

Lecture 18

Total search and TFNP

Instructor: Prof. Constantinos Daskalakis (costis@mit.edu)

[Corollary L1.10](#) guarantees a Nash equilibrium, but its existence proof does not give a polynomial-time algorithm. We have already met a similar distinction in [Sperner's lemma \(Theorem L2.1\)](#): a fully labeled simplex exists, yet the path certifying existence can be exponentially long. Today we formalize search problems whose solutions are guaranteed to exist, and isolate the directed parity argument underlying PPAD [\[Pap94\]](#).

L18.1 From decision to search

A decision problem asks for a yes/no answer. A search problem asks for a witness. For example, satisfiability asks whether a Boolean formula has a satisfying assignment; the corresponding search problem asks us to produce one when it exists.

Definition L18.1 (Polynomially verifiable search). A relation $R(x, y)$ specifies the valid solutions y to instance x . It is polynomially balanced if there is a polynomial p such that

$$R(x, y) \implies |y| \leq p(|x|).$$

The class FNP consists of search problems with a polynomially balanced relation decidable in polynomial time. A search algorithm must return a valid witness whenever one exists.

The length bound is part of the definition: efficient verification alone would not ensure that a solution can even be written in polynomial time. Instances and witnesses here are finite binary strings. When a mathematical solution is a real vector, its required finite representation must be specified.

Definition L18.2 (TFNP). A search problem is in TFNP if it is in FNP and is total:

$$\forall x, \exists y : R(x, y).$$

An existence theorem supplies the last condition. Polynomial balance and verification must still be established separately. Ill-formed encodings can be handled by a designated, efficiently recognizable witness; they should not silently become exceptions to totality.

For a rational finite game and rational $\epsilon > 0$, an ϵ -Nash equilibrium can be represented by a rational profile of polynomial encoding length in the game description and $\log(\frac{1}{\epsilon})$. To see the length bound, round an exact equilibrium to a sufficiently fine rational grid: multilinearity and bounded payoffs control the resulting change in every deviation gain. Payoffs and deviations can then be evaluated in polynomial time for an explicitly represented game.

Remark L18.3 (Exact and approximate equilibria). Two-player games with rational payoffs have rational equilibria of polynomial encoding length. With three or more players, exact equilibria can require [irrational probabilities](#) ([Section L3.1.3](#)), so one cannot simply use a rational profile as the FNP witness for exact Nash. The algebraic fixed-point class FIXP captures this different issue [\[EY10\]](#).

L18.2 Reductions between search problems

Definition L18.4 (Search reduction). A polynomial-time reduction from relation R to relation Q consists of polynomial-time functions f and g such that, whenever R has a solution on x , the target problem has a solution on $f(x)$ and every such solution satisfies

$$Q(f(x), y) \implies R(x, g(x, y)).$$

Thus f constructs the target instance and g converts any of its solutions into a solution of the original instance. For reductions between total problems, the implication holds for every x .

The decoder may use both the original instance and the target witness. It must work for every target solution; it cannot rely on a solver returning a specially chosen one. This matters when a graph has many endpoints or a game has many equilibria.

Totality also explains why the usual SAT decision reduction is not an immediate model of hardness here. A total solver always returns something, including on instances constructed from unsatisfiable formulas. A reduction intended to decide SAT would need to specify how to interpret that answer.

Remark L18.5 (A useful NP versus coNP argument). Suppose a polynomial construction maps a formula F to an instance of a total polynomially verifiable search problem, and a polynomial decoder labels every valid witness with the correct satisfiable/unsatisfiable answer for F . Then unsatisfiability has an NP certificate: provide a valid target witness that the decoder labels unsatisfiable. Totality supplies such a witness, and correctness prevents a false one. Hence such a reduction would imply $\text{NP} = \text{coNP}$. This statement concerns reductions that recover the decision answer, not just a search reduction required to work on satisfiable formulas.

