arXiv:1503.00278v1 [cs.DM] 1 Mar 2015

An Introduction to Temporal Graphs: An Algorithmic Perspective*

Othon Michail

Computer Technology Institute & Press "Diophantus" (CTI),
N. Kazantzaki Str., Patras University Campus,
Rio, P.O. Box 1382, 26504,
Patras, Greece
Email: `michailo@cti.gr`
Phone: +30 2610 960300

Abstract. A *temporal graph* is, informally speaking, a graph that changes with time. When time is discrete and only the relationships between the participating entities may change and not the entities themselves, a temporal graph may be viewed as a sequence $G_1, G_2 \ldots, G_l$ of static graphs over the same (static) set of nodes V. Though static graphs have been extensively studied, for their temporal generalization we are still far from having a concrete set of structural and algorithmic principles. Recent research shows that many graph properties and problems become radically different and usually substantially more difficult when an extra time dimension in added to them. Moreover, there is already a rich and rapidly growing set of modern systems and applications that can be naturally modeled and studied via temporal graphs. This, further motivates the need for the development of a temporal extension of graph theory. We survey here recent results on temporal graphs and temporal graph problems that have appeared in the Computer Science community.

DO NOT
nodes change
edges change with
time

1 Introduction

The conception and development of graph theory is probably one of the most important achievements of mathematics and combinatorics of the last few centuries. Its applications are inexhaustible and ubiquitous. Almost every scientific domain, from mathematics and computer science to chemistry and biology, is a natural source of problems of outstanding importance that can be naturally modeled and studied by graphs. The 1736 paper of Euler on the Seven Bridges of Königsberg problem is regarded as the first formal treatment of a graph-theoretic problem. Till then, graph

* Supported in part by the project "Foundations of Dynamic Distributed Computing Systems" (FOCUS) which is implemented under the "ARISTEIA" Action of the Operational Programme "Education and Lifelong Learning" and is co-funded by the European Union (European Social Fund) and Greek National Resources.

theory has found applications in electrical networks, theoretical chemistry, social network analysis, computer networks (like the Internet) and distributed systems, to name a few, and has also revealed some of the most outstanding problems of modern mathematics like the four color theorem and the traveling salesman problem.

Graphs simply represent a set of objects and a set of pairwise relations between them. It is very common, and shows up in many applications, the pairwise relations to come with some additional information. For example, in a graph representing a set of cities and the available roads from each city to the others, the additional information of an edge (C_1, C_2) could be the average time it takes to drive from city C_1 to city C_2. In a graph representing bonding between atoms in a molecule, edges could also have an additional bond order or bond strength information. Such applications can be modeled by weighted or, more generally, by labeled graphs, in which edges (and in some cases also nodes) are assigned values from some domain, like the set of natural numbers. An example of a classical, very rich, and well-studied area of labeled graphs is the area of graph coloring [MR02].

edges hold
additional
information

Temporal graphs (also known as *dynamic*, *evolving* [Fer04], or *time-varying* [FMS09, CFQS12] graphs) can be informally described as *graphs that change with time*. In terms of modeling, they can be thought of as a special case of labeled graphs, where labels capture some measure of time. Inversely, it is also true that any property of a graph labeled from a discrete set of labels corresponds to some temporal property if interpreted appropriately. For example, a proper edge-coloring, i.e. a coloring of the edges in which no two adjacent edges share a common color, corresponds to a temporal graph in which no two adjacent edges share a common time-label, i.e. no two adjacent edges ever appear at the same time. Still, the time notion and the rich domain of modern applications motivating its incorporation to graphs, gives rise to a brand new set of challenging, important, and practical problems that could not have been observed from the more abstract perspective of labeled graphs.

Though the formal treatment of temporal graphs is still in its infancy, there is already a huge identified set of applications and research domains that motivate it and that could benefit from the development of a concrete set of results, tools, and techniques for temporal graphs. A great variety of both modern and traditional networks such as information and communication networks, social networks, transportation networks, and several physical systems can be naturally modeled as temporal graphs. In fact, this is true for almost any network with a dynamic topology. Most modern communication networks, such as mobile ad-hoc, sensor, peer-to-peer, opportunistic, and delay-tolerant networks, are inherently dynamic. In social networks, the topology usually represents the social connections between a group of individuals and it changes as the social relationships between the individuals are updated, or as individuals leave or enter the group. In a transportation network, there is usually some fixed network of routes and a set of transportation units moving over these routes and dynamicity refers to the change of the positions of the transportation units in the network as time passes. Physical systems of interest may include several systems of interacting particles or molecules reacting in a well-mixed solution. Temporal relationships and temporal ordering of events are also present in the study of epidemics, where a group of individuals (or computing entities) come into contact with each other and we want to study the spread of an infectious disease (or a computer virus) in the population.

