Another coin bites the dust

Sergi Delgado Segura

Cristina Pérez-Solà, **Sergi Delgado-Segura**, Guillermo Navarro-Arribas, Jordi Herrera-Joancomartí
Another coin bites the dust: An analysis of dust in UTXO based cryptocurrencies
https://eprint.iacr.org/2018/513.pdf

sr_gi

Bitcoin fees 101

Bitcoin fees 101

- Bitcoin fees are the difference between the **sum of all input values** and the **sum of all output values**

Bitcoin fees 101

- Bitcoin fees are the difference between the **sum of all input values** and the **sum of all output values**

- The fees of each transaction can be **claimed by miners** when the transactions are included in a block

Bitcoin fees 101

- Bitcoin fees are the difference between the **sum of all input values** and the **sum of all output values**

- The fees of each transaction can be **claimed by miners** when the transactions are included in a block

- All unconfirmed transactions sit in the mempool waiting to be included in blocks

Bitcoin fees 101

- Bitcoin fees are the difference between the **sum of all input values** and the **sum of all output values**

- The fees of each transaction can be **claimed by miners** when the transactions are included in a block

- All unconfirmed transactions sit in the mempool waiting to be included in blocks

- Who decides the transaction inclusion order?

Bitcoin fees 101

- Bitcoin fees are the difference between the **sum of all input values** and the **sum of all output values**

- The fees of each transaction can be **claimed by miners** when the transactions are included in a block

- All unconfirmed transactions sit in the mempool waiting to be included in blocks

- Who decides the transaction inclusion order? **Miners**

Bitcoin fees 102

Bitcoin fees 102

- Incentive for miners to include your transaction in a block

Bitcoin fees 102

- Incentive for miners to include your transaction in a block

- The block size is limited, so the “better” the fee, the higher the chance of inclusion

Bitcoin fees 102

- Incentive for miners to include your transaction in a block

- The block size is limited, so the “better” the fee, the higher the chance of inclusion

- Does better means higher?

Bitcoin fees 102

- Incentive for miners to include your transaction in a block

- The block size is limited, so the “better” the fee, the higher the chance of inclusion

- Does better means higher? **Not always**

Calculating for the fees

- Two factors define the fee of a transaction:

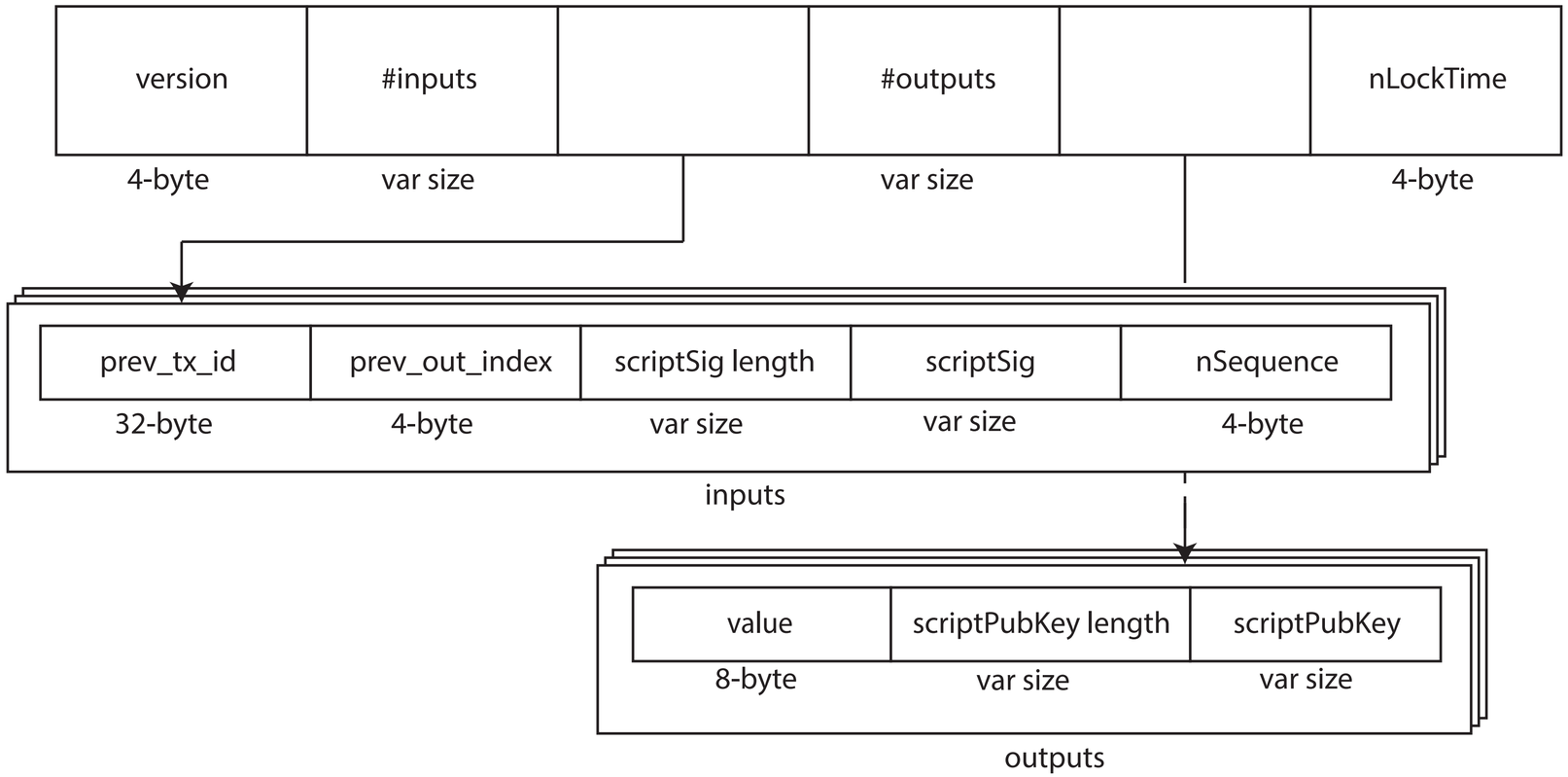
Calculating for the fees

- Two factors define the fee of a transaction:
  - Transaction size

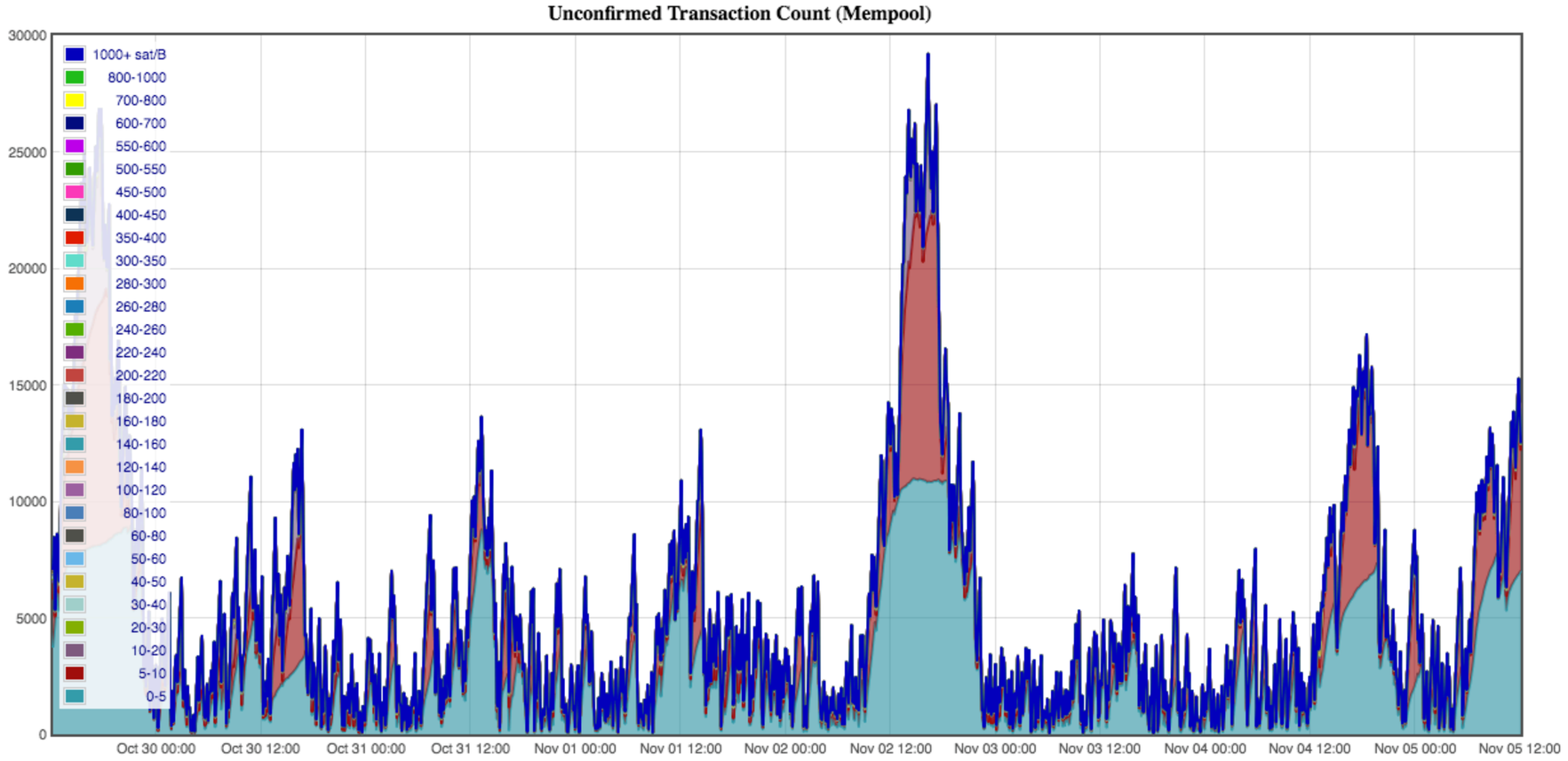
Calculating for the fees

- Two factors define the fee of a transaction:
  - Transaction size
  - Fee rate

Transaction size



Fee rate



Source: https://core.jochen-hoenicke.de

What if…

What if…

- the value of a transaction is low, but the size is high?

