I. Introduction to NP Functions and Local Search

Sam Buss (UCSD)
sbuss@math.ucsd.edu

Prague, September 2009

NP Functions — TFNP

[JPY'88, Papadimitriou'94].

Definition

TFNP, the class of Total NP Functions is the set of polynomial time relations $R(x, y)$ such that $R(x, y)$ implies $|y| = |x|^{O(1)}$ and such that R is *total*, i.e., for all x, there exists y s.t. $R(x, y)$.

Thm. If TFNP problems are in P, then $\mathrm{NP} \cap \mathrm{coNP} = P$.

Pf. If $(\exists y \leq s)A(x, y) \leftrightarrow (\forall y \leq t)B(x, y)$ is in $\mathrm{NP} \cap \mathrm{coNP}$, then $A(x, y) \vee \neg B(x, y)$ defines a TFNP predicate. $\square$

Thus, any $\mathrm{NP} \cap \mathrm{coNP}$ predicate gives a TFNP problem.

Examples of TFNP problems

Prime factorization: Using Pratt's polynomial size certificates for primes, the predicate

$$R(x, y) \ := \ \text{“}y \text{ is a prime factorization of } x\text{”}$$

is in TFNP.

Local Optimization: Any local optimization problem with polytime checkable condition

$$R(x, y) \ := \ \text{“}y \text{ is a local optimum for instance } x\text{”}$$

is in TFNP. Examples include Dantzig's algorithm for linear programming, Lin-Kernighan for Traveling Salesman, Kernighan-Lin for graph partition, etc.

Def'n: $[n] = \{0, 1, \ldots, n-1\}$.

Pigeonhole principle. PHP_n^{n+1}. If f is a mapping from $[N]$ to $[N-1]$, then there is some $y = \langle i, j\rangle$, $i < j$ such that $f(i) = f(j)$.

Graph properties. If G is a graph, all vertices of degree ≤ 2, and node 0 has degree 1, then there is another node y with degree 1. Input: Function f specifying G; for each vertex s, $f(s)$ gives the edges incident to s. Solution: $y \neq 0$ of degree 1.

LEAF: Undirected graph, vertex 0 degree 1, all other nodes degree ≤ 2. Solution: Another node of degree 1.

SINK.OR.SOURCE: Directed graph, node 0 indegree 0, outdegree 1. All indegrees and outdegrees ≤ 1. Solution: another node with indegree or outdegree zero.

SINK: Same as SINK.OR.SOURCE, but solution is a node with outdegree 0.

Polynomial Local Search (PLS)

Inspired by Dantzig's algorithm and other local search algorithms:

Definition (JPY'88.)

A PLS problem consists of polynomial time functions: $N(x,s)$, $i(x)$, and $c(x,s)$, polynomial time predicate $F(x,s)$, and polynomial bound $b(x)$ such that

0. $\forall x(F(x,s) \rightarrow s \le b(x))$.
1. $\forall x(F(x,i(x)))$.
2. $\forall x(N(x,s) = s \vee c(x,N(x,s)) < c(x,s))$.
3. $\forall x(F(x,s) \rightarrow F(x,N(x,s)))$.

A solution is a point s such that $F(x,s)$ and $N(x,s) = s$.

Thus, a solution is a local minimum. Clearly, a PLS problem is in TFNP.

Reductions among TFNP problems

Definition: Let $R(x,y)$ and $Q(x,y)$ be TFNP problems. A polynomial time many-one reduction from R to Q (denoted $R \preccurlyeq Q$) is a pair of polynomial time functions $f(x)$ and $g(x,y)$ so that, for all x, if y is a solution to $Q(f(x),y)$, then $g(x,y)$ is a solution to R, namely $R(x,g(x,y))$.

A polynomial time, Turing reduction, $R \preccurlyeq_T Q$ is a polynomial time Turing machine that solves R making (multiple) invocations of Q. It must succeed no matter which solutions y are returned in response to its queries $Q(-,-)$.

A PLS-complete circuit problem.

Definition

A instance of FLIP is a Boolean circuit, m inputs and n outputs interpreted as n-bit integer. The feasible points s are m-bit inputs. Cost $c(s)$ is the integer output. Neighbors of s are the m points at Hamming distance one. $N(s) =$ any neighbor of lower cost.

