MACS: A Multi-Asset Coin Selection Algorithm for UTXO-based Blockchains

Gholamreza Ramezan, Manvir Schneider, and Mel McCann
Cardano Foundation
Zug, Switzerland
{gholamreza.ramezan, manvir.schneider, mel.mccann}@cardanofoundation.org

***Abstract*—In this paper, we propose *MACS* — a new coin selection algorithm for multi-asset UTXO-based blockchains. MACS addresses the primary coin selection requirements regarding transaction fee, transaction size, UTXO pool size, transaction age, and privacy. We compare the performance of MACS with other coin selection algorithms. The performance evaluation shows that MACS achieves the mentioned requirements while outperforming other coin selection algorithms in terms of having a wider distribution of their number of UTXO inputs. Moreover, UTXOs will not remain in the UTXO pool forever when the MACS algorithm is used since MACS keeps the fraction of old UTXOs oscillating between 0% and 20%.**

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

I. Introduction

In the realm of blockchain, there are two primary ledger classes: Unspent Transaction Outputs (UTXO)-based and account-based ledgers. In UTXO-based ledgers, as exemplified by Bitcoin, the blockchain's state is stored within UTXOs. Blockchain transactions alter the state by consuming specific UTXOs and creating new ones. Each UTXO is associated with a blockchain user's wallet. To calculate the total tokens of a user, the wallet software must calculate the balance by summing up all UTXOs linked to that user. On the contrary, account-based ledgers, as seen in Ethereum, assign each user a dedicated account. Transactions in this system primarily increase or decrease the token balance in a user's account by transferring tokens to other users. Hence, the user wallet software simply shows the user's account balance without the need to calculate the balance. Although calculating the user balance by summing tokens in the user's UTXOs may appear more complicated compared with account-based ledgers, UTXO-based ledgers offer distinct advantages, including the predictability of the results of UTXOs processing and the facilitation of parallel transaction processing.

In this paper, we focus on UTXO-based blockchains. Every blockchain has its own native token that is used to pay the transaction fee and reward block producers when generating a new block. Additionally, blockchains have introduced features to allow their users to create new types of tokens, called *user tokens*. Hence, today's blockchains are considered multi-asset blockchains, as transactions may involve the native token as well as several user tokens.

The multi-asset feature of the UTXO-based blockchain introduces new challenges in applications such as wallet software. Before creating a transaction, the software must select a set of UTXOs as transaction inputs. The algorithm to make this selection is called the *coin selection algorithm*. This algorithm has to consider several factors, including different user tokens, privacy of users, etc.

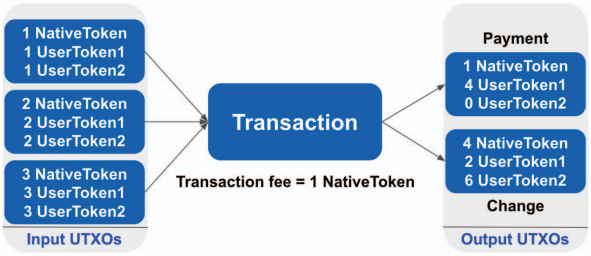


Fig. 1: Transaction

In Ethereum, as an account-based blockchain, to create a new type of user token, a new smart contract should be generated. Therefore, the security of the user tokens in Ethereum is at the level of the smart contract programmability. On the other hand, the user-defined tokens in UTXO-based blockchains, like Cardano, do not require smart contracts. They can be created through an internal feature of the blockchain. Hence, they have the same level of security at the blockchain native token. This capability allows UTXOs to carry multiple user-defined tokens as well as the native token of the blockchain. It is important to note that user-defined tokens in a UTXO cannot be used for transaction fee payments; only the blockchain's native asset can serve that purpose. Fig. 1 illustrates an example of a transaction on a UTXO-based blockchain involving two user-defined tokens, as well as the blockchain's native asset.

In the past, various coin selection algorithms have been studied and implemented [1]–[8] and [9]. Examples of these algorithms include Highest Value First (HVF), Lowest Value First (LVF), Highest Priority First, Greedy, First In First Out, Last In First Out, and slightly more advanced algorithms like Random Draw, Random Improve, Branch and Bound (BnB), and Knapsack. Each algorithm has its own set of advantages and disadvantages. A comprehensive review of existing coin selection algorithms is given in [10].

