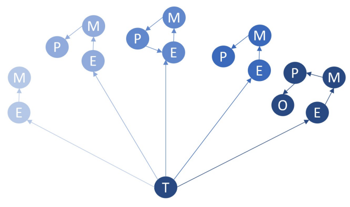
TEMPO

Management and Processing of Temporal Networks

H.F.R.I. Project No. 03480

D1.5: Final Project Report



Computer Engineering & Informatics Department
University of Patras
Greece
30/11/2025

Final Project Report

Kostas Christopoulos (PhD student), Alexandros Spitalas (PhD student),
Kostas Tsichlas (PI)

November 30, 2025

Abstract

This is the final project report, in which we describe the progress we achieved thoughout the whole time span of the project (42 months in total). We report on the deliverables and milestones, as well as on the risks and deviations from the initial plan. We also discuss the work performed within each WorkPackage (WP), and report on dissemination activities.

1 Introduction

The TEMPO project ("Management and Processing of Temporal Networks") represents a significant, forty-two-month research initiative funded by the Hellenic Foundation for Research and Innovation (H.F.R.I.) designed to fundamentally advance the infrastructure and algorithmic capability available for analyzing dynamic, time-evolving networks. In an era where data is generated continuously by social platforms, sensor networks, biological systems, and logistical operations, the static graph models that have dominated computer science for decades are increasingly insufficient. These traditional models compel researchers to aggregate data - thereby discarding critical temporal causality - or to view history as a disjointed series of snapshots, a method that is both storage-inefficient and analytically limiting. TEMPO was conceived to bridge this widening gap by treating time not as an attribute, but as a first-class structural dimension within the database and analytical stack.

This Final Project Report synthesizes the extensive work conducted by the research team at the University of Patras and Aristotle University of Thessaloniki. It documents the successful delivery of a comprehensive ecosystem for temporal graph management, centering on the development of T-JanusGraph (formerly prototyped as MAGMA), a production-grade distributed graph database capable of handling massive historical graphs. Unlike the initial proposal to build a system from scratch, the project strategically pivoted to extending the robust, open-source JanusGraph architecture. This decision, while introducing an initial learning curve that delayed intermediate milestones, ultimately allowed the team to deliver a far more mature and feature-rich system than originally improved, boasting ACID transaction support, multi-backend storage capabilities, and enterprise-grade fault tolerance.

Beyond the storage layer, the project has made profound contributions to query processing and algorithmic theory. The team developed T-Gremlin, a temporal extension of the industry-standard Gremlin traversal language, which integrates Allen's Interval Algebra directly into the query execution flow, enabling researchers to express complex historical questions (e.g., "Find all communities that formed during the breakdown of this specific hub node") with unprecedented ease and efficiency. Furthermore, theoretical advancements in Community Detection, and to a lesser degree, in Outlier Detection have made advancements towards dynamic network analysis.

The project concludes with all major objectives met. The delivered artifacts include open-source software libraries for distributed community detection and outlier detection, a functional temporal database prototype, and a collection of 11 publications that have disseminated these findings to the global research community.

In the remainder of this section, we give an overview of the progress in terms of deliverable preparation and milestone achievements. We also include a discussion regarding risks and deviations from the plan. In Section 2, we provide more details about the technical work in each WP. In Section 3 we discuss deviations from our initial research plan that is natural to happen when looking at research projects. In Section 4 we discuss dissemination activities and the expected impact, and finally, in Section 5 we conclude with a discussion and future directions.

Research Team and Organization The project was executed by a collaborative team of researchers from the University of Patras and the Aristotle University of Thessaloniki. The composition of the team remained stable throughout the project lifecycle, ensuring continuity in the research vision and implementation strategies.

1. Assoc. Prof. Konstantinos Tsichlas (PI, University of Patras): Provided overall scientific coordination, defining the algorithmic direction for community detection and the architectural vision for the HiNode implementation. His expertise in distributed algorithms guided the theoretical aspects of the project.
2. Prof. Spyros Sioutas (University of Patras): Contributed deep expertise in graph-distributed databases, specifically guiding the design of data structures for efficient storage and retrieval in distributed environments.
3. Prof. Anastasios Gounaris (Aristotle University of Thessaloniki): A specialist in distributed systems, Prof. Gounaris guided the optimization of the query engine and the distributed execution of analytical algorithms.
4. Assoc. Prof. Apostolos Papadopoulos (Aristotle University of Thessaloniki): Provided crucial guidance on graph database architectures and OLAP technologies, particularly the integration with Apache Spark for large-scale analytics.
5. Alexandros Spitalas (PhD Student): Led the engineering of the system prototype (WP2 and WP3). His work focused on forking JanusGraph and developing T-JanusGraph, and developing the T-Gremlin query language extensions.
6. Konstantinos Christopoulos (PhD Student): Focused on the algorithmic deliverables (WP4 and WP5). He developed the LCDS-A framework for community detection, the distributed triangle counting algorithms, and the outlier detection methodologies.

