SHIELD: Evolutionary Synthesis of Privacy-Preserving Pipelines for Live Stream Data Sharing

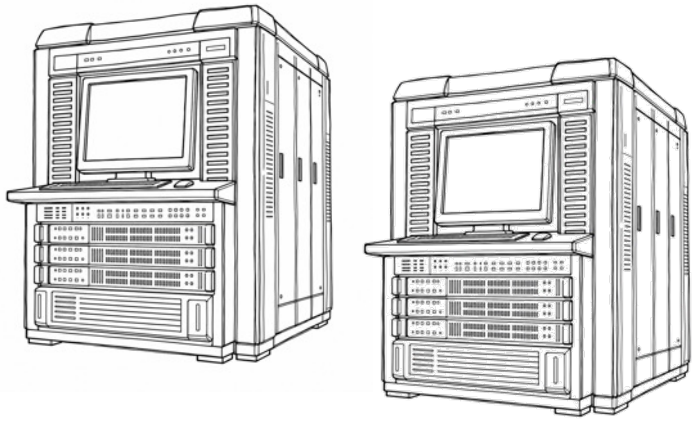
Silvia Perelli
Department of Civil Engineering and Computer Science Engineering (DICII)
University of Rome Tor Vergata
Roma, Italy
silvia.perelli@alumni.uniroma2.eu

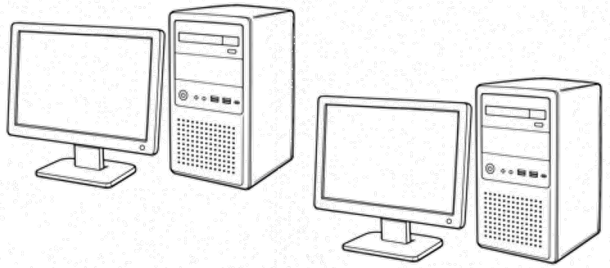
Eric Medvet
Department of Engineering and Architecture,
University of Trieste
Trieste, Italy
emedvet@units.it

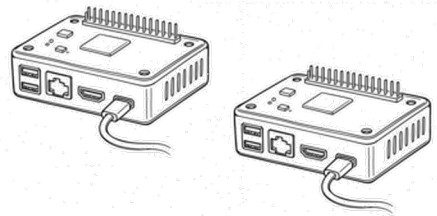
Vincenzo Gulisano
Department of Computer Science and Engineering, Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden
vincenzo.gulisano@chalmers.se

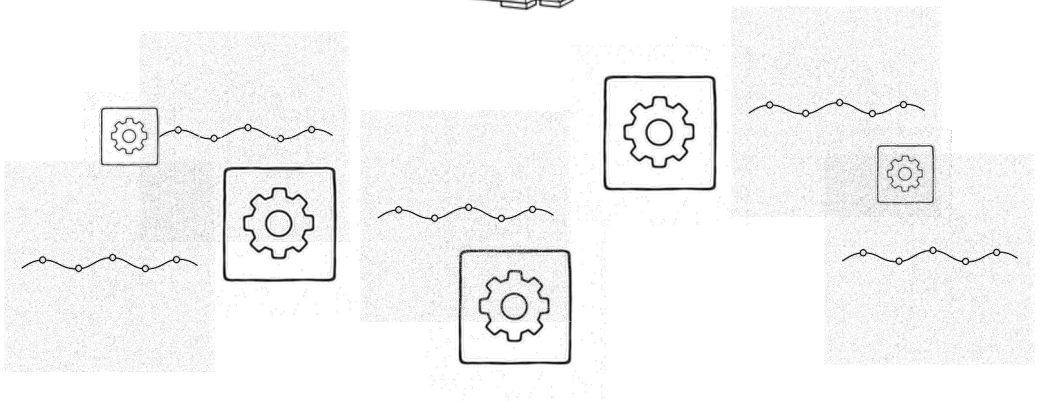
Agenda

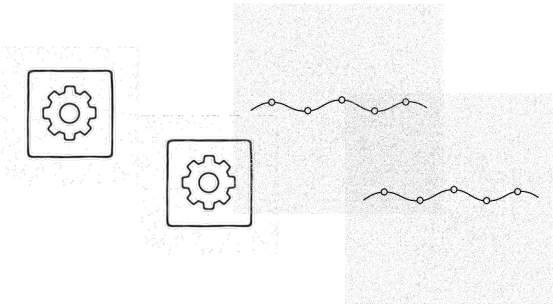
- Motivation & problem (& related work)
- Intuition
- Method
- Use-cases & results
- Wrapping up & future work

