The Complexity of Pure Nash Equilibria

[Extended Abstract]

Alex Fabrikant*
Computer Science Division
UC Berkeley, CA 94720
alexf@cs.berkeley.edu

Christos Papadimitriou†
Computer Science Division
UC Berkeley, CA 94720
christos@cs.berkeley.edu

Kunal Talwar‡
Computer Science Division
UC Berkeley, CA 94720
kunal@cs.berkeley.edu

ABSTRACT

We investigate from the computational viewpoint multi-player games that are guaranteed to have pure Nash equilibria. We focus on congestion games, and show that a pure Nash equilibrium can be computed in polynomial time in the symmetric network case, while the problem is PLS-complete in general. We discuss implications to non-atomic congestion games, and we explore the scope of the potential function method for proving existence of pure Nash equilibria.

Categories and Subject Descriptors

F.1.3 [**Computation by Abstract Devices**]: Complexity Measures and Classes—*Reducibility and completeness*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Theory

Keywords

Games, pure Nash equilibria, complexity, congestion games, PLS, PLS-completeness, local search

1. INTRODUCTION

As the ice separating Game Theory from Theoretical Computer Science is melting, some of the fundamental results in Game Theory come under increased complexity-theoretic scrutiny — chief among them the important classical existence theorem due to Nash [16]. Nash's proof that every game has a randomized Nash equilibrium is non-constructive in the polynomial sense. The problem of computing a randomized Nash equilibrium is among the "inefficient proofs of existence" identified in [19], lying, together with Brouwer fixed points, in the class PPAD (for "polynomial parity argument, directed version") — but, unlike Brouwer's problem, Nash's is not known to be PPAD-complete. The two-person case is known to be delightfully combinatorial, but again with an exponential catch (see the exposition in [26] and the recent exponential construction [23]). We now have a clever algorithm for finding approximate Nash equilibria in sub-exponential time [13], while it is known that any twist of the problem quickly makes it NP-hard (see [2] for a unifying proof).

Given the apparent difficulty of finding a randomized Nash equilibrium (let alone the criticism that the concept of randomized strategies has attracted within Game Theory), it is natural to ask what kinds of games possess a *pure* Nash equilibrium.

This is the question that motivates the present work: *Which games have pure Nash equilibria? And under what circumstances can we find such in polynomial time?* We immediately note that, for such a problem to be computationally meaningful, the number of players should be large, and the payoff table must be given in some implicit way (because otherwise exhaustive search of the entries of the payoff table does the trick).

There is a famous and well-studied class of games (and, in fact, one with obvious affinity to networks) that is guaranteed to have pure Nash equilibria: the *congestion games.* Figure 1 shows a congestion game in the setting of networks: three players want to move one unit of flow between designated endpoints of a network by choosing one path each. The cost of each combination of path choices to each player

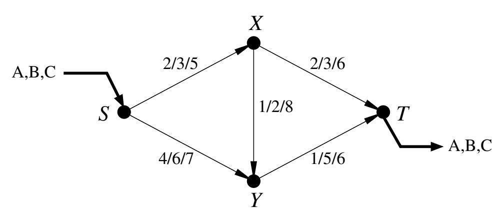

Figure 1: A network congestion game: three players are picking paths from S to T. Each edge is labeled with per-player delays when used by 1, 2, and 3 players.

*Supported by the Fannie and John Hertz Foundation

†Supported by an NSF ITR grant.

‡Research partially supported by the NSF via grants CCR-0121555, CCR-0105533 and ITR grant 331494.

STOC'04, June 13–15, 2004, Chicago, Illinois, USA.

is calculated by adding the *delays* of the edges of the path chosen, where the delay of an edge depends on the number of players using this edge (given here as an explicit function). In the present example, if Player A chooses the path $SXYT$, B chooses SXT, and C chooses SYT, then the costs to the players are 9, 5, and 9 respectively. This is not a Nash equilibrium, because C can defect profitably to path SXT.

In a classical paper [20], Rosenthal proves that, in any such game, "the Nash dynamics converges" (i.e., the directed graph with action combinations as nodes and payoff-improving defections by individual players as edges is acyclic), and hence the game has pure Nash equilibria (the sinks of the graph). The proof, outlined in Section 2, is based on a simple *potential function*. This existence theorem, however, again leaves open the question, does a polynomial-time algorithm for finding pure Nash equilibria in congestion games exist?

We show that the answer is positive when all players have the same origin and destination (the so-called symmetric case, Theorem 2); a pure Nash equilibrium is found by computing the optimum of Rosenthal's potential function, through a reduction to min-cost flow. However, we show (Theorem 3) that computing a pure Nash equilibrium in the general network case is *PLS-complete* [10], which means intuitively "as hard to compute as any object whose existence is guaranteed by a potential function" (see Section 2 for the precise definition). Our proof has as corollary the existence of examples with exponentially long shortest paths, as well as the PSPACE-completeness of the problem of finding a Nash equilibrium reachable from a specified state. The completeness proof is complicated, as it requires the reworking of the reduction, due to [24], to the problem of finding local optima of weighted MAX2SAT instances (possibly the most complex reduction in the literature, if one excludes PCP). When congestion games are posed in the abstract (in terms of sets of resources instead of paths in a network, this being the original formulation), Nash equilibria are PLS-complete to find even in the symmetric case.

Our algorithm for pure Nash equilibria has an application to the non-atomic congestion games studied by Roughgarden and Tardos [21], in which delays are continuous functions. We show that, under the necessary smoothness assumptions, we can approximate the Nash equilibria of such games in strongly polynomial time (Theorem 4).

What other games can be shown to have Nash equilibria by potential functions? Monderer and Shapley [15] have provided an early and devastating answer: only for (inconsequential generalizations of) congestion games can we have a function $\phi(s)$ of the state such that for each defection by a player from s to s' the improvement to the payoff of this player is precisely $\phi(s') - \phi(s)$.

Consider, however, the *party affiliation game*: n players have two actions ("parties") $\{-1, 1\}$ to choose from and the payoff for i of choices $(s_1, \ldots, s_n)$ is $\mathrm{sgn}(\sum_j s_i \cdot s_j \cdot w_{ij})$, where w_{ij} are given symmetric integer weights (positive or negative). Intuitively, people are happy when they are in the same party as their friends, and at different parties than their enemies, and the weights capture the warmth of the relationship between two people. It is easy to see that in this game the Nash dynamics converges, and the function $\phi(s) = \sum_{i,j} s_i \cdot s_j \cdot w_{ij}$ can serve as a potential function in a sense. And still, this game is definitely *not* a congestion game (it is easy to see that it is a local optimality version of the MAX CUT problem, and related to the convergence of Hopfield neural networks), in apparent contradiction with the negative result of [15]. What is going on?