In this paper, we consider multi-asset blockchains with different user tokens on a UTXO-based blockchain and propose a coin selection algorithm called MACS to satisfy the different

2834-9946/23/$31.00 ©2023 IEEE
DOI 10.1109/Blockchain60715.2023.00029

objectives (see Section III-A) of the coin selection algorithm. The main contributions of this paper are as follows.

- We formulate an optimization problem that aims to jointly minimize transaction fees, transaction sizes, and the size of the UTXO pool, as well as preserve the privacy of the user.
- We solve the formulated problem and propose a multi-asset coin-selection algorithm called MACS.
- We compare the performance of the proposed algorithm with five algorithms, including HVF, LVF, Random Draw, Greedy and BnB.

This paper is structured as follows. In Section 2, we review the essential components required for coin selection algorithms. Section 3 will introduce the proposed coin selection algorithm. In Section 4, we perform a performance analysis and compare MACS to other coin selection algorithms. Section 5 concludes.

II. Methdology and Definitions

UTXO is a fundamental concept in blockchain. UTXOs are transaction outputs that have been received but have not yet been spent. As in Fig. 1 depicts, every transaction has some UTXOs as input and generates new UTXOs as output. The output consists of the payment UTXO that transfers the tokens to the target users or recipients, and, the change UTXO which returns the remaining tokens to the sender user's wallet.

Let $\mathcal{C} = \{c^*, c', c'', \ldots\}$ be the set of all assets consisting of the native asset c^* and other user-defined assets.

Definition 1 (UTXO (Unspent Transaction Output))**.** A UTXO is a set of tuples $(value, asset) \in \mathbb{Q} \times \mathcal{C}$ describing the values of assets.

For a UTXO u, $\mathcal{C}_u$ is the set of all assets in UTXO u, $\mathcal{C}_u \subseteq \mathcal{C}$ and $v(u, c)$ is the value of the asset c in UTXO u. Also, define $v(u, c) = 0$ if $c \notin \mathcal{C}_u$. Note that, by definition, for any UTXO u, $c^* \in \mathcal{C}_u$ and $v(u, c^*) > 0$. That is, each UTXOs needs to carry a non-zero amount of the native asset c^*. Furthermore, for any set of UTXOs U, $\mathcal{C}_U$ is the set of all unique assets in UTXOs of S, that is, $\mathcal{C}_U = \bigcup_{u \in U} \mathcal{C}_u$.

Let T be a set of pairs $(value, asset) \in \mathbb{Q} \times \mathcal{C}$. T is called the *target* (or payment). The studied problem investigates transactions where a certain target has to be paid and chooses the optimal set of UTXOs to pay the target. Also, $\mathcal{C}_T$ is the set of all unique assets in target T and $v(T, c)$ is value of asset c in target T for $c \in \mathcal{C}$.

Definition 2 (UTXO Size)**.** The size of a UTXO u is given by

$$size(u) := \sum_{c \in \mathcal{C}_u} \left\lceil \frac{v(u,c)}{K_0} \right\rceil + len(c) I^{BytesPerChar}, \quad (1)$$

for some given constant $K_0 > 0$ and $len(c)$ returning the character length of c. $I^{BytesPerChar}$ is the number of bytes needed to store one character.

By setting $K_0 = 256$, $\left\lceil \frac{v(u,c)}{K_0} \right\rceil$ is the number of bytes required to save the value $v(u, c)$.

Example 1. *Let $c^*, c', c'' \in \mathcal{C}$ and let $u = \{(a, c^*), (b, c'')\}$ be a UTXO with $a, b > 0$. Then, $\mathcal{C}_u = \{c^*, c''\}$, $v(u, c^*) = a$ and $v(u, c'') = b$. Also, $v(u, c') = 0$ since $c' \notin \mathcal{C}_u$.*

We present our algorithm for a multi-asset blockchain with different types of tokenized assets. Consequently, each UTXO can possess a variable number of tokens associated with each specific asset type. One token in the asset list represents the blockchain's native asset, and the rest are various user-defined assets.

