

Market Mechanisms and Alternative Equilibria

Algorithmic Game Theory Using Correlated Equilibria and
Computationally Feasible Algorithms

What We'll Cover

- Dubey's limit order market mechanism
- The complexity of computing Nash Equilibria
- Correlated and Coarse Correlated Equilibria
- Constraints on Blockchain Algorithms
- Tractability concerns of computing equilibria
- Machine learning algorithms for equilibrium estimation

Definitions of the Game

- A market E consists of N players $i \in \{1, 2, \dots, n\}$ and k goods
- Each player i has endowment over the goods $a^i \in \mathbb{R}_+^k$ and a utility function $u^i: \mathbb{R}_+^k \rightarrow \mathbb{R}$
- A strategy for player i consists of $\{p^i, q^i, \tilde{p}^i, \tilde{q}^i\}$ where $p^i, q^i, \tilde{p}^i, \tilde{q}^i \geq 0$ and $\forall j \in k, \tilde{q}_j^i \leq a_j^i$
 - Statement of player i saying “I am willing buy q_j^i units of j if the price is p_j^i or less, and I am willing to sell $\tilde{q}_j^i$ units if the price is $\tilde{p}_j^i$ or more”
- A player's strategies is the set S^i , and $S = S^1 \times S^2 \times \dots \times S^N$ is the full strategy space
- An outcome consists of $x = \{x^1, \dots, x^n\}$, where $x^i \in \mathbb{R}_+^K$ is the bundle of goods i receives
- The game $\Gamma(E, \lambda)$ denotes the entire market and mechanism, where λ is a penalty to players whose net credit after trade is negative

Dubey Mechanism Setup

- There exists a trading post for each of the k goods
- In each period, all N players submit strategies s^i
- For each trading post, the highest buyer purchases from the lowest seller at the price quoted by the buyer
 - If there is surplus demand from the buyer after the first seller, he purchases from the second lowest seller. Similarly, if the seller has excess quantity after the buyer he then sells to the second highest buyer
- Each player can borrow at no interest the amount needed to fund their purchases, however if they have net negative credit at the end of the period, they incur a cost $\lambda * \beta$, where β is their net credit after trade

Dubey Mechanism Properties

- For a given strategy $s = \{s^i: i \in N\} \in S$, payoff function Π , and a deviation for M players: $e = \{s^i: i \in M\} \in \times_{i \in M} S^i$, define s to be M -efficient if **no** e exists such that:
 - $\Pi^i(s|e) \geq \Pi^i(s) \quad \forall i \in N$ No coalition of size M can find
 - $\Pi^j(s|e) > \Pi^j(s) \quad j \in N$ a *pareto dominant* strategy
- s is a *Noncooperative Equilibrium* (N.E.) if it is $\{i\}$ -efficient $\forall i \in N$, and if it is M -efficient $\forall M \subset N$, it is a *Strong Noncooperative Equilibrium*
- A N.E. is *active* if each trading post has at least 2 buyers and 2 sellers
- A N.E. is *tight* if all active traders quote the same price

- For any market E and penalty $\lambda > 0$:
 - The active N.E. of $\Gamma(E, \lambda)$ coincide with the Competitive Equilibria
 - The tight, active N.E. of $\Gamma(E, \lambda)$ also coincides with the Competitive Equilibria
 - Every tight, active N.E. of $\Gamma(E, \lambda)$ is also strong

Mechanism Application: SPEEDEX

- SPEEDEX (NDSI 23) is a decentralized limit-order exchange built on blockchain technology
- Players submit exchange bids through smart contracts to the mechanism (i.e. 100 USD : 110 EUR)
- In each block, SPEEDEX calculates the market clearing prices for each good, and executes all outstanding orders at that price
 - This may seem like a better clearing strategy, however Dubey shows that this policy can lead to noncooperative equilibria that are not competitive or pareto optimal
- The prices are calculate using a Tâtonnement iterative process

