Selective Evolving Centrality in Temporal Heterogeneous Graphs

Landy Andriamampianina
Activus Group
Toulouse, France
landy.andriamampianina@activus-group.fr

Franck Ravat
IRIT-CNRS (UMR 5505) - Université Toulouse Capitole
Toulouse, France
franck.ravat@irit.fr

Jiefu Song
IRIT-CNRS (UMR 5505) - Université Toulouse Capitole
Toulouse, France
jiefu.song@ut-capitole.fr

Nathalie Vallès-Parlangeau
Université de Pau et des Pays de l'Adour, France
Pau, France
nathalie.valles-parlangeau@univ-pau.fr

Yanpei Wang
Université Toulouse Capitole
Toulouse, France
yanpei.wang@ut-capitole.fr

ABSTRACT

Centrality metrics allow quantitatively sorting the nodes of a graph. However, classic centrality metrics are inadequate when dealing with a graph that evolves over time and that contains multiple types of nodes and edges, what is known as *Temporal Heterogeneous Graph*. In this paper, we propose an extension of centrality metrics which takes into account temporal evolution and heterogeneity of a graph. We define a calculation function of centrality that depends on several user-defined parameters (namely, any metric, any graph model, any edge type, any time period). Accordingly, we propose new usages of centrality values. Finally, we propose an algorithm coupled with a decentralized implementation method to perform a centrality analysis based on our calculation function. We carry out a series of experimental assessments to study the efficiency and analytical power of our algorithm. The experimental results have shown that globally, the algorithm's runtimes increase linearly with the datasets' volume and user-defined parameters. Moreover, the algorithm offers multiple perspectives for analyzing nodes' centrality.

1 INTRODUCTION

Graph is a useful structure to model a variety of real-world networks such as social networks, epidemiological networks, transportation networks [7]. Entities, such as people, are modeled as nodes, and relationships connecting entities, such as friendships, are modeled as edges. A typical analysis of graphs is identifying central nodes. The *centrality* of a node refers to the importance of a node in a graph with respect to its position in the graph [25]. Diverse centrality metrics enable to quantify this importance, and to rank nodes accordingly. For example, degree centrality counts the number of incident edges to a node [13]. This can be used to measure the influence of individuals in a social network based on the number of relationships they have with others and to find the most influential ones [25].

Real-world networks are inherently temporal (or dynamic), with their topology evolving over time as nodes and edges are added or removed [7]. In social networks, for example, individuals may gain or lose followers as time progresses [2]. Consequently, the influence of individuals within these networks is not static; it fluctuates over time with changes in connections. Moreover, real-world networks are heterogeneous. A network is homogeneous if the entities and relationships are of the same type. In contrast, it is heterogeneous if entities and relationships have different types [15]. For instance, individuals in a social network may belong to different types, such as influencers and casual users, and may engage in distinct types of interactions, such as follows, mentions, or shares. By leveraging these distinctions, we can differentiate the ranking of influencers and casual users to better capture their varying levels of influence within the network. Besides, we can further refine the analysis by considering how the centrality of entities varies according to the types of interactions they engage in. For example, the centrality of an influencer based on mentions may differ from their centrality based on shares, highlighting their specific roles in content creation versus content amplification. *Temporal heterogeneous graphs* have therefore emerged as models to represent temporal and heterogeneous networks. They refer to a graph that evolves over time, where nodes or edges of different types change dynamically [1, 2, 4].

However, existing centrality metrics are typically designed for either temporal graphs or heterogeneous graphs, and even when applied to the corresponding cases, they have limitations. On the one hand, centrality metrics designed for temporal graphs provide a coarse-grained perspective of the importance of a node. Indeed, they return a single value of a node for a given time interval [5, 6, 11, 17, 27, 28]. This prevents from tracking the variation of the centrality value of a node at each time step of the time interval. On the other hand, centrality metrics designed for heterogeneous graphs capture the multiple types of entities and their relationships by incorporating *meta-paths*, which are sequences of node types and edge types that define a path schema from an origin node type to a destination node type [15, 26]. However, this approach is limited to path-based centrality metrics and cannot be applied to all centrality metrics and use cases. For instance, if we aim to capture a person's influence in a social network using the degree metric, which counts the number of connections, a meta-path would only capture one type of connection, such as *Person-Shares-Person*. This excludes other semantically related types of connections, such as *Likes*, which also impact a person's influence.

A general challenge with centrality metrics is their computational cost. Historically, these metrics were designed for relatively small graphs, typically consisting of hundreds of nodes and edges (Table 1). As a result, they fail to scale efficiently when applied to large-scale networks, which may contain millions or even billions of nodes and edges. This scalability issue becomes especially problematic in temporal heterogeneous graphs, where the complexity of the model adds further computational overhead. In response to this issue, one of the two alternatives is generally taken: (i) approximation algorithms focusing on obtaining an approximation of a centrality metric by using various forms of sampling and heuristics to avoid its full computation [29, 30], (ii) parallel and distributed algorithms that consist of exploiting technical resources (CPUs, GPUs, machines) to speed up the computation [21, 24].

In this paper, we propose a new approach to measure centrality, in terms of both metric calculation and centrality usages, in a graph including temporal evolution and heterogeneity. Precisely, we make the following contributions.

First, we propose a selective centrality function that enables to measure the centrality of an entity given selected inputs: a graph model, a metric, a set of relationship types and a time interval. It has the advantages of (i) returning a list of centrality values to follow the evolution of an entity's importance, (ii) allowing the selection of a set of relationship types in the calculation to capture multiple contributors of an entity's importance and, iii) being generic by enabling the calculation of any metric applied to any graph model.

Second, our function returns a fine-grained perspective of the importance of an entity, contrary to the coarse-grained perspective in existing works. This new perspective opens new usages of the centrality values. Indeed, we propose approaches to use the resulting centrality values of our function. It includes the extension of the classic ranking of entities based on their centrality values.

Third, we propose an algorithm for computing a complete centrality analysis, from the calculation of centrality, using on our selective centrality function, to their results' usage, using our proposed approaches. Moreover, we propose an implementation of our algorithm in a parallel and distributed environment.

Finally, we conduct a systematic study of the application of our algorithm over several metrics, graph models and real-world datasets to evaluate its efficiency and analytical power.

The remainder of the paper is organized as follows. First, we review the literature on centrality in both temporal graphs and heterogeneous graphs (Section 2). Second, we define the selective centrality function of centrality and usage approaches of centrality values (Section 3). Third, we present the systematic study of the application of our solution (Section 4). Finally, we conclude the paper (Section 5).

2 RELATED WORK

We have studied existing works according to two axes: centrality metric calculation and usage of centrality.

2.1 Centrality Calculation

Many researchers have studied the importance of nodes in static graphs, i.e., graphs that do not evolve over time. Among the commonly used centrality metrics, one can cite neighborhood-based centralities such as degree, path-based centralities like betweenness and closeness, and iterative refinement centralities such as eigenvector centrality [13]. Centrality metrics are designed with respect to a specific graph model, meaning their calculation depends on the chosen graph representation. To the best of our knowledge, these metrics have been applied either to temporal graphs or heterogeneous graphs, but not to temporal heterogeneous graphs, which combine both temporal and heterogeneous aspects.

Centrality Metrics for Temporal Graphs. To account for the evolution of graphs, traditional centrality metrics have been extended. These extensions generally rely on two graph models: graph snapshots and temporal property graphs (Table 1).

Graph snapshots represent a sequence of time-ordered static graphs, each capturing the state of the graph at a particular moment. Based on this model, centrality metrics are calculated for each snapshot, and the resulting values are aggregated to produce a single value over a given time interval. For instance, [11] define temporal degree centrality as the sum of the degree values across all snapshots within a specified time interval. Similarly, [5] proposes a weighted temporal degree metric that incorporates weights in snapshot calculations.

A temporal property graph is a single graph where nodes and/or edges carry time-dependent properties. This modeling alternative allows constructing temporal paths (or time-respecting paths) beginning from a node of one instant ending on a node of another instant. This model allows the construction of temporal paths (also called time-respecting paths), which start at a node at one time instant and end at a node at another instant. Temporal property graphs are especially useful in applications such as disease spread modeling or information diffusion, where centrality measures the importance of nodes as bridges for spatio-temporal flows [27]. In this context, betweenness and closeness centralities have been extended by replacing the notion of shortest paths (i.e., the shortest spatial distance between two nodes) with temporal shortest paths (i.e., paths that minimize either the number of hops between two nodes within a specific time interval and/or time elapsed from the start to the end of the path within the given time interval) to return a single value valid for a time interval [5, 6, 17, 27, 28]. Similarly, [20] proposes a temporal PageRank by integrating the concept of time-respecting random walks. [16] defines a temporal H-index of a node based on the temporal reachability of its neighborhood nodes for a certain time interval.

Existing centrality metrics for temporal graphs do not fully exploit the temporal aspects of graphs. They provide a coarse-grained perspective of the importance of a node by returning a single value for a given time interval. Consequently, they fail to capture the evolution of a node's importance over time through a series of values. At best, incremental graph algorithms retain previously computed values to update the centrality score when changes occur in the graph, thereby avoiding recalculations from scratch [8–10]. However, their goal remains to compute a single centrality value for a node rather than to track its evolution over time.

