TRƯỜNG ĐẠI HỌC BÁCH KHOA, ĐẠI HỌC QUỐC GIA TP. HCM
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

MÔ HÌNH HÓA TOÁN HỌC

BÀI TẬP LỚN

Mô hình hóa cho việc chọn lựa UTXO

Tutor: Huỳnh Tường Nguyên (htnguyen@hcmut.edu.vn)
Class: L02, Group: 1
Student members: Lê Công Linh - 1711948
Tô Phú Quý - 1712892
Nguyễn Đức Anh Tài - 1713015
Huỳnh Ngọc Tú - 1713835

Ho Chi Minh, 05/2019

Contents

1 Giới thiệu

Tiền mã hóa phi tập trung(decentralized cryptocurrency) là một tài sản kỹ thuật số sử dụng hệ thống mã hóa chung để bảo đảm giao dịch và tính toàn vẹn mà không cần can thiệp của bên thứ ba. Tất cả các giao dịch trong hệ thống được đăng kí trên một sổ cái hay còn gọi là blockchain. Blockchain là một hệ thống được cấu thành tuần tự từ các khối (block). Và mỗi block chứa một số lượng giao dịch không đổi và một mã băm của block trước, vì thế các giao dịch trong blockchain là không thể thay đổi được và hợp lệ. Một ví dụ quan trọng của loại tiền mã hóa này là Bitcoin, được giới thiệu năm 2008 và hiện tại có hơn 141 tỉ USD trên thị trường tiền mã hóa, với trung bình 229 ngàn giao dịch mỗi ngày và khoảng 183.89GB lưu trữ.[1].

Đối với hệ thống blockchain dùng cho giao dịch, chiến lược lựa chọn UTXOs cho giao dịch đóng một vai trò vô cùng quan trọng trong việc quản lý số dư tiền điện tử của bất cứ ví điện tử nào. Một chiến lược lựa chon tối ưu có thể thỏa mãn được những ràng buộc khó khăn và các mục tiêu quan trọng của ba nhóm chính là những người dùng, cộng đồng và các thợ mỏ. Đối với người dùng, họ muốn tạo ra một giao dịch có phí giao dịch được giảm thiểu tối đa và đảm bảo được quyền riêng tư của các hoạt động của họ. Ngược lại, các thợ mỏ tập trung vào việc đào được các giao dịch với phí cao nhất có thể. Đối với cộng đồng, tập UTXO có kích thước lớn trở thành ấn đề nan giải vì nó sẽ làm chậm hiệu năng xử lý giao dịch cũng như tăng tiêu thụ bộ nhớ. Như hệ thống Bitcoin, một ảnh (snapshot) của trạng thái hiện tại yêu cầu thêm khoảng trống trong bộ nhớ để xử lý các giao dịch [11].

Trong bản báo cáo này, chúng tôi xem xét về việc nghiên cứu chiến lược để lựa chọn các UTXO trong một tập các UTX0 cho một giao dịch được cho trước. Các UTXO được chọn phải tốn chi phí nhỏ nhất cho các thở mỏ đồng thời thu hẹp kích thước tập UTXO. Các phần còn lại của bản báo cáo được chia thành các phần sau: Phần 2 đưa ra vấn đề và yêu cầu của bài toán. Sau đó, chúng tôi đưa ra đề xuất giải quyết ở Phần 3. Phần 4 sẽ tổng kết đánh giá hiệu năng và kết quả thí nghiệm. Cuối cùng, chúng tôi sẽ tổng kết toàn bộ công việc của mình ở phần cuối.

2 Xây dựng vấn đề

Đối với một giao dịch nhất định, ví sẽ chọn một số UTXO từ tập UTXO(UTXO pool) sao cho có đủ giá trị để cấp vốn(funding) được gọi là giá trị mục tiêu(target value). Lựa chọn tốt nhất là có một kết quả khớp chính xác với mục tiêu(target) vì nó sẽ không sinh ra sự thay đổi đầu ra(output) khi trả về cho người gửi để giảm thiểu kích thước giao dịch(transaction size) cũng như không làm cho kích thước tập UTXO(UTXO pool) bùng nổ(exploded). Sự thay đổi đầu ra là số lượng tiền còn lại sau khi cấp vốn và phải lớn hơn ngưỡng DUST (ngưỡng DUST là đầu ra của giao dịch trong đó phí để chuộc lại lớn hơn 1/3 giá trị của nó). Mục tiêu của DUST là ngăn chặn các giao dịch rác khi mà ai đó cố gắng làm suy giảm mạng bằng cách cố ý tạo ra các giao dịch rất nhỏ có thể tiêu tốn băng thông lớn. Ngoài ra, Bitcoin và các hệ thống tương tự khác hiện đang tính phí cho mỗi giao dịch để ngăn chặn các hành vi xấu và đảm bảo chỉ các giao dịch hợp lệ dựa trên mạng blockchain. Mục tiêu của chúng ta là đề xuất một chiến lược hiệu quả trong việc lựa chọn một tập UTXO thích hợp cho một giao dịch nhất định, có khả năng đáp ứng nhiều ràng buộc sau:

1. Giảm thiểu kích thước giao dịch(transaction size) để có phí giao dịch(transaction fee) tối thiểu.
2. Thu nhỏ tập UTXO(UTXO pool).

Điều đáng chú ý là nhận ra rằng đề xuất của chúng ta rõ ràng mang lại lợi ích cho người dùng và mục tiêu của cộng đồng. Thêm vào đó, chiến lược đề xuất của chúng ta muốn các giao dịch được xác nhận nhanh nhất có thể bằng cách sử dụng mức phí phù hợp tùy thuộc vào nhu cầu của người sử dụng. Đây là một lợi ích ngầm cho các thợ mỏ.

Mục tiêu: Xác định một tập hợp con của tập UTXO có giá cả phải chăng sao cho thỏa mãn ràng buộc cứng H_1 và ràng buộc mềm S_1.

