental, of such practical importance, and philosophically so accessible and compelling, that even in the absence of mathematical progress, there is still interest (and usefulness, we would argue) in the philosophical appraisal of the problem, and in informed opinions about how it might turn out, what techniques might be relevant, and how difficult it might be. The (notably diverse) 2002 "poll" of expert opinion on these questions by Bill Gasarch [Gas02] can be taken as one primary reference point for discussions of this sort. The Gasarch survey has been widely noted, and used in educational settings to stimulate thinking (e.g., courses on algorithms and computational complexity).

The issue has also proved irresistable in the theoretical computer science blogosphere. Discussions that can also be taken as reference points are the blogs

* This research has been supported by the Australian Research Council through the Australian Centre for Bioinformatics, by the University of Newcastle Parameterized Complexity Research Unit under the auspices of the Deputy Vice-Chancellor for Research, and by a Fellowship to the Durham University Institute for Advanced Studies. The authors also gratefully acknowledge the support and kind hospitality provided by a William Best Fellowship at Grey College while the paper was in preparation.

by Aaronson [Aar06a], and the videoblog rejoinder by Fortnow and Gasarch [FG06]. Aaronson has given a thoughtful list of ten reasons for believing $\mathcal{P} \neq \mathcal{NP}$ that provides some useful initial "order" to the area [Aar06a]. What we do here is simply *add another opinion*, not already represented in the Gasarch catalog, and (hopefully) contribute some novel points of discussion.

We discuss the "$\mathcal{P}$ *versus* $\mathcal{NP}$" question in relation to two other conjectures:

- The conjecture that FPT is not equal to $W[1]$.
- A conjecture that we introduce here and term the *Hard Puzzles Conjecture*.

In Gasarch's survey, 61 complexity theory experts testified in favor of $\mathcal{P} \neq \mathcal{NP}$, 9 surmised that $\mathcal{P} = \mathcal{NP}$, 4 gave the opinion that it is independent (e.g., of ZFC) and 22 offered no opinion.

It can happen in mathematics that statement S_1 implies statement S_2, while S_1 may seem "more intuitive" than S_2.

In considering the "$\mathcal{P}$ *versus* $\mathcal{NP}$" question, it is natural to ask if there are other statements that might be *even more accessible* to mathematical intuition, that might help in forming an opinion about it. In particular, are there candidate statements that might be at least as defensible, in the sense of the kind of intuitive argumentation offered in [Aar06a], as the $\mathcal{P} \neq \mathcal{NP}$ conjecture?

One of the remarkable characteristics of the $\mathcal{P}$ *versus* $\mathcal{NP}$ question is that some of the arguments in favor of $\mathcal{P} \neq \mathcal{NP}$ can be explained to *anybody* (in particular, Reason 9, "The Philosophical Argument" of the Aaronson list). If such questions about computational complexity are going to be with us a long time, then it is reasonable to ask if there might be other such statements that are *natural resting points* for mathematical intuition.

We are with the majority: we believe $\mathcal{P} \neq \mathcal{NP}$, for the following reasons.

2 Because FPT is not Equal to $W[1]$

There are three natural forms of the Halting Problem, all three of which play central roles in metamathematics and contemporary theoretical computer science. These are:

The Halting Problem (I)
Instance: A description of a Turing machine $\mathcal{M}$.
Question: On an empty input tape, will $\mathcal{M}$ halt?

The Polynomial-Time Halting Problem for Nondeterministic Turing Machines (II)
Instance: A description of a nondeterministic Turing machine $\mathcal{M}$.
Question: On an empty input tape, can $\mathcal{M}$ halt in at most $|\mathcal{M}|$ steps?

This problem can also be defined with a halting time of $|\mathcal{M}|^c$ for any fixed constant c, without effect on the discussion.

THE k-STEP HALTING PROBLEM FOR NONDETERMINISTIC TURING MACHINES (III)
Instance: A description of a nondeterministic Turing machine $\mathcal{M}$ where the alphabet size is unlimited and the amount of nondeterminism (the number of transitions that might be possible at any computation step) is unlimited.
Parameter: k
Question: On an empty input tape, can $\mathcal{M}$ halt in at most k steps?

