Aggregated datasets for fast LCA tools

Presentation for Brightcon2020

October 20, 2020

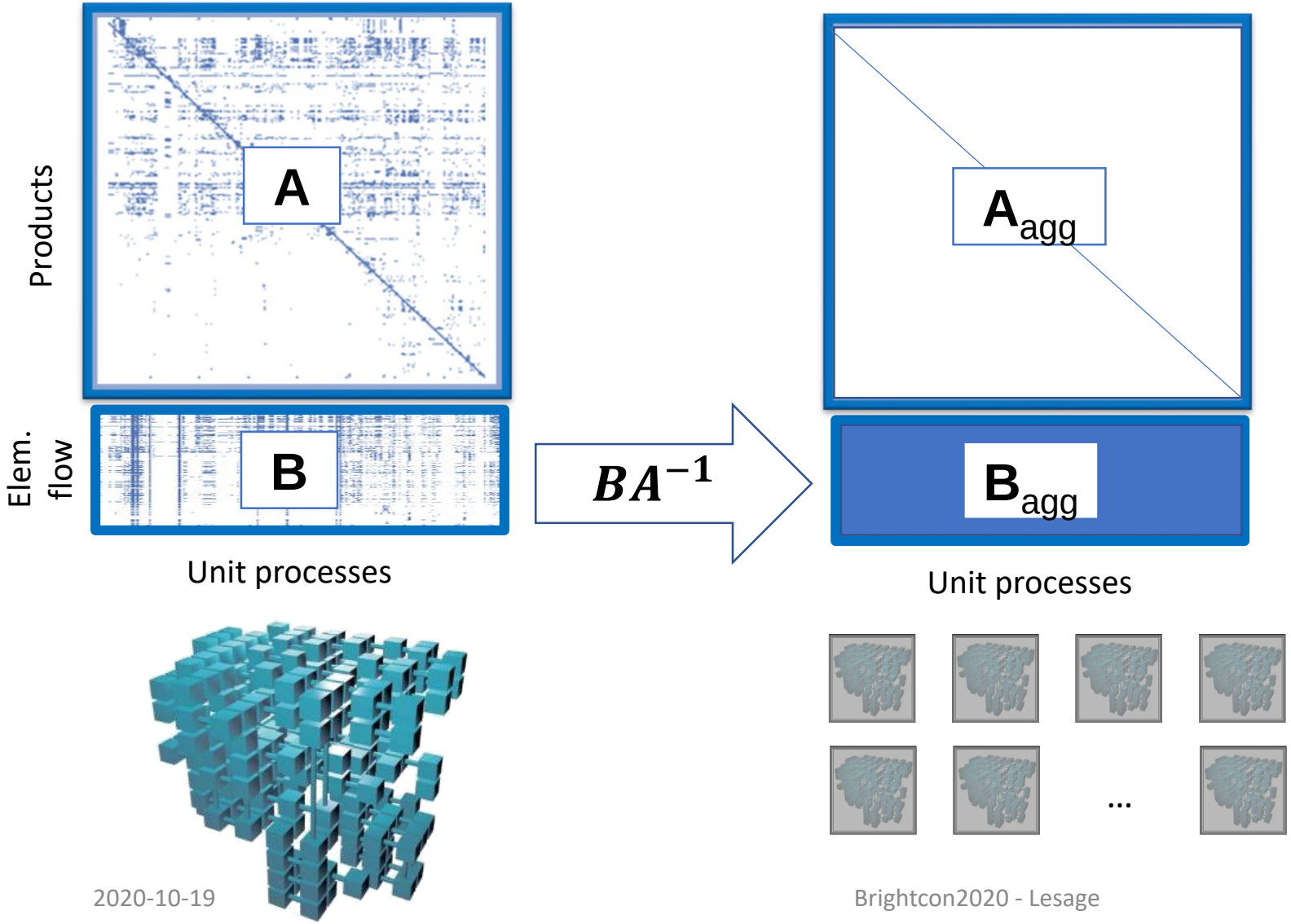
Pascal Lesage

pascal.lesage@gmail.com

Fast!

- Aim: make LCA *fast* to reduce deterrent
- Strategy, from perspective of someone used with commercial software:
  - Use aggregated datasets!
    - SimaPro: Gains ≈ 10x (from 19.5 to 2 seconds)

What are aggregated datasets

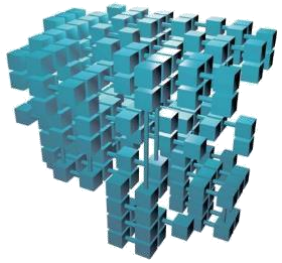


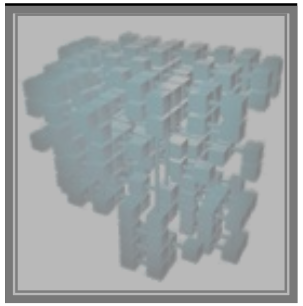
Aggregated AKA system terminated, System, S, LCI, cradle-to-gate

