Algorithmic Game Theory*

Tim Roughgarden†

May 12, 2009

1 Introduction

The widespread adoption of the Internet and the emergence of the Web changed society's relationship with computers. The primary role of a computer evolved from a stand-alone, well-understood machine for executing software to a conduit for global communication, content-dissemination, and commerce. The algorithms and complexity theory community has responded to these changes by formulating novel problems, goals, and design and analysis techniques relevant for modern applications. Game theory, which has studied deeply the interaction between competing or cooperating individuals, plays a central role in these new developments. Research on the interface of theoretical computer science and game theory, an area now known as *algorithmic game theory (AGT)*, has exploded phenomenally over the past ten years.

The primary research themes in AGT differ from those in classical microeconomics and game theory in important, albeit predictable, respects. Firstly in application areas: Internet-like networks and non-traditional auctions motivate much of the work in AGT. Secondly in its quantitative engineering approach: AGT research typically models applications via concrete optimization problems and seeks optimal solutions, impossibility results, upper and lower bounds on feasible approximation guarantees, and so on. Finally, AGT usually adopts reasonable (e.g., polynomial-time) computational complexity as a binding constraint on the feasible behavior of system designers and participants. These themes, which have played only a peripheral role in traditional game theory, give AGT its distinct character and relevance.

The next three sections touch on the current dominant research trends in AGT, loosely following the organization of the first book in the field [30]. We focus on contributions of the algorithms and complexity theory community; see two recent CACM articles [18, 40] and the references therein for alternative perspectives on computer science and game theory.

2 Algorithmic Mechanism Design

Algorithmic mechanism design studies optimization problems where the underlying data — such as the value of a good or the cost of performing a task — is initially *unknown* to the algorithm designer, and must be implicitly or explicitly elicited from self-interested participants (e.g., via a

*This article is a revised and abridged version of [35].

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Supported in part by NSF CAREER Award CCF-0448664, an ONR Young Investigator Award, an AFOSR MURI grant, and an Alfred P. Sloan Fellowship. Email: tim@cs.stanford.edu.

bid). Auction settings are canonical examples, where the private data is the willingness to pay of the bidders for the goods on sale, and the optimization problem is to allocate the goods to maximize some objective, such as revenue or overall value to society. A "mechanism" is a protocol that interacts with participants and determines a solution to the underlying optimization problem.

There is a complex dependence between the way a mechanism employs elicited data and participant behavior. For example, consider the sale of a single good in a sealed-bid auction with several bidders. In a "first-price" auction, the selling price is the bid of the winner (i.e., the maximum bid). Bidders naturally shade their bids below their maximum willingness to pay in first-price auctions, aspiring to achieve the lowest-possible price subject to winning the auction. Determining how much to shade requires guessing about the behavior of the other bidders. A different auction is the "second-price" auction, in which the selling price is only the *second-highest* bid. A famous result of Vickrey [43] is that every participant of a second-price auction may as well bid its true value for the good: intuitively, a second-price auction optimally shades the bid of the winner on its behalf, to the minimum alternative winning bid. eBay and Amazon auctions are similar to second-price auctions in many (but not all) respects; see Steiglitz [42] for a detailed discussion. Keyword search auctions, such as those run by Google, Yahoo!, and Microsoft, are more complex variants of second-price auctions with multiple heterogeneous goods, corresponding to the potential ad slots on a search results page. Lahaie et al. [30, Chapter 28] provide an overview of theoretical work on search auctions.

While the economic literature on mechanism design is quite mature [20], computer scientists have initiated a number of new research directions. We concentrate here on the emphasis in algorithmic mechanism design on complexity bounds and worst-case approximation guarantees, as first proposed by Nisan and Ronen [29]. Additional aspects including prior-free revenue-maximization, distributed (or "Internet-suitable") mechanism design, and online (or "real-time") mechanism design are discussed in [30, Part II].

The technical core of this part of algorithmic mechanism design is the following deep question:

(Q1) *to what extent is "incentive-compatible" efficient computation fundamentally less powerful than "classical" efficient computation?*

To translate question (Q1) into mathematics, reconsider the Vickrey (second-price) auction for selling a single good. Each bidder i has a private willingness-to-pay v_i and submits to the auctioneer a bid b_i. The auction comprises two algorithms: an *allocation algorithm*, which picks a winner, namely the highest bidder; and a *payment algorithm*, which uses the bids to charge payments, namely 0 for the losers and the second-highest bid for the winner. We argued intuitively that this auction is *truthful* in the following sense: for every bidder i and every set of bids by the other participants, bidder i maximizes its "net value" — its value for the good, if received, minus its payment, if any — by bidding its true private value: $b_i = v_i$. Moreover, no false bid is as good as the truthful bid for all possible bids by the other participants. Assuming all bidders bid truthfully (as they should), the Vickrey auction solves the *social welfare maximization* problem, in the sense that the good is allocated to the participant with the highest value for it.