[1] https://coinmarketcap.com

Input

Input Parameters	Description
$U = \{u_1, ..., u_n\}$	set of UTXOs
$O = \{o_1, ..., o_n\}$	set of transaction outputs
$V^u = \{v_1^u, ..., v_n^u\}$	set of UTXO's values
$V^o = \{v_1^o, ..., v_m^o\}$	set of transaction output's values
$S^u = \{s_1^u, ..., s_n^u\}$	set of transaction input size, with input is chosen from UTXO ui
$S^o = \{s_1^o, ..., s_m^o\}$	set of transaction output's size
M	maximum size of a transaction
α	fee rate
T	dust threshold
ϵ	minimum of change output

Output

- Một tập hợp UTXO được chọn có thể chỉ chứa một output trùng khớp chính xác.
- Một đầu ra có thể thay đổi.

Ràng buộc cứng H_1

1. Một giao dịch phải có đủ giá trị để tiêu thụ.
2. Kích thước giao dịch không được vượt quá kích thước khối dữ liệu tối đa.
3. Tất cả các đầu ra giao dịch(transaction outputs) phải cao hơn ngưỡng DUST(DUST threshold) để chắc chắn rằng giao dịch này được chuyển tiếp đến mạng(network) và được xác nhận(confirmed).

Ràng buộc mềm S_1

1. Kích thước giao dịch được giảm thiểu.
2. Số lượng UTXO đã chọn được tối đa hóa để thu nhỏ kích thước nhóm UTXO.

3 Mô hình được đề xuất

3.1 Module 1

1. Variables:
   Biến quyết định

$$x_i \begin{cases} 1, & \text{if UTXO } u_i \text{ is chosen} \quad (1a) \\ 0, & \text{otherwise} \quad (1b) \end{cases}$$

Biến trung gian :

- sigma : biến flag (biến nhị phân)
- z_v : Tổng số giá trị thay đổi đầu ra
- z_s : Kích thước thay đổi đầu ra

```
bigM = 10000000000000
#Decision variables
Xi = [LpVariable('X' + str(i), 0, None, LpBinary) for i in range(n)]
sigma = pp.LpVariable('sigma',0,None,LpBinary)
z_s = pp.LpVariable('z_s',0,None,LpInteger)
z_v = pp.LpVariable('z_v',0,None,LpContinuous)
```

2. Constraints:

- Kích thước giao dịch không được vượt quá kích thước khối dữ liệu tối đa.

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

```
# A transaction size may not exceed maximum block data size
Problem1+= pp.lpDot(S_u,Xi) + pp.lpSum(S_o) + z_s <= M, "1st constraint"
```

- Một giao dịch phải có đủ các giá trị để tiêu thụ

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

```
# A transaction must have sufficient value for consuming
Problem1 += pp.lpDot(V_u,Xi) == pp.lpSum(V_o) + alpha*(pp.lpDot(S_u,Xi) +
    pp.lpSum(S_o) + z_s) + z_v , "2nd constraint"
```

- Tất cả các đầu ra giao dịch phải cao hơn ngưỡng DUST để chắc chắn rằng giao dịch này được chuyển tiếp đến mạng và được xác nhận.

$$\forall v \in V^o, v \geq T$$

```
# All the transaction outputs must be higher than the dust threshold
# Problem1+= pp.lpSum(V_o) >= T, "3rd constraint"
for i in range(len(V_o)):
     Problem1 += V_o[i] >= T , "3rd constraint"
```

- Mối quan hệ giữa giá trị đầu ra thay đổi zv và kích thước zs của nó được xác định như sau.

$$z_s \begin{cases} 0, & \text{nếu } 0 \leq z_v \leq \epsilon \quad (2a) \\ beta, & \text{nếu } z_v \geq \epsilon \quad (2b) \end{cases}$$

```
# The relation between change output value z_v and its size z_s
# If z_v = epxilon, z_s should be zero; otherwise, z_s should be equal to beta
Problem1 += z_v >= epsilon + 0.0000001 - bigM*(1-sigma), "4th constraint" ##
    0.0001->1
Problem1 += epsilon >= z_v - bigM*sigma, "5th constraint"        ## 0.0001-> bo
```

3. Hàm mục tiêu:

$$minimize \text{ y}$$

```
# Objective function
Problem1 += pp.lpDot(S_u,Xi) + pp.lpSum(S_o) + z_s, "The objective function"
```

3.2 Module 2

Mục tiêu của Model 2 là để tìm maximize số lượng mà UTXO được chọn để thu hẹp lại kích thước của nhóm UTXO ban đầu. Model 2 sẽ được xây dựng dựa trên kết quả thu được từ Model 1 như sau:

1. Model 2 - Các biến: bao gồm tất cả các biến trong Model 1

```
bigM = 10000000000000
#Decision variables
Xi = [LpVariable('X' + str(i), 0, None, LpBinary) for i in range(n)]
sigma = pp.LpVariable('sigma',0,None,LpBinary)
z_s = pp.LpVariable('z_s',0,None,LpInteger)
z_v = pp.LpVariable('z_v',0,None,LpContinuous)
```

2. Model 2 - Các ràng buộc: bao gồm tất cả các ràng buộc trong Model 1 và thêm một ràng buộc như sau:

$$y \leq (1 + \gamma) \times Y$$

```
# Extra constraint (extend transaction size)
Model2 += pp.lpDot(S_u,Xi) + pp.lpSum(S_o) + z_s + 0.0000000001 <= (1 + gamma)*Y
```

- Y là min của kích thước giao dịch thu được từ Model 1
- γ: là hệ số $(0 \leq y \leq 1)$

Nếu γ tiến đến 0, ta muốn giữ lại kích thước giao dịch nhỏ nhất thu được từ kết quả của Model 1. Mặt khác, một giao dịch có kích thước phù hợp khi nó được tạo ra bởi một số lượng UTXO càng lớn càng tốt.

