Transforming Time Series into Graphs and Back with HyGraph

HyGraph Project – DFG/ANR funded project with collaboration between Leipzig University and Lyon University

Mouna Ammar, Shubhangi Agarwal, Angela Bonifati, Erhard Rahm

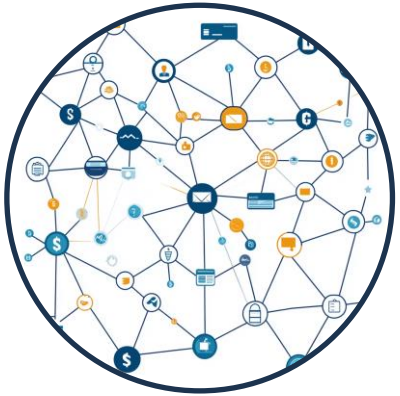
TGD Workshop
26/03/2025

Motivation - The Missing Bridge between Graphs and Time series

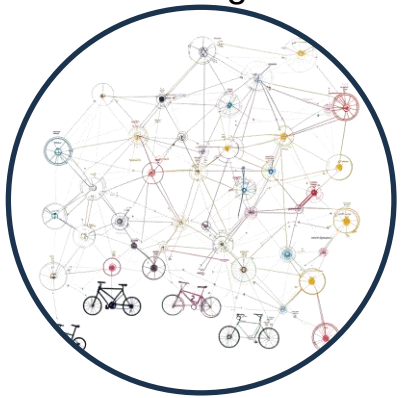
- Real world dataset existing in both graph and time series forms but separately
- The need to integrate relationships, temporal changes, and evolving patterns in one single system.

Social Network

Financial Network



Bike Sharing Network



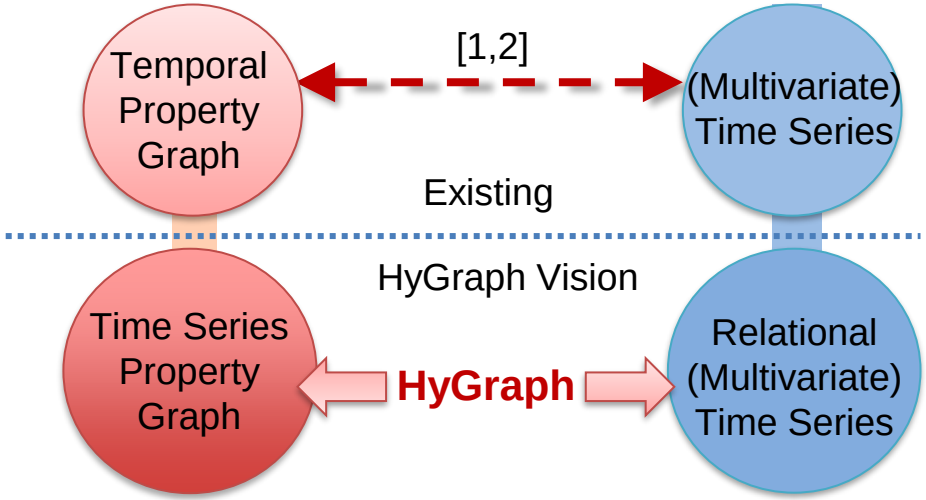
IOT Network

Motivation - The Missing Bridge between Graphs and Time series

- Existing approaches are all task specific
- Lack of a **unified**, **general** approach to transform time series and graph representations

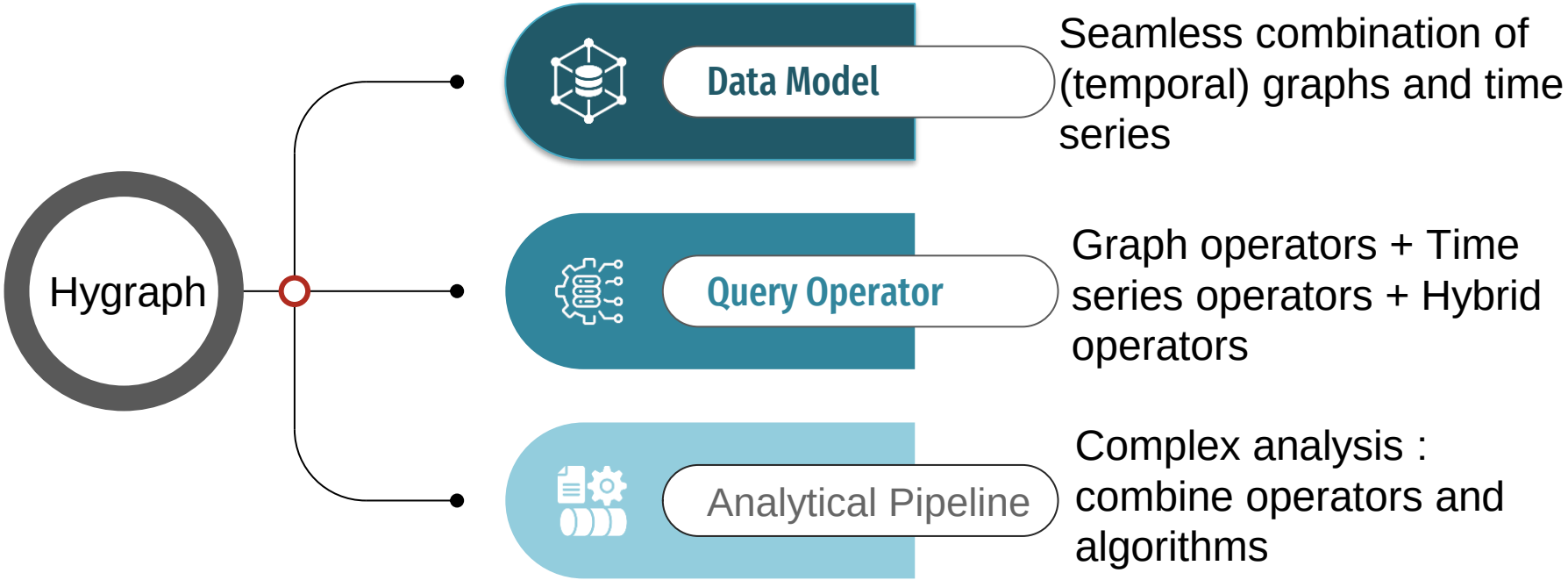
➡ HyGraph:

- Offer a range of configurable transformations.
- Make the edge-creation rule flexible.
- Perform queries that join temporal patterns with graph patterns



[1] Donato Tiano et al.. 2021. FeatTS: Feature-based Time Series Clustering. In SIGMOD '21: International Conference on
[2] E. Bollen et al. 2024. Managing Data of Sensor-Equipped Transportation Networks using Graph Databases. GeoscientificInstrumentation,