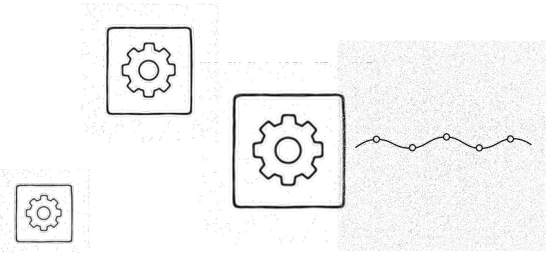












The one with the data

The one with the streaming application

The performance engineer (or researcher)

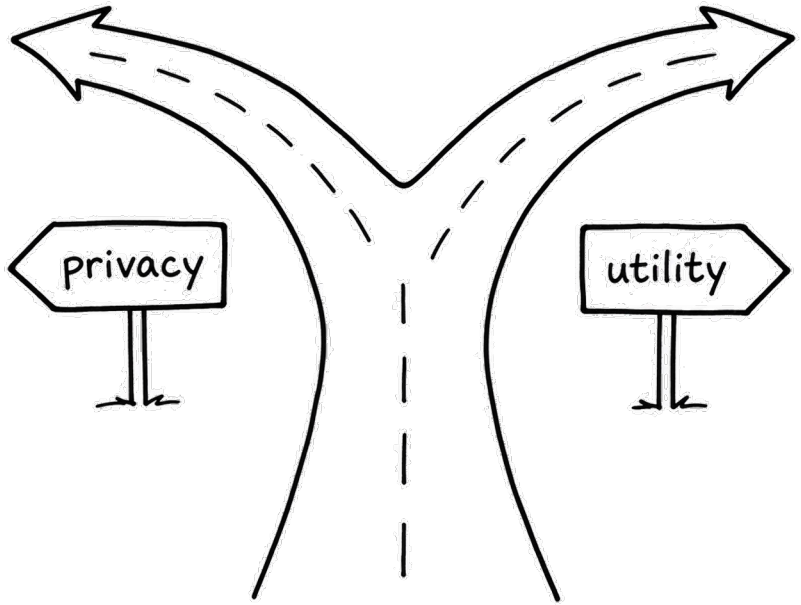
The one with the data

The one with the
streaming application

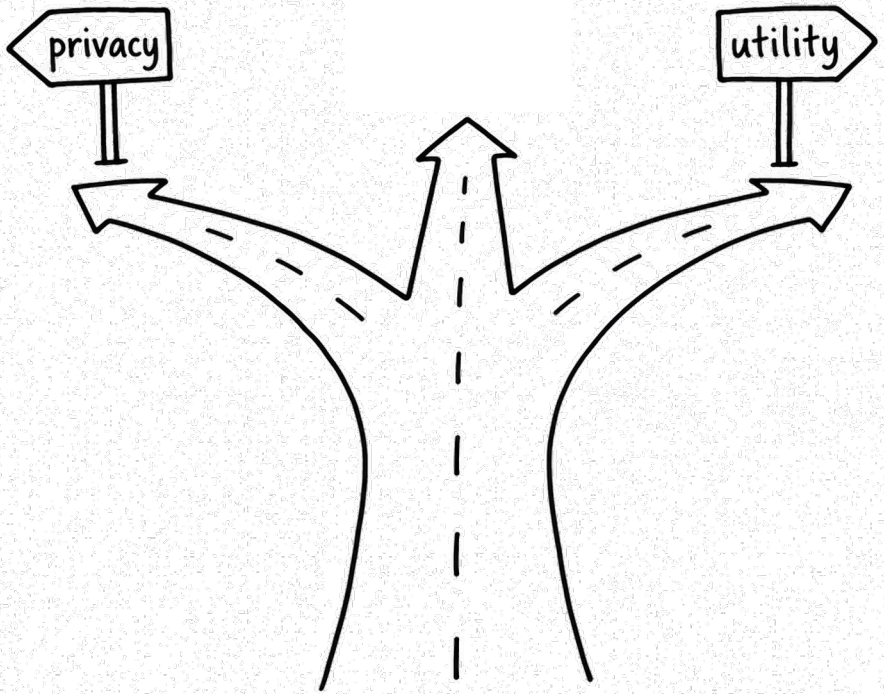
The performance engineer
(or researcher)

Anonymize the data
(e.g., using k-anonymity)!

