Temporal networks and applications

Graph Matching Algorithms for Temporal Networks

GIORGIO LOCICERO

UNIVERSITY OF CATANIA

BACHELOR'S DEGREE IN COMPUTER SCIENCE (L-31)

THESIS

Supervisor: ALFREDO FERRO
Co-Supervisor: MICALE GIOVANNI

ACADEMIC YEAR 2019/2020

Contents

List of Algorithms

Abstract

Technologies have been growing rapidly in recent decades, the links between various objects and subjects are more and more, new links are formed with the passage of time at certain precise temporal moments, and therefore new technologies are used to study these links, model them and reproduce the same real behavior that is seen in daily interactions, studying the most common structures to steal meaningful but hidden information. In particular, temporal graphs-networks are able to model any system in which there are interactions that occur at certain times (whether there are quantum times or intervals) and reproduce the same behaviors in a formal way, so they are of great importance to understand how interactions work between various agents (whether they are humans, machines, animals, emerging behaviors), study them in detail by extrapolating relevant information to the analyzes to be carried out, reproducing the characteristics of complex systems such as neural networks (for the modeling of artificial neural networks on the form of studies based on temporal graphs) and brain networks much more interesting also for the future of digitization. In the construction of a usable and adaptable basic infrastructure there will be many formalities, since maintaining the generality is very difficult to incorporate the temporal component, so techniques that focus on a local structure of interactions will be used.

Introduction

Static graphs are of great importance in many fields, and the studies concerning them are innumerable. Despite their undoubted usefulness in many applications, simple graphs are unable to model many situations and systems that it would be useful to emulate and model for studies and analyzes, see for example social networks, where it is possible to represent the links between individuals , but it is difficult to represent other characteristics such as sending messages between individuals or other characteristics which can be multiple per pair of nodes / elements of the system.

Time networks allow for the addition of additional details and attributes that increase the possibility of representing real networks, in particular representation of dynamic networks (neural, internet, infections, information and information passing between nodes).

As with other types of networks that approximate real situations and add details (such as multigraphs and multidimensional graphs), time networks try to represent the temporal component of reality and the interspersed interaction between various entities.

Precisely to represent this type of interactions complex graphs have been studied and used which, in the case mentioned above, are the multigraphs that allow more edges between two nodes or, in the case of representation of more types of relationships in the network to be represented, multilayer graphs (specifically multidimensional networks which are a specific type of multilayer) which allow to represent the different types of relations of the system and in the system.

An important part of many real situations that has not yet been clearly defined is the temporal component of a network, which is very often found in any complex system. The temporal component refers to the temporality of the interactions between the elements of a network, which can be expressed as time intervals or as individual contacts (these characteristics will be defined specifically in section 1: **Temporal networks** where they will be defined more formally, together with the logical and implementation problems that will arise), in particular the use of contacts is very useful and important to represent situations where a process of temporal and state expansion takes place, in fact the temporal contact graphs are very often accompanied by models of Markov and other state transition models, which thus are able to represent and model situations that could not be obtained with simple static graphs.

The simulation and modeling of real networks that have a time component (which can be a **timestamp** or a**temporal interval**)is of great importance in many areas ranging from any object of research:

- **Medicine**: in the modeling of brain networks and interactions between parts of the body or cellular interactions, such as metabolism, or synthesis of proteins, synaptic connections and microscopic and mesoscopic cerebral interactions, very useful for neurological and surrounding studies.

- **Transports**: in traffic modeling and road simulation, which can therefore be used for the dynamic change of routes based on time, or for the construction of new structures if certain situations require it.

- **Blockchain**: where the data is temporally concatenated therefore the temporal component is totally used for the implementation also for the sharing interactions between users.
- **Social Network**: as already specified, interactions between users are also identified by timestamps which, as already mentioned, can be single contacts or time intervals, thus increasing the accuracy of algorithms that use interactions between users and use the data for the benefit of together.
- **Distributed systems**: to study the distribution and connections of distributed systems in order to find the optimal configuration for maximizing performance.
- **Espansion models in general**: for example epidemic and tumor models, or the study of information distribution or other fields where the expansion of some type of information is brought about by well-defined events, such as contact with other people, and interactions with communities and large groups of people , with the creation of temporal subgroups.

In the first section we will give the useful definitions to define the problems that will be faced, in particular we will see a simple description of simple graphs which will then be expanded to temporal graphs.

In the second part we see the main problem that will be treated, that is the **matching of graphs** with the **temporal component**, and the important components for the solution of the problem, already mentioned in the first chapter, will be presented.

Finally, the algorithms that have been used for the matching and calculation of the isomorphisms will be presented, which will be described and seen in detail below.

We will also see the consequences of adding the temporal component, which may be simple at first glance, but hides a **complexity** typical of real representations that try to incorporate the temporal component and unfinished and unpredictable components in the construction.

In fact, construction and analysis of a temporal network are practices that involve several disciplines, a simple topological analysis may not be enough since the temporal information would be completely lost.

From another point of view, the pure analysis of the temporal component leads to the loss of very important information of a static structure that could be fundamental (this contrast of analysis techniques will be seen later when we talk about the various techniques that can be used for the analysis. of a temporal graph).

Of fundamental importance for temporal networks is the part of the theory that studies the processes and dynamics of expansion and propagation (spreading) of phenomena, compartmental models as a subdivision into characteristic states, represented by the nodes of the temporal network, i.e. each node belongs to a compartment.

For example, in the **SIR** (susceptible infected recovered) or **SIS** expansion model (where one returns susceptible after infection) a node can belong to three different compartments, susceptible, infected and recovered, the transitions from one compartment to another they occur through functions that take into account the contact time between the various compartments.

Other more complicated models have more compartments and model the reality of epidemic expansion situations even more in detail.

The probability of infection and recovery (**infection rate** and **recovery rate**) are important parameters for the model. Applying this model means assuming that all interactions occur uniformly over time, which is not true most of the time. For example, the infection of a virus also occurs based on the time of contact with the infected, the same speech can be made on the transition from infected to cured.

Vaccinations decrease the likelihood of getting infected, we can introduce models of vaccination and prevention strategies. This model, like all models that refer to real situations, assumes a bursty behavior during the various interactions and compartment passages.

This kind of epidemic and **expansion models** in general has become very fashionable in this period especially for the **SARS-CoV-2** emergency, therefore of great importance both for the modeling of solutions and for the analysis of the dynamics of expansion itself. .

Specific epidemic models will not be dealt with during this study, although small parentheses will be opened regarding any expansion model.

1 Temporal networks

Temporal networks will be presented below, starting first with the definition of simple graphs and then adding the time component.

An essential concept to initially understand is that each link (arc) of a time network transmits-carries information only during the time in which it is **active**, this activation time can be described by an established **quantum** (therefore the network in a set time can be seen as a state) or by a time interval in which contact can occur by adopting a real approach.

Well-defined time quanta are used when, rather than a pure temporal approach, it is necessary to study-represent a state approach with passage from state to state in well-defined (unique) times. Time frames are used in almost all situations involving real interactions.

Usually, real networks based on intervals are studied, but dynamic networks based on contacts are easier to use and analyze, also thanks to the immense literature concerning static networks, which are quite adaptable to some approaches used in the analysis of dynamic networks. During this study, time networks with single contacts will be used (only one time per arc), because the implementation has proved quite difficult for time networks with multiple contacts per arc and for time networks at intervals, but the reasoning that is made in all chapters are absolutely general for all types of time networks, only the implementation and the algorithms used change.

Specific algorithms for uncommon functionalities will not be dealt with in detail, but general utility algorithms will be seen which will then also be used for isomorphisms between temporal graphs.

Other specific formal definitions for graph isomorphism will be seen later and only hinted at, therefore the components of the temporal networks useful for the ultimate purpose of the isomorphism calculation will be seen.

Many useful definitions for other problems have also been omitted because they are of little relevance to the graph matching problem that uses local optimality measures to search for the global solution, and many of the measures described in [2] use global measures or that use a good part of the graph and that, in the end, found no use for the graph matching problem, so refer to that text for other possibilities.

Many of the definitions have also been created to help define isomorphism for time networks, for example the definition of *propagation*, with related classification, was created to describe the time structure of a graph node, and this time structure was used to describe the global graph matching problem, which other authors have entrusted to the definition of time-respecting-path.

We consider single contacts or single intervals per arc, but during the definitions there will be small brackets to expand the concepts to more contacts or multiple intervals for single edges.

On the definition of temporal graph matching we will see the various definitions given by the various authors [11, 12, 4] with the consequent problems that arise and the lack of generality of some.

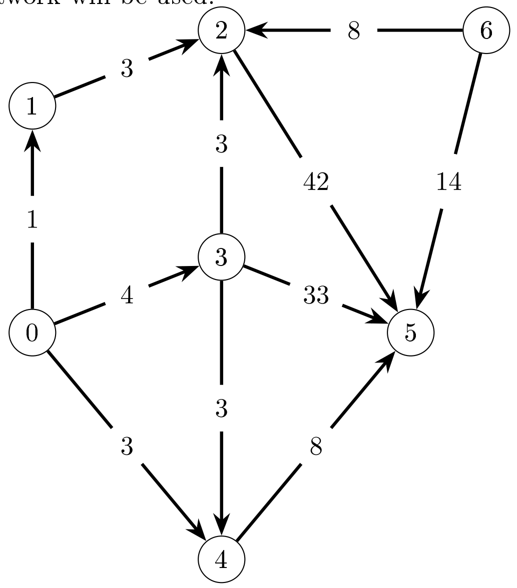
The problems that emerge from the use of contact and interval time networks will also be examined, with consequent choices and tradeoffs.

An important parenthesis will be given by interesting characteristics of time networks, for example the distribution of contacts or the behavior of certain real networks, which will influence the construction of the models and the temporal graph matching problem by delineating the contours of the various applications of time networks. with solutions to problems that manage to encompass most of the most relevant problems.

To see other important characteristics of temporal graphs (such as one-to-one, many-to-many, one-to-many relationships and events, or if the contacts occur synchronously or asynchronously, or which strategies to use if you don't want to work directly on the temporal network but on surrogates that lose some temporal or topological characteristics) read carefully [2] since it is the most up-to-date text and it contains a lot of insight useful for defining individual problems.

Directed temporal graphs are considered where the edges have only one direction, but the results can be expanded to indirect graphs very simply.

Throughout the definition and problem path, the following contact time network will be used:



1.1 Formal definitions

First of all we give a definition of a static graph and other basic definitions to introduce the basic concepts that will be expanded later:

Definition 1. (Static graph). Static Graph G is an ordered pair (V, E) of sets, where the set V is the set of nodes and $E \subseteq V^2$ is the set of edges between two nodes.

This definition is the most basic that you can have, it does not contain complex elements such as multiple attributes per arc, weights of the edges, labels associated with the nodes or the possibility of edges of different types between the same two nodes. To expand this definition with additional components such as the temporal component or the type of each arc and the possibility of various attributes, these characteristics can be added directly in the edges between the various nodes. assigning to each arc additional components such as lists or sets, or maintaining external structures that associate additional characteristics to elements of the static graph. In the case of multigraphs, the set of edges becomes a multi-set. At the end of this section, we will see how the addition of the temporal component adds considerable complexity to all problems related to graphs.

Definition 2. (Direct and indirect graph). If E is a symmetric relation then the graph is said to be undirected (or indirect), otherwise it is said to be oriented (or direct).

In the case of a static graph, the concept of direction is optional, but very often in temporal graphs the direction is mandatory, also because for Time-

Respecting paths the transitive and symmetric properties do not hold, so the paths can be seen as unidirectional.

Below we will give some basic definitions for the introduction of the concepts that are used for the description of temporal systems, then we start with the definition of temporal contact and then finish with the definition of temporal structure, which will be used entirely during the definition of the temporal graph matching problem.

Definition 3. (Time contact). A contact is a triple $(source, destination, time) \in V^2 \times \mathbb{R}$ which defines an interaction between a source node and a destination node in a certain moment in time.

This definition of temporal contact expresses a single moment in which the event can occur, in the case of intervals we speak of a contact interval and the nomenclature used remains the same (except for time which becomes interval), although usually the moment in the propagation occurs is in any case a single moment, but it still depends on the needs and on the individual descriptions of the problems that you are trying to simulate or solve.

Definition. Very often contacts or activation intervals are also associated with the nodes of the temporal graph themselves, also to simulate in a more concrete way real systems where there is this behavior, during this study this concept will not be considered, but the framework of definitions remains however expandable , especially the temporal fingerprint that will be exposed shortly, formed by a set of classes expandable according to your needs.

Although a temporal graph can be defined without the help of the definition of contacts, it is very useful during many formal definitions and in the description of the various algorithms (which in fact use the concept of contact actively).

Definition 4. (temporal graph for contacts). It is called temporal graph for contacts G_T an ordered couple (V, E) of sets, where the set $V$is the set of nodes (which can be identified through additional functions that associate a unique id to each node) and $E \subseteq V^2 \times \mathbb{R}$ is the set of edges between two nodes where the time component is a single number that expresses the moment of contact.

Other components can be associated with a temporal graph, depending on both topological and temporal structures, for example the static structure can be associated as infrastructure and use of measures and metrics derived from the temporal components to describe the temporal graph through global and local characteristics.

In the case of several contacts per node, these can be seen as single associations for the single arc or as separate edges.

Among the local characteristics that will later serve for the definition of Matching on temporal networks, the concept of propagation and temporal structure will be seen.

Definition 5. (temporal graph for contact intervals). It is called temporal graph for contacts G_T and ordered couple (V, E) of sets, where the set $V$is the set of nodes and $E \subseteq V^2 \times \mathbb{I} | \forall e \in E; a, b \in \mathbb{R}, a \leq b, e.interval = [a, b] \subseteq \mathbb{R}$

is the set of edges between two nodes where the time component is an interval identified by two real numbers which are the extremes that express the time in which the arc was active to send information.

During the activation of an arc in the specified time interval, the information propagates from node to node, so it can pass from one node to another only during the established time intervals, not before and not after.

The same goes for several intervals per node, which can be seen and defined as associated with the single arc or independent from each other.

It is important to specify this concept also because in the case of contacts there was an additional δ with respect to the individual contacts which allowed transmission between one node and another only if the time constraints were respected.

To the ordered pair defined above it is possible to associate other components such as the set of names for nodes or edges, functions that mathematically identify the individual nodes or edges and associate them with unique identifiers, as well as the addition of additional attributes for each component already defined, i.e. for each node, for each arc or for each contact, but during this study we will keep a simple definition for simplicity and to define a very easily expandable basic infrastructure

For the case of temporal graph at intervals, more or less the same reasoning applies, even if the complexity increases considerably in technicalities that are not very relevant in the case of contacts, but which assume great importance in the case of definition of propagation intervals that will be seen later. .

Definition 6. (Time-respecting walk). A time-respecting journey is a sequence of contacts with non-decreasing times, more formally:

Given a sequence of edges $\{e_1, \dots, e_n\}$, this sequence forms a time-respecting path if and only if $r \in [1, n[\subseteq \mathbb{N}, n = |V_G|, e_r = (s_r, d_r, t_r) \in E_G, d_r = s_{r+1}, t_r < t_{r+1}$, that is, a time-respecting path must have interactions between the various nodes that respect the normal flow of time.

In the case of multiple contacts for a single arc, this definition does not change.

A different speech must be made in the case of contact intervals, which slightly complicate the definition since the intervals carry with them the minimum of the intersections between the intervals of the incoming and outgoing arc. The definition is however dependent on the singular conditions of each case, therefore no additional definitions will be given, but the useful insights for the definition of problems in the case of an Interval temporal graph will be mentioned.

Definition 7. (Connectivity between two nodes i, j). Two nodes are said to be connected if there is a Time-Respecting path between the two nodes.

This definition defines the connection between the two nodes as the existence of a path connecting these two that respects the temporal ordering.

It can be noted that if two nodes (i, j) are connected to each other and (j, t) also, it is not certain that there is a Time-Respecting path between i and t, so the transitive property does not hold.

Another important factor to note is the fact that the connections are also temporal, as if two nodes are connected at a defined time, they do not necessarily have to be connected at the next instant, so you must also report some temporal attributes for the connection for do not waste time.

Another important graph that can be derived from the connections is the reachability graph. In a reachability graph a node has an arc oriented towards another node if and only if the first is connected with the second. Obviously, the arc is oriented because a path from a node a to a node b which is Time-Respecting cannot be used in reverse since the contact times will all be decreasing.

During the next chapters and the section in which the various graph matching possibilities are visited, other possibilities of representation of temporal graphs are seen that simplify or enrich the model to be analyzed.