What if…

- the value of a transaction is low, but the size is high?

- the fee rate at the time of creating the transaction is too high?

What if…

- the value of a transaction is low, but the size is high?

- the fee rate at the time of creating the transaction is too high?

Dust definition

Dust definition

- The definition is coined by the Bitcoin Core

Dust definition

- The definition is coined by the Bitcoin Core

- Dust is computed in terms of outputs, instead of transactions

Dust definition

- The definition is coined by the Bitcoin Core

- Dust is computed in terms of outputs, instead of transactions

- An output is said to be dust **if the value it is holding is lower than the fees required to spend it**

Dust definition

- The definition is coined by the Bitcoin Core
- Dust is computed in terms of outputs, instead of transactions
- An output is said to be dust **if the value it is holding is lower than the fees required to spend it**
- Dust depends, therefore, in the state of the network

Transaction propagation

Transaction propagation

- Are then dust transactions invalid?

Transaction propagation

- Are then dust transactions invalid?

Transaction propagation

- Are then dust transactions invalid?

Transaction propagation

- Are then dust transactions invalid?

NO!

Transaction propagation

- Are then dust transactions invalid?

NO!

- Dust transactions are seen as uneconomical transactions, therefore they are not propagated by the Core client

Transaction propagation

- Are then dust transactions invalid?

NO!

- Dust transactions are seen as uneconomical transactions, therefore they are not propagated by the Core client

Transaction propagation

- Are then dust transactions invalid?

NO!

Relay rules $\neq$ **Consensus rules**

- Dust transactions are seen as uneconomical transactions, therefore they are not propagated by the Core client

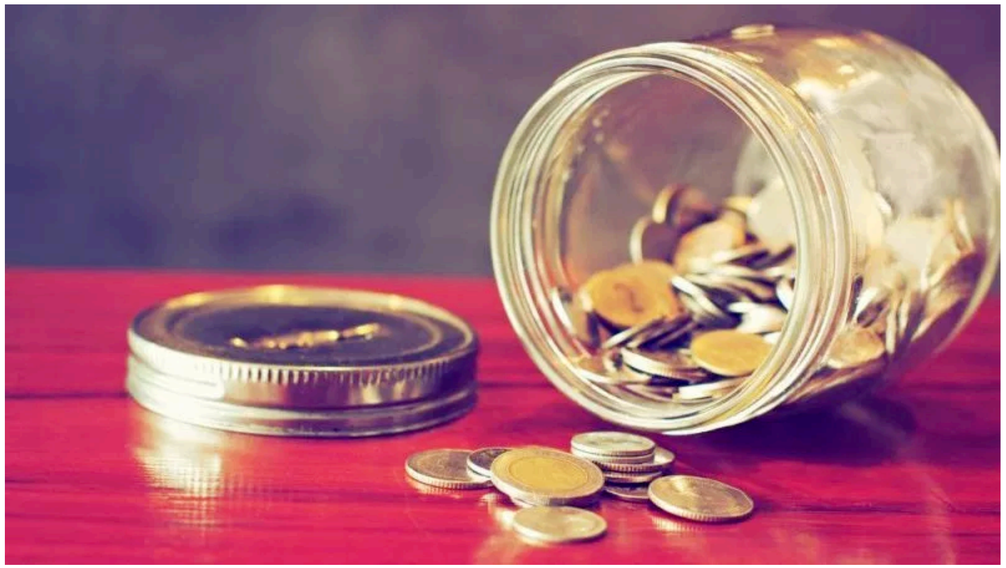
The usual suspect

The usual suspect

Really low value outputs

The usual suspect

Really low value outputs



Can we flag which uneconomical outputs?

Can we flag which uneconomical outputs?

The UTXO set

The UTXO set

- The UTXO is a collection of **all** UTXOs (Unspent Transaction Outputs)

The UTXO set

- The UTXO is a collection of **all** UTXOs (Unspent Transaction Outputs)

- Every (Bitcoin Core client) full node stores a copy of the set

The UTXO set

- The UTXO is a collection of **all** UTXOs (Unspent Transaction Outputs)
- Every (Bitcoin Core client) full node stores a copy of the set
- Each UTXO is an entry in the set

The UTXO set

- The UTXO is a collection of **all** UTXOs (Unspent Transaction Outputs)
- Every (Bitcoin Core client) full node stores a copy of the set
- Each UTXO is an entry in the set
- The value of the UTXO does not affect its size (**bigger value != bigger size**)

The UTXO set

- The UTXO is a collection of **all** UTXOs (Unspent Transaction Outputs)
- Every (Bitcoin Core client) full node stores a copy of the set
- Each UTXO is an entry in the set
- The value of the UTXO does not affect its size (**bigger value != bigger size**)
- In general, **the larger the output script, the bigger the UTXO size**

Analyzing the set

Analyzing the set

- By analyzing the set we can see which outputs are likely to be uneconomical under a given fee rate

Analyzing the set

- By analyzing the set we can see which outputs are likely to be uneconomical under a given fee rate

- Every Bitcoin (Core client) fork shares the same UTXO set format

Analyzing the set

- By analyzing the set we can see which outputs are likely to be uneconomical under a given fee rate

- Every Bitcoin (Core client) fork shares the same UTXO set format

- A tool to analyze them all

Analyzing the set

- By analyzing the set we can see which outputs are likely to be uneconomical under a given fee rate

- Every Bitcoin (Core client) fork shares the same UTXO set format

- A tool to analyze them all

STATUS

STATUS

- STATUS (STatistical Analysis Tool for Utxo Set)

STATUS

- STATUS (STatistical Analysis Tool for Utxo Set)

https://git.io/vAzHL

STATUS

- STATUS (STatistical Analysis Tool for Utxo Set)
- Open source tool (Python)

https://git.io/vAzHL

STATUS

- STATUS (STatistical Analysis Tool for Utxo Set)
- Open source tool (Python)
- Easy way to access, decode and analyze data from the UTXO set

https://git.io/vAzHL

STATUS

- STATUS (STatistical Analysis Tool for Utxo Set)
- Open source tool (Python)
- Easy way to access, decode and analyze data from the UTXO set
- We will see more about it during the working groups

https://git.io/vAzHL

The question(s)

The question(s)

- How many UTXOs are actually worth spending

The question(s)

- How many UTXOs are actually worth spending

- How much space is every full node devoting to store uneconomical outputs?

The metrics

The metrics

- We will define two metrics for uneconomical outputs:

The metrics

- We will define two metrics for uneconomical outputs:
    - **Dust**

The metrics

- We will define two metrics for uneconomical outputs:
    - **Dust**
    - **Unprofitable**

The metrics

- We will define two metrics for uneconomical outputs:
    - **Dust**
    - **Unprofitable**
- Both are defined as outputs that hold less value than the required fees

The metrics

- We will define two metrics for uneconomical outputs:
    - **Dust**
    - **Unprofitable**
- Both are defined as outputs that hold less value than the required fees
- Each one is computed using a different approach

Dust outputs

Dust outputs

To compute dust, **both the analyzed output and a P2PKH input** are used

Dust outputs

To compute dust, **both the analyzed output and a P2PKH input** are used

$$\textbf{is\_dust}(\mathfrak{out}) = \begin{cases} 1, & \mathfrak{out}_v < \mathfrak{f} \cdot (41 + 107/\alpha + \mathfrak{out}_s) \\ 0, & otherwise \end{cases}$$

Dust outputs

To compute dust, **both the analyzed output and a P2PKH input** are used

$$\textbf{is\_dust}(\mathfrak{out}) = \begin{cases} 1, & \mathfrak{out}_v < \mathfrak{f} \cdot (41 + 107/\alpha + \mathfrak{out}_s) \\ 0, & otherwise \end{cases}$$

where α is 4 for SegWit outputs, and 1 otherwise

Unprofitable outputs

Unprofitable outputs

To compute it, **only an input spending from the analyzed output** is used

Unprofitable outputs

To compute it, **only an input spending from the analyzed output** is used

$$\textbf{is\_unprofitable}(\mathfrak{out}) = \begin{cases} 1, & \mathfrak{out}_v < \mathfrak{f} \cdot \mathfrak{pred\_in}_s/\alpha \\ 0, & otherwise \end{cases}$$

Unprofitable outputs

To compute it, **only an input spending from the analyzed output** is used

$$\textbf{is\_unprofitable}(\mathfrak{out}) = \begin{cases} 1, & \mathfrak{out}_v < \mathfrak{f} \cdot \mathfrak{pred\_in}_s/\alpha \\ 0, & otherwise \end{cases}$$

$\mathfrak{pred\_in}_s$ is the predicted size of the input that will spend output $\mathfrak{out}$

Unprofitable outputs

To compute it, **only an input spending from the analyzed output** is used

$$\textbf{is\_unprofitable}(\mathfrak{out}) = \begin{cases} 1, & \mathfrak{out}_v < \mathfrak{f} \cdot \mathfrak{pred\_in}_s/\alpha \\ 0, & otherwise \end{cases}$$

$\mathfrak{pred\_in}_s$ is the predicted size of the input that will spend output $\mathfrak{out}$

and α is 4 for SegWit outputs, and 1 otherwise

Unprofitable outputs

To compute it, **only an input spending from the analyzed output** is used

$$\textbf{is\_unprofitable}(\mathfrak{out}) = \begin{cases} 1, & \mathfrak{out}_v < \mathfrak{f} \cdot \mathfrak{pred\_in}_s/\alpha \\ 0, & otherwise \end{cases}$$

