Optimizing a Decentralized Data Market Protocol via Agent-Based Simulation

Rei Chiang[1], Bharath Ramsundar[2], Tarun Chitra[1], and John Morrow[1]

[1]Gauntlet Networks
[2]Computable

Traditional markets allow for resource distribution that can increase participant utility. Blockchain systems can provide ways to guarantee that the share of that increase is distributed equitably in the market, ensuring that there are not structural hurdles to growth. However, it is not always obvious how to design a protocol that achieves these goals, and even less so in permission-less systems. We analyze a decentralized data marketplace protocol that allows users to control data and capture fair value for it. We use a platform for agent-based modeling that allows for making statistical predictions about protocol behavior and market outcomes. In this paper, we describe how we create a simulation environment that allows us to optimize fees and other key protocol parameters in this decentralized data market.

The architecture of the simulation platform was influenced by the design of algorithmic trading and reinforcement learning systems. In this framework, we define the utility or reward function for individual agents and allow them to optimize their behavior in a fully autonomous fashion. We then optimize the network-level parameters by observe the emergent properties of the agents' behavior in aggregate. Our results include the identification of phase transitions in market outcomes, Gini coefficients of agent wealth distribution under different conditions, and changes to pricing mechanisms that resulted in a more effective market design.

Introduction

Building decentralized systems presents new challenges that are not often seen in traditional software development. In particular, the adage of "move fast and break things" is no longer a viable strategy as we have seen time and time again how even a single critical security vulnerability can be very difficult for a project to recover from. Also, the success of these protocols depends on the design of economic incentives that encourage balanced participation and growth between different types of users in order to create a multi-sided market that will ultimately accrue value. These incentive structures can be difficult to modify once deployed since there is no centralized, governing authority.

We have built an agent-based simulation platform to help developers validate their protocol designs, understand trade-offs between different parameterizations, and ensure that applications are resilient to attacks by bad actors.

This paper presents a case study of using the simulation platform to analyze a decentralized data market protocol. It highlights some of the lessons learned by the simulation and protocol teams from early 2019 as we worked together to design a custom scenario to optimize parameters for the protocol's Reserve contract on the simulation platform. By leveraging the tools and simulation results, the protocol team was then able to rapidly iterate and verify a more robust design for a couple of the incentive mechanisms in the initial implementation of the contract. Simulation analysis was always a part of the protocol team's plan to test and refine the token economics, and they chose to use our simulation platform to achieve that goal.

Note: The protocol team has just released a new whitepaper, which contains a number of mechanism and terminology changes. The simulation model described here was built on the original version. We will call out some of the differences below to avoid confusion. The updated contracts can be found on the team's Github.

The Decentralized Data Market Protocol

The protocol team aims to create a decentralized data market that will incentivize the curation of high-quality datasets at scale, while providing trust and transparency around data privacy and usage. The protocol aims to be flexible enough to accommodate data markets for different industry applications. For example, certain markets may have the property that a handful of large players own the majority of the relevant data, while the success of other data markets may depend on many individual users making contributions over time. Each dataset has a unique token associated with it to incentivize curation and growth, and the participants, or "agents" in the network are grouped into the following roles:

- **Buyer** - Represents the demand for the data. Pays to query from the dataset
- **Datatrust** - Provides compute resources for executing queries
- **Maker** - Supplies data to data markets for sale in the form of listings
- **Patron** - Provides initial capital to incentivize makers to contribute data -

The mechanism for determining whether a listing is valid is similar to a Token-Curated Registry. The dynamics of TCR voting can resemble other blockchain systems such as Proof-of-Stake consensus and decentralized oracles. However, for the rest of this post we will focus on macro features that are more specific to the Data Market Protocol.

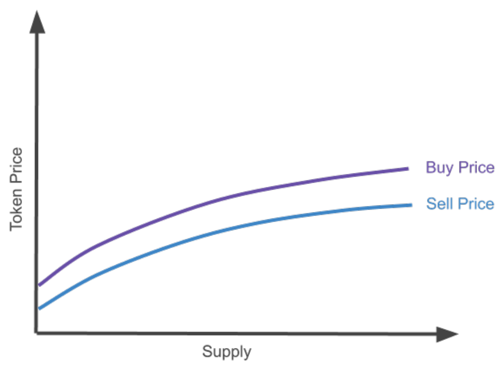

Fig. 1. A Bonding Curve determines prices for buying and selling a token as a function of token supply.

Bonding Curve.