It is quite natural to focus on this "natural descent of the HALTING PROBLEMS" for a number of reasons:

1. HALTING PROBLEM I is fundamental to what we can prove about unsolvable problems. It is also worth noting that Gödel's Incompleteness Theorem (in a very natural form — the form concerned with proving things about computation) follows as a one paragraph corollary.
2. HALTING PROBLEM II is trivially complete for NP, basically a re-presentation of the definition of NP.
3. HALTING PROBLEM III is complete for $W[1]$. The conjecture that FPT $\neq W[1]$ plays a role in parameterized complexity analogous to the $\mathcal{P} \neq \mathcal{NP}$ conjecture in classical complexity.

We argue that the FPT $\neq W[1]$ conjecture is at least as defensible, in the sense of Aaronson, as the $\mathcal{P} \neq \mathcal{NP}$ conjecture. We argue that it displays the essential issues in a way that is easier for mathematical intuition to approach. This opinion might be disputed, but if there is to be any discussion of why $\mathcal{P} \neq \mathcal{NP}$ is true, then a claim that there is a more defensible conjecture which implies it is at least worth examining.

Because the subject of parameterized complexity theory is not so well known, some background discussion is in order.

2.1 Some Background on Parameterized Complexity

Some concrete results of the theory may serve to set the stage for readers unfamiliar with the subject:

- The primordial combinatorial problem of VERTEX COVER seeks to determine if it is possible to choose a set $V' \subseteq V$ of k nodes in a conflict graph $G = (V, E)$ on $|V| = n$ nodes, so that every edge of the graph is incident on at least one vertex of V' (all the conflict edges are "covered" in the sense that if the vertices of V' are deleted, then there are no more conflicts between the "data points" represented by $V - V'$). Although NP-complete, after several rounds of increasingly sophisticated algorithmic improvements developed by many researchers in the context of parameterized complexity, this can now be solved in time $O(1.274^k n^2)$ [CKX05]. This is a result of an increasingly sharp *positive* toolkit of FPT design techniques. As one can easily imagine, conflict graphs are used to model issues in a vast range of computing applications.

– There is a natural hierarchy of parameterized complexity classes

$$FPT \subseteq M[1] \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[P] \subseteq XP$$

with respect to which hundreds of diverse combinatorial problems have been classified, revealing the surprising fact that there are only four or five natural *degrees* (under parameterized reducibility) that capture "almost all" natural parameterized problems. This empirical fact is analogous to the surprising ubiquity of the NP-complete degree in classical (one-dimensional) computational complexity. It is now known that $FPT = M[1]$ if and only if n-variable 3SAT can be solved in time $O(2^{o(n)})$ [DEFPR03]. The conjecture that FPT $\neq M[1]$ is equivalent to the Exponential Time Hypothesis introduced by Inpaggliazzo et al. [IPZ01] (see also the survey by Woeginger [Woe03]). A fundamental result, part of a parameterized analog of Cook's Theorem, is that the k-Step Halting Problem for Turing machines of unrestricted nondeterminism (trivially solvable in time $n^{O(k)}$) cannot be in FPT (that is, solvable in time $O(f(k)n^c)$) unless $FPT = W[1]$, i.e., this natural 2-dimensional form of the Halting Problem is complete for $W[1]$ [DF99]. Determining whether a graph has an independent set of size k is complete for $W[1]$ [DF99]. Determining whether a graph has a dominating set of size k is complete for $W[2]$ [DF99]. Obviously, both of these problems can be solved by the brute force approach of trying all k-subsets in time $n^{O(k)}$.
– The *negative toolkit* of parameterized complexity is producing deeper and deeper information. Recent results show that k-Dominating Set cannot be solved in time $n^{o(k)}$ unless $FPT = M[2]$, and k-Independent Set cannot be solved in time $n^{o(k)}$ unless $FPT = M[1]$ [CCFHJ+04]. We also now know that there exists a constant $\epsilon_{VC} > 0$ such that the Vertex Cover problem cannot be solved in time $O((1 + \epsilon_{VC})^k n^c$ unless $FPT = M[1]$, hence the striking improvements given by the positive toolkit for this problem must eventually end, although we currently have no concrete bounds on the barrier constant ϵ_{VC}. The positive algorithmic improvements for Vertex Cover have led directly and strikingly to improved practical algorithms for this NP-hard problem.
– As another example of the way that parameterized complexity interacts with practical computing, there is a natural problem in the context of the implementation of programming language compilers known as Type Checking (checking the consistency of the type declarations for variables). For the logic-based language ML this was shown to be hard for EXP (a superset of NP). However, the implementors of compilers for ML noted that this result seemed to be irrelevant, "The compiler works just fine." In the context of parameterized complexity, an explanation has emerged. The natural input distribution for the problem (ML programs written by human beings), tends to have small nesting depth of type declarations. That is, there is natural parameter k for this problem, measuring the maximum nesting depth of the type declarations, for which in practice almost always $k \leq 5$. The ML implementations naively employed a type-checking algorithm that runs in time