No inputs from the technosphere
B matrix contains cradle-to-gate LCI

Some critiques of use of aggregated datasets

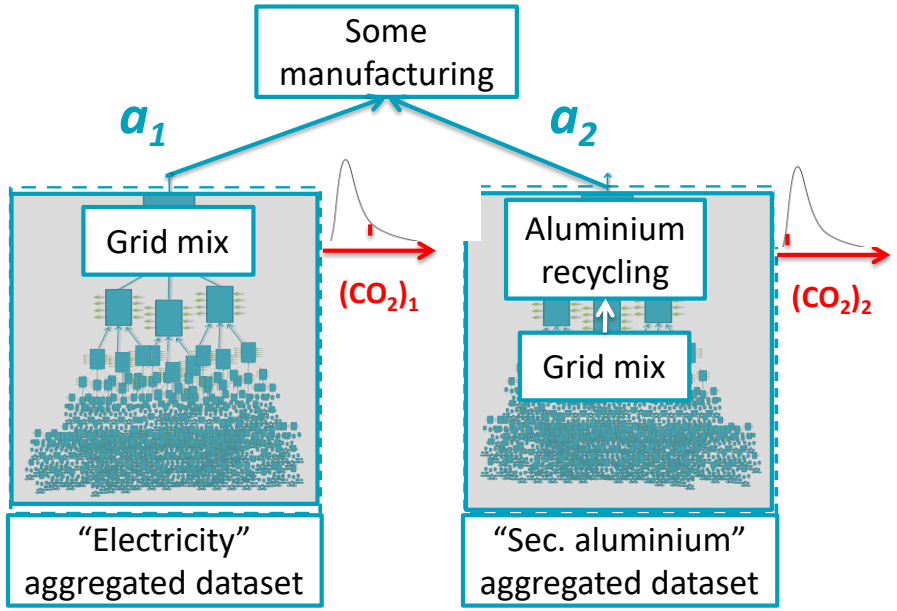
- Loss of transparency





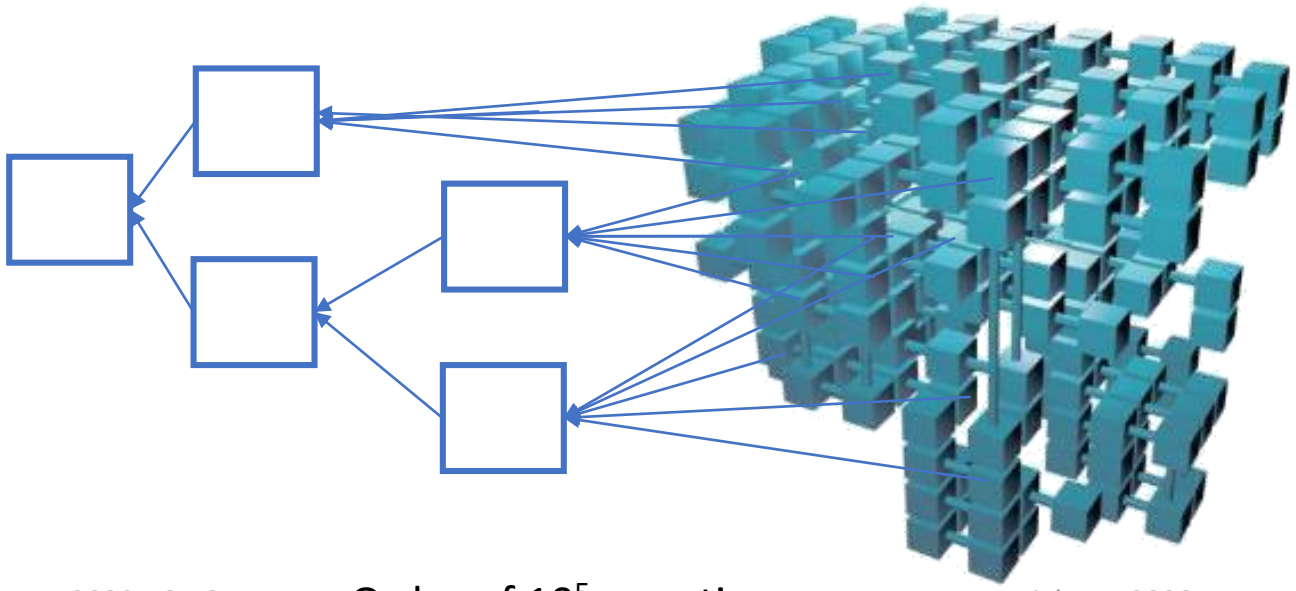
- Dependent sampling *within* a product system not possible

  (Don't believe the hype, it *can* matter)

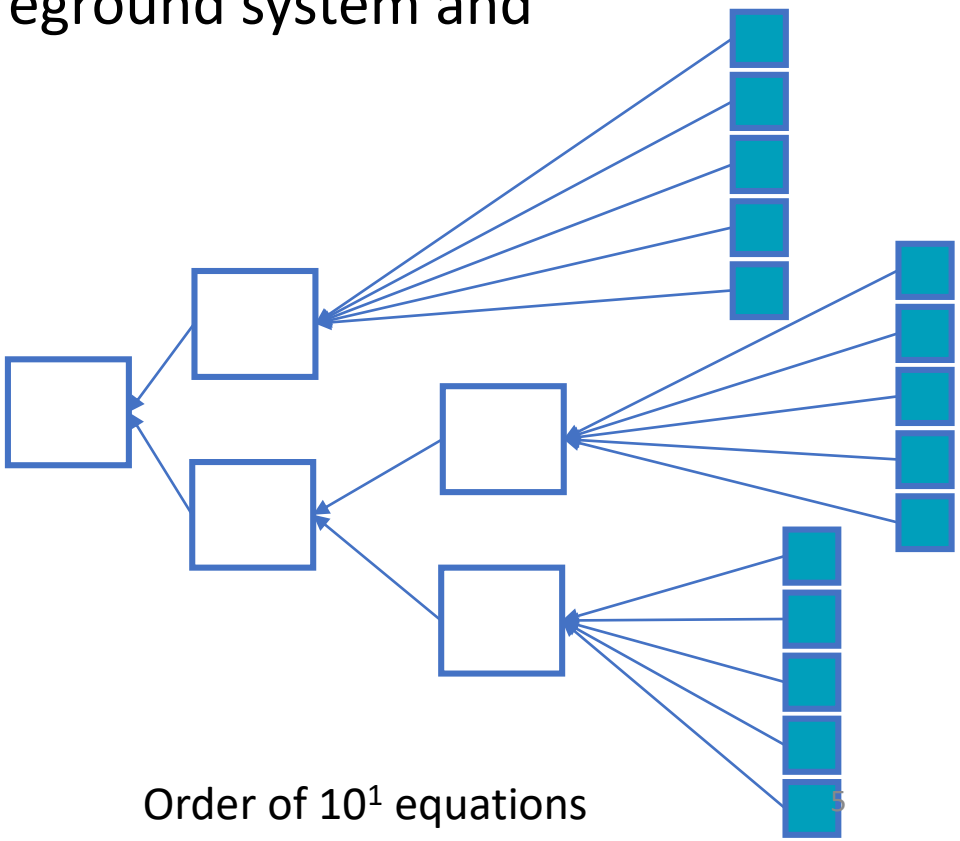


Why are aggregated datasets fast?

- Assumption:
  - Heavy work (solving system of linear equations for large background LCI database) is done *ahead* of time
  - What is left is solving a much more limited foreground system and some matrix multiplications