A very rich motivating domain is that of distributed computing systems that are inherently dynamic. The growing interest in such systems has been mainly driven by the advent of low-cost wireless communication devices and the development of efficient wireless communication protocols. Apart from the huge amount of work that has been devoted to applications, there is also a steadily growing concrete set of foundational work. A notable set of works has studied (distributed) computation in *worst-case* dynamic networks in which the topology may change arbitrarily from round to round subject to some constraints that allow for bounded end-to-end communication [OW05, KLO10, MCS14, MCS13, DPR$^+$13, APRU12]. Population protocols [AAD$^+$06] and variants [MCS11a, MS14a] are collections of finite-state agents that move passively, according to the dynamicity of the environment, and interact in pairs when they come close to each other. The goal is typically for the population to compute (i.e. agree on) something useful or construct a desired network or structure in such an adversarial setting. Another interesting direction assumes that the dynamicity of the network is a result of randomness (this is also the case sometimes in population protocols). Here the interest is on determining "good" properties of the dynamic network that hold with high probability (abbreviated w.h.p. and meaning with probability at least $1 - 1/n^c$ for some constant $c \geq 1$), such as small (temporal) diameter, and on designing protocols for distributed tasks [CMM$^+$08, AKL08]. In all the above subjects, there is always some sort of underlying temporal graph either assumed or implied. For introductory texts on the above lines of research in dynamic distributed networks the reader is referred to [CFQS12, MCS11b, Sch02, KO11].

Though static graphs [1] have been extensively studied, for their temporal generalization we are still far from having a concrete set of structural and algorithmic principles. Additionally, it is not yet clear how is the complexity of combinatorial optimization problems affected by introducing to them a notion of time. In an early but serious attempt to answer this question, Orlin [Orl81] observed that many dynamic languages derived from **NP**-complete languages can be shown to be **PSPACE**-complete. ① Among the other few things that we do know, is that the max-flow min-cut theorem holds with unit capacities for time-respecting paths [Ber96]. ② Additionally, Kempe *et al.* [KKK00] proved that, in temporal graphs, the classical formulation of Menger's theorem is violated and the computation of the number of node-disjoint *s*-*z* paths becomes **NP**-complete. ③ A reformulation of Menger's theorem which is valid for all temporal graphs was recently achieved in [MMCS13]. These results are discussed in Section 3. Recently, building on the distributed online dynamic network model of [KLO10], Dutta *et al.* [DPR$^+$13], among other things, presented *offline centralized algorithms* for the *k-token dissemination* problem. In k-token dissemination, there are k distinct pieces of information (tokens) that are initially present in some distributed processes and the problem is to disseminate all the k tokens to all the processes in the dynamic network, under the constraint that one token can go through an edge per round. These results, motivated by distributed computing systems, are presented in Section 4.

Another important problem is that of *designing* an efficient temporal graph given some requirements that the graph should meet. This problem was recently studied in [MMCS13], where the authors introduced several interesting cost minimization parameters for optimal temporal network design. One of the parameters is the *temporality* of a graph G, in which the goal is to create a temporal version of G minimizing the maximum number of labels of an edge, and the other is the *temporal cost* of G, in which the goal is to minimize the total number of labels used. Optimization of these parameters is performed subject to some *connectivity constraint.* They proved several upper and lower bounds for the temporality of some very basic graph families such as rings, directed acyclic graphs, and trees, as well as a trade-off between the temporality and the maximum label

minimize (optimize) labels (of time) on edges
$\rightarrow$ faster network?