HyGraph Vision

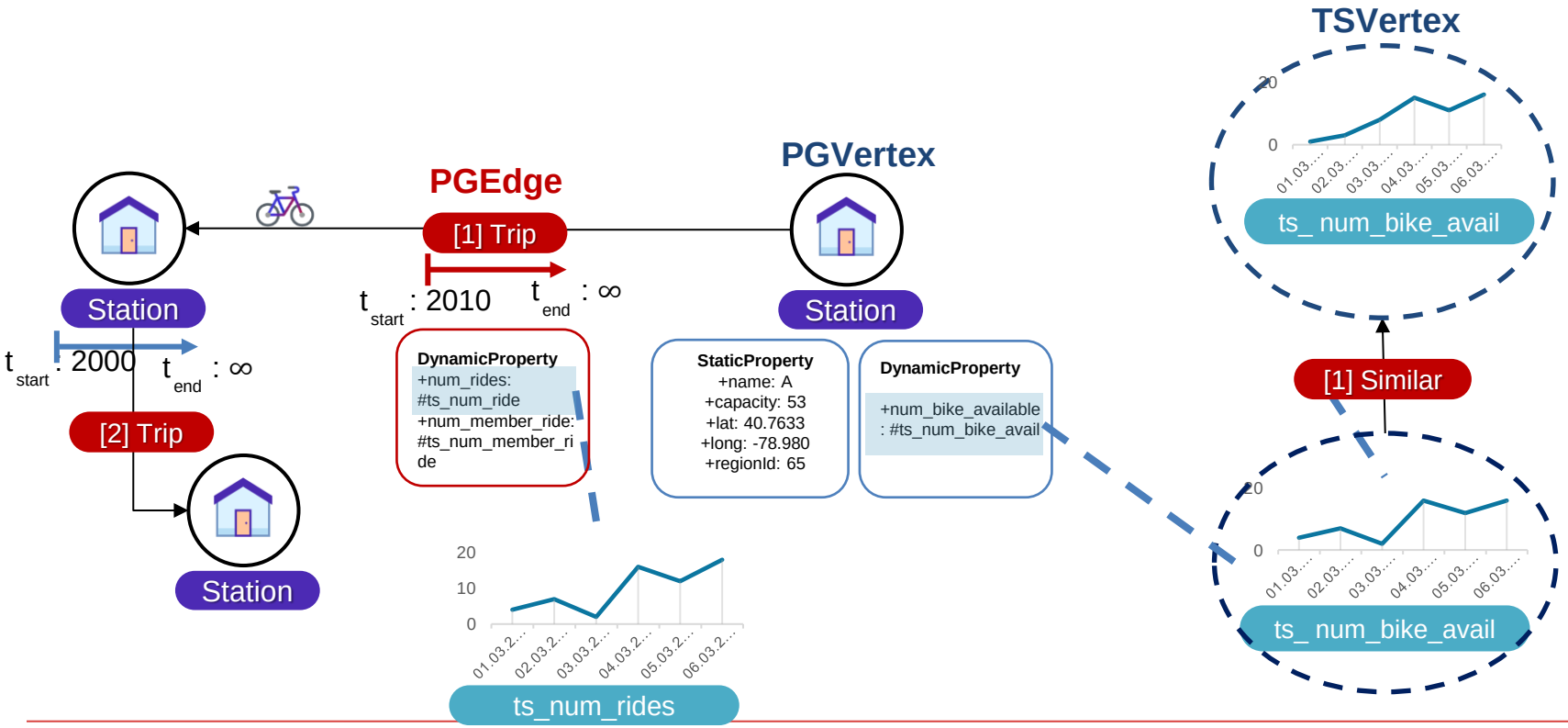


HyGraph Data Model

HyGraph Tuple *HG = (V, E, S, TS, K, N*):

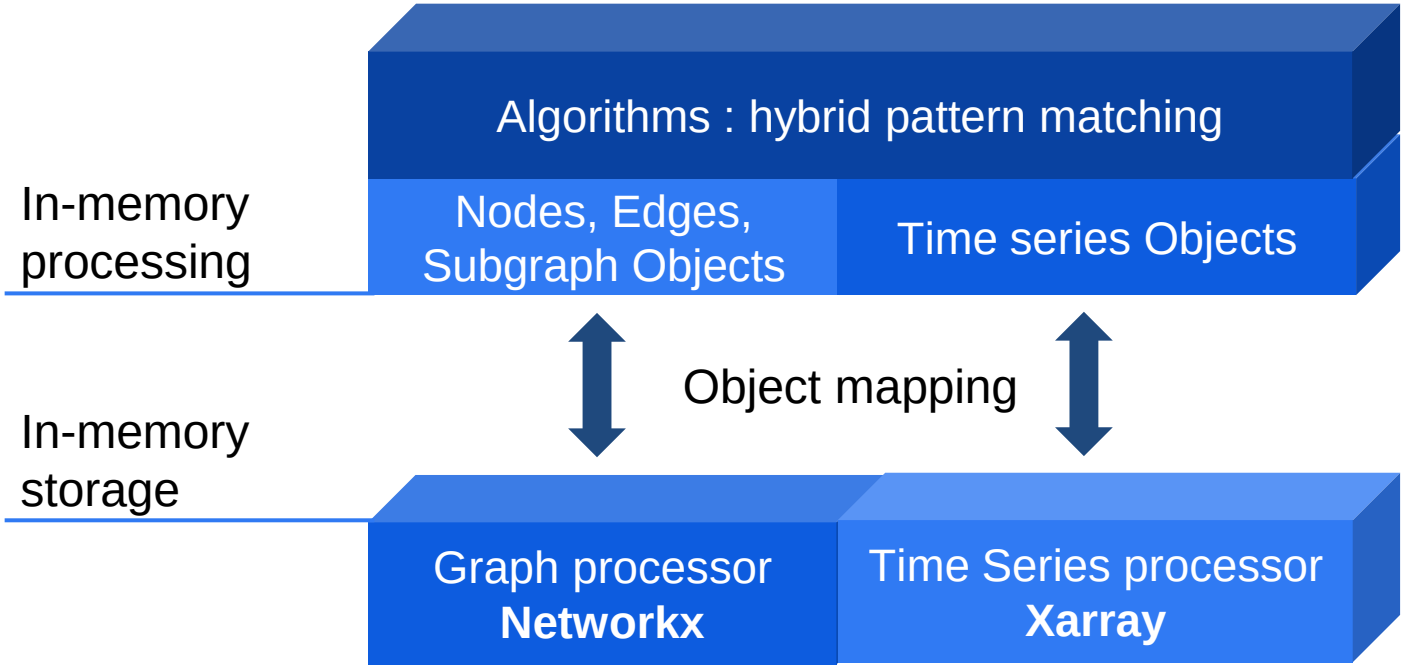
- **Vertices (*V = Vpg* ∪ *Vts*): Property Graph** vertices and **Time Series** vertices.
- **Edges (*E= Epg* ∪ *Ets*): Property Graph** edges **and Time Series** edges).
- **Subgraphs (*S*):** A collection of specific vertices and edges **evolving** over time
- **Time Series (*TS*):** The series of data points over time connected to both vertices and edges.
- **Properties (*K, N*):** Static and dynamic properties linked to different Vpg, Epg, and S.