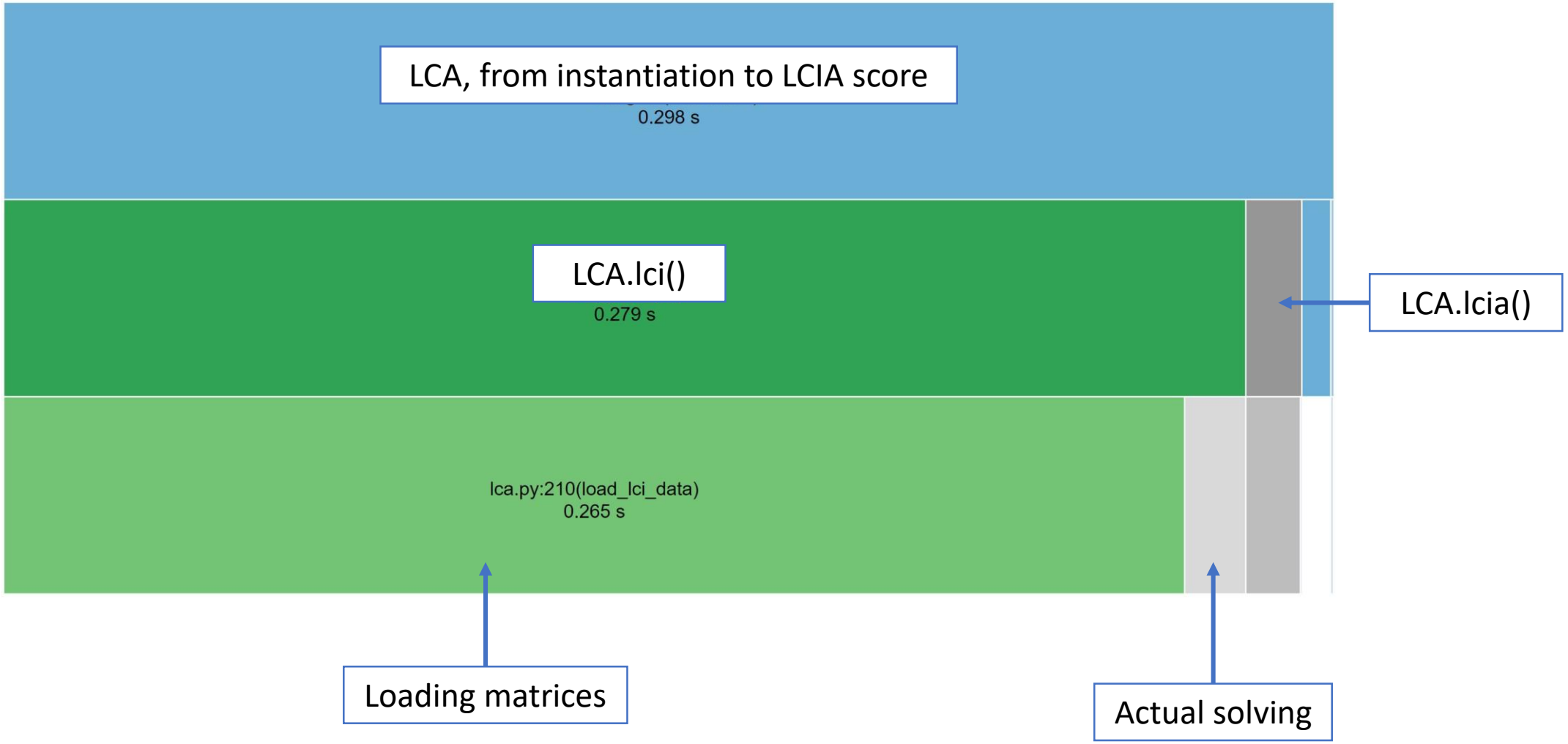


Order of 10^5 equations

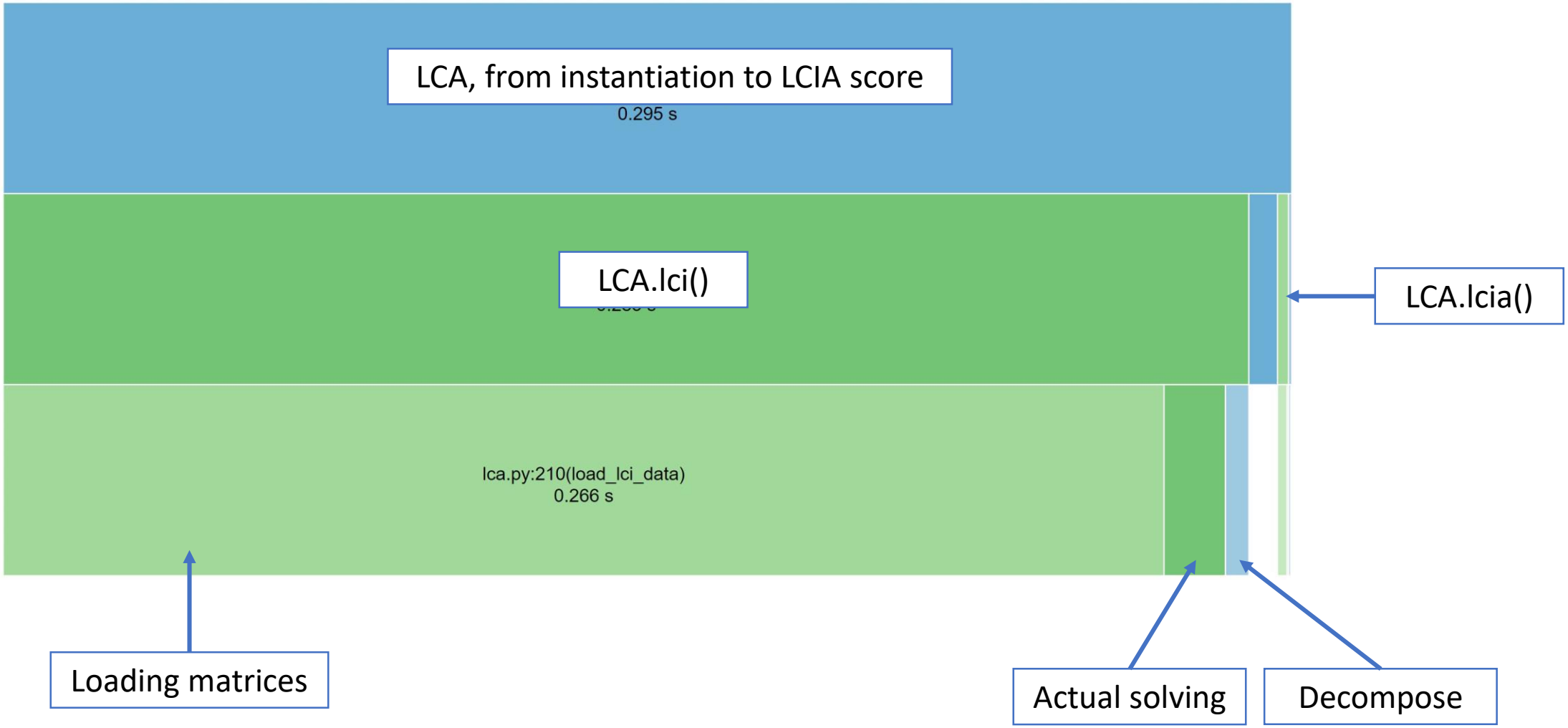


Order of 10^1 equations

Profiling an bw LCA calculation (w/o factorization)



Profiling an bw LCA calculation (w/ factorization)



Why are aggregated datasets fast?

- Maybe there is more than just “precalculating” → minimizing what gets loaded may also help

Objective of presentation

- Main objective:
  - Present some packages that help integrate aggregated datasets in brightway2:

    - bw2agg
      - Aggregate whole database
      - Save activities with LCIA scores only
    - bw2preagg
      - Generate dependently-sampled LCI arrays for whole databases for reuse in LCA
    - presamples
      - Integrate LCI arrays in LCA calculations
    - bw2tree
      - Minimize size of system and terminate in aggregated datasets

- Sidetracked objective:
  - Can we save even more time by having many smaller databases?

bw2agg

- `conda install --channel pascallesage bw2agg`
- `pip install bw2agg`
- https://brightway2-aggregated.readthedocs.io/en/latest/quickstart.html

- Two main tasks:
  - Convert database of unit process datasets in database of aggregated datasets
  - Facilitate working with “unit impact”

bw2agg – aggregated LCI databases

- Convert to LCI:

```
agg_db = bw2agg.DatabaseAggregator(
    up_db_name="ecoinvent 3.6 cutoff - Unit process",
    agg_db_name="ecoinvent 3.6 cutoff - Aggregated (LCI)",
    database_type="LCI",
    overwrite=False
    )
agg_db.generate()
```

```
Writing activities to SQLite3 database:
0% [##############################] 100% | ETA: 00:00:00
Total time elapsed: 01:38:45
```

```
Title: Writing activities to SQLite3 database:
  Started: 10/19/2020 21:08:26
  Finished: 10/19/2020 22:47:12
  Total time elapsed: 01:38:45
  CPU %: 232.40
  Memory %: 2.26
```

You should probably NOT do this:
the resulting biosphere matrix is dense,
and brightway2 is not equipped to deal with dense matrices

bw2agg – Augmenting with unit impacts

- `bw2agg.scores.add_unit_score_exchange_and_cf(method_id)`

- Adds:
  - a biosphere activity flow
  - a corresponding cf=1 to LCIA results

- Key: `(‘biosphere3’, Method(method_id).get_abbreviation())`
  - E.g.