[1] In this article, we use "static" to refer to classical graphs. This is plausible as the opposite of "dynamic" that is also commonly used for temporal graphs. In any case, the terminology is still very far from being standard.

of rings. Furthermore, they gave a *generic method* for computing a lower bound of the temporality of an arbitrary graph with respect to (abbreviated w.r.t.) the constraint of preserving a time-respecting analogue of every simple path of G. Finally, they proved that computing the temporal cost w.r.t. the constraint of preserving at least one time-respecting path from u to v whenever v is reachable from u in G, is **APX**-hard. Most of these results are discussed in Section 5. review

Other recent papers have focused on understanding the complexity and providing algorithms for temporal versions of classical graph problems. For example, the authors of [MS14b] considered temporal analogues of *traveling salesman problems* (TSP) in temporal graphs, and in the way also introduced and studied temporal versions of other fundamental problems like MAXIMUM MATCHING, PATH PACKING, MAX-TSP, and MINIMUM CYCLE COVER. One such version of TSP is the problem of exploring the nodes of a temporal graph as soon as possible. In contrast to the positive results known for the static case strong inapproximability results can be proved for the dynamic case [MS14b, EHK15]. Still, there is room for positive results for interesting special cases [EHK15]. Another such problem is the TEMPORAL TRAVELING SALESMAN PROBLEM WITH COSTS ONE AND TWO (abbreviated TTSP(1,2)), a temporal analogue of TSP(1,2), in which the temporal graph is a complete weighted graph with edge-costs from $\{1, 2\}$ and the cost of an edge may vary from instance to instance [MS14b]. The goal is to find a minimum cost temporal TSP tour. Several *polynomial-time approximation algorithms* have been proved for TTSP(1,2) [MS14b]. The best approximation is $(1.7 + \varepsilon)$ for the generic TTSP(1,2) and $(13/8 + \varepsilon)$ for its interesting special case in which the lifetime of the temporal graph is restricted to n. These and related results are presented in Section 6. review

Additionally, there are works that have considered *random temporal graphs*, in which the labels are chosen according to some probability distribution. We give a brief introduction to such models in Section 8. Moreover, Section 2 provides all necessary preliminaries and definitions and also a first discussion on temporal paths and Section 7 discusses a temporal graph model in which the availability times of the edges are provided by a set of linear functions. review

As is always the case, not all interesting results and material could fit in a single document. We list here some of them. Holme and Saramäki [HS12] give an extensive overview of the literature related to temporal networks from a diverge range of scientific domains. Harary and Gupta [HG97] discuss applications of temporal graphs and highlight the great

review

review importance of a systematic treatment of the subject. Kostakos [Kos09] uses temporal graphs to represent real datasets, shows how to derive various node metrics like average temporal proximity, average geodesic proximity and temporal availability, and also gives a static representation of a temporal graph (similar to the *static expansion* that we discuss in Section 2). Avin *et al.* [AKL08] studied the cover time of a simple random walk on Markovian dynamic graphs and proved that, in great contrast to being always polynomial in static graphs, it is exponential in some dynamic graphs. Clementi *et al.* [CMM$^+$08] studied the flooding time (also known as information dissemination; see a similar problem discussed in Section 4) in the following type of edge-markovian dynamic graphs: if an edge exists at time t then, at time $t+1$, it disappears with probability q, and if instead the edge does not exist at time t, then it appears at time $t+1$ with probability p. There are also several papers that have focused on temporal graphs in which every instance of the graph is drawn independently at random according to some distribution [CPMS07, HHL88, Pit87, KK02] (the last three did it in the context of dynamic gossip-based mechanisms), e.g. according to $\mathcal{G}(n, p)$. A model related to random temporal graphs, is the *random phone-call model*, in which each node, at each step, can communicate with a random neighbour [DGH$^+$87, KSSV00]. Other authors [XFJ03, FT98] have assumed that an edge may be available for a whole time-interval $[t_1, t_2]$ or several such intervals and not just for discrete moments. Aaron *et al.* [AKM14] studied the DYNAMIC MAP VISITATION problem, in which a team of agents must visit a collection of critical locations as quickly as possible in a dynamic environment. Kontogiannis *et al.* [KMP$^+$15], among other things, presented oracles for providing time-dependent min-cost route plans and conducted their experimental evaluation on a data set of the city of Berlin. review

set of nodes
node $v \in V$

2 Modeling and Basic Properties

$G = (V, E)$

set of edges
edge $e \in E$

When time is assumed to be discrete, a temporal graph (or digraph) is just a static graph (or digraph) $G = (V, E)$ with every edge $e \in E$ labeled with zero or more natural numbers. The labels of an edge may be viewed as the times at which the the edge is *available*. For example, an edge with no labels is never available while, on the other hand, an edge with labels all the even natural numbers is available every even time. Labels could correspond to seconds, days, years, or could even correspond to some artificial discrete measure of time under consideration.