Centrality Metrics for Heterogeneous Graphs. A very limited number of centrality metrics have been designed for heterogeneous graphs, which consist of graphs containing various types of entities and relationships, modeled as labels attached to nodes and edges [2]. To capture heterogeneity, these centrality metrics use the concept of meta-paths. A meta-path is composed of a pre-defined sequence of relationship types $(l_1, l_2, ..., l_k)$ from an origin entity type l_1 to a destination entity type l_k.

In [15], the centrality of an entity is measured using an entropy-based method, which calculates the probability of a node reaching other nodes via a given meta-path. They gave the example of the DBLP bibliographic network[1], where authors are connected through different meta-paths. Their analysis consists in calculating the influence of an author on other authors via separate meta-paths with a fixed length, such as *Author-Conference-Author*.

Similarly, [26] defines a meta-path based betweenness centrality of a node n_i of a type A as the fraction of the shortest paths that follow a meta-path (with no fixed length) having starting and ending nodes of type A and that pass through the node n_i. As an example, in a movie network, given the meta-path *Actor-Movie-Director-Movie-Actor*, we can search for actors who play important "bridge" roles in promoting cooperation between other unfamiliar actors with the help of directors.

The concept of meta-path limits the scope of a centrality analysis. First, it is specifically designed for path-based centrality metrics, ignoring other centrality metrics such as degree. Second, a meta-path's structure imposes a single relationship type between each couple of entity types from the beginning to the end of the path. For instance, in the meta-path *Author-Conference-Author*, we cannot calculate the influence of an author on another by also considering another semantically close relationship type, such as *Journal*. Third, a meta-path's structure imposes a fixed order of connections between entities and, in some cases, a fixed length of these connections.

2.2 Centrality Usage

In classic graphs (i.e., static and homogeneous), nodes are typically ranked after computing a centrality metric. This consists in sorting a list of numerical values, each of which is a centrality value calculated according to a chosen metric. In both temporal graphs and heterogeneous graphs, they apply the same ranking approach [5, 11, 15–17, 26, 27]. The only difference lies in the interpretation of rankings. In the case of temporal graphs, the ranking is for a particular time period [5, 11, 16, 17, 27]. In the case of heterogeneous graphs, the ranking is relative to a specific meta-path [15, 26]. However, these ranking approaches ignore the evolution perspective of centrality in a temporal heterogeneous graph.

3 PROPOSITION

Our proposal covers both the calculation of centrality in temporal heterogeneous graphs and its usage. Firstly, we propose a calculation function of the centrality of an entity. Second, we propose approaches to use the results of our function.

3.1 Preliminaries

Definition 3.1 (temporal heterogeneous graph). A temporal heterogeneous graph $G = \langle E, R, T, \phi \rangle$ is composed by a set of entities E and their relationships R and a timeline T. Entities are modeled as nodes. Relationships between entities are modeled as edges. Each entity and relationship can have several "versions" over time, as they change over time. These versions can be captured through existing temporal graph modelizations: graph snapshots [4] or, by the new concept of "states" in temporal property graphs [2], which associates each entity and relationship with a time interval indicating how long the current version is valid. Additionally, G is associated with an entity type set denoted $L_E = \{l_{E^1}, ..., l_{E^n}\}$ and a relationship type set denoted $L_R = \{l_{R^1}, ..., l_{R^q}\}$. An entity is connected to other entities via several relationship types. The function $\phi : w, e|l_{E^*}, L_{R^*} \rightarrow G'$ returns: (i) for a time interval $w \subseteq T$, a specific entity $e \in E$ and a set of relationship labels L_{R^*}, the subgraph $G' \subset G$ including the entity e and its connected entities via relationships having a type in the set L_{R^*} and present in the time interval w; (ii) for a time interval $w \subseteq T$, an entity

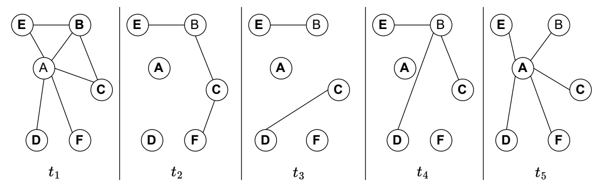
[1]https://dblp.org/

Table 1: Related work on centrality metrics. |N|= number of nodes, |E|=number of edges, |T|= number of time instances.

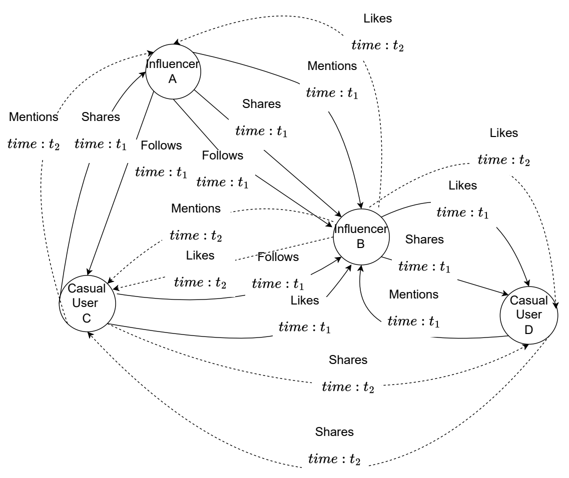
Article	Graph Model	Extended metric	Dataset size
[11]	graph snapshots	degree, closenesss, betweenness	2 contact networks: - |N|: [12,100] - |E|: [~4K,~64K] - |T|: [5,280]
[5]	weighted graph snapshots	degree, closeness	1 social network: - |N|: 100 - |T|: 3
[16]	temporal property graph	H-index	16 multi-domain datasets: - |N|: [75, ~3000K] - |E|: [~30K,~40000K] - |T|: [49,~2K]
[22]	graph snapshots	betweenness, closeness	1 contact network: - |N|: 151 - |E|: ~250K - |T|: ~1K
[17]	temporal property graph	closeness	2 social networks: - |N|:[~1K, ~3K] - |E|: [~5K,~30K] - |T|: [81,120]
[28]	temporal property graph	closeness	12 multi-domain datasets: - |N|: [~8K, ~3200K] - |E|: [~86K,~40000K] - |T|: [70,~730K]
[6]	temporal property graph, graph snapshots	closeness, eigenvector	6 multi-domain datasets: - |N|: [62, ~10K] - |E|: [~27K,~400K] - |T|: [~39,~500]
[23]	graph snapshots embedded in supra-centrality matrix	eigenvector	1 social network: - |N|: 277 - |T|: 65 1 actor network: - |N|: 55 - |T|: 10 1 citation network: - |N|: ~25K - |T|: 20
[20]	temporal property graph	page rank	3 social networks: - |N|: 100 - |E|: ~100K
[15]	heterogeneous graph	entropy	3 citation networks: - |N|: [~ 1K,~400K] - |E|: [~30K,~1800K]
[26]	heterogeneous graph	betweenness	2 movie networks: - |N|: [~ 34K,~1500K] - |E|: [~56K,~2600K] 1 citation network : - |N|: ~ 2600K - |E|: ~5900K 1 restaurant website: - |N|: ~ 9100K - |E|: ~15000K

Table 2: Notation table.

$G = \langle E, R, T, \phi \rangle$	a temporal heterogeneous graph
$G' \subset G$	a subgraph of G
E	the set of entities in G
e	an entity in E
R	the set of relationships in G
T	the timeline of G
$\phi : w, e\|l_{E^*}, L_{R^*} \rightarrow G'$	a function to return a subgraph G'
$L_E = \{l_{E^1}, ..., l_{E^n}\}$	the set of entity types in G
l_{E^*}	a type of entity
$L_R = \{l_{R^1}, ..., l_{R^q}\}$	the set of relationship types in G
l_{R^*}	a type of relationship
m	a centrality metric
$w = [t_{start}, t_{end}]$	a time interval starting at the time t_{start} and ending at the time t_{end}
v_{t_i}	the value of m at the time t_i



(a) Graph snapshots denoted as G_1



(b) an extract of a temporal property graph denoted as G_2
Solid edges represent interactions between entities at t_1.
Dotted edges represent interactions between entities at t_2.

Figure 1: Different types of graph datasets.

label l_{E^*} and a set of relationship label L_{R^*}, the subgraph $G' \subset G$ containing entities of a type l_{E^*} and its connected entities via relationships having a type in the set L_{R^*} and present in the time interval w.

It is worth noticing that no matter how the temporal heterogeneous graph is modeled, our centrality solution is generic enough to adapt to different graph temporality and heterogeneity representations.