HyGraph Data Model example – Bike Sharing use case



HyGraph package - System Architecture

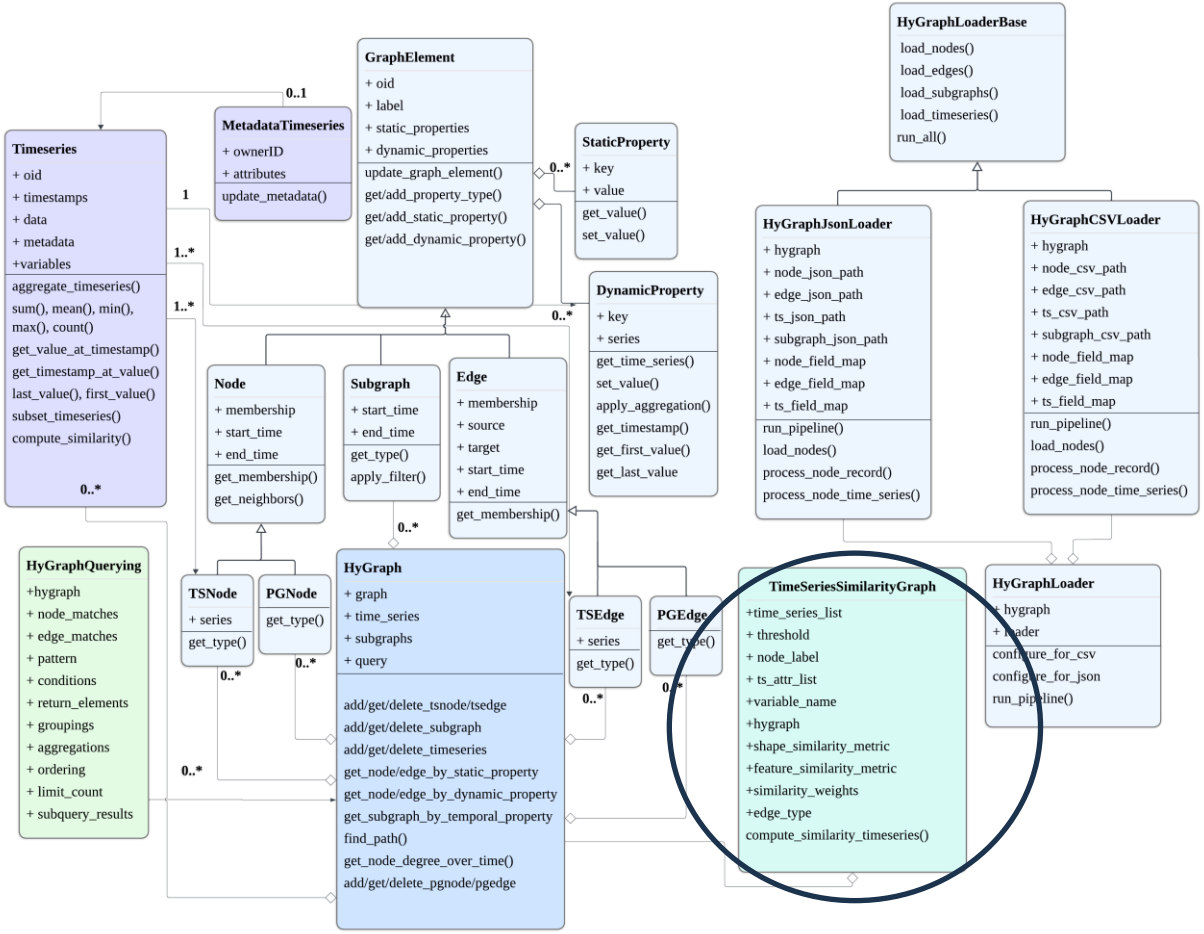
HyGraph open source python Library https://pypi.org/project/hygraph-core/



Main modules

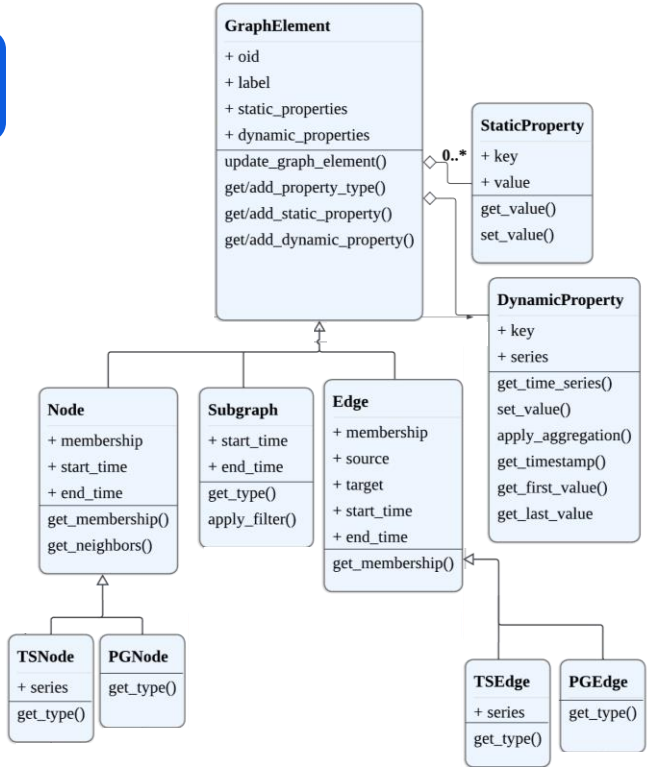
HyGraph Module

- Graph_Operator Module
- Timeseries_op Module
- HygraphQuery Module
- GraphConstruct Module
- HyGraphFileLoaderBatch Module