Mechanism Application: SPEEDEX

- Runtime of $O(\# assets^2 * \log(\# orders))$ per block
- Clearing prices are consistent across goods, so $\frac{P_A}{P_B} * \frac{P_B}{P_C} = \frac{P_A}{P_C}$, eliminating arbitrage opportunities
- All transactions occur at clearing prices, preventing frontrunning attacks on the exchange
 - Frontrunning is a prevalent issue in other DEX implementations, where well placed agents can ‘spy’ on submitted offers, notice a transaction T , and submit a new transaction T' that executes before T and buys and re-sells an asset to T at a slightly higher price
 - Block producers can do this by reordering transactions in a block to increase their own profits, known as “Miner Extractable Value”

Tractability Concerns

- Dubey's mechanism, and by extension SPEEDEX, benefit from efficient properties in equilibrium, however, have no guarantees on convergence
- Finding the N.E. of the game requires each agent to rationally predict the strategies of others and submit the optimal bid vector accordingly
- With (almost) continuous valued prices across many goods, the strategy space for each agent is extremely large and very hard to search over
- Even with many iterations of the mechanism over a large time horizon, realizing an efficient N.E. may be intractable

- Nash equilibrium computation is PPAD-complete, meaning all known algorithms have complexity exponential in number of players and strategies
- Competitive equilibrium is also PPAD-complete for general Arrow-Debreu exchange economies
- This formalizes our concern that an agent faced with many goods and (almost) continuous prices has a very complex optimization problem, and convergence of the mechanism to a Nash equilibrium may be prohibitively difficult

- NP denotes all problems that, given a potential solution, can be verified in polynomial time
- A “Hard” problem in a complexity class is any problem known to be at least as difficult to compute as any other problem in the class
- A “Complete” problem in a complexity class is a problem A where a polynomial time algorithm exists that reduces any other problem in the class B to A , i.e. a solution to A is valid if and only if it is a solution to B

- Nash proved that every game has a Nash equilibrium in mixed strategies
- Simple classes of games, such as 2-player zero-sum games, have polynomial time algorithms to find Nash equilibria
- For general games, researchers created the complexity class TFNP, or “NP total functions”, consisting of all search problems with a guaranteed solution
 - A “semantic class”, known to have no complete problems, subset of NP
- A subclass of TFNP, PPAD, is the class defined by the problem: *In any directed graph with one unbalanced node (node with outdegree different from its indegree), there must be another unbalanced node*
- Computing a Nash equilibrium was proven to be PPAD-complete by Daskalakis, Goldberg, and Papadimitriou

- As an alternative to Nash equilibria, there exist equilibrium concepts that are computationally feasible, however require coordination among players
- Instead of each player individually computing their optimal strategy profile, let there be a central coordinator, who chooses the strategies of each player
- The coordinator keeps a joint probability distribution $p \in S^1 \times S^2 \times \dots \times S^N \rightarrow [0,1)$ across the strategies of all players
- For each iteration t of the game, the coordinator randomly samples a strategy profile s_t from joint probability distribution p
- Each player knows the distribution p , as well as their selected strategy s_t^i

Correlated Equilibrium

- A Correlated equilibrium arises when each player i , knowing their signaled strategy s_t^i and p , has no incentive to deviate from the signal

$$\sum_{s^{-i} \in S^{-i}} u^i(s^i, s^{-i}) p_{s^i, s^{-i}} \geq \sum_{s^{-i} \in S^{-i}} u^i(s', s^{-i}) p_{s^i, s^{-i}}, \quad \forall s' \in S^i$$

- Each player faces $O(\# \text{ strategies}^2)$ constraints
- Called a Correlated equilibrium because knowing your strategy gives you information on the possible strategies of the other players