More generally, an allocation algorithm x is *implementable* if, for a judiciously chosen payment algorithm π, coupling x with π yields a truthful mechanism: every participant is guaranteed to maximize its payoff by reporting its true preferences. For a single-good auction, the "highest-bidder" allocation algorithm is implementable (as we have seen); the "second-highest bidder" allocation algorithm is not (a straightforward exercise). Thus some but not all algorithms are implementable.

We can mathematically phrase the question (Q1) as follows: *are implementable algorithms less powerful than arbitrary algorithms for solving fundamental optimization problems?*

Understanding this question involves two interrelated goals: characterization theorems and approximation bounds.

(G1) Usefully characterize the implementable allocation algorithms for an optimization problem.

(G2) Prove upper and lower bounds on the best-possible solution quality of an implementable algorithm for a given objective function, possibly subject to additional constraints, such as polynomial running time.

The second goal quantifies the limitations of implementable algorithms via an approximation measure; the most commonly used such measure is the worst-case ratio, over all possible inputs, between the objective function value of the algorithm's solution and the optimal objective function value. The first goal aims to reformulate the unwieldy definition of implementability into a more operational form amenable to both upper and lower approximation bounds. Both goals, and especially (G1), seem to grow more complex with the number of independent parameters required to describe the private information of a participant.

Versions of (G2) pervade modern algorithmic research: for a given "constrained computational model", where the constraint can be either computational (as for polynomial-time approximation algorithms) or information-theoretic (as for online algorithms), quantify its limitations for optimization and approximation. Goal (G1) reflects the additional difficulty in algorithmic mechanism design that even the "computational model" (of implementable algorithms) induced by strategic constraints is poorly understood — for example, determining whether or not a given algorithm is online is intuitively far easier than checking if one is implementable.

Single-Parameter Mechanism Design. This two-step approach is vividly illustrated by the important special case of *single-parameter problems*, where goal (G1) has been completely resolved. A mechanism design problem is *single-parameter* if the possible outcomes are real n-vectors ω and each participant i has an objective function of the form $v_i\omega_i$ for a private real number v_i (the "single parameter"). The numbers ω_i and v_i can be thought of as the quantity received and the value-per-unit of a good, respectively. A single-item auction is the special case in which each ω is either a standard basis vector or the all-zero vector. Keyword search auctions are also single-parameter, under the assumptions that every advertiser cares only about the probability ω_i of a click on its sponsored link and has a common value v_i for every such click.

An algorithm for a single-parameter problem is *monotone* if a greater bid begets a greater allocation: increasing the value of a bid (keeping the other bids fixed) can only increase the corresponding value of the computed ω_i. For example, the "highest bidder" allocation algorithm for a single-good auction is monotone, while the "second-highest bidder" allocation algorithm is not. In general, monotonicity characterizes implementability for single-parameter problems.

Myerson's Lemma ([27]) *An allocation algorithm for a single-parameter mechanism design problem is implementable if and only if it is monotone.*

Myerson's Lemma is a useful solution to the first goal (G1) and reduces implementable algorithm design to monotone algorithm design. For example, consider the following "rank-by-weighted bid" allocation algorithm for a keyword search auction. Advertisers' bids are sorted in decreasing order, possibly after scaling by advertiser-specific "relevance" factors, and ad slots are populated in this

order. Assuming that the probability of a click is higher in higher slots, every such algorithm is monotone: increasing one's bid can only increase one's position in the ordering, which in turn leads to an only higher probability of a click. Thus, Myerson's Lemma guarantees an analog of the second-price rule that extends the allocation algorithm into a truthful auction.[1]

Despite our thorough understanding of goal (G1), question (Q1) remains open for single-parameter problems. A single-parameter scheduling problem proposed by Archer and Tardos [1] had been the most natural candidate for differentiating between the optimization power of monotone and arbitrary polynomial-time algorithms, but Dhangwatnotai et al. [14] recently gave a (randomized) polynomial-time monotone algorithm for the problem with approximate guarantee as good as the best-possible polynomial-time algorithm (assuming $P \neq NP$).