Definition 8. (Temporal subgraph of a graphG_T). It is called a time sub-graph of a temporal graph $G_t = (V, E)$ the couple $G_s = (V_s, E_s)$ formed by the subset of nodes $V_s \subseteq V$ and the subset of contacts
$E_s \subseteq E | \forall e_s \in E_s, e_s.source \in V_S \wedge e_s.destination \in V_S.$

This definition is independent of the type of temporality of the interactions, so it is a valid definition for both contacts and intervals.

Definition 9. (Propagation between two nodes (i, j) through a node t). It is called propagation (or flow) $p_{itj} \in E \times E$ the ordered pair of edges
$p_{itj} = (e_r, e_s), e_r, e_s \in E | e_r.destination = e_s.source$.

Other characteristics are associated with a propagation such as the central propagation node (in the case of p_{itj} ,$p_{itj}.node = t$), propagation latency (which will be seen in the next definition) and, in the case of time intervals, the propagation interval which is obtained in the following way:

$$p_{itj}.intersect = p_{itj}.e_s.interval \cap p_{itj}.e_r.interval \tag{1}$$

$$p_{itj}.propInterval = \begin{cases} [min(p_{itj}.intersect), max(p_{itj}.e_s.interval)] & \text{if } p_{itj}.intersect \neq \emptyset \\ \emptyset \quad \text{if } p_{itj}.intersect = \emptyset \end{cases} \tag{2}$$

It can be noted that in the case of contacts the first time is always that of the outgoing arc, and the last time of the interval is given by δ defined, even if the ultimate goal is quite different and some results would not actually coincide.

In the case of multiple contacts, an additional distinction must be made between propagations with the same incoming and outgoing edges, if the single contacts (or intervals) are considered as separate edges, i.e. triples, the distinction is automatic, but the notation it must be changed slightly (by putting the incoming and outgoing contact times at the top), or by defining propagation sets for an incoming and outgoing arc for each contact-interval combination.

For example, the notation should be changed to $p_{itj}^{t_1 t_2}$ to express a propagation formed by the two edges that have multiple times, and therefore must be identified univocally, putting the times at the top. Or the notation may remain the same, but a propagation would no longer be the ordered pair $(e_r, e_s), e_r, e_s \in E | e_r.destination = e_s.source$ but the set of ordered pairs $\{(e_r, e_s), e_r, e_s \in E | e_r.destination = e_s.source\}$ where propagations can exist with the same incoming and outgoing edges, but with different time-intervals.

The case of multiple contacts-intervals will be revisited several times later also because it represents the generalization of the problem par excellence, as well as bringing with it considerable technicalities both in the logical-deductive factor, and in the performance of the algorithms that will be seen, which will have to compare. a factorial quantity greater (for the combinations obtainable between the various contacts of each arc).

Definition 10. (Propagation latency for contacts). Each propagation is associated with a number that represents the time interval between the contact time of the incoming arc and that of the outgoing arc, more formally:

Given p_{itj}, it is called latency $\Delta(p_{itj}) = p_{itj}.e_s.time - p_{itj}.e_r.time$, to limit redundancy, latency will be directly associated with each propagation as an attribute.

The concept of latency can also be expanded in other ways, for example by defining an information latency as the time interval between the transmission of information from one node and its arrival at another node, but this definition does not serve us for the problem which will be dealt with later on the matching since this type of information is global and has few implications in the resolution of the matching.

For further information on other measures that can be used for the creation of models, see [2].

Definition 11. (Set of propagations $P_t$of a node t). The set of all propagations of a node is defined as the set formed by all the propagations that pass through node t, more formally $P_t = \{p_{iqj} | p_{iqj}.node = t\}$

This definition establishes a good basis from which to derive other meaningful definitions for many problems, but it actually includes only graphs for individual contacts and ranges for each arc.

In the case of multiple contacts or intervals per arc, it is actually necessary to choose two intuitive paths already mentioned above, that is to consider the contacts as independent edges or belonging to a set associated with the arc.

In the first case, the definition does not change, and therefore the set would be substantially the same with the necessary distinctions for the propagations, which will have to take into account the different times, therefore using the notation already defined previously $p_{itj}^{t_1 t_2}$ which also takes into account the contact times as well as the incoming and outgoing edges.

In the second case we can refer to the definition of propagation as a set of ordered pairs of edges. So to construct the definition of the set of propagations as a set of sets, each formed by the combinations of the single sets contained in the corresponding propagation, the following formal definition is quite difficult and I cannot describe its accuracy.

Be $prop_{a_i t b_i}[i]$ the i-th propagation (as a set of ordered pairs) associated with node t and having source and destination a_i and b_i respectively, with

$i \in \mathbb{N} | 0 < i < inDeg(t) * outDeg(t)$,
then $P_t = \{\{(a_1, a_2), (a_3, a_4), ...(a_{Deg(t)*2-1}, a_{Deg(t)*2})|$
$(a_1, a_2) \in prop_{atb}[1] \wedge$
$(a_3, a_4) \in prop_{atb}[2] \wedge$
$... \wedge (a_{inDeg(t)*outDeg(t)-1}, a_{inDeg(t)*outDeg(t)*2}) \in prop_{atb}[Deg(t)]\}|$
$p_{atb}.node = t\}$

Example 12. (Set of propagations in a temporal graph with multiple contacts per arc). Given a temporal graph G_t with the following two sets of propagations associated with a node:
$p_{123} = \{((1, 2, 1), (2, 3, 4)), ((1, 2, 3), (2, 3, 4))\}, p_{428} = \{((4, 2, 6), (2, 8, 10)), ((4, 2, 11), (2, 8, 10))\}$
The set of propagations associated with node 2 is the following:
$P_2 = \{\{((1, 2, 1), (2, 3, 4)), ((4, 2, 6), (2, 8, 10))\},$
$\{((1, 2, 1), (2, 3, 4)), ((4, 2, 11), (2, 8, 10))\},$
$\{((1, 2, 3), (2, 3, 4)), ((4, 2, 6), (2, 8, 10))\},$
$\{((1, 2, 3), (2, 3, 4)), ((4, 2, 11), (2, 8, 10))\}\}$
Notice how the cardinality of the resulting set is $|p_{123}| * |p_{428}| = 4$, then 4 different time structures associated with a single node, which are the combinations of the arc contacts $1 \to 2$ with the possible arc contact times $4 \to 2$, therefore the resulting set will have cardinality equal to that of the Cartesian product of the sets of propagations associated with the node.

The set of all propagations belonging to a temporal graph is defined as $Propagations = \bigcup_{i}^{n} P_i$

Definition 13. (classification of a propagation). A propagation p_{itj}it can be of two different types:
In the case of contacts:

1. (Time-respecting propagation). A propagation p_{itj} is called Time-Respecting if $p_{itj}.e_s.time > p_{itj}.e_r.time$, in turn it is divided into two other sub-types:
   (a) (δ-time-respecting propagation). A propagationp_{itj} is called δ-Time-Respecting if $p_{itj}.e_s.time - p_{itj}.e_r.time < \delta$
   (b) (not δ-time-respecting propagation). A propagationp_{itj} is called non δ-Time-Respecting if $p_{itj}.e_s.time - p_{itj}.e_r.time \geq \delta$
2. (non Time-respecting propagation). A propagation p_{itj} is called not Time-Respecting if $p_{itj}.e_s.time \leq p_{itj}.e_r.time$

In the case of intervals:

1. (Time-respecting propagation). A propagation p_{itj} is called Time-Respecting if $p_{itj}.e_s.interval \cap p_{itj}.e_r.interval \neq \emptyset$
2. (not Time-respecting propagation). A propagation p_{itj} is called not Time-Respecting if $p_{itj}.e_s.interval \cap p_{itj}.e_r.interval = \emptyset$

As you can easily guess, the case of propagation on intervals generalizes that on contacts, also because the definition of the δ can be seen as an interval that starts at the established time and ends after a certain time ($[e_r.time, e_r.time + \delta[$), but actually the function of the δ is very often different, so we will keep the distinction.

You can add other distinctions or remove categories, the classification definition is very easily expandable with respect to the basic problem you are trying to solve. For example, if we wanted to consider the case of Time-Respecting temporal graph (i.e. the one used by many authors [8, 11, 12]) just consider the δ-time-respecting propagations and make comparisons accordingly.

The set of possible classes (which can be strings or enumerations) of a propagation is the following:

$PropType = \{TimeRespecting, NotTimeRespecting\}$

Eventually it is possible to associate other classes such as the $deltaTimeRespecting$ class but in any case it depends on the needs of the single problem.

The function that associates its specific class to each propagation is:

$classProp : Propagations \rightarrow PropType$

According to the following law:

(caso contatti)

$$classProp(prop) = \begin{cases} TimeRespecting & \text{if } 0 < p_{itj}.e_s.time - p_{itj}.e_r.time < \delta \\ NotTimeRespecting & \text{if } 0 < p_{itj}.e_s.time - p_{itj}.e_r.time \geq \delta \end{cases} \tag{3}$$

(caso intervalli)

$$classProp(prop) = \begin{cases} TimeRespecting & \text{if } p_{itj}.intersect \neq \emptyset \\ NotTimeRespecting & \text{if } p_{itj}.intersect = \emptyset \end{cases} \tag{4}$$

As already said we could generalize everything considering the delta for the definition of interval, but the definition has been defined separately to make it clear that the δ is a global parameter, and also because the intersection of certain intervals (incoming contact less than an outgoing contact, therefore the intersection between the resulting intervals would be non-zero, therefore ambiguous and contradictory with respect to the definition of contacts and time-respecting path for contacts) would result in Time-Respecting propagations when they should not be.

In the case of graphs with more contacts or intervals per arc, one of the two definitions previously given can be used, changing the equation described above considering each single element of the propagation, and returning the set of associated classes, or always using the same function , but going to work on the single elements of the propagation in the case in which a propagation is the set of pairs of contacts associated with the node, having the same sources and destinations, but different times.

Definition 14. (fingerprint or temporal character). The ordered multiset formed by all kinds of propagations of a node s is called a temporal fingerprint (or character) and is defined as follows:

$fingerprint_s = (PropType, classProp)$

$fingerprint_s = \{\forall prop_s \in P_s | prop_s^{classProp(prop)}\}$

This multiset is formed by the classes of the propagations of node s together with the multiplicity of the single elements.

In the case of intervals the definition does not change since the classes are well defined and a unique fingerprint of the node can be constructed.

For more contacts or intervals per arc the situation changes more, but using the propagation definitions as a set of contacts or intervals, it is possible to define the fingerprint as a set of sub-fingerprints, associated with the set of node propagations (i.e. P_t), thus building sub-fingerprints for each set contained in P_t.

In this way we can carry out the comparisons that will be seen in the future and which are essential for the calculation of similarity without problems of ambiguity associated with the first approach in the case of multiple contacts, where the propagations were all independent, given that the resulting fingerprint in that case would have been unique, but the cases would not have been severable from each other, and the actual time frame could lead to false positives during the comparison. This topic will be expanded in the near future.

Example 15. $fingerprint_3 = \{(TimeRespecting, 2), (NotTimeRespecting, 1)\}$

This example considers node 3 of the graph seen in section 1, all propagations are considered, and between them 2 are Time-Respecting and 1 not Time-Respecting

Example 16. (footprint in the case of several contacts per arc). Taking the example already discussed when defining a set of propagations for multiple contacts associated with an arc, the resulting footprint will be the following:

$$fingerprint_2 = \{(\{(TimeRespecting, 2), (NotTimeRespecting, 0)\}, 2), \\ (\{(TimeRespecting, 1), (NotTimeRespecting, 1)\}, 2)\}$$

Definition 17. (temporal structure of a node t). It is called the temporal structure of a node t and is marked ST_t the couple $(P_t, fingerprint_s)$, that is, the temporal structure of a node formed by its set of propagations and by the fingerprint itself.

This definition remains the same in the case of intervals and multiple contact-intervals per arc.

Having multiple contacts or gaps for a single arc might seem like nonsense compared to the definition of the main problem, but in reality they hide an immense additional complexity, especially during the implementation phase, particularly when calculating the similarity between two nodes.

For example, to check the time structure of a node, one should see all the possible combinations between the various contacts of each arc and the other edges that share a quantity of contact-intervals greater than 1.

Definition 18. (equivalence of time structure). Two nodes have the same time structure if they have the same fingerprint. More formally:

$j \equiv s \iff fingerprint_j = fingerprint_s$

In reality it would be more logical to separate the concept of fingerprint from that of temporal structure, which is however closely linked to the graph to which the node belongs, and not only to the types of propagations that pass through the node, for example by constructing a bijective function that maps each propagation of node j to propagation of node s, but the result would still be the same for the purposes that will be seen later.

In particular, it would be better to define the temporal equivalence based on an additional function that maps the propagations of the node to be compared to the propagations of the other node, in addition to the comparison of the fingerprints, this has not been done also because it is more a formality than a requirement , given that the equivalence of the fingerprints suggests the same temporal structure of the compared nodes.

The same argument made previously for multiple arc contact-intervals also applies in this case, the cardinality of the fingerprint increases exponentially to the amount of multiple contacts-intervals per arc if the entire fingerprint is considered for all possible propagations, otherwise it is possible to build more fingerprints on the basis of all combinations of contacts and define the same time structure if all fingerprints have a correspondent, not necessarily different.

Furthermore, again in the case of multiple contact-intervals per arc, the comparison of the fingerprints is not taken for granted, so one must be particularly careful in defining the problem.

Corollary 19. *The comparison between temporal structures is an equivalence relationship*

Proof. (equivalence of temporal structures) The comparison between temporal structures is an equivalence relationship since the symmetric, transitive and reflexive properties hold.

- (symmetry). $\forall j, s \in V | j \equiv s \iff s \equiv j$
- (reflexivity). $\forall j \in V | j \equiv j$
- (transitivity). $\forall a, b, c \in V | a \equiv b \cap b \equiv c \implies a \equiv c$

□

Definition 20. (similarity of temporal structure). Two nodes are temporally similar if from a subset of the propagations of a node the same fingerprint of the other node is obtained, formally:

S is similar to J and it is written $(S \simeq J) \iff J \in V, P_{subJ} \subseteq P_J \Rightarrow fingerprint_{subJ} = fingerprint_S$

It can be seen that this relationship is not symmetric but anti-symmetric, that is, if J is similar to S, the opposite is not said, and this factor is very important for the definitions of Temporal Graph Matching in the next section.

Corollary 21. *The similarity of two nodes is an order relationship (partial because the connex is not valid, so it is not total).*

Proof. (similarity as relation of order) The similarity between two nodes is a relation of order since the anti-symmetric, transitive and reflexive properties hold.

- (anti-symmetry). $\forall j, s \in V | j \simeq s \cap s \simeq j \iff j \equiv s$
- (reflexivity). $\forall j \in V | j \simeq j$
- (transitivity). $\forall a, b, c \in V | a \simeq b \cap b \simeq c \implies a \simeq c$
- (no connex). $\forall a, b \in V a \not\simeq b \not\Rightarrow b \simeq a$

□

Corollary 22. *More simply j is similar to s if and only if $fingerprint_j \subseteq fingerprint_s$*

Proof. (similarity for subsets of the fingerprints) If J is similar to S, among the possible propagation subsets of the considered nodes there will be one of which will have a fingerprint that will be equal to that of node S □

The similarity of one node to another has been defined because it will be central during the definition of Temporal Graph Matching.

Corollary 23. *If j and s have the same time structure then they are similar. Formally $J \equiv S \Rightarrow J \simeq S$*

The same is not true, on the contrary, of course, so during the comparisons it is necessary to check if a node i is similar to another j, then also $j \simeq i$.

Very important for the pruning part for the search for equivalent nodes in a subgraph is the contrapositive of this corollary, that is $J \not\simeq S \Rightarrow J \not\equiv S$, useful during the control phase for the mapping between nodes which will be seen in the next section.

1.1.1 Stream Graphs

A parenthesis of not negligible relevance must be brought for flow graphs, since they are an attempt at the generalization of formal definition for time networks.

Despite the malleability and adaptability of temporal networks to individual cases (which adapt both the definitions and the implementation for the specific case of problem to be addressed), an attempt has been made to define a general framework of formal work that we want to relate to the theory of graphs and expand it through the temporal component.

Definition 24. (flow graph). A flow graph S is defined as the tuple (T, V, W, E), where T is the temporal set , V is the set of nodes , $W \subseteq T \times V$ is a set of temporal nodes $E \subseteq T \times V \otimes V$ is the set of temporal edges.