Definition 3 (Transaction)**.** A transaction $\mathcal{T} = (S_I, S_P, S_C, T, TrFee(\mathcal{T}))$ is a quintuple consisting of the set of input UTXOs S_I, the sets of output UTXOs — the payment UTXO set S_P and the change UTXO set S_C — the target T and the transaction fee $TrFee(\mathcal{T})$. The transaction fee $TrFee(\mathcal{T}) = \{(\phi_{\mathcal{T}}, c^*)\}$ is defined as follows:

$$\phi_{\mathcal{T}} = size(\mathcal{T}) I^{FeeRate} + I^{FeeMin}, \quad (2)$$

where

$$size(\mathcal{T}) = \sum_{u \in S_I \cup S_P \cup S_C} size(u) + size(TrFee(\mathcal{T})). \quad (3)$$

Also, $size(TrFee(\mathcal{T}))$ is size of the transaction fee. $I^{FeeRate}$ reflects the dependence of the transaction fee on the size of the transaction [11] and I^{FeeMin} is a fee payable regardless of the size of the transaction. This value is used to prevent an attack on the blockchain, as described in [11]. Note that the calculation of the transaction fee, in particular, requires a fixed point calculation. Note that by definition, the calculation of the transaction fee requires a fixed-point calculation. For simulations we will relax this assumption and assume that the transaction fee is linear in the number of input UTXOs and has negligible size.

Next, we introduce the notion of a balanced transaction which postulates that the sum of the input values must be equal to the sum of the output values.

Definition 4 (Balanced Transaction)**.** A transaction $\mathcal{T} = (S_I, S_P, S_C, T, TrFee(\mathcal{T}))$ is said to be a *balanced transaction* iff for all $c \in \mathcal{C}_{S_I}$,

$$\sum_{u \in S_I} v(u, c) = \sum_{o \in S_P \cup S_C} v(o, c) + \mathbb{1}_{\{c = c^*\}} \phi_{\mathcal{T}}. \quad (4)$$

Definition 5 (Valid-Value Transaction)**.** A transaction $\mathcal{T} = (S_I, S_P, S_C, T, TrFee(\mathcal{T}))$ is said to be a *valid-value transaction* iff for all $c \in \mathcal{C}_T \setminus \{c^*\}$

$$\sum_{u \in S_I} v(u, c) \geq v(T, c), \quad (5)$$

$$\sum_{o \in S_P} v(o, c) = v(T, c), \quad (6)$$

and for the native asset c^*,

$$\sum_{u\in S_I} v(u,c^*) \geq \max\left\{v(T,c^*), \sum_{o\in S_P\cup S_C} Min_{c^*}(o)\right\} + \phi_{\mathcal{T}}, \tag{7}$$

$$\sum_{o\in S_P} v(o,c^*) = \max\{v(T,c^*), \sum_{o\in S_P} Min_{c^*}(o)\}, \tag{8}$$

where $Min_{c^*}(o)$ is minimum required value of the native asset in UTXO o which is defined as follows for constants $K_1, K_2 > 0$:

$$Min_{c^*}(o) = K_1 + size(o)K_2 \tag{9}$$

Note that the definition of equation (9) also requires a fixed point calculation, similar to the transaction fee. The reason is that the minimum value depends on the size of the UTXO, whereas the size of the UTXO itself depends on the values. Hence, if adding the minimum amount of native asset to the UTXO does not change the size of the UTXO, we arrive at a fixed point. Again, as for the transaction fee, we will relax this assumption and rather consider a fixed value for the minimum amount of the native asset per UTXO.

Definition 6 (Valid-Size Transaction)**.** A transaction $\mathcal{T} = (S_I, S_P, S_C, T, TrFee(\mathcal{T}))$ is a *valid-size transaction* iff

$$\sum_{o\in S_P\cup S_C} size(o) \leq I^{MaxTrOutputSize}, \tag{10}$$

and

$$size(\mathcal{T}) \leq I^{MaxTrSize} \tag{11}$$

where $I^{MaxTrSize}$ is the maximum size of a transaction, $I^{MaxTrOutputSize}$ is the maximum output size, and $size(o)$ and $size(\mathcal{T})$ are given by equations (1) and (3), respectively.

Definition 7 (Valid-Value UTXO)**.** For any UTXO u, and any $c \in \mathcal{C}_u$, we define

$$range(u,c) = \begin{cases} [Min_{c^*}(u), v(I^{MaxVal}, c)] & c = c^*, \\ [0, v(I^{MaxVal}, c)] & c \neq c^*, \end{cases} \tag{12}$$

where $v(I^{MaxVal}, c)$ is the maximum possible value for each asset c in a UTXO. A UTXO u is called a *valid-value* UTXO iff for all $c \in \mathcal{C}_u$, $v(u,c) \in range(u,c)$. Also, a set of UTXOs is called *valid-value* iff all UTXOs in the set are *valid-value* UTXOs.