('IPCC 2013',
'climate change',
'GWP 100a') → `('biosphere3’, 'ipcc-2013cg.bd5af3f67229a1cc291b8ecb7f316fcf’)`

bw2agg – aggregated score databases

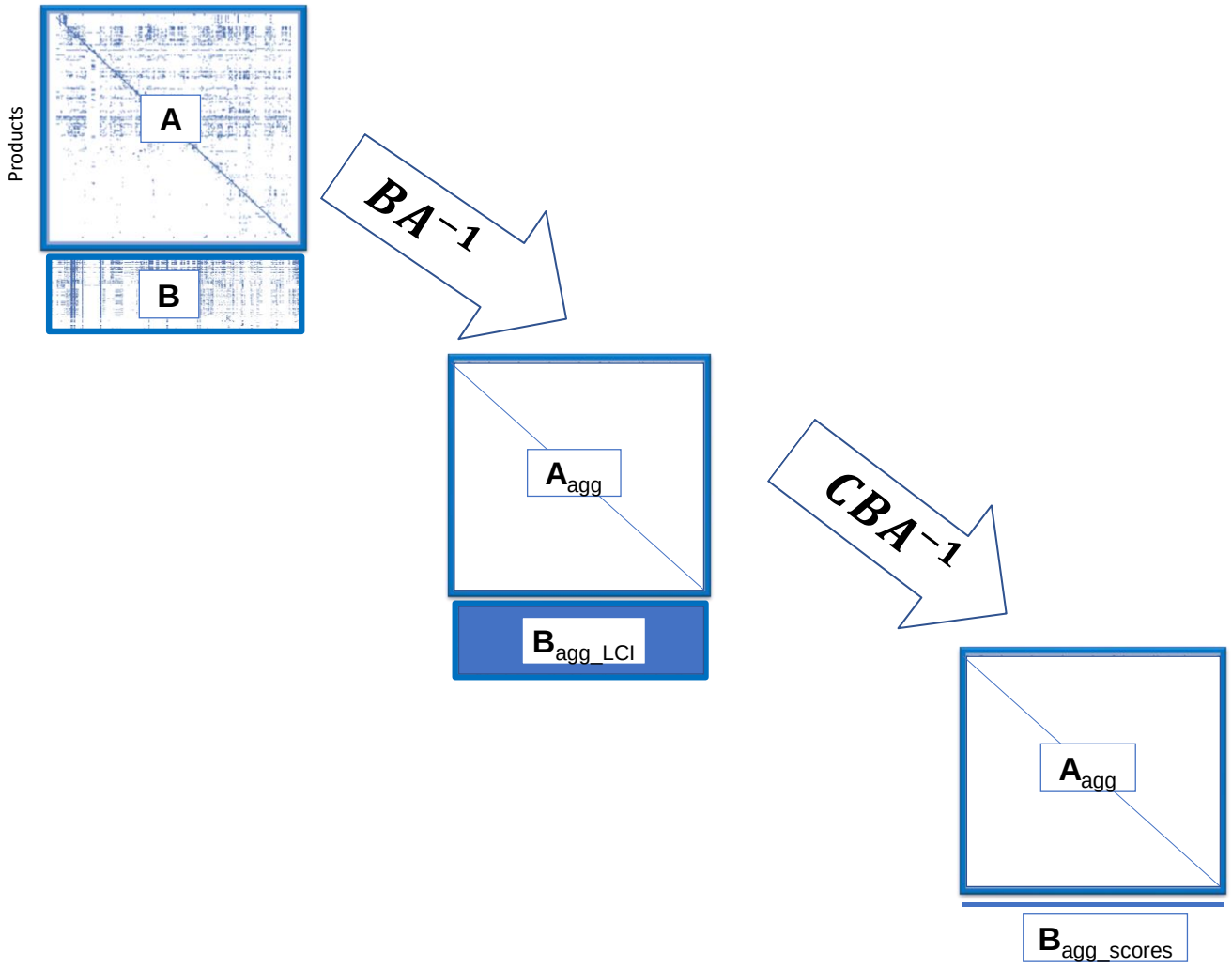
- Convert to LCIA scores:

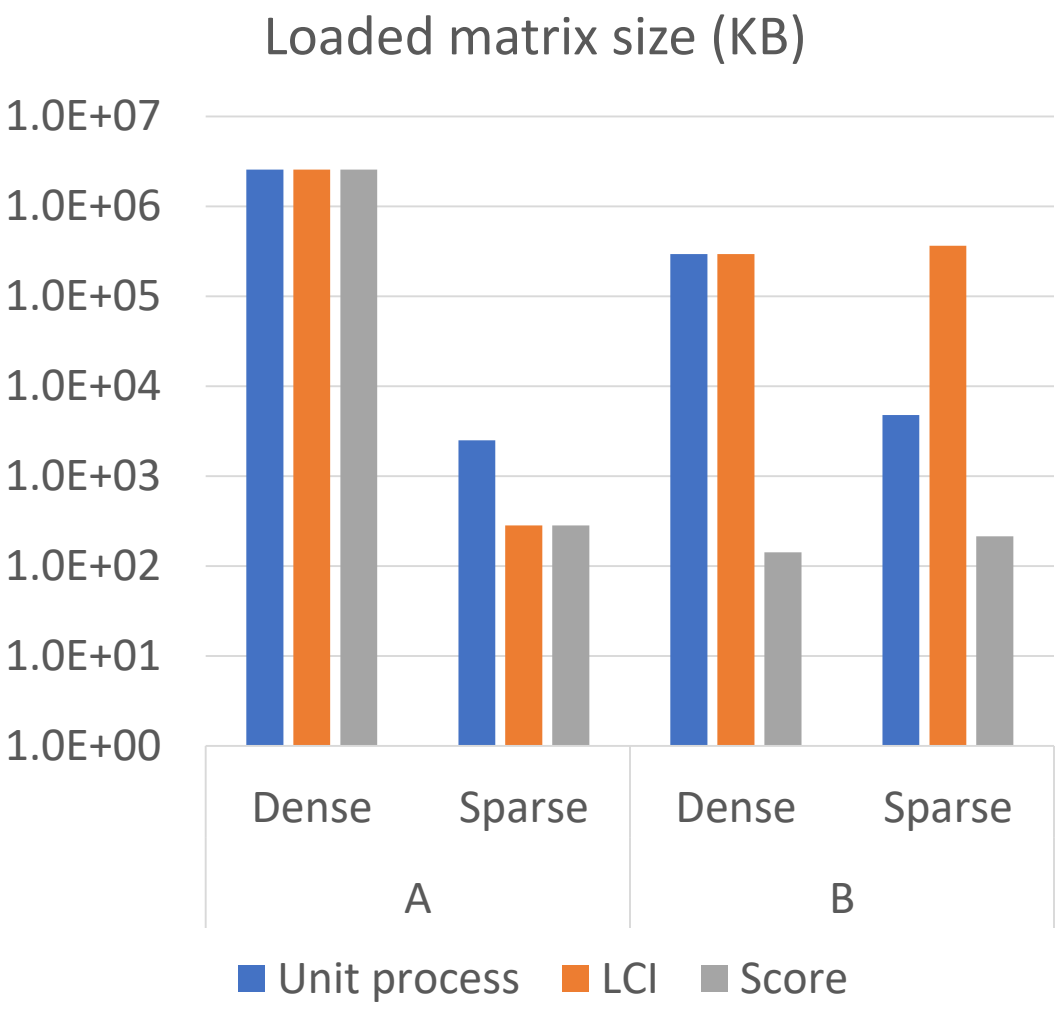
```
agg_db = bw2agg.DatabaseAggregator(
    up_db_name="ecoinvent 3.6 cutoff - Unit process",
    agg_db_name="ecoinvent 3.6 cutoff - Aggregated (scores)",
    database_type="LCIA",
    method_list=[ipcc],
    overwrite=False
    )
agg_db.generate()
```

```
Writing activities to SQLite3 database:
0% [##############################] 100% | ETA: 00:00:00
Total time elapsed: 00:06:13
```