$O(2^k n)$, where n is the size of the program — thus an FPT algorithm, although not deliberately developed or known in these terms by the computing practitioners who implemented the compiler. There are many such examples where the concepts of parameterized complexity clarify and mathematically deepen naive approaches that have been adopted by programmers facing hard problems.

Further background can be found in the survey of Downey [Dow03] and the recent monographs of Flum and Grohe [FG06] and Niedermeier Nie06.

2.2 Why the FPT $\neq W[1]$ Conjecture Seems More Accessible to Intuition

Aaronson's Reason 9, "The Philosopical Argument" might be of genuine interest to philosophers of science. This is the argument that one generally uses to explain the issue to a nonmathematician. Paraphrasing Aaronson, if $\mathcal{P} = \mathcal{NP}$, then the world would be a profoundly different place than we usually assume and experience it to be. "There would be no special value in creative leaps, no fundamental gap between solving a problem and recognizing a solution." Wouldn't life have evolved to take advantage of such an equality, and the biological world be very different? The Philosophical Argument interprets the mathematical conjecture physically and metaphorically, and under this translation is quite accessible to anyone's intuition.

The heart of our argument that the FPT $\neq W[1]$ Conjecture is more directly accessible to intuition is that "The Philosophical Argument" seems to apply with even greater force. Most creative breakthroughs involve five key steps, or ten, not "a polynomial number". Halting Problems I and II can be directly interpreted in terms of searching unstructured mazes. Trying to find a way out that involves a small or moderate fixed number of steps seems more directly comparable with the way the world, and mazes, are experienced. At any given moment, having *many* different possible choices also fits well with the way the world is experienced. The conjecture inhabits a more "finitistic" conceptual space, closer to the one that we do in fact physically inhabit, as discussed in the concluding section of Aaronson's essay on the possible independence of the $\mathcal{P}$ *versus* $\mathcal{NP}$ question [Aar06b]. How do you solve the 10 step Halting Problem in $O(n^9)$?

Aaronson's list of ten reasons for believing $\mathcal{P} \neq \mathcal{NP}$ includes a number of other reasons that reflect accumulated mathematical experience, as well as structural complexity results.

Reason 1, The Obvious Argument, is that people have tried for a long time to find polynomial-time algorithms for canonical NP-complete problems such as Circuit Satisfiability and have consistently failed. The $W[t]$ classes are defined in terms of weight k circuit satisfiability, a quite natural restriction that has also received attention, although perhaps somewhat less. Here too, no algorithm essentially better than brute force has been found.

Reason 2, The Empirical Argument, is about the evidence provided by half a century of evidence from the efforts of practical algorithm designers in the computer industry. This same history, viewed differently, also offers evidence of great

efforts spent to devise FPT algorithms (before there was a name for this notion). Practitioners have long paid attention to parameters and tried to exploit them where possible. One of the successes of the parameterized complexity framework is the way it has succeeded in capturing more sensitively the way that practical computing has actually been conducted all along.