III. Problem Formulation

We first introduce the requirements of the coin selection algorithm. Then, we formulate our proposed optimization problem.

A. Coin Selection Algorithm Requirements

We can outline five primary requirements for a coin selection algorithm, representing the desired properties that such a method should fulfill, as discussed in [10]:

- Minimize transaction fees by selecting UTXOs as inputs, taking into consideration that the transaction fee depends on the transaction's size, including the number of input and output UTXOs.
- Enhance privacy by minimizing the selection of input UTXOs originating from the same address.
- Minimize the size of the UTXO pool by avoiding the accumulation of dust UTXOs. Some coin selection methods can lead to a significant increase in the size of the UTXO pool [12], [10], which can cause issues, such as the inability to send assets. This occurs when a wallet has sufficient total value to spend for a specific target value but in the form of numerous dust UTXOs. However, due to block size limitations, there is an upper bound on transaction size, limiting the number of input UTXOs that can be included in a transaction to spend a specific target.
- Ensure that the transaction offers sufficient transaction fees to have a high probability of being included in one of the next blocks.
- Promote diversity in the UTXO pool with respect to UTXO values to enable spending a wider range of values.

It should be noted that the performance and speed of the coin selection method are also of significant importance. However, for the scope of this paper, we primarily focus on the five requirements mentioned above.

B. Problem Formulation

To start formulating the problem, we define UTXO priority as follows:

Definition 8 (Priority of UTXO)**.** Given a set of UTXOs S, the priority of UTXO $u \in S$ for asset c is calculated as follows:

$$P(u,c) = \frac{v(u,c)\text{conf}(u)}{\big(|\text{link}(u)|+1\big)\big(|v(u,c) - \text{avg}(S,c)|+1\big)}, \tag{13}$$

where $\text{conf}(u)$ is the age (confirmation count) of the UTXO u, $\text{link}(u)$ is the set of UTXOs that are linked to the UTXO u. The mean of the UTXOs' values of asset c in set S is denoted by $\text{avg}(S,c) := \frac{1}{|S|}\sum_{u\in S} v(u,c)$. The second term in the denominator shows the deviation of the UTXO values from their average. In summary, we use (13) to prioritize different UTXOs based on their value, age, links to previous UTXOs, and their deviation from the average.

Example 2. *Let's assume there are* 20 *UTXOs in the pool. We denote* $U = \{u_1, ..., u_{20}\}$*, Also, let* $\mathcal{C}_U = \{$*NativeToken*, *UserToken1*, *UserToken2*$\}$*. We assume* $v(u_i,$ *UserToken2*$) = i$ *for* $i \in [20]$*. Furthermore, assume that for each* $u \in U$, $|$*link*$(u)| = 0$ *and* *conf*$(u) = 1$*. Then, the priorities are as in Fig. 2. In particular,* $P(u_{11},$ *UserToken2*$) > P(u_{10},$ *UserToken2*$) > P(u_{12},$ *UserToken2*$) > P(u_9,$ *UserToken2*$) > ... > P(u_2,$ *UserToken2*$) > P(u_1,$ *UserToken2*$)$*. Additionally, note that being closer to the mean has a greater impact on the priority than having a high UTXO value.*

We now define the main components of our problem-formulation objectives. Considering the requirement to design

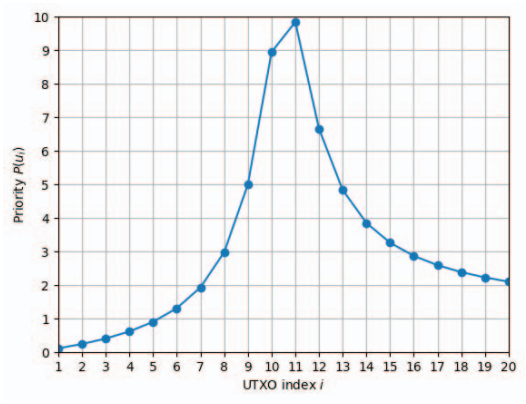


Fig. 2: Priorities.