There is a bootstrapping problem since each dataset has its own token, as these tokens will have limited liquidity and be hard to value initially. A Bonding Curve is a contract that determines token price when buying/selling and acts as an automated market-maker for the token to encourage early participation. The diagram below illustrates how one might use a bonding curve to issue tokens. Note that there are separate buy/sell curves, where the sell price is lower than the buy price, to discourage short-term price manipulation while allowing for organic price discovery and liquidity since market participants can agree to trade tokens at any price between two curves.
The protocol uses a bonding curve for issuing tokens when patrons deposit cryptocurrencies to the reserve, and when tokens are issued to makers for providing data. For the rest of this post, we will use the term "Network token" to denote the tokens deposited to the reserve. In practice, Network tokens could be either be a token that is native to the Data Market protocol and shared across multiple data markets, or it could be a token that is native to the underlying blockchain (e.g. ETH). We will show that the shape and parameterization of the bonding curves has a significant impact on network growth.
For this analysis we use a linear bonding curve-https://www.overleaf.com/project/5d422a29f7d162572282d9c1 defined as follows:

$$support\_price = conversion\_rate + conversion\_slope \cdot reserve \tag{1}$$

$$withdraw\_price = \frac{reserve}{total\_supply} \tag{2}$$

where $support\_price$ is the price to buy, $withdraw\_price$ is t he price to sell, $reserve$ is the total value of currency locked in the bonding curve, and $total\_supply$ is the total number of Market tokens issued by the bonding curve.

Agent Model.

Buyer - We model demand to query data in aggregate, rather than as individual agents. This demand is realized in the form of payment for queries during each simulation time step. The demand is a function of the number of listings in the dataset, with a predefined upper bound (market size) and bounded growth rate per time step. The fee to query data (in Network tokens) is split as follows:

- **datatrust_rate_network** - Percentage of query fee for datatrust agents, in Network tokens
- **reserve_fee_network** - Percentage that goes to the Market contract's reserve, which increases both the support and withdraw prices, in Network Tokens.
- **maker_fee_network** - Percentage that goes to listing owners in Network tokens. In our model, this portion is split equally across all listings. If the listing is no longer owned by a Maker (see **convert_listing** below), then its share goes to the Market reserve. Note: This parameter has since been removed from the protocol, and maker payments are now done exclusively with Market tokens.
- **maker_fee_market** - Percentage that goes to listing owners in Market tokens. This component is split equally across all listings, and is locked up (see **convert_listing** below) to encourage long-term maker participation. This portion of the fee is paid by minting (creating) Market tokens at the bonding curve's support price, and is inflationary because it increases the supply of Market tokens without increasing the reserve.

Datatrust - These agents will process queries if fee that they receive is greater than their marginal cost of doing the computation.
Maker - We assume an upper bound on the number of makers (i.e. there are only so many participants that have high quality data to contribute to the dataset), and that the number of makers that will want to list their data is a function of the expected utility of being listed. We also assume that each maker can have at most one listing. Makers can take the following actions:

- **list** - apply to be listed in the dataset to receive a share of query revenue. The expected utility of being listed can be modeled as:

$$\mathbb{E}[utility] = DF_i \cdot \frac{(maker\_fee\_network + maker\_fee\_market * demand_t)}{num\_listings_t} + listed\_reward \cdot divest\_price_t - listing\_cost \tag{3}$$

where $demand_t$ is query revenue at time t, $num\_listings_t$ is the number of listings, DF is the discount factor that the agent applies to future earnings, $listed\_reward$ is number of market tokens received for getting listed, $divest\_price_t$ is the sell price given by the bonding curve, and $listing\_cost$ is the overhead or anti-sybil cost associated with creating a listing.

- **convert_listing** - A maker can transfer ownership of the listing to the Market contract in order to unlock the Market tokens in the listing (from $listed\_reward$ and $maker\_fee\_market$), but forgoing future query revenue. A maker will do this if the return on investment (ROI) on the value of the locked tokens falls below the agent's $convert\_roi$. The maker's ROI can be estimated by taking the value of dividends received (including Market tokens) over an observation window, dividing by the market value of Market tokens locked, and converting to an annualized return. Note: This function is no longer a part of the protocol. The current protocol no longer requires makers to surrender ownership of listings to withdraw.

Patron - Can buy or sell Market tokens via the bonding curve:

- **support** - Buy tokens at $support\_price$ given by the bonding curve. A patron will buy tokens if the risk-adjusted return on the $withdraw\_price$ is greater than the agent's $support\_roi$ threshold and if the expected time to break even (since $support\_price_t > withdraw\_price_t$) is less than the agent's $support\_breakeven\_time$.