3. Hàm mục tiêu:

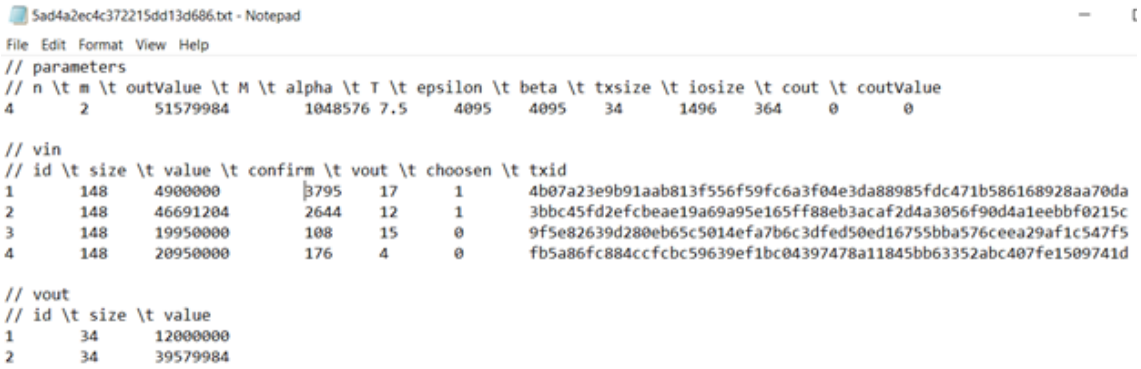
$$maximize(\sum_{i|u_i \in U} x_i - z_s/\beta)$$

```
Model2 = pp.LpProblem("Max of chossen UTXOs", pp.LpMaximize)
# Objective function
Model2 += pp.lpSum(Xi) - z_s*1/beta, "The objective function"
```

4 Thí nghiệm đánh giá

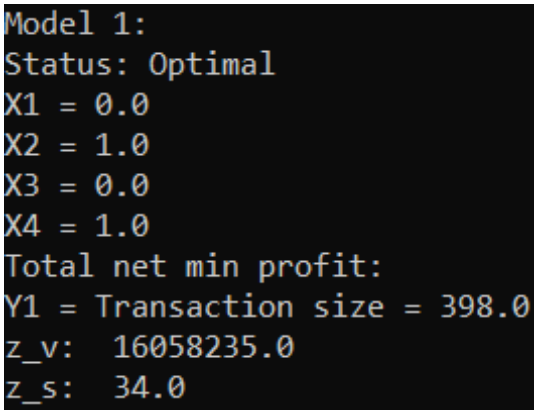
4.1 Module 1

1. Input format:
   file: 5ad4a2ec4c372215dd13d686.txt trong dataset0



2. Output format:

   - x_i : Các UTXO được chọn
   - y_1 : Giá trị kích thước tối thiểu giao dịch
   - z_v : Giá trị thay đổi đầu ra
   - z_s : Kích thước đầu ra



3. Inplementation in python/PuLP:
Source code

```
#Model 1
import pulp as pp
import numpy as np
from pulp import *

def Model1(m,n,M,alpha,beta,epsilon,T,V_u,S_u,V_o,S_o,Xi):
    bigM=10000000000000
    #Decision variables
    Xi=[LpVariable('X' + str(i), 0, None, LpBinary) for i in range(n)]
    sigma=pp.LpVariable('sigma',0,None,LpBinary)
    z_s=pp.LpVariable('z_s',0,None,LpInteger)
    z_v=pp.LpVariable('z_v',0,None,LpContinuous)

    Problem1=pp.LpProblem("Min of transaction size", pp.LpMinimize)
    #OBJECTIVE
        FUNCTION----------------------------------------------------------------------------
    Problem1+=pp.lpDot(S_u,Xi) + pp.lpSum(S_o) + z_s, "The objective function"

    #CONSTRAINTS-------------------------------------------------------------------------------------

    # A transaction size may not exceed maximum block data size
    Problem1+=pp.lpDot(S_u,Xi) + pp.lpSum(S_o) + z_s<=M,"1st constraint"

    # A transaction must have sufficient value for consuming
    Problem1+=pp.lpDot(V_u,Xi)== pp.lpSum(V_o) + alpha*(pp.lpDot(S_u,Xi) + pp.lpSum(S_o) + z_s)
        + z_v , "2nd constraint"

    # All the transaction outputs must be higher than the dust threshold
    #Problem1+= pp.lpSum(V_o)>=T, "3rd constraint"
    for i in range(len(V_o)):
        Problem1+= V_o[i]>=T , "3rd constraint"

    # The relation between change output value z_v and its size z_s
    #If (z_v > epxilon) sigma = 1 else sigma = 0;
    Problem1+=z_v >= epsilon + 0.0000001 - bigM*(1-sigma), "4th constraint"
        ###################################### 0.0001->1
    Problem1+=epsilon >= z_v - bigM*sigma, "5th constraint"
        ###################################### 0.0001-> bo
    # If z_v = epxilon, z_s should be zero; otherwise, z_s should be equal to beta

    Problem1+= z_s >= beta*sigma, "6th constraint" ######################################
    #Problem1+= 1 - bigM*(1-sigma) <= z_s <= 1 + bigM*(1-sigma)
    #Problem1+= 0 - bigM*sigma <= z_s <= 0 + bigM*sigma

    Problem1.solve()
    print('Model 1:')
    print("Status:", pp.LpStatus[Problem1.status])
    for i in range(n):
        print("X"+ str(i+1),"=",value(Xi[i]))
    #  for variable in Problem1.variables():
    #      if(variable.name!='u' and variable.name!='z_s' and variable.name!='z_v'):
    #          continue
    #      print(variable.name, "=", variable.varValue)
    print("Total net min profit:")
    print('Y1 = Transaction size =', pp.value(Problem1.objective)) ######################
    print("z_v: ",value(z_v))
    print("z_s: ",value(z_s))
    print("-----------------------------------------------------------")
   #print("Optimal   "+str(pp.value(Problem1.objective))," "+str(pp.value(z_v))," "
        "+str(pp.value(z_s)))
    YYY=pp.value(Problem1.objective)
    return pp.value(Problem1.objective)
```

```
#Run function
import glob
def Run():
    if(1):
        #file='I:\dataset0\5ad4a2ec4c372215dd13d686.txt'c
        with open('D:\\Mathematical Modeling\\dataset0\\5ad4a2ec4c372215dd13d686.txt') as f:
            V_u=[]
            S_u=[]
            V_o=[]
            S_o=[]
            Xi=[]
            line = f.readline()
            line = f.readline()
            line3 = f.readline()
            n=int(line3.split()[0])
            m=int(line3.split()[1])
            M=int(line3.split()[3])
            alpha=float(line3.split()[4])
            T=int(line3.split()[5])
            epsilon=int(line3.split()[6])
            beta=int(line3.split()[7])
            line = f.readline()
            line = f.readline()
            line = f.readline()
            for i in range(1,n+1):
                line = f.readline()
                S_u.append(int(line.split()[1]))
                V_u.append(int(line.split()[2]))
            line = f.readline()
            line = f.readline()
            line = f.readline()
            for i in range(1,m+1):
                line = f.readline()
                S_o.append(int(line.split()[1]))
                V_o.append(int(line.split()[2]))
            f.close()
            print('--------------------------------------------------------------------------------')
        print('m = ',m) ############################################################
        print('n = ',n)
        print('M = ',M)
        print('alpha = ',alpha)
        print('beta = ',beta)
        print('epsilon = ',epsilon)
        print('T = ',T)
        print('V_u = ',V_u)
        print('S_u = ',S_u)
        print('V_o = ',V_u)
        print('S_o = ',S_u)
        print('--------------------------------------------------------------------------------')
        Y=Model1(m,n,M,alpha,beta,epsilon,T,V_u,S_u,V_o,S_o,Xi)
        print('--------------------------------------------------------------------------------')
        #print(file)
    return


Run()
```