Labels

discrete set of Labels (of an edge)

time availability

Multiple ways to model temporal graphs

underlying static model

labels λ

static graphs with labels (time)

There are several ways of modeling formally discrete temporal graphs. One is to consider an underlying static graph $G = (V, E)$ together with a labeling $\lambda : E \to 2^{\mathbb{N}}$ of G assigning to every edge of G a (possibly empty) set of natural numbers, called *labels*. Then the temporal graph of G with respect to λ is denoted by $\lambda(G)$. This notation is particularly useful when one wants to explicitly refer to and study properties of the labels of the temporal graph. For example, the multiset of all labels of $\lambda(G)$ can be denoted by $\lambda(E)$, their cardinality is defined as $|\lambda| = \sum_{e\in E} |\lambda(e)|$, and the maximum and minimum label assigned to the whole temporal graph as $\lambda_{\max} = \max\{l \in \lambda(E)\}$ and $\lambda_{\min} = \min\{l \in \lambda(E)\}$, respectively. Moreover, we define the *age* (or *lifetime*) of a temporal graph $\lambda(G)$ as $\alpha(\lambda) = \lambda_{\max} - \lambda_{\min} + 1$ (or simply α when clear from context). Note that in case $\lambda_{\min} = 1$ then we have $\alpha(\lambda) = \lambda_{\max}$.

time-edge model

Another, often convenient, notation of a temporal graph D is as an ordered pair of disjoint sets (V, A) such that $A \subseteq \binom{V}{2} \times \mathbb{N}$ in case of a graph and with $\binom{V}{2}$ replaced by $V^2 \setminus \{(u, u) : u \in V\}$ in case of a digraph. The set A is called the set of *time-edges*. A can also be used to refer to the structure of the temporal graph at a particular time. In particular, $A(t) = \{e : (e, t) \in A\}$ is the (possibly empty) set of all edges that appear in the temporal graph at time t. In turn, $A(t)$ can be used to define a snapshot of the temporal graph D at time t, which is usually called the *t-th instance of* D, and is the static graph $D(t) = (V, A(t))$. So, it becomes evident that a temporal graph may also be viewed as a *sequence of static graphs* $(G_1, G_2, \ldots, G_{\lambda_{\max}})$.

static graph equivalent of temporal graph

Finally, it is typically very useful to expand in time the whole temporal graph and obtain an equivalent static graph without losing any information. The reason for doing this is mainly because static graphs are much better understood and there is a rich set of well established tools and techniques for them. So, a common approach to solve a problem concerning temporal graphs is to first express the given temporal graph as a static graph and then try to apply or adjust one of the existing tools that works on static graphs. Formally, the *static expansion* of a temporal graph $D = (V, A)$ is a DAG $H = (S, E)$ defined as follows. If $V = \{u_1, u_2, \ldots, u_n\}$ then $S = \{u_{ij} : \lambda_{\min} - 1 \le i \le \lambda_{\max}, 1 \le j \le n\}$ and $E = \{(u_{(i-1)j}, u_{ij'}) : \lambda_{\min} \le i \le \lambda_{\max}$ and $j = j'$ or $(u_j, u_{j'}) \in A(i)\}$. In words, for every discrete moment we create a copy of V representing the instance of the nodes at that time (called *time-nodes*). We may imagine the moments as levels or rows from top to bottom, every level containing a copy of V. Then we add outgoing edges from time-nodes of one level only to time-nodes of the level below it. In particular, we connect a

Connect static graphs between levels through time-connected nodes only

time-node $u_{(i-1)j}$ to its own subsequent copy u_{ij} and to every time-node $u_{ij'}$ s.t. $(u_j, u_{j'})$ is an edge of the temporal graph at time i. Observe that the above construction includes all possible vertical edges from a node to its own subsequent instance. These edges express the fact that nodes are usually not oblivious and can preserve their on history in time (modeled like propagating information to themselves). Nevertheless, depending on the application, these edges may some times be omitted.

