TFNP: An Update

Paul W. Goldberg[1] and Christos H. Papadimitriou[2]

[1] University of Oxford, Oxford, UK
paul.goldberg@cs.ox.ac.uk
[2] University of California at Berkeley, USA
christos@cs.berkeley.edu

Abstract. The class TFNP was introduced a quarter of a century ago to capture problems in NP that have a witness for all inputs. A decade ago, this line of research culminated in the proof that the NASH equilibrium problem is complete for the subclass PPAD. Here we review some interesting developments since.

1 Introduction

Many apparently intractable problems in NP are *total*, that is, they are guaranteed to have a solution for all inputs. FACTORING (given a non-prime integer, find a prime factor) is perhaps the most accessible example of such a problem, and was the first to be identified, but by now many natural problems of this sort are known. (The hardness of FACTORING stands in contrast with primality testing, which is well known to be polynomial-time solvable [1].) This computational phenomenon is captured by the class TFNP (the initials stand for *total functions in NP*) [20, 21].

How does one provide evidence of intractability for such a problem? Since problems in TFNP are unlikely to be NP-complete, and TFNP appears to have no complete problems ([23], Section 6 constructs an oracle relative to which there is no single TFNP problem to which all others reduce), focus quickly shifted to subclasses of TFNP with complete problems. To show that a problem is in TFNP, one must establish a theorem of the form $\forall x \exists y \Phi(x, y)$ stating that in all situations x of some kind (for example, in every bimatrix game) corresponding to the problem's input, a certain pattern y can be found (a solution, in this example a Nash equilibrium). If the proof of this theorem is constructive in a computationally meaningful sense, then the problem is in P. Hence, all intractable problems in TFNP must harbor an exponentially non-constructive step in their proof, presumably a combinatorial lemma guaranteeing the existence of a particular kind of element in an exponential structure. A productive classification of the problems in TFNP is in terms of the particular combinatorial lemma of the form $\forall x \exists y \Phi(x, y)$ employed in their proof. The following subclasses of TFNP had been known since the early 1990s.

1. PPP, for polynomial pigeonhole principle, the combinatorial lemma stating that *"for every function* $f : \{0,1\}^n \mapsto \{0,1\}^n$ *there must be either*

an $x \in \{0,1\}^n$ *such that* $f(x) = 0^n$*, or* $x, y \in \{0,1\}^n, x \neq y$ *such that* $f(x) = f(y)$*.".*

2. PPA, for polynomial parity argument: *"In every finite graph containing an odd-degree node, a second such node must exist."*

3. PPAD, for polynomial parity argument for directed graphs: *"In every finite directed graph containing an unbalanced (in-degree ≠ out-degree) node, a second such node must exist."* It is not hard to see that PPAD is a subset of PPA, and also of PPP.

4. PPADS. A variant of PPAD, based on a slightly stronger lemma stating that *"a second* oppositely unbalanced *node must exist."* PPADS includes PPAD.

5. PLS, for polynomial local search: *"Every dag has a sink."*

It is often quite nontrivial to capture these classes in terms of a concrete "basic complete problem," but in all these cases it can be done [17, 22]; typically, the class is then defined as all search problems reducible to the basic complete problem. In the case of PPAD, the basic complete problem is called END OF THE LINE, seeking a second degree-one node in a directed graph where no node has in-degree or out-degree greater than one, and 0^n has in-degree zero and out-degree one.

But are these classes different from P and one another? In [3] and elsewhere evidence was provided, through oracle constructions, that essentially all these classes are "compellingly different" from P and from each other, and that no easy inclusions seem to hold beyond the ones noted above.

Arguably, the whole TFNP research direction was motivated by one main quest, establishing that the NASH problem mentioned above is intractable; this was resolved in 2006 [7, 8], by showing that NASH is PPAD-complete.

2 Recent Developments

Rogue problems. The FACTORING problem does not immediately belong to any of these classes, as its totality seems to draw from the Fundamental Theorem of Arithmetic: *"any non-prime has a prime divisor that is smaller"*. What is the relation between this important total problem and the subclasses of TFNP already defined? Recently, Emil Jeřábek [16] employed elementary algebraic number theory to show that FACTORING belongs to *both* PPAD and PPP through randomized reductions. *Combinatorial Nullstellensatz*, another "rogue" problem, was recently shown to be in PPA [25]. There are further rogue problems still defying classification within TFNP. Two examples are RAMSEY: *"Given a Boolean circuit encoding the edges of a graph with* $4n$ *nodes, find* n *nodes that are either a clique or an independent set"* and BERTRAND-CHEBYSHEV: *"Given* n*, produce a prime between* n *and* $2n$*"*, both embodying important and classical eponymous existence theorems in combinatorics and number theory, respectively.

