HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY, VNU-HCM
FACULTY OF COMPUTER SCIENCE & ENGINEERNG

MATHEMATICAL MODELING (CO2011)

Assignment

Mathematical model for UTXO selection

Giảng viên: Huỳnh Tường Nguyên (htnguyen@hcmut.edu.vn)
Lớp: L01, Nhóm: 7
SV thực hiện: Phạm Tấn Đại – 1710929
Nguyễn Lê Hoàng Hiệu – 1711355
Trần Xuân Hội – 1711457
Cao Minh Khôi – 1710148
Nguyễn Bảo Phúc – 1712674
Nguyễn Vũ Hoàng Phúc – 1712691

Thành phố Hồ Chí Minh, Tháng 05/2019

Contents

Đường link truy cập các file trong bài tập lớn:

http://bit.ly/submit_gr7

1 Introduction

Tiền mã hóa phân tán là một loại tài sản số sử dụng hệ thống mật mã học để bảo vệ các giao dịch, ngăn các bên thứ ba có thể can thiệp,thay đổi thông tin trong hệ thống. Tất cả các giao dịch trong hệ thống đều được ghi lại trong sổ cái gọi là blockchain (chuỗi các block). Mỗi block chứa một số các giao dịch và hash của block trước đó. Vì thế tất cả các giao dịch trong blockchain là hợp lệ và không thể thay đổi. Ví dụ nổi tiếng nhất về tiền mã hóa là Bitcoin, hình thành từ năm 2008, với tổng giá trị trên thị trường là hơn $141 tỉ USD, 183.89 GB dung lượng và hơn 229,000 giao dịch mỗi ngày[1].

Đối với loại blockchain sử dụng transaction-base, chiến lược chọn các "UTXO" cho giao dịch là cực kì quan trọng trong việc quản lí số dư trong ví. Một phương pháp tối ưu phải thỏa mãn các ràng buộc và các yêu cầu cần thiết cho ba nhóm đó là người dùng, "miners" và cộng đồng. Đối với người dùng, họ muốn thực hiện một giao dịch sao cho giao dịch đó có mức phí nhỏ nhất có thể và đảm đảo tính ẩn danh của giao dịch. Ngược lại,"miners" muốn "mining" các giao dịch có mức phí càng cao càng tốt. Đối với cộng đồng, "UTXO pool" lớn làm giảm hiệu suất của giao dịch, và gây tốn bộ nhớ,do vậy cần đảm bảo kích thước "UTXO pool" càng nhỏ càng tốt.

Trong bài tập lớn này, để ưu tiên cho lợi ích của người dùng và cộng đồng, vấn đề chúng ta cần giải quyết là tạo ra một mô hình để chọn các UTXO của một giao dịch sao cho phải trả phí thấp nhất cho "miners" và chọn càng nhiều các "UTXO" có kích thước nhỏ càng tốt để giảm kích thước của "UTXO pool".

2 Problem formulation

- Problem statement:
  - Mỗi người dùng Bitcoin có thể sử dụng bất kì ví nào để quản lí khoá và giao dịch với các từ "UTXOs pool" và có thể có nhiều đầu ra.
  - Trong giao dịch theo dạng giao dịch dựa trên chuỗi khối (transaction-based blockchains) thì việc lựa chọn UTXOs đóng vai trò quan trọng trong việc quản lí ví điện tử.
  - Giao dịch sẽ chưa được thêm vào "sổ cái" (legder) cho đến khi nó được giải ra với các thợ đào người sẽ nhận khoản tiền tương ứng khi đóng góp phần trăm trong việc xác nhận giao dịch.
  - Trường hợp tốt nhất là tồn tại trong UTXOs những ví chính xác với số tiền cần gửi. Nhưng việc này ít xảy ra thường xuyên.
  - Việc lựa chọn UTXOs cần thoả mãn 2 tiêu chí:

[1] https://coinmarketcap.com

  - Giảm thiểu kích thước giao dịch dẫn đến giảm phí giao dịch
  - "UTXOs pool" nhỏ lại

- Requirements statement:
  - Kích thước giao dịch không vượt quá kích thước của một block.
  - Tất cả các output phải lớn hơn một giá trị gọi là "dust threshold" để tránh tạo ra các giao dịch "spam".