Multi-Parameter Mechanism Design. Many important mechanism design problems are not single-parameter. *Combinatorial auctions* [11], in which each participant aims to acquire a heterogeneous set of goods and has unrelated values for different sets, are a practical and basic example. Combinatorial auctions are used in practice to sell wireless spectrum (where the goods are different licenses), with recent auction designs by theoretical economists generating billions of dollars of revenue over the past decade [11]. Their complexity stems from "complements", meaning goods that are more useful when purchased in tandem (e.g., spectrum licenses for small but adjacent regions); and "substitutes", meaning goods that are partially redundant (e.g., two different but functionally identical licenses for the same region). Each bidder in a combinatorial auction has, in principle, an exponential number of private parameters — one private value for each subset of goods.

Multi-parameter mechanism design is complex and our current understanding of goals (G1) and (G2) is primitive for most problems of interest. Here, there can be a provable gap between the worst-case approximation ratio of implementable and arbitrary polynomial-time algorithms for natural optimization problems. This fact was first proved by Lavi et al. [23]; recently, Papadimitriou et al. [33] showed that this gap can be very large (polynomial in the number of bidders). Because of its importance and bounty of open questions, multi-parameter mechanism design has been a hotbed of activity over the past few years. See [35] for a survey of the primary research threads, including upper and lower approximation bounds for polynomial-time welfare maximization for combinatorial auctions, and work toward multi-parameter analogs of Myerson's Lemma.

3 Quantifying Inefficiency and the Price of Anarchy

The truthful mechanisms studied in Section 2 are — by design — strategically degenerate in that the best course of action of a participant (i.e., truthtelling) does not depend on the actions taken by the others. When a designer cannot specify the rules of the game and directly dictate the allocation of resources — or when there is no central designer at all — dependencies between different participants' optimal courses of action are generally unavoidable and preclude exact optimization of standard objective functions. This harsh reality motivates adopting an *equilibrium concept* — a rigorous proposal for the possible outcomes of a game with self-interested participants — and an *approximation measure* that quantifies the inefficiency of a game's equilibria, to address the following basic question:

[1] Modern search engines use allocation algorithms that are similar to rank-by-weighted bid algorithms. By historical accident, they use a slightly different pricing rule than that advocated by Myerson's Lemma, although the two pricing rules lead to comparable outcomes and revenue at equilibrium. See Lahaie et al. [30, Chapter 28] for more details.

(Q2) *when, and in what senses, are game-theoretic equilibria guaranteed to approximately optimize natural objective functions?*

Such a guarantee implies that the benefit of imposing additional control over the system is small, and is particularly reassuring when implementing an optimal solution is infeasible (as in a typical Internet application).

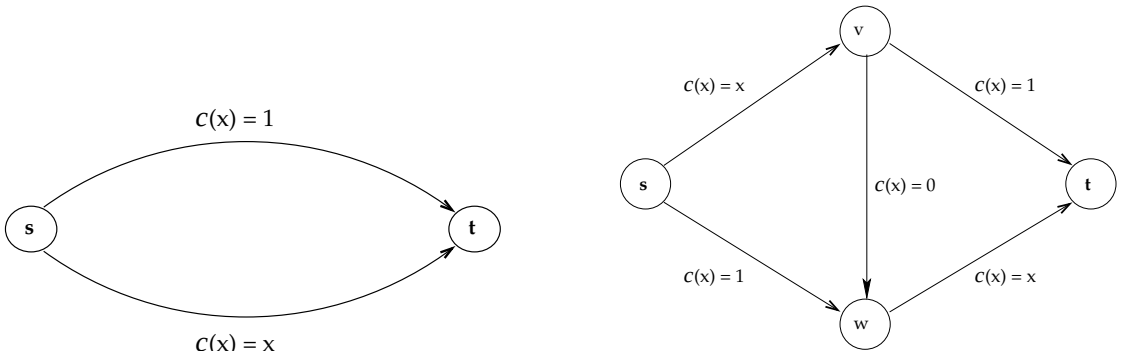
Routing with Congestion. There are now numerous answers to question (Q2) in different models; we describe one by Roughgarden and Tardos [37, 39], for a model of "selfish routing" originally proposed for road traffic (see [4]) and subsequently adapted to communication networks (see [5]). This was the first general approximation bound on the inefficiency of equilibria; the idea of quantifying such inefficiency was explored previously in a scheduling model [22].