Thm: [JPY'88] FLIP is many-one complete for PLS.

Thm: [JPY'88] The Kernighan-Lin problem for minimum weight graph partitioning (LOKL) is many-one complete for PLS.

Relativized (type 2) TFNP problems

Definition

A *relativized* or *Type 2* TFNP problem $R(1^n, f, y)$ is a (oracle) polynomial time predicate that takes as input a size bound 1^n and one or more functions $f : [2^n] \to [2^n]$. A solution is any value such that $R(1^n, f, y)$.

Definition: A many-one reduction from R to Q is a triple of oracle polynomial time functions:
- a mapping $1^n \mapsto 1^m$, that is computing $m = m(n)$.
- a function $\beta^f : [2^m] \to [2^m]$,
- a function γ^f such that if y is a solution to $Q(1^m, \beta^f, y)$, then γ^f computes a solution to $R(1^n, f, \gamma^f(1^n, y))$.

In many cases, the first function does not need to make oracle calls, and we henceforth assume this is the case.

Some subclasses of TFNP [P'94, BCEIP'98]

Class	$\preccurlyeq_T$-Complete problem
PPA	LEAF
PPAD	SINK.OR.SOURCE
PPADS	SINK
PPP	PHP_n^{n+1}

Theorem (BCEIP'98)

In the relativized (oracle) setting, we have:
$\mathrm{PPAD} \subset \mathrm{PPADS} \subset \mathrm{PPP}$ *and*
$\mathrm{PPAD} \subset \mathrm{PPA}$
Furthermore, no other inclusions hold.

Theorem (Morioka)

In relativized setting, $\mathrm{PPAD} \not\leq_T \mathrm{PLS}$*.*
The same holds for PPADS, PPP, PPA *in place of* PPAD*.*

Proof: (By contradiction.) Let M be an oracle Turing machine that reduces instances $\langle \alpha, n \rangle$ of $\textsc{Sink.or.Source}$ to a PLS problem. α is to specify an undirected graph G on $[n]$. W.l.o.g., α is a function $\alpha(i) = \langle j, k \rangle$ meaning that the only edge into i is from j, and the only edge out from i is to k. The goal is find that node 0 has degree $\neq 1$, or to find another node with degree 1, or to find an inconsistency in α's specification of the undirected graph G.
M can query α and can invoke PLS problems P with inputs β, x where β is a code for a polynomial-time (fixed time bound) Turing machine M^* that can query the oracle α and that computes values of the F, i, N, and c for the queried instance P of PLS.
Any such query by M returns a solution to P.

Proof continued: As M runs, it fixes a finite portion of the graph G so that 0 has degree 1 and all other nodes have in- and out-degrees ≤ 1. We call such partial graphs "good".

In the end, only polynomially many edges will have been specified. This suffices to obtain a contradiction, since M will have no way of outputting a node that must have in- or out-degree 0.

Each time M queries α, set its value arbitrarily so as to maintain goodness. When a PLS problem P is invoked: find a least cost value c_0 such that some feasible point x_0 has cost c_0 when computed using some *good* (polynomial size) extension of α. Set α accordingly. Then arbitrarily extend α further to a good partial graph that sets enough values of α to allow the values of F, c and N at x_0 to be computed.
Q.E.D.

On the other hand, we have:

Theorem (Buresh-Oppenheim, Morioka '04)

$\mathrm{PLS} \not\preccurlyeq \mathrm{PPA}$.

Hence $\mathrm{PLS} \not\preccurlyeq \mathrm{PPAD}$.

Open question: Does $\mathrm{PLS} \preccurlyeq \mathrm{PPADS}$ hold?

This is known not to hold with a "nice" (=parsimonious) reduction.

More classes

1. α-PLS, for α an ordinal ϵ_0 or Γ_0. PLS is modified so that costs are notations for ordinals $< \alpha$. [Beckmann-Buss-Pollett'02].
2. Colored PLS [Krajíček-Skelley-Thapen'07]
3. RAMSEY. Input: an undirected graph G on N nodes. Output: a homogeneous set of size $\frac{1}{2} \log N$.
4. Weak PHP, PHP_n^{2n}, etc.