This team structure effectively bridged the gap between theoretical algorithm design and practical systems engineering, a synergy that was essential for the project's success.

1.1 Milestones

There were two milestones anticipated for the final year: MS3 (planned at month 24) and MS4 (planned at month 36). Milestone MS3, achieved approximately by Month 36, marks the delivery of the Initial Version of the Query Engine Module, validating the project's strategic pivot to the T-Gremlin traversal language. This milestone signifies the successful integration of simple analytic capabilities into the T-JanusGraph system, characterized by the native implementation of Allen's Interval Algebra as first-class traversal steps. Its completion confirmed the feasibility of performing semantic temporal operations directly within the query execution flow.

Milestone MS4, achieved at Month 42, marks the completion of the prototype of the T-JanusGraph system, a distributed historical graph database that fulfills the core objectives of WP2. Verified by Deliverable D2.2, this final prototype successfully implements the HiNode storage model by extending the JanusGraph architecture to support ACID transactions and integrating Apache Solr for high-precision temporal indexing. The milestone represents a robust, scalable system featuring a novel temporal placement strategy for optimized data locality and efficient retrieval, serving as the necessary infrastructure to support the project's complex analytical queries and community detection algorithms.

2 Summary of Work per WP

A detailed report for each work-package follows.

2.1 WP1: Project Management - Result Dissemination

The main activity that took place was the preparation of the Final Project Report as well as 3 presentations at international conferences, as will be detailed in Section 4.

2.2 WP2: MAGMA System Implementation

The cornerstone of the TEMPO project is T-JanusGraph, a sophisticated temporal graph database. The genesis of this system involved a critical decision-point early in the project: whether to build a system (code-named MAGMA) entirely from scratch or to extend an existing platform. The team chose to fork and extend JanusGraph, a leading open-source distributed graph database.

Strategic Pivot: From Scratch to Extension The initial proposal considered building a custom engine to ensure maximum control over the HiNode implementation. However, modern graph databases require a massive infrastructure - consensus protocols, replication strategies, ACID transaction managers, and multi-backend adapters - before any novel graph functionality can be implemented. Building this from scratch would have consumed the project's resources on solved problems rather than innovation. By pivoting to JanusGraph, the team inherited a production-grade foundation:

1. *Storage Agnosticism:* JanusGraph natively supports Apache Cassandra, HBase, and BerkeleyDB. This allows T-JanusGraph to leverage Cassandra's wide-column store architecture, which is ideal for the high write throughput required by temporal graph updates.

2. *ACID Transactions:* Supporting atomic, consistent, isolated, and durable transactions in a distributed system is notoriously difficult. JanusGraph's existing transaction manager handles complex coordination, allowing the TEMPO team to focus on ensuring temporal consistency rather than basic data safety.

3. *Horizontal Scalability:* The underlying architecture supports linear scalability by adding nodes to the cluster, a prerequisite for handling massive historical graphs as outlined in the project goals.

While this decision ultimately accelerated the project, it introduced an initial delay (affecting Milestone MS2) as the team engaged in a deep reverse-engineering of JanusGraph to understand how to inject the HiNode model and temporal semantics without breaking the core architecture.

The TemporalPlacementStrategy One of the most persistent performance bottlenecks in distributed graph databases is the graph partitioning problem. To execute a query like "find the neighbors of node v," the database must retrieve v and its edges. If v resides on server A, but its edges connect to nodes on server B and server C, the query requires network communication between servers, which is orders of magnitude slower than local memory or disk access. In temporal graphs, this problem is exacerbated. Queries often ask for interactions over a specific time window. If the historical versions of a node or its relevant temporal neighbors are scattered across the cluster, performance degrades catastrophically. T-JanusGraph addresses this with a novel Temporal Placement Strategy that optimizes across two dimensions: the total weight of the temporal cuts and load balancing.

