Complexity of Game Dynamics

by

Alexander Fabrikant

B. S. (UC Berkeley) 2002
B. A. (UC Berkeley) 2002

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Christos H. Papadimitriou, Chair
Professor Alistair Sinclair
Professor Shmuel Oren

Fall 2008

The dissertation of Alexander Fabrikant is approved:

Chair Date

Date

Date

University of California at Berkeley

Fall 2008

Complexity of Game Dynamics

Abstract

Complexity of Game Dynamics

by

Alexander Fabrikant
Doctor of Philosophy in Computer Science

University of California at Berkeley

Professor Christos H. Papadimitriou, Chair

What happens when independent agents interact based on their selfish interests? Game theory's earliest and most natural simple model of such interaction is the best-response Nash dynamics, the process of agents making unilateral moves that are their best response to the actions of others. Pure Nash equilibria, when they do exist, arise naturally as the steady states of this process. When they don't exist, behavioral predictions can be made from "sink equilibria", a universal (guaranteed to exist) generalization of Nash equilibria to "steady clusters of states."

We now know that it is vital for a model of naturally-occuring behavior to be computationally tractable, not only for simulations, but also to check how realistic the model is, since we expect that nature (aside maybe from quantum physics) cannot produce systems with fundamentally more computational power than computers as we know them.

In this dissertation, I show several results about the tractability of analyzing a game's best-response dynamics. If the game payoff tables are given explicitly (in normal form), searching for pure Nash or sink equilibria is trivial. However, most interesting games are represented more succinctly. For pure Nash equilibria, I show that in potential games, a well-studied class of succinct games guaranteed to have pure equilibria, finding pure Nash equilibria is PLS-complete, and the best-response dynamics takes an exponentially long time to converge in the worst case, even when the games are restricted to network routing games, or to symmetric games. For sink equilbria, I prove that it is PSPACE-complete to analyze them in graphical games.

On the practical side, I resolve a decade-old well-known open problem in network-

ing by establishing that it is PSPACE-complete to predict whether Internet inter-domain routing may be destabilized by large-scale oscillations; that is, whether a system of path preferences in the BGP protocol may lead to flapping. This turns out to be a question about the best-response dynamics in a special kind of game.

Lastly, I propose several enhanced equilibrium concepts inspired by game dynamics that allow for higher rationality by the players while mostly retaining the tractability and universality of sink equilibria in normal-form games.

Professor Christos H. Papadimitriou
Dissertation Committee Chair

To a certain "UC Berkeley grad student" dedicated

Contents

List of Figures

Acknowledgements

I am deeply indebted to my thesis advisor, Christos Papadimitriou. His charismatic and inspiring introduction to theoretical computer science was what first drew me to the field almost a decade ago, and his enthusiasm and insights have kept me spellbound ever since. Without his guidance, encouragement, support, patience, and all-but-infallible wisdom, not to mention collaboration, ideas, and constructive criticism, none of the work presented here would have ever taken off the ground.

This dissertation also would not have been possible without the support of the John and Fannie Hertz Foundation, which provided not only generous material support via the Hertz Fellowship, but also useful advice, regular mentoring, and unique networking opportunities.

Many people have been particularly remarkable as teachers and mentors to me over the years, whether in an official capacity or otherwise, and to them I am also in debt. Lev Fedorovich Kolesnikov, the staff of LMSh-92, Linda Colley, Jenny Leung, and Wayne Lee helped set me on this career path in general. Sanjoy Dasgupta, Tad Hogg, and Lior Pachter provided a gentle and inspiring introduction to CS research. Mark Sandler, Shirley Hung, Aaron Snyder, David Y. Chen, Danny Tom, and Kris Hildrum have, over the years, helped me navigate the maze of academia while keeping my head above the water.

On the technical side, all of the results given here were obtained in collaboration with Christos Papadimitriou, and the results in Chapter 2, also in collaboration with Kunal Talwar. Vijay Ramachandran turned my attention to the BGP problem in Chapter 4, and discussions with Robert Kleinberg and P. Brighten Godfrey contributed to the results in chapters 3, 4, and section 5.1.

I also thank Alistair Sinclair and Shmuel Oren, who graciously volunteered to serve on my dissertation committee, and, in addition to doubling this oeuvre's likely readership, contributed several insights and suggestions during my thesis proposal.

I am particularly grateful to David Molnar for, among many other things, volunteering his valuable assistance with the end game of this dissertation; and to Ranjit Jhala, and generations of LaTeXgeeks before him for automating the painful process of typesetting this thesis to UC Berkeley's exacting rules.

My graduate school experience would have been intolerably bleak without the company and support of — and many discussions with, at all hours of the day and night and

on every topic under the sun — a multitude of friends, colleagues, roommates, officemates, and others, all of whom I cannot possibly list here. For their friendship and their patience with me through the high and the low points of the past few years, I thank, lexicographically, Alex Jaffe, Alex Simma, Alice Gutman, Bem Jones-Bey, Boriska Toth, Brighten Godfrey, Costis Daskalakis, David Molnar, David Sun, Devin Jones, Francis Hsu, Franz Cheng, Gagan Prakash, Geoff Morrison, Grant Schoenebeck, Greg Valiant, Helena Chia, Henry Lin, Hoeteck Wee, Jack Lin, Jack Sampson, James Cook, Kamalika Chaudhuri, Karen Nguyen, Lena and Daniel Koslover, Lorenzo Orecchia, Madhur Tulsiani, Michael Constant, Michael Schapira, Omid Etesami, Sam Riesenfeld, Scott Aaronson, Sharon Goldberg, Yan Nusinovich, Yaron Singer, among many others.

My graduate school experience would have been shorter, but also far less enjoyable, without the various distractions contributed by Jody Lewen, Donna Hamamoto, and Maureen Lahiff, inter alia; Geoffrey Skinner, Dave Croker, Tim Oren, and Scott Heeschen, inter alia; David Campbell, Marc "Scuba" Young, and Darius Monsef, inter alia; Marsha Jean Falco; and Offer Grembek, Shadi Anani, and Brian Gross, inter alia; for all of which I'm very grateful even if my advisor might not have been.

And, lastly, and by far most importantly, I could not have done this without the unwavering love, support, patience, and encouragement from my parents, Anatoly and Galina Fabrikant, and from Nicole Holland, my favorite and most important result from my graduate student years, which is nevertheless otherwise omitted from this dissertation other than for one oblique reference. No words can do justice to describe everything I owe to you. I hope to find some approximation to such words in the many years to come.

Chapter 1

Introduction

Games, that is, interactions between self-interested parties, are everywhere. Beyond their original and natural domain in economics, they appear in political science and military strategy, ecology and animal behavior, and now, in the era of global networking, in Internet-scale computer systems, among various other domains. Game theory, the study of such systems, has sought to model the interests of the "players" and the interactions between them, and to forecast the outcomes.

As with any modeling endeavour, game theory has had to balance between the duelling objectives of realism and tractability — the more realistic sophistication is added to a model, the more difficult is it to analyze it and make predictions based on it. Traditionally, these two objectives have been fundamentally separate, both in game theory and elsewhere. The arguments defending a game-theoretic model's realism were based on economics, psychology, or other aspects of the application domain. On the other hand, the tractability of analyzing the model and of making predictions depended on available mathematical tools.

With the rise of computer modeling, the computational tractability of simulating the model, or of otherwise computing its predicted outcome, has become an important consideration. But, beyond simulation as such, computational complexity can, via lower bounds, also offer a window on the other side of the above trade-off. The strong version of the Church-Turing Thesis, believed widely enough by modern computer science to be considered all but axiomatic[1], states that effectively that no physical system can perform computation asymptotically much more efficiently than a (probabilistic) Turing Machine, or a computer as we know it. So, if a model looks like a prima facie reasonable representation

[1] except perhaps for a still-controversial caveat about quantum computation

of its intended application domain, but is then found to be computationally intractable to simulate, how can we claim it to be realistic? For if the real world (in our case, the market, or the network of autonomous entities) is indeed behaving like the model predicts, it is then "computing" a problem more efficiently than a computer. Thus, the Church-Turing Thesis casts immediate doubt on just how well this model represents the real system.

It is with this two-pronged motivation that theoretical computer scientists have recently turned their attention to game theory, in hope of either discovering algorithms for analyzing game theoretical models, for both games that arise in other areas of computer science and games at large; or of proving lower bounds on their tractability, as a tool for critiquing the plausibility of the models.

Game theory's earliest and simplest model is the normal-form game. Each of m players has a finite set of *actions* (or *strategies*), S_i. When every player selects an action from his set, each player i's *utility* (or *payoff*) is a function of everyone's choice of action. A *strategy profile* (combination of everyone's strategies) where no one player can unilaterally change his strategy and improve his payoff is called a *pure Nash equilibrium*. While pure Nash equilibria aren't guaranteed to exist, extending the model to allow *mixed strategies* — each player picks a distribution over his strategy set rather than a single strategy, and "improvement" is measured based on expected payoff — produces *mixed Nash equilibria*.

Pure and mixed Nash equilibria are game theory's earliest and most prominent *solution concepts*, that is, predictions of game outcome. Nash's seminal 1950 paper [Nas50], the first major study of normal-form games in full generality, a model that by far subsumed the prior study of the restricted case of zero-sum games [NM44] and specific economic models [Cou38], proved that mixed Nash equilibria exist in all games. This paper is now considered one of the founding works of modern game theory [Mye99], and Nash equilibria, both mixed and pure, are the standard against which new solution concepts are first compared. They are thus the natural place to begin the study of computational complexity of selfish behavior.

While heuristics for computing mixed Nash in practice date back to the Lemke-Howson algorithm from 1964 [vS02], they come without a good upper bound on the running time, with Lemke-Howson in particular even proven to have exponential worst-case behavior [SvS03]. The first major result in understanding the worst-case complexity of games was a pair of recent breakthroughs [DGP06; CD06], where finding a mixed Nash equilibrum has been shown to be PPAD-complete [Pap94], with the subsequent research thus directed at approximation algorithms [LMM03; DMP06; KPS06; CDT06; DMP06; FNS]. Pure Nash

equilibria do not present a computational problem if the utilities are given as explicit tables, which are exponential in the number of players, since evaluating each cell in the table (each strategy profile) to see if it's a pure Nash is easy. On the other hand, in most games encountered in practice, the utilities are represented succinctly. In succinct games, even under reasonable constraints, it becomes intractable to even detect whether a pure Nash exists (see, e.g., [GGS03]). Interestingly, succinct representations don't make mixed Nash equilibria any harder to find — for all major classes of succinct games, finding a mixed Nash is still in PPAD [DFP06], and thus no harder than in explicit normal form games.

With equilibrium states intractable to find, or even absent, as with pure Nash, the question of "what actually happens when the game is played?" looms large, since the agents consistently and quickly reaching equilibrium would violate the Church-Turing Thesis.

In this dissertation, we consider complexity issues pertaining to the *dynamics* of agent behavior, limiting ourselves to pure strategies for simplicity. Hereafter, "Nash equilibria", "strategies", and the like are, when unqualified, used as shorthand for *pure* Nash equilibria, *pure* strategies, etc. The natural starting point is the *Nash best response dynamics*, the state space of strategy profiles, which the system traverses by agents, one at a time, making unilateral moves to their current optimal strategy. This model is the "native habitat" of Pure Nash equilibria, which arise naturally as the fixed points, i.e. sink nodes, of the resulting directed graph. The restriction to sequential moves is reasonable not only for the sake of simplicity, but also because quick changes in strategy and instant propagation of information about the changes made by others are the reality in many modern economic and computer systems.