$\mathfrak{pred\_in}_s$ is the predicted size of the input that will spend output $\mathfrak{out}$

and α is 4 for SegWit outputs, and 1 otherwise

Unprofitable outputs

To compute it, **only an input spending from the analyzed output** is used

$$\textbf{is\_unprofitable}(\mathfrak{out}) = \begin{cases} 1, & \mathfrak{out}_v < \mathfrak{f} \cdot \mathfrak{pred\_in}_s/\alpha \\ 0, & otherwise \end{cases}$$

We need a way of computing the input size

$\mathfrak{pred\_in}_s$ is the predicted size of the input that will spend output $\mathfrak{out}$

and α is 4 for SegWit outputs, and 1 otherwise

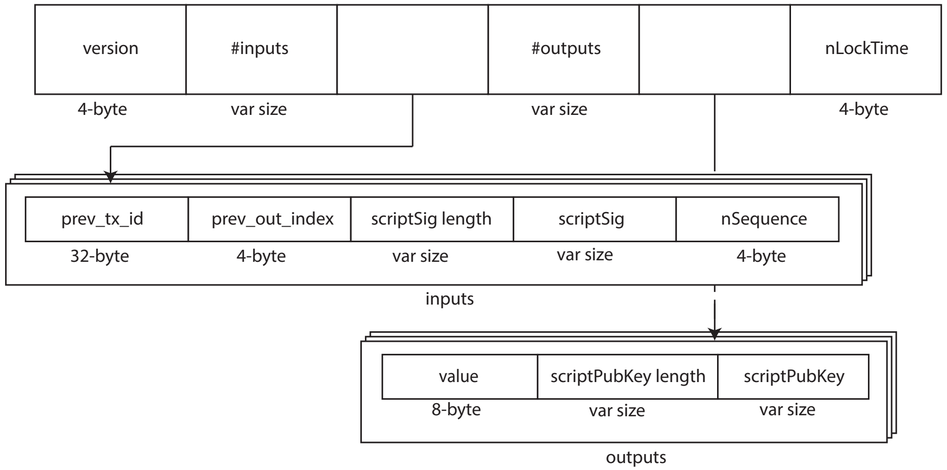
Computing the inputs size

Computing the inputs size

$$\mathfrak{pred\_in}_s = fixed\_size + variable\_size$$

Computing the inputs size

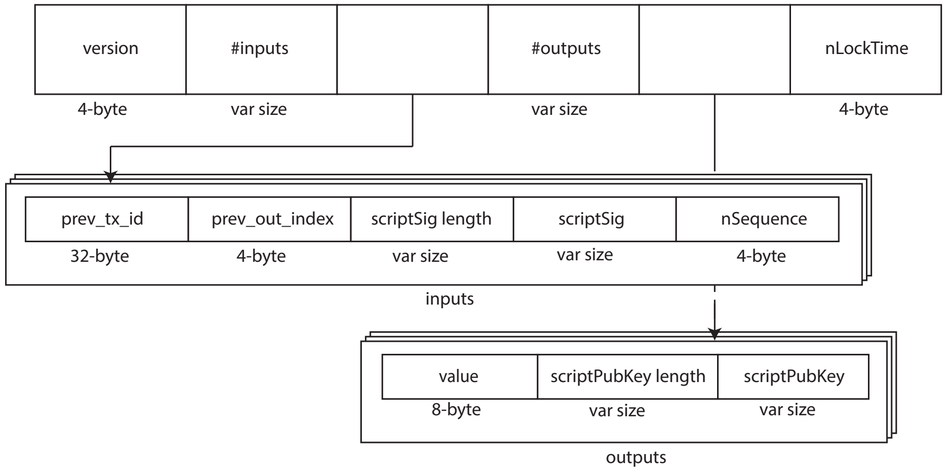
$$\mathfrak{pred\_in}_s = fixed\_size + variable\_size$$



Computing the inputs size

$$\mathfrak{pred\_in}_s = fixed\_size + variable\_size$$

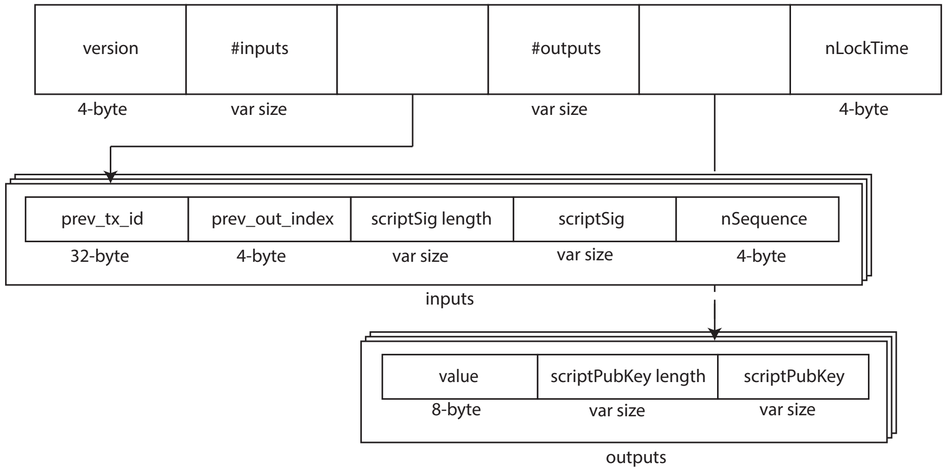
$$fixed\_size = outpoint + nSequence = 40\,bytes$$



Computing the inputs size

$$\mathfrak{pred\_in}_s = fixed\_size + variable\_size$$

$$fixed\_size = outpoint + nSequence = 40\,bytes$$

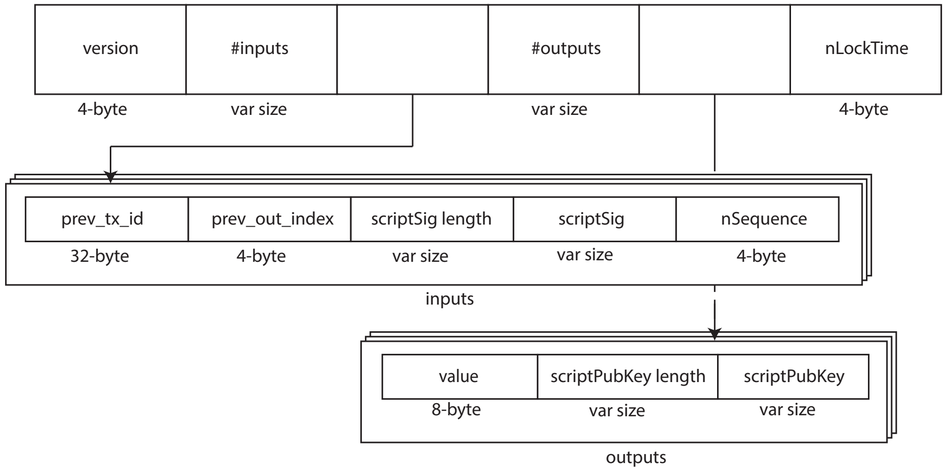


Computing the inputs size

$$\mathfrak{pred\_in}_s = fixed\_size + variable\_size$$

$$fixed\_size = outpoint + nSequence = 40\,bytes$$

$$variable\_size = scriptSig\_len + scriptSig$$



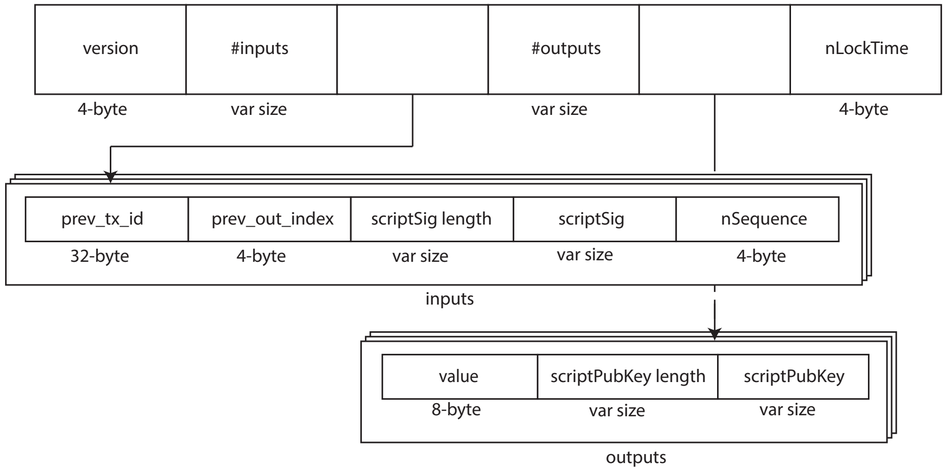
Computing the inputs size

$$\mathfrak{pred\_in}_s = fixed\_size + variable\_size$$

$$fixed\_size = outpoint + nSequence = 40\,bytes$$

$$variable\_size = scriptSig\_len + scriptSig$$

Depends on the UTXO type



Variable size (non-SegWit)

Variable size (non-SegWit)

- Pay-to-PubKey (P2PK) outputs:

Variable size (non-SegWit)

- **Pay-to-PubKey (P2PK) outputs:**
    - PUSH sig (1 byte) + sig (71-73 bytes)

Variable size (non-SegWit)

- **Pay-to-PubKey (P2PK) outputs:**
    - PUSH sig (1 byte) + sig (71-73 bytes)
- **Pay-to-PubKeyHash (P2PKH) outputs:**

Variable size (non-SegWit)

- **Pay-to-PubKey (P2PK) outputs:**
  - PUSH sig (1 byte) + sig (71-73 bytes)
- **Pay-to-PubKeyHash (P2PKH) outputs:**
  - PUSH sig (1 byte) + sig (71-73 bytes) + PUSH pk (1 byte) + pk (33-65 bytes)

Variable size (non-SegWit)

- **Pay-to-PubKey (P2PK) outputs:**
    - PUSH sig (1 byte) + sig (71-73 bytes)
- **Pay-to-PubKeyHash (P2PKH) outputs:**
    - PUSH sig (1 byte) + sig (71-73 bytes) + PUSH pk (1 byte) + pk (33-65 bytes)
- **Pay-to-multisig (P2MS) outputs:**