Some of the other Reasons that concern mathematical evidences, such as the surprising collapses that would occur if $\mathcal{P} = \mathcal{N}P$, are here less developed (which should make this an attractive area for investigation) but some striking things are known. The tower of parameterized intractability classes begins with:

$$FPT \subseteq M[1] \subseteq W[1] \subseteq M[2] \subseteq W[2] \cdots$$

and it is known that FPT is equal to $M[1]$ if and only the *Exponential Time Hypothesis* (ETH) fails [DEFPR03]. Thus FPT = $W[1]$ would imply that n-variable 3SAT can be solved in $2^{o(n)}$ time. Some known lower bounds on circuit complexity carry over to provide indirect evidence that the $W[t]$ hierarchy is proper [DF99].

Incidentally, but relevantly, we do not feel much intuitive connection to the ETH — this seems difficult (to us) to form much direct intuition about. How would a Gasarch-style survey of the ETH turn out? [3]

3 Because There Are Hard Puzzles

We first describe some concrete background experiences that have led to the *Hard Puzzles Conjecture* (HPC). We then show that the HPC implies $\mathcal{P} \neq \mathcal{N}P$. Finally, we propose a way in which the HPC can be concretely explored, and some possible payoffs of such an empirical investigation.

3.1 Background

For many years the authors have been active in popularizing topics in theoretical computer science to elementary school children, using concrete activities such as described in the books *This Is MEGA-Mathematics!* [CF92] and *Computer Science Unplugged* [BWF96], and in the paper by Fellows and Koblitz about presenting modern cryptography to children [FK93]. The "$\mathcal{P}$ *versus* $\mathcal{N}P$" question can be made concretely accessible to children in a number of ways.

For example, to get the issue across concretely, one can first pass out a 2-colorable graph (without telling the children that it is 2-colorable), and explain the basic rule for a proper coloring: that vertices joined by an edge must get different colors, and the basic objective: finding a coloring with a minimum number of colors. Almost always, the solutions will improve for a few rounds, before someone discovers that a 2-color solution is possible (basically re-discovering the 2-coloring algorithm).

[3] Experts in parameterized complexity are divided on whether perhaps $M[1] = W[1]$. This may be accessible to "holographic" proof techniques (if it is true).

In the second part of the presentation, one can pass out a 3-colorable graph, such as the one shown in Figure 1(b). This particular instance has proved itself in dozens of classroom visits as usually defying everyone's efforts for most of an hour. Figure 1(a) shows a similarly "proven useful" small hard instance of Minimum Dominating Set.

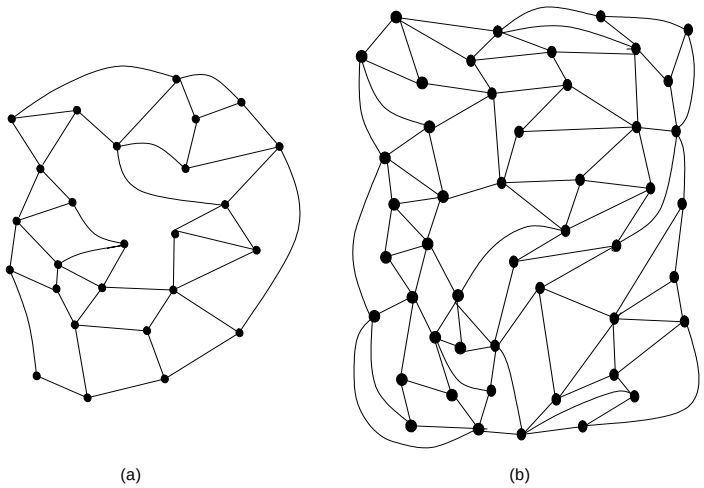

Fig. 1. Some small hard instances of NP-hard problems used in classrooms: (a) Minimum Dominating Set, (b) 3Coloring

Based on these experiences we wish to raise a few issues about hard combinatorial problems and algorithm design that may have bearing on the $\mathcal{P}$ *versus* $\mathcal{NP}$ question. It seems to us remarkably easy to design relatively small hard instances for most NP-hard problems, and one can't help sensing a kind of amorphous opacity beginning to assert itself even in these small instances: there just isn't much structure to hang on to, and one doesn't get much "inkling" of algorithmic possibility. It seems to us that for most of the problems for which we have P-time algorithms, one can point to such "inklings" and "partial algorithms" and "possible strategies" that begin to appear if you spend some time solving some concrete instances. *Amorphous opacity* is the quality of an unstructured maze. This is essentially Argument 6, The Known-Algorithms Argument, of Aaronson, but here experienced afresh in small hard instances in an elementary school classroom, using our heads as computers.