Yes, but...



Try to find (automatically)
some data transformation
pipeline that satisfies both

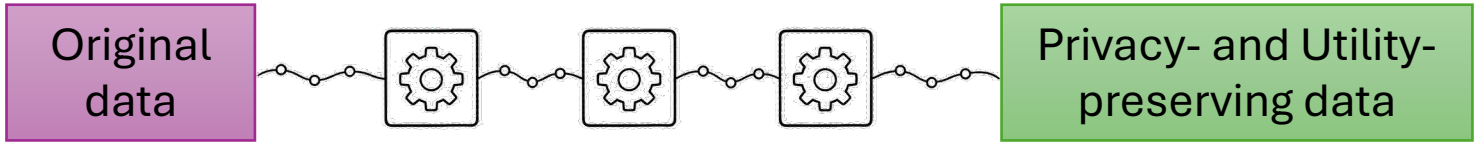


Agenda

- Motivation & problem (& related work)
- Intuition
- Method
- Use-cases & results
- Wrapping up & future work

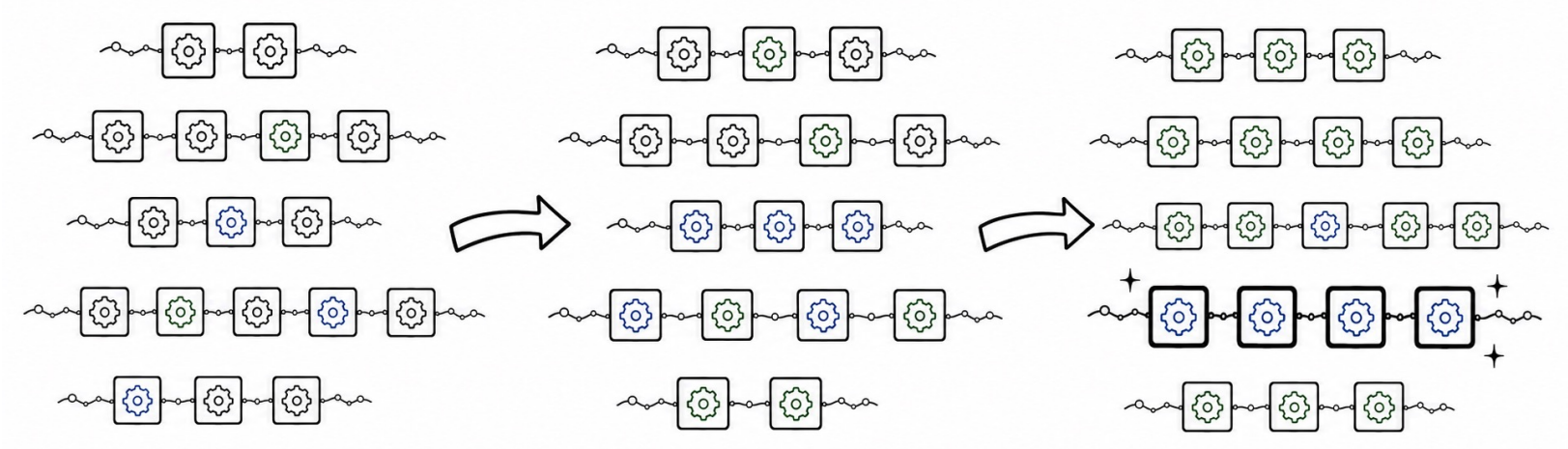
What if…

- We define a set of operators that can modify data
  With predefined/known semantics
- We compose them into a stream processing query
- If it works, then:



- And since these are stream processing operators, they can perform this on-the-fly, on live data!
- Nice, **but how do we find such a composition?**

Use evolutionary computing to find the pipeline!