3 Proposed model

3.1 Model 1: Minimizing transaction size

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

Figure 1: ALL INPUT PARAMETERS OF OUR DEFINED WORK

1. Các biến quyết định

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

2. Các biến trung gian
   - y : Quy mô giao dịch.
   - z_v : Giá trị của Output thay đổi (Số lượng)
   - z_s : Kích thước của Output thay đổi

$$z_s = \begin{cases} 0, & 0 \leq z_v \leq \varepsilon \\ \beta, & z_v > \varepsilon \end{cases} \quad (2)$$

3. Những ràng buộc: Được định nghĩa trong chiến lược lựa chọn đề xuất có thể được xây dựng như sau:

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

$$y = \sum_{i|u_i \in U} s_i^u * x_i + \sum_{j|o_j \in O} s_j^o + z_s + (txsize - iosize) \leq M \quad (3)$$

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

$$\sum_{i|u_i \in U} v_i^u * x_i = \sum_{j|o_j \in O} v_j^o + \alpha * y + z_v \quad (4)$$

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

$$T \leq \sum_{j|o_j \in O} v_j^o \quad (5)$$

   - Mối quan hệ giữa giá trị Output thay đổi z_v và kích thước của nó z_s được định nghĩa như sau.

$$z_s \leq \left\lfloor \frac{z_v}{\varepsilon} \right\rfloor * \beta \quad (6)$$

     Nếu $z_y < \epsilon$, $z_s = 0$, nếu không $z_s = \beta$.

   - a_i là biến nhị phân.

$$\forall i | u_i \in U : x_i \in \{0, 1\} \quad (7)$$

4. Hàm mục tiêu : Tối thiểu hóa kích thước giao dịch.

$$\text{minimize} \quad y \quad (8)$$

3.2 Model 2: Maximizing the number of UTXOs

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

$$y < (1+\gamma) \times Y, \tag{9}$$

Với:

- Y là kích thước giao dịch tối thiểu thu được ở Model 1.
- γ: là 1 hệ số ($0<\gamma<1$)

Nếu γ dần về 0, chúng tôi muốn giữ kích thước giao dịch tối thiểu thu được từ Model 1. Mặt khác, giao dịch có kích thước phù hợp được tạo bởi một số UTXO càng lớn càng tốt.

3. Hàm mục tiêu: Tối đa hóa số lượng UTXO

$$\text{maximize} \quad \Big(\sum_{i|u_i \in U} x_i - z_s/\beta \Big) \tag{10}$$

4 Experimental evaluation

4.1 Input format

4.1.1 Model 1

Ví dụ với file data mẫu như sau:

```
data;
param n := 7;
param m := 1;
param M := 1048576;
param alpha := 8.29435483870968;
param T := 4529;
param epsilon := 4529;
param beta := 34;
param txsize := 1488;
param iosize := 364;
```

```
param:                  size        value       :=
        1                 148        3802043
        2                 148        9202262
        3                 148        449963
        4                 148        285272
        5                 148        49661
        6                 148        14312
        7                 148        2484;
param:                  Size        Value       :=
        1                 34        12542000;
end;
```

4.1.2 Model 2

Có thêm biến Y ở model 1, file data như sau:

```
data;
param n := 7;
param m := 1;
param M := 1048576;
param alpha := 8.29435483870968;
param T := 4529;
param epsilon := 4529;
param beta := 34;
param txsize := 1488;
param iosize := 364;
param:                  size        value       :=
        1        148        3802043
        2        148        9202262
        3        148        449963
        4        148        285272
        5        148        49661
        6        148        14312
        7        148        2484;
param:                  Size        Value       :=
        1        34        12542000;
param Y:= 1488;
end;
```

4.2 Output format

4.2.1 Model 1

Đầu ra bao gồm:

- Tập các UTXOs được lựa chọn.
- Kích thước giao dịch.
- z_v: Giá trị của đầu ra thay đổi.
- z_s: Kích thước của đầu ra thay đổi.

