Shapley Value-based Approach for Redistributing Revenue of Matchmaking of Private Transactions in Blockchains

Presented at AAMAS'25 as Extended Abstract

Rasheed M
IIIT Hyderabad
Hyderabad, India, India
mohammad.ahmed@research.iiit.ac.in

Parth Desai
IIIT Hyderabad
Hyderabad, India, India
parth.desai@research.iiit.ac.in

Sujit Gujar
IIIT Hyderabad
Hyderabad, India, India
sujit.gujar@iiit.ac.in

ABSTRACT

Background: In the context of blockchain, MEV refers to the maximum value that can be extracted from block production through the inclusion, exclusion, or reordering of transactions. Searchers often participate in order flow auctions (OFAs) to obtain exclusive rights to private transactions offered by entities known as *matchmakers*, also known as order flow providers (OFPs). The utility gained by searchers stems from private transactions; thus, redistributing revenue generated by such auctions to the transaction creators is desirable. **Objectives and Research Questions:** We aim to maximize searchers' social welfare while fairly distributing auction revenue among transaction creators. **Methods:** Using cooperative game theory, we formalize the notion of fair revenue redistribution in matchmaking and present its potential possibilities and impossibilities. Precisely, we define a characteristic form game, referred to as `RST-Game`, for the transaction creators. We propose to redistribute the revenue using the Shapley value of `RST-Game`. **Results:** We show that the Shapley value computation in `RST-Game` with additive valuations is polytime; however, in the case of a single-minded setting, the problem incurs runtime $2^{O(\sqrt{n})}$, where n is the number of transactions; therefore, approximating the Shapley value is necessary. We show that the sampling-based approach provides a good approximation for the Shapley value with $O(n^2)$ samples in practice, where n is the number of transactions. **Conclusions:** We show that computing the redistribution vector in single-minded settings can incur subexponential complexity. We propose a randomized algorithm for computing the Shapley value in `RST-Game` and empirically demonstrate its efficacy.

KEYWORDS

Blockchain, MEV, Matchmaking, Shapley Value

ACM Reference Format:

Rasheed M, Parth Desai, and Sujit Gujar. 2025. Shapley Value-based Approach for Redistributing Revenue of Matchmaking of Private Transactions in Blockchains: Presented at AAMAS'25 as Extended Abstract. In *Appears at the 8th Games, Agents, and Incentives Workshop (GAIW-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 25 pages.

Appears at the 8th Games, Agents, and Incentives Workshop (GAIW-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Armstrong, Curry, Hosseini, Mattei, Tsang, Wąs (Chairs), May 2026, Paphos, Cyprus.

1 INTRODUCTION

A *blockchain* is a distributed ledger of digital transactions maintained by a network of computers (or nodes) that reach consensus using one of a variety of mechanisms, such as *Proof-of-Work* (PoW) [34] or *Proof-of-Stake* (PoS) [6]. [19] defines *Maximal Extractable Value* (MEV) as "the maximum value that can be extracted from block production in excess of the standard block reward and gas fees by including, excluding, and changing the order of transactions in a block." As MEV extraction becomes increasingly vital for sustaining the Ethereum blockchain's economic ecosystem, developments in the Ethereum landscape [8], such as its transition to Proof-of-Stake, have reformed how transactions are processed. This shift introduced the concept of *proposer-builder separation* (PBS), in which *builders* are responsible for gathering transactions from the network, assembling them into blocks, and submitting them to the *proposer*, who has the authority to publish them on the blockchain. Builders compete in an auction, known as the *PBS auction*, to get their block included in the current slot [26]. Proposers are heavily invested in the staking process and prioritize earning through staking rewards rather than generating profits through block building and MEV extraction. *Searchers* are individual or institutional entities that seek to capitalize on profitable opportunities arising from various factors such as market inefficiencies during periods of high volatility [3]. They use techniques such as front-running, back-running, and sandwiching [9, 43] to extract MEV.

Although MEV has negative externalities such as poor execution price to transaction creators due to front-running and sandwiching [18], consensus security risks [36], centralization [2, 24], it enhances overall revenue in the system, supports sustainable post-block rewards, and contributes to the overall health of DeFi (Decentralized Finance) activities [5, 14, 20]. As these added advantages come at the cost of the price paid by transaction creators, in this work, we focus on sharing some of the value generated by searchers with transaction creators, whom we refer to as *users*.

Typically, searchers find MEV opportunities by analyzing unconfirmed transactions in the public mempool. Searchers strategically create a bundle of transactions to earn profit and bid (based on their potential profit) for block space with block builders. Unconfirmed transactions in a public mempool are accessible to all blockchain participants, leading to intense competition among searchers [28]. Hence, searchers look for private transactions (private order flows) not available in the public mempool to increase their chances of winning [24].

Such MEV-rich private transactions are made available by *order flow providers* (OFPs), which typically include wallet service providers. OFPs collect user intents, process them into transactions, and are supposed to put the transactions into the public mempool. However, to improve efficient MEV extraction, OFPs auction transactions to searchers who can extract more MEV in *order flow auctions* (OFAs) [42]. OFPs typically charge users for their wallet services. Thus, to enhance the user experience, OFPs share the revenue generated from the OFAs with them, leading to the mechanism of *matchmaking* [46].

Matchmaking is a mechanism for redistributing revenue generated by order flow auctions to users [46]. A *matchmaker* in a blockchain is an OFP with access to users' private transactions. Figure 1 shows that as of 2025, private transactions made up more than 30% of all smart contract transactions on Ethereum and almost make up to 55% of the block value [42]. Although multiple authors have argued for such redistribution [21, 22], designing a matchmaking mechanism remains an open problem. This paper addresses how a matchmaker should fairly redistribute the revenue generated through an OFA among users. Some transactions add more value to the system than others; thus, sharing the revenue uniformly is unfair. Thus, sharing should be proportional to how much value they add to the system.

The challenge in achieving redistribution arises from the limited information available to the matchmaker, i.e., searchers' bids and their preferred transactions. Without any inherent valuation of transactions for the matchmaker, it must compensate users fairly. In this paper, we show how it can achieve fair redistribution in such settings using cooperative game theory. First, we introduce a `RST-Game` (Definition 4), a characteristic form game amongst users with an appropriately defined characteristic function. We propose that the user receive compensation proportional to their Shapley Value [38]. Next, we analyze it for two important types of valuations of the searchers: (i) *additive* and (ii) *single-minded*. Additive valuations refer to a type of searcher preference in which the total value of a searcher's bundle of transactions is the sum of the valuations of its individual transactions. A single-minded valuation refers to the type of searcher preference in which the searcher does not prefer (an allocation of) any strict subset of his interested bundle of transactions, but does not mind (an allocation of) a superset of his interested bundle.

In general, the complexity of computing the Shapley value of a game is in EXP [15]. However, the Shapley value computation can be in polynomial time under restrictive settings, where the game's structure or representation is particularly simple or admits efficient algorithms [32, 40]. In the additive valuation case, we show that the proposed characteristic function of `RST-Game` is additive and hence, its Shapley value computation is in polynomial time [13]. Instead of relying on such algorithms, we provide a simple yet very useful closed-form solution for users' Shapley Values. For the single-minded case, we present a construction in which determining the Shapley value of `RST-Game` requires subexponential computation (in the number of transactions). Hence, we conjecture that the

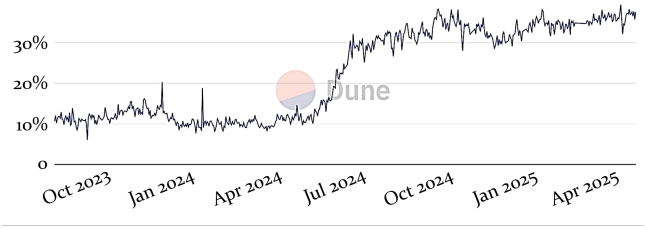


Figure 1: Private Transactions on Ethereum [16]

complexity of Shapley value computation of `RST-Game` with single-minded searchers is SUBEXP [1]. Since an analytical solution to the `RST-Game` under single-minded settings is infeasible when the number of transactions or searchers is large, we resort to approximating the Shapley values via sampling. We show that a simple sampling algorithm, **R**andomized **S**hapl**Y** **P**rocedure `RSYP` (Algorithm 2) approximates the Shapley value of transactions. Sampling-based algorithms have been used in many real-world settings for computing the Shapley value. We contribute to determining the sample complexity required to get PAC guarantees for `RST-Game`. Further, we show that $O(n^2)$ samples of subsets of transactions provide a good approximation to the Shapley value for practical purposes.

Contributions. In summary, the following are our contributions: (i) we introduce a cooperative game, `RST-Game`, that uses information of searcher-market to define the value of coalitions of transactions, (ii) when the searcher valuations are additive, we provide a closed-form solution for the Shapley value of users in the `RST-Game`, which is polynomial-time computable, (iii) when the searchers are single-minded bidders, we show that computing Shapley value of users in the `RST-Game` is possibly SUBEXP, (iv) we obtain the PAC guarantees for `RST-Game`, and (v) we empirically show that `RSYP` with $O(n^2)$ samples well approximates Shapley value of users for practical purposes.

We believe our results provide valuable insights for the practical deployment of matchmaking.

Related Work. Prior works on MEV redistribution include modeling the MEV setting as a dynamical system with a fraction of MEV going to the miner as a dynamical variable updated with every time step [10]. The miners and builders are assumed to be a single entity, with the rest of the MEV returned to users. However, [10] does not discuss MEV distribution among the users. [31] explores rebates in the context of liquidity providers in constant function market makers and discusses the auction between searchers and builders with the assumption of a perfect MEV oracle that can compute the MEV extracted given the state of the blockchain and a new block of transactions. Our work, in contrast, does not assume the existence of such an oracle. Similarly, [48] proposes a MEV redistribution between users and liquidity providers. Our model does not pertain to liquidity providers but rather to the revenue of the matchmaker. [11] proposes a dynamical system in which agents' lending and staking portfolios within a system are influenced by an externally selected parameter that determines the proportion of

[1] Of the two commonly used definitions of SUBEXP, we use the following: SUBEXP(n) = $2^{o(n)}$

MEV extracted from a block allocated to staking. This work focuses on balancing the redistribution of MEV profits to staking, which differs from ours.

2 PRELIMINARIES

Smart contracts on blockchains, such as Ethereum, have led to the rise of Decentralized Finance (DeFi). In such systems, the strategic reordering of transactions can generate additional value beyond user transaction fees, known as *maximal extractable value* (MEV). In leader-based blockchain protocols, the next *round* proposer is known at the end of the current round.

Searchers are entities that actively look for bundling transactions to generate MEV and bid with the proposer or builder (in the case of *proposer-builder separation* in Ethereum [26]) to write their preferred order. Searchers collect transactions from *order flow providers* (OFPs) to increase their profits. Let $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ be a set of transactions available at an OFP, each created by a single user. We treat transaction t_i and its creator interchangeably as user t_i. Let $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$ be the set of searchers interested in transactions $\mathcal{T}$. Let $v_{s_i} : 2^{\mathcal{T}} \to \mathbb{R}_+$ be the valuation function of the searcher s_i.

OFPs leverage an auction mechanism to allocate transactions to searchers who value them most. Typically, such an auction is known as an *order flow auction* (OFA) [28]. Let the payment by the searcher s_i be p_{s_i} and the total revenue $\mathcal{R} = \sum_i p_{s_i}$.

Order Flow Auction (OFA). OFA is combinatorial, with searchers competing for private transactions. Valuations for different transactions and bundles may have complex relationships, which are captured via valuation functions. The OFP, in practice, can implement a first-price auction; however, as is standard in theory, assume that a truthful auction is implemented [1, 44, 45]. In truthful auctions, rational searchers would bid truthfully; hence, we work with v_{s_i}s. In real-world implementation, all the proposed things in this paper work with bids.

Often, searchers utilize individual transactions or a bundle of transactions [17]. The former can be seen as searchers with *additive valuations* and the latter as searchers with *single-minded valuations*.

Definition 1 (Additive Valuation Function). *A valuation function v is called* additive *if $\forall B \subseteq \mathcal{T}, v(B) = \sum_{t_j \in B} v(t_j)$, where $v(t_j)$ is the value of transaction t_j.*

The auction setting in which all searchers have additive valuations is called the *additive setting*. In additive settings, the most popular truthful auction, the VCG auction [12, 23, 41], is as follows. The optimal allocation A is to give each transaction to that searcher who values it the most, and its payment would be the second-highest valuation (for the transaction).

To extract MEV, searchers often require transactions in their bundles to be executed atomically; hence, they only value a bundle if it is received in its entirety. Such searchers are called *single-minded* searchers. A single-minded searcher s_i values only one specific set or bundle of transactions, say B_{s_i}, and has no interest in any other proper subset of B_{s_i}. More formally,

Definition 2 (Single-minded Valuation Function). *A valuation function v is called* single-minded *if there exists a bundle of transactions B and a value $\bar{v} \in \mathbb{R}^+$ such that $v(B') = \bar{v}$ for all $B' \supseteq B$, and $v(B') = 0$ for all other B'.*

An auction setting in which all searchers have single-minded valuations is called a *single-minded* setting. The VCG auction in a single-minded setting requires finding bundles that maximize social welfare while ensuring that the optimal allocation bundles are mutually disjoint. Finding an optimal allocation or an optimal allocation that is better than $m^{\frac{1}{2}-\epsilon}$th fraction of optimal welfare is known to be *NP-hard* [4]. Thus, we look for approximate mechanisms for such auctions. Due to DeFi markets' high volatility, we resort to greedy approaches such as the one proposed in [29] to determine the winning bids in minimum time, ensuring truthful bidding, zero loss to searchers, and producing a $\sqrt{m}$-approximation of the optimal social welfare. The rationale behind choosing this algorithm is that it balances the value a bidder brings with the size of their bundle. We refer to this algorithm as *ICA-SM*. Algorithm 1 shows the greedy allocation of transactions using ICA-SM. We introduce matchmaking in the following section.