- Define each candidate's fitness as f(privacy, utility)
- Let users customize the multi-objective problem user-defined metrics for:
  - Privacy (for the modified vs. original data)
  - Utility
    - Semantics (for the outputs produced by the modified vs. original data)
    - Fidelity (for the application characteristics observed for modified vs. the original data)

Agenda

- Motivation & problem (& related work)
- Intuition
- Method
- Use-cases & results
- Wrapping up & future work

Goal: define a  that is good for $\begin{cases} \text{Privacy} \\ \text{Utility} \end{cases}$

- Recipe:
  1. Define what a  is
  2. Define how it can be created/mutated
  3. Evolve!

Define what a  is

- **Filter**, based on
  - random field among the ones defined by the input data
  - random condition (> or <)
  - random value
- **Noise**, based on
  - random field among the ones defined by the input data
  - random value for σ (Gaussian centered at 0)
- **Duplicate**, based on
  - Random probability
- **Aggregate**, based on
  - random field among the ones defined by the input data
  - random aggregation function (min, max, or avg)
  - random window length

Define how  can be evolved/mutated

⟨ops⟩ ::= ⟨op⟩ | ⟨op⟩-> ⟨ops⟩
⟨op⟩ ::= ⟨filter⟩ | ⟨dupl⟩ | ⟨noise⟩ | ⟨aggr⟩
⟨filter⟩ ::= filter(⟨attr⟩⟨cond⟩⟨val⟩)
⟨dupl⟩ ::= duplicate(⟨prob⟩)
⟨noise⟩ ::= noise(⟨attr⟩,⟨sigma⟩)
⟨aggr⟩ ::= aggregate(⟨attr⟩,⟨count⟩,⟨type⟩)
⟨cond⟩ ::= < | >
⟨type⟩ ::= avg | min | max
problem dependent
⟨attr⟩ ::= a_1 | ... | a_n
⟨val⟩ ::= ⟨d⟩.⟨d⟩⟨d⟩E⟨sign⟩⟨d⟩
⟨d⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
⟨sign⟩ ::= + | -
⟨prob⟩ ::= 0.⟨d⟩
⟨sigma⟩ ::= 0.01 | 0.05 | 0.10 | 0.25
⟨count⟩ ::= 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10

- The initial population is created randomly using **ramped half-and-half**: 50% **full** (reach max depth) and 50% **grow** (ok to stop earlier)
- Then, grammar-based subtree mutation selects a subtree in an individual, and regenerates it using the grammar (for syntactically validity)

Evolve!

- A random population is initialized (size is fixed at X individuals).
- At each evolution step, X new offspring from current population.
- Parents are selected with a bias toward better-ranked individuals.
  - Metrics used for ranking are presented next
- Each offspring is created by grammar-based subtree mutation.
- Parents and offspring are merged into a pool of 2X candidates and ranked by Pareto dominance.
- The best X candidates are kept as the next population.

Agenda

- Motivation & problem (& related work)
- Intuition
- Method
- Use-cases & results
- Wrapping up & future work

Performance metrics

- **Privacy**
  - Based on **density** and **size similarity** compared to the original data
  - For each point in the modified dataset:
    - Compute normalized distance from other points in the input dataset
    - Find closest k points (k=50) and compute its dispersion (based on standard deviation)
  - Aggregate over all points and modulate accounting for size dissimilarity
- **Fidelity**
  - Based on the difference in #tuples and #keys in all intermediate streams in the query
- **Semantics**
  - Based on the difference in output tuples
- All are in [0,1] and to be **maximized**