There is a weakness in the negative result of [15]: the requirement that the two differences be the same is far too strict; they need only *have the same sign* for ϕ to be a valid potential function for the purposes of local search; we will refer to this as a *general potential function*. We show (Theorem 6) that, under the relaxed definition, the space of "potential games" is much richer, essentially encompassing all of the class PLS: any problem in PLS can be coached as a game whose pure equilibria are guaranteed to exist by a potential function argument.

Finally, in Section 5 we discuss several open problems, the most general and important of which is understanding better the nature of games that are guaranteed to have pure Nash equilibria; we also review some other known classes of such games, proving more general results in some cases.

2. DEFINITIONS AND NOTATION

Games. A *game* with $n \geq 2$ players is a finite set of actions S_i for each player, and a payoff function u_i for each player mapping $S_1 \times \cdots \times S_n$ to the integers. The elements of $S_1 \times \cdots \times S_n$ will be called *action combinations* or *states*. A *(pure) Nash equilibrium* is a state $s = (s_1, \ldots, s_n)$ such that for each i $u_i(s_1, \ldots, s_i, \ldots, s_n) \geq u_i(s_1, \ldots, s_i', \ldots, s_n)$ for any $s_i' \in S_i$. In general a game may not have pure Nash equilibria. (However, Nash proved [16] that if we extend the game to include as strategies for i all possible distributions on S_i, with the obvious extension of the u_i's to capture expectation, then an equilibrium is guaranteed to exist.)

A game is *symmetric* if all S_i's are the same, and all u_i's, considered as a function of the choices of the other players, are identical symmetric functions of $n-1$ variables.

Consider a graph with node set $S_1 \times \cdots \times S_n$ and an edge (s, s') whenever s and s' differ only in one component, say the ith, and $u_i(s') > u_i(s)$. If this graph is acyclic then we say that, for this game, *the Nash dynamics converges*.

PROPOSITION 1. *If the Nash dynamics converges, then there is a pure Nash equilibrium.*

Sketch: The sinks of the graph are precisely the Nash equilibria of the game. □

Congestion Games. We shall consider games in which the u_i's are given implicitly in terms of efficient algorithms computing the utilities based on the input and the state. For example, in a *congestion game* the input is a set of n players, a finite set E of *resources*, and the action sets are $S_i \subseteq 2^E$; we are also given the *delay function* d mapping $E \times \{1, \ldots, n\}$ to the integers. $d_e(j)$ is nondecreasing in j. The payoffs are computed as follows. Let $s = (s_1, \ldots, s_n)$ be a state, and let $f_s(e) = |\{i : e \in s_i\}|$. Then $c_i(s) = -u_i(s) = \sum_{e \in s_i} d_e(f_s(e))$. Intuitively, each player chooses a set of resources (from among the sets available to her), and to compute the cost incurred by i (the negative of her payoff) we add the delay of each resource used by i, where the delay of a resource e depends on the congestion $f_s(e)$, the total number of players using e.

In a *network congestion game* the families of sets S_i are presented implicitly as paths in a network. We are given a network (V, E), two nodes $a_i, b_i \in V$ for each player i and again a delay function with the edges playing the role of the resources. The subset of E available as actions to the player i is the set of all paths from a_i to b_i. We shall assume the network is directed.

THEOREM 1 (ROSENTHAL, [20]). *Every congestion game has a pure Nash equilibrium.*

PROOF. The potential function establishing the result is $\phi(s) = \sum_e \sum_{j=1}^{f_s(e)} d_e(j)$. For the proof, reverse the summations: $\phi(s) = \sum_{i=1}^{n} \sum_{e \in s_i} d_e(f_s^{\leq i}(e))$, where by $f_s^{\leq i}(e))$ we denote the total number of players $j \leq i$ using e. Suppose now that (s, s') is an improving defection, and suppose (without loss of generality, since players were ordered arbitrarily) that the defecting player is n. Then:

$$\begin{aligned} \phi(s') - \phi(s) &= \sum_{e \in s_i} d_e(f_{s'}^{\leq n}(e)) - \sum_{e \in s_i} d_e(f_s^{\leq n}(e)) \\ &= \sum_{e \in s_i} d_e(f_{s'}(e)) - \sum_{e \in s_i} d_e(f_s(e)) \\ &= c_i(s') - c_i(s) \end{aligned}$$

Hence, ϕ decreases along all edges of the Nash dynamics graph, and hence the Nash dynamics converges. □

Notice that $\phi(s)$ has no intuitive interpretation as "social welfare" or as any related notion; it just accurately absorbs progress, as a potential function should.

PLS. A problem in PLS [10] is given by (a) a set of instances $I = \Sigma^*$; (b) for each instance $x \in I$ a set of *feasible solutions* $F_x \subseteq \Sigma^{p(|x|)}$; (c) a polynomial oracle c which, given $x \in I$ and $s \in \Sigma^{p(|x|)}$ determines whether $s \in F_x$ and, if so, computes an integer $c(x, s)$ — the cost of s (to simplify matters we assume minimization); and (d) for each $x \in I, s \in F_x$ a neighborhood $N_x(s) \subseteq F_x$; and a polynomial function g which, on input $x \in I$ and $s \in F_x$ returns an $s' \in N_x(s)$ with $c(s') < c(s)$, or, if no such s' exists, returns **"no"**. An instance of the PLS problem is this: "Given $x \in I$, find a local optimum, that is, an $s \in F_x$ such that $g(s) =$ **"no"**."

Since the introduction of this class in [10], many local search problems were shown PLS-complete, including weighted versions of satisfiability, aspects of graph bisection, and the traveling salesman problem [12, 24, 18]. PLS-completeness results are proved in terms of *PLS reductions*, providing also a mapping from local optima of the target problem to local optima of the original. Let us immediately note that, by the proof of Rosenthal's Theorem above, finding a pure Nash equilibrium for a congestion game is in PLS, as it is equivalent to finding a local optimum of ϕ, where the feasible solutions are all states. Notice that this does not imply a polynomial algorithm, since improvements of ϕ can be small and exponentially many. It is shown in [17] that problems in PLS have a PTAS (by appropriately rounding the potential function, and re-rounding after enough steps if necessary to retain accuracy, the improvements become coarse enough, and thus guaranteed to end before too long). However, this does not immediately imply a PTAS for finding ϵ-Nash equilibria, as approximation of the potential does not imply approximation of the individual player's cost.

In the next section we characterize the complexity of computing pure Nash equilibria in congestion games.

3. THE COMPLEXITY OF CONGESTION GAMES

The Algorithm

A network potential game is symmetric if all players have the same endpoints a and b (and thus they all have the same set of paths/strategies).

THEOREM 2. *There is a polynomial algorithm for finding a pure Nash equilibrium in symmetric network congestion games.*

PROOF. The algorithm computes the optimum of $\phi(s)$; since the optimum is also a local optimum, the resulting state $\hat{s}$ is a pure Nash equilibrium.