Algorithm 1 ICA-SM [29]

Input : Bids $\{(B_{s_i}, v_{s_i}(B_{s_i}))\}_{i=1}^{n}$
EF $\leftarrow \mathcal{S}$ sorted by $v_{s_i}(B_{s_i})/\sqrt{|B_{s_i}|}$
$W \leftarrow \emptyset$.
for $k \in EF$ **do**
  if $B_{s_k} \cap (\bigcup_{s_j \in W} B_{s_j}) = \emptyset$ **then**
    $W \leftarrow W \cup \{B_{s_k}\}$.
  end if
end for
Payments : For each $s_i \in W$, $p_{s_i} = v_{s_i}(B_{s_i})/\sqrt{|B_{s_j}|/|B_{s_i}|}$, where $j \in \{i+1, \ldots, n\}$ is the smallest index such that $B_{s_i} \cap B_{s_j} \neq \emptyset$, and for all $l < j, l \neq i$, $B_{s_l} \cap B_{s_j} = \emptyset$ (if no such s_j exists then $p_{s_i} = 0$).
Output : Set of Winners W

2.1 Matchmaking

As OFP charges users for using its services, redistributing $\mathcal{R}$ back to the users supports better retention and long-term engagement. The allocation of resources and the redistribution of $\mathcal{R}$ is known as *matchmaking* [37], and an OFP is known as *matchmaker* (MM). matchmaking executes as follows: MM announces the auction along with information on private transactions. Searchers join the auction, submit their bids, and select their preferred bundle of transactions. MM determines the winning bids, allocates transactions to respective searchers, and collects payments. Since the utility to searchers is due to private transactions, MM shares the auction revenue among users via rebates. Let the rebate to user t_j be r_j. Typically, we assume budget-balanced redistribution, and hence $\sum r_j = \mathcal{R}$. Let $\gamma_j = \frac{r_j}{\mathcal{R}}$ and $\Gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n)$. We refer to Γ as a *redistribution vector*. Figure 2 describes the matchmaking process. More formally, matchmaking is defined as follows:

Definition 3 (Matchmaking). *Matchmaking is a function $\mathcal{M}$ which takes, $\mathcal{T}, \mathcal{S}, (v_{s_i})_{s_i \in \mathcal{S}}$ as inputs and outputs $(A, (p_{s_i})_{s_i \in \mathcal{S}}, \Gamma)$*

where A represents the transaction allocation to searchers, p_{s_i} represent the payment of the searcher s_i, and Γ is redistribution vector.

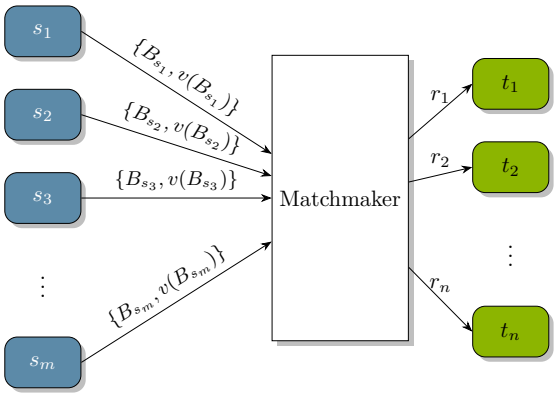


Figure 2: Matchmaking of Searchers and Users

In this work, A, $(p_{s_i})_{s_i \in \mathcal{S}}$ are according to a truthful mechanism. Thus, MM's goal is to design an appropriate Γ. To this end, we leverage the Shapley value concept from cooperative game theory. The exact bidding structure and the matchmaking process have not been discussed here due to space constraints. For more details on these aspects of matchmaking, refer to Appendix B.1.

2.2 Cooperative Game Theory

Cooperative game theory analyzes scenarios where players, or agents, can form coalitions to achieve collective goals. A *characteristic form game* (N, v) is (i) a set of players N, and (ii) $v : 2^N \to \mathbb{R}$, where $\forall S \subset N, v(S)$ represents the value that coalition S can generate. In a cooperative game, there are many ways to redistribute value among players.

Shapley value redistributes a cooperative game's total value or payoff to individual players based on their marginal contribution to every possible coalition. Shapley value is the only solution concept that satisfies all desirable properties of fair redistribution, such as efficiency, symmetry, and additivity [38]. Shapley value of a player $j \in N$ in a characteristic form game (N, v) is denoted by $\varphi_j(v)$ and can be computed using Eq. 1.

$$\varphi_j(v) = \sum_{S \subseteq \mathcal{N} \setminus j} \frac{|S|!(|\mathrm{N}| - |S| - 1)!}{|\mathrm{N}|!} \big(v(S \cup \{j\}) - v(S)\big) \quad (1)$$

The expression $v(S \cup \{j\}) - v(S)$ represents the marginal contribution of player j to coalition S. Alternatively, the Shapley value can be computed via permutations of players $\pi \in \Pi$, where Π is the set of all permutations of players in which players could join the coalition. $\pi(j)$ represents the set of players that precede j in the permutation π.

$$\varphi_j(v) = \frac{1}{|N|!} \sum_{\pi \in \Pi} [v(\pi(j) \cup \{j\}) - v(\pi(j))] \quad (2)$$

Towards this end, we explain our approach for redistribution in matchmaking, i.e., designing Γ.

3 OUR APPROACH

To leverage Shapley value, we define RST-Game withing matchmaking using $\mathcal{S}, \mathcal{T}$. We then propose Γ^{SHAP}, a redistribution vector based on the Shapley value of the users, and analyze RST-Game in additive settings as a warm-up, followed by more practically relevant, single-minded settings.

3.1 RST-Game

We model the redistribution in matchmaking as a cooperative game with $N = \mathcal{T}$. As we are interested in redistributing $\mathcal{R}$ among the users, the players are only the users, not searchers. To determine users' contributions, we aim to leverage bids and transaction preferences from the searcher-MM market to resolve the fair revenue distribution problem. We define a characteristic function $v : 2^{\mathcal{T}} \to \mathbb{R}_+$ that assigns a value to each coalition. $\forall\, T \subseteq \mathcal{T}$, $v(T)$ is the revenue generated, when only transactions in T are auctioned to searchers $\mathcal{S}^T$, where $\mathcal{S}^T \subseteq \mathcal{S}$ is the set of searchers who are interested in any of the transactions in T. $\mathcal{R}^T = \sum_{s_i \in \mathcal{S}^T} p_{s_i}$, thus, $v(T) = \mathcal{R}^T$. Formally, we define the characteristic form game in Def. 4.

Definition 4 (RST-Game). RST-Game *is a cooperative game $(\mathcal{T}, v)$ where $\mathcal{T}$ are the set of transactions being auctioned off by OFP and $v : 2^{\mathcal{T}} \to \mathbb{R}$ is the characteristic function such that $\forall T \subseteq \mathcal{T}, v(T) = \mathcal{R}^T$, where $\mathcal{R}^T = \sum_{s_i \in \mathcal{S}^T} p_{s_i}$ and $\mathcal{S}^T = \{s_i \in \mathcal{S} : s_i$ is interested in any of the transactions in $T\}$.*

revenue Redistribution. Using the aforementioned characteristic function, the matchmaker can compute the significance of any transaction as the externality it imposes on the system. However, given the very combinatorial (bundling) nature of the auction, we believe that the average marginal value addition of a transaction t_j to all possible coalitions of $\mathcal{T} \setminus \{t_j\}$ better captures its significance. This naturally leads us to the user's Shapley value, φ_j. There are specific challenges in designing Γ with $r_j \propto \varphi_j$.

Since the nature of the auction is combinatorial, v is not necessarily monotonic, which means that adding a transaction can reduce the revenue generated and therefore negative marginal utility. (We show such non-monotonicity through example in B.5) Non-monotonicity in turn may lead to φ being negative. We are interested in sharing revenue with users rather than charging them. So, we propose redistribution vector as Γ^{SHAP} as follows

$$\gamma_j^{\mathsf{SHAP}} = \frac{\max(0, \varphi_j)}{\sum_{t_l \in \mathcal{T}} \max(0, \varphi_l)} \quad (3)$$

We first discuss the easier additive setting.

3.2 Shapley Value of RST-Game with Additive Settings

When v is additive, we have $v(T_1 \cup T_2) = v(T_1) + v(T_2) \;\; \forall T_1, T_2 \subset \mathcal{T}$ and $T_1 \cap T_2 = \phi$. If v is additive, the Shapley value computation is in polynomial time [13]. Note that the additive setting we refer here concerns the valuation function v_{s_i} but not v. However, we show that when v_{s_i}s are additive, v also becomes additive in RST-Game,

(Proposition 1). Hence, existing polynomial-time algorithms [13] can compute the Shapley value efficiently.

VCG auction for additive settings turns out to be n independent second-price auctions with VCG payments. For each transaction, $t_j \in \mathcal{T}$, the matchmaker determines a searcher with the highest bid for t_j as the winner. The winner pays the amount of the second-highest bid. This leads to the following proposition.

Proposition 1. *In additive settings,* RST-Game *admits v to be additive.*

Proof Sketch. The proof is simple and intuitive. As the auction reduces to n independent auctions and the revenue sums up across all transactions, v becomes additive. We defer the complete proof to Appendix A. □

Instead of relying on algorithms for additive characteristic functions, we show that for RST-Game in additive settings, the γ_js admit a simple closed form that can be computed in $O(n)$. We compute users' Shapley values using the permutation method. Let Π denote all the possible permutations of transactions. Let L be the number of transactions allocated[2]. For each permutation $\pi \in \Pi$, $\pi(t_j)$ denotes set of transactions present before t_j in π. For any $T \subseteq \mathcal{T}, mc_v(t_j, T)$ denote the marginal contruibution of t_j to T. Formally, $mc_v(t_j, T) = v(T \cup \{t_j\}) - v(T) = \mathcal{R}^{T\cup\{t_j\}} - \mathcal{R}^T$. We compute the Shapley value of t_j as the average of the marginal contributions of t_j. That is, the average of $v(\pi(t_j) \cup \{t_j\}) - v(\pi(t_j))$ $\forall \pi \in \Pi$.

$$\varphi_{t_j}(v) = \frac{1}{n!} \sum_{\pi\in\Pi} v(\pi(t_j) \cup \{t_j\}) - v(\pi(t_j)) \tag{4}$$

Theorem 1. *The Shapley value of user t_j is $\varphi_{t_j}(v) = \frac{(n-1)}{n} v_{sh}(t_j)$. Thus, the computation of Γ^{SHAP} in* RST-Game *under additive settings has time complexity $O(n)$.*

Proof. The marginal contribution of t_j to any $T \subseteq \mathcal{T}$ is given by $mc_v(t_j, T) = \mathcal{R}^{T\cup\{t_j\}} - \mathcal{R}^T$.

Since the searchers' valuations are additive and they bid for individual transactions, MM reduces the auction to a set of independent VCG auctions; the payment for each transaction is the second-highest bid received for that transaction. Let $v_{sh}^{t_j}$ be the second-highest bid received for transaction t_j.

$$\mathcal{R}^{T\cup\{t_j\}} = \textstyle\sum_{t_l\in T} v_{sh}^{t_l} + v_{sh}^{t_j}; \quad \mathcal{R}^T = \sum_{t_k\in T} v_{sh}^{t_k}$$

Since, $\forall T \subseteq T^*, mc_v(t_j, T) = v_{sh}$, Eq. 4 is rewritten as:

$$\varphi_{t_j}(v) = \tfrac{1}{n!} \textstyle\sum_{\pi\in\Pi} mc_v(t_j, \pi(t_j)) = \frac{(n-1)}{n} v_{sh}(t_j)$$

Hence, the redistribution fraction is $\Gamma_{t_j}^{\mathsf{SHAP}} = \frac{v_{sh}(t_j)}{\sum_{t_j\in\mathcal{T}} v_{sh}(t_j)}$. Since there can be at most n winning transactions, the complexity of computing the Shapley value is $O(n)$. □

We now discuss the single-minded setting.

[2]It might be possible for a particular transaction, no searcher is interested

3.3 Shapley Value of RST-Game for Single-Minded Settings

Each searcher $s_i \in \mathcal{S}$ submits only a single subset $\mathcal{B}_{s_i} \subseteq \mathcal{T}$ in bid $\{\mathcal{B}_{s_i}, v_{s_i}\}$, where s_i's valuation v_{s_i} is single-minded. M reduces this auction to a combinatorial auction with single-minded bidders.

Example 1. *Consider* RST-Game *with $S = \{s_1, s_2, s_3\}$ and $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$ transactions. Let the bundle-bid pair submitted by searchers be $(B_1, v_1) = (\{1, 2\}, 10)$, $(B_2, v_2) = (\{3, 4\}, 9)$, and $(B_3, v_3) = (\{2, 4\}, 8)$.*

In this case, s_1 gets t_1, t_2 and s_2 gets t_3, t_4 and their payments are: $p_{s_1} = (\frac{8}{\sqrt{2}}) * \sqrt{2} = 8$ and $p_{s_2} = (\frac{8}{\sqrt{2}}) * \sqrt{2} = 8$
The total revenue $\mathcal{R} = 16$ and the individual revenue distribution fractions are $\gamma_{t_1} = \gamma_{t_3} = 0.154$, $\gamma_{t_2} = \gamma_{t_4} = 0.346$.

