Five AI Challenges in Strategyproof Computing

David C. Parkes
Division of Engineering and Applied Sciences,
Harvard University,
33 Oxford Street,
Cambridge, MA 02138
parkes@eecs.harvard.edu

Abstract

Computational systems are now distributed by default, and designed, owned and used by multiple self-interested parties. In the face of this growing system complexity, we need a unifying design paradigm, that supports continued innovation and competition while promoting easy deployment, operation, and use. Focusing on the central problem of *resource allocation*—the arbitration of shared resources among the competing demands of users—we introduce a paradigm of *strategyproof computing*, a vision in which individual users can treat other resources as their own. We layout the benefits of strategyproof computing, and identify five AI challenges in making this vision a reality.

1 Introduction

The widespread deployment of high-speed Internet access, including rapid build-out of WiFi capability, is leading to massively distributed computing systems, that are designed, owned and used by multiple self-interested parties. Witness the explosion of next-generation systems, including peer-to-peer systems, pervasive systems [48; 39], computational grids [14], 3G cell-phone services, and server farms providing on-demand computing for businesses.

Two points cannot be understated. First, these systems are truly *open*, with innovation and competition, and little central control beyond agreement on fundamental communication protocols such as those within the current Internet. This open nature is vital for the continued design and deployment of new and rich capabilities, and must be explicitly recognized and embraced.[1] Second, the actors in these systems are truly self-interested (at least to the same extent that people and businesses in human societies are self-interested) and heterogeneous in both their capabilities and their goals. This self-interest is natural as computing systems increasingly adopt characteristics of economies, and must be explicitly recognized and embraced.

[1]Indeed, the original Internet design was built around an end-to-end principle, that explicitly left the "smarts" to the edges of the network to allow for the development and deployment of rich and unanticipated applications [36].

Resource allocation—the arbitration of shared resources among the competing demands of self-interested users—is among the most basic functions of a distributed computing system. Taking this as our focus, we introduce the paradigm of *strategyproof computing*, a vision which embraces and enables innovation and competition by defining principles while promoting "plug-and-play" build-out, *and* addresses self-interest by allowing individual users to treat other resources as their own. Strategyproof computing builds on two intellectual threads: the theory of economic mechanism design, and market-based approaches to resource allocation. In particular, we seek to use mechanism design to design distributed systems that *simplify* the game-theoretic problem facing agents, while promoting competition across mechanisms and the development and deployment of new mechanisms around new services.

Strategyproof computing can be viewed within the emerging research agenda of *autonomic computing*. The thesis of autonomic computing is that the complexity of distributed systems, with multiple tuning parameters, massive scale, heterogeneity, and dynamic interactions, which require systems that can *self-configure* and *self-repair*. Naturally, one of the key challenges in the autonomic-computing agenda is to address the issue of decentralized autonomy, namely how to define standards for interoperability and how to define high-level protocols to promote the effective collective behavior in multiagent systems. Strategyproof computing provides a compelling paradigm with which to address this particular challenge.

Indeed, strategyproof computing can be viewed as a subset of autonomic computing. Within strategyproof computing there are both autonomous elements— individual mechanisms of limited scope —and aspects of adaptive and decentralized control, with the emergence of new mechanisms and the tuning of existing mechanisms. As such, strategyproof computing will require an extensive use of AI techniques such as learning, search, description languages, and concepts of bounded-rationality.

In a companion paper we lay out a related set of systems-related challenges [29]. Here, we focus on some of the AI-related research challenges in making the vision of strategyproof computing a reality.

2 Strategyproof Computing

In this section, we first layout the guiding principles and intellectual threads that underpin the paradigm of strategyproof computing. Continuing, we define the central components of strategyproof computing.

SPC is best distinguished from previous proposals for market-based methods for the control of distributed systems (e.g. [12; 41; 47; 18; 16; 4; 42; 23; 35; 11; 44]) in that it is designed for large and complex systems in which it is infeasible to suppose a single entity with the authority to impose and implement a single resource allocation mechanism. Instead, SPC is an infrastructure-level effort to support the massive deployment and interoperation of competing economic mechanisms. We contend that no single mechanism can possibly be appropriate for all requirements in a heterogeneous multiagent system.

Central research questions include that of *design*, i.e. the construction of local mechanisms with robust incentive properties, *validation*, i.e. the validation of the incentive properties of mechanisms, and *composition*, i.e. the development of algorithmic methods and mechanisms that make it easy for participants to compose services across multiple mechanisms. We return to these questions in Section 3.

