UPPSALA
UNIVERSITET

UPTEC STS 18009
Examensarbete 30 hp
Mars 2018

Partitioning temporal networks

A study of finding the optimal partition of temporal networks using community detection

Axel Lindegren

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Partitioning temporal networks

Axel Lindegren

Many of the algorithms used for community detection in temporal networks have been adapted from static network theory. A common approach in dealing with the temporal dimension is to create multiple static networks from one temporal, based on a time condition. In this thesis, focus lies on identifying the optimal partitioning of a few temporal networks. This is done by utilizing the popular community detection algorithm called Generalized Louvain. Output of the Generalized Louvain comes in two parts. First, the created community structure, i.e. how the network is connected. Secondly, a measure called modularity, which is a scalar value representing the quality of the identified community structure. The methodology used is aimed at creating a comparable result by normalizing modularity. The normalization process can be explained in two major steps: 1) study the effects on modularity when partitioning a temporal network in an increasing number of slices. 2) study the effects on modularity when varying the number of connections (edges) in each time slice. The results show that the created methodology yields comparable results on two out of the four here tested temporal networks, implying that it might be more suited for some networks than others. This can serve as an indication that there does not exist a general model for community detection in temporal networks. Instead, the type of network is key to choosing the method.

Handledare: Matteo Magnani
Ämnesgranskare: Georgios Fakas
Examinator: Elísabet Andrésdóttir
ISSN: 1650-8319, UPTEC STS 18009

Populärvetenskaplig sammanfattning

Inom nätverksteori är det av intresse att hitta så kallade *communities* (grupperingar eller kluster). De definieras vanligen som en samling aktörer inom nätverket som är starkare kopplade till varandra än vad de är till andra aktörer i nätverket genom så kallade kontakter (*edges*). Målet med att hitta grupperingar är kunna göra observationer om det studerade nätverket, exempelvis identifiera kärngrupperingar (*community detection*). Inom sociala nätverk, till exempel Facebook-nätverk, kan det exempelvis vara av intresse att se hur information sprids. För att identifiera detta krävs då att man kan finna de specifika grupper som agerar och framförallt skapar och sprider information. Tidigare har man vanligtvis modellerat sociala nätverk med hjälp av så kallade statiska nätverk. Dessa förändras inte över tid vilket leder till att identifiering av utveckling är problematiskt. Av detta skäl myntades den nya forskningsgrenen tidsnätverk (*temporal networks*). Inom tidsnätverk är målet ofta likaså att identifiera grupperingar men till skillnad från statiska nätverk innehåller de en tidsdimension. Därav anses tidsnätverk i mindre utsträckning förenkla verkligheten än vad statiska nätverk gör. De är således en fördjupning inom forskningsfältet nätverksteori. De algoritmer som används inom tidsnätverk är emellertid hämtade från det äldre forskningsfältet statiska nätverk. Med anpassningen till en mer komplex nätverksuppfattning har det även följt problem: Hur ska tid hanteras?

Förenklat går det att säga att det finns tre övergripande typer av metoder som används vid identifiering av grupperingar. Den första innefattar att man reducerar tidsnätverken till ett statiskt nätverk men försöker att bibehålla en del av tidsdimensionen genom att lägga till så kallade vikter (*weights*). Dessa fungerar som en indikator på intensiteten av kontakterna mellan aktörerna i nätverket. Därefter appliceras en algoritm från statisk nätverksteori som har blivit modifierat något. Den andra metoden innebär att tidsnätverket delas in i så kallade tidsfönster (*time slices* eller *time windows*), varefter man sedan analyserar dessa med algoritmer från statisk nätverksteori. Målet är då att hitta grupperingar som är någotsånär lika från ett tidsfönster till ett annat. Argumentet är då att grupperingarna uppvisar en viss utveckling. I den tredje och sista övergripande metoden delar man upp tidsnätverk på samma sätt. Skillnaden ligger i hur grupperingarna identifieras. Alla tidsfönster har en koppling till varandra, ordnade efter tid, vilket påverkar vilka typer av grupperingar som hittas. Alla ovan nämnda metoder för att identifiera grupperingar inom tidsnätverk delar ett gemensamt problem, nämligen *hur* tidsnätverket ska delas upp. Eftersom forskningsfältet är nytt ligger mycket av dess fokus på att försöka skapa nya algoritmer snarare än att lösa detta grundläggande problem. Vidare är det svårt att säga *varför* identifieringen och avgränsningen av en viss gruppering bättre relaterar till något objektivt påvisbart än en annan avgränsning. Forskningsfältet är således subjektivt och resultat påverkas av vilken typ av algoritm som används samt vilket nätverk som studeras.

Inom den ovan sistnämnda övergripande metoden är algoritmen Generalized Louvain ett populärt val för att identifiera grupperingar. Generalized Louvain traverserar ett nätverk och maximerar en kvalitetsfunktion. Mätvärdet kallas för modulairtet och avser beskriva hur modulärt eller "välanpassad" en viss gruppering är. Mätvärdet är en skalär mellan -1 och 1. Det är beroende av hur många aktörer och kopplingar

som existerar. Ett underliggande problem med modularitet är dock att det inte är jämförbart från ett nätverk till ett annat. Problemet ligger i att varje gång ett nätverk har blivit uppdelat i tidsfönster kan det anses bilda ett "nytt" nätverk. Exempelvis kan ett tidsnätverk delas upp i fem respektive tio tidsfönster vid två olika tillfällen. Vid applicering av algoritmen kommer då den resulterande modulariteten att vara olika i dessa båda fall. Detta är en konsekvens av att många kopplingar mellan aktörer kommer att brytas upp, samt att andra skapas. Således kommer den slutgiltiga övergripande grupperingen att se annorlunda ut. Vidare kommer det skalära värdet modularitet att vara baserat på olika antal aktörer och kopplingar tidsfönsterna sinsemellan. Av detta skäl skapades en normaliserad version av modularitet genom en ingående studie i Generalized Louvain. Studien resulterade i att det gick att påvisa att en jämförbar modularitet hade uppnåtts vid två av de fyra tidsnätverk som studerades. En möjlig förklaring till att tydliga resultat inte kan hittas i alla nätverk kan vara att det beror på tidsnätverken själva. Exempelvis är det en möjlighet att vissa topologiska förhållanden inom nätverket påverkar algoritmen och i förlängning, resultaten. Förslag till vidare forskning är således att studera hur nätverkets beståndsdelar kan påverka ett mätvärde av denna typ. Vidare kan det vara av intresse att skapa ett likande mätvärde baserat på en av de andra övergripande metoderna för identifiering av grupperingar inom tidsnätverk.

Förord

Detta examensarbete har varit en avslutande del av civilingenjörsprogrammet system i teknik och samhälle på Uppsala universitet. Matteo Magnani har varit till ovärdelig hjälp som handledare gällande de komplexa frågor som existerar inom det nya forskningsfältet tidsnätverk. Vidare skulle jag vilja tacka Davide Vega för hjälp med programmeringsproblem, Amin Kaveh för skämten och Geogrios Fakas för feedback. Sist men inte minst vill jag tacka min familj och min vapendragare för ett enormt stöd.

Axel Lindegren,

Uppsala, 14 Mars, 2018

Contents

Abbreviations and definitions

Actor	Vertex in a graph. In a social network, the actors are the people which interact within it.
Clique	An all-to-all connected community.
Degree	The degree of a node is the number of edges connected to it.
Edge	The interaction or contact between two actors.
Horizontal community	A community that spans multiple time slices (often all).
Layer	A time slice. Layer is the more general term when operating with multiplex or multilayer network. Time slice is temporal network specific.
Multilayer network	A network which is comprised of multiple layers. Each layer can represent some type of environment, e.g. if one layer is a school network, another could be a football team network. The full network would then be both of the afore mentioned networks, divided into different layers, sharing a large set of the same actors.
Multiplex network	A type of multilayer network. In a multiplex network *all* actors are replicated in *all* layers.
Node	Time slice specific actor.
OGL	The Ordered Generalized Louvain.
Singleton	A community consisting of only one node (Bazzi et al. 2014).
UGL	The Unordered Generalized Louvain.

1 Introduction

For the last few decades, the empirical study of social networks has been a popular field, especially with the growth of social media. For example, Yogeeswaran et al. (n.d.) studied the Twitter network of the top five presidential candidates — with focus towards Donald Trump and Hillary Clinton — for the election in the US in 2016. The data was collected for a total of 9 weeks, 4 weeks prior and 4 weeks post-election. The goal of the study was to see if there were decidedly more people connected to leaders or followers of American hate groups whose networks coincided with any of the candidates'. Using network theory, and other means, they could conclude that there was a significant number more strongly connected to Donald Trump than to the other candidates.

The study of large networks, social and other types, requires tools, many of which have been adapted from graph theory. In its most reduced form, a network can be seen as a graph. The graph is built of a set of *nodes*, which represent entities in the network, such as people in a social network, and a set of *edges*, connecting the nodes as a form of *contact* or *interaction*. A network graph can also be *weighted* or *unweighted* (Bazzi et al. 2014). This indicates the intensity of the interaction between two nodes, compared to the network as a whole. Networks can also be considered to be *static* or *dynamic*. The former type implies that the network does not change its structure over a set condition, whereas dynamic networks do. A *temporal* network is a type of dynamic network, where the network structure changes over time. This causes a number of different problems whilst trying to study it using algorithms from the well developed field of static networks. As mentioned before, a static network graph can be depicted by nodes which are connected by edges. The difference to a temporal network is that the edges (contacts) may not be present at all times. The general goal of studying temporal, instead of static, networks is avoiding the significant simplification that a static representation of such a network imposes.

As an example, think of a small social network of three people, person A, B and C. If A interact with B at time t_1 and B interact to C at time t_2, then both of the two edges (contacts/interactions) are not active at the same time. Compare this to a static network where the edges always are present, meaning that the network does not change. In a temporal network the edges are extended to contain a time t, showing when the edge was active. Since the emergence of social media, the study of social networks has seen an ever increasing interest as a field of study. There are also other types of networks worth mentioning, for example technological and biological networks. They will not be actively treated in this thesis, but temporal networks of such composition use the same theoretical framework.

1.1 Problem statement

An ever prominent challenge when studying temporal networks is the definition of time. For some sociotemporal networks, such as a network where the population exchanges emails, the aspect of time is quite straight forward, i.e. the time when an email is sent or received. In contrast, other temporal networks can have contacts taking place over a time period. An example of such a network is one with

meetings between individuals. A meeting can then be described with a time interval.

Often when studying temporal networks, the goal is to find or detect *meso-scale structures*. These can be described as non-random occurring groups of actors, often referred to as *communities*. Communities are commonly defined as clusters or subgraphs of densely connected actors in a network (Fortunato 2010; Bazzi et al. 2014; He et al. 2017). Several algorithms have been developed for community detection in static networks. However, many possible application areas contain a temporal dimension. With temporal networks, one tries to reduce the simplification which lies in static networks. Following such an argument, there are several applications for community detection in temporal networks: one-to-many information spreading through social media like Facebook (Zhao et al. 2010) or Twitter (González-Bailón et al. 2011), neuroscience (Bassett et al. 2010), ecology (Holme & Saramäki 2011), among others. While all of these are not social networks, it is important to mention the wide area of application of temporal networks.

A few main approaches have been used in the later years to detect communities in temporal networks (Bazzi et al. 2014; Holme 2014). The first includes constructing a static network by aggregating contacts from a temporal network. By using weights on the edges, the created static network contained more information than a regular static network. A second way of attacking the problem is to use community detection algorithms developed for static networks on multiple partitions of a temporal network. Then from each of the partitions (or aggregations), one tries to match the communities found, identifying communities spanning some time interval. The third way entails similarly aggregating parts of a temporal network into slices (or layers) of a multilayer network. While this share similarities with the second approach, the main difference is that the slices or layers are connected, creating a type of dependency (Bazzi et al. 2014). The two latter, and arguably more promising, above mentioned strategies of community detection share a prominent feature in common; the partitioning of a temporal network. However, it is not abundantly clear as to *how* the aggregation affect the final outcome of the community detection algorithm.

Formally, a temporal network can be defined as a network where edges have time annotations, which are based on an ordered set of times T. While in general, one can think of time being continuous, and maybe even unbounded on one or both sides, the focus of this thesis is on temporal networks that can be represented and processed using computers. Following that argument, consider a discrete, lower- and upper-bounded set $T = \{t_{min}, t_2, \ldots, t_{max}\}$, with $t_i < t_j \forall i < j$. The annotation t_i here represents a *time* or *timestamp*. In some cases one can also assume that $t_{i+1} - t_i = t_{j+1} - t_j$, for all i and j — that is, both ordinal and interval attribute types are allowed.

Sometimes it is practically useful to transform T into a coarser set T', so that a larger number of edges is present at each timestamp. This operation is called *slicing*. For example, if we have a set T where each element $t \in T$ corresponds to one minute, one can aggregate groups of 60 timestamps so that every $t' \in T'$ corresponds to one hour. The objective of slicing is typically to have enough edges associated to

each slice, so that existing network analysis algorithms can be applied slice by slice.

Figure 1 shows an example of a temporal network. The actors are annotated with $A \in \{1, 2, 3, 4, 5\}$ and are connected with edges (black lines). Each edge have some time points, indicating when it was active, i.e. when two actors had a *contact*. Running a static community detection algorithm on the network in figure 1 does not work. The extra information of a set of times when the edges were active cannot be properly handled by a such an algorithm. What is possible though is to aggregate an ordered series of networks, i.e. *slicing* the temporal network, based on some time condition. Figure 2 shows the hypothetical result of partitioning a network into *three* static networks, and then applying a community detection algorithm. Communities can then be visualized as in the figure, some actors are in community 1, while the others are in community 2. Continuing the argument, figure 3 represents hypothetical results of running the *same* community detection algorithm on the *same* temporal network (shown in figure 1). The hypothetical results yield a different number of communities and internal community structure. The only difference is that the network has been partitioned into *four* static networks instead of three. The goal is then to decide which partition is better, based on the *quality* of the found communities. As a real-life example, one can think of the network in figure 1 as a family of five individuals, where some of the family members talk or interact more often than others. In such a case, a community detection algorithm will most likely find a community within the network, i.e. find the family members that talk more often to each other. In a sense, the community is then based on the frequency of the interaction in time. That being said, if the input is different, i.e. if the temporal network is partitioned differently, the community detection algorithm might find other communities yet again. This is a consequence of partitioning a temporal network, to enable for a current community detection algorithm to be applied.

For one of the two above mentioned temporal community detection approaches, the quality of the found communities is described as a scalar value between -1 and 1 where higher is better. That the measure is scalar, and dependent on the network of study, implies that it is not feasible to compare one networks qualitative measure to another networks. In extension, this implies that when a temporal network has been sliced differently twice, e.g. into 3 and 4 slices at separate times (like figure 2 and 3), the output of the algorithm *cannot* be compared. Normalizing a measure of this type, i.e. making it comparable, is then paramount in deciding which of the two partitions is better. As an example of this problem from previous research, He et al. (2017) partitions a network into multiple layers according to the *Moore visualization method*. They create different partitions of the same network which overlap heavily and then apply their community detection algorithm. This implies that they do not know of an optimal partition for their studied networks, thus the reason for testing different settings.

As a result of this inconclusiveness, it is of importance to study the effect of partitioning temporal networks. The goal is then to find an optimal, or at least measurably better, partition. Creating a good measure is key to understanding how a community detection algorithm might be affected by partitioning temporal networks. This will be done by taking a few temporal networks as input and applying a modified com-

munity detection algorithm which utilizes partitioning, to decide where in time the split should be done. The output will then be a specific number of wanted partition for each temporal network individually, but is retrieved from the attained community structure, i.e. the original output from the community detection algorithm. This could further the research of temporal networks, aiding in other applications than social temporal networks alone.

