ACE Market Game Examples

Presenter:

Leigh Tesfatsion
Professor of Economics and Mathematics
Department of Economics
Iowa State University
Ames, Iowa 50011-1070
http://www.econ.iastate.edu/tesfatsi/
tesfatsi@iastate.edu

Outline

- ACE double-auction trading game
- An ACE two-sector trading game

EX 1: ACE Double-Auction Trading Game

- J. Nicolaisen, V. Petrov, L. Tesfatsion, *IEEE Transactions on Evolutionary Computation*,5(5),2001,504-523
  http://www.econ.iastate.edu/tesfatsi/mpeieee.pdf

- **Key Issue Addressed:**

Relative role of structure vs. learning in determining performance of a double-auction design for a day-ahead electricity market.

Key Issues We Address

* Sensitivity of market performance to changes in **market structure:**

  RCON = Relative seller/buyer **concentration**

  RCAP = Relative demand/supply **capacity**

* Sensitivity of market performance to changes in **trader learning:**

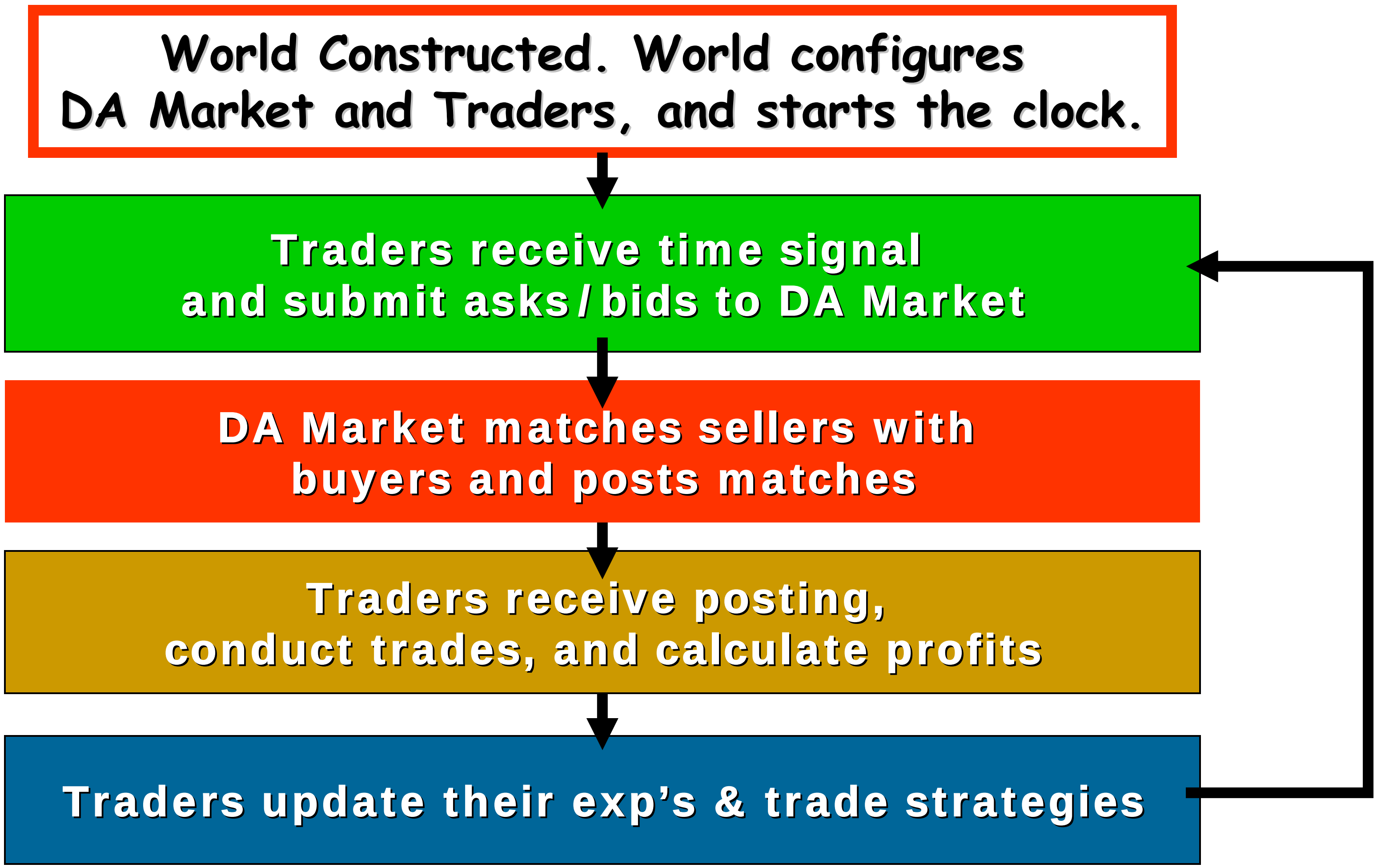
  Individual learning via **Reinforcement Learning (RL)**