In Chapter 2, we consider what happens when there's a guarantee of *eventual* convergence of the best-response dynamics to pure Nash equilibria. There is a famous and

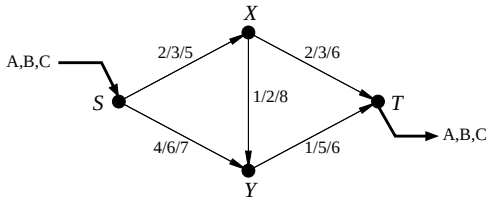

Figure 1.1: A network congestion game: three players are picking paths from S to T. Each edge is labeled with per-player delays when used by 1, 2, and 3 players.

well-studied class of games (and, in fact, one with obvious affinity to computer networks) that are not only guaranteed to have pure Nash equilibria, but are guaranteed to eventually converge to them: the *congestion games*. Figure 1.1 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 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 [Ros73], Rosenthal proves that, in any such game, the Nash dynamics converges, and hence the game has pure Nash equilibria. The proof, outlined in Section 2.1, 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, and are the best-response dynamics guaranteed to converge to one in polynomial time?

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* [JPY88], which means intuitively "as hard to compute as any object whose existence is guaranteed by a potential function" (see Section 2.1 for the precise definition).

Our proof has, as a corollary, strong consequences for the convergence of the Nash dynamics. It guarantees the existence of examples with exponentially long convergence times to the nearest pure Nash, as well as the PSPACE-completeness of finding some pure Nash equilibrium that the dynamics might converge to from a specified starting state. The completeness proof is complicated, as it requires the reworking of the reduction, due to [SY91], 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), the lower bounds on the dynamics hold even in the symmetric case.

What other games can be guaranteed to converge to Nash equilibria by potential functions? Monderer and Shapley [MS96] 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 $\phi(s') - \phi(s)$. However, the requirement that the two differences be the same is far too strict, as 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 an *ordinal potential function*. In Section 2.3, we establish a different kind of a connection between computational complexity and this game theoretic concept. It turns out that, under this relaxed definition, the space of "ordinal potential games" is much richer, essentially encompassing all of the complexity class PLS: any problem in PLS can be represented as a game whose pure equilibria are guaranteed to exist by a potential function argument.

Since pure Nash equilibria are not universally present, any general solution concept claiming to be a behavioral prediction must also account for games which don't have any. In Chapter 3, we consider *sink equilibria*, an interesting approach to analyzing the dynamics of games without Nash equilibria, proposed and pursued recently [GMV05] in the context of the "price of anarchy". These are an extension of pure Nash equilibria, in cases where they do not exist, to their natural dynamic generalization: *the ergodic states of the Nash dynamics*. That is, if the Nash dynamics graph has no sink nodes, i.e. pure Nash equilibria, we consider other, larger sink connected components of this graph, and postulate that a distribution on them is the desired solution. Two alternatives emerge: (1) We could focus on one sink component and consider its steady-state distribution of the corresponding Markov chain, assuming that the chain starts there, or (2) consider the steady-state distribution of the Markov chain associated with the Nash dynamics, perhaps assuming a uniform initial distribution (a natural alternative which, to our knowledge, has not been pursued explicitly).

We find that the sink equilibria approach is hindered by devastating complexity-theoretic obstacles. Even though the distribution (and the expected payoffs) of sink equilibria are easy to calculate for normal-form games, it is PSPACE-complete to say anything nontrivial about them in succinct games (Theorem 6). Our proof for the case of graphical games is a novel simulation of a space-bounded Turing machine by the Nash dynamics of such a game.

Next, in Chapter 4, we examine a very practical application domain where the best-response Nash dynamics are a particularly realistic model of actual agent behavior.

Using tools similar to those developed for sink equilibria, we derive an important (and unexpected) applied result related to *BGP oscillations* in Internet inter-domain routing. *Autonomous systems* (ASes) comprising the Internet route traffic originating in their domain or in the domains of other ASes using a proverbially complex protocol called *the Border Gateway Protocol (BGP)* [RL95; Ste98]. BGP allows each AS to specify arbitrary preferences on the paths to a destination, and to selectively offer such paths to others. Possibly because of the complexity of these preferences and offers (reflecting obscure agreements and opaque interests), BGP is known to occasionally cause *oscillations* in the Internet, that is, lengthy, drawn-out sequences of route reassignments by many AS's. This is a very disruptive, and consequently well-studied, phenomenon [GW97; LMJ98; VGE96; MGWR01]. A formal approach to BGP oscillations was initiated by [GW97]; they defined the *stable paths problem* as an abstraction of the phenomenon. They defined *safety* as the absence of oscillatory possibilities, and gave a necessary condition for safety (the presence of a stable path configuration, known to be NP-complete to test) and a sufficient condition for safety (the absence of a *dispute wheel,* a set of circular preferences), which is coNP-complete to test. But no necessary and sufficient condition had been known.

We point out that *the BGP oscillation problem is in fact also a problem of Nash dynamics* in a new kind of succinct game. The ASes are players, strategies correspond to choice of a next hop, and BGP preferences define (in a succinct way) the utility of the resulting graph for each of the players. Safety is tantamount to the game having *only* pure Nash equilibria, and no other sink components with two or more states. Our major practical contribution is that *BGP safety is PSPACE-complete* (Theorem 7), which resolves the complexity of this important question in networking. The reduction is involved and novel (and has to be quite different from that of Theorem 6).

Incidentally, on an aesthetic level, it is amusing to note that the observed BGP oscillations among a few ASes are known to often go on for hours or days. Exponentially long computation confined in a limited space is the hallmark of PSPACE-completeness, so this is a hint that the formal worst-case intractability proven here may occasionally be echoed in actual observed behavior.

Are sink equilibria and other variants of steady states in Nash dynamics natural and compelling solution concepts? The main shortcoming of such concepts is that they postulate complete lack of strategic behavior by the players, beyond a desire for local improvement. In the last part of this dissertation, we take a step back from best-response

dynamics, and explore several models of higher-level strategies where the players are aware of the game's dynamics beyond their immediate payoffs. Our objective here is to find models which allow higher player rationality, without detracting from the benefits sink equilibria have to offer — their tractability in normal form games, and their universality.

A game will have many sink components, and the payoffs will differ wildly from one component to the other. It would make perfect sense for a player to be strategic in the beginning (transient phase) of the game, sacrificing short-term gains so as to land on her favorite sink. How can we model such strategic behavior?

Allowing full strategic behavior brings us to the time-honored and well-studied — if somewhat inconclusively so — area of *repeated games* [Sor97; PY94; LS03], historically the first domain of interaction between complexity and game theory. The basic problem with this is that repeated games allow and predict strategic behavior that can be extremely sophisticated ([PY94]; see [FS98] for an insightful discussion of the kinds of strategic behavior one should expect on the Internet). In the interest of exploring the effect of limited strategic behavior in Nash dynamics, we consider *unit recall games*. In such games, players move simultaneously, and each player's next strategy choice depends only on the current state (strategy choices by everybody). A player's strategy is thus a finite automaton whose states are the player's own strategies, and whose transitions are labeled by the strategy combinations of everybody else. Once everyone adopts such a strategy, the Nash dynamics becomes deterministic and ends up in a cycle; the payoff is the time-average payoff of the states on the cycle.

Does the unit recall version of every game (that is, the game in which the available strategies are all possible unit recall automata) have a pure Nash equilibrium? The answer, at least for bimatrix games, is "almost": we show (Theorem 8) that a random bimatrix game has such an equilibrium with probability approaching 1 as the number of strategies grows. Furthermore, w.h.p., such an equilibrium is easy to find in polynomial time.

However, somewhat surprisingly, these equilibria aren't completely universal: even the extremely simple *matching pennies game* has no such equilibrium (Proposition 4, proven by a computer analysis of the 32×32 game). Then, how hard is it to tell whether a game has a unit-recall equilibrium in the worst case? The problem is, a priori, in the complexity class $\Sigma_2 P$ (it asks whether there exists a combination of automata such that all defection-automata are unfavorable). We conjecture that it is in P. *And we are half-way there*: we can prove, by a reduction to the problem of finding in a graph the cycle with the smallest

average edge length, that the best-response problem for such games is in P, and therefore the overall problem is in NP.

Another cause for optimism is the behavior of a relaxation of the unit recall equilibrium obtained from the following observation: Going from one strategy of a unit recall game to another requires changing a large number of transitions in the automaton/strategy; what if we allow changing *just one transition* at a time?

This yields a novel kind of equilibrium that we call *componentwise equilibrium*, a generalization of pure Nash.[2] Thinking of the strategies/automata as vectors of transitions, we can define an equilibrium concept by only allowing defections that change a single component. We show that *every multi-player game has a componentwise unit recall equilibrium (CURE)*. That is, in any game there are finite automata with the strategies of each player as state space such that no player can increase her long-term expected payoff by changing one transition (Theorem 10). In fact, we can find such an equilibrium in polynomial time, which puts it into the rarefied club of equilibria that are both guaranteed to exist and easy to find in normal-form games, sink equilibria and correlated equilibria [Pap05] being the only other members we're aware of. In view of our stated goal of seeking tractable, universal characterizations of game dynamics, these results suggest that the CURE concept may merit further theoretical attention, and that its more appealing subset, the unit-recall equilibrium, may also be tractable.

Lastly, in section 5.3 we consider what happens when, instead of committing to their full "strategy automata" as above, the players can look ahead into the future, decide that the Nash dynamics cycle that lies ahead won't be better for them on average than the state they are at right now, and stop early, instead of continuing to make best-response moves. The same reasoning can then be applied recursively by the players, one at a time, to "roll back" the original, potentially non-terminating, walk through the best-response dynamics to a single fixed state from where no one wants to make best-response moves due to what lies ahead for them if they do. This equilibrium concept, which we call *lasso equilibrium* after the shape of the walk in the 2-player case, may turn out to be a useful tool for analyzing games where several other moves must happen before the player feels the

[2] It is also an instance of a more general, and very promising, idea that extends the pure Nash equilibrium: Suppose that the strategies of each player are equipped with a cost matrix, and the player defects from a strategy to another only if the payoff difference is bigger than the cost; this is an idea also treated in [BSKK06]. In the case of componentwise equilibria, the cost is zero if the two strategies differ in one component, and infinity otherwise.

actual impact of the move he just made. As with unit recall equilibria, we show that, with high probability lasso equilibria exist and are easy to find in random two-player normal-form games.

Bibliography

Chapter 2 is based on work that was presented at the ACM Symposium on the Theory of Computing, 2004 [FPT04]. Chapter 3 and parts of chapters 4 and 5 are based on work that was presented at the SIAM Symposium on Discrete Algorithms, 2008 [FP08].

Chapter 2

Convergence to Pure Nash: Congestion Games

2.1 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 [Nas50] 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.*

Proof 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, [Ros73]) *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 [JPY88] 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 [JPY88], many local search problems were shown PLS-complete, including weighted versions of satisfiability, aspects of graph bisection, and the traveling salesman problem [Kre89; SY91; Pap92]. 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 [OPS04] 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.

