Fundamental Open Questions In Distributed Mechanism Design

Scott Shenker
cowritten with Joan Feigenbaum

Game Theory and Computer Science

Computer science:

- Focus: computational efficiency
- Nodes: obedient or adversarial

Game theory:

- Focus: incentives
- Nodes: selfish

Reality:

- Both computation and incentives matter
- Must look at complexity of economic mechanisms
- Combinatorial auctions are a compelling example

Seminal paper:

- *Algorithmic Mechanism Design* (AMD) [Nisan-Ronen]
- Most AMD work focuses on *centralized complexity*

This Talk

Considers *distributed resource allocation problems*

- Users are distributed
- Resources are distributed
- Computation is distributed

Focuses on *network complexity*

- Assume that the economic mechanism involves a distributed computation carried out over a network
- *Network complexity* measures the computational and communication efficiency of the distributed algorithm

Theme: *distributed algorithmic mechanism design* (DAMD)

Distributed Algorithmic Mechanism Design

Not yet mature:

- No particularly compelling example yet
- Some isolated results, but no coherent framework
- Many fundamental issues unresolved
- Most of these are never even addressed

Purpose of this talk:

- Encourage discussion of these unresolved issues
- Pose both general and specific open questions

Outline

- Review of Mechanism Design Paradigm
- Four Distributed Resource Allocation Problems
- Six Fundamental Questions (long)
- One Final Comment about Canonical Hard Problems

Mechanism Design Paradigm (review)

Resource allocation problem:

- Set of possible allocations or outcomes $\mathcal{O}$
- Utilities u_i over $\mathcal{O}$, $u_i \in \mathcal{U}$
- Social Choice Function (SCF):
  - $F : \mathcal{U}^n \mapsto \mathcal{O}$
- Social Choice Correspondence (SCC):
  - $H : \mathcal{U}^n \mapsto 2^{\mathcal{O}}$

Strategyproof SCFs

F is strategyproof if

- $u_i(F(u)) \geq u_i(F(u|^i v_i)$ for all $v_i \in \mathcal{U}$

Revelation or *direct* mechanism:

- No incentive to lie, modulo collusional behavior
- Achieves truthful outcome

Examples:

- VCG mechanisms

Why Not Always Use Strategyproof Direct Mechanisms?

Strategic reason:

- Greatly limits choice of SCF
  - General: Gibbard-Satterthwaite
  - Differentiable: Satterthwaite-Sonnenschein
  - Exchange: Barbera-Jackson
  - Single-peaked: Moulin, Sprumont
  - ...

Practical reasons:

- Communication overhead
- Sometimes agents don't know utility explicitly
  - Probably quite common in network resource cases

(Indirect) Mechanism Design Paradigm

Pick Social Choice Function/Correspondence

Solution concept: C

- C_G describes the selfish outcome in game G
- Models reality, not something you can design

Design mechanism $< M, S >$

- $M : S^n \mapsto \mathcal{O}$
- Induces game $< G, S >$ with $G_i(s) = u_i(M(s))$
- Denote solution concept by $C_M(u)$

Desired Property:

- SCF: $M(C_M(u)) = F(u)$ for all $u \in \mathcal{U}^n$
- SCC: $M(C_M(u)) \subseteq H(u)$ for all $u \in \mathcal{U}^n$

Results:

- With the common solution concepts (*e.g.*, Nash) this approach can implement many nonstrategyproof SCFs

Distributed Resource Allocation Problems

Distributed AMD:

- Considers all distributed resource allocation problems
- The Internet is the biggest and most successful distributed system, making it a natural source of DAMD problems

Four examples (in following slides):

- All Internet-related
- Varying degrees of reality
- Varying degrees of distributed mechanism design

Example #1: Congestion Game

Problem:

- Agent utilities $u_i(r_i, d_i)$
- Delays d function of rates r: $d = D(r)$
- D represents local packet scheduling algorithm

Results:

- If D=FIFO, Nash is very inefficient (for large n)
- If D=FQ, Nash is fair, reasonably efficient

Comment:

- Distributed resources, local (not centralized) mechanism

Example #2: Alternate Path Game

Problem: (simplest form)