a coin selection algorithm, we first, sum up the priorities of the input UTXOs and form the first objective as follows:

$$O_1 = \sum_{u \in S_I} P(u, c)$$

Second, we define

$$M(\mathcal{T}, c^*) := \max\{v(T, c^*), \sum_{o \in S_P \cup S_C} Min_{c^*}(o)\} + \phi_{\mathcal{T}}.$$

which is the maximum of the target value and the total minimum number of native tokens in output UTXOs plus the required transaction fee value. Then, the second objective is

$$O_2 = \frac{\mathbb{1}_{\{c \neq c^*\}} v(T, c) + \mathbb{1}_{\{c = c^*\}} M(\mathcal{T}, c)}{1 + \sum_{u \in S_I} v(u, c) - \mathbb{1}_{\{c \neq c^*\}} v(T, c) - \mathbb{1}_{\{c = c^*\}} M(\mathcal{T}, c)}$$

that shows the ratio between the target value and the change value. Third, we use the number of input and output UTXOs, and define the third objective.

$$O_3 = \frac{1}{|S_P| + |S_C| + |S_I|}$$

Fourth, we define

$$V(S) := \frac{\sqrt{\sum_{o \in S} (v(o, c) - \text{avg}(S, c))^2}}{(|S| + 1)(\text{avg}(S, c) + 1)}$$

where $S \in \{S_P, S_C\}$. The fourth objective is the dispersion of the change and payment values defined below:

$$O_4 = V(S_P) + V(S_C)$$

Now we formulate the optimization problem called MACS whose objective is the weighted sum of the objectives mentioned above. Given a target T and an initial set of UTXO U, we have

$$\max_{\mathcal{T}} \quad \sum_{c \in \mathcal{C}_T \cup \{c^*\}} \lambda_1 O_1 + \lambda_2 O_2 + \lambda_3 O_3 + \lambda_4 O_4 \tag{14a}$$

$$\text{s.t.} \quad \mathcal{T} \text{ is a balanced transaction} \tag{14b}$$

$$\mathcal{T} \text{ is a valid-value transaction} \tag{14c}$$

$$\mathcal{T} \text{ is a valid-size transaction} \tag{14d}$$

$$S_P \text{ is a valid-value UTXO set} \tag{14e}$$

$$S_C \text{ is a valid-value UTXO set} \tag{14f}$$

$$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1 \tag{14g}$$

$$S_I \subseteq U, \quad 0 \leq \phi_{\mathcal{T}}, \quad \lambda_1, \lambda_2, \lambda_3, \lambda_4 \in [0, 1] \tag{14h}$$

where $\mathcal{T} = (S_I, S_P, S_C, TrFee(\mathcal{T}))$. Also, λ_1,λ_2,λ_3, and λ_4 represent the importance of each objective.

To solve the proposed optimization problem, we use the GEKKO optimization suite [13] which allows us to tackle a mixed-integer non-linear program (MINLP). Specifically, we employ steady-state optimization with the APOPT (Advanced Process OPTimizer) solver package. We evaluate the performance in the next section.

IV. Performance Evaluation

In this section, we study the performance of MACS and compare it with other coin selection algorithms including *Highest-Value-First (HVF), Lowest-Value-First (LVF), Random Draw (RndDrw), Greedy* and *Branch-and-Bound (BnB)*[1]. For all the latter mentioned established algorithms, we need an extension for multiple assets. In particular, all algorithms are performed asset-by-asset, that is, UTXOs are selected to meet the target in the native asset first and then successively for the following user-defined assets.

We consider the setting of two user-defined tokens *UserToken1* and *UserToken2*. That is, in total we have three assets — the native asset and two user-defined assets. We start with a UTXO pool consisting of three UTXOs and a total balance of 100,000 in each asset. In particular, the first UTXO has 80,000 native tokens, the second UTXO has 10,000 native tokens and 100,000 *UserToken1*, and the third UTXO has also 10,000 native tokens and 100,000 *UserToken2*. We run 1,000 iterations where in each iteration UTXOs are added and removed from the UTXO pool.

To model real-world behavior, we assume that in every iteration one larger target is paid and three smaller UTXOs (deposits) are added to the UTXO pool. This particular choice target-deposit ratio is due to [10], [12]. Each target consists of three values — one for each asset. The target in each asset is drawn from a Poisson distribution with a mean of 3,000 and each deposit consists of three randomly drawn values from a Poisson distribution with a mean of 1000 [10].

We consider that only one change and one payment UTXO are created. The optimization problem involves selecting the optimal input UTXOs to pay a given target. We set $\lambda_1 = \lambda_2 =$

[1]BnB is one of the better performing existing algorithms [7], [10].