2.1 Journeys

As is the case in static graphs, the notion of a *path* is one of the most central notions of a temporal graph, however it has to be redefined to take time into account. A *temporal* (or *time-respecting*) *walk* W of a temporal graph $D = (V, A)$ is an alternating sequence of nodes and times $(u_1, t_1, u_2, t_2, \ldots, u_{k-1}, t_{k-1}, u_k)$ where $(u_i u_{i+1}, t_i) \in A$, for all $1 \le i \le k-1$, and $t_i < t_{i+1}$, for all $1 \le i \le k-2$. We call $t_{k-1} - t_1 + 1$ the *duration* (or *temporal length*) of the walk W, t_1 its *departure time* and t_{k-1} its *arrival time*. A *journey* (or *temporal/time-respecting path*) J is a temporal walk with pairwise distinct nodes. In words, a journey of D is a path of the underlying static graph of D that uses strictly increasing edge-labels. A u-v journey J is called *foremost from time* $t \in \mathbb{N}$ if it departs after time t and its arrival time is minimized. The *temporal distance* from a node u at time t to a node v is defined as the duration of a foremost u-v journey from time t. We say that a temporal graph $D = (V, A)$ has *temporal (or dynamic) diameter d*, if d is the minimum integer for which it holds that the temporal distance from every time-node $(u, t) \in V \times \{0, 1, \ldots, \alpha - d\}$ to every node $v \in V$ is at most d.

foremost journey Minimized duration of journey

temporal distance

A nice property of foremost journeys is that they can be computed efficiently. In particular there is an algorithm that, given a source node $s \in V$ and a time t_{start}, computes for all $w \in V \setminus \{s\}$ a foremost s-w journey from time t_{start} [MMCS13, MMS15]. The running time of the algorithm is $O(n\alpha^3(\lambda) + |\lambda|)$, where n here and throughout this article denotes the number of nodes of the temporal graph. It is worth mentioning that this algorithm takes as input the whole temporal graph D. Such algorithms are known as *offline* algorithms in contrast to *online* algorithms to which the temporal graph is revealed on the fly. The algorithm is essentially a temporal translation of the breadth-first search (BFS) algorithm (see e.g. [CLRS01] page 531) with path length replaced by path arrival time. For every time t, the algorithm picks one after the other all nodes that have been already reached (initially only the source node s) and inspects all edges that are incident to that node at time t. If a time-edge (e, t) leads to

Picks one node

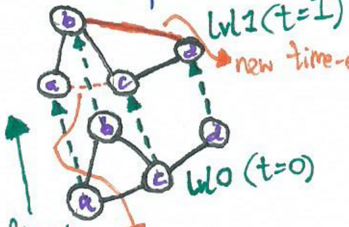
offline algorithm → whole temporal graph at once

online algorithm → revealed on the fly

review BFS algorithm, and MMCS13/15

a node w that has not yet been reached, then (e, t) is picked as an edge of a foremost journey from the source to w. This greedy algorithm is correct for the same reason that the BFS algorithm is correct. An immediate way to see this is by considering the static expansion of the temporal graph. The algorithm begins from the upper copy (i.e. at level 0) of the source in the static expansion and essentially executes the following slight variation of BFS: at step $i+1$, given the set R of already reached nodes at level i, the algorithm first follows all vertical edges leaving R in order to reach in one step the $(i+1)$-th copy of each node in R, and then inspects all diagonal edges leaving R to discover new reachabilities. The algorithm outputs as a foremost journey to a node u, the directed path of time-edges by which it first reached the column of u (vertical edges are interpreted as waiting on the corresponding node). The above algorithm computes a shortest path to each column of the static expansion. Correctness follows from the fact that shortest paths to columns are equivalent to foremost journeys to the nodes corresponding to the columns.

static
layered copies of temporal graph



3 Connectivity and Menger's Theorem

Assume that we are given a static graph G and a source node s and a sink node z of G. [2] Two paths from s to z are called node-disjoint if they have only the nodes s and z in common. *Menger's theorem* [Men27], which is the analogue of the max-flow min-cut theorem for undirected graphs, is one of the most basic theorems in the theory of graph connectivity. It states that *the maximum number of node-disjoint s-z paths is equal to the minimum number of nodes that must be removed in order to separate s from z* (see also [Bol98] page 75).

It was first observed in [Ber96] and then further studied in [KKK00] that this fundamental theorem of static graphs, is violated in temporal graphs if we keep its original formulation and only require it to hold for journeys instead of paths. In fact, the violation holds even for a very special case of temporal graphs, those in which every edge has at most one label, which are known as *single-labeled temporal graphs* (as opposed to the more general *multi-labeled* temporal graphs that we have discussed so far). Even in such temporal graphs, the maximum number of node-disjoint journeys from s to z can be strictly less than the minimum number of nodes whose deletion leaves no s-z journey. For a simple example, observe in Figure 1 that there are no two node-disjoint journeys from s to z but

[2] The sink is usually denoted by t in the literature. We use z instead as we reserve t to refer to time moments.

watched Mengers theorem video.
only s & v in common, but can have different nodes on the way.

