Reinforcement Learning for Query Pricing in The Graph

Tomasz Kornuta
VP of Engineering/Head of AI

SEMIOTIC LABS
Outline

- **Introduction**
  - About Semiotic Labs
- **Automated Price Discovery: AutoAgora**
  - Problem Formulation: Dynamic Pricing
- **Reinforcement Learning 101**
- **Agent-Based Modeling** for
  - Testing system properties and outcomes
  - Single- and multiple-agent setup
- **AutoAgora in production**
- **Summary**

Team effort! Special shout outs to Alexis Asseman and Matt Deible
Introduction
About

- Founded in 2020 by AI & Cryptography researchers
- Funding from NSF, DARPA, The Graph Foundation and Infinity Ventures
- Focus on Applied Research
- Core Developer of thegraph Protocol
- Developer of ODOS the Optimal DEX Aggregator
Automated Price Discovery: AutoAgora
Automated Price Discovery: The Scenario

- **Customers** send queries to **The Gateway**
- **The Gateway** distributes queries between **indexers**
  - The decision is based on each indexer’s price-bid and its quality of service
- **Indexers** earn money by serving queries
  - **Indexers** can control the prices of served queries
- **AutoAgora = Dynamic pricing** based on query volume received by an indexer
Agent Based Modeling

- Price bids expressed as models in DSL called Agora

Queries distributed amongst agents depending on their price bids

- Queries simulated as a total query volume (q/s)
Assumptions (selected)

- **Normalized** query volume with noise \( \uparrow \) (additive white Gaussian noise)
- Customers have **limited budget** that can change over time \( \leftrightarrow \)
- Query serving **costs** are not considered => agents operate purely on revenue

- **Game:** Agent’s **revenue maximization** vs Gateway’s **quality of service**
Reinforcement Learning 101

- **Agent** interacts with the **Environment** by executing an **action**
- Agent’s actions change the **state** of the **Environment**
- Agent gets a **reward** and **observes** the new state of the Environment
- Agent updates its **policy** based on the received **reward**

Image credits: Lilian Weng (2018). A (Long) Peek into Reinforcement Learning [link]
Agents and algorithms

- **Types of agents** used in our simulations
  - Trainable (RL) vs Rule-based (i.e. with predefined behaviors)
  - Deterministic vs Stochastic

- **Types of RL algorithms** (update rules):

  ![RL Algorithms Diagram](link)
Gaussian bandits

- Gaussian bandits = trainable, stochastic agents with:
  - Policy is represented as a **gaussian distribution** over the possible query prices
  - Action is sampled from the **policy distribution** (continuous action space)
  - No internal representation of the environment (bandit)

\[
\theta = [\mu, \sigma]^T \\
\pi(a|\theta) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(a-\mu)^2}{2\sigma^2}\right)
\]
Testing Properties with Agent-Based Modeling
**Experiment 1.1**

- **Market Conditions**: Fixed customer budget
- **Bandit property tested**: Customer budget discovery

- Gaussian bandit (modified PPO)
- Distribution inversely-proportional to price bids (inverse softmax)
- Fixed customer budget, with noise
Experiment 1.2

- Query volume + consumer budget (white)
- Agent’s initial policy (dashed red)
- Agent’s current policy (red)
Experiment 1.2

- Query volume served by the agent
Experiment 1.2

- Aggregated query volume served by the agent (red)
- Aggregated volume of unserved queries (cyan)
• Agent’s revenue
• Aggregated agent's revenue
Experiment 1.3
Market Conditions: Dynamic customer budget

Bandit property tested: Customer budget discovery

Gaussian bandit (modified PPO)

Distribution inversely-proportional to price bids (inverse softmax)

Dynamic customer budget, with noise
Experiment 2.2
Experiment 3.1

- Market Conditions: No demand
- Bandit property tested: Fallback and recovery

- Gaussian bandit (modified PPO)
- Distribution inversely-proportional to price bids (inverse softmax)
- Dynamic customer budget, with noise
Experiment 3.2
Experiment 3.3

- Graceful Init Pull!
Experiment 4.1

- Market Conditions: Competition with deterministic agents
- Bandit property tested: Discovery of price bids of competitive agents

- Gaussian bandit (modified PPO)
- Deterministic agents (rule-based, no update)

- Distribution inversely-proportional to price bids (inverse softmax)
- Static customer budget, with noise

Environment

- Query distributor
- Traffic generator
- Agora models

Agent

- q/s
- price bid

total q/s
**Market Conditions**: Competition with stochastic agents

**Bandit property tested**: Discovery of price bids of competitive agents

- **Gaussian bandit** (modified PPO)
- **Stochastic agents** (rule-based, no update)
- **Static** customer budget, with noise

