JOURNAL OF COMPUTER AND SYSTEM SCIENCES **48**, 498–532 (1994)

# On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence

CHRISTOS H. PAPADIMITRIOU*

*Department of Computer Science and Engineering, University of California at San Diego*

Received March 25, 1991; revised June 9, 1992

We define several new complexity classes of search problems, "between" the classes FP and FNP. These new classes are contained, along with factoring, and the class PLS, in the class TFNP of search problems in FNP that always have a witness. A problem in each of these new classes is defined in terms of an implicitly given, exponentially large graph. The existence of the solution sought is established via a simple graph-theoretic argument with an inefficiently constructive proof; for example, PLS can be thought of as corresponding to the lemma "every dag has a sink." The new classes are based on lemmata such as "every graph has an even number of odd-degree nodes." They contain several important problems for which no polynomial time algorithm is presently known, including the computational versions of Sperner's lemma, Brouwer's fixpoint theorem, Chévalley's theorem, and the Borsuk–Ulam theorem, the linear complementarity problem for P-matrices, finding a mixed equilibrium in a non-zero sum game, finding a second Hamilton circuit in a Hamiltonian cubic graph, a second Hamiltonian decomposition in a quartic graph, and others. Some of these problems are shown to be complete. © 1994 Academic Press, Inc.

## 1. INTRODUCTION

The classes FNP and FP of search problems (problems in which an output more elaborate than "yes" or "no" is sought) are traditionally studied in terms of their surrogates NP and P. There are certain aspects of the issue, however, that cannot be easily captured by recognition problems. Consider for example the class TFNP defined in [MP]. TFNP (for *total* multivalued functions with polynomial-time verification) contains all search problems for which a witness always exists, independently of the input. Obviously, $\mathrm{FP} \subseteq \mathrm{TFNP} \subseteq \mathrm{FNP}$, and no proof of a proper inclusion is in sight. Is $\mathrm{TFNP} = \mathrm{FP}$? Is it true, that is, that it is easy to find a solution when you know that there is always one? A positive answer would imply, among other things, that $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$, that there is some kind of polynomial pivoting rule for simplex, and that there are polynomial-time algorithms for factoring, as well as for finding Brouwer fixpoints (despite the lower bounds in [HV]). Naturally, a negative answer would have even more important consequences.

* Research supported by the ESPRIT Basic Research Action No. 3075 ALCOM and by the NSF.

Even a qualified negative answer, such as the existence of an FNP-complete problem in TFNP, would imply that NP = coNP.

As with any conundrum in complexity, it would be nice to isolate problems that are TFNP-complete and thus capture this interesting computational phenomenon. However, it appears unlikely that such problems exist. The reason is that, along with NP $\cap$ coNP, RP, ZPP, BPP, and so many other complexity classes, TFBP is a *semantic* class.[1] By this informal notion we mean that a syntactic object (in our case, a nondeterministic Turing machine with output) defines a search problem in TFNP iff it satisfies a property quantified over all inputs (in the case of TFNP, the property states that the machine has at least one conclusive computation of all inputs). Such properties (like polynomial-time termination) can be handled only if they are properties of single computations. Semantic classes seem to have no complete problems. We are thus led to the following question: *Are there important, syntactically definable, subclasses of TFNP*? ZPP does not qualify, as it is a semantic class.

Every natural member of a semantic class is equipped with a mathematical proof that it belongs to that class. One idea is to group together problems in TFNP in terms of "proof styles." Factoring and discrete logarithms have too specialized, algebraic proofs (but see the open problems section). Another proof style, employed in the case of linear programming before Khachian's algorithm, and in the case of other combinatorial optimization problems, is *duality* (essentially, the ability to recognize optimality). Obviously, all optimization problems with duality are in TFNP, but there has been little success in formalizing duality in a syntactic way. Yet another proof style may be the "probabilistic method in combinatorics" [Sp], usually implying not only existence, but *abundance*, and thus membership in RP (see Section 5 for a brief discussion of an exception, the *local lemma*).

In [JPY] we defined a broad and natural subclass of TFNP, namely PLS (for polynomial local search). A problem $A$ in PLS is defined in terms of two polynomial algorithms $N$ and $c$. For each input $x$, $S(x) = \Sigma^{p(|x|)}$ is the set of all solutions (nodes of the dag alluded to above), where $p$ is a polynomial. The two polynomial algorithms compute, for each input $x$ and node $s \in S(x)$, the cost $c(x, s)$ and the (polynomially bounded in cardinality) set of neighbors $N(x, s)$. We wish to find a solution such that no neighbor has better cost. Thus, totality for functions in PLS is established by invoking the following "lemma:"

*Every finite directed acyclic graph has a sink.*

The dag for invoking the lemma is the graph whose adjacency lists are the $N(x, s)$, with arcs leading to nodes with no better $c$ omitted. In other words, we can view $N$ and $c$ as an *implicit syntactic way for specifying an exponentially large dag*. Class PLS contains a host of problems that are not known to be in FP (the difficulty is, of course, that the dag may have exponential depth). Several important problems

[1] Stuart Kurtz proposed to me the term *non-categorical* for the same thing.

are now known to be PLS-complete, including finding a local optimum in the Lin–Kernighan heuristic for the TSP and finding a stable configuration in Hopfield neural nets [JPY, PSY, Kr].

Are there other important examples of "unifying proof styles" of totality, defining new syntactic subclasses of TFNP? *In this paper we identify several such classes, and a host of natural, important problems contained in them; some of them are complete.* As usual, these classes first manifested themselves through a number of important computational problems which refused to be categorized in terms of existing complexity classes. Perhaps the most important among them is the computational version of *Brouwer's fixpoint theorem,* stating that any continuous function $f$ from the $d$-dimensional simplex to itself has a fixpoint. Computing Brouwer fixpoints (within accuracy $\varepsilon$) is a central computational problem in optimization and mathematical economics (see [SS] for a detailed discussion). We call this problem BROUWER (see Section 2 for a careful definition.) Ideally, we would like an algorithm for BROUWER that is polynomial in $d$ and $-\log \varepsilon$. Unfortunately, it was shown in [HV, HPV] that any algorithm *that treats $f$ as an oracle* must be in the worst case exponential in *both parameters.* The assumption in this result is rather unsatisfying (despite the fact that every one of the many algorithms that have been proposed for this important problem is indeed an oracle algorithm). Naturally, in the absence of this assumption the problem is in TFNP; thus no exponential lower bound is forthcoming. In this paper we derive convincing evidence of a different sort for the difficulty of Brouwer's problem by showing that BROUWER is complete in a rich new complexity class.

Like PLS, each of our new complexity classes can be seen as based on a graph-theoretic "lemma." Perhaps the most basic one is the *parity argument*:

*Any finite graph has an even number of odd-degree nodes.*

There is an interesting problem in graph theory that evokes the parity argument, namely Smith's theorem [Th]. Any graph with odd degrees has an even number of Hamilton cycles through edge $xy$. The proof constructs a graph $\Gamma$ whose nodes are all Hamilton paths starting from node $x$ and not continuing with node $y$. There is an edge between two paths iff they differ in only one edge (that is, one Hamilton path is a "rotation" of the other). It is easy to see that, since all degrees of $G$ are odd, the odd-degree nodes of $\Gamma$ are precisely the Hamilton paths starting from $x$ and ending at $y$—and these can be extended to a Hamilton cycle through $xy$. The computational problem SMITH is this: Given a graph $G$ with odd degrees, and a Hamilton cycle, *find another one.* It is in TFNP, and it is not known to be in FP. It is a prime specimen of our class PPA (for polynomial parity argument), defined in Section 3.

To feel the difficulty of turning an existence proof based on the parity argument to an efficient algorithm, imagine the following situation: Everybody remembers all games of chess they have played in their life. You have played an odd number of games, and you must find a fellow odd player (known to exist by the parity argument). Depth-first search can solve this problem, but it is too memory-consuming. Here is a solution: We require that each player has paired up his/her past games so that game $2i-1$ is the "mate" of game $2i$. The algorithm is this: Ask your last opponent if he is odd; if so, you are done. If not, you ask the address of his playmate in the game that is the mate, in his game history, of the game with you, and visit her. If she is odd you are done, but otherwise you ask for the address of her opponent in the game that is the mate (in her history) of her game with the previous player. And so on. You may come back to yourself many times (in which case you disregard your own parity), but the algorithm is guaranteed to terminate at another odd player. Alas, this algorithm may take time proportional to the number of all games of chess ever played!

Note that the "chessplayer algorithm" above must converge because of a trivial graph-theoretic fact, the "even leaves argument," which is in fact a special case of the parity argument:

*All graphs of degree two or less have an even number of leaves.*

Therefore, the path started by the algorithm must end somewhere. To illustrate the even leaves argument, imagine the special case of SMITH in which the graph is cubic (see Fig. 1). It is easy to find a second Hamilton cycle by the following (alas, exponential in the worst case!) algorithm: Delete an edge of the given cycle, fix an endpoint of the resulting path, and start "rotating" from the other endpoint. Since the graph is cubic, rotations are unique. There is no danger of "cycling" (repeating Hamilton paths), because cycling would mean that rotations are not unique. And there is no way for the process to end, other than in another Hamilton cycle through the deleted edge (see Fig. 1).

It appears that the even leaves argument is much more specialized than the more general parity argument. As it turns out, the "chessplayer algorithm" above can be formalized to show that the class based on the even leaves argument coincides with that based on the general parity argument (Theorem 1).

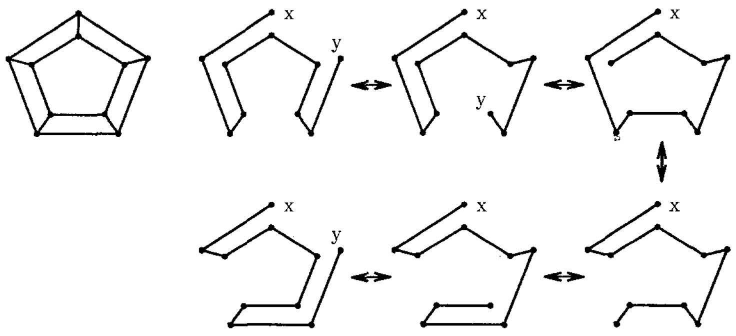

FIG. 1. Smiths theorem in the case of a cubic graph.

Sperner's lemma states that *any admissible coloring of any triangulation of the unit triangle as a trichromatic triangle* (in fact, by the parity argument, an odd number of them). Consider a triangulation of the unit triangle 012, say the standard $n \times n$ triangulation (Fig. 2, ignore for the moment the long triangles to the left). A coloring of all vertices with colors 0, 1, 2 is admissible if each vertex of the big triangle obtains its own name, and no vertex on edge $ij$ of the original triangle contains color $3-i-j$. A trichromatic triangle is found by extending the triangulation as shown in Fig. 2. Note an external 01 edge has been added. In fact, it is now the only external 01 edge. If we cross it, we end up in a triangle that is bound to have colors 0 and 1. So, it has another 01 edge, which we cross. The process cannot exit the boundary (that was the only 01 edge) and cannot fold upon itself (a triangle cannot have three 01 edges). It must end in a trichromatic triangle.

Q.E.D.

Note a subtle difference from the application of the pure even-leaf argument in the cubic case of Smith's theorem: In the present case we do have a sense of "progress" on the path from the standard starting position to the tricromatic triangle, suggested by the orientation of the bichromatic triangle in hand. Edges colored 0–1 are traversed in the direction that leaves the color 0 to the right. As a result, the implicit graph here is directed, but any leaf (a sink or a source other than the original one) is sought. The class of such problems is called PPAD (for "polynomial parity argument in a directed graph"). In contrast, in Smith's problem and other problems in PPA, if we start in the middle of a path we have no clue which of the two directions is the right one (see Fig. 1). This is the major difference between PMPA and PPAD. We do not know whether PPA = PPAD. As a general rule, problems in PPA that have a topological-geometric flavor seem to crowd into PPAD, whereas those of a more generic combinatorial or algebraic nature do not seem to be in PPAD.

The simple proof of Sperner's theorem explained above can be used as the basis for a proof of Brouwer's fixpoint theorem (this proof is sketched in Section 3). It is therefore not surprising that the computational problems BROUWER and SPERNER are closely related.