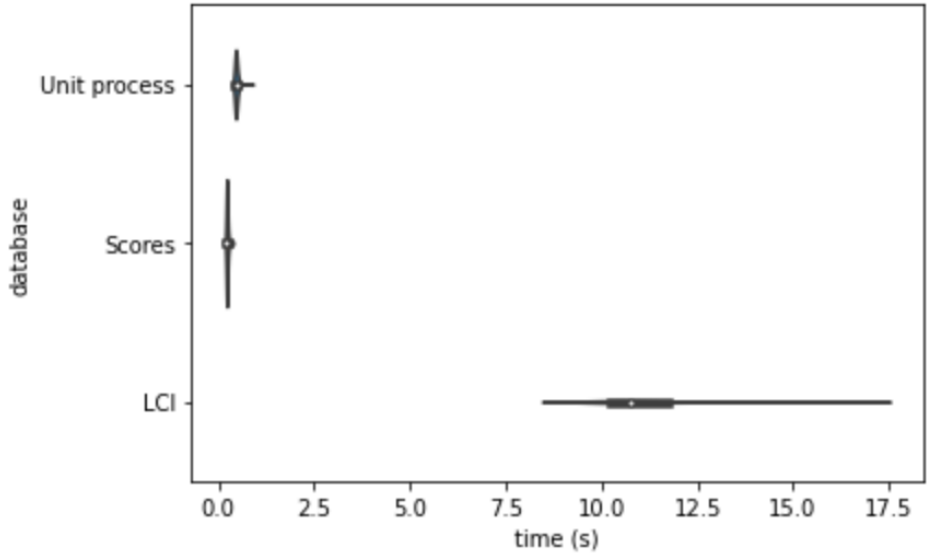
```
Title: Writing activities to SQLite3 database:
  Started: 10/19/2020 10:31:24
  Finished: 10/19/2020 10:37:38
  Total time elapsed: 00:06:13
  CPU %: 387.70
  Memory %: 2.61
```

bw2agg and size of matrices



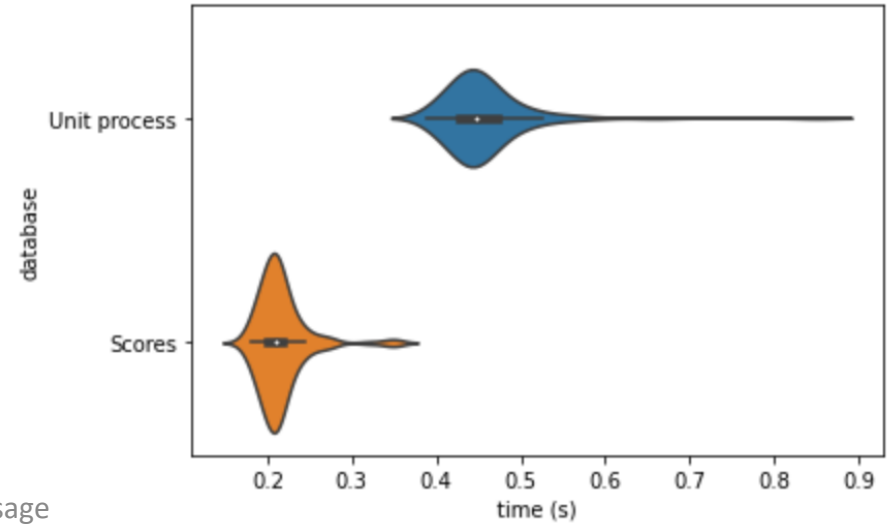


bw2agg and calculation time

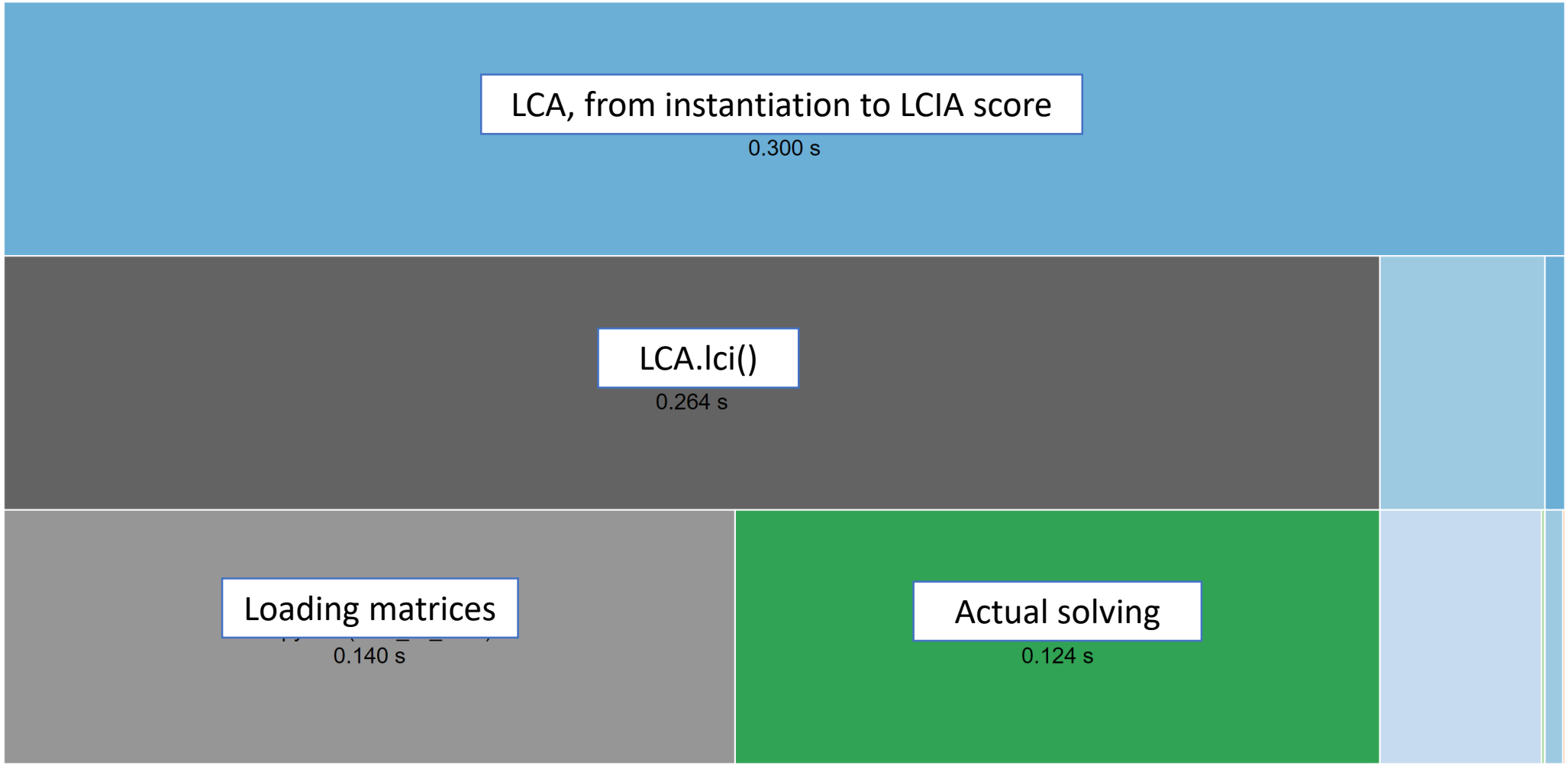


Conclusion 1: Don't use datasets that are aggregated at the LCI level in brightway2!

