An Introduction to Temporal Graphs: An Algorithmic Perspective*

Othon Michail

Computer Technology Institute and Press "Diophantus" (CTI), Patras, Greece
Department of Computer Science, University of Liverpool, Liverpool, UK
Email: michailo@cti.gr

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 is 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.

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. Since then, graph 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].

Temporal graphs (also known as *dynamic*, *evolving* [Fer04], or *time-varying* [FMS13, 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. a temporal graph in which 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 papers have 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, Mic15b] 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 sometimes the case 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 of 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. Even though his model is not equivalent to the modern temporal graph models that concern us in this article, still it gives a first indication of the complexity increase that accompanies temporal versions of combinatorial optimization problems, therefore (and also for historical reasons) we have included a brief presentation of it in Section 8. 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]. Informally, a *time-respecting path* (or *temporal path*, but usually called *journey* in the recent literature and also in this article) is like a usual path whose edges, additionally, use *strictly increasing* times (or just *non-decreasing* in some cases); see Section 2 for formal definitions. Additionally, Kempe *et al.* [KKK00] proved that, in temporal graphs, the classical formulation of Menger's theorem [2] is violated and the computation of the number of node-disjoint s-z paths becomes **NP**-complete. 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.

Also 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 (called *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 contribution of the Distributed Computing community to the theory of temporal graphs concerns the investigation of how instantaneous properties (or properties satisfied for some larger but restricted time-windows) relate to global properties of the temporal graph. Such properties may be continuously-connected instances, guaranteed connectivity or guaranteed propagation of information in a given time-window length (holding for all times), small instantaneous diameter, etc. The goal is to determine whether these properties imply some "good" temporal-connectivity properties of the temporal graph as a whole, for example a small temporal diameter. These investigations, together with the novel notions and metrics that have been proposed for capturing such properties formally, are presented in Section 5.

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 has to be performed subject to some *connectivity constraint*. They proved several

---

[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.

[2] *Menger's theorem* [Men27], which is the analogue of the max-flow min-cut theorem for undirected graphs, states that the maximum number of node-disjoint s-z paths is equal to the minimum number of nodes needed to separate s from z (see also [Bol98] page 75).

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 of rings. Furthermore, they gave a *generic method* for computing a lower bound of the temporality of an arbitrary graph G 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 6.

Other recent papers have focused on understanding the complexity of temporal versions of classical graph problems and providing algorithms for them. For example, the authors of [MS14b] considered 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 MATCH-ING, 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. A first set of *polynomial-time approximation algorithms* for TTSP(1,2) (the only ones known at the time of writing this article) was given in [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 [3] of the temporal graph is restricted to n. These and related results are presented in Section 7.

A family of temporal graphs which is of particular interest as its graphs underlie many real-world systems, is the family of *recurrent* and *periodic* temporal graphs. A characteristic example of a real system with a topology that changes with time but in a periodic (and therefore also predictable) way, is the public transportation system. Apart from their practical importance, such temporal graphs usually have the convenient property of being susceptible to a succinct representation of their evolution (for example, there could be a set of functions, e.g. linear functions, describing/generating the availability times of the edges). Recurrent and periodic temporal graphs are discussed in Section 8. Additionally, there are works that have considered *random temporal graphs*, another succinctly representable model, in which the labels are chosen according to some probability distribution. An obvious motivation for this, is that the dynamicity of a great variety of real temporal networks is the result of some underlying random process. But apart from this, randomness can usually simplify the understanding of the behavior of several interesting network properties (e.g. many of the average properties of the Erdös-Renyi random graph model can be calculated exactly in the limit) and can assist the efficient design of a temporal graph with a desired property (by sacrificing correctness to hold w.h.p.), e.g. to have, for all u, v, a time-respecting path from u to v. Therefore, we expect randomness to be a valuable tool in the study of temporal graphs, as it has been for static graphs. We give a brief introduction to such models in Section 9. Moreover, Section 2 provides all necessary preliminaries and definitions and also a first discussion on temporal paths and Section 10 concludes the article and gives a selection of open problems and research directions.

---

[3] Essentially, the *lifetime* of a temporal graph is equal to the number of steps for which the temporal graph exists. Restricting it, shortens the state space and can make some problems easier to solve or approximate; for example, this seems to be the case for TTSP(1,2).

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. Very recently, Holme [Hol15] prepared a more up-to-date survey (compared to [HS12]) focusing on the developments (mainly methods, systems modeled, and questions addressed) during the period 2012-2015. Harary and Gupta [HG97] discuss applications of temporal graphs and highlight the great 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). Clementi *et al.* [CMM$^+$08] studied the *flooding time* (*flooding* is also known as *information dissemination*; see a similar problem discussed in Section 4) in the following type of edge-markovian temporal 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, or that it has time-dependent travel-times [KZ14]. Finally, Aaron *et al.* [AKM14] studied the DYNAMIC MAP VISITATION problem, in which a team of agents, operating in a dynamic environment, must visit a collection of critical locations as quickly as possible.

2 Modeling and Basic Properties

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.

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 $\alpha(\lambda) = \lambda_{\max}$.

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* $(G_1, G_2, \ldots, G_{\lambda_{\max}})$.

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 time-node $u_{(i-1)j}$ to its own subsequent copy u_{ij} and to every time-node $u_{ij'}$ such that $(u_j, u_{j'})$ 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.

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 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.

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. [4] 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 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.

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

---

[4] The sink is usually denoted by t in the literature. We use z instead as we reserve t to refer to time moments.
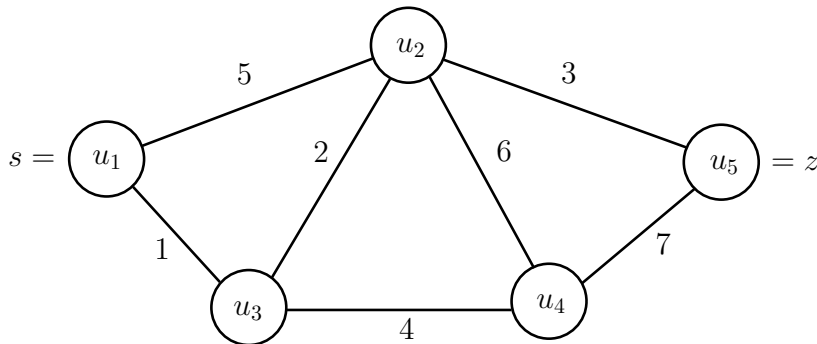
Fig. 1. A counterexample of Menger's theorem for temporal graphs (adopted from [KKK00]). Each edge has a single time-label indicating its availability time.

