Money Laundering Detection: Unsupervised Analysis on Banking Transaction Data

Faculty of Information Engineering, Informatics, and Statistics
Master Degree in Data Science

Candidate:
Cristiano Di Crescenzo

Supervisor:
Prof. Stefano Leonardi

Co-Supervisor:
Ph.D. Adriano Fazzone

External Supervisor:
Ph.D. Raffaele Miele

The Money Laundering Detection Problem

Money laundering is the process of transforming the profits of crime and corruption into ostensibly "legitimate" assets.

- 3 steps:

  - **Placement:** Initial entry of the "dirty" cash or proceeds of crime into the financial system.

  - **Layering:** Separate the illicit money from its source.

  - **Integration:** Money is returned to the criminal from what seem to be legitimate sources.

Bankitalia's Rules for Money Laundering Involving Groups of Cards

- **Placement**: Initial entry of the "dirty" cash or proceeds of crime into the financial system.

- **Layering**: Separate the illicit money from its source.

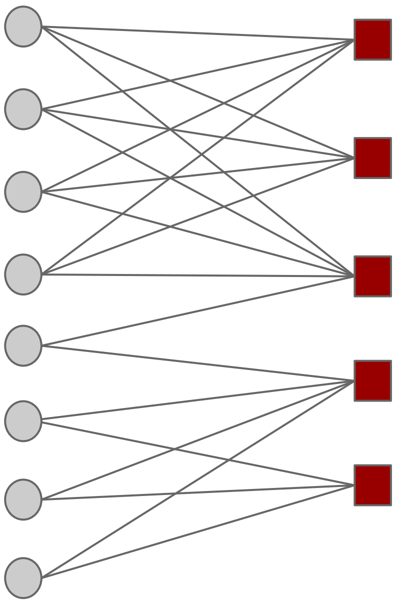
  - Transactions of relevant amount, involving cash
  - Transactions carried out at the same operative point (or neighboring operative points)
  - Transactions carried out inside a narrow time frame (inside a temporal interval of 60 minutes, for instance)

- **Integration**: Money is returned to the criminal from what seem to be legitimate sources.

State of the Art**: GIANOS** → find anomalies related to single cards behaviors, elaborated by hundreds of pre-established rules

Dataset

- **Real Data** (25 days of transactions)
- Each record is a financial operation (cash advance, purchase of goods, ...)
  `[ID_card, channel, ID_operative_point, timestamp, transaction_amount, transaction_type]`
- Focus the attention just in cash advance operation
- Dataset Size:
  #transactions: `17.8M`
  #cards: `5.1M`
  #ATMs: `135K`



Bankitalia's Rules for Money Laundering Involving Groups of Cards

- <u>**Transactions of relevant amount, involving cash**</u>

- Transactions carried out at the same operative point (or neighboring operative points)

- Transactions carried out inside a narrow time frame (inside a temporal interval of 60 minutes, for instance)

Transactions of Relevant Amount

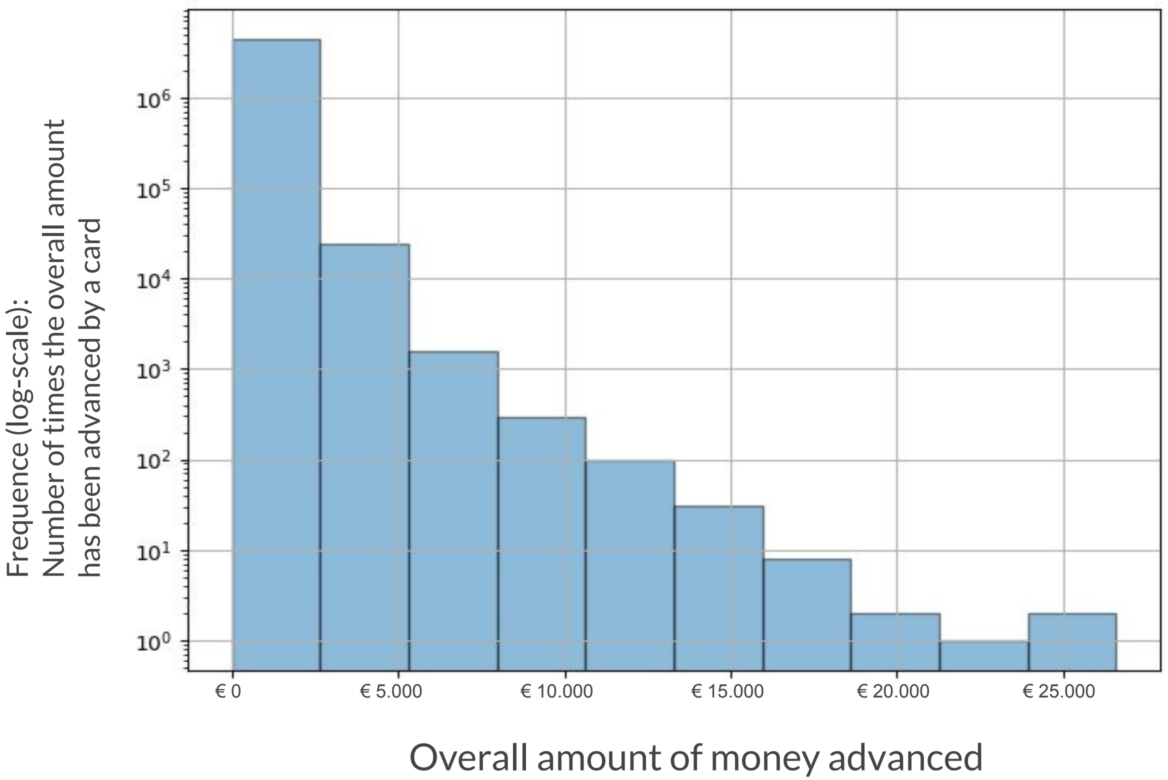
Transactions of interest:

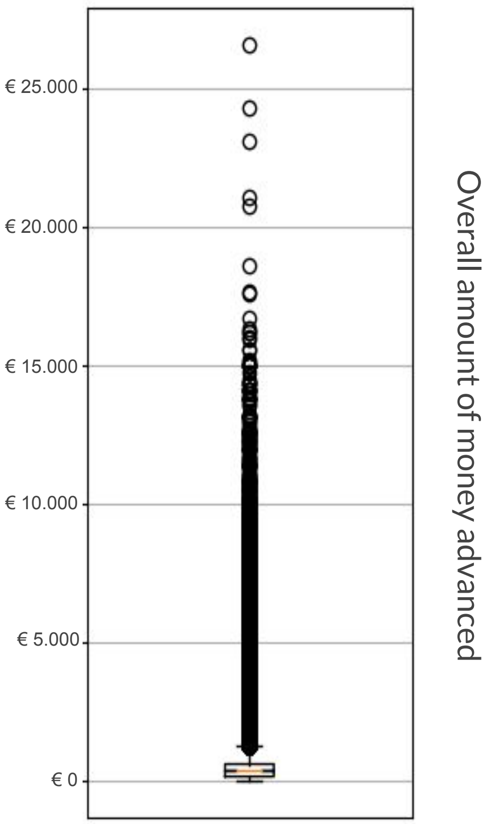
- high amount of money
- made by the most active cards

Method:

- Detect the most active cards through a data driven threshold (?€)
- Exclude the single transactions of amount lower than a second data driven threshold (?€)

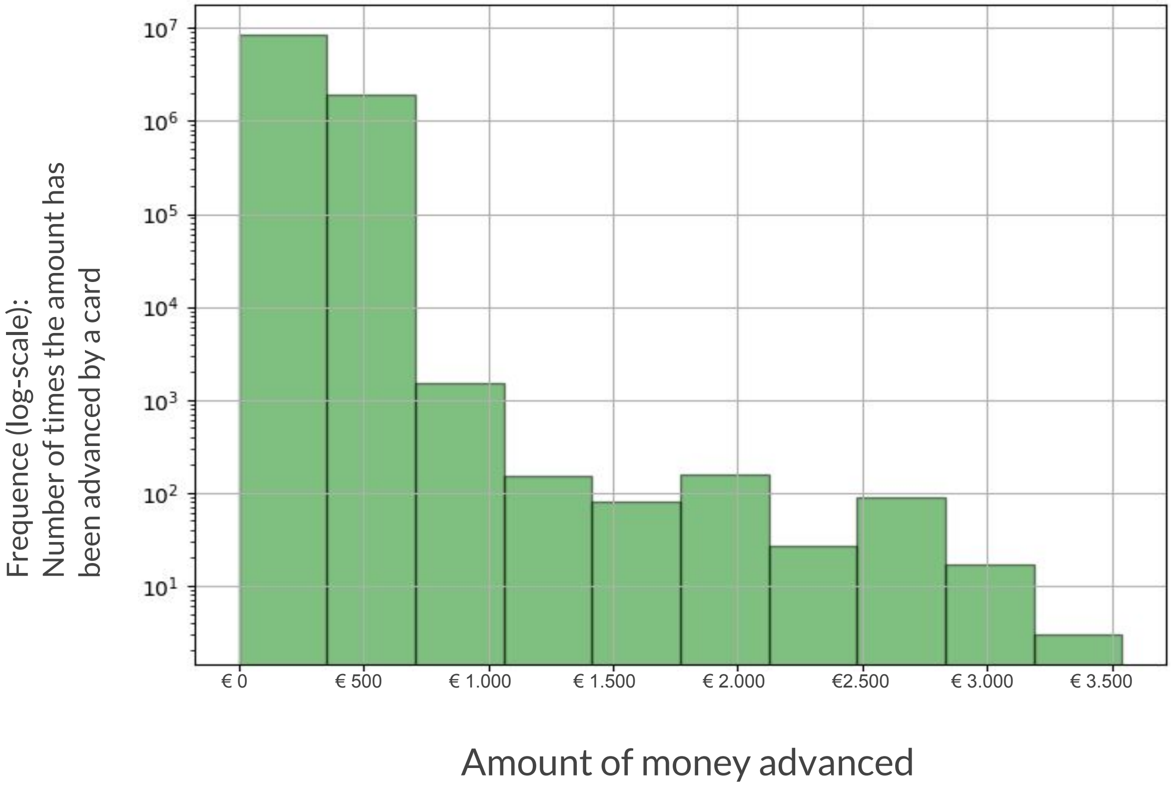
Distribution of Overall Advanced Amount of Money