Strategyproof computing (SPC) is most applicable in distributed computing environments in which resource allocation decisions are ubiquitous and exposed to users (or their computational agents), and in which users (or their agents) are sophisticated and have figured out how to behave in order to maximize their individual utility.

2.1 Guiding Principles.

We start with five guiding principles for the design of standards to promote effective collective behavior in distributed computational systems.

incentives-first Resource allocation decisions should be made in a framework that considers the incentives of participants and assumes that users will try to extract as much individual benefit from a system as possible. Models that assume cooperation and ignore incentives are not useful when defining systems with useful long-term equilibrium behavior.[2]

utility-based Resource allocation decisions should be made in a framework that considers the *utility* of users for different outcomes as the basis for arbitration. Moreover, this utility should be measured in terms of a common currency, to allow for an explicit tradeoff between multiple users and across between different components of a system.[3]

simple As designers we should seek to simplify the decisions facing participants in distributed multiagent systems, such as the optimal statements to make about the local capabilities (e.g. available storage space, speed of Internet connection, etc.), or the optimal statements to make about the utility of a party for different levels of resources (e.g. bandwidth connectivity).

open Systems must be *open* to allow for innovation and competition in the design of new computing services. Rather than advocating one particular mechanism for resource allocation, we propose to provide an infrastructure in which multiple mechanisms can be designed and deployed within an open system.

decentralized The control structure in distributed computing systems must be decentralized, both to respect the autonomy of the nodes that own the property rights to resources, but also for reasons of computational scale and timeliness of information.

[2]Current systems seldom expose the rules of resource allocation arbitration to participants, and instead rely on users downloading cooperative software layers. A good example is Kazaa, which relies on enough users choosing to provide files for upload to other users. Those systems that do expose the rules of resource allocation, such as emerging computational grids for scientific experiments (e.g. GriPhyN), currently provide users with a simple priority-based method to request and allocate resources. The allocation mechanisms (e.g. Condor) are not incentive-compatible, and users can improve their allocation by misstating priorities for jobs. While current grids appear to be working well for small groups of friendly scientists local to the organization that provides the resources, incentive issues will surely arise as grids are expanded to include corporations and inter-organization computing.

2.2 Intellectual Threads.

In support of these principles— *incentives-first, utility-based, simple, open*, and *decentralized* —we seek to infuse fundamental ideas from economic mechanism design and markets into the microstructure of distributed system design, while retaining the principle of supporting the decentralized and open build-out of new functionalities and services.

We build on two intellectual threads: market-based approaches and economic mechanism design. *Price-based markets* are interesting because when equilibrium prices, when exist they support an efficient allocation of resources. Market-based schemes (e.g. [12]) are intrinsically decentralized, with resource allocations left to individual participants (and coordinated via equilibrium prices). Moreover, there are simple (but usually centralized) tâtonnenment schemes to adjust prices towards equilibrium, with guaranteed convergence properties in classical economies [10].

Economic mechanism design (MD) presents an alternative approach, in which each participant is modeled as a game-theoretic agent and a protocol is selected to achieve particular desiderata (e.g. allocative efficiency, fairness, or maximal throughput) in equilibrium (e.g. [20; 25]). Mechanism design formulates the design of an allocation scheme as a constrained optimization problem, in which an optimal set of rules are selected subject to incentive constraints [28]. The theory of mechanism design extends to very general problems, and there are a number of celebrated mechanisms addressing various requirements (e.g., [46; 17]).

However, both approaches have their limits. First, price-based methods are inapplicable when resources are indivisible or when the preference structure of the economic agents

[3]Shenker (e.g. [40]) has long advocated the use of utility-based metrics rather than more traditional systems metrics such as throughput and utilization. Recently, Vadhat [43] has promoted a utility-based computing research agenda.

are more expressive and complex (such as "I only want A if I also get B"), because equilibrium prices often fail to exist [5]. Second, although MD presents an elegant mathematical theory it is often unworkable in practice (e.g. [21]). Difficulties include problems that are not analytically solvable for any but the simplest settings; formulations that are centralized and ignore the computational constraints on a solution; lack of general agreement on desiderata; and inability to capture the entire *scope* of a problem within a single mechanism.

2.3 Basic Elements.

In strategyproof computing we seek to combine the decentralized resource-allocation aspects and open nature of traditional price-based market models with the generality and clarity of mechanism-based approaches.