3.2 Calculation Function

Current centrality metrics provide a static vision of centrality metric through a single value of an entity for a certain time interval. Moreover, they provide a strict sequence of relationships between entities in the calculation. To do so, we propose a calculation function of an entity's centrality which is characterized by two aspects : (i) its ability to return a dynamic version of a centrality metric to follow changes over time and, (ii) its ability to take into account a graph model, a metric, a time interval and a flexible structure of relationships to adapt to diverse analytical needs. We call this function *selective centrality.*

Definition 3.2 (selective centrality function). The selective centrality function returns the evolution of an entity's importance as a series of values based on several user-defined inputs:

- a temporal heterogeneous graph G;
- the subject of the centrality to compute. It is an entity $e \in G$;
- a centrality metric m to compute;
- a time interval $w = [t_{start}, t_{end}]$ based on the time unit in G;
- an optional relationship label set L_{R^*} to focus on a subset group of relationships $R^* \in G$. It is by default the whole set of relationships $R \in G$.

$$Scent(G, e, m, w, L_{R^*}) = \begin{cases} \bullet \ \{v_{t_{start}}, ..., v_{t_i}, ..., v_{t_{end}}\} \\ \text{where } v_{t_i} \text{ is the centrality value of } e \\ \text{at the time } t_i \in w \\ \bullet \ \text{not defined, otherwise} \end{cases}$$

Example 3.3 (static centrality VS dynamic centrality). Suppose that we want to study the evolution of people's influence based on the degree metric in a social network during the time interval $[t_1, t_5]$. As a reminder, the degree metric consists in counting the number of edges incident to a node [13]. The social network is modeled as graph snapshots presented in Fig 1 (a) and denoted $G_1 = \langle G_{t_1}, G_{t_2}, G_{t_3}, G_{t_4}, G_{t_5} \rangle$ where each $G_t = \langle E_t, R_t \rangle$ is the set of entities and relationships captured at the time $t_i \in [t_1, t_5]$. Changes in the graph are therefore captured by entire graph snapshots created at different time points. Here, G_1 does not distinguish relationship types. However, our selective centrality function also works on temporal homogeneous graphs by skipping the selection of the set of relationship labels. For each person e_i (i.e, A, B, C, D, E and F) in the graph, we compute two versions of its centrality: (i) a static version (i.e., a single value) by aggregating the degree values of each time point of the time interval as in current work, (ii) a dynamic version (i.e, a time series) by using our function, i.e., applying for each entity e_i the operation $Scent(G_1, e_i, degree, [t_1, t_5], \emptyset)$. We obtain the centrality values in Table 3. We observe that if we consider a static version of centrality, A and B have the same degree values on the time period $[t_1, t_5]$. If we make a fine-grained analysis using our selective centrality function, we observe that they have different evolution trends. We observe that the degree of A is 5 at t_1 but stagnates to 0 from t_2 to t_4 and then increases sharply to 5 at t_5. Meanwhile, B has a more stable degree over the period.

Example 3.4 (homogeneous centrality VS heterogeneous centrality). We consider that the heterogeneity of a graph model impacts the centrality of an entity. In the previous example, we consider a social network in the form of graph snapshots that are homogeneous (Fig 1 (a)). They do not include different types of entities and relationships. Now, we consider a social network in the form of a temporal property graph based on [2]. It is presented in Fig 1 (b) and denoted as, $G_2 = \langle E, R, T \rangle$ where E is all entities and R is all relationships over the timeline $T = [t_1, t_2]$. Compared to graph snapshots, changes are captured by using a single graph where a new version of each relationship is created whenever a change occurs. This avoids repeating edges that do not change, as it would introduce redundancy. To do so, this temporal property graph attaches properties to edges to depict the time occurrence of interactions. Moreover, it integrates heterogeneity using labels attached to nodes and edges to distinguish entity and relationship types. Here, we have two types of entities, *Influencer* and *Casual User* and four types of relationships, *Shares, Likes, Mentions* and *Follows.* Suppose that we want to compute the degree of the individual B, i.e., the number of received likes and/or shares and/or mentions and/or followers, in t_1 to show its influence in the social network. Using our selective centrality function, we have

Table 3: Static centrality VS dynamic centrality for the degree metric in the graph G_1 in Fig 1 (a). The static degree centrality is the sum of degree values calculated at each time point t_i of the time interval $[t_1, t_5]$. The dynamic degree centrality is calculated with the selective centrality function.

Entity	Degree	
	Static version	Dynamic version
A	10	{5,0,0,0,5}
B	10	{3,2,1,3,1}
C	7	{2,2,1,1,1}
D	4	{1,0,1,1,1}
E	6	{2,1,1,1,1}
F	3	{1,1,0,0,1}

several perspectives of the influence of B in the social network. We can obtain a homogeneous version of the degree of B without distinguishing relationship type: $Scent(G_1, B, degree, t_1, \emptyset) = 5$. We can obtain several heterogeneous versions of the degree of B, i.e., with a selected subset of relationship types. If we want to measure the influence of B only based on the number of its followers, we have: $Scent(G_1, B, degree, t_1, \{Follows\}) = 2$. It is also possible to measure the influence of B with respect to his posts' visibility through shares, likes and mentions: $Scent(G_1, B, degree, t_1, \{Shares; Likes; Mentions\}) = 4$.

Example 3.5 (different metrics). Our selective centrality function is independent of the metric. Indeed, it can be applied for different metrics. In Table 4, we compute the degree-, eigenvector- and betweenness- centrality for each individual in the graph G_2 in Fig 1 (b) for the relationship types *Shares, Likes, and Mentions* using our selective centrality function. The goal is to compare their influence based on the *Shares, Likes,* and *Mentions* they received and the metric used. The details of computations are presented on the website https://gitlab.com/2573869/centrality_temporal_graph_edbt. As a reminder, eigenvector centrality computes for each node a score (between 0 and 1) based on the sum of its neighbors' centrality values [13]. In other words, the influence of an individual depends on the influence of the individuals with whom he interacts (through likes, mentions and shares). Betweenness centrality of a node is calculated as the sum of the ratios of all shortest paths that pass through the node to the total number of shortest paths in the graph [13]. Accordingly, betweenness centrality looks at the influence of an individual as a bridge between other individuals. Meanwhile, the degree centrality here refers to the number of likes, mentions and shares received by an individual. We notice that in t_1 even if the individual C is not influential according to its degree value, he is one of the most influential person according to the eigenvector score. Moreover, we observe that B is the only one to have a non-null betweenness centrality in t_1 because he connects indirectly other individuals. It is the same for C in t_2.

In previous examples, we show that our selective centrality function enables to choose interchangeable parameters, namely a graph model, a metric, a time interval and a set of relationship types. This is in the objective to offer multi-perspectives of the importance of an entity: its evolution and a different meaning according to the chosen metric and the relationship types on which it relies. All these different perspectives imply to rank entities differently as well.

Table 4: Application of different metrics on the graph G_2 in Fig 1 (b) where $L_{R^*} = \{Shares; Likes; Mentions\}$.

Operation	Centrality		
	Degree	Eigenvector	Betweenness
$Scent(G_2, A, m, [t_1, t_2], L_{R^*})$	{1,2}	{0.19,0}	{0,0}
$Scent(G_2, B, m, [t_1, t_2], L_{R^*})$	{4,0}	{0.31,0.5}	{2,0}
$Scent(G_2, C, m, [t_1, t_2], L_{R^*})$	{0,3}	{0.31,0.25}	{0,1}
$Scent(G_2, D, m, [t_1, t_2], L_{R^*})$	{2,2}	{0.19,0.25}	{0,0}

3.3 Usage

Traditionally, after computing the centrality metrics for all entities in the graph, the resulting list of numerical values are sorted to produce a rank to the user to identify the most important entities. However, this ranking approach is not sufficient if the user takes account for the evolution and heterogeneity perspectives proposed by our selective centrality. Indeed, with this new perspectives, multiple lists of centrality values (one for each time step of a given interval) are produced. Aggregating these multiple centrality lists into a single list disregards the evolution and heterogeneity of the entities. To address these challenges, we propose three approaches for utilizing centrality values: (i) global centrality ranking, (ii) dynamic centrality ranking and (iii) centrality spreading.

3.3.1 Global Centrality Ranking. A user has a need to rank entities based on their centrality, typically using traditional ranking approaches. However, these traditional approaches may fail to provide insights into long-term trends, particularly when short-term fluctuations dominate. Our approach addresses this need by providing a global view of centrality that reflects the overall importance of entities over a given time interval. To do so, we define an average selective centrality function to rank entities based on a single centrality value, calculated over a selected time. The goal is to simplify the analysis for non-expert users, who may be more familiar with traditional ranking, and to offer decision-makers a quick way to assess the most important entities. The approach proceeds in three steps: (i) compute the selective centrality of each entity considering a centrality metric m, a selected group of entities given the type l_{E^*} (by default all entities E), a selected set of relationship types given the set of relationship labels L_{R^*} (by default, all relationships R), (ii) compute the average of these centrality values over the time interval, and (iii) rank entities in descending order of their average centrality. The average selective centrality is denoted as follows:

$$Scent_{avg}(G, e, m, w, L_{R^*}) = \frac{\sum_{t_i \in w} Scent(G, e, m, t_i, L_{R^*})}{\sum_{i=1}^{|w|} i}$$

This provides a way to calculate a ranking based on the average selective centrality for the selected entities, metric, relationship types, and time period:

$$Scent_{rank,avg}(G, l_{E^*}, m, w, L_{R^*}) = (e_1, e_2, ..., e_k \mid e_1, e_2, ..., e_k \in E^* \text{ and } Scent_{avg}(G, e_1, m, t_i, L_{R^*}) \geq ... \geq Scent_{avg}(G, e_k, m, t_i, L_{R^*}))$$

Example 3.6. Suppose the user has the graph G_2 in Fig 1 (b), and wants to rank entities using the degree metric over the time interval $[t_1, t_5]$ and the relationship types $\{Shares; Likes; Mentions\}$. Using the traditional ranking approach, the classic degree for each entity is computed for

Table 5: Centrality for entities in the graph G_2 (Fig 1 (b)), the degree metric, the time period $[t_1, t_5]$ and the set of relationship types $\{Shares; Likes; Mentions\}$.