2.2 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 $\phi(s)$. □

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.

Proof 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 [SY91].

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.

It should be noted that, since the publication of this result in [FPT04], a followup paper by Ackermann, et al, [ARV06] gave a much more elegant and general proof of part (iii) of the theorem, based on matroid theory. We thus only present an outline of our original proof 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 [SY91]. We need three new kinds of clauses besides POSNAE3FLIP to replace clusters of POSNAE3FLIP clauses of [SY91] 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 list 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.

Proof 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

[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

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.*

Proof sketch:

We show that the reduction in [SY91] from CIRCUITFLIP to POSNAE3FLIP yields instances of the latter which, once cast as XPNAE3FLIP, always have a valid witness.

The reduction in [SY91] 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 2.1 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 [SY91] 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 2.2. Notation of the form $\{$(sequence of clauses with k 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 $(\alpha_1^1)_{\mathrm{t}}$.

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 of the proof of the lemma, and this completes our outline of the proof of theorem 3. □ □

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,*

[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

Figure 2.1: Clauses in the PLS-reduction from CIRCUITFLIP to POSNAE3SAT in [SY91].

3D. $\forall k, i$:

- c1. $\psi^1_{k,i}$, $\alpha^1_{k,i}$
- c2. $\psi^2_{k,i}$, $\alpha^2_{k,i}$
- c3. $\psi^3_{k,i}$, $\gamma^1_{k,i}$
- c4. $\psi^4_{k,i}$, $\gamma^2_{k,i}$
- c5. $\psi^5_{k,i}$, $\beta^3_{k,i}$
- c6. $\psi^6_{k,i}$, $\omega_{k,i}$
- c7. $\psi^1_{k,i}$, $y_{k,i-1}$
- c8. $\psi^2_{k,i}$, $y_{k,i-1}$
- c9. $\psi^3_{k,i}$, $y_{k,i-1}$
- c10. $\psi^4_{k,i}$, $y_{k,i-1}$
- c11. $\psi^5_{k,i}$, $y_{k,i-1}$
- c12. $\psi^6_{k,i}$, $y_{k,i-1}$
- c13. $\zeta^1_{k,i}$, $\beta^1_{k,i}$
- c14. $\zeta^2_{k,i}$, $\beta^2_{k,i}$
- c15. $\zeta^3_{k,i}$, $\delta^1_{k,i}$
- c16. $\zeta^4_{k,i}$, $\delta^2_{k,i}$
- c17. $\zeta^5_{k,i}$, $\gamma^3_{k,i}$
- c18. $\zeta^1_{k,i}$, $z_{k,i-1}$
- c19. $\zeta^2_{k,i}$, $z_{k,i-1}$
- c20. $\zeta^3_{k,i}$, $z_{k,i-1}$
- c21. $\zeta^4_{k,i}$, $z_{k,i-1}$
- c22. $\zeta^5_{k,i}$, $z_{k,i-1}$

3E. $\forall k, i$:

- c1. $\rho_{k,i}$, $\alpha^1_{k,i}$
- c2. $\rho_{k,i}$, $\alpha^2_{k,i}$
- 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^1_k, 1
- c2. $\hat{d}_k$, θ^2_k, 0
- c3. w_k, θ^1_k, η_k
- c4. w_k, θ^2_k, η_k
- c5. v_k, η_k
- c6. θ^1_k, η_k
- c7. θ^2_k, η_k

7. $\forall k$

- c1. μ^1_k, v_k, w_k
- c2. $\hat{\mu}^2_k$, v_k, w_k
- c3. μ^2_k, $\hat{\mu}^2_k$
- c4. μ^1_k, μ^2_k, e_k
- c5. e_k, $\hat{e}_k$
- c6. e_k, 0
- c7. μ^1_k, 0
- c8. $\hat{\mu}^2_k$, 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 2.1: Clauses in the PLS-reduction from CIRCUITFLIP to POSNAE3SAT in [SY91] (continued)

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)

Figure 2.2: Per-variable ordered lists that comprise the valid witness.

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 2.2: Per-variable ordered lists that comprise the valid witness (continued)

and (b) the problem, given a state s, find a Nash equilibrium reachable from s is PSPACE-complete.

Proof sketch: Our reductions preserve these properties of POSNAE3FLIP [SY91]. □

2.3 Ordinal 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 a *cardinal 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 [MS96] establishes the following disappointing fact (as restated in [VBvM$^+$99]):

Theorem 4 ([MS96]) *Any cardinal potential game is isomorphic to a congestion game.*

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

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 $\text{sgn}(\sum_j s_i \cdot s_j \cdot w_{ij})$, 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 [MS96]. What is going on?

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 actually requires only that $\text{sgn}(\phi(s') - \phi(s)) = \text{sgn}(u_i(s') - u_i(s))$. Let *ordinal potential games* be games that have ordinal 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 ordinal 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 5 *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.*

Proof 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. □

2.4 Discussion and Open Problems

What other games are guaranteed to have pure Nash equilibria? Vetta identifies in [Vet02] 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 [FLM+03] are another example, and so are congestion games with *subjective delays* played on a network of parallel edges [Mil96]. 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 [FKK+02] 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 $S_1, \ldots, S_m$ of the set of players to the m edges, and consider the multi-set of numbers $\mu(s) = \{d_{e_1}(S_1), \ldots, d_{e_m}(S_m)\}$. 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 $\mu(s)$. □

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 exponential 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.

Chapter 3

Sink Equilibria

3.1 Definitions

In a *normal form game* G we have $n \geq 2$ players $1, 2, \ldots, n$; for each player i we have a finite set of strategies S_i. We denote by $\mathcal{S}$ the *set of pure strategy profiles*, or *states*, the Cartesian product $S_1 \times S_2 \times \cdots \times S_n$; by $\mathcal{S}_{-i}$ we denote the product of all strategy sets except S_i. Finally, for each player i we have a *utility function* u^i mapping S to the integers.

Formally, we define the *Nash (best-response) dynamics* of a game as a directed graph with $\mathcal{S}$ as its set of vertices, and an edge (s, s') if s and s' agree on all components except for one, say the ith, and $u^i(s') > u^i(s)$. The *strict Nash dynamics* is a subgraph of the Nash dynamics, where an edge s, s' signifies, in addition, that i's strategy in s' is i's *best response* to s, that is, one of the strategies in S_i which, if substituted for the ith component of s, yields the largest possible utility. (We shall henceforth specify strict or not strict Nash dynamics only in cases that do not apply to both kinds.) A *pure Nash equilibrium* of a game is a state that is a *sink* of the Nash dynamics.

Consider now the strongly connected components of the Nash dynamics, and among those, one that has no outgoing edges. Such a component is called a *sink equilibrium* of the game. The Nash dynamics of a graph define a Markov chain, by assigning equal probabilities to all outgoing edges.

3.1.1 Normal form versus graphical games

The following are then computational problems[1] of interest:

- BEST (WORST) SINK: Given a game, what is the highest (or lowest) expected utility for a given player in the steady-state distribution of a sink equilibrium?
- IN A SINK: Does a given state s belong to any sink equilibrium?

The following result only involves depth-first search and matrix inversion:

Proposition 3 *Given a game in normal form, the problems* BEST (WORST) SINK *and* IN A SINK *can be solved in polynomial time*

An n-player game requires an exponential, in n, number of bits to be represented; hence, the only computationally meaningful ways of representing multi-player games must be succinct. A *graphical game* is a particularly useful succinct way of representing a game. It is defined in terms of a graph $G = ([n], E)$ whose set of vertices is the set of players. The utility function u^i is now a function mapping $\mathcal{S}_{N(i)}$ to the integers, where $\mathcal{S}_{N(i)}$ is the Cartesian product of the strategy sets of player i and of all players that are adjacent to i in G.

Our main result in this chapter is this:

Theorem 6 IN A SINK *for graphical games is PSPACE-complete.*

Here and in the BGP result in Chapter 4, we seek to use sink equilibria of a graphical game to capture the halting behavior of a special space-bounded Turing machine. We set up the intermediate problem in section 3.2 and present the main reduction in 3.3.

3.2 Intermediate automaton problem

We define an intermediate construction, a *local-dependence resetting cycle machine* (LRCM), which moves continously and unidirectionally around a circular tape, and updates tape cells based on its internal state and the state of *nearby* cells, but not the current one. This mirrors the game constraint that a player cannot decide on its new strategy

[1] More precisely, these are families of problems, parametrized by one's choice of game representation

based on its current strategy. The reduction to a graphical game then creates a graph which locally models the tape with two players per cell (a "tape cell" player and a "state" player), and $O(1)$ edges to players modeling nearby cells. We design the utilities so that the Nash dynamics, starting at the canonical starting state, have a "ripple" of activity move around the cycle until the corresponding computation halts. To guarantee that the canonical starting state of a game is in a large sink equilibrium iff the machine does *not* halt, we have the LRCM reset to its starting state after it takes enough steps to have exhausted all possible computation states and looped somewhere.

Formally, let an LRCM be a tuple of a state space, a designated starting state, a tape alphabet, the length of its circular tape, and a transition function mapping the machine state and the contents of tape cells $i-2$, $i-1$, and $i+2$, where i is the current tape cell, to the new state and the new content of the current cell, or a "halt" command: $C = (Q, q_0 \in Q, \Gamma, t, f : Q \times \Gamma^3 \to (Q \times \Gamma) \cup \{\text{halt}\})$ (we'll use (f_q, f_T) for components of f's output). We require LRCMs to obey two additional semantic constraints:

1. (Resetting constraint) When started on a blank tape, it must either halt or return to the same configuration (state q_0 and a blank tape) in finite time.

2. (Self-loop-free constraint) There may not be any self-loops in the finite-state control, i.e. whenever $f(q, \ldots) = (q', \ldots)$, $q \neq q'$.

C is said to halt on starting tape state $(T_0, \ldots, T_t) \in \Gamma^t$ if the following loop halts (using $T_{N(i)}$ as shorthand for $(T_{i-2}, T_{i-1}, T_{i+2})$):

1: $i \leftarrow 0$
2: **while** $f(q, T_{N(i)}) \neq$ halt **do**
3: $\quad (q, T_i) \leftarrow f(q, T_{N(i)})$
4: $\quad i \leftarrow i + 1$
5: **end while**

Lemma 3 *It is PSPACE-complete to tell whether an LRCM C will halt on a starting tape state.*

Proof:

We start with the generic PSPACE-complete problem, the space-bounded halting problem for TMs. We first reduce an instance of this problem (a Turing machine M, an input x, and a tape bound t), to the halting problem for a TM variant $M' = (Q', \Sigma', \Gamma', \delta', q'_0, q'_a, q'_r)$, with a circular tape of length t' which, independently of its own input, simulates M on x with space bound t and halts iff M accepts. Additionally, we use $t \log |Q|$ extra cells of the tape to store a step counter that counts up to the total number of possible configurations of M, and, when the counter overflows, have the machine enter a special state which forces the entire tape, including the counter, to be reset to blank, moves the tape head to the starting position, and sets the control to q_0.