A relatively more complex definition than previously seen, which also adds the temporal component to the nodes, as well as adding limits to the temporal component.

Definition 25. (times of presence of a node). Each knot $v \in V$ has a set of times of presence $T_v = t, (t, v) \in W$.

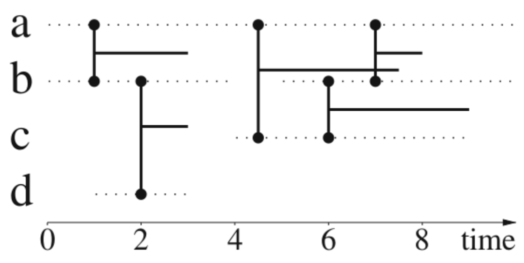


Figure 1 : Example of flux graph:$S = (T, V, W, E)$ with $T = [0, 10] \subseteq R, V = a, b, c, d, W = [0, 10] \times a \bigcup ([0, 4] \bigcup [5, 10]) \times b \bigcup [4, 9] \times c \bigcup [1, 3] \times d$, and $E = ([1, 3] \bigcup [7, 8]) \times ab \bigcup [4.5, 7.5] \times ac \bigcup [6, 9] \times bc \bigcup [2, 3] \times bd$. In other words, $T_a = [0, 10], T_b = [0, 4] \bigcup [5, 10], T_c = [4, 9], Td = [1, 3], T_{ab} = [1, 3] \bigcup [7, 8], T_{ac} = [4.5, 7.5], T_{bc} = [6, 9], T_{bd} = [2, 3]$, and $T_{ad} = T_{cd} = \emptyset$

Definition 26. (times of presence of an arc). $T_{uv} = t, (t, uv) \in E$is the set of times of presence of an arc $u \to v$.

Definition 27. (set of nodes and derived edges). $V_t = v, (t, v) \in W$ and $E_t = uv, (t, uv) \in E$ are the nodes and edges associated with a time or interval, thanks to which it is possible to define $G_t = (V_t, G_t)$.

This is the temporally induced graph very similar to a snapshot, although in reality t would be an interval, so it can be seen as a set of layers of the graph.

It is also possible to derive other measures and metrics (such as for example the average temporal degree of a node or the density of a flow graph which is the probability for which, when one randomly chooses an instant in time and two nodes present at that moment, these two nodes are connected together at that time).

It is also possible to define the concepts of bipartite graph (also exploiting the time components), weighted (with functions that change with time according to the moment in which the contact interval is found, very useful for many cases) and direct, but they are not useful for the ultimate purpose of this study, for further information see [5, 2].

During this study, however, they will not be used also because they do not add anything particularly useful to the problem that is mainly addressed, which is that of Temporal Graph Matching, and the formalization of definitions useful for the description of the problem has already been done previously.

1.2 Significant characteristics of temporal networks

There are many characteristics that can be relevant and significant during an analysis, for graph matching the measures of local identification and similarity are of fundamental importance to be able to identify the same characteristics

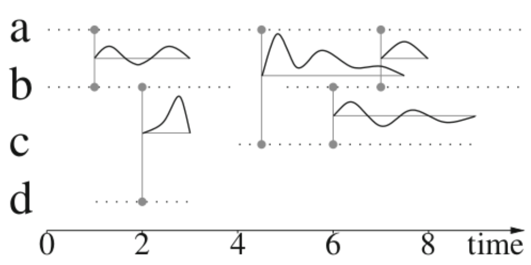


Figure 2: An example of a weighted flow graph, as you can see, the weights are functions that vary with the time of the interval, only the arcs are weighted.

of certain nodes and compare their similarity to see if it is possible find map functions to implement isomorphisms and endomorphisms in the same graph (in the graph matching section it will be seen that the calculation of endomorphisms will be needed to break the symmetries in the query graph).

Other measures that describe the characteristics of a time network are the distribution of degrees and contact times, very useful to see if the network follows a normal distribution of poisson, or the most frequent for Burstiness time networks which describes many contacts in short strokes. temporal.

An important part to be defined and modified according to the needs of the case is the concept of synchrony and asynchrony of the interactions between the nodes of the graph (the agents of the system), which may require particular attention during the construction of the structure of the models and algorithms. , as these characteristics actively influence the underlying infrastructure and problem definition.

The same can be said about the types of relationships, that is, one to one, one to many and many to many, which could significantly decrease the performance of any algorithm that works on temporal and topological components.

An additional consideration can be made on contacts that occur in a very close time interval, i.e., by defining a temporal neighborhood for the synchronized propagation from the same node, non-Time-Respecting propagations of the differences between the contacts or between the intervals can be considered. relatively small, perhaps caused by the latencies of the measuring instruments used to capture the base system.

Other additional conditions that can be added to check the similarity between nodes concern the valence of a node with respect to the whole graph, or similarity measures that exploit a sub-structure of the temporal graph.

Additional features can be achieved by using the transformation of the dynamic graph into other shapes and structures, such as special or time-network-derived static graphs, or by using structures such as lists or min-heaps to derive additional properties.

Even if representing the data as a temporal network means that some information must be discarded for reasons of simplification and digitization of the temporal component, this is often not sufficient to obtain a large-scale representation of the system that can be understood and analyzed with common techniques.

Perhaps the most obvious way to simplify a time network is to transform it into a static network using techniques that maintain some form of temporality among the characteristics of the graph. In the next subsection we will see ways to obtain static networks from the basic dynamic network.

As you can easily guess, temporal graphs have many points in common with statistic state models such as Markov models and the like, in fact in some implementations the temporal graphs are usually accompanied by Markov models built on the basic compartments and classes of single nodes.

Global metrics for temporal graphs are mainly derived from the topological structure of the graph, with minimal integrations of the time component.

Some temporal metrics can be the distribution of times for each node, the distribution of the times of each Time-Respecting path, and the parameters derived from these distributions and the fits to the best known distributions.

Other metrics can be derived from the flow graphs briefly described above, for further information always refer to the bibliography [2, 5].

Of great importance to many are the **mesoscopic** structures and features..

The term mesoscopic refers to the scale between macroscopic and microscopic. In network science, this would mean structures larger than nodes but smaller than the entire network and, in fact, the term is often used in contexts where there is a need to group nodes into classes based on how they are connected to each other. and the rest of the network. The main example of mesoscopic structures is the community structure where some networks have groups of nodes that are strongly connected within the group and loosely connected to each other.

Important sets during many analyzes are the **influence set** and the **source set of information**.

The set of influence is defined as the set of nodes reachable from the source node.

Alternatively, the source set of information can be defined as the set of all source nodes from which it is possible to reach the considered node.

These two sets can be seen as the past and the future of the node and information shared in the network.

Various definitions for metrics and measures on lead temporal graphs will be presented below but the definitions can be expanded at intervals:

Definition 28. (distance). The definition of distance is the same as for static networks adapted using Time-Respecting paths and the basic definitions that depend on the problem to be treated, formally:

Distance is a measure of path optimality that depends on what you want to optimize, for example the number of edges or the weights of the edges.

Definition 29. (time duration). A new quantity is introduced which is the duration, defined in a path as the difference of the arrival time of the information and the start time of the information transmission, formally:

having i and j the starting node and the final node of a Time-Respecting path, and $i^{>}$ ed $j^{<}$the nodes belonging to the Time-Respecting path connecting i and j such that $i^{>} \in i.Adj \cap j^{<} \in j.Adj$, the number $e_{jj^{<}}.time - e_{ii^{>}}.time$ is called duration.

In the case of intervals, it is necessary to take into account the first contact with the node in the interval between the first and the second node of the path, and the first contact between the penultimate and last node of the path.

Definition 30. (temporal latency). Latency is the minimum duration between all the times used in traveling paths between two nodes.

This definition is parallel to that of the minimum distance between two nodes, in fact both are based on the minimization of a characteristic associated with two nodes.

Definition. Algorithmic considerations on the search for time latency are very similar to the search for shortest paths, therefore as a dynamic problem that can be broken down into sub-problems with optimal solutions.

Definition 31. The quantity $\varphi_{i,t}(j)$ is defined as the last time before t in which the information starting from j has arrived at I.

Definition 32. (information latency). The function is called information latency $\lambda_{i,t}(j) = t - \varphi_{i,t}(j)$, what measure of how old the information is.

Definition 33. (vector clock). It is also defined $[\varphi_{i,t}(1), ..., \varphi_{i,t}(N)]$ as the vector clock, i.e. the vector of information latencies that shows all the freshness of the information that arrive at the node from other nodes.

Definition 34. (Closeness Centrality). Closeness centrality is defined according to the following law:

$C_c(i) = \frac{N-1}{\sum\limits_{i,j\neq i}^{n} \lambda_{i,t}(j)}$

where N is the number of nodes in the graph

Definition 35. (Betweenness Centrality). Betweenness centrality is defined according to the following law:

$C_B(i) = \frac{\sum\limits_{i,k\neq j\neq i}^{n} v_i(j,k)}{\sum\limits_{i,k\neq j\neq i}^{n} v(j,k)}$

Where $\nu_i(j,k)$ is the number of shortest paths that pass through i and $\nu(j,k)$ is the total number of shortest paths between j and k.

There are other measures of centrality and other ways of interpreting the various metrics that are used to calculate them (Path lengths, correlations, and centrality) but they will not be seen as they are very situational.

Other measures are important for the creation of random models useful for the creation of random time networks.

The creation of the random temporal network starts from an initial hypothesis (from this the model takes the name of null model since the infrastructure is built from the hypothesis starting from the arbitrary hypothesis), which can be for example the average degree of each node, the distribution of average arc activation intervals, or the probability of making contacts between nodes in certain time intervals.

There is no generalized model of randomized temporal networks, given the arbitrary nature of null models, rather you can either follow guidelines knowing the basic structure of the network you want to analyze and model, or you can follow a brute force approach up to exhaustion of possibilities, that is, try different null models and see which of them gives the most satisfactory results. Following a basic structure, one can take for example as an infrastructure a temporal graph with certain properties and follow topological and temporal structures.

Randomized edges (RE) is a random model for creating temporal graphs, where we change destinations with probability p.

Randomly permutated times is the temporal counterpart, the static structure is maintained, the contact times remain the same number but are permuted between the various edges, the cardinality of the contact times remains the same.

RP preserves the set of contact times, but destroys the typical behavior of burstiness time networks, another model that destroys these behavior patterns even more is the random times (RT) where random times are chosen for each contact, obviously the generation of random times does not necessarily have to follow a uniform distribution.

RE + RP is a hybrid between the two models.

Random contacts (RC) maintains the topology and redistributes the contacts.

Equal weight edge randomization (EWER) where edges with the same number of contacts can be replaced. This model keeps the burstiness, but it needs a large enough network to have enough leads.

Edge randomization (ER) is the generalization in which any arc can be swapped. The network topology is destroyed with this model, but the temporality is relatively conserved.

Time reversal (TR) with inverse times, ie the contacts are crossed in the opposite direction, correlations are sought between the normal and the inverse network.

Heuristic and empirical is mainly almost the whole process of research and construction of null models, each model favors some characteristics and destroys the others, usually an analysis is made using a group of these models in order to analyze a good part of the characteristics of the network temporal, or hybrids are exploited (eg RE + RP) that allow to maintain a balance between predictability and randomness.

For graphs with many contacts the result in using EWER is quite satisfactory, alternating with other models results as one of the "best" methods adopted in the creation of random temporal graphs.

Generative models will not be dealt with in detail during this study, even if the need will arise during the experimentation phase, but we did not want to spend any more space for this type of study since the study focuses on a general definition of a flexible infrastructure and usable in most applications.

1.3 Considerations for the implementation of temporal networks

During the definitions, nodes were used as identifiers, but usually in reality there are labels associated with integers that are the entries of the various adjacency lists, so we must not fall into deception with the theory.

The representation of a time network for contacts is very often advantageous.

This type of approach favors the vision of the temporal network as a series of slides (snapshots) of the network at certain times, in order to see the temporal network as a sequence of static networks, but there are also other types of representations that are favored by the contacts, for further information see [2].

In the case of contact networks, rather than defining a sequence of static networks, a three-dimensional adjacency matrix can be defined where the third dimension is time (if the contacts belong to the $\mathbb{N}$ set of course, perhaps de-normalizing the values or , in the case of intervals, transform the numbers into integers, and mark all the places of a matrix that belong to the set obtained $interval \subset \mathbb{N} \times \mathbb{N}$), and working on this matrix, through this definition is possible.

This adjacency matrix is defined as follows:

case contacts

$$TemporalAdj(i,j,t) = \begin{cases} 1 & \text{if there is an edge from i}\rightarrow\text{j at time t} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

case intervals

$$TemporalAdj(i,j,t) = \begin{cases} 1 & \text{if there is an edge from i}\rightarrow\text{j}\cap\text{t} \in e_{ij}.interval \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

In the case of adjacency lists, it is simply necessary to take into account the contacts if you want to keep individual contacts for each arc as different, otherwise you can keep sets associated with each arc that take into account the contact times of the arc to which they are Associates.

As already mentioned previously, time networks can have other representations (transmission graph, line graph, reachability graph).

The simplest static networks that truly encode some temporal effects are reachability graphs. These graphs have a direct arc (i, j) if an event can reach j from i through a Time-Respecting path. In fact, thanks to these reachability graphs, it is possible to establish quite relevant metrics for the calculation of various statistics and for the modeling of temporal graphs, for example a kind of betweeness can be defined on the basis of the degree of a node.

It can be shown that for each time network one or more reachability graphs can be constructed, and it can be shown that for each reachability graph a time network that respects it can be constructed.

A more elaborate way of reducing temporal networks to static ones is the extraction of backbones specifically adapted with respect to diffusion processes on temporal networks, for example by using time-respecting propagations from node to node, and by building a network in which all nodes are somehow reachable from a node of your choice, and then build the models on the basis of this new easily usable static graph.

Another way of representing dynamic networks as static networks uses the concept of line graph, where the role of nodes and edges is reversed. A transmission graph is then constructed, which incorporates in the definition of the nodes and edges of the line graph also the temporal component (both of the moment of contact and for how long it occurs).

A common approach that can be interpreted as static network projection is to use multilayer networks, using snapshots at intervals according to the following reasoning: time is divided into consequent intervals and the layers of a multilayer network correspond to aggregated networks for each interval. Once the layers have been coupled, modeling and mining methods for multilayer (static) networks can then be applied to the system. Importantly, the layers in such a projection are sorted by time.

Instead of reducing the time network data to static networks, you can try to preserve some but not all of the time components.

One example is the statistics of the times between contacts. It was recognized from early studies that often the times between events, both for nodes and for links, have heavy-tailed distributions (bursty as already mentioned above). So it is sufficient to recognize and simulate this property without taking into account other temporal characteristics to have a temporal network that simulates well most of the situations that one would like to study and model.

Another way to simplify time webs is to ignore the dynamics of contacts and simply think about the links between the first and last observation of a contact in the data and ignore the precise timing of temporal interactions. Compared to simplifying the system in the form of bursty dynamics on static networks, this technique emphasizes more lasting time structures such as general growth and decline in data activity.

There are also other types of representation (for example you can add the fact that the crossing of an arc involves a well-defined time, and build the network and the algorithms accordingly) that are based on other properties, but will not be seen in the detail since the purpose of this study is the construction of a framework that is useful in most cases and particularly detailed for graph matching.

For the representation used during graph matching, a standard representation enriched with contacts between the various nodes is used through a hashset to have good performance in adjacency checking. graphs with single arc contacts are used but the algorithms used should be general for multiple arc contacts.

The interval case has not been implemented but the basic reasoning is always the same (only the way to classify the propagations and the problem of initial contact times of the interval that will be seen in the next subsection changes).

As you can guess, it is quite simple to add the time component starting from a static graph, but one must not fall into the superficial banality of the problem, since it hides a considerable amount of problems related to many real structures, such as consistency internal structure of other support structures that depend on the temporal component, perhaps recursively, and which therefore require a complexity relevant to the analysis of large graphs (eg the duration of a path, the latencies of information, or other measures that depend on several contacts-intervals).

An alternative to the direct implementation in the static graph and its structure is the creation of a representation completely based on the temporal components, for example that exploits contacts and propagations for the model that was defined during this study, or that implements all the possibilities offered by flow charts described briefly in the previous sections.

Despite the importance of propagations in the definition and understanding of the problem that is faced and will be addressed in detail later, that is graph matching, very often the information they possess is redundant and derivable from some components of the simple temporal graph, also because integration of the temporal components has been made directly on a static representation of a graph so it is not possible to obtain an advantage on keeping structures for propagations.