Correlated Equilibrium

P1	L	R
T	3	1
B	2	7

P2	L	R
T	4	8
B	6	5

D	L	R
T	p_{TL}	p_{TR}
B	p_{BL}	p_{BR}

P1:

$$3p_{TL} + 1p_{TR} \geq 2p_{TL} + 7p_{TR}$$

$$2p_{BL} + 7p_{BR} \geq 3p_{BL} + 1p_{BR}$$

P2:

$$4p_{TL} + 6p_{BL} \geq 8p_{TL} + 5p_{BL}$$

$$8p_{TR} + 5p_{BR} \geq 4p_{TR} + 6p_{BR}$$

- A Coarse Correlated equilibrium is a similar notion of equilibrium, except now the distribution p must only be incentive compatible prior to player i learning their strategy

$$\mathbf{E}[u^i(p)] \geq \sum_{s^{-i} \in S^{-i}} u^i(s', s^{-i}) \sum_{s^i=1}^{|S^i|} p_{s^i, s^{-i}}, \quad \forall s' \in S^i$$

- The distribution must be better in expectation than any *fixed* strategy s'
- Each player faces $O(\# \text{ strategies})$ constraints

Coarse Correlated Equilibrium

P1	L	R
T	3	1
B	2	7

P2	L	R
T	4	8
B	6	5

D	L	R
T	p_{TL}	p_{TR}
B	p_{BL}	p_{BR}

P1:

$$u^1(D) = 3p_{TL} + 1p_{TR} + 2p_{BL} + 7p_{BR}$$
$$u^1(D) \geq 3(p_{TL} + p_{BL}) + 1(p_{TR} + p_{BR})$$
$$u^1(D) \geq 2(p_{TL} + p_{BL}) + 7(p_{TR} + p_{BR})$$

P2:

$$u^2(D) = 4p_{TL} + 8p_{TR} + 6p_{BL} + 5p_{BR}$$
$$u^2(D) \geq 4(p_{TL} + p_{TR}) + 6(p_{BL} + p_{BR})$$
$$u^2(D) \geq 8(p_{TL} + p_{TR}) + 5(p_{BL} + p_{BR})$$

(Coarse) Correlated Equilibrium

- $NE \subseteq CE \subseteq CCE$
 - A Nash equilibrium is a special case of CE where $p_{s^1, \dots, s^N} = p_{s^1} * \dots * p_{s^N}$, or joint distribution is independent between players
- The difference between CE and CCE can be thought of as a difference in commitment: CCE requires commitment prior to learning your signal, while CE requires commitment after receiving your signal
- The coordinator allows players to “correlate” their strategies, greatly reducing the complexity of each player’s optimization problem
- (C)CE requires trust in the coordinator, all players must believe they are receiving fair draws from the distribution
 - Smart contracts allows this trust, as each player can independently inspect the code to verify the distribution and fair sampling

(Coarse) Correlated Equilibrium

- Constraints are all linear combinations of the joint probabilities
- Can easily solve with linear programming methods in polynomial time
 - N players, k strategies leads to $O(Nk^2)$ constraints for CE, $O(Nk)$ for CCE
 - Added $O(1)$ constraints for $\sum p = 1, p_s \geq 0, \forall s \in S$
- Using linear programming, no extra complexity to maximize an objective function while computing equilibria, such as $\max \sum_{i=1}^N \lambda_i * E(u^i)$, the pareto planner problem for weights λ

(Coarse) Correlated Equilibrium

P1 L R

T	3	1
B	2	7

P2 L R

T	4	8
B	6	5

D L R

T	p_{TL}	p_{TR}
B	p_{BL}	p_{BR}

$$\max_{\substack{p_{TL}, p_{TR}, p_{BL}, p_{BR} \\ \lambda_1, \lambda_2 = 1}} (3 + 4)p_{TL} + (1 + 4)p_{TR} + (2 + 6)p_{BL} + (7 + 5)p_{BR}$$