Thm: α-PLS is many-one equivalent to PLS for $\alpha = \epsilon_0$ and $\alpha = \Gamma_0$. [BBP'02]

Thm: Colored PLS is strictly stronger than PLS. [KST'07]

Question: What is the strength of RAMSEY? Of PHP_n^{2n}?

Question: What is the strength of FACTORING? [P'94]

First-order representations for search problems.

Definition

A first-order, existential formula ϕ, interpreted in finite structures $[N]$, defines a TFNP-problem, provided ϕ is valid in all finite structures.

Examples:

PHP: $(\exists x)(\exists y)(f(x) = 0 \lor (x \neq y \land f(x) = f(y)))$

$\mathrm{ontoPHP}$:
$(\exists x)(\exists y)(f(g(x)) \neq x \lor f(x) = 0 \lor (x \neq y \land f(x) = f(y)))$

Translations to propositional logic [Wilkie-Paris]

To translate $\exists\vec{x}\phi$ to a family of propositional formulas. First write ϕ as a DNF in which no functions are nested, w.l.o.g. For each $N > 0$, translate as the following family of formulas using variables $p_{i,j}$ for $f(i) = j$, $q_{i,j}$ for $g_{i,j}$, etc.

(a) Function totality. $\bigvee_{j\in[N]} p_{i,j}$ (for each $i \in [N]$, each f).

Functionality. $\overline{p_{i,j}} \vee \overline{p_{i,k}}$, for each i, each $j \neq k$.

(b) For assignment of values to $\vec{x}$ from $[N]$, each disjunct of ϕ becomes a conjunction of atoms.

Propositional translation is sequent: $(a) \longrightarrow (b)$.

More commonly, we wish to a refutation system. $\neg\exists\vec{x}\phi$ becomes an (unsatisfiable) set Γ_ϕ of clauses: namely, clauses (a), and the clauses which are the negations of conjunctions (b).

Example: The Paris-Wilkie translation of PHP^N_{N-1} is Γ_{PHP} containing the clauses

$\bigvee_j p_{i,j}$,	for all $i = 0, \ldots, N-1$.
$\neg p_{i,j} \vee \neg p_{i,k}$,	for all $j \neq k$, all i.
$\neg p_{i,0}$,	for all $i = 0, \ldots, N-1$.
$\neg p_{i,j} \vee \neg p_{i',j}$,	for all $i \neq i'$, all j.

Since Γ_{PHP} is unsatisfiable, it has a refutation. Let the depth of a formula be the maximum nesting of (blocks of) $\wedge$'s and $\vee$'s in the formula. The *depth* of a proof is the max depth of any formula in the proof. It is a now-classic theorem that Γ_{PHP} requires exponential size to refute with bounded depth propositional proofs [BIKPPW].

Definition

$\Gamma_R \le_{bdLK} \Gamma_Q$ provided there are quasipolynomial size, bounded depth propositional proofs, from Γ_R, of each 'clause' of some substitution instance of Γ_Q.

Theorem (B-O,M)

If $R \preccurlyeq Q$, then $\Gamma_R \le_{bdLK} \Gamma_Q$.

They also prove a similar result about reducibility with respect to Nullstellensatz proofs, $\le_{HN(d)}$, via constant degree d reductions.

As corollaries, one gets nearly all known independence results for $\preccurlyeq$-reductions among TFNP.

Theorem (B-O&M)

If $R \preccurlyeq Q$, then $\Gamma_R \leq_{bdLK} \Gamma_Q$.

Proof. (Sketch.) Let $R = R(f, 1^n)$ and $Q = (g, 1^m)$. The $\preccurlyeq$ reduction gives $m = m(n)$ and gives a polynomial time oracle machine M^* computing $g(i)$, $i \in [2^m]$. An execution of M^* on input i can be viewed as a polynomial depth decision tree T_i. Each node in T_i queries some value $j \in dom(f)$ and branches 2^n ways to give the value of $f(j) = \ell$ — this branch is labeled with $p_{j,\ell}$. Each leaf of T_i has a label k, indicating $g(i) = k$.

Identify paths in T_i with the conjunction of literals $p_{i,\ell}$ on the path. Define the condition $g(i) = k$ as the disjunction over the paths in T_i that end with label k.

The DNF formulas for $g(i) = k$ give the substitution instance of Γ_Q.