This parenthesis on the low utility of propagations (only in this case given that other code had to be reused and adapted to search for isomorphisms on temporal graphs) does not remove their indeterminable importance for the definition of many problems that consider a local structure for the search of something, such as a search for cliques, motifs, communities, and other structures linked to a microscopic and mesoscopic vision of the temporal graph.

A possible optimization to the continuous calculation of structures and fingerprints lies in keeping an internal structure for each node that changes with the addition of edges between nodes (obviously correlated with contact time or contact interval), even if this computation of structure is carried out only for the computation of the compatibility domains that will be seen later.

1.4 Typical problems of time networks

Most of the problems that might arise during the construction of a time network and its analysis are usually connected with the basic problem that one tries to analyze.

For example, the transposition of a network of events as epidemic or metabolic models entails innumerable additional features to be taken into account during both temporal and topological analysis.

Other possible complications that can come to light come from the banality of analysis and understanding with which the basic problem is treated, which usually does not depend only on the temporal component or the topological component of the network with which it is represented, but on many other factors that influence the behaviors within the network.

Therefore, the more you try to represent an actual reality, the more problems you will encounter along the way of construction, implementation and analysis.

For example, in the case of time intervals, which do not seem to add additional complexity, it can easily be seen that at the moment of propagation it is necessary to take into account the interval in which the previous propagation took place, then start the interval of the new propagation in a range of the previous one or, to simplify, from the minimum of the resulting interval (this consideration of the minimum of the intersection has already been made in the definition of the propInterval of a propagation).

Definition 36. Given p_{itj} and p_{tje} two propagations, the propagation interval $p_{tje}.e_s.interval = p_{itj}.propInterval$ and not the arc propagation interval tj.

This definition was created to simulate the decomposition of paths into single propagations, adapting the previously defined infrastructure to a correct definition with respect to the common problem.

This definition traces back many problems since defining the interval according to a specific propagation from other propagations of the nodes preceding it is very ambiguous, therefore not isolated as a problem, which instead hides an internal recursion for the effective calculation of these intervals, dependent recursion from the path that is being considered.

In the implementation this problem has been avoided thanks to additional controls very similar to those of the fingerprint, but which use the single arc to be mapped (or the interval in the case of an interval graph) to see the feasibility of the actual mapping, this to see if the propagation is actually correctly mapped with respect to the path used to get from the source node to the destination node.

Before and during the construction of time networks, it is necessary to take into account how much information is lost and how much it is worth building a time network, i.e. consider the feasibility, accuracy and maintenance of the network, carrying out an effective risk analysis, given that the construction of a network, correlated by the construction of the associated models, is not something to be taken lightly.

In the construction of random temporal networks, considerations have already been made on the basis of random models, which in any case did not take into account some usable characteristics or metrics, but which used trial and error approaches, which were quite underperforming.

Very often there are also problems of scalability, as if the changes of a static network are minimal in time (too slow or too fast), but topologically enough (addition of additional nodes or edges), the construction of the temporal infrastructure becomes tedious. and useless in the long term, so a comparison of the speed of change of the static and temporal network must be carried out before continuing with the construction of a network.

Always concerning the scalability is the size of the network itself and the type, as we have seen during the definition of the various problems, in the case of multiple contacts or intervals, the solutions are very computationally heavy since you visit all the possible possibilities, so you the asymptotic temporal complexity of the possible algorithms that could be used increases factorial.

However, these problems can be avoided by keeping additional structures, sacrificing space for time, and decreasing the redundancy of repeated computation for the computation of the temporal structures of the components of a graph.

Not even the visualization and representation of a network contributes to the banality of the superficial construction of a time network, which could compromise both its complexity and interpretation, especially in the eyes of third parties who do not have much experience on the time infrastructure on which they should work.

One of the most important problems and which precludes the use of many techniques based on it is the intransitivity, a basic characteristic of many algorithms that work on graphs, such as the search for optimal paths or the search for connected components.

It is possible to add additional components and constraints to a time network, but particular attention must be paid to the possibility of making the infrastructure too specific for a single problem and inflexible for code expansions.

Although the scientific literature on time networks has been growing over the last few years, it is still not at all comparable to the literature on static networks, so most of the topics are still uncharted territory.

The literature on time networks is very scattered, there are many ambiguities during the study of the topic and the construction of a specific network for a single particular case, and usually freedom of choice and interpretation is left. Attempts at standardization and formalization have been made with flow graphs [5, 2]mentioned above, but still have not been fully tested for any case given their youth.

mentioned above, but still have not been fully tested for any case given their youth.

Time networks are complex systems that are particularly delicate and deceptive, also because the time component is not a simple number or interval, the passage of time is somehow something complex and not easy to study.

In the case of random models, to be meaningful the construction and analysis of the network it is necessary to have binary interactions, i.e. between two nodes, in order to maintain both a certain level of predictability and a certain randomness in the behavior of the network, so that it simulates real systems well. Furthermore, the temporal networks built should not be too random or too regular in order not to stray too far from the static network to which they have been interfaced.

Particular attention must be paid to the speed of the contacts compared to the speed of change of the dynamic system of the network. Being a dynamic system, the temporal network is particularly difficult to analyze, nodes can reenter and exit, edges may not be traversed, many variables must be taken into account and some may not be understood.

In the search for motifs, cliques, communities, and other types of patterns that have a certain regularity, there are different interpretations, as for the problem treated in this study which is that of Temporal Graph Matching, where many authors give completely different or unclear definitions, which often leave quite simple and useful cases in many applications, this type of problem will be covered in the next section.

Important in the analysis of temporal networks are not only the topological or practical causes for which certain interactions occur, there are not only different interpretations and analyzes of the temporal nature but, very often, there is also a search for the origins of the interactions between the various agents , for example the study of the contact between two nodes which, as already said, assumes a bursty behavior rather than following a known distribution (eg Poisson).

The study of time networks could benefit from many studies that are done on static networks (such as the coloring of graphs, or the large number of methods useful for the search of shortest paths, for the search of transitive closures), but given that the subject is quite young, although studies have been done since the nineties in an approximate way, there is a need to expand many of the researches that are done on graphs, multigraphs, etc. also to temporal graphs.

We mentioned the modeling and generation of temporal networks with particular characteristics, in this field of study there are methods very similar to those of static networks, with appropriate modifications that vary from case to case and which see the temporal component of particular relevance, therefore they build networks on the basis of what they would like to obtain, perhaps by adopting unique and non-standardized methods, therefore not very general.

Throughout this study, particular attention has been given to the descriptors of temporal structure, such as the temporal fingerprint defined previously, it could however take advantage of other types of structures that reflect the true nature of the temporal component, which are created and designed to be useful. in the analysis of dynamic networks, and which are able to describe global or mesoscopic properties in order to find similarities between the various temporal networks.

Nonetheless, temporal contact networks remain the state of the art of many studies (almost all studies use single contacts). This is also because they manage to represent quite normal and not too trivial situations without heavily affecting the complexity of the basic system. The fact remains that single moments do not represent many of the most important real situations, which instead have agents acting in intervals.

For other problems concerning the single system to be represented see [2] which does an immense job in the representation possibilities of systems that contain and incorporate the temporal component in some way.

2 Graph isomorphism

The search for isomorphisms and repeated patterns within the structure of a graph is something that has produced an immense amount of scientific material, also due to the fact that the results are very often relevant in many fields and shed light on certain mechanics that are not they could learn in complex systems, such as emerging behaviors or agents that play different roles but who, in the end, communicate very often with each other by exchanging messages and information.

In a very rough and general way, the search for isomorphisms in graphs is reduced to the search for structures that respect the conditions established according to the case and the type of structure being analyzed.

In the case of temporal graphs, the search for isomorphisms is very important in many fields, and often very different on the basis of the basic problem we are trying to solve.

In fact, as will be seen in the following subsections, many authors do not seem to agree on a single framework to use and on the definitions to be exploited for the description of the basic problem, although the problems they deal with are very often very similar and independent from the basic problem.

For example, on the one hand there is a search for temporal motifs that adopts an edge-driven search, which takes into account only a single definition that is quite narrow also due to the fact that the actual algorithm is focused on optimization for search and the count of small query-patterns (three nodes per query, see below)[11] and on the other hand a more generic definition but which concerns only a series of sub-problems and depends on simple conditions that do not really expose the complexity of the problem [12, 10, 8].

The problem therefore seems not to have been addressed in detail yet and an actually usable framework has not yet been found (or at least it has not been found during the searches, perhaps an unknown researcher has already thought about it but is not receiving the attention it deserves. and search engines don't help).

Throughout this study, definitions have been given that will be key to the definition of the basic problem, and therefore it is necessary not to forget what has been said previously and to revise the definitions in case there are any doubts.

The definitions given previously and in the near future are however not totally general, since there will almost certainly exist a case that has different conditions, even if the whole framework described has been built to be as general as possible, thanks to the definition of propagation classes that is easily expandable with other classes.

Other definitions and concepts will be mentioned that are exposed in other research [13, 12, 11, 7, 6, 10, 3], and the concepts will sometimes be implemented in the case study that we will take as an example, but global controls will not be implemented since they do not take into account groups of independent interactions and are almost always dependent on the basic problem. In case you need to expand the infrastructure with additional controls that are not local, just add controls at the end of the isomorphism search to check if the established constraints (whether mesoscopic or global) are respected in the sub-graph corresponding.

The graph matching on static graphs will be shown below to present the algorithm that will be used during the matching and its modifications, we will not go down too much on the details for the Sub-Graph Isomorphism for static graphs but the basic ideas will be presented, since the concept will be explored during the discussion of the possible modifications to the basic system of structures and algorithms for adapting to temporal graphs.

Towards the end, the algorithms used and the descriptions of these algorithms will be presented for a complete understanding.

Finally, problems regarding isomorphism on graphs and possible solutions will be seen and discussed, these problems are very relevant since they clarify once again the connection of the definitions of temporal graphs and isomorphisms to the case study of the problem.

2.1 Graph matching for static graphs

The problem of graph matching on static graphs has been dismembered in many ways since the beginning of studies on graph theory.

The studies are innumerable, the solutions range from simple visit solutions of all possible solutions to the more complex ones, which use the parallel properties of the sub-graph search computation in order to divide the problem and optimize the use of the single machine (using the integrated or discrete GPU, multi-threading and other technologies that allow the parallelization of computation), in addition to the use of additional heuristics for the reduction of the search space, also because the problem of searching for isomorphisms in graphs is NP -hard as a minimum, while the search for isomorphisms on sub-graphs is NP-complete, therefore solutions that decrease the computation time and the asymptotic complexity in general are absolutely welcome.

Among the implementations that perform an isomorphism check on a single graph there are mainly algorithms that implement a transformation of the graph into a unique canonical form that can be identified and compared to see if the isomorphism conditions are satisfied (among these algorithms Bliss, Saucy , Conauto and Traces).

The search for the canonical form is not a simple problem, but there are many studies on it.

For the problem concerning the search for isomorphisms on sub-graphs there are therefore as many solutions, since the problem is a generalization of the search for isomorphisms between graphs.

These solutions of the Sub-Graph isomorphism will not be seen all, but will only be hinted at approximately in the most important components that directly influence the performance and the logical reasoning for the search for optimal solutions.

The algorithm for the search of isomorphisms on sub-graphs is RI, It will be seen in detail[1, 9], it incorporates many heuristics of other algorithms, and favors the speed of the algorithm and the reduction of the search space for solutions.

A simple enumeration algorithm to find all the subgraph isomorphisms (i.e. occurrences) of a pattern graph in a target graph works as follows: all possible matches between the vertices of the two graphs are generated and checked for any matches generated is a subgraph isomorphism (which will be called match). Considering that this algorithm is totally inefficient when implemented naively, as well as being a real brute force, it serves as a starting point.

All matches can be represented using a search space tree.

The tree has a fictitious root. Each node represents a possible correspondence between some vertex u of the query graph G and some vertex u 'of the target graph G'. The path from the root to a given node represents a partial match between G and G '. Only some paths from the root to the leaves correspond to the isomorphisms of the subgraph between the query graph and the target graph, and it is precisely those that must be sought primarily in order to have a fast matching.

Finding a solution for the isomorphism search problem in sub-graphs is inherently difficult and therefore the efficiency of any software that uses isomorphism search algorithms largely depends on:

1. find efficient heuristics to make the isomorphism algorithms faster, then use them in a hybrid way so as to have the best possible characteristics;

2. reduce the number of isomorphism calls of the subgraph, therefore summarily decrease the execution time;

3. decrease and make the isomorphism conditions simple and fast, in order to have a solid base that does not have too much load on the system.

Many algorithms work by carrying out local comparisons with respect to the node, implementing lookahead rules for checking the conditions for a candidate node and the nodes not yet mapped connected to the node to be mapped, this function is useful in many cases (it has also been implemented in implementation code of the temporal graph a function that checks lookahead rules up to the third level with the name of testNodeMapping, but will not be used in the actual study because RI does not use expensive lookahead rules) but is very expensive in performance, and the pruning can be done without too much trouble for small neighborhoods of a node without spending too much time checking for redundant conditions. When a node does not meet the conditions to be mapped, it goes back (rollback or backtracking to a node higher in the search tree) and check other possible mappings.

During the visit the isomorphism conditions are checked to verify the partial combinations. When the conditions are not met, the algorithm drops the underlying branches and goes back to the main nodes of the search tree. The size of the search space tree increases exponentially with the size of the graph.

The main goal is to eliminate paths from root to leaves that are not isomorphisms as quickly as possible.

A feature that greatly influences the solution search strategy is the ordering of the query graph nodes that are analyzed during the matching phase.

Thanks to this sorting, it is possible to decrease the search space very quickly, and finding the solutions is consequently faster.

The ordering of nodes can be of two types: static and dynamic.

Using a static sort, you define an initial sort that is the same for any path from root to leaf. In dynamic ordering, there is an order for each individual path and these orders are usually established at runtime.

Among the most used algorithms to search for isomorphisms in subgraphs today there is VF (with subsequent versions that optimize and increase performance, until today the version is VF3).

VF (acronym for Vento-Foggia) uses a dynamic search strategy.

Given a partial solution, first it chooses the vertices of the non-corresponding pattern having edges that have source vertices of the partial solution; then it chooses those mismatched vertices that have edges with destination in vertices of the partial solution. To reduce the search space, the approach uses the following two lookahead heuristics. A mapping pair (u, u'), where u and u' are vertices of the query and target graphs, respectively, is considered a valid match if it satisfies the following rules:

1. u and u 'are both neighbors of the vertices that have already been mapped;

2. the number of unmatched pattern vertices that are close to matched vertices and are connected with u must be less than or equal to the number of unmatched target vertices that are close to matched vertices and are connected with u' .

Rule (ii) is divided into four cases according to the direction of the edges involved between the neighbors of u and the set in (i).

There was also another lookahead rule that considered all connected nodes, but it is not described as it is only used for induced subgraphs.

However, there are two major families of isomorphism search algorithms: Tree Search (TS) and Constraint Propagation (CP) algorithms.

Tree Search algorithms formulate the correspondence problem in terms of a State Space Representation (SSR), which consists of exploring a tree of the search space. Each state of the SSR corresponds to a partial solution. The search space is visited in a deep first order, and the heuristics is designed to avoid exploring parts of the search space using additional rules. The algorithms of this class include the Ullmann algorithm, VF2, VF3, RI and RI-DS.

In Constraint Propagation methods, the subgraph matching problem is represented as a Constraint Satisfaction Problem. The nodes of the query graph are represented as variables and the nodes of the target graph represent the values that can be assigned to those variables. The edges are translated into constraints that must be satisfied.

The CP algorithms first compute a compatibility domain for each node of the pattern-query graph and then iteratively propagate the constraints to reduce those domains and filter candidate nodes for matching. The most famous CP methods are nRF +, Focus-Search and LAD.

Several filtering techniques, such as forward-checking, eliminate branches of the search tree by propagating constraints to remove values from potential compatibility domains. A branch is deleted when a domain becomes empty.

In forward-checking, a variable is first assigned, then all constraints involving those variables are propagated to remove values from other domains that are inconsistent with the current assignment (inference).

Inference-based methods, which propagate constraints to convergence (e.g. LAD), minimize the seek time. Unfortunately, this inference comes at the cost of higher computational and preprocessing cost. On the other hand, when the constraint checking is applied only locally (for example, the local inference used by FocusSearch and the VFlib pruning rules), it is essential to define a search strategy that seeks to thin the search space in optimally and as soon as possible at low cost.

The most important characteristics for the performance of an algorithm described above are, for the most part, implemented in RI.