$P_1 = s\,\underline{axyn}\,v$
$P_2 = s\,\underline{bcdg}\,v$
internally / node disjoint.

In static graphs, # node disjoint paths (MAXIMUM) = # nodes that must be removed to separate s & v (MINIMUM)

NOT FOR JOURNEYS ON TEMPORAL GRAPHS

after deleting any one node (other than s or z) there still remains a s-z journey. To see this, notice that every journey has to visit at least two of the inner-nodes u_2, u_3, u_4. If u_2 is one of them, then a vertical obstacle is introduced which cannot be avoided by any other journey. If u_2 is not, then the only disjoint path remaining is (s, u_2, z) which is not a journey. On the other hand, any set of two inner vertices has a s-z journey going through them implying that any s-z separator must have size at least 2. As shown in [KKK00], this construction can be generalized to a single-labeled graph with $2k-1$ inner nodes in which: (i) every s-z journey visits at least k of these nodes, ensuring again that there are no two node-disjoint s-z journeys and (ii) there is a journey through any set of k inner nodes, ensuring that every s-z separator must have size at least k.

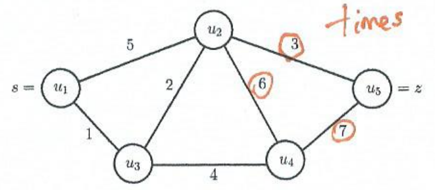


Fig. 1. A counterexample of Menger's theorem for temporal graphs (adopted from [KKK00]). Each edge has a single time-label indicating its availability time.

On the positive side, the violation does not hold if we replace node-disjointness by edge-disjointness and node removals by edge removals. In particular, it was proved in [Ber96] that for single-labeled temporal graphs, the maximum number of edge-disjoint journeys from s to z is equal to the minimum number of edges whose deletion leaves no s-z journey, that is, that the max-flow min-cut theorem of static graphs holds with unit capacities for journeys in single-labeled temporal graphs. The construction (which we adopt from [KKK00]) is simply an *ad-hoc* static expansion for the special case of single-labeled temporal graphs. Let $G = (V, E)$ be the underlying graph of an undirected single-labeled temporal graph. We construct a labeled directed graph $G' = (V', E')$ as follows. for every $\{u, v\} \in E$ we add in G' two new nodes x and y and the directed edges (u, x), (v, x), (y, u), (y, v), (x, y). Then we relax all labels required so that there is sufficient "room" (w.r.t. time) to introduce (by labeling the new edges) both a (u, x, y, v) journey and a (v, x, y, u) journey. The goal is to be able to both move by a journey from u to v and

Single labels, single active times on each edge.

from v to u in G'. An easy way to do this is the following: if t is the label of $\{u, v\}$, then we can label $(u, x), (x, y), (y, v)$ by $(t.1, t.2, t.3)$, where $t.1 < t.2 < t.3$, and similarly for $(v, x), (x, y), (y, u)$. Then we construct a static directed graph $G'' = (V'', E'')$ as follows: For every $u \in V$ let $y_1, y_2, \ldots, y_i, \ldots$ be its incoming edges and $x_1, x_2, \ldots, x_j, \ldots$ its outgoing edges. We want to preserve only the time-respecting y, u, x traversals. To this end, for each one of the (y_i, u) edges we introduce a node w_i and the edge (y_i, w_i) and for each one of the (u, x_i) edges we introduce a node v_j and the edge (v_j, x_j) and we delete node u. Finally, we introduce the edge (w_i, v_j) iff $(y_i, u), (u, x_j)$ is time-respecting. This reduction preserves edge-disjointness and sizes of edge separators and if we add a super-source and a super-sink to G'' the max-flow min-cut theorem for static directed graphs yields the aforementioned result. Another interesting thing is that reachability in G under journeys corresponds to (path) reachability in G'' so that we can use BFS on G'' to answer questions about foremost journeys in G, as we did with the static expansion in Section 2.1.