The algorithm is a reduction to min-cost flow. Given the network $N = (V, E, a, b)$ and the delay functions d_e, we replace in N each edge e with n parallel edges between the same nodes, each with capacity 1, and with costs $d_e(1), \ldots, d_e(n)$. It is easy to see that any (integer) min-cost flow in the new network is a state of the game that minimizes $\phi(s)$. □

As we shall see soon (Theorem 4) this simple algorithmic idea has implications for non-atomic congestion games.

PLS-completeness

In contrast, all three other cases of congestion games are PLS-complete:

THEOREM 3. *It is PLS-complete to find a pure Nash equilibrium in network congestion games of the following sorts:*

(i) General congestion games.

(ii) Symmetric congestion games.

(iii) Asymmetric network congestion games.

Sketch: We explain the simple reduction for (i) because it is the basic framework for the much harder proof for case (iii). We reduce from the following problem: given an instance of not-all-equal-3SAT with weights on its clauses and containing positive literals only, find a truth assignment satisfying clauses whose total weight cannot be improved by flipping a variable. Call this problem POSNAE3FLIP; it is known to be PLS-complete [24].

Given an instance of POSNAE3FLIP, we construct a congestion game as follows. For each 3-clause c of weight w we have two resources e_c and e'_c, with delay that is 0 if there are two or fewer players, and w otherwise. The players are variables. Player x has two strategies: one strategy contains all e_c's for clauses that contain x, and another that contains all e'_c's for the same clauses. Smaller clauses are implemented similarly. It is not hard to see that any Nash equilibrium of the congestion game is a local optimum of the POSNAE3FLIP instance.

The proof of (ii) is by a reduction of the non-symmetric case to the symmetric case. Given a congestion game with action sets $S_1, \ldots, S_n$, we construct the following symmetric game. Let $S'_i = \{s \cup \{e_i\} : s \in S_i\}$ for each i, where the e_i's are distinct new resources with delay function $d_{e_i}(j) = 0$ if $j = 1$, and $d_{e_i}(j) = M$, a very large number, if $j \geq 2$. Consider the symmetric game with the same edges and common strategy set $\bigcup_i S'_i$. It is easy to see that any equilibrium of this game will have one player using a strategy from S'_i, and hence will correspond to (by omitting the e_i's) a specific equilibrium of the original game.

We only present an outline of (iii) here. In order to make the idea in (i) work in a concrete network, we need several modifications and extensions of the original construction of [24]. We need three new kinds of clauses besides POSNAE3FLIP to replace clusters of POSNAE3FLIP clauses of [24] that are incompatible with our proof: a single clause over m variables which is satisfied if *exactly one* of its arguments is true and whose penalty scales linearly with the number of extraneous true arguments; 2SAT clauses with positive literals; and 2SAT clauses with negative literals. We call this problem EXTENDED POSNAE3FLIP, or XPNAE3FLIP. For each such instance we have "network gadgets" for variables and clauses of each type, and we can put them together in a network congestion game where the players are the variables and any truth assignment can be simulated by a state of the game.

The hard part is proving that all Nash equilibria of the resulting game are of this "standard" form and not hybrids that correspond to no truth assignment. The property of the XPNAE3FLIP instance needed for our proof to go through can be stated in terms of a weighted directed graph, called the *witness graph* of the instance, which we define next.

Consider an instance F of XPNAE3FLIP with a set of variables X and a set of clauses C, where $C = C_0 \cup C_1 \cup C_2 \cup C'_2 \cup C_s$, with C_0 being the 2- or 3-literal NAE clauses, C_1 being $\{c_1\}$, where c_1 is the single "one-out-of-m" clause over some set of variables X_1, C_2 and C'_2 — the positive and negative 2-SAT clauses, and C_s — all the other clauses, all of which are single-variable (i.e. of form $x \neq 0$ or $x \neq 1$). Define the set of nodes V to be $V = (X \times \{\mathrm{s}, \mathrm{t}\}) \cup (C_0 \times \{0, 1\}) \cup (C_1 \times \{X_1 \cup \{1\}\}) \cup C_2 \cup C'_2$.

Suppose now that, for every variable $x \in X$, we arrange the nodes corresponding to the clauses in which x appears in two ordered lists. The list $L_1(x)$ starts with $(c_1, 1)$ if $x \in X_1$, and also contains $(c, 1)$ for all clauses $c \in C_0$ in which x appears, and c for each clause c in C_2 in which x appears. The list $L_0(x)$ starts with (c_1, x) if $x \in X_1$, and also contains $(c, 0)$ for all clauses $c \in C_0$ in which x appears, and c for each clause c in C'_2 in which x appears. Suppose then that we are given, besides F, this set of $2|X|$ lists, call them $\mathcal{L}$. The *witness graph* $WG(F, \mathcal{L})$ is a directed graph $(V, E_{\mathcal{L}})$, whose edges are defined as follows: for every variable x, if $L_1(x)$ is $(v^1_1, \ldots, v^1_{k_1})$ and $L_0(x)$ is $(v^0_1, \ldots, v^1_{k_0})$, then the witness graph contains the edges $(x_{\mathrm{s}}, v^1_1), (v^1_1, v^1_2), \ldots, (v^1_{k_1}, x_{\mathrm{t}})$, and $(x_{\mathrm{s}}, v^0_1), (v^0_1, v^0_2), \ldots, (v^0_{k_1}, x_{\mathrm{t}})$. The paths from x_{s} to x_{t} consisting of these edges are called *the two standard paths of variable x*. This definition references some edges multiple times, but we are *not* defining a multi-graph; only one edge between any 2 nodes is added, independently of the number of times it's referenced by the above expression. In particular, there are multiple references to edges connecting two C_0 clauses which share 2 variables (or, inconsequentially, identical clauses in any class), since they appear in the least for each repeated variable. Note that C_s clauses are not involved in the construction of the witness.

Consider now an instance F of XPNAE3FLIP, a set of lists $\mathcal{L}$, and the witness graph $(V, E_{\mathcal{L}})$ *with non-negative integer weights y on $E_{\mathcal{L}}$*. We say that the weighted witness graph $(V, E_{\mathcal{L}}, y)$ is *valid* for F if the following holds: for any variable $x \in X$, the two standard paths for x have the same length (under y), and are strictly the shortest paths from x_{s} to x_{t}. The WITNESSED XPNAE3FLIP problem is the following: given an instance F of XPNAE3FLIP, and a valid weighted witness graph for F, find a truth assignment whose total weight is maximal.

The proof now follows from two results:

LEMMA 1. *There is a PLS reduction from* WITNESSED XPNAE3FLIP *to* NETWORK CONGESTION GAME.