RI adopts a search strategy based only on the topology of the query graph. The order is chosen to create the constraints as soon as possible in the matching phase. Roughly speaking, vertices that have high valence and that are highly connected with vertices previously present in the sort tend to come first in the final sort of variables. During the matching phase, RI does not apply burdensome pruning or inference rules for the computation and computation. A powerful ordering of the vertices of the query graph together with the verification of not too onerous constraints, is more efficient than a local or global inference procedure.

The RI algorithm is divided into four main phases:

1. The first phase of RI calculates the compatibility domains $Dom(q)$ for each node of the query graph q , this domain is the set of nodes in the target graph that can be mapped to one of the nodes of the query graph based on topological conditions such as the degree of the node and other comparable metrics or structures (then we will see which conditions to add to integrate the temporal component). This phase increases performance precisely because only the nodes of the target graph in the domain of q will be considered as possible candidates for the match of a node during the isomorphism search phase. Formally, it is $Q = (V_Q, E_Q)$ a graph called query and $T = (V_T, E_T)$ a graph called target. A knot $t \in V_T$ is compatible with a node $q \in V_Q$if the following condition is satisfied:

   - $deg(q) \leq deg(t)$

   After the computation of the compatibility domains, an Arc Consistency (AC) technique is applied to remove possible candidates who do not meet the minimum requirements. The procedure AC states c and if there is an arc between two target nodes, q and q ', then for each target node among the compatible ones (Dom (q) as source and Dom (q') as destination) there must exist an arc between the nodes of the domain of the source node and at least one node among those of the destination node and vice versa. This implies that if a target node t belongs to the domain of a query node q but does not satisfy the condition AC, it can be removed from Dom (q).

2. The second phase concerns the ordering of the nodes of the query graph where RI calculates the order in which the nodes of the query graph will be processed during the matching phase. The basis of the algorithm is to define a possible order for the processing of the query nodes. In RI, query nodes that both have a high rank and are highly connected to nodes already present in the partial sort come first in the final sort. Formally, let n be the number of nodes in the query graph and $\mu^{i-1} = (q_1, q_2, ..., q_{i-1})$ the partial sort up to (i − 1)-th node, with i < n. A set U^{i-1} of nodes not yet in the sort is also defined. To choose the next node in the sort, three sets are defined for each node of the candidate query graph q:

   (a) $V_{q,vis}$: The set of nodes in μ^{i-1} and adjacent to q;

   (b) $V_{q,neig}$: The set of nodes in μ^{i-1} which are adjacent to at least one node in U^{i-1} and connected to q q;

   (c) $V_{q,unv}$: The set of nodes in adjacent to q that are not in μ^{i-1} and are not adjacent to nodes inμ^{i-1} .

   The next node in the sorting is the one that meets the following conditions: (i) has the greatest value of $|V_{q,vis}|$, (ii) if there are more matches in (i), the one that has the greater value in $|V_{q,neig}|$, (iii) in case there are more matches in (ii), the node that has the greater value in $|V_{q,unv}|$. In the case of several candidates in each condition, the node is chosen arbitrarily. This definition can be expanded with other conditions dependent on the single problem, but in reality this phase will not be modified since the purpose of the search tree reduction is already fully implemented even in the case of temporal graphs with this definition.

3. A problem that arises in the computation of isomorphisms in sub-graphs is that the matching process can produce redundant occurrences. In order to exclude redundant occurrences during the solution search process and prune the search space tree from redundant solutions, RI defines symmetry breaking conditions based on the node identifiers of the query graph. The symmetry breaking conditions are related to the concepts of automorphisms and orbits of a query graph (defined in section 2.3). Given two query nodes$q$ and q_0 belonging to the same orbit, with $id(q) < id(q_0)$, a symmetry breaking condition is an inequality of form $q \prec q_0$, indicating that node q must precede the node q_0. In other words, a symmetry breaking condition is a condition that imposes a relative order between two query nodes belonging to the same orbit.

4. Last and most important phase is the matching phase. Following the previously defined order of the query graph nodes, RI starts the matching process to find the occurrences of the query within the target graph,

using the symmetry breaking conditions to eliminate redundancies as it proceeds. Matching is done by building a mapping function $f : V_Q \to V_T$ (initially not defined for all query nodes) and the corresponding match M, which is initially empty. Whenever a new match is found between a query node q and a target node t, the pair (q, t) is added to M. When all the nodes of the query graph have been matched, M constitutes a new match of Q in T, so it can be added to the list of found matches.

2.2 Graph matching for temporal graphs

During this study we will perform a matching on temporal graphs with single contacts per arc, this is both to simplify the understanding of the basic problem that can be easily expanded, and to establish a solid basis that can be evolved according to one's needs.

The theoretical part will however remain general with respect to any type of temporal graph, and additional considerations for more contacts or intervals per arc will be made during the formal definitions of the problem.

However, it must be kept in mind that the implemented algorithm only considers graphs with single contacts per arc (which is however easily expandable from the basic problem for more single contacts, not so simple to implement for too many contacts and for intervals in general).

We also used single contacts because most of the works seen in the Temporal Graph Matching field used either single contacts, or they established very weak conditions and constraints that will be seen below in some definitions given by some authors[11, 12, 8].

In un altro studio per la ricerca di motif su grafi temporali [11], the study mainly focuses on defining a counting process based on edge-driven comparisons, i.e. comparing groups of edges that respect a condition defined in the definition of temporal motif, but this definition is very similar to that of walking, and does not leave much space for expansion with other concepts, thus remaining a single and specific definition with respect to the problem of counting motifs in a graph (the study also focuses on motifs with 3 nodes -3 edges, dedicating a very small part to the general problem).

The definition of motif given in [11] it does not allow contacts that can have the same times, thus precluding most of the existing time networks that rely on this feature. The algorithm used sorts the edges and performs an edge-driven comparison, optimized for finding motifs with 3 edges and 3 nodes. The general problem is very poorly defined and does not seem to be general enough to be used in practice either.

For the definition of isomorphisms on subgraphs,[12] did not give a concrete definition, concentrating on the definition of a temporal subgraph that respects the time-respecting property, and left out the generalization of the problem by going directly to the solution suitable for the problem he wanted to solve (therefore considering the subgraphs in the target graph that respected the definition of time-respecting subgraph, where each path had to be Time-Respecting, so no other types of query graphs could be defined except those that respected this property).

The definition of sub-graph isomorphism on temporal graphs given in [12] it rests on the definition of Time-Respecting paths, not defining actual temporal structures that can be used for comparisons and conditions that can be used during matching, thus keeping the definition quite ambiguous and not very general.

There can be three possibilities for searching for isomorphisms on temporal sub-graphs:

- Time before topology: extract the temporal sub-graphs that respect the time-respecting hypotheses, then proceed with the matching of the simple topological subgraphs. This approach is particularly heavy in terms of performance, also because non-negligible redundancies can occur.

- Topology before time: perform the matching directly on the static topology of the graph, and then perform operations on the temporal components of the graphs obtained, then pruning on the actual temporal subgraphs that respect the hypotheses, this is the simplest strategy, also because they can be used directly known subgraph matching algorithms, to then manage the results obtained as desired.

- Time and topology together: The time control and the matching are done at the same time and then, taking as an example a simple Ullman algorithm, a check is made on the matching of each node and each edge, pruning solutions that do not respect the hypotheses of time-respecting match.

Of particular importance for the definition of isomorphisms on temporal subgraphs is the definition of query graph and target in the temporal case, since this definition is usually not given formally or in detail by the authors, and therefore very often leaves many doubts.

In this study, the query temporal graph is seen as a temporal pattern to search for. This means that the contact times are not important, but the types of propagations they produce with the consequent time fingerprints are at the center of the search for isomorphisms.

For the definition, only two possible classes were chosen for the classification of a propagation, however during the real implementation 3 classes (or more during the comparison of the edges) were used, depending on the propagation latency (less than 0, greater than 0 but less than δ, greater than delta).

During the search for usable possibilities to carry out the matching in a consistent way with the problem, two main methods were found from which it is possible to derive other more complex methodologies:

- Use a transformation of the query temporal graph into a static graph, always according to the transformation patterns that have been defined and will be explored practically in the next chapter, and then use this static graph for submatching on the target temporal graph.

- Carry out comparisons and matching based on complex comparisons, alternating and parallelizing comparisons and computation both on the static and temporal structure of each node and of the whole.

To implement the topology strategy before time, you simply need to match the static structure of the target graph and then check the dynamic structure, for example by carrying out a check on the mapped nodes and on the contact times of each node, so that they respect the defined time-respecting path rules.

To implement the Time and topology together strategy, the reasoning should be quite simple and would mainly focus on the actual matching phase (taking Ullman for example).

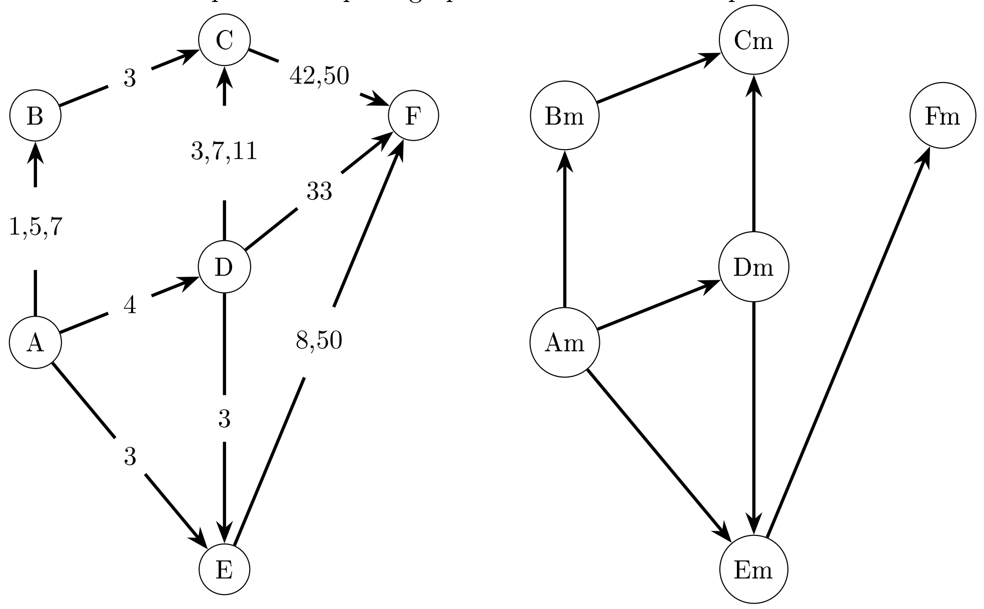
After the definition of the problem and the key points on temporal graph matching, it should not be difficult to use the time-respecting path and contact assumptions to broaden and implement look-ahead and rollback rules.

A solution possibility for subgraph matching algorithms of temporal graphs used by other authors [11, 8] is the conversion of the temporal graph into one or more induced static forms that allow the static matching of the network, in particular the strategies used introduce very simple methods in the conversion, which simply count the edges that respect one or more time conditions, followed by a possible quite simple conversion algorithm:

Algorithm 1 InducedStaticGraphFromTemporal(G)

```
1: function InducedStaticGraphFromTemporal($G$,$\delta$)
2:     edges sorted in sequence, select a node that have the less time of contact
3:     BFS modified with control over temporal attribute and $\delta$, add edge to
       Gtemp if conditions are satisfied
4:     for $tedge \in E_{Gtemp}$ do
5:         SG.addedge(tedge.s,tedge.d)
6:     end for
7:     return SG
8: end function
```

and below an example of a temporal graph conversion with multiple contacts:



An activation time interval has been chosen $\delta = 10$. As you can easily see, only one static graph has been obtained, this is not always the situation, especially when one needs to take into account all the activation patterns that occur in the temporal graph, so the solution presented is quite temporary and it would not bring concrete results (the problem still remains the case of arches with more contacts).

In the end, however, a hybrid approach of Time and Topology Together is used, where the temporal control is done dynamically together with the topological structure control.

An algorithm is used that does not make use of too complicated look-ahead rules that would compare a considerable amount of time structures that is previously exposed RI modified for the control of the similarity between nodes and of the temporal structures per node during various phases.

It is however possible to expand other concepts, such as for example the preprocessing phases of the RI calculation of the compatibility domains and the order of the nodes of the query graph for the matching phase. [9, 1], always using the definitions previously defined.

For example, for the calculation of the compatibility domains, a check on the degree of the node can still be done, and it is possible to alternate with a check on the contact times for each arc outgoing from the node, or directly a check of the fingerprint of the node to be analyzed. .

In fact, this is the solution that has been adopted, in addition to various checks on the degree and on the symmetries, a check has been added for the compatibility of a node of the target temporal graph with one of the query graph, exploiting the fingerprints of the two and seeing self$v_{query} \simeq v_{target}$,and then proceeding to add the node among the candidate nodes for mapping.

For the calculation of the ordering of the nodes of the query graph, one could do a local ordering (with respect to the degree of the nodes) and then an ordering on the number of contacts per arc (problems could arise which will be seen in the next chapter).

In the actual implementation, the calculation of the sorting was not modified, also because it could lead to particular and unpredictable behaviors (node with many incoming contacts and few outgoing contacts), so we wanted to leave the basic algorithm based on the degree of a node.

A very important part that has been changed is the phase of checking the edges of a possible node to be mapped, to which an additional check has been added on some characteristics derived from the arc (very similar to the calculation of the time structure, but actually more focused to the structure of the individual edges with respect to the one taken as input, which should be the arc to be mapped, which must therefore give the same type of derived alternative time structure), the functionality to be changed for checking the edges (edgeCheck) performs the following function:

Be $t \in G_T$ and $q \in G_Q$ and we want to know if it is possible to map t to q, associating it with the partial match. be $M = (t1, q1), (t2, q2), \dots, (tk, qk)$ the partial mapping obtained up to the moment of checking the possible mapping between node q and candidate t.

The edgeCheck function checks that if q is connected to node q_ {i} in M, then t must also be connected with the node t_ {i} already contained in the partial mapping and mapped with the query graph node q_ {i }. If this check is true for all q_ {i} already mapped, then edgeCheck returns true, which means that node t can be mapped to node q.

So, if you want to add the time component while checking, just take the arc and check that the connected node is similar to the node in the mapping (thus comparing fingerprints or other arc-dependent temporal information).

The new temporal information to be compared is given by the arc to be mapped, which has a single time, and defines the complete structure of all possible information to see if the arc can actually be mapped to the consequent candidate (i.e. nodes can be mapped).

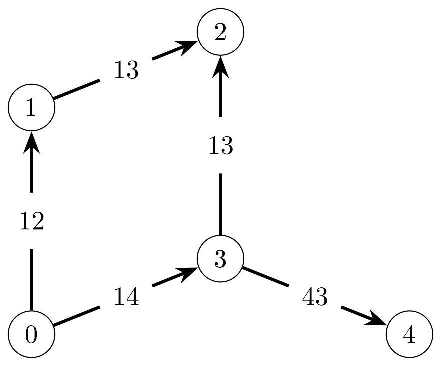
A direct fingerprint check was not used because it does not establish a well-defined temporal structure in the case in which there are only incoming nodes and only outgoing nodes.

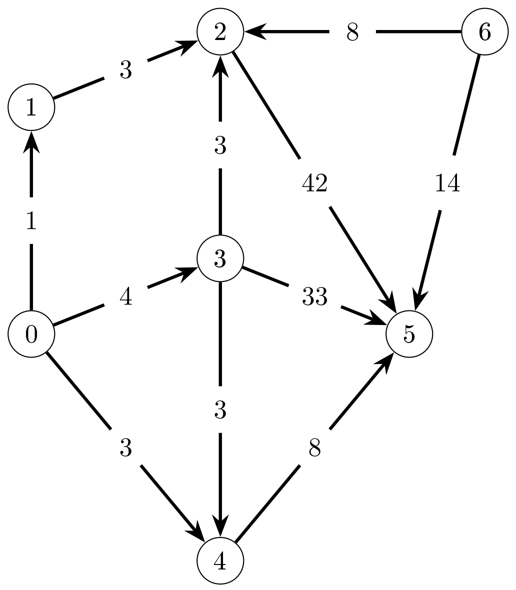
Therefore in the arc control (q, qi) it is necessary to verify that:

- (t,ti) is and edge of target graph
- Temporal costraints for (t,ti) are satisfied.

The implementations of everything that has been described and of this control in particular are available online (the code is available in both Java and Scala in my github `https://github.com/josura/university-sad/tree/master/tesi/realtesi`).