To map M' to a LRCM, we:

- simulate bidirectional movement by making the CM do an "idle round" around the tape after each step of M'
- store the direction of tape movement in the state during the idle round
- remove the transition function's dependence on the current tape cell by storing a copy of cell i's contents in cell $i + 2$
- double the state space by adding a "flip-flop" component which flips at each step to ensure the absence of control self-loops

Formally, let $\Gamma'' = \Gamma' \times \Gamma' \times \{0, 1\} \times \{0, 1\}$ and $Q'' = Q' \times \{\pm 1\} \times \{\text{active}, \text{idle}\} \times \{0, 1\}$. Use $T_i = (T_i^t, T_i^c, T_i^l, T_i^{cl})$ for the "this cell", "copy of cell $i - 2$", "last-change bit", and "copy of $i - 2$'s last-change bit" components of T_i; and $q = (q^q, q^d, q^i, q^f)$ for "state", "direction of movement", "idle indicator", and "flip-flop" components of the state q.

Then, let f be the following:

As above, we claim that if we start LRCM $C = (Q'', (q'_0, \text{active}, +1, 0), \Gamma'', t' \geq 3, f)$ on tape $T_i = (\llcorner, 0, \llcorner, 0)$ $\forall i$, it will halt iff M' halts, and otherwise will eventually return to state q_0 and the starting tape state, with each step of M' getting simulated by an action step of C.

An induction on the steps of C shows that, at the beginning of any step of C at cell i:

- $T_{k+2}^{c,cl} = T_k^{t,l}$ for at least all k other than $i - 1$ and $i - 2$.
- C's state is one of the following 4 configurations:

---

if $q^i = \text{active}$ **then**
  $(\tilde{q}^q, \tilde{T}^t, \tilde{q}^d) \leftarrow \delta'_t(q^q, T^c_{i+2})$ // "action step"
  if $\tilde{q}^q \in \{q'_a, q'_r\}$ **then**
    return halt
  end if
  $\tilde{T}^l \leftarrow 1$; $\tilde{q}^i = \text{idle}$
else
  $\tilde{T}^l \leftarrow 0$; $\tilde{T}^t \leftarrow T^c_{i+2}$; $\tilde{q}^{q,d} \leftarrow q^{q,d}$ // "idle step"
  if $T^l_{i+2} = 1 \wedge q^d = -1$ or $T^{cl}_{i+2} = 1 \wedge q^d = +1$ **then**
    $\tilde{q}^i = \text{active}$ // prepare to write in next cell
  end if
end if
$\tilde{T}^{c,cl} \leftarrow T^{t,l}_{i-2}$; $\tilde{q}^f \leftarrow \neg q^f$
return $(\tilde{q}, \tilde{T})$

---

1. Exactly 1 last-change bit T^l_k is on, $q^i = \text{idle}$. If $k = i + 1$ or $k = i$, q^d may only be $+1$. Corresponds to M' having just written to tape cell k, switched the state to q^q, and about to move in direction q^d

2. Only T^l_{i+1} is on, $q^i = \text{active}$, $q^d = -1$. Same M' state as in 1 above.

3. Only T^l_{i-1} and T^l_i are on, $q_i = \text{idle}$. Corresponds to M' having just written to $i - 1$, switched to state q^q, about to move by q^d (having moved by -1 on the previous step).

4. No last-change bit T^l_k is on, $q_i = \text{active}$, $q_d = +1$. Corresponds to M' having just written to $i - 1$, switched to state q^q, and about to move by $+1$.

- Each step of C which starts with $q^i = \text{active}$ changes the state q^q and the sequence $\{T^t_j\}$ to match the state and tape contents of the next step of M''s computation, hence causing C to halt iff M' halts.

Allowing a succinct representation of f as above, the reduction is clearly polynomial, hence yielding PSPACE-completeness. □

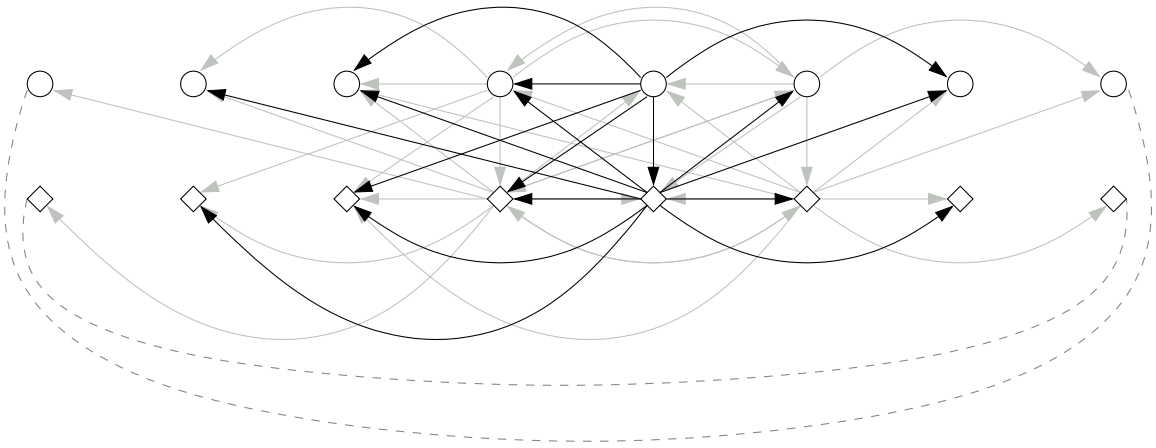

Figure 3.1: A segment of a graphical game gadget to simulate an LRCM.

3.3 From LRCMs to sink equilibria

Our reduction from an LRCM $C = (Q, q_0, \Gamma, t, f)$ and starting tape T will create a game $\mathcal{G}$ which, for each tape cell i, has a "cell" player C_i and a "state" player S_i, with the utilities of both depending on several nearby C_j and S_j players. We will arrange $\mathcal{G}$ so that its Nash dynamics follow the computation of C, with no players ever given an incentive to deviate from "canonical" game states, i.e. those that directly correspond to a configuration of C. Thus, the game state corresponding to the starting state of the LRCM will be in a sink equilibrium if and only if the LRCM does not halt.

The dependencies in the game follow the pattern in Figure 3.1. C_i players are shown as circles and S_i — as diamonds. The black links show the dependencies for a one pair of C_i and S_i nodes. The same pattern of links repeats for all t pairs of players (C_j, S_j), which are arranged in a closed loop.

The strategy set for each C_i is just Γ; the strategy set for S_i is $Q \cup \{\emptyset\}$. The utility function for S_i depends on the strategies of C_{i-3}, C_{i-2}, C_{i-1}, C_{i+1}, C_{i+2}, S_{i-3}, S_{i-2}, S_{i-1}, S_{i+1}, S_{i+2}, and itself; and that for C_i — on C_{i-2}, C_{i-1}, C_{i+2}, S_{i-2}, S_{i-1}, S_i, and itself. We set the utility function for S_i according to the rules in Table 3.1.

For C_i, we set $u^{C_i}(C_{i-2}, C_{i-1}, C_i, C_{i+2}, S_{i-2}, S_{i-1}, S_i)$ to 1 iff $S_{i-2} = \emptyset$, $\emptyset \neq S_i \neq S_{i-1} \neq \emptyset$, and $C_i = f_T(S_{i-1}, C_{i-2}, C_{i-1}, C_{i+2})$. For any other configuration of strategies, we set it to 0.

We start the game $\mathcal{G}$ with the tape players C_i playing the contents of the C's

C_{i-3}	C_{i-2}	C_{i-1}	C_{i+1}	C_{i+2}	S_{i-3}	S_{i-2}	S_{i-1}	S_{i+1}	S_{i+2}	S_i	u^{S_i}
*	*	*	*	*	$\emptyset$	$\emptyset$	q_1	q_2	$\emptyset$	q_2 else	1 0
V	W	$f_T(q_1, V, W, Y)$	Y	*	$\emptyset$	q_1	$q_2 \neq q_1$	$\emptyset$	$\emptyset$	q_2 else	1 0
*	W	X	*	Z	$\emptyset$	$\emptyset$	q_1	$\emptyset$	$\emptyset$	$f_q(q_1, W, X, Z) \neq$ halt else	1 0
*	*	*	*	*	$\emptyset$	$\emptyset$	$\emptyset$	q_1	$\emptyset$	*	0
else										$\emptyset$ else	1 0

Table 3.1: The utility function for S_i. Asterisks indicate don't-cares, q_1 and q_2 are arbitrary elements of Q (i.e. not $\emptyset$), and V, W, X, Y, Z are arbitrary elements of Γ.

starting tape, S_0 and S_{-1} playing q_0, and all the other S_i's playing $\emptyset$. This puts the game state in "phase" (iv) as shown in Table 3.2. By induction, it can be seen that, given that starting configuration:

- at each step of the Nash dynamics, the game state is in one of the four phases listed in the table,
- only the player whose strategy is marked by a square may have an incentive to deviate, and
- the game will cycle through states in phases (i)-(iv), simulating the computation of C, unless and until C halts, at which point the game will be in a pure Nash equilibrium.

Due to the resetting constraint on C, if C does not halt, $\mathcal{G}$ must return to its starting state after a finite number of steps. Since no other players have an incentive to deviate at any step, this cycle in $\mathcal{G}$'s state space forms a sink equilibrium. Conversely, if C does halt, there is a path from the starting configuration to the pure Nash equilibrium corresponding to the halted computation of C, thus assuring that the starting state of $\mathcal{G}$ is not part of a sink equilibrium.

Since the degree of $\mathcal{G}$'s graph is 10, we thus have that IN A SINK is PSPACE-complete. This concludes the proof of theorem 6.

As an easy corollary, by taking liberties with the scale of the utility functions, we get that the problems BEST (WORST) SINK for graphical games cannot be approximated at all.

phase	$C_{\dots}$	C_{i-3}	C_{i-2}	C_{i-1}	C_i	C_{i+1}	C_{i+2}	C_{i+3}	$C_{\dots}$
	$S_{\dots}$	S_{i-3}	S_{i-2}	S_{i-1}	S_i	S_{i+1}	S_{i+2}	S_{i+3}	$S_{\dots}$
(i)	$\cdots$	A	B	C	**D**	E	F	G	$\cdots$
	$\emptyset$	$\emptyset$	$\emptyset$	a	b	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
(ii)	$\cdots$	A	B	C	H	E	F	G	$\cdots$
	$\emptyset$	$\emptyset$	$\emptyset$	a	b	**$\emptyset$**	$\emptyset$	$\emptyset$	$\emptyset$
(iii)	$\cdots$	A	B	C	H	E	F	G	$\cdots$
	$\emptyset$	$\emptyset$	$\emptyset$	**a**	b	b	$\emptyset$	$\emptyset$	$\emptyset$
(iv)	$\cdots$	A	B	C	H	E	F	G	$\cdots$
	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	b	**b**	$\emptyset$	$\emptyset$	$\emptyset$
(i)	$\cdots$	A	B	C	H	**E**	F	G	$\cdots$
	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	b	c	$\emptyset$	$\emptyset$	$\emptyset$

Table 3.2: Canonical states of $\mathcal{G}$. Capital letters are tape symbols in Γ, lower-case letters are states in Q. $H = f_T(a, B, C, E)$, $c = f_q(b, C, H, F)$

3.4 Open Problems

We believe that sink equilibria are intractable in other general situations as well, for example in congestion games (with large, splitable flows).