Main modules

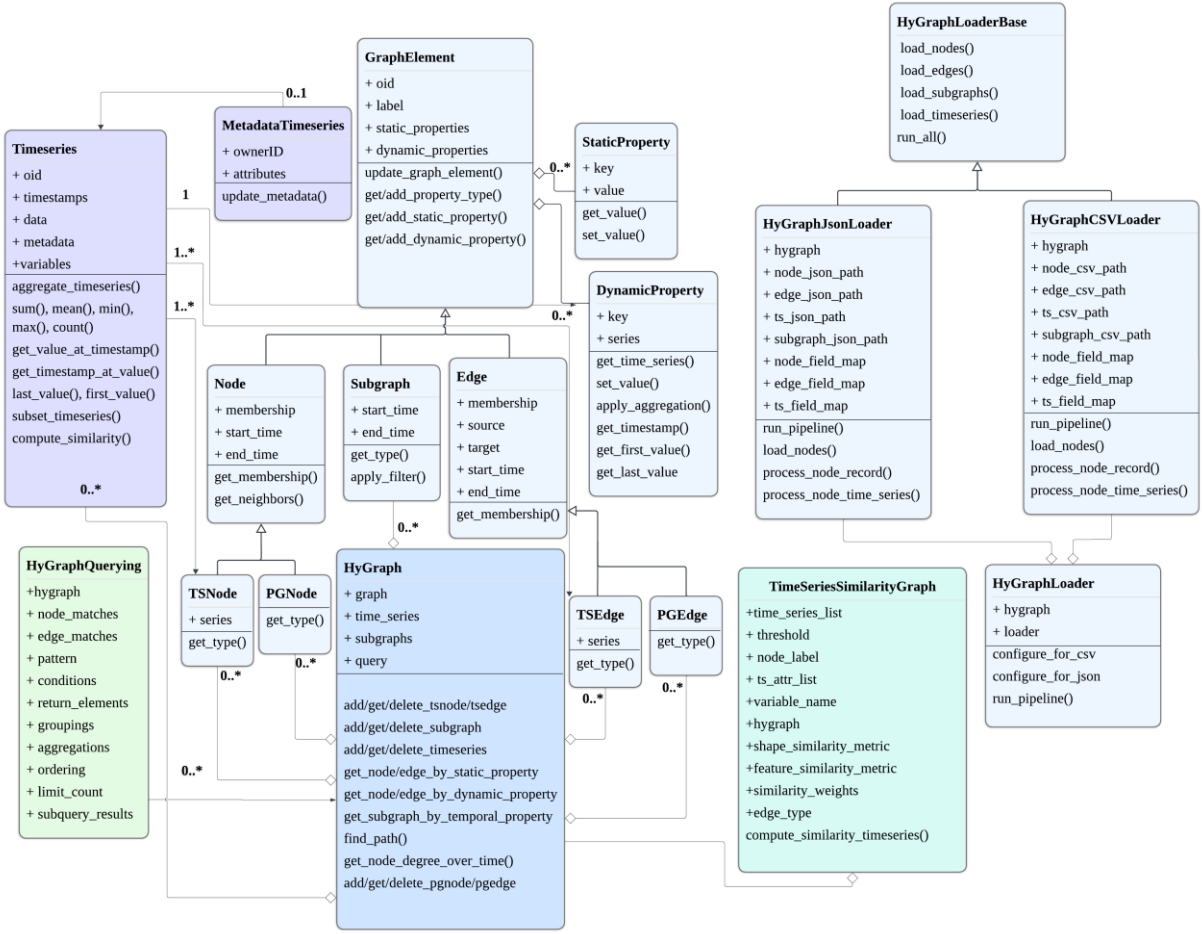
Graph_Operator Module



Main modules

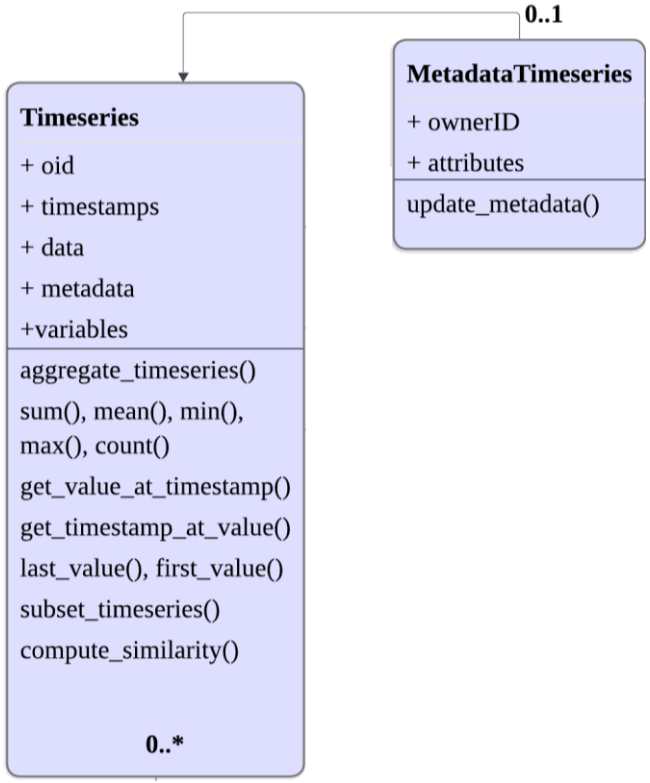
HyGraph Module

- Graph_Operator Module
- Timeseries_op Module
- HygraphQuery Module
- GraphConstruct Module
- HyGraphFileLoaderBatch Module



Main modules

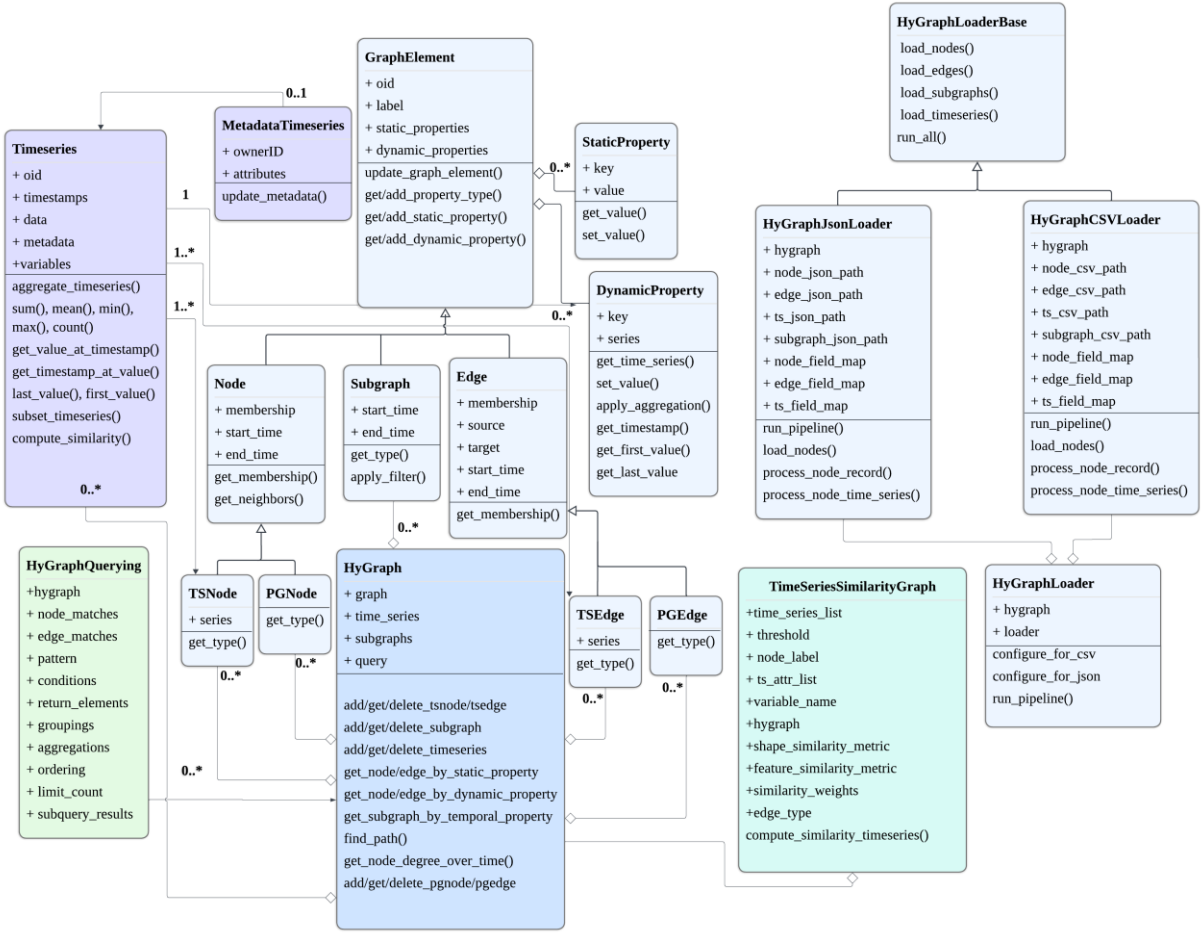
Timeseries_op Module



Main modules

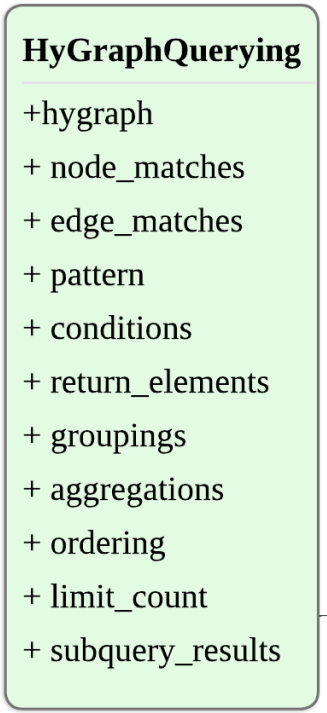
HyGraph Module

- Graph_Operator Module
- Timeseries_op Module
- HygraphQuery Module
- GraphConstruct Module
- HyGraphFileLoaderBatch Module