- Flows choose from n parallel links
- Congestion on links function of their utilization
- Compare worst-case Nash to social optimal [Koutsoupias-Papadimitriou]

Results: [Roughgarden-Tardos]

- Nash allocation bad, but increasing bandwidth by factor of two offsets selfishness

Comment:

- Distributed resources, but no mechanism

Example #3: Interdomain Routing Game

Problem: [Feigenbaum, Papadimitriou, Sami, S]

- Routing among ASs, currently handled by BGP
- Each AS incurs a cost, and gets paid, for carrying traffic
- Want packets to travel on true lowest-cost paths
- Use VCG pricing scheme so ASs reveal their true costs
- Like *shortest-path* problem [Nisan-Ronen, H-S] except
  1. Nodes (ASs) are strategic entities
  2. Consider all source-destination pairs, naively introducing additional n^2 complexity.
  3. Distributed BGP-like computational model

Results: (Feigenbaum's talk)

- We can calculate VCG prices without greatly increasing network complexity of BGP

Example #4: Multicast Cost Sharing

Problem:

- Multicast transmission to multiple receivers along a shared delivery tree
- Receivers have utilities u_i for receiving transmission
- Traversing each link l costs c_l
- Mechanism decides which receivers get transmission and how much to charge
- Want a strategyproof pricing mechanism that is budget-balanced and efficient
- Game-theoretic results: [Moulin, S]
  - Classical result: can't do SP, BB, and Eff
  - Single "best" SP and Eff mechanism: MC (VCG)
  - Single "best" SP and BB mechanism: SH

Results: [Feigenbaum, Krishnamurthy, Papadimitriou, Sami, S]

- MC can be computed in one bottom-up pass followed by one top-down pass of the tree, resulting in each link having at most one message traversal in each direction
- SH can require a linear number of messages crossing at least one link
- MC and SH are two extreme cases:
  - MC is as “easy” as possible
  - SH is as “hard” as possible
- More recent results:
  - Lower bounds on SH apply to a wide class of BB mechanisms (Krishnamurthy’s talk)
  - A roughly approximate version of the SH mechanism has low network complexity (Sami’s talk)

(Mostly) Common Features

Setting:

- Little information about infrastructure and other players
- Dynamic environment
- Asynchronous

Not standard game-theoretic setting:

- Game theory has treated each of these issues individually, but not jointly
- The confluence is crucial

Computational constraints:

- Communication/computation costs are important
- Mechanisms should have low network complexity

Two Classes of Issues

DAMD raises two different classes of issues:

- Game-theoretic issues in distributed systems
- Distributed computational issues in resulting economic mechanisms

Six Fundamental Questions

1. What is the strategic model?
2. What is the solution concept?
3. Is the selfish outcome sufficiently bad?
4. What can be implemented through mechanism design?
5. What can be feasibly implemented?
6. What can be approximately implemented?

Q1: What is the Strategic Model?

Defining basic aspects:

- Who are the strategic agents?
- How much information do they have?
- Are they collusional or not?
- What can be observed by other agents?
- Is the environment static or dynamic?
- . . .

These are questions about reality, not mathematics:

- Analysis of the problem depends on these assumptions
- Need to make the assumptions explicit

Q2: What is the Solution Concept?

For indirect mechanisms, this is the most basic question:

- What is the result of selfish play?

Answer:

- It depends greatly on the strategic model
- Even given specific strategic model it isn’t always clear

Standard Game Theory Setting

Environment:

- Static (and known) infrastructure
- Synchronous play

Standard Solution Concepts:

- One-shot game with common knowledge of G:
  - Rationalizable strategies
  - Nash and refinements thereof
  - ...
- Repeated game knowing only own payoff function G_i:
  - Agents learn what strategies to play from history
  - Solution concept is the set of asymptotic plays, which depends on the nature of learning
  - Adaptive: Serially undominated set [Milgrom-Roberts]
  - Calibrated: Correlated equilibria [Foster-Vohra]
  - ...