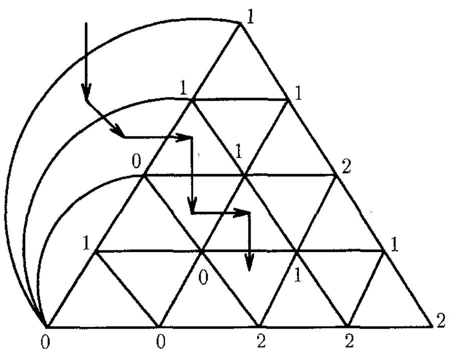

FIG. 2. Sperner's lemma in two dimensions.

There are many more problems that we consider in this paper; many of them were first identified in our joint work with Nimrod Megiddo [MP]. One of them, NASH, asks for a mixed equilibrium in a bimatrix game. This is arguably one of the few most important problems for which no polynomial algorithm is known (and no proof of NP-completeness seems possible). Another, SECOND HAMILTON DECOMPOSITION, asks for a second way to decompose a graph into two disjoint Hamilton cycles (its existence follows from a complicated parity argument, see [Th] and Theorem 12 below). A third, related problem is ANOTHER HAMILTON PATH: Given a Hamilton path in a graph (directed or not), find either another one or one in the complement (the total number of Hamilton paths in a graph and its complement is even!).

We also discuss certain problems related to Brouwer's fixpoint theorem. Problem KAKUTANI, a generalization of BROUWER, asks for a fixpoint of an *upper semi-continuous correspondence*, roughly, a continuous mapping from the unit triangle to convex regions of the unit triangle. We also introduce two important computational problems from mathematical economics, namely, computing equilibrium prices in certain appropriate economic models that guarantee the existence of equilibria. Existence is proved by Kakutani's theorem, which is in turn based on Brouwer's, which uses Sperner's lemma, and so on, down to the parity argument. We also introduce perhaps the major open algorithmic question in an important subfield of optimization: Our problem P-LCP defined in Section 2 asks for the solution of a linear complementarity problem or a negative minor of the matrix; one of the two must exist. Finally, we show that a classical result in number theory, Chévalley's theorem, has the parity argument at its root and the corresponding computational problem is thus in PPA. In fact, so are the computational problems associated with several interesting graph-theoretic applications of Chévalley's theorem. For none of these applications (one due to [AFK] and one new in this paper) is there a known polynomial-time algorithm.

In Section 2 we introduce the basic complexity classes discussed in this paper and show some interesting inclusions and collapses. We also point out a rather unexpected fact: If we insist that the algorithm return not any other leaf of the graph, but the particular leaf at the other end of the path (recall the discussion of Smith's and Sperner's theorems above) the problems become much harder: The resulting class is PSPACE! In Section 3 we introduce the various problems and show why they are in the classes discussed. In Section 4 we prove that BROUWER is PPAD-complete and so are several other problems. Finally, in Section 5 we introduce and briefly discuss certain other interesting classes of problems, based on "inefficiently constructive existence proofs" of different varieties. These new sub-classes are related to the probabilistic method [Sp], the local lemma [Sp], and the pigeonhole principle. We also point out some of the many problems left open by our work.

## 2. Complexity Classes

Let $R \subseteq \Sigma^* \times \Sigma^*$ be a polynomial-time computable, polynomially balanced relation (that is, $\{(x, y) \in R\} \in P$, and $(x, y) \in R$ implies $|y| \leqslant |x|^k$ for some integer $k$). The *search problem* associated with $R$ is this: Given input $x \in \Sigma^*$, return a $y \in \Sigma^*$ such that $(x, y) \in R$, if such a $y$ exists, and return the string "no" otherwise. The class of all such search problems is called FNP. We denote by FP the subclass of FNP containing all those search problems that can be solved in polynomial time. TFNP is the subclass of FNP containing all problems such that for all $x$ there is a $y$ with $R(x, y)$. It is clear that FP $\subseteq$ TFNP $\subseteq$ FNP, and it is open whether these inclusions are strict.

A problem in TFNP can be described by a polynomial-time machine that decides $R$. However, given such a description of a problem in TFNP, there is no way to tell whether $R$ is indeed total (the problem is obviously undecidable). A similar problem already exists in the case of FNP, since it is undecidable whether a given machine halts after polynomially many steps. The problem here, however, is much less severe and can be easily taken care of by "standardization," for example, by restricting our machines to incorporate a polynomial clock. In other words, there is a recursive enumeration of the problems in FNP, whereas such an enumeration is not known to exist for TFNP. The difficulty is quite familiar: It is already present in the language classes RP, ZPP, BPP, NP $\cap$ coNP, etc. Such classes, whose definition is given in terms of a non-recursive enumeration of machines, can be informally called *semantic* (all others are thus *syntactic*). It is well known that semantic classes tend not to have complete problems [Si].

The definition of our classes is similar in spirit with that of PLS: A problem in PPA is described in terms of an algorithm which implicitly defines an exponentially large graph. In the case of PLS, the graph was a partial order, defined by a directed graph and a cost function. In our case, the graph is cyclic, sometimes directed, sometimes undirected, usually (but not always) of bounded degree.

We start by defining PPA. A problem A in PPA is defined in terms of a polynomial-time deterministic Ruring machine $M$. Let $x$ be an input for $A$. The *configuration space* $C(x)$ is $\Sigma^{[p(|x|)]}$, the set of all strings of length at most $p(|x|)$, where $p$ is a polynomial. Given a configuration $c \in C(x)$, $M$ outputs in time $O(p(n))$ a set $M(x, c)$ of at most two configurations. Note that we do not require that $c' \in M(x, c)$ iff $c \in M(x, c')$; symmetry is guaranteed syntactically by the following definition: We say that two configurations $c, c'$ are *neighbors*, written $[c, c'] \in G(x)$, if $c \in M(x, c')$ and $c' \in M(x, c)$. Obviously, $G(x)$ is a symmetric graph of degree at most two. In applications, we expect $M(x, c)$ to be empty often, thereby rendering many configurations $c$ irrelevant. $M$ is such that $M(x, 0 \cdots 0) = \{1 \cdots 1\}$, and $0 \cdots 0 \in M(x, 1 \cdots 1)$, so that $0 \cdots 0$ is always a leaf (the *standard leaf*). This is not a semantic restriction, since $M$ can be standardized so that it is syntactically guaranteed to behave in the required way. Problem A associated with $M$ is the following search problem: "Given $x$, find a leaf of $G(x)$ other than $0 \cdots 0$." PPA is the class of all problems A defined as above.

PPA, as defined above, embodies the "even leaves argument," and not the full "parity argument." It may thus appear that it fails to include problems such as "Given an odd-degree graph and a Hamilton cycle, find another" (the generalization of Fig. 1 to arbitrary odd-degree graphs). Suppose that we define PPA$'$ to be the class with the same definition, only that $|M(x, c)|$ is bounded by a polynomial in $|x|$, as opposed to two. That is, we allow the degree of $G(x)$ to be polynomially large. We are seeking any odd-degree node.

THEOREM 1 (The Chessplayer Algorithm). *PPA$'$ = PPA.*

*Proof.* It is obvious that PPA $\subseteq$ PPA$'$. We shall show that, given any problem A in PPA$'$, presented by a machine $M$, we can define an equivalent problem in PPA. For each node $c$ of $G(x)$ we can compute its neighborhood $N(c)$. If $|N(c)| = k \geqslant 1$, we create new nodes $(c, i)$, $i = 1, ..., \lceil k/2 \rceil$. To compute the connectivity of the new graph, the two neighbors of $(c, i)$ are the two nodes of the form $(c', j)$, where the (lexicographically) $2i$th or $(2i+1)$th edge out of $c$ is $[c, c']$, which is also the $2j$th or $(2j+1)$th edge out of $c'$. It is clear that the nonstandard leaves of the new graph coincide with the nonstandard odd-degree nodes of the former, and that the neighbors of each node of the new graph can be computed in polynomial time. ∎

We can even extend the definition of PPA to cases in which the degree of the graph is not bounded by a polynomial. Suppose that we have a polynomial algorithm for deciding, given two nodes $c, c' \in C(x)$, whether $[c, c'] \in G(x)$. We call this the *edge recognition algorithm.* Naturally, if a graph is presented this way, its degree may very well be exponential. However, assume that we are also given a polynomial algorithm that computes a *pairing function* $\phi$ between the edges out of each node. That is, given $c, c'$, where $c$ is an even-degree node and $[c, c'] \in G(x)$, $\phi(c, c') = c''$ with $c' \neq c''$ and $[c, c''] \in G(x)$, and also $\phi(c, c'') = c'$. If $c$ has odd degree, then for exactly one $c'$ $\phi(c, c') = c'$. We are again asking for any nonstandard odd-degree node.

COROLLARY. *Any problem defined in terms of an edge recognition algorithm and a pairing function is in PPA.*

*Proof.* The nodes of the new graph are the edges of the original, unbounded-degree one, and the only adjacencies are these: edges $[c, c']$ and $[c, c'']$ (considered as unordered pairs) are adjacent if and only if $\phi(c, c') = c''$. ∎

Note that there is no syntactic way to tell whether an algorithm indeed computes a pairing function; however, this does not interfere with the validity of the corollary above. If $\phi$ fails to be a pairing function, then simply there will be a nonstandard leaf in the constructed graph that does not correspond to an odd-degree node of $G(x)$, and the problem in PPA will fail to be the intended one.

To define PPAD (the parity argument for directed graphs), we modify the definition of PPA so that $M(x, c)$ is an ordered pair of configurations. The graph $G(x)$ is now directed: $(c, c') \in G(x)$ iff $c'$ is the second component of $M(x, c)$, and $c$ is the first component of $M(x, c')$. We are asking for any node (other than $0 \cdots 0$) with indegree + outdegree $= 1$. In other words, a correct output is any source or sink of the directed graph other than the standard source.

In the context of search problems, a *reduction* from A to B is defined as a pair of polynomially computable functions $f$ and $g$ such that, for any input $x$ of problem A, and for any legal output of problem B on input $f(x)$, $g(x, y)$ is a legal output of problem A on input $x$. As defined so far, PPA and PPAD are *not* closed under reductions, if $g$ is not one-to-one on outputs. We must therefore modify our definitions so that PPA is the closure under reductions of the class of search problems defined so far, and similarly for PPAD.

PROPOSITION 1. *$FP \subseteq PPAD \subseteq PPA \subseteq FNP$.*

As Theorem 1 suggests, these classes are quite robust under certain variants of their definition. Modifying the definition in other directions can have a devastating effect. Suppose, for example, that in the definition of PPA we insist that the output be not *any* other leaf, bu the *particular* other leaf connected to $0 \cdots 0$. This is not unnatural, since that leaf is the only one guaranteed to exist. Call this class PPA″, and the corresponding directed class PPAD″.

THEOREM 2. *PPA″ = PPAD″ = FPSPACE.*

*Sketch.* We shall show that PPA″ $\supseteq$ FPSPACE, the other inclusions being immediate. Consider any problem in FPSPACE. It follows from a result by Bennett [Be] that this problem can be solved by a polynomial-space bounded Turing machine $T$ that is *reversible*; that is, each configuration is the successor and predecessor of at most one other configuration. We can thus define the following problem in PPA″: For each configuration $c$ of $T$, $M(x, c)$ returns the (at most) two configurations that are the predecessor and/or successor of $c$ on input $x$. The standard leaf is connected to the initial configuration on input $x$. Thus, the nonstandard leaf on the same component as the standard one is precisely the halted configuration, containing the desired output. ▮

Another interesting observation (due to Steve Bloch and Sam Buss) is this: Imagine that PPA‴ is the variant in which $|M(x, c)| \leqslant 3$; that is, the configuration graph has degree at most three. Naturally, a second leaf is now not guaranteed to exist. It can be shown that PPA‴ = FNP.

## 3. THE PROBLEMS

In defining some of the computational problems that we study, we encounter a complication reminiscent of the "semanticity" issue in classes, but of course, one set-theoretic level below. At the problem level, the analog of a semantic class is a "promise problem," that is, one whose input domain is restricted in some way. Since TFNP trivially contains a host of uninteresting promise problems, we are eager to avoid considering such problems. As a result, whenever our inputs need to be restricted in some way (see the problems BROUWER and SPERNER below), the input will be supplied with a syntactic proof that it satisfies the restriction. Even when the input is an algorithm, this is usually easy to do by standardizing the algorithm so that its output can be checked and, if illegal, replaced by a standard output.