Main modules

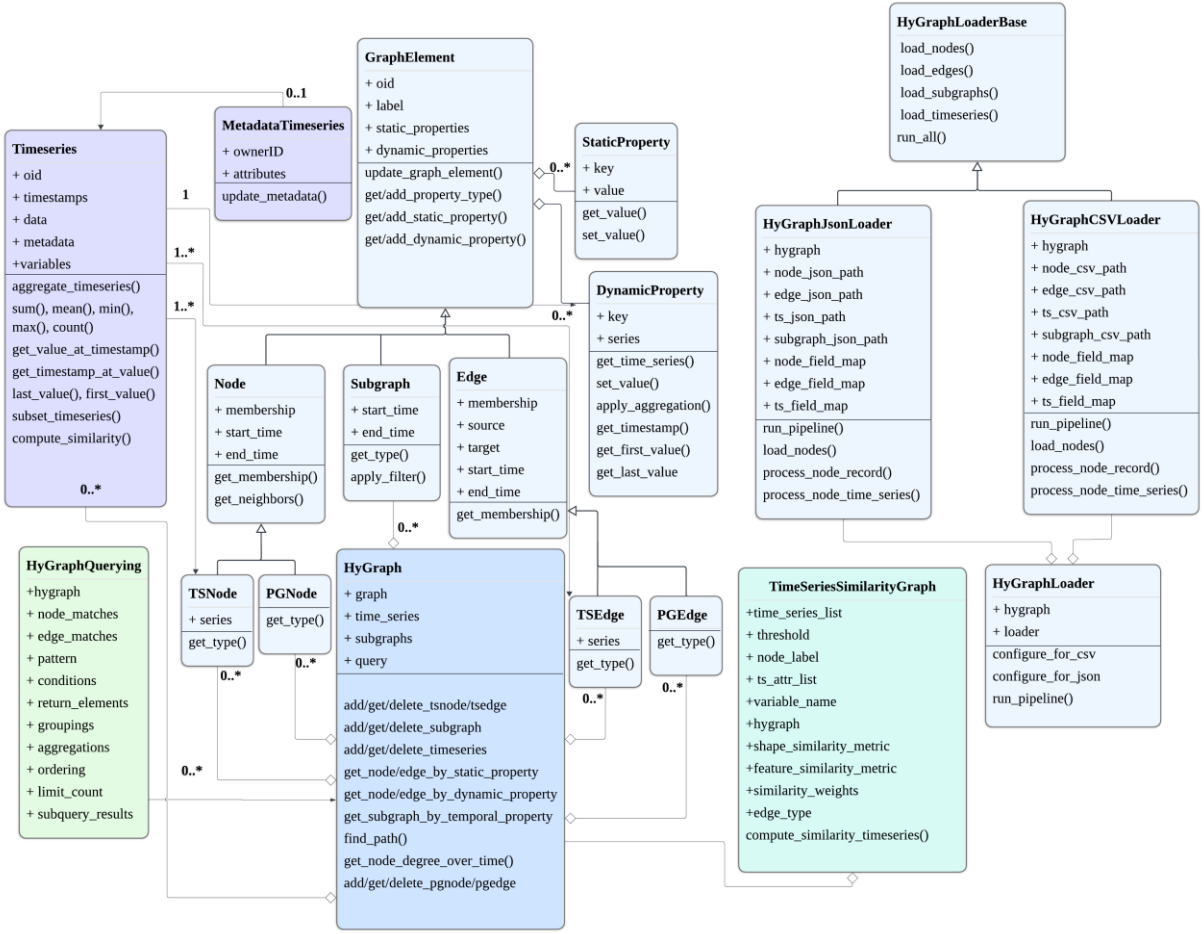
HygraphQuery Module



Main modules

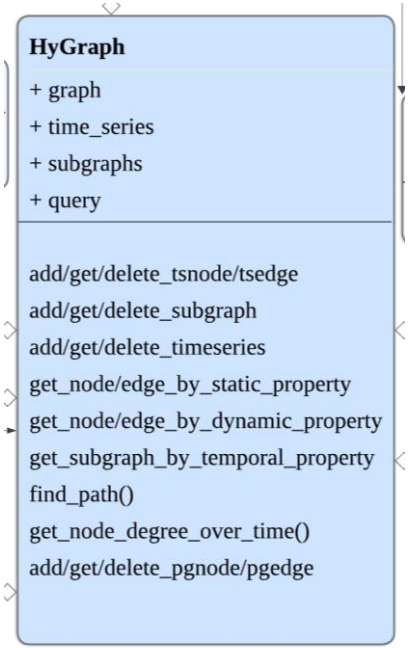
HyGraph Module

- Graph_Operator Module
- Timeseries_op Module
- HygraphQuery Module
- GraphConstruct Module
- HyGraphFileLoaderBatch Module



Main modules

HyGraph Module



HyGraph Operators

Operator	Output
Hybrid Pattern Matching	HyGraph elements matching the defined hybrid pattern.
Temporal Subgraph	Create a subgraph with evolving nodes and edges.
Graph similarity ($GraphSim$)	Constructs a graph based on correlated time-series pattern.
Timeseries metrics generation ($ExtractTS$)	Edge aggregation, membership changes, graph metrics changes.

From Time series to HyGraph - Graph similarity on top of time series data (*GraphSim*)

- Feature Extraction: basic statistical features (e.g., mean, standard deviation, etc)
- Feature Similarity: *compute_feature_similarity* computes the similarity between the feature vectors of two time series.
- Shape Similarity: similarity functions (e.g., euclidean_distance).
- Edge Creation: if avg (similarity) > threshold, edge created:
  - For **PGEdge**, similarities are stored as static properties.
  - For **TSEdge**, similarities over time stored as a TimeSeries object.