The Class CLS. There had been a number of interesting total problems which have, frustratingly, defied polynomial-time algorithms for too long, and which were known to be inside *both* PPAD and PLS. Examples: finding an approximate fixpoint of a contraction map; finding the max-min of a simple stochastic game; finding a solution of a linear complementarity problem with a P-matrix; finding a Nash equilibrium in a network coordination, or congestion, game; or finding a stationary point of a multivariate polynomial. A new class called CLS (for *"continuous local search"*) lying within (and probably well within) the intersection of the two classes PLS and PPAD was defined in [9]; there are no known non-generic complete problems for this class (roughly, generic problems are ones whose instances contain generic boolean circuits). Very recently, a new problem was added to this collection, a version of the END OF THE LINE problem in which there is only one line starting at zero (no floating cycles or paths) and the nodes of the line are numbered $0, 1, 2, 3, \ldots$, while the remaining nodes are numbered ∞. This new problem, END OF THE METERED LINE, renders existing black-box complexity lower bounds for PPAD and PLS [26], as well as cryptographic lower bounds (see next), applicable to CLS as well.

Cryptographic assumptions. There are obvious connections between TFNP classes and Cryptography, as cryptosystems can be based on intractable TFNP problems such as FACTORING. In the other direction, if there are hashing functions that are secure with respect to collisions, then PPP is clearly not P. Recently, sophisticated arguments were articulated establishing hardness for other subclasses of TFNP, based on other standard (if not universally accepted) cryptographic assumptions. In [5, 11] it is shown that, if indistinguishability obfuscation of software is possible, then PPAD (and therefore NASH) is intractable. Such results can be extended to CLS, as noted above. Furthermore, it was recently established in [14] that, under the assumption that NP has a problem that is hard on the average for some distribution (an assumption that has many consequences of interest to cryptography but is not per se cryptographic), then PPAD also has such a problem. A new paper [18] shows that hardness of RAMSEY follows from the existence of collision-resistant hash functions.

Approximate NASH. The major concrete problem in complexity left open by the establishment of the PPAD-completeness of NASH had been whether there is a PTAS for approximate Nash equilibria, or whether there is a finite ε such that finding an ε-approximate Nash equilibrium is PPAD-complete. This was recently resolved by Aviad Rubinstein [24] in favor of the latter eventuality through a brilliant PCP construction for PPAD.

Finitary lemmata. One of the curiosities of the class TFNP and its subclasses is the surprising poverty (or, depending on your point of view, parsimony) of proof techniques needed to establish totality of functions in NP: in a quarter of a century, no new classes, no new combinatorial lemmata, have been added to the original five (we are not counting CLS, which lies within the intersection of all five classes). A look at the list of the five "combinatorial lemmata" reveals

that they share an intriguing property: *They are all finitary*, that is, they fail to hold if the underlying structure is infinite. It has been recently pointed out in [12], by using a classical theorem from Logic [13], that this is necessary: Any combinatorial lemma that is not finitary must yield a subclass of TFNP that is necessarily a subset of P — an observation that may help focus the pursuit of a sixth combinatorial lemma.

Provable TFNP and Wrong Proof

The fragmented nature of TFNP has been quite productive over the years, but it is also intriguing, and a little disturbing. Is there a way to unify all these genres of totality? Are there problems, perhaps complete for a master class, that generalize all total problems known to be complete for the subclasses?[3] Can we think of the phenomenon of total problems as a whole?

We have recently proposed such a unified theory [12]. The idea is to define a new class that we call *provable TFNP*, or PTFNP, which includes all total problems whose totality is proven within some minimalistic logical framework. The logical framework should be strong enough to supply proofs of the five lemmata, and versatile enough for the other important mission, namely identifying a complete problem for the whole class.

In [12] this was achieved through the framework of a first-order propositional logic. The language has a polynomial (in the underlying complexity parameter n) collection of Boolean variables of the form x_i, as well as the Boolean constants $0-1$, connectives such as $\vee, \rightarrow$ and quantifiers $\forall, \exists$, all with the standard meaning. Importantly, the framework also allows an exponential collection of n-ary Boolean function symbols, represented as $f_i(x_1, \ldots, x_n)$, where "f" is a symbol of the language and the index i is expressed in binary (it is at present an open question whether these function symbols are necessary). The framework also includes a rather standard axiom system, encompassing a complete semantic understanding of the roles of connectives, functions, and quantifiers (see [12] for details).