$$(3 - 2)p_{TL} + (1 - 7)p_{TR} \geq 0$$

$$(2 - 3)p_{BL} + (7 - 1)p_{BR} \geq 0$$

$$(4 - 8)p_{TL} + (6 - 5)p_{BL} \geq 0$$

$$(8 - 4)p_{TR} + (5 - 6)p_{BR} \geq 0$$

$$0 \leq p_{TL}, p_{TR}, p_{BL}, p_{BR} \leq 1$$

$$p_{TL} + p_{TR} + p_{BL} + p_{BR} = 1$$

Why Complexity Matters

- If algorithms for computing Nash in PPAD time exist, why do we care about the complexity?
- When new blocks are committed on Ethereum, all nodes must run transactions on smart contracts to update the state
- To ensure the blockchain can add new blocks in a reasonable time, there are limits on the complexity allowed by each contract
- While complexity constraints differ, most blockchains require that all smart contract code must be written such that polynomial execution time is guaranteed for any input

(C)CE in Dubey's Game

- Linear programming gives polynomial time algorithms for CE, however it is linear in the number of outcomes, which is of the size $O(k^N)$
- For nearly continuous action spaces or many players, the strategy space grows exponentially and quickly becomes intractable
 - Example: 2 goods, 2 players, integers $[0, 9]$ for possible prices and quantities $p, q, \tilde{p}, \tilde{q}$
 - Each player has 100,000,000 possible strategies, meaning 10^{16} joint strategy outcomes
 - 10^{24} for 3 players!
- Even with polynomial time algorithms, further optimization is needed

No Regret Learning

- No regret learning is a class of machine learning algorithms where an agent tries to make actions that maximizes its reward
- It is an *online* algorithm, meaning that it receives input at each timestep $0 \leq t \leq T$, and must make decisions without knowing all T inputs
- At each step t the algorithm probabilistically chooses some action $a \in S$, and receives feedback on the reward of the chosen action
- We usually consider cost, the dual of utility, in no regret learning. The cost of an action can be computed as the inverse utility, scaled and normalized from $[0,1]$ such that
$$c_t(a) = \frac{u_{\max} - u(a)}{u_{\max} - u_{\min}}$$
- In general, the cost of any action at time t is decided by an adversary who arbitrarily chooses a mapping from action to costs $c_t: [0,1] \times N$, however we are specifically interested in multiplayer dynamics, where costs are determined by the interactions of many players submitting limit orders according to Dubey's mechanism

- To evaluate our algorithm, we do not compare its choices to the best possible choices
 - It is impossible to be competitive with the optimal algorithm. Imagine an adversary which picks costs such that all actions have cost 1, except a randomly chosen a_t has $c(a_t) = 0$
 - Optimal choices will always guess a_t correctly, netting 0 cost. Best online algorithm will incur expected cost $\frac{N-1}{N} * T$
- Instead, we use *regret*, or compare the choices in retrospect to some simple alternative policy $\pi \in G$, where G is some simple class of decisions
- We then look for algorithms that perform well against G as T grows

No Regret Learning

- *External regret* compares the chosen actions to best fixed action in retrospect, where $G_{ex}(t) = X$, for fixed $X \in S$, is the class of fixed strategies
- External regret algorithms H attempt to minimize $R_H - R_G$, where R_H is the cost of the actions selected by H , and R_G is $\min_{a \in S} \sum_{\tau=1}^t c_\tau(a)$
- Randomized Weighted Majority is an algorithm for minimizing external regret over time, with regret bounded by $O(\sqrt{T \log N})$

No Regret Learning

Randomized Weighted Majority (RWM) Algorithm

Initially: $w_i^1 = 1$ and $p_i^1 = 1/N$, for $i \in X$.

At time t : If $\ell_i^{t-1} = 1$, let $w_i^t = w_i^{t-1}(1 - \eta)$; else ($\ell_i^{t-1} = 0$) let $w_i^t = w_i^{t-1}$
 Let $p_i^t = w_i^t / W^t$, where $W^t = \sum_{i \in X} w_i^t$.