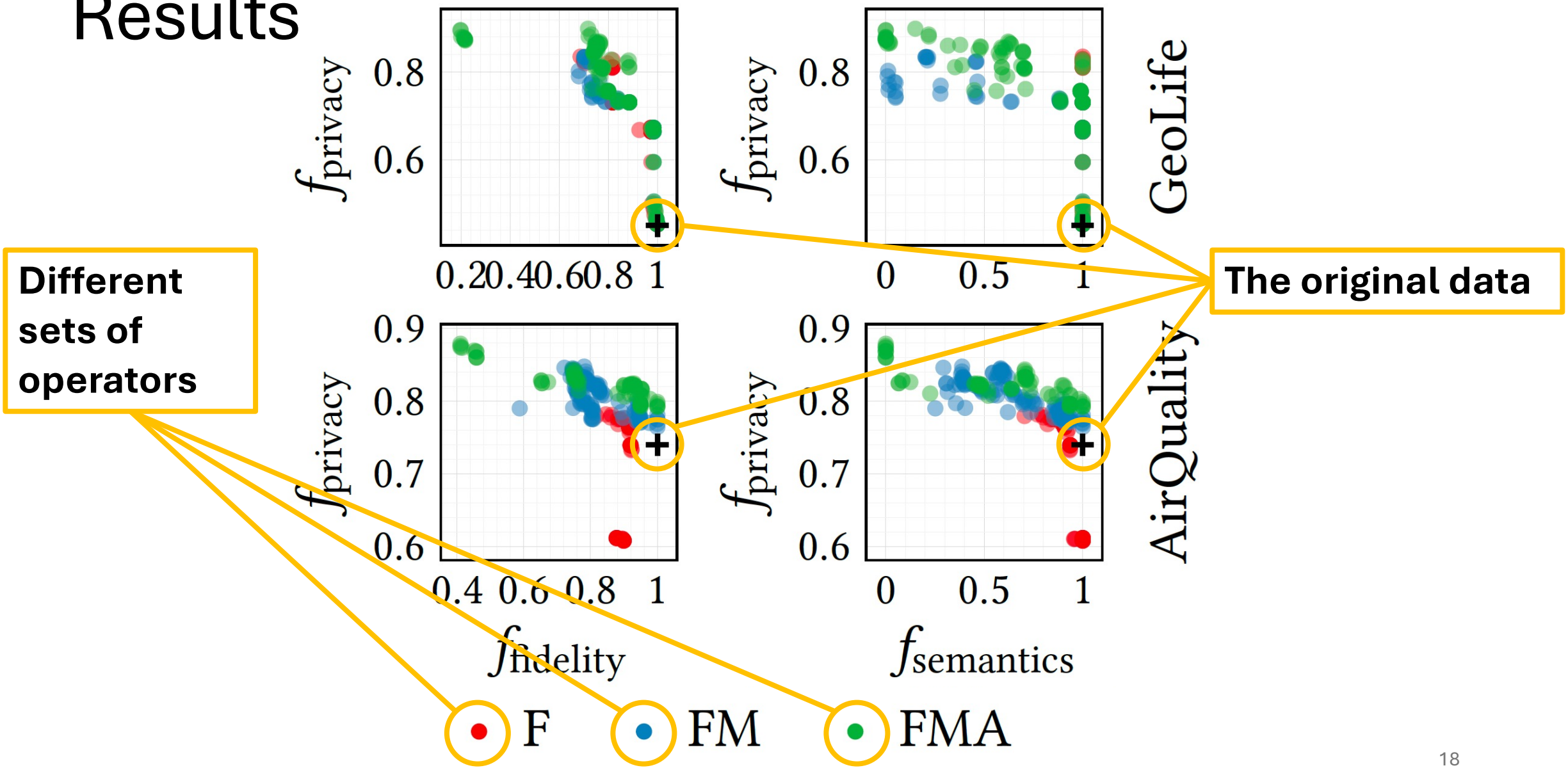
Use-cases

- **GeoLife**
  - Mobility monitoring (180 users, mainly in Beijing, several years of data)
  - Data contains aggregate per-user hourly statistics (coordinates X,Y)
  - Query: report moving average of users close to a given location
- **AirQuality**
  - Sensors measuring air quality, reporting hourly statistics over a year
  - Query: After filtering/averaging, report values that indicate low air quality