- **withdraw** - Sell tokens at $withdraw\_price$ given by the bonding curve. A patron will sell tokens if the risk-adjusted return on the $withdraw\_price$ is less than the agent's $withdraw\_roi$ threshold.

Simulation Environment.

Our simulation platform is built around an agent-based model where users can specify the initial conditions of the network, including distributions of agent behaviors and agent-specific parameters. We generally follow the Byzantine-Altruistic-Rational (BAR) model for describing agent behavior, though this is not a requirement. Each time step of the simulation involves the following:

- Update environment state variables
- Introduce new agents
- Evaluate agent utility functions for each action being considered and execute those that have highest utility

For the analysis below, we make the following assumptions:

- The maximum buyer demand (i.e. market size) is 100,000 Network tokens per year
- To improve performance, the maximum number of makers is 25, each maker can only have one listing, and $listed\_reward$ is 3 Network tokens
- There are 5 initial patrons that contribute 1000 Network tokens each
- 80% of the makers are rational (behavior/utility described above), and the remaining 20% are altruistic (will not try to **convert_listing**)
- We run each simulation scenario for 5 years, or 1825 time-steps

Findings

Maker Compensation.

We ran the simulation over different values of $maker\_fee\_network$ and $maker\_fee\_market$ to explore the impact of different fee structures on Maker behavior. The $reserve\_fee$ is held constant, and the remainder of the query revenue is paid to datatrust agents. Recall that $maker\_fee\_market$ is the share of query revenue that are paid in Market tokens and locked up to encourage long-term participation. In the heat-map below, each square represents an independent run of the simulation and the color represents the percentage of the total query revenue that is captured by the rational makers.
This is shown in Figure 2. A few observations:

- When $maker\_fee\_network + maker\_fee\_market$ is too high, the network fails to create value because $datatrust\_fee$ is not high enough to cover the marginal cost of running computations.
- When $maker\_fee\_network + maker\_fee\_market$ is too low, the network also fails to generate significant value because makers quickly convert their listings or are not sufficiently incentivized to **list** in the first place.
- Under these parameterizations, the $maker\_fee\_market$ does not seem to be long-term beneficial for makers. When $maker\_fee\_network$ is held constant, increasing $maker\_fee\_market$ has little effect on increasing maker utility, and utility can actually decrease a bit in some cases.

The result that increasing $maker\_fee\_market$ generally does not benefit makers much in the long-run is definitely a bit counter-intuitive. Upon closer inspection, we realized that it could make sense for the following reasons:

- Once buyer demand plateaus, $maker\_fee\_market$ continues to increase the value of tokens locked up in the listing, while revenue remains constant. This means that the listing's ROI continues to fall and at some point the maker will convert the listing, trading future revenue for liquidity.

