Resource Allocation in Competitive Multiagent Systems

Kevin Leyton-Brown

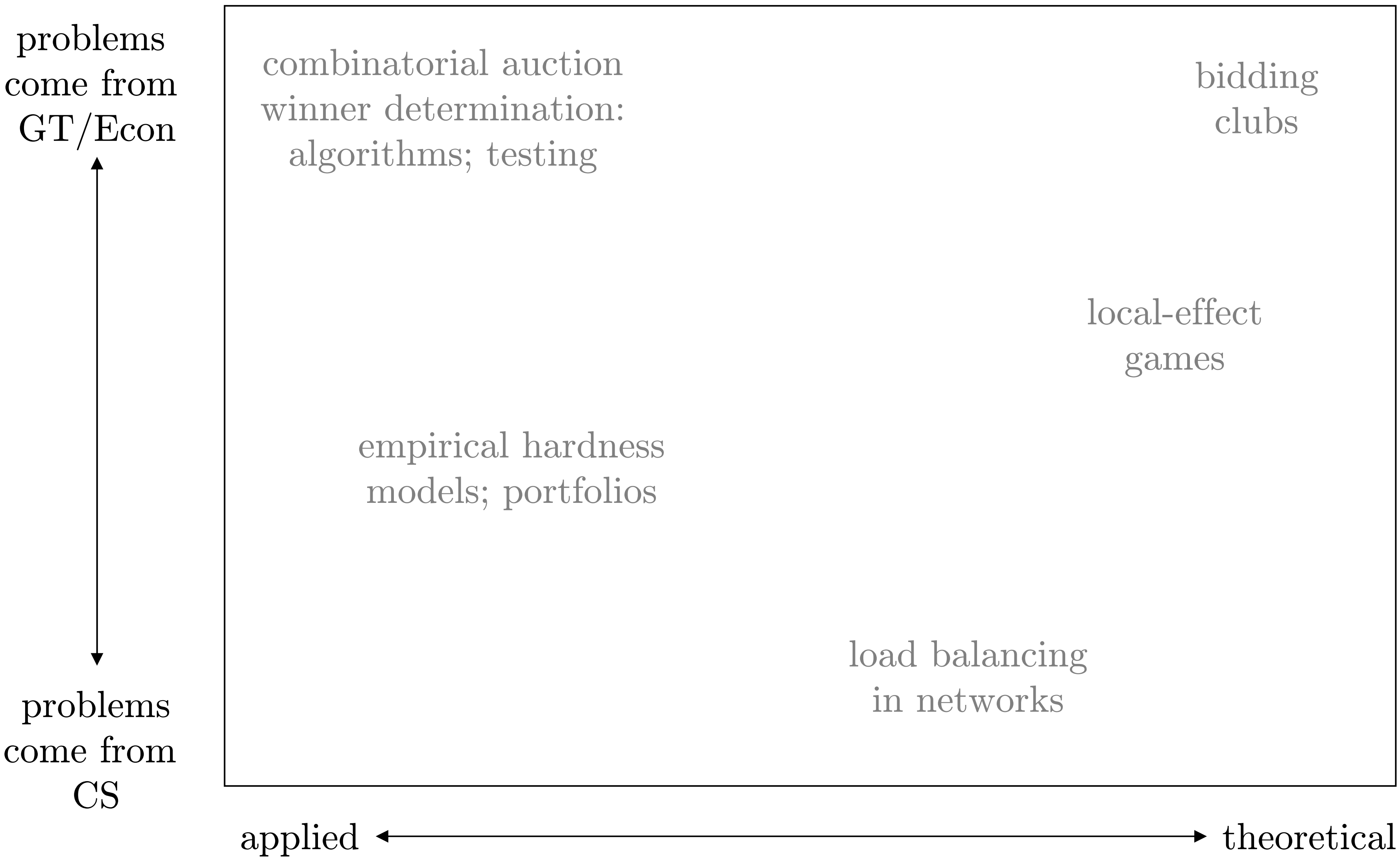
Overview

- Multiagent systems
  - autonomy; asymmetric information
  - cooperative: same interests
  - competitive: selfish

- Resource allocation in multiagent systems
  - cooperative: behavioral protocol can be imposed
  - competitive: agents can't be trusted to follow a protocol

- Explore interactions between Economics/Game Theory and Computer Science
  1. GT problems with CS solutions
  2. CS problems with GT solutions
  3. Bidirectional interactions; synthesis

Topics


problems come from GT/Econ
problems come from CS
combinatorial auction winner determination: algorithms; testing
bidding clubs
local-effect games
empirical hardness models; portfolios
load balancing in networks
applied
theoretical

Why auctions?

- Theoretical framework for resource allocation among self-interested agents
  - e.g., social welfare maximization; revenue maximization
- They're big ($$$)
  - and the internet is changing the way they're used

home | my eBay | site map

Browse | Sell | Services | Search | Help | Community

item view

500 Pounds of Marijuana For Sale! NO RESERVE!

Item #168659968

Toys, Bean Bag Plush:Hobbies:General

Description

Bid!

Currently	$10,000,000.00	First bid	$50.00
Quantity	1	# of bids	7 (bid history)
Time left	9 days, 3 hours +	Location	Ownedville, Florida
Started	09/21/99, 20:42:19 PDT		(mail this auction to a friend)
Ends	10/01/99, 20:42:19 PDT		(request a gift alert)
Seller	killpeople (0) (view comments in seller's Feedback Profile) (view seller's other auctions) (ask seller a question)		
High bid	rdmustang (11)		
Payment	Visa/MasterCard, American Express, Discover, Money Order/Cashiers Checks, COD (collect on delivery), Personal Checks, Other, See item description for payment methods accepted		
Shipping	Buyer pays actual shipping charges, Seller ships internationally, See item description for shipping charges		

Seller assumes all responsibility for listing this item. You should contact the seller to resolve any questions before bidding. Currency is U.S. dollar ($) unless otherwise noted.

Description

500 Pounds of the best marijuana Holland has to offer! ***This is a good deal for this much marijuana. street value is off the charts!*** **Please dont email me about free samples, sorry :-P {[-Yomasta-]}**

home | my eBay | site map | sign in/out

Browse | Sell | Services | Search | Help | Community

item view

HALF-EATEN ROAST BEEF SANDWICH VERY TASTY

Item #545658477

Collectibles:Cultural:Western Americana:General

Description

Bid!

Currently	**$10.50**	First bid	**$0.01**
Quantity	**1**	# of bids	**3** bid history
Time left	**6 days, 4 hours +**	Location	**slappy's funtown**
		Country/Region	**USA/Philadelphia**
Started	Jan-19-01 19:04:54 PST		mail this auction to a friend
Ends	Jan-26-01 19:04:54 PST		watch this item

Seller (Rating)	**moongoober** (14)
	view comments in seller's Feedback Profile | view seller's other auctions | ask seller a question
High bid	**baron_von_nutterbutter** (0)
Payment	See item description for payment methods accepted
Shipping	Buyer pays actual shipping charges, Will ship to United States and the following regions: Canada, Europe, Australasia, Asia, South America, Africa, Mexico and Central America, Middle East, Caribbean
Item Revised Before First Bid	To review revisions made to this item by the seller, click here.

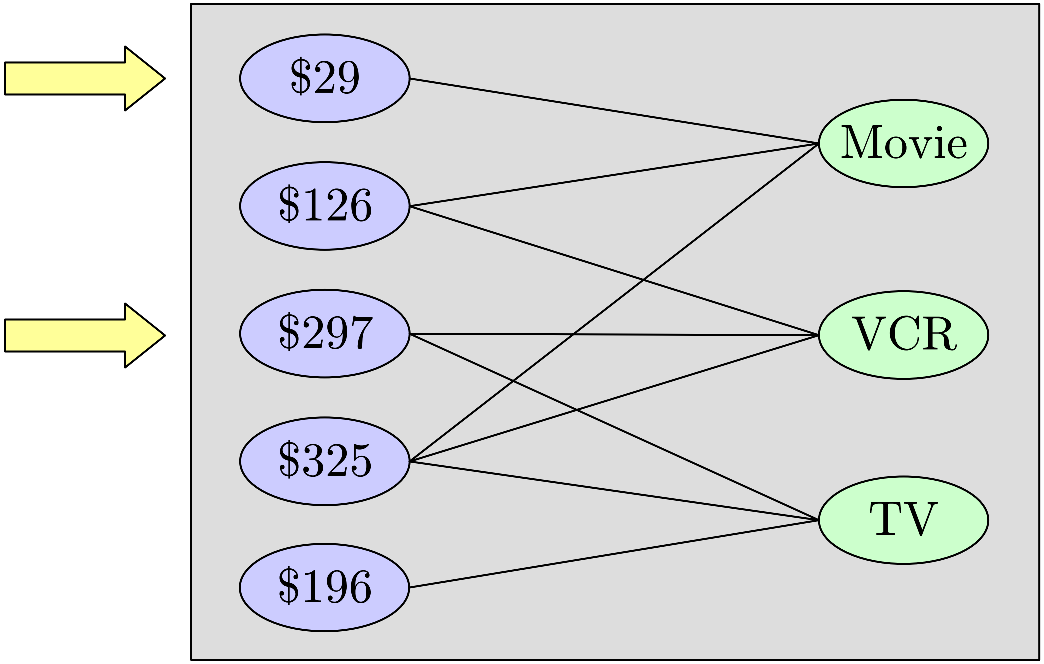
Seller assumes all responsibility for listing this item. You should contact the seller to resolve any questions before bidding. Auction currency is U.S. dollars ($) unless otherwise noted.