Sketch: The construction of the network uses the witness graph as a blueprint. Each clause-related node is expanded to an edge between two nodes, with all the incoming edges attached to its source, and all the outgoing edges attached to its destination. The weights (delay functions) of these "clause edges" are chosen to reflect the exact penalties[1] for more than 1 variable being true in the case of C_1, and, in case of the other clauses, the penalty for the clause being violated by all variables being equal. The delays of the other edges, those specific to the variables, are set to be incomparably larger than the clause weights at the "proper load", and to be incomparably larger than even that if they are used by too many variables (2 in most cases, 3 if the edge is in the standard path of 2 variables). This ensures that the standard paths are the only ones taken by Nash defectors, and thus there are no spurious Nash equilibria. Any clauses containing 2 variables and a literal are forced to be in C_2 or C'_2, and any clauses containing just 1 variable and a literal are accommodated by charging their weight to the penalty of an arbitrary private edge of that variable (chosen from $L_0(x)$ or $L_1(x)$ depending on the literal). □

LEMMA 2. WITNESSED XPNAE3FLIP *is PLS-complete.*

Sketch: See the appendix. □

This completes our outline of the proof of the theorem. □

COROLLARY 1. *For the three cases in the theorem, (a) there are examples of game instances states from which all Nash equilibria/sinks are exponentially far in the Nash dynamics graph, and (b) the problem, given a state s, find a Nash equilibrium reachable from s is PSPACE-complete.*

Sketch: Our reductions preserve these properties of POSNAE3FLIP [24]. □

The Non-atomic Case

The non-atomic congestion game, studied extensively and productively in the work of Roughgarden and Tardos [22], is the limit of the congestion game as n, the number of players, goes to infinity. We are given a network (V, E) and endpoint pairs $(a_i, b_i), i = 1, \ldots, k$ as well as flow requirements $r_i, i = 1, \ldots, k$, rational numbers adding to 1; also for each edge $e \in E$ a non-decreasing delay function $d_e : [0, 1] \mapsto \Re_+$. For a path p and flow f, define the delay of the path $d_p(f)$ to be $\sum_{e \in p} d_e(f)$. We wish to find a k-commodity flow f satisfying the flow requirements that is a Nash equilibrium, that is, for all pairs of endpoints a_i, b_i, any flow path between

[1]It is here that the symmetry of the penalty function for C_1 clauses is needed — the penalty has to be independent of which variables are true

a_i and b_i (i.e. a path with nonzero a_i-b_i flow) has a delay at no larger than any other a_i-b_i path p'.

We note that, as observed in [1, 3, 22] this problem can be rephrased as a convex optimization problem and hence can be solved by the Ellipsoid method. As pointed out by Lisa Fleischer [5], an approximation scheme for the above convex optimization problem also follows from combining the techniques in [9, 8, 11], though that does not necessarily give an approximate Nash Equilibrium (see definition below). Our algorithm is also combinatorial and runs in strongly polynomial time. We make the following assumption on the latency functions d_e.

Lipschitz assumption: There exists a constant K such that for edges e, for all $0 \le x < y \le 1$, $|d_e(y) - d_e(x)| \le K|y - x|$.

Approximate Nash Equilibria: We say a state $s = (s_1, \ldots, s_k)$ is an ϵ-approximate Nash equilibrium if for every i, every flow path p carrying at least ϵ units of flow, and every a_i-b_i path p', the delay $d_p(f)$ is no larger than $d_{p'}(f) - \epsilon$. In words, no player has a non-negligible defection that decreases her delay by more than ϵ.

We outline an algorithm for computing an ϵ-approximate Nash equilibrium in any congestion game. With some foresight, we set $\delta = \frac{\epsilon}{4mK}$ (where K is an upper bound on the lipschitz constants of the latency functions above, and $m = |E|$).

Recall that a Nash equilibrium of the non-atomic congestion game is a flow that optimizes the potential function $\phi(f) = \sum_e \phi_f(f_e)$, where $\phi_e(f_e) = \int_0^{f_e} d_e(t)\mathrm{d}t$. Taking a cue from our algorithm for the symmetric atomic case, we define an instance of the multicommdity min cost flow problem. We replace edge e by a sequence of parallel arcs each of capacity δ, where the cost per unit flow of the i^{th} arc is set to $\frac{\phi(i\delta) - \phi((i-1)\delta)}{\delta} = \frac{\int_{(i-1)\delta}^{i\delta} d_e(t)\mathrm{d}t}{\delta}$. Since the delay function is increasing, so are the costs of successive arcs corresponding to a particular edge. We say a flow f on this instance is *canonical* if for every edge in the original graph, f uses the first i arcs to their capacity, and does not use arcs $i+2$ onwards. Clearly there is always a canonical optimum to the min cost flow instance. Moreover, there is a one-to-one correspondence between flows in the original graph and canonical flows in min cost flow instance.

Consider an optimal canonical solution to the multicommodity min cost flow instance. This defines a flow f on the original graph. Consider a rounded down version $\overline{f}$ of this flow, where $\overline{f}_e = \delta\lfloor \frac{f_e}{\delta} \rfloor$. Note that $\overline{f}$ does not necessarily satisfy flow conservation, and hence need not be a valid flow.

Since $\overline{f}$ is an "integral" (pseudo)flow, the cost of this flow is exactly equal to the potential function value evaluated at this flow. Recall that f optimizes the cost function $C(f) = \sum_e C_e(f_e)$, where $C_e(f_e)$ is a piecewise linear approximation of $\phi_e(f) = \int_0^{f_e} d_e(t)\mathrm{d}t$. Thus $C_e(i\delta) = \phi_e(i\delta)$ for any integer i. Now, for any edge e,

$$\begin{aligned}
|\phi_e(f_e) - C_e(f_e)| &= \left|(\phi_e(f_e) - \phi_e(\overline{f}_e)) - (C_e(f_e) - C_e(\overline{f}_e))\right| \\
&= \left| \int_{\overline{f}_e}^{f_e} d_e(t)\mathrm{d}t - \frac{(f_e - \overline{f}_e)\int_{\overline{f}_e}^{\overline{f}_e+\delta} d_e(t)\mathrm{d}t}{\delta} \right| \\
&= \int_{\overline{f}_e}^{f_e} \left| d_e(t) - \left(\frac{\int_{\overline{f}_e}^{\overline{f}_e+\delta} d_e(t)\mathrm{d}t}{\delta} \right) \right| \mathrm{d}t \\
&\le \int_{\overline{f}_e}^{f_e} \left| d_e(\overline{f}_e + \delta) - d_e(\overline{f}) \right| \mathrm{d}t \\
&\le \delta \cdot K\delta = K\delta^2
\end{aligned}$$

where in the first inequality, we have used the fact that for any function f and any point t, $|f(t) - f_{av}|$ is at most $|f_{max} - f_{min}|$. The second inequality uses the lipschitz condition.

Thus $|\phi(f) - C(f)| \le Km\delta^2 \le \frac{\epsilon^2}{16mK}$. Thus the function $C(f)$ approximates the potential function $\phi(f)$ within an additive error of $\frac{\epsilon^2}{16mK}$.