The sink equilibria concept remains one important direction to explore in our search for characterizations of game dynamics. It would be interesting to consider strategic enhancements of this concept, besides the unit recall idea offered in section 5.1, in line with the realities of the Internet: Asynchronous players with different reaction speeds whose move depends not on the state, but on a rough (and perhaps delayed...) estimate of their utility.

Chapter 4

Stability in BGP Inter-Domain Routing

4.1 Modelling BGP dynamics: the Stable Paths Problem

We now turn our attention to a concrete and important engineering application of the Nash best-response dynamics: Internet inter-domain routing with the BGP protocol. While the real BGP is, by necessity, a complex, practical protocol, we restrict discussion to a mathematical model of it based on [GW97], which actually does not stray far from the true protocol while abstracting away many engineering details which are irrelevant to our results.

In our abstraction, a *BGP system* $\mathcal{B}$ is a *directed* graph $G = (V, E)$ whose nodes are called *autonomous systems* or *ASes*, except for $v_0 \in V$, called the *target node* (we assume all traffic is directed to it). Also, for each $v \in V - \{v_0\}$ there is a function λ^v, represented as a Boolean circuit, say, mapping all simple paths between v and v_0 to the integers. λ^v also assigns an integer preference value to $\perp$, the absence of a path (in which case all paths whose utility is less than that of $\perp$ can be considered "disallowed paths").

The BGP system's *state* is a *path assignment* $\pi(u)$ mapping node u to a path from u to v_0 (or the absence thereof). Given a state, let us define $C(\pi, u) = \{(u, \pi(v))|(u, v) \in E\}$ to be the list of paths u can choose from, starting with a link to a neighbor v, and continuing with the current path for v. A path assignment π is *stable* if, for all u, $\pi(u) = \arg\max_{P \in C(\pi,u)} \lambda^u(P)$. We'll abbreviate path assignments by showing just $\nu(u)$, the next

hop of the paths from u to v_0, i.e. $\pi(u) = (u, \nu(u), \nu(\nu(u)) \ldots, \nu^k(u) = v_0)$.

From a given state π, a node $a \in V$ can be *activated*, in which case a's path is updated with $\pi'(a) = \arg\max_{P \in C(\pi,a)} \lambda^a(P)$. That is, if there is a better option for a (a is "dissatisfied") then the best such option is adopted, otherwise, $\pi' = \pi$. We then say $\pi \xrightarrow{a} \pi'$. An infinite activation sequence is *fair* ([GW97]), if, for all $v \in V$ and $i \in \mathbb{Z}$, there is a $j > i$ such that $a_j = v$. A system is said to be *convergent* if for all fair sequences $(a_1, a_2, \ldots)$, there exists an x such that $\pi_0 \xrightarrow{a_1} \pi_1 \xrightarrow{a_2} \ldots \xrightarrow{a_x} \pi_x$ and π_x is stable.

Our BGP model follows closely the *stable paths problem (SPP)* model of [GW97], see also [GSW02]. One exception is that they provide an explicit list of ordered paths, instead of an algorithm for comparing paths. Our version is closer to the realities of both the formal BGP protocol specification, in which ASes are allowed to implement arbitrary preference functions, and real BGP implementations in routers, which somewhat limit the preference functions, but allow more freedom than just an explicit ordering of paths. Another important difference is that our model allows *ties* between paths; the real BGP actually does not (this is BGP's "Phase 2" tie-breaking as described in Sec. 9.1.2.1 of [RL95]). Our main lower bound proof below uses this possibility of ties. However, we can extend the result to not require ties; a sketch of this is given at the end of this section. A final difference is that the model of [GW97] is explicit about the asynchronous nature of the communication between the ASes by introducing message queues to the state, something that we view as an unnecessary complication that obscures the finitary, combinatorial nature of the problem.

4.2 The complexity of BGP safety

We are interested a computational problem which we call

BGP SAFETY: given a BGP system as above, is it convergent?

Characterizing safety has been the main goal of the literature on BGP oscillations ([GW97; LMJ98; VGE96; MGWR01], inter alia). In [GSW02] only a necessary condition, and another sufficient one, for safety are given, both NP-hard. We show below that the problem is PSPACE-complete.

One key observation is that *the BGP convergence problem is actually one about Nash dynamics.* Starting with a BGP system $\mathcal{B} = ((G, E), \lambda)$, consider a game $\mathcal{G}[\mathcal{B}]$ with a set $V - \{v_0\}$ of players, each with a strategy set equal to the set of outgoing edges from it.

Every pure strategy profile s defines now a subgraph G_s of G with outdegree one for each player. The utility of v is then defined as $u^v(s) = \lambda^v(\pi[v, G_s])$, the value of the unique path from v to v_0 in G_s, if such a path exists, and $\lambda^v(\perp)$ otherwise.

BGP SAFETY is hence just the question of whether the strict Nash dynamics of $\mathcal{G}[\mathcal{B}]$ is completely acyclic. Though the problem is similar in spirit to those about sink equilibria considered in Chapter 3, the situation here is much more specialized than with the graphical games result, and the proof of the following result is quite a bit more sophisticated (and very different) than the proof of Theorem 6.

Theorem 7 BGP SAFETY *is PSPACE-complete.*

Proof sketch: We start from the following computational problem:

STRING-OSCILLATIONproblem: Given a function $f : \Gamma^{t-2} \mapsto \Gamma \cup \{\text{halt}\}$ for some alphabet Γ and integer $t > 2$ (assume that the function is given as a Boolean circuit), is there an initial string $T_1 \cdots T_t \in \Gamma^T$ such that the following program does not halt? (Indices of T are understood modulo t, and $T_{-(i,i+1)}$ means $T_{i+2}T_{i+3} \cdots T_{i-1}$.)

1: $i \leftarrow 0$
2: **while** $f(T_{-(i,i+1)}) \neq$ halt **do**
3: $\quad T_i \leftarrow f(T_{-(i,i+1)})$
4: $\quad i \leftarrow i + 1$
5: **end while**

To show the STRING-OSCILLATIONproblem PSPACE-complete, we start, as in Theorem 6, from the problem of telling whether a linear-bounded Turing machine will halt if started on n blanks. We modify the Turing machine (in a way reminiscent of the construction in [GKMP06]) so that (1) it has a clock that is "syntactically integrated" in its operation (meaning, it is easy to check whether a configuration contains a legitimate clock state); (2) when the clock overflows, or if the machine is about to accept, it erases the tape, zeroes the clock, and restarts; (3) if it rejects, it halts; (4) therefore, the machine cycles if and only if it accepts the empty string. The details are messy and have to be done exactly right (and are omitted here).

We conclude that it is PSPACE-complete to tell if a Turing machine (with embedded clock) will cycle. The reduction from the cycling problem to STRING-OSCILLATION

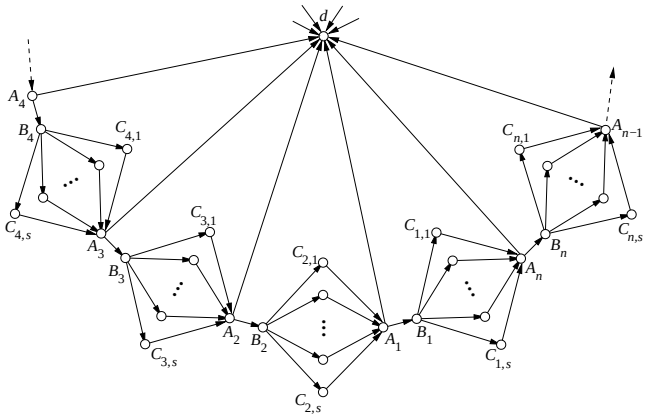

Figure 4.1: A BGP gadget to simulate String-Oscillation.

is not difficult, once it is clear that f can recognize legitimate configurations (and halt in every other case).

Starting from such a function f, we use the BGP system shown in Figure 4.1 to simulate the operation of the program above in such a way that, at any step, no more than 2 nodes are interested in changing their path in a way that keeps the system "alive", i.e. in a state from which it might still loop (oscillate) indefinitely. The system can oscillate iff there exists a starting configuration from which f never halts.

Given an instance $I = (\Gamma, n, f)$, with $|\Gamma| = s$ of String-Oscillation, we define the BGP system $\mathcal{B}(I) = ((V, E), \{\lambda^i\}$, where $V = \{d, A_i, B_i, C_{i,j} | i \in N = \{1, \ldots, n\}, j \in S = \{1, \ldots, s\}\}$ and $E = \{(A_i, d), (A_i, B_i), (B_i, C_{i,j}), (C_{i,j}, A_{i-1}) | i \in N, j \in S\}$ (operations on "tape" indices are implicitly modulo n). $d = v_0$ is the target node. Since $C_{i,j}$'s have only one outgoing link, they never update. B_i's path choices, i.e. its preference function, λ^{B_i}, implements f, while λ^{A_i} is set up to assure that updates happen in clockwise (increasing i) order.

Let $\mathcal{T}(t_x, \ldots, t_y)$ be shorthand for the path segment:

$$(C_{x,t_x}, A_{x-1}, B_{x-1}, C_{x-1,t_{x-1}}, \ldots, B_y, C_{y,t_{y-1}})$$

Note the reverse order of the tape indices. We can now define the preference functions by:

$$\lambda^{B_i}(P) = \begin{cases} 4 & \text{if } P = (B_i, \mathcal{T}(t_i, \ldots, t_{i+1}), A_i, d) \\ & \quad \text{and } f(t_{i+2}, \ldots, t_{i-1}) = t_i \\ 1 & \text{if } P = (B_i, \mathcal{T}(t_i, \ldots, t_{i+1}), A_i, d) \\ & \quad \text{and } f(t_{i+2}, \ldots, t_{i-1}) \neq t_i \\ 0 & \text{if } P = \bot \\ 2 & \text{else} \end{cases}$$

$$\lambda^{A_i}(P) = \begin{cases} 3 & \text{if } P = (A_i, d) \\ 1 & \text{if } P = (A_i, B_i, C_{i,j}, A_{i-1}, d) \\ 1 & \text{if } P = (A_i, B_i, \mathcal{T}(t_i, \ldots, t_{i+2}), A_{i+1}, d) \\ & \quad \text{and } f(t_{i+2}, \ldots, t_{i-1}) \in \{t_{j \neq i}, \text{halt}\} \\ 4 & \text{if } P = (A_i, B_i, \mathcal{T}(t_i, \ldots, t_{i+2}), A_{i+1}, d) \\ & \quad \text{and } f(t_{i+2}, \ldots, t_{i-1}) = t_i \\ 0 & \text{if } P = \bot \\ 2 & \text{else} \end{cases}$$

We call a state "alive" if either (a) exactly one A_i has $\nu(A_i) = d$, or (b) only $\nu(A_{i+1})$ and $\nu(A_i)$ are d and $\nu(B_i) = C_{i,f(T_{-(i,i+1)})}$, with f not returning "halt".

If a state is not alive, we can inductively show that no B_j will ever again have an incentive to switch (since it will never have a "clear view" to A_i) and no A_j will ever have an incentive to switch from d to B_j, so any fair activation sequence will lead to all A_j's switching to d, creating a stable configuration.

If a state is alive, its $\nu(B_j)$'s correspond to a configuration of f's tape, with the loop at cell i.