Therefore, $\Gamma^{\mathsf{SHAP}} = (0.154, 0.346, 0.154, 0.346)$

In general, marginal values are essential for computing the Shapley value. We show that the number of unique values of the marginal contribution in RST-Game can be subexponential (Theorem 2). Towards this, we define an *unique marginal contribution* (UMC) set.

$$UMC_v = \{v(T \cup \{t_j\}) - v(T), \forall T \subseteq \mathcal{T}, \forall t_j \in T\}$$

Theorem 2. *Let $UMC_v = \{v(T \cup \{t_j\}) - v(T), \forall T \subseteq \mathcal{T}, \forall t_j \in T\}$ be the set of unique marginal contributions obtained in the Shapley value computation of* RST-Game *under single-minded searcher valuations. $\exists$ instances such that $|UMC_v|$ is $\Omega(2^{\sqrt{n}})$.*

Proof Sketch. Let there be $n \geq 16$ transactions and $m = \lfloor 2\sqrt{n} \rfloor$ searchers. Partition the searchers in $\mathcal{S}_0 = \{1, 2, \ldots, \sqrt{n}\}, \mathcal{S}_1 = \{\sqrt{n}+1, \sqrt{n}+2, \ldots, 2\sqrt{n}\}$. Consider the following matrix arrangement of transactions $E = [a_{i,j}]_{\forall i,j\in[\lfloor\sqrt{n}\rfloor]}$ where $a_{i,j} = t_{\sqrt{n}*(i-1)+j}$

$$E = \begin{bmatrix} t_1 & t_2 & t_3 & \cdots & t_{\sqrt{n}} \\ t_{\sqrt{n}+1} & t_{\sqrt{n}+2} & t_{\sqrt{n}+3} & \cdots & t_{2\sqrt{n}} \\ t_{2\sqrt{n}+1} & t_{2\sqrt{n}+2} & t_{2\sqrt{n}+3} & \cdots & t_{3\sqrt{n}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{n-\sqrt{n}+1} & t_{n-\sqrt{n}+2} & t_{n-\sqrt{n}+3} & \cdots & t_n \end{bmatrix}$$

Let $\mathcal{B}_0 = \{B_1^0, B_2^0, \ldots, B_{\sqrt{n}}^0\}$ denote the bundles that $\mathcal{S}_0$ are interested in and $\mathcal{B}_1 = \{B_1^1, B_2^1, \ldots, B_{\sqrt{n}}^1\}$ for $\mathcal{S}_1$. Bundles in $\mathcal{B}_0$ are formed by selecting transactions from each row in E, and Bundles in $\mathcal{B}_1$ are formed by selecting one transaction from each row in a circular-shift manner. Specifically,

$$\mathcal{B}_0 = \left\{ \begin{array}{c} \{t_1, t_2, \ldots, t_{\sqrt{n}}\}, \\ \{t_{\sqrt{n}+1}, t_{\sqrt{n}+2}, \ldots, t_{2\sqrt{n}}\}, \\ \ldots, \\ \{t_{n-\sqrt{n}+1}, t_{n-\sqrt{n}+2}, \ldots, t_n\} \end{array} \right\}$$

and

$$\mathcal{B}_1 = \left\{ \begin{array}{c} \{t_1, t_{\sqrt{n}+2}, t_{2\sqrt{n}+3} \ldots, t_n\}, \\ \{t_{\sqrt{n}+1}, t_{2\sqrt{n}+2}, \ldots, t_{n-1}, t_{\sqrt{n}}\}, \\ \ldots, \\ \{t_{n-\sqrt{n}+1}, t_2, t_{\sqrt{n}+3} \ldots, t_{n-\sqrt{n}}\} \end{array} \right\}$$

Let searcher valuations be sufficiently far such that

$$v_i = a^{2\sqrt{n}-2i+1}, v_{\sqrt{n}+i} = a^{2\sqrt{n}-2i}, \text{where } a \geq 3$$

For the instance above, we compute marginal contributions for each transaction with the following subset, $\{T \subseteq \mathcal{T} : T = \bigcup_{x \in D} B_x^0 \cup B_x^1$ where $D \in 2^{[\sqrt{n}]}\}$. corresponding to every bundle in $\mathcal{B}_0$, there is a unique transaction which belongs to T iff $\mathcal{B}_0$ is selected. Thus, among all non-empty selections of bundles, we consider $2^{\sqrt{n}} - 1$ unique subsets. It can be shown that each of these unique sets of transactions can produce a unique marginal contribution. More detailed proof is provided in Appendix A. □

Games in which Shapley value computation is polynomial-time often compute the unique marginal contributions and the frequency with which they occur. Both of these steps must take polynomial time. In addition, such games have a more well-defined structure that allows for optimizing the computation of the Shapley value. RST-Game allows searchers to propose arbitrary bundles, and any polytime algorithm must work for all such arbitrary bundles. With these two insights, we conjecture the following:

Conjecture 1. *In single-minded settings of* RST-Game *Shapley value computation of users can be SUBEXP in the number of users, n.*

Why we believe the above conjecture? We empirically verify Theorem 2 and observe that variation in the number of marginal contributions with an increasing number of transactions is sub-exponential (Appendix C.2). One possibility appears to solve via Unanimity games [38]. The following subsection discusses why it fails to get a polynomial-time algorithm. Also, there are some similarities between RST-Game in single-minded settings and weighted voting games whose Shapley value computation is known to be #P-Complete [30]. Such a reduction is elusive at this point, and we leave it for future work.

3.4 Unanimity Games and Shapley Value

RST-Game in single-minded settings and Unanimity games are closely related. For unanimity games, the Shapley value can be computed in polynomial time [38]. Hence, we explore it for a polynomial-time algorithm via unanimity games. A neat analysis shows that the Shapley value computation still requires sub-exponential computations. Intuitively, we need to solve $2^{\sqrt{n}}$ unanimity games to solve RST-Game.

Consider an unanimity game $U_S = (\mathcal{T}, \omega_S)$ as,

$$\forall T \subseteq \mathcal{T}, \omega_S(T) = \begin{cases} 1 & S \subseteq T \\ 0 & \text{Otherwise} \end{cases} \tag{5}$$

The Shapley value of $t_j \in \mathcal{T}$ in U_S is given by $\varphi_{t_j} = \frac{1}{|S|}$. The set of all unianimity functions $W = \{\omega_S | S \subseteq T\}$ forms a basis for the characteristic function $v : 2^{\mathcal{T}} \to \mathbb{R}$ of RST-Game, i.e., $\forall C \subseteq T \setminus \phi, v(C) = \sum_{T \in 2^{\mathcal{T}} \setminus \phi} \Delta_T w_T(C)$, where Δ_T is referred to as *Harsanyi dividend* [25]. Based on the unanimity games, the Shapley value can be written as

$$\varphi_{t_j} = \sum_{T \in 2^{\mathcal{T}} \setminus \phi,\ t_j \in T} \frac{\Delta_T}{|T|} \tag{6}$$

With the construction provided in the proof of Theorem 2, the number of unique Harsanyi dividends is sub-exponential. Therefore, the Shapley value computation of RST-Game via Unanimity games would be sub-exponential. Refer to Appendix E for more details on Harsayni dividends.

To this end, we propose to use a sampling-based approach to approximate the Shapley value of RST-Game. Such an approach has performed remarkably well in many settings, e.g, [7, 33, 35, 47].

3.5 Approximating Shapley Value

The complexity of Shapley value computation can go sub-exponential due to the game's underlying structure, as shown in our construction. The occurrence of such structures is typically rare as the real world closely follows some distribution [3]. As only a few bundles($\ll$ the total number of all possible bundles) are being vied, $|UMC_v|$ remains $O(n)$. We show that the permutation-sampling-based approach approximates the exact Shapley values of RST-Game, and hence Γ^{SHAP}. We refer to this as RSYP (Algorithm 2). From a set of all permutations of transactions Π, we sample k different permutations. Let $\bar{\Pi}_k$ denote these permutations. For each transaction $t_j \in A$, the approximate Shapley value $\tilde{\varphi}_j$ is computed as marginal contribution of t_j to each $\pi \in \bar{\Pi}_k$, averaged over k. The approximate Shapley value $\tilde{\varphi}_{t_j}(v) = \frac{1}{k} \sum_{\pi \in \Pi_k} v(\pi(j) \cup \{j\}) - v(\pi(j))$ and thus $\gamma_{t_j}^{RSYP} = \frac{\max(\tilde{\varphi}_{t_j}(v),\, 0)}{\sum_{j \in [n]} \max(\tilde{\varphi}_{t_j}(v),\, 0)}$. We empirically show, for $k = O(n^2)$, $\forall t_j \in \mathcal{T}, \gamma_{t_j}^{\text{RSYP}}$ approximates $\gamma_{t_j}^{\text{SHAP}}(v)$.

Algorithm 2 RSYP

```
Input : Π_k, n, k
for j = 1 to n do
    MC_sum = 0
    for π ∈ Π̄_k do
        MC = v(π(j) ∪ {j}) − v(π(j))
        MC_sum += MC
    end for
    φ̃_{t_j}(v) = max(MC_sum,0) / k
end for
for j = 1 to n do
    Γ^{RSYP}_{t_j} = φ̃_{t_j}(v) / Σ_{j∈[n]} φ̃_{t_j}(v)
end for
Output :( Γ^{RSYP}_{t_j} )_{j∈[n]}
```

3.5.1 PAC Analysis. The computation of the Shapley value of any transaction t_j is essentially a sum of weighted marginal contributions. A marginal contribution of t_j to $T \subseteq \mathcal{T}$ in RST-Game is the difference in revenues. Thus, the marginal contribution in φ_j computation is at most the maximum revenue and at least the minimum revenue. Thus, $\forall\ mc \in UMC_v, mc \leq \bar{R}$, where $\bar{R} = \sum_{s_i \in \mathcal{S}} v_{s_i}$. Using Hoeffding's inequality[27], we provide an upper bound on the approximation of Shapley values in RST-Game using RSYP.

Theorem 3. *In single-minded settings of* RST-Game, *let* $\tilde{\varphi}_{t_j}$ *be the estimate of the Shapley value* φ_{t_j} *obtained using* RSYP *with*

[3] We often see some transactions being more lucrative to most of the searchers and occasionally, a few transactions being relatively highly valued by only a few (specialized) searchers [17]

k uniformly sampled permutations. Suppose the marginal contributions are bounded as $X_i^j \in [-\bar{R}, \bar{R}]$. *If* $k \geq \frac{2\bar{R}^2}{\epsilon^2} \ln \frac{2}{\delta}$, *then* $\Pr\left(\left|\tilde{\varphi}_{t_j} - \varphi_{t_j}\right| \geq \epsilon\right) \leq \delta$.

In the absence of real-world matchmaking data, we empirically validate the efficacy of RSYP on synthetically generated datasets.

4 EXPERIMENTAL ANALYSIS

4.1 Setup

We compare γ^{RSYP} of RST-Game in single-minded settings with γ^{SHAP} for smaller instances.[4] i.e., $(n \leq 8, m \leq 14)$ on 10K randomly generated instances, each with varying searcher bids and transaction bundles. We consider the following distributions:
(i) D_1: $v_i \sim \mathcal{U}(0, c)$ (ii) D_2: $v_i \sim \mathcal{N}(c, 1)$
(iii) D_3: $v_i \sim \text{EXP}(c/2)$ (iv) D_4: $v_i \sim \text{Triangular}(0, c/2, c)$

4.2 Results

We explain our results for D_1, defer D_2, D_3, D_4 results to Appendix D. Figures 3a and 3b show the behavior of γ^{RSYP} and γ^{SHAP} for a randomly selected user by varying n and m, where μ represents the mean redistribution fraction. It is clear, $\Gamma_{t_j}^{\text{RSYP}} \to \Gamma_{t_j}^{\text{SHAP}}$ in RSYP for $k = O(n^2)$. The graphs are similar for all the users.

5 MANIPULATIONS AND PRIVACY LOSS

We briefly discuss strategic manipulations by searchers and users, as well as privacy implications, in the context of RST-Game.

Searchers: The existing matchmaking algorithms are susceptible to manipulation by searchers who create multiple identities (*Sybil Attacks*). The introduction of RST-Game does not add more manipulations or attacks from searchers over existing matchmaking.

Users: If users have multiple transactions and merging them will only increase the combined value, searchers will capture them, and thus, users need not act towards them. Further, if users split transactions, it can cause a decrease in (i) overall revenue and (ii) the combined Shapley value of split transactions and it may result in reduced rewards for the user. RST-Game is sybil-proof against users.

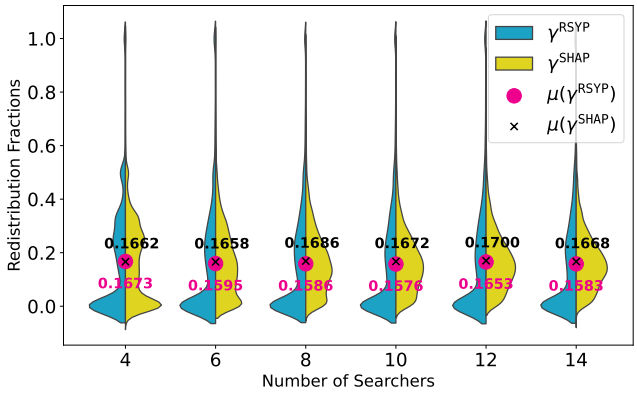
Privacy Loss: Our approach does not incur additional privacy loss. Hence, privacy guarantees the matchmaker offers hold in RST-Game too.

6 CONCLUSION

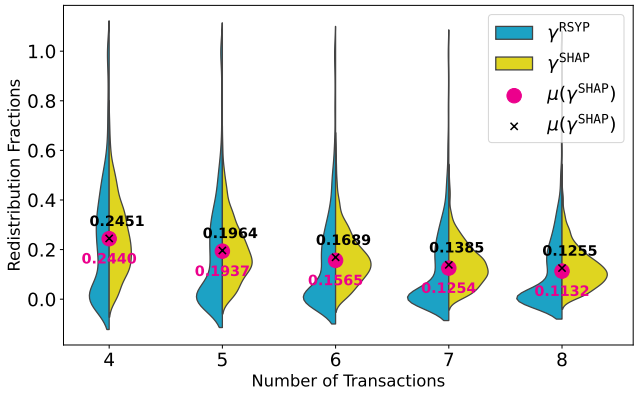
In this work, we propose a redistribution mechanism for matchmaking in blockchains. We model the redistribution as a cooperative game, RST-Game with transactions as players and propose redistributing revenue based on each transaction's Shapley value. We show that, in single-minded settings where each bidder is interested in only one specific bundle of items and has a fixed value for that bundle, computing the redistribution vector can be subexponential. We show that the sampling-based approach provides a good approximation for the Shapley value with $O(n^2)$ samples, where n is the number of transactions.

We aim to explore fair revenue redistribution in matchmaking to better reflect searchers' general valuations as part of our future work.

[4]Since computing the exact Shaley value for large instances is computationally intensive, we show the efficacy of the sampling approach for smaller instances.