In this context, the following seemingly natural conjecture concerning the existence of such hard puzzles has suggested itself.

The Hard Puzzles Conjecture (for Graph 3-Coloring).
There exists a sequence of increasingly larger 3-colorable graphs $G_1, G_2, G_3, \ldots G_n$,

$|G_i| = i$, such that $\forall c$, $\exists n_c$ such that $\forall n \geq n_c$, and $\forall$ algorithm ϕ for which:
(1) $|\phi| \leq c$ (c is thus a bound on the program size), and
(2) ϕ computes a 3-coloring for G_n (the single "puzzle" graph G_n; there are no requirements on ϕ being able to solve any other puzzle)
the program ϕ will necessarily require $\Omega(n^c)$ time to compute the 3-coloring.

We believe that, over time, this may come to be seen as a very natural conjecture (with versions, of course, for any NP-hard problem). [4]

3.2 The Hard Puzzles Conjecture Implies $\mathcal{P} \neq \mathcal{NP}$

The proof is immediate: if $\mathcal{P} \neq \mathcal{NP}$ then there exists a single algorithm ϕ and two constants c_0 and c_1 such that:
(1) $|\phi| \leq c_0$, and
(2) for every 3-colorable graph G of size n, ϕ takes time at most $O(n^{c_1})$.
In the conjecture, c plays both of these roles: placing a bound on the size of the algorithm, and giving us an exponent for a target polynomial running time.

The conjecture essentially says that there exists a series of 3-coloring puzzles that are *so intrinsically hard*, that no matter how generous you are about the size c allowed for the algorithm, and for the running time of the exponent — if you go far enough out in the list of puzzles, and even if you are only looking for algorithms that solve a single puzzle, then you will be disappointed: any algorithm that is small enough will take too much time. (The reason for the restriction on the size of the algorithm should be clear, since without this restriction, for an algorithm designed to solve a single puzzle, there would always be the trivial linear time algorithm that simply prints the solution.)

Any proof of the HPC would seem to have to be *extremely nonconstructive*, and in any case, cannot be expected anytime soon.

We remark that the HPC seems to be a different, albeit, related notion to *instance complexity*, as defined by several authors [OKSW94,Mu00].

4 Engaging the Hard Puzzles Conjecture: A Proposal for a Stock Market of Hard Instances

If we could somehow find the hard puzzles promised by the conjecture, this could be useful and interesting.

There seem to be several routes to "where the hard instances are". We mention two that have previously been investigated:

1. **Random Instances from the Phase Transition.** Cheeseman, et al. [CKT91] introduced the notion of *phase transitions* in the distribution of hard instances to NP-hard problems. One of their key examples was Graph Coloring where they discuss polynomial-time reduction rules to simplify an instance.

[4] It might also be seen to be in harmony with the possible terrible existential destiny of mathematics sketched in Aaronson's essay [Aar06b], where we may be confronted, perhaps forever, with key puzzles of mathematics that defy all efforts.

2. **FPT polynomial-time kernelization.** This is discussed in the monograph by Niedermeier [Nie06]. The hard instances are the kernelized instances. Whether a strong set of kernelization rules will *take you* to the phase transition boundary is an interesting unresolved issue. For example, if (G, k) is an instance of the VERTEX COVER problem, then in polynomial time we can reduce to an equivalent instance (G', k') where $k' \leq k$ and where the number of vertices in G' is bounded by $2k'$.