## 3.1. *Problems in PPAD*

SPERNER. We have sketched Sperner's theorem for two dimensions, and its proof, in the Introduction (Fig. 2). The corresponding computational problem is 2D SPERNER: Given an integer $n$ (in binary) and an algorithm $M$ for assigning to each point $p=(i_1, i_2, i_3)$ with $i_1, i_2, i_3 \geqslant 0$ and $i_1+i_2+i_3=n$ a color $M(p) \in \{0, 1, 2\}$, such that $i_j=0$ implies $f(p) \neq j$; find three points $p$, $p'p''$ such that their pairwise distances are one, and $\{f(p), f(p'), f(p'')\} = \{0, 1, 2\}$. (The three points define a trichromatic triangle). To guarantee that $M$ indeed produces an admissible color, we equip it with a subroutine which, before halting, examines the output and, if it is not legal, outputs something standard.

Sperner's theorem and its proof above can be generalized to three and more dimensions: Consider the $d$-dimensional simplex with vertices $0, 1, ..., d$, and a given *simplicization* (the $d$-dimensional equivalent of the triangulation) of the simplex. Suppose that all vertices of this simplicization have been assigned colors $0, 1, ..., d$ such that, if a vertex $v$ is in the exterior of the simplex, on face $i_0, ..., i_k$ (recall that a $k$-dimensional face of the simplex is determined by its $k+1$ vertices), then its color is among $i_0, ..., i_k$. Then Sperner's lemma guarantees that there is a *panchromatic* simplex in the simplicization (one that contains all colors). The proof proceeds by an induction on the dimension, with the panchromatic simplex in a $(d-1)$-dimensional face serving as the starting point of a path that must lead to a panchromatic simplex (just as a bichromatic edge started us in two dimensions, recall Fig. 2).

However, although Sperner's theorem holds in higher dimensions, there is a difficulty in generalizing the *computational problem* 2D SPERNER to three and more dimensions. The reason is that *there is no easy standard simplicization of the tetrahedron*, no simple three-dimensional analog of the triangles in Fig. 2. For example, I believe that there is no way to divide a regular tetrahedron into eight identical regular tetrahedra (the reader is encouraged to try). There are several approaches we can take to handle this problem, each with different advantages and disadvantages:

(1) We could encode the simplicization in the machine that defines the coloring. This would make the problem unnecessarily complex (and, therefore, the completeness result in Theorem 14 unnecessarily weak). Syntactic guarantees

that the machine indeed provides a simplicization are possible, but would further complicate the problem and obscure the issue. We come back to this problem after Theorem 14.

(2) To divide the equilateral triangle into four equal triangles we first "cut off" the three triangles with parallel lines to the sides (Fig. 3); fortunately, the remainder also happens to be an equilateral triangle. If we try the same in the regular tetrahedron, we end up with a regular octahedron in the middle. We can now divide this octahedron into eight equal (not regular) tetrahedra with a vertex at the center of the octahedron. Thus we have subdivided the regular tetrahedron into 12 tetrahedra. To divide any tetrahedron into $12^k$ tetrahedra, repeat the above subdivision $k$ times. This results into an arbitrarily fine simplicization. The vertex coordinates are rationals with denominators of polynomial length in $k$, and the vertices can be encoded easily. Unfortunately, had we adopted this approach, the geometric and analytical details of our reduction in Theorem 14 would be extremely messy.

(3) Another idea is to embed a solid with an easy simplicization, such as a cube, into the tetrahedron. Simplicize the remaining space in some standard coarse way (see Fig. 4 for the idea in the two-dimensional case, leading to an obvious three-dimensional generalization). This version has certain advantages, but the following variant of the same idea is much cleaner.

(4) The simplest and most intuitive way is to state the computational problem related to Sperner is in terms of a *hypercube*. To explain and motivate it in two dimensions, suppose that we are given a square grid with a coloring, such that (see Fig. 5): (a) Vertex (0, 0) is colored 0, vertex (1, 0) is colored 1, and vertex (0, 1) is colored 2. (b) No point on the edge (0, 0)–(1, 0) is colored 2, and no point on the edge (0, 0)–(0, 1) is colored 1. (c) Points on the other two edges are not colored 0. Sperner's theorem implies that there is a little square with all three colors. Note that the square "simulates" a triangle, with its two sides

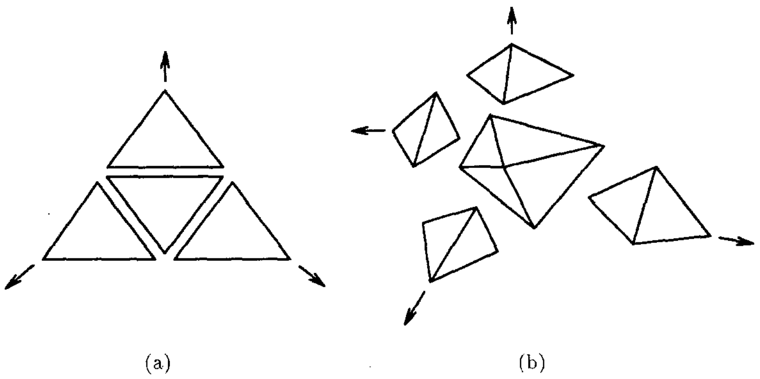

FIG. 3. Subdividing a simplex.

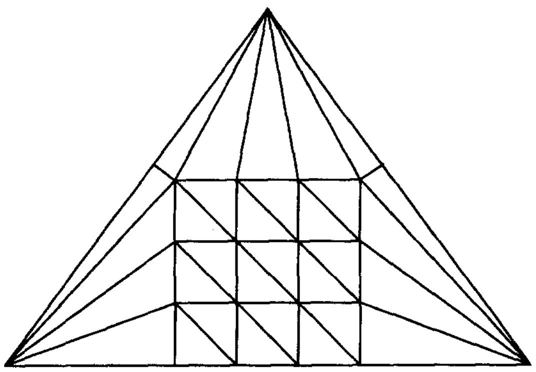

FIG. 4. Embedding a hypercube.

(0, 1)–(1, 1)–(1, 0) playing the role of the third side. The proof is immediate if we triangulate the little squares and then "pull" the vertices (0, 1) and (1, 0) diffeomorphically (as indicated in Fig. 5), so that a triangle is formed with vertices (0, 0), (0, 1), (1, 0).

We can now generalize this to three dimensions. Suppose that we are given the standard subdivision of the unit cube into $n^3$ "cubelets" (see Fig. 6). Each point $(x_1/n, x_2/n, x_3/n)$, $0 \leqslant x_1, x_2, x_3 \leqslant n$ (we call the vertices of the cubelets "points") is assigned a color among 0, 1, 2, 3, subject to the following conditions: (a) Vertex (0, 0, 0) obtains 0; also vertex (1, 0, 0) obtains 1, vertex (0, 1, 0) obtains 2, and vertex (0, 0, 1) obtains 3. Call these the *principal* vertices. (b) Any point on a face containing (0, 0, 0) cannot obtain the color of the principal vertex missing from it. (c) Any point on the three faces not containing (0, 0, 0) cannot obtain the color 0. Intuitively, we "simulate" the tetrahedron by a cube, where the three faces incident upon (0, 0, 0) simulate the three faces of the tetrahedron, and the other three faces

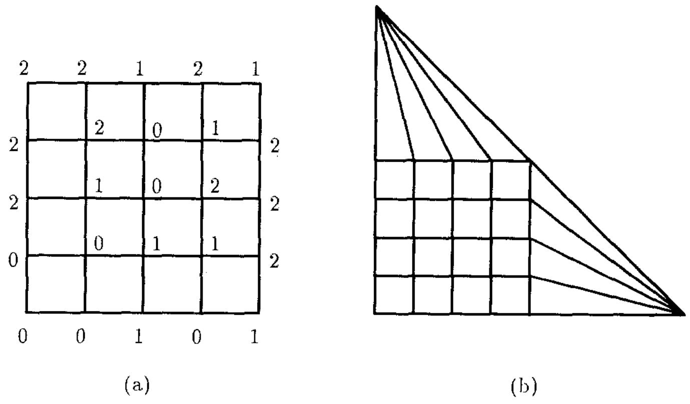

FIG. 5. Sperner's lemma in the square.

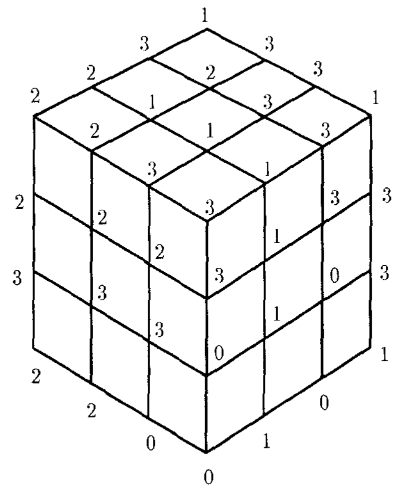

FIG. 6. Sperner's lemma in the cube.

of the cube simulate the fourth face of the tetrahedron. Sperner's Theorem implies that there is a cubelet with all four colors. The proof is by subdividing each cubelet into tetrahedra (such a subdivision of the cube into five tetrahedra is possible and well known) and transforming the cube into a tetrahedron by "pulling" the principal vertices outwards. The result follows immediately from Sperner's lemma in the tetrahedron.

We define the computational problem 3D SPERNER thus: Given an integer $n$ (in binary) and a polynomial-time algorithm computing for each point of the $n \times n \times n$ subdivision of the cube a legal color, find a tetrachromatic cubelet (one that has all four colors). We assume as before that the polynomial algorithm is such that the colors computed by it are legal. The generalization to more dimensions is immediate. This simple problem captures the intricacy of Sperner's theorem and avoids the problem alluded to above.

THEOREM 3. *For any $k \geqslant 2$, k-D SPERNER is in PPAD.*

*Proof.* In the two-dimensional case the nodes are the triangles in Fig. 2 (including the "long" triangles on the left), and there is an arc from one triangle to the other if they have colors 0 and 1, and they share a 0–1 edge. The arc is directed away from the simplex in which the colors on the common arc in the clockwise direction are 0–1. The standard leaf corresponds to the outermost 0–1 triangle. All other leaves are adjacent to trichromatic triangles.

For 3D SPERNER we generalize the same construction. We embed the cube into a tetrahedron and fix a simplicization. We complete this simplicization by adding edges from node 0 to all points on the 0–1–2 face. The vertices are either

0–1 triangles on that face or trichromatic triangles on that face or 0–1–2 simplices in the interior of the tetrahedron: 0–1 triangles are connected as before; 0–1 triangles are connected to adjacent 0–1–2 triangles and each of the latter, to the 0–1–2 simplex that contains it. Finally, two 0–1–2 simplices are connected if they share a 0–1–2 face and the tail is the one for which the common face is read 0–1–2 in the clockwise sense. We omit the generalization to more dimensions. ∎

Brouwer. Brouwer's theorem states that any continuous function $f$ from the unit simplex (or cube, or any convex compact body) to itself has a fixpoint, that is, a point $x$ such that $f(x)=x$. A simple proof is based on Sperner's lemma. Consider the $d$-dimensional regular simplex and the vector from vertex $i$ and perpendicular to the opposite facet; call it "vector $i$." Now, consider any simplicization of the simplex, and a coloring of the points in it with colors in $\{0, 1, ..., d\}$, such that if the color of $p$ is $i$, then vector $p-f(p)$ has a positive projection on the $i$th vector. It is easy to see that any such coloring of the points is acceptable. Hence, by Sperner's theorem, it has a panchromatic simplex. The center of this simplex is called $x_1$.

Consider now a sequence of finer and finer such triangulations and centers of panchromatic simplices $(x_1, x_2, ...)$. Pick a converging subsequence of that sequence. It is easy to see that its limit, call it $x^*$, must satisfy $f(x^*)=x^*$. End of proof!