(a) Distribution of $\gamma^{\text{RSYP}}, \gamma^{\text{SHAP}}$ vs n for $m = 6, c = 1$ for a randomly selected user



(b) Distribution of $\gamma^{\text{RSYP}}, \gamma^{\text{SHAP}}$ vs m for $n = 6, c = 1$ for a randomly selected user

Figure 3: RSYP Performance Analysis

REFERENCES

[1] Gagan Aggarwal, Ashish Goel, and Rajeev Motwani. 2006. Truthful auctions for pricing search keywords. In *Proceedings of the 7th ACM Conference on Electronic Commerce* (Ann Arbor, Michigan, USA) *(EC '06)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/1134707.1134708

[2] Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. 2025. Centralization in Block-Building and Proposer-Builder Separation. In *Financial Cryptography and Data Security: 28th International Conference, FC 2024, Willemstad, Curaçao, March 4–8, 2024, Revised Selected Papers, Part I* (Willemstad, Curaçao). Springer-Verlag, Berlin, Heidelberg, 331–349. https://doi.org/10.1007/978-3-031-78676-1_19

[3] Jan Arvid Berg, Robin Fritsch, Lioba Heimbach, and Roger Wattenhofer. 2022. An empirical study of market inefficiencies in Uniswap and SushiSwap. In *International Conference on Financial Cryptography and Data Security*. Springer, 238–249.

[4] Liad Blumrosen and Noam Nisan. 2007. *Combinatorial Auctions*. Cambridge University Press, 267–300.

[5] Jonah Burian. 2024. The Future of MEV. *arXiv preprint arXiv:2404.04262* (2024).

[6] Vitalik Buterin et al. 2013. Ethereum white paper. *GitHub repository* 1 (2013), 22–23.

[7] Javier Castro, Daniel Gómez, and Juan Tejada. 2009. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research* 36, 5 (2009), 1726–1730. https://doi.org/10.1016/j.cor.2008.04.004 Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X).

[8] Yash Chaurasia, Parth Desai, Sujit Gujar, et al. 2024. MEV Ecosystem Evolution From Ethereum 1.0. *arXiv preprint arXiv:2406.13585* (2024).

[9] Tianyang Chi, Ningyu He, Xiaohui Hu, and Haoyu Wang. 2024. Remeasuring the Arbitrage and Sandwich Attacks of Maximal Extractable Value in Ethereum. *arXiv preprint arXiv:2405.17944* (2024).

[10] Georgios Chionas, Pedro Braga, Stefanos Leonardos, Carmine Ventre, Georgios Piliouras, and Piotr Krysta. 2024. Who gets the Maximal Extractable Value? A Dynamic Sharing Blockchain Mechanism. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2024)*.

[11] Tarun Chitra and Kshitij Kulkarni. 2022. Improving proof of stake economic security via MEV redistribution. In *Proceedings of the 2022 ACM CCS Workshop on Decentralized Finance and Security*. 1–7.

[12] Edward H Clarke. 1971. Multipart pricing of public goods. *Public choice* (1971), 17–33.

[13] Vincent Conitzer and Tuomas Sandholm. 2004. Computing Shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In *AAAI*, Vol. 4. 219–225.

[14] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. 2020. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE symposium on security and privacy (SP)*. IEEE, 910–927.

[15] Xiaotie Deng and Christos H Papadimitriou. 1994. On the complexity of cooperative solution concepts. *Mathematics of operations research* 19, 2 (1994), 257–266.

[16] Dune. 2024. Private Order Flow. https://dune.com/dataalways/private-order-flow. [Accessed 10-12-2024].

[17] EigenPhi. 2022. MEV Data | Eigen Phi — eigenphi.io. https://eigenphi.io/. [Accessed 1-10-2024].

[18] Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. 2020. Sok: Transparent dishonesty: front-running attacks on blockchain. In *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*. Springer, 170–189.

[19] Ethereum. 2024. Maximal extractable value (MEV) | ethereum.org — ethereum.org. https://ethereum.org/en/developers/docs/mev/. [Accessed 23-09-2024].

[20] Christof Ferreira Torres, Albin Mamuti, Ben Weintraub, Cristina Nita-Rotaru, and Shweta Shinde. 2024. Rolling in the shadows: Analyzing the extraction of mev across layer-2 rollups. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 2591–2605.

[21] Flashbots. 2023. MEV-Share: programmably private orderflow to share MEV with users — collective.flashbots.net. https://collective.flashbots.net/t/mev-share-programmably-private-orderflow-to-share-mev-with-users/1264. [Accessed 10-10-2024].

[22] Flashbots Research Proposals. 2023. FRP-30: Quantifying Shareable MEV collective.flashbots.net. https://collective.flashbots.net/t/frp-30-quantifying-shareable-mev/1618. [Accessed 10-10-2024].

[23] Theodore Groves. 1973. Incentives in teams. *Econometrica: Journal of the Econometric Society* (1973), 617–631.

[24] Tivas Gupta, Mallesh M. Pai, and Max Resnick. 2023. The Centralizing Effects of Private Order Flow on Proposer-Builder Separation. In *5th Conference on Advances in Financial Technologies (AFT 2023) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 282)*, Joseph Bonneau and S. Matthew Weinberg (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 20:1–20:15. https://doi.org/10.4230/LIPIcs.AFT.2023.20

[25] JC Harsanyi. 1959. RA bargaining model for cooperative n person games. *Contributions to the Theory of Games IV (eds. Tucker AW, and RD Luce), Princeton UP, Princeton* (1959), 325.

[26] Lioba Heimbach, Lucianna Kiffer, Christof Ferreira Torres, and Roger Wattenhofer. 2023. Ethereum's Proposer-Builder Separation: Promises and Realities. In *Proceedings of the 2023 ACM on Internet Measurement Conference* (Montreal QC, Canada) *(IMC '23)*. Association for Computing Machinery, New York, NY, USA, 406–420. https://doi.org/10.1145/3618257.3624824

[27] Wassily Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association* 58, 301 (1963), 13–30.

[28] Quintus Kilbourn. 2022. order flow, auctions and centralisation I - a warning | Flashbots Writings — writings.flashbots.net. https://writings.flashbots.net/order-flow-auctions-and-centralisation. [Accessed 02-10-2024].

[29] Daniel Lehmann, Liadan Ita Oćallaghan, and Yoav Shoham. 2002. Truth revelation in approximately efficient combinatorial auctions. *J. ACM* 49, 5 (Sept. 2002), 577–602. https://doi.org/10.1145/585265.585266

[30] Yasuko Matsui and Tomomi Matsui. 2001. NP-completeness for calculating power indices of weighted majority games. *Theoretical Computer Science* 263, 1-2 (2001), 305–310.

[31] Bruno Mazorra and Nicolás Della Penna. 2023. Towards optimal prior-free permissionless rebate mechanisms, with applications to automated market makers & combinatorial orderflow auctions. *arXiv preprint arXiv:2306.17024* (2023).

[32] Tomasz P Michalak, Karthik V Aadithya, Piotr L Szczepanski, Balaraman Ravindran, and Nicholas R Jennings. 2013. Efficient computation of the Shapley value for game-theoretic network centrality. *Journal of Artificial Intelligence Research* 46 (2013), 607–650.

[33] Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. 2022. Sampling permutations for shapley value estimation. *Journal of Machine Learning Research* 23, 43 (2022), 1–46.

[34] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. *Satoshi Nakamoto* (2008).

[35] Ramasuri Narayanam and Yadati Narahari. 2011. A Shapley Value-Based Approach to Discover Influential Nodes in Social Networks. *IEEE Transactions on Automation Science and Engineering* 8, 1 (2011), 130–147. https://doi.org/10.1109/TASE.2010.2052042

[36] Kaihua Qin, Liyi Zhou, and Arthur Gervais. 2022. Quantifying blockchain extractable value: How dark is the forest?. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 198–214.

[37] Miller Robert. 2023. MEV-Share: programmably private orderflow to share MEV with users — collective.flashbots.net. https://collective.flashbots.net/t/mev-share-programmably-private-orderflow-to-share-mev-with-users/1264. [Accessed 23-07-2024].

[38] Lloyd S. Shapley. 1952. *A Value for N-Person Games*. RAND Corporation, Santa Monica, CA. https://doi.org/10.7249/P0295

[39] shea. 2023. Announcing MEV-share beta — collective.flashbots.net. https://collective.flashbots.net/t/announcing-mev-share-beta/1650. [Accessed 25-07-2024].

[40] Tom C van der Zanden, Hans L Bodlaender, and Herbert JM Hamers. 2023. Efficiently computing the Shapley value of connectivity games in low-treewidth graphs. *Operational Research* 23, 1 (2023), 6.

[41] William Vickrey. 1961. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance* 16, 1 (1961), 8–37.

[42] Shuzheng Wang, Yue Huang, Wenqin Zhang, Yuming Huang, Xuechao Wang, and Jing Tang. 2025. Private Order Flows and Builder Bidding Dynamics: The Road to Monopoly in Ethereum's Block Building Market. In *Proceedings of the ACM on Web Conference 2025* (Sydney NSW, Australia) *(WWW '25)*. Association for Computing Machinery, New York, NY, USA, 2144–2157. https://doi.org/10.1145/3696410.3714754

[43] Ye Wang, Patrick Zuest, Yaxing Yao, Zhicong Lu, and Roger Wattenhofer. 2022. Impact and user perception of sandwich attacks in the defi ecosystem. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–15.

[44] Fei Xiao, Haijun Wang, Shuojia Guo, Xu Guan, and Baoshan Liu. 2021. Efficient and truthful multi-attribute auctions for crowdsourced delivery. *International Journal of Production Economics* 240 (2021), 108233.

[45] Yidan Xing, Zhilin Zhang, Zhenzhe Zheng, Chuan Yu, Jian Xu, Fan Wu, and Guihai Chen. 2023. Truthful auctions for automated bidding in online advertising. *arXiv preprint arXiv:2301.13020* (2023).

[46] zeroXbrock. 2022. Searching Post-Merge | Flashbots Writings — writings.flashbots.net. https://writings.flashbots.net/searching-post-merge. [Accessed 22-09-2024].

[47] Jiayao Zhang, Qiheng Sun, Jinfei Liu, Li Xiong, Jian Pei, and Kui Ren. 2023. Efficient Sampling Approaches to Shapley Value Approximation. *Proc. ACM Manag. Data* 1, 1, Article 48 (May 2023), 24 pages. https://doi.org/10.1145/3588728

[48] Mengqian Zhang, Sen Yang, and Fan Zhang. 2024. RediSwap: MEV Redistribution Mechanism for CFMMs. arXiv:2410.18434 [cs.GT] https://arxiv.org/abs/2410.18434

APPENDIX

A MAIN PROOFS

Proposition 1. *In additive settings,* RST-Game *admits ν to be additive.*

Proof. Since the searchers' valuations are additive, and they bid for individual transactions, MM reduces the auction into independent VCG auctions; the payment for each transaction is the second-highest bid received for the transaction. Let $v_{sh}(t_j)$ be the second-highest bid received for transaction t_j.

Then for any $T_1 \subseteq \mathcal{T}$, $\nu(T_1) = \mathcal{R}^{T_1} = \sum_{t_i \in T} v_{sh}^{t_i}$ Similarly, for any $T_2 \subseteq \mathcal{T}$, $\nu(T_2) = \mathcal{R}^{T_2} = \sum_{t_j \in T_2} v_{sh}^{t_j}$

For $T = T_1 \cup T_2$, $\nu(T) = \mathcal{R}^T$ which is revenue from transactions T alone. As the auction reduces to $|T|$ independent auctions and revenue is added across all transactions exactly once,

$$\begin{aligned}
\nu(T) &= \sum_{t_l \in T} t_l \\
&= \sum_{t_i \in T_1 \setminus T_2} t_i + \sum_{t_j \in T_2 \setminus T_1} t_j \\
&= \sum_{t_i \in T_1} t_i + \sum_{t_j \in T_2} t_j - \sum_{t_k \in T_1 \cap T_2} t_k \\
&\nu(T_1) + \nu(T_2) - \nu(T_1 \cap T_2)
\end{aligned}$$

Hence, ν turns out to be additive. □

Theorem 2. Let $UMC_\nu = \{\nu(T \cup \{t_j\}) - \nu(T), \forall T \subseteq \mathcal{T}, \forall t_j \in T\}$ be the set of unique marginal contributions obtained in the Shapley value computation of RST-Game under single-minded searcher valuations. $\exists$ instances such that $|UMC_\nu|$ is $\Omega(2^{\sqrt{n}})$.

Proof. Consider an instance of the RST-Game with set of transactions $\mathcal{T} = \{t_1, \dots t_n\}$. Let the set of searchers $\mathcal{S} = \{s_1, \dots, s_m\}$ be divided into two classes of searchers with equal cardinality, $\mathcal{S}_0$ and $\mathcal{S}_1$, with corresponding sets of bundles $\mathcal{B}_0$ and $\mathcal{B}_1$ satisfying the following 2 properties:

- None of the bundles in $\mathcal{B}_0$ or $\mathcal{B}_1$ intersects with any other bundle in $\mathcal{B}_0$ or $\mathcal{B}_1$, respectively, i.e., $\forall B_{s_i}, B_{s_j} \in \mathcal{B}_0, B_{s_i} \cap B_{s_j} = \phi$. In other words, bundles within $\mathcal{B}_0$ are mutually exclusive, and bundles within $\mathcal{B}_1$ are mutually exclusive.
- Each bundle in $\mathcal{B}_0$ intersects with every bundle in $\mathcal{B}_1$ and vice versa, i.e., $\forall B_{s_i} \in \mathcal{B}_0, \forall B_{s_j} \in \mathcal{B}_1, B_{s_i} \cap B_{s_j} \neq \phi$.