Thought Experiment. We propose to employ "the minds of children" in an implicit genetic algorithm. Suppose there were a **Hard Instance Stock Market** where (small?) investors could contribute a small, presumably hard, instance of an NP-hard problem for a small fee, such as \$1. The STOCK MARKET would evaluate the invested instances by gathering statistics on competitive puzzle-solving. Perhaps each instance in the top 10% is awarded a \$60 prize, and every instance in the top 0.01% is awarded \$1000. Investing in the market might allow one to freely download puzzle instances from the stock market to a cellphone. This business model would seem to combine two of mankind's greatest addictions: puzzling, and something like gambling.

Inevitably, if such a venture succeeded (and besides serving to popularize theoretical computer science on a grand scale), some developments can be predicted. If there were big money in it, then adults with computers would get involved.

Who would win, in designing the best small hard instances? This is somewhat akin to the problems confronted by, e.g., chess-playing programs, and it is not at all clear that adults with computers would have an advantage over the minds of children or other natural players. This seems to be an interesting scientific question about human cognitive abilities in relation to the power of computers that we propose to call the *reverse Turing test*.

A collection of small, somehow certifiably, intrinsically hard instances would also seem to be useful for the testing of heuristics for hard problems.

Perhaps, evidence that the HPC is true would gradually emerge.

5 Discussion and Open Problems

In considering a discussion such as offered in this paper, it is necessary to confront an issue of traditional mathematical culture head on. There is a point of view held by some that any discussion of "mathematical intuition" is oxymoronic. Of course, this is an old discussion, taken up on the other side by, e.g., Lakatos [Lak76]. There are also "softer forms" of the opinion discouraging discussion, for example, the one contributed by Fortnow to the Gasarch collection, where essentially it is argued that since we currently have so little programmatic traction towards settling the question, that there is not enough relevant mathematical material to support a credible discussion of, say, *serious intuition*. (It is notable, and we think reflects this traditional mathematical taboo, that some of the responses to the Gasarch survey seem a bit flippant.)

In the spirit of Aaronson's thoughtful and serious attempt to catalog the reasons to believe $\mathcal{P} \neq \mathcal{NP}$, we offer here an initial catalog of:

5.1 Four Reasons Why The Discussion is Interesting, Useful and Worthy

Reason 1. Inevitability and Responsibility. Like or not it, the discussion is already well underway, and will inevitably continue for the forseeable future. *Others* will certainly be engaging in such discussion and speculation, because of the absolutely compelling nature of the question, and also its relative intuitive accessibility. The mathematical community might as well contribute what it can from an expert perspective, out of general responsibility to intellectual life.

Reason 2. Stimulation. The discussion is fun and irresistable (even to those who initially dismiss it, following the traditional mathematical taboos). It may serve to stimulate creative thinking about the problem, or energize and boost the morale of those spending time on it (as advocated by Ron Fagin in the Gasarch opinion catalog). Given the relative failure of mathematical approaches to date, speculative thinking on the question (and related issues) might help to stir up some fresh mathematical approaches.

Reason 3. Creative and Responsible Engineering. The fact is, the $\mathcal{P} \neq \mathcal{NP}$ conjecture is already "operationally true", and it is far from the only mathematical conjecture that now has this status. The point here is that there is a distinction that should be made between the *high church* role of computational complexity theory, and what might be called its *pastoral role* in the world of algorithm design, where theoretical computer science attempts to minister to the needs of industry and the other sciences. (This might be considered as possibly its *central mission*, by analogy with theoretical physics and other theoretical sciences that have an explanatory or predictive or engineering-support mission.) At the pastoral level, the $\mathcal{P} \neq \mathcal{NP}$ conjecture is already "true enough for government work", in the sense that on a day-to-day basis it functions (quite reasonably) to direct the efforts of algorithm designers. *Nobody* at the pastoral level gets paid to continue working to find a polynomial time constant factor approximation algorithm, after somebody else has shown that this is impossible unless $\mathcal{P} = \mathcal{NP}$. If you explained to the famous "boss" in the Garey and Johnson cartoons, that based on your high church belief that $\mathcal{P} = \mathcal{NP}$, you wanted to continue the lead role in this project, you'd be fired.