Temporal Indexing with Apache Solr Storage is only half the story; retrieval is the other half. Temporal queries are often stab queries (e.g., "What was the state of the network at 10:00 AM on Jan 1st?") or range queries (e.g., "Find all edges active between 2020 and 2022"). Without an index, the database would have to scan every edge in the graph to check its valid interval, an $O(n)$ operation that is non-viable for massive graphs. T-JanusGraph integrates tightly with Apache Solr to provide a specialized temporal indexing layer. In a nutshell, the system indexes `startTime` and `endTime` properties with millisecond precision. The team implemented mixed indices (e.g., EdgeIndexStartEnd) that combine vertex properties with temporal bounds. This allows the query optimizer to execute complex lookups - such as "find all edges of type 'Transfer' active in 2023 with value $>$ 1000 euros" - using a single index lookup rather than filtering results in memory. This indexing infrastructure is what powers the T-Gremlin language extensions. When a user writes a query using temporal Overlaps, T-JanusGraph translates this into a Solr range query that efficiently retrieves only the relevant graph elements.

Fault Tolerance and Adaptive Replication T-JanusGraph inherits the robust replication mechanisms of its storage backends (e.g., Cassandra's replication factor), ensuring that data is durable even if individual nodes fail. However, the project explored theoretically a more sophisticated approach: Temporal Adaptive Replication. In a static graph, all nodes are often treated as equally important for replication purposes. In a temporal graph, some entities are "temporally heavy" - they persist for the entire lifetime of the network and are involved in constant interactions, while other nodes are transient. The adaptive replication strategy, currently under testing, dynamically adjusts the replication factor based on the "temporal importance" of the entity. Long-lived, high-interaction nodes are replicated more aggressively across the cluster to serve high read traffic, while transient nodes are replicated less to save storage. This represents a nuanced optimization of the trade-off between availability, performance, and storage cost.
For more information, we refer the reader to deliverable D2.2.

2.3 WP3: Complex Data Analytic Queries

Traditional graph query languages like Gremlin or Cypher were designed for static graphs. While they can store time as a property (e.g., `edge.property('time', 100)`), they lack the semantic understanding of time. Writing a query to find overlapping intervals in standard Gremlin requires verbose, error-prone Boolean logic. TEMPO delivered T-Gremlin, a comprehensive fork of the Apache TinkerPop Gremlin language, which elevates time to a first-class citizen. This means temporal concepts are built into the language's syntax and execution engine.

The Structured Temporal Property Model T-Gremlin enforces a strict schema for temporal data to ensure consistency and enable optimization. It introduces the lifetime (`startTime, endTime`) step, a dedicated construct for defining validity intervals. Unlike standard Gremlin where users might name properties time, time stamp, date, or ts, T-Gremlin mandates standardized property keys (`startTime, endTime`). This predictability allows the underlying engine to automatically apply temporal indices without requiring the user to manually specify them. In addition, the language enforces logic constraints at the API level. For example, it prevents the creation of an edge with a validity interval that falls outside the existence intervals of its connecting vertices, preventing logical inconsistencies in the historical graph. Finally, the system handles the parsing and serialization of dates automatically, supporting various formats (ISO 8601) and ensuring that temporal comparisons are mathematically valid.

Integration of Allen's Interval Algebra The most significant contribution of T-Gremlin is the native implementation of Allen's Interval Algebra. Proposed by James F. Allen in 1983, this algebra defines the 13 mutually exclusive relationships that can exist between any two time intervals (e.g., Before, After, Meets, Overlaps, Starts, Finishes, During, etc.). Implementing these as native traversal steps transforms considerably the expressiveness of the language. In this way, we get syntactically cleaner and at the same time semantically precise queries. All 13 operators are implemented, enabling nuanced historical queries. For example, temporalMeets can be used to find causal chains where one event ends exactly as the next begins, crucial for analyzing logistics or process flows.
For more information, we refer the reader to deliverable D3.2.

2.4 WP4: Local Community Detection

Community detection - the identification of dense clusters of nodes - is a staple of network analysis. In static graphs, the problem is well-defined. In temporal networks, it faces the "identity problem": if a community grows, shrinks, splits, or merges over time, is it still the "same" community? TEMPO addressed this through a local, anchor-based framework for tracking evolution. In addition, TEMPO also addressed the computational intractability of finding community in massive historical graphs by utilizing contemporary parallel/distributed frameworks (SPARK) that can be executed over T-JanusGraph.