- w_i^t is the weight the player assigns to strategy i at time t
- ℓ_i^{t-1} is the cost of action i in the previous timestep. The algorithm receives the cost of **all actions**, not just the one selected
- Given $\ell_i^{t-1} \in \{0,1\}$, update w_i^t by learning rate η
- At time t , player randomly chooses strategy i with probability $p_i^t = \frac{w_i^t}{\sum_i w_i^t}$

- In the multiplayer setting, where multiple agents are using the same algorithm H to choose actions, time averaged strategies converge to CCE

Proposition 3.1 *Suppose after T iterations of no-regret dynamics, every player of a cost-minimization game has regret at most ϵ for each of its strategies. Let $\sigma^t = \prod_{i=1}^k p_i^t$ denote the outcome distribution at time t and $\sigma = \frac{1}{T} \sum_{t=1}^T \sigma^t$ the time-averaged history of these distributions. Then σ is an ϵ -approximate coarse correlated equilibrium, in the sense that*

$$\mathbf{E}_{\mathbf{s} \sim \sigma}[C_i(\mathbf{s})] \leq \mathbf{E}_{\mathbf{s} \sim \sigma}[C_i(s'_i, \mathbf{s}_{-i})] + \epsilon$$

for every player i and unilateral deviation s'_i .

© Tim Roughgarden. All rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use/>.

Note: ϵ is time averaged regret, or $\frac{R_H - R_G}{T}$

- Consider a broader comparison class G_{sw} such that, instead of a fixed action, we consider all possible *swaps* of actions
 - i.e. what if every time H played i , it had instead played j
 - G_{sw} therefore encompasses all mappings $G_{sw} : N \rightarrow N$
- $G_{ex} \subset G_{sw}$, external regret is the subset where $G_{sw} : N \rightarrow X$ for fixed $X \in S$

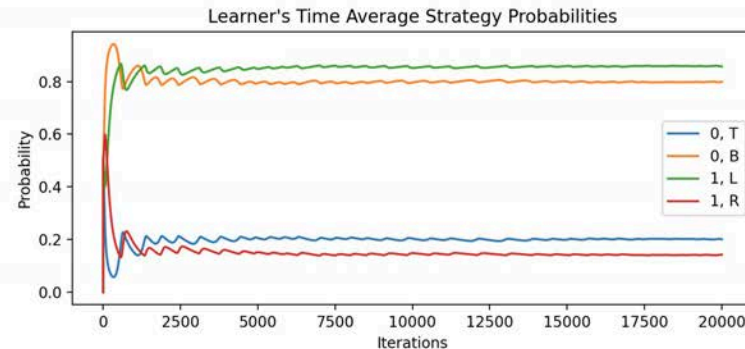
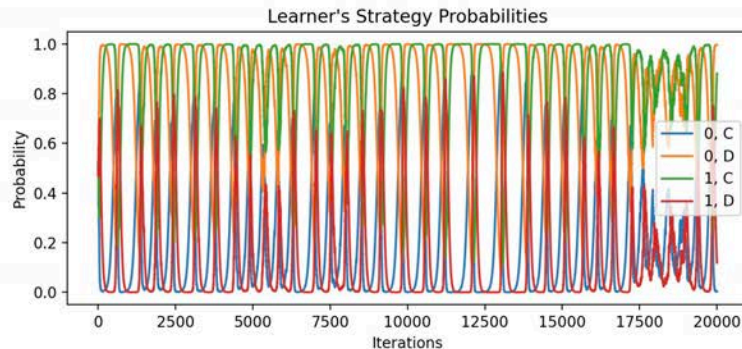
Theorem 4.12 *Let $G = \langle M, (X_i), (s_i) \rangle$ be a game and assume that for T time steps every player follows a strategy that has swap regret of at most R . Then, the empirical distribution Q of the joint actions played by the players is an (R/T) -correlated equilibrium.*