The *responsibility* part here has to do with a more general situation that has developed: the entire enterprise of mathematical sciences and engineering has become immensely "forward leaning". For example, the world economy now runs on the conjecture that Prime Factorization is difficult. And not just that. The companies that develop and sell elliptic curve cryptosystems have to gather investor money, and convince customers that the system is secure. Essentially they have to explain why investing in conjectures about elliptic curves is reasonable. Most investors would not have a problem investing in $\mathcal{P} \neq \mathcal{NP}$.

The *creativity* part here has to do with another distinction, between conjectures that have *operational value*, and those that do not. The Exponential Time

Hypothesis and the conjecture that FPT is not equal to $W[1]$ have day-to-day uses at the pastoral level. It is *useful* to identify plausible conjectures that can help routinely to establish lower bounds on problem kernel sizes for parameterized problems that are FPT (see the discussion in Burrage, et al. [BE+06], also [Fe06]), and we currently do not have such conjectures identified. Similar issues are confronted in the recent work of Harnik and Naor [HN07].

Reason 4. It is Good for the Field. Computational complexity theory might offer more job opportunities in the long run, if it conceived of itself as concerned with the identification of statements that are "probably true" and that have operational value, whether provable or not, and doing what it can to gather some evidence to support such conjectural frameworks that are useful at the pastoral level. Since most of the candidate statements imply $\mathcal{P} \neq \mathcal{NP}$, the field is not going to find these gems by waiting for a mathematical resolution of the $\mathcal{P}$ *versus* $\mathcal{NP}$ question, before finding time and inclination.

It is surprising that there is not a single mention of the $FPT \neq W[1]$ conjecture in such comprehensive surveys of the field as, for example, Fortnow and Homer [FH02], even though the pastoral role of the conjecture is so strong. The field of computational complexity theory seems to have centered itself on the $\mathcal{P}$ *versus* $\mathcal{NP}$ (and related) questions in a manner that one might expect for Recursion Theory II, but not necessarily for a theoretical science accountable to functional criticism in the manner of theoretical physics (which is held accountable to explanation and prediction). No other theoretical discipline has so tied itself to a central question that:
(1) It apparently cannot answer, in the terms it has set for finding an answer.
(2) The answer, were it proven, would surprise nobody.
(3) The outcome would have absolutely no impact on practice.

It would be unfortunate if the state of the field were to become the state of the question.

5.2 Two "New" Reasons for Believing $\mathcal{P} \neq \mathcal{NP}$

We have offered here two reasons for believing $\mathcal{P} \neq \mathcal{NP}$, the first being the conjecture that FPT is different from $W[1]$. We believe that this conjecture, while closely related and less well-known, is actually easier to think about and admits a defense at least as strong as $\mathcal{P} \neq \mathcal{NP}$, despite that it is mathematically a "stronger" conjecture.

The *Hard Puzzles Conjecture* (HPC) seems to us to arise quite naturally, and we have described its experiential origins in popularizing to children the $\mathcal{P}$ *versus* $\mathcal{NP}$ question. The HPC implies $\mathcal{P} \neq \mathcal{NP}$, but it also seems compelling on intuitive grounds (at least to us). Basically the argument comes down to an "Extended Philosophical Argument". Routinely, we experience situations where small hard puzzles *defy everybody* (who play the role of the many possible algorithms).

More importantly, we have proposed a possible experiment towards exploring the intellectual terrain in the direction of the HPC that seems to be of independent scientific interest.

Final Note. The small hard instances depicted in Figure 1, do have some structure that can be exploited (they are really not good representatives of *completely* amorphous opacity): they are *planar*. It is known that both 3Coloring and Minimum Dominating Set can be solved in time $O(n^{O(\sqrt{k})})$ for planar graphs, and cannot be solved in time $O(n^{o(\sqrt{k})})$ unless FPT = $M[1]$ (equivalently, ETH fails).

References

[Aar06a] S. Aaronson. Reasons to Believe. *Shtetl-Optimized: The Blog of Scott Aaronson* (http://www.scottaaronson.com/blog/), posted 4 September 2006.

[Aar06b] S. Aaronson, Is P versus NP formally independent? Manuscript, 2006.