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 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).
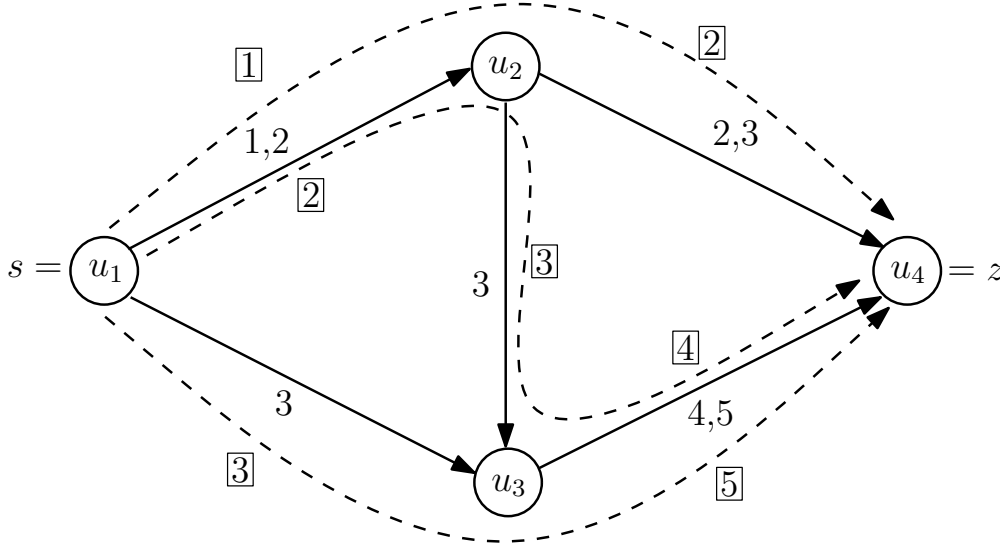

Fig. 2. An example of a temporal graph. The dashed curves highlight the directions of three out-disjoint journeys from s to z. The labels used by each of these journeys are indicated by the labels that are enclosed in boxes.

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.*

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
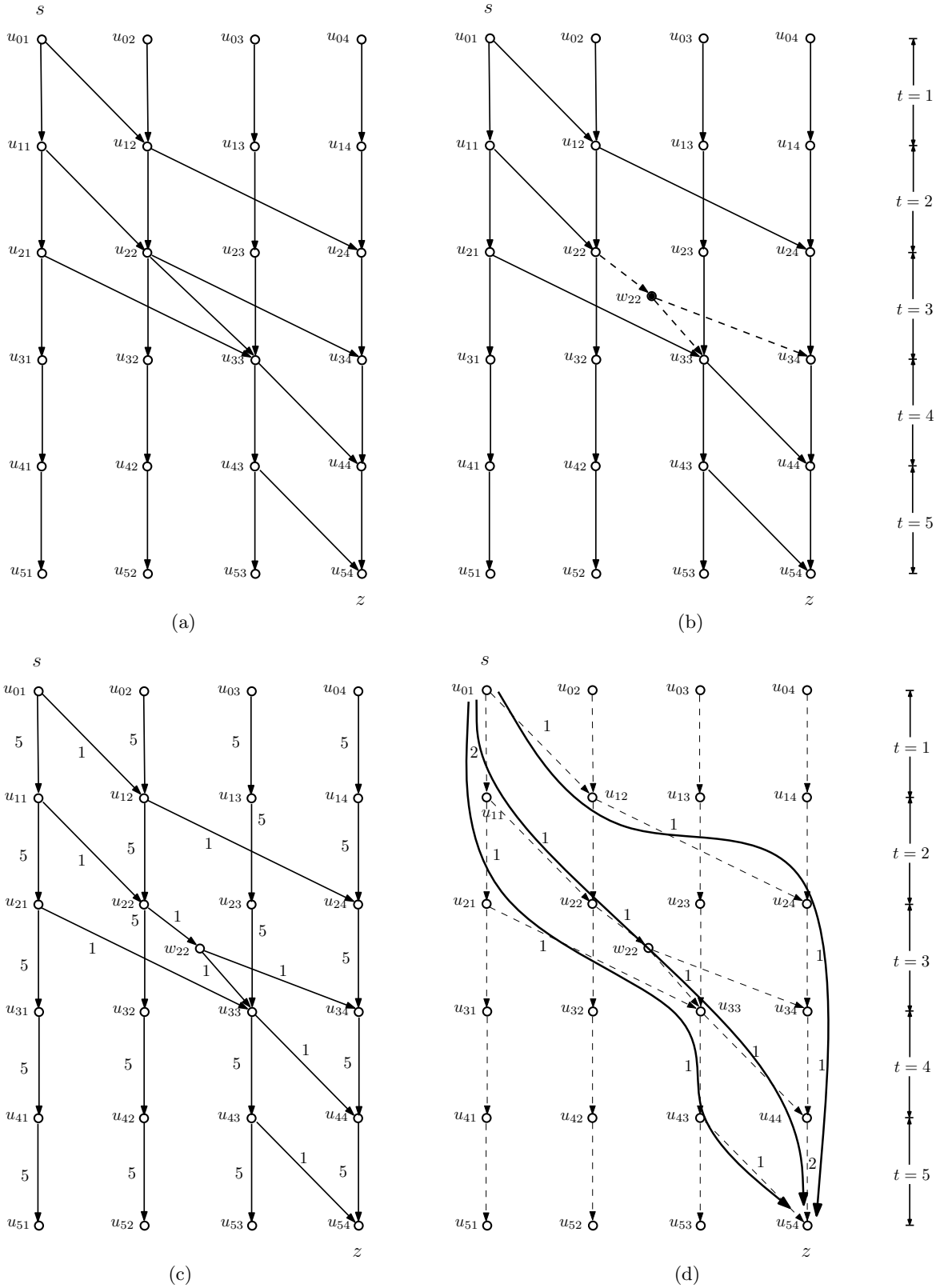
Fig. 3. (a) The static expansion of a temporal graph. Here, only two edges leave from the same node at the same time: (u_{22}, u_{33}) and (u_{22}, u_{34}). (b) Adding a new node w_{22} and three new edges. This we ensures that a node departure time can be removed by removing a single diagonal edge. removing edge (u_{22}, w_{22}) removes all possible departures from u_{22}. This ensures that separation of s and z by node-departure times is equivalent to separation by a usual static cut. (c) Adding capacities to the edges. Vertical edges take capacity $\lambda_{max} = 5$ and diagonal edges take capacity 1. (d) The maximum number of out-disjoint journeys from s to z is equal to the maximum flow from s to z and both are equal to 3.

in the same spirit, comes from the very young area of distributed computing in highly dynamic networks [OW05, KLO10, KO11, CFQS12, MCS14, MCS13].