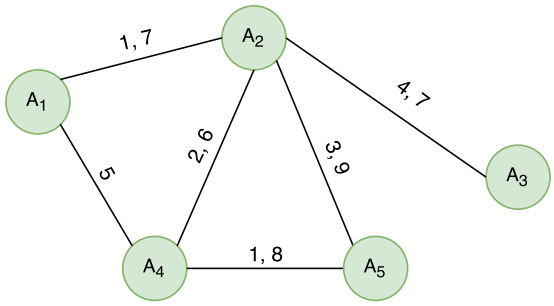


Figure 1: A temporal network consisting of actors $A \in \{1, 2, 3, 4, 5\}$ *and edges. Edges are annotated with discrete time, indicating when a contact between two actors happened.*

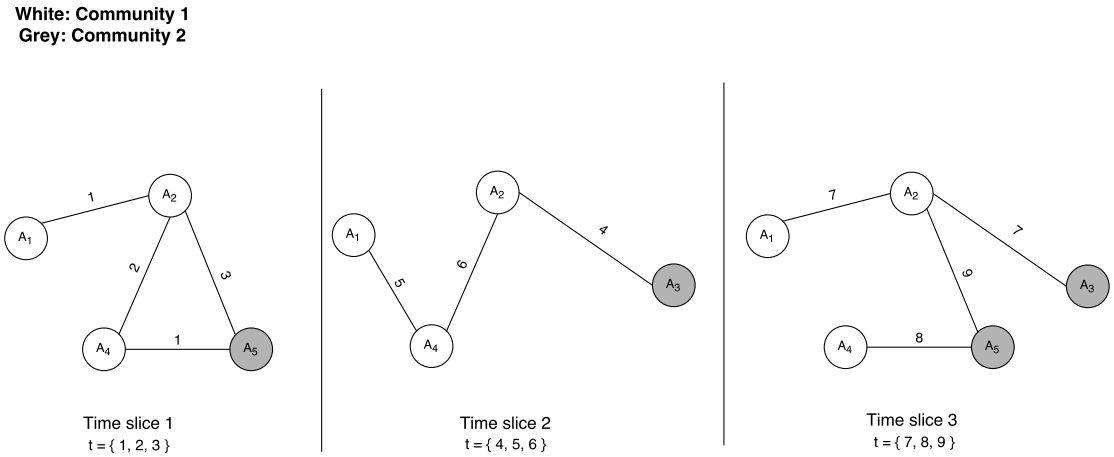


Figure 2: Toy results of running a community detection algorithm on the network in figure 1, after is has been partitioned into 3 slices.

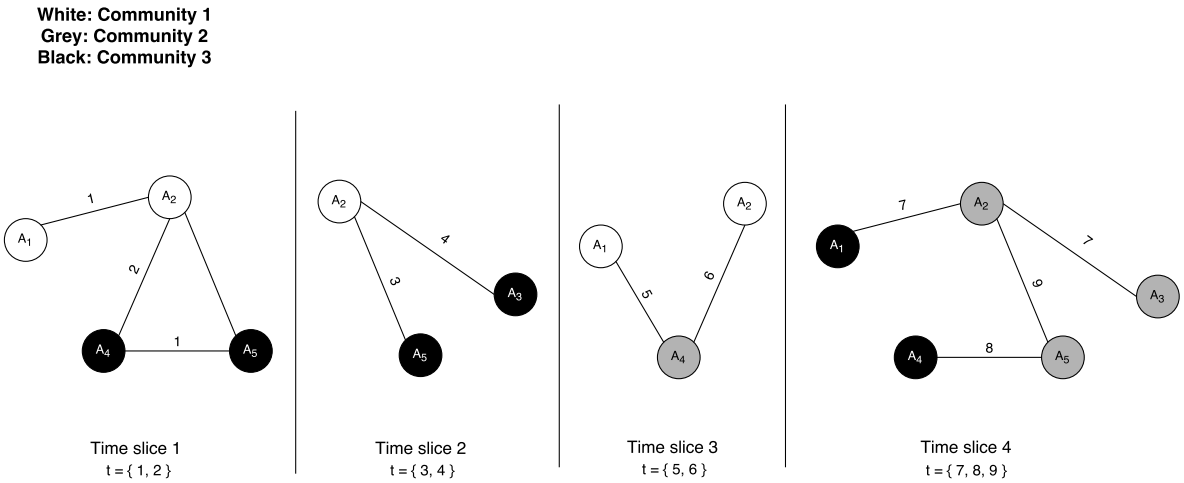


Figure 3: Toy results of running a community detection algorithm on the network in figure 1, after is has been partitioned into 4 slices.

1.2 Thesis purpose

The purpose of this thesis is to research the effect of partitioning temporal networks, with the intention of finding an optimal aggregation in comparison to previous research using a community detection algorithm. The algorithm will be decided through a literature review of the state-of-the-art research in community detection in temporal networks, with the intention of choosing a popular algorithm that takes partitioned temporal networks as input.

1.2.1 Research questions

The purpose of this thesis can be summarized in the following research questions:

- What is state-of-the-art in community detection in temporal networks?
- Which algorithm is the best fit for optimizing a partitioning method for temporal networks?
- What is an optimal partition of a temporal network?

1.2.2 Limitations

As a consequence of the scope of this thesis, a few limitations will be introduced to limit the field of study. First, the state-of-the-art section will group community detection techniques into the three mentioned approaches in section 1.1. Secondly, the thesis focus will be to continue exploring the limitations of *one* of the surveyed techniques, as part of one approach. Thirdly, only unweighted and undirected networks will be considered.

2 State-of-the-art in temporal networks

In this section, definitions used in temporal network theory will presented under collected definitions and the three common approaches for community detection

will be described. This was done as to survey the relatively new field of temporal networks, where the jargon remains fluid or indecisive.

2.1 Data models

Recall the formal definition of a temporal network and the operation of slicing a temporal network from 1.1. In literature, time slices have been called in different ways in the literature: Holme (2014) describes *time windows* which are aggregated contacts (edges) over some time period, while Wang et al. (2016) use the term *snapshots*. Here, all afore mentioned versions will be seen as synonymous. Formally, when given a set T, that can also be the result of slicing, one can then make a distinction between two types of time annotations:

- A set of points in time (P), each defined by an element t in T.
- A set of time intervals (I), each defined by a pair of elements t_i, t_j in T.

While edges must have some kind of time annotation in temporal networks, actors may have no time annotation (N), which practically corresponds to the interval $[t_{min}, t_{max}]$ indicating that all actors are always present.

Table 1 shows which combinations of time annotations that are meaningful for actors and edges, leading to five main types of temporal network models. The relevant combinations are numbered 1 to 5 below.

Table 1: Main types of temporal network data models. If both actors and edges are associated with time (*), the edges' recorded times are constrained by their actors' recorded times, that is, an edge cannot be associated to a time that is not associated to both its actors.

Actors \ Edges	N	P	I
N	Not temporal	Yes	Yes
P	Not consistent	Yes*	Not typical
I	Not consistent	Yes*	Yes*

The following are examples of the types of temporal networks in Table 1.

1. Actor (N), Edge (P): Email, forums, and other online communication networks where the users are always considered potentially active. This is the most popular format for raw network data.
2. Actor (N), Edge (I): Face-to-face contacts, with discussions spanning a time interval.
3. Actor (P), Edge (P): This is typically used when elements of T correspond to long periods of times, e.g., after slicing. For example, a set of daily email networks.

4. Actor (I), Edge (P): Actors who are on-line on a social media platform "liking", "re-tweeting" or similar, where the actors' on-line time are being recorded as well as the time of the operation.

5. Actor (I), Edge (I): Workshop, conference or similar, where both the actors' time spent at the event as well as their social interactions (edge) times are recorded.

Formally, there is no need for any special data model for sliced temporal networks. An Actor (P) Edge (P) model works both for sliced networks and networks with finer timestamps. However, a practical difference can be identified between data where the networks at each timestamp t are very sparse and data inside some timestamps where we have enough active edges to be able to observe some community structure. This difference is important because the two cases are typically handled by different types of algorithms.

2.1.1 Visualizing temporal networks

One way to represent a temporal network, which works for all types of data models, is to draw the network and specify time annotations on its actors and edges. An example of a temporal network visualized in this way is shown in Figure 4. Table 1 can then serve as an indication of how temporal networks can be represented. Any of the possible combinations — motivated above — of time annotations can be used.

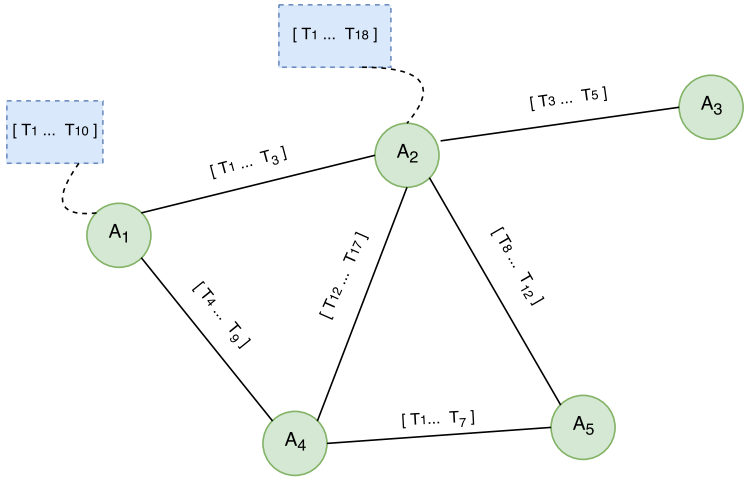


Figure 4: Overall visual representation of a temporal network. The different time annotations associated with edges can be thought of as either point time (P) or interval time (I). Following the statement above, actors cannot have point times (P) associated with them and can therefore only be associated with no time (N) or interval time (I).

Contact sequences are another way of modeling a temporal network, as in figure 5. This format encapsulates the same information as the time-annotated network graph, but can be useful to understand the different types of community that can exist in a temporal network. At the same time, this visualization does not work well in general if too many edges have the same timestamp (Holme 2014).

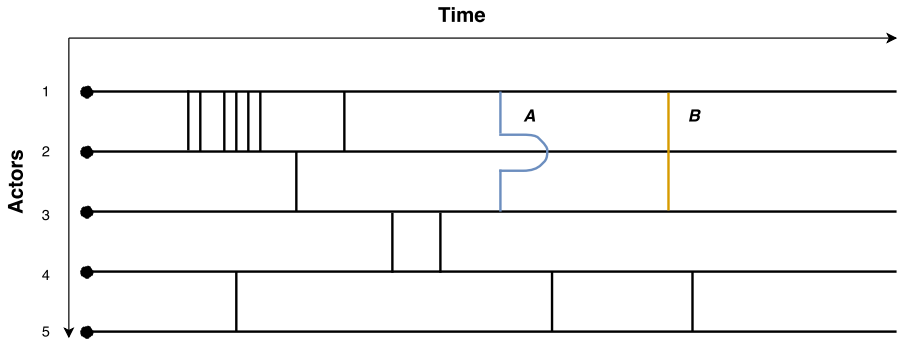


Figure 5: An alternative graphical representation of a temporal network. Each black vertical line serves as an interaction or contact (Holme 2014) between two actors at a given time. Notice contact labeled as A. The curvature indicates that actor 2 is not included in the contact. Compare that to contact B, which spans actor 1, 2 and 3. This indicates that all three actors have some interaction at the same time.

Another way of representing temporal networks is to show the actors and edges that are active at each timestamp. This representation, shown in Figure 6, is typically meaningful only when enough edges are active inside the same timestamp. Therefore in the figure, a network that has already been divided into slices is presented, each time slice corresponding to a set of consecutive timestamps.

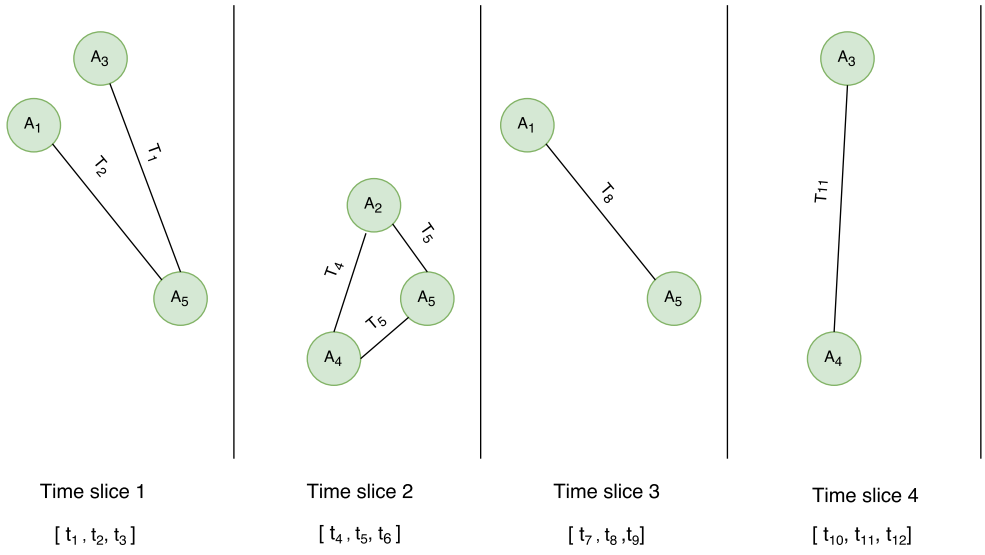


Figure 6: Visual representation of a temporal network where the network has been partitioned into time slices or snapshots. Each time slice only includes the edges (and attached actors), which were active in a given interval $[t_i, t_j]$. In this example, the time slices are evenly partitioned, i.e. the time intervals span the same number of discrete points in time or time intervals.

2.2 Definition of temporal community

There seems to exist an overall consensus in current temporal network literature that the definition of a community is a variation of: "A community — cluster, sub-

graph, module — in a network is a set of nodes more tightly — densely, strongly — connected to each other than they are to the rest of the nodes outside of the community" (Fortunato (2010); Bazzi et al. (2014); He et al. (2017); Kuncheva & Montana (2015); Wang et al. (2016)). Interestingly enough, only a few authors try to discuss the core of the definition, namely the part about "densely, strongly or tightly connected". Bazzi et al. (2014) argue that it is difficult giving a precise definition of what *densely connected* means. What they do conclude though, is that the definition is subjective and in particular, may depend on the proposed application.

In this section, an attempt will be made as to define what *types* of communities exist in the literature. In fact, while most researchers seem to be in agreement regarding the loose definition of what a community is, existing algorithms produce different structures. While the line between the definition of community and a community type sometimes feels more philosophical than factual, this section is aimed at shedding some light on the current situation. The different types of communities are presented in the list below.

1. Actor-only (Ao). This type of community has no temporal annotation. In this case we assume an underlying stable division of the actors into communities, that one can identify based on their timed interactions. The community itself is persistent through time and the actors in the community do not change. Mathematically, this community type can be represented as a set of actors: $\{a_1, \dots a_m\}$

2. Actor-static (As). An actor static community can be represented mathematically as the cross product of a set of actors A and one time interval: $\{a_1, \dots a_n\} \times [t_i, t_j]$. Ao communities are a special case of As communities, where $[t_i, t_j] = [t_{min}, t_{max}]$

3. Actor-varying (Av). This type of community can be mathematically represented as a set of time-reachable triplets $\{ (a_1, t_1^{min}, t_1^{max}), \dots, (a_m, t_m^{min}, t_m^{max}) \}$. Time-reachability means that it is possible to go from each triplet in the set to every other triplet through a set of steps $(a_i, t_i^{min}, t_i^{max}) \rightarrow (a_j, t_j^{min}, t_j^{max})$ so that the intersection between $[t_i^{min}, t_i^{max}]$ and $[t_j^{min}, t_j^{max}]$ is not empty. Av communities may contain different actors at different times, and the same actor can be active inside disjoint time intervals if it is present in more than one triple. An As community is a special case of an Av community where t_i^{min} and t_i^{max} are the same for each actor a_i in the community.

Figure 7 shows how the three types of communities defined above look like:

- Community 1 ($C1$): The first community in Figure 7 is what has been defined as an actor only (Ao) community.