Variable size (non-SegWit)

- **Pay-to-PubKey (P2PK) outputs:**
  - PUSH sig (1 byte) + sig (71-73 bytes)
- **Pay-to-PubKeyHash (P2PKH) outputs:**
  - PUSH sig (1 byte) + sig (71-73 bytes) + PUSH pk (1 byte) + pk (33-65 bytes)
- **Pay-to-multisig (P2MS) outputs:**
  - OP_0 (1 byte) + [PUSH sig (1 byte) + sig (71-73 bytes)] * req_sigs (1-20)

Variable size (non-SegWit)

- **Pay-to-PubKey (P2PK) outputs:**
  - PUSH sig (1 byte) + sig (71-73 bytes)
- **Pay-to-PubKeyHash (P2PKH) outputs:**
  - PUSH sig (1 byte) + sig (71-73 bytes) + PUSH pk (1 byte) + pk (33-65 bytes)
- **Pay-to-multisig (P2MS) outputs:**
  - OP_0 (1 byte) + [PUSH sig (1 byte) + sig (71-73 bytes)] * req_sigs (1-20)
- **Pay-to-ScriptHash (P2SH) outputs:**

Variable size (non-SegWit)

- **Pay-to-PubKey (P2PK) outputs:**
  - PUSH sig (1 byte) + sig (71-73 bytes)
- **Pay-to-PubKeyHash (P2PKH) outputs:**
  - PUSH sig (1 byte) + sig (71-73 bytes) + PUSH pk (1 byte) + pk (33-65 bytes)
- **Pay-to-multisig (P2MS) outputs:**
  - OP_0 (1 byte) + [PUSH sig (1 byte) + sig (71-73 bytes)] * req_sigs (1-20)
- **Pay-to-ScriptHash (P2SH) outputs:**
  - ∅

Variable size (SegWit)

Variable size (SegWit)

- Pay-to-Witness-Public-Key-Hash (P2WPKH) outputs:

Variable size (SegWit)

- **Pay-to-Witness-Public-Key-Hash (P2WPKH) outputs:**
    - PUSH sig (1 byte) + sig (71-73 bytes) + PUSH pk (1 byte) + pk (33 bytes)

Variable size (SegWit)

- **Pay-to-Witness-Public-Key-Hash (P2WPKH) outputs:**
    - PUSH sig (1 byte) + sig (71-73 bytes) + PUSH pk (1 byte) + pk (33 bytes)
- **Pay-to-Witnes-ScriptHash (P2WSH) outputs:**

Variable size (SegWit)

- **Pay-to-Witness-Public-Key-Hash (P2WPKH) outputs:**
    - PUSH sig (1 byte) + sig (71-73 bytes) + PUSH pk (1 byte) + pk (33 bytes)
- **Pay-to-Witnes-ScriptHash (P2WSH) outputs:**
    - ∅

Variable size (SegWit)

- **Pay-to-Witness-Public-Key-Hash (P2WPKH) outputs:**
    - PUSH sig (1 byte) + sig (71-73 bytes) + PUSH pk (1 byte) + pk (33 bytes)
- **Pay-to-Witnes-ScriptHash (P2WSH) outputs:**
    - ∅

Witness scripts discounted α = 1/4

Two metrics for non-profitable

Two metrics for non-profitable

- **Lower bound (unprofitable_low):** Takes into account the minimum size of the input

Two metrics for non-profitable

- **Lower bound (unprofitable_low):** Takes into account the minimum size of the input
    - Signatures: 71 bytes

Two metrics for non-profitable

- **Lower bound (unprofitable_low):** Takes into account the minimum size of the input
    - Signatures: 71 bytes
    - Unknown variable size: Not counted

Two metrics for non-profitable

- **Lower bound (unprofitable_low):** Takes into account the minimum size of the input
    - Signatures: 71 bytes
    - Unknown variable size: Not counted
- **Estimation (unprofitable_est):** Estimates the size of the input using blockchain data

Two metrics for non-profitable

- **Lower bound (unprofitable_low):** Takes into account the minimum size of the input
    - Signatures: 71 bytes
    - Unknown variable size: Not counted
- **Estimation (unprofitable_est):** Estimates the size of the input using blockchain data
    - Signatures: 72 bytes

Two metrics for non-profitable

- **Lower bound (unprofitable_low):** Takes into account the minimum size of the input
    - Signatures: 71 bytes
    - Unknown variable size: Not counted
- **Estimation (unprofitable_est):** Estimates the size of the input using blockchain data
    - Signatures: 72 bytes

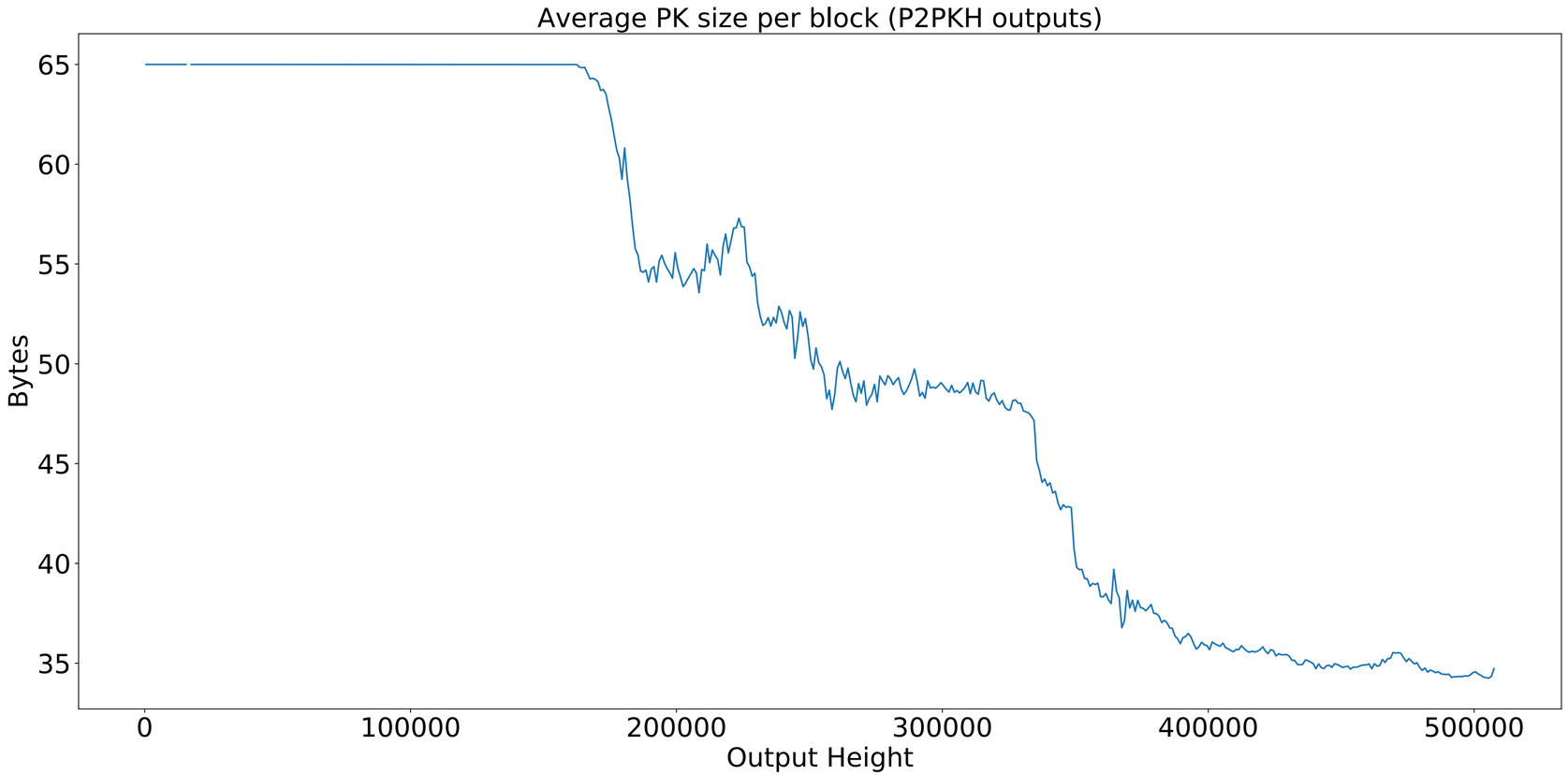
Two metrics for non-profitable

- **Lower bound (unprofitable_low):** Takes into account the minimum size of the input
  - Signatures: 71 bytes
  - Unknown variable size: Not counted
- **Estimation (unprofitable_est):** Estimates the size of the input using blockchain data
  - Signatures: 72 bytes **Most probable size for an ECDSA signature**