4.2.2 Model 2

Đầu ra Tương tự như Model 1, có thêm một số thành phần mới như :

- Số lượng UTXO.
- Tổng giá trị đầu vào.
- Tổng giá trị đầu ra.

4.3 Implementation in GLPK/AMPL:

4.3.1 Model 1

Hiện thực tất cả các file đầu vào với file **model1.mod** như dưới đây:

```
set U := {1..n};
set O := {1..m};

param alpha; param beta;
param T; param M; param epsilon;

param value{U} >= 0;
param size{U} >= 0;
param Value{O} >= 0;
param Size{O} >= 0;
param txsize >= 0;
param iosize >= 0;
param big := 1000000000 integer;

var X{U} binary;          # Decision variables
var z_s >=0;                 # Intermediate variables
```

```
var z_v >=0;                # A value of change output
var u binary;                 # Variables check (z_v > epxilon) ? 1 : 0;
#Objective function
minimize y : sum{j in U} (size[j] * X[j]) + sum{i in O} (Size[i]) + z_s+(txsize-iosize);

#file to save output
param f, symbolic := "result.txt";
#Constraint

# A transaction size may not exceed maximum block data size
s.t. constraint1: sum{j in U} (size[j] * X[j]) + sum{i in O} (Size[i])
+ z_s + (txsize-iosize) <= M;

# A transaction must have sufficient value for consuming
s.t. constraint2: sum{i in U} (value[i] * X[i]) = sum{j in O} (Value[j]) +
alpha * (sum{j in U} (size[j] * X[j]) + sum{i in O} (Size[i]) + z_s) + z_v;

# All the transaction outputs must be higher than the dust threshold
s.t. constraint3: T <= sum{i in O} Value[i];

# If z_v = epxilon, z_s should be zero; otherwise, z_s should be equal to beta
s.t. constraint4: z_s = beta*u;
#If (z_v > epxilon) u = 1 else u = 0;
s.t. c1: z_v          >= epsilon + 1 - big*(1-u);
s.t. c2: epsilon >= z_v  - big*u;
solve;

printf {i in U} " x[%d] = %d\n",i,X[i];
printf " zs = %d\n",z_s>>f;
printf " zv = %d\n", z_v>>f;
printf " minimize y = %d\n",sum{j in U}(size[j]*X[j]) + sum{i in O} (Size[i])
+ z_s + (txsize-iosize) >>f;
data;

end;
```

Với file data mẫu ở phần 4.1 ta có kết quả nhận được là:

```
zs = 34
zv = 459286
minimize y = 1488
x[1] = 1
```

```
x[2] = 1
x[3] = 0
x[4] = 0
x[5] = 0
x[6] = 0
x[7] = 0
```

Với kết quả này thì giá trị của đầu ra thay đổi là 34, kích thước của đầu ra thay đổi là 459286, kích thước giao dịch là 1488 và có 2 UTXOs được lựa chọn là 1 và 2.