From Time series to HyGraph - Graph similarity on top of time series data (*GraphSim*)

- **Goal.** Convert time-series **patterns** into **graph**
- **How it works**. Given a set of time series

$$TS_i = \{x_1^i x_2^i, \dots, x_T^i\}, TS_j = \{x_1^j x_2^j, \dots, x_T^j\}$$

  - Compute **pairwise similarity** $S(i,j)$ using one of the similarity technics (e.g., DTW) Define an **edge creation threshold** τ

$$\textbf{if } S(i,j) > \tau, create\ edge\ e(i,j)$$

TSEdge: Similarity score evolution

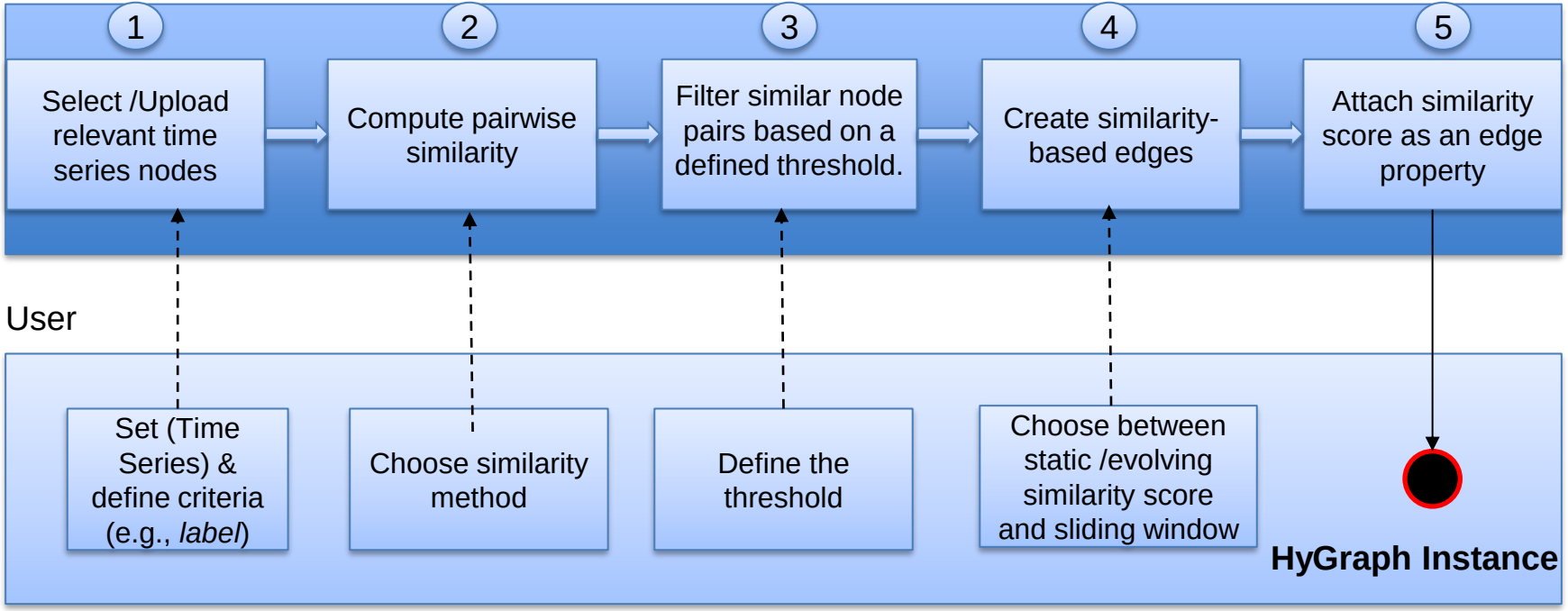
PGEdge: Static Similarity score

Why ?

  - Applying graph operators on time series data (e.g.,community detection)

From Time series to HyGraph - Graph similarity on top of time series data (*GraphSim*)

HyGraph system



From Time series to HyGraph - Graph similarity on top of time series data – Stock Market use case

```
from hygraph_core.construct_graph import (
    build_timeseries_similarity_graph,
    compute_similarity_timeseries
)
```

Importing the function from the package

```
threshold = 0.40
node_label = "BikeStation"              # Label for the newly created TSNodes
variable_name = "num_bike_"             # Variable name in case of a multivariate time seriees
hygraph = build_timeseries_similarity_graph(
    time_series_list=time_series_list, #list of time series data
    threshold=threshold,                #Similarity threshold for edge creation
    node_label=node_label,              #label of teh created node
    ts_attr_list=ts_attr_list,          #metadata about the time-series data
    variable_name=variable_name,
    hygraph=hygraph,
    shape_similarity_metric='correlation',
    feature_similarity_metric='cosine',
    similarity_weights=None,
    edge_type='PGEdge'                  #type of edge created
)
```

From Time series to HyGraph - Graph similarity on top of time series data – Stock Market use case

- Node data

Node ID	Label	TS	Start Time	End Time	Region ID	NodeID	Coordinates
240e9fee-cee3	BikeDemande	#530e9fce-dee4	2015-01-01	2124-04-10	71	5d34te-Der7	(40.7036, -74.0131)
B53df64c-dse5	BikeDemand	#D70e9fee-coe8	2014-02-03	2124-04-10	65	4r34te-ruic9	(40.7036, -74.0131

- Edge data

Edge ID	From	To	Start Time	End Time	Total Similarity Score	Shape Similarity Score	Feature Similarity Score
350e9fee-cte3	B53df64c-dse5	240e9fee-cee3	2024-01-01	2024-04-10	0.93	0.94	0.92
U33df64c-dre5	240e9fee-cee3	738e9fee-uee7	2024-01-01	2024-04-10	0.95	0.94	0.96

HyGraph Query - Most connected domain to ‘Tech’ domain

```
source_region = 'Newport PATH'
query = HyGraphQuery(hygraph)
results = (
    query
    .match_node(alias='node_source', label='Station')
    .where(lambda node_A: node_A.series.metadata.attributes['region'] == source_region)
    .match_edge(alias='sim_edge', label='Similar')
    .match_node(alias='node_target', label='Station')
    .connect('node_A', 'sim_edge', 'node_target')
    .where(lambda connected_node: connected_node.series.metadata.attributes['region'] !=
source_region)
    .group_by(lambda res: res['node_target'].series.metadata.attributes['region'])
    .aggregate(
        alias='connection_count',
        property_name='count',
        method='count',
      direction='both'
    )
    .order_by('connection_count', ascending=False)
    .limit(1)
    .return_(region=lambda res: res['group_key'],
        connections=lambda res: res['connection_count']
   ).execute())
```

Filtering nodes and edges

Grouping nodes

Aggregating edges

Returned values

From Graph To HyGraph – Extract Time series from Graph (ExtractTS)

- **Goal.** Convert **graph dynamics** into **time-series data**
- **How it works.** Given a time-evolving Graph G_t, extract time series for a node, an edge, or a subgraph

$$TS_i = f(G_t)$$

Where f can be:

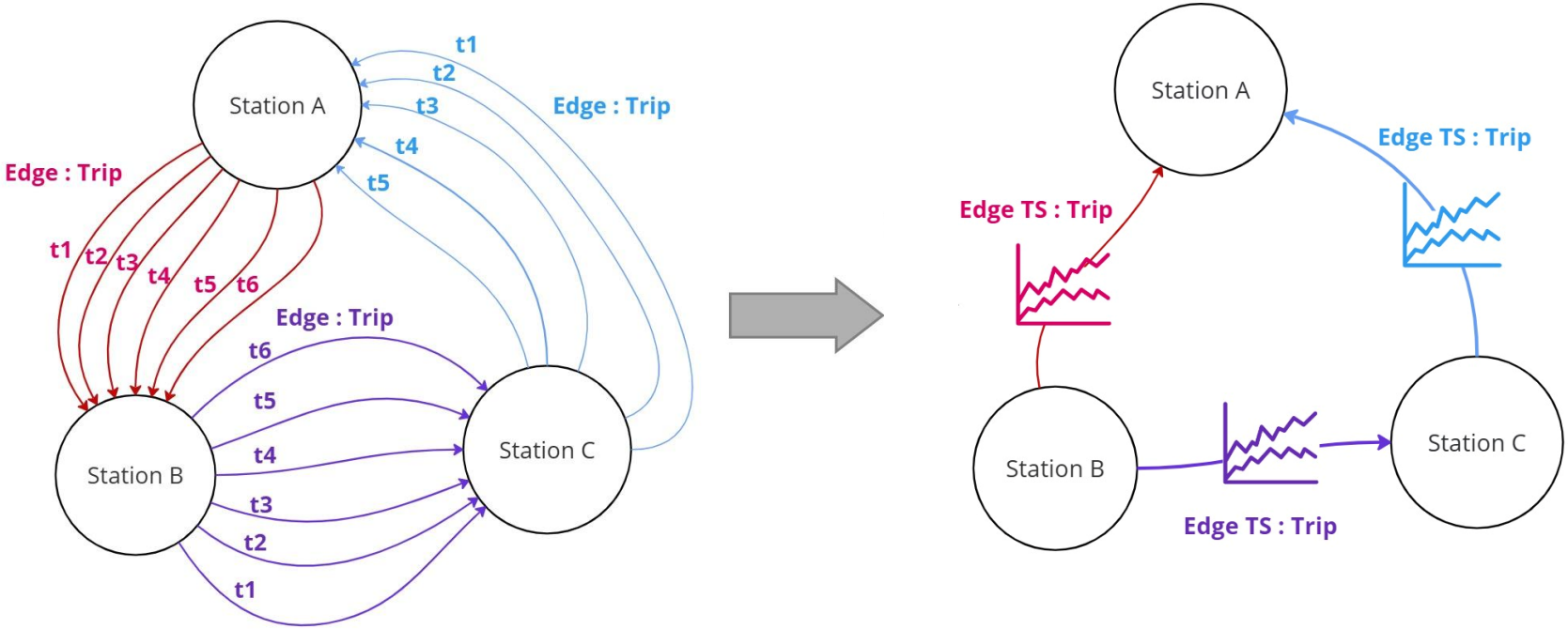
- Aggregated edge values
- Node degree evolution
- Aggregated nodes and edges for a subgraph

Why ?

- Further analysis of graph metrics (e.g., trends in node degree)
- Optimization (Graph summarization)

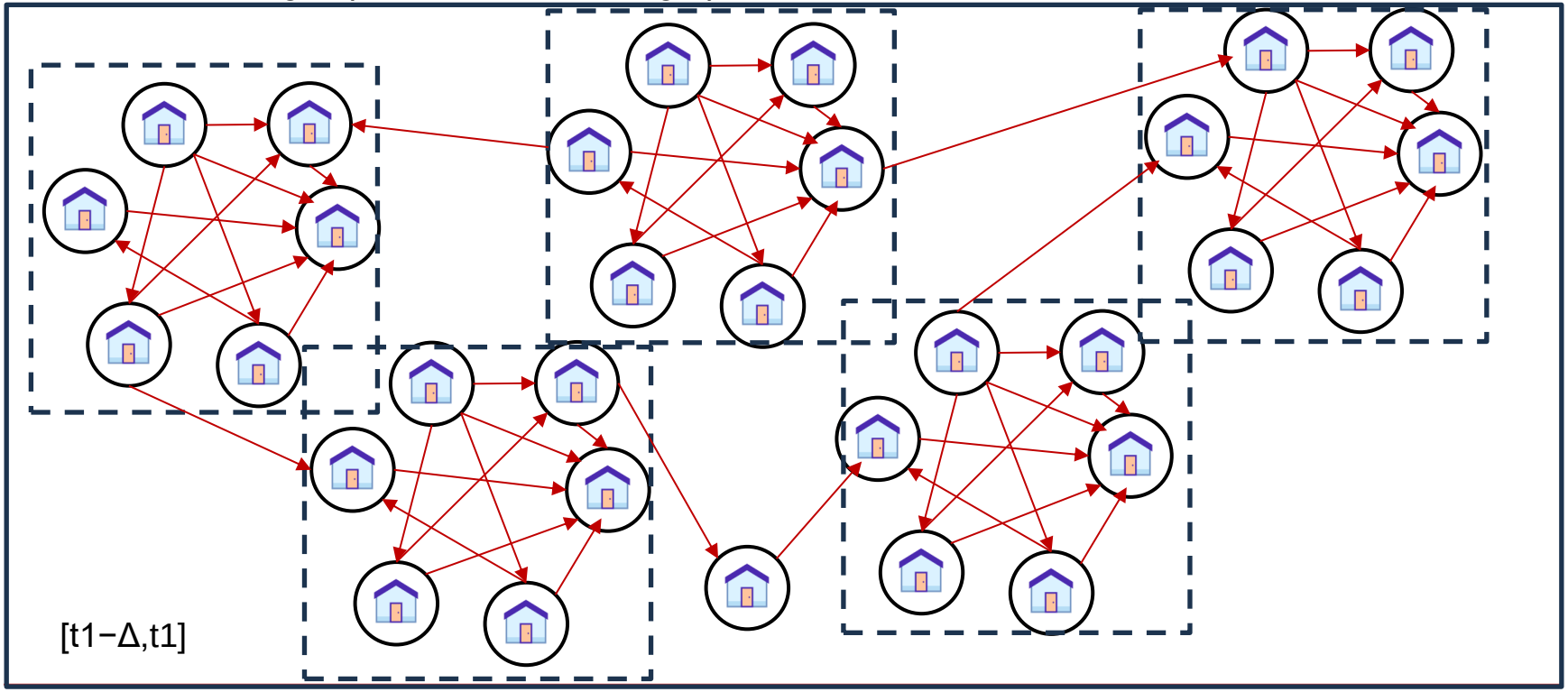
From Graph To HyGraph – Extract Time series from Graph (ExtractTS)