Conclusion 2: We can cut time by about half with precalculated scores

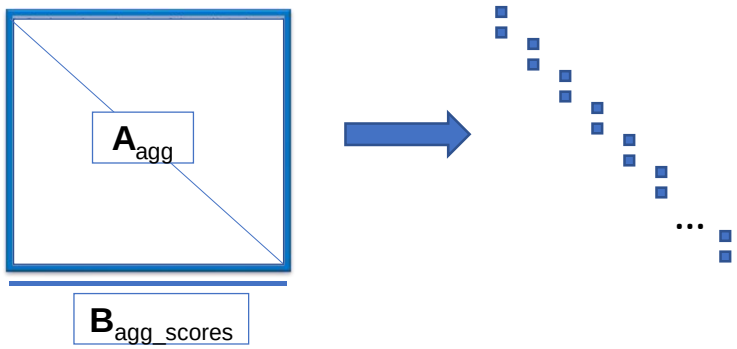


What takes time with scores?



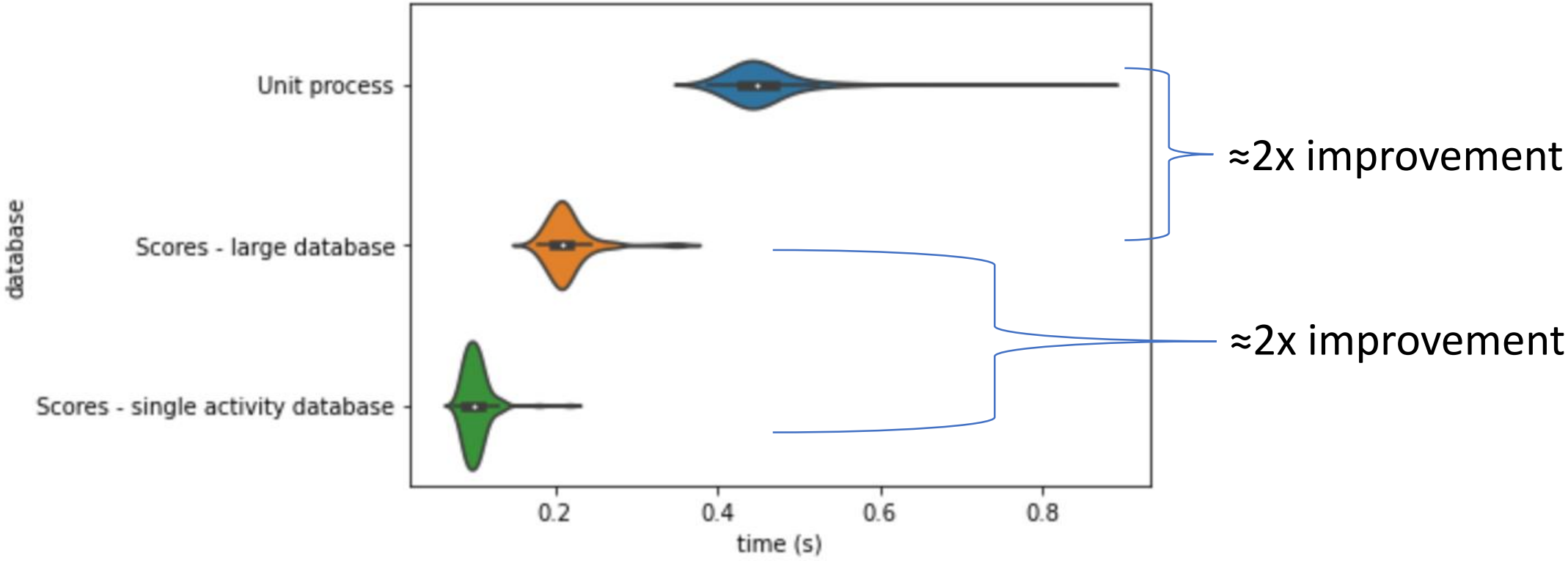
Testing multiple small matrix approach

- Can we cut time down more by saving aggregated activities in separate databases?
  - 1 database with 18k activities →18k databases with 1 activity each
- This would allow us to cut down on:
  - Loading time (smaller matrices)
  - Calculation time (smaller system)



Speed of LCA with different databases

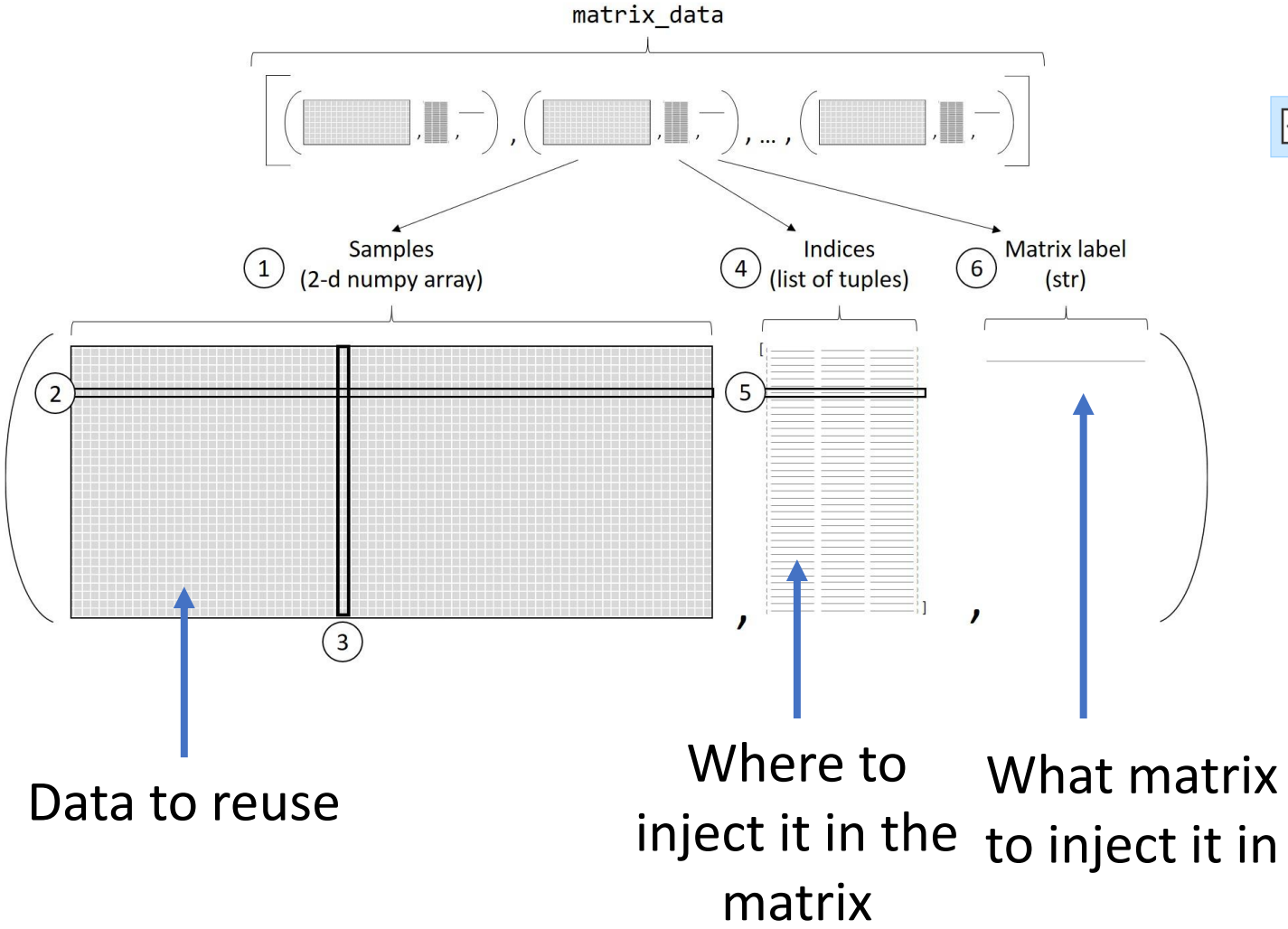
25 LCA with 5 random activities in demand
Time calculated 5 times per LCA