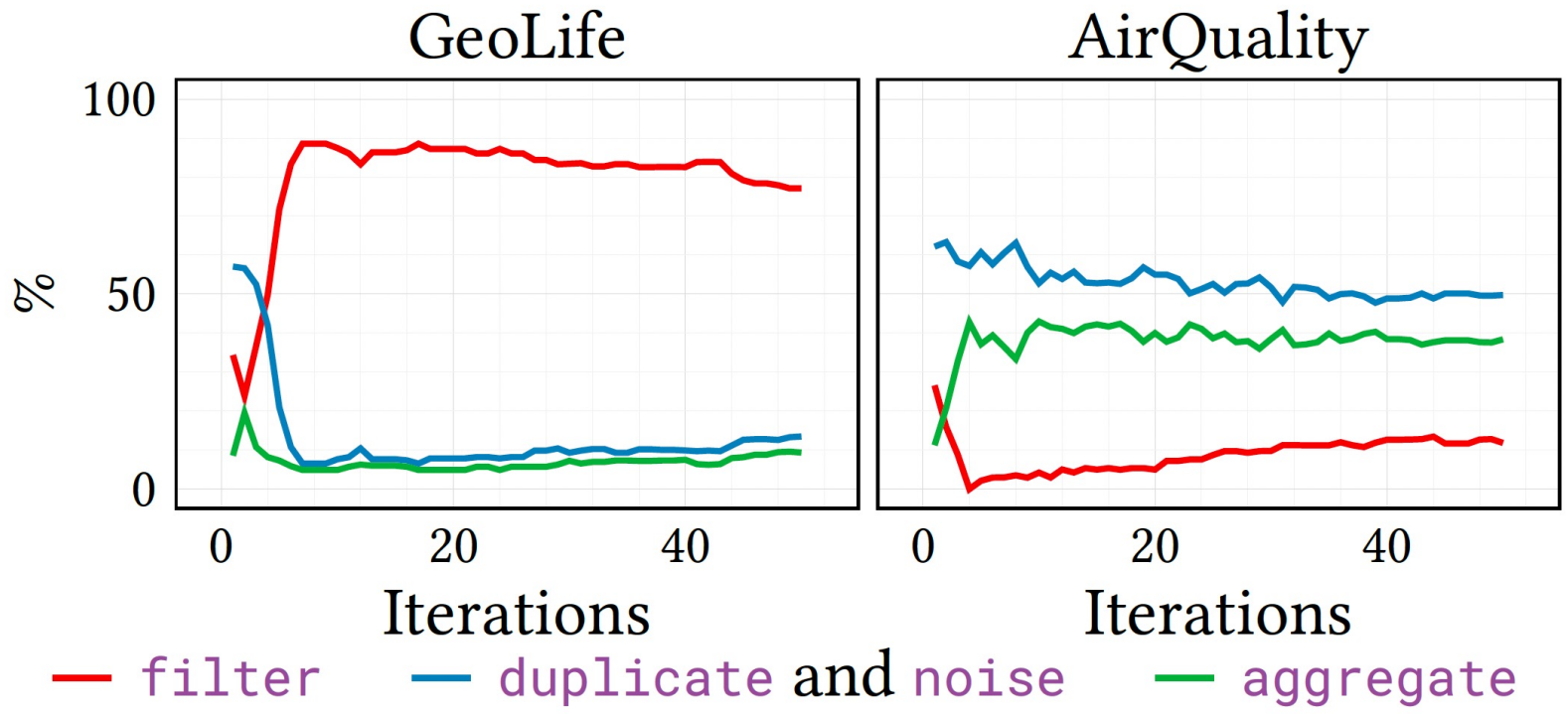
Evaluation setup

- Stream Processing Engine: Liebre
- Evolutionary optimizer: JGEA
- Each experiment repeated 5 times
- Setup for the evolutionary process
  - 50 evolutionary iterations
  - For populations of size 100
- For composing the anonymization pipelines, we consider 3 cases
  - Only Filters
  - Filters + Noise/Duplicate
  - Filters + Noise/Duplicate + Aggregate