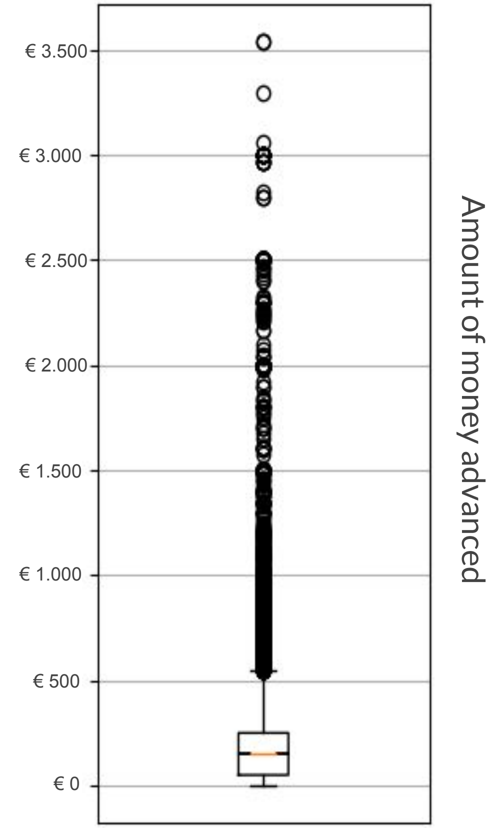




- First quartile ⇒ Data driven threshold ⇒ € 150
- FROM `17.8M` transactions, `5.1M` cards
  TO `10.4M` transactions, `3.4M` cards

Distribution of Advanced Amount of Money





- First quartile ⇒ Data driven threshold ⇒ € 50

- FROM `10.4M` transactions, `3.4M` cards
  TO `9M` transactions, `3.4M` cards

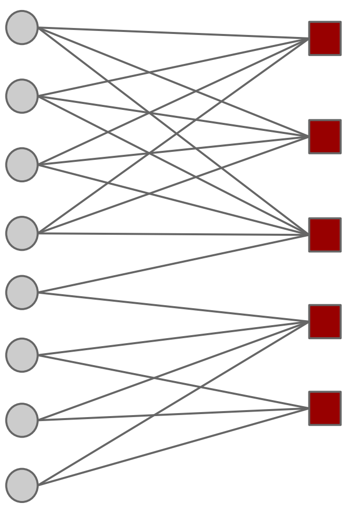
Transactions of Relevant Amount

Transactions of interest:

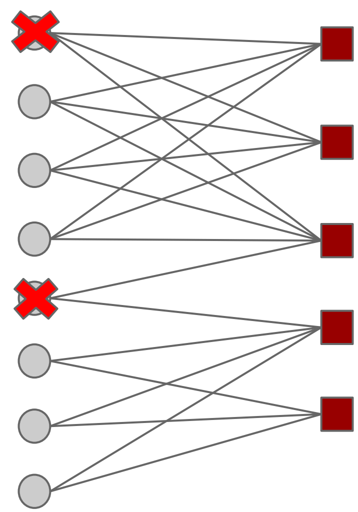
- high amount of money
- made by the most active cards

Method:

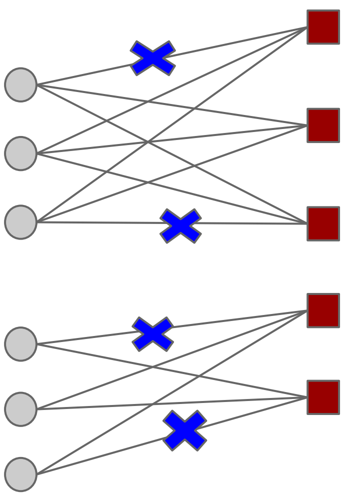
- Detect the most active cards through a data driven threshold (€ 150)
- Exclude the single transactions of amount lower than a second data driven threshold (€ 50)



17.8M transactions
5.1M cards



10.4M transactions
3.4M cards



9M transactions
3.4M cards

Bankitalia's Rules for Money Laundering Involving Groups of Cards

- ☑ Transactions of relevant amount, involving cash
- ☐ **Transactions carried out at the same operative point (or neighboring operative points)**
- ☐ Transactions carried out inside a narrow time frame (inside a temporal interval of 60 minutes, for instance)

Find Groups of Neighboring ATMs

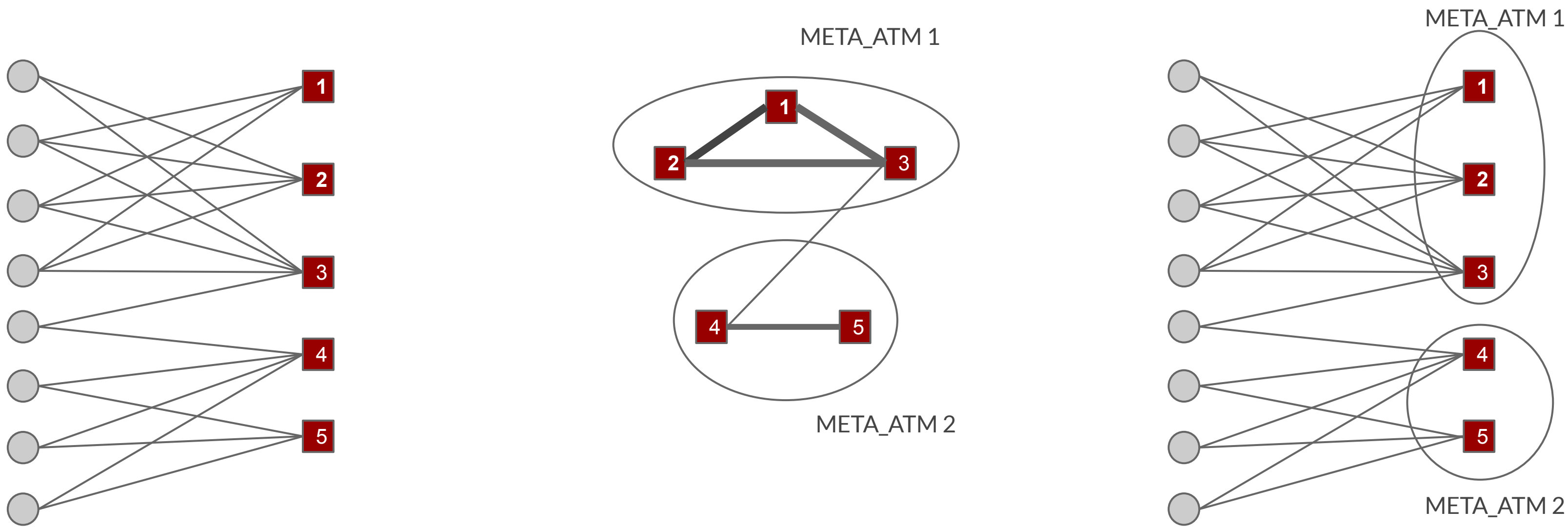
- **Goal**: Find geographically close groups of ATMs.
- **Problems/limitations**: No geospatial data on ATMs :(
- **Idea/Assumption**: the more cards two ATMs have in common, the closer they are ;)

How to Find Groups of Neighboring ATMs?

1. Model the dataset as a bipartite graph
   - **Nodes**: 5.183.308 cards and 135.503 ATMs.
   - **Edges**: 17.866.556 transactions.