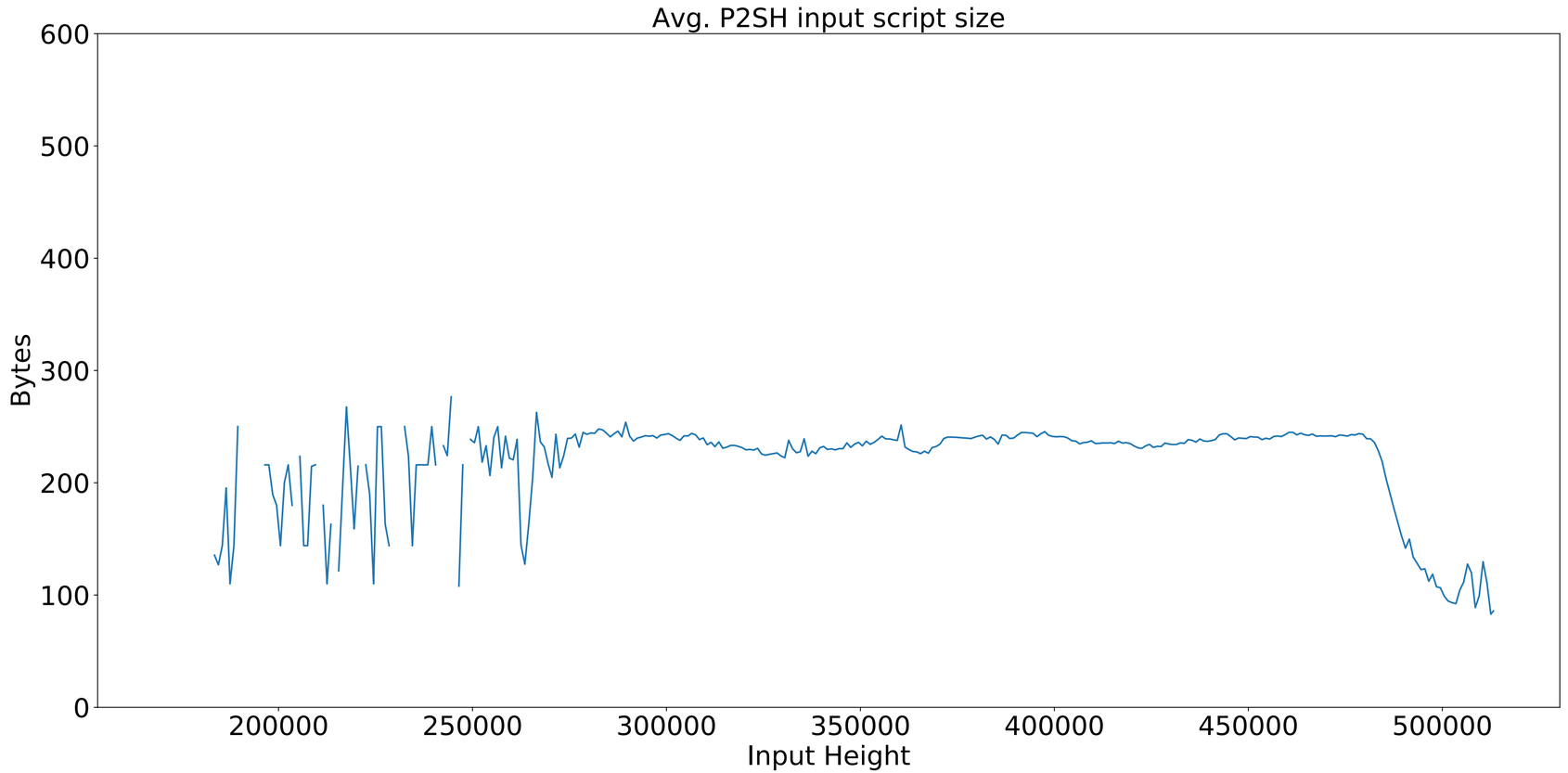
Two metrics for non-profitable

- **Lower bound (unprofitable_low):** Takes into account the minimum size of the input
  - Signatures: 71 bytes
  - Unknown variable size: Not counted
- **Estimation (unprofitable_est):** Estimates the size of the input using blockchain data
  - Signatures: 72 bytes **Most probable size for an ECDSA signature**
  - Uses BlockSci (public key sizes and redeem scripts sizes)

BlockSci data (PKs)

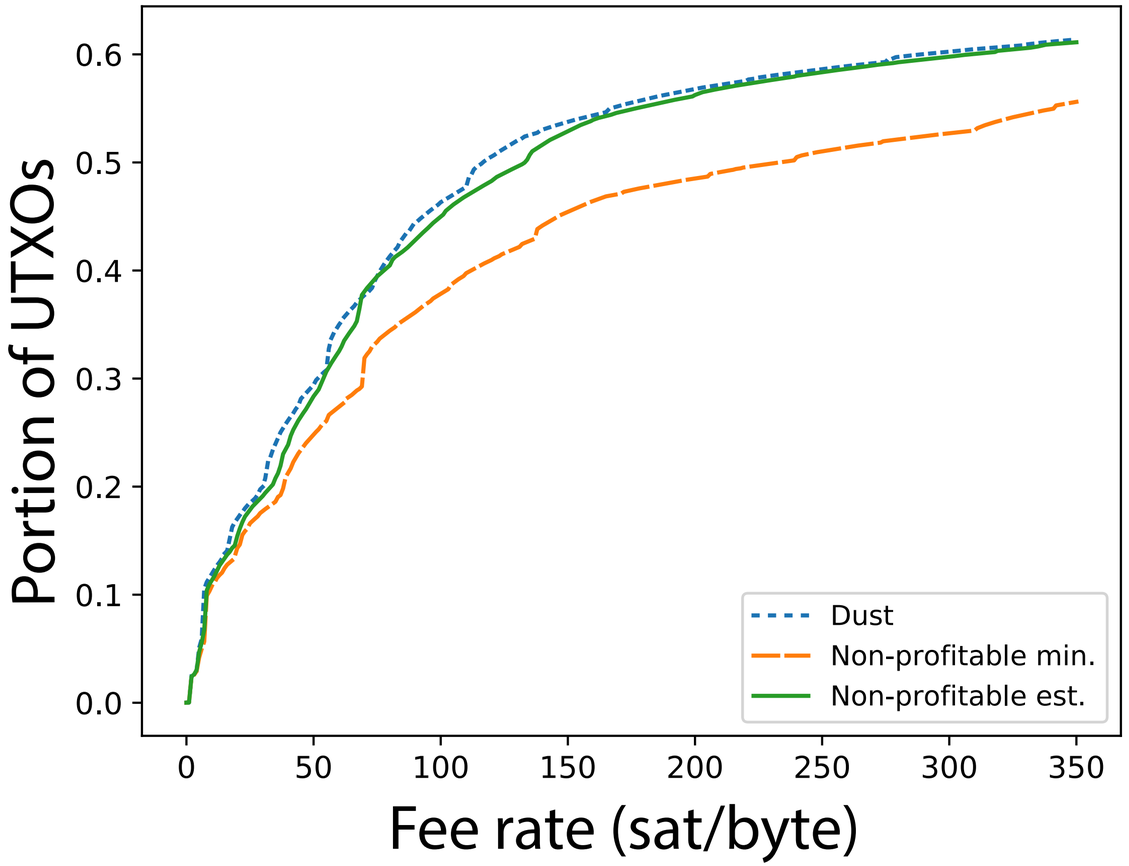


BlockSci data (redeem scripts)