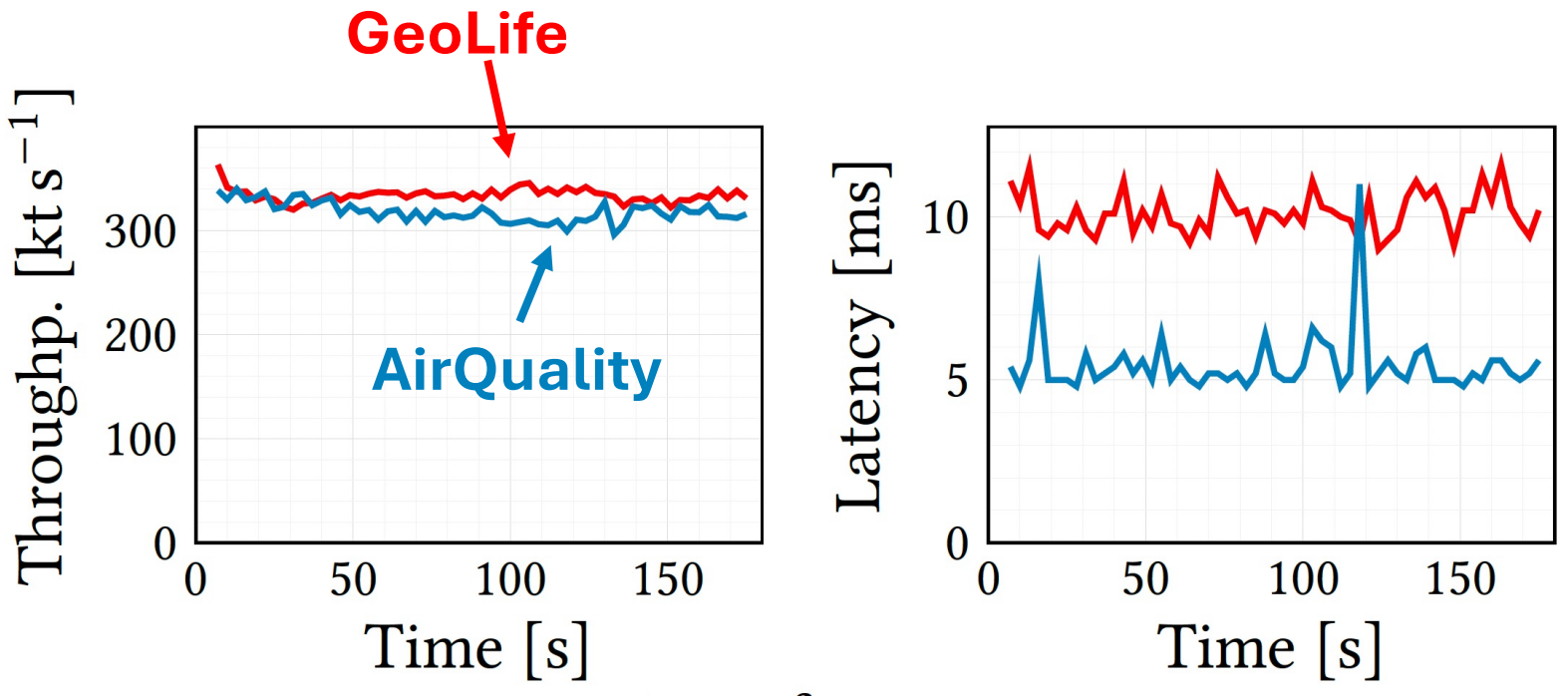
Results


Different sets of operators
The original data
GeoLife
AirQuality
$f_{privacy}$
$f_{fidelity}$
$f_{semantics}$
0.8
0.6
0.9
0.7
0.2 0.4 0.6 0.8 1
0.4 0.6 0.8 1
0 0.5 1
F
FM
FMA

Results


GeoLife
AirQuality
100
50
0
%
0
20
40
Iterations
filter
duplicate and noise
aggregate

Results


GeoLife
AirQuality
Throughp. [kt s^{-1}]
300
200
100
0
0
50
100
150
Time [s]
Latency [ms]
10
5
0
0
50
100
150
Time [s]