There are n nodes. In this context, nodes represent distributed processes. Note, however, that most of the results that we will discuss, concern centralized algorithms (and in case of lower bounds, these immediately hold for distributed algorithms as well). The nodes communicate with other nodes in discrete rounds by interchanging messages. In every round, an adversary scheduler selects a set of edges between the nodes and every node may communicate with its current neighbors, as selected by the adversary, usually by *broadcasting* a single message to be delivered to all its neighbors. So, the dynamic topology behaves as a discrete temporal graph where the i-th instance of the graph is the topology selected by the adversary in round i. The main difference, compared to the setting of the previous sections, is that now (in all results that we will discuss in this section, apart from the last one) the topology is revealed to the algorithms in an online and totally unpredictable way. An interesting special case of temporal graphs consists of those temporal graphs that have *connected instances*. A temporal graph D is called *continuously connected* (also known as *1-interval connected*) if $D(t)$ is connected for all times $t \geq 1$ [OW05, KLO10]. Such temporal graphs have some very useful properties concerning information propagation in a distributed setting, like, for example, that if all nodes broadcast in every round all information that they have heard so far, then in every round at least one more node learns something new, which implies that a piece of information can in principle be disseminated in at most $n - 1$ rounds. Naturally, the problem of information dissemination becomes much more interesting and challenging if we do not allow nodes to transmit an unlimited amount of information in every round, that is, if we restrict the size of the messages that they can transmit.

An interesting problem of token dissemination in such a setting, called the *k-token dissemination problem*, was introduced and first studied in [KLO10]. In this problem there is a domain of *tokens* $\mathcal{T}$, each node is assigned a subset of the tokens, and a total of k distinct tokens is assigned to the nodes. The goal is for an algorithm (centralized or distributed) to organize the communication between the nodes in such a way that, under any dynamic topology (from those described above), each node eventually terminates and outputs (i.e. has learned) all k tokens. In particular, the focus here is on *token-forwarding algorithms*. Such an algorithm is quite restricted in that, in every round r and for every node u, it only picks a single token from those already known by u (or the empty token $\bot$) and this token will be delivered to all the current neighbors of u by a single broadcast transmission. Token-forwarding algorithms are simple, easy to implement, typically incur low overhead, and have been extensively studied in static networks [Lei92, Pel00]. We will present now a lower bound from [KLO10] on the number of rounds for token dissemination, that holds even for centralized token forwarding algorithms. Such centralized algorithms are allowed to see and remember the whole state and history of the entire network, but they have to make their selection of tokens to be forwarded without knowing what topology will be scheduled by the adversary in the current round. So, first the algorithm selects and then the adversary reveals the topology, taking into account the algorithm's selection. For simplicity, it may be assumed that each of the k tokens is assigned initially to exactly one (distinct) node.

Theorem 2 ([KLO10]). *Any deterministic centralized algorithm for k-token dissemination in continuously connected temporal graphs requires at least $\Omega(n \log k)$ rounds to complete in the worst case.*

The idea behind the proof is to define a potential function that charges by $1/(k - i)$ the i-th token learned by each node. So, for example, the first token learned by a node comes at a cheap

price of $1/k$ while the last token learned costs 1. The initial total potential is 1, because k nodes have obtained their first token each, and the final potential (i.e. when all nodes have learned all k tokens) is $n \cdot H_k = \Theta(n \log k)$. Then it suffices to present an adversarial schedule, i.e. a continuously connected temporal graph, that forces any algorithm to achieve in every round at most a bounded increase in potential. The topology of a round can be summarized as follows. First we select all edges that contribute no cost, called *free edges*. An edge $\{u, v\}$ is free if the token transmitted by u is already known by v and vice versa. The free edges partition the nodes into l components $C_1, C_2, \ldots, C_l$. We pick a representative v_i from each component C_i. It remains to construct a connected graph over the v_is. An observation is that each v_i transmits a distinct token t_i, otherwise at least two of them should have been connected by a free edge (because two nodes interchanging the same token cannot learn anything new). The idea is to further partition the representatives into a small set of nodes that know many tokens each and a large set of nodes that know few tokens each. We can call the nodes that know many tokens the *expensive* ones, because according to the above potential function a new token at a node that already knows a lot of tokens comes at a high price, and similarly we call those nodes that know few tokens the *cheap* ones. In particular, a node is expensive if it is missing at most $l/6$ tokens and cheap otherwise. Roughly, a cheap node learns a new token at the low cost of at most $6/l$, because the cost of a token is inversely proportional to the number of missing tokens before the token's arrival. First we connect the cheap nodes by an arbitrary line. As there are at most l such nodes and each one of them obtains at most two new tokens (because it has at most two neighbors on the line and each node transmits a single token), the total cost of this component is at most 12, that is, bounded as desired. It remains to connect the expensive nodes. It can be shown that there is a way to match each expensive node to a distinct cheap node (i.e. by constructing a matching between the expensive and the cheap nodes), so that no expensive node learns a new token. So, the only additional cost is that of the new tokens that cheap nodes obtain from expensive nodes. This additional cost is roughly at most 6, so the total cost have been shown to be bounded by a small constant as required. It is worth mentioning that [KLO10], apart from the above lower bound, also proposed a simple distributed algorithm for k-token dissemination that needs $O(nk)$ rounds in the worst case to deliver all tokens.

The above lower bound can be further improved by exploiting the probabilistic method [DPR$^+$13]. In particular it can be shown that any randomized token-forwarding algorithm (centralized or distributed) for k-token dissemination needs $\Omega(nk/\log n)$ rounds. This lower bound is within a logarithmic factor of the $O(nk)$ upper bound of [KLO10]. As is quite commonly the case in probabilistic results, the interesting machinery used to establish the lower bound is in the analysis and not in the construction itself. Now *all* the representatives of the connected components formed by the free edges are connected arbitrarily by a line. The idea is to first prove the bound w.h.p. over an initial token distribution, in which each of the nodes receives each of the k tokens independently with probability $3/4$. It can be shown in this case that, w.h.p. over the initial assignment of tokens, in every round there are at most $O(\log n)$ new token deliveries and an overall of $\Omega(nk)$ new token deliveries must occur for the protocol to complete. Finally, it can be shown via the probabilistic method, that, in fact, any initial token distribution can be reduced to the above distribution for which the bound holds. The above lower bounding technique, based on the probabilistic method, was applied in [HK12] to several variations of k-token dissemination. For example, if the nodes are allowed to transmit $b \leq k$ tokens instead of only one token in every round, then it can be proved that any randomized token-forwarding algorithm requires $\Omega(n + nk/(b^2 \log n \log \log n))$ rounds.

In [DPR$^+$13], also offline token forwarding algorithms were designed, that is, algorithms provided with the whole dynamic topology in advance. One of the problems that they studied, was that of delivering all tokens to a given sink node z as fast as possible, called the *gathering problem*. We now present a lemma from [DPR$^+$13] concerning this problem, mainly because its proof constitutes a nice application of the temporal analogue of Menger's theorem presented in Section 3 (the simplified proof via Menger's temporal analogue is from [MMCS13]).