In defining the computational problem associated with Brouwer's theorem, we must somehow represent a continuous function by a Turing machine. We know of no simple syntactic general way of doing this (basically, we know of no simple enumeration of the algorithms computing continuous functions). We define here a syntactic special case of the problem, which is sufficiently general in that the negative results of [HPV] apply (and the Corollary to Theorem 14 still holds). We are given an integer $n$ Turing machine $M$ which, given a point $x$ in the unit cube $C_d$ with coordinates multiples of $1/n$ returns in polynomial time a vector $\mu(x)$ such that $|\mu(x)| \leqslant 1/n^2$ ands $f(x)=x+\mu(x) \in C_d$ (these conditions can be syntactically enforced by checking the output, and returning 0 if they are violated). The function $f(x)$ thus defined can be extended to a piecewise linear map by computing the values within each $1/n$ cubelet by interpolating the values at the vertices, according to a standard simplicization of the cubelet. We are seeking a point $x$ such that $f(x)=x$.

Theorem 4. *BROUWER is in PPAD.*

*Proof.* The proof above reduces BROUWER to SPERNER. ∎

We next examine KAKUTANI, a computational problem inspired by a generalization of Brouwer's theorem. $f(x)$ is now a *subset* of $C_d$, say a simplex with vertices of the form $x+\mu_i(x)$ for $i=0, ..., d$ such that again $x+\mu_i(x) \in C_d$ for all $x$

and $i$. We are seeking a point $x$ such that $x \in f(x)$. The existence of such a point follows from a reduction to Brouwer's theorem. Therefore we have

COROLLARY. *KAKUTANI is in PPAD.*

EQUILIBRIUM. Brouwer's theorem is used in the proof of certain important theorems in mathematical economics (results in the so-called "Arrow–Debreu theory"). Accordingly, algorithms for problem BROUWER are at the core of computer models of large-scale economies. Here we shall only try to give the flavor of these important results.

A vector $x \in \mathfrak{R}^n$ stands for amounts (perhaps negative) of each of $n$ goods. A (normalized) *price vector* is a vector in $\Delta_n$ (the unit simplex). One application supposes that each price vector $p$ generates among producers and consumers a *total excess demand* $D(p) \in \mathfrak{R}^n$, where $D$ is a continuous function satisfying *Walras's Law* $p \cdot D(p) \leqslant 0$. An equilibrium in this situation is a price vector such that $D(\hat{p}) = 0$. It always exists, and this result is equivalent to Brouwer's theorem. The corresponding computational problem, called EXCHANGE EQUILIBRIUM, is defined as follows: We are given an $\varepsilon = 2^{-m}$, and a machine $D$ computing the demand for each price vector, up to $2m$ bits. We are asked to find a price vector such that $|D(\hat{p})| \leqslant \varepsilon$.

In a related but more detailed setting, economic agents are studied as individuals, not through their aggregate behavior. We have $m$ agents, each with a convex, upwards unbounded set $X_i \subset \mathfrak{R}^n$ of *acceptable* vectors of goods. A negative amount of a good means production, a positive amount consumption. Presumably, the acceptable vectors of a consumer will have nonnegative components everywhere except for the component *labor*; those of a factory will positive labor and raw material components, and a negative component for the products. Each agent has a *utility* function $u_i$ mapping $\mathfrak{R}^n$ to $\mathfrak{R}$ (a measure of the satisfaction he/she draws from each vector of commodities) and also an initial *endowment* $e_i \in \mathfrak{R}^n$, $e_i \geqslant 0$. We assume that $u_i$ is piecewise linear and convex, say.

Then prices are announced. Each agent will sell his/her endowment and obtain the best (under $u_i$) acceptable vector $\hat{x}_i$ he/she can afford ($\max u(\hat{x}_i)$ subject to $x_i \in X_i$, $p \cdot \hat{x}_i \leqslant p \cdot e_i$); at least one such $\hat{x}_i$ is assumed to exist, no matter what the prices are. A famous theorem by Arrow and Debreu (based on Kakutani's theorem) [AD] states that there is a price $\hat{p}$, the *equilibrium* price vector, such that the individual optimizing agents will end up clearing all markets: $\sum_{i=1}^{m} \hat{x}_i = \sum_{i=1}^{m} e_i$. We omit the details of the definition of the corresponding computational problem, called COMPETITIVE EQUILIBRIUM.

THEOREM 5. *COMPETITIVE EQUILIBRIUM and EXCHANGE EQUILIBRIUM are in PPA.*

*Sketch.* For EXCHANGE EQUILIBRIUM the result follows from the equivalence to BROUWER described in [Uz]. For COMPETITIVE EQUILIBRIUM, it follows from the corollary to Theorem 4, since the existence of a competitive equilibrium is proven by a simple application of Kakutani's theorem [AD]. ∎

Borsuk, Ulam, and Tucker. The Borsuk–Ulam theorem states that any continuous function $f: S^d \mapsto \Re^d$ (from the surface of the $(d+1)$-dimensional unit sphere to $\Re^d$) has equal values at some pair of antipodal points; that is, there is a point $x$ such that $f(x) = f(-x)$.

An immediate application of the Borsuk–Ulam theorem is the "ham sandwich problem:" Given $d+1$ "nice" bodies in $d+1$ dimensions, there is a hyperplane that cuts all of them in half by volume. To prove it, consider the mapping from $S^d$ to $\Re^d$ defined as follows: For each point $x \in S^d$ (that is, direction in $\Re^{d+1}$), let $f(x)$ be the vector of volumes cut from bodies 1, ..., $d$ by the hyperplane normal to $x$ that bisects body 0. The result then follows from the Borsuk–Ulam Theorem. Another interesting application of the Borsuk–Ulam theorem is the necklace problem [Al1]. It is shown that any necklace with $mnk$ beads, equally divided into $k$ colors, can be cut by $(m-1)k$ cuts into pieces that can be reassembled to $m$ smaller necklaces, each containing $n$ beads of each color (note that $(m-1)k$ is optimal: consider a necklace with contiguous colors). The proof is a direct application of the Bursuk–Ulam theorem, not unlike the one for the ham sandwich problem.

Perhaps the most natural proof of the Borsuk–Ulam theorem is based on a combinatorial result due to Tucker (very much the same way that Brouwer's theorem follows from Sperner's lemma). Tucker's lemma states the following: Consider the $d$-dimensional hypercube $-1 \leqslant x_i \leqslant 1$. Consider a subdivision of the hypercube into an even number of equal cubelets (so that the origin is a point). Consider now a coloring $f$ of all points by colors $\pm 1, ..., \pm d$ which is *antipodal preserving*, that is, for all points $x$ on the surface of the hypercube $f(-x) = -f(x)$. Then there is a cubelet that contains two opposite colors. For a proof of Tucker's lemma see Theorem 6 below.

To prove the Borsuk–Ulam theorem from Tucker's lemma, consider the function $f$ on the surface of the $(d+1)$-dimensional sphere. Consider now a hemisphere $\frac{1}{2} S_d$, and define on it the function $g$: $g(x) = f(x) - f(-x)$. Note that, at the "rim" of the hemisphere, $g(x) = -g(-x)$. Now, map the hemisphere continuously and symmetrically to the $d$-dimensional hypercube (say, by first projecting to the $d$-dimensional ball, and then mapping the ball "radially" to the hypercube). Finally, define the color of a point $p$ of a subdivision of the cube to be the index of the largest (in absolute value) coordinate of $g(p)$ times the sign of that coordinate; ties favor smaller coordinates, say. Precisely as in Brouwer's theorem, any sequence of subdivisions defines a sequence of centers of the cubelets with two opposite colors, which converges to a point that corresponds (inverting the mappings) to a point $x^*$ on the surface of the sphere satisfying $f(x^*) = f(-x^*)$. One can define the following computational problems:

*Borsuk–Ulam.* Given an integer $n$ and a Turing machine computing for each point $p = (x_1, ..., x_d)$ with $-n \leqslant x_i \leqslant n$ and $\max_i |x_i| = n$ (the surface of the $L_1$ sphere) a function $f(p)$ with $f(p) \leqslant 1/Kn$. Find an $x$ with $|f(x) - f(-x)| \leqslant 1/n^2$.

*Discrete ham sandwich.* Given $2n^2$ points in generic position in $n$ dimensions, separate them into $n$ groups with $2n$ points each, find a hyperplane dividing all groups in half. (Note that, unlike other geometric-topological problems in this section, this problem is in FP for any fixed dimension $d$.)

*Necklace splitting.* Given a cyclic sequence $a_1, a_2, ..., a_{mnk}$ of integers between 1 and $k$, such that each integer appears $mn$ times, find $(m-1)k$ integers $1 \leqslant c_1 \leqslant \cdots \leqslant c_m \leqslant mnk$ (the cuts) such that there is a way to partition the cyclic intervals $[c_i \cdots c_{i+1 \bmod (m-1)k}]$ into $m$ sets of intervals, each of which contains $n$ appearances of each integer.

*Tucker.* Given an integer $n$ and a Turing machine computing for each point $p=(x, y, z)$ with $-n \leqslant x, y, z \leqslant n$ a color $M(p) \in \{\pm 1, ..., \pm d\}$ such that for all $p$ with $\max\{|x|, |y|, |z|\} = nM(p) = -M(-p)$, find a two points $p, p'$ with $|p-p'|_\infty \leqslant 1$ and $M(p) = -M(p')$.

None of this problems is known to be in FP. However, we can show the following:

THEOREM 6. *TUCKER is in PPAD.*

*Proof.* The construction of $G(x)$ is based on the elegant proof of Tucker's lemma in [FT]. We consider a standard simplicization of the hypercube that refines the subdivision into cublets. The nodes of $G(x)$ are the *fully colored simplices* in this simplicization. We call a simplex (of any dimension including 0) fully colored if for any nonzero coordinate $i$ of its center, one of the colors $\pm i$ appears at some point in the simplex. Note that the zero-dimensional simplex at the origin (necessarily a part of the simplicization) is trivially fully colored; it is the standard source of $G(x)$. Two simplices are adjacent if one of the following circumstances occur: (a) One is a facet of the other, and the smaller one has the same set of colors as the larger; or (b) They are both on the surface of the hypercube, and antipodal. It is easy to see (see [SF] for a careful proof) that all degrees are all at most two, and any nonstandard leaf must be a simplex containing two opposite colors.

However, this only establishes that the problem is in PPA. To prove it is in PPAD we must define consistent directions of the arc of $G(x)$. This is possible, but requires the introduction of involved techniques, see [Fr]. ▌

COROLLARY. *BORSUK–ULAM, DISCRETE HAM SANDWICH, and NECKLAGE SPLITTING are in PPAD.*

LINEAR COMPLEMENTARITY. We are given an $n \times n$ matrix $A$ and an $n$-vector $b$. We are seeking vectors $x, y \geqslant 0$ such that $Ax+b=y$, and $x \cdot y = 0$. The latter constraint requires that, for each $i$, at least one of $x_i$, $y_i$ be zero (hence the term complementarity).

This problem can always be *attacked* (but not always solved) by *Lemke's algorithm* [LH, Le, CD]. Let us modify the equation to $Ax+b+\lambda \cdot \mathbf{1} = y$, where

$\lambda$ is a scalar. By taking $x=0$ and $\lambda$ appropriately large to make for the negative components of $b$, we can find a *basic* solution (at most $n$ nonzero among the $y_i$, $x_i$'s, and $\lambda$) of this equation with all $x$'s zero and at least one $y$ zero. Unfortunately, $\lambda$ will in general be nonzero as well, and so we have no solution. Thus, there is an $i$ for which $x_i=y_i=0$. If we choose one of these two variables and increase its value, the other positive variables will change linearly (this is very much like traditional simplex pivoting, except that we have only one pivoting choice). If they all increase, this means that the variable can be increased indefinitely, and we are at an *infinite ray*. We shall return to this case. Suppose then that at least one other variable decreases and eventually becomes zero. If it is one of the $x_j$, $y_j$'s, say $x_j$, we are back where we started: We can start to increase $y_j$ away from zero and continue. But if $\lambda$ ever becomes zero then we have solved the original problem.

But, unfortunately, this "algorithm" may indeed end up in a ray (actually, the linear complementarity problem is NP-complete). However, there are certain cases in which this cannot happen. For example, if all entries of $A$ are positive, or if $A$ is derived in a simple way from a bimatrix game (see the proof of Theorem 8 below), then the algorithm cannot end up in a ray. We shall concentrate on perhaps the most intriguing such special case, namely that of *P-matrices*. A matrix $A$ is a P-matrix if for all nonempty subsets $S$ of $\{1, ..., n\}$, $\det(A_S)>0$, where $A_S$ is $A$ restricted to the rows and columns in $S$. For symmetric matrices this coincides with the matrix being positive-definite and can be tested in polynomial time by standard techniques. For general matrices, however, it is open whether checking the P-property can be done in polynomial time. What is known is that if $A$ is a P-matrix, then the linear complementarity problem has a unique solution for any $b$, and Lemke's algorithm eventually converges to it [CD]. Let P-LCP be the following problem: "We are given $A$ and $b$, and we are asked to find either a solution $x$, $y$ to the linear complementarity problem, or a subset $S$ such that $\det(A_S)\leqslant 0$ (proof that $A$ is not a P-matrix)."