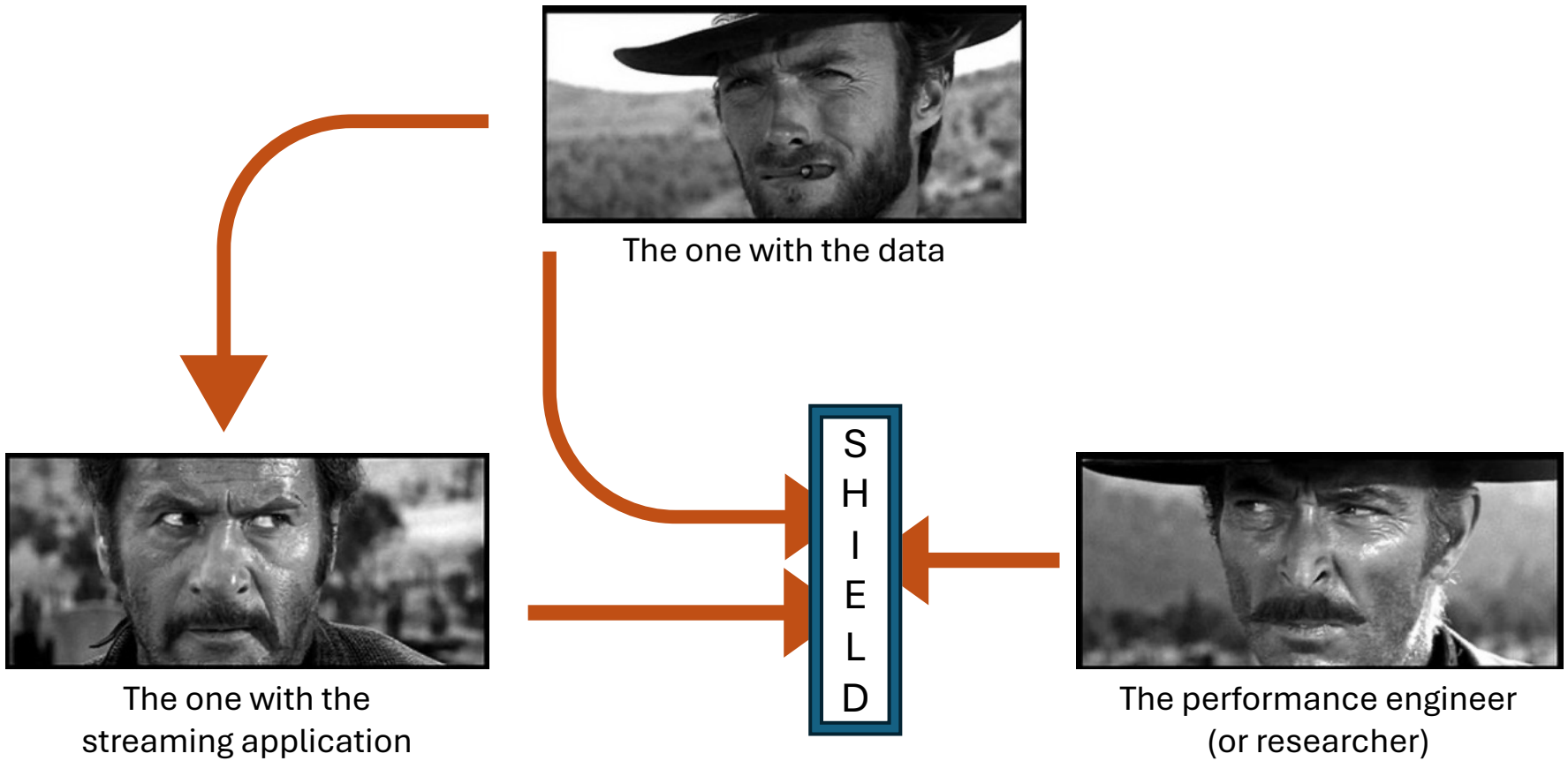
Agenda

- Motivation & problem (& related work)
- Intuition
- Method
- Use-cases & results
- Wrapping up & future work

The one with the data

The one with the
streaming application

The performance engineer
(or researcher)


The one with the data
S
H
I
E
L
D
The one with the
streaming application
The performance engineer
(or researcher)

Future work

- Improving efficiency/expressiveness of the search
  - Simplify individuals
  - Provide a larger set of “building blocks” to compose anonymization queries
- Move from single-source use cases to multi-source use cases
- Make the anonymization pipelines both
    - Data-aware (it already is)
    - Application-aware (it is not in this study)