– Aggregate edges of same Station source and target



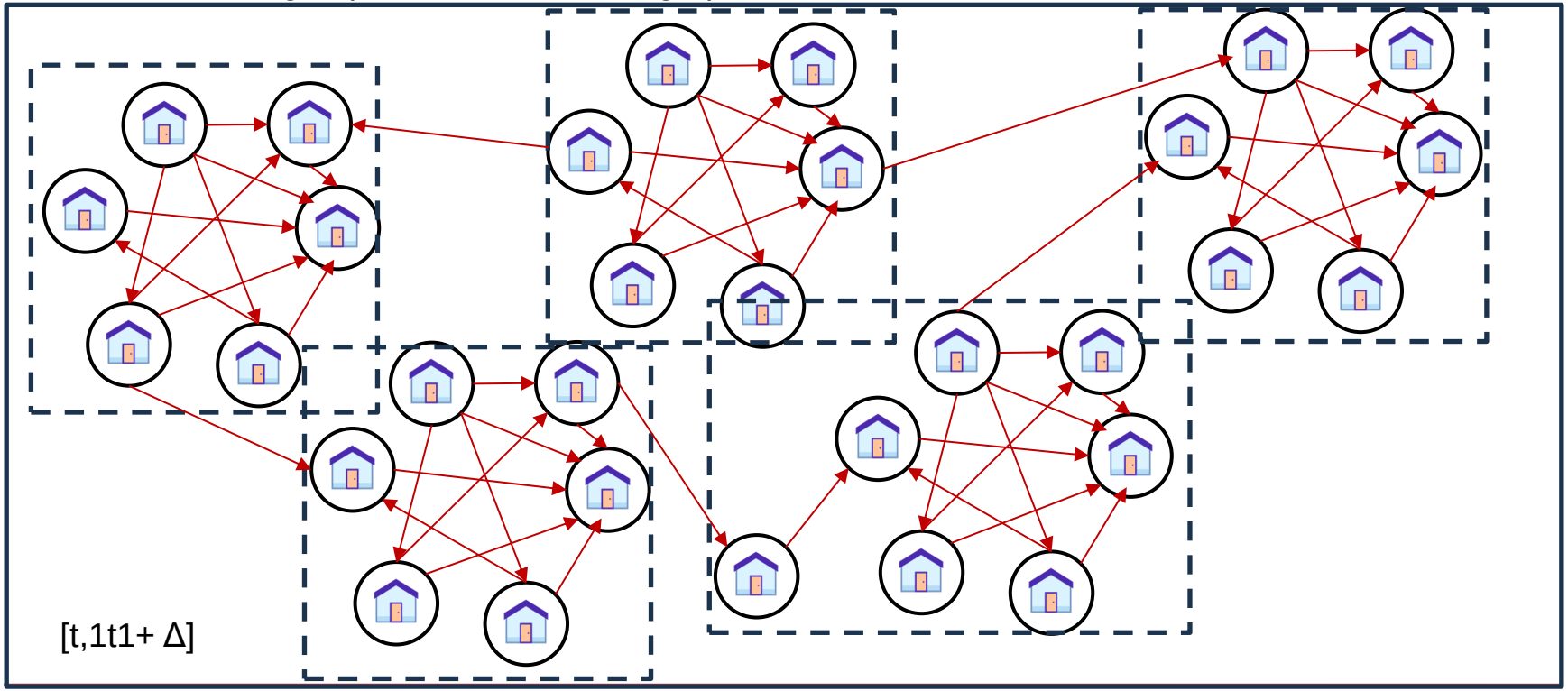
Example: Extracting Time series for an evolving Subgraph

Similar nodes are grouped in the same Subgraph

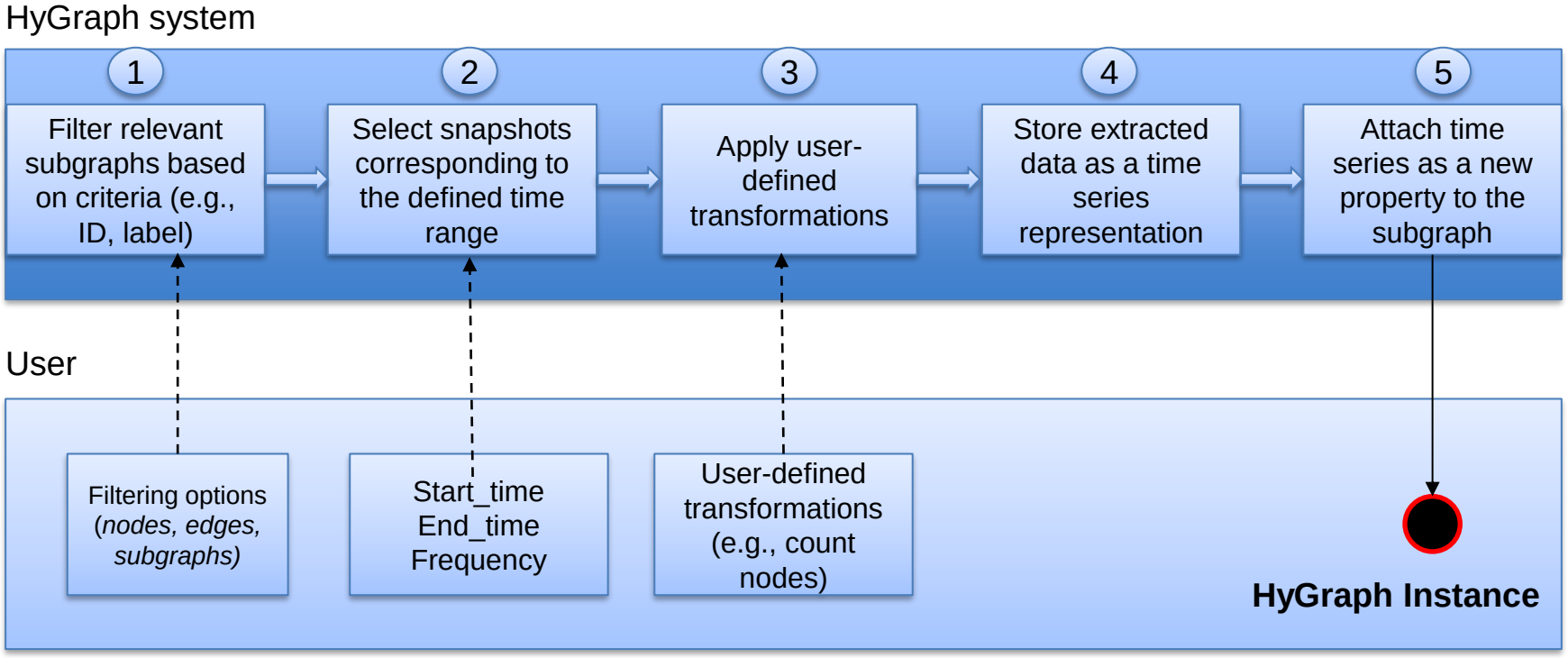


Example: Extracting Time series for an evolving Subgraph

Similar nodes are grouped in the same Subgraph



From Graph To HyGraph – Extract Time series from Graph (ExtractTS)



From Graph To HyGraph – Extract Time series from Graph (ExtractTS)

```python
def subgraph_aggregate(hy, element_type, oid, attribute, dt):
    subgraph = hy.get_subgraph_at(oid, dt)
    node_count = subgraph.number_of_nodes()
    edge_count = subgraph.number_of_edges()
    return (node_count, edge_count)
```

User-defined function

```python
subgraph_query = {
    'element_type': 'subgraph',
    'subgraph_id': 'subgraph_1',
    'time_series_config': {
        'start_date': datetime(2024, 1, 1),
        'end_date': datetime(2024, 1, 31),
        'freq': 'D',
        'attribute': 'subgraph_evolution',
        'aggregate_function': subgraph_aggregate,
        'use_actual_timestamps': False
    }
}
hy.create_time_series_from_graph(subgraph_query)
```

Filtering using Subgraph ID + Run the query

```python
hy.get_element('subgraph',
'subgraph_1').get_temporal_property('subgraph_evolution')
```

Display the newly created property

Conclusion

- **GraphSim**: Extracts **graph structures from time-series similarity** (e.g., linking entities based on evolving correlations).
- **ExtractTS**: Converts **graph dynamics into time-series representations** (e.g., tracking centrality over time).

Open Challenges

- Improve scalability of the operators
- Extend GraphSim by integrating ML-based embeddings
- Support path-based time-series extraction
- Learning-Based TSExtract (Auto-Extract Meaningful Time-Series)

UNIVERSITÄT LEIPZIG

HyGraph

Thanks for your attention !

Contact us to collaborate or to suggest a use case

Visit our website !

https://hygraph.net/