  Social mimicry via **Genetic Algorithms (GAs)**

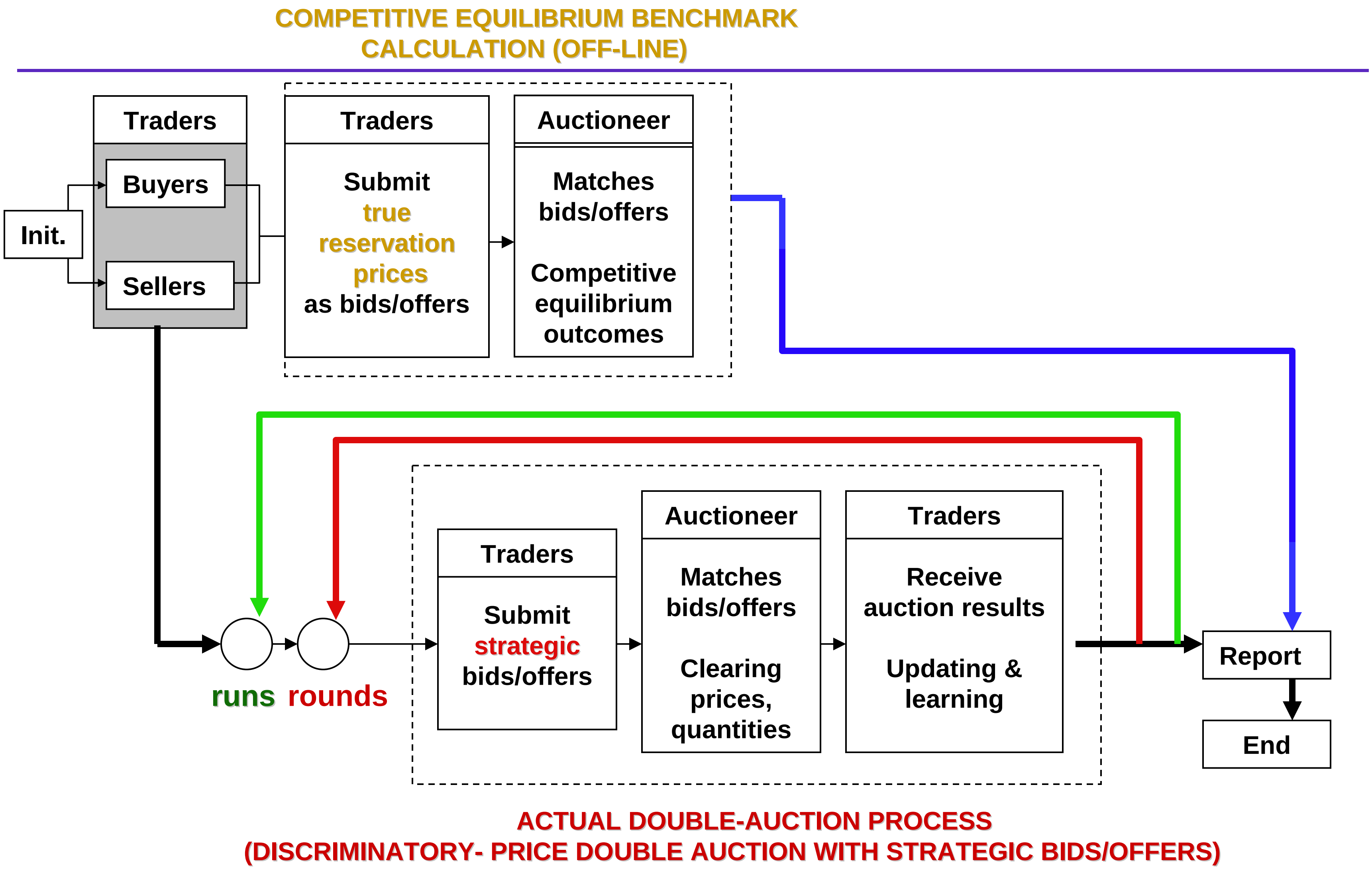
Market Performance Measures

- **Market Efficiency:** Actual total net benefits extracted from the market relative to maximum possible total net benefits (competitive benchmark).
- **Market power:** The manner in which extracted total net benefits are distributed among the market participants.