4.3.2 Model 2

Hiện thực tất cả các file đầu vào với file **model2.mod** như dưới đây:

```
param n; param m;

set U := {1..n};
set O := {1..m};

param alpha; param beta;
param T; param M; param epsilon;
param Y >=0;
param value{U} >= 0;
param size{U} >= 0;
param Value{O} >= 0;
param Size{O} >= 0;
param big := 1000000000 integer;
var X{U} binary;          # Decision variables
var z_s >=0;                 # Intermediate variables
var z_v >=0;                 # A value of change output
var u binary;                  # Variables check (z_v > epxilon) ? 1 : 0;
#Objective function

maximize UTXO : sum{j in U} ( X[j]) - z_s/beta;

#file to save output
param f, symbolic := "resultmodel2.txt";

#Constraint

# A transaction size may not exceed maximum block data size
```

```
s.t. constraint1: sum{j in U} (size[j]*X[j]) + sum{i in O}(Size[i])
+ z_s <= (1+0.05)*(Y-(txsize-iosize));

# A transaction must have sufficient value for consuming
s.t. constraint2: sum{i in U} (value[i] * X[i]) = sum{j in O} (Value[j])
+ alpha * (sum{j in U} (size[j] * X[j])+ sum{i in O} (Size[i]) + z_s) + z_v;

# All the transaction outputs must be higher than the dust threshold
s.t. constraint3: T <= sum{i in O} Value[i];

# If z_v = epxilon, z_s should be zero; otherwise, z_s should be equal to beta
s.t. constraint4: z_s = beta*u;
#If (z_v > epxilon) u = 1 else u = 0;
s.t. c1: z_v          >= epsilon + 1 - big*(1-u);
s.t. c2: epsilon >= z_v  - big*u;
solve;

printf {i in U} " x[%d] = %d\n",i,X[i];
printf " zs = %d\n",z_s;
printf " zv = %d\n", z_v;
printf " objective function = %d\n",sum{j in U} ( X[j]) - z_s/beta;
printf " size : %d\n",sum{j in U} (size[j] * X[j]) + sum{i in O} (Size[i]) + z_s;
printf "value in: %d\n",sum{i in U} (value[i] * X[i]);

######### write to file
printf " zs = %d\n",z_s>>f;
printf " zv = %d\n", z_v>>f;
printf "number of UTXOs = %d\n",sum{j in U} ( X[j])>>f;
printf "transaction size y= %d\n",sum{j in U}(size[j]*X[j])+sum{i in O}(Size[i])
+ z_s + (txsize - iosize) >> f;
printf "value of input : %d\n",sum{i in U} (value[i] * X[i])>>f;
printf "value of output : %d\n", sum{ i in O}Value[i]>>f;

data;

end;
```

Kết quả nhận được là:

```
zs = 34
zv = 459286
number of UTXOs = 2
```

```
transaction size y = 1488
value of input : 13004305
value of output : 12542000
 x[1] = 1
 x[2] = 1
 x[3] = 0
 x[4] = 0
 x[5] = 0
 x[6] = 0
 x[7] = 0
```

Với kết quả này thì giá trị của đầu ra thay đổi là 34, kích thước của đầu ra thay đổi là 459286, kích thước giao dịch là 1488 và số lượng UTXO là 2 đó là 1 và 2, tổng giá trị đầu vào 13004305, tổng giá trị 12542000.

4.4 Experimental results

4.4.1 Model 1

4.4.1.a Thu thập dữ liệu

Dựa vào tập dữ liệu mẫu với kích thước là 133 trường hợp. Tập dữ liệu có 3 bộ dữ liệu: DS1 chứa đầy 0 trường hợp, DS2 có 121 trường hợp với số lượng UTXO từ 2 đến 1000 và DS3 có 12 trường hợp có UTXO lớn hơn 1000.

4.4.1.b Kết quả thực nghiệm

So sánh kết quả với hai phương pháp hiện có bao gồm HVF và LVF để đánh giá và so sánh hiệu suất.

Bảng 1 tóm tắt tổng kích thước giao dịch cho cả ba tập dữ liệu DS1, DS2, DS3 tương ứng với lựa chọn UTXO bằng các giao dịch thực, HVF và Mô hình 1 tương ứng. Ngoài ra, Hình 1 cho thấy sự khác biệt về quy mô giao dịch của các giao dịch thực và các giao dịch khác do HVF tạo ra, Mô hình 1. Dữ liệu thể hiện rằng các UTXO được chọn bởi HVF và người giải quyết trên.

