Enhancing Coin Selection in UTXO-based Blockchains through Modified Greedy Algorithms

***Abstract*—This paper delves into the coin selection algorithms employed by UTXO-based blockchains, such as Bitcoin and Cardano, and introduces a modification to the standard greedy algorithm utilizing randomization and adaptiveness. Our proposed modification aims to optimize the selection process by minimizing the UTXO pool size, albeit at the expense of an increased transaction fee. The adjustment is simple yet remarkably effective, resulting in a substantial average decrease of approximately 85% in the UTXO pool size, with a marginal average transaction fee increase of only approximately 0.07%. This paper prompts a critical inquiry into the nature of this trade-off, raising the question of whether the balance achieved is indeed a trade-off or if the focus should be redirected towards further optimizing the conventional greedy algorithm.**

***Index Terms*—Coin Selection, UTXO, Greedy, Blockchain**

I. Introduction

Coin selection algorithms play a crucial role in the operation of blockchains. These algorithms determine the specific coins or tokens chosen for a given transaction, significantly influencing the overall efficiency, privacy, and reliability of blockchain systems. Coin selection refers to the algorithmic process of selecting unspent transaction outputs (UTXOs) from a user's wallet to initiate a transaction by transferring tokens to designated recipients. In this process, the chosen set of UTXOs serves as the input for the transaction, while the transaction output includes the payment to the target recipient and the change that returns to the user's wallet. It is therefore of the utmost importance to have a good algorithm in place. Among many others, one possible class of algorithms are greedy algorithms. A greedy algorithm is an approach that quickly identifies a solution to problems by selecting the most promising option at each step, prioritizing local optimization without considering the global optimization of the final solution. This can be very effective but may yield unwanted outcomes. In this paper we study versions of the standard greedy algorithm employing randomness and adaptiveness. In particular, we observe through simulations that the standard greedy algorithm selects a minimal number of input UTXOs which is directly correlated with minimizing the transaction fee—a desired property. However, as we model a real world user wallet's behavior there are both incoming and outgoing transactions. In the particular, in the studied setting the UTXO pool size experiences a pronounce increase, while minimizing the transaction fee. Having a large UTXO pool is an undesired property, as (a) there are more UTXO to be processed to run the coin selection algorithm; (b) there is an increased risk of having dust UTXOs. Keeping the pool size low, bounds the risk of having dust UTXOs.

In this paper, we focus on the two objectives of minimizing the transaction fee and minimizing the UTXO pool size. An inherent trade-off arises, as the reduction of the UTXO pool size necessitates the selection of larger subsets of UTXOs as transaction inputs. Consequently, this augmentation in the selected UTXO subsets contributes to an increase in the transaction fee. Simulations suggest surprising results and indicate that the trade-off is less pronounced.

The paper is organized as follows. The next section contains related literature. Section II introduces notation and objectives. In Section III, we study the different versions of the greedy algorithm. In Section IV, we study their performance. Section V concludes.

Related Literature

Coin selection algorithms for UTXO-based blockchain have been studied earlier but mostly informally. In particular, many suggested algorithms were are only part of blog posts. [1] provide an extensive survey on coin selection algorithms and provide a comparison of the advantages and disadvantages of different algorithms. [2] study an coin selection algorithm for multi-asset UTXOs based on optimization.

The literature related to randomized adaptive greedy search includes works such as [3] and [4]. It's important to note that these papers adopt a distinct approach, initially identifying candidate solutions and then optimizing within the set of candidates.

Greedy algorithms in the context of coin selection for UTXO selection have been studied in [5].

II. Terminology and Notation

The advent of the Bitcoin era brought about the necessity for cryptocurrency wallets. A pivotal element within these wallets is the coin selection algorithm, responsible for choosing a set of unspent transaction outputs (UTXOs) to construct the input for a transaction. Each blockchain user stores their tokens in the form of UTXOs within the blocks of the blockchain. The user's UTXO pool is managed by wallet software, keeping track of the UTXOs on the blockchain, as illustrated in Fig. 1.