Example 37. In this example we see an instance of isomorphisms on temporal subgraphs, the query and target graph are as follows:





The query node compatibility domains are as follows:
$Dom(0) = \{0, 6\}$
$Dom(1) = \{1, 4\}$
$Dom(2) = \{2, 4, 5\}$
$Dom(3) = \{3, 5\}$
$Dom(4) = \{1, 2, 3, 4, 5\}$
A $\delta = 10$ was chosen and the classes of a propagation have been defined have been defined as $notTimeRespecting, \delta - TimeRespecting, \delta - notTimeRespecting$
The time stamps of each node of the query temporal graph are as follows:
$fingerprint_0 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 0)\}$
$fingerprint_1 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 1), (\delta{-}notTimeRespecting, 0)\}$
$fingerprint_2 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 0)\}$
$fingerprint_3 = \{(notTimeRespecting, 1), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 1)\}$
$fingerprint_4 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 0)\}$
Instead, the time fingerprints of each node of the target temporal graph are the following:
$fingerprint_0 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 0)\}$
$fingerprint_1 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 1), (\delta{-}notTimeRespecting, 0)\}$
$fingerprint_2 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 3)\}$
$fingerprint_3 = \{(notTimeRespecting, 1), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 2)\}$
$fingerprint_4 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 2), (\delta{-}notTimeRespecting, 0)\}$
$fingerprint_5 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 0)\}$
$fingerprint_6 = \{(notTimeRespecting, 1), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 1)\}$
During the matching phase, the possible candidates are passed, the edges and fingerprints are checked, and the mapping continues until all occurrences have been found.

In the end the number of occurrences of the query graph in the target graph in this case is 1.

2.3 Definitions

Definition 38. (Sub Graph matching on static graphs). A graph G1 is isomorphic to a subgraph of a graph G2 if and only if there is a one-to-one correspondence between the nodes of the subgraph of G2 and G1 that preserves adjacency, more formally $\exists f : V_{G1} \to V_{G2} | (s,d) \in E_{G1} \implies (f(s), f(d)) \in E_{G2}$

Definition 39. (Sub Graph matching on contact temporal graphs). A temporal graph G1 is isomorphic to a subgraph of a temporal graph G2 if and only if there is a one-to-one correspondence between the nodes of the subgraph of G2 and G1 that preserves adjacency, more formally $\exists f : V_{G1} \to V_{G2} | (s,d) \in E_{G1} \implies (f(s), f(d)) \in E_{G2}$ and if the target node to map has the same temporal structure as the query node, formally $s \equiv f(s)$

For the definition the relation of equivalence of temporal structure was exploited because the sub-graph does not have all the edges included between the nodes of the sub-graph, later a definition will be given with the similarity that removes this limit, but which must be however used only for the definition of pruning and feasibility rules, since the similarity with the node of the target graph does not guarantee the temporal equivalence between the node of the query graph and the node of the target subgraph.

Definition 40. (isomorphism between temporal graphs through equivalence and similarity). A GT1 temporal graph is isomorphic to a temporal subgraph of a GT2 temporal graph if and only if there is a one-to-one correspondence between the nodes of the temporal subgraph of GT2 and GT1 that preserves adjacency and the resulting subgraph (as a subset of the nodes and edges of the target graph) has all mapped nodes with the same time structure as those of the query graph to which they are mapped, i.e. the nodes of the query graph are similar to the nodes of the mapped target graph.

By defining the isomorphism of sub-graphs through similarity, it is possible to perform a matching without too many problems simply by comparing the fingerprints of the single nodes that are mapped.

However, it is necessary to be careful to carry out the checks on the sub-structure and on the temporal equivalence between the nodes of the sub-graph (formed by a subset of nodes and a subset of contacts) and the query graph, as already mentioned above.

Definition 41. (automorphism of a temporal graph). Given a temporal graph $G_t = (V, E)$, an automorphism of G_t is a permutation ϱ of the set of nodes of V which meets the following conditions:

- $\forall u, v \in V : (u, v, t) \in E \iff (\rho(u), \rho(v), t') \in E$
- $\forall u, v \in V : u \equiv v \iff \rho(u) \equiv \rho(v)$

In other words, an automorphism is a rearrangement of the set of nodes that preserves the structure of a graph. The result of applying an automorphism is a new graph $G_0 = (V, E)$, obtained from G by permuting the identifiers of its nodes according to ρ. It is said that G_0 is automorphic for G.

Definition 42. (Matrix of automorphisms). Given a graph $G = (V, E)$ with k nodes $v_1, v_2, ..., v_k$ and h automorphisms $\rho_1, \rho_2, ..., \rho_h$. The automorphism matrix A is a matrix containing h rows and k columns, where $A[i, j] = \rho_i(v_j)$.

The definition is the same as for a static graph, since the time component does not directly invalidate this definition.

Definition 43. (Orbits of a graph). Given a matrix of automorphisms A, for a certain column of the matrix, $A[, j]$, Its node set is a G orbit. It is denoted by $Orb(u)$ the orbit to which node u belongs.

This definition does not change for temporal graphs.

The definitions of automorphism and orbit will be used to search for symmetries that could give rise to redundancies during the matching phase.

2.4 Problems with isomorphisms and the definition of query and target graph

As already briefly described previously during the definition of the problem of isomorphisms on temporal sub-graphs, considerable problems arise in the interpretation of the query graph, which has been defined as the temporal pattern to be found in the target graph.

What to find in the case of several contacts per arc? How to do the matching and how to adapt it according to the needs? How to adapt the model in the case of intervals?

All these and other questions have answers depending on the case study you want to bring, so you have to adapt the basic model to the shape of the system you are trying to capture or simulate.

For example, the implementation adopted during this study of the research of isomorphisms on temporal graphs finds graphs that have nodes with the same temporal structure or that are similar, moreover additional classes are added for the classification of propagations that do not respect the conditions and that are greater or less than the initial contact time.

This feature can be misunderstood and not in keeping with the meaning of propagation classification given previously which only classifies in two categories, but in reality the system was designed so precisely to show how the implementation is not always the same under the basic conditions established, and that most of the time the matching framework has to be adapted to cover well the needs of the single problem, as already said many times.

In the case of a matching solution based on Time and topology together, the modification of the matching algorithm could lead to the uselessness of the pre-processing phase, in particular the ordering of the query nodes (because in some cases there would be the obligation to start from the node that has the edge with the shortest time), even if this risk is questionable and an actual implementation should be tried.

Since there is no univocal and standard definition of the problem of isomorphisms on graphs (and of techniques that can be used on temporal graphs in general), very often the authors do not define a basic framework to be used for defining the problem, but build a solution specification of the single problem according to fairly strict definitions that leave little possibility of expansion.

For the search of temporal patterns the previously defined framework may not be suitable, since it finds sub-graphs with a specific temporal structure, therefore the sub-structure must always be adapted to the unique case.

In the implementation it must be remembered that the similarity is useful for pruning the possible candidates, but for the computation of the isomorphism on a sub-graph, it is necessary to consider the temporal structure of the sub-graph only (without additional edges or nodes), then perform additional checks during the matching phase (in the adjacency check) or at the end of the matching, which can slow down the computation considerably.

In the next section we will look at the algorithms and implementation considerations for searching for isomorphisms on temporal sub-graphs.

3 Algorithms and implementation

Below are the algorithms used to calculate the basic functions that will be used for graph matching:

Algorithm 2 classifyPropagation(prop,δ)

```
    Input:  prop: propagation to classify,
            δ: conditional parameter used to classify propagations
    Output:  the type of the propagation prop
 1: function classifyPropagation(prop,δ)
 2:     if 0 <prop.Δ < δ then
 3:         return timeRespecting
 4:     else
 5:         return notTimeRespecting
 6:     end if
 7: end function
```

This function is particularly important for the calculation of the fingerprints, as it classifies the individual propagations to make them fall into the class to which they should belong.

In this case, as defined above, only two classes are used to facilitate understanding, but there are three classes defined in the code.

Basically, for each propagation passed to the command line, the conditions established by the possible classes that can be obtained are checked, then the type of class it belongs to is returned. To expand this function with other well-defined cases and classes, just add the additional conditions and conditionals, implementing them through a switch case, or treating the propagation as an object and assigning a state (the class it belongs to) every time a propagation is created.

This function is very simple and could be directly substituted for the parts of code that require it, but it has been left independent given its nature dependent on definable classes and for refactoring since it is also used in other parts of the code (which are not seen given which are not used for actual matching).

The last possibility was not implemented since we did not use the actual propagations but pairs of edges, and keeping additional structures involved relatively redundant extra space, and since the classification does not carry out complex calculations but simple decisions this was left a choice that adds almost nothing to the actual complexity of the problem.

Note that the function works for contacts, but does not distinguish between multiple contacts of a single arc or different edges, so it is independent of the contact temporal graph used.

The case of the interval graph is slightly different, but the principles are in any case those described previously during the definition of the problem.

In the case of an interval temporal graph, the algorithm is the following:

Algorithm 3 classifyPropagationInterval(prop,δ)

```
Input: prop: propagation to classify,
       δ: conditional parameter used to classify propagations
Output: the type of the propagation prop
1: function classifyPropagationInterval(prop,$\delta$)
2:     if prop.$interval \neq \emptyset$ then
3:         return timeRespecting
4:     else
5:         return notTimeRespecting
6:     end if
7: end function
```

Notice how it is possible to make a simple comparison with the empty set to see if the propagation is actually Time-Respecting or not, this only thanks to the definition of interval in the propagation, which incorporates the recursive intersection with the different propagations (which however remains ambiguous but this factor will be overlooked as this study focuses on a one-on-one implementation).

Here is the function that uses the classification of a propagation (by contact graph) to calculate the time fingerprint of a node:

Algorithm 4 temporalFingerprint(G_t,node,δ)

```
Input: $G_t$: Temporal Graph of the node,
       node: is the node to test,
       δ: is the conditional parameter used to classify propagations
Output: the Temporal fingerprint of the node in the Temporal Graph
1: function temporalFingerprint($G_t$,node,$\delta$)
2:     $fingerprint \leftarrow vector[2]$
3:     for $prop \in G_t.P_{node}$ do
4:         if classifyPropagation(prop,$\delta$) = timeRespecing then
5:             fingerprint[1] ← fingerprint[1] +1
6:         else
7:             fingerprint[2] ← fingerprint[2] +1
8:         end if
9:     end for
10:    return fingerprint
11: end function
```

The code is substantially the same both in the case of contact graph and in the case of propagation intervals.

Ambiguity could arise, however, in the case of a graph with more contacts per arc, a case already discussed above, in which there are two possibilities: either consider the propagations as independent, or propagations with the same destination node and source node will end up in different fingerprints, therefore combinations between the same and different propagations.

In the first case, ambiguities could come out since a single time stamp that incorporates information from different patterns for the same edges could lead to false positives, i.e. nodes whose edges are greater than those of the query node, but having the same final fingerprint.

The second case should be the safest one, but also the most complex algorithmically, since it would be necessary to make the combinations between the various edges on the basis of the contacts of each arc (returning a set of time stamps of all the combinations), then compare all the possible permutations and see if they are similar.

This case will not be dealt with since the study focuses on single contacts, but in the future we will see the possible implications and uses of temporal graphs at multiple intervals, so as to have more information on the possible most common needs that can be met with possible implementations. and definitions.

To check two temporal structures and verify the similarity, it is sufficient to perform a vector operation on the components of the fingerprint of the query node q and the target node t, thus checking that it is

$$result = temporalFingerprint(q) \overset{component-wise}{\leq} temporalFingerprint(t) | \\ \forall i \in [1, result.size], result[i] = true$$

otherwise the condition is not verified and the node is not similar.

Here is the algorithm that is called to perform the matching between temporal graphs:

Algorithm 5 TemporalRI(G_{query},G_{target},δ)

```
Input:  $G_{query}$: query Temporal Graph,
        $G_{target}$: target Temporal Graph,
        $\delta$: is the conditional parameter used to classify propagations
Output: number of occurrences of Query in Target
1: function TemporalRI($G_{query}$,$G_{target}$,$\delta$)
2:     $Dom \leftarrow$ ComputeDomains($G_{query}, G_{target}$,$\delta$)
3:     $\mu \leftarrow$ OrderQueryNodes($G_{query}, Dom$)
4:     $\varsigma \leftarrow$ ComputeSimmetryBreakingCond($G_{query}$,$\delta$)
5:     $Matches \leftarrow$ SubgraphMatching($G_{query}, G_{target}, Dom, \mu, \varsigma, \delta$)
6:     return $Matches$
7: end function
```

Not all the algorithms used are seen specifically (the pseudocode is available in the Appendix of[9]) since the changes are minimal or nil for many parts of the code, the only parts where something has actually changed are the computation of the compatibility domains, the computation of automorphisms during the search for symmetries and a part of the code during the of matching in which the edges between the nodes already mapped and a candidate are checked (checking the similarity).

The latter case will not be reported as a pseudocode because the change is minimal and concerns a control of the temporal structure starting from the arc that is being mapped with respect to the candidate and to the node already mapped (there is however the usual control of the similarity of the temporal fingerprints , the algorithm has already been described).

Below is the code for the calculation of symmetry breaks:

Algorithm 6 ComputeSimmetryBreakingCond(G_{query},δ)

```
    Input:  $G_{query}$: query Temporal Graph,
            $\delta$: is the conditional parameter used to classify propagations
    Output: set of symmetry breaking condition of query graph
 1: function ComputeSimmetryBreakingCond($G_{query}$,$\delta$)
 2:     $\varsigma \leftarrow \emptyset$
 3:     $Aut_{|\varsigma} \leftarrow$ ComputeAutomorphismMatrix($G_{query}$,$\delta$)
 4:     $Orb_{|\varsigma} \leftarrow$ ComputeOrbits($Aut_{|\varsigma}$)
 5:     while $|Aut_{|\varsigma}| > 1$ do
 6:         $q' \leftarrow argmin_{q \in V_Q : |Orb_{|\varsigma}| > 1} \{id(q)\}$
 7:         for $q \in V_Q | q \neq q' \cap Orb_{|\varsigma}(q) = Orb_{|\varsigma}(q')$ do
 8:             $\varsigma \leftarrow \varsigma \cup \{q' \prec q\}$
 9:         end for
10:         $Aut_{|\varsigma} \leftarrow \{Aut_{|\varsigma}[i] | \rho_i(q') = q'\}$
11:         $Orb_{|\varsigma} \leftarrow$ ComputeOrbits($Aut_{|\varsigma}$)
12:     end while
13:     return $\varsigma$
14: end function
```

A possible optimization for both single contacts and multiple contacts or multiple intervals per arc is to perform a binary search on contact times to speed up the similarity and rank check.

The algorithm for the calculation of symmetry breaking conditions is based on an iterative calculation of the query and orbit automorphisms starting from the current set of symmetry breaking conditions discovered by the algorithm. Given a set of failure conditions C, we denote with $Aut_{|\varsigma}$ and $Orb_{|\varsigma}$ the automorphism matrix and the set of orbits that respect the failure conditions in C.

When C is empty $Aut_{|\varsigma}$ corresponds to the automorphism matrix of Q and $Orb_{|\varsigma}$ is the corresponding set of orbits. A breaking condition $q_0 \prec q$ in C, prevents that q_0 be mapped to q in any automorphism. Therefore, $Aut_{|\varsigma}$becomes the matrix of automorphisms obtained from the set of all automorphisms of the where graph q_0 is not mapped to q, for all nodes q_0 and q such that $q \prec q_0 \in C$. $Orb_{|\varsigma}$ is the set of orbits related to $Aut_{|\varsigma}$. The first step of the algorithm is the computation of the Q automorphism matrix.

This can be achieved using any graph matching algorithm. Next, the orbits of Q are computed and the query graph node is computed q_0 with minimum id on all orbits with at least two equivalent nodes. For each node $q \neq q_0$ belonging to the same orbit as q_0, a new symmetry breaking condition is defined $q_0 \prec q$ which is added to the final set of conditions. This step is equivalent to preventing the node q_0 be mapped to any other node and put q_0 in a separate orbit. To be consistent with this, we must keep from the current automorphism matrix only the rows corresponding to the automorphisms that map q_0 to itself and update its orbit set $Orb_{|\varsigma}$.