An Internet-like Setting

Environment:

- Prolonged, not one-shot, interactions
- No information about payoff function G
  - Even your own payoff function G_i
  - Only know the result of actual play
- Dynamic (and unknowable) infrastructure
- Moderate or extreme asynchrony

Modeling choices:

- Prolonged interaction: repeated game
- Low-information: agents learn what strategies to play
- Dynamic: must adapt to changes in environment
- Asynchrony: agents learn at different speeds

The Corresponding Internet-like Solution Concept:

- No one knows!

Preliminary Work

Several papers: [Friedman, Greenwald, Shor, Sopher, S]

- Theory, simulation, and experiments

Theory:

- Assumed agents use *reasonable* learning algorithms
- Defined minimal criteria for *reasonable* learning
- Derived bounds for the resulting solution concept

Simulations:

- Results consistent with theory
- In this setting, agents don't converge to a small asymptotic set even when using very sophisticated learning algorithms

Preliminary Work (cont'd)

Experiments:

- Human subjects playing for real money
- Convergence sometimes "worse" than in theory
  - Experimentation cascades
- Too early to make sweeping generalizations

Caveat:

- This is just one Internet-like context (but common)
- Applies to several of the examples presented earlier
- But there are many other Internet problems that don't fit this model and to which traditional solution concepts may apply

Open Questions about Solution Concept

General:

- What are the appropriate solution concepts for Internet-like settings?
  - Can't just assume Nash is the right solution concept
- Can you design mechanisms to be more "learnable"?
  - But must be in the agents' self-interest

Specific:

- If you allow only limited asynchrony, does the solution concept change?
- Can you scalably transform the model from a low-information environment to a high-information one by giving the agents more information?

Q3: Is the Selfish Outcome Sufficiently Bad?

If not, then don't bother with mechanism design!

Previous work:

- Koutsoupias-Papadimitriou formulation:
  - Compare social optimum to worst-case Nash
- Roughgarden-Tardos formulation:
  - Look at increasing resources: Alternate Path Game
  - Degradation in the Nash outcome is offset by doubling the bandwidths
- But the congestion game is different
  - Total Nash utility vanishes in the limit of large n
  - Adding fixed fraction of bandwidth doesn't help

Open Questions about Selfish Outcomes

General:

- Can we characterize the class of problems where increasing the resources by a fixed fraction offsets selfish behavior?

Specific:

- Does the Roughgarden-Tardos result continue to hold with other solution concepts?

Q4: What Can Be Implemented?

Strategyproof direct mechanisms:

- Strategyproofness usually a very restrictive requirement

Indirect mechanisms:

- Depends greatly on the solution concept
- Don't know solution concept for many Internet settings
- For the Internet-like setting and solution concept considered by Friedman *et al.*:
  - Only a subset of strategyproof SCFs are implementable

Strategyproof vs Nonstrategyproof Mechanism Design

Both approaches have significant limitations:

- Strategyproof mechanisms
  - Small subset of SCFs
  - Computational and practical limitations
- Nonstrategyproof (indirect) mechanisms
  - May only implement an even smaller subset of SCFs in some Internet-like settings

Role of nonstrategyproof (indirect) mechanisms:

- Game theory: used to implement a wider set of SCFs
- DAMD: in some Internet-like settings they may only be useful for overcoming computational and practical limitations

Open Questions about Implementation

General:

- What social choice functions and correspondences can be implemented with Internet-like solution concepts?

now leaving pure game theory behind. . .

Q5: What Can Be Feasibly Implemented?

Three separate feasibility concerns.

- Complexity
- Integrity
- Privacy

Complexity

Must consider both:

- Computational complexity at each node
- Communication complexity (between nodes)

The term *network complexity* refers to both

To evaluate network complexity:

- Need to define computational model of network

Computational Model

Options:

- Traditional TCS computational models (*e.g.*, PODC)
  - It isn't clear how realistic these models are
- Use existing protocols as computational substrate
  - Example: BGP in the interdomain routing game
- Intermediate approach: incorporate certain basic protocol design styles into computational model.
  - Example: soft-state protocols

Open Questions about Complexity

General:

- Which computational models are appropriate for the Internet?
- What mechanisms are computationally feasible with these models?
- Are there reductions, complete problems and, more generally, a complexity theory for Internet computations?