- Distribution inversely-proportional to price bids (inverse softmax)
Experiment 5.2
**Experiment 6.1**

- **Market Conditions**: Competition with Gaussian bandits
- **Bandit property tested**: Discovery of price bids of competitive agents

- Gaussian bandit (modified PPO)
- **Static** customer budget, with noise

- Distribution inversely-proportional to price bids (inverse softmax)

---

![Diagram](image)
Experiment 6.2

- Race to the bottom!
On the Expected Outcomes

- **Agents** and **Environment** form a **Game**
  - When Agents’ rewards are driven purely by query volume (x price) and
  - Environment naively distributes the queries based on the price bids, then
    **Race to the bottom is the expected outcome!**

- Different outcomes can be achieved in various ways

- The Graph protocol desired features and outcomes (selected):
  - All Indexers should have freedom with their pricing models
  - All Indexers should be able to make (some) profit
  - Conclusion: **The Gateway** should implement the anti-domination rules
    **So happens it already does!**
Experiment 7.1

- **Indexer Selection Algorithm (ISA)** (wrapped one of components of The Graph's Gateway)
- **Gaussian bandit** (modified PPO)
- **Static** customer budget, with noise
- **Market Conditions**: Competition with Gaussian bandits
- **Bandit property tested**: Discovery of price bids of competitive agents

**Environment**

- **Agent**
  - q/s
  - price bids

- **ISA (wrapped)**
  - Agora models

- **Traffic generator**
  - total q/s
Experiment 7.2
Experiment 7.3

- **Market Conditions**: Competition with Gaussian bandits (more agents)
Experiment 7.4

- **Market Conditions**: Competition with Gaussian bandits (different init conditions)
Experiment 7.5

- **Market Conditions**: Competition with Gaussian bandits (weaker/stronger update rules)
AutoAgora in production 1

- Deployed AutoAgora on our Graph indexer agent (semiotic-indexer.eth) [graphscan]

Gaussian moves right!

Mean goes up!
AutoAgora in production 2

Variance goes down!

Gaussian gets narrower!
AutoAgora In Production 3

Reward goes up!
AutoAgora In Production 4

Revenue goes up!
Summary

- Agent-based Modelling (ABM) for cryptoeconomics
  - Focus on Dynamic Pricing applied to Automated Price Discovery
  - Focus on agents using reinforcement learning for revenue maximization

- We have shown how to use ABM for
  - Testing the properties of the protocol
  - Discovering (and designing!) the outcomes of the game

- Finally, we have deployed AutoAgora in a real Graph indexer!
Summary 2

- **Feature works**
  - Better update rules/policies
  - Agents with multiple rewards (taking QoS into account)
  - Modelling and putting consumer agents into play
  - Redesigning the game (e.g. perfect information)

- **AutoAgora resources**
  - A. Asseman (2022): “Automated Query Pricing in The Graph” [blogpost]
  - A. Asseman (2022), Special Graph Hack Episode: “Automated Cost Modeling” [youtube]
  - AutoAgora GitHub Repository (open-source!) [link]
Oct 12th, 10:00am, Matt: “Overview of AMM mechanisms”

Oct 12th, 11:30am, Seve: “A SNARK’s Tale: A Story of Building SNARK Solutions on Mainnet”

We are hiring | full time & interns | remote & Los Altos, California

- **AI Researchers** (RL, DL)
- **Cryptographers** (SNARKs, ZK proofs, FHE)
- **Developers** (general | web3, Rust, Solidity)
- **DevOps Engs** (infrastructure, CI, real-time services, AWS)
- **Data Scientists** (general | arbitrage strategy and capture)
- **BizDev Officers** (general | web3)

contact@semiotic.ai
Thank you!

Tomasz Kornuta (Tom)
tomasz@semiotic.ai
@TomaszKornuta
Agent-Based Simulation (in Protocol Economics)
Dynamic Pricing In Competitive Markets

- **Dynamic pricing** happens where the **price is flexible**
  - **Flexibility**: price can be based on demand, supply, competition price, and/or subsidiary product prices
  - **Personalization**: Price may change from customer-to-customer based on their purchase habits

- Protocols like RAI and Filecoin already rely on Dynamic Pricing
- **Reinforcement Learning** is often cited as a **future option** for automated decision-making in web3 protocols
Independent simulations

- Fast: good for rapid prototyping, unit testing etc.
- Huge “sim2real gap”: deployment is *the actual testing*
Multi-fidelity Simulation 1

- Reduction of the sim2real gap
- Modeling off- & on-chain with varying “realism”
Multi-fidelity Simulation 2

- Component mockup
- Simulation speed
- Simulation fidelity
- Complexity
- “Sim2real” gap

Testing new ideas (off-chain tests)
Verification (partial off- & on-chain tests)
E2E testing (full off- & on-chain simulation)