Description

This is a half-eaten roast beef sandwich I made earlier today. It is very tasty.

What you need to know about auctions

- They're a broader category than often perceived
- Of special interest: Combinatorial auctions
  - hot topic in CS for past four years
  - auctions where bidders can request bundles of goods
  - interesting because of complementarity and substitutability



Winner Determination Problem

- Input: n goods, m bids

  $< S_i, p_i >, S_i \subseteq \{1, \ldots, n\}$

- Objective: find revenue-maximizing non-conflicting allocation

$$
\begin{array}{lll}
\text{maximize:} & \displaystyle\sum_{i=1}^{m} x_i p_i & \\
\text{subject to:} & \displaystyle\sum_{i|g \in S_i} x_i \leq 1 & \forall g \\
& x_i \in \{0, 1\} & \forall i
\end{array}
$$

What's known about WDP

Equivalent to weighted set packing, $\mathcal{NP}$-Complete

1. Approximation
   - best guarantee is within factor of $\sqrt{n}$
   - economic mechanisms can depend on optimal solution
2. Polynomial special cases
   - very few (ring; tree; totally unimodular matrices)
   - allowing unrestricted bidding is the whole point
3. Complete heuristic search
   - CASS [Fujishima, Leyton-Brown, Shoham, 1999]
   - CABOB [Sandholm, 1999; Sandholm, Suri, Gilpen, Levine, 2001]
   - GL [Gonen & Lehmann, 2001]
   - CPLEX [ILOG Inc., 1987-2003]

Where do we stand?

- Best solutions (e.g., CPLEX):
  - often blindingly fast
  - but sometimes very slow

- Problem I: Are we testing on the right data?
  - Legacy [Sandholm, 1999]; [Fujishima, Leyton-Brown, Shoham, 1999]
  - CATS [Leyton-Brown, Pearson, Shoham, 2000]

- Problem II: How can we understand why performance varies so drastically?
  - use machine learning to predict running time

    [Leyton-Brown, Nudelman, Shoham, 2002]

Empirical Hardness Models

- Our goal: emulate success in understanding the hardness of (e.g.) satisfiability instances, but:
  - we have an optimization problem
  - and a very high dimensional one
- If we are nonetheless successful, we will be able to:
  - go get coffee while the algorithm is running
  - build algorithm portfolios
  - tune distributions for hardness
  - in general, gain insight into the sources of hardness
- Case study of these models on WDP
  - recent work: applied these ideas to SAT

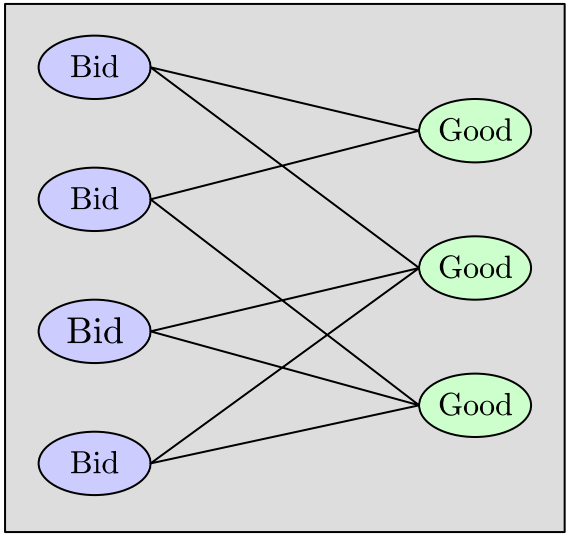
Empirical Hardness Methodology

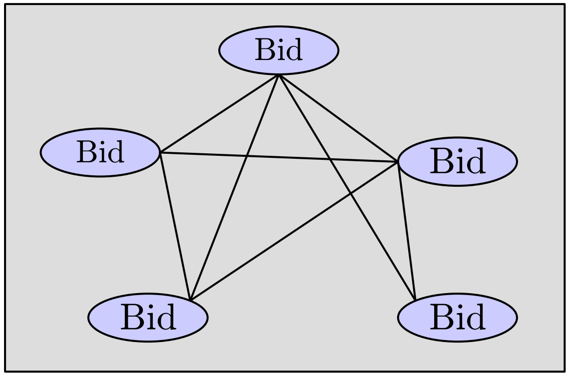
1. Select optimization algorithm
2. Select set of distributions
3. Define problem size
4. Select features
5. Generate instances
6. Compute running time, features
7. Learn running time model

Features

1. Linear Programming
   - L_1, L_2, L_∞ norms of integer slack vector
2. Price
   - stdev(prices)
   - stdev(avg price / num goods)
   - stdev(average price / sqrt(num goods))
3. Bid-Good graph
   - node degree stats (max, min, avg, stdev)
4. Bid graph
   - node degree stats
   - edge density
   - clustering coefficient (CC), stdev
   - avg min path length (AMPL)
   - ratio of CC to AMPL
   - eccentricity stats (max, min, avg, stdev)

$$\begin{aligned} \text{maximize:} \quad & \sum_{i=1}^{m} x_i p_i \\ \text{subject to:} \quad & \sum_{i \mid g \in S_i} x_i \leq 1 \quad \forall g \\ & 0 \leq x_i \leq 1 \quad \forall i \end{aligned}$$