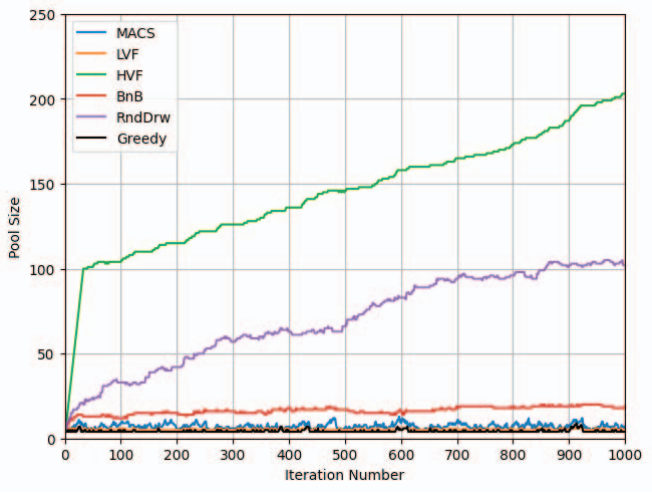
$\lambda_3 = \lambda_4 = 0.25$. Furthermore, we assume that the transaction fee is the number of input UTXOs divided by two, which is a linear function. The minimum native token amount for a UTXO is assumed to be fixed and set to 1. The number of links associated with a change in UTXO is the sum of the links of the input UTXOs and the age is set to zero. Deposit UTXOs are added to the pool with one link each and an age of zero. The parameters $I^{MaxTrOutputSize}$ and $I^{MaxTrSize}$ are set such that no transaction will exceed the numbers in the simulations. Similarly, targets are drawn such that I^{MaxVal} for each asset $c \in \mathcal{C}$ has no impact on UTXOs, i.e. no UTXO will exceed the upper bound.

In Fig. 3, we illustrate the evolution of the UTXO pool sizes, i.e. the number of UTXOs in the UTXO pool after each iteration for the different methods. In Fig. 3a, we observe that the pool size explodes with HVF. The same holds true for RndDrw but with a weaker increase than HVF. MACS outperforms HVF, RndDrw, and BnB in this regard, and performs similar to LVF and Greedy. In Fig. 3b, we compare only MACS, LVF, BnB and Greedy. To better illustrate MACS we use a median filter and illustrate this as a smoothed graph. For the median filter we divide the total number of iterations into equally sized batches of size 15 and calculate the median over each batch. We observe that the pool size under MACS stays under nine for the majority of iterations and also goes below six. The lower bound of four for all algorithms is given by the change and the three deposits that are added to the pool in every iteration. The constant periods in BnB, LVF, and Greedy are due to the fact that the number of input UTXOs in consecutive iterations is the same. Fig. 4 undermines this observation as the majority of transactions has an input of four UTXOs for all methods, except MACS and therefore multiple consecutive transaction will have the same number of input UTXOs, keeping the pool size constant.

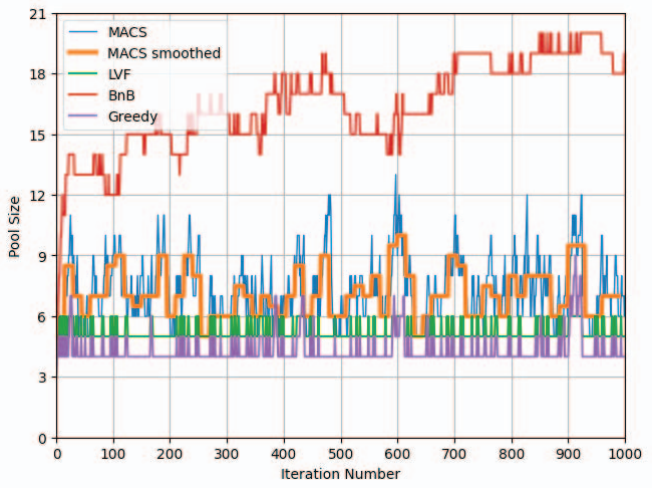
In Fig. 4, we illustrate the number of input UTXOs and their frequency. We observe that all algorithms, except MACS, have exactly four input UTXOs in the majority of transactions, i.e. around 800 times. However, our proposed MACS algorithm has a much wider distribution of input UTXOs.