We mentioned "plug and play" in the introduction, as a useful characteristic of a SPC system. Briefly, the intention is that new services and new resources can be **plugged** into a SPC infrastructure, along with an accompanying *public interface* to describe their incentive- and performance characteristics. Then, the existing user base can seamlessly start to **play** the new services and resources, because the appropriate method with which a self-interested user should interact with a new service is already clear. The companion paper on systems related issues makes these components more precise.

For now, we content ourselves with defining the central elements of the SPC paradigm.

Limited Scope.
Individual parties can design and deploy mechanisms with limited scope, to implement resource allocation decisions for the resources and overlay services under their control. Realistic mechanisms must necessarily have limited scope, both to maintain a reasonable computational and informational scale and because the sphere of influence and control for a single party must necessarily be limited (see [49] for a related discussion).

Strategyproof.
Each mechanism must be *locally strategyproof* (LSP). Technically, this means that truth-revelation is a dominant strategy for a user that can express her utility for resources and services within the scope of a mechanism *independently* of other events in the world, and chooses to submit requests for those resources *exclusively* to that mechanism. For example, if I know that I want ice-cream for desert irrespectively of the kind of pie that I acquire, then I should be able to truthfully express my value for different flavors of ice-cream to a LSP mechanism.

A little more formally, but sticking with a special case, consider a set of items $\mathcal{G}$ that are partitioned $\Pi = \{G_1, \ldots, G_L\}$ such that $G_l \cap G_{l'} = \emptyset$ for all l, l' and $\bigcup_l G_l = \mathcal{G}$. Now, consider a family of LSP mechanisms $\mathcal{M}_l$, each defined on one of the components G_l of the partition. As long as the valuation $v_i(S) \geq 0$ of agent i for bundle $S \subseteq \mathcal{G}$ is *additive* across the partition, i.e. $v_i(S) = \sum_l v_{i,l}(S_l)$ where $S_l = S \cap G_l$, for restricted valuations $v_{i,l}(\cdot)$, then agent i should submit its true valuation function v_i to each mechanism (perhaps within a bidding language that restricts this valuation to the subset of goods G_l).

We advocate locally strategyproof mechanisms because of their *simple* game-theoretic properties. A self-interested participant need not model the strategies and behaviors of other parties [45]. This, the computational benefits and strategic robustness provided by strategyproofness are very compelling. Indeed, one driving vision of strategyproof computing is that parties can begin to treat the resources of the network as though they were their own, with no need to game the system.

By itself, local strategyproof (LSP) mechanisms are not sufficient to completely remove any need for game-theoretic reasoning. In general, it is not necessary that the patchwork of mechanisms exactly partition the outcome space, nor that agents' valuations exactly decompose along the boundaries of the patchwork. In this case, there remain strategic questions facing participants that cannot decompose their local problems to fit the decomposition in the world provided by the patchwork of mechanisms, and also for participants that want to get the best deal across multiple mechanisms.

However, as we noted earlier, it is most definitely necessary to relax the extreme position of MD theory in which one can (informally speaking) build one combinatorial auction for the entire world. We return to the question of composition across LSP mechanisms in the next section, when we identify a number of important research challenges.

Open.
Strategyproof computing allows anyone with property rights, for example resource or service providers, or intermediaries, to define and deploy a LSP mechanism. We envision an open and competitive market for mechanisms, in which economic forces should naturally lead this to the emergence of mechanisms with the right scope and the right complexity. This decision about scope represents a tradeoff between providing a large enough scope to sufficiently simplify the game-theoretic decision facing a participant— for example bringing resources that are *complementary* for a large number of users into the same scope —while maintaining a small enough scope to build computationally reasonable resource allocation mechanisms.

3 AI Challenges

Progress in this research agenda calls for a careful synthesis of a number of AI techniques. In this section we identify five AI-related challenge areas in making progress towards this strategyproof computing vision.

A companion paper lays out some of the *systems-centric* challenges [29]. These systems challenges include those of providing *development tools* for LSP mechanisms, developing methods to validate and certify LSP mechanisms, issues of *scalability and decentralization*, *deployment issues*, and the need to find effective methods to *measure* the performance of strategyproof systems.

3.1 Design

We need to build useful LSP mechanisms, with good computational properties—for both agents and the mechanism infrastructure— and good economic properties. There has already been some progress in designing strategyproof mechanisms (e.g. [2; 24; 32; 27]). For example, a general paradigm has emerged that observes that the family of strategyproof mechanisms for combinatorial auction settings can be completely characterized by price-based mechanisms in which each agent faces a set of prices that are entirely determined by the information announce by other agents. Computational mechanism design (see [33, chapter 3] for an introduction) has considered topics such as the design of *bidding languages* (e.g. [8; 31]), and winner-determination algorithms that can leverage the structure in problems (e.g. [38; 6]).