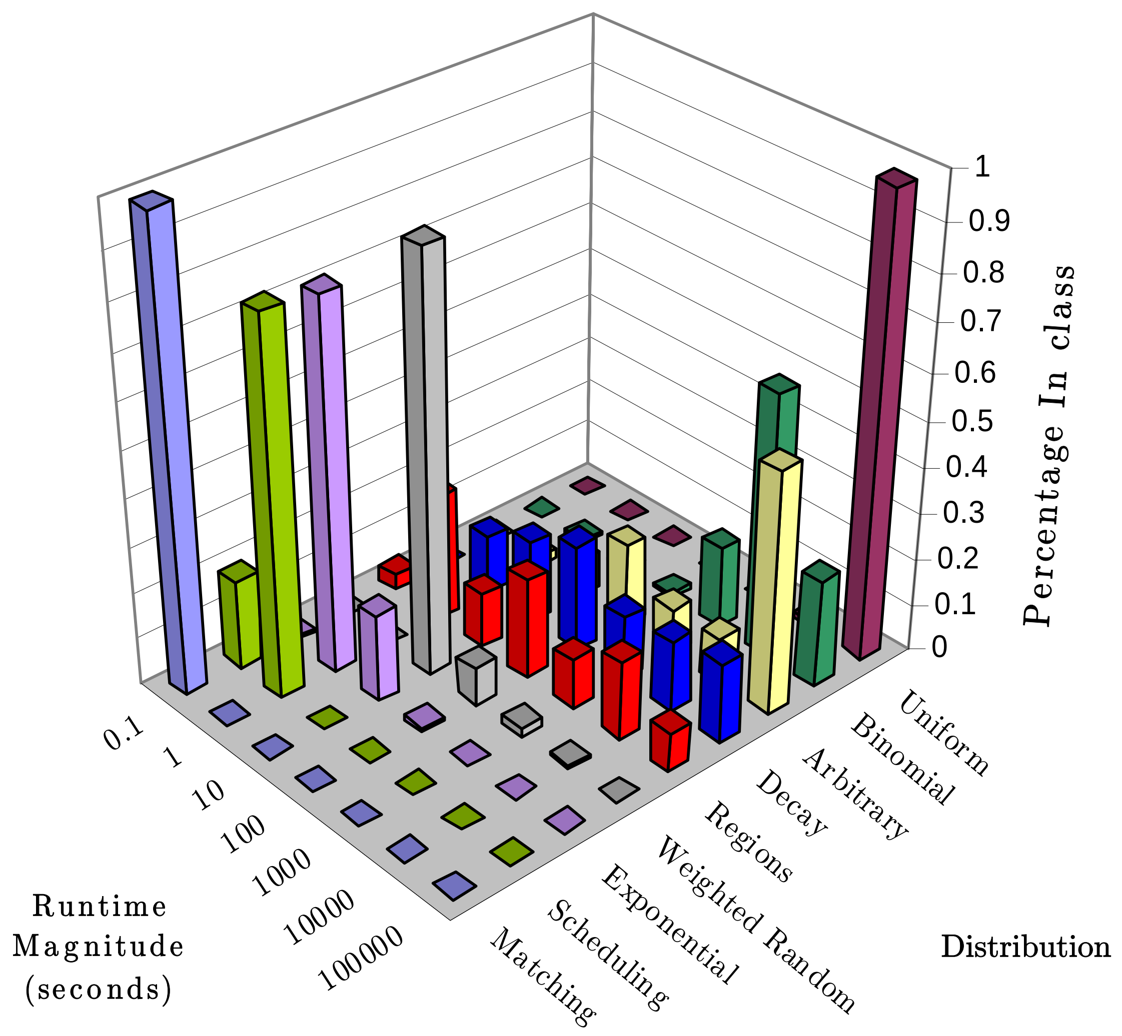




Experimental Setup

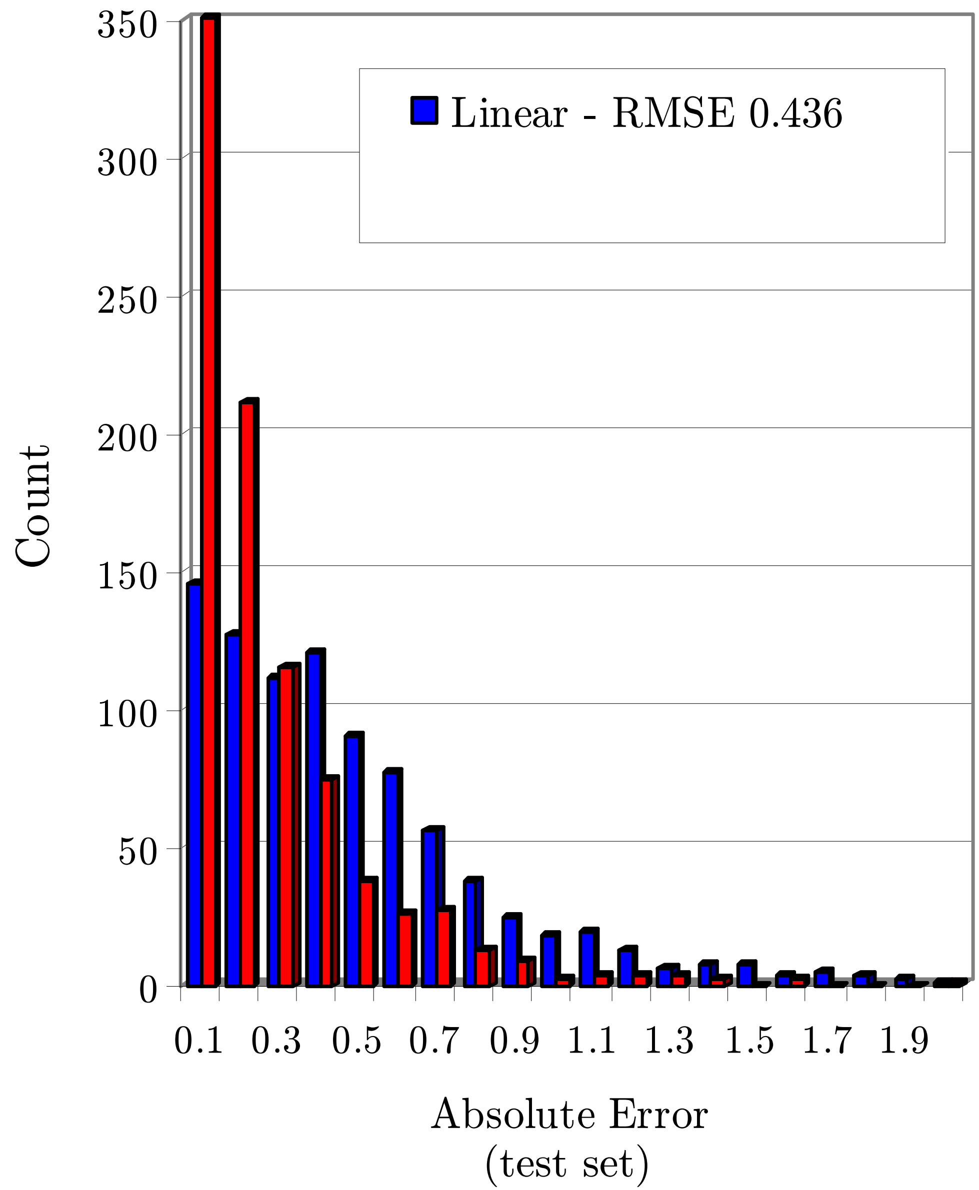
- Problem size: goods, undominated bids
- Nine distrib
  Bid: $100
  - sample par ranges
  - generate 500 instances/distribution: 4500 per dataset
- Three datasets:
  - 256 goods, 1000 non-dominated bids
  - 144 goods, 1000 non-dominated bids
  - 64 goods, 2000 non-dominated bids
- Experiments:
  - 32-machine cluster of 550 MHz Xeons, Linux 2.12
  - collecting data took approximately 3 years of CPU time!
  - running times varied from 0.01 sec to 22 hours (CPLEX capped)

Gross Hardness (144 goods, 1000 bids)