Now suppose that the optimal min cost flow f is not an ϵ-approximate Nash equilibrium. Then there is an i, a flow path p and a path p' such that $d_p(f) > d_{p'}(f) - \epsilon$. Rerouting $\epsilon/2mK$ units of flow from p to p' then improves ϕ by $\frac{\epsilon^2}{4mK}$. This however implies that C can be improved by at least $\frac{\epsilon^2}{8mK}$, contradicting the optimality of f. Thus we have established that any flow f optimizing C corresponds to an ϵ-approximate Nash equilirium of the congestion game. Moreover, a near optimal min cost flow is still an ϵ-approximate Nash Equilibria.

What remains then is to compute the min cost flow f. This however is a linear programming problem, and also has strongly polynomial time combinatorial algorithms (see e.g. [9, 7]). Thus we have shown that

THEOREM 4. *Given a non-atomic congestion game with delay functions satisfying the Lipshitz assumption with constant C, an ϵ-approximate Nash equilibrium can be computed in time poly$(m, C, \frac{1}{\epsilon})$.*

4. GENERAL POTENTIAL GAMES

We have seen that potential functions are valuable for proving the existence of pure Nash equilibria. What is the precise scope of this method?

Call a game an *exact potential game* if there is a function ϕ such that for any edge of the Nash dynamics graph (s, s') with defector i we have $\phi(s') - \phi(s) = u_i(s') - u_i(s)$. A result of Monderer and Shapley [15] establishes the following disappointing fact (as restated in [27]):

THEOREM 5 ([15]). *Any exact potential game is isomorphic to a congestion game.*

Hence, the applicability of the potential function method is limited essentially to Rosenthal's theorem.

Recall however the party affiliation game from the introduction. Existence of Nash equilibria *was* proved by a potential function — albeit not an exact one. For any edge (s, s') of the Nash dynamics graph with defector i we have $u_i(s') - u_i(s) = 2$, whereas $\phi(s') - \phi(s)$ can be any positive number. The potential function argument for convergence requires only that $\mathrm{sgn}(\phi(s') - \phi(s)) = \mathrm{sgn}(u_i(s') - u_i(s))$.

Let *general potential games* be games that have general potential functions, i.e. ones satisfying this inequality. The question now becomes, how rich is this class of games? We note immediately that if a family of games has polynomially computable general potentials, then the problem of finding a pure Nash equilibrium is in PLS. Our next result is a converse statement: the class of general potential games essentially comprises all of PLS.

THEOREM 6. *For any problem in PLS with instances I there is a family of general potential games indexed by I such that, for problem instance x, the game G_x has poly$(|x|)$ players each with strategy set that includes the alphabet Σ, and such that the set of pure Nash equilibria of G_x is precisely the set of local optima of x.*

Sketch: By generalizing the construction that took us from the MAX CUT local optimality under the natural neighborhood for the party affiliation game. The players are dimensions of the solution space, and a local search improvement is translated into a sequence of Nash defections (first by a lead player, then by others) leading to a new feasible solution. $\square$

5. DISCUSSION AND OPEN PROBLEMS

What other games are guaranteed to have pure Nash equilibria? Vetta identifies in [25] the "basic utility games" as another class of games where the Nash dynamics converges, as proven by a general potential function. The network creation games [4] are another example, and so are congestion games with *subjective delays* played on a network of parallel edges[14]. In these cases, however, *some* equilibrium can be produced in polynomial time by an inductive argument.

Consider yet another variant of congestion games, the one with *player-specific delays*. We have n players and m parallel edges (strategies), each with a delay function $d_e(S)$, a non-decreasing function of the specific set of the players choosing e (as opposed to their *number*). Generalizing slightly a result in [6] we can show:

PROPOSITION 2. *In any congestion game with player-specific delays the Nash dynamics converges.*

PROOF. Consider a state s, inducing a partition $S_1, \dots, S_m$ of the set of players to the m edges, and consider the multiset of numbers $\mu(s) = \{d_{e_1}(S_1), \dots, d_{e_m}(S_m)\}$. Suppose that a player defects from S_i to S_j to form a state s'; it is easy to see that $\mu(s')$ is *lexicographically smaller* than $\mu(s)$. $\square$

The above argument shows that a quite large class of games has pure equilibria, with the corresponding problem in PLS. However, the potential function used here is rather novel (sort the components and weigh them by the powers of a large number), and we have no idea if such problems can be PLS-complete. Incidentally, counter-examples show that such games in more general networks fail to have pure Nash equilibria.

Are potential functions (the discrete analog of Lyapunov functions) the only way to establish convergence of the Nash dynamics? If the dynamics is acyclic, then there is always an awkward potential function (the topological ordering of the state), but it seems to require exponentially time to compute. Are there examples of convergent games that do not have polynomial-time computable potential functions?

Finally, yet another genre of pure equilibrium existence argument, in fact one of an algebraic, combining nature, seems to be this: If two games are known to have pure equilibria, and their payoff functions are (in some precise sense not defined here) cross-monotonic, then their *union* (same players, the union of the strategy spaces, and the same payoffs) is also guaranteed to have pure Nash equilibria, by a continuity argument. Facility location-related games are an example where this type of argument applies.

6. ACKNOWLEDGEMENTS

We thank Tim Roughgarden for enlightening discussions.

7. REFERENCES

[1] M. Beckmann, C. McGuire, and C. Winsten. *Studies in the Economics of Transportation*. Yale University Press, 1956.

[2] V. Conitzer and T. Sandholm. Complexity results about nash equilibria. In *Proc. of IJCAI*, pages 765–771, 2003.

[3] S. C. Dafermos and F. T. Sparrow. The traffic assignment problem for a general network. *Journal of Research of the National Bureau of Standards*, 73(2):91–118, 1969.

[4] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. On a network creation game. In *Proc. of ACM PODC*, pages 347–351, 2003.

[5] L. Fleischer, 2004. Personal Communication.

[6] D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. Spirakis. The structure and complexity of nash equilibria for a selfish routing game. In *Proc. of ICALP*, pages 123–134, 2002.

[7] N. Garg and J. Konemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *IEEE Symposium on Foundations of Computer Science*, pages 300–309, 1998.

[8] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by successive approximation. *Mathematics of Operations Research*, 15(3):430–466, 1990.

[9] M. D. Grigoriadis and L. G. Khachiyan. Coordination complexity of parallel price-directive decomposition. *Mathematics of Operations Research*, 21:321–340, 1996.

[10] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37:79–100, 1988.

[11] A. V. Karzanov and S. T. McCormick. Polynomial methods for separable convex optimization in unimodular linear spaces with applications. *SIAM Journal of Computing*, 26(4):1245–1275, 1997.

[12] M. W. Krentel. Structure in locally optimal solutions. In *Proc. of IEEE FOCS*, pages 216–221, 1989.

[13] R. J. Lipton, E. Markakis, and A. Mehta. Playing large games using simple strategies. In *Proc. of ACM E-Commerce*, pages 36–41, 2003.

