A logical approach to TFNP

Neil Thapen

Institute of Mathematics
Czech Academy of Sciences

Reflections on Propositional Proofs in Algorithms and Complexity

FOCS 2021, online, 7 February 2022

A logical approach to TFNP – some references

- Buss, Krajíček 1994
  An application of Boolean complexity to separation problems in bounded arithmetic
- Jeřábek 2009
  Approximate counting by hashing in bounded arithmetic
- Skelley, Thapen 2011
  The provably total search problems of bounded arithmetic
- Beckmann, Buss 2014
  Improved witnessing and local improvement principle for second-order bounded arithmetic
- Kołodziejczyk, Thapen 2018
  Approximate counting and NP search problems

The talk is based on the last paper, which has detailed references.

Outline

Total Functional NP

Some logic

Bounded arithmetic and TFNP

Approximate counting and TFNP

Total Functional NP

Some logic

Bounded arithmetic and TFNP

Approximate counting and TFNP

Total Functional NP

Complexity theory usually studies **decision problems**.
E.g. given a graph, decide whether it has a Hamiltonian path.

We will discuss instead **search problems**.
E.g. given a graph, find and output a Hamiltonian path.

We are interested in search problems where

- the thing being searched for always exists
- it is recognizable in polynomial time
- it is not too big.

The class of such problems is Total Functional NP, or TFNP.

Two examples

TFNP problems appear in yellow.

PIGEON

Input: a number n (in binary) and a circuit C

The circuit C specifies a function $f : n + 1 \to n$.

Output: a collision in f.

That is, distinct $x, x' < n + 1$ such that $f(x) = f(x')$.

RAMSEY

Input: a number n (in binary) and a circuit C

The circuit C specifies a graph G on vertices $[0, n)$.

Output: A clique or anticlique in G of size $\frac{1}{2} \log n$.

Formal definition

Definition

A TFNP problem is specified by a p-time relation $R(x, y)$ and a polynomial bound p satisfying $\forall x \exists y < 2^{p(|x|)} R(x, y)$.

We use R as the name for the problem, suppressing the bound p.

The variables x, y range over binary strings, which we identify with natural numbers whenever convenient.

Usually the input x is a compact description of an exponential size object. In **relativized** problems, this object may be directly described by an oracle.

Comparing problems

Let P, Q be problems in TFNP.

P is reducible to Q, written $P \leq Q$, means

"we can efficiently solve P with the help of one call to Q"

Definition

$P \leq Q$ if there are p-time functions f and g such that

$$\forall x \forall z \qquad Q(f(x), z) \rightarrow P(x, g(x, z)).$$

$$(\text{ "solve } Q \text{ at } f(x)\text{"} \implies \text{"solve } P \text{ at } x\text{" })$$

P is equivalent to Q, written $P \equiv Q$, if $P \leq Q$ and $Q \leq P$.

Structure of TFNP

FP $:=$ problems solvable in p-time.

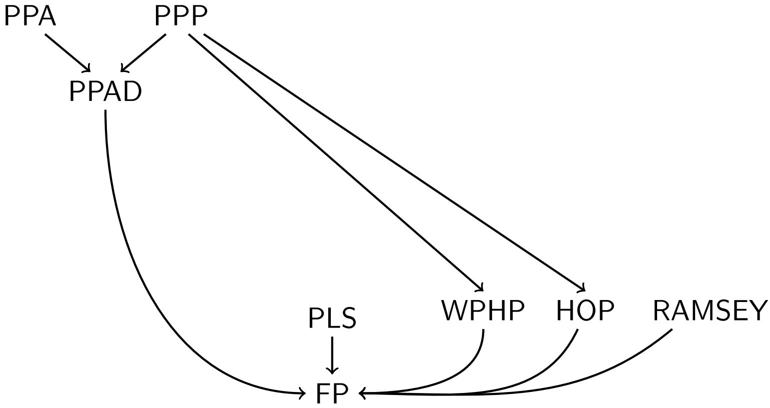
Other subclasses of TFNP are PPAD, PLS, PPP, ...
These consist of all problems reducible to a specified problem.
E.g. PPP $:= \{Q \in \text{TFNP} : Q \leq \text{PIGEON}\}$.

