VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MATHEMATICAL MODELING

Assignment

Modeling The Coin Selection Problem In Blockchain Technology

Lecturer: Huynh Tuong Nguyen
CO2011 CCO1, Group 4
Students: Vo Dong Ho - 1752219
Mai Trong Nhan - 1752390
Nguyen Minh Nhat - 1752039
Nguyen Minh Quan - 1752450
Huynh Gia An Tien - 1752538
Pham Minh Tuan - 1752595

HO CHI MINH CITY, April 2019

Contents

1 Introduction

1.1 Bitcoin and Blockchain

- **Decentralized System**

  The idea behind cryptocurrencies is a decentralized system for transactions. Instead of a traditional client-server network, cryptocurrencies like Bitcoin utilize a peer-to-peer network in which every machine on the network, called node, has the same copy of the history of transactions on the network.

- **Blocks and Blockchain**

  The history of transactions is stored in a system called **blockchain**. On this "chain", each block stores a set of transactions, together with the hash of the previous block. A transaction is said to be confirmed only when a block containing it is put onto the chain.

- **Miners and Transaction Confirmation**

  On a network of cryptocurrency, there needs to be some kind of mechanism to confirm transactions by putting them in a new block on the blockchain. The nodes that are responsible for transaction confirmation are called **miners**.

  To be able to create a new block on the blockchain, a miner needs to participate in a race with other miners on the network to find a solution for a "puzzle". This kind of puzzle is somewhat like lottery tickets. The more computational power a miner has, the higher chance it will be the first one to solve the puzzle - just like the more lottery tickets one buys, the higher chance he or she will win the lottery.

  Because the confirmation mechanism is based on computational work, it is super hard for a miner to create a fraudulent transaction, unless it has more than half of the computational power of all miners on the network combined.

  After a successful confirmation, a miner will receive bounty from two different sources:

  - **Transaction fee**: each transaction has an amount of transaction fee to attract miners to include it into a new block
  - **Coin base**: the network automatically includes an amount of currency called "coin base" as a prize for the miner.

- **Bitcoin Transactions and UTXOs**

  Some cryptocurrencies, such as Bitcoin, use UTXOs (which stands for Unspent Transaction Outputs) as "inputs" of transactions. We can think of UTXOs as banknotes. When one buys a product in real life, he or she pays with banknotes and then gets an amount of change if the paid amount exceeds the price of the product. It is the same for UTXOs: a transaction takes some UTXOs as inputs, produces some outputs and also a change output if required.

- **Transaction Size and Transaction Fee**

  On the current Bitcoin network, an input has a constant size of 148 bits and an output has a constant size of 34 bits. The size of a transaction is the size of all inputs and outputs combined. At a moment, the transaction fee can be computed with the following formula:

$$\text{Fee} \approx \text{Fee rate} \times [148 * \#\text{ inputs} + 34 * \#\text{ outputs}]$$

- **Dust**

  Roughly speaking, dust outputs are transaction outputs that are so small that the fee to retrieve it is even larger than the value of itself. The network would try to prevent the creation of this type of outputs by including dust output into the transaction fee.

1.2 The Coin Selection Problem

To understand the Coin Selection problem, we need to understand two objectives:

- Minimizing the transaction fee
- Shrinking the UTXO pool

1.2.1 Minimizing the Transaction Fee

Minimizing the transaction fee is what a user wants. In the formula of transaction fee, fee rate is constant at a point in time, and the number of outputs depends on the decision of the user for a particular transaction. What is left is the number of inputs. The larger the number of inputs is, the higher the fee becomes. Therefore, a user would want to use the least number of inputs possible for a transaction.

1.2.2 Shrinking the UTXO pool

It is important to keep the UTXO pool relatively small for performance purpose: the larger the UTXO pool is, the more it hampers processing speed and consumes memory. To shrink the UTXO pool, there are two things we can do:

- Prevent the creation of change output
- Select the largest number of inputs possible for one transaction

As we can see, the two objectives somewhat contradict each other: for the previous objective, we want to select as few inputs as possible, while for this objective, we want to do the opposite. Therefore, we always have some kind of trade-off between the two objectives.