Lemma 1 ([DPR$^+$13]). *Let there be $k \leq n$ tokens at given source nodes and let z be an arbitrary node. Then, if the temporal graph D is continuously connected, all the tokens can be delivered to z, using local broadcasts, in $O(n)$ rounds.*

Let $S = \{s_1, s_2, \ldots, s_h\}$ be the set of source nodes, let $N(s_i)$ be the number of tokens of source node s_i and let the age of the temporal graph be $n + k = O(n)$. It suffices to prove that there are at least k out-disjoint journeys from S to any given z, such that $N(s_i)$ of these journeys leave from each source node s_i. Then, all tokens can be forwarded in parallel, each on one of these journeys, without conflicting with each other in an outgoing transmission and, as the age is $O(n)$, they all arrive at z in $O(n)$ rounds. To show the existence of k out-disjoint journeys, we create a *supersource* node s and connect it to the source node with token i (assuming an arbitrary ordering of the tokens from 1 to k) by an edge labeled i. Then we shift the rest of the temporal graph in time, by increasing all other edge labels by k. The new temporal graph D' has asymptotically the same age as the original and all properties have been preserved. Now, it suffices to show that there are at least k out-disjoint journeys from s to z, because the k edges of s respect the $N(s_i)$'s. Due to Menger's temporal analogue, it is equivalent to show that at least k departure times must be removed to separate s from z. Indeed, any removal of fewer than k departure times must leave at least n rounds during which all departure times are available (because, due to shifting by k, the age of D' is $n + 2k$). Due to the fact that the original temporal graph is connected in every round, n rounds guarantee the existence of a journey from s to z.

5   Instantaneous vs Global Properties

It has probably already become evident that a crucial property of temporal graphs is *connectivity*. The connectivity of a static graph ensures that any pair of nodes in the graph can communicate and influence each other, even if this can only happen implicitly via one or more intermediate nodes. Its analogue in a temporal graph is the notion of *temporal connectivity*. An important characteristic of static connectivity is that it holds once and for all, in the sense that a path between two nodes will be either present or missing forever. Unfortunately, this is not the case in temporal graphs, where a journey from one node to another may not be available in future steps. A possible definition of temporal connectivity could be the following: *a temporal graph $D = (V, A)$ is temporally connected if for every ordered pair of nodes (u, v) there is a journey from u to v that arrives before the lifetime of D expires.* Though this definition allows every node to reach any other node it is still useless for many practical purposes. The reason is that, in contrast to static connectivity, it does not provide us with some continuously available reachabilities between the nodes. A natural way to avoid this is to assume an unlimited lifetime and require the temporal reachabilities to hold repetitively. A very abstract way to formalize this, is via the notion of the *temporal diameter*, already defined in Section 2. Formally, if a temporal graph $D = (V, A)$ has temporal diameter d, then for every time-node (u, t) and every node v there is a journey from (u, t) to v of duration at most d. This

type of temporal connectivity is indeed persistent because given any source node u, any sink node v, and any time t we know that we can start a walk from u at time $t + 1$ that will arrive at v at most by time $t + d$.

In several cases, the temporal graph under consideration may satisfy some stronger guarantees. One such example is to a temporal graph that comes with the guarantee that every one of its instances is connected (already discussed in the previous section). An immediate question to ask is whether and how such instantaneous properties (or properties that hold for a particular time-window), which can be viewed as local properties in the static expansion of the temporal graph, translate into more global properties, like bounds on the temporal diameter.

Let us begin from the continuously connected case. The fact that every instance of the temporal graph is connected implies that its temporal diameter d is at most $n - 1$. There is an intuitive constructive way to see this [KLO10]. Take any node u and any time t. At time $t + 1$ there must be an edge in the $(\{u\}, V \setminus \{u\})$-cut because otherwise u would be disconnected from the rest of the graph at that time. So there is a journey from u to some other node arriving by $t + 1$. Denote by S the set of nodes reached by u via a journey so far, u inclusive. [5] At any time t' there must be an edge (w, z) in the $(S, V \setminus S)$ cut. Observe now that $w \in S$ implies a journey (J, t'', w) from u to w arriving by time $t'' < t'$, therefore (J, t'', w, t', z) is a journey from u to z arriving by time t', which implies that z is added to S and, in turn, that $|S|$ increases by at least one in every step. As $S = \{u\}$ at step 0, in $n - 1$ steps $|S|$ must be equal to n and we conclude that $d \leq n - 1$. Of course this could happen much faster in some cases. For example, if it happens that every instance is a clique, then $|S_u|$ becomes n in a single step, for every $u \in V$, and the temporal diameter is 1.

It could even hold that every instance G_i of the temporal graph is f-connected, meaning that the removal of any $f - 1$ nodes from G_i does not disconnect it. Does this stronger instantaneous property imply something better about the dynamic diameter of the temporal graph? The answer is *yes* [KLO10]. The dynamic diameter now becomes at most $\lceil (n-1)/f \rceil$ (i.e. $O(n/f)$). Take again a node u and any time t and initialize again the set $S = \{u\}$ as before. At time $t + 1$ there must be at least f edges (u, v_i) in the $(\{u\}, V \setminus \{u\})$-cut, otherwise the removal of the at most $f - 1$ v_is would disconnect u from $V \setminus \{u\}$. So, S increases by at least f after the first step. Take now any time t'. At time $t' + 1$ there must be edges (u_i, v_i) in the $(S, V \setminus S)$-cut to at least f distinct v_is in $V \setminus S$, otherwise the removal of the at most $f - 1$ v_is would disconnect S from $V \setminus S$. So, also in the generic case, S increases by at least f, therefore in at most $\lceil (n-1)/f \rceil$ steps $|S|$ is equal to n which implies that $d \leq \lceil (n-1)/f \rceil$.

On the other hand, not all instantaneous guarantees can be used to infer useful global properties. For example, a good instantaneous diameter does not necessarily imply a good temporal diameter. In a temporal star graph D in which all leaf-nodes $(u_1, u_2, \ldots, u_{n-1})$ but one (u_n) go to the center one after the other in a modular way, any message from the node that enters last the center to the node that never enters the center needs $n - 1$ steps to be delivered; that is the temporal diameter of D is $n - 1$ even though the diameter of every instance of D is just 2. This star graph, with additional self-loops at all nodes, was used in [AKL08] to show that, in contrast to the cover time of a random walk on a static graph which is always $O(d \cdot |E| \cdot \log n)$ (i.e. polynomial in n), the cover time of a random walk on a temporal graph may be $\Omega(2^n)$ (i.e. exponential in n). The reason is that the only way for a random walk (that begins from u_{n-1}) that is on a leaf u_i to reach u_n is to

---

[5] Such a set S is typically called the *future set* of a time-node in the relevant literature (see e.g. [MCS14]). A similar set that shows up often in the design and analysis of distributed protocols, is the *past set* of a time-node u, containing all nodes that have reached u by a journey so far (see e.g. [KOM11]).

stay at u_i for $n-2$ consecutive steps until u_i goes to the center and then move to u_n. Because if the walk moves sooner to the center, then in the next step that center will become a leaf and the process will start over. The probability of staying for $n-2$ consecutive steps at a leaf is $1/2^{n-2}$ which gives the $\Omega(2^n)$ expected time to cover all nodes.