2. Compress the graph on the ATMs nodes: 135.503 ATMs, 1.391.665 edges.

3. Apply a community detection algorithm to find communities of neighboring ATMs ;)
   #META_ATMs: 22.964

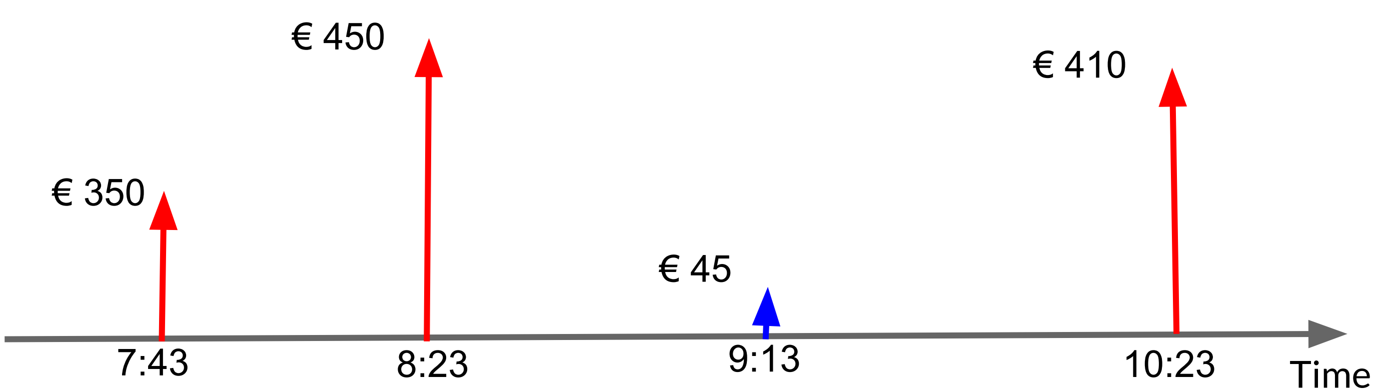


Bankitalia's Rules for Money Laundering Involving Groups of Cards

- ☑ Transactions of relevant amount, involving cash
- ☑ Transactions carried out at the same operative point (or neighboring operative points)
- ☐ **Transactions carried out inside a narrow time frame (inside a temporal interval of 60 minutes, for instance)**

Suspected Cards from a Temporal Point of View

- **Definition:** two cards are suspected from a temporal point of view when they carry out at least two temporally close transactions.
- **Temporally close (assumption):** Two transactions are temporally close when they lie inside a temporal interval of 60 minutes.
- **Degree of Suspicion of a Couple of Cards:** the greater the number of times two cards make temporally close transactions, the greater their degree of suspicion is.
- **Goal:** detect the pairs of cards with the highest degree of suspicion.



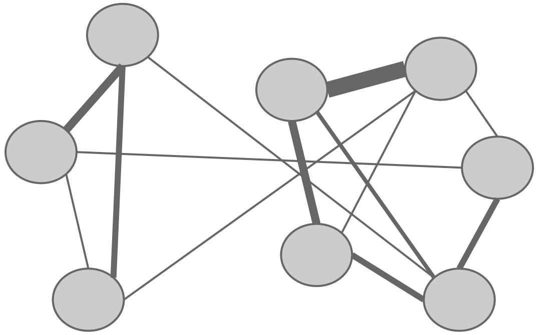
Temporal Locality Algorithm

- For each Meta_ATM

- Scan of all transactions in ascending order of time
  `[ID_card, ID_Meta_ATM, timestamp_↑]`

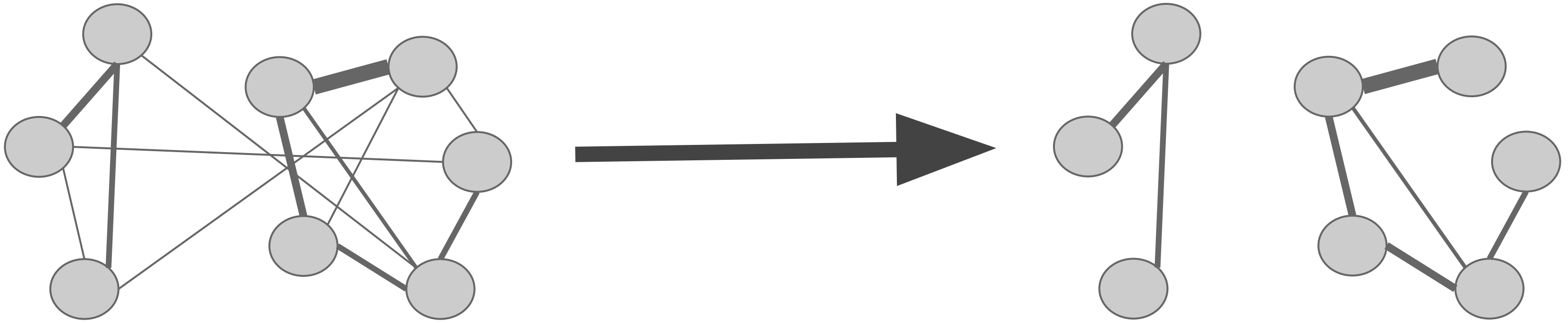
- Count the number of times a couple of cards make temporally close transactions
  ⇒ evaluating the degree of suspicion of a couple of cards

- **OUTPUT:** Mapping between pairs and their strength of interactions ⇒
  Weighted graph connecting cards with their degree of suspicion



Temporal Locality Algorithm: Relevant Data

1. We are interested in groups of connected cards.
2. A lot of not relevant edges inside the weighted edgelist (low degree of suspicion).
3. Removing not relevant edges (low degree of suspicion).
4. Find the connected components of the resulting graph.