THEOREM 7. *P-LCP is in PPAD.*

*Sketch.* The construction of the directed graph $G(A, b)$, where $A, b$ is an instance of P-LCP and $A$ is $n\times n$, is outlined in [To]. The nodes are pairs of the form $(S, T)$, where $S$ and $T$ are subsets of $\{0, 1, 2, ..., n\}$, such that $S=T-\{a\}\cup\{b\}$ (as a special case, $a$ could equal $b$ in which case $S=T$). Intuitively, $S$ is the set of coordinates in the vector $(0, x_1, ..., x_n)$ that are zero, and $T$ is the set of coordinates in the vector $(\lambda, y_1, ..., y_n)$ that are not zero. Two such nodes are adjacent if they represent a pivot. The direction is determined by the signs of the subdeterminants of $A$ defined by the sets, as explained in [To]. The standard source is $(\{0\}, \{0\})$. All other sources and sinks are of the form $(S, S)$, with $S\neq\emptyset$. If $\det A_S>0$ then we have a solution of the LCP; otherwise we have a proof that $A$ is not a P-matrix. ▌

NASH. We are given two $m \times n$ matrices $A$ and $B$. Intuitively, $A_{ij}$ is the payoff of player I when I plays strategy $i$ and II plays strategy $j$; $b_{ij}$ is the payoff of player II. In general, $A + B \neq 0$ (the game is not *zero-sum*). A *Nash equilibrium* is a pair of strategies $i$ for I and $j$ for II, such that neither I nor II have an incentive to change strategy, that is, $a_{kj} \leqslant a_{ij}$ for all $k$, and $b_{ik} \leqslant b_{ij}$ for all $k$. It is easy to see that a *bimatrix game may have no equilibrium.*

The problem here is that the set of strategies is *not convex* (since it is discrete). For a convex space of strategies, Nash has shown that an equilibrium exists. There is an attractive may to make our situation convex: Accept as strategies any *probabilistic distribution* over the rows and columns. A (row) $m$-vector $x$ is a *mixed strategy of* I if $x \geqslant 0, x \cdot \mathbf{1} = 1$; similarly for a column vector $y$ for II. We say that a pair of such mixed strategies $x$ and $y$ are an *equilibrium* if $x'Ay \leqslant xAy$ for all mixed strategies $x'$, and $xBy' \leqslant xBy$ for all $y'$. Such an equilibrium always exists. Nash's proof used Brouwer's fixpoint theorem. Another proof uses the linear complementarity problem; in fact, this latter proof shows that the equilibrium will be rational with small coefficients (this is open for games with more than two players, where equilibria are also guaranteed to exist). The problem NASH is this: Given integer matrices $A$ and $B$, find a mixed strategy equilibrium. There is no known polynomial algorithm for this fundamental problem.

THEOREM 8. *NASH is in PPAD.*

*Proof.* Any solution to the linear complementarity problem with matrix $\left(\begin{smallmatrix} 0 & A \\ B^T & 0 \end{smallmatrix}\right)$ and vector $\mathbf{1}$ is a mixed equilibrium, and can be found by Lemke's algorithm, assuming (with no loss of generality) that $A$ and $B$ are positive [CD]. ▌

## 3.2. *Problems in PPA*

SMITH. We are given an odd-degree graph $G$ and a Hamilton cycle $C$ of $G$. We are asked to find a second Hamilton cycle of $G$. Existence proof is by the parity argument, as in the Introduction. This same proof yields the following result.

THEOREM 9. *SMITH is in PPA.*

A similar problem is based on the following result: Let $G$ be an undirected graph and $\bar{G}$ its complement. If $H(G)$ is the number of Hamilton paths of $G$, then $H(G) + H(\bar{G})$ is even [Lo]. The same is true when $G$ is directed. ANOTHER HAMILTON PATH is the following problem: Given a graph (or digraph) $G$ and a Hamilton path in it, find either another Hamilton path in it, or a Hamilton path in its complement $\bar{G}$.

THEOREM 10. *ANOTHER HAMILTON PATH (directed and undirected case) is in PPA.*

*Proof.* We shall construct a graph $\Gamma$ such that its odd-degree nodes are precisely the Hamilton paths of $G$, and those of $\bar{G}$. The graph is bipartite. The left-hand side has as nodes all Hamilton paths of the complete graph over the nodes of $G$. The right-hand side has as nodes all subgraphs of $\bar{G}$ (including the empty subgraph). There is an edge from Hamilton path $h$ on the left side to subgraph $s$ on the right *if and only if $s$ is a proper subgraph of $h$.*

We claim that the only odd-degree nodes are the Hamilton paths of $G$ and $\bar{G}$ on the left side. Suppose a node on the left is not a Hamilton path of $G$ or of $\bar{G}$. Then it contains a non-empty subset of edges of $\bar{G}$, so it will be adjacent to the nodes on the right that correspond to all subsets of that non-empty set; hence it has even degree. On the other hand, a Hamilton path of $G$ on the left will only be adjacent to the empty subgraph, and will thus be a leaf. And a Hamilton path of $\bar{G}$ will be adjacent to all of its $2^{n-1}-1$ proper subgraphs and will also have odd degree. Any subgraph on the right will be connected to all possible completions to a Hamilton path (for most subgraphs this number will be zero); an easy calculation shows that, for graphs with five or more nodes, this number is always even. Since it is easy to match together the edges out of all even-degree nodes, the Corollary to Theorem 1 applies, and the problem is in PPA. ▮

Incidentally, the proof that all tournaments have an odd number of Hamilton paths (the exercise in [Lo] that comes right after the previous one) is efficiently constructive, and the corresponding problem is in FP.

Hamilton Decomposition. Thomason shows that each four-regular graph has an even number of decompositions into two disjoint Hamilton cycles [Th] (a proof is given in Theorem 11 below). The problem HAMILTON DECOMPOSITION is the following: Given a graph that is the disjoint union of two given Hamilton cycles, find another such decomposition.

This problem is equivalent to the following question from combinatorial optimization: Given two disjoint traveling salesman tours, provide convincing evidence that they are not *adjacent* on the traveling salesman polytope. Two tours are called adjacent if there is a distance matrix that makes them the only optimum tours. The simplest possible proof of non-adjacency would be two other disjoint tours using the same edges (their costs would always be in between). There are more complex proofs: Any set of tours such that they contain as a convex combination the average of the two given ones is a proof; finding a general proof for an arbitrary pair of tours is an FNP-complete problem [Pa1] (called TSP NON-ADJACENCY). Naturally, for arbitrary tours such a proof may not exist, and therefore the problem can be FNP-complete. In the case of disjoint tours, however, existence is guaranteed by Thomason's theorem, although the proof is, for all we know, hard to find.

Theorem 11. *HAMILTON DECOMPOSITION is in PPA.*

*Proof.* Suppose that $G$ is a loopless graph of degree four with three or more nodes (and possibly parallel edges), and let $x$, $y$ be two of its edges. We denote by $D(x, y)$ the set of all Hamilton decompositions of $G$ in which $x$ and $y$ belong to the same Hamilton cycle, we denote by $C(x, y)$ the set of all other Hamilton decompositions, and let $D = D(x, y) \cup C(x, y)$. For each such set we shall construct graphs $\Gamma(D(x, y))$, $\Gamma(D(x, y))$, and $\Gamma(D)$, with degrees $O(n^2)$, where $n$ is the number of nodes of $G$, such that their sets of odd-degree nodes are $D(x, y)$, $C(x, y)$, and $D$, respectively. (In general, in this proof we shall denote by $\Gamma(S)$ a graph whose set of odd-degree nodes is $S$.) All graphs will be such that adjacency can be computed efficiently and locally. Let us first note the following basic construction: If $\Gamma(S)$ and $\Gamma(T)$ are two such graphs, we can take the union of their nodes and edges to form a new graph (allowing multiple edges). It is easy to see that the resulting graph is $\Gamma(S \oplus T)$ (where $\oplus$ is the symmetric difference). It follows that, if we have constructed all $\Gamma(D(x, y'))$'s then we can also construct $\Gamma(D)$ and $\Gamma(C(x, y))$ by noting that $D = (D(x, y) \oplus D(x, y')) \oplus D(x, y'')$, where $x, y, y', y''$ are the edges adjacent to some node $v$; and $C(x, y) = D(x, y') \oplus D(x, y'')$ (alternatively, $D(x, y) + D$).

Thus, we shall show how to construct $\Gamma(D(x, y))$. The construction is inductive on the number of nodes of $G$ and follows closely the proof in [Th]. It is trivial when $G$ has three nodes, so let us suppose it has four or more. Suppose first that $x$ and $y$ are edges having one endpoint in common. Then consider the graph $G''$ that results by deleting their common node and its edges, and adding two more edges, $z$ connecting the endpoints of $x$ and $y$, and $w$ the endpoints of the other two edges. It is easy to see that Hamilton decompositions in $D(x, y)$ and $C'(z, w)$ (where by $C'$ we denote the $C$ sets of $G'$) are in one-to-one correspondence. Let $\phi$ be the one-to-one mapping assigning to a Hamilton decomposition in $D(x, y)$ the Hamilton decomposition in $C'(z, w)$ for which the edges $x$, $y$ are "shortcut" by $z$, and the other two edges by $w$. Consider next the graph $\Gamma(C'(z, w))$, whose construction we are assuming by induction. Add now to $\Gamma(C'(z, w))$ a node for each Hamilton decomposition in $D$, and the edges $\{[d, \phi(d)] : d \in D(x, y)\}$. The resulting graph is $\Gamma(D(x, y))$. Since, by induction. the odd-degree nodes of $\Gamma(C'(z, w))$ are the Hamilton decompositions in $C'(z, w)$, and our construction adds one edge to each of these (they are the range of $\phi$), the odd degree nodes of $\Gamma(D(x, y))$ are precisely the Hamilton decompositions in $D(x, y)$. This completes the construction for the case in which $x$ and $y$ are adjacent edges—and therefore the construction of $\Gamma(D)$.

Suppose now that $x$ and $y$ are not adjacent. The induction will be now on the length of the shortest path connecting them. We have already proved it when the distance is zero. Suppose then that $x \cdots y'y$ is the shortest path from $x$ to $y$. We assume that we have constructed $\Gamma(D(x, y'))$ and $\Gamma(D(y', y))$. By observing that $C(x, y) = D(x, y') \oplus D(y', y)$, we construct $\Gamma(C(x, y))$ and from it and $\Gamma(D)$ we finally obtain $\Gamma(D(x, y))$. ▮

COROLLARY. *TSP NONADJACENCY in the special case of disjoint tours is in PPA.*

CHÉVALLEY. Chévalley's theorem in number theory considers a system of polynomial equations in $n$ variables in the field modulo $p$ (a prime). It states that if the sum of the degrees of the polynomials is less than $n$, then the number of roots

($n$-vectors of integers modulo $p$ that make all given polynomials zero) is divisible by $p$. The computational problem CHÉVALLEY MOD $p$ is this: Given such a system and a root, find another.

This beautiful number-theoretic result has some unexpected combinatorial applications. Here is one, which, to our knowledge, has not been observed before: Given a graph with $n$ nodes whose set of edges is the union of $k<n/2$ (not necessarily disjoint) classes (with multiple edges *and loops*), there is a non-empty induced graph with an even number of edges from each class. (*Proof.* Apply Chévalley's theorem with $p=2$ to the $k$ bilinear forms defined by the upper-triangular adjacency matrices of the color classes, and exclude the trivial root.) This immediately suggests a problem which we call TOTALLY EVEN SUBGRAPH. We know of no polynomial algorithm that finds such a subgraph.

More sophisticated combinatorial applications of Chévalley's theorem can be found in [AFK], where it is shown that any graph of degree four *plus an edge* contains a cubic subgraph. The problem CUBIC SUBGRAPH is the following: Given a graph with degree four, except for two adjacent nodes that have degree five, find a cubic subgraph. The proof is a direct application of Chévalley's theorem for $p=3$.