The pseudocode for computation of compatibility domains is shown below:

Algorithm 7 ComputeDomains(Q,T,δ)

```
    Input:  $Q$: query Temporal Graph,
            $T$: target temporal graph
            $\delta$: is the conditional parameter used to classify propagations and for the similarity
    Output:  set of compatibility domains of nodes in $G_{query}$
 1: function ComputeDomains($Q$,$T$,$\delta$)
 2:     for $q \in V_Q$ do
 3:         $Dom(q) \leftarrow \emptyset$
 4:     end for
 5:     for $t \in V_T$ do
 6:         for $q \in V_Q$ do
 7:             if $deg(q) \leq deg(t) \wedge q \simeq t$ then
 8:                 $Dom(q) \leftarrow Dom(q) \cup \{t\}$
 9:             end if
10:         end for
11:     end for
12:     for $q' \in V_Q$ do
13:         for $t' \in Domq'$ do
14:             for $q'' \in V_Q | (q', q") \in E_Q$ do
15:                 if $\nexists t'' \in Dom(q'') | (t', t") \in E_T$ then
16:                     $Dom(q') \leftarrow Dom(q') \setminus \{t'\}$
17:                 end if
18:             end for
19:         end for
20:     end for
21:     return $Dom$
22: end function
```

As you can easily see, a target graph node enters the candidate domain of a query graph node only and only if it has the same rank and if the query node is similar to the target node.

It is also seen the minimum requirement already described of the Arc Consistency, it is then seen if there is an arc between two nodes of the query graph, then for each node of the source domain it is checked if there is at least one arc towards the nodes in the destination domain.

The δ parameter seems not to be used, but is actually used when calculating the similarity between the two nodes (for computation of fingerprints).

The modified Match algorithm to integrate the temporal component is briefly described below.

The core of the matching algorithm is the recursive MATCH procedure. Let M be the partial match found and q a node of the query temporal graph that has not yet been mapped. If $q_0$is the node preceding q in the order μ, the set $Cand(q)$ of candidate target nodes to match to q is given by $Neigh(f(q_0)) \bigcap Dom(q)$, the set of nodes that are adjacent to the target node that has already been matched to q_0 and are in the compatibility domain of q.

If q is the first node in μ,the set of candidate nodes is just the compatibility domain of q. If some feasibility rules are satisfied (including the similarity between the nodes and the additional control on the associated edges), RI adds the pair (q, t) to the partial match M and updates the mapping function f. When all the nodes in the query have been mapped, a new occurrence of Q in T is found and the matching match is added to the list of matches found. Whenever all query temporal graph nodes have been mapped or no match is found for a query graph node, the algorithm backtracks and continues the search from the last (previously) matched node. Backtracking involves removing the last pair of matching queries and target nodes from M and mapping between those nodes using f. When no more matches can be created for any query node, TemporalRI stops. At the end of the matching process, the algorithm returns the list of all matches found.

Furthermore, since the algorithm checks the similarity between nodes using the complete target graph, it is necessary to check at the end that the query graph and the subgraph (without additional edges compared to the necessary ones), are isomorphic, in particular each node has the same structure temporal (and that it is no longer just similar).

Eventually you will have all isomorphic subgraph matches to the query graph.

3.1 Complexity analysis

Spatial and temporal complexities are analyzed, starting with the analysis of algorithms and exploring alternatives both as a change of code and functionality, and as a different use of additional structures, for example by adding parameters to the definition of a temporal graph.

Ben_Q the number of nodes of the query temporal graph, ed n_T the number of nodes of the destination. The first step of TemporalRI is the computation of the compatibility domains. Building domains requires $O(n_Q n_T)$, while the AC procedure requires $O(n_Q^2 n_T^2)$ because, in the worst case, for each arc out of the query from q_0 you have to check all outgoing arcs from destination nodes in $Dom(q'')$. Hence, computation of compatibility domains requires summarily $O(n_Q^2 n_T^2)$. Then, TemporalRI computes the ordering of the query graph nodes for the matching process. The important part of this step is building the sets $V_{q,vis}$,$V_{q,neig}$ and $V_{q,unv}$, that requires $O(n_Q^2)$.

The third phase of TemporalRI is the calculation of the symmetry breaking conditions. The computation of the automorphism matrix for the query is at least as difficult as the resolution of the graph's isomorphism and its complexity is limited by $O(2^{n_Q})$. The orbits are calculated by scanning the automorphism matrix column by column. Since the number of automorphisms of a graph with n nodes is at most n! (in case the graph is complete), the calculation of the orbits requires $O(n_Q!n_Q)$. In the worst case, at each step of the while loop the number of automorphisms only decreases by one and the loop is executed $n_Q!$ times. Hence, the calculation of the failure conditions requires $O(n_Q!2n_Q)$.

The matching process will not be commented specifically, but it is enough to know that the calculation of the similarities between nodes adds nothing in the asymptotic analysis, which remains the same as that already described in [1, 9],moreover, if the temporal structures of the single nodes are kept directly as an updated structure, the control becomes a simple comparison.

Regarding the spatial complexity of TemporalRI, given the possibility of having a contact for each arc, the query and target temporal graph require $O((2n_Q)^2)$ and $O((2n_T)^2)$ space in the worst case, respectively, so they always remain in the order of the quadratic of RI. The additional data structures used by the algorithm when searching include the set of compatibility domains for each query node (space $O(n_Qn_T)$ in the worst case), the set of symmetry breaking conditions for the query $O(n_Q^2)$ space), the mapping and partial matching (both require a space$O(n_Q)$) and the set of target nodes candidates for matching for each query node $O(n_Qn_T)$. Therefore, the total spatial complexity of TemporalRI is $O((2n_Q)^2)+O((2n_T)^2)+2*O(n_Q))+2*O(n_Qn_T)+O(n_Q^2)$. Since very often the query graph is much smaller than the target graph ($n_Q << n_T$),the resulting complexity will be asymptotically dominated by $O(n_T^2)$, ie the space needed to store the target temporal graph.

However, additional structures or variables can be added, such as the set of propagations which would add an additional spatial complexity of $(O(|inedges(Q)|* |outedges(Q)|) + O(|inedges(T)| * |outedges(T)|))$ that it is absolutely not advisable to keep given its situation and easy derivability, or the fingerprint (or the set of fingerprints in the case of several contacts per arc) of the single node, which is updated with each addition of an arc entering or leaving a node $(O(n_T) + O(n_Q))$,and thanks to which it is possible to avoid the continuous calculation of the fingerprints to carry out a direct control on the auxiliary structures.

4 Conclusions

The development of temporal networks is particularly difficult because either we lack the basic measures and the knowledge necessary to model real networks, or we simplify the analysis itself too much.

When common conventions are established that are of considerable importance for enough real networks and do not lose relevance with the change of the basic structure (in short, they are independent of chance and common to all problems that incorporate the temporal component) you may be able to simulate reality more accurately. The final problem, however, remains that of minimizing entropy and maximizing information.

As we have seen several times, very often the problem of time networks is that of adapting the infrastructure to the individual case, according to the needs.

Very often this is exploited by other authors to create their own solutions to the problem, thus creating very selective and not very general solutions, without defining a real solid infrastructure, but very often confusing ideas with definitions that are too simple to fully describe the temporal component.

Also thanks to this trend, many authors end up taking as an example the single specific solutions for the single problem in a superficial way, adapting the solution seen and explained in other studies for their own case, thus managing to define a fairly confusing infrastructure with respect to the final goal that we wanted to achieve.

Some authors, on the other hand, have focused on a definition of a general framework that can be used in many contexts, without being heavily influenced by bias towards individual cases, but taking the descriptions and solutions adopted by other studies and integrating a general explanation capable of allowing the construction of a general framework that can be adapted to the individual case[2].

In this study it was decided to adopt the same approach, not focusing on a single problem, but first defining a flexible basic structure, and then adapting it for the definition of the problem (which in this case was the Temporal Sub-Graph Isomorphism), in order to bring a real example of modeling a solution starting from clear and concretely useful definitions in most cases.

After the definition of the basic framework, the Temporal Sub-Graph Isomorphism problem was defined by introducing it first with a description of the basic problem in static graphs with a possible solution that uses a fast algorithm based on heuristics useful for reducing the space of research (RI), and expanding the concept to temporal graphs in order to have a solid base that can be expanded easily thanks to the definitions created during the description of the basic structure and operations in time networks.

We have seen some definition problems in the description of the query and target temporal graph, since the search for the structure is totally dependent on the problem you are trying to solve, that is the type of structure you are looking for.

A complexity analysis was carried out which showed how the basic problem is actually not easy to solve and at the same time very dependent on the variables that are passed (i.e. query graph and target), and it will be seen in experimental data such as this statement is substantially true since execution times very often depend on the basic structure of temporal graphs used as variables.

An additional consideration must be made on the performance analysis which will be seen below and which are very important in many studies. In particular, particular attention must be paid to the implementation itself and the methods used.

Almost all the algorithms use technologies designed to be implemented on a single core, only some use a multithreading that exploits a parallelization of the instructions, thus increasing the overall performance of the algorithm, but none uses specific methods actually designed to be normally translated in a computation aimed at parallelization.

None of the authors mention distributed computing technologies like SPARK, but this is quite understandable as the technology is quite modern.

Furthermore, the possibility of exploiting the parallelization of computation and data in order to use the machine to the maximum is absolutely not mentioned, for example using OpenCL or CUDA for writing programs usable in GPU or FPGA, which would exploit the basic parallel nature of many of the algorithms used (which is however based on a concept of dynamic programming, where partial solutions can be used to arrive at global solutions).

All the possible implementations presented are therefore blocked from being single thread (in most algorithms), or multithreaded optimized for a single task (the case of SNAP for the search of temporal motifs).

In conclusion, this research and construction of a new general infrastructure easily adaptable to individual cases was made on the principles of local similarity, in order to exploit the subproblems, and build a global solution in a simple way by adapting an algorithm that is not easy to modify. In contrast, many authors use global definitions that do not consider concepts such as independence between groups of interactions (defining orderings on the whole graph), or establishing mesoscopic constraints on paths and subgraphs that must be respected (which always depend on the case that you are studying and simulating).

We will see in the next subsection how the execution trend depends not on the quantity of nodes in a graph, but on the contact-edges, which define the temporal structure and directly influence the quantity of nodes to be analyzed (which belong to the compatibility domains) and the control used in the matching phase.

4.1 Experimental data

The graphs that represent the performance as the size and structure of a temporal graph increase are presented below.

It should be noted that this study was not focused on an implementation that exploits the hardware in an optimal way, but on the definition of the general problem and the strategies to solve it, therefore the performance of many general algorithms (Mackey and TemporalRI) is very unsatisfactory, and in fact, particularly complex graphs were not chosen.

Other specific algorithms for single problems where the implementation has been taken care of in order to exploit a minimum of parallelism (such as SNAP's temporal-motif) will obviously be optimized for the tasks for which they were created, and therefore faster than the algorithms presented.

For the construction of the target graphs the intention was to use a temporal graph generator based on two distributions, one for the possible edges (which should be a Poisson for the degree of a node, therefore a Barabasi-Albert model based on preferential attachment can be fine), and one of the possible contact times (which actually establishes interval fashions where contacts can occur, so it is not easy to define), in the end we opted for the use of graphs slightly modified time frames already available (taking the subgraphs) to facilitate the search for occurrences and not significantly increase complexity, and for the construction of the synthetic target graphs a very simple Barabasi model was used slightly modified to add contact times to each added arc .

For the construction of the query graph, graphs created ad-hoc were also used to simulate a real procedure, where you want to find specific patterns. Query graphs of increasing size (one with 4 nodes, one with 16 nodes, one with 32 and the last at 50 nodes) were tried, the edges were created by hand.

In the following we will consider only the cases of 32 and 50 node query graphs, since it is the heaviest computationally.

A comparison with the algorithm used in the [8] modified since it did not find the same occurrences of the algorithm used in this study and also because it was not very efficient, even after the modification it does not find the same occurrences, but at least it works for fairly complex graphs.

The basic implementation of this algorithm can be downloaded on github (`https://github.com/pnnl/temporal_subgraph_isomorphism`). No other algorithms were found that could be used, functioning or modifiable to make a complete comparison, moreover the results could not be those seen but slightly different (since the experiment period was around August and the temperature prevented the normal functioning of the available devices) , however, the relationships between timing and performance should be the same.

A comparison of SNAP and its implementation of searching for temporal motifs and the TemporalRI algorithm was then made, using the defined queries that are used in SNAP (i.e. 2-3 nodes and 3 edges), within the limits of individual possible contacts in the implementation made for TemporalRI.

Individual measures are presented for each variable because during some tests the execution time did not seem to fluctuate particularly (also because the procedure is not very variable in all the algorithms).

An Acer Nitro 5 AN515 was used during the experimental tests.

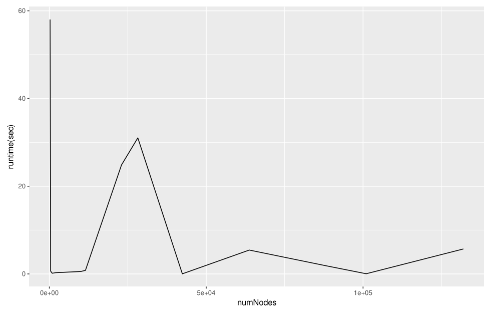
The target graphs taken are relatively few also because the execution time did not vary particularly with some variables (the nodes of the target graph), while with the variation of the edges in the target graph it seems to follow an increasing trend (with a correlation quite tending to 0 because in any case depends on the structure of the whole graph and how the edges are distributed, as well as the trend of the time structure).

Also, the execution times for certain types of query and target graphs were long enough to crash my system.

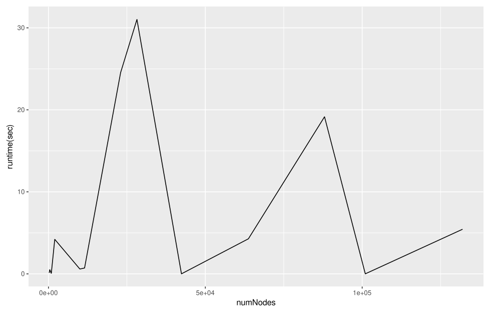
The following graphs present the general preliminary run times of the algorithm on finding query temporal graphs defined on real target temporal graphs.

The datasets used for these tests are not the same as in Table 1, but were lost during the tests as these experiments were among the first for performance control.

These graphs were left because it can be seen that the execution times are relatively low, this is because the graphs used came from networks that were not strictly temporal but were given for recommendation systems with a type of temporal component that did not convince, and given that the times fell within a minimum time interval, the resulting compatibility domains were very small and the computation of the actual matching was heavily affected.

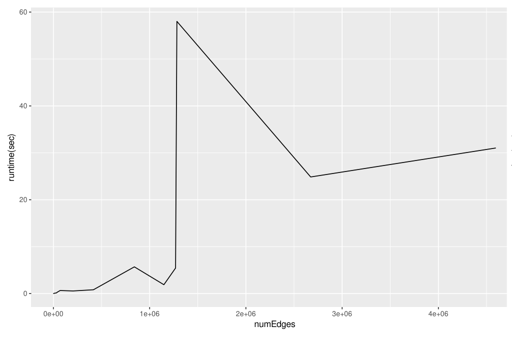


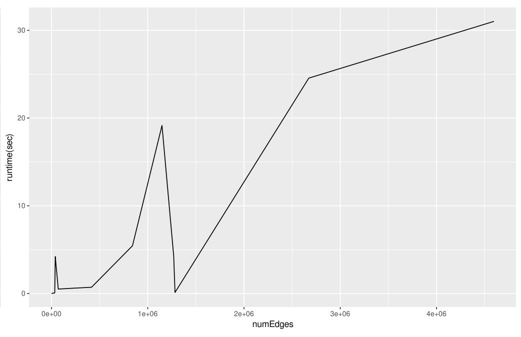
Query of 50 nodes



Query of 32 nodes

The number of nodes will not seem to directly affect the execution time, since it does not directly affect the search for possible matches.





Query di 50 nodi Query di 32 nodi

In the case of the correlation with the increase of edges in a graph, it seems instead that the algorithm is affected by the greater computational load in the control of the various edges, therefore the execution time assumes an increasing trend as the number of contacts in a temporal graph increases.

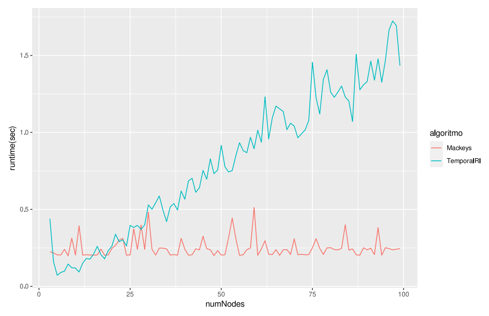
An additional consideration can be made on the fact that the query graphs used were created ad-hoc, this is because during other experiments with query graphs extracted from other temporal graphs, the execution times increased or decreased without a stable pattern, also because, in the in most cases, these depend heavily on the time structure of both the query graph and the target graph.