Bankitalia's Rules for Money Laundering Involving Groups of Cards

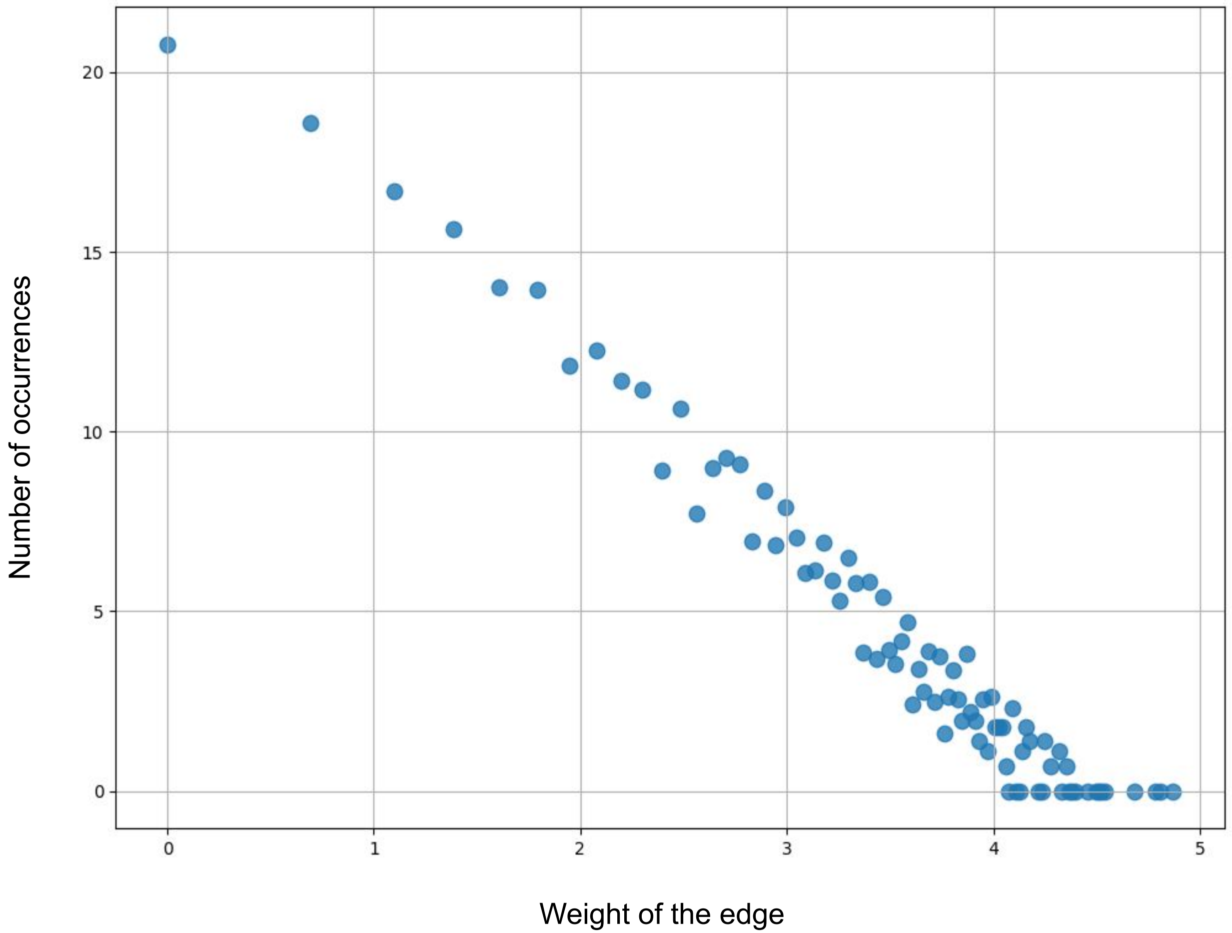
- Transactions of relevant amount, involving cash

- Transactions carried out at the same operative point (or neighboring operative points)

- Transactions carried out inside a narrow time frame (inside a temporal interval of 60 minutes, for instance)