Dynamic Flow of DA Market: Simple View



Dynamic Flow of DA Market: Detailed View

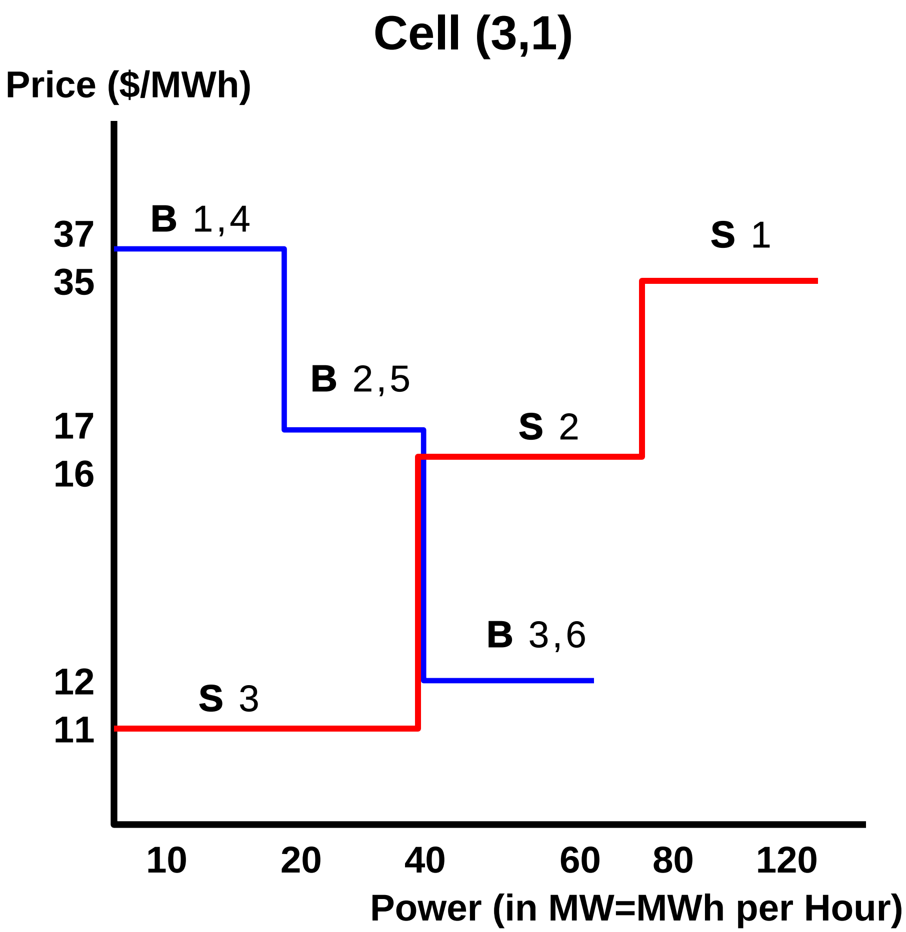


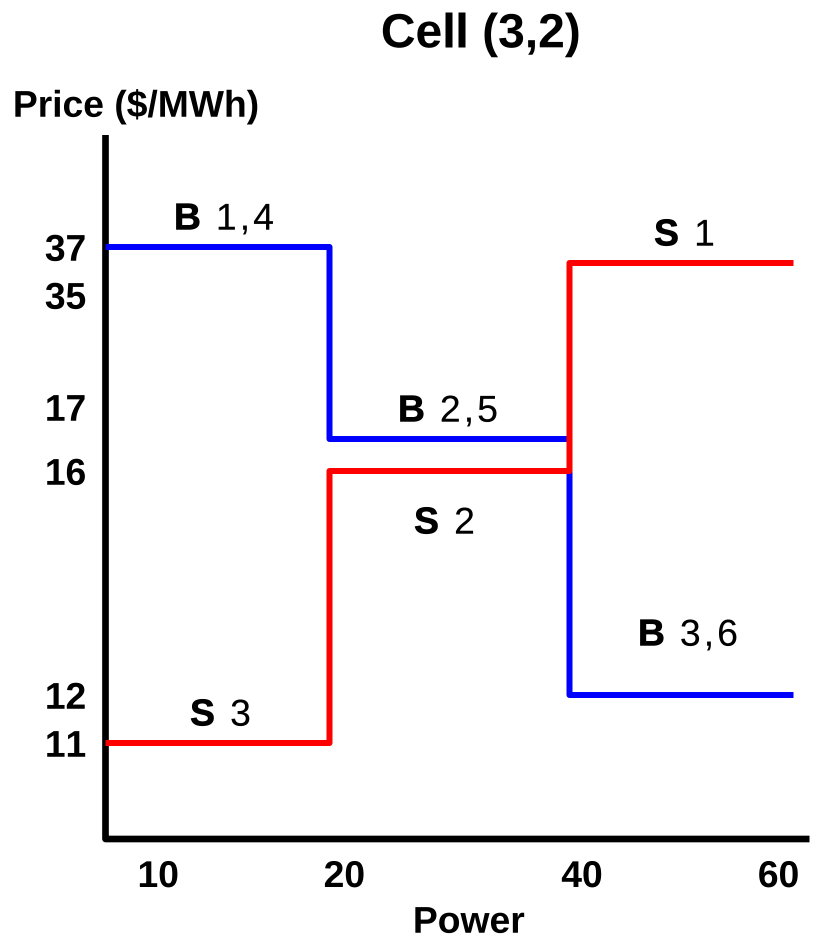
Structural Treatment Factor Values
(tested for each learning treatment)