Every transaction incorporates existing UTXOs as input and generates new UTXOs as output. The output UTXO comprises two primary components: the payment UTXO and the change UTXO. The payment UTXO represents the tokens transferred to the recipient's wallet, with its value termed the *target value* denoted by T. Meanwhile, the change UTXO denotes the tokens sent back to the sender's wallet. The value of the change UTXO is calculated as the difference of the total

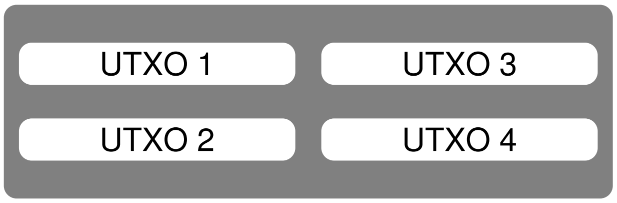


Fig. 1: Example of a user's UTXO pool. The user is eligible to spend four UTXOs which each carry some value in the native token.

value of the input UTXOs and the target value. The disparity between the total values of the input UTXOs and the total values of the output UTXOs is referred to as the *transaction fee*. Fig. 2 illustrates a sample transaction with input UTXOs and the output UTXOs—the payment and change UTXO. A transaction is said to be *balanced* if the total input value is equal to the total output value plus the transaction fee.

Once a transaction is submitted to a blockchain network, it undergoes processing by block producers. The block producer selecting a given transaction and incorporating its UTXO outputs into a new block receives the transaction fee as part of the block generation reward.

Transaction fee: The transaction fee depends on the number of input and output UTXOs. We assume a unified structure of UTXOs, i.e. ignoring additional data that a UTXO can hold and assuming the sizes of UTXOs are constant for any value it holds. Therefore, we assume that the transaction fee is linear in the number of input UTXOs. In Section IV we will compare the number of input UTXOs used by the different greedy algorithms and derive conclusions about the incurred transaction fees.

Notation: We introduce some notations about UTXOs and UTXO pools, following [1]. Let $U = \{u_1, ..., u_n\}$ be a pool of UTXOs with n UTXOs. As we are considering the case of a single asset only, all UTXOs are represented by there respected amount of tokens it holds. Let v denote the *value function*, that takes any set of UTXOs as input and returns the total value of the set. That is, for any set of UTXOs U, the total value of U is given by $v(U) = \sum_{u \in U} u$. Also note that by definition we have that for any UTXO $u \in U : v(u) = u$.

Objectives: We focus on analyzing the following two objectives.

O.1 Minimize transaction fee $\simeq$ Minimize no. input UTXOs
O.2 Minimize UTXO pool size

The initial objective holds evident significance from a user standpoint, whereas the subsequent goal pertains to the execution time of algorithms and mitigating the inherent risks associated with an excessively large set of dust UTXOs.

Next, we shift our focus to analyzing greedy algorithms.

III. Analysis of Greedy Algorithms

In this section, we analyze three different versions of the greedy algorithm. First, we study the standard Greedy algorithm which serves as benchmark. Second, we study a version of the Greedy algorithm with randomness. And finally, we present our randomized adaptive Greedy algorithm.

Algorithm 1 Greedy Algorithm — $\mathcal{G}(U, T)$

```
Input: $U = \{u_1, ..., u_n\}$ with $u_i \geq u_{i'}$ for $i < i'$
Input: Target $T > 0$
Output: Set of selected UTXOs $S^{\text{greedy}}$
Require: $v(U) > T$
 1: $S^{\text{greedy}} \leftarrow \{\}$
 2: $rem \leftarrow T$
 3: for $i = 1$ to $n$ do                                   ▷ Part P.1
 4:     if $u_i \leq rem$ and $rem > 0$ then
 5:         $S^{\text{greedy}}$.add($u_i$)
 6:         $rem \leftarrow rem - u_i$
 7:     end if
 8: end for
 9: while $rem > 0$ do                                      ▷ Part P.2
10:     $S^{\text{greedy}}$.add(min$\{U \setminus S^{\text{greedy}}\}$))
11:     $rem \leftarrow rem -$ min$\{U \setminus S^{\text{greedy}}\}$
12: end while
13: return $S^{\text{greedy}}$
```

A. Benchmark: Standard Greedy Algorithm

Given a target payment value T and a set of UTXOs U, the greedy algorithm aims at finding an optimal subset $S^{greedy} \subseteq U$ such that the total value of the subset $v(S^{greedy})$ is sufficient to pay the target payment value[1], i.e. $v(S^{greedy}) \geq T$. We start with an empty set S^{greedy} of selected UTXOs and set variable *rem*—that reflects the remaining amount—equal to T. The standard greedy algorithm consists of two parts:

P.1 Take UTXOs in descending order and select UTXOs that are below the remaining value *rem*. Once a UTXO u is selected and added to the set S^{greedy}, its value $v(u)$ is subtracted from the remaining value *rem*. Whenever the target value is reached, i.e. if $rem \leq 0$, the algorithm returns S^{greedy}.