If P$=$NP then all these classes collapse to FP.

Why study the structure of TFNP classes?

- we want to solve search problems
- some are easy (e.g. in FP)
- some appear hard. What are the ways they can be hard?

Selected TFNP classes



Solid arrows indicate strict containment of one class in another. Equivalently, a reduction $\leq$ that goes strictly in one direction. (I am not distinguishing carefully between the names of classes and the names of their complete problems.)

Total Functional NP

Some logic

Bounded arithmetic and TFNP

Approximate counting and TFNP

Background - how strong is a theory?

A **theory** is just a set of sentences (describing properties of the natural numbers).

PA := algebraic axioms + induction for all formulas

$\mathsf{I}\Sigma_k$:= algebraic axioms + induction for Σ_k formulas [Σ_k-IND]

$$\mathsf{I}\Sigma_0 < \mathsf{I}\Sigma_1 < \mathsf{I}\Sigma_2 < \cdots < \mathsf{PA} < \cdots < \mathsf{ATR}_0 < \dots$$

As we move to the right, we can define more recursive functions.
That is, we can prove more recursive functions are total.
This gives a common measure for quite different theories.

There are sophisticated ways to measure this increase in strength, using ordinals or measuring the growth rate of functions.

From computability to complexity

We replace recursive functions with TFNP problems.

Definition

For a theory T,

$$\text{TFNP}(T) := \{R \in \text{TFNP} : T \text{ proves } R \text{ is total }\}.$$

What is language is T in? How do we express "R is total"?

- The language consists of a name for every p-time relation and function. E.g. $x < y$, $x \mapsto 2^{p(|x|)}$ and $R(x, y)$.
- "R is total" is the formal sentence $\forall x \exists y < 2^{p(|x|)} R(x, y)$.
- As before, variables are binary strings, identified with numbers.

A convenient base theory

Definition

The theory BASE consists of every true sentence of the form $\forall \bar{x} \phi(\bar{x})$ where ϕ is quantifier free (that is, p-time).

Recall that the condition $Q \leq P$ is such a sentence, namely

$$\forall x \forall z \quad Q(f(x), z) \rightarrow P(x, g(x, z)).$$

Proposition

If T contains BASE, then TFNP(T) is closed under reductions.

BASE also contains the algebraic axioms of PA.

For our questions, it is harmless to work only with theories containing BASE (called in the literature e.g. $\forall$PV($\mathbb{N}$)).

Examples

TFNP(BASE) $=$ FP (the problems solvable in p-time)

TFNP(BASE+"R is total"), for some $R \in$ TFNP

Depends on robustness of R.
If R is PLS, this is just PLS.

TFNP(PA) or TFNP(ZFC) $= \ldots ?$

Cannot be all of TFNP, unless you add BASE.
"Given a purported proof of $0 = 1$ in PA, find a mistake"
is a TFNP problem that PA does not prove is total.
Discussed in [Beckmann 09, Pudlák 17]

TFNP($\{$all true sentences$\}$) $=$ TFNP

Total Functional NP

Some logic

Bounded arithmetic and TFNP

Approximate counting and TFNP

Bounded arithmetic

This is a "resource bounded" version of PA [Buss 85].

Definition

The theory Σ_k^P-IND consists of

- BASE, and
- the induction axiom for every Σ_k^P formula

The induction axiom for a formula ϕ is

$$\phi(0) \wedge \forall x(\phi(x) \to \phi(x+1)) \quad \to \quad \forall x \phi(x).$$

A Σ_k^P property is one at level k in the polynomial hierarchy.
Σ_1^P is NP, $\quad \Sigma_2^P$ is $\mathrm{NP}^{\mathrm{NP}}$, $\quad$ etc.

We need to specify a particular machine to express a Σ_k^P property in a formula. But it does not matter which one, as BASE proves they are all equivalent.

Bounded arithmetic

Bounded arithmetic provides a natural way to reason with and about concepts from complexity theory.