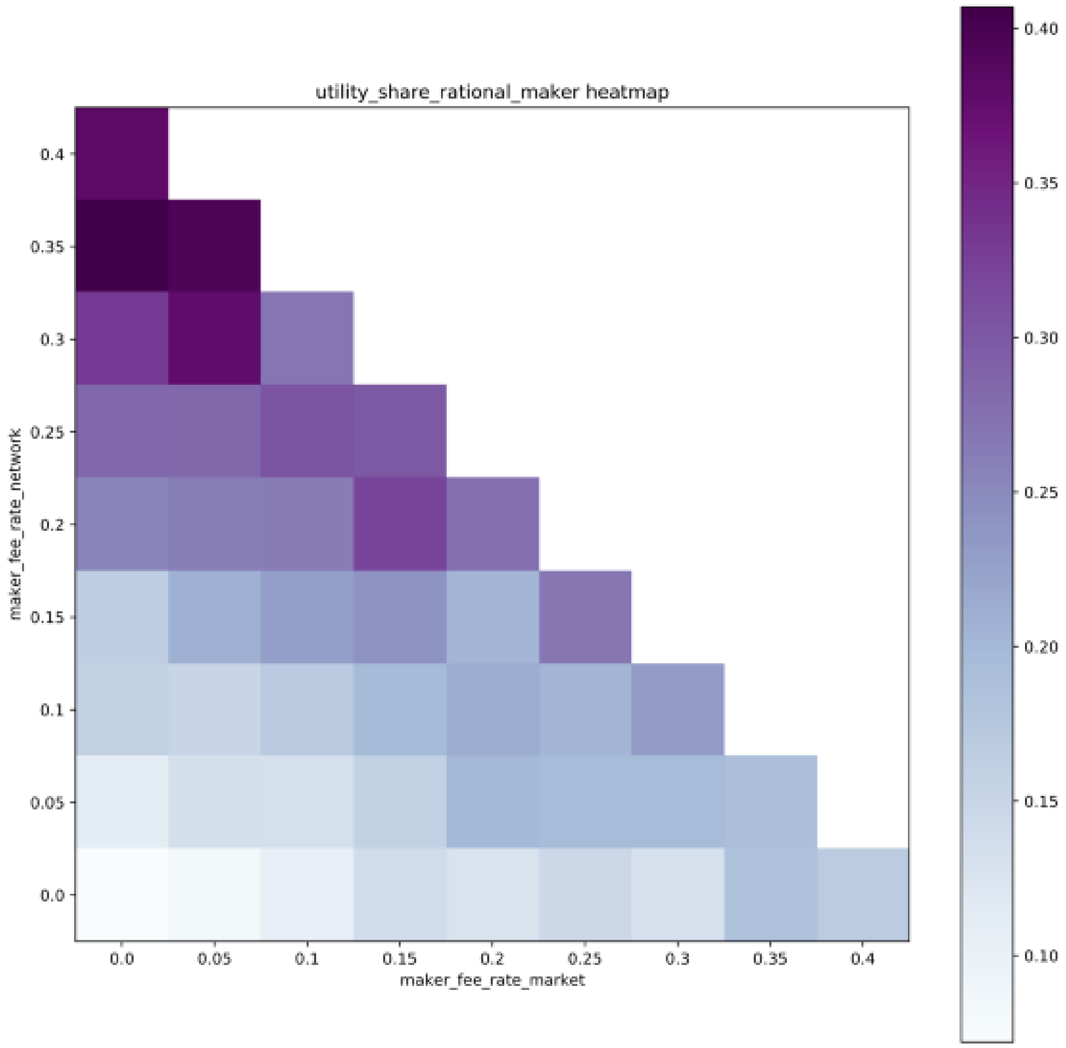

Fig. 2. Rational Maker Utility Share Heatmap shows the percentage of the total query revenue that is captured by the rational makers

- $maker\_fee\_market$ is inflationary (increases supply of Market token but not the reserve). The percentage of reserve ownership for each maker does not increase much later in the simulation when makers own most of the Market tokens (vs initial patrons).
- Most bonding curve parameterizations result in an support price that is substantially greater than the withdraw price once the supply is large, so the conversion is inefficient, resulting in the overall contribution of $maker\_fee\_market$ to maker ROI being small relative to $maker\_fee\_network$.

Out of these three factors, we suspected that the shape of the bonding curve was likely to have the largest impact, so we decided to dig a bit further.

In the original formulation of the bonding curve, the $support\_price$ did not track the $withdraw\_price$ very closely so we decided to update the definition of $support\_price$. We now define the curve as:

$$\begin{aligned} support\_price = conversion\_rate + \\ support\_multiplier * \frac{reserve}{max(1, total\_supply)} \end{aligned} \quad (4)$$

$$withdraw\_price = \frac{reserve}{total\_supply} \quad (5)$$

We re-ran the above analysis and got the following results:
Now you can see in Figure 3 it appears that the $maker\_fee\_market$ parameter is actually useful! Increasing $maker\_fee\_market$ generally increases the utility of the rational makers, and we see that for a given maker fee allocation ($maker\_fee\_network + maker\_fee\_market$), it is better to split the fee between network and market components rather than paying purely in either Network or Market tokens. Note that there is still a trade-off between makers and patrons when paying in Market tokens since $maker\_fee\_market$ dilutes initial patrons.

Bonding Curve Analysis.

The success of the protocol depends on initial patrons contributing a large amount of capital to the reserve to incentivize makers to list data. Patrons ultimately profit if the $withdraw\_price$ exceeds the initial $invest\_price$. We ran the simulation over the new bonding curve's parameters $conversion\_rate$ and $support\_multiplier$. In the heat-map below, each square represents an independent run of the simulation and the color represents the percentage of total query revenue that is captured by the initial patrons.
A graph of this is shown in Figure 4. A few observations:

- High $support\_multiplier$ benefits initial patrons because it limits dilution when minting Market tokens
- When $conversion\_rate$ is too high, the network does not generate enough value within the time constraints we set for initial patrons to break even on their original deposit
- When $conversion\_rate$ and $support\_multiplier$ are too low, the network fails to generate any value because there is not enough incentive for Makers to join in the first place