P.2 If the target value is not reached during part **P.1**, the smallest UTXOs are selected one-by-one and added to S^{greedy} until the remaining value *rem* is non-positive.

The algorithm is described in Algorithm 1.

Note that the greedy algorithm cannot always minimize the number of inputs. We show this in the following example.

Example 1. *Let UTXO pool* $U = \{0.5, 0.4, 0.4, 0.2, 0.1\}$ *and target* $T = 0.8$. *The greedy algorithm will find the solution* $S^{Greedy} = \{0.5, 0.2, 0.1\}$, *whereas the optimal solution that minimizes the number of inputs is* $S^{opt} = \{0.4, 0.4\}$.

B. Randomized Adaptive Greedy

For the randomized version of the greedy algorithms we first note that we do not require the UTXO pool to be ordered.

[1]To be strict, $v(S^{greedy})$ should be sufficient to also cover the transaction fee.

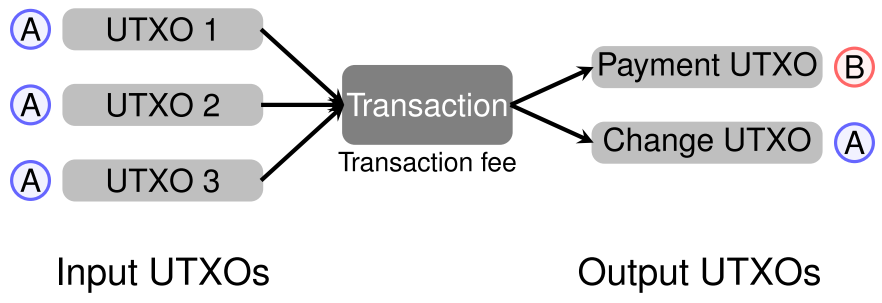


Fig. 2: Example of a transaction structure. User A sends a payment to user B. The payment UTXO goes to user B and the change UTXO is added to user A's UTXO pool. The three input UTXOs are all removed from user A's UTXO pool. Note that we have the following equality in this example: *UTXO 1 + UTXO 2 +UTXO 3 = Payment UTXO + Change UTXO + Transaction fee*.

1) Randomness: As a first simple modification, we introduce some randomness. In particular, in Part **P.1** in line 4 of Algorithm 1 we add a random bool generation function `randBool()` which returns *true* or *false* with equal probability. This means, that a UTXO that would have been chosen under the standard greedy algorithm is now chosen with a probability of 0.5 only. The second part remains the same as in **P.2**, i.e. if the selected subset does not suffice to pay the target, the smallest remaining UTXOs are selected one-by-one until the target can be paid. We call this random greedy algorithm $\mathcal{RG}(\cdot,\cdot)$.

2) Adaptiveness: We further modify the second part of algorithm $\mathcal{RG}(\cdot,\cdot)$ by introducing a new variable $R > 0$ that denotes the target size of the UTXO pool. In particular, assume that the first part of algorithm $\mathcal{RG}(\cdot,\cdot)$ selectes a subset $S \subseteq U$ with $v(S) < T$. Then the remaining value is $rem = T - v(S)$. Now, instead of adding the smallest remaining UTXOs one-by-one until rem is covered, we demand more. We recalculate rem as follows:

$$rem \leftarrow \lfloor (1 + |U|/R) \rfloor \cdot rem,$$

that is, we adapt the remaining value depending on the *current* UTXO pool size. Specifically, if the size of the UTXO pool is larger than the target size R, rem is increased accordingly. This has the following effect: As the remaining value is larger, more UTXOs have to be selected to meet this value. This ultimately leads to shrinking the UTXO pool. If the size of the UTXO pool is below the target size, rem is not scaled and stays as in $RG(\cdot,\cdot)$.

The random adaptive greedy algorithms is described in Algorithm 2.