In Fig. 5, we illustrate the fraction of UTXOs with an age larger than ten.[2] In particular, we increase the age of a UTXO by one in each iteration whenever the UTXO remains in the pool. In Fig. 5a, we display all algorithms. We observe that MACS performs very good, that is, the fraction of UTXOs with an age larger than ten moves between 0% and 20%. Specifically, this means that all UTXOs will eventually be chosen as input, and no UTXO will remain in the pool for too long periods. The Greedy method performs well but has several spikes where the fraction of old UTXOs reaches close to 40%. The other methods clearly lead to many "old" UTXOs being in the pool, which is an undesired property. Such as, LVF which keeps the fraction approximately constant and equal to 20%. In Fig. 5b, we apply a median filter to MACS and BnB to

[2]We can consider any threshold to compare the age. In this case, we chose 1/100 of the total iteration steps.



(a) Comparison of all algorithms.



(b) Comparison without HVF and RndDrw.

Fig. 3: Pool Size.

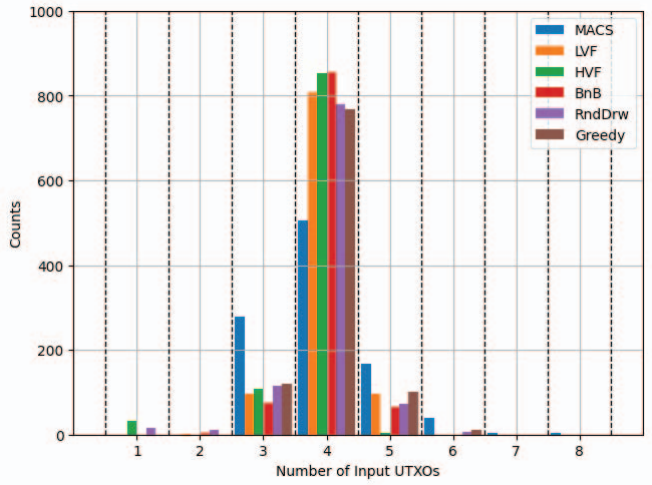
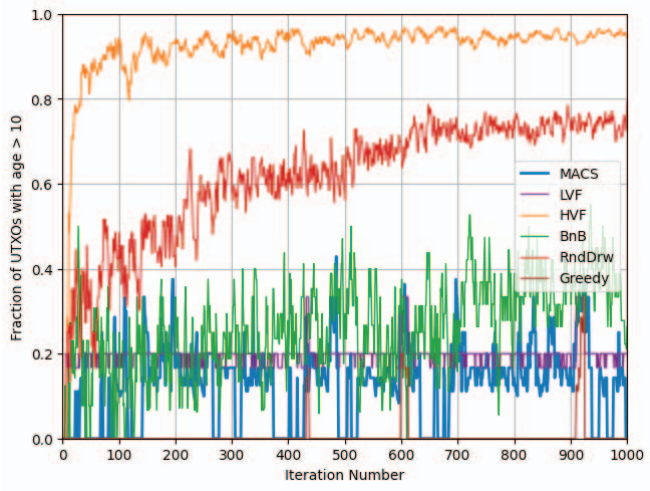
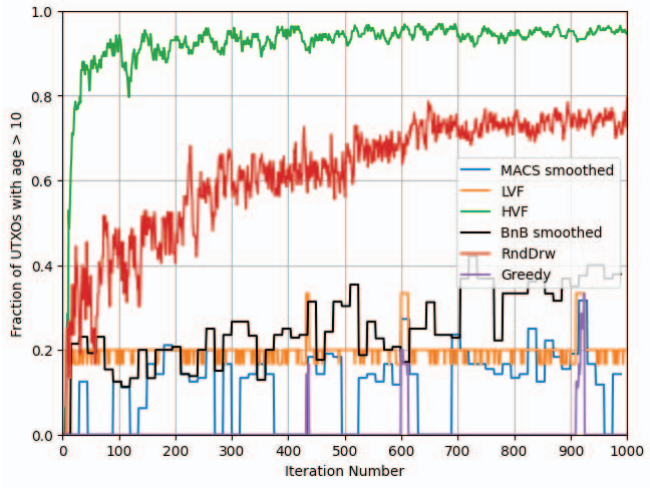


Fig. 4: Number of Input UTXOs.



(a) Comparison of all algorithms.



(b) Smoothed graphs for MACS and BnB.

Fig. 5: Fraction of UTXOs with age larger than 10.

reduce the variance. In particular, we divide the total number of iterations into equally sized batches of size 15 and take the median over each batch. The smoothed graphs are displayed in Fig. 5b.