Specific:

- Are there many easy-to-compute VCG mechanisms in the multicast cost sharing problem?
- Does the *revelation principle* still apply?
  - Are there cases where direct mechanism has bad network complexity but an indirect one has good network complexity?
  - Related work on special case [Parkes]

Integrity

The Issue:

- When agents are both the strategic agents *and* the computational nodes, how can we preserve the integrity of the computation?

One approach: *observability* [Mitchell-Teague]

- Agents observe the protocol actions of neighboring agents
- Agents verify that neighboring agent's actions are consistent with her declared private information
- Extreme punishment for any inconsistency maintains the integrity of the computation

Open Questions about Integrity

General:

- Can we formalize the observability approach?
- Are there other approaches to the Integrity problem?

Specific:

- Does observability constrain the mechanism?

Privacy

The Issue:

- Can we design distributed mechanism-design algorithms such that agents' utilities remain private knowledge?

Observation [Nisan'99]:

- Yes, in theory: Use Secure, Multiparty Function Evaluation (SMFE) developed by crypto community

Problems with generic SMFE protocols:

- Assume large fraction of agents are obedient
- Assume *set* of agents known by all agents
- Require n^2 private channels (information-theoretic model)
- Have unacceptable network complexity

Open Questions about Privacy

General:

- Are there general approaches to agent privacy in DAMD other than the SMFE approach?

Specific: For specific mechanism-design problems...

- Are there SMFE protocols with low network complexity?
- Are there information-theoretic SMFE protocols that:
  - Don't require the agents to know about, or communicate explicitly with, each other?
  - Don't require n^2 private channels and use the natural network topology for the mechanism?
- Does settling for *partial* privacy of agents' utilities make the problem easier?

Related work:

- [Naor-Pinkas-Sumner], [Monderer-Tennenholtz], [Canetti-Kushilevitz-Ostrovsky-Rosen], [Cramer-Damgaard], [Beaver]

Q6: What Can Be Approximately Implemented?

Approximation may help remove barriers arising from:

- Incentive compatibility

  and/or

- Feasibility

Key point:

- Approximating a hard-to-compute mechanism with an easier one is not sufficient
- Must consider the strategic properties of the approximate mechanism

Possible Approaches to Approximation (partial list)

- Loosen strategyproof requirement
  - Approximately strategyproof [Schummer]
  - Feasible dominance [Nisan-Ronen]
  - Tolerable manipulability
- Asymptotic implementation: large number of agents
  - This may not work if agents are *idiosyncratic* and not all resources have many users
  - One approach: [Mehta-Vazirani] Assume agents are idiosyncratically *located* but have *iid* utilities
- Restrict utilities to tractable subset
  - *E.g.*, restricted languages for auctions
- Lotteries: virtual implementation
  - Impressive results for Nash
- Use metric space on outcomes to define approximation

Open Questions about Approximation

General:

- Which approximation approaches are effective?

Specific:

- What can be virtually implemented with strategyproof mechanisms? Other Internet-like solution concepts?
- Which SCCs can you (approximately) achieve knowing the distribution of utilities?
- Do any implementation impossibility results disappear when you allow metric-space approximations?
  - One negative result: Cannot always achieve approximate efficiency and approximate budget-balance with strategyproof mechanisms (Krishnamurthy's talk)

Summary: Distributed Algorithmic Mechanism Design

State of the field:

- Growing consensus that both incentive and computation constraints are important
- Several interesting DAMD problems and results
  - But no single compelling example
  - No coherent framework
- Many fundamental issues unresolved (and unaddressed)

What we need to make progress:

- Work on these unresolved issues (focus of talk)

  and

- Some canonical hard problems

Canonical Hard Problems

Computer Science:

- Has a collection of canonical hard problems
- Teach us what functions are inherently hard to compute

Game Theory:

- Has a collection of impossibility results
- Teach us what SCFs/SCCs are impossible to implement

Canonical Hard Problems for DAMD

Want distributed allocation problems and an SCC where:

- The computation, ignoring incentives, has low network complexity
- Implementation, ignoring the computational limitations, is possible
- The centralized implementation has low complexity
- But distributed implementations have inherently high network complexity

Such problems teach us about the interaction of incentives and distributed computation

- BB (+ other minor conditions) multicast cost sharing
- We need more!