Of course, the intention in SPC is not to include as a subcomponent the entire agenda of computational mechanism design. Rather, it seems more pertinent to focus on three important directions that will be required for successful mechanisms within computational systems: a) **approximation** and relaxed definitions of strategyproofness (e.g. [13; 22]); b) **online** mechanisms in which systems are "always on" and agents can dynamically arrive and leave over time (e.g. [15; 30]); and c) **two-sided** mechanisms in which there is intermediation across both buyers and sellers of resources (e.g. [34; 26; 3]).

3.2 Description

We need languages to *describe* the resources provided by a mechanism, the semantics of messages that a user can send to a mechanism, and the incentive properties of mechanisms. In particular, the incentive properties must be stated with respect to some *assumptions* about the utilities of users. For example, "my mechanism is LSP for a user that only wants access to WiFi bandwidth for a single minute." In the context of a market for mechanisms, it also seems useful to enable a mechanism to describe the *utility* that users have received from its service to promote market transparency. These utility statistics should aid in the composition problem that faces participants in systems with multiple mechanisms.

3.3 Verification

Perhaps the central role of the *infrastructure*, in the support of strategyproof computing, is in its ability to verify the stated incentive properties and utility statistics of a mechanism, or at least to enable the verification of these properties by other parties. This is essential, because the truth-revealing equilibrium of well-designed mechanisms can quickly unravel without the ability to commit to a particular set of rules. Consider a Vickrey (second-price sealed-bid) auction in which the auctioneer cannot commit to clearing the auction at the second highest price. Such an auction would degenerate into a first-price auction.

One interesting direction would require that mechanisms, and users, maintain a (partial) audit trail, to record sequences of requests for resources and decisions made. Collected (reported utility, outcome) tuples from successive queries can be used to check that no user could have received a better outcome with respect to her reported utility by reporting the utility of another user. This can be viewed as a problem in reinforcement learning: given the data try to learn a better strategy than truthful reporting. The challenge is to investigate how much data is necessary to provide enough counterfactual information to validate a mechanism's incentive properties, and to understand the best way to use existing data and to augment the data, as necessary, with additional queries. In Ng et al. [29] we term this distinction the difference between *passive*- and *active* checking.

3.4 Composition

We need to be able to compose the resources and services provided across multiple mechanisms. This problem occurs because of the limited scope of individual mechanisms. There has been some progress on the design of trading agents to participate in multiple sequential (e.g. [7]) and parallel (e.g. [1]) auctions, but less emphasis on the design of general methods to support composition. The basic problem is one of *exposure*, which can occur when what is desired in one negotiation is contingent on the outcome of another negotiation.[4] One natural idea is to provide LSP mechanisms that offer *real options* to users, such that the most useful option is offered through a truthful statement about a user's utility. This reduces the problem to a decision-theoretic problem, with a user able to select the appropriate combination of options once they have been awarded. We are aware of only one mechanism that grants options [19], although the options in that mechanism are not designed to be held for any length of time after the auction clears. Options are closely related to the idea of *leveled commitment* contracts, that have been proposed to address exposure to risk in the setting of decentralized bilateral negotiation between multiple parties [37].

3.5 Learning

Given that mechanisms are of limited scope, and that the design of individual mechanisms represents a difficult tradeoff between expressiveness and complexity, it seems that machine learning techniques will be useful to promote the automatic adaption of mechanisms of the right scope and to identify and introduce new mechanisms to act as intermediaries for certain combinations of resources.[5] One very useful property of SPC is that the *information* is available to enable learning because users are revealing utility information directly through their requests to mechanisms.

For a fixed set of resources and in a fixed environment, it will be interesting to explore whether reinforcement learning, and search, can be used to adapt towards mechanism rules with useful properties. Similarly, given a limit on the number of mechanisms, and again for a fixed environment, it will be interesting to explore whether reinforcement learning (or perhaps evolutionary computing methods) can be used to recognize a useful resource scope for mechanisms.

Another related topic is that of the automatic tuning of the parameters within a particular mechanism. The family of false-name bid proof mechanisms due to Yokoo et al. [50]

[4]This was recognized early on in the FCC auction design problem, with bidders in a non-combinatorial auction becoming exposed to one license without being able to win the complementary license [9].