Regret Learning

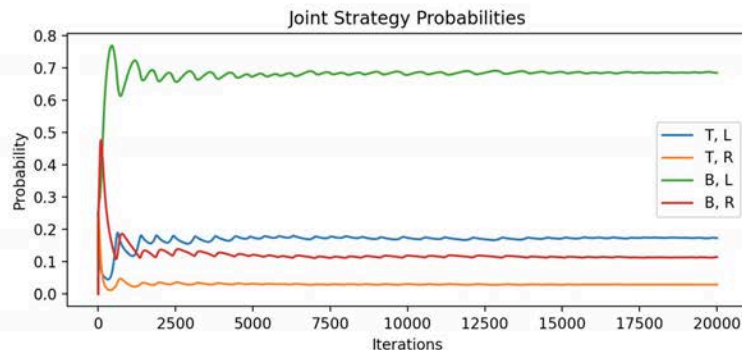
- External Regret and Swap Regret algorithms allow estimation of (C)CE without explicitly computing the entire joint probability distribution
- Importantly, efficient algorithms for minimizing regret exist, and work well in multiplayer settings
- By running many agents in a multiplayer setting and taking the average of their strategies, algorithms quickly converge on good estimates of equilibria
- For 2 player, zero-sum games, regret algorithms converge on Nash equilibria
 - Additionally for 2 player, general-sum games when agents begin with equal weights

External Regret Minimization

P1	L	R
T	3	1
B	2	7



P2	L	R
T	4	8
B	6	5



In this general-sum, 2 player game, the time average of strategies is the unique mixed strategy Nash equilibrium

- So far, we have considered algorithms that can see the entire cost vector, i.e. they can update based on the cost of all actions, not just the one chosen
- This is the *full information* setting, where each player submits a strategy and learns the expected cost of all strategies, $C(a_t) = E[c_t(a_t)], \forall a_t \in S$
- Alternatively, the *partial information* setting learns the entire cost vector, but only for the submitted strategies of the other players, $C(a_t) = c(a_t, a_{-t}), \forall a_t \in S$, where a_{-t} is the submitted strategies of the other players
- Finally, the *multi-armed bandit* setting, where the agent only learns the cost of the played action $C(a_t) = c(a_t, a_{-t}), a_t = s_t^i$

Regret Learning

- Algorithms for limited information and multi-armed bandit regret learning exist, however they converge slower than those for full information
- When designing algorithms for regret learning, consider the information readily available to the mechanism, and choose the learning algorithms accordingly
- Algorithms exist for regret learners subject to continuous action spaces, even under very limited information

Regret Learning

- Regret learning agents are online algorithms which take actions and in hindsight try to minimize regret against some simple alternative strategy
- External regret learners minimize cost compared to the best fixed strategy
- Swap regret learners minimize cost over the best fixed mapping between strategies
- Time averaged strategies of swap/external regret learners in multiplayer settings converge to correlated/coarse correlated equilibria
- These estimations can be made without computing the entire joint probability distribution, greatly reducing the complexity from the linear programming

Bringing it All Together

- Design regret learning agents and a training mechanism that can efficiently estimate a (coarse) correlated equilibrium of the Dubey game
- Use the estimated equilibrium as a distribution q which can be sampled by a coordinator
- Deploy this coordinator along with the Dubey mechanism onto the blockchain
- For each iteration of the mechanism, signal to players the equilibrium strategies, which agents are incentivized to play under proper modeling

Remaining Questions

- What is the efficiency of the equilibria computed by regret learners?
- Can the learning mechanism be adjusted to impact the efficiency of the computed strategies?
- How does the distribution q change as new agents enter the market and agents experience endowment shocks?
- How can learning agents adapt to shocks in endowments or preferences?
- What is the optimal information setting for the learners in the Dubey mechanism?

MIT OpenCourseWare

<https://ocw.mit.edu>

14.129 Advanced Contract Theory Spring 2025

For more information about citing these materials or our Terms of Use, visit <https://ocw.mit.edu/terms>.