The LCDS-A Framework: Solving the Identity Problem The project introduced the LCDS-A (Local Community Detection in graph Streams with Anchors) framework. This novel approach shifts the analytical lens from global partitioning (finding all communities at once) to local tracking (following one community through time). Instead of defining a community solely by its topology (which changes), LCDS-A defines it relative to a stable reference point: the *anchor*. An anchor is a node with external semantic significance - for example, the manager in a corporate email network, or a patient zero in an epidemic. The community is defined as the dense cluster surrounding the anchor at any given time t.

Distributed Community Detection in Historical Graphs For scenarios where the user needs to analyze the entire graph structure over a specific query time interval (e.g., "Find all communities that existed during 2023"), the project developed distributed algorithms capable of running on Spark clusters. Based on two algorithmic approaches (label propagation and weighted clustering coefficient), we proposed various versions, among which:

- t-iWCC (Temporal Interval Weighted Clustering Coefficient): This algorithm explicitly computes the intersection of validity intervals. A triangle is only counted if all three edges exist simultaneously for a duration within the query interval. This provides the most rigorous temporal accuracy.
- t-wWCC (Temporal Weighted WCC): This variant converts temporal overlap into a scalar weight (the *Observation Ratio*). If an edge exists for 6 months of a 1-year query window, it gets a weight of 0.5. This simplifies the computation while preserving temporal significance.
- t-iLPA and t-wLPA (Label Propagation): Parallel versions of the fast Label Propagation Algorithm, adapted to propagate labels based on temporal weights or interval intersections.

The efficiency and effectiveness of these algorithms have been verified experimentally. For more information, we refer the reader to deliverables D4.2 and D4.3.

2.5 WP5: Outlier Detection

The final pillar of the project focused on identifying anomalies - patterns that deviate significantly from the norm. In temporal networks, anomalies are often structural events (e.g., the sudden dissolution of a stable group) rather than simple statistical outliers. TEMPO explored two methodologies: community-based detection and deep learning. During the project, deep results could only be achieved if we focused either on community detection or on outlier detection. We chose to focus on the former, since outlier detection can be accomplished by using community detection methods as shown in our first approach.

Community-Based Anomaly Detection Leveraging the work from WP4, this approach detects anomalies by monitoring the lifecycle of communities. If a community that has been stable for months suddenly exhibits erratic behavior, it is flagged as an anomaly. To accomplish this, we formalized six pivotal evolutionary events to characterize community dynamics: a) *Growth*: Significant increase in size, b) *Contraction*: Significant decrease in size, c) *Birth*: Emergence of a new community, d) *Vanish*: Disappearance of a community, e) *New "Expanded"*: A community splits or merges into a larger form, and f) *New "Shrank"*: A community splits or degrades into a smaller form.

Deep Learning Approaches: DyGED Recognizing the power of neural networks for pattern recognition, the team investigated DyGED (Dynamic Graph Event Detection), a state-of-the-art deep learning framework. DyGED combines Graph Neural Networks (GNNs) to capture the spatial topology of the graph (who connects to whom) with Recurrent Neural Networks (LSTMs) and Attention mechanisms to model temporal evolution (how connections change). Unlike simple node embeddings, DyGED focuses on macro-dynamics - it pools node representations to create a graph-level embedding. Preliminary experiments on synthetic datasets were highly promising. While the complexity of training GNNs makes them computationally heavier than the algorithmic approaches (LCDS-A), they offer superior sensitivity to subtle, non-linear patterns that heuristic algorithms might miss.
For more information, we refer the reader to deliverables D5.2 and D5.3.

3 Risks and Deviation from the Plan

Research is an iterative process, and the TEMPO project successfully navigated significant technical risks by adapting its plan to technical realities.

3.1 The Pivot to JanusGraph (WP2)

The most critical deviation was the decision to abandon the ground-up build of the MAGMA system. The team realized that replicating the fundamental distributed systems primitives required for a modern database (consensus, replication, write-ahead logging) would consume the entire project timeline. This decision caused a 6-month delay in Milestone MS2 (Initial Prototype) due to the steep learning curve of the JanusGraph codebase. The team had to essentially become contributors to a complex open-source project rather than architects of their own system. However, this was a high-return investment. It allowed the project to deliver a system with ACID transactions and Cassandra integration - features that were effectively impossible to build from scratch within the project timeframe. The delay was absorbed, and the final deliverable is significantly more robust than the original plan anticipated.

3.2 The Query Engine Pivot (WP3)

Concurrently, the team replaced the planned custom query engine with Gremlin. This decision effectively neutralized the schedule slip from WP2. Because Gremlin provides a mature traversal machine, the team could focus entirely on implementing the temporal semantics (Allen's Algebra) rather than building a query parser and optimizer.