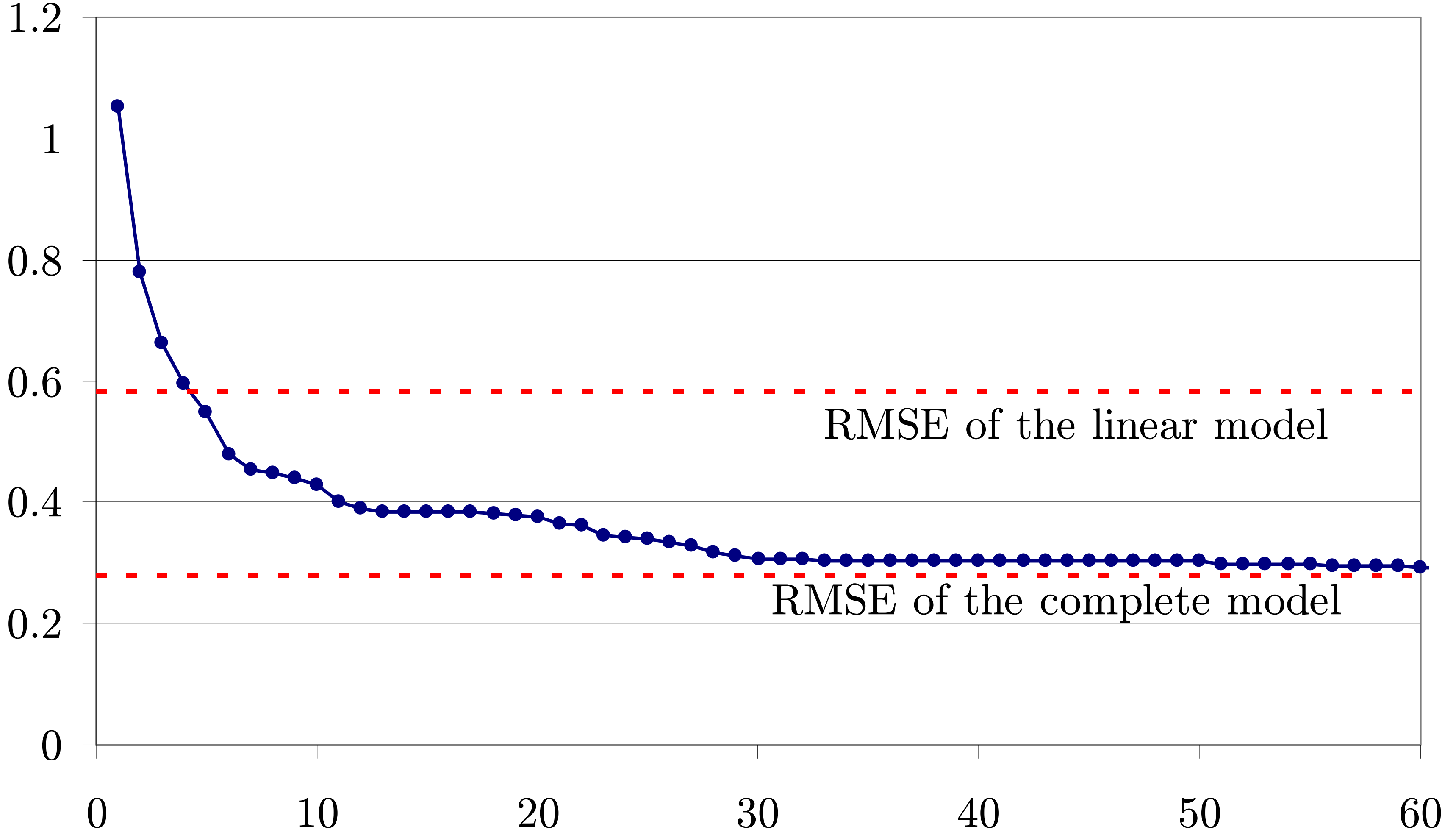


Learning

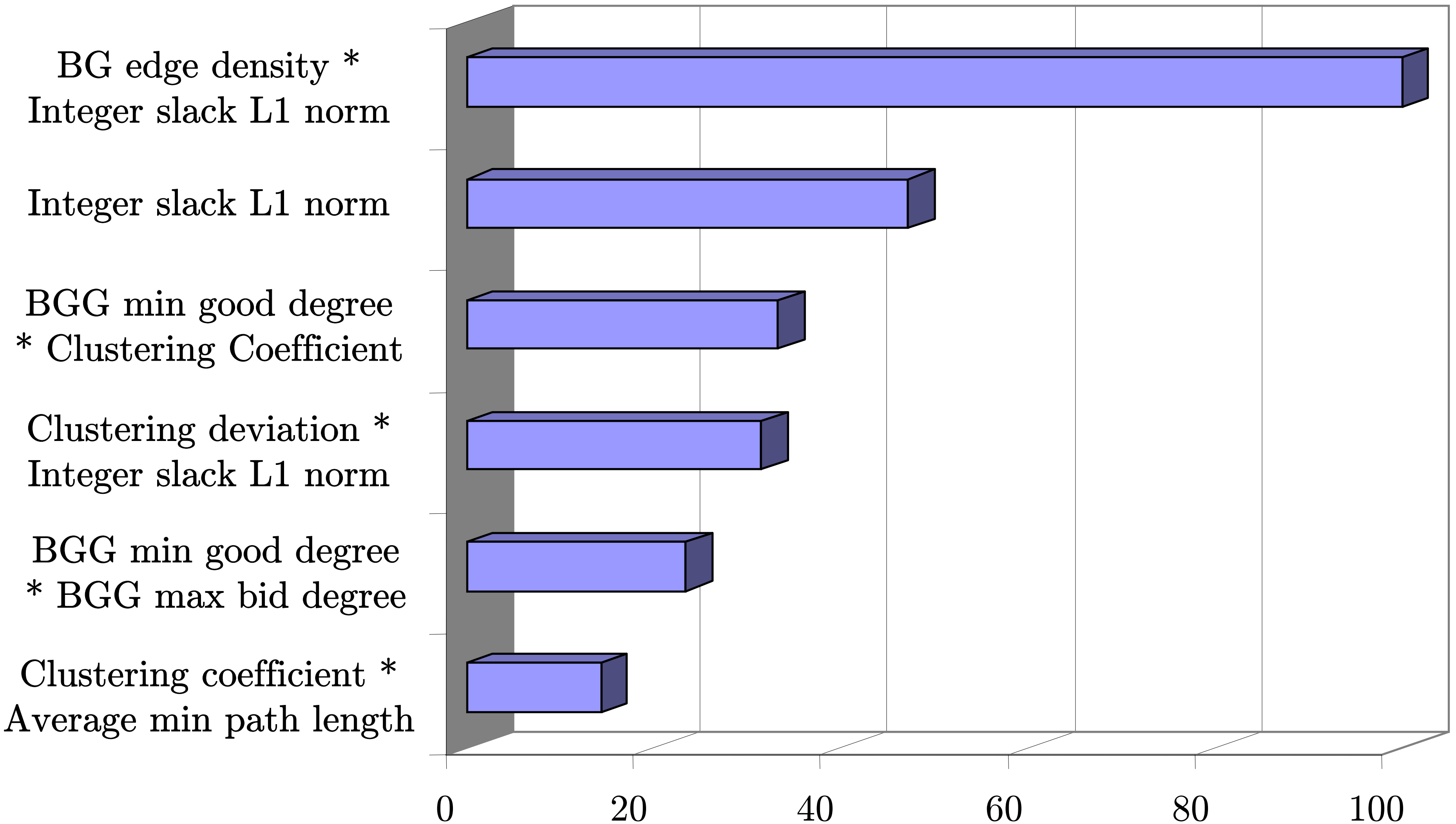
- Linear regression
  - ignores interactions between variables
- Consider 2nd degree polynomials
  - variables: pairwise products of original features
  - total of 325
- We tried various other non-linear approaches; none worked better.



Understanding Models: RMSE vs. Subset Size



Cost of Omission (subset size 6)



Boosting as a Metaphor for Algorithm Design

[Leyton-Brown, Nudelman, Andrew, McFadden, Shoham, 2003]

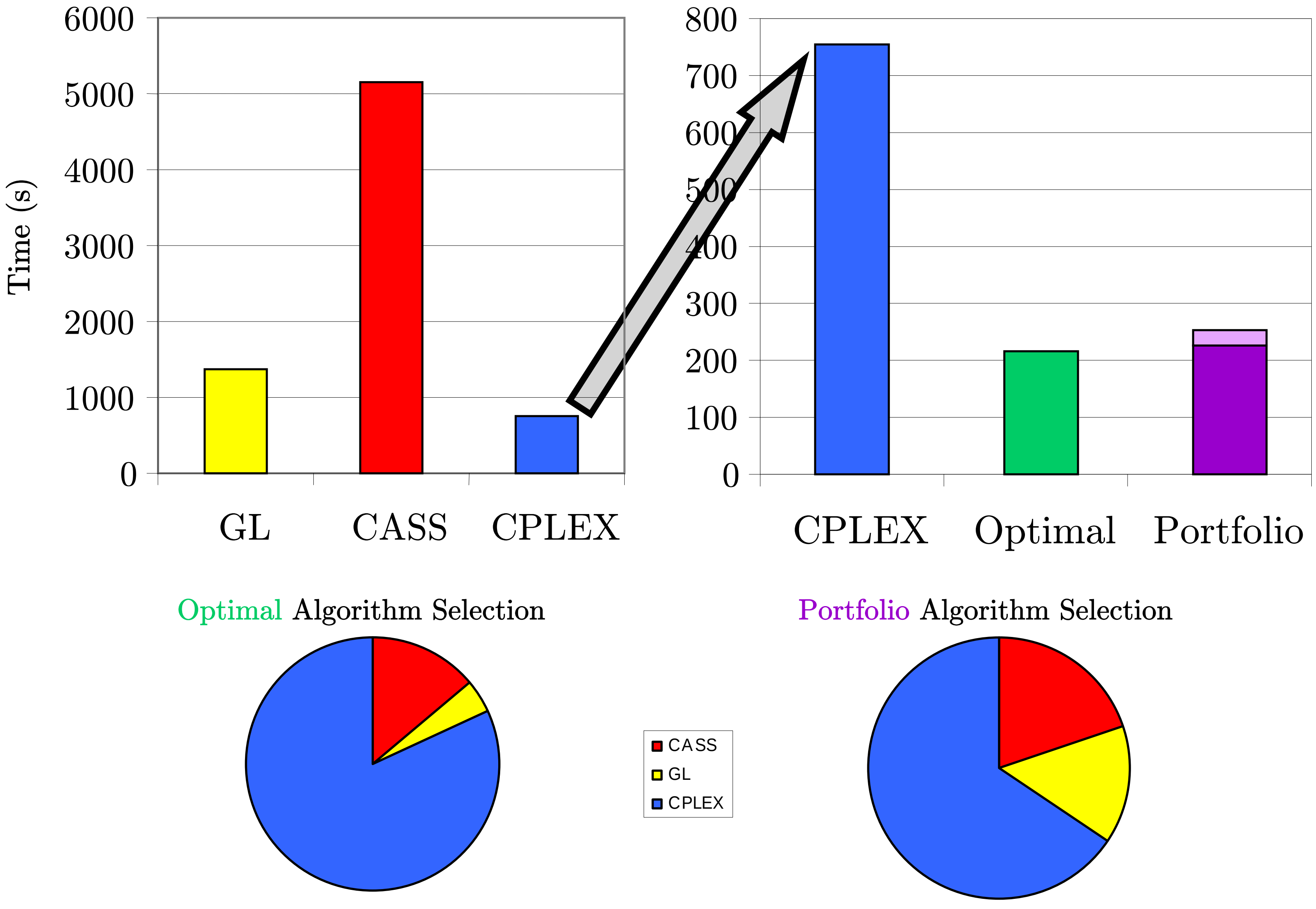
Boosting (machine learning technique):