Ns = Number of Sellers
Nb = Number of Buyers
Cs = Seller Supply Capacity
Cb = Buyer Demand Capacity
RCON=Ns/Nb
RCAP=NbCb/NsCs

		RCAP		
		1 / 2	1	2
RCON	2	Ns = 6 Nb = 3 Cs = 10 Cb = 10	Ns = 6 Nb = 3 Cs = 10 Cb = 20	Ns = 6 Nb = 3 Cs = 10 Cb = 40
	1	Ns = 3 Nb = 3 Cs = 20 Cb = 10	Ns = 3 Nb = 3 Cs = 10 Cb = 10	Ns = 3 Nb = 3 Cs = 10 Cb = 20
	1 / 2	Ns = 3 Nb = 6 Cs = 40 Cb = 10	Ns = 3 Nb = 6 Cs = 20 Cb = 10	Ns = 3 Nb = 6 Cs = 10 Cb = 10

True Total Demand and Supply Schedules
(True Reservation Prices)





The Computational World

Public Access:

// **Public Methods**

The ***World Event Schedule,*** i.e., a system clock that permits inhabitants to time and synchronize activities (e.g., submission of asks/bids into the DA market);

Protocols governing trader collusion;

Protocols governing trader insolvency;

Methods for receiving data;

Methods for retrieving World data.

Private Access:

// **Private Methods**

Methods for gathering, storing, and sending data;

// **Private Data**

World attributes (e.g., spatial configuration);

World inhabitants (DA market, buyers, sellers);

World inhabitants’ methods and data.

The Computational DA Market

Public Access:

// **Public Methods**

getWorldEventSchedule(clock time);
Protocols governing the public posting of bids/offers;
Protocols governing matching, trades, and settlements;
Methods for receiving data;
Methods for retrieving Market data.

Private Access:

// **Private Methods**

Methods for gathering, storing, and sending data.

// **Private Data**

Data recorded about sellers (e.g., seller offers);
Data recorded about buyers (e.g., buyer bids);
Address book (communication links).

A Computational DA Trader

Public Access:

// **Public Methods**

getWorldEventSchedule(clock time);
getWorldProtocols (collusion, insolvency);
getMarketProtocols (posting, matching, trade, settlement);
Methods for receiving data;
Methods for retrieving Trader data.

Private Access:

// **Private Methods**

Methods for gathering, storing, and sending data;
Methods for calculating expected & actual profit outcomes;
Method for updating my bid/offer strategy (**LEARNING**).

// **Private Data**

Data about me (history, profit function, current wealth,…);
Data about external world (rivals' bids/offers, …);
Address book (communication links).

What Do DA Traders Learn?

Supply Offers and Demand Bids

- **Offer for each Seller i** = ***reported*** supply q_i^S of real power in Mega-Watts (MWs) together with a ***reported*** unit (i.e., per-MW) price p_i in dollars \$ per MW

- **Bid for each Buyer j** = ***reported*** demand q_j^D for real power in MWs together with a ***reported*** unit price p_j in \$ per MW

- *Action choices for sellers* = Their possible OFFERS

- *Action choices for buyers* = Their possible BIDS

How Might DA Traders Learn?

- One possibility:

  Reactive Reinforcement Learning (RL)

Asks….

Given ***past*** events, what action should I take ***now*** ?

Examples: Three-parameter RL based on human-subject experiments (Roth-Erev, 1995,1998), Modified Roth-Erev RL for electricity double auctions (Nicolaisen, Petrov, Tesfatsion, IEEE TEC, 2001)

How Might DA Traders Learn…

* Another possibility:

  Anticipatory Learning

Asks….