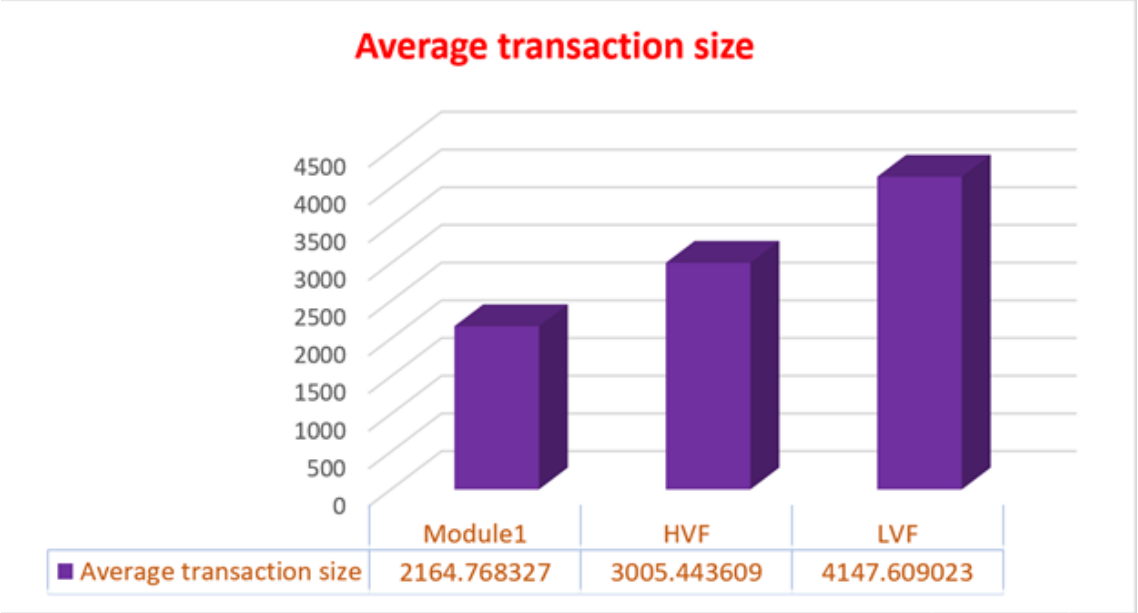
4. Experimental results:

```
Model 1:
Status: Optimal
X1 = 0.0
X2 = 1.0
X3 = 0.0
X4 = 1.0
Total net min profit:
Y1 = Transaction size = 398.0
z_v:  16058235.0
z_s:  34.0
```