Samples for aggregated datasets

- Uncertainty information lost when aggregating
- Solution: generate samples that can be reused
- Both the generation and the use of these samples is based on *presamples*

Creating presamples packages (LCA matrices)



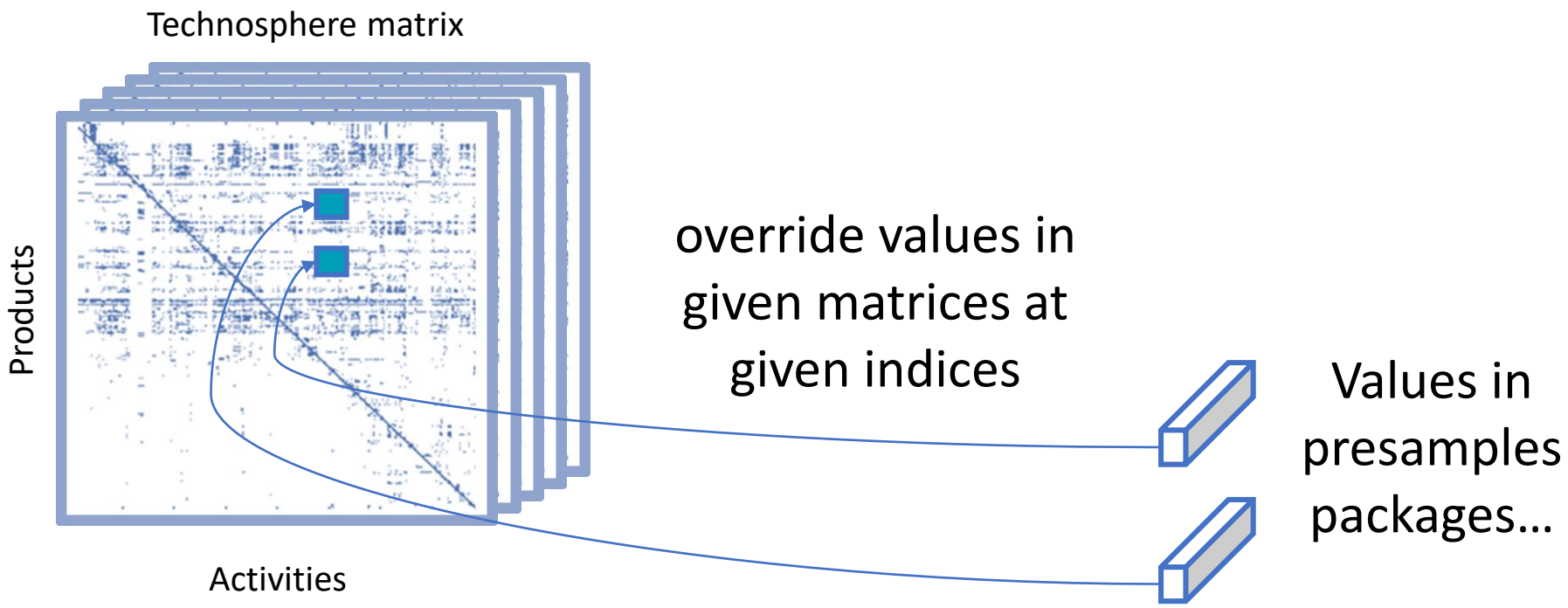
datapackage.json

21be9b5b205588118423bc7604918706.0.samples.npy

21be9b5b205588118423bc7604918706.0.indices.npy

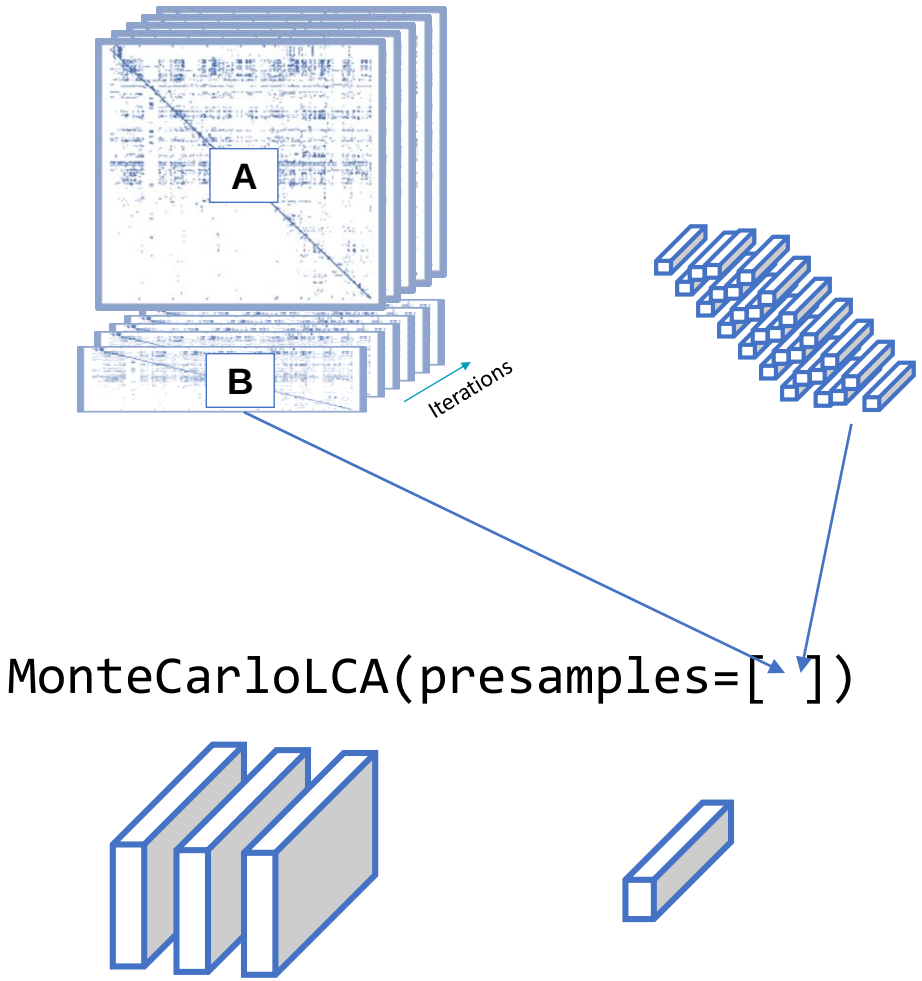
How does bw use presamples packages?