Two properties we use here are that it can be translated into small, uniform constant-depth Frege proofs, and the following:

Theorem [Buss 85]

Σ_k^P-IND proves
"Every $\mathrm{P}^{\Sigma_k^P}$ machine has a computation on every input".
For our purposes, Σ_k^P-IND is equivalent to this statement.

Polynomial Local Search and Σ_1^P-IND

PLS (SINK OF DAG) [JPY 88]

Input: A number n (in binary) and a circuit C

C specifies a DAG G on $[0, n)$, in which node 0 is a source.

Output: A sink in G.

Here DAG means that every edge (x, y) in G has $x < y$.

This has many natural equivalent problems.

E.g. find a local minimum of a function over a low degree graph.

(Formally, PLS is the class of problems $\leq$ SINK OF DAG.)

Theorem [BK 94]

$\mathrm{TFNP}(\Sigma_1^P\text{-IND}) = \mathrm{PLS}$

Coloured Polynomial Local Search

CPLS [KST 07]

A version of PLS with some extra structure.
Find a sink in a DAG, where each node is labelled with a set of colours that must satisfy some local conditions.

Theorem

$\mathsf{TFNP}(\Sigma_2^P\text{-}\mathsf{IND}) = \mathsf{CPLS}$

This principle is useful for proof complexity lower bounds, as it is in some sense the strongest combinatorial principle with a short proof in resolution.

Game Induction Principle

k-turn Game Induction Principle, GI_k [ST 11]

Input: A sequence of k-turn, two-player games, together with a winning strategy for Player B in the first game, and functions to change a strategy for game i into a strategy for game $i+1$.

Output: Winning moves for Player B in the last game.

Theorem

$\mathrm{TFNP}(\Sigma_k^P\text{-IND}) = \mathrm{GI}_k$

GI_k is equivalent to [PT 12]:

Generalized PLS, GPLS_k - A version of PLS where you have to optimize an alternating min of max of min etc. rather than just find a minimum.

Polynomial Time Equilibrium, PE_k - Find a Nash equilibrium in a k-turn, succinctly given game, where players can only make p-time revisions to their strategies.

TFNP

$$
\begin{tikzcd}
\mathsf{PPA} \arrow[dr] & & \mathsf{PPP} \arrow[dl] & & & & \\
& \mathsf{PPAD} \arrow[ddddrr, bend right] & & \mathsf{GI}_4 \arrow[d, dotted] & & & \\
& & & \mathsf{GI}_3 \arrow[d, dotted] \arrow[ddrrr] & & & \\
& & & \mathsf{CPLS} \equiv \mathsf{GI}_2 \arrow[d] \arrow[dr] \arrow[drr] & & & \\
& & & \mathsf{PLS} \equiv \mathsf{GI}_1 \arrow[d] & \mathsf{WPHP} \arrow[dl, bend left] & \mathsf{HOP} \arrow[dll, bend left] & \mathsf{RAMSEY} \arrow[dlll, bend left] \\
& & & \mathsf{FP} & & &
\end{tikzcd}
$$

$\mathsf{GI}_k \equiv \mathsf{TFNP}(\Sigma_k^P\text{-IND})$

New arrows follow from known proofs in bounded arithmetic [Pudlák 91, MPW 02]

Main open problem

For $k \geq 2$, show $\mathsf{GI}_k \not\equiv \mathsf{GI}_{k+1}$, w.r.t. some oracle.
In other words, show Σ_{k+1}^P-IND is stronger than Σ_k^P-IND.

Second order theories

[Buss 85] also introduced "second-order" bounded arithmetic theories. These are similar to Σ_k^P-IND, but also allow quantification over exponentially large objects; essentially oracles.

Two important theories are U_2^1 and its extension V_2^1.

Theorem [Buss 85]

U_2^1 proves
"Every PSPACE machine has a computation on every input".

V_2^1 proves
"Every EXPTIME machine has a computation on every input".

For our purposes, U_2^1 and V_2^1 are equivalent to these respective statements.

These are uniform versions of Frege and Extended Frege proof systems. U_2^1 can formalize counting and is very good at proving combinatorial statements.