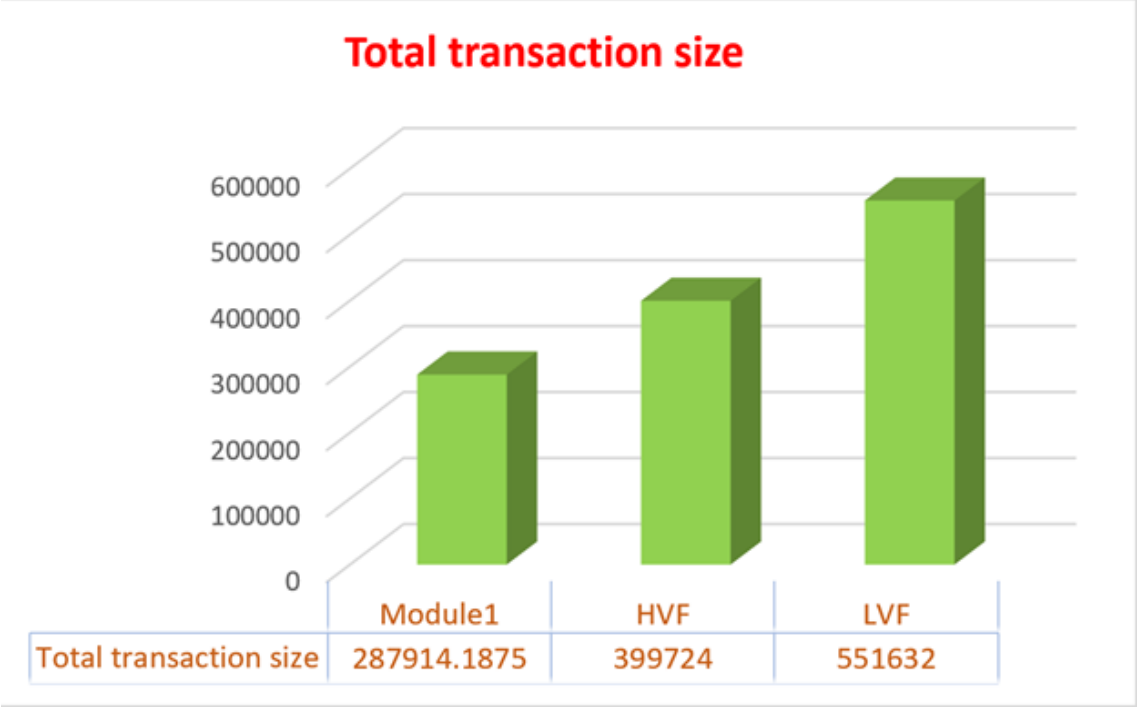
OutputModule1.txt - Notepad

File Edit Format View Help

5ad4a2ec4c372215dd13d686.txt	Optimal	398.0	16058235.0	34.0
5ad4a9e64c372215dd13d68b.txt	Optimal	398.0	10969.33	34.0
5ad4a25d4c372215dd13d685.txt	Optimal	398.0	11509.938	34.0
5ad4a3084c372215dd13d687.txt	Optimal	512.0	11509.938	34.0
5ad4a4934c372215dd13d688.txt	Optimal	1138.0	4145.6857	34.0
5ad4aa604c372215dd13d68c.txt	Optimal	364.0	1369.5	34.0
5ad4ab444c372215dd13d68e.txt	Optimal	512.0	40247.807	34.0
5ad4ac8e4c372215dd13d692.txt	Optimal	398.0	5775.7627	34.0
5ad4acaf4c372215dd13d693.txt	Optimal	546.0	14263.5	34.0
5ad4ad324c372215dd13d694.txt	Optimal	660.0	4936.381	34.0
5ad4ad534c372215dd13d695.txt	Optimal	3616.0	4515558400.	34.0
5ad4ada24c372215dd13d697.txt	Optimal	398.0	2844.8182	34.0
5ad4ae9d4c372215dd13d698.txt	Optimal	216.0	14210213000.0	34.0
5ad4b8d74c372215dd13d6a8.txt	Optimal	1514.0	222.49603	0.0
5ad4b33c4c372215dd13d69f.txt	Optimal	182.0	3774.6637	0.0
5ad4bd964c372215dd13d6b2.txt	Optimal	364.0	1641.6283	34.0
5ad4be974c372215dd13d6b4.txt	Optimal	660.0	12893.543	34.0
5ad4bf004c372215dd13d6b5.txt	Optimal	808.0	29614.396	34.0
5ad4b8154c372215dd13d6a7.txt	Optimal	398.0	10958.602	34.0
5ad4b9324c372215dd13d6a9.txt	Optimal	182.0	1043.6545	0.0



Biểu đồ: Kích thước giao dịch trung bình của Module1, HVF và LVF



Biểu đồ: Tổng kích thước giao dịch của Module1, HVF và LVF

4.2 Module 2

1. Input format:
   file: 5ad4a2ec4c372215dd13d686.txt trong dataset0

```
// parameters
// n \t m \t outValue \t M \t alpha \t T \t epsilon \t beta \t txsize \t iosize \t cout \t coutValue
4	2	51579984	1048576	7.5	4095	4095	34	1496	364	0	0

// vin
// id \t size \t value \t confirm \t vout \t choosen \t txid
1	148	4900000	3795	17	1	4b07a23e9b91aab813f556f59fc6a3f04e3da88985fdc471b586168928aa70da
2	148	46691204	2644	12	1	3bbc45fd2efcbeae19a69a95e165ff88eb3acaf2d4a3056f90d4a1eebbf0215c
3	148	19950000	108	15	0	9f5e82639d280eb65c5014efa7b6c3dfed50ed16755bba576ceea29af1c547f5
4	148	20950000	176	4	0	fb5a86fc884ccfcbc59639ef1bc04397478a11845bb63352abc407fe1509741d

// vout
// id \t size \t value
1	34	12000000
2	34	39579984
```

- y : Giá trị Minimize y từ Model 1
- γ : Khởi chạy lần lượt [0.05, 0.1 , 0.2 , 0.3 , 0.4 , 0.5]

2. Output format:

- x_i : Các UTXO được chọn
- Tổng số UTXO được chọn
- γ : Giá trị gamma khởi chạy

```
Model 2:
Status: Infeasible
X1 = 0.15116144
X2 = 0.21302775
X3 = 1.0
X4 = 1.0
Total max chossen UTXOs:  2.3641891900000003
gamma:  0.05
```

```
Model 2:
Status: Optimal
X1 = 0.0
X2 = 1.0
X3 = 0.0
X4 = 1.0
Total max chossen UTXOs:  2.0
gamma:  0.1
```

```
Model 2:
Status: Infeasible
X1 = 0.60182504
X2 = 0.16574253
X3 = 1.0
X4 = 1.0
Total max chossen UTXOs:  2.7675675699999998
gamma:  0.2
```

```
Model 2:
Status: Optimal
X1 = 0.0
X2 = 1.0
X3 = 0.0
X4 = 1.0
Total max chossen UTXOs:  2.0
gamma:  0.3
```

```
Model 2:
Status: Optimal
X1 = 0.0
X2 = 1.0
X3 = 1.0
X4 = 1.0
Total max chossen UTXOs:  3.0
gamma:  0.4
```

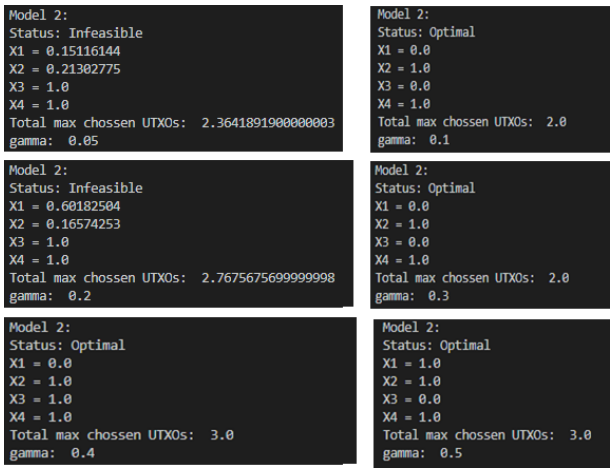
```
Model 2:
Status: Optimal
X1 = 1.0
X2 = 1.0
X3 = 0.0
X4 = 1.0
Total max chossen UTXOs:  3.0
gamma:  0.5
```