Method	DS1	DS2	DS3
Real	0	256504	158940
HVF	0	249296	149068
Model 1	0	248888	150264

Table 1: Kết quả thực nghiệm kích thước giao dịch tổng thể

Method	DS1	DS2	DS3
HVF	0	7208	9872
Model 1	0	7616	8676

Table 2: So sánh hiệu suất về quy mô giao dịch

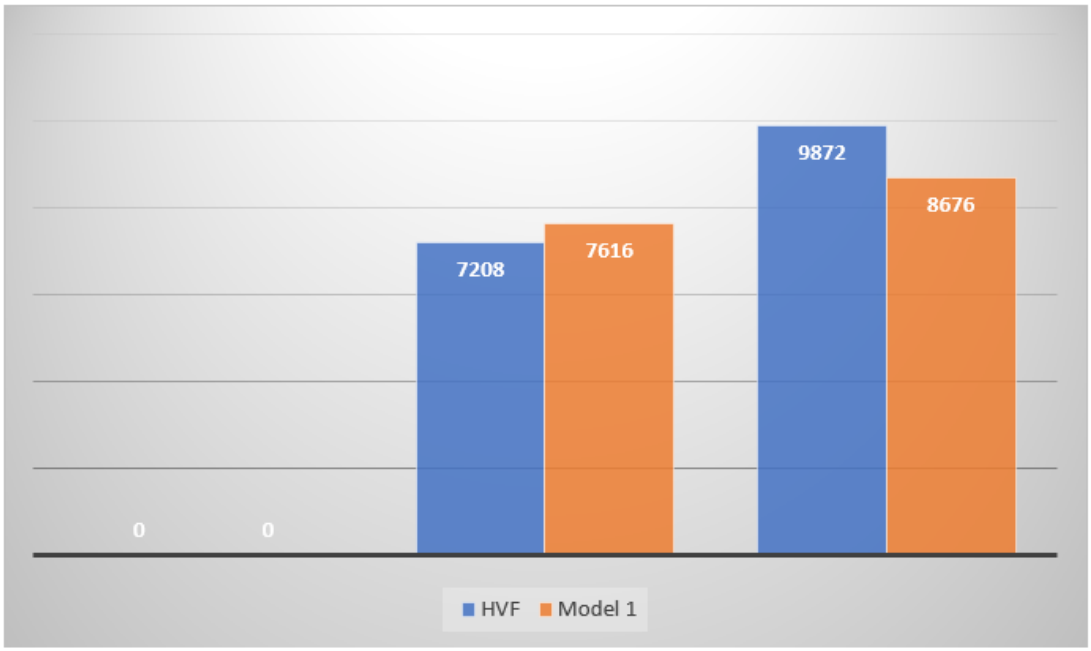

Figure 2: So sánh hiệu suất về quy mô giao dịch

Mô hình 1 cho kết quả tốt hơn so với các giao thực trong thời hạn của quy mô giao dịch. Khi so sánh cả hai chiến lược khi chọn UTXO, tất cả các địa chỉ Bitcoin có từ 2 đến 1000 UTXO (khoảng 90,98%) (DS2) và trên 1000 UTXO (DS3), thì đối với DS2 model 1 có hiệu suất về quy mô giao dịch cao hơn với HVF (1.06 lần), nhưng đối với DS3 thì hiệu suất về quy mô giao dịch lại thấp hơn so với HVL (0.88 lần)

4.4.2 Model 2

Ở Model 2 với việc chọn các giá trị cho lần lượt là 5%, 10%, 20%, 30%, 40%, 50% thì ta có kết quả thực nghiệm như bên dưới đây:

Method	DS1	DS2	DS3
Real Transition	0	367	6
LVF	0	785	189
Model 2(5%)	0	293	2

Model 2(10%)	0	298	2
Model 2(20%)	0	304	2
Model 2(40%)	0	333	2
Model 2(50%)	0	352	2