Consider a directed graph with fixed traffic rates between various origin-destination pairs in which the traffic chooses routes to minimize individual cost; see also Figure 1. In this section, we assume that the traffic comprises a large number of selfish users, each of negligible size — such as drivers on a highway or packets in a network. Edge costs are *congestion-dependent*, with the continuous, nondecreasing function $c_e(x)$ denoting the per-unit cost incurred by traffic on edge e when x units of traffic use it. In an *equilibrium*, each user travels along a minimum-cost path from its origin to its destination, given the congestion caused by the traffic. These selfish routing games are strategically non-trivial in that the minimum-cost path for a given user generally depends on the paths chosen by the others.

For example, in a "Pigou-like network" (Figure 1(a)), r units of selfish traffic autonomously decide between parallel edges e_1 and e_2 that connect the origin s to the destination t. Suppose the second edge has some cost function $c_2(\cdot)$, and the first edge has a constant cost function c_1 everywhere equal to $c_2(r)$. Such networks are strategically trivial, just like the truthful mechanisms of Section 2: the second edge's cost is never larger than that of the first, even when it is fully congested. For this reason, all traffic uses the second edge at equilibrium. This equilibrium does not generally minimize the average cost of all users. For example, if $r = 1$ and $c_2(x) = x$ as in Figure 1(a), the average cost at equilibrium is 1, while splitting the traffic equally between the two edges yields a routing with average cost 3/4. The latter traffic pattern is not an equilibrium because of a "congestion externality": a selfish network user routed on the first edge would switch to the second edge, indifferent to the fact that this switch (slightly) increases the cost incurred by a large portion of the population. Similarly, in the "Braess's Paradox" [7] network of Figure 1(b), the average cost at equilibrium is 2 (with all traffic on the zig-zag path), while a benevolent dictator could route the traffic at average cost 3/2 (by splitting traffic between the two two-hop paths).[2]

The *price of anarchy (POA)* of a selfish routing network is the ratio of the average user cost at equilibrium and in an optimal routing — 4/3 in both of the networks in Figure 1. The closer the POA is to 1, the lesser the consequences of selfish behavior. Replacing the cost function of the second edge in Figure 1(a) by $c_2(x) = x^d$ for large d shows that the POA can be arbitrarily large, even in Pigou-like networks, and suggests that the POA is governed by the "degree of nonlinearity" of the cost function c_2. A key result formalizes and extends this intuition to *arbitrary* networks: among all networks with cost functions lying in a set $\mathcal{C}$ (e.g., bounded-degree polynomials with nonnegative

[2]This network is called a "paradox" because removing the intuitively helpful zero-cost edge — depriving users of one of their options — recovers the optimal solution as an equilibrium, thereby decreasing the cost incurred by all users. Analogously, cutting a taut string in a network of strings and springs that carries a heavy weight can cause the weight to levitate further off of the ground! [10]



(a) Pigou's Example (b) Braess's Paradox

Figure 1: Two selfish routing networks with price of anarchy 4/3. One unit of selfish traffic travels from s to t. At equilibrium, all traffic travels on the bottom path and the zig-zag path, respectively. In an optimal solution, traffic is split equally between the two edges and between the two two-hop paths, respectively.

coefficients), the largest-possible POA is achieved already in Pigou-like networks [37]. Conceptually, *complex topologies do not amplify the worst-case POA*. This reduction permits the easy calculation of tight bounds on the worst-case POA for most interesting sets $\mathcal{C}$ of cost functions. For example, the POA of every selfish routing network with affine cost functions (of the form $c_e(x) = a_e x + b_e$ for non-negative a_e, b_e) is at most 4/3, with a matching lower bound provided by the examples in Figure 1. See [30, Chapter 18] for a recent survey detailing these and related results.

These POA bounds provide a theoretical justification for a common rule of thumb used in network design and management: *overprovisioning networks with extra capacity ensures good performance*. Precisely, suppose every edge e of a network has a *capacity* u_e and a corresponding cost function $c_e(x) = 1/(u_e - x)$; see Figure 2(a). (If $x \geq u_e$, we interpret the cost as infinite.) This is the standard M/M/1 queueing delay function with service rate u_e. We say that a network is β-*overprovisioned* for $\beta \in (0, 1)$ if, at equilibrium, at least a β fraction of each edge's capacity remains unused. The following is a tight bound on the POA for such networks; the bound is illustrated in Figure 2(b).