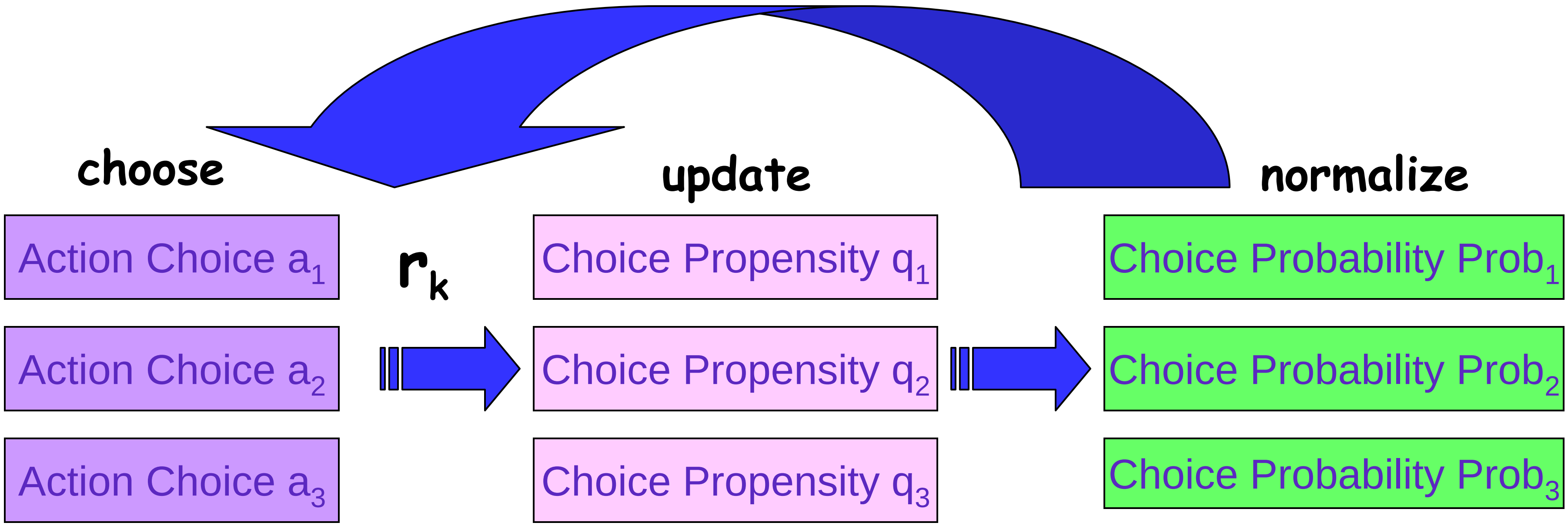
If I take this action ***now***, what will happen in the ***future*** ?

Examples: Q-Learning (Watkins, 1989); Temporal-Difference Reinforcement Learning (Sutton/Barto, 1998)

Learning Method Used for This study:

Reactive Reinforcement Learning

MRE = Modified Roth-Erev RL (Nicolaisen et al.,2001)



- Each trader maintains action choice propensities q, normalized to action choice probabilities Prob, to choose actions. A good (bad) profit r_k for action a_k results in a strengthening (weakening) of the propensity q_k for a_k.

MRE = Modified Roth-Erev RL

1. **Initialize** action propensities to an initial propensity value.
2. **Generate** choice probabilities for all actions using current propensities.
3. **Choose** an action according to the current choice probability distribution.
4. **Update** propensities for all actions using the reward for the last chosen action.
5. **Repeat** from step 2.

MRE Updating of Action Propensities

Parameters:

- $q_j(1)$ Initial propensity
- ϵ Experimentation
- ϕ Recency (forgetting)

Variables:

- a_j Current action choice
- q_j Propensity for action a_j
- a_k Last action chosen
- r_k Reward for action a_k
- t Current time step
- N Number of actions

$$q_j(t+1) = [1-\phi]q_j(t) + E_j(\epsilon, N, k, t)$$

$$E_j(\epsilon, N, k, t) = \begin{cases} r_k(t)[1-\epsilon] & \text{if } j = k \\ q_j(t)\frac{\epsilon}{N-1} & \text{if } j \neq k \end{cases}$$

From Propensities to Probabilities for MRE

$$p_j(t) = \frac{q_j(t)}{\sum_{j=0}^{N-1} q_j(t)}$$

$p_j(t)$ = Probability of choosing action j at time t

N = Number of available actions at each time t

Sample Table of Experimental Results

TABLE VI

EXPERIMENTAL MARKET POWER AND EFFICIENCY OUTCOMES FOR THE BEST FIT MRE ALGORITHM WITH 1000 AUCTION ROUNDS AND PARAMETER VALUES $s(1) = 9.00$, $r = 0.10$, AND $e = 0.20$