Inductively, after any activation sequence $(a_1, \ldots, a_x)$, we see that the system can only be in one of these types of live states as it simulates f:

1. For all $j \neq i$, $\nu(A_j) = B_j$, $\nu(B_j) = C_{i,T_j}$; $\nu(A_i) = d$, and $\nu(B_i) = C_{i,f(T_{-(i,i+1)})}$. Only A_j for $j \notin \{i, i-1\}$ are dissatisfied. Updating A_{i+1} yields state type 3; updating any of the others creates a dead state.

2. For all $j \neq i$, $\nu(A_j) = B_j$, $\nu(B_j) = C_{i,T_j}$; $\nu(A_i) = d$, and $\nu(B_i) \neq C_{i,f(T_{-(i,i+1)})}$. Only A_j for $j \notin \{i, i-1\}$ and B_i are dissatisfied. Updating B_i yields state type 1. Updating any of the A_j's yields a dead state.
3. For all $j \notin \{i, i+1\}$, $\nu(A_j) = B_j$, $\nu(B_j) = C_{i,T_j}$; $\nu(A_i) = \nu(A_{i+1}) = d$, and $\nu(B_i) = C_{i,f(T_{-(i,i+1)})}$, with f not returning "halt". Only A_j, $j \neq i+1$ are dissatisfied. Updating A_i "advances" the simulation of C to the next step, and updates to state type 2, or, if the next step of C doesn't change the cell value, 1. Updating any of the other A_j's yields a dead state.

Thus, if I does not halt, the activation sequence $(B_1, A_2, A_1, B_2, A_3, \ldots)$ will make this system cycle between states 2, 1, and 3 (perhaps occasionally skipping 2 in the sequence when a type-3 state updates directly to type-1). If I does halt, a fair activation sequence will force the system to enter a dead and then reach a stable state in finite time. With the λ's clearly succinctly representable, we have that BGP SAFETY is PSPACE-complete. □

The above setup relies on B_j's maintaining their state by having their preference function remain "indifferent" when the "tape head" (the A_i-to-d link) is elsewhere; this is the sole element above that ever requires "ties for first place" in the preference functions, which BGP disallows. We can eliminate this by introducing an extra state D_i for each B_i, with links from B_i to D_i and from D_i directly to d. By setting B_i's preference function to 3 for (B_i, D_i, d), and making A_i and $B_{j\neq i}$ dislike paths through D_i, we would introduce another category of dead states, with the same implications of forthcoming termination as above. We omit here the details of the proof that the reduction still works.

4.3 BGP open problems

Can our complexity result on BGP oscillations be extended to the case in which the utilities are given explicitly as a preference order of paths, as in the model of [GW97]? We cannot see how to prove this, but we expect that this problem is, indeed, also PSPACE-complete.

There is also the question of how to model more complex real-world incentives behind the actions of ISPs, which, externally to the protocol-mandated local preferences, are actually driven by revenue from their potential customers and payments to their upstream [GHJ+08]. In section 5.3, we propose a new equilibrium concept that may be a good

approach for modelling such a system.

Chapter 5

Strategizing About Dynamics

5.1 Unit Recall Equilibria

Sink equilibria restrict the strategic behavior by players to myopic local improvement. On the other hand, unrestricted strategic play on the states is a very involved subject [Sor97]. The following is an interesting compromise between the two extremes, a minimal instance of bounded-recall games as studied by the game theory community [Ney85; Aum00]:

Fix a game. A *unit recall strategy* by player i is a finite state automaton that has S_i as its state space and $\mathcal{S}_{-i}$ as its alphabet. It specifies a starting strategy, and a next strategy for every combination of plays by the other players. Notice that, assuming that the players act synchronously, a set of unit recall strategies, one for each player, defines a function from $\mathcal{S}$ to itself, as well as a start state, and thus an infinite walk from the start state ending up in a cycle. The payoff of this combination of automata is then the average utility in this cycle.

Notice that the above concepts allow us to define, for each game, a game by the same players whose strategies are unit recall strategies. We call the pure Nash equilibria of this game *unit recall equilibria* (UREs). It would be very exciting if every game had a unit recall equilibrium. Unfortunately, this is not the case. The following can be proved by exhaustion:

Proposition 4 *The game of matching pennies (with $n = 2$, $S_1 = S_2 = \{0, 1\}$, and $u^i(a, b) = (-1)^{i+a+b \bmod 2}$) has no unit recall equilibria.*

However, unit-recall equilibria are still *common*. Take a random 2-player $m \times m$ normal-form game. Pure Nash equilibria are known to exist with probability approaching $1 - 1/e$ as the number of strategies tends to infinity [GGN68]. The probability of a URE existing, however, approaches 1:

Theorem 8 *In an $m \times m$ bimatrix game with payoffs chosen independently from an arbitrary distribution, the probability that a unit recall equilibrium exists and can be found in polynomial time is $1 - 1/\Omega(m^{1-\varepsilon})$, for all $\varepsilon > 0$.*

Proof: We show that a particular kind of URE exists w.h.p.: a *simple URE* in a bimatrix game with payoff matrices (R, C) is a pair of starting strategies (r, c) and a pair of a "punishment column" $c_p \neq c$ and "punishment row" $r_p \neq r$ for the row and the column player, respectively, such that $R_{k,c_p} \leq R_{r,c}$ and $C_{r_p,k} \leq C_{r,c}$ for all k. The unit-recall strategies are set so that, from (r, c), both players continue playing the same strategy, and from any other state, the row player defects to r_p to punish the column player, and vice versa. Such equilibria can be found in polynomial time by exhaustion.

Set, with foresight, $f(m) = \varepsilon(m-1)\log_3 m$, and pick the top $f(m)$ entries from all of R, and the top $f(m)$ entries from all of C (break ties uniformly at random). Let I^R and I^C be 0-1 matrices which have 1's wherever those top $f(m)$ entries occur, and 0 elsewhere. For a simple URE to exist, it is sufficient to have (i) event M: there exist (r, c) such $I^R_{r,c} = I^C_{r,c} = 1$, and (ii) event P: there exist some r_p and c_p such that I^R_{k,c_p} and $I^C_{r_p,k}$ are 0 for all k.

Since the entries of R and C are chosen independently, I^R and I^C are chosen uniformly at random from the set of 0-1 $m \times m$ matrices with exactly $f(m)$ 1's. We will also use the fact that $(1 - 1/x)^x$ monotonically approaches $1/e$ from below as x grows, and thus, for all $x \geq 6$, $1/3 < (1 - 1/x)^x < 1/e$.

We bound the probability of M via a birthday-paradox argument:

$$\Pr[\overline{M}] = \frac{\binom{m^2-f(m)}{f(m)}}{\binom{m^2}{f(m)}} \tag{5.1}$$

$$= \prod_{i=0}^{f(m)-1} \left(1 - \frac{f(m)}{m^2-i}\right) \tag{5.2}$$

$$\leq \left(1 - \frac{f(m)}{m^2}\right)^{f(m)} \tag{5.3}$$

$$\leq e^{-f(m)^2/m^2} \tag{5.4}$$

$$\leq m^{-\Theta(1)(\log m)((m-1)^2/m^2)} \leq m^{-\Theta(\log m)} \tag{5.5}$$

The more involved part is the following lemma:

Lemma 4 *The probability that R has no empty columns is at most $\frac{1}{m^{1-\varepsilon}}$.*

The same lemma applies, by symmetry, for C.

Proof: Let X_i be the indicator of column i of R being empty, with $X = \sum_i X_i$. For each i, we have:

$$\Pr[X_i = 1] = \frac{\binom{m^2-m}{f(m)}}{\binom{m^2}{f(m)}} \tag{5.6}$$

By linearity, we have:

$$\mathrm{E}[X] = m\frac{\binom{m^2-m}{f(m)}}{\binom{m^2}{f(m)}} \tag{5.7}$$

$$= m\prod_{i=0}^{m-1} \left(1 - \frac{f(m)}{m^2-i}\right) \tag{5.8}$$

$$\geq m\left(1 - \frac{f(m)}{m^2-m}\right)^{m} \tag{5.9}$$

$$\geq m\left(\frac{1}{3}\right)^{f(m)/(m-1)} \tag{5.10}$$

$$\geq m^{1-c} \tag{5.11}$$

$$\tag{5.12}$$

Since $X_i \cdot X_j$ is just the indicator that columns i and j are empty, the variance is bounded

by:

$$\mathrm{Var}[X] = \sum_i \mathrm{E}[X_i^2] + \sum_{i,j} \mathrm{E}[X_i X_j] - \mathrm{E}[X]^2 \tag{5.13}$$

$$= \mathrm{E}[X] + m(m-1)\frac{\binom{m^2-2m}{f(m)}}{\binom{m^2}{f(m)}} - m^2\frac{\binom{m^2-m}{f(m)}^2}{\binom{m^2}{f(m)}^2} \tag{5.14}$$

$$\leq \mathrm{E}[X] + \frac{m^2}{\binom{m^2}{f(m)}^2}\left(\binom{(m^2-m)-m}{f(m)}\binom{(m^2-m)+m}{f(m)} - \binom{m^2-m}{f(m)}^2\right) \tag{5.15}$$

Now observe that:

$$\binom{A}{B}^2 = \frac{1}{B!^2}\prod_{i=0}^{B-1}(A-i)^2 \geq \frac{1}{B!^2}\prod_{i=0}^{B-1}(A-i+C)(A-i-C) = \binom{A-C}{B}\binom{A+C}{B} \tag{5.16}$$

. Thus, $\mathrm{Var}[X] \leq \mathrm{E}[X]$, and the Chebyshev inequality gives us:

$$\Pr[X=0] \leq \Pr[|X - \mathrm{E}[X]| \geq \mathrm{E}[X]] \tag{5.17}$$

$$\leq \frac{\mathrm{Var}[X]}{\mathrm{E}[X]^2} \leq \frac{1}{\mathrm{E}[X]} \leq \frac{1}{m^{1-\varepsilon}} \tag{5.18}$$

□

Thus, asymptotically, the union bound guarantees that Pr[simple URE exists] $\geq 1 - 1/\Theta(m^{1-\varepsilon})$. We expect that a tighter analysis of $\Pr[P]$ as an occupancy problem with nearly-negligible dependencies will tighten the result to $1 - 1/\mathrm{superpoly}(m)$. □

How difficult is then the problem UNIT RECALL EQUILIBRIUM (given a game, find a unit recall equilibrium)? Though the problem is prima facie a non-trivial member of $\Sigma_2 P$, the answer is, "not quite that difficult":

Theorem 9 *There is a polynomial algorithm which, given an n-player game and unit recall strategies for $n-1$ players, calculates the best response by the nth player. Therefore,* UNIT RECALL EQUILIBRIUM *is in NP.*

Proof sketch: The $n-1$ given unit recall strategies specify a function ϕ from $\mathcal{S}$ to $\mathcal{S}_{-i}$. Define a graph with vertex set $\mathcal{S}$ and edges $\{(s, (\phi(s), a)) : s \in \mathcal{S}, a \in S_i\}$, and assign to each state s the weight $u^i(s)$. It is not hard to see that the best response is determined by the cycle in this graph that has the highest average weight, a problem solvable in polynomial time [Kar78]. □

Conjecture 5.1.1 UNIT RECALL EQUILIBRIUM *is in P.*

5.2 CURE: URE with slow strategy updates

Let us now weaken this equilibrium concept. A unit recall equilibrium requires that, for each player, there be no way to change some subset of its transitions so that an improvement results. But suppose now that we only require that no improvement result by changing *any one* of the transitions. This could be a reasonable relaxation, if one assumes that transitions in the automaton get changed slowly and sequentially, and that lower utility will prevent a player from even starting the defection process. We call this weaker notion *componentwise unit recall equilibrium*, or *CURE*.