Local Improvement Principle

Local Improvement Principle, LI [KNT 11]

Input: A large DAG, with an initial labelling of nodes, and local rules about how to relabel nodes to improve their "score".

Output: A labelling of part of the graph with maximal score.

Linear Local Improvement Principle, $\mathsf{LLI}_{\mathsf{log}}$

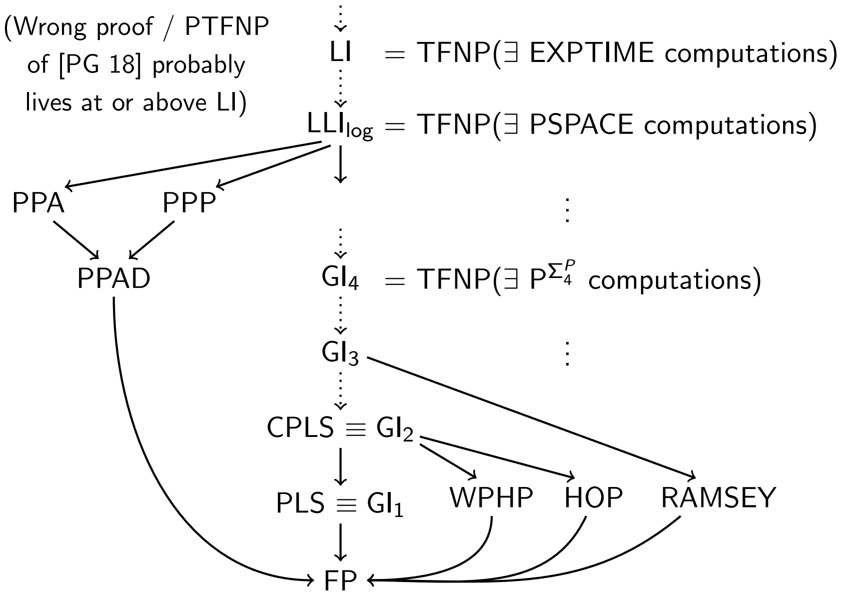
A restriction of LI, where the DAG is just a line and scores are at most logarithmic in the size parameter.

Theorem [KNT 11, BB 14]

$\mathsf{TFNP}(U_2^1) = \mathsf{LLI}_{\mathsf{log}}$

$\mathsf{TFNP}(V_2^1) = \mathsf{LI}$

TFNP


(Wrong proof / PTFNP
of [PG 18] probably
lives at or above LI)
LI = TFNP(∃ EXPTIME computations)
$\mathsf{LLI}_{\log}$ = TFNP(∃ PSPACE computations)
PPA
PPP
PPAD
GI_4 = TFNP(∃ $\mathsf{P}^{\Sigma_4^P}$ computations)
GI_3
CPLS ≡ GI_2
PLS ≡ GI_1
WPHP
HOP
RAMSEY
FP

Total Functional NP

Some logic

Bounded arithmetic and TFNP

Approximate counting and TFNP

Approximate counting

The problems WPHP, HOP and RAMSEY can all be proved total using arguments based on approximate counting.

Proof of the finite Ramsey Theorem

We are given a graph G of size n. Choose the first node in G. Either it is has an edge to at least $(\frac{1}{2} - \epsilon)n$ other nodes, or it is has no edge to at least $(\frac{1}{2} - \epsilon)n$ other nodes ...

Q1. Is there a formal theory capturing this kind of counting?

Q2. Is **every** TFNP-style principle, provable in first-order bounded arithmetic, already provable in approximate counting?

Q3. Is **every** TFNP-style principle, provable in first-order bounded arithmetic, already provable in Σ_2^P-IND / reducible to CPLS?

A1. Yes

A2. No (at least relative to an oracle)

A3. This is the main open problem again

Approximate counting

Definition [Jeřábek 09]

The theory APC_2 consists of

- Σ_1^P-IND, and
- $\mathrm{rWPHP}(\mathrm{P}^{\mathrm{NP}})$, the P^{NP} retraction weak pigeonhole principle stating there is no P^{NP} surjection $n \twoheadrightarrow 2n$. with P^{NP} inverse.