Comparisons between TemporalRI and the algorithm used in [8] in synthetic and real graphs, towards the end a comparison with SNAP of TemporalRI with real temporal graphs is also presented.

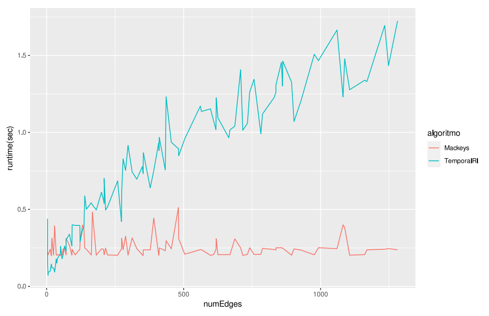
4.1.1 Performance over synthetics graphs

The synthetic graphs used are, in the case of query graphs, sub-graphs of other time networks (100 sub-graphs of the fb-forum network, with nodes from 3 to 103), and in the case of the study of the variation in performance based on dimension of the target graph, are temporal graphs generated with a basic Barabasi model (i.e. exploiting the principle of preferential attachment) modified for the addition of the temporal component, which is established based on when the node enters the network (to simulate the entry of an agent and the interaction between other agents in a certain time), the average grade is approximately 3.

Below is the comparison based on the number of nodes and contacts in the query (and not in the target graph), again by TemporalRI with Mackey:



Incremental Query over number of nodes



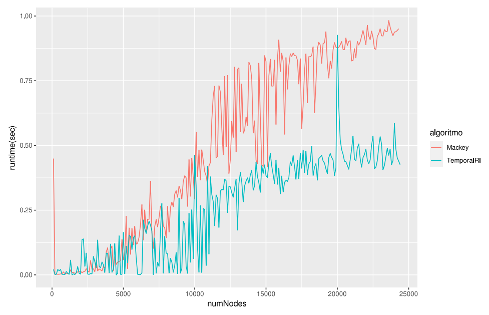
Incremental Query over number of edges

As you can see, Mackey's algorithm is better in almost all cases since it maintains constant execution times regardless of the number of nodes or edges of the query graph, where instead TemporalRI seems to assume linear behavior.

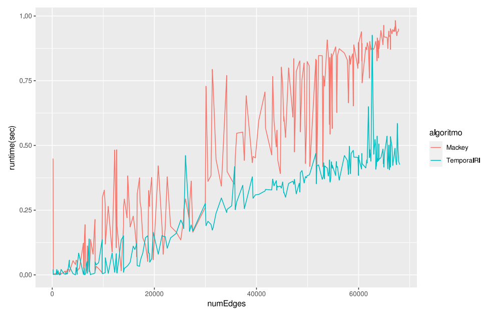
It is also presented a comparison of the execution times of the search for isomorphisms of a defined graph (with 10 nodes) in target graphs with variable nodes or edges (according to what previously explained), the results below:

Although TemporalRI seems to prevail in performance, in reality the two algorithms have very similar times, and on average it takes the same time to search for isomorphisms-motifs.

From this result it can be seen that the edge-driven approach adopted by many authors is not always the best choice since, in addition to the fact that only certain sub-graphs can be found without counting certain occurrences dependent on the structure defined by the problem (which it is not always the same with time-respecting paths), one has to deal with the increasing number of edges to be compared, which is buffered by the approach used in RI with the preventive



Incremental Target over number of nodes



Incremental Target over number of edges

control of the fingerprints and during the matching phase to remove candidates who do not respect certain rules.

4.1.2 Performance over real graphs

During this subsection, real temporal graphs found in `http://networkrepository.com/temporal-networks.php`.

The dataset used are below

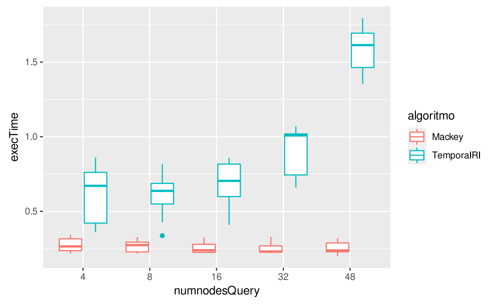
Graph	|V|	|E|	d_{max}	d_{avg}
SFHH-conf-sensor	403	70K	2K	348
ca-cit-HepPh	28K	5M	11K	327
copresence-InVS15	219	1M	40K	12K
edit-enwiki-books	8K	162K	11K	38
email-dnc	2K	37K	6K	40
fb-forum	899	34K	2K	74
fb-wosn-friends	64K	1M	2K	39
ia-contacts-dublin	11K	416K	616	75

Table 1: dataset utilizzati

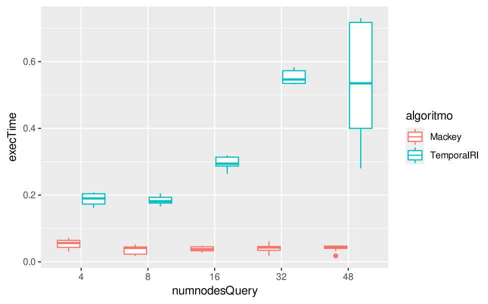
As will be seen from the next experimental data, the number of nodes will not seem to directly affect the execution time, and although the algorithm compared with RI is quite underperforming for certain types of temporal graphs, it appears to be faster than TemporalRI for temporal graphs. small and with few contacts, and very often the two algorithms have the same execution times, considering the limits of possible edges.

During the experiments TemporalRI will be better in the case in which the edges of considerable quantity, since the comparison is not edge-driven like the Mackey algorithm, besides giving the right result based on the definition given during this study, these results do not they will still be very reliable, given that the graphs used as targets are extremely variable, and very often the contacts that are provided all occur in a very small interval, so TemporalRI performs a computation of the compatibility domains that completely destroys the structure used by Mackey for the motif search, and at the same time the Mackey algorithm sifts the contacts in order to perform the search in a performing way.

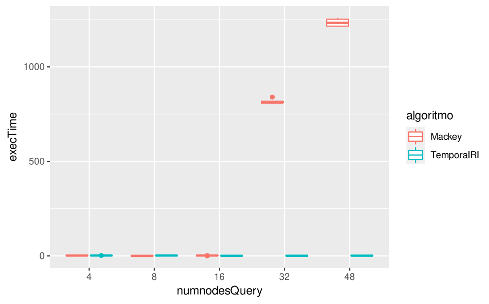
A boxplot is made based on the number of nodes of multiple queries, in order to analyze the behavior of the two algorithms on multiple datasets (some of those specified previously) and to vary the query graphs:



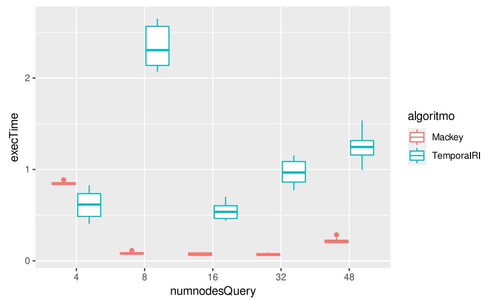
Boxplot 1:dataset edit-enwikibooks



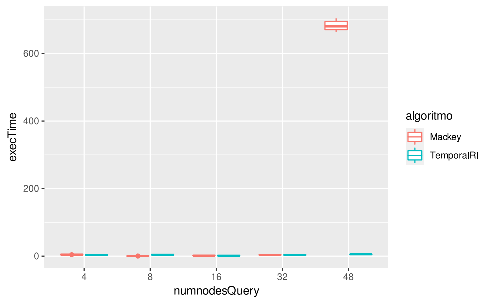
Boxplot 2:dataset fb-forum



Boxplot 3:dataset SFHH-conf-sensor



Boxplot 4:dataset email-dnc

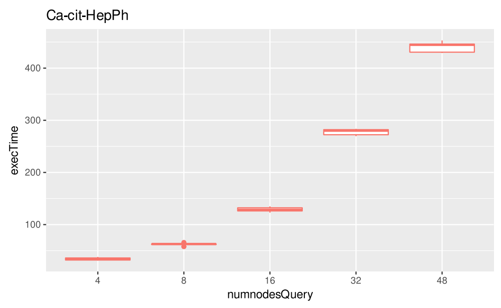


Boxplot 5:dataset ia-contacts-dublin

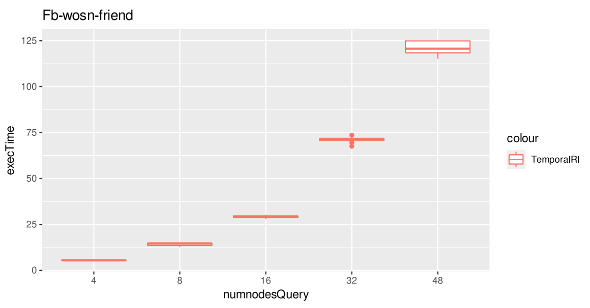
Mackey's algorithm appears to be faster in most experiments, but sometimes it assumes unexpected behaviors with no apparent criteria (for example by increasing the number of nodes, the algorithm becomes immensely slower).

These results are also implemented on fairly small graphs, in fact both algorithms finish the computation within a few seconds. It was not possible to use larger datasets as Mackey's algorithm has quite naive memory management, and for relatively large graph sizes (above 300,000 edges), the algorithm takes up all memory and eventually commits suicide.

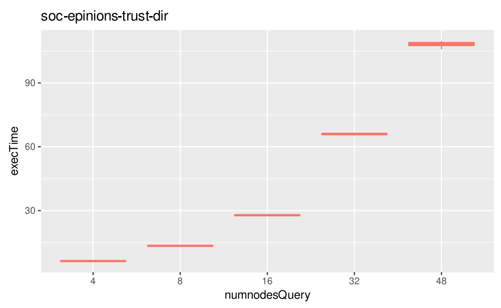
The boxplots of real graphs are presented where Mackey's algorithm went out of memory:



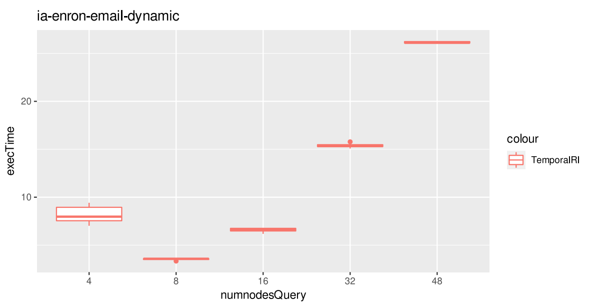
Boxplot 6:dataset Ca-cit-HepPh



Boxplot 7:dataset fb-wosn-friend

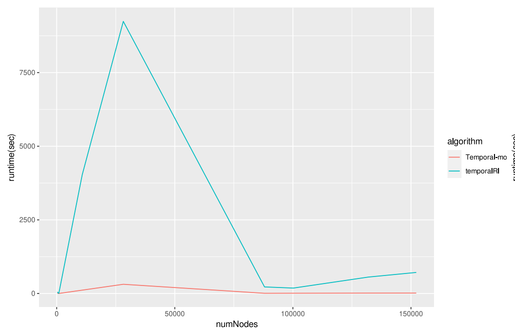


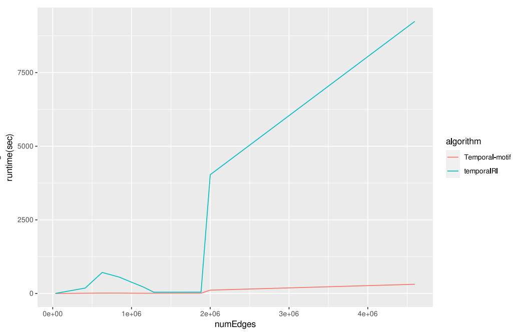
Boxplot 8:dataset soc-epinions-trust-dir



Boxplot 9:dataset ia-enron-email-dynamic

Below is the comparison between the execution times of SNAP for searching for temporal motifs (called Temporal-motif in the graph) and TemporalRI, in which all the execution times of all the queries for a single target graph are considered (represented by the number of nodes or contacts belonging to the temporal graph) since SNAP does not allow to know the single execution times for each query graph, the Mackey algorithm has not been compared since for graphs of considerable size it does not seem to work (with my hardware), the datasets used are always those defined in Table 1.





Obviously, SNAP's temporal motif search algorithm is faster for any input, since it was created for the single task of finding all occurrences of 2-3 nodes and 3 edges, as well as taking advantage of OpenMP for the parallel use during computation, and multithreading.

In the end you can see how an optimized edge-driven comparison that also uses edge pruning and control parallelism strategies is better than a normal structure-based search algorithm that does not use data and task parallelism, therefore not optimized to use very components. useful to decrease the execution time.

However, it must be remembered that the algorithms presented do not give the same results (quantity of match-motif) also because they use different definitions of isomorphism or motif as well as having quite fluctuating times with respect to the basic temporal structure of both the query time networks used and the targets. where to search for isomorphisms.

5 Future

For the future there is a clear definition of a basis of formality for isomorphisms for multiple intervals for each arc, since it is much more complicated and dependent on what has been exposed previously.

Since the study was not focused on performance, the algorithm was not written in an optimal way to increase its performance, so in the future it is possible to check the time structure and equivalence directly during the matching and control phase, given that on the date this research was written, the time structure check is done at the end of the comparison on the subgraph to see the equivalence.

In addition, the code was also written to scale to facilitate a possible implementation with SPARK and the transposition into the world of stream processing or distributed computation of graphs with libraries such as Neo4j or GraphX.

Possible implementation that takes advantage of the dynamic and parallel nature of the problem using GPGPU technologies such as OpenCL or CUDA, increasing the overall throughput of the algorithm.

An unvisited possibility is the implementation of temporal graphs together with Markov models or other statistical models in order to model the change of states and links in a purely statistical way, creating a more complete and hybrid basic structure.

Nomenclature

p_{itj}	(e_{it},e_{tj}) propagation from node i to node j through node t
P_t	set of propagations of a node t
E	set of edges of a graph
G	(V,E) static graph
GT	(V,E) temporal graph
V	set of nodes of a graph

References

[1] Vincenzo Bonnici, Rosalba Giugno, Alfredo Pulvirenti, Dennis Shasha, and Alfredo Ferro. A subgraph isomorphism algorithm and its application to biochemical data. *BMC bioinformatics*, 14(S7):S13, 2013.

[2] Petter Holme and Jari Saramäki. *Temporal Network Theory.* Springer, 2019.

[3] Ali Jazayeri and Christopher C Yang. Motif discovery algorithms in static and temporal networks: A survey. *arXiv preprint arXiv:2005.09721*, 2020.

[4] Lauri Kovanen, Márton Karsai, Kimmo Kaski, János Kertész, and Jari SaramÃ€ki. Temporal motifs in time-dependent networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(11):P11005, nov 2011.

[5] Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, 8(1):61, 2018.

[6] Paul Liu, Austin R Benson, and Moses Charikar. Sampling methods for counting temporal motifs. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 294–302, 2019.

[7] Penghang Liu, Valerio Guarrasi, and A Erdem Sarıyüce. Temporal network motifs: Models, limitations, evaluation. *arXiv preprint arXiv:2005.11817*, 2020.

[8] Patrick Mackey, Katherine Porterfield, Erin Fitzhenry, Sutanay Choudhury, and George Chin. A chronological edge-driven approach to temporal subgraph isomorphism. *mackey2018chronological*, pages 3972–3979, 2018.

[9] Giovanni Micale, Vincenzo Bonnici, Alfredo Ferro, Dennis Shasha, Rosalba Giugno, and Alfredo Pulvirenti. Multiri: Fast subgraph matching in labeled multigraphs. *arXiv preprint arXiv:2003.11546*, 2020.

[10] Majid H Mohajerani, Allen W Chan, Mostafa Mohsenvand, Jeffrey LeDue, Rui Liu, David A McVea, Jamie D Boyd, Yu Tian Wang, Mark Reimers, and Timothy H Murphy. Spontaneous cortical activity alternates between motifs defined by regional axonal projections. *Nature neuroscience*, 16(10):1426, 2013.

[11] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 601–610, 2017.

[12] Ursula Redmond and Pádraig Cunningham. Subgraph isomorphism in temporal networks. *arXiv preprint arXiv:1605.02174*, 2016.

[13] Xiaoli Sun, Yusong Tan, Qingbo Wu, Jing Wang, and Changxiang Shen. New algorithms for counting temporal graph pattern. *Symmetry*, 11(10):1188, 2019.