[14] I. Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13:111–124, 1996.

[15] D. Monderer and L. S. Shapley. Potential games. *Games and Economic Behavior*, 14:124–143, 1996.

[16] J. F. Nash. Equilibrium points in n-person games. In *Proc. of National Academy of Sciences*, volume 36, pages 48–49, 1950.

[17] J. B. Orlin, A. P. Punnen, and A. S. Schulz. Approximate local search in combinatorial optimization. In *Proc. of SODA*, pages 580–589, 2004.

[18] C. H. Papadimitriou. The complexity of the lin-kernighan heuristic for the traveling salesman problem. *SIAM Journal on Computing*, 21(3):450–465, 1992.

[19] C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

[20] R. W. Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[21] T. Roughgarden and É. Tardos. Bounding the inefficiency of equilibria in nonatomic congestion games. *Games and Economic Behavior*. To appear.

[22] T. Roughgarden and É. Tardos. How bad is selfish routing? *J. ACM*, 49(2):236–259, 2002.

[23] R. Savani and B. von Stengel. Long lemke-howson paths. Technical Report LSE-CDAM-2003-14, LSE, 2003.

[24] A. A. Schäffer and M. Yannakakis. Simple local search problems that are hard to solve. *SIAM Journal on Computing*, 20(1):56–87, 1991.

[25] A. Vetta. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *Proc. of IEEE FOCS*, pages 416–425, 2002.

[26] B. von Stengel. Computing equilibria for two-person games. In R. J. Aumann and S. Hart, editors, *Handbook of Game Theory, Vol. 3*, chapter 45, pages 1723–1759. North-Holland, Amsterdam, 2002.

[27] M. Voorneveld, P. Borm, F. van Megen, S. Tijs, and G. Facchini. Congestion games and potentials reconsidered. *International Game Theory Review*, 1:283–299, 1999.

APPENDIX

A. PROOF SKETCH OF LEMMA 2

To prove Lemma 2, we show that the reduction in [24] from CIRCUITFLIP to POSNAE3FLIP yields instances of the latter which, once cast as XPNAE3FLIP, always have a valid witness.

The reduction in [24] produces, given a circuit with n gates (without loss of generality, all gates are NORs, with 2 inputs and a fan-out of at most 3), m outputs, and p inputs, a POSNAE3FLIP instance with the following variables:

- Numbering gates from 2 to $2n$, we have, for each gate $i = 2h$ (and separately, for each pair (input,gate), with index i replaced with k, i): g_i, y_{2h-1}, y_{2h}, z_{2h-1}, z_{2h}, and "local variables" (α_i^1, α_i^2, β_i^1, β_i^2, β_i^3, γ_i^1, γ_i^2, γ_i^3, δ_i^1, δ_i^2, ω_i, ρ_i, ψ_i^1, ψ_i^2, ψ_i^3, ψ_i^4, ψ_i^5, ψ_i^6, ζ_i^1, ζ_i^2, ζ_i^3, ζ_i^4, ζ_i^5)
- Implicitly, each output gate is also labeled c_j, $t_{k,j}$, $\hat{c}_j$, or $\hat{t}_{k,j}$ (the existence of negatives of outputs is guaranteed, i.e. built into the circuit beforehand); the c_j's correspond to g_i's, and $t_{k,j}$'s — to $g_{k,i}$'s
- For each input k: d_k, $\hat{d}_k$, e_k, $\hat{e}_k$, v_k, w_k, and "local variables" (θ_k^1, θ_k^2, η_k, μ_k^1, μ_k^2, $\hat{\mu}_k^2$)
- $1+p$ extra variables: y_{2n+1}, and $y_{k,2n+1}$ for each input k.

Figure 2 lists all the clauses produced in the reduction; all indexes k and k' range over inputs, indexes i range over "real" gates (even numbers from 2 to $2n$), indexes h range over all numbers between 1 and $2n+1$, and indexes j range over all outputs. The notation $I_1(g_i)$ refers to the first input to gate i, whether it's the output of some other gate or an input to the circuit (respectively, either a $g_{i'}$ or a v_k/w_k variable). See [24] for the full treatment of the original reduction.

To translate this into the necessary XPNAE3FLIP form, note that (a) any POSNAE3FLIP clause that contains 2 variables and a literal can be put into C_2 or C_2', and (b) the sole C_1 clause arises from clauses in the "clique" in group 1.c1.

Then, the witness is constructed according to the per-variable ordered lists shown in Figure 3. Notation of the form $\{(\text{sequence of clauses with } k \text{ as an argument})\}_{k=1\ldots p}$ means "that sequence for $k = 1$, then that sequence for $k = 2$, etc." Since some of the g, v, w, c, and $\hat{t}$ variables are "aliased" to each other[2], the table indicates where, e.g., a sequence of clauses for a c variable may actually be preceded by a sequence of clauses for the g variable that this c is synonymous with. Both C_2 and C_2' clauses are included in the lists, even though they appear in only 1 of the lists. Clauses in parentheses are the single-variable clauses; these do not affect the witness. Lastly, translating from the per-gate variables (those indexed with i) to the per-(input,gate) variables (those indexed with k, i) is just a matter of replacing "2" with "3" as the "clause class."

The full proof that this witness is valid proceeds roughly by:

1. Showing that most "local variables" (both per-gate and per-input) and g's, v's, w's, c's, and $\hat{d}$'s do not interact with variables outside their respective gate or input — that is, there is no way for the path of, e.g., α_1^1 to diverge to areas of the witness corresponding to variables unrelated to gate 1, and then come back to the node $(\alpha_1^1)_t$.
2. Inspecting all exceptions to the above (γ, ψ, and ζ variables) and all the local interactions to verify that all alternate paths are longer.
3. Showing that, in most cases, the d, $\hat{d}$, e, $\hat{e}$, y, and z variables do not permit paths to a variable in the same group but with a lower index. This way, any diversion by a lower-numbered variable is bound to cause prohibitive congestion for some higher-numbered variable in the same group.
4. Inspecting all the exceptions to this (d's in the C_1 clause, and y/z variables going back by 1 index in 2B/3B), and connections between variable groups to verify that all alternate paths are longer.

We omit the remaining details here.