Once we have all that, and having fixed the complexity parameter n, we can now have *succinctly represented proofs* in our system; these proofs play an important role in providing us with a complete problem (and from that, as it is common with total functions, a definition of the class PTFNP). In particular, consider a Boolean circuit C, with n input gates and of size polynomial in n, which maps an input j (a binary integer with n bits) to $C(j)$ where $C(j)$ can be of one of two forms:

> *Either $C(j)$ is a sentence that holds due to an axiom (and this can be checked easily), or it is of the form $(F(j), k, \ell)$, where $F(j)$ is (the*

[3] Notice immediately that there is a trivial way of combining any finite number of classes with complete problems via some kind of "direct product" construction to obtain one all-encompassing class and complete problem. The challenge is to do this in a way that does not explicitly refer to the parts.

Boolean encoding of) a logical formula in the language, and k, ℓ are integers smaller than j, such that $F(j)$ follows from $F(k)$ and $F(\ell)$ due to an inference rule.

Call the above condition "correctness of C at j." Note that a circuit C satisfying the correctness condition at all $j = 0, \ldots, 2^n - 1$ encodes a *proof* of length 2^n in our system.

But suppose now that we are given such a circuit C standing for a purported proof in our language (it will be an actual proof if it is correct at all j), and we notice that the last line $C(2^n - 1)$ is of the form $(F(2^n - 1), k, \ell)$ *with* $F(2^n - 1) =$ FALSE. Since our logical system is consistent, it must be the case that there is a $j < 2^n$ such that C is not correct at j. The point is that *finding this j may be nontrivial!*

This is the total problem which we call WRONG PROOF:. Given n and a Boolean circuit C polynomial in n such that $F(2^n - 1) =$ FALSE, find a j such that C is not correct at j. Finally, we define PTFNP (for *provable* TFNP) as the class of all search problems in NP that reduce to WRONG PROOF.

The main theorem in [12] is the following:

Theorem 1. *PTFNP contains the five classes.*

We note that, in related work, Arnold Beckmann and Sam Buss prove in a recent paper [4] certain results that appear to be closely related to Theorem 1. They define two problems similar to WRONG PROOF, one corresponding to Frege systems, and another to extended Frege, and show these complete for two classes of total function problems in NP whose totality is provable within the bounded arithmetic systems U_2^1 and V_2^1, respectively. Theorem 1 differs from [4] in that we reduce TFNP problems to a propositional proof system capable of encoding concisely instances of these problems, without resorting to bounded arithmetic.

3 Open problems

There are many proof systems, some of them more powerful than others. It is possible that the one we propose in [12] can be simplified, while continuing to generalise PPP and the related complexity classes. So, one direction of future research is to look for minimal proof systems that continue to have this capability. One can also look in the opposite direction, and study more powerful proof systems, which may allow us to construct a hierarchy of increasingly general TFNP problems.

Some more concrete open problems include the following:

- *Is* FACTORING *in PPAD?* is a consequential and tempting conjecture, as the problem is in both PPP and PPA, the two important classes containing PPAD. Note that inclusion in PPAD would probably be through randomized reductions, as such are the reductions of [16] to PPP and PPA.

- *How about the remaining "rogue problems"* Bertrand-Chebyshev *and* Ramsey, discussed in the introduction? We conjecture that they are both in PPP.

- Are there natural complete problems for PPA? Even though several discrete fixpoint-type problems are by now known to be PPA-complete, they all contain in their input the description of a computational device such as a circuit or a Turing machine; see [10] for an intriguing possibility.

- *How about PPP?* It is shown in [2] that several natural problems in PPP such as Equal Sums (given n integers, find two subsets with the same sum modulo 2^n-1) and Dirichlet (Given n rational numbers and integer n find $\frac{1}{Nq}$ approximations for some denominator q) can be reduced to Minkowski (given an matrix A with determinant less than one, find a nontrivial combination of its rows within the unit square) and thus the latter becomes an interesting target in which to encode all of PPP.

- *How robust is PPP?* The class PPP can be parametrised as $\text{PPP}_K(n)$, for any function K: given $f : \{0,1\}^n \mapsto \{0,1\}^n$ identify either a collision $x \neq y$ with $f(x) = f(y)$, or a $x \in \{0,1\}^n$ such that $f(x) \leq K(n)$, where $\leq$ is meant in the standard binary notation. Obviously, PPP_0 is the standard PPP, and it is easy to see that $\text{PPP}_{p(n)} = \text{PPP}$, and $\text{PPP}_{2^n-p(n)} = \text{P}$ for all polynomials $p(n)$. But what happens for faster growing $K(n)$, for example if $K(n) = 2^{n-1}$ (the case known as "weak pigeonhole principle")? Are there oracle separation results?