1. Combine uncorrelated weak classifiers into aggregate
2. Train new classifiers on instances that are hard for the aggregate

Algorithm Design with Hardness Models:

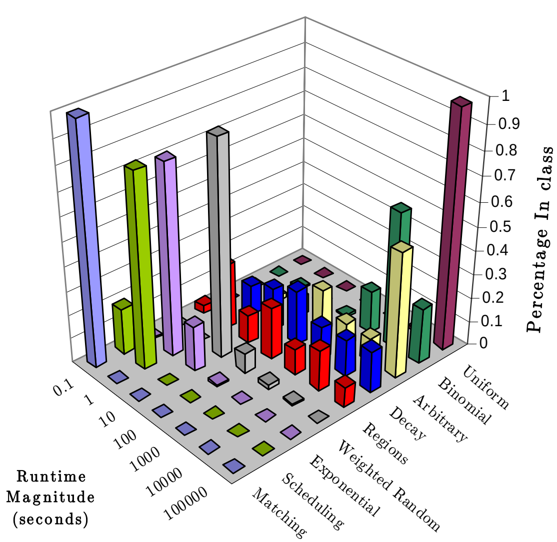
1. Hardness models can be used to select an algorithm to run on a per-instance basis
2. Use portfolio hardness model as a PDF, to induce a new test distribution for design of new algorithms

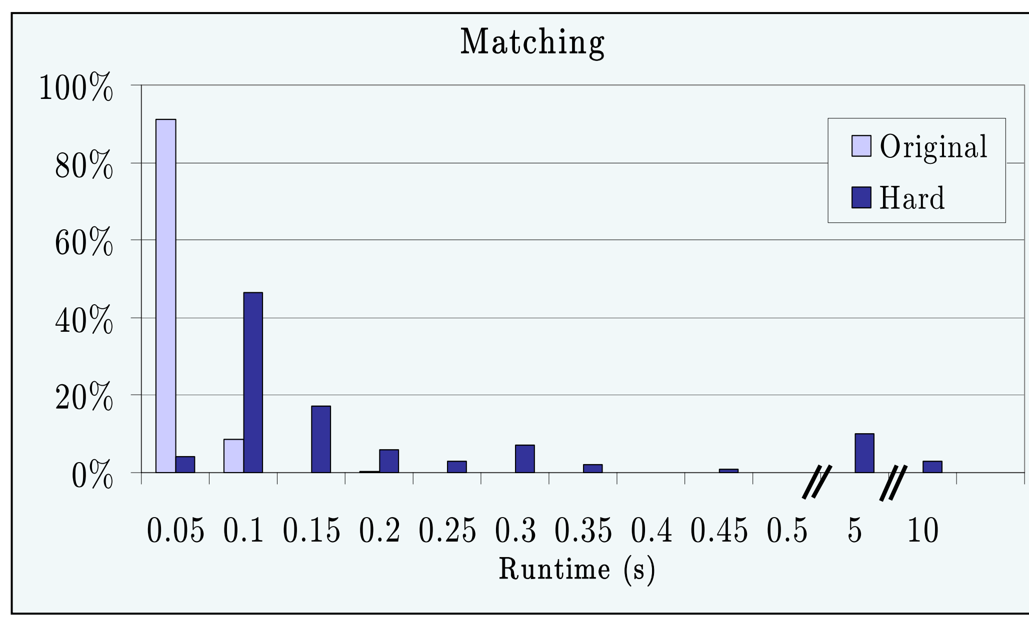
Portfolio Results



Distribution Induction

- We want our test distribution to generate problems in proportion to the time our portfolio spends on them
  - D: original distribution of instances
  - H_f: model of portfolio runtime (h_f: normalized)
- Goal: generate instances from $D \cdot h_f$
  - D is a distribution over the parameters of an instance generator
  - h_f depends on features of generated instance
- Use rejection sampling





Topics

problems come from GT/Econ

combinatorial auction winner determination: algorithms; testing

bidding clubs

local-effect games

empirical hardness models; portfolios

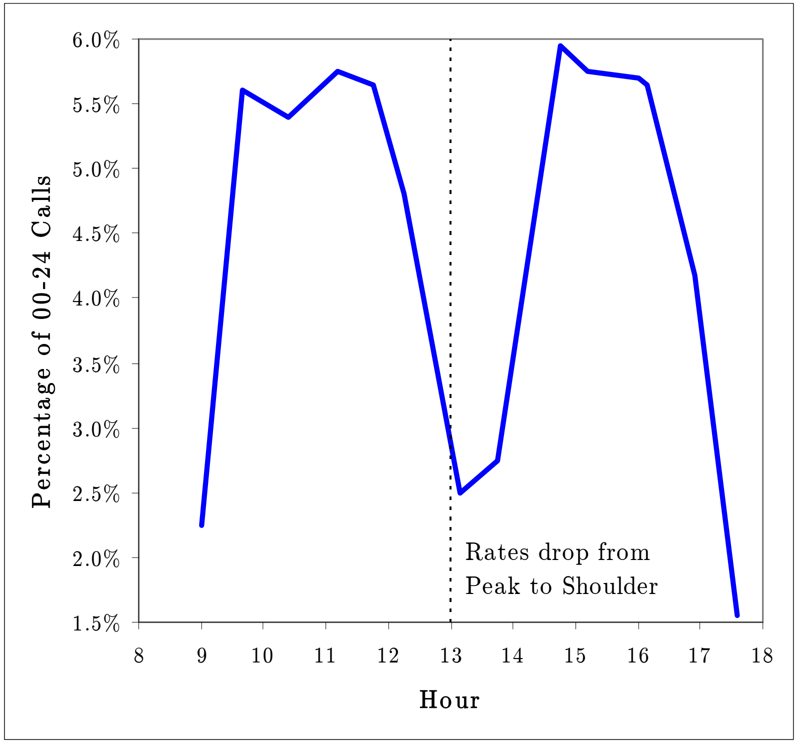
load balancing in networks

problems come from CS

applied ↔ theoretical

Focused Loading

- Many users demand network resources at a focal time
- Example: long distance phone
  - want to talk as early as possible, minimize cost
  - max utility when rates drop: network demand spikes
- Computer networks: load can be even more focused
  - sudden onset: TicketMaster server as tickets go on sale
  - deadline: IRS server just before taxes are due
- Idea: provide incentives for users to defocus their own loads



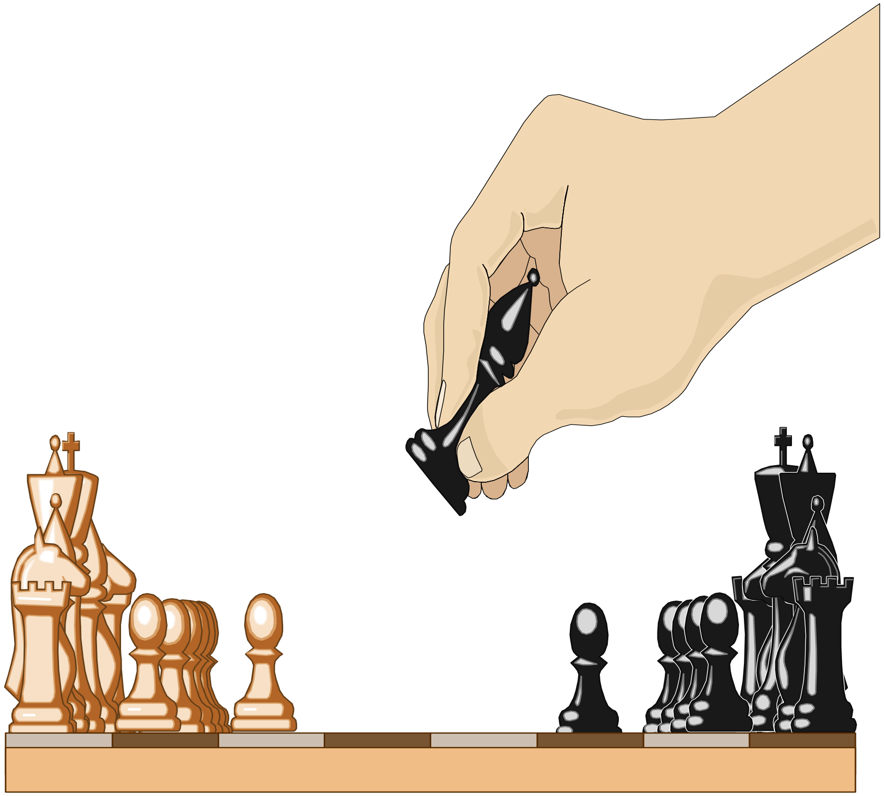
Quarterly Trunk Calls on Weekdays in the United Kingdom, December 1975