Our project try to incorporate these two objectives through two mathematical models solved by linear optimization.

2 Problem Statement and Modeling

2.1 Problem Statement

Given a transaction with:

- A set of inputs: each input has a certain value and a size of 148 bits.
- A list of outputs: each output has a certain value and a size of 34 bits.
- Transaction fee rate
- Dust threshold: any output with the value lower than this threshold is considered dust.

Find a way to select inputs such that:

- The transaction size is as low as possible
- The number of selected inputs is as large as possible.

2.2 Model 1: Minimizing transaction size

This part is also included as a Jupyter notebook in the source code

2.2.1 Problem Modelling

- **Input parameters**

Variable	Meaning
$U = \{u_1, \ldots, u_n\}$	set of UTXOs
$O = \{o_1, \ldots, o_m\}$	set of outputs
$V^u = \{v_1^u, \ldots, v_n^u\}$	set of value of UTXOs
$V^o = \{v_1^o, \ldots, v_m^o\}$	set of value of outputs
$S^u = \{s_1^u, \ldots, s_n^u\}$	set of size of UTXOs
$S^o = \{s_1^o, \ldots, s_m^o\}$	set of size of outputs
M_s	maximum size of a transaction
α	fee rate
T	dust threshold
ϵ	minimum change output

- **Decision variables**

$$x_i = \begin{cases} 1, \text{if UTXO } u_i \text{ is chosen} \\ 0, \text{otherwise} \end{cases}$$

- **Immediate variables**:
  - y: transaction size
  - z_v: change output
  - z_s: change value
- The relationship between change output value z_v and its size z_s:

$$z_s = \begin{cases} 0, 0 \leq z_v \leq \epsilon \\ \beta, z_v > \epsilon \end{cases}$$

- **Constraint 1**: A transaction size may not exceed maximum block data size.

$$y = \sum_{i|u_i \in U} (s_i^u * x_i) + \sum_{j|o_i \in O} s_j^o + z_s \leq M_s$$

- **Constraint 2**: A transaction must have sufficient value for consuming.

$$\sum_{i|u_i \in U} (v_i^u * x_i) = \sum_{j|o_i \in O} v_j^o + \alpha y + z_v$$

- **Constraint 3**: All the transaction outputs must be higher than the dust threshold to certain that this transaction is relayed to the network and confirmed.

$$\sum_{j|o_i \in O} v_j^o \geq T$$

- **Objective function**: Minimize transaction size

$$min(y)$$

2.2.2 Reformulating

- Define M_c as the maximum value of change.

$$M_c = \sum_{i|u_i \in U} (v_i^u * x_i) - \sum_{j|o_i \in O} v_j^o$$

- Define a binary variable t:

$$t = \begin{cases} 0, \text{if } z_v - \epsilon \leq 0, \text{ or if } z_s = 0 \\ 1, \text{if } z_v - \epsilon > 0, \text{ or if } z_s = \beta \end{cases}$$

- The relationship between t and z_v can be rewritten as a linear inequality:

$$\begin{aligned} & z_v - \epsilon \leq M_c t \\ \Rightarrow & -M_c t + z_v \leq \epsilon \end{aligned}$$

- Substitute t into y:

$$y = \sum_{i|u_i \in U} (s_i^u * x_i) + \sum_{j|o_i \in O} s_j^o + t \times \beta \leq M_s$$