Entity	Selective Degree	Classic Degree	Average Selective Degree
A	{1,2,6,1,7}	17	3.4
B	{4,0,1,1,15}	21	4.2
C	{0,3,1,2,0}	6	1.2
D	{2,2,2,2,4}	12	2.4

each time step, and a sum is obtained for the entire interval (Table 5). In contrast, in the global centrality ranking approach, the selective degree values are computed for each entity and then the average degree for each entity is computed over the time period $[t_1, t_5]$ (Table 5). The classic degree and the average selective degree rankings are similar, but the average degree provides a more balanced view, avoiding the dominance of entities that only appear influential during certain time windows. The resulting ranking for all entities is: $Scent_{rank,avg}(G_2, \emptyset, degree, [t_1, t_5], \{Shares; Likes; Mentions\}) = (B, A, D, C)$. In addition, we can refine the ranking by entity type. For example, ranking just the "Influencers" versus "Casual Users" might yield: $Scent_{rank,avg}(G_2, Influencer, degree, [t_1, t_5], \{Shares; Likes; Mentions\}) = (B, A)$ and $Scent_{rank,avg}(G_2, Casual\,User, degree, [t_1, t_5], \{Shares; Likes; Mentions\}) = (D, C)$.

3.3.2 *Dynamic Centrality Ranking.*

A user wants to track how the importance of entities evolves over time. Traditional ranking approaches fail to highlight key moments when entities become more or less influential. By using dynamic ranking, we offer an approach to track rankings at each time step, providing insights into temporal shifts in centrality. This approach ranks entities at each time step of a time interval w in three steps: (i) compute the selective centrality of each entity considering a centrality metric m, a selected group of entities given the type l_{E^*} (by default all entities E), a selected set of relationship types given the set of relationship labels L_{R^*} (by default, all relationships R), (ii) for each time step in w, create a list grouping entities and their centrality values, (iii) in each list, sort entities by descending order of their centrality values. The dynamic ranking is represented as follows:

$$Scent_{rank}(G, l_{E^*}, m, w, L_{R^*}) = \{rank_{t_{start}}, ..., rank_{t_i}, ..., rank_{t_{end}}\}$$
$$\text{where } rank_{t_i} = (e_1, e_2, ..., e_k \mid e_1, e_2, ..., e_k \in E^*$$
$$\text{and } Scent(G, e_1, m, t_i, [L_{R^*}]) \geq ... \geq Scent(G, e_k, m, t_i, [L_{R^*}]))$$

Example 3.7. Suppose the user has the graph G_2 in Fig 1 (b), and wants to analyze the evolution of the ranking of entities using the degree metric over the time interval $[t_1, t_5]$ and the relationship types $\{Shares; Likes; Mentions\}$. Using the traditional ranking approach, we obtain a single degree value for each entity which prevents from tracking the evolution of entity rankings (Table 5). Using the dynamic ranking approach, we obtain: $Scent_{rank}(G_2, \emptyset, degree, [t_1, t_5], \{Shares; Likes; Mentions\}) = \{(B, D, A, C), (C, A - D, B), (A, D, B - C), (B, A, D, C)\}$. This reveals the fluctuations in entity rankings across time, such as entity B being top-central individual at t_1 but less influential at t_2 and t_3, before regaining prominence by t_5.

3.3.3 *Centrality Spreading.*

A user wants to better understand the magnitude of differences in centrality among entities within a ranking. While rankings show the order of entities, they do not reveal the spread of centrality values. We propose an approach to analyze how centrality values are distributed among entities. This approach involves two scenarios: calculating the centrality values for each entity, using either the selective centrality or the average selective centrality. Then, based on the previous computed values, the following aggregate statistics are computed: the minimum, the maximum, the first quartile[2], the second quartile[3] and the thrid quartile[4].

In the first scenario, the approach works as follows: (i) calculate the selective centrality of each entity considering a centrality metric m, a selected group of entities given the type l_{E^*} (by default all entities E), a selected set of relationship types given the set of relationship labels L_{R^*} (by default, all relationships R), (ii) for each time step in w, create a list grouping entities and their centrality values, (iii) for each list, apply the minimum, maximum and first quartile, second quartile and third quartile functions. In the second scenario, we have to : (i) calculate the average selective centrality for each entity considering some selected parameters, (ii) apply the minimum, maximum and first quartile, second quartile and third quartile functions on the average selective centrality values. The corresponding functions are:

$$Scent_{min}(G, l_{E^*}, m, w, L_{R^*}) = \begin{cases} \bullet \{min_{t_{start}}, ..., min_{t_i}, ...min_{t_{end}}\} \\ \text{ s.t } min_{t_i} = min(\cup_{e_j \in E^*} Scent(G, e_j, m, t_i, L_{R^*})) \\ \bullet\, min(\cup_{e_j \in E^*} Scent_{avg}(G, e_j, m, w, L_{R^*})) \end{cases}$$

$$Scent_{max}(G, l_{E^*}, m, w, L_{R^*}) = \begin{cases} \bullet \{max_{t_{start}}, ..., max_{t_i}, ..., max_{t_{end}}\} \\ \text{ s.t } max_{t_i} = max(\cup_{e_j \in E^*} Scent(G, e_j, m, t_i, L_{R^*})) \\ \bullet\, max(\cup_{e_j \in E^*} Scent_{avg}(G, e_j, w, t_i, L_{R^*})) \end{cases}$$

$$Scent_{q1}(G, l_{E^*}, m, w, L_{R^*}) = \begin{cases} \bullet \{q1_{t_{start}}, ..., q1_{t_i}, ..., q1_{t_{end}}\} \\ \text{ s.t } q1_{t_i} = q1(\cup_{e_j \in E^*} Scent(G, e_j, m, t_i, L_{R^*})) \\ \bullet\, q1(\cup_{e_j \in E^*} Scent_{avg}(G, e_j, m, w, L_{R^*})) \end{cases}$$

$$Scent_{q2}(G, l_{E^*}, m, w, L_{R^*}) = \begin{cases} \bullet \{q2_{t_{start}}, ..., q2_{t_i}, ..., q2t_{end}\} \\ \text{ s.t } q2_{t_i} = q2(\cup_{e_j \in E^*} Scent(G, e_j, m, t_i, L_{R^*})) \\ \bullet\, q2(\cup_{e_j \in E^*} Scent_{avg}(G, e_j, m, w, L_{R^*})) \end{cases}$$

$$Scent_{q3}(G, l_{E^*}, m, w, [L_{R^*}]) = \begin{cases} \bullet \{q3_{t_{start}}, ..., q3_{t_i}, ..., q3_{t_{end}}\} \\ \text{ s.t } q3_{t_i} = q3(\cup_{e_j \in E^*} Scent(G, e_j, m, t_i, L_{R^*})) \\ \bullet\, q3(\cup_{e_j \in E^*} Scent_{avg}(G, e_j, m, t_i, L_{R^*})) \end{cases}$$

Example 3.8. Suppose the user has the graph G_2 in Fig 1 (b), and wants to analyze the evolution of the spreading of "Influencers" and "Casual Users" based on the degree metric over the time interval $[t_1, t_2]$ and the set of relationship types $\{Shares; Mentions; Likes\}$. To do so, we apply the first scenario of the approach. Using boxplots[5] to visualize the results (Fig 2), we find that degree values for "Casual Users" are more concentrated at lower values, while "Influencers" exhibit a greater spread. For example, 75% of casual users have fewer than 6 interactions over the period (q3), while 75% of influencers have more than 8

[2]The first quartile marks the 25th percentile, meaning 25% of the data points fall below this value.

[3]The second quartile marks the 50th percentile, dividing a dataset in half.

[4]The third quartile represents the 75th percentile, meaning 75% of the data points fall below this value.

[5]In a box plot, the minimum is at the end of the lower whisker. The first quartile q1 is at the bottom of the box. The median q2 is the line inside the box. The third quartile q3 is at the top of the box. The maximum is at the end of the upper whisker.

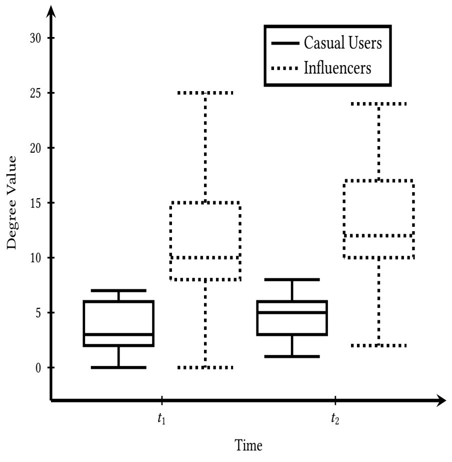


Figure 2: Evolution of box plots of degree values of entities in G_2.

interactions (q1). Moreover, the top 25% casual users have degree values than close to the rest of entities. Conversely, the top 25% influencers have degree values significantly higher than the rest of the group.

3.4 Algorithm

In this section, we propose an algorithm, independent of a technical environment, called SelectiveCentralityAnalysis (or SCA) to execute a complete analysis of the centrality of a group of entities in a graph, i.e., from the computation of entities' centrality to their rankings and spreadings (Algorithm 1). It takes as inputs a graph, an optional entity group defined by a type, a metric, a time interval and an optional relationship type set. Then, it implements the propositions of the previous sections to return several outputs: the evolving centrality of entities, the evolving ranking of entities, the evolving spreading of entities, the average centrality of entities, the average ranking of entities, and the average spreading of entities.