Consider the following construction satisfying the above properties. Let $m = \lfloor 2\sqrt{n} \rfloor$ where $n \geq 16$. Thus, $|\mathcal{B}_0| = |\mathcal{B}_1| = m/2 = \sqrt{n}$. Let the size of each bundle be $\sqrt{n}$, i.e., $|B_{s_i}| = \sqrt{n}, \forall s_i \in \mathcal{S}$. Consider matrix $E = [a_{i,j}]_{\forall i,j \in [\lfloor \sqrt{n} \rfloor]}$ where $a_{i,j} = t_{\sqrt{n}*(i-1)+j}$:

$$E = \begin{bmatrix}
t_1 & t_2 & t_3 & \cdots & t_{\sqrt{n}} \\
t_{\sqrt{n}+1} & t_{\sqrt{n}+2} & t_{\sqrt{n}+3} & \cdots & t_{2\sqrt{n}} \\
t_{2\sqrt{n}+1} & t_{2\sqrt{n}+2} & t_{2\sqrt{n}+3} & \cdots & t_{3\sqrt{n}} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
t_{n-\sqrt{n}+1} & t_{n-\sqrt{n}+2} & t_{n-\sqrt{n}+3} & \cdots & t_n
\end{bmatrix}$$

$$\mathcal{B}_0 = \left\{ \begin{array}{c}
\{t_1, t_2, \dots, t_{\sqrt{n}}\}, \\
\{t_{\sqrt{n}+1}, t_{\sqrt{n}+2}, \dots, t_{2\sqrt{n}}\}, \\
\dots, \\
\{t_{n-\sqrt{n}+1}, t_{n-\sqrt{n}+2}, \dots, t_n\}
\end{array} \right\}$$

$$\mathcal{B}_1 = \left\{ \begin{array}{c}
\{t_1, t_{\sqrt{n}+2}, t_{2\sqrt{n}+3} \dots, t_n\}, \\
\{t_{\sqrt{n}+1}, t_{2\sqrt{n}+2}, \dots, t_{n-1}, t_{\sqrt{n}}\}, \\
\dots, \\
\{t_{n-\sqrt{n}+1}, t_2, t_{\sqrt{n}+3} \dots, t_{n-\sqrt{n}}\}
\end{array} \right\}$$

Let $\mathcal{B}_0 \cup \mathcal{B}_1$ be the set of bundles submitted by searchers such that bundles in $\mathcal{B}_0$ are essentially individual rows from the matrix E and Bundles in $\mathcal{B}_1$ contain one transaction from each row and each column of matrix E [5]. The essence of this construction is that none of the bundles in $\mathcal{B}_0$ or $\mathcal{B}_1$ intersect with any other bundle in $\mathcal{B}_0$ or $\mathcal{B}_1$, respectively. It can also be shown that the intersection of a bundle from $\mathcal{B}_0$ and a bundle from $\mathcal{B}_1$ is a singleton [6].

[5] Note that $(\sqrt{n})!$ such selections of $\mathcal{B}_1$ are possible, given the same $\mathcal{B}_0$.

[6] More specifically, $B_x^1 \cap B_y^0$ is the transaction in $B_x^1 \in \mathcal{B}_1$ with the row number y in matrix E where bundle $B_y^0 \in \mathcal{B}_0$ is the bundle corresponding to row y in matrix E.

Let $B^0_1, \ldots, B^0_{\sqrt{n}}$ be the bundles of $\mathcal{B}_0$ and the corresponding bids, with slight abuse of notation, be $v^0_1, v^0_2, \ldots, v^0_{\sqrt{n}}$ such that $v^0_1 > v^0_2 > \ldots > v^0_{\sqrt{n}}$. Similarly, let $B^1_1, \ldots, B^1_{\sqrt{n}}$ be the bundles of $\mathcal{B}_1$ and the corresponding bids be $v^1_1, v^1_2, \ldots, v^1_{\sqrt{n}}$ such that $v^1_1 > v^1_2 > \ldots > v^1_{\sqrt{n}}$. Further let the searcher valuations be sufficiently far such that $v^0_{s_i} = a^{2\sqrt{n}-2i+1}, v^1_{s_i} = a^{2\sqrt{n}-2i}$, and $a \geq 3$

$$v^0_1 = a^{(2\sqrt{n}-1)} > v^1_1 = a^{(2\sqrt{n}-2)} > v^0_2 = a^{(2\sqrt{n}-3)} > \ldots > v^0_{\sqrt{n}} = a^1 > v^1_{\sqrt{n}} = a^0$$

We now show that while computing Shapley values of all transactions using ICA-SM for the above construction, we will encounter at least $2^{\sqrt{n}} - 1$ unique marginal values. The main insight in our approach is that corresponding to any of the non-empty $2^{\sqrt{n}} - 1$ combinations of bundles from $\mathcal{B}_0$, there exists a unique transaction set T and transaction t_j such that $\nu(T) - \nu(T \setminus \{t_j\})$ gives a unique marginal contribution.

Transaction selection: Let the bundles selected from $\mathcal{B}_0$ in the combination be $B^0_a, \ldots, B^0_z$. Then, we define our transaction set as $T = \bigcup_{x=a}^{x=z} B^0_x \cup B^1_x$. We note the following:

(1) Let $\{t_x\} = B^0_x \cap B^1_x$ and $\{t_y\} = B^0_y \cap B^1_y$. Then, if $B^0_x \neq B^0_y \implies t_x \neq t_y$.
We can prove this by contradiction. Suppose

$$\begin{aligned} \exists B^0_x, B^0_y | B^0_x \neq B^0_y \wedge t_x = t_y & \\ \implies & t_x \in B^0_x \wedge t_x \in B^0_y \\ \implies & t_x \in B^0_x \cap B^0_y \\ \implies & B^0_x \cap B^0_y \neq \phi \end{aligned}$$

Given that all bundles in $\mathcal{B}_0$ are mutually exclusive, this is a contradiction.

(2) Let $\{t_x\} = B^0_x \cap B^1_x$. If B^0_x is selected, $t_x \in T$.
This is trivially true because $T = \bigcup B^0_x \cup B^1_x$ for all selected bundles B^0_x.

(3) Let $\{t_x\} = B^0_x \cap B^1_x$. If B^0_x is not selected, $t_x \notin T$.
We prove this by contradiction. Let B^0_y and B^1_y represent any selected bundle. Suppose $t_x \in T$

$$\begin{aligned} \implies & (\exists B^0_y | t_x \in B^0_y) \vee (\exists B^1_y | t_x \in B^1_y) \\ \implies & (t_x \in B^0_x \wedge \exists B^0_y | t_x \in B^0_y) \vee \\ & (t_x \in B^1_x \wedge \exists B^1_y | t_x \in B^1_y) \\ \implies & (\exists B^0_y | t_x \in B^0_y \cap B^0_x) \vee (\exists B^1_y | t_x \in B^1_y \cap B^1_x) \\ \implies & (\exists B^0_y | B^0_y \cap B^0_x \neq \phi) \vee (\exists B^1_y | B^1_y \cap B^1_x \neq \phi) \end{aligned}$$

Given that all bundles within $\mathcal{B}_0$ and $\mathcal{B}_1$ are mutually exclusive, this is a contradiction.

(4) While aggregating which searchers are allotted their bundles given the set of transactions T, only the following two cases are possible:
- All allotted bundles are from $\mathcal{B}_0$ and their payments are some bids v^1_x, i.e. bid value of a bundle in $\mathcal{B}_1$.
- All allotted bundles are from $\mathcal{B}_1$ and their payments are some bids v^0_x, i.e. bid value of a bundle in $\mathcal{B}_0$.

This is true because our construction obeys the two properties mentioned above: i) mutual exclusion within $\mathcal{B}_0$ and $\mathcal{B}_1$, and, ii) non-empty intersection between any bundle from $\mathcal{B}_0$ and $\mathcal{B}_1$.

(5) The ICA-SM payment [7] corresponding to B^0_x is v^1_x.
Transactions of both B^0_x and B^1_x are present in T. Also, according to our assigned bid values, $\nexists v_y | v^0_x < v_y < v^1_x$, i.e., bid v^1_x is the highest bid smaller than bid v^0_x. Given that B^0_x and B^1_x intersect, the searcher corresponding to bundle B^0_x pays v^1_x.

(6) The ICA-SM payment corresponding to the bundle B^1_x is either 0 (if B^0_x was the bundle with the least bid among the selected bundles) or v^0_y for some bundle B^0_y, where v^0_y is the highest bid less than v^1_x.

(7) Let B^0_a be the selected bundle with the highest bid. Let B^0_w be an unselected bundle (if all bundles are selected, consider the least valued bundle B^0_z instead). Thus, $\exists t_u \in B^0_a \cap B^1_w$, when the transaction set is $\tau = T \setminus \{t_u\}$, $\nu(\tau \cup \{t_u\}) = \sum_x v^1_x$ and $\nu(\tau) = (\sum_x v^0_x) - v^0_a$, where B^0_a is the bundle with the highest bid.
Thus, the marginal contribution of coalition τ for Shapley value of t_u is $\sum_x v^1_x - \sum_x v^0_x + v^0_a$.

From (1), (2), and (3), it follows that for every bundle B^0_x, there is a unique transaction t_x in T if and only if B^0_x is selected. Thus, while considering all non-empty selections of bundles, we are considering $2^{\sqrt{n}} - 1$ unique subsets of $\mathcal{T}$. Each of these unique sets of transactions is involved in marginal contributions of some transaction t_u, as shown in (7). The marginal contributions $\sum_x v^1_x - \sum_x v^0_x + v^0_a$ are the sum of all bids with coefficients either $-1, 0$, or 1. Thus, for these bid values (powers of 3) and bundles and coefficients $-1, 0$, or 1, all possible summations, and hence the marginal contributions, are unique.

[7] As the size of all bundles in $\mathcal{B}_0$ and $\mathcal{B}_1$ is $\sqrt{n}$, ICA-SM ordering is the same as ordering of the bids – the common factor $\sqrt{n}$ cancels out. Similarly, the payments (Step 9 in Algorithm 1) $p_{s_i} = v_{s_j} / \sqrt{|B_{s_j}| / |B_{s_i}|} = v_{s_j}$ as $|B_{s_j}| = |B_{s_i}| = \sqrt{n}$

Thus, the number of unique marginal contributions: □

Theorem 2. In single-minded settings of RST-Game, let $\tilde{\varphi}_{t_j}$ be the estimate of the Shapley value φ_{t_j} obtained using RSYP with k uniformly sampled permutations. Suppose the marginal contributions are bounded as $X_i^j \in [-\bar{R}, \bar{R}]$. If $k \geq \frac{2\bar{R}^2}{\epsilon^2} \ln \frac{2}{\delta}$, then $\Pr\left(\left|\tilde{\varphi}_{t_j} - \varphi_{t_j}\right| \geq \epsilon\right) \leq \delta$.

Proof. Let Π denote the set of all permutations, and let X_i^j denote the marginal contribution of player t_j in permutation π_i. Then,

$$\varphi_{t_j} = \mathbb{E}[X_i^j] = \frac{1}{N!} \sum_{i=1}^{N!} X_i^j.$$

Let K be a set of k permutations drawn independently and uniformly at random from Π. The estimator is

$$\tilde{\varphi}_{t_j} = \frac{1}{k} \sum_{\ell \in K} X_\ell^j.$$

Thus, $\tilde{\varphi}_{t_j}$ is the empirical mean of k i.i.d. samples of X_i^j, and $\mathbb{E}[\tilde{\varphi}_{t_j}] = \varphi_{t_j}$.

Since the marginal contributions are bounded, i.e., $-\bar{R} \leq X_i^j \leq \bar{R}$, we can apply Hoeffding's inequality for bounded independent random variables. This gives:

$$\Pr\left(\tilde{\varphi}_{t_j} - \varphi_{t_j} \geq \epsilon\right) \leq \exp\left(\frac{-2k\epsilon^2}{(2\bar{R})^2}\right) = \exp\left(\frac{-k\epsilon^2}{2\bar{R}^2}\right).$$

Similarly, for the lower tail:

$$\Pr\left(\varphi_{t_j} - \tilde{\varphi}_{t_j} \geq \epsilon\right) \leq \exp\left(\frac{-k\epsilon^2}{2\bar{R}^2}\right).$$

Applying the union bound, we obtain:

$$\Pr\left(\left|\tilde{\varphi}_{t_j} - \varphi_{t_j}\right| \geq \epsilon\right) \leq 2\exp\left(\frac{-k\epsilon^2}{2\bar{R}^2}\right).$$

To ensure this probability is at most δ, it suffices that

$$2\exp\left(\frac{-k\epsilon^2}{2\bar{R}^2}\right) \leq \delta.$$

Taking logarithms and rearranging gives:

$$k \geq \frac{2\bar{R}^2}{\epsilon^2} \ln \frac{2}{\delta}.$$

This completes the proof. □

B MATCHMAKING

B.1 Matchmaking Implementation

Consider the set of transactions $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ submitted by transaction creators. Let $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$ be the set of searchers interested in participating in the sealed-bid auction with the matchmaker M. Matchmaker M collects the set of transactions $\mathcal{T}$ from transaction creators and announces the auction, along with metadata/ partial information about the transactions, to the set of all participating searchers $\mathcal{S}$. For example, rather than disclosing the entire transaction details as illustrated in the JSON below, the Matchmaker filters the data and shares only that the transaction interacted with SushiSwap and UniSwap contracts and executed a swap between different tokens.

```
            {
               "to": SushiSwap_router",
               "tokens_traded": ["DAI", "WBTC"]
            }

            {
               "to": nuniswap_v3_router,
               "tokens_traded": ["ETH", "USDC"]
            }
```

Only limited transaction data is disclosed to searchers to safeguard the user's transactions. If full transaction details were available, searchers could independently submit a bundle directly to the builder or proposer and extract MEV without compensating the user. This would make MEV extraction appear as if the user's transactions were publicly available.

Each searcher $s_i \in \mathcal{S}$ runs their strategies locally and submits a partial bundle of transactions $B_{s_i} \subseteq \mathcal{T}$ to M along with bids v_{s_i}. Below is a sample JSON structure sent by the Searcher to the Matchmaker. The bundle includes an empty string ("") o as a placeholder for any

transaction matching the given criteria. The hints field specifies the conditions the searcher is interested in. When the user's transaction runs, any transaction that calls the SushiSwap contract ("0x9f8c") or Uniswap_v3_contract ("0xab12"), and produces the event log "0x7f200" and "0x7g100" respectively, is expected to be placed within the placeholders in the same order as addresses touched. Note that, instead of a simple order-based system, hints can be made more expressive to provide more complex matching criteria.

```
{
    "txs": [ "", "0xf145hbOt5", ""]
    "blockNumbern: 1000000,
    "hints": {
        "addresses_touched": ["0x9f8c","0xab12"]
        "Logs_emitted": "0x7f200", "0x7g100"
}
```

The Matchmaker receives a partial bundle from the searcher and identifies transactions that satisfy the specified criteria. It then assembles a complete bundle, simulates each bundle locally, performs sanity checks, and discards invalid bundles. The Matchmaker determines the amount of compensation allocated to the user and attaches a validity condition to ensure the user can reclaim a portion of the transaction's rewards. The complete bundle is then sent to the builder/ proposer. Upon successful execution of the block, the rewards are shared with the users. Figure 4 illustrates the process of matchmaking.