Algorithm 2 Randomized Adaptive Greedy — $\mathcal{RAG}(U, T)$

```
Input: UTXO pool $U$
Input: Target payment $T > 0$
Output: Set of selected UTXOs $S^{\text{greedy}}$
Require: $v(U) > T$
 1: $S^{\text{greedy}} \leftarrow \{\}$
 2: $rem \leftarrow T$
 3: $R \leftarrow 20$
 4: for $i = 1$ to $n$ do
 5:     if $u_i \leq rem$ and $rem > 0$ and randBool() then
 6:         $S^{\text{greedy}}$.add($u_i$)
 7:         $rem \leftarrow rem - u_i$
 8:     end if
 9: end for
10: $rem \leftarrow \lfloor (1 + |U|/R) \rfloor \cdot rem$
11: while $rem > 0$ do
12:     $S^{\text{greedy}}$.add(min$\{U \setminus S^{\text{greedy}}\}$)
13:     $rem \leftarrow rem -$ min$\{U \setminus S^{\text{greedy}}\}$
14: end while
15: return $S^{\text{greedy}}$
```

IV. Simulations

For each algorithm $\mathcal{R}, \mathcal{RG}, \mathcal{RAG}$ we start with a UTXO pool consisting of one UTXO with value 100'000. We run 50'000 iterations, 100 times and take the average values. Following [1], [2], [6] to model real world behavior of a user's wallet, in every iteration there is one larger payment and three smaller deposits such that on average the balance stays constant. The payment and deposits are drawn from a Normal distribution, see [1], [6], [7] In each iteration we run the three algorithms and capture the sizes of the UTXO pools and the number of inputs used in each transaction. The simulation procedure is described in Algorithm 3. The `update()` function used in Algorithm 3 is described in Algorithm 4.

In Fig. 3 we see the average evolution of the UTXO pool size for the three different algorithms. Notably, the randomized adaptive greedy algorithm maintains the pool size consistently below the designated target of 20. In contrast, both the standard greedy and randomized greedy algorithms exhibit a gradual increase. Our proposed method manages to maintain the pool size at approximately 85% lower than that of the standard greedy algorithm.

In Figure 4, we present the average counts of input UTXOs across all iteration steps. We observe that the frequencies are largely similar, with only minor and negligible differences. Specifically, when comparing the total cumulative count of

Algorithm 3 Simulation Procedure

```
Input: $U^G, U^{RG}, U^{RAG} = \{100'000\}$.
 1: $iterations \leftarrow 50'000$
 2: for $i = 1$ to $iterations$ do
 3:     $T \leftarrow Normal(3000, 500)$
 4:     $U^G$.sortDescending()
 5:     $U^G$.update($\mathcal{G}, U^G, T$)
 6:     $U^{RG}$.update($\mathcal{RG}, U^{RG}, T$)
 7:     $U^{RAG}$.update($\mathcal{RAG}, U^{RAG}, T$)
 8:     for $k = 1$ to 3 do
 9:         $deposit \leftarrow Normal(1000, 250)$
10:         $U^G$.add($deposit$)
11:         $U^{RG}$.add($deposit$)
12:         $U^{RAG}$.add($deposit$)
13:     end for
14: end for
```

Algorithm 4 update($\mathcal{A}$,U,T)

```
Inputs: Algorithm $\mathcal{A}$, UTXO set $U$, Target value $T$
 1: $S \leftarrow \mathcal{A}(U, T)$
 2: $U \leftarrow U \setminus S$
 3: $U$.add($v(S) - T$))
 4: return $U$
```

input UTXOs for each algorithm, we find that the randomized adaptive greedy algorithm utilized approximately 0.07% more input UTXOs on average. While this results in a higher transaction fee, it is a negligible impact when compared to the substantial 85% decrease in the pool size.

Based on our simulations, it appears worthwhile to investigate this specific trade-off. If accomplishing certain objectives entails only a slight increase in transaction fees, it is certainly logical to delve deeper into this direction.

V. Conclusion

We proposed a simple yet effective modification for the standard greedy algorithm that shrinks the UTXO pool size adaptively. This comes at a cost of higher transaction fee. However, as our simulations suggested this increase in transaction fee is very small. Therefore, further research on greedy algorithms can help to improve existing coin selection algorithms implemented in practice.