Another issue has to do with the fact that such instantaneous guarantees are quite strong and are not always present. For example, it is quite typical for a dynamic system, and its underlying temporal graph, to be disconnected at any given time (or most of them) but still to be *temporally connected*. In [MCS14], the authors proposed some metrics that can be used in such cases. These metrics capture properties that do not necessarily hold for every given instance but instead may require several steps until they are satisfied.

The first such metric is called the *connectivity time* of a temporal graph, which is the maximal time of keeping the two parts of any cut of the graph disconnected. Formally, the connectivity time (ct) of a temporal graph $D = (V, A)$ is the minimum $k \in \mathbb{N}$ such that for all times $t \in \mathbb{N}$ the static graph $(V, \bigcup_{i=t}^{t+k-1} A(i))$ is connected. In the extreme case in which the ct is 1, every instance of the temporal graph is connected and we obtain a continuously connected graph. On the other hand, greater ct allows for different cuts to be connected at different times in the ct-step interval and the resulting temporal graph can very well have disconnected instances. By using elementary arguments as those used in the beginning of this section, one can prove the following relation between the connectivity time and the temporal diameter d of a temporal graph: $\mathsf{ct} \le d \le (n-1)\mathsf{ct}$.

Another such metric is the *outgoing influence time* (oit), which is essentially the maximal time until the t-state of a node influences the state of another node (by a journey initiating at the former node and arriving at the latter) and captures the speed of information spreading. Formally, it is defined as the minimum $k \in \mathbb{N}$ such that for all $u \in V$ and all times $t, t' \ge 0$ such that $t' \ge t$ it holds that $|S_{(u,t)}(t' + k)| \ge \min\{|S_{(u,t)}(t')| + 1, n\}$, where $S_{(u,t)}(t')$ is just a more refined definition of the future set of a node u, already used implicitly in some of the previous paragraphs, containing all nodes $v \in V$ such that some journey initiating from u after time t arrives at v at most by time t'. Clearly, a continuously connected temporal graph has oit 1. Still there are temporal graphs with all their instances being disconnected that also have oit 1. A minimal such example is the following temporal graph, called the alternating matchings temporal graph in [MCS14]. Take a ring of an even number of nodes $n = 2l$, partition the edges into two disjoint perfect matchings A and B (each consisting of l edges) and alternate round after round between the edge sets A and B. In this temporal graph, every instance consists of $n/2$ components, still it is not hard to see that its oit is 1 and that its temporal diameter is $n/2$. It is also possible (but harder to construct) for a temporal graph to have disconnected instances (in fact, always being perfect matchings as before), oit 1, and additionally none of its edges to reappear in less than $n-1$ steps. Such a temporal graph can be defined (as shown in [MCS14]) based on an edge-coloring method due to Soifer [Soi09]. The node set is $V = \{u_1, u_2, \ldots, u_n\}$ where $n = 2l$, $l \ge 2$. Place u_n on the center and $u_1, \ldots, u_{n-1}$ on the vertices of a $(n-1)$-sided polygon. For each time $t \ge 1$ make available only the edges $\{u_n, u_{m_t(0)}\}$ for $m_t(j) := (t - 1 + j \mod n - 1) + 1$ and $\{u_{m_t(-i)}, u_{m_t(i)}\}$ for $i = 1, \ldots, n/2 - 1$; that is make available one edge joining the center to a polygon-vertex and all edges perpendicular to it. (e.g. see Figure 4 for $n = 8$ and $t = 1, \ldots, 7$). We should also mention that the oit of a temporal graph is always a lower bound on its ct, however the inverse is violated in the worst possible way as there are temporal graphs with oit 1 but $\mathsf{ct} = \Omega(n)$ [MCS14].

Most of the above special cases of temporal graphs have been exploited in the distributed computing literature for solving the *k-token dissemination problem* (see Section 4) and the related
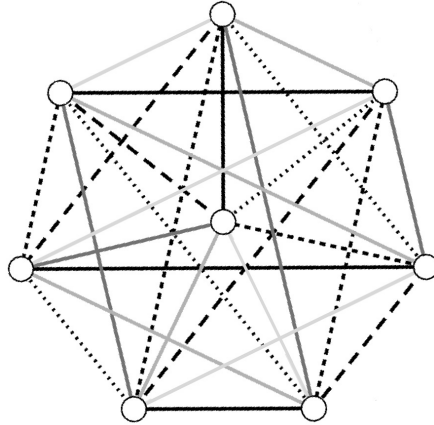
Fig. 4. Soifer's temporal graph for $n = 8$ and $t = 1, \ldots, 7$. In particular, in round 1 the graph consists of the black solid edges, then in round 2 the center becomes connected via a dotted edge to the next peripheral node clockwise and all edges perpendicular to it (the remaining dotted ones) become available, and so on, always moving clockwise. Figure adopted from [MCS14].

problems of counting the number of nodes in the network and computing functions over input values assigned to the nodes. First, as already mentioned in Section 4, in the case of a continuously connected temporal graph, [KLO10] gave a distributed algorithm that solves all of the above problems in $O(nk)$ rounds (in the worst case), provided that the nodes (i.e. the distributed processes) have unique identifiers and broadcast messages of size $O(\log n)$ bits. If there are no such identifiers, it was proved in [MCS13] that a pre-elected unique leader and the ability to transmit different messages to different neighbors (this is in contrast to *broadcast* communication, where in every round a node transmits a single message that is delivered to all its current neighbors) are all that one needs to assign unique identifiers to the processes and terminate (and then execute the algorithm of [KLO10] on the named system). In case of an f-connected temporal graph, it was shown in [KLO10] that the running time of the dissemination algorithm can be improved roughly by a factor of f, i.e. becomes $O(nk/f)$. If the temporal graph D is guaranteed to be "more stable", in the sense that every window of T consecutive instances of D has an underlying connected spanning subgraph that remains stable during the window (such temporal graphs, introduced in [KLO10], are called *T-interval connected* temporal graphs and by setting $T = 1$ we obtain the family of continuously connected temporal graphs as a special case), then $T/2$ tokens can be disseminated in only n rounds and the time-complexity of the above problems can be further improved by a factor of $T/2$ by "pipelining" transmissions through the edges that remain stable. If the temporal graph has ct upper bounded by C and the distributed processes know C, then there is a time-optimal algorithm that only needs time linear in the dynamic diameter d and in T (however, using large messages) [MCS14]. If, instead, the temporal graph has oit upper bounded by K then the above problems can be solved in $O(d + K)$ rounds (which is time-optimal) by using messages of size $O(n(\log K + \log n))$ or in $O(dn^2 + K)$ rounds by using messages of size $O(\log d + \log n)$ [MCS14].