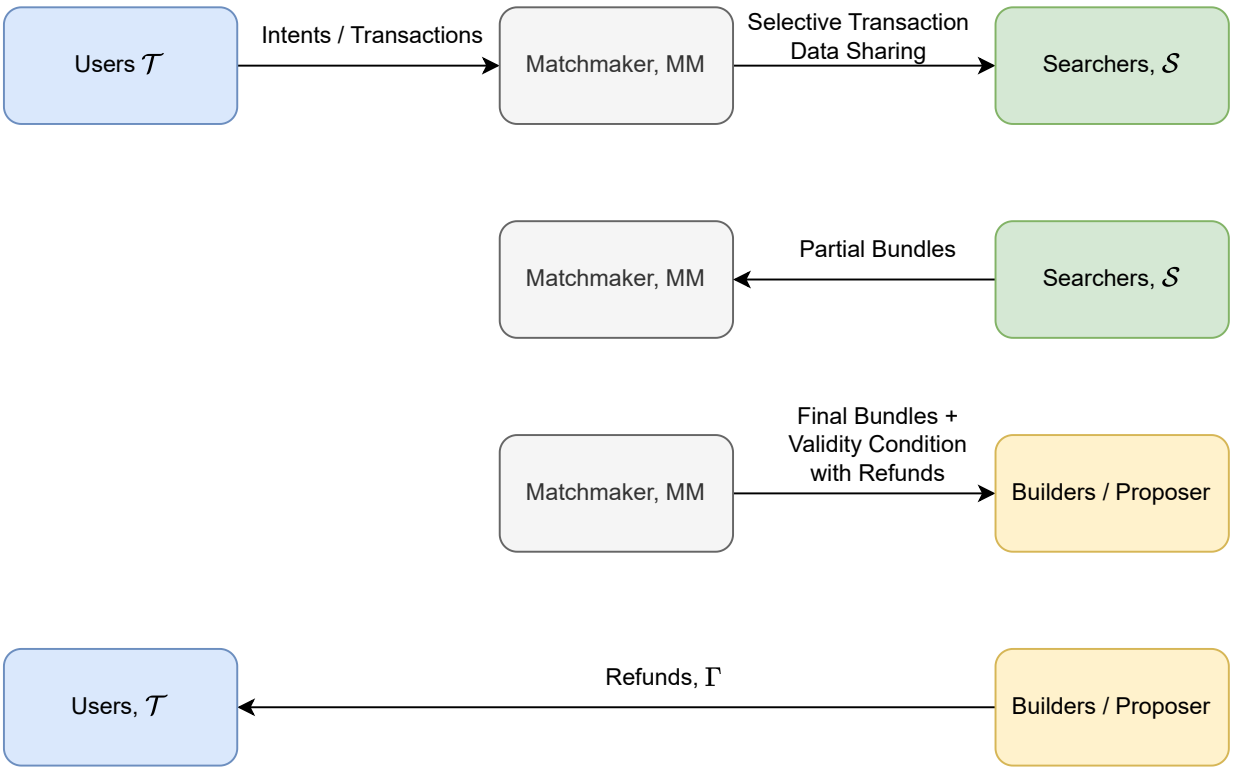


Figure 4: Matchmaking Real Implementation Architecture [39]

B.2 Matchmaking vs OFA

A regular OFA only ensures that private transactions are sold off to the highest bidder but doesn't ensure the timely execution of the transaction. Consider the scenario where the searcher chooses not to send the transactions to a block builder and retain them for the future. While an OFA can introduce a penalty for the searcher that misses the slot, where an OFA waits for a deterministic amount of time and forwards the transaction by itself or auctions it again, it is hard to identify if the transactions did not make it to execution solely due to searcher and not due to network latency, high competition for blockspace. Hence, potentially enforcing cooperation is not easy. Further, the

auctioneer can increase the revenue by selling multiple transactions simultaneously as a bundle to searchers rather than individually [37]. Matchmaking is a recent introduction in the MEV world, where a *matchmaker* aggregates transactions from transaction creators, exposes transaction data to searchers, collects bids along with partial bundles, creates final bundles, and bids to builders for inclusion [39]. Searchers bid on different subsets of transactions, and the matchmaker computes to find the optimal allocation. This requires optimizing for the best bundles in a finite amount of time and redistributing the revenue back to the transaction creators to compensate for the value they create.

B.3 Desirable Properties of Matchmaking

We now discuss the desirable properties of matchmaking. Matchmaking involves allocating transactions to searchers and compensating transaction creators. Mathematically, let:

- v_i be the true valuation of searcher i for the allocation $A \in \mathcal{A}$ (set of possible allocations).
- b_i be the bid of searcher i.
- $A(v)$ be the allocation rule based on all bids $v = (v_1, v_2, \ldots, v_n)$.
- $p_i(v)$ be the payment rule for searcher i.
- $r_i(v)$ be the reward for transaction creator i

Some of the generally desired properties of auction mechanisms are:

1. Incentive Compatibility (Truthfulness): A mechanism is incentive-compatible (or truthful) if each searcher's best strategy is to bid their true valuation of the good, regardless of what others are doing. In such a mechanism, searchers have no incentive to misrepresent their preferences. The mechanism is incentive-compatible for searchers if:

$$v_i(A(v_i, b_{-i})) - p_i(v_i, b_{-i}) \geq v_i(A(b'_i, v_{-i})) - p_i(b'_i, v_{-i}), \tag{7}$$

for all b'_i, where b_{-i} are the bids of all other agents.

2. Allocative Efficiency (Social Welfare) Allocative Efficiency or Social Welfare Maximizing allocation refers to the allocation that maximizes total value for all searchers, i.e.,

$$\max_{A \in \mathcal{A}} \sum_{i=1}^{m} v_i(A(b)). \tag{8}$$

3. Individual Rationality A matchmaking is individually rational if it is individually rational for both searchers and transaction creators. Each searcher is at least as well off by participating in the matchmaking as they would be if they chose not to participate. Similarly, the utility of each transaction creator is at least as well by including it in matchmaking as they would if they choose not to be involved in matchmaking. Formally, the utility $u_i^{\mathcal{S}}$ of searcher i and utility $u_i^{\mathcal{T}}$ should be non-negative:

$$u_i^{\mathcal{S}} = v_i(A(b)) - p_i(b) \geq 0. \tag{9}$$

$$u_i^{\mathcal{T}} = r_i \geq 0 \tag{10}$$

4. No-deficit A matchmaking is no-deficit if the total payments collected from the searchers are equal to the total rebates paid to transaction creators of the allocated transactions. That is, no money is lost from the matchmaking process. The condition for no deficit is:

$$\sum_{i \in \mathcal{S}^*} p_i(v) = \sum_{i \in \mathcal{T}^*} r_i. \tag{11}$$

5. Fair redistribution A matchmaking is fair to transaction creators if rebate to the transaction creators is such that i) $\sum_{i \in [n]} r_i = R$, where R is the revenue generated from searchers, (ii) $\forall i, j$ similar MEV transactions generating same value, $r_i = r_j$, and (iii) $r_i = 0$ for all transactions i that do not create any MEV.

B.4 Winner Determination in Additive Settings

B.5 `RST-Game` Examples

Example 2 (Non-Monotonocity of v in Single-minded Setting). *Consider* `RST-Game` *with* $\mathcal{T} = \{t_1, t_2, t_3, t_4, t_5\}$ *and* $\mathcal{S} = \{s_1, s_2, s_3, s_4\}$ *in single-minded setting where the bundle-bid pairs are* $(B_1, v_1) = (\{t_1, t_2, t_3, t_4\}, 9), (B_2, v_2) = (\{t_2, t_3\}, 8), (B_3, v_3) = (\{t_4, t_5\}, 7), (B_4, v_4) = (\{t_3, t_4\}, 6)$.

Based on the above bundle-bid pairs we have the following:
The order of searchers sorted by bid/bundle is s_1, s_2, s_3, s_4.
Consider the grand coalition $T = \{t_1, t_2, t_3, t_4, t_5\}$. $\mathcal{S}^T = S$, s_1 get t_1, t_2, t_3, t_4 and $p_{s_1} = 5.6 * 2 = 11.2$.
Therefore, $v(T) = 11.2$
Consider $T' \subseteq T$ such that $T' = T \setminus \{t_1\}$. $\mathcal{S}^{T'} = \{s_2, s_3\}$ and $p_{s_2} = 6/\sqrt{2} * \sqrt{2} = 6, p_{s_3} = 6/\sqrt{2} * \sqrt{2} = 6$.
Therefore, $v(T') = 12$
Clearly, $T' \subset T$ and $v(T) < v(T')$. Hence, v is not monotonic.

Algorithm 3 SPA with VCG Payments

Input: Bids $v_{s_1}, v_{s_2}, \ldots, v_{s_m}$
Initialize $p_{s_i} = 0$ for each searcher s_i
for each transaction $t_j \in T$ **do**
  Find $v_{s_i}[t_j]$ of each searcher s_i.
  Let ${s_i}^* = \arg\max_{s_i} v_{s_i}[t_j]$ {Identify the participant with the highest bid}
  Let $v_{s_{\text{second}}}[t_j] = \max_{s_i \neq {s_i}^*} v_{s_i}[t_j]$ {Find the second-highest bid}
  Payments:
  for each searcher s_i **do**
    if $s_i = {s_i}^*$ **then**
      $p_{s_i} = p_{s_i} + v_{s_{\text{second}}}[t_j]$ {Winner pays the second-highest bid}
    else
      $p_{s_i} = p_{s_i} + 0$ {Non-winners pay nothing}
    end if
  end for
end for
Output: Winner ${s_i}^*$ and payments $p_{s_1}, p_{s_2}, \ldots, p_{s_n}$

Searcher	B	v	$\sqrt{\lvert B \rvert}$	ratio
s_1	t_1, t_2, t_3, t_4	12	2	4.5
s_2	t_2, t_3	8	$\sqrt{2}$	5.6
s_3	t_4, t_5	7	$\sqrt{2}$	4.9
s_4	t_3, t_4	6	$\sqrt{2}$	4.2

C UNIQUE MARGINAL CONTRIBUTIONS

C.1 Proof of Unique Marginal Contributions

Theorem 4. *$\forall (a_0, \ldots, a_{n-1}) \in \{-1, 0, 1\}^n$, the summation $s = \sum_{i=0}^{i=n-1} 3^i * a_i$ is unique.*

Proof. Proof by induction:
Base case: $n = 1$

For $a_0 \in \{-1, 0, 1\}$, the summation s can take value either $-1, 0$ or 1. Thus, all three possible summations for $n = 1$ are unique.
Inductive step: Assume the theorem to be true for $n = k$.

The minimum summation possible for $(a_0, \ldots, a_{k-1})$ is when all $a_i = -1$.

$$s_{\min} = -1 \times \frac{3^k - 1}{3 - 1} = -\frac{3^k - 1}{2}$$

The maximum summation possible for $(a_0, \ldots, a_{k-1})$ is when all $a_i = 1$.

$$s_{\max} = 1 \times \frac{3^k - 1}{3 - 1} = \frac{3^k - 1}{2}$$

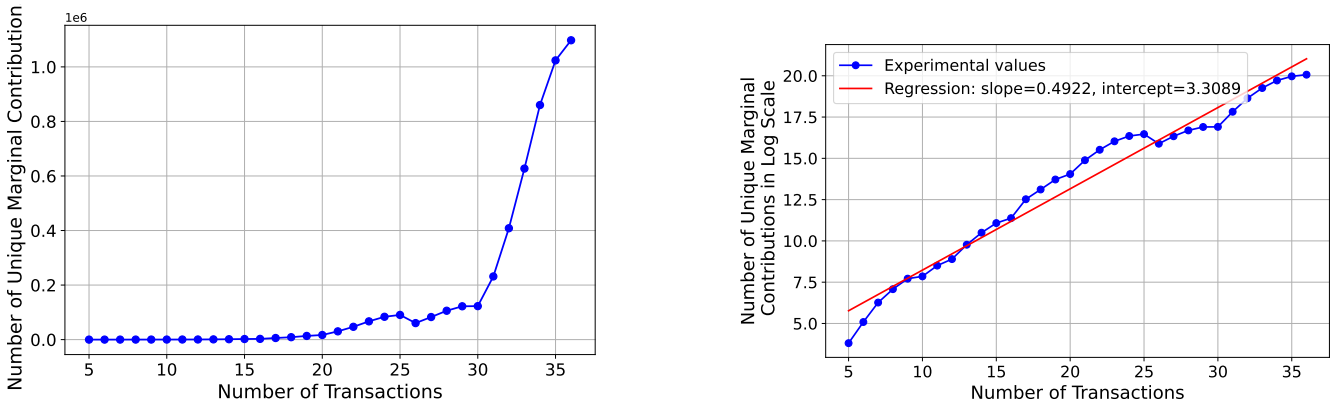
For $n = k + 1$, a_{k+1} can take values either $-1, 0$ or 1.

- $a_{k+1} = -1$: $s^0_{\min} = -\frac{3^{k+1}-1}{2}$ and $s^0_{\max} = -\frac{3^k+1}{2}$
- $a_{k+1} = 0$: $s^1_{\min} = -\frac{3^k-1}{2}$ and $s^1_{\max} = \frac{3^k-1}{2}$
- $a_{k+1} = 1$: $s^2_{\min} = \frac{3^k+1}{2}$ and $s^2_{\max} = \frac{3^k-1}{2}$

As $s^0_{\max} < s^1_{\min}$, $s^0_{\max} < s^2_{\min}$ and $s^1_{\max} < s^2_{\min}$, neither ranges of values overlap. Thus, if all possible summations for $n = k$ are unique, then the summations for $n = k + 1$ are also unique.
By the principle of mathematical induction, the summations $s = \sum_{i=0}^{i=n-1} 3^i * a_i$ are unique. □

C.2 Empirical Analysis

Figure 5 demonstrates the relationship between the number of unique values and the number of transactions, n based on the construction provided for Theorem 2. For 10k different instances, we generate different instances of `RST-Game` and observed that the observed average growth rate of the number of unique values closely follows the theoretical prediction of $\Omega(2^{\sqrt{n}})$. This validates our theory that the number of unique values scales asymptotically as $\Omega(2^{\sqrt{n}})$ with respect to n, confirming the expected behavior.