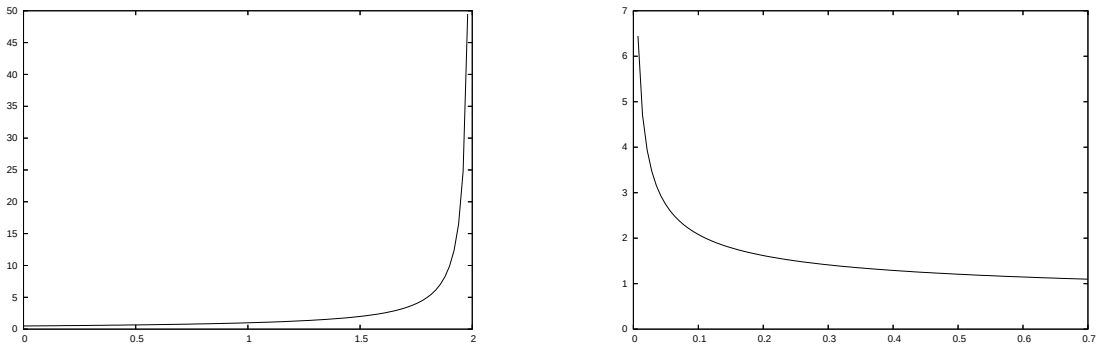
Theorem (Consequence of [37]) *The POA of every β-overprovisioned network is at most*

$$\frac{1}{2}\left(1 + \frac{1}{\sqrt{\beta}}\right).$$

Thus even 10% extra capacity reduces the worst-case price of anarchy of selfish routing to roughly 2.

Further Aspects of Quantifying Inefficiency. We have barely scratched the surface of recent work on equilibrium efficiency analyses. See [30, Part III] for an overview of work on some other application domains, including resource allocation, scheduling, facility location, and network design.

An important emerging trend in this area is to prove POA-type bounds under increasingly weak assumptions on the rationality of participants. Recall that in Section 2, our only assumption was that participants will make use of a "foolproof" strategy (one that dominates all others), should one be available. This section implicitly assumed that selfish participants can reach an equilibrium of a



(a) M/M/1 delay function

(b) Extra capacity vs. POA curve

Figure 2: Modest overprovisioning guarantees near-optimal routing. The left-hand figure displays the per-unit cost $c(x) = 1/(u - x)$ as a function of the load x for an edge with capacity $u = 2$. The right-hand figure shows the worst-case price of anarchy as a function of the fraction of unused network capacity.

game without such foolproof strategies, presumably through repeated experimentation. This much stronger assumption has been addressed in two different ways in the recent literature. The first is to formally justify it by positing natural experimentation strategies and proving that they quickly reach a (possibly approximate) equilibrium; see [9] and the references therein for a sampling of such results. The second is to prove POA-like guarantees that apply "on average," even when such experimentation strategies fail to converge to an equilibrium. Remarkably, such approximation bounds hold in interesting classes of games, including in selfish routing networks. See [2, 6, 19] for initial formalizations of this approach, and [36] for a recent general result that shows that, under weak conditions, POA bounds for equilibria extend automatically to the results of repeated experimentation.

4 Complexity of Equilibrium Computation

Equilibrium concepts — most famously the Nash equilibrium [28] — play a starring role in game theory and microeconomics. If nothing else, a notion of equilibrium describes outcomes that, once reached, persist under some model of individual behavior. In engineering applications we generally demand a stronger interpretation of an equilibrium, as a credible *prediction* of the long-run state of the system. But none of the standard equilibrium notions or the corresponding proofs of existence suggest how to arrive at an equilibrium with a reasonable amount of effort. This fact motivates the following questions.

(Q3) *When can the participants of a game quickly converge to an equilibrium? More modestly, when can a centralized algorithm quickly compute an equilibrium?*

These questions are interesting for two reasons. First, algorithms for equilibrium computation can be useful practically, for example in game-playing and for multi-agent reasoning [41]. Second, assuming that players can invest only polynomial computation in playing a game, resolving the complexity of computing an equilibrium concept has economic implications: a polynomial-time

algorithm is an important step toward establishing the concept's credibility, while an intractability result casts doubt on its predictive power.

There has been a frenzy of recent work on these questions, for many different fundamental equilibrium concepts. Perhaps the most celebrated results in the area concern the $PPAD$-completeness of computing mixed-strategy Nash equilibria in finite games with two or more players [8, 12]. To briefly convey the spirit of the area with a minimum of technical fuss, we instead discuss the complexity of converging to and computing pure-strategy Nash equilibria in a variant of the routing games studied in Section 3. We then discuss the key differences between the two settings. For work on the complexity of computing other equilibrium concepts, such as market, correlated, and approximate Nash equilibria, and for a discussion of equilibrium computation in extensive-form, compact, randomly generated, and stochastic games, see [30, Part I] and [38] and the references therein.