3. Inplementation in Python/Pulp:
Source code

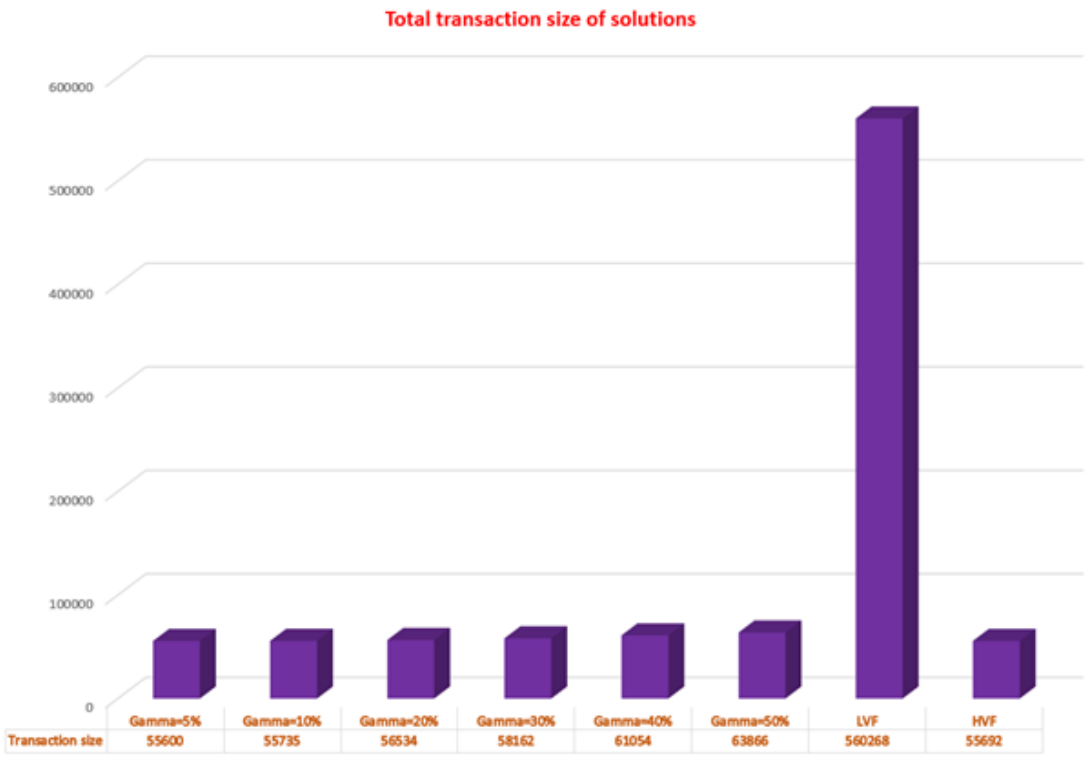
```
#
#  def Model 1 (presented above : 4.1.3 page 7)
#

#Model 2
import pulp as pp
import numpy as np
from pulp import *
def RunModel2(m,n,M,alpha,beta,epsilon,T,V_u,S_u,V_o,S_o,Y,gamma):
    bigM=1000000000000
    #Decision variables
    Xi=[LpVariable('X' + str(i), 0, None, LpBinary) for i in range(n)]
    sigma=pp.LpVariable('u',0,None,LpBinary)
    z_s=pp.LpVariable('z_s',0,None,LpInteger)
    z_v=pp.LpVariable('z_v',0,None,LpContinuous)

    Model2=pp.LpProblem("Max of chossen UTXOs", pp.LpMaximize)
    #Objective function
    #*********************************************************
    Model2+=pp.lpSum(Xi)-z_s*1/beta, "The objective function"
    #*********************************************************

    #Constraints
    # A transaction size may not exceed maximum block data size
    Model2+=pp.lpDot(S_u,Xi) + pp.lpSum(S_o) + z_s<=M,"1st constraint"

    # A transaction must have sufficient value for consuming
    Model2+=pp.lpDot(V_u,Xi)== pp.lpSum(V_o) + alpha*(pp.lpDot(S_u,Xi) + pp.lpSum(S_o) + z_s) +
        z_v , "2nd constraint"

    # All the transaction outputs must be higher than the dust threshold
    #Model2+= pp.lpSum(V_o)>=T, "3rd constraint"
    for i in range(len(V_o)):
        Model2+= V_o[i]>=T , "3rd constraint"

    # The relation between change output value z_v and its size z_s
    #If (z_v > epxilon) u = 1 else u = 0;
    Model2+=z_v >= epsilon+0.00000001- bigM*(1-sigma), "4th constraint"
    Model2+=epsilon+0.000000001 >= z_v - bigM*sigma, "5th constraint"
    # If z_v = epxilon, z_s should be zero; otherwise, z_s should be equal to beta
    Model2+= z_s >= beta*sigma, "6th constraint"

    #Extra constraint (extend transaction size)
    Model2+=pp.lpDot(S_u,Xi) + pp.lpSum(S_o) + z_s + 0.0000000001 <= (1+gamma)*Y

    Model2.solve()
    SUM=0
    print("Model 2")
    print("Status:", pp.LpStatus[Model2.status])
    for i in range(n):
        print("X"+ str(i+1),"=",value(Xi[i]))
        SUM+=value(Xi[i])
    print("Total max chosen UTXOs: ", SUM)
    print("z_v: ", value(z_v))
    print("z_s: ", value(z_s))
    print("gamma: ", value(gamma))
    return
```

```
#Run function
import glob
def Run():
    if(1):
    # path='I:\dataset0'
    # files=[file for file in glob.glob(path+'\*.txt')]

    # for file in files:
    #     if(file=='I:\dataset0\logs.txt'):
    #         continue
    #     print(file)
    #     with open(file,"r") as f:
        with open('I:\\dataset0\\5ad4a2ec4c372215dd13d686.txt ') as f:
            V_u=[]
            S_u=[]
            V_o=[]
            S_o=[]
            line = f.readline()
            line = f.readline()
            line3 = f.readline()
            n=int(line3.split()[0])
            m=int(line3.split()[1])
            M=int(line3.split()[3])
            alpha=float(line3.split()[4])
            T=int(line3.split()[5])
            epsilon=int(line3.split()[6])
            beta=int(line3.split()[7])
            line = f.readline()
            line = f.readline()
            line = f.readline()
            for i in range(1,n+1):
                line = f.readline()
                S_u.append(int(line.split()[1]))
                V_u.append(int(line.split()[2]))
            line = f.readline()
            line = f.readline()
            line = f.readline()
            for i in range(1,m+1):
                line = f.readline()
                S_o.append(int(line.split()[1]))
                V_o.append(int(line.split()[2]))
            f.close()
        gammaSet=[0.05, 0.1, 0.2, 0.3, 0.4, 0.5]
        Y=RunModel1(m,n,M,alpha,beta,epsilon,T,V_u,S_u,V_o,S_o)
        for j in range(len(gammaSet)):
            gamma=gammaSet[j]
            RunModel2(m,n,M,alpha,beta,epsilon,T,V_u,S_u,V_o,S_o,Y,gamma)
    return

Run()
```