L18.3 The End-of-Line problem

Consider a finite directed graph in which every vertex has at most one incoming edge and at most one outgoing edge. Its nontrivial components are directed paths or cycles. If one vertex has unequal in-degree and out-degree, some other vertex must also be unbalanced, because the sum of out-degree minus in-degree over all vertices is zero.

The graph we use is exponentially large but succinctly represented. Its vertices are the n -bit strings, and two Boolean circuits $P, S : \{0, 1\}^n \rightarrow \{0, 1\}^n$ propose predecessors and successors. Define an edge $v \rightarrow w$ precisely when

$$S(v) = w, \quad P(w) = v, \quad w \neq v.$$

Requiring agreement between the two circuits ensures in-degree and out-degree at most one. Self-loops are treated as absent edges.

Definition L18.6 (End-of-Line, with a total encoding). Given P, S , let 0 denote the all-zero vertex. Return:

- 0 if its in-degree equals its out-degree; or
- any vertex $v \neq 0$ whose in-degree differs from its out-degree.

This convention handles every circuit pair. On the usual valid instances, 0 is a specified source with one outgoing edge and no incoming edge, and the task is to find another endpoint. An isolated vertex has both degrees zero and is not an endpoint.

Theorem L18.7. End-of-Line is a total polynomially verifiable search problem.

Proof. If 0 is balanced, output it. Otherwise the directed degree-sum identity implies another unbalanced vertex. A witness needs only n bits. To check the outgoing edge at v , test $S(v) \neq v$ and $P(S(v)) = v$; to check the incoming edge, test $P(v) \neq v$ and $S(P(v)) = v$. These require only a constant number of circuit evaluations. $\square$

Example L18.8 (Which endpoint may a solver return?). On eight vertices, consider paths $0 \rightarrow 1 \rightarrow 3$ and $4 \rightarrow 6$, a cycle $2 \rightarrow 5 \rightarrow 2$, and isolated vertex 7. Set the missing predecessor of a source and missing successor of a sink equal to the vertex itself. The valid outputs are 3, 4, 6. The solver need not return the endpoint of the path starting at 0. Neither a cycle vertex nor isolated vertex 7 is valid.

Following the path from 0 will eventually find an endpoint, but can require exponentially many steps. The input contains circuits of polynomial size, not an explicit list of all 2^n vertices. The gap between verification and search is therefore compatible with a very simple graph-theoretic existence proof.

L18.4 PPAD and other total-search classes

Definition L18.9 (PPAD). PPAD is the class of total polynomially verifiable search problems that admit a polynomial-time search reduction to End-of-Line. A problem is PPAD-hard if every problem in PPAD reduces to it, and PPAD-complete if it is both PPAD-hard and in PPAD.

End-of-Line is complete by definition and transitivity of reductions. PPAD is a subclass of TFNP: composing a reduction with a locally verifiable endpoint certificate gives the requisite search relation, with the standard witness encoding that includes the target endpoint when necessary.

Other classes organize total search by different existence principles. PPA uses parity of odd-degree vertices in undirected graphs; PPP uses a pigeonhole principle; PLS uses the existence of a local optimum in a finite search space with efficiently computable improving moves. These names identify formal reduction classes. A proof that merely mentions parity or a potential function still needs an efficient encoding and a reduction before it establishes membership.

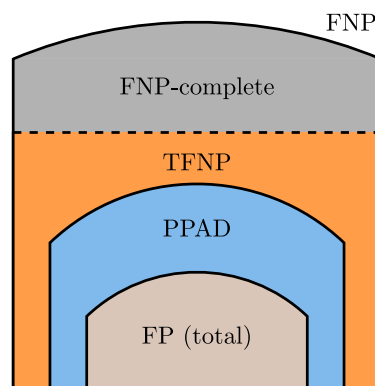


Figure L18.1: Schematic complexity-class inclusions, with the total-search region labeled explicitly. Nesting denotes inclusions; the displayed strict separations and placement of FNP-complete problems outside TFNP are conjectural, not proved by this diagram. FP here denotes polynomial-time solvable total search problems.