3.3 Integration Strategy for Analytics

Due to the complexity of the T-JanusGraph core, the advanced analytics (WP4/WP5) were developed as standalone distributed libraries (Scala/Spark) rather than native database kernels. This decoupling allowed parallel development. While they cannot yet run inside the database transaction loop, they are fully functional applications that process data exported from T-JanusGraph, satisfying the project's analytical requirements.

4 Dissemination and Impact

The project has made a substantial contribution to the scientific literature, producing 11 key publications that validate its theoretical and practical outputs:

1. **State-of-the-art in Community Detection in Temporal Networks** [8] that contains the work within deliverable D4.1.
2. **MAGMA: Proposing a Massive Historical Graph Management System** [13] that contains part of the survey within deliverable D1.2 as well as a preliminary version of the system architecture of MAGMA.
3. **Dynamic Local Community Detection with Anchors** [4] that contains a streaming algorithm for local community detection by using anchors instead of seeds.
4. **Local Community Detection: A survey** [5] that contains a survey on local community detection algorithms based partially on deliverable D4.1. Note that no such stand-alone survey existed for the problem of local community detection.
5. **Local Community Detection in Graph Streams with Anchors** [9] that contains extended research on local community detection in a streaming environment based on [4].
6. **Adopting Different Strategies for Improving Local Community Detection: A Comparative Study** [3] that introduce new variants of an existing local community detection algorithm that uncover a single community.

7. **Local Community-Based Anomaly Detection in Graph Streams** [1] that introduces an established algorithm for dynamic local community detection, aimed at identifying anomalies based on community evolution over time.

8. **Degree Distribution Optimization in Historical Graphs** [14] that builds HiNode model on top of CockroachDB, a Distributed SQL database, for more resilience of data, and also introduces a new space-efficient algorithm for degree distribution query in historical graphs. It was given the **best paper award**.

9. **Triangle Counting in Large Historical Graphs** [16] that provides definitions and methods for counting triangles in massive historical graphs, implemented in SPARK.

10. **Partition Strategies for Vertex-Centric Historical Graph Systems** [15] that proposes new strategies for partitioning of historical graphs with a time-interval representation of history in a distributed system.

11. **Distributed Community Detection in Temporal Graphs** [17] that proposes new algorithms for identifying communities in historical graphs with time-interval representation of history in a distributed environment.

Our primary objective during the project's final year was to disseminate our findings regarding T-JanusGraph, T-Gremlin, and distributed community detection algorithms. To this end, we have submitted a manuscript titled "Distributed Community Detection in Historical Graphs" to the International Journal of Data Science that investigates the efficiency and effectiveness of distributed community detection for historical graphs; this submission marks the conclusion of K. Christopoulos's doctoral studies. Concurrently, A. Spitalas is preparing a submission for a top-tier conference (e.g., VLDB or SIGMOD) focusing on T-JanusGraph and T-Gremlin. Furthermore, pending access to our department's 28-node cluster, we will initiate experiments on partitioning strategies to support a subsequent journal publication. These two contributions will complete A. Spitalas's PhD requirements. Collectively, these three publications represent the culmination of the core research conducted within TEMPO. However, the results of TEMPO will impact our research for the coming years in community detection as well as for anomaly detection.

Open Source Impact All software components are open-source, ensuring reproducibility and community adoption. All systems/libraries have been on github, free to access for everyone.

5 Conclusion and Future Directions

The TEMPO project successfully met its ambitious goal: to create a holistic ecosystem for the management and processing of temporal networks. It moved the field beyond the limitations of static snapshots, providing a valid-time database (T-JanusGraph), a temporal query language (T-Gremlin), and a suite of temporally-aware analytical algorithms. Our key achievements include:

1. Time as a First-Class Citizen: Temporal logic is no longer an application-layer burden but a native database capability.

2. Locality-Aware Storage: The TemporalPlacementStrategy provides a solution for the critical performance bottleneck of distributed historical traversals.

3. Algorithmic Innovation: The Anchor framework provides a robust solution to the community identity problem in evolving networks. In addition, we are the first to provide algorithmic approaches for historical graphs (interval-based) for community detection.