6   Design Problems

So far, we have mainly presented problems in which a temporal graph is provided somehow (either in an offline or an online way) and the goal is to solve a problem on that graph. Another possibility

is when one wants to *design* a desired temporal graph. In most cases, such a temporal graph cannot be arbitrary, but it has to satisfy some properties prescribed by the underlying application. This design problem was introduced and studied in [MMCS13] (and its full version [MMS15]). An abstract definition of the problem is that we are given an underlying (di)graph G and we are asked to assign labels to the edges of G so that the resulting temporal graph $\lambda(G)$ minimizes some parameter while satisfying some connectivity property. The parameters studied in [MMCS13] were the maximum number of labels of an edge, called the *temporality*, and the total number of labels, called the *temporal cost*. The connectivity properties of [MMCS13] had to do with the preservation of a subset of the paths of G in time-respecting versions. For example, we might want to preserve all reachabilities between nodes defined by G, in the sense that for every pair of nodes u, v such that there is a path from u to v in G there must be a temporal path from u to v in $\lambda(G)$. Another such property is to guarantee in $\lambda(G)$ time-respecting versions of all possible paths of G. All these can be thought of as trying to *preserve* a connectivity property of a static graph in the temporal dimension while trying to minimize some cost measure of the resulting temporal graph.

The provided graph G represents some given static specifications, for example the available roads between a set of cities or the available routes of buses in the city center. In scheduling problems it is very common to have such a static specification and to want to organize a temporal schedule on it, for example to specify the precise time at which a bus should pass from a particular bus stop while guaranteeing that every possible pair of stops are connected by a route. Furthermore, it is very common that any such solution should at the same time take into account some notion of cost. Minimizing cost parameters may be crucial as, in most real networks, making a connection available and maintaining its availability does not come for free. For example, in wireless sensor networks the cost of making edges available is directly related to the power consumption of keeping nodes awake, of broadcasting, of listening the wireless channel, and of resolving the resulting communication collisions. The same holds for transportation networks where the goal is to achieve good connectivity properties with as few transportation units as possible.

For an example, imagine that we are given a directed ring $u_1, u_2, \ldots, u_n$ and we want to assign labels to its edges so that the resulting temporal graph has a journey for every simple path of the ring and at the same time minimizes the maximum number of labels of an edge. In more technical terms, we want to determine or bound the *temporality* of the ring subject to the *all paths* property. It is worth mentioning that the temporality (and the temporal cost) is defined as the *minimum* possible achievable value that satisfies the property, as, for example, is also the case for the chromatic number of a graph, which is defined as the minimum number of colors that can properly color a graph. Looking at Figure 5, it is immediate to observe that an increasing sequence of labels on the edges of path P_1 implies a decreasing pair of labels on edges (u_{n-1}, u_n) and (u_1, u_2). On the other hand, path P_2 uses first (u_{n-1}, u_n) and then (u_1, u_2) thus it requires an increasing pair of labels on these edges. It follows that in order to preserve both P_1 and P_2 we have to use a second label on at least one of these two edges, thus the temporality is at least 2. Next, consider the labeling that assigns to each edge (u_i, u_{i+1}) the labels $\{i, n+i\}$, where $1 \leq i \leq n$ and $u_{n+1} = u_1$. It is not hard to see that this labeling preserves all simple paths of the ring. Since the maximum number of labels that it assigns to an edge is 2, we conclude that the temporality is also at most 2. Taking both bounds into account, we may conclude that the temporality of preserving all simple paths of a directed ring is 2. Moreover, it holds that the temporality of graph G is lower bounded by the maximum temporality of its subgraphs, because if a labeling preserves all paths of G then it has to preserve all paths of any subgraph of G, paying every time the temporality of the subgraph.

So, for example, if the input graph G contains a directed ring then the temporality of G must be at least 2 (and could be higher depending on the structure of the rest of the graph).
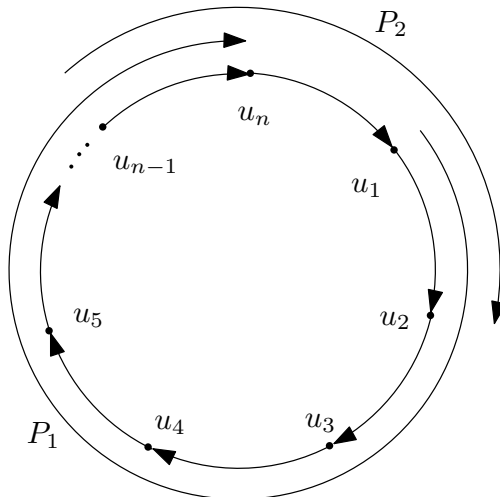

Fig. 5. Path P_2 forces a second label to appear on either (u_{n-1}, u_n) or (u_1, u_2).

Rings have very small temporality w.r.t. the all paths property, however there is a large family of graphs with even smaller. This is the family of directed acyclic graphs (DAGs). DAGs have the very convenient property that they can be topologically sorted. In fact, DAGs are the only digraphs that satisfy this property. A *topological sort* of a digraph G is a linear ordering of its nodes such that if G contains an edge (u, v) then u appears before v in the ordering. So, we can order the nodes from left to right and have all edges pointing to the right. Now, we can assign to the nodes the indices $1, 2, \ldots, n$ in ascending order from left to right and then assign to each edge the label of its tail, as shown in Figure 6. In this way, every edge obtains exactly one label and every path of G has been converted to a journey, because every path moves from left to right thus always moves to greater node indices. As these indices are also the labels of the corresponding edges, the path has strictly increasing labels which makes it a journey. This, together with the fact that the temporality is at least 1 in all graphs with non-empty edge sets, shows that the temporality of any DAG w.r.t. the all paths property is 1.
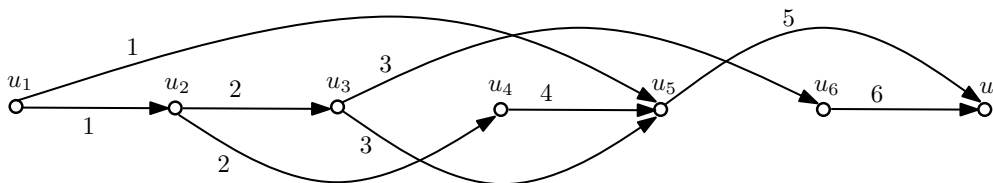

Fig. 6. A topological sort of a DAG. Edges are labeled by the indices of their tails (which are strictly increasing from left to right) and this labeling converts every possible path of the dag to a journey. For example, $(u_1, u_2, u_3, u_5, u_7)$ is a journey because its labels $(1, 2, 3, 5)$ are strictly increasing.