THEOREM 12. *CHÉVALLEY MOD 2 is in PPA.*

*Proof.* Suppose that we are given polynomials $p_i(x_1, ..., x_n)$, $i=1, ..., m$ (explicitly as sums of monomials) such that $\sum_{i=1}^{m} \deg(p_i)<n$. We shall construct a graph $\Gamma$ whose odd-degree nodes are precisely the roots of the simultaneous equations $p_i(x_1, ..., x_n)=0$, $i=1, ..., m$ modulo 2. The graph is bipartite. The nodes of the left side are all vectors in $\{0, 1\}^n$. The nodes of the other side are the *terms of the polynomial* $Q=\prod_{i=1}^{m}(1+p_i)$; recall that $Q$ has degree less than $n$. Each such term is the product of monomials from the given polynomials, with at most one monomial from each polynomial. That is, each term can be represented by an $m$-tuple of integers $(a_1, ..., a_m)$, where $0 \leqslant a_i \leqslant L_i$, and $L_i$ is the number of monomials in $p_i$. There is an edge between vector $x$ and term $t$ if and only if $t(x)=1$.

Obviously, any vector $x \in \{0, 1\}^n$ is a root of the equations if and only if it has odd degree (since $x$ is a root of the equations if and only if it is not a root of $Q$). We claim that all nodes in the right-hand side have even degree. That is, each term of $Q$ has an even number of roots. But this is obvious since, because the degree of $Q$ is less than the number of variables, there is a variable that does not appear in $Q$. Therefore, the roots are precisely the odd-degree nodes, and the construction is correct. This completes the proof of Chévalley's theorem.

However, the nodes of this graph have *exponentially large* degrees, so to apply the chessplayer algorithm we must exhibit a pairing function between the edges out of a node (recall the corollary to Theorem 1). For a vector $x$ which is not a root, we pair up the terms $t$ such that $t(x)=1$ as follows: Since $x$ is not a root, for some

$i$, $1+p_i(x)=0$. Pick the smallest such $i$. An even number of terms of $1+p_i$ are 1 at $x$. We pair those terms up by a pairing function $\phi$. Then the mate of term $(a_1, ..., a_i, ..., a_n)$ is $(a_1, ..., \phi(a_i), ..., a_n)$. For a term $t$ represented by $(a_1, ..., a_n)$, we pair up the vectors $x$ for which $t(x)=1$ as follows: Since the degree of $Q$, and thus of $t$, is less then $n$, there is a variable $x_j$ not appearing in $t$; let $x_j$ be the lowest-indexed such variable. We pair up $x$ with $(x_1, ..., 1-x_j, ..., x_n)$. ∎

COROLLARY. *TOTALLY EVEN SUBGRAPH is in PPA.*

For $p>2$, CHÉVALLEY MOD $p$ fails to be in PPA; it is instead in a class that may be called PPA-$p$, in which existence proofs are based on the following generalization of the parity argument:

> *If in a bipartite graph a node has degree not a multiple of p, then there is at least another such node.*

THEOREM 13. *For every odd prime p, CHÉVALLEY MOD p is in PPA-p.*

*Sketch.* The previous construction generalizes quite readily. $Q$ is defined as $\prod_{i=1}^{m}(1-p_i^{p-1})$. The bipartite graph must now have parallel edges, with multiplicity reflecting the value of $t(x)$. The roots are the nodes whose degree is not a multiple of $p$. ∎

COROLLARY. *CUBIC SUBGRAPH is in PPA-3.*

*Proof.* By the reduction to Chévalley (with $p=3$) described in [AFK]. ∎

## 4. COMPLETE PROBLEMS

We show the following:

THEOREM 14. *3D SPERNER is PPAD-complete.*

*Proof.* We shall show that any problem in PPAD reduces to 3D SPERNER. Suppose that we are given a problem A in PPAD, that is, an algorithm $M$ for computing, for each input $x$ and each node $v \in \{0,1\}^N$ (where $N=|x|^k$) of $G(x)$ its possible parent–possible child pair $M(x,v) \in (\{0,1\}^{2N}$. We shall construct in polynomial time, based on $x$ and $M$, an integer $n$ and a polynomial algorithm $M'$ for computing legal colors for the points of the $n \times n \times n$ subdivision of the unit cube such that a solution of the instance $x$ of problem A (that is, a node with a parent and no child in $G(x)$, or vice versa) can be easily recovered from any tetrachromatic cubelet.

We describe the construction informally first: $n$, the number of subdivisions per dimension, will be $2^{3N}$, that is, the cube of the number of nodes of $G(x)$. The vast majority of the points are going to be colored 0. For example, all points on the three faces containing (0, 0, 0) are colored 0, except for points on the edges to the other faces that cannot be legally colored 0. To each of the other three faces we assign a different legal color among 1, 2, 3 and give all its points this color (except for the single boundary edge where this color is not legal). The vertex (1, 1, 1) takes color 1, say. Note that the cubelet containing this vertex is the only one on the exterior that contains all three "rare" colors 1, 2, and 3. Since we have said that the vast majority of points are colored 0, the cubelets where all three rare colors meet are very interesting, since they are suspect tetrachromatic cubelets. These trichromatic cubelets will in fact form thin "lines" within the cube. If any of these lines ever touched the main, 0-colored part of the cube, we would immediately have a tetrachromatic cubelet. For this reason, these lines are almost always surrounded by protective "tubes" (Fig. 7). The points in these tubes are colored with the three rare colors, separated into three distinct sectors as shown. The circular order of the three sectors will determine a "direction" in the tube; that is, we shall think that the tube "moves" in the direction for which the colors 1, 2, and 3 appear in the clockwise order. The thickness of all tubes (in units of $1/n$) will be an adequately large integer, such as 10; call it $\theta$.

*The purpose of the tubes is to simulate paths in* $G(x)$. Since the tubes are essentially determined by the line at their core, we shall informally describe these lines first. The vertices of $G(x)$ (except for the isolated ones, those that have neither parent nor child) will be themselves simulated also by short tubes (of length a few multiples of $\theta$). The vertices are all arranged on a straight line very close to (a few $\theta$'s away from) the edge (1, 1, 1)–(1, 1, 0) of the overall cube and parallel to it (Fig. 8). This line is divided into $2^N$ equal parts, and vertex $j$ of $G(x)$ is simulated by a short tube within the $j$th subdivision. Node $11\cdots1$ of $G(x)$ (adjacent to the standard leaf) is thus located next to the vertex (1, 1, 1) of the cube, which is, very

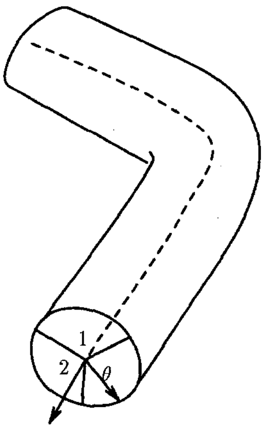

FIG. 7. The tubes.

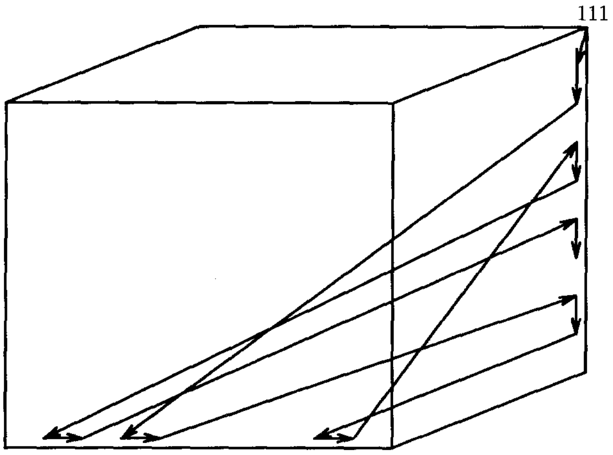

FIG. 8. The broken line, schematically.

appropriately, the known origin of the trichromatic lines. Incidentally, all tetrachromatic cubelets (solutions of the problem) will lie on this line, in positions corresponding to nonstandard source or sink vertices of $G(x)$.

Similarly, each pair of vertices of $G(x)$ will be assigned a position on a line parallel to and near the edge $(0, 0, 0)$–$(1, 0, 0)$ of the cube. Only those pairs that correspond to directed edges of $G(x)$ will be assigned short tubes. Note that the two edges of the cube chosen to host these short tubes are orthogonal and skewed to each other, on parallel faces. Each short tube representing a vertex of edge of $G(x)$ will have a beginning and an end, and its direction will be from the beginning to the end.

We can now complete our sketch of the construction: *For each edge $(i, j)$ of $G(x)$, there is a straight-line tube from the end of the tube corresponding to $i$ to the beginning of the tube corresponding to $(i, j)$, and one from the end of the $(i, j)$ tube to the beginning of the $j$ tube.* Because edges and vertices are arranged far apart from one another, and on lines that are skewed and orthogonal to each other, an easy geometric argument proves that the connecting lines never come close to each other.

Care must be taken so that the division of tubes into monochromatic sectors passes smoothly from one piece of tube to the next (this will be achieved by slowly "rotating" the sectors appropriately). It should be intuitively clear now that the tetrachromatic cubelets sought are precisely at the beginnings or ends of those vertices of $G(x)$ that are nonstandard leaves; this means that the reduction is a valid one. It will be crucial to show that our coloring is such that *the color of each point can be decided locally*, based only on $x$, $M$, and the coordinates of the point.

We are now ready to describe the construction formally. We shall first describe the geometry of the core of the tubes (a possibly disconnected broken line) in terms of $G(x)$, and then we will describe a polynomial-time algorithm for computing the color of a point, based only on $x$, $M$, and the coordinates of the point.

Let us fix $\theta = 10$ (the thickness of the tubes in units of $1/n$). We are assuming that $x$ is long enough so that $2^N$ is much larger than $\theta^3$ (for small $x$, a standard reduction is used, say). The broken line will consist of straight-line segments. Since the line and its segments are directed, each segment has a beginning and an end, both points of the subdivision. For each node $i$ of $G(x)$ that is not isolated, where $i$ is an integer between 1 and $2^N - 1$, there is a segment from point $B(i) = (n - \theta^2, n - \theta^2, (i+1)\,2^{2N} - \theta^2)$ to point $E(i) = (n - \theta^2, n - \theta^2, (i+1)\,2^{2N} - 2\theta^2)$ (we are omitting the factor $1/n$ in coordinates of points). Note that we are also omitting the standard leaf 0, since we are picking up its path from the next node, $11 \cdots 1$. For each pair of nodes $(i, j)$ such that there is a directed edge $(i, j)$ from $i$ to $j$, we have a broken line from $E(i)$ to $B'(i, j) = (2^N i + 2^{2N} j + \theta^2, \theta^2, \theta^2)$, from there to $E'(i, j) = (2^N i + 2^{2N} j + 2\theta^2, \theta^2, \theta^2)$, and from there to $B(j)$. Finally, there is a segment from $(n, n, n)$ to $(n - \theta^2, n - \theta^2, \theta^2)$. See Fig. 8 for a schematic illustration of the construction. Call a point from which a segment starts but no segment ends a source; the opposite is called a *sink*. It is easy to see that the construction guarantees the following:

LEMMA 1. *The broken line described has a source at $(n, n, n)$. Any other point is a source if and only if it is of the form $B(i)$ and $i$ is a nonstandard source of $G(x)$. A point is a sink if and only if it is of the form $E(i)$, and $i$ is a sink of $G(x)$.*

So far we have essentially shown that an intermediate problem, that of finding non-standard sources and sinks in a directed broken line with breakpoints that are points in a cube, given by an algorithm that recognizes its breakpoints in the cube and their connections, is PPAD-complete. The purpose of the remaining part of the construction is to reduce this problem to 3D SPERNER.

The next step of our construction is to create appropriate "tubes" around the broken lines described so far. We first describe the colors of the points in the exterior of the cube. Any point of the form $(0, x, y)$, $(x, 0, y)$, or $(x, y, 0)$ with $x, y < n$ obtains color 0. Any point of the form $(n, x, y)$ takes color 1. Any point of the form $(x, n, y)$, with $x < n$, obtains color 2. And any point of the form $(x, y, n)$, $x, y < n$, takes color 3. All other points will obtain color 0 *unless they are $\theta$ or closer to the broken line.* We shall have to describe how one can easily decide this from the coordinates of a point, and then describe how the correct color among 1, 2, 3 is chosen for each such point.