Relative Concentration		Relative Capacity 1/2 MP	1/2 StdDev	1 MP	1 StdDev	2 MP	2 StdDev
2	All Buyers:	-0.13*	(0.09)	-0.15*	(0.09)	0.10	(0.30)
	All Sellers:	0.55*	(0.38)	0.38*	(0.33)	-0.10	(0.25)
	Buyer[1]:	-0.12*	(0.08)	-0.13*	(0.10)	0.10	(0.30)
	Buyer[2]:	-0.20	(0.40)	-0.75*	(0.33)	ZP	(0.00)
	Buyer[3]:	ZP	(0.00)	ZP	(0.00)	ZP	(0.00)
	Seller[1]:	ZP	(0.00)	ZP	(0.00)	ZP	(0.00)
	Seller[2]:	ZP	(0.00)	-0.50	(1.34)	-0.12	(0.34)
	Seller[3]:	0.54	(0.63)	0.45*	(0.40)	-0.10	(0.22)
	Seller[4]:	ZP	(0.00)	ZP	(0.00)	ZP	(0.00)
	Seller[5]:	ZP	(0.00)	-0.42	(1.67)	-0.08	(0.36)
	Seller[6]:	0.55	(0.60)	0.46*	(0.41)	-0.09	(0.24)
	Efficiency:	99.81	(0.02)	96.30	(0.05)	99.88	(0.06)
1	All Buyers:	-0.22*	(0.12)	-0.13*	(0.10)	0.13	(0.33)
	All Sellers:	0.80*	(0.53)	0.28	(0.35)	-0.10	(0.26)
	Buyer[1]:	-0.21*	(0.11)	-0.11*	(0.10)	0.13	(0.33)
	Buyer[2]:	-0. 31	(0.44)	-0.80*	(0.40)	ZP	(0.00)
	Buyer[3]:	ZP	(0.00)	ZP	(0.00)	ZP	(0.00)
	Seller[1]:	ZP	(0.00)	ZP	(0.00)	ZP	(0.00)
	Seller[2]:	ZP	(0.00)	-0.37	(1.89)	-0.10	(0.34)
	Seller[3]:	0.76*	(0.63)	0.34	(0.45)	-0.11	(0.24)
	Efficiency:	92.13	(0.09)	94.59	(0.07)	100.00	(0.00)
1/2	All Buyers:	-0.21*	(0.12)	-0.14*	(0.08)	0.09	(0.24)
	All Sellers:	0.67*	(0.46)	0.30	(0.31)	-0.07	(0.19)
	Buyer[1]:	-0.18*	(0.12)	-0.14*	(0.10)	0.09	(0.27)
	Buyer[2]:	-0.37	(0.47)	-0.77*	(0.44)	ZP	(0.00)
	Buyer[3]:	ZP	(0.00)	ZP	(0.00)	ZP	(0.00)
	Buyer[4]:	-0.20*	(0.11)	-0.11	(0.11)	0.10	(0.25)
	Buyer[5]:	-0.38	(0.47)	-0.73*	(0.46)	ZP	(0.00)
	Buyer[6]:	ZP	(0.00)	ZP	(0.00)	ZP	(0.00)
	Seller[1]:	ZP	(0.00)	ZP	(0.00)	ZP	(0.00)
	Seller[2]:	ZP	(0.00)	0.14	(2.69)	-0.08	(0.27)
	Seller[3]:	0.63*	(0.55)	0.32	(0.48)	-0.07	(0.17)
	Efficiency:	91.84	(0.09)	94.24	(0.07)	100.00	(0.00)

ZP indicates that zero profits were earned both in the auction and in competitive equilibrium.