[Mitchell, 1978]

[Leyton-Brown, Porter, Prabhakar, Venkataraman, Shoham, 2001; 2003]

Things you need to know from Game Theory

- Game:
  - players/agents
  - actions
  - strategies
  - payoffs
- Equilibrium
  - stable strategies
  - weak/strict
  - mixed/pure
- Mechanism design



Our Model

- Network resource: use is divided into t timeslots
- Each slot has a fixed usage cost m
- n agents will use the network resource for one slot each
  - slot $\bar{\boldsymbol{s}}$ is preferred by all agents
- Agent a_i's valuation for slot s is $v_i(s)$. Two cases:
  1. all agents have the same v
  2. mechanism designer knows bounds: v^l and v^u
- $d(s)$ is the number of agents who choose slot s
- Give agents incentive to balance load, but make small computational demands on the network resource
  - waive the usage fee for slot s with probability $p(s)$
  - q: expected number of free slots

Mechanism Evaluation, Optimality

The mechanism designer has two goals:

1. maximize expected revenue
2. balance load caused by the agents' selection of slots

Expressed in tradeoff function z

Optimality: A mechanism-equilibrium pair is optimal if it maximizes z, as compared to all other equilibria in other mechanisms (constant n, participation rational)

ε-optimality: z - z_{opt} is bounded by $n\varepsilon$

Theorem 1: The optimal mechanism-equilibrium pair has a weak equilibrium (complete indifference). [same v]

Theorem 2: No strict, optimal equilibrium exists

"Collective Reward"

1. The mechanism signals agents with slot numbers
   - $c(s)$: the number of agents given signal s
2. Each agent chooses a slot
3. The mechanism computes p, and determines which slots will be made (retroactively) free

$$p^*(s) = \begin{cases} \frac{1}{m}\left(v^u(\bar{s}) - v^l(s)\right) & \text{if } s \neq \bar{s} \\ 0 & \text{if } s = \bar{s} \end{cases}$$

$$p(s) = \begin{cases} \max(p^b(s) + \varepsilon, 1) & \text{if } d(s) \leq c(s) \\ 0 & \text{if } d(s) > c(s) \end{cases}$$

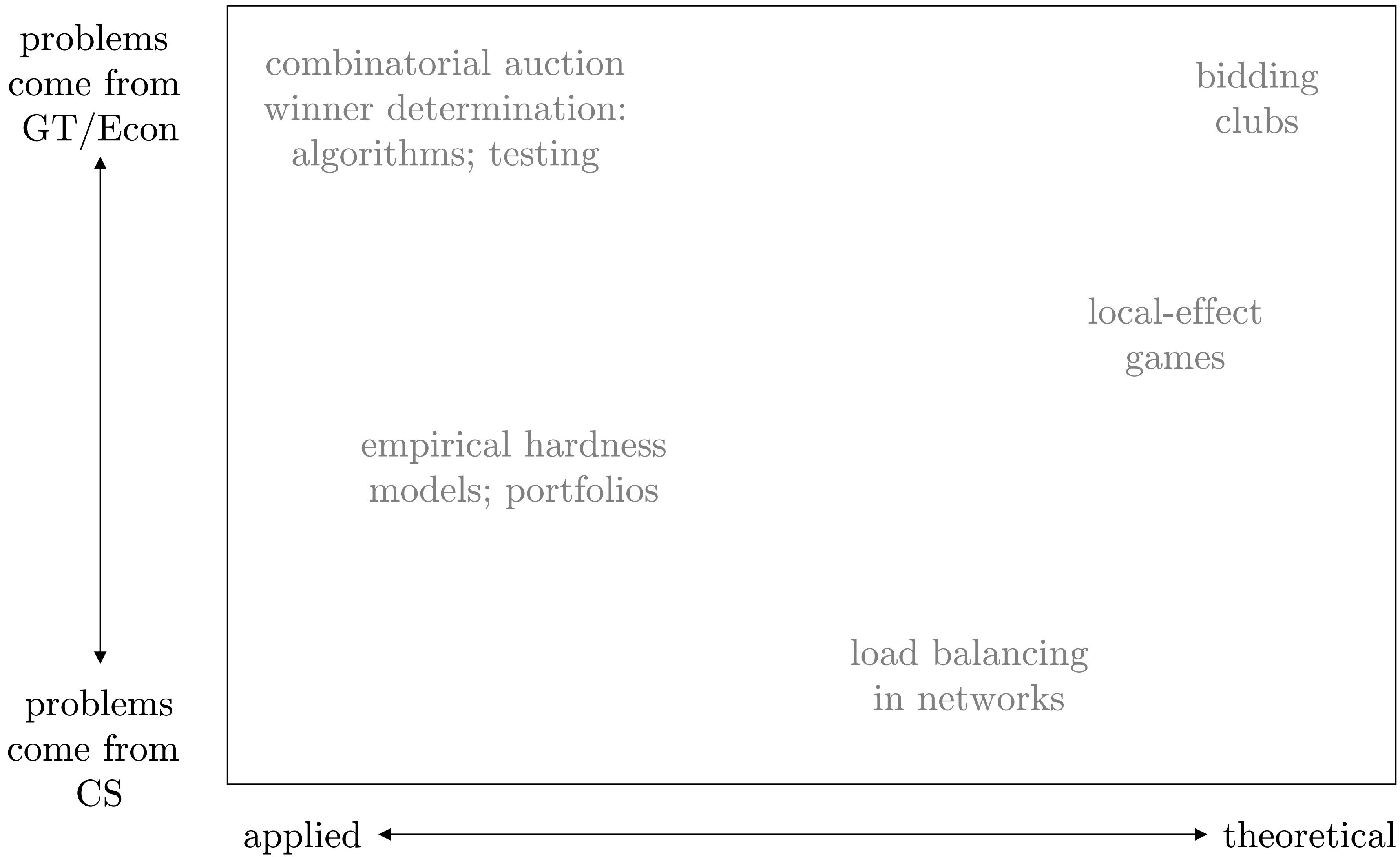
Lemma 1: Assigning each agent the signal that greedily improves z gives rise to optimal d

Lemma 2: Strict equilibrium φ: a_i chooses slot $c(i)$

Theorem 3: (CR, φ) is ε-optimal [same v]

Theorem 4: (CR, φ) is k-optimal, $k = \max_s(v^u(s) - v^l(s)) + \varepsilon$ [different v]

Topics


problems come from GT/Econ
problems come from CS
applied
theoretical
combinatorial auction winner determination: algorithms; testing
bidding clubs
local-effect games
empirical hardness models; portfolios
load balancing in networks

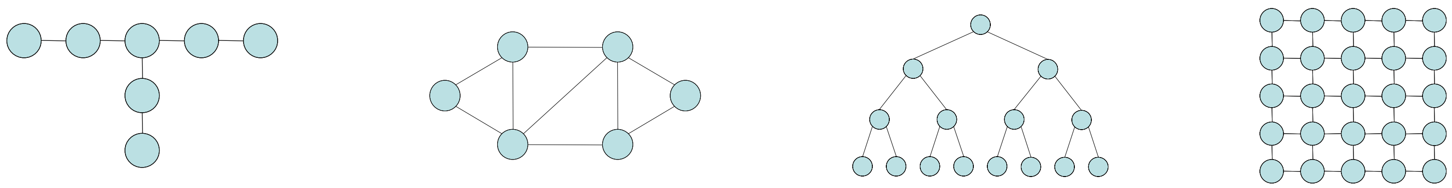
Computation-Friendly Game Representations

- In practice, interesting games are large; computing equilibrium is hard
- CS agenda: compact representation, tractable computation
  - independencies/modularity [La Mura, 2000], [Kearns, Littman, Singh, 2001], [Vickrey & Koller, 2002]
  - symmetries [Roughgarden & Tardos, 2001], [Kearns & Mansour, 2002]
- Congestion games (slightly simplified) [Rosenthal, 1973]
  - each agent i selects an action a
  - $D(a)$ is the number of agents who choose action a
  - $F_a(\cdot)$ are arbitrary functions for each a
  - agent i pays $p_i(a_i, D) = F_{a_i}(D(a_i))$
- Example: traffic congestion

Local Effect Games

[Leyton-Brown & Tennenholtz, 2003]