Let $p = (x, y, z)$ be a point in the interior of the cube. There are two cases. If $p$ is at a distance of less than $2^{2N}$ from the line $(\lambda, \theta^2, \theta^2)$, where all $B'$s and all $E'$s lie, then it is easy to see that there are at most two such points close enough to it and thus at most three lines and two points to check. If on the other hand, $p$ is at a distance greater than $2^{2N}$ from the line $(\lambda, \theta^2, \theta^2)$, then (see Fig. 9) a segment connecting that line to the line $(n - \theta^2, n - \theta^2, \lambda)$ (where all $B$'s and all $E$'s lie) and passing $\theta$ close to $p$ can only originate from an interval of length at most $\theta^2$ on the

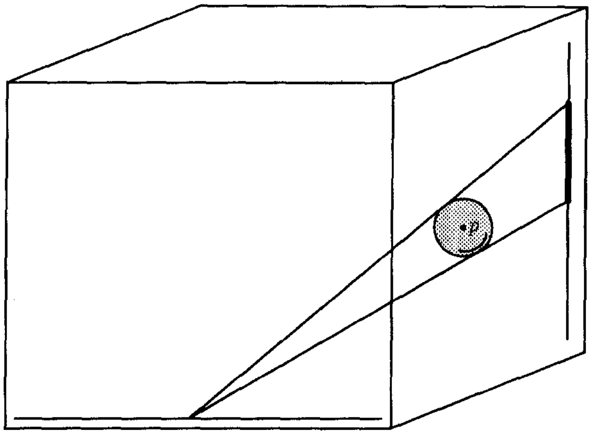

FIG. 9. Determining the closest segment.

line $(n-\theta^2, n-\theta^2, \lambda)$. However, on such an interval at most one $B$ and one $E$ can exist and thus at most three lines and two points must again be checked.

LEMMA 2. *There is a polynomial-time algorithm which, given $M$, $x$, and point $p$ returns "yes" if $p$ is closer than $\theta$ from any part of the broken line, and returns "no" if there is no part of the broken line closer than $\theta+1$ to $p$.*

It is thus easy for a point to decide whether its color is 0 or not. It is somewhat more involved to decide between the three rare colors. Consider any segment of the broken line. By the construction so far it is surrounded by a tube of radius between $\theta$ and $\theta+1$. Consider the cross section of the tube with a plane perpendicular to the segment (Fig. 10). We shall define the boundaries of the three colors in the tube. In fact, we need only define the boundary between colors 1 and 2, as the other two will be determined by rotations of 120°.

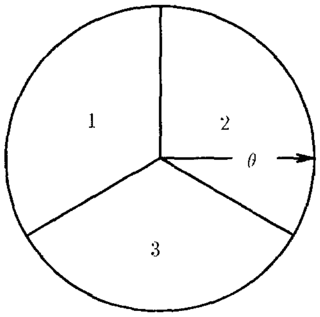

FIG. 10. The three sectors.

For the tubes on the $(n-\theta^2, n-\theta^2, \lambda)$ line, as well as for the tube beginning at vertex $(1, 1, 1)$, the boundary is the direction towards the $(1, 1, 1)$–$(1, 1, 0)$ edge of the cube (note that, appropriately, this edge also separates the faces colored 1 and 2). This assures that there is no tetrachromatic cubelet near vertex $(1, 1, 1)$. For the tubes along the line $(\lambda, \theta^2, \theta^2)$, this is much less crucial; let us say that the boundary is in the direction towards the $(0, 0, 0)$–$(1, 0, 0)$ edge of the cube.

What happens when a segment ends and another begins? The tubes then „turn" as well. We assume that the boundary changes so that it forms the same angle with the normal to the plane defined by the two segments (Fig. 11). We have thus defined the boundary for all tubes, except for those going between the two lines. Even for these, this latter convention defines the boundary at their two endpoints. Naturally, the boundaries at the two endpoints may not coincide. Then the boundary is completely determined by the following rule: *We assume that the boundary rotates in the clockwise direction at a constant rate so that it coincides with the two boundaries at the endpoints, and it makes less than a full clockwise rotation totally.* This completes the definition of the coloring.

LEMMA 3. *There is a polynomial-time algorithm which, given $M$, $x$, and a point $p$ decides the correct color for point $p$.*

*Proof.* If $p$ is on the exterior, this is easy. Otherwise, we first determine whether $p$ is close to a segment by the algorithm in Lemma 2; if not, the color is 0. Otherwise, the algorithm determines the boundaries between the colors at the cross section of the tube, where it belongs, and decides on the color according to the position of $p$ on the cross section. ∎

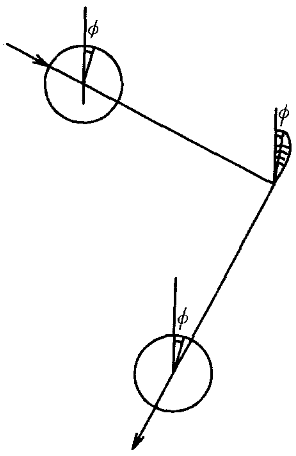

FIG. 11. Turning a tube.

The algorithm of Lemma 3 can be easily rendered as a Turing machine $M'$, whose description can be constructed in polynomial time from the input $x$ and the description of $M$; it is in fact the output of our reduction. Finally, from a tetrachromatic cubelet we can easily recover a nonstandard leaf of $G(x)$.

LEMMA 4. *A tetrachromatic cubelet always contains as a vertex either $B(i)$, where $i$ is a nonstandard source of $G(x)$, or $E(i)$, where $i$ is a sink of $G(x)$.*

*Proof.* The only places where the three rare colors meet is at cubelets at the core of the tubes. And the only places where these cubelets are not surrounded by a tube are the ones corresponding to nonstandard leaves. ▌

This completes the proof of Theorem 14. ▌

COROLLARY. *The problems BROUWER, KAKUTANI, and EXCHANGE ECONOMY are PPAD-complete.*

*Proof.* For BROUWER, we can assign a small "displacement" vector to each color and define a Brouwer function which at the points is the point plus the displacement associated with its color. The function is then extended to the cube by interpolation and to the simplex by any convenient continuous mapping. The other two problems are generalizations of BROUWER. ▌

THEOREM 15. *TUCKER in three dimensions is PPAD-complete.*

*Sketch.* The proof is very similar to that of Theorem 14, and we only present the basic idea, namely, how to implement "tubes" in the context of Tucker's lemma. The idea is shown in Fig. 12 for two dimensions: The tube is a thin region of $-1$'s, "protected" from touching its environment of $+1$'s by thin layers of $+2$'s and $-2$'s. Obviously, the tube starts at the origin, and the only solution of Tucker's problem is at the end of the tube. Once more, we need the third dimension to avoid the crossing of tubes implementing edges of $G(x)$. ▌

By applying the reduction implicit in the proof of the Borsuk–Ulam theorem in Section 3, we obtain

COROLLARY. *BORSUK–ULAM is PPAD-complete.*

In fact, the idea in Fig. 12 yields a result analogous to the lower bound in [HPV] (the proof is omitted).

THEOREM 16. *Any algorithm for solving BORSUK–ULAM in the $(d+1)$-dimensional sphere, with $d>1$, within $\varepsilon$ (that is, finding an $x$ such that $|f(x)-f(-x)| \leqslant \varepsilon$) using the machine that computes $f$ as an oracle requires $(1/\varepsilon)^{cd}$ steps for some $c>0$.*

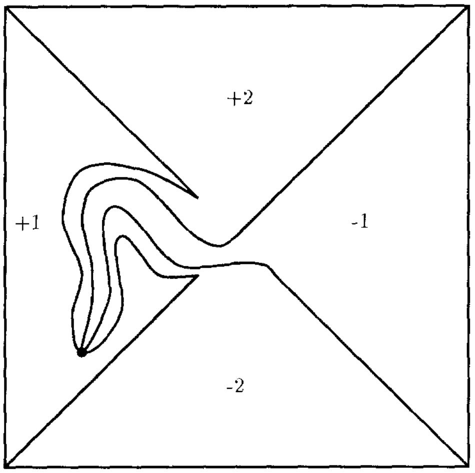

FIG. 12. The construction for TUCKER (two dimensions).

## 5. RELATED CLASSES AND OPEN PROBLEMS

We have introduced and studied PPA and PPAD, two apparently important syntactic subclasses of TFNP; PLS [JPY] is another such class. Another important source of "inefficiently constructive" proofs of existence is Lovász's local lemma [Sp]. We can define a class PLL (for *polynomial local lemma*).[2] A problem A in PLL is defined in terms of a polynomial algorithm $M$, much like PLS and PPA. Given input $x$, the set of possible solutions is again $\{0, 1\}^{[p(|x|)]}$. $M$ takes as inputs triples of the form string–integer–string; on input $(x, j, \lambda)$ (where $x$ is the original input, $\lambda$ is the empty string, and $j \leqslant p(|x|))$ $M$ generates a set $D_j \subset \{1, 2, ..., p(|x|)\}$, with $|D_j| \leqslant \log |x|$ for all $j$. Intuitively, the $D_j$'s are the domains of local conditions on the strings. On input $(x, j, y)$ where $1 \leqslant j \leqslant p(|x|)$ and $y \in \{0, 1\}^{|D_j|}$, $M$ outputs "yes" or "no" (whether or not the string satisfies the local conditions).

To define the desired output for input $x$, let $N(x, j)$ be $\{i \neq j : D_i \cap D_j \neq \emptyset\}$, and let $a(x, j)$ be the number of $y$'s for which $M(x, j, y) =$ "yes" (note that these sets and quantities are polynomial-time computable, because the length of the $y$'s is logarithmic in that of $x$). If it so happens that for some $j$

$$\sum_{i \in N(x,j)} \frac{a(x, i)}{2^{|D_i|}} \geqslant \frac{1}{e},$$

[2] Our definition builds on ideas by J. Kratochvil and M. Fellows [Kra], who defined a sequence of classes PLL[$k$], one for each size of the $D_j$'s in our definition.

then the correct output is the string "*local lemma does not apply*"; otherwise, the correct output is any string $z$ in $\{0, 1\}^{[p(|x|)]}$ such that $M(x, j, z|_{D_j}) =$ "no" for all $j \leqslant q$. *The existence of at least one such string $z$ is guaranteed by the local lemma* [Sp]. It is easy to see that $\text{FP} \subseteq \text{PLL} \subseteq \text{TFNP}$.

Class PLL contains several interesting problems in combinatorics, for which no polynomial algorithm is known. Examples: Given a directed graph with indegree and outdegree eight, find an even cycle [Al1]; or, given a set of clauses with eight literals in each, and with at most 10 appearances of each variable, find a satisfying truth assignment [Kra]. In fact, it has some (unfortunately, rather unnatural) complete problems. Certain important problems in PLL have been recently shown to be in FP (in fact, NC [Al2]). For example: Given an appropriately sparse regular hypergraph, find a coloring of its vertices with two colors so that no hyperedge is monochromatic. Despite this breakthrough, however, a general technique for making the local lemma constructive seems to remain an important open question.

If we rely on the crude probabilistic method, the situation is much less interesting. Define for a moment the class PPM (for *polynomial probabilistic method*) to have the same definition with PLL, except that the condition now becomes weaker:

$$\sum_{j=1}^{q} \frac{a(x,j)}{2^{|D_j|}} \geqslant 1.$$

If this condition does not hold, a simple argument establishes that at least one string satisfying $M(x, j, z|_{D_j}) =$ "no" for all $j \leqslant q$ exists [Sp]. In fact, a calculation shows that a *constant fraction of all strings* satisfy this condition, and hence finding one is solvable by a randomized algorithm: $\text{PPM} \subseteq \text{FZPP}$ (the class of search problems that can be solved conclusively by randomized algorithms). A little more care establishes a stronger result.

PROPOSITION 2. *$PPM = FP$.*

The algorithm in the proof of Proposition 3 finds the bits of $z$ one-by-one, always choosing the bit that maximizes the probability that a string satisfying all conditions on the remaining bits exists. Unfortunately, there seems to be no obvious way of extending this strategem to PLL.