- Substitute t into constraint 1:

$$\begin{aligned} & \sum_{i|u_i \in U} (v_i^u * x_i) = \sum_{j|o_i \in O} v_j^o + \alpha y + z_v \\ \Rightarrow \quad & \sum_{i|u_i \in U} (v_i^u * x_i) = \sum_{j|o_i \in O} v_j^o + \alpha \left[\sum_{i|u_i \in U} (s_i^u * x_i) + \sum_{j|o_i \in O} s_j^o + t \times \beta \right] + z_v \\ \Rightarrow \quad & \sum_{i|u_i \in U} \left[\left(v_i^u - \alpha s_i^u \right) x_i \right] - \alpha \beta t - z_v = \sum_{j|o_i \in O} v_j^o + \alpha \sum_{j|o_i \in O} s_j^o \end{aligned}$$

2.2.3 Solving with Linear Optimization

After reformulating, the model can be solved by linear optimization:

Minimize the objective function:

$$y = \sum_{i|u_i \in U} (s_i^u * x_i) + \sum_{j|o_i \in O} s_j^o + t \times \beta$$

subject to the linear constraints:

$$\sum_{i|u_i \in U} (s_i^u * x_i) + \sum_{j|o_i \in O} s_j^o + t \times \beta \leq M_s$$

$$\sum_{i|u_i \in U} \left[\left(v_i^u - \alpha s_i^u \right) x_i \right] - \alpha\beta t - z_v = \sum_{j|o_i \in O} v_j^o + \alpha \sum_{j|o_i \in O} s_j^o$$

$$-M_c t + z_v \leq \epsilon$$

and the bounds:

$$\begin{aligned} x_i \in \{0,1\} \forall u_i \in U \\ t_i \in \{0,1\} \\ 0 \leq z_v \leq M_c \end{aligned}$$

2.3 Model 2: Maximize the number of selected inputs based on an upper bound of transaction fee

This part is also included as a Jupyter notebook in the source code

2.3.1 Problem Modelling

The same with Model 1 with some modifications:

- **Constraint 1**: The size of a transaction cannot exceed the minimum size Y found by Model 1 times $(1+\gamma)$, where $\gamma \in [0,1]$ is a choosen coefficient:

$$y = \sum_{i|u_i \in U} (s_i^u * x_i) + \sum_{j|o_i \in O} s_j^o + z_s \leq (1+\gamma)Y$$

- **Objective function**: Maximize the number of selected UTXOs (and also try to produce no change output)

$$\max(\sum_{i|u_i \in U} u_i - t)$$

2.3.2 Solving with Linear Optimization

After reformulating, the model can be solved by linear optimization:
Minimize the objective function:

$$\max(\sum_{i|u_i \in U} u_i - t)$$

subject to the linear constraints:

$$\sum_{i|u_i \in U} (s_i^u * x_i) + t \times \beta \leq (1+\gamma)Y - \sum_{j|o_i \in O} s_j^o$$

$$\sum_{i|u_i \in U} \left[\left(v_i^u - \alpha s_i^u \right) x_i \right] - \alpha\beta t - z_v = \sum_{j|o_i \in O} v_j^o + \alpha \sum_{j|o_i \in O} s_j^o$$

$$-M_c t + z_v \leq \epsilon$$

and the bounds:

$$\begin{aligned} x_i \in \{0,1\} \forall u_i \in U \\ t_i \in \{0,1\} \\ 0 \leq z_v \leq M_c \end{aligned}$$

3 Implementation and Results

3.1 Coefficient of Model 2

We produced the result of Model 2 with 4 different values of γ:

- 10% (`model2-1`)
- 25% (`model2-2`)
- 40% (`model2-3`)
- 50% (`model2-4`)

3.2 Programming Language and Library

The two models mentioned in section 2 are implemented using Python 3. The library that we used to solve Linear Optimization is `MOSEK`.