Temporal Locality Algorithm: Results

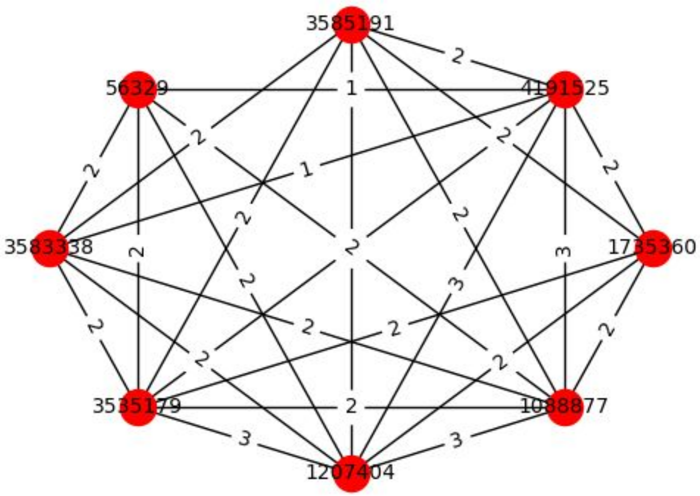
Number of occurrences of each particular weight in the set of edges



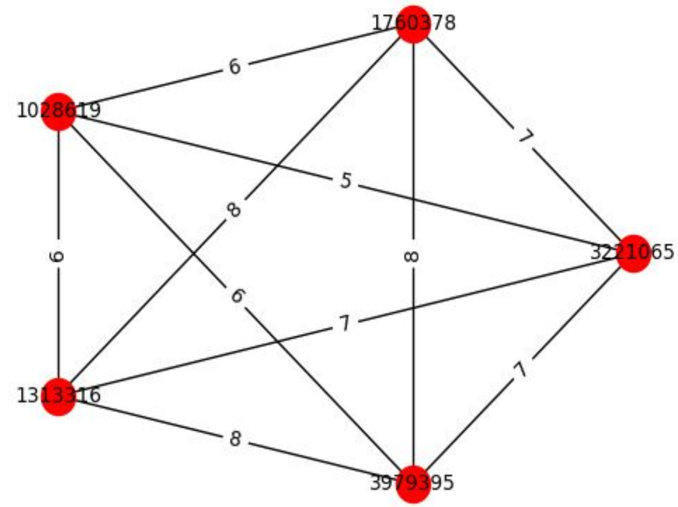
- Applied to META_ATM with the highest number of transactions (688k transactions, 312k cards)
- Number of edges: `1.193.539.139 (1.049.755.424` **`with weight=1`**`, 88%)`
- loglog scale
- trend similar to a power law