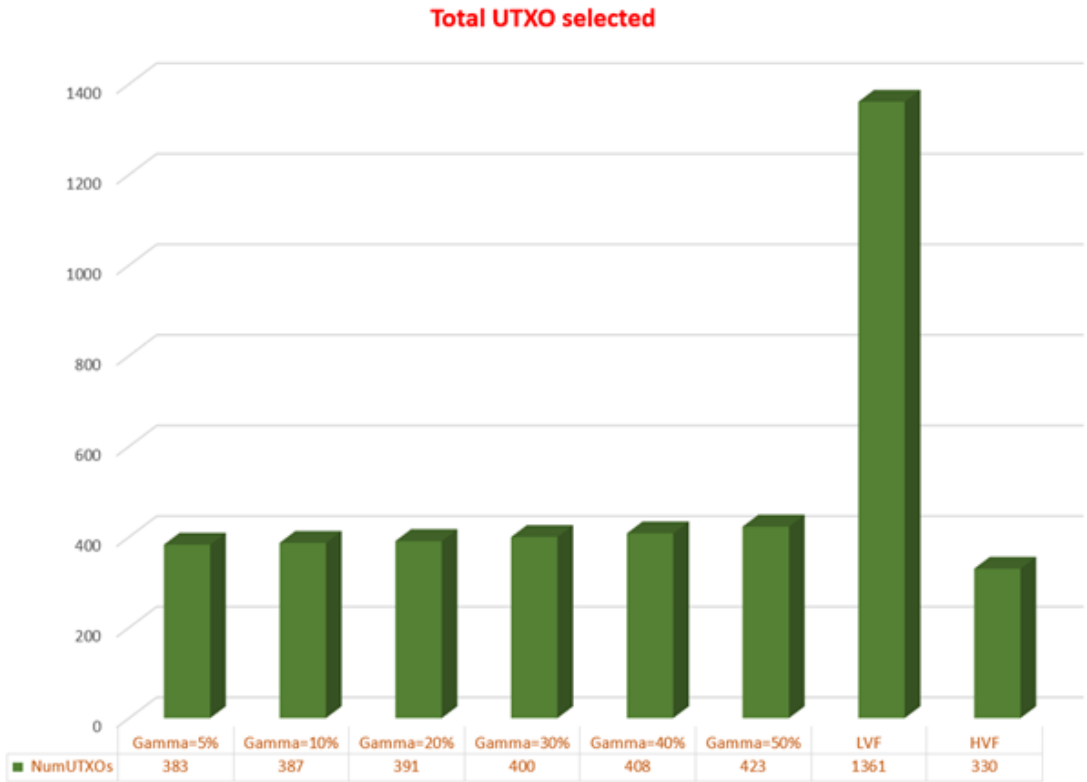
4. Experimental result



	A	B	C	D	E	F	G	H	I	J	K	L	M
1		x(5%)	num(5%)	x(10%)	num(10%)	x(20%)	num(20%)	x(30%)	num(30%)	x(40%)	num(40%)	x(50%)	num(50%)
2	5ad448a9	15,	1	0,	1	17,	1	1,15,61,	3	1,15,61,	3	1,15,61,	3
3	5ad44bfdc	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2
4	5ad44e0e	2,	1	2,	1	2,	1	2,	1	2,	1	0,1,2,	3
5	5ad44e1b	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2
6	5ad4503e	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,4,	3
7	5ad4517c	3,	1	3,	1	12,	1	12,	1	12,	1	12,	1
8	5ad4519c	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2
9	5ad45307	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2
10	5ad45353	0,1,2,	3	0,1,2,	3	0,1,2,	3	0,1,2,	3	0,1,2,	3	0,1,2,	3
11	5ad4537e	1,	1	1,	1	1,	1	1,	1	1,	1	1,	1
12	5ad45aac	0,4,	2	0,4,	2	0,4,	2	0,6,	2	0,6,	2	0,1,2,4,	4
13	5ad45bc6	0,1,2,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2
14	5ad45f554	5,	1	5,	1	5,	1	5,	1	5,	1	5,	1
15	5ad46006	0,2,3,	2	0,2,3,	2	0,3,	2	0,3,	2	0,3,	2	0,3,	2
16	5ad4601c	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2
17	5ad46027	0,1,2,	3	0,1,2,	3	0,1,2,	3	0,1,2,4,	4	0,1,2,5,	4	0,1,2,9,	4
18	5ad462d6	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2
19	5ad46427	0,1,2,3,	4	0,1,2,3,	4	0,1,2,3,	4	0,1,2,3,	4	0,1,2,3,	4	0,1,2,3,	4
20	5ad46607	1,2,3,4,5,	3	0,1,2,	3	2,3,4,5,	4	1,2,3,4,5,	4	1,2,3,4,5,	4	1,2,3,4,5,	5
21	5ad46696	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2
22	5ad466b8	0,1,2,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,2,	3
23	5ad46769	0,1,2,3,	4	0,1,2,3,	4	0,1,2,3,	4	0,1,2,3,	4	0,1,2,3,	4	0,1,2,3,	4
24	5ad46823	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2
25	5ad469f14	0,1,2,3,4,5	9	0,1,2,3,4,5	9	0,1,2,3,4,5	9	0,1,2,3,4,5	9	0,1,2,3,4,5	9	0,1,2,3,4,5	9
26	5ad46c15	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2
27	5ad46c3a	2,	1	2,	1	2,	1	2,	1	2,	1	0,1,2,	2
28	5ad46ca8	0,1,2,	3	0,1,2,	3	0,1,2,	3	0,1,2,	3	0,1,2,	3	0,1,2,	3
29	5ad46ec1	0,1,2,3,4,5	7	0,1,2,3,4,5	7	0,1,2,3,4,5	7	0,1,2,3,4,5	7	0,1,2,3,4,5	7	0,1,2,3,4,5	7
30	5ad46ef24	0,1,2,	2	0,1,2,	2	0,1,	2	0,1,	2	0,1,	2	0,1,	2
31	5ad47128	0,1,2	2	0,1,2	2	0,1	2	0,1	2	0,1	2	0,1	2