[BE+06] K. Burrage, V. Estivill-Castro, M. Fellows, M. Langston, S. Mac and F. Rosamond. The undirected feedback vertex set problem has polynomial kernel size. *Proceedings IWPEC 2006*, Springer-Verlag, *Lecture Notes in Computer Science* 4169 (2006), 192–202.

[BWF96] T. Bell, I. H. Witten and M. Fellows. *Computer Science Unplugged: Offline activities and games for all ages*, 231 pp., 1996. (Web available).

[CF92] N. Casey and M. Fellows. *This is MEGA-Mathematics!*, 1992, (134 pp.), available for free from: http://www.c3.lanl.gov/ captors/mega-math.

[CCFHJ+04] J. Chen, B. Chor, M. Fellows, X. Huang, D. Juedes, I. Kanj and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. *Proceedings of the IEEE Conference on Computational Complexity* (2004), 150–160.

[CKT91] P. Cheeseman, R. Kanefsky, and W. Taylor. Where the Really Hard Problems Are. *Proceedings IJCAI-91* (1991), 163–169.

[CKX05] J. Chen, I. Kanj and G. Xia. Simplicity is beauty: improved upper bounds for vertex cover. Manuscript, 2005.

[CM97] S. A. Cook and D. G. Mitchell. Finding Hard Instances of the Satisfiability Problem: A Survey. in D. Du, J. Gu, and P. Pardalos, Eds. *The Satisfiability Problem. DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 35 (1997), 1–17.

[DEFPR03] R. Downey, V. Estivill-Castro, M. Fellows, E. Prieto-Rodriguez and F. Rosamond. Cutting up is hard to do: the complexity of k-cut and related problems. *Electronic Notes in Theoretical Computer Science* 78 (2003), 205–218.

[DF99] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[Dow03] R. G. Downey. Parameterized complexity for the skeptic. *Proc. 18th IEEE Annual Conf. on Computational Complexity* (2003), 147–169.

[Fe06] M. Fellows. The lost continent of polynomial time. *Proceedings IWPEC 2006*, Springer-Verlag, *Lecture Notes in Computer Science* 4169 (2006), 276–277.

[FG06] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Springer-Verlag, 2006.

[FK93] M. Fellows and N. Koblitz. Kid Krypto. *Proceedings of Crypto '92*, Springer-Verlag, Lecture Notes in Computer Science Vol. 740 (1993), 371–389.

[FG06] L. Fortnow and W. Gasarch. Number 60: a videoblog response, in "61 Responses to *Reasons to Believe*." *Shtetl-Optimized: The Blog of Scott Aaronson* (http://www.scottaaronson.com/blog/) posted 21 December 2006.

[FH02] L. Fortnow and S. Homer. A Short History of Computational Complexity. (http://www.neci.jn.nec.com/homepages/fortnow), 2002.

[Gas02] W. I. Gasarch. Guest column: The P=?NP poll. *Sigact News 36* (www.cs.umd.edu/users/gasarch/papers/poll.ps), 2002.

[HN07] D. Harnik and M. Naor. On the compressibility of NP instances and cryptographic applications. Manuscript, 2006.

[Hem02] L. A. Hemaspaandra. Sigact news complexity theory column 36. *Sigact News*, 2002.

[IPZ01] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? Journal of Computer System and Science, 63(4):512–530, 2001.

[Lak76] I. Lakatos. *Proofs and Refutations.* Cambridge University Press (New York), 1976.

[Mu00] M. Mundhenk. On hard instances. *Theoretical Computer Science*, 242 (2000) 301–311.

[Nie06] R. Niedermeier. *Invitation to Fixed Parameter Algorithms.* Oxford University Press, forthcoming, 2005.

[OKSW94] P. Orponen, K. Ko, U. Schöning, and O. Watanabe. Instance complexity. *Journal of the Association for Computing Machinery* Vol. 41, No. 1 (1994), 96-121.

[Woe03] G. J. Woeginger. Exact algorithms for NP-hard problems: a survey. *Proceedings of 5th International Workshop on Combinatorial Optimization-Eureka, You Shrink! Papers dedicated to Jack Edmonds,* M. Junger, G. Reinelt, and G. Rinaldi (Festschrift Eds.) Springer-Verlag, LNCS (2003).