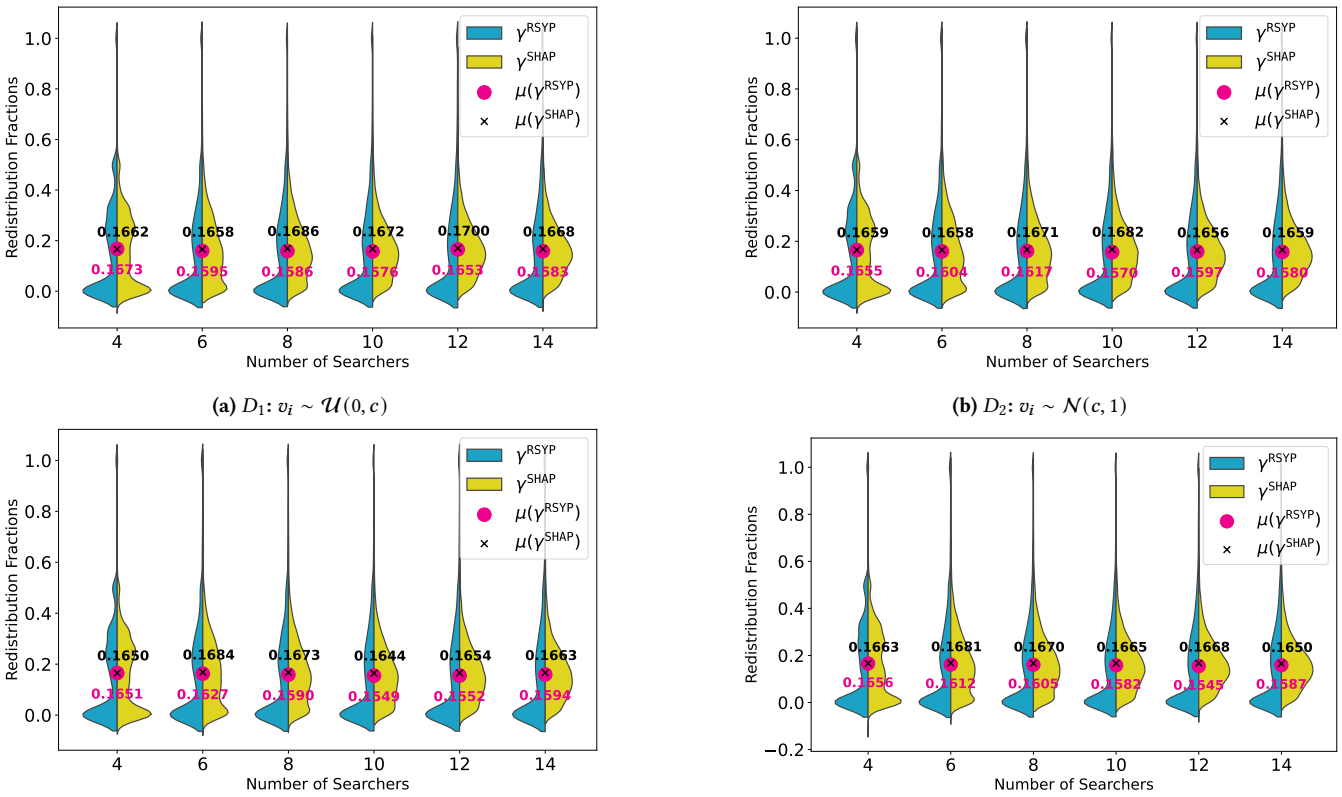
(a) Linear Scale **(b) Log Scale**

Figure 5: Variation of Unique Marginal Contributions with Transactions

D PERFORMANCE OF RSYP

Figures 6 and 7 show the performance analysis of RSYP with varying searchers and transaction creators. Specifically, they show the distribution of redistribution fractions obtained by transaction creators across 10k random instances generated using the following distributions: (i) D_1: $v_i \sim \mathcal{U}(0, c)$ (ii) D_2: $v_i \sim \mathcal{N}(c, 1)$ (iii) D_3: $v_i \sim \text{EXP}(c/2)$ (iv) D_4: $v_i \sim \text{Triangular}(0, c/2, c)$.

For each distribution, we show the distribution of redistribution fractions, γ_i, of a random transaction creator computed via RSYP and exact Shapley value $c = 1$. We observe that the expected redistribution fraction $\mu(\gamma_i^{\text{RSYP}}) \to \mu(\gamma_i^{\text{SHAP}})$.



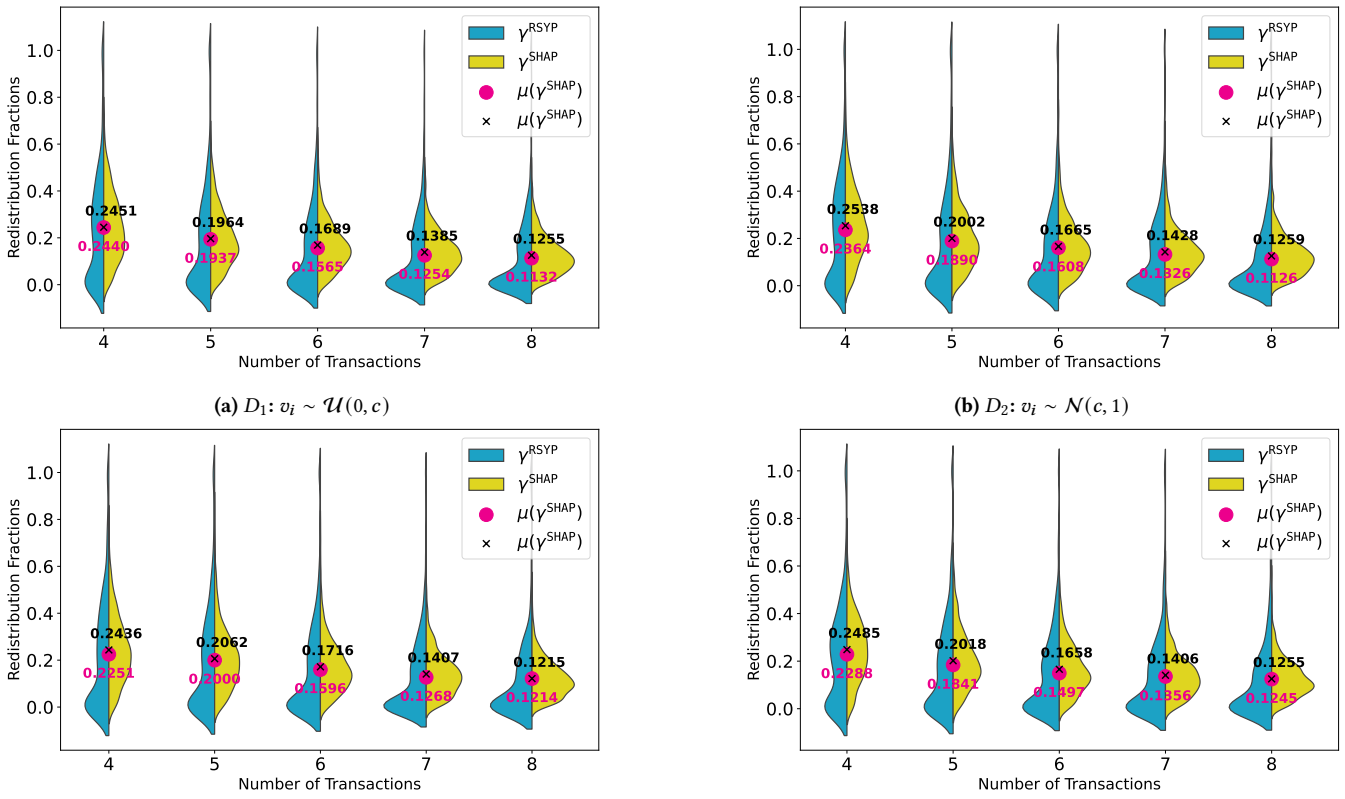
(a) D_1: $v_i \sim \mathcal{U}(0, c)$

(b) D_2: $v_i \sim \mathcal{N}(c, 1)$

(c) D_3: $v_i \sim \mathbf{EXP}(c/2)$

(d) D_4: $v_i \sim \mathbf{Triangular}(0, c/2, c)$

Figure 6: Distribution of $\gamma^{\mathbf{RSYP}}$, $\gamma^{\mathbf{SHAP}}$ vs m for $n = 6, c = 1$ for a randomly selected user



(a) D_1: $v_i \sim \mathcal{U}(0, c)$

(b) D_2: $v_i \sim \mathcal{N}(c, 1)$

(c) D_3: $v_i \sim$ **EXP**$(c/2)$

(d) D_4: $v_i \sim$ **Triangular**$(0, c/2, c)$

Figure 7: Distribution of $\gamma^{\text{RSYP}}, \gamma^{\text{SHAP}}$ vs n for $m = 14, c = 1$ for a randomly selected user

We perform our analysis using the Python programming language on a Windows 10 Machine with AMD Ryzen 5 4600H with Radeon Graphics, 3000 MHz, and 8GB RAM.

E UNIANIMITY GAMES

A unanimity game is a fundamental concept in cooperative game theory, characterized by a coalition structure in which a specific subset of players, called a winning coalition, must act unanimously for any payoff to be generated. The game's value depends on whether a coalition contains the required subset.

Let N be a finite set of players, and let $S \subseteq N$ be a coalition (a subset of players). A unanimity game is represented by a characteristic function $v : 2^N \to \mathbb{R}$ defined as:

$$v(T) = \begin{cases} 1, & \text{if } S \subseteq T, \\ 0, & \text{otherwise.} \end{cases}$$

where S is the winning coalition, i.e., the minimal subset of players required to generate a value. T is any coalition under consideration. $v(T)$ gives the value of the coalition T. It is 1 if T contains all players in S, and 0 otherwise.

The Shapley value of `RST-Game` can derived from unanimity games in the following way. Let $U_S = (\mathcal{T}, \omega_S)$ be unanimity game defined on set of transactions $\mathcal{T}$ such that

$$\forall T \subseteq \mathcal{T}, \omega_S(T) = \begin{cases} 1 & S \subseteq T \\ 0 & \text{Otherwise} \end{cases} \tag{12}$$

Theorem 5. *$\forall T \subseteq \mathcal{T}$, the value function can be uniquely expressed in terms of unanimity functions in B_N with Harsanyi dividends as the coefficients*

$$v(T) = \sum_{S \in 2^{\mathcal{T}} \setminus \phi} \Delta_{v,S} \omega_S(T) \tag{13}$$

Proof. Let $E = 2^{\mathcal{T}} \setminus \phi$ i.e., $E = \{T_1, \ldots, T_{2^{|\mathcal{T}|}-1}\}$

Let Λ be the matrix formed using unanimity functions $\Lambda_{(i,j)} = \omega_{T_i}(T_j) \forall i, j \in \{0, \ldots, 2^{|\mathcal{T}|} - 1\}$

ω	$\mathbf{T_1}$	$\mathbf{T_2}$	$\mathbf{T_3}$	$\mathbf{T_4}$	$\cdots$	$\cdots$	$\mathbf{T_{2^{|\mathcal{T}|}-1}}$
ω_{T_1}	1	0	0	1	$\cdots$	$\cdots$	1
ω_{T_2}	0	1	0	1	$\cdots$	$\cdots$	1
ω_{T_3}	0	0	1	0	$\cdots$	$\cdots$	1
ω_{T_4}	0	0	0	1	$\cdots$	$\cdots$	1
ω_{T_5}	0	0	0	0	$\cdots$	$\cdots$	1
ω_{T_6}	0	0	0	0	1	$\cdots$	1
ω_{T_7}	0	0	0	0	0	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\omega_{T_{2^{|\mathcal{T}|}-1}}$	0	0	0	0	0	0	1

Assume there is a linear combination of unanimity functions that equals the zero function:

$$\sum_{S \in E} v_S \omega_S(T) = 0, \quad \forall T \in E$$

For this to hold, v_S must be zero for all $S \subseteq \mathcal{T}$. The unanimity function $\omega_S(T)$ is nonzero (equal to 1) if and only if $S \subseteq T$. This means $\omega_S(T)$ activates only those terms where $S \subseteq T$. If $v_S \neq 0$ for some $S \subseteq \mathcal{T}$, then $\sum_{S \subseteq \mathcal{T}} v_S \omega_S(T) \neq 0$ for some T. This contradicts the assumption that the sum equals 0 for all T. Thus, $v_S = 0$ for all S. This proves $\{\omega_S : S \in E\}$ are linearly independent.

Further, consider the following equations:

$$\begin{aligned}
v(T_1) &= \sum_{S \in E} v_S \omega_S(T_1) \\
v(T_2) &= \sum_{S \in E} v_S \omega_S(T_2) \\
v(T_3) &= \sum_{S \in E} v_S \omega_S(T_3) \\
&\vdots \\
v(T_{2^{|\mathcal{T}|}-1}) &= \sum_{S \in E} v_S \omega_S(T_{2^{|\mathcal{T}|}-1})
\end{aligned}$$

From the Λ, it can be observed that there are $2^{|\mathcal{T}|} - 1$ independent equations and $2^{|\mathcal{T}|} - 1$ variables, hence there exists unique solution $\{v_T : T \subseteq \mathcal{T}\}$.