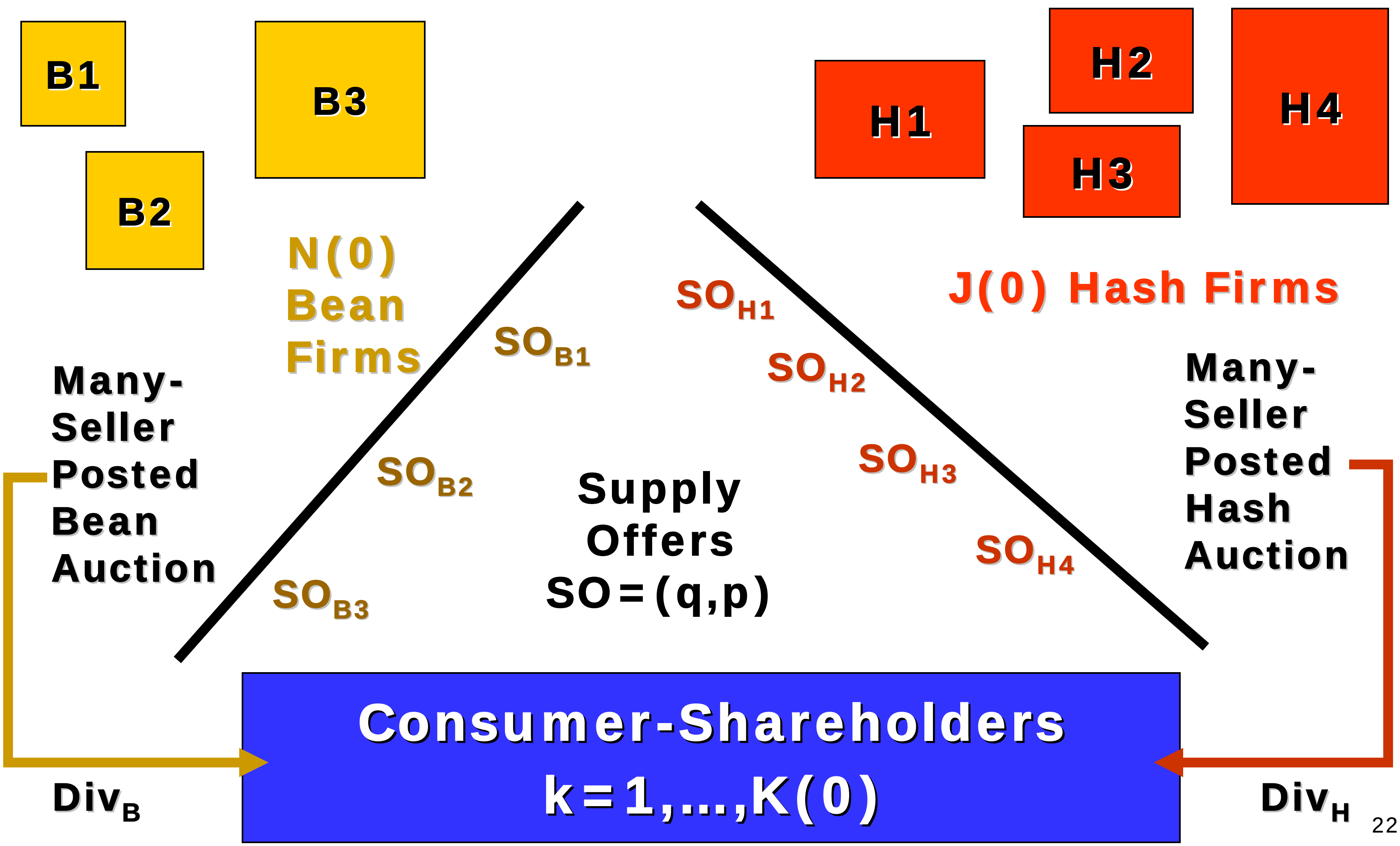
Summary of Policy-Relevant DA Findings

- **Market Efficiency:** Generally high when traders use MRE (Modified Roth-Erev) reinforcement learning **but not** when traders use GA (genetic algorithm) social mimicry (***type of learning can matter***).

- **Structural Market Power:** Microstructure of the DA market is strongly predictive for the relative market power of traders (***rule details matter***).

- **Strategic Market Power:** Traders are **not** able to change their relative market power through learning (***importance of countervailing power***).

Ex 2: An ACE Bilateral Trade Hash-and-Beans Economy



Dynamic Flow of ACE H&B Economy

World Constructed. World configures Markets, Firms, Consumers and starts the clock.

↓

Firms receive time signal and post quantities/prices in H & B markets

↓

Consumers receive time signal and begin price discovery process

↓

Firms-consumers match, trade, calculate profits/utilities & update wealth levels

↓

Firms update their exp's & prod/price strategies

Dynamic Flow of Activity for H & B Firms

- Each firm f starts out (T=0) with ***money $M_f(0)$*** and a ***production capacity $Cap_f(0)$***
- Firm f's ***fixed cost $FC_f(T)$*** in each $T \geq 0$ is proportional to its current capacity $Cap_f(T)$
- At beginning of each $T \geq 0$, firm f selects a ***supply offer = (production level, unit price)***
- At end of $T \geq 0$, firm f is ***solvent*** if it has $NetWorth(T) = [Profit(T)+M_f(T)+ValCap_f(T)] > 0$
- If solvent, firm f ***allocates its profits*** (+ or -) between M_f, CAP_f, and dividend payments.