[5]This is akin to the role of arbitrager agents in stock markets, where the role of arbitrage is to identify and fix pricing inefficiencies across markets and improve liquidity by executing complex deals such as swaps and portfolio rebalancing trades.

are examples of mechanisms that are tunable, e.g. via reserve prices. One subtle problem with this will be to maintain incentives along the path of learning.

4 Conclusions

We have introduced the paradigm of strategyproof computing, in which resources and services are wrapped around an economic mechanism interface that provides local strategyproofness. Individual mechanisms compete in an open marketplace, with deployment enabled through a shared infrastructure that enables the description and validation of the incentive properties of mechanisms. We identified five AI-specific challenges areas in strategyproof computing, that provide a compelling and rich research agenda.

We believe that strategyproof computing presents a compelling vision to simplify the development of decentralized systems in which there are multiple self-interested parties and incentives matter. Further, we believe that strategyproof computing provides an intellectually stimulating research agenda around at the interface between AI, systems, and economics.

Acknowledgments

This work described in this note represents an ongoing research by the author and his colleagues at Harvard University. The author wishes to gratefully acknowledge an energetic and active collaboration with Chaki Ng, Margo Seltzer, Jeff Shneidman, and also more recent conversations with John Wroclawski and Michael Wellman. This work is supported in part by NSF grant IIS-0238147.

References

[1] P Anthony and N R Jennings. Developing a bidding agent for multiple heterogeneous auctions. *ACM Trans. On Internet Technology*, 2003. to appear.

[2] A Archer and E Tardos. Truthful mechanisms for one-parameter agents. In *Proc. 42nd IEEE Symp. on Foundations of Computer Science*, 2001.

[3] Moshe Babaioff and Noam Nisan. Concurrent auctions across the supply chain. In *Proc. 3rd ACM Conference on Electronic Commerce*, pages 1–10. ACM Press, 2001.

[4] J S Banks, J O Ledyard, and D Porter. Allocating uncertain and unresponsive resources: An experimental approach. *The Rand Journal of Economics*, 20:1–25, 1989.

[5] Sushil Bikhchandani and Joseph M Ostroy. The package assignment model. *Journal of Economic Theory*, 2002. Forthcoming.

[6] Craig Boutilier. Solving concisely expressed combinatorial auction problems. In *Proc. 18th National Conference on Artificial Intelligence (AAAI-02)*, 2002.

[7] Craig Boutilier, Moises Goldszmidt, and Bikash Sabata. Sequential auctions for the allocation of resources with complementarities. In *Proc. 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 527–534, 1999.

[8] Craig Boutilier and Holger Hoos. Bidding languages for combinatorial auctions. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, 2001.

[9] Mark M Bykowsky, Robert J Cull, and John O Ledyard. Mutually destructive bidding: The FCC auction design problem. *J. of Regulatory Economics*, 2000. To appear.

[10] J Q Cheng and M P Wellman. The WALRAS algorithm: A convergent distributed implementation of general equilibrium outcomes. *Computational Economics*, 12:1–24, 1998.

[11] B Chun and D Culler. Market-based proportional resource sharing for clusters. Technical Report CSD-1092, University of California, Berkeley, 2000.

[12] Scott H Clearwater, editor. *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1996.

[13] Joan Feigenbaum and Scott Shenker. Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13, 2002.

[14] I T Foster and C Kesselman. Computational grids. In *VECPAR*, pages 3–37, 2000.

[15] Eric Friedman and David C. Parkes. Pricing WiFi at Starbucks– Issues in online mechanism design. In *Fourth ACM Conf. on Electronic Commerce (EC'03)*, 2003. Poster. Extended version at http://www.eecs.harvard.edu/econcs/online.pdf.

[16] Robert L Graves, Linus Schrage, and Jayaram Sankaran. An auction method for course registration. *Interfaces*, 23(5):81–92, 1993.

[17] Theodore Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[18] Bernardo A Huberman and Scott Clearwater. A multi-agent system for controlling building environments. In *Proc. 1st International Conference on Multi-Agent Systems (ICMAS-95)*, pages 171–176, 1995.

[19] A Iwasaki, M Yokoo, and K Terada. A robust open ascending-price multi-unit auction protocol against false-name bids. In *Proc. 5th ACM Conf. on Electronic Commerce*, 2003.

[20] Matthew O. Jackson. Mechanism theory. In *The Encyclopedia of Life Support Systems*. EOLSS Publishers, 2000.