Are there other natural examples of inefficiently constructive proofs of existence, or have we exhausted all such naturally occurring problems? Let us end by introducing another subset of TFNP which seems to be distinct from FP and the classes defined above: A problem in the class PPP (for polynomial pigeonhole principle)[3] is again defined in terms of a machine $M$. Given an input $x$, the set of „solutions" is once more $S(x) = \sum^{[p(|x|)]}$ and for any $y \in S(x)$, $M(x, y) \in S(x)$. The desired output is either a solution $y \in S(x)$ such that $M(x, y) = 00 \cdots 0$, or a pair of $y, y' \in S)x$ such that $M(x, y) = M(x, y')$; one of the two is guaranteed to exist by the pigeonhole principle. It is easy to see that PPP has a rather natural complete problem:

[3] The same class was independently defined by Steve Cook [Co].

PIGEONHOLE CIRCUIT. Given a Boolean circuit $C$ with $n$ inputs and $n$ outputs—by this we mean a directed acyclic graph with nodes labeled AND (indegree two), OR (indegree two), NOT (indegree one), $x_i$ with $1 \leqslant i \leqslant n$ (indegree zero), and with $n$ designated output nodes—either find an input $x$ such that $C(x) = 0^n$ (here by $C(x)$ we denote the output of $C$ on input $x$, defined the obvious way) or two inputs $x \neq x'$ such that $C(x) = C(x')$.

It is perhaps amusing to note that the same problem *only with monotone circuits* (Boolean circuits without NOT gates) is in FP (*Proof.* If the circuit is monotone, then it is easy to see that among $0^i 1^{n- \ +i}$, $i = 0, ..., n$, either there are two inputs that produce the same output or an input that outputs $00 \cdots 0$). For another example, the following natural problem is in PPP, but is not known to be in FP (or PPP-complete).

EQUAL SUMS. Given $n$ positive integers whose sum is less than $2^n - 1$, find two subsets with the same sum.

As it turns out, PPP is closely related to much more familiar issues. Recall that a one-way permutation is a polynomial-time algorithm $\pi$ mapping $\Sigma^n$ to itself for each $n$, such that for all $x \neq y$, $\pi(x) \neq \pi(y)$ such that it is exponentially hard to recover $x$ from $\pi(x)$. The existence of such algorithms is a widely accepted strengthening of $\mathrm{P} \neq \mathrm{NP}$.

PROPOSITION 3. *If PPP = FP then one-way permutations do not exist.*

*Proof.* Suppose that PIGEONHOLE CIRCUIT is in FP. Then we could invert any alleged one-way length-preserving polynomially computable permutation $\pi$ by the following: Given $x$, in order to find a $y$ such that $\pi(y) = x$, we turn the algorithm for $\pi$ (on inputs of length $|x|$) into a circuit, and add an exclusive-or of the output with the bitstring $x$. We then apply to the resulting circuit $C$ the assumed polynomial algorithm for PIGEONHOLE CIRCUIT. Since $\pi$ is a permutation, the algorithm cannot return two inputs with the same output, so it must return a $y$ such that $C(y) = 0$; that is, $\pi(y) = x$. ▌

As was observed by Steve Vavasis and Mihalis Yannakakis, PPP is also related with the other classes we considered.

PROPOSITION 4. *PPAD $\subseteq$ PPP.*

*Proof.* We can reformulate any problem in PPAD with graph $G$ as a problem in PPP by defining $\pi(x)$ to be equal to: (a) $x$ itself if $x$ is a sink; (b) $G^2(x)$ if $x$ is a nonstandard source; and (c) $G(x)$ otherwise. It is easy to see that from any confluence of values of $\pi$ we can recover a nonstandard source or a sink of $G$. ▌

There is a host of open problems left open by this work. We mention some below:

(1) Define other complexity classes that are syntactic subclasses of TFNP and containing natural, important problems, by isolating other styles of "inefficiently constructive proofs of existence." We currently know of no such problems.

(2) Naturally, one has to consider whether PPA = FP. Even the weaker result PPAD = FP would have the remarkable consequence that BROUWER $\in$ FP, obviously by an algorithm that somehow circumvents the lower bound in [HPV]. We consider this very unlikely.

(3) Is 2D-SPERNER PPAD-complete? At present we seen no way of defeating the problem created by crossings of the tubes. (A similar difficulty is present at the two-dimensional case of TUCKER.) The graph $G(x)$ related to any problem in PPA is, of course, always planar, but this cannot be used, as the construction requires strictly local knowledge at each point. If 2D-SPERNER is not PPAD-complete, then we have an interesting hierarchy of problems collapsing to the third level, where 1D-SPERNER represents FP (it embodies binary search in an exponential domain), 3D-SPERNER represents PPAD, and 2D-SPERNER, some other intermediate class.

(4) Can we show more problems complete for PPA and PPAD—especially ones that do not have a Turing machine embedded in the input? For example, is SMITH PPA-complete? We strongly suspect that it is. Also, is the "equal sums problem" complete for PPP?

(5) Is PPA = PPAD? If we define a version of PPAD so that we are only looking for a nonstandard sink, not a source (a class apparently adding no natural problems of interest), is this version equal to PPAD? (In a preliminary version of this paper [Pa2] it was erroneously stated that it is.) Is PPP = PPAD?

(6) Is PPAD a subset of PLS? Is PPA related to FZPP?

(7) Although our complexity classes group together total functions whose totality is proved by fairly elementary techniques (the parity argument, the pigeonhole principle, and so on), apparently these proofs are still not elementary enough. It follows from the results in Sam Buss's thesis [Bu] that, if the proof of totality for a function in FNP can be expressed in the logical language $S_2^1$ (a natural formulation of polynomially bounded arithmetic), then the function is polynomially computable. The question thus arises: As a small first step towards separating these classes from FP, can one show that for some of our problems totality cannot be shown within this language? It is known that a relativized version of the pigeonhole principle cannot be proved in the $S_2^1$ framework.

(8) I suspect that all these classes can be separated in all possible ways by oracles. See [BCPP] for several such separation results.

(9) Finally, is FACTORING in one of these classes (or some other class capturing a more general combinatorial style of existence proofs)? We would not be very surprised if it were, as the arguments underlying Fermat's theorem and Pratt's proof [Pr] do have a subtly combinatorial flavor.

## Acknowledgments

While wondering aloud about the problems in this paper, I benefited tremendously from discussions with several people. Nimrod Megiddo introduced me to problems NASH and P-LCP and collaborated with me in my first attempt at understanding the issue. Mihalis Yannakakis encouraged me to look for a syntactic formulation of PPA. Steve Vavasis pointed out to me the difficulty related to the directed version of the parity argument, by catching a key error in an earlier version of Theorem 14. Noga Alon brought to my attention many of the references related to the local lemma and Chévalley's theorem, and Svatopluk Poljak reminded me of Smith's theorem. Mike Saks turned me to the local lemma and suggested Proposition 3. Elias Koutsoupias pointed out Proposition 2, and Steve Vavasis and Mihalis Yannakakis, Proposition 4. Emo Welzl suggested the EQUAL SUMS problem, and Sam Buss, the connection fo $S_2^1$. Steve Cook independently came up with the idea of the class PPP based on the pigeonhole principle. Manuel Blum and Stuart Kurtz contributed to my understanding of the issue of "semantic classes." Finally, my coauthors in [BCPP] made several valuable corrections and suggestions.

## References

[AFK] N. Alon, S. Friedland, and G. Kalai, Regular subgraphs of almost regular graphs, *J. Combin. Theory Ser. B* **37**, No. 1 (1984), 79–91.

[Al1] N. Alon, private communication, May 1990.

[Al2] N. Alon, A parallel algorithmic version of the local lemma, *in* "Proceedings, 1991 FOCS," pp. 586–593.

[AD] K. J. Arrow and G. Debreu, Existence of an equilibrium for a competitive economy, *Econometrica* **27**, No. 3 (1954).

[BCPP] P. Beame, S. A. Cook, C. H. Papadimitriou, and T. Pitassi, The relative complexity of NP search problems, manuscript, 1993.

[Be] C. H. Bennet, "[TIME, SPACE] $(T, S) \subseteq$ REVERSIBLE [TIME, SPACE] $(T^{1.585}, S \log T)$ for Multitape Turing Machines," IBM Yorktown Heights report, Novermber 1984.

[Bu] S. Buss, "Bounded Arithmetic," Bibliopolis, Naples, Italy, 1986; Ph.D. thesis, Princeton University, 1985.

[Co] S. A. Cook, private communication, February 1991.

[CD] R. W. Cottle and G. B. Dantzig, Complementary pivot theory of mathematical programming, *Linear Algebra Appl.* **1** (1986), 103–125.

[Fr] R. M. Freund, Variable dimension complexes. Part I. Basic theory, *Math. Oper. Res.* **9**, No. 4 (1984), 479–497.

[FT] R. M. Freund and J. Rodd, A constructive proof of Tucker's combinatorial lemma, *J. Combin. Theory Ser. A* **30** (1981), 321–325.

[HV] M. Hirsch and S. Vavasis, Exponential lower bounds for finding Brouwer fixpoints, *in* "Proceedings, 1987 FOCS," pp. 401–410; M. Hirsch, C. H. Papadimitriou, and S. Vavasis, Exponential lower bounds for finding Brouwer fixpoints, *J. Complexity* **5** (1989), 379–416.

[GN] D. Gale and H. Nikaidô, The Jacobian matrix and global univalence of mappings, *Math. Ann.* **159** (1965), 81–93.

[JPY] D. S. JOHNSON, C. H. PAPADIMITRIOU, AND M. YANNAKAKIS, How easy is local search?, *in* "Proceedings, 26th Annual Symp. Foundations Comput. Sci., 1985," pp. 39–42; *J. Comput. System Sci.* **37** (1988), 79–100.

[Kra] J. KRATOCHVIL, private communication, July 1990.

[Kou] E. KOUTSOUPIAS, private communication, August 1990.

[Kr] M. W. KRENTEL, Structure of locally optimal solutions, *in* "Proceedings, 30th Annual Symp. Foundations Comp. Sci., 1989," pp. 216–221.

[Le] C. E. LEMKE, Bimatrix equilibrium points and mathematical programming, *Management Sci.* **11** (1965), 681–689.

[LH] C. E. LEMKE AND J. T. HAWSON, JR., Equilibrium points in bimatrix games, *J. SIAM* **12** (1964), 413–423.

[Lo] L. LOVÁSZ, "Combinatorial Problems and Exercises," North Holland, Amsterdam, 1981.

[MP] N. MEGIDDO AND C. H. PAPADIMITRIOU, "A Note on Total Functions, Existence Theorems, and Computational Complexity," IBM Research Report RJ 7091, 1989.

[Pa1] C. H. PAPADIMITRIOU, The adjacency relation on the travelling salesman polytope is NP-complete, *Math. Programming* (1977).

[Pa2] C. H. PAPADIMITRIOU, On graph-theoretic lemmata and complexity classes, *in* "Proceedings, FOCS, 1990.

[PSY] C. H. PAPADIMITRIOU, A. S. SCHÄFFER, AND M. YANNAKAKIS, On the complexity of local search, *in* "Proceedings, STOC, 1990."

[PPT] S. POLJAK, D. TURZIK, AND P. PUDLAK, Extensions of $k$-subsets to $(k+1)$-subsets: Existence versus constructability," *Comment. Math. Univ. Carolin.* **23**, No. 2 (1982), 337–349.

[Pr] V. R. PRATT, Every prime has a succinct certificate, *SIAM J. Comput.* **5**, (1974).

[SS] H. SCARF AND J. SHOVEN (Eds.), "Applied General Equilibrium Analysis," Cambridge Univ. Press, Cambridge, UK, 1984.

[Si] M. SIPSER, On relativization and the existence of complete sets, *in* "Proceedings, 9th ICALP, 1982."

[Sp] J. SPENCER, "Then Lectures on the Probabilistic Method," SIAM, CBMS-NSF Regional Conference Series, No. 52, Philadelphia, 1987.

[Th] A. THOMASON, Hamilton cycles and uniquely edge colourable graphs, *Ann. Discrete Math.* **3** (1978), 259–268.

[To] M. TODD, Orientation in complementary pivot algorithms, *Math. Oper. Res.* **1**, No. 1 (1976), 54–66.

[Uz] H. UZAWA, Walras's existence theorem and Brouwer's fixpoint theorem, *Econ. Stud. Quart.* **13** (1962), 59–62.