Pure Nash Equilibria in Network Congestion Games. In the *atomic* variant of selfish routing, there are a finite number k of players that each control a non-negligible amount of traffic (say one unit each) and choose a single route for it. Each edge cost function $c_e : \{1, 2, \ldots, k\} \rightarrow \mathcal{R}^+$, describing the per-player cost along an edge as a function of its number of users, is non-decreasing. An outcome $(P_1, \ldots, P_k)$ — a choice of a path P_i for each player i — is a *pure-strategy Nash equilibrium (PNE)* if each player simultaneously chooses a *best response*: a path with minimum-possible cost, given the paths chosen by the other players. For instance, consider Pigou's example (Figure 1(a)) with the constant cost on the upper edge raised from 1 to 2. If there are two players (with origin s and destination t), then there are three PNE: one with both players on the lower link, and two in which each link is used by a single player. In every case, a deviating player would incur cost 2 and be no better off than in the equilibrium.

Best-response dynamics is a simple model of experimentation by players over time: while the current outcome is not a PNE, choose an arbitrary player that is not using a best response, and update its path to a best response. The update of one player usually changes the best responses of the others; for this reason, best-response dynamics fails to converge in many games (such as "Rock-Paper-Scissors"). In an atomic selfish routing network, however, every iteration of best-response dynamics strictly decreases the potential function

$$\Phi(P_1, \ldots, P_k) = \sum_{e \in E} [c_e(1) + c_e(2) + \cdots + c_e(x_e)],$$

where x_e denotes the number of paths P_i that contain edge e, and is thus guaranteed to terminate, necessarily at a PNE [26, 34]. Does convergence require polynomial or exponential time? Can we compute a PNE of such a game by other means in polynomial time?

Assume for the moment that the problem of computing a PNE of an atomic selfish routing network is not solvable in polynomial time; how would we amass evidence for this fact? An obvious idea is to prove that the problem is NP-hard. Remarkably, a short argument [21, 25] shows that this is possible only if $NP = coNP$! Intuitively, solving an NP-hard problem like satisfiability means to either exhibit a satisfying truth assignment of the given Boolean formula or to correctly determine that none exist. Computing a PNE of an atomic selfish routing game appears easier because the latter situation (of there being no PNE) can be ruled out a priori — the "only" challenge is to exhibit a solution in polynomial time.[3]

[3]The complexity classes P and NP are usually defined for *decision* problems, where the answer sought is a simple

To motivate the definition of the appropriate complexity class, recall that problems in the class NP are characterized by short and efficiently verifiable witnesses of membership, such as satisfying truth assignments or Hamiltonian cycles. There is thus a generic "brute-force search" algorithm for NP problems: given an input, enumerate the exponentially many possible witnesses of membership, and check if any of them are valid. Computing a PNE of an atomic selfish routing game appears to be easier than an NP-hard problem because there is a *guided search* algorithm (namely, best-response dynamics) that navigates the set of possible witnesses and is guaranteed to terminate with a legitimate one. At worst, computing a PNE might be as hard as all problems solvable by such a "guided search" procedure. This is in fact the case, as we formalize next.

What are the minimal ingredients that guarantee that a problem is solvable via guided search? The answer is provided by the complexity class PLS (for "polynomial local search") [21]. A PLS problem is described by three polynomial-time algorithms: one to accept an instance and output an initial candidate solution; one to evaluate the objective function value of a candidate solution; and one that either verifies local optimality (for some local neighborhood) or else returns a neighboring solution with strictly better objective function value. To "solve" a PLS problem means to compute a local optimum, by local search or by other means. For example, computing a PNE of an atomic selfish routing game can be cast as a PLS problem by adopting the potential function as an objective function, and defining two outcomes to be neighbors if all but one player choose the same path in both. Local minima then correspond to the PNE of the game. A problem in PLS is then PLS-*complete* if every problem in PLS reduces to it in polynomial time, in which case the complete problem is solvable in polynomial time only if *every* problem in PLS is.

The problem of computing a PNE of an atomic selfish routing network is PLS-complete [17]. It is therefore polynomial-time solvable if and only if $P = PLS$. In the spirit of the P vs. NP question, it is generally believed that $P \neq PLS$ but researchers seem far from a resolution in either direction. Since PLS contains several important problems that have resisted all attempts at a computationally efficient solution, PLS-hardness is viewed as strong evidence that a problem will not be solved in polynomial time (at least in the near future).