- *Prove that one of the problems known to be in CLS is CLS-complete.* Alternatively, show that CLS is in P.

Acknowledgments: Many thanks to the "PPAD-like classes reading group" at the Simons Institute during the Fall 2015 program on Economics and Computation for many fascinating interactions. We also thank Arnold Beckmann, Pavel Pudlák, and Sam Buss for helpful discussions. Work supported by NSF grant CCF-1408635.

Bibliography

[1] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Annals of Mathematics* 160 (2): pp. 781–793 (2004).

[2] Reductions in PPP. F. Ban, K. Jain, C.H. Papadimitriou, C.A. Psomas, and Aviad Rubinstein. Manuscript, 2016

[3] P. Beame, S. Cook, J. Edmonds, R. Impagliazzo and T. Pitassi. The relative complexity of NP search problems. *27th ACM Symposium on Theory of Computing* pp. 303–314 (1995).

[4] A. Beckmann, Sam Buss. "The NP Search Problems of Frege and Extended Frege Proofs," preliminary draft December 2015.

[5] N. Bitansky, O. Paneth, and A. Rosen. On the cryptographic hardness of finding a Nash equilibrium. FOCS 2015, pp. 1480–1498 (2015).

[6] Sam Buss. "Bounded Arithmetic," Bibliopolis, Naples, Italy, 1986, www.math.ucsd.edu/∼sbuss/ResearchWeb/BAthesis/

[7] X. Chen, X. Deng and S.-H. Teng. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM* **56**(3) pp. 1–57 (2009).

[8] C. Daskalakis, P.W. Goldberg and C.H. Papadimitriou. The Complexity of Computing a Nash Equilibrium *SIAM Journal on Computing* **39**(1) pp. 195–259 (2009).

[9] C. Daskalakis and C. H. Papadimitriou. Continuous local search. SODA 2011, p. 790–804 (2011).

[10] A. Filos-Ratsikas, S. K. S. Frederiksen, P. W. Goldberg, and J. Zhang. Hardness Results for Consensus-Halving. *Corr*, arXiv:1609.05136 [cs.GT] (2016).

[11] S. Garg, O. Pandey and A. Srinivasan. Revisiting the Cryptographic Hardness of Finding a Nash Equilibrium. CRYPTO (2016).

[12] P.W. Goldberg and C.H. Papadimitriou. Towards a Unified Complexity Theory of Total Functions. Submitted (2017).

[13] J. Herbrand. *Récherches sur la théorie de la démonstration*, PhD thesis, Université de Paris, 1930.

[14] P. Hubáček, M. Naor, and E. Yogev. The Journey from NP to TFNP Hardness. 8th ITCS (2017).

[15] P. Hubáček and E. Yogev. Hardness of Continuous Local Search: Query Complexity and Cryptographic Lower Bounds. 2017 SODA (2017).

[16] E. Jeřábek. Integer factoring and modular square roots. *J. Comput. System Sci.* **82**(2) pp. 380–394 (2016).

[17] D. S. Johnson, C. H. Papadimitriou and M. Yannakakis. How Easy is Local Search?, *Journal of Computer and System Sciences*, 37(1): 79–100, 1988.

[18] I. Komargodski, M. Naor, and E. Yogev. White-Box vs. Black-Box Complexity of Search Problems: Ramsey and Graph Property Testing. *ECCC* Report 15 (2017).

[19] Jan Krajíček. Implicit Proofs, *Journal of Symbolic Logic*, **69**(2), pp. 387–397 (2004).

[20] Nimrod Megiddo. A note on the complexity of P-matrix LCP and computing an equilibrium. *Res. Rep. RJ6439, IBM Almaden Research Center, San Jose.* pp. 1–5 (1988).
[21] N. Megiddo and C. H. Papadimitriou. On Total Functions, Existence Theorems and Computational Complexity. *Theoretical Computer Science*, 81(2): 317–324, (1991).
[22] C.H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. System Sci.* **48** pp. 498–532 (1994).
[23] P. Pudlák. On the complexity of finding falsifying assignments for Herbrand disjunctions. *Arch. Math. Logic*, 54, pp. 769–783 (2015).
[24] A. Rubinstein. Settling the complexity of computing approximate two-player Nash equilibria. FOCS (2016).
[25] L. Varga. Combinatorial Nullstellensatz modulo prime powers and the Parity Argument. arXiv:1402.4422 [math.CO] (2014).
[26] S. Zhang. Tight bounds for randomized and quantum local search. SIAM J. Comput., 39(3), pp 948–977 (2009).