References

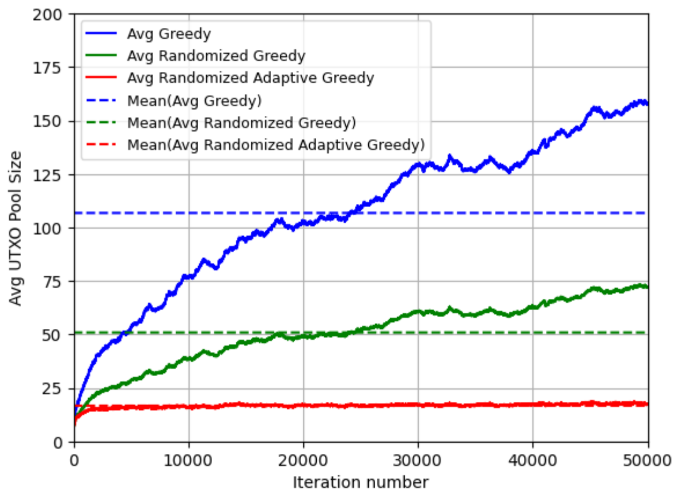
[1] G. Ramezan, M. Schneider, and M. McCann, "A Survey on Coin Selection Algorithms in UTXO-based Blockchains," *arXiv preprint, arXiv:2311.01113*, 2023.

[2] G. Ramezan, M. Schneider, and M. McCann, "MACS: A Multi-Asset Coin Selection Algorithm for UTXO-based Blockchains," *mimeo*, 2023.

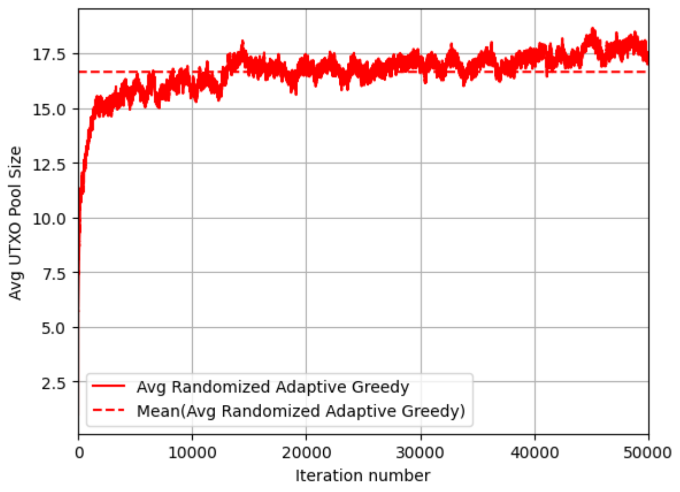
[3] T. A. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, pp. 109–133, 1995.

[4] M. G. Resende and C. C. Ribeiro, *Greedy Randomized Adaptive Search Procedures: Advances, Hybridizations, and Applications*, pp. 283–319. Boston, MA: Springer US, 2010.

[5] X. Wei, C. Wu, H. Yu, S. Liu, and Y. Yuan, "A Coin Selection Strategy Based on the Greedy and Genetic Algorithm," *Complex Intelligent Systems*, vol. 9, pp. 421–434, 2023.



(a) Average UTXO pool size for all three algorithms, including the mean over 50'000 iterations.



(b) Average pool size for the randomized adaptive greedy algorithm.

Fig. 3: Comparison of UTXO pool sizes.

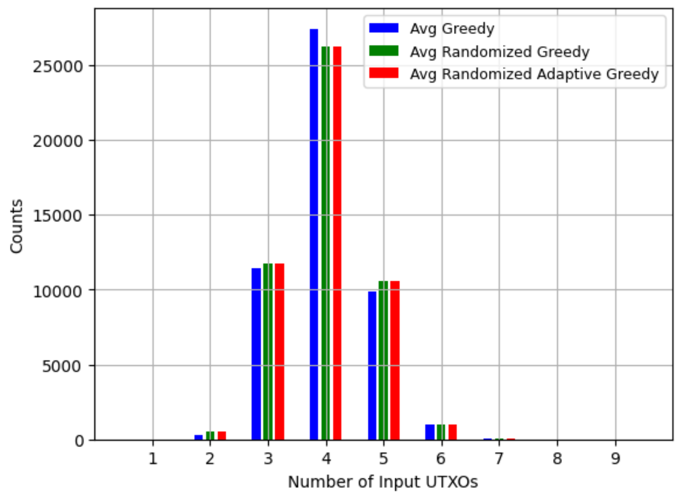


Fig. 4: Frequency of number of input UTXOs.

[6] E. de Vries, “Self Organisation in Coin Selection.” https://iohk.io/en/blog/posts/2018/07/03/self-organisation-in-coin-selection/ (accessed July 25, 2023).

[7] M. Erhardt, “An Evaluation of Coin Selection Strategies.” https://murch.one/wp-content/uploads/2016/11/erhardt2016coinselection.pdf, 2016.