- Community 2 ($C2$): This community is an actor static (As) type. The difference to $C1$ above is that the time interval is a subset of the full recorded time in the temporal network.

- Community 3 ($C3$): Shows an actor varying (Av) community type. The community is still considered to be the same, even though the two "original" actors (actor 7 and 8) have left it at the end.

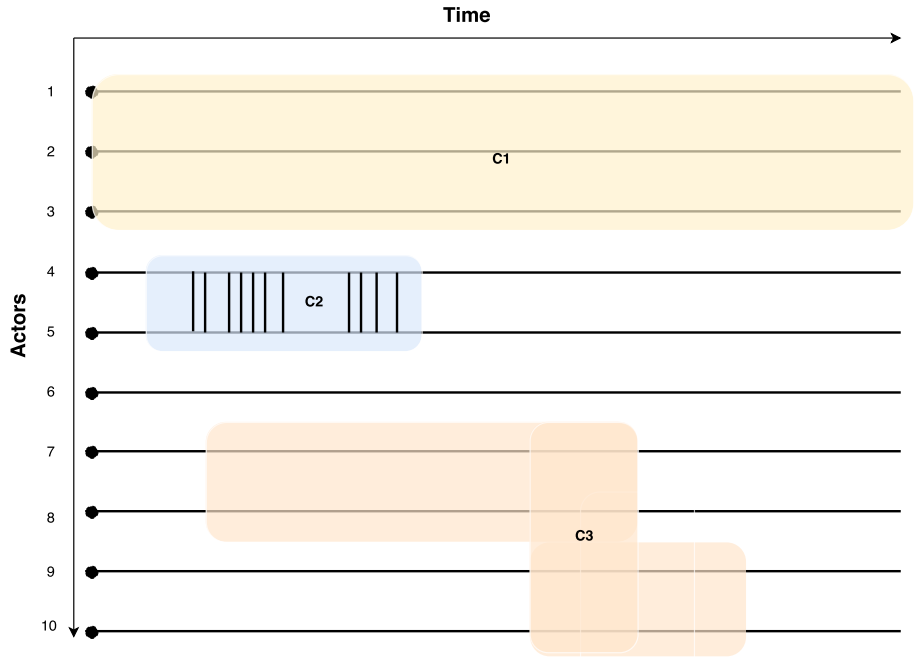


Figure 7: Graphical representation of the three main types of communities. Communities are shown as a dense region of contacts between actors. To make the figure more understandable, only contacts in community 2 (C2) are shown as an example.

Apart from their external shape, it is also possible for communities to have specific internal time patterns: inside a community one can find denser interactions at specific times. As an example, a group of students may work together during a course, forming a community for the whole duration of the course, and during this time students interact more with each other during the day, or every Monday morning in case they have weekly meetings, etc.

2.3 Temporal community structure

It is important to clarify the difference between a *community* and a *community structure*. A community is defined as one of the cases in the section above. A community structure is a set of communities. This section is further divided into three subsection, each describing one type of community structure. These are not always mutually exclusive, but will be treated so here as a consequence of the scope of this thesis.

2.3.1 Dynamic community structure

Sometimes, instead of only having a set of communities one may also want to identify the relationships between them. In particular, two or more communities can be connected by events of type: grow, shrink, merge and split. There are various definitions in the literature related to dynamic communities. He et al. (2017) define a dynamic community as a time line of several "step communities", ordered by time (see Figure 8). A "step community" is a community in each *time step*, which here is considered to be synonymous with a *time slice*. Wang et al. (2016) describe community evolution in terms of a small set of core vertices. For instance, when two successive communities share a common core vertex but possibly more vertices

overall, then the second community (from a temporal point of view) is considered as an evolved version of the first. Wang et al. (2016) definition of evolving communities is here interpreted as dynamic communities.

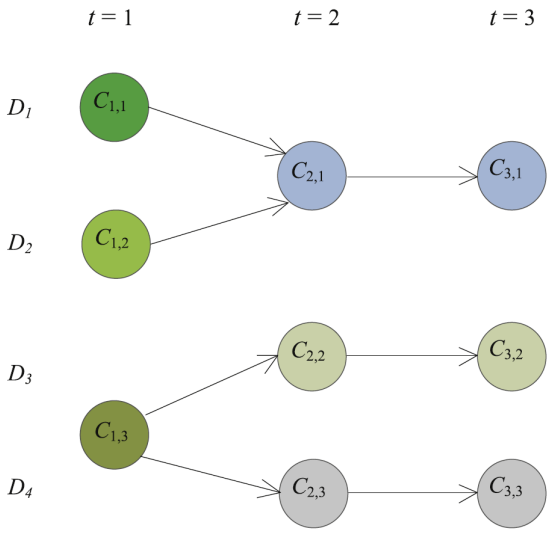


Figure 8: Dynamic communities (represented by D_1 to D_4), with step communities (represented by $C_{1,1}$ to $C_{3,3}$), for each given time step t. He et al. (2017)

2.3.2 Overlapping community structure

A feature of temporal networks is that actors can exist in more than one community at the same time. This definition has an intuitive explanation in real-life. As individuals, we are more often than not associated with multiple social groups, which is also the motivation behind overlapping communities. As in real-life, many social group can, for example, partly share the same individuals (Wang et al. 2016). Take three friends who play in the same football team and go to school together as an example. If the football team and the school are communities, then the three friends are what makes the communities *overlap*.

There are two ways of defining an overlapping community structure. First, actors can exist in two or more communities at the same time, i.e. communities are *time overlapping* (mathematically represented in equation 1). Figure 9 shows a *time overlapping* community structure. Secondly, actors can exist in two or more communities at *different* times. Communities are then *actor overlapping*. This is formally defined mathematically as one actor existing in two or more communities at two or more time points. The base case of two different time points and two different communities are shown in equation 2 and 3. Lastly, it is important to clarify that actors can exist outside of a community. All actors do not have to be partitioned into communities (shown in equation 4).

$$(a, t) \in c_1 \cap c_2 \cap ..., \cap c_n \tag{1}$$

$$(a, t\ ) \in c_i \tag{2}$$

$$(a, t') \in c_j \quad i \neq j \tag{3}$$

$$(a, t) \in \emptyset \tag{4}$$

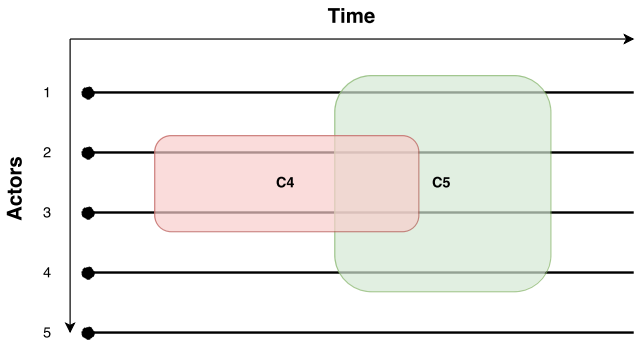


Figure 9: A special feature of temporal networks is that actors can exist in multiple communities at the same time. Actor 1 and two are here present in both $C4$ and $C5$ over the time period of the communities intersect.

2.3.3 Hard partitioned community structure

Bazzi et al. (2014) describes what they call hard partitioned communities. A hard partitioned community is when an actor only can be subject to one community at any given time. Another way of describing hard partitioned communities is presented by Kuncheva & Montana (2015), who defines a community which is limited to one layer (here interpreted as a *time slice*), as a non-shared community. A difference between hard partitioned community and non-shared, is that a non-shared cannot exist in the following time slice. Therefore, a non-shared community can arguably be though of as a discretization of a hard partitioned community structure. This implies that a non-shared community can be represented as a hard partitioned community which has, in turn, been discretized into time slices. It is therefore a generalization of a hard partitioned community. Kuncheva and Montana continue describing a shared community, which is a community that exist in multiple time slices, i.e. a hard partitioned community.

2.4 State-of-the-art approaches for community detection

In this section, an attempt will be made to make a brief classification of state-of-the-art approaches for community detection in the field of temporal network mining. The approaches are divided into three major approaches: 1) Two step approach; 2) Generative models; 3) Null models. As mentioned in section 1.1, the two latter mentioned approaches have one thing in common: the temporal network used as in-data has to be partitioned. Furthermore, they imply an arguably lesser simplification than the first mentioned approach. Each approach will be presented briefly in the following sections, respectively.

2.4.1 Two step approach

The two step approach, is based on the thought that every temporal network can be partitioned into *time slices*. The first step of the approach, disregarding the actual

partitioning, is to find communities in each *time slice*, often using some community detection algorithm borrowed from static network theory. The second step is to map the communities across the *time slices* according to some principle.Wang et al. (2016) Similarity indices are often used in this step. The belief behind a similarity index is that if two communities in different adjacent *time slices* are similar enough (with some ratio), they can be considered as a dynamic temporal community. For example, He et al. (2017) uses the Blondel method for detecting communities in each *time slice*.

2.4.2 Generative models

Generative models are often based on a combination of Stochastic Block Model and a dynamic system. These approaches represent a temporal network as a sample of a dynamic generative model. Generative models are designed to emulate statistical behavior in networks, e.g. degree distribution, which can then be used for validating analysis Tantipathananandh & Berger-wolf (n.d.). The popularity of the stochastic block model came from its wide use in static network community detection, and a lot of attention has been put into adapting it to temporal networks (see Guo et al. (2007); Xu et al. (2014); Matias & Miele (2017)) A major underlying assumption of the stochastic block model is that each node i of a network belongs to one of K-hidden communities with some probabilities. The goal of using a stochastic block model is to find a benchmark for finding "real" communities in the studied dataset. Tang & Yang (2014) describe an extensive approach using stochastic block modeling as a base. They built their approach on Yang et al. (2011) presented approach for community detection in 2011.

2.4.3 Null models

Null models are created to study different topological relationships within a temporal network. Generating a null model can be done in multiple ways, and each way correspond to a different fundamental constraint. The constraint itself depends on the system and type of analysis that is to be done (He et al. 2017). Null models are often used in conjunction with a modularity function. The idea is to optimize the modularity function, which includes comparing the actual connections (edges) in a network to the connections produced by a random graph (null model). The communities are then the set of nodes that are more densely connected than would otherwise be expected from the comparison to the null model (Sarzynska et al. 2014). A popular choice of null network is the Newman-Girvan null network. It is generated by randomizing edges in the studied network with the constrain that each edge (or vertex) has the same degree as in the original network (Bazzi et al. 2014). One popular approach for community detection in temporal networks is called the Generalized Louvain. The original version (called the Louvain) was created by Blondel et al. (2008) for static community detection. It was later adapted for multilayer — which a temporal network can be represented as — by Mucha et al. (2010). The generalized version of the algorithm applies an adapted version of the Newman-Girvan null model for multilayer networks.

2.5 Summary table of approaches for community detection

The community definitions and community structures used in the table are defined in section 2. Notice that the actor static, time static community type (Ao) has been excluded. This is a result of the definition being a static network community.

Table 2: This table provides an overview over which approaches, models and definition of community an article includes.

Article	Data model	Approach	Community type	Community structure
He et al. (2017)	Time slice	Blondel for communities in time slices. Jaccard index for matching across slices	Av	Hard partition
Tang & Yang (2014)	Time slice	Stochastic block model (heavily extended)	As	Hard partition
Xu et al. (2014)	Time slice	Evolutionary clustering	As	Hard partition (Overlapping in certain cases)
Sarzynska et al. (2014)	Time slice	Null model / modularity	Av	Hard partition
Wang et al. (2016)	Time slice	Nonnegative matrix factorization	As	Overlapping
Kuncheva & Montana (2015)	Point time	Random walk	Av	Hard partition

3 Methodology proposition

In section 2.4 the three main approaches for temporal community detection are presented. Most of the algorithms grouped into any of the three approaches often yield different results, i.e. they find different communities, when applied on the same network. However, the communities found is not what ties them together. Their adhesive is the act of partitioning temporal networks to achieve a data model that a community detection algorithm can be used on. Therefore, it is not necessarily the approach it self that is important for this thesis, but rather a measurable community structure that emerges from any of the state-of-the-art community detection approaches. A measure can thus help in researching the effect partitioning a temporal network and in extension, optimize it. Thus one has to study the impact of the different approaches. However, within the scope of this thesis, it is not feasible to test more than one. For practical reasons this has led to the choice of the popular algorithm the Generalized Louvain. It's characteristics and inner workings is presented more in detail below.

This section consists of four subsection. The first describe the two used partitioning methods in this thesis. The second gives a detailed explanation of the Generalized Louvain, its adapted version for temporal networks and a proposed method for testing the validity of said adaptation. Thirdly, a measure is proposed for comparing different partitions of temporal networks. This measure is based on two underlying assumptions further explained in section 3.3.1. Lastly, a summary table of all proposed tests for this study are presented.

3.1 Partitioning methods

In this section, two partitioning (slicing) methods of aggregating a temporal network into time slices are presented. Using different methods when aggregating a temporal network into time slices have potential to yield different results when applying the same community detection algorithm on the different aggregations. For example, He et al. (2017) uses the so called *Moore's Visualization Method*, which implies a certain overlap of time slices. They try out a few different settings, which yield different results. The partitioning method they use is not studied here, as it does not comply well with the Generalized Louvain.

Let figure 10 represent a base for visualizing how the two partitioning methods function. Each method will then be visualized in their respective section.

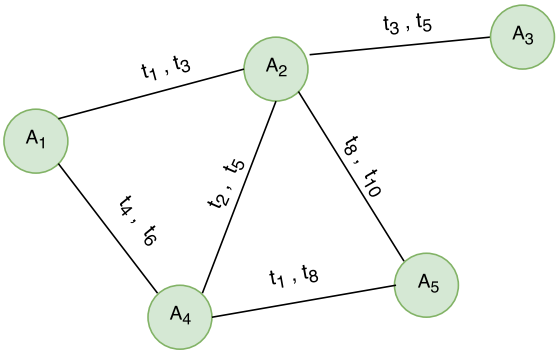


Figure 10: Visual representation of a temporal network.

3.1.1 Equal time

Partitioning a temporal network based on *equal time* implies that the full set of times T in the network is split as evenly as possible. The edges are then aggregated into time slices (layers) of a multilayer network. Partitioning the toy network in figure 10 yields the results displayed in figure 11, if assuming that the time is ordinal. In the setup of figure 11, time slice 1, 2 and 3 contain, 5, 4 and 3 edges respectively. Notice that some edges have two *timestamps*, implying that it is in fact two edges, i.e. edge (A_1, A_2, t_1) and (A_1, A_2, t_3) in time slice 1.

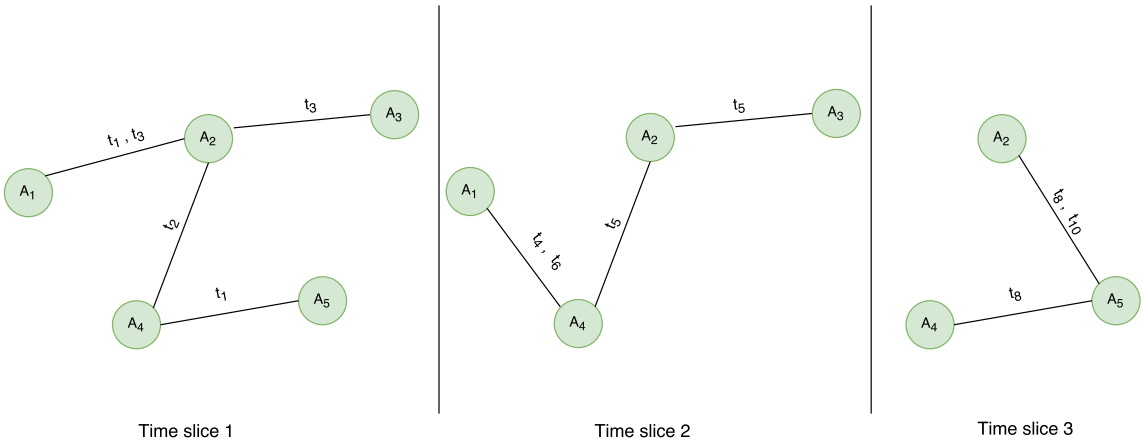


Figure 11: Visual representation of applying time partitioning on the network in figure 10.