Dynamic Flow of Activity for Consumer-Shareholders

- Each consumer k starts out (T=0) with a ***lifetime money endowment profile***

  $$(Mk_{youth} , Mk_{middle} , Mk_{old})$$

- In each T ≥ 0, consumer k's **utility** is measured by $U_k(T) = (hash(T) - h_k^*)^{\alpha_k} \bullet (beans(T) - b_k^*)^{[1-\alpha_k]}$

- In each T ≥ 0, consumer k seeks to secure maximum utility by ***searching*** for beans and hash to buy at ***lowest possible prices***.

- At end of each T ≥ 0, consumer k ***dies*** unless consumption meets subsistence needs (b_k^*, h_k^*).

Experimental Design Treatment Factors

- **Initial size of consumer sector** [K(0)]
- **Initial concentration** [N(0), J(0), Cap(0) values]
- **Firm learning** (supply offers & profit allocations)
- **Firm cost functions**
- **Firm initial money holdings** [$M_f(0)$]
- **Firm rationing protocols** (for excess demand)
- **Consumer price discovery processes**
- **Consumer money endowment profiles/TMax** (rich, poor, ↗ , ↘ , life cycle u-shape)
- **Consumer preferences** (θ values)
- **Consumer subsistence needs** (b*,h*)

The Computational World

Public Access:

// **Public Methods**

The ***World Event Schedule,*** i.e., a system clock that permits inhabitants to time and synchronize activities (e.g., opening/closing of H & B markets);

Protocols governing firm collusion;

Protocols governing firm insolvency;

Methods for receiving data;

Methods for retrieving World data.

Private Access:

// **Private Methods**

Methods for gathering, storing, and sending data;

// **Private Data**

World attributes (e.g., spatial configuration);

World inhabitants (H & B markets, firms, consumers);

World inhabitants' methods and data.

A Computational Market

Public Access:

// **Public Methods**

getWorldEventSchedule(clock time);
Protocols governing the public posting of supply offers;
Protocols governing matching, trades, and settlements;
Methods for receiving data;
Methods for retrieving Market data.

Private Access:

// **Private Methods**

Methods for gathering, storing, and sending data.

// **Private Data**

Data recorded about firms (e.g., sales);
Data recorded about consumers (e.g., purchases);
Address book (communication links).

A Computational Consumer

Public Access:

// **Public Methods**

getWorldEventSchedule(clock time);
getWorldProtocols (stock share ownership);
getMarketProtocols (price discovery process, trade process);
Methods for receiving data;
Methods for retrieving stored Consumer data.

Private Access:

// **Private Methods**

Methods for gathering, storing, and sending data;
Method for determining my budget constraint;
Method for searching for lowest prices.

// **Private Data**

Data about me (history, utility function, current wealth,…);
Data about external world (posted supply offers, …);
Address book (communication links).

A Computational Firm

Public Access:

// **Public Methods**

getWorldEventSchedule(clock time);

getWorldProtocols (collusion, insolvency);

getMarketProtocols (posting, matching, trade, settlement);

Methods for receiving data;

Methods for retrieving Firm data.

Private Access:

// **Private Methods**

Methods for gathering, storing, and sending data;

Methods for calculating expected & actual profit outcomes;

Method for allocating my profits to my shareholders;

Method for updating my supply offers (**LEARNING**).

// **Private Data**

Data about me (history, profit function, current wealth,…);

Data about external world (rivals' supply offers, …);

Address book (communication links).

Interesting Issues for Exploration

- Initial conditions → **carrying capacity?** (Survival of firms/consumers in long run)
- Initial conditions → **market clearing?** (Walrasian equilibrium benchmark)
- Initial conditions → **market efficiency?** (Walrasian equilibrium benchmark)
- Standard concentration measures at T=0 → **good predictors of long-run market power?**
- Importance of **learning vs. market structure** for market performance? *(Gode/Sunder, JPE, 1993)*

ACE Hash-and-Beans Economy: Comp Lab Implementation

Christopher Cook and Leigh Tesfatsion, **"Agent-Based Computational Laboratories for the Experimental Study of Complex Economic Systems,"** Working Paper, ISU Department of Economics, in progress.

- Computational laboratory under construction for the ACE Hash-and-Beans Economy
- Programming language C#/.Net (all WinDesktops)
- Under development for Econ 308 (ACE course)

www.econ.iastate.edu/classes/econ308/tesfatsion/

ACE Hash & Beans Economy: Comp Lab Main Screen