Mixed-Strategy Nash Equilibria and *PPAD*. A *mixed strategy* is a probability distribution over the pure strategies of a player. In a *mixed-strategy Nash equilibrium (MNE)*, every player simultaneously chooses a mixed strategy maximizing its expected payoff, given those chosen by the others. For example, in "Rock-Paper-Scissors", with each player receiving payoff 1 for a win, 0 for a draw, and -1 for a loss, the only MNE has each player randomizing uniformly over its three strategies to obtain an expected payoff of 0. Nash proved that every game with a finite number players and strategies has at least one MNE [28]. Computing an MNE of a finite game is a central equilibrium computation problem.

We focus on the two-player ("bimatrix") case, where the input is two $m \times n$ payoff matrices (one for each player) with integer entries; with three or more players, the problem appears to be harder in a precise complexity-theoretic sense [15]. We emphasize that the two payoff matrices are completely unrelated, and need not be "zero-sum" like in Rock-Paper-Scissors. (When the two payoff matrices sum to a constant matrix, an MNE can be computed in polynomial time via linear programming; see e.g. [30, Chapter 1] for details.)

There is a non-obvious "guided search" algorithm for two-player games called the *Lemke-Howson*

"yes" or "no". Here we refer to the similar but more general *search* versions of P and NP, where for a "yes" instance, the deliverables include a correct solution.

algorithm [24]; see von Stengel [30, Chapter 3] for a careful exposition. This algorithm is a path-following algorithm in the spirit of local search, but it is not guided by an objective or potential function and thus does not prove that computing an MNE of a bimatrix game is in PLS. In conjunction with our earlier reasoning, however, the Lemke-Howson algorithm shows that the problem is not NP-hard unless $NP = coNP$ [25].

A complexity class that is related to but apparently different from PLS is $PPAD$, which stands for "polynomial parity argument, directed version". This class was defined in [32] to capture the complexity of computing MNE and related problems, such as computing approximate Brouwer fixed points. Its formal definition parallels that of PLS, with a $PPAD$ problem consisting of the minimal ingredients necessary to execute a Lemke-Howson-like path-following procedure (again easily phrased as three polynomial-time algorithms). A problem in $PPAD$ is $PPAD$-complete if every problem in $PPAD$ reduces to it in polynomial time; the complete problem is then polynomial-time solvable only if all problems in $PPAD$ are. Since $PPAD$ contains several well-studied problems that are not known to be solvable via a polynomial-time algorithm, a proof of $PPAD$-completeness can be interpreted as a significant intractability result.

A few years ago, the problem of computing an MNE of a bimatrix game was shown to be $PPAD$-complete [8, 12]. Thus, if $P \neq PPAD$, there is no general-purpose and computationally efficient algorithm for this problem, and in particular there is no general and tractable way for players to reach a Nash equilibrium in a reasonable amount of time. This hardness result casts doubt on the predictive power of the Nash equilibrium concept in arbitrary games. See the papers [8, 12] for the details of this tour de force result and the recent CACM article [13] for a high-level survey of the proof.

5 Future Directions

The astonishing rate of progress in algorithmic game theory, nourished by deep connections with other areas of theoretical computer science and a consistent infusion of new motivating applications, suggests that it will flourish for many years to come. There is a surfeit of important open research directions across all three of the AGT areas surveyed here, such as developing theory for the design and analysis of mechanisms for multi-parameter problems, for minimizing the inefficiency of equilibria (e.g., via a mediating network protocol), and for the computation of approximate equilibria. See [35] and the concluding sections of many chapters in [30] for more details and many concrete open problems.

A broad challenge, mentioned also in Shoham's recent CACM article [40], is to develop more appropriate models of agent behavior. All of the results described in this article, even the welfare guarantee of the simple second-price auction, depend on some kind of behavioral assumptions about the participants. Such assumptions are required to address modern applications, yet are largely foreign to the theoretical computer science mindset, which is characterized by minimal assumptions and worst-case analysis. But a number of new types of worst-case guarantees, coupled with novel behavioral models, have already begun to sprout in the AGT literature. For example: mechanism implementation in undominated strategies [3] and in ex post collusion-proof Nash equilibrium [31]; the price of total anarchy [6, 36]; and the complexity of unit-recall games [16]. We expect that these are only the vanguard of what promises to be a rich and relevant theory.