3.1.2 Edge density

When a temporal network has been partitioned using *edge density*, the split is based on aggregating an equal amount of edges in each time slice. This is done by ordering the edges according to ordinal time, then aggregating them in a specified number of time slices. When partitioning in such a way, the overall edge distribution in each time slice is as even as possible. In comparison, when aggregating using equal time, the edge distribution is seldom as even as when aggregating using edge density. As an example, figure 12 shows the result of aggregating the temporal network displayed in figure 10 using edge density into 3 time slices. In the example of figure 12, *all*

time slices contain 4 edges. The difference between the two partitioning methods for this small example is apparent, which implies even more noticeable differences for larger networks.

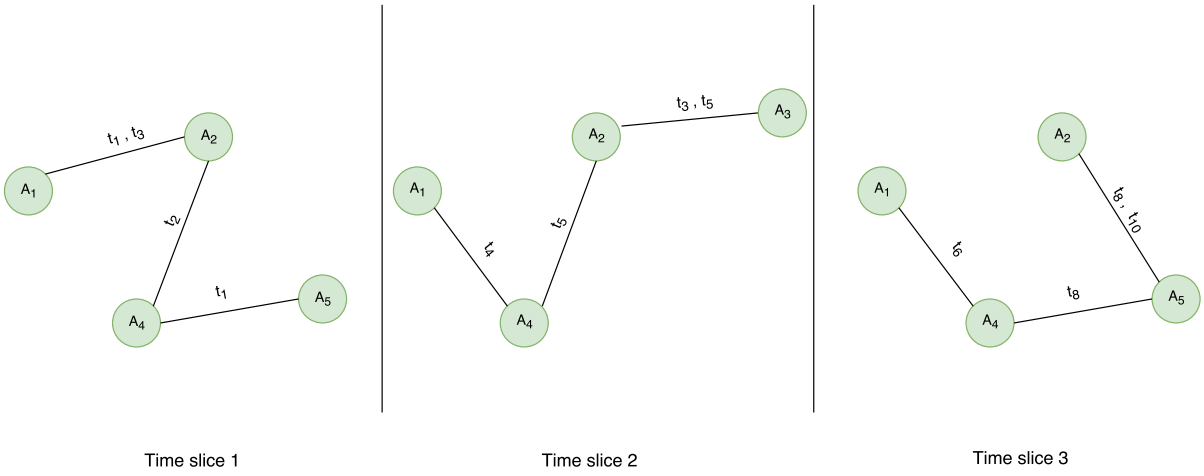


Figure 12: Visual representation of applying edge partitioning on the network in figure 10.

3.2 Generalized Louvain

The version of the Generalized Louvain used in this thesis is a C++ multilayer implementation of the "GenLouvain" created by Jeub et al. (2011-2017) for Matlab. The algorithm was created for community detection in multilayer networks and defines a quality function in terms of a generalized modularity null model framework. In turn, the Generalized Louvain applies a two-step approach similar to that of the original Louvain (see Blondel et al. (2008)). Furthermore, the algorithm utilizes an adjacency matrix for creating a matrix representation of a multilayer network. The goal of the algorithm is to maximize the modularity calculated by some quality function, which in this thesis is a multilayer generalization of Newman-Girvan modularity, denoted by the authors $Q_{multilayer}$ (Mucha et al. 2010).

The already implemented C++ version of the Generalized Louvain was further adapted, according to Jeub et al. (2011-2017) Matlab code, to work on ordered multilayer networks, i.e. sliced temporal networks. In this thesis, the earlier version will be denoted Unordered Generalized Louvain (UGL), and the later version Ordered Generalized Louvain (OGL). A special subset of multilayer networks are here used to represent temporal networks, called multiplex networks. In multiplex networks, all actors are present in all layers, even in the layers where they have no edges. In this section, the adjacency matrix and modularity will be presented and explained respectively.

3.2.1 Adjacency matrix

In multilayer networks, the adjacency matrix is often called a *supra-adjacency matrix*. It specifies both the intra-layer adjacencies between nodes and the between layer couplings (inter-layer edges). The intra-layer adjacency matrix specifies which nodes are connected within a layer. If node n_1 and n_2 are connected, the adjacency matrix A will have a value of 1 at element $x_{1,2}$ and element $x_{2,1}$. In the supra-adjacency matrix, all diagonal matrices are intra-layer adjacencies. Take a toy network with 3 layers (time slices) and 5 actors as an example. The supra-adjacency matrix will then be of size $(3 * 5) * (3 * 5)$, where the 3 diagonal matrices specifies intra-layer adjacencies for actors. All other matrices except the diagonal will then specify the inter-layer edges. These indicate to what degree the other layers affect each other, specified with the parameter ω (omega). Omega will be further explained in section 3.2.3. The supra-adjacency matrix is different for an ordered version of the Generalized Louvain in comparison to an unordered version Jeub et al. (2011-2017).

3.2.2 Modularity

Modularity is originally defined as a scalar value between -1 and 1 that measure the number of links inside communities as opposed to between communities Blondel et al. (2008). The modularity is calculated using a null model which can be described as a randomly generated network based on some topological condition, for example having the same node degree, i.e. the overall same amount of edges as the network of study. The Generalized Louvain computes the Newman-Girvan modularity matrix (based on the adjacency matrix), using quality function $Q_{multilayer}$ described by

Mucha et al. (2010). The quality function is a "multilayerization" of the Newman-Girvan null model and is presented below in equation 5 (Mucha et al. 2010):

$$Q_{multislice} = \frac{1}{2\mu} \sum_{ijsr} \left[\left(A_{ijs} - \gamma_s \frac{k_{is} k_{js}}{2m_s} \right) \delta_{sr} + \delta_{ij} C_{jsr} \right] \delta(g_{is}, g_{jr}) \tag{5}$$

where μ specifies all edges in the network (intra-layer and inter-layer edges). In the parenthesis, A_{ijs} is an element in the adjacency matrix and details a direct connection between node i and j in layer s, γ_s is the resolution parameter, k_{is} and k_{js} are the degree of node i and j in layer s and m_s are the total number of edges in layer s. The second part of the sum, C_{jsr} indicate the conditional probability of stepping from layer to s to layer r along a inter-layer edge from node j to itself, i.e. (j, s) to (j, r). The function δ is the Kronecker delta and takes the value of 1 or 0 depending on whether the multislice structure allows for movement between layer s and r. The last term of the equation, $\delta(g_{is}, g_{jr})$ is 1 if the community assignments g_i and g_j are the same and 0 otherwise Mucha et al. (2010). The Generalized Louvain can compute modularity based on other null models (see Bazzi et al. (2014) for more null models), but they will not be considered here.

3.2.3 Parameters

The Generalized Louvain take five input parameters:

- The temporal network: A C++ representation of a network, from InfoLab's MultiEdgeNetwork.
- Move: Specifies if the move function is random or not. The Generalized Louvain find a local optimum depending on which node it starts. An implication is that the results will be differing from one run to another. If the move function is set to random, where ever the Generalized Louvain start, it have some probability to jump to another node. If the move function is set to "normal", the algorithm will iterate through the nodes, always choosing connected nodes.
- Gamma (γ): A resolution parameter which decides the wanted resolution of communities. Mucha et al. (2010) recommend setting gamma to 1, since interpreting resulting output from the algorithm can be problematic. Gamma was therefore set to 1 in this thesis.
- Omega (ω): Is the inter-layer coupling weight parameter which specifies how much time slices affect each other. If omega is set to 0, the each partition of the temporal network (time slice) can be interpreted as a static network. The Generalized Louvain will then yield a modularity of 0. However, communities can still be found in each time slice, but the overall modularity will be 0. If omega is set to 1, each time slice of the temporal network will affect all other equally. For the OGL, only adjacent time slices may affect each other.
- Limit: How large the modularity and adjacency matrices can be, i.e. how much data that fits in memory.

3.2.4 Synthetic network creation

In this study, three synthetic networks were created for two reasons. First, to test if the implementation of the Ordered Generalized Louvain was done correctly, i.e. to enable for a validity check in a controlled environment. Secondly, a known community structure can help understand the effect the inter-layer weight coupling parameter omega (ω) has on an ordered version of the Generalized Louvain. When creating synthetic networks, focus was on trying to identify "corner cases", i.e. cases where the algorithm's community detection could be predicted at the same time as being unique enough to identify potential issues. The choice of only studying one of the two hyperparameters (omega), consists of two factors. First, studying the effects only one parameter heavily reduces the scope of this thesis test section. Secondly, Mucha et al. (2010) describe omega as a sensitive parameter which should be chosen for the studied network. For most networks, and in cases of uncertainty, the authors recommend setting omega to the value of 1. This will be taken into consideration in the study. If there is support for setting the parameter omega to 1, or close to, in the validity study, the two factors together will be seen as proof that the input value is satisfactory. If the validity does not yield any preferable omega, the standard value of 1 will be chosen. Four temporal networks are studied in this thesis, and choosing an individual omega for each is not feasible, therefore the validity study will partly have the goal of finding a preferable omega. To make it easier for the reader, the synthetic networks are presented in the section they are studied (see section 5.1).

3.2.5 Effect of omega on synthetic networks

After the OGL was adapted to temporal networks (represented as ordered multi-layer networks), a validity check had to be made. This was done by comparing the results of the OGL and the UGL when varying the input parameter omega for three different small synthetic networks. Each synthetic network was used as input to the OGL with three different values of omega: 0, 0.5 and 1. These values can be considered meaningful to test for a few different reasons. When omega is 0, the layers do not affect each other at all. An implication is that each layer can be considered as a static network and the community detection algorithm is then run on each of the layers in turn. When omega is 0.5, the impact of actors in other layers are halved, which could be a breaking point of the algorithm. As a result, the algorithm would either favor the more static network approach previously mentioned, or find communities closer to when omega is 1. Lastly, when omega is 1, all layers affect each other equally, implying that communities can be found horizontally through the layers. Results is presented showing the structure of the communities found for different values of omega. A community is represented as a set of nodes where, in turn, each node is represented as an actor and a layer (time slice), separated by a colon. Communities can contain one node or more.

To further explore the effect of omega on small synthetic networks, the structure of the communities is then kept the same, but the number of time slice is increased. This implies a larger temporal distance between communities present in all synthetic networks. A few different settings is tried: 1) one time slice is added *between* the original two, 2) then two more, 3) and lastly six more. For the last setting, one time slice is added *before* the first time slice and *after* the last time slice. This is done to

see if any noticeable difference can be discerned for the OGL in comparison to the UGL as it takes the order of the time slices into account. All settings for omega and the number of time slices are presented in the results.

3.3 Proposed measure to compare partitions of temporal networks

Figure 13 shows a multilayer representation of a network. Each of the rectangles represent one layer of a partitioned network, numbered 1, 2, 3 and 4. Within each layer, nodes are represented as dots and they are connected through edges (lines). The part of the modularity function marked with "within community" shows the total unweighted modularity that each *community* has (assuming the four nodes in the circle are a community). The part marked "interlayer edges" represent edges connecting two of the *same* actors in two different layers, i.e. node-to-node inter-layer edges. Finally, the *community index* is the index of one community. If the Kronecker delta is 1, then community index g_{is} is the same as community index g_{jr}. An important intuition when studying the Generalized Louvain on multilayer networks is that the modularity will increase when more layers are added. Therefore, the same applies if a temporal network is partitioned into an increasing number of time slices, i.e. when each time slice span a shorter time frame. This can be primarily be explained with an increase of inter-layer edges, where the increase is linear to the number of actors. Recall that the multilayer representation of temporal networks used in this thesis is a so-called multiplex network. In a multiplex network, actors in one layer are connected (with inter-layer edges) to themselves in adjacent layers. Therefore, when partitioning a temporal network in an increasing number of time slices, the number of inter-layer edges increases linearly to the number of actors and time slices. A proposed methodology to study this effect is presented in subsection 3.3.1.

A second important intuition about partitioning temporal networks is that the amount of edges in each layer might be different (see section 3.1). Therefore, it is important to study the effect of having a varying number of edges present in each time slice to create a comparable measure. A proposed method for researching this effect is presented in subsection 3.3.2.

The proposed measure is then presented in section 3.3.3 as an adapted version of the two normalizations from section 3.3.1 and 3.3.2 for temporal networks.

3.3.1 Temporalizing static networks

The term "temporalize" is here introduced and holds the meaning of replicating a static network in multiple time slices. The temporalized network is exactly the same in all time slices with the sole exception that the timestamps associated with edges are time slice specific. For example, if the same static network is replicated into two time slices, all edges in time slice 1 will have time $t_1 = 1$ and all edges in time slice 2 will have time $t_2 = 2$. This is done to study the increase of modularity when the number of time slices are increased (recall the intuition mentioned in section 3.3). To isolate the afore mentioned effect, the modularity was measured using the OGL on the temporalized networks. Since the network structure (number of nodes

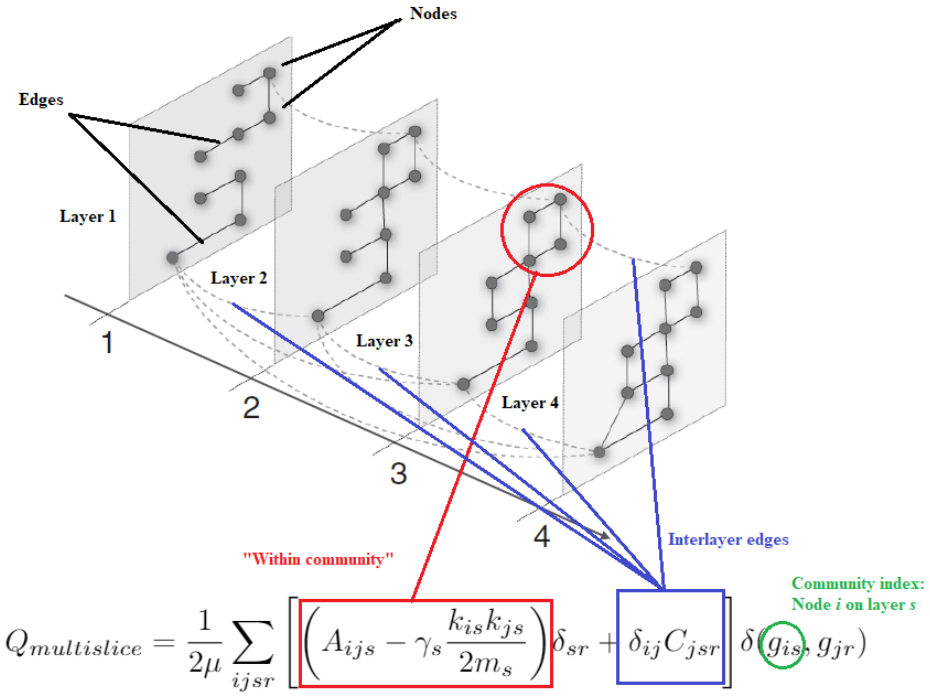


Figure 13: A multilayer network representation by Mucha et al. (2010). The figure has been modified as to show the different parameters of the quality function (modularity).

and edges) is known in *all* time slices, a normalization of the modularity could be proposed.