We summarize that MACS fulfills the desired properties (see Section III-B) and outperforms other coin selection algorithms. Next, we provide a more detailed summary of the simulation procedure.

Details of the Simulation

For every analyzed algorithm we start with a UTXO pool consisting of three UTXOs with a total balance of 100,000 in each asset—the native asset c^*, UserToken1 c_1 and UserToken2 c_2. In particular, the three UTXOs u_1, u_2, u_3 are defined as $u_1 = \{(80,000, c^*)\}$, $u_2 = \{(10,000, c^*), (100,000, c_1)\}$ and $u_3 = \{(10,000, c^*), (100,000, c_2)\}$. For every studied algorithm $alg \in \{$MACS, LVF, HVF, BnB, RndDrw, Greedy$\}$, we have an initial UTXO pool $U^{alg} = \{u_1, u_2, u_3\}$.

In every iteration we do the following steps:

1) Generate a target (T_1, T_2, T_3) where T_1 is the target for the native asset and the other two are targets for the user-defined tokens. For $i = 1, 2, 3$, $T_i \sim Poisson(3000)$.
2) For every coin selection algorithm alg, choose the input UTXOs set to pay the target (T_1, T_2, T_3), remove the chosen UTXOs from U^{alg} and add the change UTXO.
3) Generate deposits UTXOs $D_j = (d_1^j, d_2^j, d_3^j)$ for $j = 1, 2, 3$ where each $d_i^j \sim Poisson(1000)$ for $i = 1, 2, 3$ and add the three deposit UTXOs to every pool U^{alg}.

V. Conclusion

We proposed a multi-asset coin selection algorithm called MACS for UTXO-based blockchains. The proposed method covers the major requirements for coin selection algorithms. The performance evaluation showed that MACS outperforms other coin selection algorithms. The generated transactions using MACS have a wider distribution in terms of their number of UTXO inputs. Moreover, UTXOs will not remain in the UTXO pool forever when the MACS algorithm is used.

Acknowledgment

We would like to thank Jonathan Knowles for helpful comments.

References

[1] "Github Bitcoin Core — Bitcoin Coin Selection Code." https://github.com/bitcoin/bitcoin/blob/74f1a27f2f45af7dafcc34df766cf76d29c7c6ed/src/wallet/coinselection.cpp#L1 (accessed July 25, 2023).
[2] "Greedy Algorithm." https://www.log2base2.com/algorithms/greedy/greedy-algorithm.html (accessed July 25, 2023).
[3] S. Abramova and R. Böhme, "Your Money or Your Privacy: A Systematic Approach to Coin Selection," *Cryptoeconomic Systems*, 2020.
[4] Cardano Foundation, "CIP 2 - Coin Selection Algorithms for Cardano." https://cips.cardano.org/cips/cip2/(accessed July 25, 2023).
[5] M. M. T. Chakravarty, J. Chapman, K. MacKenzie, O. Melkonian, M. Peyton Jones, and P. Wadler, "The Extended UTXO Model," in *Financial Cryptography and Data Security* (M. Bernhard, A. Bracciali, L. J. Camp, S. Matsuo, A. Maurushat, P. B. Rønne, and M. Sala, eds.), (Cham), pp. 525–539, Springer, 2020.
[6] D. J. Diroff, "Bitcoin Coin Selection with Leverage," *arXiv preprint, arXiv:1911.01330*, 2019.
[7] M. Erhardt, "An Evaluation of Coin Selection Strategies." https://murch.one/wp-content/uploads/2016/11/erhardt2016coinselection.pdf, 2016.
[8] V.-H. Nguyen, H.-S. Trang, Q.-T. Nguyen, N. Huynh-Tuong, and T.-V. Le, "Building mathematical models applied to utxos selection for objective transactions," in *2018 5th NAFOSTED Conference on Information and Computer Science (NICS)*, pp. 160–164, 2018.
[9] 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.
[10] G. Ramezan, M. Schneider, and M. McCann, "A Survey on Coin Selection Algorithms in UTXO-based Blockchains," *arXiv preprint, arXiv:2311.01113*, 2023.
[11] "Cardano Fee Structure." https://docs.cardano.org/explore-cardano/fee-structure/ (accessed July 25, 2023).
[12] 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).
[13] L. Beal and J. Hedengren, "GEKKO Optimization Suite." https://gekko.readthedocs.io/en/latest/ (accessed September 20, 2023).