References

[1] A. Archer and É. Tardos. Truthful mechanisms for one-parameter agents. In *FOCS '01*, pages 482–491.

[2] B. Awerbuch, Y. Azar, A. Epstein, V. S. Mirrokni, and A. Skopalik. Fast convergence to nearly optimal solutions in potential games. In *EC '08*, pages 264–273.

[3] M. Babaioff, R. Lavi, and E. Pavlov. Single-value combinatorial auctions and algorithmic implementation in undominated strategies. *Journal of the ACM*, 56(1), 2009. Article 4.

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

[5] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, 1989.

[6] A. Blum, M. Hajiaghayi, K. Ligett, and A. Roth. Regret minimization and the price of total anarchy. In *STOC '08*, pages 373–382.

[7] D. Braess. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, 12:258–268, 1968.

[8] X. Chen, X. Deng, and S.-H. Teng. Settling the complexity of two-player Nash equilibria. *Journal of the ACM*, 2009. To appear.

[9] S. Chien and A. Sinclair. Convergence to approximate Nash equilibria in congestion games. In *SODA '07*, pages 169–178.

[10] J. E. Cohen and P. Horowitz. Paradoxical behavior of mechanical and electrical networks. *Nature*, 352(8):699–701, 1991.

[11] P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006.

[12] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a Nash equilibria. *SIAM Journal on Computing*, 2009. To appear.

[13] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a Nash equilibria. *Communications of the ACM*, 52(2):89–97, 2009.

[14] P. Dhangwatnotai, S. Dobzinski, S. Dughmi, and T. Roughgarden. Truthful approximation schemes for single-parameter agents. In *FOCS '08*, pages 15–24.

[15] K. Etessami and M. Yannakakis. On the complexity of Nash equilibria and other fixed points. In *FOCS '07*, pages 113–123.

[16] A. Fabrikant and C. H. Papadimitriou. The complexity of game dynamics: BGP oscillations, sink equlibria, and beyond. In *SODA '08*, pages 844–853.

[17] A. Fabrikant, C. H. Papadimitriou, and K. Talwar. The complexity of pure Nash equilibria. In *STOC '04*, pages 604–612.

[18] J. Feigenbaum, D. C. Parkes, and D. M. Pennock. Computational challenges in e-commerce. *Communications of the ACM*, 52(1):70–74, 2009.

[19] M. X. Goemans, V. Mirrokni, and A. Vetta. Sink equilibria and convergence. In *FOCS '05*, pages 142–151.

[20] M. O. Jackson. A crash course in implementation theory. *Social Choice and Welfare*, 18(4):655–708, 2001.

[21] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988.

[22] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. In *STACS '99*, pages 404–413.

[23] R. Lavi, A. Mu'alem, and N. Nisan. Towards a characterization of truthful combinatorial auctions. In *FOCS '03*, pages 574–583.

[24] C. E. Lemke and J. T. Howson, Jr. Equilibrium points of bimatrix games. *SIAM Journal*, 12(2):413–423, 1964.

[25] N. Megiddo and C. H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991.

[26] D. Monderer and L. S. Shapley. Potential games. *Games and Economic Behavior*, 14(1):124–143, 1996.

[27] R. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

[28] J. F. Nash, Jr. Equilibrium points in N-person games. *Proceedings of the National Academy of Science*, 36(1):48–49, 1950.

[29] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1/2):166–196, 2001.

[30] N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[31] N. Nisan, M. Schapira, G. Valiant, and A. Zohar. Best-reply mechanisms. Working paper, 2009.

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

[33] C. H. Papadimitriou, M. Schapira, and Y. Singer. On the hardness of being truthful. In *FOCS '08*, pages 250–259.

[34] R. W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

[35] T. Roughgarden. Algorithmic game theory: Some greatest hits and future directions. In *TCS '08*, pages 21–42.

[36] T. Roughgarden. Intrinsic robustness of the price of anarchy. In *STOC '09*.

[37] T. Roughgarden. The price of anarchy is independent of the network topology. *Journal of Computer and System Sciences*, 67(2):341–364, 2003.

[38] T. Roughgarden. Computing equilibria: A computational complexity perspective. *Economic Theory*, 2009. To appear.

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

[40] Y. Shoham. Computer science and game theory. *Communications of the ACM*, 51(8):75–79, 2008.

[41] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game Theoretic and Logical Foundations*. Cambridge University Press, 2008.

[42] K. Steiglitz. *Snipers, Shills, and Sharks: eBay and Human Behavior*. Princeton University Press, 2007.

[43] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1):8–37, 1961.