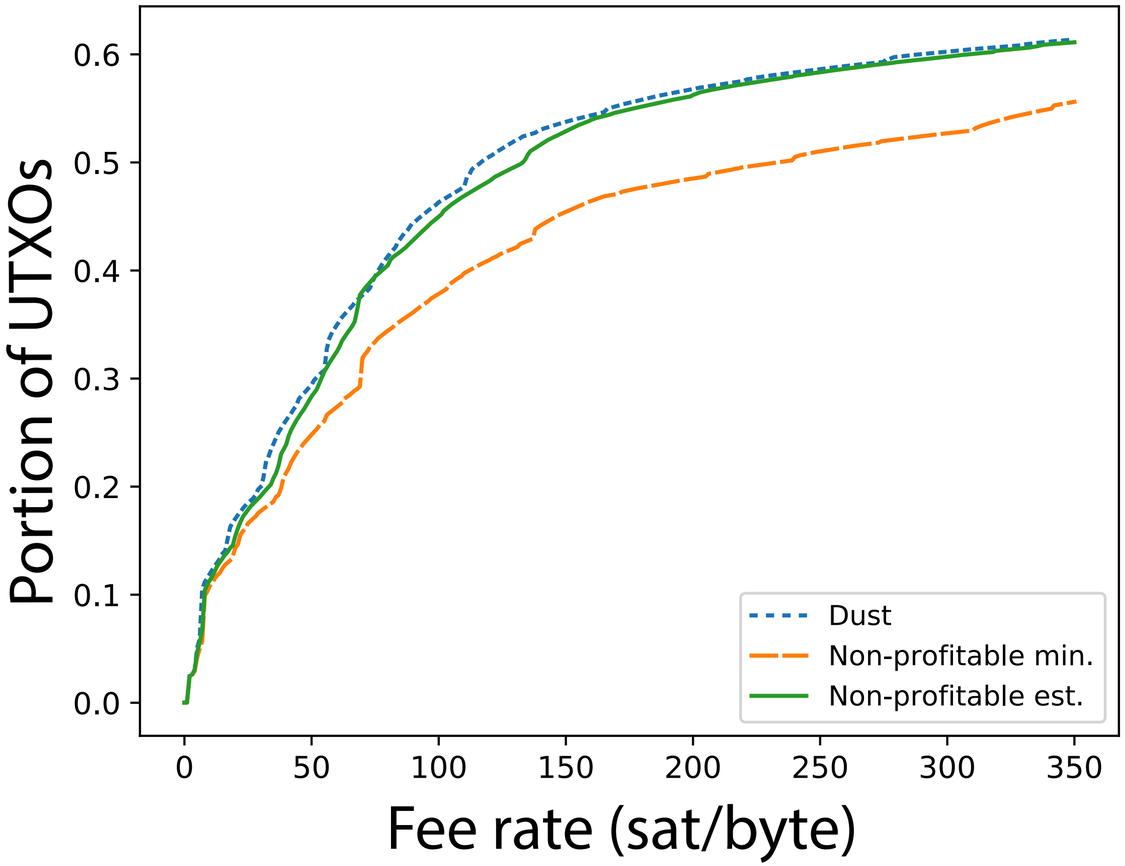


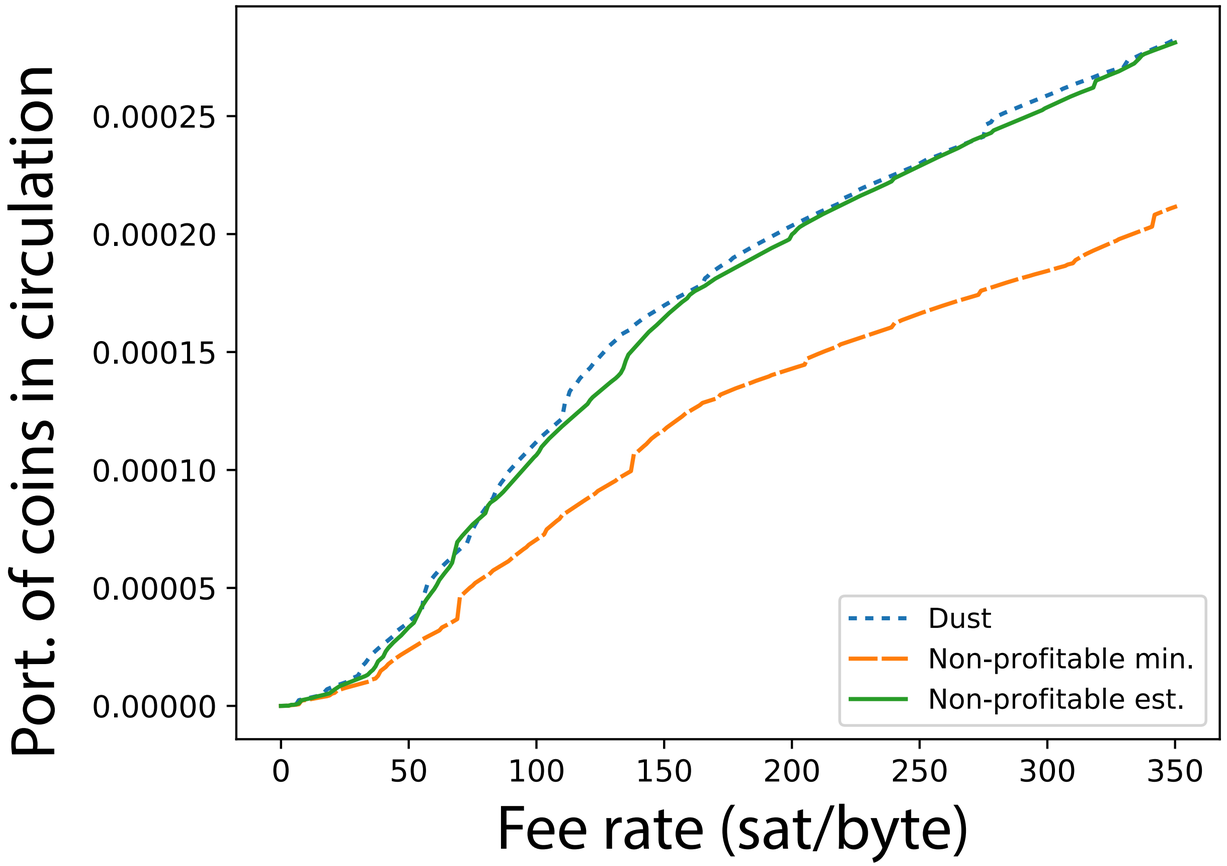
Results - single snapshot

Results - single snapshot



Results - single snapshot





Results - implications

Results - implications

- 1/2 of the outputs are dust/unprofitable at 100-125 sat/byte (30M UTXOS)

Results - implications

- 1/2 of the outputs are dust/unprofitable at 100-125 sat/byte (30M UTXOS)

- It represents around 1/2 of the size of the UTXO set (1.7 GB)

Results - implications

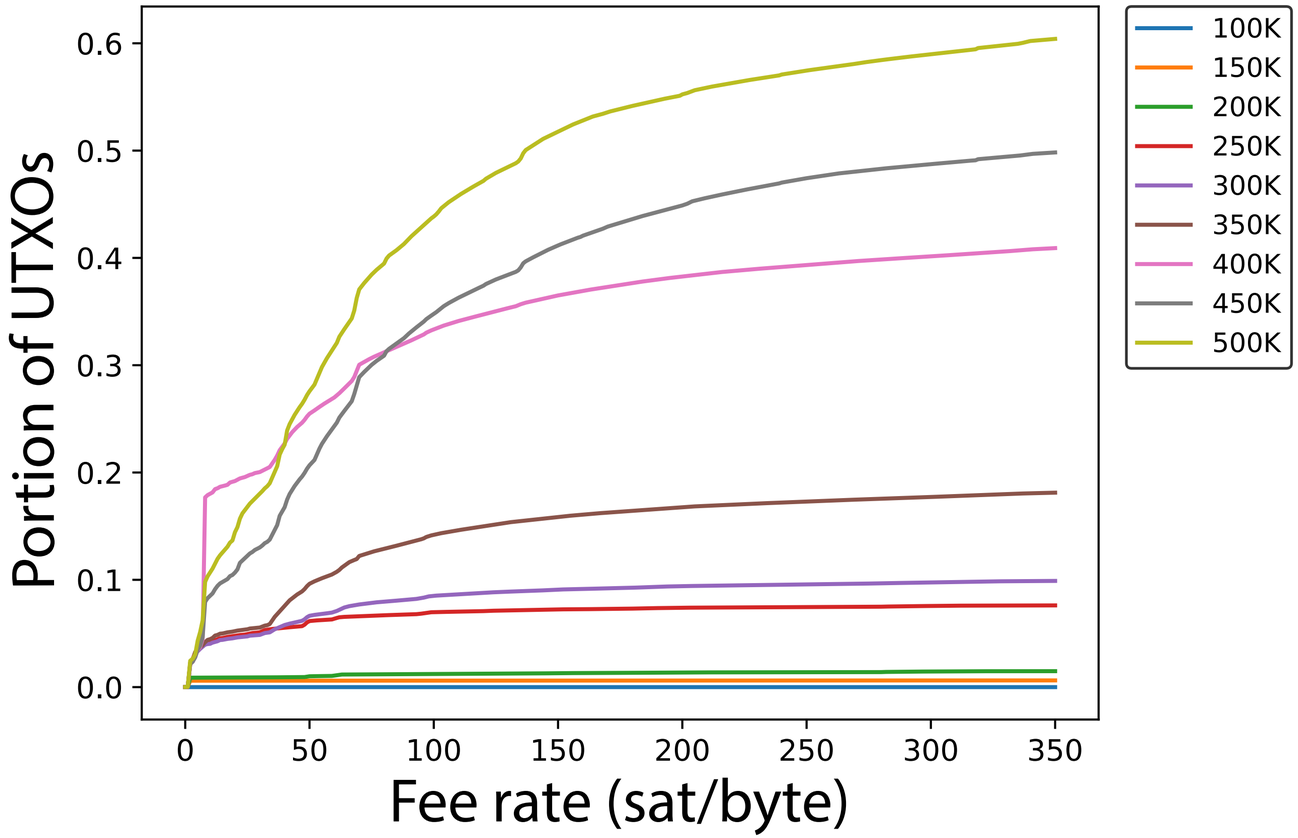
- 1/2 of the outputs are dust/unprofitable at 100-125 sat/byte (30M UTXOS)
- It represents around 1/2 of the size of the UTXO set (1.7 GB)
- Its total value is almost worthless (0.015% of total value)

Results - implications

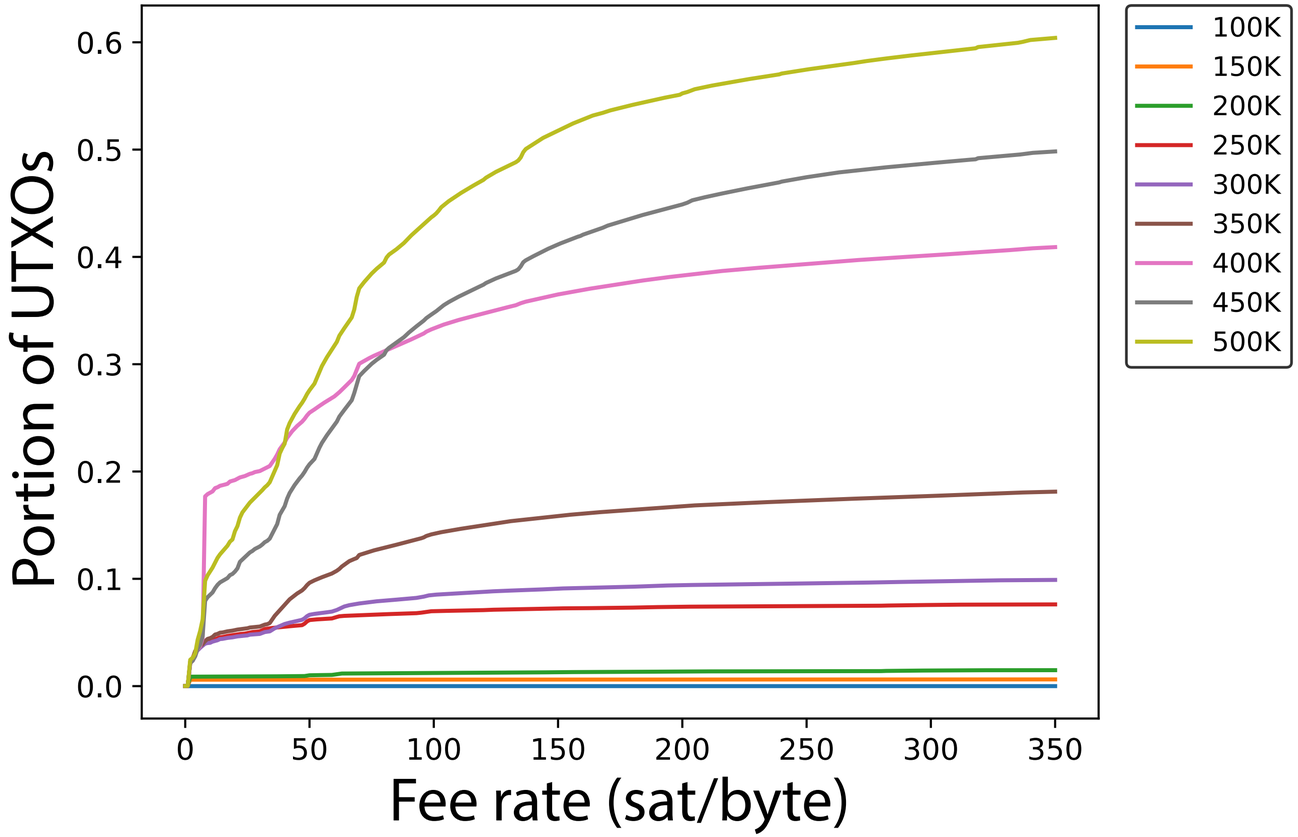
- 1/2 of the outputs are dust/unprofitable at 100-125 sat/byte (30M UTXOS)
- It represents around 1/2 of the size of the UTXO set (1.7 GB)
- Its total value is almost worthless (0.015% of total value) $\approx 252740\,BTC$