Conclusion

The simulation work with the protocol team prompted many updates to the initial protocol design, including an improved bonding curve and simplification of the Maker payment and $convert\_listing$ interface. Along the way, we have also identified key parameters to optimize, trade-offs associated with different parameterizations, and areas where a different mechanism altogether may achieve the desired result more efficiently. Hopefully this was a convincing example of how simulations can guide the protocol design process!
Within the context of a single data market, simulation allows us to analyze mechanism design as a distributed constrained optimization problem. More broadly, the framework can generate a reference set of parameters and initial conditions that are catered to serving data markets with all sorts of different properties and industry applications. The goal is to design a system that maximizes buyer demand, while maintaining equitable incentives between datatrust providers, makers, and patrons in a way that is statistically verifiable.

Economic incentives are of paramount importance for the long-term security and success of blockchain applications. Trying to reason about incentive mechanism design without simulation is tricky as the emergent properties of a network can be difficult to predict from local changes, and people often resort to making overly simplistic assumption about user behavior in order to get tractable results or closed-form solutions. Agent-based simulation can be a valuable tool for helping developers to validate security assumptions, and understand how value is created for network participants over time.

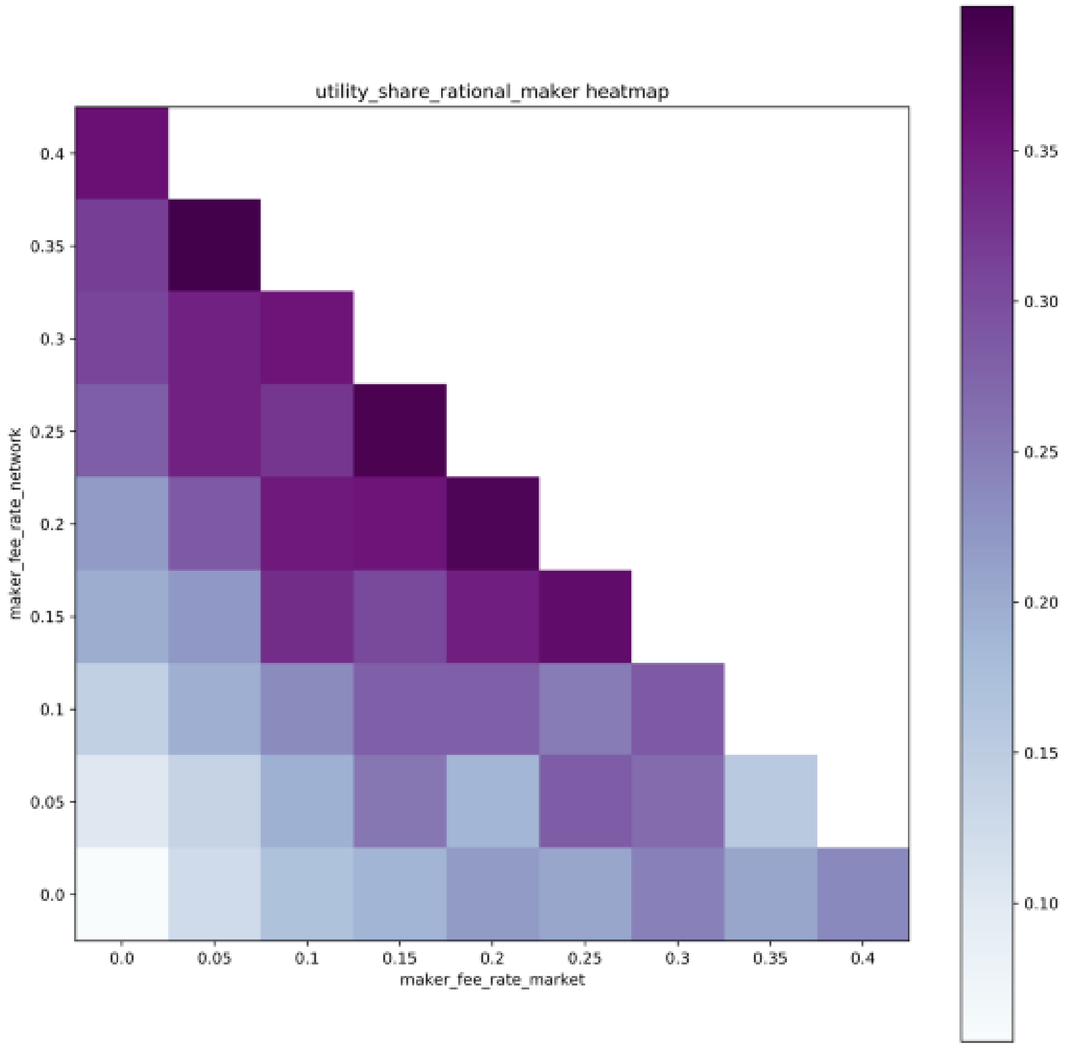

Fig. 3. Rational Maker Utility Share Heatmap shows the percentage of the total query revenue that is captured by the rational makers