Assuming that we only consider the part of the modularity function when all i's and j are equal and when all s and r are equal yields the following simplified equation:

$$Q_{multislice} = \frac{1}{2\mu}\sum_{ijsr}\left[\left(A_{ijs} - \frac{k_{is}k_{js}}{2m_s}\right) + C_{jsr}\right]\forall i = j \quad \forall s = r \tag{6}$$

where all Kronecker deltas ($\delta_{i,j}$, $\delta_{s,r}$, $\delta(g_{is}, g_{jr})$) are equal to 1, i.e. when only considering nodes, time slices and communities which are the same. All other annotations hold the same meaning as in equation 5. Notice that the resolution parameter gamma (γ) is set to 1 (see section 3.2.3). In the first part of the sum ($A_{ij} - \frac{k_{is}k_{js}}{2m_s}$), the negative term is a value that varies in the interval of [-0.5, 1] and is unique for each solution of the optimization problem, i.e. it varies from one run of the algorithm to another. Likewise, the sum of A_{ij} varies from solution to solution since it is the total number "within community"-edges. To normalize the quality function based on the number of time slices, the left part of the sum was annotated as M_1 for the base case, when the static network was temporalized into only one time slice, i.e. it was still a static network but with temporal information. The *sum* of C_j — the total number of inter-layer edges — can also be rewritten as the number of actors a since it is an always-known constant for each number of time slices in turn in this controlled testing environment. When only considering one time slice, the term μ = m and the term $C_j = a = 0$, since there are no inter-layer edges. The equation

for the quality function could then be rewritten as follows:

$$Q_1 = \frac{1}{2m}\big(M_1 + 0 * 2a\big) = \frac{1}{2m}M_1 \tag{7}$$

where Q_1 is the total modularity for a network of 1 time slice. Multiplying a with two comes from the fact that the networks of study are undirected (or otherwise handled likewise), which means that all inter-layer edges are counted twice. Adding one more time slice to the base equation yields the following, assuming that the *optimal* partitioning is the same for all time slices since:

$$Q_2 = \frac{1}{2(m+m+a)}\big(2 * M_1 + 1 * 2a\big) \tag{8}$$

where Q_2 is the total modularity for a network. The pattern that emerges can then be used to write the general formula for the modularity for any number of time slices, based on M_1 and a:

$$Q_N = \frac{1}{2(Nm + (N-1)a)}\big(N * M_1 + (N-1) * 2a\big) \tag{9}$$

where Q_N represents the total expected modularity for N time slices. Notice that this *only* holds for the cases where the exact same number of actors and edges are present in *all* time slices.

3.3.2 Random edge removal

To study the effect that the number of *edges* has on modularity, the edges were removed iteratively. The modularity was then calculated using the OGL. Notice that the static network was only temporalized into *one* time slice, while for every iteration one edge was removed until there were none left in the temporal network. Assuming that the same community structure is maintained when removing one random edge at a time, each element of the "within community"-sums ($\sum A_{ij}$ and $\sum \frac{k_i k_j}{2m}$) can be multiplied by the expected remaining fraction.

Let m' represent the new number of edges while m represent the regular number of edges. The expected fraction of edges can then be written as $\frac{m'}{m}$. Multiplying each element of the "within community"-sums yields the following equation for the expected modularity for one time slice:

$$M_{m'} = \frac{1}{2m'}\sum_{ij}\left(A_{ij} * \frac{m'}{m} - \frac{m'}{m} * \frac{k_i k_j}{2m} * \frac{m'}{m}\right) = \sum_{ij}\left(A_{ij} * \frac{1}{2m} - \frac{k_i k_j}{2m} * \frac{m'}{2m^2}\right) \tag{10}$$

where $M_{m'}$ represents the modularity for one time slice with m' edges remaining. Annotating the sums for the "within community"-edges and degrees ($\sum_{ij}\left(A_{ij}\right)$, $\sum_{ij}\left(\frac{k_i k_j}{2m}\right)$) with $\bar{A}$ and $\bar{K}$ respectively yields the following equation:

$$M_{m'} = \bar{A} * \frac{1}{2m} - \bar{K} * \frac{m'}{2m^2} \tag{11}$$

3.3.3 Final measure

Merging equation 11 from section 3.3.2 and equation 9 from section 3.3.1 yield the final model adapted for temporal networks:

$$Q_{norm}(m, \overrightarrow{m'}, \bar{A}, \bar{K}) = \frac{1}{2(Nm + (N-1)a)} \left(N * \left(\bar{A} * \frac{1}{2m} - \bar{K} * \frac{m'}{2m^2} \right) + (N-1) * 2a \right) \tag{12}$$

where Q_{norm} is the expected modularity of a temporal network where the two underlying assumptions have been applied. The input parameters are the following: m represent the maximum number of edges of a temporal network, $\overrightarrow{m'}$ is a vector of the number of edges in each time slice, $\bar{A}$ is the total number of edges existing in all communities and $\bar{K}$ is the sum of the node degrees existing in all communities. $\bar{A}$ and $\bar{K}$ The goal of applying the created model to temporal networks is to create a comparable measure for modularity. Modularity is a scalar value and is therefore unique for each network. It is dependent on the structure of the network, i.e. how many edges and inter-layer edges that are present and how the nodes are connected. The intuition behind applying the created model from temporalized networks is to estimate the effect of partitioning a temporal network into more time slices. By then taking the fraction of the measured and estimated modularity, the effect of increasing the number of time slices can be factored out. Therefore, the resulting graph should show a peak, which is an indication that the estimated modularity grows faster than the measured. At the peak, the optimal number of time slices for the studied network can be attained. No discernible peak could serve as an indication that the temporal network is not suited for the applied method. Another possible explanation for such an event is that not enough time slices have been iterated through. However, for each new time slice added, the size of the modularity matrix grows quadratically against the number of actors times the number of time slices. For example, if we have a network with 100 actors and partition it into 300 time slices, the modularity matrix will contain approximately $9 * 10^8$ elements (mostly zeros). Therefore, the method is computationally demanding and the highest number of possible time slices that can be iterated through is dependent of the size of the network.

3.4 Summary table of proposed tests

In this section a summary table is presented showing all tests which were proposed including the input and expected output to each test, respectively.

Table 3: This table provides an overview over all proposed tests studied in this thesis. The tests are ordered as they appear in the thesis. Note that the number in front of each test only acts as an indication of which tests are connected and is not the actual section number.

Test	Input	Expected output
1.1 Synthetic network (test case 1)	Synthetic network of 4 actors	Preferable omega
1.2 Synthetic network (test case 2)	Synthetic network of 6 actors	Preferable omega
1.3 Synthetic network (test case 3)	Synthetic network of 13 actors	Preferable omega
2.1 Temporalized static networks	Static networks replicated into multiple time slices	Discernible pattern of modularity
2.2 Random edge removal	Static networks replicated into 1 time slice	Discernible pattern of community structure
3. Comparison of measured and modeled modularity	Temporal networks	Comparable modularity

4 Data

There are a total of 8 chosen data sets used in this study which are divided into two subgroups. The first subgroup consists of four temporal networks, i.e. networks with temporal data on the edges. The second subgroup are static networks, which consequently have no temporal data on the edges. These networks will be redesigned into temporal networks for a specific reason motivated in section 3.3.1. For each subsection, the networks of choice will be motivated and their characteristics will be accounted for. All data sets used in this thesis are social networks, even though one is fictional.

4.1 Temporal data sets

The four different data sets used in this thesis was chosen by the following two criteria. Firstly, they have time stamped edges. Secondly, they are small enough to run multiple times each without computationally taking too much time to run. The characteristics of each data set will be presented below.

4.1.1 Data set 1: Infectious, stay away

This network describes face-to-face interactions between visitors during the exhibition Infectious: stay away in 2009 at the Science Gallery in Dublin. Each edge is time stamped and represent a face-to-face contact lasting for at least 20 seconds. There are a total of 410 actors (visitors) and 17 298 edges (contacts). Edges are unique to their actors and time stamp, but the same two actors can have multiple contacts at different times (*Infectious network dataset – KONECT* 2017).

4.1.2 Data set 2: Hypertext 2009

Hypertext 2009 is the network of face-to-face contacts between visitors at the ACM Hypertext 2009 conference in Turin. The face-to-face contacts had to last for at least 20 seconds to be recorded. Each edge is time stamped and represents a contact between two visitors. The same two visitors can have multiple contacts at different times. There are a total of 113 actors (visitors) and 20 818 edges (contacts) (*Hypertext 2009 network dataset – KONECT* 2017)

4.1.3 Data set 3: Reality Mining (MIT)

This undirected network describe the contact between students at the Massachusetts Institute of Technology (MIT), collected through an experiment named *Reality Mining*. The experiment was done in 2004 and the data was collected during a period of 9 months, using 100 mobile phones, distributed to students. Actors represent students, and edges represent contacts between students. There are a total of 94 actors (students) and 1 086 404 edges (contacts) (*Reality Mining network dataset – KONECT* 2017).

4.1.4 Data set 4: Cambridge/Haggle

This network represents contacts between people measured by Bluetooth devices in different areas around the city of Cambridge, England. Each edge is time stamped and represents that two people were close in proximity for a small amount of time.

The network consists of a total of 274 actors (people) and 28 244 edges (contacts) (*Haggle network dataset – KONECT* 2017).

4.2 Static network data sets

The networks presented below are ordered in ascending order by their size, where the smallest network is smaller (in total number of actors) than any of the temporal networks. The largest static network is similarly larger than any of the temporal networks. All static networks have been chosen mainly because of their increasing size, but also to show that there exist a large diversity in *types* of networks. Information about the networks are presented in table 4.

Table 4: This table provides an overview over static networks chosen in this study.

Network	# Actors	# Edges	Information
Zachary Karate Club	34	78	Popular data set often used for community detection in static networks (*Zachary karate club network dataset – KONECT* 2017).
Train bombing	64	243	Contacts between suspected terrorists of the train bombing in Madrid, 2004 (*Train bombing network dataset – KONECT* 2017).
Highschool	70	366	A network of friendship between students at a high school in Illinois, US. The data was collected during a year, from 1957 to 1958 (*Highschool network dataset – KONECT* 2017).
Les Misérables	77	254	A network of co-occurrences between the characters of the novel "Les Misérables" by Victor Hugo (*Les Misérables network dataset – KONECT* 2017).

5 Results

The results are presented in four subsections. First, the result of the validity study of the adapted version of the Generalized Louvain are presented. Secondly, the results of the normalized modularity function are presented. Lastly, the results of applying the created algorithm to temporal networks are shown.

5.1 Omega on synthetic networks

This section is further divided into three subsections, one for each synthetic network. As mentioned in section 3.2.4, the networks are introduced in this section to make it easier for the reader to follow.

5.1.1 Test case 1: Synthetic network comprised of 4 actors

As seen in figure 14, the network is comprised of two communities, one in each time slice. Both of the time slices contain one time, t_1 and t_2 respectively. This implies that each edge — interaction between two actors — was active during that exact time. Running the OGL and the UGL on the network in figure 14 yielded the following result:

- $\omega = 0$: Both the OGL and the UGL found four communities, namely the two cliques in each time slice and two singular node communities:

$$\{A_1 : t_1,\ A_2 : t_1,\ A_3 : t_1\}, \quad \{A_4 : t_1\}$$

$$\{A_1 : t_2\}, \quad \{A_2 : t_2,\ A_3 : t_2,\ A_4 : t_2\},$$

- $\omega = 0.5$: Similarly as above, the OGL and the UGL gave the same results. They both found one large community, encompassing all nodes in the network:

$$\{A_1 : t_1,\ A_2 : t_1,\ A_3 : t_1,\ A_4 : t_1,\ A_1 : t_2,\ A_2 : t_2,\ A_3 : t_2,\ A_4 : t_2\}$$

- $\omega = 1$: Once again, as above, the OGL and the UGL yielded the same result, one large community:

$$\{A_1 : t_1,\ A_2 : t_1,\ A_3 : t_1,\ A_4 : t_1,\ A_1 : t_2,\ A_2 : t_2,\ A_3 : t_2,\ A_4 : t_2\}$$

The results of adding more time slices to the tests are presented below. Notice that if nothing else is stated, omega = 0.5 and omega = 1 yield the same result:

1. 1 extra time slice:

   - Omega = 0: Both the OGL and the UGL found similar communities with the difference that the UGL did not take the order of the time slices into consideration, which lead it to partition the communities in two random time slices. That being said, the *structure* of the communities were preserved, mirroring the structure of the OGL. All nodes outside of the found communities were partitioned into singular node communities. In total, six singular node communities were found, which was a result of all actors being replicated into the new time slice *without* any edges:

$$\{A_1 : t_1,\ A_2 : t_1,\ A_3 : t_1,\}, \quad \{A_4 : t_1\}$$

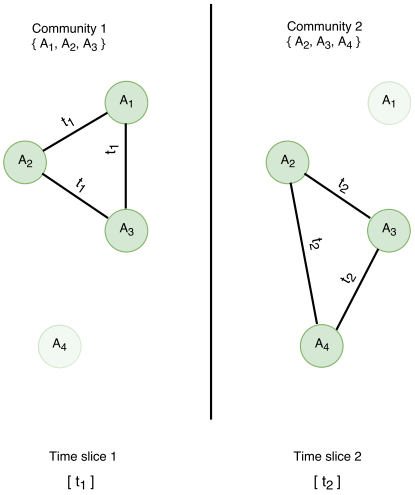


Figure 14: Visual representation of a small synthetic network. The temporal network was divided into two time slices containing four actors, denoted $A_i, i \in \{1, 2, 3, 4\}$. Both of the time slices had two communities (cliques) each and consisted of two sets of actors. Community 1 contained actor A_1, A_2 and A_3, while community 2 contained actor A_2, A_3 and A_4. Transparent actors, A_1 in time slice 2 and A_4 in time slice 1, implies that the nodes (layer specific actor) were accounted for by the Generalized Louvain, but did not have any active edges in those specific time slices.

$$\{A_2 : t_3,\ A_3 : t_3,\ A_4 : t_3,\},\quad \{A_1 : t_3\}$$
$$\{A_1 : t_2\},\ \{A_2 : t_2\},\{A_3 : t_2\},\ \{A_4 : t_2\}$$

- Omega = 1: Here the OGL found one large community, containing all nodes of the network. The UGL found four horizontal communities, i.e. it partitioned same-actor nodes spanning all three time slices:

$$\{A_1 : t_1,\ \ldots,\ A_4 : t_1,\ A_1 : t_2,\ \ldots,\ A_4 : t_2,\ \ldots,\ A_1 : t_3,\ \ldots,\ A_4 : t_3\} \quad (OGL)$$

$$\{A_1 : t_1,\ \ldots,\ A_1 : t_3\} \quad (UGL \quad Community\ 1)$$
$$\{A_2 : t_1,\ \ldots,\ A_2 : t_3\} \quad (UGL \quad Community\ 2)$$
$$\{A_3 : t_1,\ \ldots,\ A_3 : t_3\} \quad (UGL \quad Community\ 3)$$
$$\{A_4 : t_1,\ \ldots,\ A_4 : t_3\} \quad (UGL \quad Community\ 4)$$

2. 2 extra time slices:

   - Omega = 0: As with the case above with one extra time slice, both the OGL and the UGL found the two cliques, and partitioned the rest of the nodes into one community each. The only noticeable difference is that the four actors have been replicated in yet another time slice, resulting in 10 singular node communities.
   - Omega = 1: The OGL found one large community, spanning all time slices. In comparison, the UGL still found four *horizontal* communities, but spanning three time slices instead of the for the OGL found — which are the total number of time slices in this case. The nodes excluded from the horizontal communities are partitioned into singular node communities.

3. 6 extra time slices (figure 15):

   - Omega = 0: As with the cases above, the OGL and the UGL found the two cliques and the rest of the nodes in singular node communities. It is worth mentioning again that the communities the UGL found is not necessarily in the same time slices as the OGL. As previously mentioned this is a consequence of the UGL not being ordered, and communities are located in the time slice they have been stored in-memory.
   - Omega = 1: The behavior here is the same as the case with two extra time slices. The OGL found one large community spanning all time slices, while the UGL found four *horizontal* communities spanning *three* time slices.