In both of the above examples, all paths could be preserved by using very few labels per edge. One may immediately wonder whether converting all paths to journeys can always be achieved with few labels per edge, e.g. a constant number of labels. However, a more careful look at the previous examples may provide a first indication that this is not the case. In particular, the ring example suggests that cycles can cause an increase of temporality, compared to graphs without cycles, like DAGs. Of course, a single ring only provides a very elementary exposition of this phenomenon, however as proved in [MMCS13], this core observation can be extended to give a quite general method for lower bounding the temporality. The idea is to identify a subset of the edges of G such that, for every possible permutation of these edges, G has a path following the direction of the permutation. Such subsets of edges, with many interleaved cycles, are called *edge-kernels* (see Figure 7 for an example) and it can be proved that the preservation of all paths of an edge-kernel on k edges yields a temporality of at least k. To see this, consider an edge-kernel $K = \{e_1, e_2, \ldots, e_k\}$ and order increasingly the labels of each edge. Now take an edge with maximum first label, move from it to an edge of maximum second label between the remaining edges, then move from this to an edge of maximum third label between the remaining edges, and so on. All these moves can be performed because K is an edge-kernel, thus there is a path no matter which permutation of the edges we choose. As in step i we are on the edge e with maximum i-th label, we cannot use the 1st, 2nd, $\ldots$, i-th labels of the next edge to continue the journey because none of these can be greater than the i-th label of e. So, we must necessarily use the $(i+1)$-th label of the next edge, which by induction shows that in order to go through the k-th edge in this particular permutation we need to use a kth label on that edge.
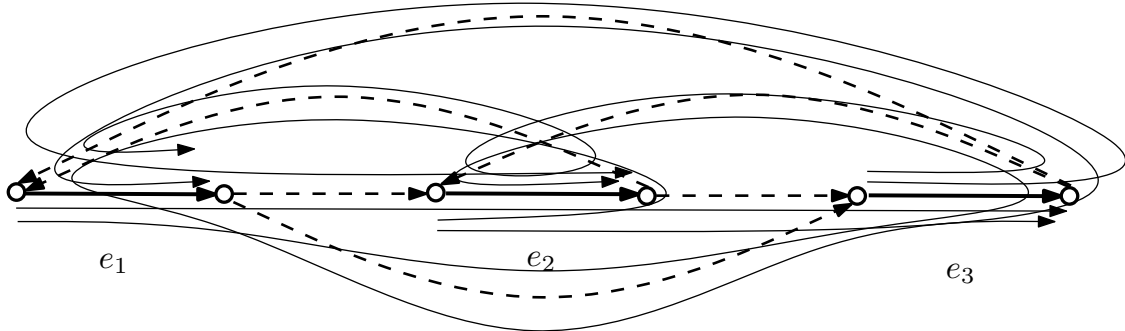

Fig. 7. The graph consists of the solid and dashed edges. The long curves highlight some of the paths that the graph defines. Edges e_1, e_2, and e_3 constitute an edge-kernel of the graph, because for every possible permutation of these edges the graph has a directed path (one of those highlighted in the figure) that traverses the edges in the order defined by the permutation. As a result, at least 3 labels must be assigned on an edge in order to preserve a temporal analogue of every possible path.

Also, as stated above, the temporality of a graph w.r.t to the all paths property is always lower bounded by the temporality of any of its subgraphs. As a consequence, we can obtain a lower bound on the temporality of a graph or of a whole graph family by identifying a large edge-kernel in it. For a simple application of this method, it is possible to show that in order to preserve all paths of a complete digraph, at least $\lfloor n/2 \rfloor$ labels are required on some edge. This is done by showing that complete digraphs have an edge-kernel of size $\lfloor n/2 \rfloor$. Moreover, it is possible to construct a planar graph containing an edge-kernel of size $\Omega(n^{1/3})$, which yields that there exist planar graphs with

temporality at least $\Omega(n^{1/3})$. It is worth noting that the absence of a large edge-kernel does not necessarily imply small temporality. In fact, it is an interesting open problem whether there are other structural properties of the underlying graph that could cause a growth of the temporality.

The above show that preserving all paths in time can be very costly in several cases. On the other hand, preserving only the reachabilities can always be achieved inexpensively. In particular, it can be proved that for every strongly connected digraph G, we can preserve a journey from u to v for every u, v for which there exists a path from u to v in G, by using at most two labels per edge [MMCS13]. Recall the crucial difference: now it suffices to preserve *a single path* from all possible paths that go from u to v. The result is proved by picking any node u and considering an in-tree rooted at u. We then label the edges of each level i, counting from the leaves, with label i, so that all paths of the tree become time-respecting (this also follows from the fact that the tree is a DAG so, as we discussed previously, all of its paths can be preserved with a single label per edge). Next we consider an out-tree rooted at u and we label that tree inversely, i.e. from the root to the leaves, and beginning with the label $i+1$. The first tree has a journey from every node to u arriving by time i and the second tree has a journey from u to every other node beginning at time $i+1$. This shows that there is a journey from every node to every other node. Moreover, this was achieved by using at most two labels per edge because every edge of the in-tree has a single label and every edge of the out-tree has a single label and an edge is in the worst case used by both trees, in which case it is assigned two labels. Furthermore, it can be proved that the temporality w.r.t. reachabilities of any digraph G is upper bounded by the maximum temporality of its strongly connected components. But we just saw that each component needs at most two labels, thus it follows that two labels per edge are sufficient for preserving all reachabilities of *any* digraph G.

Finally, we should mention an interesting relation between the temporality and the age of a temporal graph. In particular, restricting the maximum label that the labeling is allowed to use makes the temporality grow. For an intuition why this happens, consider the case in which there are many maximum length shortest paths between different pairs of nodes that all must be necessarily be preserved in order to preserve the reachabilities. Now if it happens that all of them pass through the same edge e but use e at many different times, then e must necessarily have many different labels, one for each of these paths. A simple example to further appreciate this is given in Figure 8. In that figure, each u_i-v_i path is a unique shortest path between u_i and v_i and has additionally length equal to the diameter (i.e. it is also a maximum one), so we must necessarily preserve all 5 u_i-v_i paths. Note now that each u_i-v_i path passes through e via its i-th edge. Each of these paths can only be preserved without violating $d(G)$ by assigning the labels $1, 2, \ldots, d(G)$, however note that then edge e must necessarily have all labels $1, 2, \ldots, d(G)$. To see this, notice simply that if any label i is missing from e then there is some maximum shortest path that goes through e at step i. As i is missing it cannot arrive sooner than time $d(G) + 1$ which violates the preservation of the diameter. Additionally, the following trade-off for the particular case of a ring can be proved [MMCS13]: If G is a directed ring and the age is $(n-1) + k$, then the temporality of preserving all paths is $\Theta(n/k)$, when $1 \leq k \leq n-1$, and $n-1$, when $k = 0$.