Theorem 10 *Every game has a CURE, which can be found in polynomial time, given a normal-form game.*

Proof sketch: Discard any players with only one choice of strategy. Let $m = \min_i |S_i| \geq 2$ and $M = \max_i |S_i|$. We'll give the proof that works when $n \geq 7$ or $m \geq 3$.

Consider an n-player game ($n \geq 2$), and a state s. We say that s is *i-punishable* if there is another state $s^{(i)}$, differing from s in the strategy of two or more players, such that $u^i(s^{(i)}) \leq u^i(s)$. We call a state s *punishable* if it is i-punishable for all players i.

Lemma 5 *Every game with 7 or more players, or with every player having* 3 *or more strategies, has a punishable state.*

Proof: The proof proceeds by a counting argument. For each player i, pick any s from the set $\arg\min_s u_i(s)$ to be $s^{(i)}$. For each such state, there are $\sum_i |S_i| - n + 1$ states that differ from it in at most one player's strategy. Via the union bound, the number of punishable states must be at least:

$$\prod_i |S_i| - n\Big(\sum_i |S_i| - n + 1\Big) > \prod_i |S_i| - n\sum_i |S_i| \tag{5.19}$$

$$\geq Mm^{n-1} - n^2 M \tag{5.20}$$

Since $M > 0$, this is non-negative (guaranteeing at least one punishable state) whenever $n \geq 2$ and $m \geq 7$ and whenever $n \geq 3$ and $m \geq 3$. For $n = 2$ and $m = 3$, just set $s_i \notin \{s_i^{(1)}, s_i^{(2)}\}$. □

Now, given a punishable state s, we shall construct n unit recall strategies, one for each player, which, we shall argue, form a CURE. We shall describe these strategies

implicitly, by the function ψ from $\mathcal{S}$ to itself that they induce. This function starts at s, and it maps s to itself. For each player i, any state s' such that $s_{-i} = s'_{-i}$ is mapped to the state $s^{(i)}$ guaranteed to exist by i-punishability. All other states are mapped to themselves, and this concludes the description of the CURE.

To verify that s is a CURE, notice that, since the play is on s alone, only changes of the transition for s by a player could possibly lead to disequilibrium. However, if player i defects from s_i, the play will move to a state s' with $s_{-i} = s'_{-i}$, and then immediately to $s^{(i)}$ where it stays forever, thus failing to improve i's utility (by the assumption that $u^i(s^{(i)}) \leq u^i(s)$). This completes the proof of existence.

In a normal-form game, a punishable state can be found by exhaustion.

Some geometric arguments, omitted here, extend the lemma to guarantee a punishable state for all games except 2×2, $2 \times 2 \times 2$, $2 \times 2 \times 3$, and $2 \times 2 \times 2 \times 2$ games. The finite corner cases are handled by tweaking the "punishable state" approach. □

We are not claiming that the equilibria guaranteed to exist by this result are in any sense natural. State s is supported, for each player i, by both "threats" by other players (their transitions from s' to $s^{(i)}$ and the persistence in $s^{(i)}$) and by what is essentially a commitment by player i to collaborate with the other players in punishing itself if it ever deviates from playing s. Still, besides sink and correlated equilibria, CUREs in normal-form games are the only equilibrium concept we know that is both tractable and universal.

5.3 Forecasting the average: Lasso equilibria

When evaluating the cost of the sink they are destined for, players with unit recall in the above model assume that they are committed to following their full automaton strategy in perpetuity. They expect to receive the steady-state average payoff over the sink they are "aimed" toward. What if, as they look ahead to the future, they are allowed to realize that "it's downhill from here" — that is, that the sink they are headed to is worse for them than the state they are now in?

Bilò and Flamini [BF07] propose a similar notion of *second-order Nash equilibria*, where players recursively consider the worst-case scenario if they continue doing what they are doing, and stop moving if there's a risk of ending up worse off later on. The resulting equilibria are a generalization of pure Nash equilibria to include states where players might have a best-response move they could make but don't, due to some risk of eventual decrease

in payoff. Unsurpisingly, these equilibria are present and plentiful, since the players are very risk-averse and are willing to stop moving at the slightest probability of a decrease.

We consider instead a model where players evaluate the *average* payoff in their future, and stop moving if they would currently be better off. This definition can be made recursive, with each player iteratively re-evaluating his stopping based on the other player's stopping decisions. For simplicity, we consider the bimatrix case, with players in any given state given just the option of keeping the same strategy, or switching to their best response. In this case, if no players stop, they will deterministically follow a "lasso" shaped walk in the state space (a connected subgraph with outdegree at most 1, i.e. either a path or a path connected to a cycle). The recursion is then irrelevant to our goal of evaluating the universality of such equilibria and tractability of finding them, since, as long as there is a state where one player will change his mind about the following the lasso and stop, further recursion will only result in an equilibrium point earlier on the path from the starting state to that state. Once the first stopping point in the recursion is found, finding the actual recursive equilibrium requires just a quick backward induction.

These equilibria, which we will call *lasso equilibria* in honor of the bimatrix case, are particularly applicable to situations where a player's payoff after a strategy change is only meaningful after some other players have had a chance to respond to it. For instance, the BGP model in Chapter 4 has a natural extension for "traffic attraction", a common Internet phenomenon where ASes gain revenue from having their customers route more traffic to them [GHJ+08]. There, the provider AS's immediate payoff after a routing table update is meaningless until its customers have had a chance to reconsider whether they want to route through that provider. We expect that the fully-general version of lasso equilibria will be a useful tool for modeling this.

Our results on lasso equilibria here are parallel to our URE result in Section 5.1 — we show that, with high probability, in a random bimatrix game, lasso equilibria exist and are easy to find. Thus, they are a reasonable and almost-universal generalization of pure Nash.

Theorem 11 *For any $\varepsilon > 0$, in an $m \times m$ bimatrix game with payoffs chosen independently from the uniform distribution on an interval, the probability that a lasso equilibrium exists and can be found in polynomial time is $1 - 1/\Omega(m^{1/5})$.*

Proof: Without loss of generality, assume that the interval is $[0, 1]$. Let (R, C) again

be the payoff matrices for the row and column players, whom we'll name $\mathcal{R}$and $\mathcal{C}$. We will consider the possible best-response transitions between states that are at least one player's best response, and will proceed by demonstrating that:

1. With high probability, there is a loop of non-trivial size among these transitions.

2. Such a loop, with high probability, has a state which is the best response for $\mathcal{C}$, but is also better for $\mathcal{R}$than $\mathcal{R}$'s average payoff over the loop.

We will slowly "reveal" information about the random values in the payoff matrices so as to exploit the independence conditions at each step. First, reveal *which* column is $\mathcal{C}$'s best response to $\mathcal{R}$playing his strategy 1 (with no regard for what the actual payoffs are), and label that column κ_1. Let $\rho_0 = 1$. We'll then consider the path of best-response transitions starting from (ρ_0, κ_1) (where only $\mathcal{R}$can have an incentive to move). The first step is to $\mathcal{R}$'s best response to κ_1, which we call ρ_2; the second is to $\mathcal{C}$'s best response to ρ_2, which we call κ_3. For ease of notation, we only use even-numbered ρs and odd-numbered κs. Note that at each iteration from (ρ_i, κ_{i+1}) to $(\rho_{i+2}, \kappa_{i+1})$, we only need to reveal which entry in the κ_{i+1}'th column of R is the maximum, rather than anything about the values in that column, and conversely for the $(\rho_{i+1}, \kappa_i) \to (\rho_{i+1}, \kappa_{i+2})$ transition, where we reveal which entry of C is the maximum of the ρ_{i+1}'th row. Let $k^* = \min\{k | \exists i < k \,.\, \rho_i = \rho_k \vee \kappa_i = \kappa_k\}$ be the first time where the path loops back to itself to form the lasso. This can include $k = i + 2$, i.e. the player that could've deviated doesn't actually want to, so the state is a pure Nash. To lower-bound the likely time until we cycle back, we get:

$$\Pr[k^* > k] = \prod_{i=1}^{k} \Pr[k^* \neq i | k^* \geq i] \tag{5.21}$$

WLOG, let i be even. Conditioned on $k^* \geq i$, $k^* \neq i$ is just the event that ρ_i, $\mathcal{R}$'s best response to $\mathcal{C}$playing κ_{i-1}, does not land in a row that we've previously "revealed". Since the conditioning requires that κ_{i-1} is not a repeat, we must not have revealed any information about the κ_{i-1} column of R, so this probability is just $\frac{m-i/2}{m}$, since $\mathcal{R}$'s best response to in column κ_{i-1} just can't land in any of the $i/2$ rows we passed earlier. Inductively, we get,

for odd k:

$$\Pr[k^* > k] = \prod_{i=1}^{(k-1)/2} \left(1 - \frac{i}{m}\right)^2 \tag{5.22}$$

$$\geq \prod_{i=1}^{(k-1)/2} \left(1 - \frac{k-1}{m}\right)^2 \tag{5.23}$$

$$\geq \left(1 - \frac{k-1}{m}\right)^{k-1} \tag{5.24}$$

$$\geq 1 - \frac{k^2}{m} \tag{5.25}$$

Set $k = m^{2/5}$ (we'll omit the arithmetic to deal with rounding to an odd integer). We get $\Pr[k^* < m^{2/5}] \leq m^{-1/5}$. Given $k^* > m^{2/5}$ (with k^* WLOG even), the probability that the loop from ρ_{k^*} goes to a ρ_i with $i > k^* - m^{1/5}$ is at most $\frac{m^{1/5}}{m^{2/5}} = m^{-1/5}$, since, again, the conditioning is independent of the κ_{k^*-1}'th column of R, and the probability of the best response being one of the last $m^{1/5}$ steps by $\mathcal{R}$is just $m^{1/5}/k^*$.

Let l be the length of the loop reached from (ρ_0, κ_1). By the union bound, we get that $\Pr[l] < m^{1/5} \leq 2m^{-1/5}$.

The resulting loop, as any path in this dynamic, alternates between $\mathcal{R}$'s and $\mathcal{C}$'s best responses. $\mathcal{R}$'s average payoff is thus the average between $l/2$ of his best responses, and $l/2$ of his payoffs at states which are $\mathcal{C}$'s best responses. The former are bounded by 1. The latter, conditioned on everything we've revealed thus far, are just independent samples $R_{\rho_{k^*-2l},\kappa_{k^*-2l+1}}, \ldots, R_{\rho_{k^*-2},\kappa_{k^*-1}}$ (hereafter written as $r_0, r_2, r_4, \ldots, r_{l-2}$) from the underlying uniform distribution, each conditioned only the event notmax(i) that, out of m payoffs in r_i's column of R, it is not the maximum. This distribution has the following c.d.f.:

$$\Pr[r_i < \alpha | \text{notmax}(i)] = \frac{\Pr[r_i < \alpha]}{\Pr[\text{notmax}(i)]} \Pr[\text{notmax}(i) | r_i < \alpha] \tag{5.26}$$

$$= \frac{\alpha m}{m-1} \left(1 - \int_0^\alpha r_i^{m-1} \frac{dr_i}{\alpha}\right) \tag{5.27}$$

$$= \frac{m\alpha - \alpha^m}{m-1} \tag{5.28}$$

Then, $\mathrm{E}[r_i] = \frac{m}{m-1}\left(\frac{1}{2} - \frac{1}{m+1}\right) \leq \frac{1}{2}$ and $\mathrm{Var}[r_i] = \frac{m}{3(m+2)} - \frac{m^2}{4(m+1)^2} = \frac{m^3+2m^2+4m}{12(m^3+4m^2+5m+2)} \leq \frac{1}{12}$.