3.3 Implementation Issues

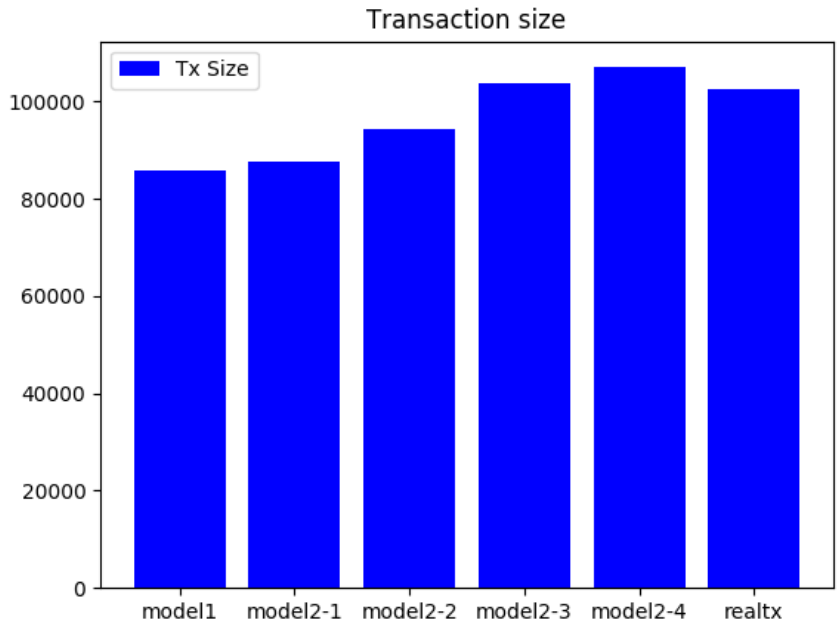
Out of 133 transactions in the dataset, model 1 successfully solved all. Meanwhile, model 2 failed on a few transactions (from 1 to 4 transactions out of 133, according to different values of γ). The unsolved transactions often have really large number of inputs (more than 19.000). Some transaction was successfully solved with a particular value of γ but was unsolved with another value. Based on our observation, as well as the `MOSEK` documentation, we suspect the unsolved transactions are caused by numerical issues.

To get the full result despite the unsolved transactions of model 2, for each unsolved transaction we will use the result of model 1 instead, since the result of model 1 is one viable solution of model 2.

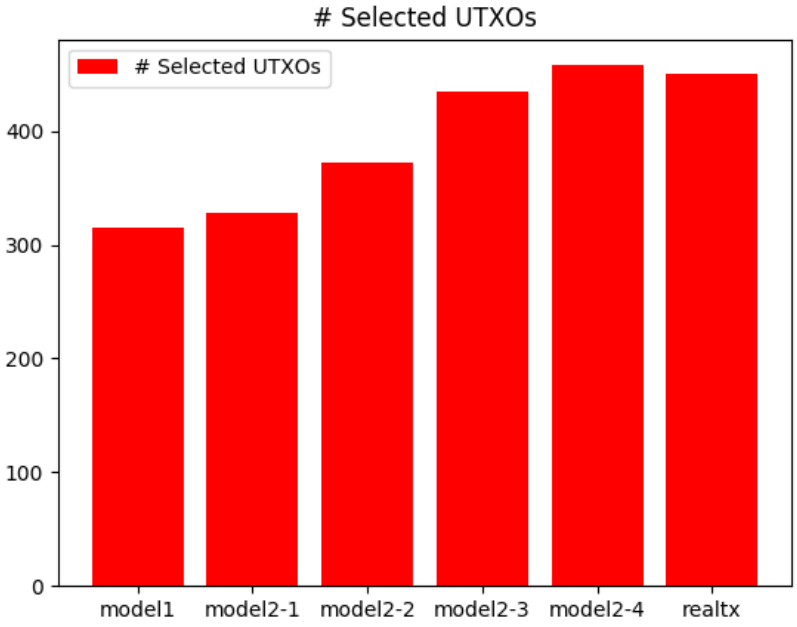
3.4 Results

The result of the two models is visualized with the following charts.

3.4.1 Transaction Size



3.4.2 Number of Selected UTXOs



3.4.3 Average Value of Selected UTXOs