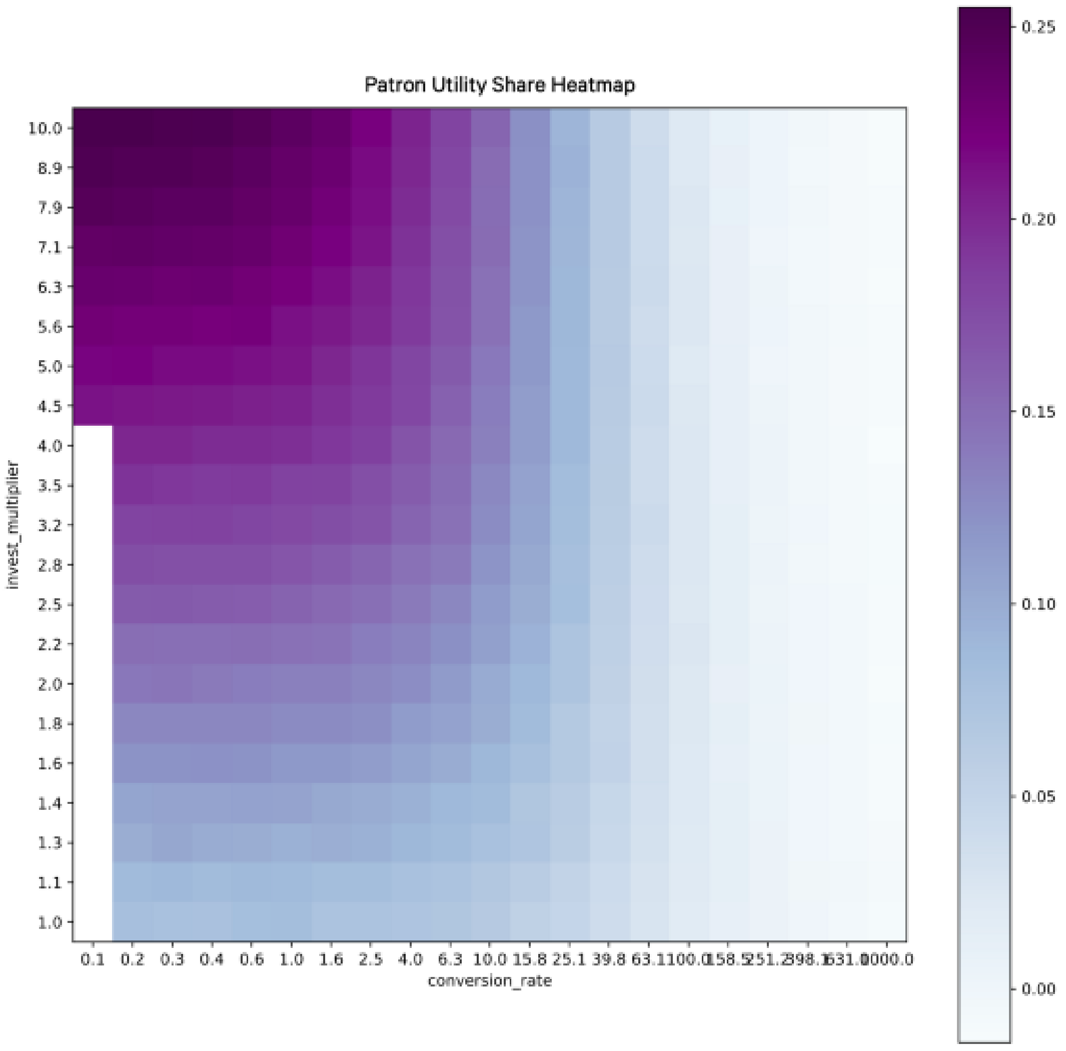

Fig. 4. Patrion Utility Share Heatmap shows the percentage of the total query revenue that is captured by the initial patrons