Theorem [Jeřábek 09]

In APC_2 we can express "a set X has size approximately n, with some multiplicative error" and can formalize proofs using this notion, such as the Ramsey theorem, the tournament principle, etc.

Theorem [KT 18]

APC_2 does not prove that CPLS is total.

Proof that APC_2 does not prove CPLS is total, part 1

Definition

A Σ_2^P search problem is like a TFNP problem, except that the relation $R(x,y)$ is coNP, rather than p-time.

Given a purported solution to such a problem, there may be a counterexample, showing that it is not a solution.

Definition

A TFNP problem $Q(x,y)$ is **PLS counterexample reducible** to a Σ_2^P search problem $R(x',y')$ if we can solve Q as follows:

1. Given x, compute in p-time an instance x' of R.
2. Let y' be a purported solution of R satisfying $R(x',y')$.
3. Compute y from x and y' by solving a PLS problem.
4. Either y is a solution satisfying $Q(x,y)$, or y is a counterexample witnessing that $R(x',y')$ is false.

Proof that APC_2 does not prove CPLS is total, part 2

We formalize the P^{NP} retraction weak pigeonhole principle as a class $rWPHP_2$ of Σ_2^P search problems.

Definition

The class APPROX consists of all TFNP problems which are PLS counterexample reducible to problems in $rWPHP_2$.

Theorem

$\mathrm{APPROX} = \mathrm{TFNP}(\mathrm{APC}_2)$

Theorem

$\mathrm{CPLS} \notin \mathrm{APPROX}$, relative to an oracle.

Proof that APC_2 does not prove CPLS is total, last part

Theorem (from last slide)

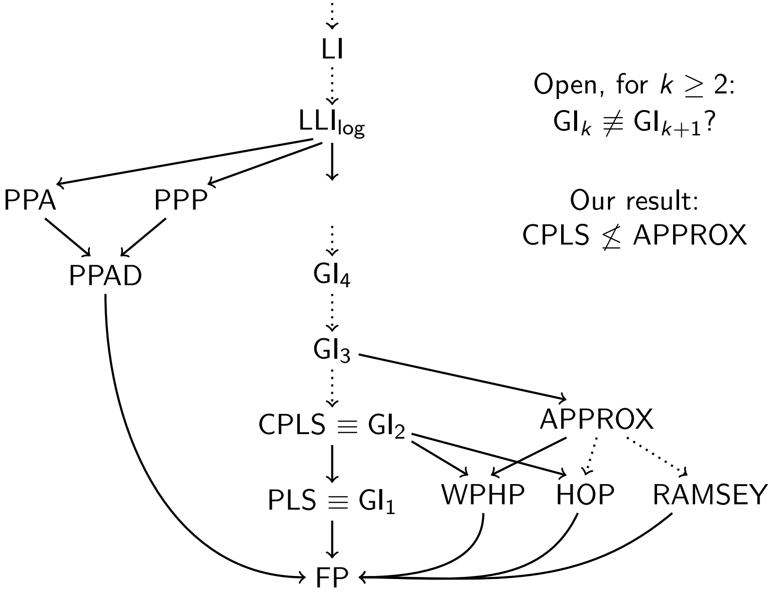
CPLS $\notin$ APPROX, relative to an oracle.

Proof. We use a random restriction and a kind of switching lemma, developed in [PT 19], to construct a partial oracle α for an instance of CPLS such that

- α does not contain a solution to CPLS
- α fixes replies to every NP query made by a P^{NP} machine on most inputs, and thus the output of the machine, for a suitable notion of "fixes".

We show that α contains a purported solution y' to our rWPHP(P^{NP}) problem, but it is still hard to find a solution to CPLS in α or to find a counterexample witnessing that y' is not a solution to rWPHP(P^{NP}), even if we have the power to solve PLS problems. □

TFNP



Open, for $k \geq 2$:
$\mathsf{GI}_k \not\equiv \mathsf{GI}_{k+1}$?

Our result:
$\mathsf{CPLS} \not\leq \mathsf{APPROX}$