Biểu đồ: Tổng số lượng UTXO được chọn của từng giải pháp



Biểu đồ: Tổng kích thước giao dịch của từng giải pháp

5 Kết luận

Trong báo cáo này, chúng ta đã đề xuất hai mô hình toán học cho việc giải quyết hai mục tiêu thiết yếu khi tạo giao dịch mới trên blockchain. Mô hình đầu thiên là giảm thiểu kích thước giao dịch để có thể tạo ra một khoản phí nhỏ cho nhiệm vụ khai thác chịu trách nhiệm xác nhận giao dịch này trên mạng. Mô hình thứ hai được chế tạo để kiềm chế sự bùng nổ của nhóm UTXO bằng cách dùng trả chi phí phải chăng phù hợp.

5.1 Module 1

- Kích thước giao dịch của module 1 đã thấy được sự tối ưu đáng kể so với các phương pháp HVF, LVF, OF.
- Cũng qua kết quả đó ta thấy module 1 đã đáp ứng được mục tiêu đề ra đó giảm thiểu tối ưu kích thước giao dịch (transaction size) nhằm để giảo thiểu phí giao dịch (transaction fee) mà người dùng (users) trả cho thợ mỏ (miners).

5.2 Module 2

- Với việc nới rộng kích thước giao với một tham số thích hợp trong phạm vi cho phép chúng ta chọn được nhiều UTXO hơn, giảm đáng kể kích thước của tập UTXO, giúp cho việc truy xuất và xác nhận giao dịch diễn ra nhanh hơn, nâng cao hiệu quả giao dịch.
- Theo kết quả nhận thấy, với tham số gamma 0.4 và 0.5 là cho kết quả tốt hơn hết

EXPERIMENTAL RESULTS OF THE NUMBER OF SELECTED UTXOS

Method	DS1	DS2	DS3
Real Transaction	17059	5717	731
LVF	**513987**	23470	479906
Model 2 (γ = 5%)	16426	5286	529
Model 2 (γ = 10%)	16489	5320	558
Model 2 (γ = 20%)	16654	5406	637
Model 2 (γ = 40%)	17004	5608	785
Model 2 (γ = 50%)	17273	5807	855

PERFORMANCE COMPARISON IN TERMS OF TRANSACTION SIZE (IN BYTES)

Method	DS1	DS2	DS3
Real Transaction	17317872	4330028	784672
LVF	**90865154**	6959648	71702334
Model 2 (γ = 5%)	17226596	4268179	755245
Model 2 (γ = 10%)	17235501	4272984	759345
Model 2 (γ = 20%)	17260814	4286764	770878
Model 2 (γ = 40%)	17313888	4318482	792234
Model 2 (γ = 50%)	17356251	4349849	803230

Tô Phú Quý – 1712892	- Chịu trách nhiệm chính phần code hiện thực module 1. - Chạy, tổng hợp kết quả và lấy dữ liệu module 1. - Nghiên cứu kỹ và phân tích đề để triển khai và giải đáp cho các thành viên. - Chỉnh sửa và bổ sung nội dung file báo cáo.
Lê Công Linh – 1711948	- Chịu trách nhiệm chính phần code hiện thực module 2. - Tham gia phân tích và hiện thực phần module 1. - Chạy, tổng hợp kết quả và lấy dữ liệu module 2. - Chỉnh sửa và bổ sung nội dung file báo cáo.
Nguyễn Đức Anh Tài - 1713015	- Chịu trách nhiệm chính phần code đọc và xuất file. - Tham gia phân tích và hiện thực các module. - Chịu trách nhiệm chính nôi dung và chỉnh sửa về file báo cáo . - Tìm kiếm tài liệu và cung cấp cho các nhóm.
Huỳnh Ngọc Tú – 1713835	- Chịu trách nhiệm phần code đọc và xuất file. - Tham gia phân tích và hiện thực các module. - Chịu trách nhiệm chính về file báo cáo. - Tìm kiếm tài liệu và cung cấp cho nhóm

References

[1] wikipedia. “**link: http://en.wikipedia.org/**”, , last access: 05/05/2015.

[2] Frey, D., Makkes, M. X., Roman, P.-L., Taiani, F., Voulgaris, S.: Bringing secure Bitcoin transactions to your smartphone. The 15th International Workshop on Adaptive and Reflective Middleware, (2016).

[3] Antonopoulos, A. M.: Mastering Bitcoin. 2nd edn. O’Reilly Media, CA 95472 (2014).

[4] Bitcoinjs: Open Source Organisation for Bitcoin JavaScript Libraries,https://github.com/bitcoinjs. Last accessed 15 August 2018.

[5] Bitcoinj: Library for working with the Bitcoin protocol,https://bitcoinj.github.io. Last accessed 10 August 2018.

[6] Yanovich, Y., Mischenko, P., Ostrovskiy, A.: Shared Send Untangling in Bitcoin, White paper, Bitfury Group Limited (2016).

[7] Dai, P., Mahi, N., Earls, J., Norta, A.: Smart-Contract Value-Transfer Protocols on a Distributed Mobile Application Platform, https://qtum. org/uploads/files/cf6d69348ca50dd985b60425ccf282f3.pdf, (2016).

[8] Sergi, D.-S., Cristina, P.-S., Guillermo, N.-A., Jordi, H.-J.: Analysis of the Bitcoin UTXO set, IACR Cryptology ePrint Archive, (2017).

[9] Erhardt, M.: An Evaluation of Coin Selection Strategies, Master thesis, Karlsruhe Institute of Technology, URL: http://murch.one/wp-content/uploads/2016/11/erhardt2016coinselection.pdf, (2016).

[10] Zahnentferner, J.: Chimeric ledgers: Translating and unifying utxo-based and account-based cryptocurrencies, Cryptology ePrint Archive, Report 2018/262, 2018. https://epri nt. iacr. org/2018/262, (2018).

[11] Chepurnoy, A., Kharin, V., Meshkov, D.: A Systematic Approach To Cryptocurrency Fees. IACR Cryptology ePrint Archive, (2018).