Hence, $B_{\mathcal{T}} = \{\omega_T : T \subseteq \mathcal{T} \setminus \phi\}$ forms the basis for the characteristic function v. These v_S are nothing but Harsanyi dividends represented as $\Delta_{v,S}$.

$$v(T) = \sum_{S \in 2^{\mathcal{T}} \setminus \phi} \Delta_{v,S} \omega_S(T)$$

□

Theorem 6. *The Shapley value of t_j in the unanimity game $(\mathcal{T}, \omega_T)$ is:*

$$\varphi_{t_j}(\omega_T) = \begin{cases} \frac{1}{|T|}, & if\, t_j \in T, \\ 0, & if\, t_j \notin T. \end{cases}$$

Proof.

$$\Delta\omega_T(C)(t_j) = \begin{cases} 1, & T \setminus \{t_j\} \subseteq C, \\ 0, & T \setminus \{t_j\} \nsubseteq C. \end{cases}$$

$$\implies \forall t_j \notin S, \quad \varphi_{t_j}(\omega_T) = 0.$$

$$\omega_T(\mathcal{T}) = 1.$$

$$\varphi_{t_j}(\omega_T) = \frac{1}{N!} \sum_{C \subseteq \mathcal{N}_i} |C|!(N - |C| - 1)!\big(\omega_T(C \cup \{t_j\}) - \omega_T(C)\big) \tag{14}$$

From symmetry, each transaction in S receives an equal share of $\nu(\mathcal{T})$:

$$\implies \forall i \in S, \quad \varphi_{t_j}(\omega_T) = \frac{1}{|T|}.$$

□

Theorem 7. *Any characterisitic function $\nu : 2^{\mathcal{T}} \setminus \{\emptyset\} \to \mathbb{R}_+$, the Shapley value can be expressed in terms of the Harsanyi dividends as:*

$$\varphi_{t_j}(\nu) = \sum_{T \subseteq 2^{\mathcal{T}} \setminus \{\emptyset\}, t_j \in T} \frac{\Delta_{\nu,T}}{|T|}.$$

Proof. Let $N_S = |C|!\,(|\mathcal{T}| - |C| - 1)!$

$$\varphi_{t_j}(\nu) = \frac{1}{|\mathcal{T}|!} \sum_{C \subseteq \mathcal{T} \setminus \{t_j\}} N_C\{\nu(C \cup \{t_j\}) - \nu(C)\}.$$

From Theorem 5, 6, we have

$$\nu(C) = \sum_{T \subseteq 2^{\mathcal{T}} \setminus \{\emptyset\}} \Delta_{\nu,T}\omega_T(C) \text{ and}$$

$$\varphi_{t_j}(\omega_T) = \frac{1}{|T|} \quad \text{or } 0.$$

$$\begin{aligned}
\varphi_{t_j}(\nu) &= \frac{1}{|\mathcal{T}|!} \sum_{C \subseteq \mathcal{T} \setminus \{t_j\}} N_C\big\{\nu(C \cup \{t_j\}) - \nu(C)\big\} \\
&= \frac{1}{|\mathcal{T}|!} \sum_{C \subseteq \mathcal{T} \setminus \{t_j\}} N_C\Bigg(\sum_{T \subseteq 2^{\mathcal{T}} \setminus \emptyset} \Delta_{\nu,T}\omega_T(C \cup \{t_j\}) - \\
&\qquad\qquad \sum_{T \subseteq 2^{\mathcal{T}} \setminus \emptyset} \Delta_{\nu,T}\omega_T(C)\Bigg) \\
&= \frac{1}{|\mathcal{T}|!} \sum_{C \subseteq \mathcal{T} \setminus \{t_j\}} \sum_{T \subseteq 2^{\mathcal{T}} \setminus \emptyset} N_C\Delta_{\nu,T}\big(\omega_T(C \cup \{t_j\}) - \omega_S(C)\big) \\
&= \frac{1}{|\mathcal{T}|!} \sum_{C \subseteq \mathcal{T} \setminus \{t_j\}} \sum_{T \subseteq 2^{\mathcal{T}} \setminus \emptyset, t_j \in T} N_C\Delta_{\nu,T}\big(\omega_T(C \cup \{t_j\}) - \omega_T(C)\big) \\
&= \frac{1}{|\mathcal{T}|!} \sum_{T \subseteq 2^{\mathcal{T}} \setminus \emptyset, i \in T} \sum_{T \subseteq \mathcal{T} \setminus \{t_j\}} N_C\Delta_{\nu,T}\big(\omega_T(C \cup \{i\}) - \omega_T(C)\big) \\
&= \frac{1}{|\mathcal{T}|!} \sum_{T \subseteq 2^{\mathcal{T}} \setminus \emptyset, t_j \in T} \Delta_{\nu,T} \sum_{C \subseteq \mathcal{T} \setminus \{t_j\}} N_C\big(\omega_T(C \cup \{i\}) - \omega_T(C)\big) \\
&= \sum_{T \subseteq 2^{\mathcal{T}} \setminus \emptyset, t_j \in T} \Delta_{\nu,T}\left(\frac{1}{|\mathcal{T}|!} \sum_{C \subseteq \mathcal{T} \setminus t_j} N_C\big(\omega_T(C \cup \{t_j\}) - \omega_T(C)\big)\right) \\
&= \sum_{T \subseteq 2^{\mathcal{T}} \setminus \emptyset, t_j \in T} \Delta_{\nu,T}\varphi_{t_j}(\omega_T) \\
&= \sum_{T \subseteq 2^{\mathcal{T}} \setminus \phi, t_j \in T} \frac{\Delta_{\nu,T}}{|T|}
\end{aligned}$$

□

F CODES

F.1 Main

```
seed = 5
np.random.seed(seed)
numIterations = 10000

c=1
numSearchersIter = [4,6,8,10,12,14]
# numTransactionsIter = [4,5,6]
numTransactionsIter = [4,5,6,7,8]
distributions = ['uniform','normal','exp','trian']
```

F.2 Generate Instances

```
def truncated_normal_vec(mean, std, c, size):
    samples = []
    while len(samples) < size:
        s = np.random.normal(mean, std, size*2)  # oversample
        s = s[(s >= 0) & (s <= c)]
        samples.extend(s[:size - len(samples)])
    return np.array(samples)

def truncated_exponential_vec(scale, c, size):
    samples = []
    while len(samples) < size:
        s = np.random.exponential(scale, size*2)
        s = s[s <= c]
        samples.extend(s[:size - len(samples)])
    return np.array(samples)

def generate_instances(dist,numSearchers,numTransactions,c):
  if dist == 'uniform':
    searcherVals = np.random.uniform(0,c,(1,numSearchers)).flatten()
    searcherBdls = [np.unique(np.random.uniform(0,numTransactions,(1,
        ... int(np.random.uniform(1,numTransactions)))).astype(int)) for bi in
        ... range(numSearchers)]
  elif dist == 'normal':
    searcherVals = truncated_normal_vec(c/2, 1, c, numSearchers)
    searcherBdls = [np.unique(np.random.uniform(0,numTransactions,(1,
        ... int(np.random.uniform(1,numTransactions)))).astype(int)) for bi in
        ... range(numSearchers)]
  elif dist == 'exp':
   searcherVals = truncated_exponential_vec(c/2, c, numSearchers)
   searcherBdls = [np.unique(np.random.uniform(0,numTransactions,(1,
       ... int(np.random.uniform(1,numTransactions)))).astype(int)) for bi in
       ... range(numSearchers)]
  elif dist == 'trian':
   searcherVals = np.random.triangular(left=0,mode=c/2, right = c, size=numSearchers)
   np.random.uniform(0,c,(1,numSearchers)).flatten()
   searcherBdls = [np.unique(np.random.uniform(0,numTransactions,(1,
       ... int(np.random.uniform(1,numTransactions)))).astype(int)) for bi in
       ... range(numSearchers)]
```

```
  else:
    raise ValueError
  return searcherVals,searcherBdls
```

F.3 ICA-SM

```
def findDisjointBunldeOwners(sortedBdl, searcherBdls):
  disjointBdlOwners = []
  disjointBdlSet = set()
  for index in sortedBdl:
    flagSet = False
    for txn in searcherBdls[index]:
      if txn in disjointBdlSet:
        flagSet = True
        break

    if not flagSet:
      disjointBdlSet = disjointBdlSet.union(set(searcherBdls[index]))
      disjointBdlOwners.append(index)

  return disjointBdlOwners


def findSumValuations(disjointBdlOwners, searcherVals):
  tvals = 0
  for i in disjointBdlOwners:
    tvals += searcherVals[i]
  return tvals

def findFirstIntersectingBundleOwner(userBdl, partial_sortedBdl, searcherBdls):
  for sIndex in partial_sortedBdl:
    for txn in userBdl:
      if txn in searcherBdls[sIndex]:
        return sIndex
  return -1

def findIcaSmPayments(numSearchers, sortedBdl, disjointBdlOwners, searcherVals,
    ... searcherBdls, searcherBdlSizes):
  payments = [0]*numSearchers

  for i in disjointBdlOwners:
      winIndex = np.where(sortedBdl == i)[0][0]
      userBdl=searcherBdls[i]
      sIndex = findFirstIntersectingBundleOwner(userBdl,sortedBdl[winIndex+1:], searcherBdls)
      if sIndex >= 0:
        payments[i] =
            ... searcherVals[sIndex]*math.sqrt(searcherBdlSizes[i]/searcherBdlSizes[sIndex])

  return payments

def findShapleyValueIcaSm(numTransactions,searcherBdls,searcherVals,searcherBdlSizes,
    ... permutations):
  shapley = []
```

```
transactions = [i for i in range(numTransactions)]

for i in range(numTransactions):
  marginalContribution = 0
  for perm in permutations:

    index = perm.index(i)
    coalitionTxns = list(perm[:index+1])

    searcherBdls_temp = []
    searcherVals_temp = []
    searcherBdlSizes_temp = []

    for sIndex,bdl in enumerate(searcherBdls):
      if set(bdl).issubset(set(coalitionTxns)):
        searcherVals_temp.append(searcherVals[sIndex])
        searcherBdls_temp.append(searcherBdls[sIndex])
        searcherBdlSizes_temp.append(searcherBdlSizes[sIndex])


    R = [round(searcherVals_temp[j]/math.sqrt(searcherBdlSizes_temp[j]), 3) for j in
        ... range(len(searcherVals_temp))]
    sortedBdl_temp = np.argsort(np.array(R)*-1)
    disjointBdlOwners_temp = findDisjointBunldeOwners(sortedBdl_temp,searcherBdls_temp)
    optVal = findSumValuations(disjointBdlOwners_temp, searcherVals_temp)
    icaSmPayments = findIcaSmPayments(len(searcherBdls_temp), sortedBdl_temp,
        ... disjointBdlOwners_temp, searcherVals_temp, searcherBdls_temp,
        ... searcherBdlSizes_temp)
    icaRevenue= sum(icaSmPayments)

    tvals_temp_with = icaRevenue

    coalitionTxns.pop(-1)

    searcherBdls_temp = []
    searcherVals_temp = []
    searcherBdlSizes_temp = []

    for sIndex,bdl in enumerate(searcherBdls):
      if set(bdl).issubset(set(coalitionTxns)):
        searcherVals_temp.append(searcherVals[sIndex])
        searcherBdls_temp.append(searcherBdls[sIndex])
        searcherBdlSizes_temp.append(searcherBdlSizes[sIndex])

    R = [round(searcherVals_temp[j]/math.sqrt(searcherBdlSizes_temp[j]), 3) for j in
        ... range(len(searcherVals_temp))]
    sortedBdl_temp = np.argsort(np.array(R)*-1)
    disjointBdlOwners_temp = findDisjointBunldeOwners(sortedBdl_temp,searcherBdls_temp)
    optVal = findSumValuations(disjointBdlOwners_temp, searcherVals_temp)
```

```
      icaSmPayments = findIcaSmPayments(len(searcherBdls_temp), sortedBdl_temp,
          ... disjointBdlOwners_temp, searcherVals_temp, searcherBdls_temp,
          ... searcherBdlSizes_temp)
      icaRevenue= sum(icaSmPayments)

      tvals_temp_without = icaRevenue

      marginalContribution += tvals_temp_with - tvals_temp_without

    shapley.append(marginalContribution)

  shapley = np.round(np.array(shapley) / math.factorial(numTransactions), decimals=3)
  return shapley
```

F.4 RSYP

```
def RSYP(sampledSubsets, numTransactions, searcherVals, searcherBdls,
    ... searcherBdlSizes,R,numSearchers):
  shapley = []

  for j in range(numTransactions):
    marginalContribution = 0

    for subset in sampledSubsets:

      bdlp = list(subset)
      if j in bdlp:

        searcherBdls_temp = searcherBdls.copy()
        searcherVals_temp = searcherVals.copy()
        searcherBdlSizes_temp = searcherBdlSizes.copy()
        R_temp = R.copy()

        popIndices1=[]

        for i in range(numSearchers-1,-1,-1):
          if not set(searcherBdls[i]).issubset(set(bdlp)):
            popIndices1.append(i)

        if(len(popIndices1) != numSearchers):

          for i in popIndices1:
            searcherBdls_temp.pop(i)
            searcherVals_temp.pop(i)
            searcherBdlSizes_temp.pop(i)
            R_temp.pop(i)


          sortedBdl_temp = np.array([i for i in range(len(searcherVals_temp))])
          disjointBdlOwners_temp = findDisjointBunldeOwners(sortedBdl_temp,searcherBdls_temp)
          optVal = findSumValuations(disjointBdlOwners_temp, searcherVals_temp)
```

```
            icaSmPayments = findIcaSmPayments(len(searcherBdls_temp), sortedBdl_temp,
                ... disjointBdlOwners_temp, searcherVals_temp, searcherBdls_temp,
                ... searcherBdlSizes_temp)
            icaRevenue1= sum(icaSmPayments)


            bdlp.remove(j)

            popIndices2=[]

            for i in range(len(searcherVals_temp)-1,-1,-1):
              if not set(searcherBdls_temp[i]).issubset(set(bdlp)):
                popIndices2.append(i)

            for i in popIndices2:
              searcherBdls_temp.pop(i)
              searcherVals_temp.pop(i)
              searcherBdlSizes_temp.pop(i)
              R_temp.pop(i)

            sortedBdl_temp = np.array([i for i in range(len(searcherVals_temp))])
            disjointBdlOwners_temp = findDisjointBunldeOwners(sortedBdl_temp,searcherBdls_temp)
            optVal = findSumValuations(disjointBdlOwners_temp, searcherVals_temp)
            icaSmPayments = findIcaSmPayments(len(searcherBdls_temp), sortedBdl_temp,
                ... disjointBdlOwners_temp, searcherVals_temp, searcherBdls_temp,
                ... searcherBdlSizes_temp)
            icaRevenue2= sum(icaSmPayments)


            diff = icaRevenue1 - icaRevenue2
          else:
            diff = 0.0
        else:
          diff = 0.0

        MCT = round(diff,3)

        if MCT <= 0:
          MCT = 0.0

        marginalContribution += MCT

      shapley.append(marginalContribution)

    shapley = np.round(np.array(shapley) / len(sampledSubsets), decimals=3)
    gammas = findGammas(shapley)
    return gammas
```

F.5 Gamma Computation

```
def findGammas(shapleyList):
    totalShap = 0
    for shap in shapleyList:
```

```
    totalShap+=shap
  if(totalShap==0):
    return np.round(shapleyList, decimals=3)
  else:
    return np.round(np.divide(shapleyList, totalShap), decimals=3)
```