[2] We assume the circuit is preprocessed to not have any inputs feeding directly to outputs, so the only aliased groupings can be $\{v, g\}$, $\{w, g\}$, $\{g, c\}$, and $\{g, \hat{t}\}$

1. $\forall k \neq k'$:
   c1. d_k, $d_{k'}$, 1

2A. $\forall i$ (i.e. even):
   c1. $I_1(g_i)$, α_i^1, 1
   c2. α_i^1, β_i^1, 0
   c3. β_i^1, γ_i^1, g_i
   c4. γ_i^1, z_i, 0
   c5. $I_1(g_i)$, δ_i^1, 0
   c6. $I_2(g_i)$, α_i^2, 1
   c7. α_i^2, β_i^2, 0
   c8. β_i^2, γ_i^2, g_i
   c9. γ_i^2, z_i, 0
   c10. $I_2(g_i)$, δ_i^2, 0
   c11. δ_i^1, δ_i^2, ω_i
   c12. δ_i^1, δ_i^2, β_i^3
   c13. β_i^3, γ_i^3, g_i
   c14. y_i, γ_i^3, 1

2B. $\forall h \neq 2n+1$:
   c1. y_h, z_h, 0
   c2. z_h, y_{h+1}, 1 $(h < 2n)$

2C. c1. z_{2n}, y_{2n+1}, 1
   c2. y_{2n+1}, 1
   c3. $\forall k$, y_{2n+1}, d_k, 0

2D. $\forall i$:
   c1. ψ_i^1, α_i^1
   c2. ψ_i^2, α_i^2
   c3. ψ_i^3, γ_i^1
   c4. ψ_i^4, γ_i^2
   c5. ψ_i^5, β_i^3
   c6. ψ_i^6, ω_i
   c7. ψ_i^1, y_{i-1}
   c8. ψ_i^2, y_{i-1}
   c9. ψ_i^3, y_{i-1}
   c10. ψ_i^4, y_{i-1}
   c11. ψ_i^5, y_{i-1}
   c12. ψ_i^6, y_{i-1}
   c13. ζ_i^1, β_i^1
   c14. ζ_i^2, β_i^2
   c15. ζ_i^3, δ_i^1
   c16. ζ_i^4, δ_i^2
   c17. ζ_i^5, γ_i^3
   c18. ζ_i^1, z_{i-1}
   c19. ζ_i^2, z_{i-1}
   c20. ζ_i^3, z_{i-1}
   c21. ζ_i^4, z_{i-1}
   c22. ζ_i^5, z_{i-1}

2E. $\forall i$:
   c1. ρ_i, α_i^1
   c2. ρ_i, α_i^2
   c3. ρ_i, g_i
   c4. ρ_i, 0

3A. $\forall k, i$ (i.e. even):
   c1. $I_1(g_{k,i})$, $\alpha_{k,i}^1$, 1
   c2. $\alpha_{k,i}^1$, $\beta_{k,i}^1$, 0
   c3. $\beta_{k,i}^1$, $\gamma_{k,i}^1$, $g_{k,i}$
   c4. $\gamma_{k,i}^1$, $z_{k,i}$, 0
   c5. $I_1(g_{k,i})$, $\delta_{k,i}^1$, 0
   c6. $I_2(g_{k,i})$, $\alpha_{k,i}^2$, 1
   c7. $\alpha_{k,i}^2$, $\beta_{k,i}^2$, 0
   c8. $\beta_{k,i}^2$, $\gamma_{k,i}^2$, $g_{k,i}$
   c9. $\gamma_{k,i}^2$, $z_{k,i}$, 0
   c10. $I_2(g_{k,i})$, $\delta_{k,i}^2$, 0
   c11. $\delta_{k,i}^1$, $\delta_{k,i}^2$, $\omega_{k,i}$
   c12. $\delta_{k,i}^1$, $\delta_{k,i}^2$, $\beta_{k,i}^3$
   c13. $\beta_{k,i}^3$, $\gamma_{k,i}^3$, $g_{k,i}$
   c14. $y_{k,i}$, $\gamma_{k,i}^3$, 1

3B. $\forall k, h \neq 2n+1$:
   c1. $y_{k,h}$, $z_{k,h}$, 0
   c2. $z_{k,h}$, $y_{k,h+1}$, 1 $(h < 2n)$

3C. $\forall k$:
   c1. $y_{k,2n+1}$, d_k, 0

3D. $\forall k, i$:
   c1. $\psi_{k,i}^1$, $\alpha_{k,i}^1$
   c2. $\psi_{k,i}^2$, $\alpha_{k,i}^2$
   c3. $\psi_{k,i}^3$, $\gamma_{k,i}^1$
   c4. $\psi_{k,i}^4$, $\gamma_{k,i}^2$
   c5. $\psi_{k,i}^5$, $\beta_{k,i}^3$
   c6. $\psi_{k,i}^6$, $\omega_{k,i}$
   c7. $\psi_{k,i}^1$, $y_{k,i-1}$
   c8. $\psi_{k,i}^2$, $y_{k,i-1}$
   c9. $\psi_{k,i}^3$, $y_{k,i-1}$
   c10. $\psi_{k,i}^4$, $y_{k,i-1}$
   c11. $\psi_{k,i}^5$, $y_{k,i-1}$
   c12. $\psi_{k,i}^6$, $y_{k,i-1}$
   c13. $\zeta_{k,i}^1$, $\beta_{k,i}^1$
   c14. $\zeta_{k,i}^2$, $\beta_{k,i}^2$
   c15. $\zeta_{k,i}^3$, $\delta_{k,i}^1$
   c16. $\zeta_{k,i}^4$, $\delta_{k,i}^2$
   c17. $\zeta_{k,i}^5$, $\gamma_{k,i}^3$
   c18. $\zeta_{k,i}^1$, $z_{k,i-1}$
   c19. $\zeta_{k,i}^2$, $z_{k,i-1}$
   c20. $\zeta_{k,i}^3$, $z_{k,i-1}$
   c21. $\zeta_{k,i}^4$, $z_{k,i-1}$
   c22. $\zeta_{k,i}^5$, $z_{k,i-1}$

3E. $\forall k, i$:
   c1. $\rho_{k,i}$, $\alpha_{k,i}^1$
   c2. $\rho_{k,i}$, $\alpha_{k,i}^2$
   c3. $\rho_{k,i}$, $g_{k,i}$
   c4. $\rho_{k,i}$, 0

4. $\forall k, j$
   c1. d_k, c_j, 0
   c2. d_k, $\hat{t}_{k,j}$, 1

5. $\forall k$
   c1. d_k, $\hat{d}_k$, 1
   c2. $\hat{d}_k$, 0

6. $\forall k$
   c1. d_k, θ_k^1, 1
   c2. $\hat{d}_k$, θ_k^2, 0
   c3. w_k, θ_k^1, η_k
   c4. w_k, θ_k^2, η_k
   c5. v_k, η_k
   c6. θ_k^1, η_k
   c7. θ_k^2, η_k

7. $\forall k$
   c1. μ_k^1, v_k, w_k
   c2. $\hat{\mu}_k^2$, v_k, w_k
   c3. μ_k^2, $\hat{\mu}_k^2$
   c4. μ_k^1, μ_k^2, e_k
   c5. e_k, $\hat{e}_k$
   c6. e_k, 0
   c7. μ_k^1, 0
   c8. $\hat{\mu}_k^2$, 1

8. $\forall k, h \neq 2n+1$
   c1. e_k, $y_{k,h}$, 1
   c2. $\hat{e}_k$, $z_{k,h}$, 0