L18.5 Encoding a PPAD reduction

Lecture 2, “Brouwer and Sperner” established the geometric ingredients: the directed Sperner graph and the conversion from a trichromatic cell to an approximate fixed point. We now use those results to explain what a polynomial-time reduction must actually construct [Pap94].

■ **Short cell encodings.** Consider the two-dimensional Sperner grid (Section L2.1) with 2^b subdivisions per coordinate. Its interior colors are given by a Boolean circuit C on the binary coordinates; the standard boundary colors are imposed by a fixed rule. Thus validity of the boundary does not require checking exponentially many outputs of C . Splitting each grid square along a fixed diagonal produces 2^{2b+1} triangles, but a triangle is specified by

only its two square indices and one bit selecting the half-square. The [boundary padding \(Section L2.2\)](#) adds only a constant number of bits to this $O(b)$ -bit encoding.

■ **Local predecessor and successor circuits.** Given a cell encoding, compute its three corners, evaluate their colors, and apply the [orientation rule for the Sperner graph \(Section L2.2.1\)](#) to identify its incoming and outgoing neighbors. This uses a constant number of calls to C and arithmetic on $O(b)$ -bit coordinates. These local procedures therefore give polynomial-size circuits P, S ; constructing them never requires listing the grid. A missing neighbor is represented by the cell itself. Relabel the known boundary source as the all-zero vertex, and make unused bit strings isolated vertices by setting $P(v) = S(v) = v$.

■ **A decoder for every endpoint.** By these Sperner-graph properties, every unbalanced vertex other than the designated boundary source is a trichromatic cell. Decoding its coordinates takes polynomial time. Consequently, *any* valid End-of-Line answer solves the succinct Sperner instance, including an endpoint on a different path from the one that starts at zero. This is the search-reduction requirement from Section L18.2.

The reduction has polynomial cost in the circuit description and b , even though the represented graph has exponentially many vertices. End-of-Line is asked to supply an endpoint; the reduction does not find one by tracing a potentially exponential path. This distinction is what makes the Sperner existence argument a PPAD membership argument.

For a fixed-point application, one must additionally derive a suitable polynomially bounded precision b and an efficiently computable coloring from the finite description of the function and the requested accuracy. The [quantitative approximation bound \(Theorem L2.4\)](#) supplies this accuracy control; continuity alone does not supply these computational guarantees [EY10]. The two-dimensional example explains the encoding step. Applications in variable dimension require the corresponding higher-dimensional construction.

L18.6 The connection to Nash computation

Membership reduces approximate Nash to End-of-Line via a fixed-point construction. Hardness reduces End-of-Line to a game whose every sufficiently accurate equilibrium decodes to a valid endpoint.

Lecture 19, “PPAD-hardness of Nash equilibrium” uses action probabilities as circuit variables and payoff incentives to enforce gate relations [CDT09; DGP09]. The gates, payoff normalization, and approximation tolerance must be specified. PPAD-completeness concerns worst-case instances; a long path-following algorithm alone proves no lower bound on all algorithms.

Bibliography

- [Pap94] C. H. Papadimitriou, “On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence,” *Journal of Computer and System Sciences*, vol. 48, no. 3, pp. 498–532, 1994, doi: 10.1016/S0022-0000(05)80063-7.

- [EY10] K. Etessami and M. Yannakakis, “On the Complexity of Nash Equilibria and Other Fixed Points,” *SIAM Journal on Computing*, vol. 39, no. 6, pp. 2531–2597, 2010, doi: 10.1137/080720826.
- [DGP09] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, “The Complexity of Computing a Nash Equilibrium,” *SIAM Journal on Computing*, vol. 39, no. 1, pp. 195–259, 2009, doi: 10.1137/070699652.
- [CDT09] X. Chen, X. Deng, and S.-H. Teng, “Settling the complexity of computing two-player Nash equilibria,” *Journal of the ACM (JACM)*, vol. 56, no. 3, pp. 1–57, 2009.