- `LCA(demand, method, presamples=[dirpaths to presamples])`
- `MonteCarloLCA(demand, method, presamples=[dirpaths to presamples])`



For database-wide presamples – `bw2preagg`

- `pip install bw2preagg`
- 6 steps:
  1. Setup
  2. Create base presamples packages
  3. Create "balancing" presamples packages, for land transformation and water[1]
  4. Generate LCI arrays
  5. Transform to LCIA arrays
  6. Concatenate



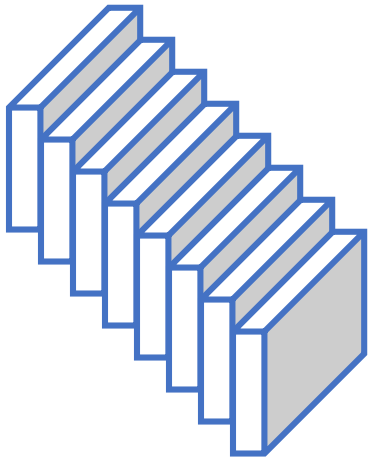
Takes weeks to run!

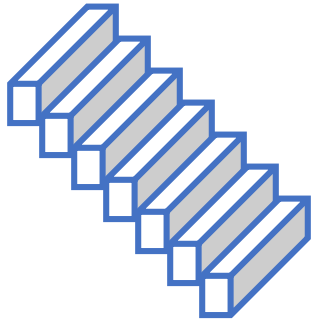
Code optimized for multiprocessing and distributing on clusters

[1] https://github.com/CIRAIG/bw2landbalancer, https://github.com/CIRAIG/bw2waterbalancer

What we get from `bw2preagg`

- LCI arrays:
  - Dependently sampled
  - As many arrays as there are activities in database
  - Rows = elementary flows
  - Columns = iterations
  - Name = code.npy
  - Need biosphere_dict for reuse
  - 1.36 TB !
- LCIA arrays:
  - Characterized LCI arrays
  - Can be created on the fly





How to use these LCI arrays

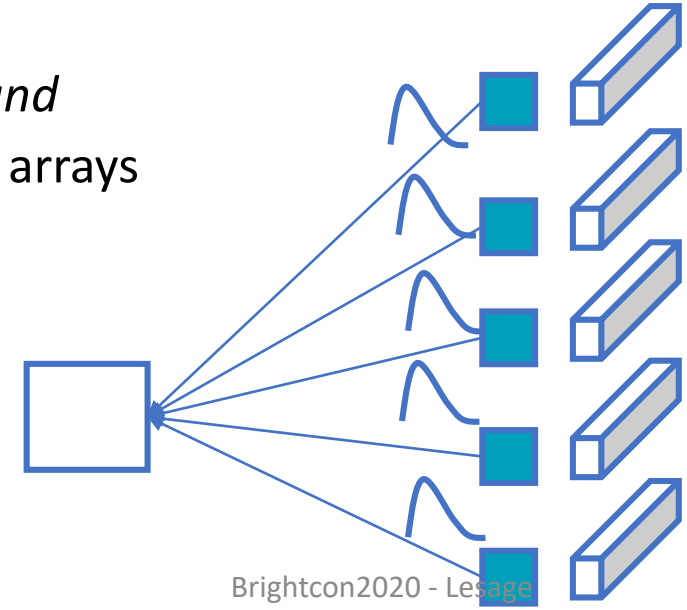
Non-exhaustive list of ways I've used these arrays in projects

1. Streamlined LCA tool (linear combination)
2. Link to tree background
3. ParameterizedBrightwayModel ←Sacrificed from presentation for all that profiling

1) Use in streamlined LCA tools

- Defined here as “tool that creates linear combination of datasets”
  - Typically one level deep
  - Acyclic
- Super fast because can be vectorized
- Simulated:
  - 5 upstream datasets
  - 10000 iterations for *demand*
  - Load 10000 iteration LCIA arrays
  - Scale (multiply)
  - Sum

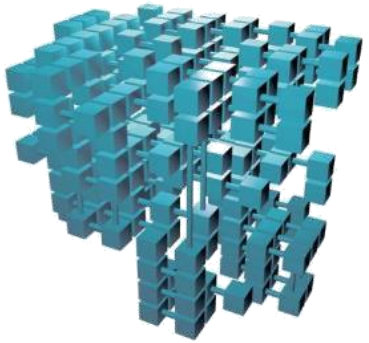
Streamlined LCA with LCI arrays	3 ms
MonteCarloLCA	38:47
Ratio	$7.8*10^5$

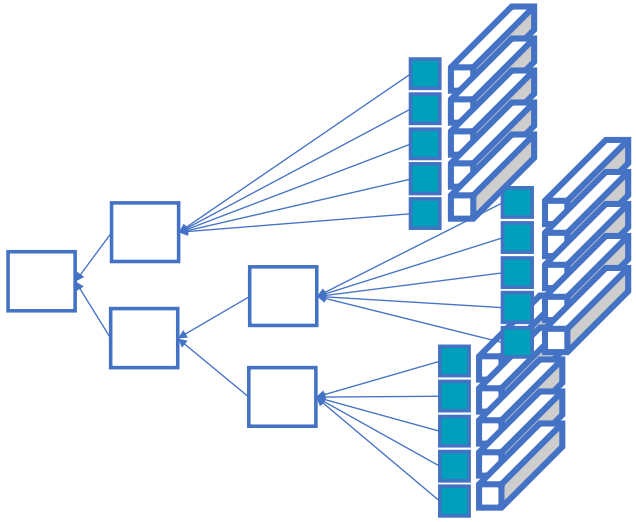


2) Link to tree background

Steps:

1. Convert network LCA to “pruned” acyclic tree
   - Nodes with impacts < cutoff criteria replaced by aggregated datasets
   - Uses `bw2tree`
2. Create presamples package for all “leaves” using LCI arrays
3. `MonteCarloLCA = (demand=tree.root, presamples = [presamples from step 2])`





Tree background	1:29
MonteCarloLCA	35:28
Ratio	24

Conclusion

- Aggregated datasets will make your models faster
  - Factor 2-4 for deterministic LCA
  - Orders of magnitude for MonteCarloLCA
    →**<u>IF</u>** you (or *someone*) invests the time to pre-emptively create required arrays
- Packages available to incorporate in real tools