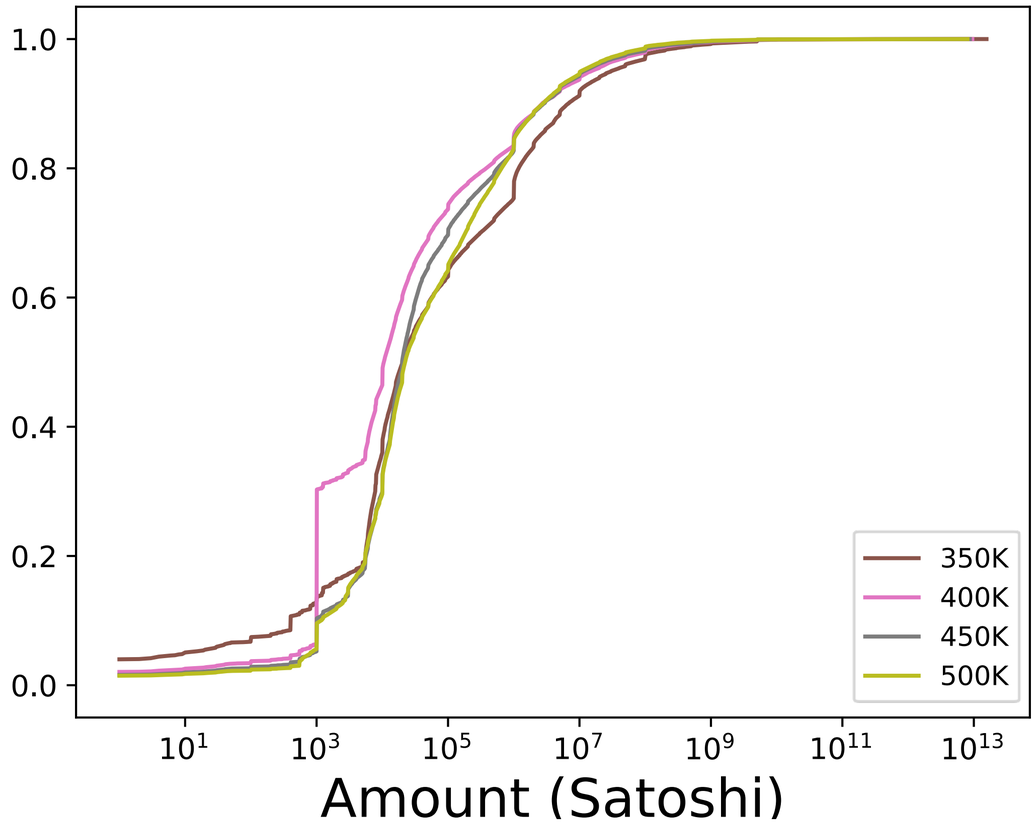
Results - evolution

Results - evolution



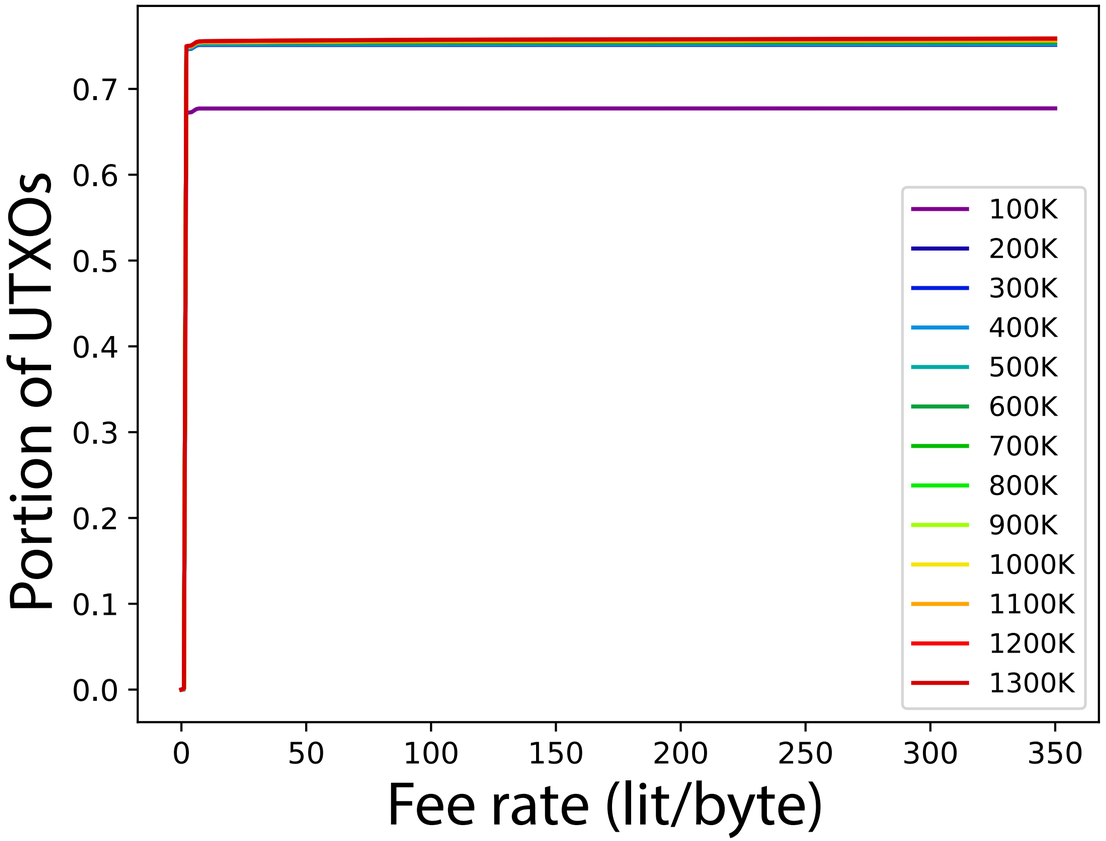
Results - evolution



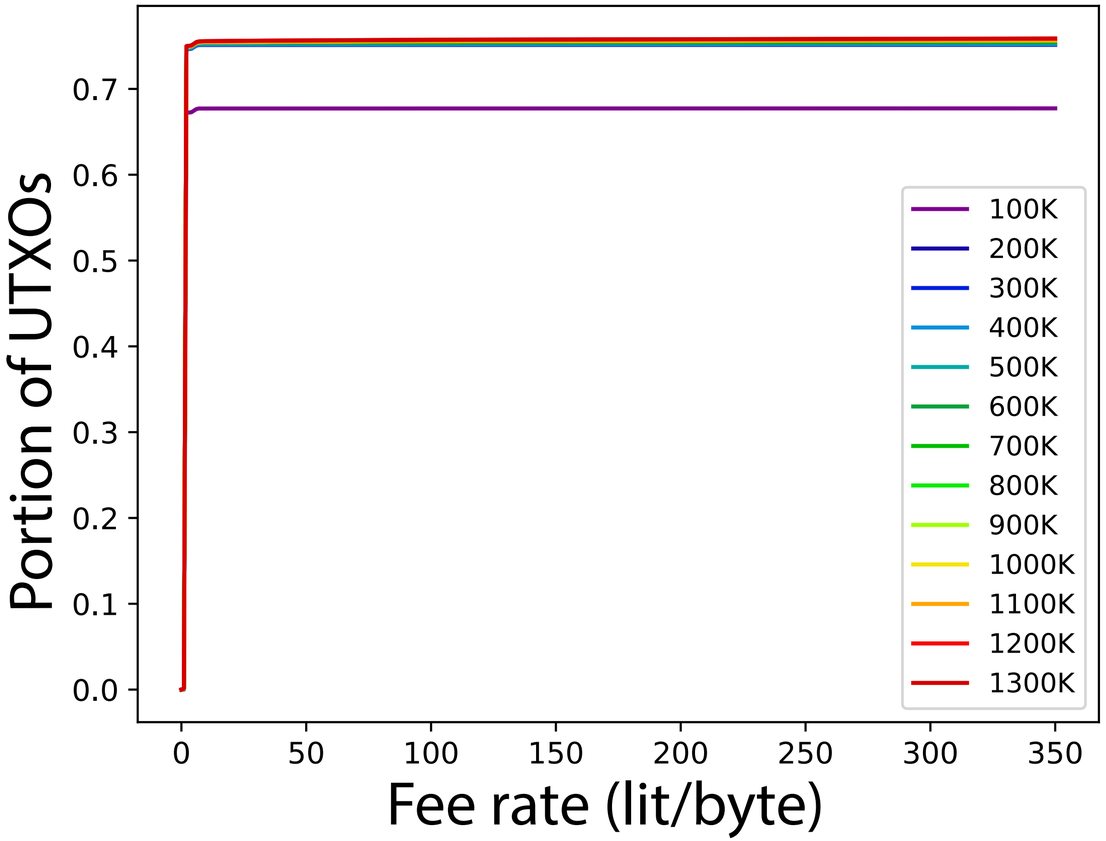


Results - Litecoin (extreme case)

Results - Litecoin (extreme case)