First, the algorithm calculates all the centrality values of a selected group of entities of the type l_{E^*} (by default E) by implementing the selective centrality function. For each time step t within the chosen time interval w, the algorithm does in parallel the following sequential tasks (lines 5-13): (i) it selects the sub-graph that includes entities from the chosen type l_{E^*} and relationships from the chosen relationship type set L_{R^*} having timestamps intersecting the time step t (line 7), (ii) for each entity in the selected subgraph, it does in parallel the calculation of the chosen metric m, and adds the triple (entity, metric value and time step) in a set of entities' centrality SC_t for the time step $t \in w$ (lines 8-11).

Second, the algorithm computes the ranking of entities for each time step in w by implementing a selective centrality ranking. To do so, for each set of entities' centrality SC_t calculated at the previous step, it sorts entities of the set by the descending order of their centrality values (lines 14-18).

Third, the algorithm computes the spread of entities for each time step in w. More precisely, for each set of entities' centrality SC_t calculated at the previous step, it applies the minimum, maximum, and quartiles functions and records the resulting values in the set $SC_{spread,t}$ (lines 19-23).

Algorithm 1 SelectiveCentralityAnalysis (SCA)

```
1: Input: a temporal heterogeneous graph $G$, an optional entity type $l_{E^*}$, a metric $m$, a time interval $w$, an optional relationship type set $L_{R^*}$
2: Output: the selective centrality values of entities $SC$, the dynamic ranking of entities $SC_{rank}$, the dynamic spreading of entities $SC_{spread}$, the average selective centrality values of entities $SC_{avg}$, the average ranking of entities $SC_{rank,avg}$, the average spreading of entities $SC_{spread,avg}$
3: Initialize $SC \leftarrow \{\}, SC_{rank} \leftarrow \{\}, SC_{spread} \leftarrow \{\}$
4: Initialize $SC_{avg} \leftarrow \{\}, SC_{rank,avg} \leftarrow \{\}, SC_{spread,avg} \leftarrow \{\}$
5: Selective Centrality Calculation:
6: for each $t$ in $w$ do in parallel
7:     $G_t \leftarrow \phi(t, l_{E^*}, L_{R^*})$
8:     for each $e$ in $G_t$ do in parallel
9:         $v \leftarrow m(G_t, e)$
10:        $SC_t \leftarrow SC_t \cup \{(e, v, t)\}$
11:    end for
12:    $SC \leftarrow SC \cup \{SC_t\}$
13: end for
14: Selective Centrality Ranking:
15: for each $SC_t \in SC$ do
16:     $SC_{rank,t} \leftarrow sort(SC_t, v, desc)$
17:     $SC_{rank} \leftarrow SC_{rank} \cup SC_{rank,t}$
18: end for
19: Selective Centrality Spreading:
20: for each $SC_t \in SC$ do
21:     $SC_{spread,t} \leftarrow \{min(v \mid (e,v,t) \in SC_t), Q_1(v \mid (e,v,t) \in SC_t), Q_2(v \mid (e,v,t) \in SC_t), Q_3(v \mid (e,v,t) \in SC_t), max(v \mid (e,v,t) \in SC_t)\}$
22:     $SC_{spread} \leftarrow SC_{spread} \cup SC_{spread,t}$
23: end for
24: Average Selective Centrality Calculation:
25: for each $e \in SC$ do
26:     $SC_{avg} \leftarrow SC_{avg} \cup \{(e, avg(\{\cup_{(e,v_i,t) \in SC_t \in SC} v_i\}))\}$
27: end for
28: Average Selective Centrality Ranking:
29: $SC_{rank,avg} \leftarrow sort(SC_{avg}, v, desc)$
30: Average Selective Centrality Spreading:
31: $SC_{spread,avg} \leftarrow \{min(v \mid (e,v) \in SC_{avg}), Q_1(v \mid (e,v) \in SC_{avg}), Q_2(v \mid (e,v) \in SC_{avg}), Q_3(v \mid (e,v) \in SC_{avg}), max(v \mid (e,v) \in SC_{avg})\}$
32: return $SC, SC_{rank}, SC_{spread}, SC_{avg}, SC_{rank,avg}, SC_{spread,avg}$
```

Fourth, the algorithm calculates the average centrality values of the entity group for the time interval w by implementing the average selective centrality (lines 24-27). To do so, for each entity, it applies the average function on all metric values over all time steps of w. Then, it adds the double (entity, average metric value) in a set of entities' average centrality SC_{avg}.

Fifth, the algorithm computes the ranking of entities based on their average centrality values for the time interval w. To do so, it sorts the entities of the set SC_{avg} calculated in the previous step by the descending order of their average centrality values (lines 28-29).

Finally, the algorithm computes the spread of the entities based on their average centrality values for the time interval w. More precisely, it applies the minimum, maximum, and quartile functions to the set of entities' average centrality SC_{avg} calculated at the previous step and records the resulting values in the set $SC_{spread,avg}$ (lines 30-31).

3.5 Algorithm's Complexity

The time and space complexity of the algorithm is primarily determined by the Selective Centrality Calculation (lines 5-13). As a result, the complexity discussion in this section will primarily focus on the Selective Centrality Calculation. Since various centrality metrics can be integrated into our algorithm, this complexity varies depending on the specific metric used.

The average-case time complexity of the algorithm is:

$$O\left(|w| \cdot (|E| + |R| + TC_m(|\bar{e}|, |\bar{r}|)\right),$$

The average-case space complexity of the algorithm is:

$$O\left(|\bar{e}| + |\bar{r}| + SC_m\left(|\bar{e}|, |\bar{r}|\right) + |w| \cdot |E|\right),$$

where:

- $|w|$: length of the selected time interval,
- $|E|$: number of entities in the graph,
- $|R|$: number of relationships in the graph,
- $|\bar{e}|$: average number of entities in subgraphs per time step,
- $|\bar{r}|$: average number of relationships in subgraphs per time step,
- TC_m: time complexity of the selected metric calculation,
- SC_m: space complexity of the selected metric calculation.

4 EXPERIMENTAL EVALUATION

We propose the Algorithm 1 to make a complete analysis of centrality in a temporal heterogeneous graph. In this section, we conducted a series of experiments that:

- assess the compatibility of our algorithm on different temporal graph models (Section 4.2);
- analyze the scalability of our algorithm across datasets of different volumes and parameters (centrality metric, relationship type set and time interval) (Section 4.3);
- highlight the analytical performance of our algorithm across evolution and heterogeneity aspects (Section 4.4).

4.1 Experimental Setup

4.1.1 Datasets. To avoid any bias in our experiments, we have tested our algorithm on six real datasets. They include both temporal homogeneous graphs and temporal heterogeneous graphs. The detailed information of these datasets is provided in the Table 6. All experiments with the DBLP dataset resulted in a timeout, which was set to one day of computation. Therefore, no results could be obtained and presented in the paper.

4.1.2 Technical Environment. Our experiments were conducted in a decentralized environment, in accordance with a parallel and distribution computation approach. The decentralized environment consists of a cluster SLURM (Simple Linux Utility for Resource Management) installed on the OSIRIM[6] plateform (Open Services for Indexing and Research Information in Multimedia contents) provided by IRIT (Computer Science Research Institute of Toulouse). It consists of a storage area with a capacity of approximately 1PB and a computing cluster of 928 cores and 31 GPUs. Our experiments were conducted on the partition named "24CPUNodes" which consists of 12 computing nodes with dual Intel Xeon Gold 6136 processors at 3 GHz, with 24 processors and 192 GB of RAM each. We installed for each computing node one Neo4j graph database, thus they could work on the same dataset in parallel. The goal is to optimize the computation of our algorithm by distributing a part of its processing over the 12 computing nodes and parallelizing another part over the 24 processors of each node.

4.1.3 Implementation of the algorithm. In this section, we present the implementation of the Algorithm 1 based on the decentralized environment presented in Section 4.1.2. The programming language used to implement the algorithm is Python 3.7. We use the library Neo4j Python Driver[7] to work with Neo4j database in Python. The algorithm is implemented through tasks that are distributed over the cluster's nodes and parallelized over the processors of each node:

- **Task 1** (line 1) In each node, the input temporal heterogeneous graph dataset G is stored in a Neo4j database. Mapping details of these datasets into Neo4j are available at: https://gitlab.com/2573869/centrality_temporal_graph_edbt.
- **Task 2** (line 5-13) Each time instance $t \in w$ is assigned to a node of the cluster. A query in CYPHER (Neo4j's query language) is executed to extract a sub-graph from the Neo4j database according to the optional input parameters: $MATCH\,(e : l_{E^*}) - (r : L_{R^*}) - ()\,WHERE\,r.time = t\,RETURN e.id, r$ where $L_{R^*} = l_1|...|l_k$. If l_{E^*} is not specified, we will have (e) as the first element in the $MATCH$ clause referring to all entities. If L_{R^*} is not specified, we will have (r) as the second element of the $MATCH$ clause referring to all relationships. Then, the calculation of the metric m for each entity of the sub-graph is done in parallel. Using Python, the result is recorded in the set SC_t and entities of the set are sorted by the descending order of their centrality values to obtain the ranking of entities for the time t.
- **Task 3** (line 19-32) Finally, the results of each node are collected on a single node. Using Python, for each set SC_t obtained in the previous step, we apply the minimum, maximum, first quartile, second quartile and third quartile functions to obtain the evolution of the spreading of entities' centrality. Then, for each entity across the sets SC_t, we apply the average of its centrality values and sort entities of the set by the descending order of their average centrality values to obtain an average ranking of entities. Finally, the minimum, maximum, first quartile, second quartile and third quartile functions are applied to the average centrality values of entities to obtain an average centrality spreading of entities.