The team is already looking ahead. Immediate next steps involve the tight coupling of the analytical libraries with the database kernel to enable real-time, in-transaction analytics. Furthermore, the team has secured follow-up research initiatives (Trust Your Stars Action I: DYNCEG - DYNamic data structures and Causal explanations for historical Evolving Graphs) to explore Time-Sensitive Replication and Multi-Version Storage optimizations (among others), building directly on the TEMPO infrastructure. The project leaves behind not just a report, but a functional, open-source platform that will support future research.

References

[1] **Local Community-Based Anomaly Detection in Graph Streams.** Konstantinos Christopoulos, and Konstantinos Tsichlas. *In Proc. of the 20th International Conference on Artificial Intelligence Applications and Innovations*, pp. 348-361, 2024.

[2] **High quality, scalable and parallel community detection for large real graphs.** Prat-Pérez, Arnau, David Dominguez-Sal, and Josep-Lluis Larriba-Pey. *In Proc. of the 23rd international conference on World wide web*, 2015.

[3] **Adopting Different Strategies for Improving Local Community Detection: A Comparative Study.** Konstantinos Christopoulos, and Konstantinos Tsichlas. *In Proc. of the 12th Intern. Conference on Complex Networks and their Applications*, 2023.

[4] **Dynamic Local Community Detection with Anchors.** G. Baltsou, K. Christopoulos and K. Tsichlas. *In Proc. of the 11th Intern. Conference on Complex Networks and their Applications*, 2022.

[5] **Local Community Detection: A survey.** G. Baltsou, K. Christopoulos and K. Tsichlas. *IEEE ACCESS*, vol. 10, pp. 110701-110726, 2022.

[6] **Local community detection with hints.** G. Baltsou, K. Tsichlas and A. Vakali. *Applied Intelligence*,doi:10.1007/s10489-021-02946-7, 2022.

[7] **Dynamic Community Detection with Anchors (Extended Abstract)** G. Baltsou and K. Tsichlas. *In Proc. of the 10th Intern. Conference on Complex Networks and their Applications*, 2021.

[8] **State-of-the-art in Community Detection in Temporal Networks.** K. Christopoulos and K. Tsichlas. *In Proc. of the 18th International Conference on Artificial Intelligence, Applications, and Innovations (AIAI) - Mining Humanistic Data Workshop (MHDW)*, 2022.

[9] **Local Community Detection in Graph Streams with Anchors.** K. Christopoulos, G. Baltsou and K. Tsichlas. *Information*, 14(6): 332, 2023.

[10] **HiNode: An Asymptotically Space-Optimal Storage Model for Historical Queries on Graphs.** A. Kosmatopoulos, K. Tsichlas, A. Gounaris, S. Sioutas and E. Pitoura. *Distributed and Parallel Databases*, 35(3-4):249–285, 2017.

[11] **Hinode: Implementing a Vertex-Centric Modelling Approach to Maintaining Historical Graph Data.** A. Kosmatopoulos, A. Gounaris and K. Tsichlas. *Computing*, pp. 1-24, 2019.

[12] **Investigation of Database Models for Evolving Graphs.** A. Spitalas, A. Gounaris, A. Kosmatopoulos, K. Tsichlas. *In proc. of the 28th International Symposium on Temporal Representation and Reasoning(TIME)*, pp. 6:1-6:13, 2021.

[13] **MAGMA: Proposing a Massive Historical Graph Management System.** A. Spitalas and K. Tsichlas. *In Proc. of the 7th International Symposium on Algorithmic Aspects of Cloud Computing ALGOCLOUD*, pp. 42-57, 2022.

[14] **Degree Distribution Optimization in Historical Graphs.** A. Spitalas, Charilaos Kapeletiotis, and K. Tsichlas. *In Proc. of the 9th International Symposium on Algorithmic Aspects of Cloud Computing ALGOCLOUD*, pp. 88-106, 2024.

[15] **Partition Strategies for Vertex-Centric Historical Graph Systems.** A. Spitalas, G. Tsolas and Kostas Tsichlas. *In Proc. of the 40th ACM/SIGAPP Symposium On Applied Computing (SAC)*, pp. 425-43, 2025.

[16] **Triangle Counting in Large Historical Graphs.** K. Christopoulos, E. Daskalakis, A. Bompotas and K. Tsichlas. *In Proc. of the 40th ACM/SIGAPP Symposium On Applied Computing (SAC)*, pp. 432-439, 2025.

[17] **Distributed Community Detection in Temporal Graphs.** K. Christopoulos and K. Tsichlas. *In Proc. of the 19th International Symposium on Spatial and Temporal Data (SSTD)*, 29-33, 2025.