- An agent can be made to pay more because another agent chooses a different but related action
  - location problem: ice cream vendors on the beach
- $neigh(a)$ is the set of actions that locally affect agents who choose action a
- $F_{a,a'}(\cdot)$ is the cost due to the local effect from action a to action a'
- Agent i pays $p_i(a_i, D) = F_{a_i,a_i}(D(a_i)) + \sum_{a' \in neigh(a_i)} F_{a',a_i}(D(a')).$



Local Effect Graphs

Local Effect Games

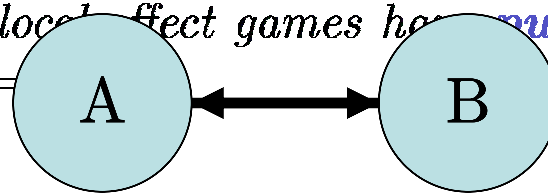
1. Compact representation
   - context-specific independence between actions
   - symmetry among players' utility functions

2. What about finding equilibria?
   - theoretical: exploit special properties
     - pure-strategy Nash equilibrium
   - computational
     - myopic best-response dynamics

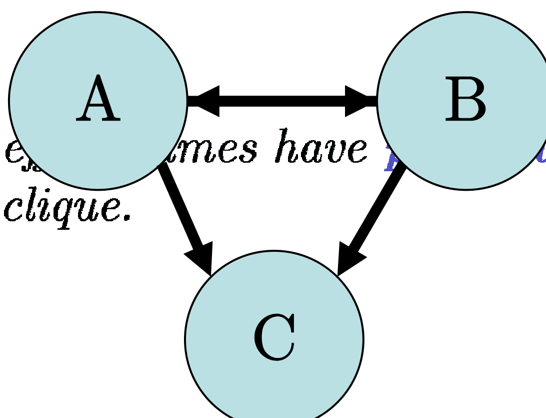
Main Technical Results

Definition 1 *A local-effect game is a **bidirectional local-effect game** when local effects are bidirectional:* $\forall a \in \mathcal{A}, \forall a' \neq a \in \mathcal{A}\, \mathcal{F}_{a,a'}(x) = \mathcal{F}_{a',a}(x)$.

Theorem 1 *Bidirectional loc… ffect games ha… **…ure strategy Nash equilibria** if* $\forall i, \forall j \neq i\, \mathcal{F}_{i,j}(x) =$ …



Definition 2 *A local-effect game is a **uniform local-effect game** when local effects are uniform:* $\forall A, B, C \in \mathcal{A}\, (B \in neigh(A) \wedge C \in neigh(A)) \rightarrow \forall x\, \mathcal{F}_{A,B}(x) = \mathcal{F}_{A,C}(x)$.



Theorem 2 *Uniform local-e… …mes have **…trategy Nash equilibria** if the local-effect graph is a clique.*

Main Technical Results

Theorem 3 *The class of congestion games contains the class of local-effect games for which any of the following hold:*

1. *the game is a* ***bidirectional local-effect game*** *and all local-effect functions are linear*
2. *the game is a* ***uniform local-effect game*** *and the local-effect graph is a clique*
3. *the local-effect graph contains* ***no edges***
4. *the local-effect graph contains* ***fewer than three nodes***

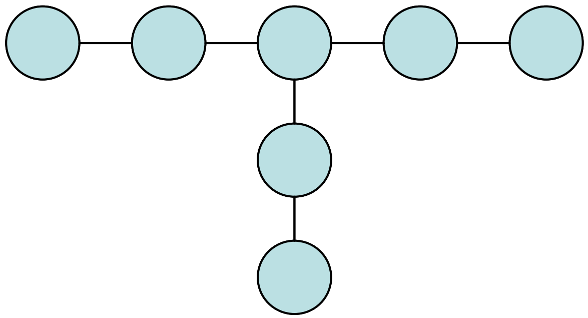
No other local-effect games are congestion games.

Theorem 4 *If a local-effect game satisfies*

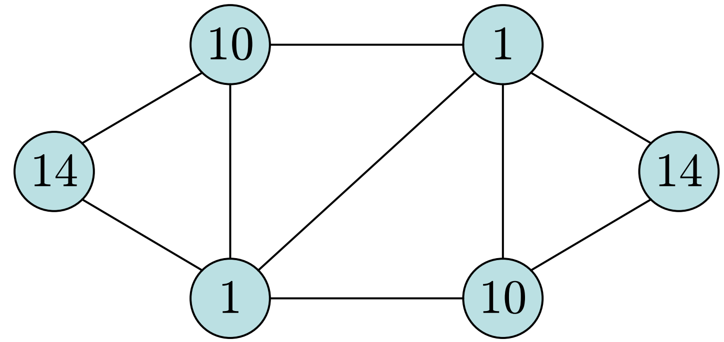
1. $\forall A \in \mathcal{A}, \forall B \in neigh(A), \forall x,\ \mathcal{F}_{A,A}(x) \leq \mathcal{F}_{A,B}(x)$
2. $\forall A, B \in \mathcal{A}, \forall x \geq 1,\ \mathcal{F}_{A,B}(x+1) - \mathcal{F}_{A,B}(x) \leq \mathcal{F}_{A,B}(x) - \mathcal{F}_{A,B}(x-1),$

then there exists a ***pure-strategy Nash equilibrium*** *in which agents choose nodes that constitute an independent set.*

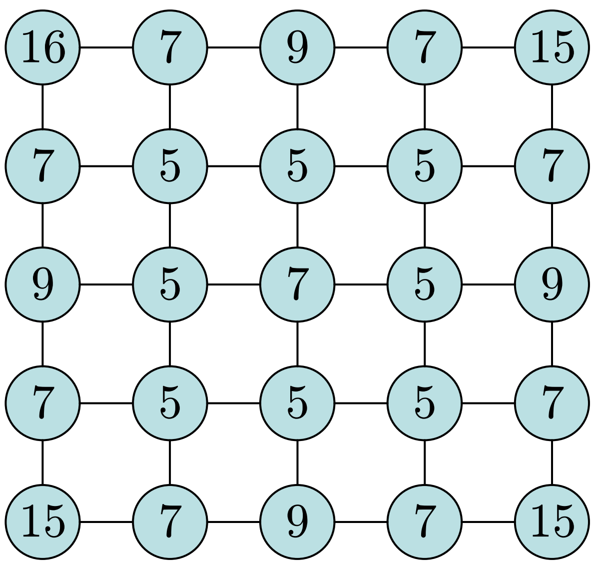
Computational Results



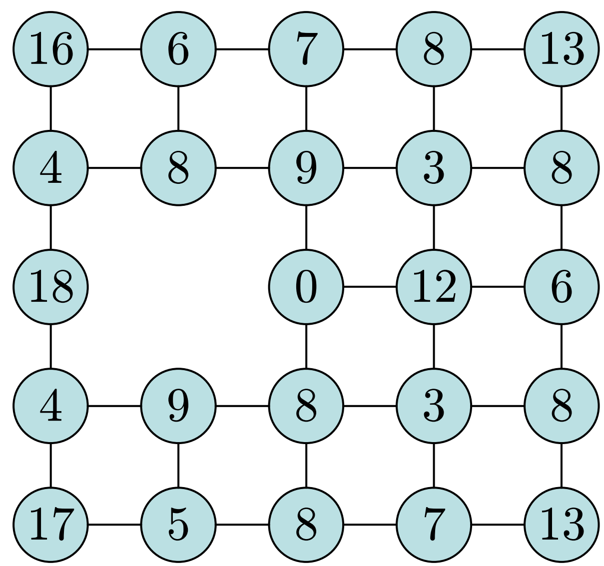
node: $3{\cdot}\log(x{+}1)$ edge: $\log(x{+}1)$
50 agents