Table 3: Bảng kết quả thực nghiệm của số lượng UTXO

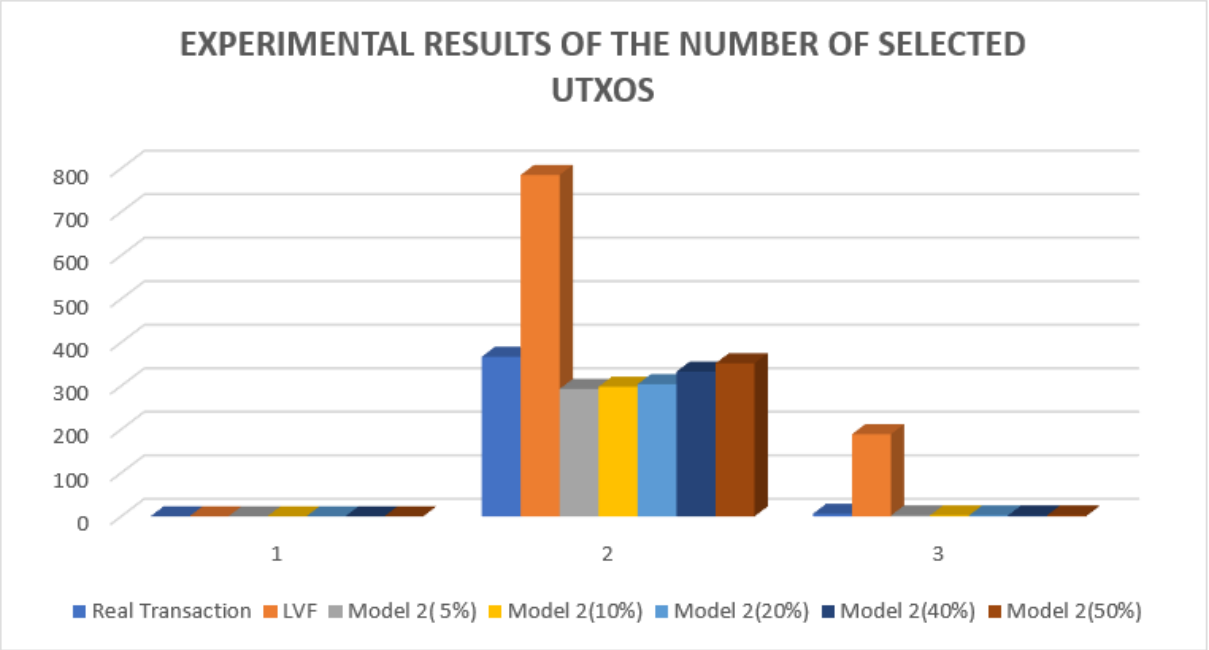

Figure 3: Biểu đồ kết quả thực nghiệm của số lượng UTXO

Method	**DS1**	**DS2**	**DS3**
Real Transition	0	256504	4928
LVF	0	318028	32012
Model 2(5%)	0	247932	4336
Model 2(10%)	0	248672	4336
Model 2(20%)	0	249526	4336
Model 2(40%)	0	253716	4336
Model 2(50%)	0	256528	4336