We propose to implement our algorithm according to three different centrality metrics, one which is neighborhood-based (degree), one which is path-based (betweenness) and another which is iterative refinement-based (eigenvector). The algorithm's implementation differs with the chosen metric since their calculation are different in the Task 2:

- **Degree centrality:** The calculation of the degree centrality for an entity at $t \in w$ is translated by the addition of the function $COUNT$ in the $RETURN$ clause of the query for extracting the sub-graph of interest. Here, the function counts the number of edges incident to an entity node at the time t.

[6] https://osirim.irit.fr/docs/slurm/

[7] https://neo4j.com/docs/api/python-driver/current/

Table 6: Dataset characteristics. Raw datasets are available here: (a) http://realitycommons.media.mit.edu/socialevolution.html, (b) https://www.kaggle.com/datasets/retailrocket/ecommerce-dataset, (c) https://www.citibikenyc.com/system-data, (d) https://snap.stanford.edu/data/sx-mathoverflow.html, (e) https://snap.stanford.edu/data/wiki-talk-temporal.html, (f) http://konect.cc/networks/dblp_coauthor/.

Dataset	Social Experiment (a)	E-commerce (b)	Citibike (c)	Math Overflow (d)	wiki-talk (e)	[DBLP (f)]
Nodes	65	4 315 735	2 861	24 818	1 140 149	1 824 701
Edges	2 168 270	4 447 430	2 181 077	506 550	7 833 140	29 487 744
Nb days	106	130	59	2 350	2 320	29 219
Nb of node types	1	3	1	1	1	1
Nb of edge types	9	5	1	3	1	1

- **Eigenvector centrality:** In Neo4j, for each time instance in w, we repeatedly query, for each entity identifier, its direct neighbors. With the result extracted, we create an adjacency matrix. Then, we use the Python library scipy[8] to compute eigenvectors and choose the one with the highest eigenvalue.
- **Betweenness centrality:** In Neo4j, for each time instance, we repeatedly query for each entity all shortest paths starting from it. For each path, start nodes, end nodes and all traversed nodes would be noted. With the result extracted, the betweenness of each time instance is calculated on Python using the Brandes algorithm [3].

4.2 Impact of Graph Models

In this analysis, we demonstrate that our algorithm can be applied to multiple graph models, including the graph snapshots [4] and the temporal property graph [2]. This experiment was executed on the Social Experiment dataset modeled in the two graph models. Transformation details of this dataset into the graph snapshots model and the temporal property graph model are available on the website https://gitlab.com/2573869/centrality_temporal_graph_edbt. For each centrality metric (degree, eigenvector, and betweenness), we capture the CPU time from the moment when the program takes the raw temporal heterogeneous graph as input until the end of execution when analysis results are produced. We measure this time throughout our experiments.

In Fig 3, we observed that the algorithm can be applied to both models. Additionally, execution times between the two graph models are almost the same, except for the betweenness centrality. The temporal property graph model allows reducing execution times of graph snapshots by 20 times for the betweenness centrality. This is partly due to the amount of data processed by the algorithm. The temporal property graph model reduces significantly data redundancy generated by graph snapshots [2].

4.3 Scalability

We executed a series of experiments to evaluate the efficiency of our algorithm and to explore how dataset nature and parameter setting impact the execution time.

We run the algorithm in the decentralized environment by progressively integrating random edges in the six datasets at the following percentages: 20%, 40%, 60%, 80%, and 100%. We did not define scale factors with respect to node volume because the variation in the number of nodes impacts the variation in the number of edges. The six datasets were modeled as temporal

[8]https://scipy.org/

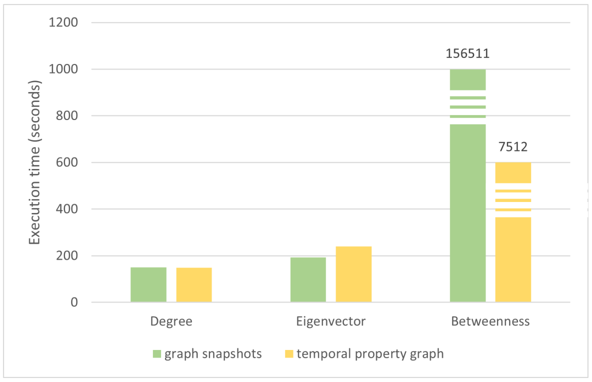
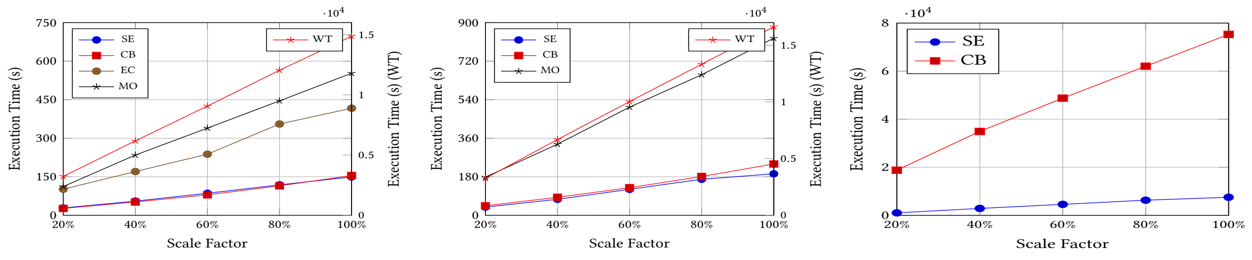


Figure 3: Application of selective centrality on different graph models.

property graphs. Transformation details are available on the website https://gitlab.com/2573869/centrality_temporal_graph_edbt. Experimentation over 24 hours was regarded as "timeout". In Figure 4, we compare the algorithm's runtimes for the different graph sizes. Due to resource limitations, the degree centrality timed out on DBLP dataset, the eigenvector centrality calculation timed out on the E-commerce and DBLP datasets, and the betweenness centrality calculation timed out on E-commerce, Math Overflow, Wiki-Talk, and DBLP datasets. We observe that, for each dataset, the algorithm's runtimes generally increases linearly with the scale factors.

Referring to parameter setting, a semantic or temporal selection can also impact the execution time of the algorithm. A semantic selection enables the inclusion of only a specific subset of relationships in centrality calculations, rather than using the entire graph. This approach helps to reduce the scale on which the subsequent processes operate. We run our algorithm with degree centrality on the Social Experiment dataset, using a subset of relationship types. To highlight the differences between execution times, this experiment was conducted in a single computation node. As shown in Fig 5, the execution time increases linearly with the number of relationships of the selected set of types.

A temporal selection corresponds to a time interval on which centrality calculations are done. The length of the time interval affects execution time of the algorithm, as each time instant of the time interval requires the selection of a corresponding subgraph. Thus, we executed our algorithm with degree centrality on the Math Overflow dataset with randomly extracted 200-days, 400-days, 600-days, 800-days, and 1000 days, each 10 times. In



a) Degree centrality **b) Eigenvector centrality** **c) Betweenness centrality**

Figure 4: Runtimes over graph sizes. (*SE=Social Experiment, CB=Citibike, EC=E-commerce, MO=Math Overflow, WT=wiki-talk. The right y-axis is for wiki-talk dataset and left y-axis for other datasets.*)

Fig 6, we observe that the average execution time for each time interval length, and they are roughly linear.

To sum up, semantic selection reduces the graph scale on which centrality metric operates, while temporal selection diminishes the length of time interval in the Selective Centrality Calculation. Therefore, both of them can impact significantly the execution time of our algorithm.

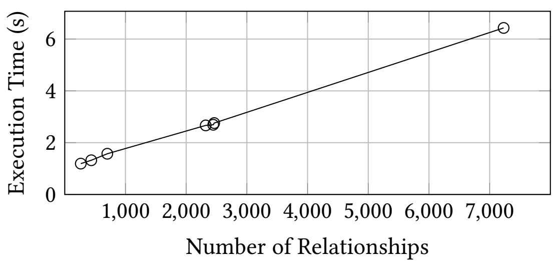


Figure 5: Runtimes with semantic selection.

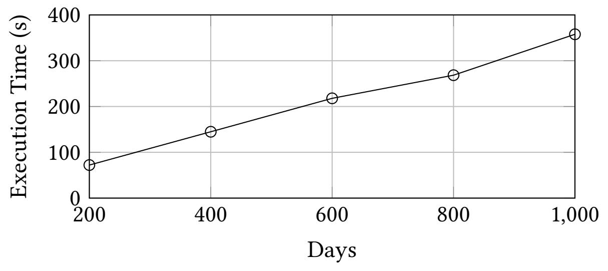


Figure 6: Runtimes with temporal selection.

4.4 Analytical Performance

In this section, we demonstrate the analytical power of our algorithm by presenting the use cases where the traditional metrics do not work well.

4.4.1 Evolution. Traditionally, the temporal centrality analysis of entities consists of (i) calculating a centrality metric of an entity, which returns an aggregated value for a time interval and (ii) producing a ranking of all entities based on previous calculated values. Our algorithm expands the temporal centrality analysis of entities by (i) calculating the centrality value of an entity at each time point of a time interval to produce a list of values, as well as (ii) producing the evolution of the ranking and spreading of a selected group of entities over time. This extension provides a complete perspective of the centrality of entities within a graph with evolution.