log/log. Node 3, edge 1. 50 agents

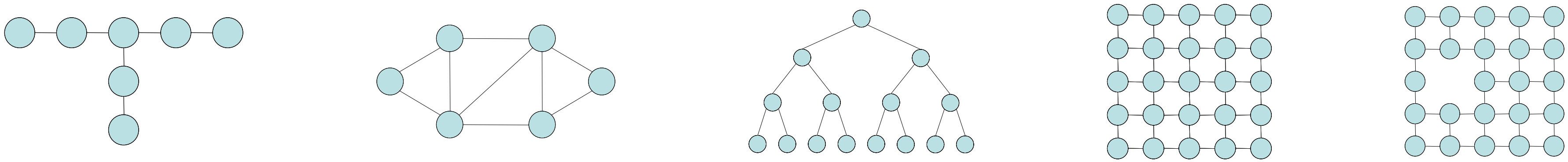


log/log; node 4, edge 1; 200 agents

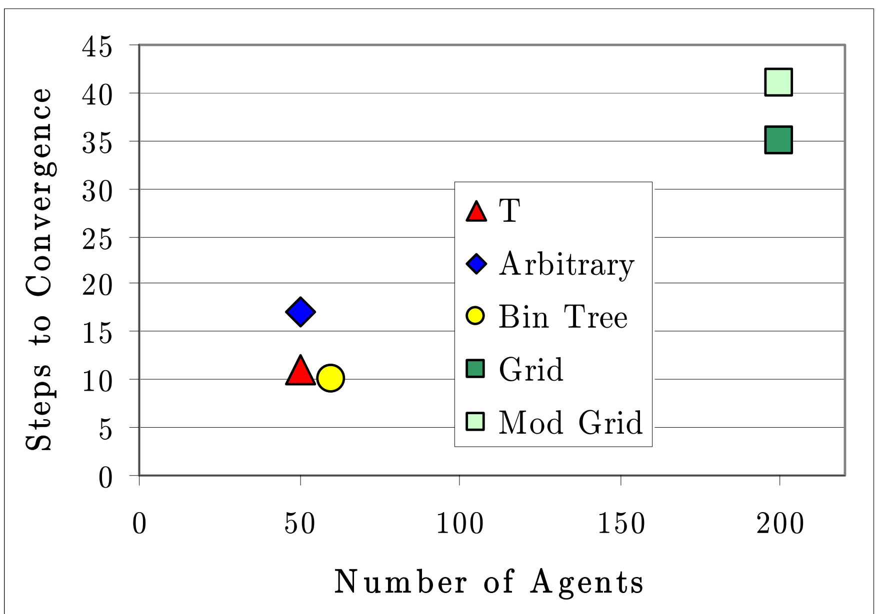


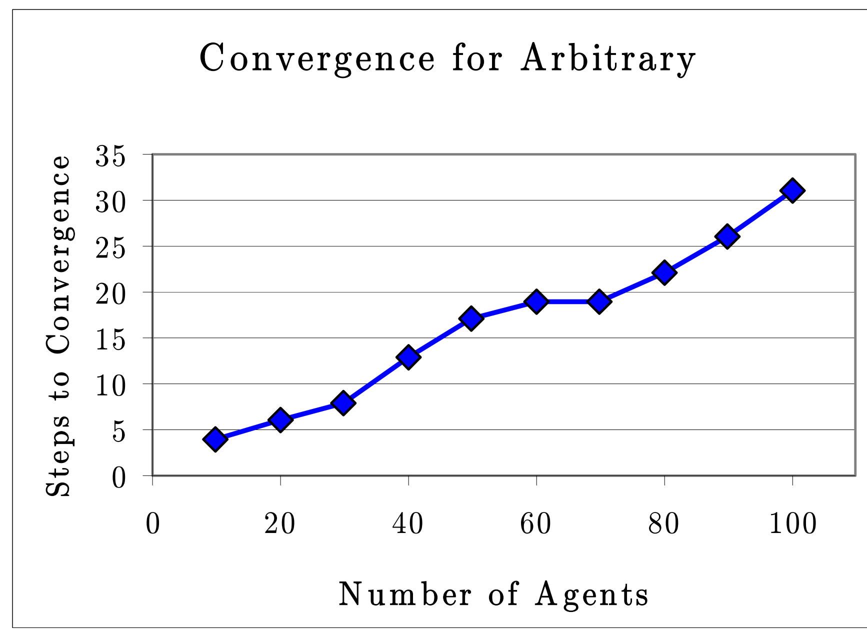
log/log; node 4, edge 1; 200 agents

Computational Results

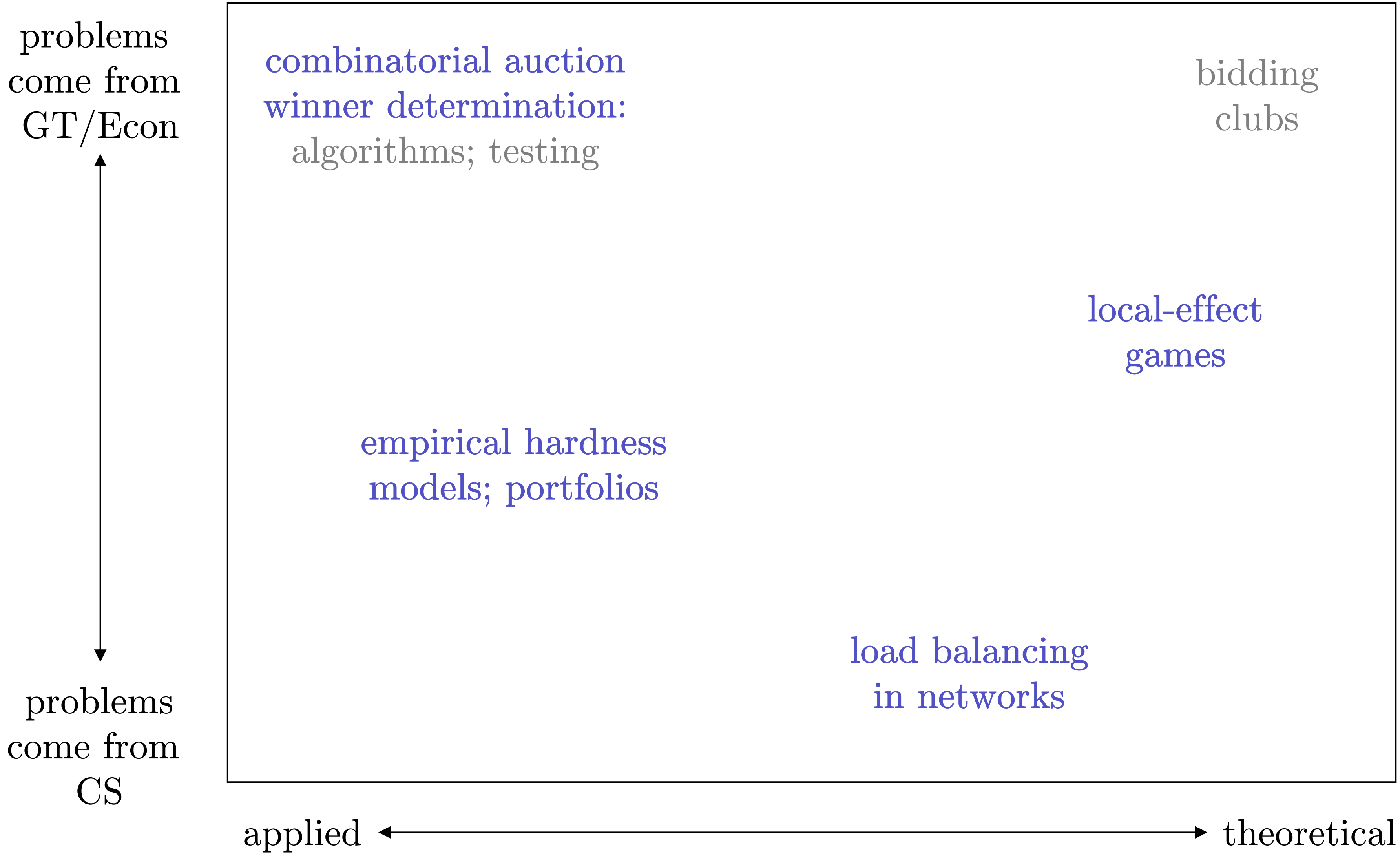


T Arbitrary Bin Tree Grid Mod Grid





Resource Allocation in Competitive Multiagent Systems



Thanks!

- to my advisor, Yoav Shoham
- to members of my committee:
  - Andrew Ng, Moshe Tennenholtz (reading)
  - Daphne Koller, David Kreps (orals)
- to coauthors of the work presented here:
  1. Eugene Nudelman; Galen Andrew; Jim McFadden; Yoav Shoham
  2. Ryan Porter; Balaji Prabhakar; Yoav Shoham; Shobha Venkataraman
  3. Moshe Tennenholtz
- to other coauthors of work in my thesis:
  - Navin A.R. Bhat, Yuzo Fujishima, Mark Pearson
- to members of the multiagent group, past & present
- to many friends who offered help and support
- to my family and my girlfriend, Judith

And thanks for your attention!

Distribution Induction

- D: original distribution of instances
- H_f: model of portfolio runtime
  - h_f: normalized for interpretation as a density function
- Goal: generate instances from $D \cdot h_f$
  - D is a distribution over the parameters of an instance generator
  - h_f depends on features of generated instance
- Rejection sampling
  1. Create model of hardness H_p using parameters of the instance generator as features; normalize it to create a PDF h_p
  2. Generate an instance from $D \cdot h_p$
  3. Keep the sample with probability proportional to $\dfrac{H_f(s)}{h_p(s)}$
  4. Else, goto 2