Proof cont'd.

The instances of the functionality and totality clauses of Γ_Q follow readily from those of Γ_R.

Consider some other clause C in Γ_Q. If (the instance of) C falsified by some values of g, then the $\preccurlyeq$ reduction runs a further oracle polynomial time procedure to find a clause of Γ_R that is falsified. Form a decision by cascading, (a) decision trees for the values of g needed for falsifying C, and (b) for the paths that falsify C, append the decision tree of queries made by the subsequent oracle polynomial time procedure, γ.

Each path in this big decision tree that falsifies C also contains an explicit falsification of some clause in Γ_R. $\square$

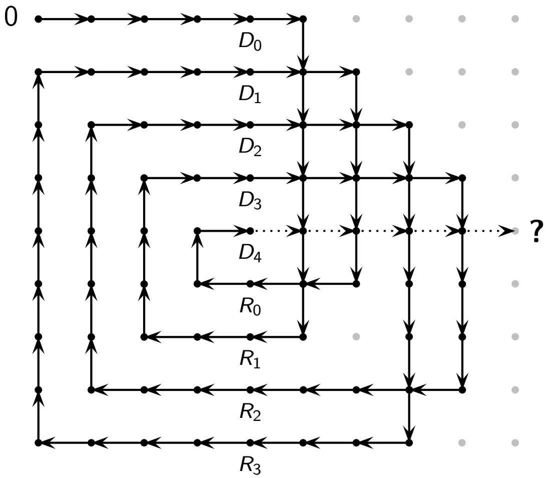
The first-order framework does not apply directly to all natural TFNP problems. For example, RAMSEY would be formulated as

$$\exists\langle x_0,\ldots,x_{\frac{1}{2}|N|}\rangle\forall i,j,k\in[\tfrac{1}{2}|N|].i\notin\{j,k\}\rightarrow(e(x_i,x_j)\leftrightarrow e(x_i,x_k))$$

The second quantifier is a kind of "sharply bounded" quantifier of the type used in bounded arithmetic, with $|N|\approx\log N$.

It is also possible to consider higher-level reductions between TFNP problems than $\preccurlyeq$- and $\preccurlyeq_T$-reductions. Consider, for example, the reduction shown on the next page from PHP^N_{N-1} to the $\textsc{Sink}$ principle.

For PHP^N_{N-1} it is a bounded depth reduction (in terms of definability in bounded arithmetic). Only for $\mathrm{ontoPHP}^N_{N-1}$ is it a $\preccurlyeq$-reduction.



How to build an instance of $\mathrm{S{\small INK}}$ from an instance of PHP_4^5. The dotted path from D_4 would connect back up to a R_i point if we had a contradiction to the pigeonhole principle.

Some selected references

- D. Johnson, C. Papadimitriou, M. Yanakakis, "How easy is local search?", *JCSS* 37 (1988) 79-100.
- C. Papadimitriou, "On the complexity of the parity argument and other inefficient proofs of existence", *JCSS* 48 (1994) 498-523.
- D. Gale, "The game of Hex and the Brouwer fixed-point theorem", *Amer. Math. Monthly* 86 (1979) 818-827.
- P. Beame, S. Cook, J. Edmonds, R. Impagliazzo,, T. Pitassi, "The relative complexity of NP search problems," *JCSS* 52 (1998) 3-19.
- J. Buresh-Oppenheim, T. Morioka, "Relativized NP search problems and propositional proof systems, 19th CCC, 2004, pp. 54-67.
- "A compendium of PPAD-complete problems", http://www.cc.gatech.edu/~kintali/ppad.html
- A. Beckmann, S. Buss, C. Pollett, "Ordinal Notations and Well-Orderings in Bounded Arithmetic", *APAL*, 120 (2003) 197-223.
- J. Krajíček, A. Skelley, N. Thapen, "NP search problems in low fragments of bounded arithmetic", *JSL* 72 (2007) 649-672.
- S. Buss, "Polynomial-size Frege and resolution proofs of st-connectivity and Hex tautologies", *TCS* 52 (2006) 35-52.
- P. Nguyen, S. Cook, "The complexity of proving the discrete Jordan curve theorem", *LICS*, 2007, pp. 245-256.