The Chebyshev bound gives us:

$$\Pr\left[\mathcal{R}\text{'s average payoff over loop} > \frac{7}{8}\right] \leq \Pr\left[\sum_{\text{even } i} r_i > \frac{3}{4}\middle|\text{notmax}(i)\forall i\right] \tag{5.29}$$

$$\leq \frac{\text{Var}\left[\sum_{\text{even } i} r_i|\text{notmax}(i)\forall i\right]}{(1/4)^2} \tag{5.30}$$

$$\leq \frac{8}{3l} \tag{5.31}$$

$$\leq \frac{8}{3m^{1/5}} \tag{5.32}$$

Lastly, we bound the probability that, among the $l/2$ positions where $\mathcal{R}$could make a best response, there are none that are good enough to beat the loop average and convince $\mathcal{R}$to stop:

$$\Pr\left[\neg\exists i\,.\,r_i > \frac{7}{8}\right] = \left(\frac{\frac{7}{8}m - \left(\frac{7}{8}\right)^m}{m-1}\right)^l \leq \frac{63}{64}^l \forall m \geq 9 \tag{5.33}$$

Thus, combining the four above via the union bound, we get that the probability that there is no place along the lasso starting from (ρ_0, κ_1) where $\mathcal{R}$is happier than he would be on average over the loop, thus ensuring that he would stop, and that thus somewhere on the path between (ρ_0, κ_1) and that point is a lasso equilibrium, is $1 - 1/\Omega(m^{1/5})$. In a normal-form game, following the path from (ρ_0, κ_1) to find such a point and applying backward induction to obtain the lasso equilibrium requires only linear time. □

This result is an initial indication that lasso equilibria show some promise for modelling games with delayed payoffs. We leave open a variety of interesting questions concerning them. Most prominently, there is a design decision about how to model the necessary non-determinism that will arise with more than 2 players — depending on the application domain, it may make sense for the players to evaluate the future based on an average over random orderings of the "activation sequence" of players, or based on the worst-case activation sequence's average loop payoff, etc. Also, there is an interesting question, similar to the URE result above, of what happens when players try to game the future by making non-best-response moves.

Bibliography

[ARV06] Heiner Ackermann, Heiko Roglin, and Berthold Vöcking. On the impact of combinatorial structure on congestion games. pages 613–622, 2006.

[Aum00] Robert J. Aumann. Survey of repeated games. In *Collected papers*, pages 411–438. MIT Press, 2000.

[BF07] Vittorio Bilò and Michele Flammini. Extending the notion of rationality of selfish agents: Second order nash equilibria. In *International Symposium on Mathematical Foundations of Computer Science*, pages 621–632, 2007.

[BSKK06] Eli Ben-Sasson, Adam Kalai, and Ehud Kalai. An approach to bounded rationality. In *Proceedings of the Conference on Neural Information Processing Systems*, 2006.

[CD06] X. Chen and X. Deng. Settling the complexity of 2-player Nash equilibrium. In *Proceedings of IEEE Symposium on Foundations of Computer Science*, pages 261–270, 2006.

[CDT06] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Computing Nash equilibria: Approximation and smoothed complexity. In *Proceedings of IEEE Symposium on Foundations of Computer Science*, pages 603–612, 2006.

[Cou38] Augustin Cournot. *Recherches sur les Principes Mathématiques de la Théorie des Richesses*. Paris: Hachette, 1838.

[DFP06] Constantinos Daskalakis, Alex Fabrikant, and Christos H Papadimitriou. The game world is flat: The complexity of Nash equilibria in succinct games. In *Proceedings of International Colloquium on Automata, Languages and Programming*, pages 513–524, 2006.

[DGP06] C. Daskalakis, P.W. Goldberg, and C.H. Papadimitriou. The complexity of computing a Nash equilibrium. In *Proceedings of ACM Symposium on Theory of Computing*, pages 71–78, 2006.

[DMP06] Constantinos Daskalakis, Aranyak Mehta, and Christos Papadimitriou. A note on approximate Nash equilibria. In *Proceedings of Workshop on Internet and Network Economics*, 2006.

[FKK+02] D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. Spirakis. The structure and complexity of Nash equilibria for a selfish routing game. In *Proceedings of International Colloquium on Automata, Languages and Programming*, pages 123–134, 2002.

[FLM+03] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. On a network creation game. In *Proceedings of ACM Symposium on Principles of Distributed Computing*, pages 347–351, 2003.

[FNS] Tomás Feder, Hamid Nazerzadeh, and Amin Saberi. Breaking the factor of two in approximate Nash equilibria. Manuscript.

[FP08] Alex Fabrikant and Christos H. Papadimitriou. The complexity of game dynamics: Bgp oscillations, sink equilibria, and beyond. In *Proceedings of SIAM Symposium on Discrete Algorithms*, pages 844–853, 2008.

[FPT04] A. Fabrikant, C. H. Papadimitriou, and K. Talwar. The complexity of pure Nash equilibria. In *Proceedings of ACM Symposium on Theory of Computing*, pages 604–612, 2004.

[FS98] E. J. Friedman and S. Shenker. Learning and implementation on the Internet. Working paper 1998-21, Rutgers Univ., Dept. of Economics, 1998. Available from http://www-snde.rutgers.edu/Rutgers/wp/rutgers-wplist.html.

[GGN68] K. Goldberg, A. J. Goldman, and M. Newman. The probability of an equilibrium point. *Journal of Research of the National Bureau of Standards USA, Section B*, 72B:93–101, 1968.

[GGS03] Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. Pure nash equilibria: hard and easy games. In *TARK '03: Proceedings of the 9th conference on*

Theoretical aspects of rationality and knowledge, pages 215–230, New York, NY, USA, 2003. ACM.

[GHJ+08] Sharon Goldberg, Shai Halevi, Aaron D. Jaggard, Vijay Ramachandran, and Rebecca N. Wright. Rationality and traffic attraction: incentives for honest path announcements in BGP. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 267–278, New York, NY, USA, 2008. ACM.

[GKMP06] Parikshit Gopalan, Phokion Kolaitis, Elitza Maneva, and Christos H Papadimitriou. The connectivity of Boolean satisfiability: Computational and structural dichotomies. In *Proceedings of International Colloquium on Automata, Languages and Programming*, pages 346–357, 2006.

[GMV05] M. Goemans, V. Mirrokni, and A. Vetta. Sink equilibria and convergence. In *Proceedings of IEEE Symposium on Foundations of Computer Science*, pages 142–151, 2005.

[GSW02] T. Griffin, F. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking*, 10(2):232–243, 2002.

[GW97] Timothy G. Griffin and Gordon Wilfong. An analysis of BGP convergence properties. In *Proceedings of ACM SIGCOMM*, pages 277–288, 1997.

[JPY88] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37:79–100, 1988.

[Kar78] Richard Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23:309–311, 1978.

[KPS06] Spyros Kontogiannis, Panagiota Panagopoulou, and Paul Spirakis. Polynomial algorithms for approximating Nash equilibria of bimatrix games. In *Proceedings of Workshop on Internet and Network Economics*, 2006.

[Kre89] M. W. Krentel. Structure in locally optimal solutions. In *Proceedings of IEEE Symposium on Foundations of Computer Science*, pages 216–221, 1989.

[LMJ98] C. Labovitz, G. R. Malan, and F. Jahanian. Internet routing instability. *IEEE/ACM Transactions on Networking*, 6:515–528, 1998.

[LMM03] Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. Playing large games using simple strategies. In *Proceedings of ACM Conference on Electronic Commerce*, pages 36–41, 2003.

[LS03] Michael L. Littman and Peter Stone. A polynomial-time Nash equilibrium algorithm for repeated games. In *Proceedings of ACM Conference on Electronic Commerce*, pages 48–54, 2003.

[MGWR01] Danny McPherson, Vijay Gill, Daniel Walton, and Alvaro Retana. BGP persistent route oscillation condition. In *Proceedings of 50th Internet Engineering Task Force*, March 2001.

[Mil96] I. Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13:111–124, 1996.

[MS96] D. Monderer and L. S. Shapley. Potential games. *Games and Economic Behavior*, 14:124–143, 1996.

[Mye99] Roger Myerson. Nash equilibrium and the history of economic theory. *Journal of Economic Literature*, 37(3):1067–1082, September 1999.

[Nas50] J. F. Nash. Equilibrium points in n-person games. In *Proceedings of National Academy of Sciences*, volume 36, pages 48–49, 1950.

[Ney85] Abraham Neyman. Bounded complexity justifies cooperation in the finitely repeated prisoners' dilemma. *Economics Letters*, 19(3):227–229, 1985.

[NM44] John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[OPS04] James B. Orlin, Abraham P. Punnen, and Andreas S. Schulz. Approximate local search in combinatorial optimization. In *Proceedings of SIAM Symposium on Discrete Algorithms*, pages 580–589, 2004.

[Pap92] C. H. Papadimitriou. The complexity of the Lin-Kernighan heuristic for the traveling salesman problem. *SIAM Journal on Computing*, 21(3):450–465, 1992.

[Pap94] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *JCSS*, 48(3):498–532, June 1994.

[Pap05] Christos H. Papadimitriou. Computing correlated equilibria in multi-player games. In *Proceedings of ACM Symposium on Theory of Computing*, pages 49–56, 2005.

[PY94] Christos H. Papadimitriou and Mihalis Yannakakis. On complexity as bounded rationality. In *Proceedings of ACM Symposium on Theory of Computing*, pages 726–733, 1994.

[RL95] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1771 (Draft Standard), 1995.

[Ros73] R. W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[Sor97] S. Sorin. Cooperation through repetition. In S. Hart and A. Mas Colell, editors, *Cooperation: Game Theoretic Approaches*, pages 169–198. Springer, 1997.

[Ste98] J. W. Stewart. *BGP4: Inter-Domain Routing on the Internet*. Addison-Wesley, 1998.

[SvS03] R. Savani and B. von Stengel. Long Lemke-Howson paths. Technical Report LSE-CDAM-2003-14, LSE, 2003.

[SY91] A. A. Schäffer and M. Yannakakis. Simple local search problems that are hard to solve. *SIAM Journal on Computing*, 20(1):56–87, 1991.

[VBvM+99] 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.

[Vet02] A. Vetta. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *Proceedings of IEEE Symposium on Foundations of Computer Science*, pages 416–425, 2002.

[VGE96] K. Varadhan, R. Govindan, and D. Estrin. Persistent route oscillations in inter-domain routing. Technical Report 96-631, USC ISI, 1996.

[vS02] 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.