Fortunately, the above important negative result concerning Menger's theorem has a turnaround. In particular, it was proved in [MMCS13] that if one reformulates Menger's theorem in a way that takes time into account then a very natural temporal analogue of Menger's theorem is obtained, which is valid for all (multi-labeled) temporal networks. The idea is to replace in the original formulation node-disjointness by *node departure time disjointness* (or *out-disjointness*) and node removals by *node departure times removals*. When we say that we remove *node departure time* (u, t) we mean that we remove *all edges leaving u at time t*, i.e. we remove label t from all (u, v) edges (for all $v \in V$). So, when we ask "how many node departure times are needed to separate two nodes s and z?" we mean how many node departure times must be selected so that after the removal of all the corresponding time-edges the resulting temporal graph has no s-z journey (note that this is a different question from how many time-edges must be removed and, in fact, the latter question does not result in a Menger's analogue). Two journeys are called *out-disjoint* if they never leave from the same node at the same time (see Figure 2 for an example).

} time into account
Menger's theorem
analogeous.

Theorem 1 (Menger's Temporal Analogue [MMCS13]). *Take any temporal graph $\lambda(G)$, where $G = (V, E)$, with two distinguished nodes s and z. The maximum number of out-disjoint journeys from s to z is equal to the minimum number of node departure times needed to separate s from z.*

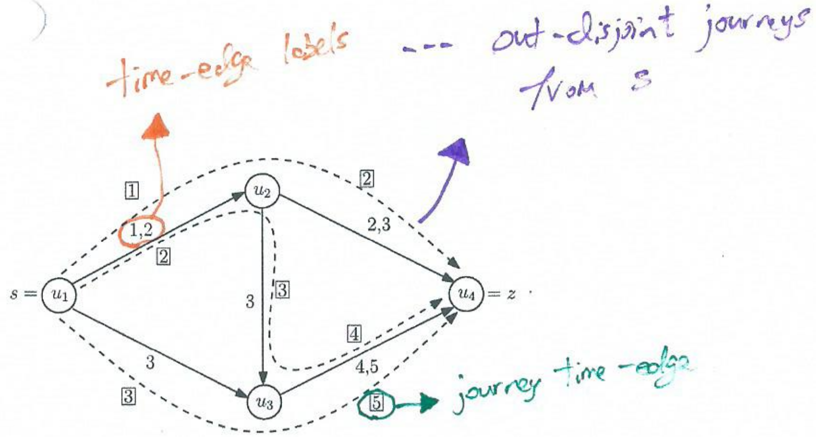


Fig. 2. An example of a temporal graph. The dashed curves highlight the directions of three out-disjoint journeys from s to z. The labels used by each of these journeys are indicated by the labels that are enclosed in boxes.

The idea is to take the static expansion $H = (S, A)$ of $\lambda(G)$ and, for each time-node u_{ij} with at least two outgoing edges to nodes different than u_{i+1j}, add a new node w_{ij} and the edges (u_{ij}, w_{ij}) and $(w_{ij}, u_{(i+1)j_1})$, $(w_{ij}, u_{(i+1)j_2}), \ldots, (w_{ij}, u_{(i+1)j_k})$. Then define an *edge capacity function* $c : A \to \{1, \lambda_{\max}\}$ as follows: edges $(u_{ij}, u_{(i+1)j})$ take capacity $\lambda_{\max}$ and all other edges take capacity 1. The theorem follows by observing that the maximum u_{01}-$u_{\lambda_{\max}n}$ flow is equal to the minimum of the capacity of a u_{01}-$u_{\lambda_{\max}n}$ cut, the maximum number of out-disjoint journeys from s to z is equal to the maximum u_{01}-$u_{\lambda_{\max}n}$ flow, and the minimum number of node departure times needed to separate s from z is equal to the minimum of the capacity of a u_{01}-$u_{\lambda_{\max}n}$ cut. See also Figure 3 for an illustration.

4 Dissemination and Gathering of Information

A natural application domain of temporal graphs is that of *gossiping* and in general of *information dissemination*, mainly by a distributed set of entities (e.g. a group of people or a set of distributed processes). Two early such examples were the *telephone problem* [BS72] and the *minimum broadcast time problem* [Rav94]. In both, the goal is to transmit some information to every participant of the system, while minimizing some measure of communication or time. A more modern setting, but in the same spirit, comes from the very young area of distributed computing in highly dynamic networks [OW05, KLO10, KO11, CFQS12, MCS14, MCS13].