Temporal Locality Algorithm: Results

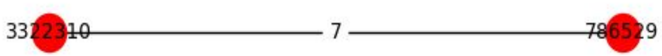
COMMUNITY 1 - INTERACTIONS



COMMUNITY 2 - INTERACTIONS

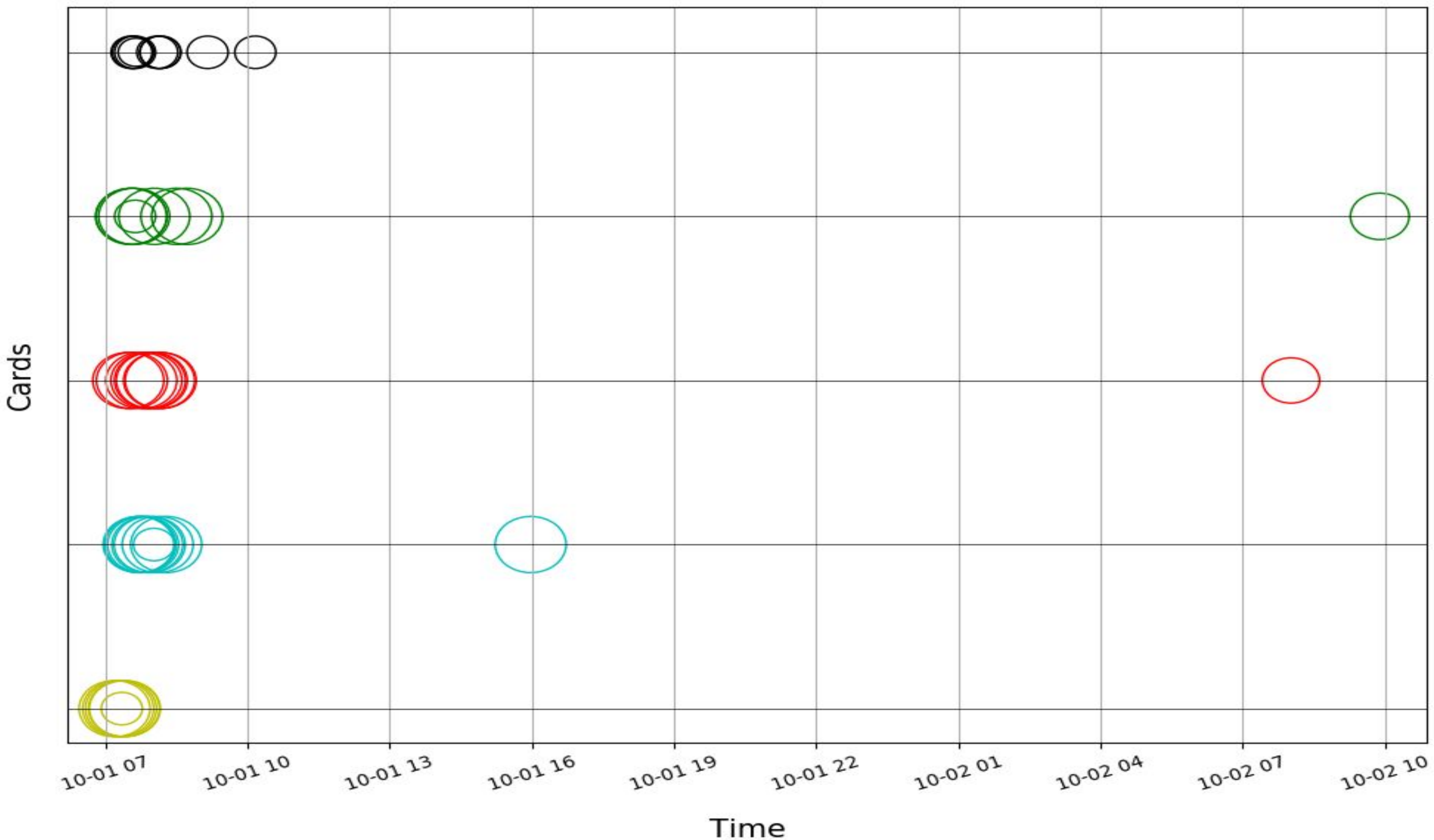


COMMUNITY 3 - INTERACTIONS



Temporal Locality Algorithm: Results

COMMUNITY 2 - TIME SERIES



Temporal Locality Algorithm: Main Memory Constraint

1. Billions of spotted pairs ⇒ cannot be fitted in main memory :(

2. Flush on disk sub-mappings every time main memory is full ;)

3. Sort the sub-mappings and merge them all

4. Obtain the overall mapping representing the strength of interaction of all the spotted pairs ⇒ Weighted graph connecting cards with their degree of suspicion

Temporal Locality: Probabilistic Approach

- **Motivation**: We are interested in pairs of cards that appear many times during the scanning of the dataset: high degree of suspicion

- **First Step** (find good candidates): Every time we spot a pair of cards, add it into a set with probability p to guarantee that a relevant pair will be in the set with a certain input confidence

- **Second Step** (compute the suspicious degree): Scan a second time the dataset, adding to a map only pairs inside the set

- **OUTPUT**: A much smaller mapping w.r.t. the one obtained before
  ⇒ possible errors: ~~FP~~, FN

DETERMINISTIC RUNNING TIME:

- `3h 25min 28secs`

PROBABILISTIC RUNNING TIME (conf=0.9 ⇒ p=0.2056):

- `53min 48secs`

RECALL:

- `0.93`

PRECISION:

- `1.0`

Conclusions

- PROBLEM: Find groups of cards acting in anomalous way, by following the Bankitalia instructions

- APPROACH:

  - Deterministic algorithm able to adapt to the available main memory (common laptop)

  - Probabilistic approach that does not generate False Positives and 3.8 times faster

- GOAL: To help the money laundering experts by providing a smaller set of groups of cards classified as suspicious

Thank You

For the Attention

Deterministic VS Probabilistic - Performance

DETERMINISTIC

- **TOTAL TIME:** `3h 25min 28secs`

PROBABILISTIC (conf=0.9 ⇒ p=0.2056)

- **TOTAL TIME:** `53min 48secs`
- **True Positives (TP):**
  `152.935 (on 164.344)`
- **False Positives (FP):**
  `273.619.520` (not a problem ;))
- **True negatives (TN):**
  `1.193.363.386`
- **False negatives (FN):**
  `11.409`
- **Recall** = `TP/(TP + FN) = 0.93`
- **Precision** = `TP/(TP + FP) = 1.0`