[21] Jayant Kalagnanam and David C Parkes. Auctions, bidding and exchange design. In David Simchi-Levi, S David Wu, and Z Max Shen, editors, *Supply Chain Analysis in the eBusiness Era*. Kluwer, 2003.

[22] Anshul Kothari, David C. Parkes, and Subhash Suri. Approximately-strategyproof and tractable multi-unit auctions. In *Fourth ACM Conf. on Electronic Commerce (EC'03)*, 2003. To appear.

[23] J Kurose and R Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Trans. on Computers*, 38:705–717, 1989.

[24] Daniel Lehmann, Liadan Ita O'Callaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, September 2002.

[25] Andreu Mas-Colell, Michael D Whinston, and Jerry R Green. *Microeconomic Theory*. Oxford University Press, 1995.

[26] R Preston McAfee. A dominant strategy double auction. *J. of Economic Theory*, 56:434–450, 1992.

[27] A. Mu'alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *Proc. 18th National Conference on Artificial Intelligence (AAAI-02)*, 2002.

[28] Robert B Myerson. Optimal auction design. *Mathematics of Operation Research*, 6:58–73, 1981.

[29] Chaki Ng, David C. Parkes, and Margo Seltzer. Strategyproof computing: Systems infrastructures for Self-interested parties. Technical report, Harvard University, 2003.

[30] Chaki Ng, David C. Parkes, and Margo Seltzer. Virtual Worlds: Fast and Strategyproof Auctions for Dynamic Resource Allocation. In *Fourth ACM Conf. on Electronic Commerce (EC'03)*, 2003. Poster. Extended version at http://www.eecs.harvard.edu/econcs/virtual.pdf.

[31] Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proc. 2nd ACM Conf. on Electronic Commerce (EC-00)*, pages 1–12, 2000.

[32] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[33] David C Parkes. *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, May 2001. http://www.cis.upenn.edu/˜dparkes/diss.html.

[34] David C Parkes, Jayant R Kalagnanam, and Marta Eso. Achieving budget-balance with Vickrey-based payment schemes in exchanges. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, 2001.

[35] Ori Regev and Noam Nisan. The POPCORN market-an online market for computational resources. In *Proceedings of the first international conference on Information and computation economies*, pages 148–157. ACM Press, 1998.

[36] J H Saltzer, D P Reed, and D D Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, pages 277–288, 1984.

[37] Tuomas Sandholm and Victor Lesser. Leveled commitment contracts and strategic breach. *Games and Economic Behavior*, 35:212–270, 2001.

[38] Tuomas W Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. CABOB: A fast optimal algorithm for combinatorial auctions. In *Proc. 17th Int. Joint Conf. on Artificial Intelligence*, 2001.

[39] M Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, pages 10–17, Aug 2001.

[40] S Shenker. Making greed work in networks: A game-theoretic analysis of switch service disciplines. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 47–57, 1994.

[41] K Steiglitz, M L Honig, and L M Cohen. A computational market based on individual auction. In Clearwater [12], chapter 1, pages 1–27.

[42] M Stonebraker, R Devine, M Kornacker, W Litwin, A Pfeffer, A Sah, and C Staelin. An economic paradigm for query processing and data migration in Mariposa. In *Proc. 3rd Int. Conf. on Parallel and Distributed Information Systems*, pages 58–67, 1994.

[43] A Vahdat, J Chase, R Braynard, D Kostic, and A Rodriguez. Self-organizing subsets: From each according to his abilities, Teach according to his needs. In *Proc. 1st Int. Peer to Peer Symp. (IPTPS)*, 2002.

[44] H Varian and J K MacKie-Mason. Generalized Vickrey auctions. Technical report, University of Michigan, 1995.

[45] Hal R Varian. Economic mechanism design for computerized agents. In *Proc. USENIX Workshop on Electronic Commerce*, 1995. Minor update, 2000.

[46] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[47] Carl A Waldspurger, Tad Hogg, Bernado Huberman, Jeffrey O Kephart, and W. Scott Stornetta. Spawn: A distributed computational economy. *IEEE Trans. on Software Engineering*, 18:103–117, 1992.

[48] M Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–100, Sept 1991.

[49] Michael P Wellman and Peter R Wurman. Market-aware agents for a multiagent world. *Robotics and Autonomous Systems*, 24(3–4):115–125, 1998.

[50] M Yokoo, Y Sakurai, and S Matsubara. Robust multi-unit auction protocol against false-name bids. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, 2001.