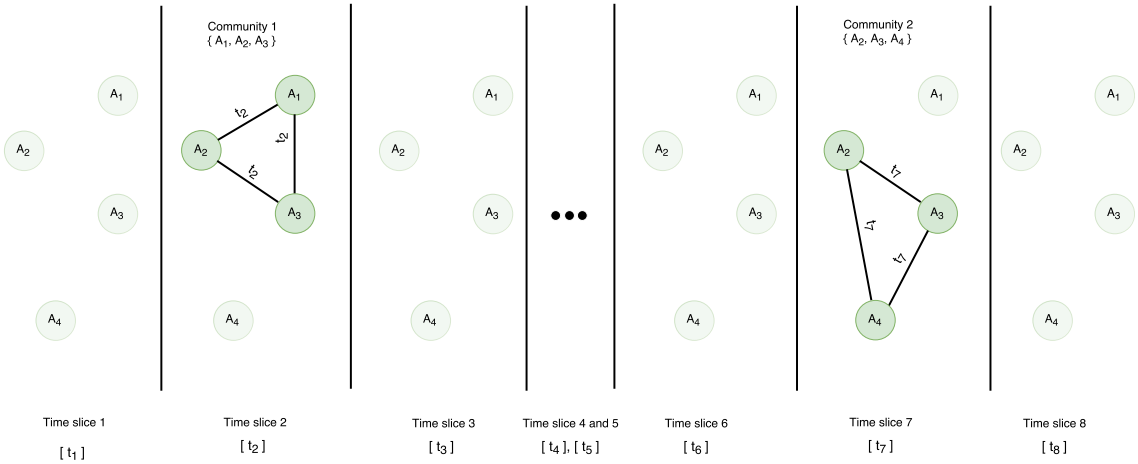


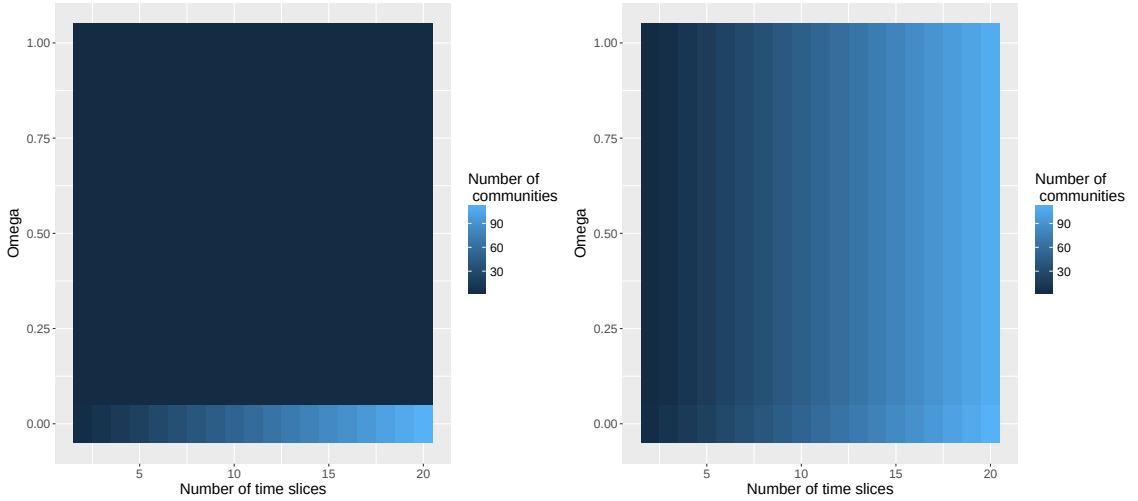
Figure 15: Visual representation of a small synthetic network. The temporal network was the same as the one in figure 14, but more time slices were added. Time slice 1, 3, 4, 5, 6 and 8 were empty, i.e. there were no edges connecting any of the nodes. Nodes were still present in these time slices, albeit without edges. Time slice 4 and 5 are not fully shown to save space, but have the same structure as any of the other empty time slices.

5.1.2 Test case 2: Synthetic network comprised of 6 actors

The second test was done on the synthetic network shown in figure 17. It consists of six individual actors, creating two cliques two different time slices. Similarly as in test case 1, the effect of omega was studied at first, and then time slices were added in conjunction with testing different values for omega. To make it easier for the reader to understand, the tests with increasing number of time slices are presented in a graph, showing the relationship of omega and the number of time slices for the network in figure 17. Running the OGL and the UGL on the network of study yielded the following result:

- Omega = 0: Both the OGL and the UGL found four communities, i.e. community C_1, C_2, C_3 and C_4 (as seen in figure 17).
- Omega = 0.5: The OGL and the UGL found one large community spanning all nodes in the network.
- Omega = 1: Same results as for omega = 0.5. One large community with all nodes are found by both versions.

As mentioned above, the results will now be presented with a graph, showing the number of communities found. Omega and the number of time slices were increased iteratively and the number of communities found are shown for each two values (see figure 16). Both the OGL and the UGL followed the same pattern as described in the beginning of this test case and therefore also in test case 1. When omega is 0, the other time slices have no impact on the structure of the partitioned communities, i.e. both the OGL and the UGL found the four cliques (C_1, C_2, C_3 and C_4) and partitioned the rest of the existing nodes into singular node communities. As the number of time slices were increased, the actors were replicated into each new time slice, creating a linear pattern for the number of communities found by the algorithms. All dark tiles in 16a indicate that the OGL found two communities. It then partitioned community C_1 and C_2 together with all nodes of the same three actors A_1, A_2 and A_3, replicated over all intermediate time slices. In comparison, the UGL displayed a linear pattern over all time slices and values of omega. The UGL followed the same pattern as in test case 1. It found six *horizontal* communities (one for each actor), spanning three time slices where two contain the constructed cliques. The rest of the nodes were partitioned into singular node communities.



(a) Ordered Generalized Louvain *(b) Unordered Generalized Louvain*

Figure 16: Generalized Louvain run on the synthetic network in figure 17. Showing the number of communities found using the OGL (a) and UGL (b) respectively, while ω was varied in the interval of [0,1], incremented with steps of 0.1. The number of time slices were increased from 2 to 20.

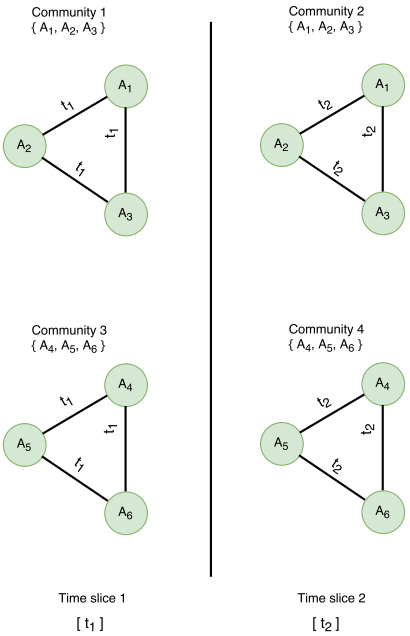


Figure 17: Visual representation of a constructed synthetic network. The temporal network was divided into two time slices containing six actors, denoted $A_i, i \in \{1, 2, 3, 4, 5, 6\}$. Both time slices had two communities (cliques) each, that consisted of three actors respectively, i.e. community 1 and 2 contained the same actors and community 3 and 4 contained the same actors.

5.1.3 Test case 3: Synthetic network comprised of 13 actors

The third test was done on the synthetic network shown in figure 18. The results are presented below.

- Omega = 0: Both the OGL and the UGL found the two cliques and partitioned

the rest of the nodes into singular node communities. In total, the algorithms found eight communities:

$$\{A_1 : t_1,\ \dots\ ,\ A_{10} : t_1\}$$
$$\{A_4 : t_2,\ \dots\ ,\ A_{13} : t_2\}$$
$$\{A_1 : t_2,\}\ ,\{A_2 : t_2\}\ ,\{A_3 : t_2\}$$
$$\{A_{11} : t_1,\}\ ,\{A_{12} : t_1\}\ ,\{A_{13} : t_1\}$$

- Omega = 0.5: The OGL and the UGL partitioned everything into one large community:
$$\{A_1 : t_1,\ \dots\ , A_{13} : t_1,\ A_1 : t_2,\ \dots\ ,\ A_{13} : t_2\}$$
- Omega = 1: As above, both algorithms found one large community.

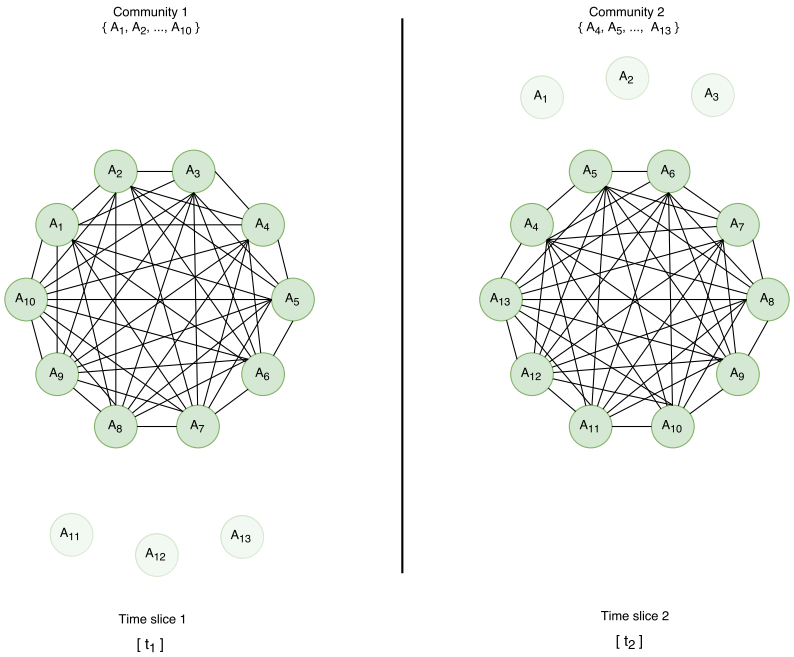
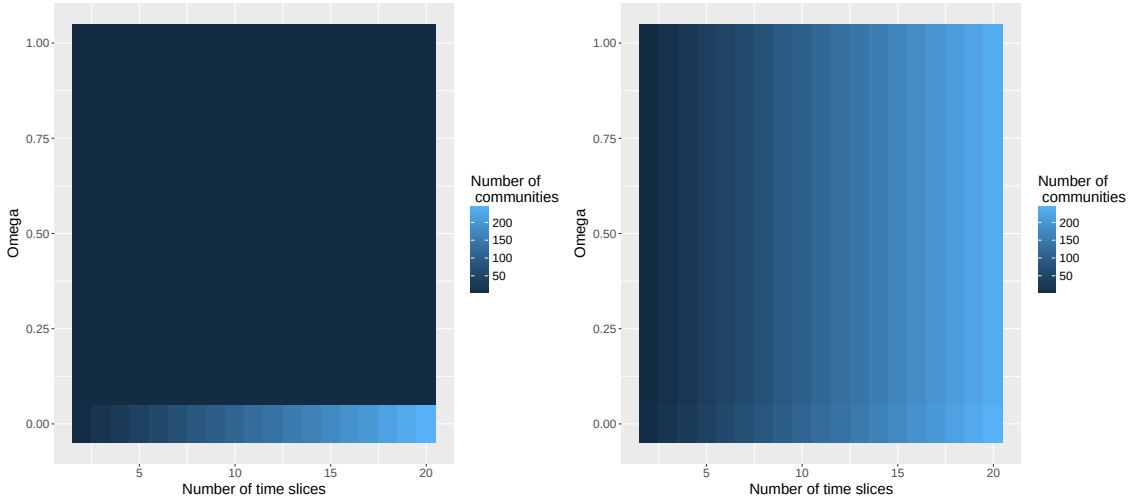


Figure 18: Visual representation of a constructed synthetic network. The temporal network were divided into two time slices containing 13 actors, denoted $A_i, i \in \{1, 2, ..., 13\}$. Both time slices have two communities (cliques) each, consisting of the 10 actors respectively. In community 1 (C_1), actor A_1, ..., A_{10} created a clique, and in community 2 (C_2), actor A_4, ..., A_{13} created another clique. All edges in time slice 1 had a time of t_1, and all edges in time slice 2 had a time of t_2. Transparent actors were replicated nodes without edges to any other actor within the same time slice.

To further study the effect of omega on the network, omega and the number of time slices will be increased iteratively. This results in a similar pattern as described in test case 2. In comparison, the number of communities found are more, but the structure of the found communities follow the same pattern as described in both previous test cases. That all three tests yielded similar results, i.e. a pattern where all omega above 0 behave similarly, serve as an indication that there is no support for any specific omega. Therefore, and When taking the recommendation of the authors into account, omega will henceforth be set to 1 in the rest of this thesis.



(a) Ordered Generalized Louvain *(b) Unordered Generalized Louvain*

Figure 19: Generalized Louvain run on the synthetic network in figure 18. Showing the number of communities found using the OGL (a) and UGL (b) respectively, letting ω vary in the interval of [0,1], incrementing with steps of 0.1. The number of time slices are increased from 2 to 20.

5.2 Modularity of temporalized static networks

The pattern that emerges from replicating the same static network (temporalized) in an increasing number of time slices is the similar for all data sets (see figure 20). The modularity levels out relatively fast — after being replicated into around 10 time slices — with the exception that the highest modularity is different for each network. Since modularity is a scalar value in the interval of $Q_{multislice} \in [-1, 1]$ and is dependent on the number of nodes and edges in the network, these values are not comparable. Modularity serves as an indicator of "goodness of fit", i.e. how well the network has been partitioned (or how *modular* it is). To be able to compare modularity the effect of the number of time slices and the number of edges had to be generalized. The results of the exploratory study — incorporating the normalizations of the modularity function from section 3.3.1 — is presented below in two steps. First, the results of removing the effect of the number of time slices is shown. Secondly, the results of removing the effect of the number of edges is presented.

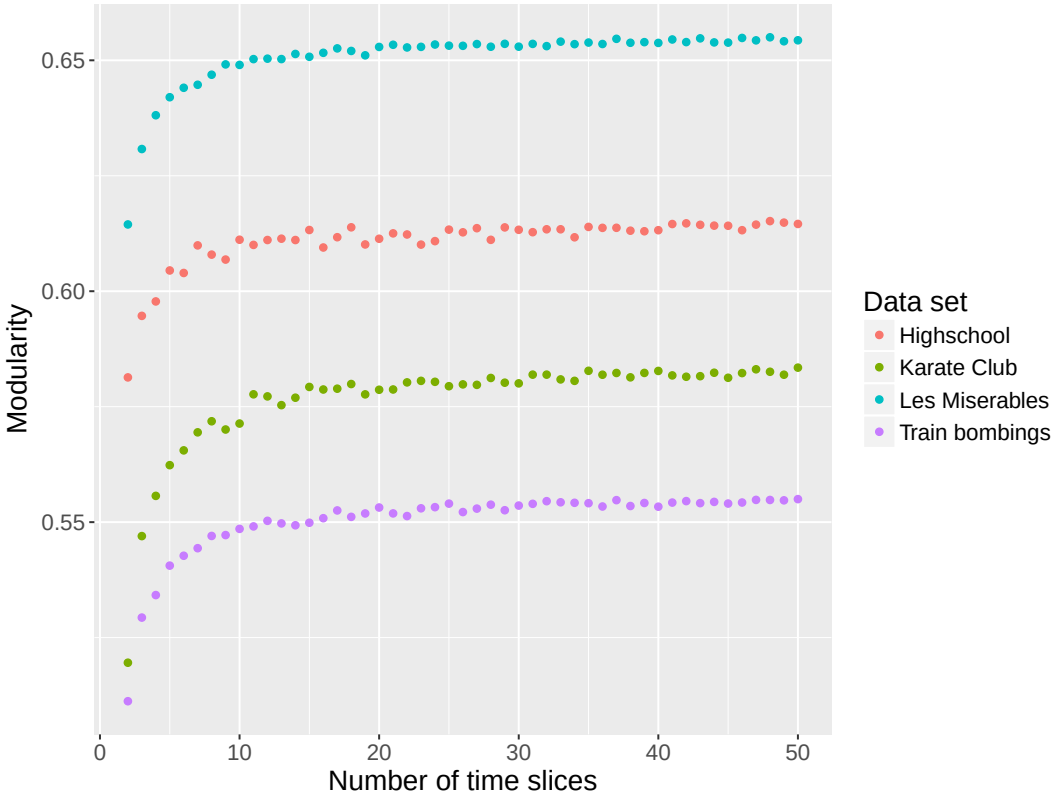
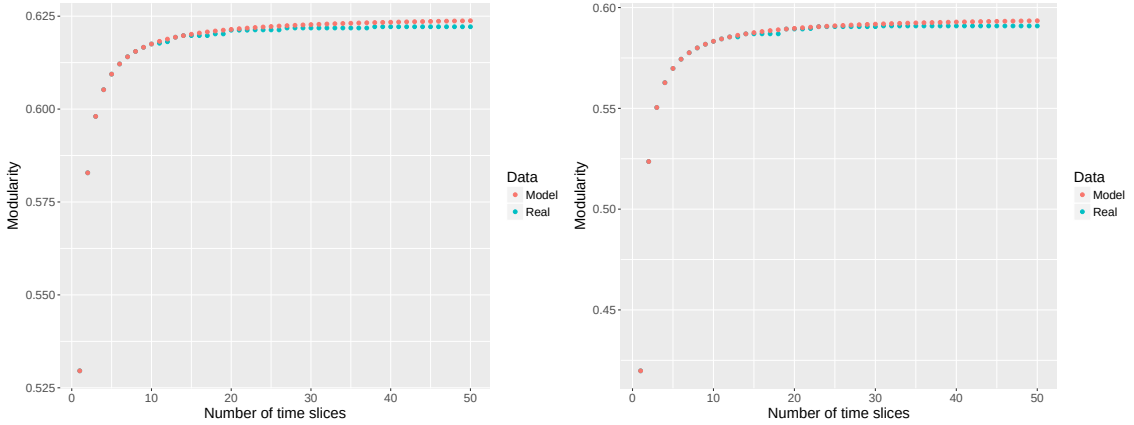


Figure 20: Ordered Generalized Louvain run on all temporalized static data sets. The maximum modularity $Q_{multislice}$ was extracted from 30 runs of each number of time slices.