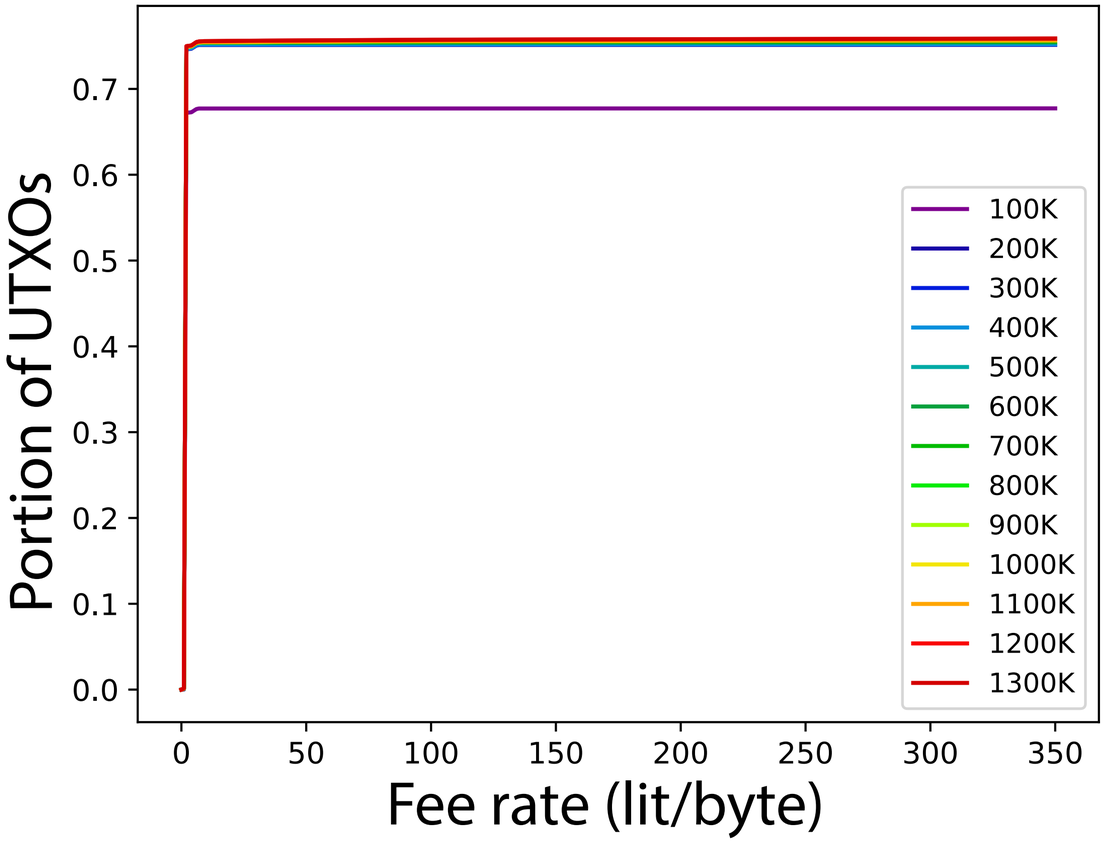


Results - Litecoin (extreme case)



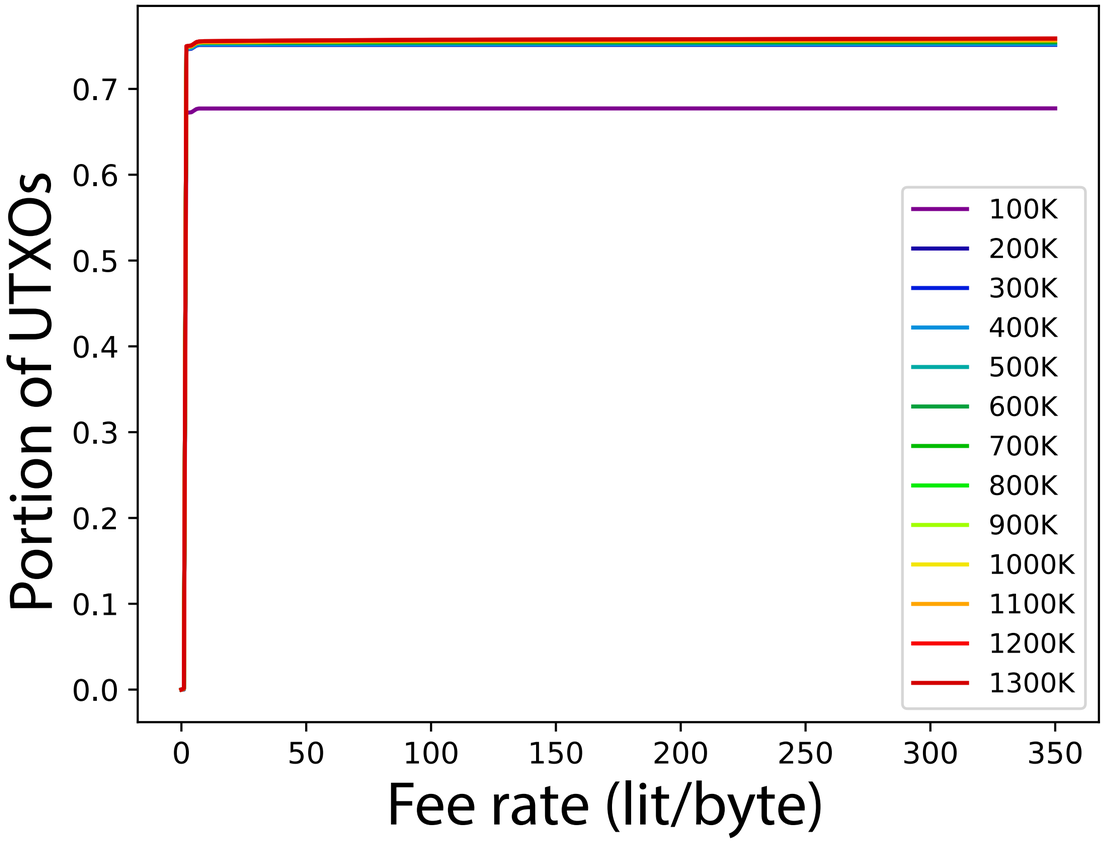
- 70%+ UTXOs are dust for $\approx 1\ lit/byte$

Results - Litecoin (extreme case)



- 70%+ UTXOs are dust for $\approx 1\ lit/byte$
- 50% are worth exactly 1 litoshi

Results - Litecoin (extreme case)



- 70%+ UTXOs are dust for $\approx 1\ lit/byte$
- 50% are worth exactly 1 litoshi
- Spam attack 2011/2012

Not your keys, not your coins

65

Posted by u/coblee **Litecoin Founder** 25 days ago

What should we do about the 50+% of Litecoin's UTXO which are 1 litoshi?

Recently, Sergi Delgado Segura and Jameson Lopp has brought this an issue to the forefront: https://twitter.com/lopp/status/1048380552691310592 The issue is that more than 50% of Litecoin's UTXO set is unspendable because they are only 1 litoshi (0.00000001 LTC) each.

Background

In late 2011 through April 2012, Litecoin was attacked with various forms of spam/dust attacks that bloated the blockchain and unspent transaction output (UTXO) set. For some context, you can read up more about what happened and how we handled it: https://bitcointalk.org/index.php?topic=47417.msg626303#msg626303

Bottom line is that the attacker was able to create a lot of dust transactions for little to no cost. Here's an example transaction: https://blockchair.com/litecoin/transaction/c3ed2cea60de33cb59c8ea6f053930e9c06cd7b4805e70ef60173c07dc2e00b9

In total there are 12,272,034 1-litoshi outputs that were created before 4/1/2012. (https://blockchair.com/litecoin/outputs?q=is_spent(false),value(1),time(..2012-04-01)#)

This is out of about 23,538,000 UTXOs in total today. So about 52% of the whole UTXO set is 1-litoshi spam. Total value of all these outpus is 0.12272014 LTC, which is about $7 today.

Conclusions

Conclusions

- A fairly big percentage of the UTXO set is by dust

Conclusions

- A fairly big percentage of the UTXO set is by dust

- The current implementation of the UTXO set can grow unbounded

Conclusions

- A fairly big percentage of the UTXO set is by dust
- The current implementation of the UTXO set can grow unbounded
- The bigger the set, the less suitable running a full node in optimal conditions

Conclusions

- A fairly big percentage of the UTXO set is by dust
- The current implementation of the UTXO set can grow unbounded
- The bigger the set, the less suitable running a full node in optimal conditions
- Spam attacks can be performed to make the set grow

Solutions

Solutions

- TXO commitments (Peter Todd)

Solutions

- TXO commitments (Peter Todd)

- High-performance merkle set (Bram Cohen)

Solutions

- TXO commitments (Peter Todd)
- High-performance merkle set (Bram Cohen)
- RSA accumulators for UTXOS (Benedikt Bünz, Benjamin Fisch and Dan Boneh)

Solutions

- TXO commitments (Peter Todd)
- High-performance merkle set (Bram Cohen)
- RSA accumulators for UTXOS (Benedikt Bünz, Benjamin Fisch and Dan Boneh)
- UTXO accumulators (Tadge Dryja)

Solutions

- TXO commitments (Peter Todd)
- High-performance merkle set (Bram Cohen)
- RSA accumulators for UTXOS (Benedikt Bünz, Benjamin Fisch and Dan Boneh)
- UTXO accumulators (Tadge Dryja)
- May be more

What can YOU do?

What can YOU do?

- Output consolidation when fees are low

What can YOU do?

- Output consolidation when fees are low

- Good coin selection algorithm (specially for exchanges)

Another coin bites the dust

Sergi Delgado Segura

Cristina Pérez-Solà, **Sergi Delgado-Segura**, Guillermo Navarro-Arribas, Jordi Herrera-Joancomartí
Another coin bites the dust: An analysis of dust in UTXO based cryptocurrencies
https://eprint.iacr.org/2018/513.pdf

sr_gi