Table 4: So sánh hiệu suất hoạt động trong giới hạn của kích thước giao dịch

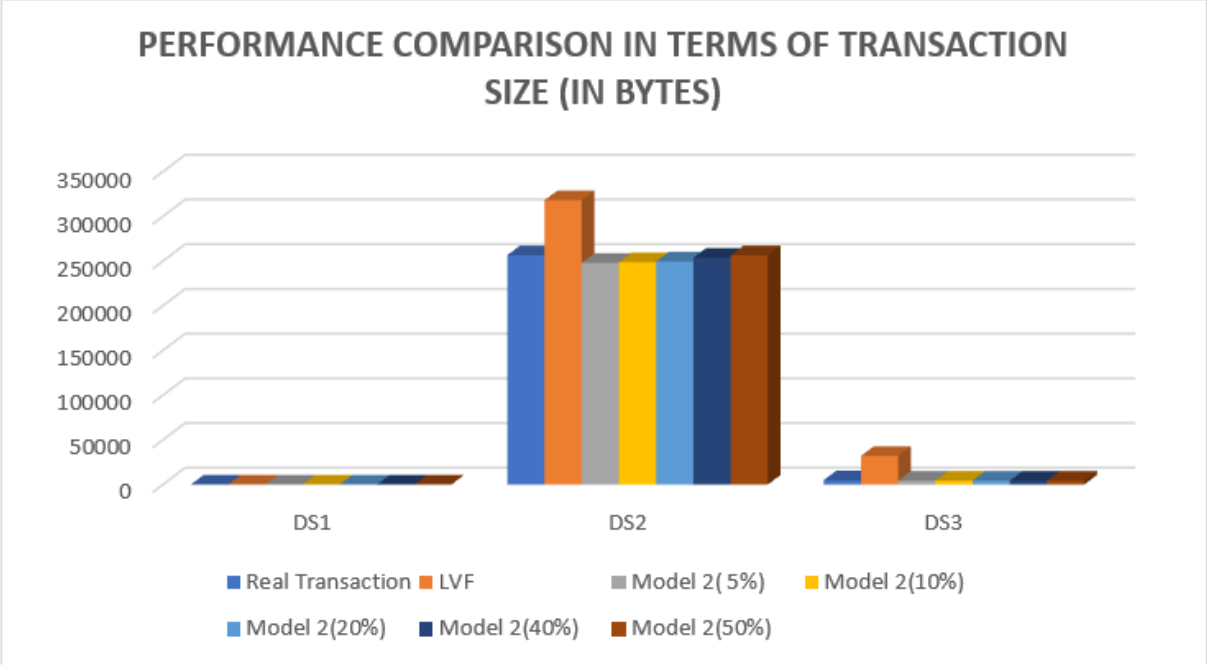

Figure 4: Biểu đồ so sánh hiệu suất hoạt động trong giới hạn của kích thước giao dịch

Bảng 3 và Hình 3 thể hiện rằng thuật toán LVF vượt trội hơn khả năng dọn sạch nhóm UTXO càng nhiều càng tốt, nhưng nó có tác dụng phụ đối với quy mô giao dịch có thể thấy trên Bảng 4 và Hình 4. LVF có thể tạo ra một giao dịch lớn mà chắc chắn chi phí cao cho các thợ mỏ. Về Model 2, kết quả của người giải kết hợp lựa chọn nhiều UTXO và kích thước giao dịch trong một hệ số phù hợp, các tham số γ 40% và 50% cho kết quả tốt hơn, nghĩa là chọn nhiều UTXO nhưng cũng đảm bảo rằng giao dịch được tạo với kích thước phù hợp.

5 Conclusion

Trong bài tập lớn này, chúng tôi đã đề xuất hai mô hình toán học để giải quyết hai mục tiêu thiết yếu khi tạo giao dịch mới trên blockchain. Mô hình đầu tiên giảm thiểu kích thước giao dịch để có thể tạo ra một khoản phí nhỏ cho nhiệm vụ khai thác chịu trách nhiệm xác thực giao dịch này trên mạng. Mô hình thứ hai được xây dựng để hạn chế sự bùng nổ của nhóm UTXO bằng cách chọn càng nhiều số lượng UTXO trong khi duy trì quy mô giao dịch để giúp người dùng trả chi phí phù hợp và phù hợp. Các thử nghiệm của chúng tôi đã cho thấy kết quả tốt hơn so với các chiến lược thực tế hiện tại, chiến lược HVF và LVF. Mặc dù các mô hình đề xuất của chúng tôi có thể áp dụng trong thực tế, chúng tôi cần thực hiện các thử nghiệm tiếp theo trên bộ dữ liệu lớn hơn để đo thời gian chạy và tiêu thụ tài nguyên, đó cũng là hai yếu tố quan trọng, đặc biệt là để triển khai trên thiết bị di động.

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).