In Fig 7, we present the evolution of degree centrality within a time interval of 106 days for 3 students in the Social Experiment dataset having closed average degree values. We notice that these students behave quite differently. Some students become more or less active together (such as the students 67 and 33); for others, their activity varies individually (such as the student 26). More precisely, we notice that the student 33 seems to be relatively evenly active day by day, while the student 26 is normally silent but with several activity picks. If using traditional metrics that only return one aggregated value, these students will be considered similar while all these mentioned details will be hidden.

Additionally, our algorithm can help to identify variations at individual and group level. In Fig 8, we present the evolution of the spreading of degree centrality values within a time interval of 15 days in the Social Experiment dataset using box plots. The dots outside the box plots represent outliers, i.e., the top-students having very high degree values compared to the rest of students. In each box plot, a blue dot represents the degree value of a specific student at each day. Thus, the blue line represents the variation of his position within the group of students over time.

4.4.2 Selectivity. Real-world graph datasets contain semantically rich information with multiple types of entities and relationships. Thus, on the same graph, diverse centrality analyses can be expressed. According to the analytical need, some relationship (or entity) types may be more relevant than another. Traditional centrality metrics do not distinguish between these types. They include all relationship (or entity) types in the calculation, even if they may be semantically unrelated to each other. Our algorithm enables to do a selection of the relationship (or entity) types identified as relevant for an analytical need. In the following, we make an experiment to show the potential gap between a non-selective centrality metric (i.e., available in traditional metrics) and our selective centrality metric.

Using our algorithm, we compute for all entities in the Social Experiment and E-commerce datasets of a given time interval (i) the average degree centrality for each type of relationship and (ii) the average degree centrality without distinguishing relationship types. Then, we produce (i) selective rankings (rankings considering a selected relationship type) and (ii) a non-selective ranking (a ranking considering all relationship types). We use the Spearman's rank correlation coefficient to evaluate the similarity

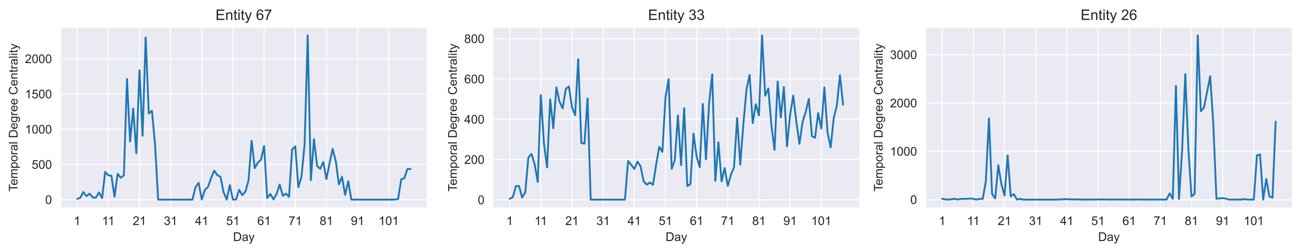


Figure 7: Evolution of degree centrality evolution over time for selected entities.

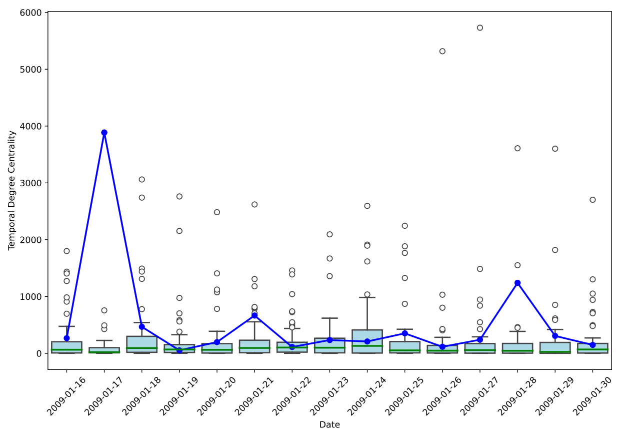


Figure 8: Evolution of degree centrality spreading

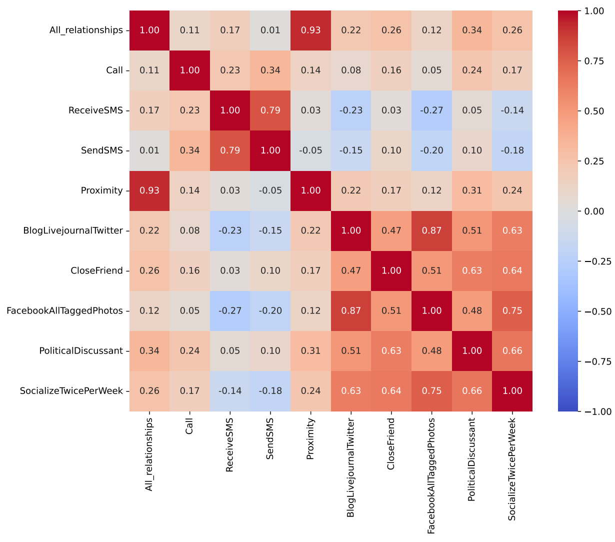


Figure 9: Correlation of average degree centrality rankings by relationship types

between two rankings. If it is close to 1, it means that both rankings are similar. If it is close to -1, it means that both rankings are dissimilar. We illustrate the results of this calculation in the form of a correlation matrix in Fig 9. Each label of a row and column corresponds to a type of relationship. Each cell of the matrix corresponds to a correlation coefficient between two rankings of two different relationship types. The more a cell tends to the red color, the more the corresponding pair of rankings of two relationship types are similar.

Table 7: Kendall Tau of centrality ranking by different types of metrics for two use cases.

Metric	Analysis 1	Analysis 2
Static Centrality	0.07	0.31
Non-selective Temporal Centrality	0.33	0.17
Temporal Centrality with meta-path	-	1
Selective Evolving Centrality	1	-

We observe that the non-selective ranking (*All_relationships*) is highly correlated with the ranking of one type of relationships, *Proximity*[9], with a correlation coefficient of 0.93. Conversely, all other selective rankings are not similar to the non-selective ranking. Indeed, selective rankings themselves are not necessarily similar to one another. We have similar observations for E-commerce dataset. The details are given on the website https://gitlab.com/2573869/centrality_temporal_graph_edbt.

In Table 7. we show how existing metrics, namely static centrality, non-selective temporal centrality, temporal centrality with meta-path, and our selective centrality cover two analytical needs. The first analysis is on the Social Experiment dataset and asks for the importance of these students in online social networks in the form of eigenvector centrality. The second analysis looks at Math Overflow data over a given time interval, to determine which response triggered discussion. For the first analysis, we can use our selective evolving centrality with a selection on relationship types concerning communication on online social networks. For the second, we can use our algorithm implemented with betweenness centrality with the meta-path: (User)<-[Answer]-(User)<-[Comments to Answers]-(User). We apply static centrality and non-selective temporal centrality to both datasets and compare their results with those obtained using semantically aware metrics—either selective evolving centrality or temporal centrality with meta-path. Thus, in Table 7, the first column presents the similarity (measured using the Kendall Tau coefficient[10]) between the entity rankings in the Social Experiment dataset, as computed by static centrality and non-selective temporal centrality, and the rankings obtained using selective evolving centrality. The second column presents the similarity between entity rankings in the Math Overflow dataset, comparing static centrality and non-selective temporal centrality with temporal centrality with meta-path.

We observe that the entity ranking by static centrality and non-selective temporal centrality has small Kendall Tau value with the ranking by selective evolving centrality and temporal centrality with meta-path. This implies their results are irrelevant to the

[9]physical proximity of students recorded by Bluetooth proximity of students' cell phone
[10]Kendall Tau, ranging from 0 to 1, indicates the similarity between two rankings. A higher value signifies greater similarity.

analysis demand and in these cases using semantic aware metrics is necessary. Additionally, the meta-path is not suitable for the first analysis because it is too challenging to enumerate patterns of online communication with different types of relationship and different length of paths. Moreover, it can only be applied to centrality metrics related to path, like betweenness centrality. For the second analysis, simply making restrictions on relationship types can involve extraneous patterns in the calculation.

In conclusion, in case of semantic concern needed, using a static or non-selective temporal metric can lead to erroneous results. We should choose the proper method based on analysis need, for example keeping irrelevant semantics out of calculation or define appropriate semantic pattern.

5 CONCLUSION

In this paper, we propose a complete approach for measuring centrality in graphs, including evolution and/or heterogeneity.

First, we define the *selective centrality function* to calculate the centrality of an entity. This function has the advantages of (i) following the changes in centrality over time, (ii) allowing the analysis to focus on a subset of relationship types, (iii) being generic by allowing the calculation of any graph model and metric.

Second, we propose three approaches to use the results of our calculation function: (i) the global centrality ranking that provides a synthetic view of ranking, (ii) the dynamic centrality ranking that provides an evolution view of ranking, and (iii) the centrality spreading that highlights the differences between entities.

Third, we propose an algorithm to implement our approach coupled with a basic parallel and distributed computation approach. On the basis of real-world datasets, the experimental results have shown that our implementation method guarantees the algorithm's efficiency in terms of runtimes. Globally, the algorithm's runtimes increase linearly with the graph size, the relationship type set and the time interval length.

Finally, our algorithm has a great analytical power thanks to the evolution and selectivity capabilities of our centrality function and approaches to use centrality values. Indeed, we were able to analyze entities' centrality from multiple perspectives.