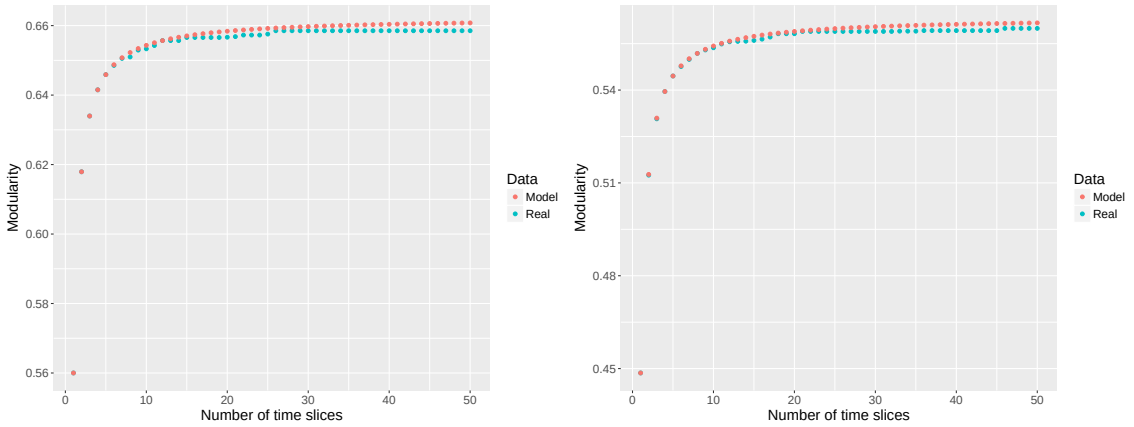
5.2.1 Normalizing the effect of the number of time slices on modularity

Figure 21 show the maximum modularity found by the OGL for the temporalized networks. What can be seen is that the model (see section 3.3.1 equation 9) has a higher expected modularity than the results from the OGL (annotated "Real" in the figures). The higher modularity in the model can be explained by the underlying assumption, which is that the maximum modularity for one time slice (see M_1 in equation 9 in section) has a linear correlation with the increase of the number of time slices.



(a) Highschool

(b) Zachary Karate Club



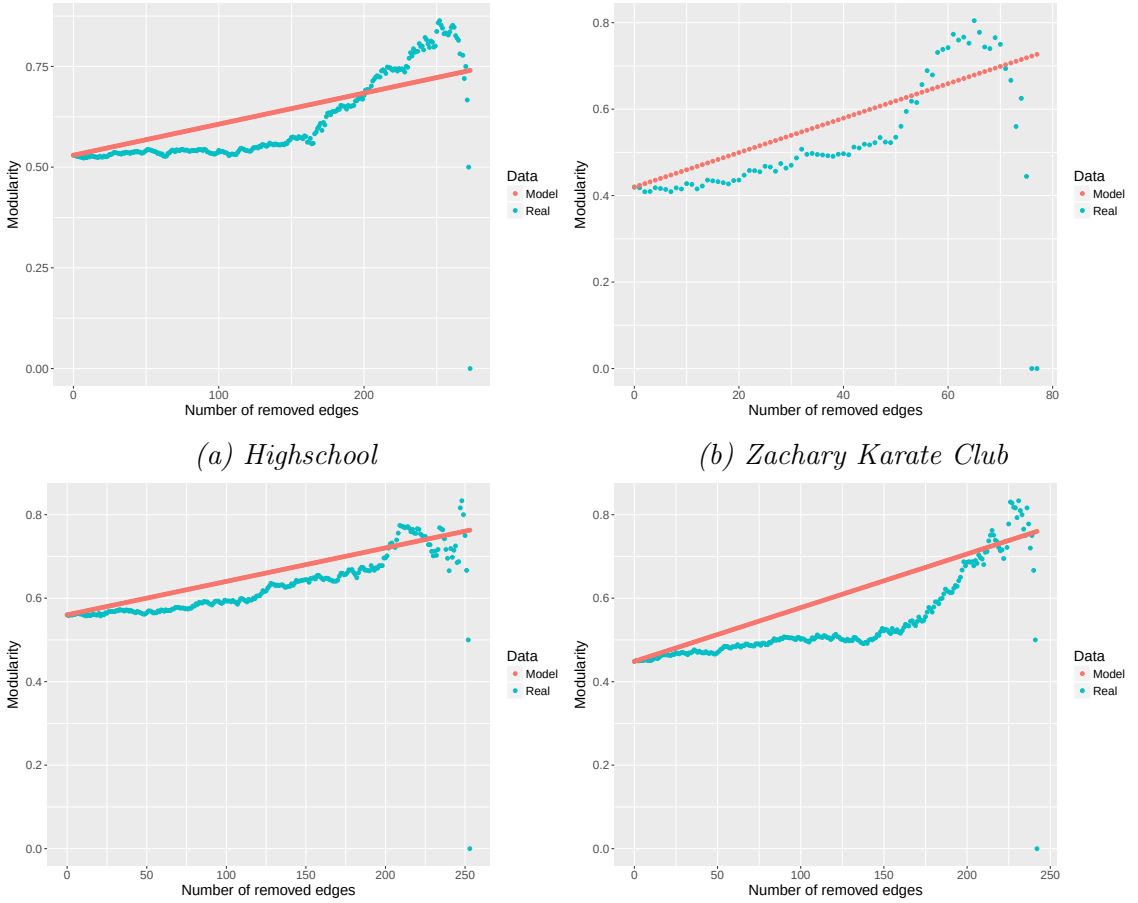
(c) Les Misérables

(d) Train bombings

Figure 21: The total modularity for each of the temporalzied static networks. The model is shown in orange while the results of the OGL is shown in light blue under the label "real". Notice the different maximum modularity each network attains.

5.2.2 Normalizing the effect of the number of edges on modularity

The pattern that emerges in figure 22 is similar for all temporalized networks. As mentioned in section 3.3.1, the underlying assumption for the edge removal model was that the community structure should be preserved if one random edge was removed at a time. The fit of the model is not perfect but describes the general trend of the effect of removing random edges from a network. After removing about two-thirds of all edges, each network show similar results; a sudden increase in modularity, then a drop to 0.



(a) Highschool

(b) Zachary Karate Club

(c) Les Misérables

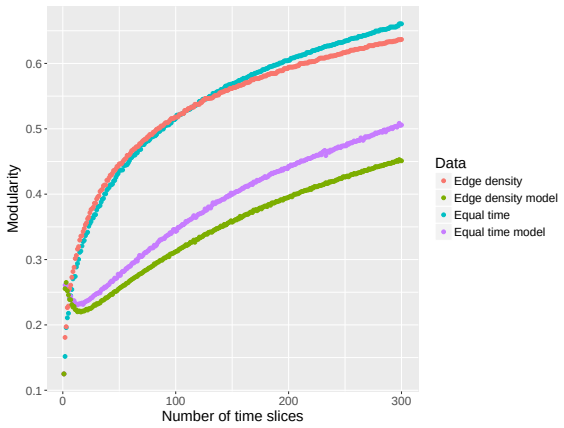
(d) Train bombings

Figure 22: The modularity for each of the temporalized static networks. Each network was replicated into one *time slice. Then the edges were removed iteratively, storing the maximum modularity attained over 30 runs of the OGL. The model is shown in orange while the results of the OGL is shown in light blue under the label "real". Notice the different maximum modularity each network attains and the different number of edges each network contains.*

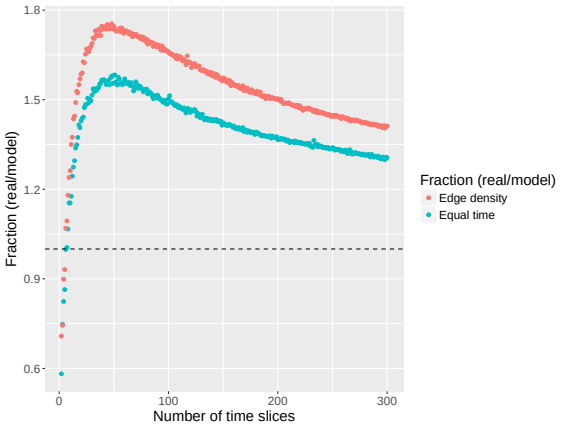
5.3 Applying the model to temporal networks

In figure 23a, the measured and expected modularity is plotted against the number of time slices that the temporal networks were partitioned into. The two partitioning methods were used separately, with their respective models created. In figure 23b, the measured modularity was divided by the modularity for the models for both partitioning methods respectively. This resulted in two peaks at 48 and 50 time slices for the *edge density* and *equal time* fractions respectively. Notice that the fraction for one time slice has been removed from the graph since it is always 1. That is a result of the model being built on the starting values of the overall modularity (Q_1 in equation 9).

For the Hypertext network, a similar pattern emerges, albeit with different curves. The modularity is overall higher for the Hypertext network in comparison to the MIT network. The results for the data set Infectious: Stay away are less conclusive. For the first 10 time slices in figure 25a, both models behave differently from the

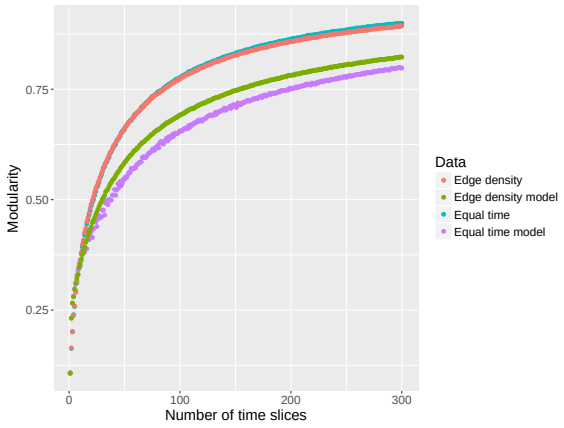


(a) Measured and expected modularity of the OGL and the model respectively.

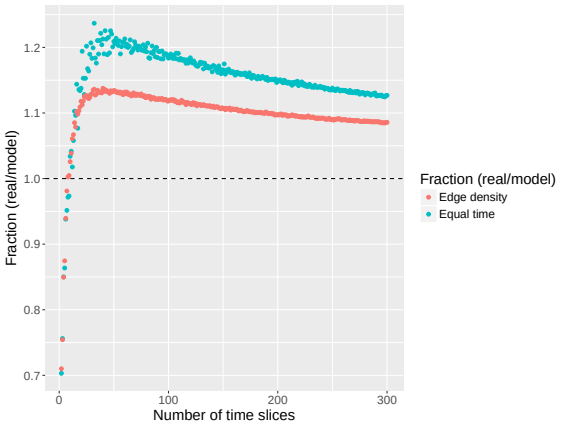


(b) Fraction of the measured and expected modularity of the OGL and the model respectively.

Figure 23: For the MIT data set. The figure in a) show the measured modularity of the OGL based on the two different ways of accumulating time slices (edge density and equal time). The model has two different inputs for the number of edges in each time slice, and similarly has two different results. The figure in b) show the fraction of the measured ("real" in the figure) and the expected ("model" in the figure) modularity.



(a) Measured and expected modularity of the OGL and the model respectively.

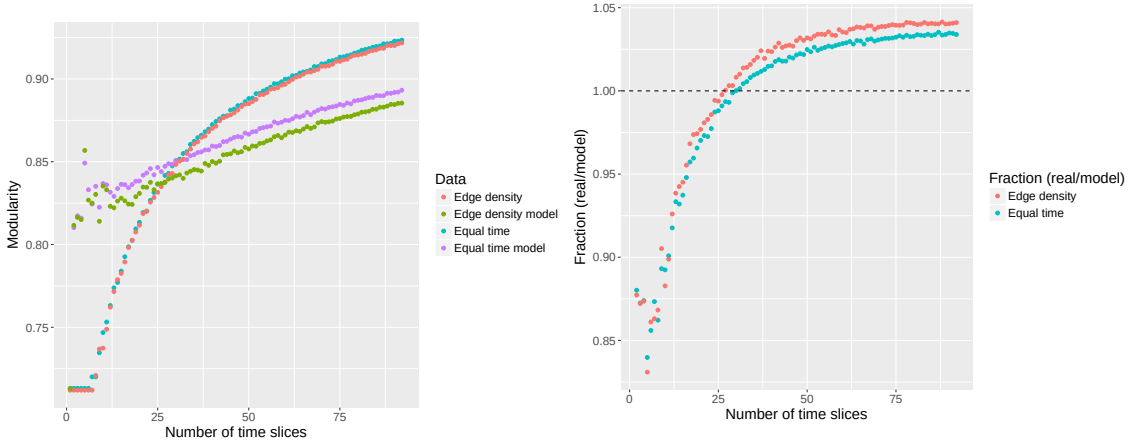


(b) Fraction of the measured and expected modularity of the OGL and the model respectively.

Figure 24: For the Hypertext data set. The figure in a) show the measured modularity of the OGL based on the two different ways of accumulating time slices (edge density and equal time). The model has two different inputs for the number of edges in each time slice, and similarly has two different results. The figure in b) show the fraction of the measured ("real" in the figure) and the expected ("model" in the figure) modularity.

measured modularity. This can either be a result of how many of the edges that were partitioned into the first time slice or that the community structure is strong enough to be maintained over the first 10 time slices. The growth of the measured modularity start to decline at around 50 time slices, while the model continues an

almost linear growth. This serves as an indication that the optimal number of time slices will be attained at a higher number of time slices, i.e. when the model grows faster than the measured modularity. Since this data set contains 410 actors, the modularity matrix quickly grows in size and further increasing the number of time slices was not feasible on the machine that the tests were run on.