9. $\forall k \neq k'$, $\forall h$
   c1. e_k, z_h, 1
   c2. e_k, $z_{k',h}$, 1
   c3. $\hat{e}_k$, y_h, 0
   c4. $\hat{e}_k$, $y_{k',h}$, 0

10. $\forall k \neq k'$, $\forall h$
   c1. d_k, y_h, 1
   c2. d_k, $y_{k',h}$, 1
   c3. $\hat{d}_k$, z_h, 0
   c4. $\hat{d}_k$, $z_{k',h}$, 0

11. $\forall k, h$
   c1. v_k, w_k
   c2. d_k, 1
   c3. z_h, 1
   c4. $z_{k,h}$, 1
   c5. y_h, 0
   c6. $y_{k,h}$, 0

Figure 2: Clauses in the PLS-reduction from CIRCUITFLIP to POSNAE3SAT in [24].

g_i	{2E.c3, 2A.c3, 2A.c8, 2A.c13} as output, or the v/w sequence; then, {2A.c1, 2A.c5} or {2A.c6, 2A.c10} as 1-3 inputs, in gate order, or the $c/\hat{t}$ sequence
α_i^1	2A.c2, 2A.c1, 2D.c1, 2E.c1
α_i^2	2A.c7, 2A.c6, 2D.c2, 2E.c2
β_i^1	2A.c2, 2A.c3, 2D.c13
β_i^2	2A.c7, 2A.c8, 2D.c14
β_i^3	2A.c12, 2A.c13, 2D.c5
γ_i^1	2A.c3, 2D.c3, 2A.c4
γ_i^2	2A.c8, 2D.c4, 2A.c9
γ_i^3	2A.c13, 2D.c17, 2A.c14
δ_i^1	2A.c11, 2A.c12, 2A.c5, 2D.c15
δ_i^2	2A.c11, 2A.c12, 2A.c10, 2D.c16
ω_i	2A.c11, 2D.c6
ρ_i	2E.c3, 2E.c1, 2E.c2, (2E.c4)
ψ_i^1	2D.c1, 2D.c7
ψ_i^2	2D.c2, 2D.c8
ψ_i^3	2D.c3, 2D.c9
ψ_i^4	2D.c4, 2D.c10
ψ_i^5	2D.c5, 2D.c11
ψ_i^6	2D.c6, 2D.c12
ζ_i^1	2D.c13, 2D.c18
ζ_i^2	2D.c14, 2D.c19
ζ_i^3	2D.c15, 2D.c20
ζ_i^4	2D.c16, 2D.c21
ζ_i^5	2D.c17, 2D.c22
y_{2h-1}	2D.c7, 2D.c8, 2D.c9, 2D.c10, 2D.c11, 2D.c12; 2B.c2, 2B.c1; 9.c3$_{k=1\ldots p}$; 10.c1$_{k=1\ldots p}$; (11.c5)
y_{2h}	2A.c14; 2B.c2, 2B.c1; 9.c3$_{k=1\ldots p}$; 10.c1$_{k=1\ldots p}$; (11.c5)
z_{2h-1}	2D.c18, 2D.c19, 2D.c20, 2D.c21, 2D.c22; 2B.c1, 2B.c2; 9.c1$_{k=1\ldots p}$; 10.c3$_{k=1\ldots p}$; (11.c3)
z_{2h}	2A.c4, 2A.c9; 2B.c1, {2B.c2 or 2C.c1 for $2n$}; 9.c1$_{k=1\ldots p}$; 10.c3$_{k=1\ldots p}$; (11.c3) for $2n$
y_{2n+1}	2C.c1; 9.c3$_{k=1\ldots p}$; {2C.c3, 10.c1}$_{k=1\ldots p}$; (2C.c2, 11.c5)
$y_{k,2h-1}$	3D.c7, 3D.c8, 3D.c9, 3D.c10, 3D.c11, 3D.c12; 3B.c2, 3B.c1; {8.c1 if $k' = k$, 9.c4 else}$_{k'=1\ldots p}$; 10.c2$_{k'=1\ldots p}$, (11.c6)
$y_{k,2h}$	3A.c14; 3B.c2, 3B.c1; {8.c1 if $k' = k$, 9.c4 else}$_{k'=1\ldots p}$; 10.c2$_{k=1\ldots p}$; (11.c6)
$z_{k,2h-1}$	3D.c18, 3D.c19, 3D.c20, 3D.c21, 3D.c22; 3B.c1, 3B.c2; {8.c2 if $k' = k$, 9.c2 else}$_{k'=1\ldots p}$; 10.c4$_{k'=1\ldots p}$, (11.c4)
$z_{k,2h}$	3A.c4, 3A.c9; 3B.c1, 3B.c2; {8.c2 if $k' = k$, 9.c2 else}$_{k'=1\ldots p}$; 10.c4$_{k'=1\ldots p}$; (11.c4)
$y_{k,2n+1}$	9.c4$_{k'=1\ldots p}$; {3C.c1 if $k' = k$, 10.c2 else}$_{k'=1\ldots p}$; (11.c6)
c_j	g output sequence, then 4.c1$_{k=1\ldots p}$
$t_{k,j}$	(not used)
$\hat{c}_j$	(not used)
$\hat{t}_{k,j}$	g output sequence, then the single 4.c2
d_k	The 1.c1 clique; 6.c1; 5.c1; 4.c1$_{j=1\ldots m}$, 4.c2$_{j=1\ldots m}$; 10.c1$_{h=1\ldots 2n+1}$, 2C.c3; {3C.c1 if $k' = k$, else 10.c2$_{h=1\ldots 2n+1}$}$_{k'=1\ldots p}$; (11.c2)
$\hat{d}_k$	6.c2; 5.c1; 10.c3$_{h=1\ldots 2n+1}$; 10.c4$_{h=1\ldots 2n+1}$; (5.c2)
e_k	7.c4, 7.c5; 9.c1, {8.c1$_{h=1\ldots 2n+1}$ if $k' = k$, else 9.c2$_{h=1\ldots 2n}$}$_{k'=1\ldots p}$; (7.c6)
$\hat{e}_k$	7.c5; 9.c3, {8.c2$_{h=1\ldots 2n}$ if $k' = k$, else 9.c4$_{h=1\ldots 2n+1}$}$_{k'=1\ldots p}$
v_k	11.c1, 7.c1, 7.c2, 6.c5; then g input sequences
w_k	11.c1, 7.c1, 7.c2, 6.c3, 6.c4; then g input sequences
θ_k^1	6.c6, 6.c3, 6.c1
θ_k^2	6.c4, 6.c7, 6.c2
η_k	6.c6, 6.c3, 6.c4, 6.c7, 6.c5
μ_k^1	7.c1, 7.c4, (7.c7)
μ_k^2	7.c3, 7.c4
$\hat{\mu}_k^2$	7.c3, 7.c2

Figure 3: Per-variable ordered lists that comprise the valid witness.