In future work, we will further improve the efficiency of our solution by using some state-of-the-art distributed systems for graph analytics [12, 14, 18, 19]. This study will be done with synthetic datasets to cover more variability (topological characteristics, etc). Moreover, in existing work, centrality metrics are used individually. Each metric integrates a fragmented vision of centrality. In the future, we would like to identify the metrics that complement each other to provide a panoramic view, as it is done in this article with the degree, eigenvector and betweenness metric. Then, we would like to propose a classification of metrics to be able to propose a framework that assists a user in the choice of metrics, as well as ranking techniques adapted to business needs.

REFERENCES

[1] Landy Andriamampianina, Franck Ravat, Jiefu Song, and Nathalie Vallès-Parlangeau. 2023. Semantic Centrality for Temporal Graphs. In *European Conference on Advances in Databases and Information Systems*. Springer, 163–173. https://doi.org/10.1007/978-3-031-42941-5_15

[2] Landy Andriamampianina, Franck Ravat, Jiefu Song, and Nathalie Vallès-Parlangeau. 2022. Graph data temporal evolutions: From conceptual modelling to implementation. *Data & Knowledge Engineering* 139 (May 2022), 102017. https://doi.org/10.1016/j.datak.2022.102017

[3] Ulrik Brandes. 2001. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology* 25, 2 (June 2001), 163–177. https://doi.org/10.1080/0022250X.2001.9990249

[4] Ariel Debrouvier, Eliseo Parodi, Matías Perazzo, Valeria Soliani, and Alejandro Vaisman. 2021. A model and query language for temporal graph databases. *The VLDB Journal* (May 2021). https://doi.org/10.1007/s00778-021-00675-4

[5] Mahmoud Elmezain, Ebtesam A Othman, and Hani M Ibrahim. 2021. Temporal degree-degree and closeness-closeness: A new centrality metrics for social network analysis. *Mathematics* 9, 22 (2021), 2850. https://doi.org/10.3390/math9222850

[6] Marwan Ghanem, Clemence Magnien, and Fabien Tarissan. 2019. Centrality Metrics in Dynamic Networks: A Comparison Study. *IEEE Transactions on Network Science and Engineering* 6, 4 (Oct. 2019), 940–951. https://doi.org/10.1109/TNSE.2018.2880344

[7] Petter Holme and Jari Saramäki. 2012. Temporal networks. *Physics reports* 519, 3 (2012), 97–125. https://doi.org/10.1016/j.physrep.2012.03.001

[8] Fuad Jamour, Spiros Skiadopoulos, and Panos Kalnis. 2018. Parallel Algorithm for Incremental Betweenness Centrality on Large Graphs. *IEEE Transactions on Parallel and Distributed Systems* 29, 3 (March 2018), 659–672. https://doi.org/10.1109/TPDS.2017.2763951

[9] Miray Kas, Kathleen M Carley, and L Richard Carley. 2013. Incremental closeness centrality for dynamically changing social networks. In *Proceedings of the 2013 IEEE/ACM International Conference on advances in social networks analysis and mining*. 1250–1258. https://doi.org/10.1145/2492517.2500270

[10] Miray Kas, Matthew Wachs, Kathleen M Carley, and L Richard Carley. 2013. Incremental algorithm for updating betweenness centrality in dynamically growing networks. In *Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining*. 33–40. https://doi.org/10.1145/2492517.2492533

[11] Hyoungshick Kim and Ross Anderson. 2012. Temporal node centrality in complex networks. *Physical Review E* 85, 2 (Feb. 2012), 026107. https://doi.org/10.1103/PhysRevE.85.026107

[12] Nicolas Kourtellis, Gianmarco De Francisci Morales, and Francesco Bonchi. 2015. Scalable online betweenness centrality in evolving graphs. *IEEE Transactions on Knowledge and Data Engineering* 27, 9 (2015), 2494–2506. https://doi.org/10.1109/TKDE.2015.2419666

[13] Linyuan Lü, Duanbing Chen, Xiao-Long Ren, Qian-Ming Zhang, Yi-Cheng Zhang, and Tao Zhou. 2016. Vital nodes identification in complex networks. *Physics Reports* 650 (Sept. 2016), 1–63. https://doi.org/10.1016/j.physrep.2016.06.007

[14] Maria Massri, Zoltan Miklos, Philippe Raipin, and Pierre Meye. 2022. Clock-G: A temporal graph management system with space-efficient storage technique. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, Kuala Lumpur, Malaysia, 2263–2276. https://doi.org/10.1109/ICDE53745.2022.00215

[15] Soheila Molaei, Reza Farahbakhsh, Mostafa Salehi, and Noel Crespi. 2020. Identifying influential nodes in heterogeneous networks. *Expert Systems with Applications* 160 (Dec. 2020), 113580. https://doi.org/10.1016/j.eswa.2020.113580

[16] Lutz Oettershagen, Nils M. Kriege, and Petra Mutzel. 2023. A Higher-Order Temporal H-Index for Evolving Networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, Long Beach CA USA, 1770–1782. https://doi.org/10.1145/3580305.3599242

[17] Raj Kumar Pan and Jari Saramäki. 2011. Path lengths, correlations, and centrality in temporal networks. *Physical Review E* 84, 1 (July 2011), 016105. https://doi.org/10.1103/PhysRevE.84.016105

[18] Christopher Rost, Kevin Gomez, Matthias Täschner, Philip Fritzsche, Lucas Schons, Lukas Christ, Timo Adameit, Martin Junghanns, and Erhard Rahm. 2022. Distributed temporal graph analytics with GRADOOP. *The VLDB Journal* 31, 2 (March 2022), 375–401. https://doi.org/10.1007/s00778-021-00667-4

[19] Christopher Rost, Kevin Gómez, Philip Fritzsche, Andreas Thor, and Erhard Rahm. 2021. Exploration and Analysis of Temporal Property Graphs. In *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021*, Yannis Velegrakis, Demetris Zeinalipour-Yazti, Panos K. Chrysanthis, and Francesco Guerra (Eds.). OpenProceedings.org, 682–685. https://doi.org/10.5441/002/edbt.2021.83

[20] Polina Rozenshtein and Aristides Gionis. 2016. Temporal pagerank. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II 16*. Springer, 674–689. https://doi.org/10.1007/978-3-319-46227-1_42

[21] Kshitij Shukla, Sai Charan Regunta, Sai Harsh Tondomker, and Kishore Kothapalli. 2020. Efficient parallel algorithms for betweenness- and closeness-centrality in dynamic graphs. In *Proceedings of the 34th ACM International Conference on Supercomputing (ICS '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3392717.3392743

[22] John Tang, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Vincenzo Nicosia. 2010. Analysing information flows and key mediators through temporal centrality metrics. In *Proceedings of the 3rd Workshop on Social Network Systems*. ACM, Paris France, 1–6. https://doi.org/10.1145/1852658.1852661

[23] Dane Taylor, Sean A. Myers, Clauset Aaron, Mason A. Porter, and Peter J. Mucha. 2017. Eigenvector-based centrality measures for temporal networks. *Multiscale modeling & simulation : a SIAM interdisciplinary journal* 15, 1 (2017), 537–574. https://doi.org/10.1137/16M1066142

[24] Ioanna Tsalouchidou, Ricardo Baeza-Yates, Francesco Bonchi, Kewen Liao, and Timos Sellis. 2020. Temporal betweenness centrality in dynamic graphs. *International Journal of Data Science and Analytics* 9, 3 (April 2020), 257–272. https://doi.org/10.1007/s41060-019-00189-x

[25] Zelin Wan, Yash Mahajan, Beom Woo Kang, Terrence J. Moore, and Jin-Hee Cho. 2021. A Survey on Centrality Metrics and Their Network Resilience Analysis. *IEEE Access* 9 (2021), 104773–104819. https://doi.org/10.1109/ACCESS.2021.3094196

[26] Xinrui Wang, Yiran Wang, Xuemin Lin, Jeffrey Xu Yu, Hong Gao, Xiuzhen Cheng, and Dongxiao Yu. 2024. Efficient Betweenness Centrality Computation over Large Heterogeneous Information Networks. *Proceedings of the VLDB Endowment* 17, 11 (July 2024), 3360–3372. https://doi.org/10.14778/3681954.3682006

[27] Matthew J Williams and Mirco Musolesi. 2016. Spatio-temporal networks: reachability, centrality and robustness. *Royal Society open science* 3, 6 (2016), 160196. https://doi.org/10.1098/rsos.160196

[28] Huanhuan Wu, James Cheng, Silu Huang, Yiping Ke, Yi Lu, and Yanyan Xu. 2014. Path problems in temporal graphs. *Proceedings of the VLDB Endowment* 7, 9 (May 2014), 721–732. https://doi.org/10.14778/2732939.2732945

[29] Shiqi Zhang, Renchi Yang, Jing Tang, Xiaokui Xiao, and Bo Tang. 2023. Efficient Approximation Algorithms for Spanning Centrality. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, Long Beach CA USA, 3386–3395. https://doi.org/10.1145/3580305.3599323

[30] Tianming Zhang, Yunjun Gao, Jie Zhao, Lu Chen, Lu Jin, Zhengyi Yang, Bin Cao, and Jing Fan. 2024. Efficient Exact and Approximate Betweenness Centrality Computation for Temporal Graphs. In *Proceedings of the ACM on Web Conference 2024 (WWW '24)*. Association for Computing Machinery, New York, NY, USA, 2395–2406. https://doi.org/10.1145/3589334.3645438