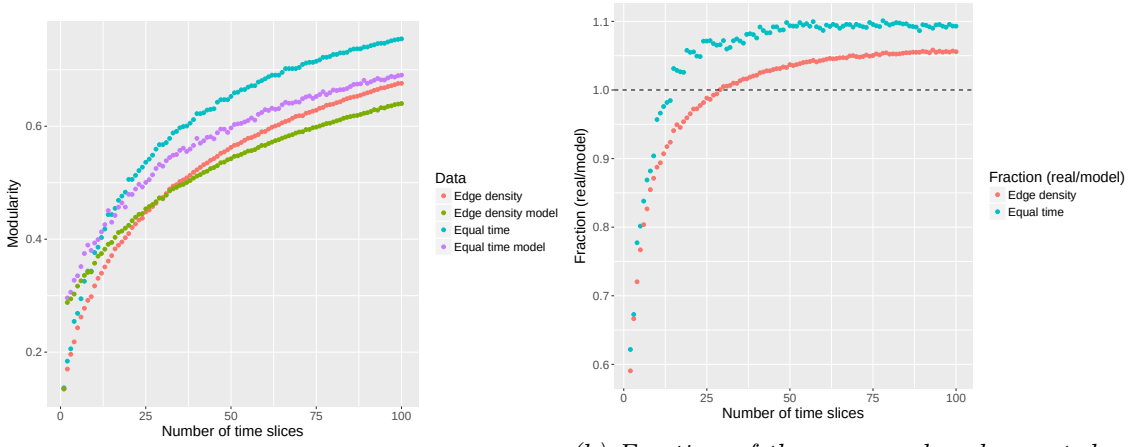


(a) Measured and expected modularity of the OGL and the model respectively.

(b) Fraction of the measured and expected modularity of the OGL and the model respectively.

Figure 25: For the data set Infectious: Stay away. The figure in a) show the measured modularity of the OGL based on the two different ways of accumulating time slices (edge density and equal time). The model has two different inputs for the number of edges in each time slice, and similarly has two different results. The figure in b) show the fraction of the measured ("real" in the figure) and the expected ("model" in the figure) modularity.

For the data set Cambridge/Haggle, a similar pattern emerges as for the data set Infectious: Stay away. This data set contains more actors than either of the MIT or the Hypertext data sets. It thus becomes computationally heavy above 100 time slices. What can be discerned though is that for partitioning with *equal time*, the curve levels out quicker than for *edge density*, which hints at a possible peak.



(a) Measured and expected modularity of the OGL and the model respectively.

(b) Fraction of the measured and expected modularity of the OGL and the model respectively.

Figure 26: For the data set Cambridge/Haggle. The figure in a) show the measured modularity of the OGL based on the two different ways of accumulating time slices (edge density and equal time). The model has two different inputs for the number of edges in each time slice, and similarly has two different results. The figure in b) show the fraction of the measured ("real" in the figure) and the expected ("model" in the figure) modularity.

6 Discussion

This section is divided into three subsections. First, the impact of setting the parameter omega is discussed. Secondly, a discussion about the results from normalizing the modularity function is presented. Lastly, the completed algorithm is discussed.

6.1 The parameter omega

The results of the study of omega was presented using the found community structure in conjunction with the measure of how many communities were found. From the results it was visible that for all synthetic networks, the OGL performed similarly when iterating over the number of time slices and omega. Whenever omega passes the threshold of 0, i.e. in the interval of [0.01, 1], the algorithm produces *horizontal* communities. With the support of such a created community structure and the fact that omega = 1 is considered to be the standard, the research preceded with this setting. Furthermore, setting omega to a value of over 1 was never considered. An omega above one would the imply that actors are clustered into communities based on both past and *future* interactions. While the former is most likely probable, i.e. that previous actions affect future in social networks, the latter is not. When taking time slicing into account, it would seem more prudent to let the actors and edges within the same time slice affect each other more, or at least equal, than actors/edges from other, adjacent time slices.

Comparing the validity of the OGL in comparison to the UGL yielded a clear difference in the community structure. The OGL created fully horizontal communities, while the UGL did not. The difference is most likely an effect of how every time slice affect all others in the UGL. Therefore the latter of the two algorithms is not equally prone to create horizontal communities. That being said, the creation of horizontal communities by the OGL can be seen as a type of validity in itself, since it implies that same-actor nodes are being partitioned into the same community. Comparing to a real-life situation this does seem probable for, at least, social temporal networks. If communities are created with sole constraint of time, then naturally each node should be partitioned into the same community. It is still the same *actor* after all, only some time have passed.

The results from the study of omega on synthetic networks yielded that there were no apparent choice of omega which was preferable as long as omega was above 0. As mentioned in section 3.2.4, if no preferable omega was found the recommended value of 1 would be used for the continued study of the temporal networks used in this thesis. That being said, it is important to mention yet again that the creators of the Generalized Louvain *did* recommend choosing a suitable for every studied network individually. Although that might have been beneficial, the scope of this thesis prevented a deeper study of each temporal network, i.e. finding a preferable omega for each temporal network here studied. Furthermore, the parameter omega is subjective to the study since different values yield at times different community structures and are thus dependent on the wanted results.

6.2 Normalizing the modularity function

The intuition behind normalizing the modularity function is based on the thought that increasing the number of time slices also increases the modularity. This is a consequence of the quality function itself. Inter-layer (inter-time slice) edges are added each time a new time slice is added. Furthermore, since the input data of this thesis is undirected networks, each inter-layer edge is counted twice, thus increasing the modularity when more time slices are added. That being said, all increase in modularity cannot be attributed to time slices alone. Therefore, studying the effect of the number of edges existing in the network could attribute the increase of modularity. In total, the created model were built on two major underlying assumptions, using temporalized networks: 1) the exact same community structure is being replicated into each time slice and 2) removing one random edge at a time will preserve the community structure (within certain limits). For the first assumption, the created model behaved very similar to the measured modularity (see figure 21). That the measured modularity is slightly lower than the model might be attributed to the fact that the community structure was *not* fully preserved. Thus, the measured modularity was slightly lower than the expected from the model. For the second assumption, the expected results from the model were less similar to the measured modularity. This is a consequence of the model not taking the randomness into account. When community detection is done with the OGL, the algorithm calculates the modularity based on the number of edges existing within communities as well as the *degree* of each node in each community. However, these values can be different from one run to another, implying an underlying randomness, dependent on which node the Generalized Louvain start on.

Since the Generalized Louvain is a greedy algorithm, it stops when it finds a local maximum of the modularity. Generally, this implies that most solutions are unique, and therefore the community structure will be somewhat different from one run to another. As a result, predicting the modularity with the removal of edges was not as apparent as the underlying reasoning implied. In reality, the removal of edges did not fully preserve the community structure, since the modularity was more often than not constant when removing up to 50 percent of the edges (see figure 22). If studying equation 10 in section 3.3.1, it becomes apparent that the second term of the sum $(\frac{k_i k_j}{2m} * \frac{m'}{m^2})$ quickly goes towards 0 as m' represent the number of edges *remaining* in the data set. A notable difference is that the model for edge removal show a linear correlation between the modularity and the number of removed edges. In reality, it is not feasible to remove more edges than there exists in the network, therefore the model is constrained by that fact. That being said, what the edge removal model generally did good was to predict the modularity when almost all edges were removed.

6.3 Optimal number of time slices

The created estimation model of modularity of temporal networks was based on temporalizing static networks. This was done under two underlying assumptions mentioned above. Applying a model based on assumptions taken from a known community structure does not necessarily translate to estimating the modularity

for any temporal network. As mentioned in section 3.2, the modularity is a scalar value, based on the number of existing edges and nodes in a network. Therefore, a modularity attained through the Generalized Louvain for one network *cannot* be compared to the modularity of another network. Take two different networks for example, even if they contain the same number of edges and nodes, the modularity will most probably be different since the *structure* of the network is most likely different. In a random network, the modularity will be close to -1, since it effectively do not have any measurable communities. A part of the Generalized Louvain is that it compares the measured number of edges and edge degrees in a community, to that of a *null model*. The null model can be built on different assumptions, but the one used in the Generalized Louvain (Newman-Girvan) calculates the probability that a certain node is in a community in comparison to a random network. Therefore, measuring the modularity of a random network is akin to measuring the modularity of a random network against a random network. The result of -1 comes from the idea that any random network is as different to the expected null model as possible. With that in mind, each partitioning of a temporal network, being 20 time slices or 100 time slices, creates it's own unique network. The modularity of one specific partitioning is therefore not comparable to another partitioning.

When partitioning the temporal networks studied in an increasing number of time slices, the modularity increased as well. Much of the increase can be attributed to the increasing number of inter-layer edges, i.e. new edges are being created since more time slices are added. The inter-layer edges are represented by the last term of the sum in the quality function for modularity: $\sum_{ijs} C_{ijs}$. When comparing the measured modularity to the estimated modularity attained through the model, the effect of increasing the number of time slices could essentially be factored out from the measured modularity. For the two networks *Hypertext* and *MIT*, the expected modularity grows faster than the measured modularity (see figure 24 for the Hypertext network). The fraction between the measured modularity and the expected, thereby removes the effect of increasing the number of time slices.

When the fraction between them hit a peak, an optimal number of time slices had been attained. The results were not conclusive for the networks *Infectious* and *Cambridge/Haggle* though. A possible explanation can be that the network structure itself affected the results. Another possible explanation is that the time line of the two networks affect the partitioning. In that case it could be favorable to introduce other types of partitioning methods. However, there still exist a possibility that an optimal number of time slices can be found at a higher number of time slices for those networks, but it is computationally heavy to iterate to a higher number.

7 Conclusion

The model created in this thesis is based on two assumptions. Both of the assumptions are quite strong in character, i.e. they might simplify the quality function used in the Generalized Louvain too much. However, it is a good start for creating a comparable measure and in extension, finding an optimal partition of potentially any temporal network.

When it comes to two out of the four temporal networks used in this thesis, the created approach fails to find an optimal number of time slices. As mentioned in the discussion it can depend on a few things. A possibility is that it is a consequence of the number of the structure of the network. Another possibility is that the two partitioning methods does not work for all networks. Community detection is no an exact science, and it is often difficult saying that a certain communities are correct where others are not. Commonly, newly created community detection algorithms are tested on social "ground truth"-networks, i.e. networks where the community structure is said to be known. However, those types of social networks have been recorded by people of people. There is not necessarily a "correct" community existing in those networks, but rather what people *believe* should exist. This can serve as an indication that community detection is *not* an exact science. Many community detection approaches find different results in the same networks.

For the two networks *MIT* and *Hypertext*, an optimal number of time slices were attained. This gives an indication that is is possible to use a measure of the community structure comparing different partitions of temporal networks among themselves. It is not necessarily the measure used that is important, but rather adapting it for comparisons. There are a lot of unknown factors when dealing with networks, especially when they have a temporal dimension. Most of the algorithms used in temporal network science today have been adapted from static networks and thus need added functionality to handle time. Much like the algorithm created here.

7.1 Future work

Since the developed approach is a naive, i.e. the optimal number of time slices are attained through iterating over a large amount of them, there is a lot of potential work to be done. If an extension of the approach could be created that could "guess" a potentially good number of time slices, then it could potentially reduce the computational complexity of the problem. Furthermore, it would enable for study of larger networks, including networks with a more extensive time line.

Another interesting approach would be to study *why* the created algorithm works on some temporal networks, and not others. Potentially, a solution could lie in studying the characteristics of each network. They are numerous and various standardized measures are commonly used. The next step is then to understand what makes the algorithm succeed or fail.

Lastly, it would be interesting try a similar approach using another algorithm as base, e.g. stochastic block modeling. Even though such an approach might yield different results (for the optimal number of time slice), it could give hints as to which network characteristics affect time slicing.

References

Bassett, D. S., Wymbs, N. F., Porter, M. A., Mucha, P. J., Carlson, J. M. & Grafton, S. T. (2010), 'Dynamic reconfiguration of human brain networks during learning', *Proceedings of the National Academy of Sciences* **108**(18), 7641.

Bazzi, M., Porter, M. A., Williams, S., McDonald, M., Fenn, D. J. & Howison, S. D. (2014), 'Community detection in temporal multilayer networks, and its application to correlation networks'.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. (2008), 'Fast unfolding of communities in large networks', *Journal of Statistical Mechanics: Theory and Experiment* **2008**(10), P10008.

Fortunato, S. (2010), 'Community detection in graphs', **486**(3-5), 75–174.

González-Bailón, S., Borge-Holthoefer, J., Rivero, A. & Moreno, Y. (2011), 'The Dynamics of Protest Recruitment through an Online Network'.

Guo, F., Hanneke, S., Fu, W. & Xing, E. P. (2007), 'Recovering temporally rewiring networks: A model-based approach', pp. 321–328.

Haggle network dataset – KONECT (2017).
URL: *http://konect.uni-koblenz.de/networks/contact*

He, J., Chen, D., Sun, C., Fu, Y. & Li, W. (2017), 'Efficient stepwise detection of communities in temporal networks', *Physica A: Statistical Mechanics and its Applications* **469**, 438–446.

Highschool network dataset – KONECT (2017).
URL: *http://konect.uni-koblenz.de/networks/moreno_highschool*

Holme, B. P. (2014), 'Analyzing Temporal Networks in Social Media', **102**(12).

Holme, P. & Saramäki, J. (2011), 'Temporal Networks', *Arxiv preprint arXiv:1108.1780* pp. 1–25.

Hypertext 2009 network dataset – KONECT (2017).
URL: *http://konect.uni-koblenz.de/networks/sociopatterns-hypertext*

Infectious network dataset – KONECT (2017).
URL: *http://konect.uni-koblenz.de/networks/sociopatterns-infectious*

Jeub, L. G. S., Bazzi, M., Jutla, I. S. & Mucha, P. J. (2011-2017), 'A generalized louvain method for community detection implemented in matlab'.
URL: *http://netwiki.amath.unc.edu/GenLouvain*

Kuncheva, Z. & Montana, G. (2015), 'Community detection in multiplex networks using locally adaptive random walks'.

Les Misérables network dataset – KONECT (2017).
URL: *http://konect.uni-koblenz.de/networks/moreno_lesmis*

Matias, C. & Miele, V. (2017), 'Statistical clustering of temporal networks through a dynamic stochastic block model', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **79**(4), 1119–1141.

Mucha, P. J., Richardson, T., Macon, K., Porter, M. A. & Onnela, J.-P. (2010), 'Community structure in time-dependent, multiscale, and multiplex networks.', *Science (New York, N.Y.)* **328**(5980), 876–8.

Reality Mining network dataset – KONECT (2017).
URL: *http://konect.uni-koblenz.de/networks/mit*

Sarzynska, M., Leicht, E. A., Chowell, G. & Porter, M. A. (2014), 'Null Models for Community Detection in Spatially-Embedded, Temporal Networks', pp. 1–35.

Tang, X. & Yang, C. C. (2014), 'Detecting Social Media Hidden Communities Using Dynamic Stochastic Blockmodel with Temporal Dirichlet Process', *ACM Transactions on Intelligent Systems and Technology* **5**(2), 1–21.

Tantipathananandh, C. & Berger-wolf, T. (n.d.), 'Dynamic network generative model', pp. 1–5.

Train bombing network dataset – KONECT (2017).
URL: *http://konect.uni-koblenz.de/networks/moreno_train*

Wang, W., Jiao, P., He, D., Jin, D., Pan, L. & Gabrys, B. (2016), 'Autonomous overlapping community detection in temporal networks: A dynamic Bayesian nonnegative matrix factorization approach', *Knowledge-Based Systems* **110**, 121–134.

Xu, K. S., Kliger, M. & Hero, A. O. (2014), 'Adaptive evolutionary clustering', *Data Mining and Knowledge Discovery* **28**(2), 304–336.

Yang, T., Chi, Y., Zhu, S., Gong, Y. & Jin, R. (2011), 'Detecting communities and their evolutions in dynamic social networks - A Bayesian approach', *Machine Learning* **82**(2), 157–189.

Yogeeswaran, K., Nash, K., Sahioun, R. & Sainudiin, R. (n.d.), 'Seeded by Hate? Characterizing the Twitter Networks of Prominent Politicians and Hate Groups in the 2016 US Election'.

Zachary karate club network dataset – KONECT (2017).
URL: *http://konect.uni-koblenz.de/networks/ucidata-zachary*

Zhao, Q., Tian, Y., He, Q., Oliver, N., Jin, R. & Lee, W.-C. (2010), 'Communication motifs: A Tool to Characterize Social Communications', *Proceedings of the 19th ACM international conference on Information and knowledge management - CIKM '10* p. 1645.