7 Temporal Versions of Other Standard Graph Problems: Complexity and Solutions

Though it is not yet clear how is the complexity of combinatorial optimization problems affected by introducing to them a notion of time, still there is evidence that complexity increases significantly
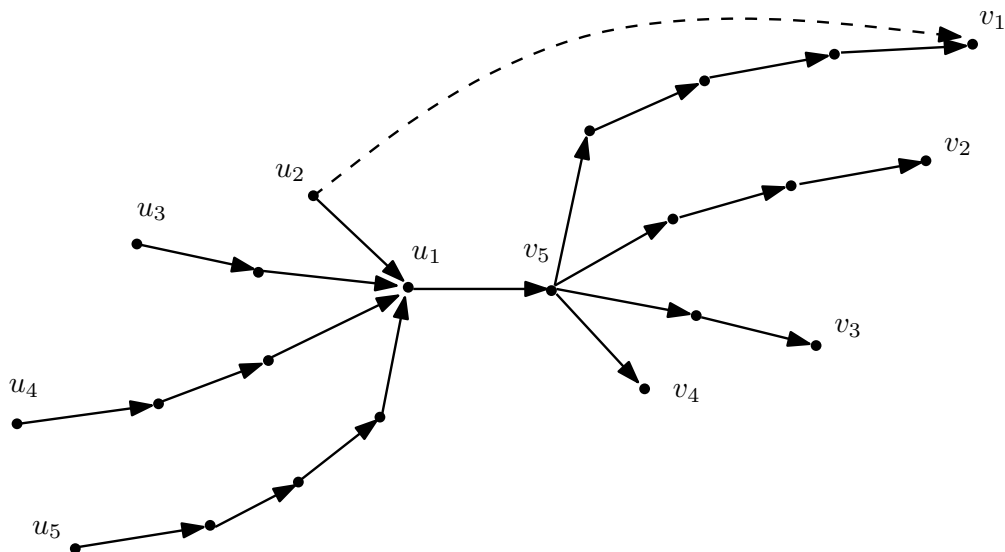
Fig. 8. In this example, restricting the maximum label to be at most equal to the diameter $d(G)$ forces the temporality to be at least $d(G)$.

and that totally novel solutions have to be developed in several cases. In an early but serious attempt to answer the above question, Orlin [Orl81] observed that many dynamic languages derived from **NP**-complete languages can be shown to be **PSPACE**-complete. This increase in complexity has been also reported in [BF03, XFJ03]. For example, [BF03] studied the computation of multicast trees minimizing the overall transmission time and to this end proved that it is **NP**-complete to compute strongly connected components in temporal graphs. Important evidence to this direction comes also from the rich literature on labeled graphs, a more general model than temporal graphs, with different motivation, and usually interested in different problems than those resulting when the labels are explicitly regarded as time moments. Several papers in this direction have considered labeled versions of polynomial-time solvable problems, in which the goal is to minimize/maximize the number of labels used by a solution. For example, the first labeled problem introduced in the literature was the LABELED MINIMUM SPANNING TREE problem, which has several applications in communication network design. This problem is **NP**-hard and many complexity and approximability results have been proposed (see e.g. [BL97, KW98]). On the other hand, the LABELED MAXIMUM SPANNING TREE problem has been shown polynomial in [BL97]. In [BLWZ05], the authors proved that the LABELED MINIMUM PATH problem is **NP**-hard and provided some exact and approximation algorithms. In [Mon05], it was proved that the LABELED PERFECT MATCHING problem in bipartite graphs is **APX**-complete (see also [TIR78] for a related problem).

A primary example of this phenomenon, of significant increase in complexity when extending a combinatorial optimization problem in time, is the fundamental MAXIMUM MATCHING problem. In its static version, we are given a graph $G = (V, E)$ and we must compute a maximum cardinality set of edges such that no two of them share an endpoint. MAXIMUM MATCHING can be solved in polynomial time by the famous Edmonds' algorithm [Edm65] (the time is $O(\sqrt{|V|} \cdot |E|)$ by the algorithm of [MV80]). Now consider the following temporal version of the problem, called TEMPORAL MATCHING in [MS14b]. In this problem, we are given a temporal graph $D = (V, A)$ and we are asked to decide whether there is a maximum matching M of the underlying static graph

of D that can be made temporal by selecting a single label $l \in \lambda(e)$ for every edge $e \in M$. For a single-labeled matching to be temporal it suffices to guarantee that no two of its edges have the same label. TEMPORAL MATCHING was proved in [MS14b] to be **NP**-complete. Then the problem of computing a maximum cardinality temporal matching is immediately **NP**-hard, because if we could compute such a maximum temporal matching in polynomial time, we could then compare its cardinality to the cardinality of a maximum static matching and decide TEMPORAL MATCHING in polynomial time. **NP**-completeness of TEMPORAL MATCHING can be proved by the sequence of polynomial-time reductions: BALANCED 3SAT $\leq_P$ BALANCED UNION LABELED MATCHING $\leq_P$ TEMPORAL MATCHING. In BALANCED 3SAT, which is known to be **NP**-complete, every variable x_i appears n_i times negated and n_i times non-negated and in BALANCED UNION LABELED MATCHING we are given a bipartite graph $G = ((X, Y), E)$, labels $L = \{1, 2, ..., h\}$, and a labeling $\lambda : E \to 2^L$, every node $u_i \in X$ has precisely two neighbors $v_{ij} \in Y$, and additionally both edges of u_i have the same number of labels, and we must decide whether there is a maximum matching M of G such that $\bigcup_{e \in M} \lambda(e) = L$ [MS14b].

Another interesting problem is the TEMPORAL EXPLORATION problem [MS14b]. In this problem, we are given a temporal graph and the goal is to visit all nodes of the temporal graph by a temporal walk, that possibly revisits nodes, minimizing the arrival time. The walk is commonly thought to be performed by an *agent*. The version of this problem for static graphs is well-known as GRAPHIC TSP. Though, in the static case, the decision version of the problem, asking whether a given graph is explorable, can be solved in linear time, in the temporal case it becomes **NP**-complete. Additionally, in the static case, there is a $(3/2 - \varepsilon)$-approximation for undirected graphs [GSS11] and a $O(\log n / \log \log n)$ for directed [AGM$^+$10].

In contrast to these, it was proved in [MS14b] that there exists some constant $c > 0$ such that TEMPORAL EXPLORATION cannot be approximated within cn unless **P** = **NP**, even in temporal graphs consisting, in every instance, of two strongly connected components (weakly connected with each other), by presenting a gap introducing reduction from HAMPATH. Additionally, it was proved that even the special case in which every instance of the temporal graph is strongly connected, cannot be approximated within $(2 - \varepsilon)$, for every constant $\varepsilon > 0$, unless **P** = **NP**. The reduction is from HAMPATH (input graph G, source s). The constructed temporal graph D consists of three strongly connected static graphs T_1, T_2, and T_3 persisting for the intervals $[1, n_1 - 1]$, $[n_1, n_2 - 1]$, and $[n_2, 2n_2 + n_1]$, respectively (it will be helpful at this point to look at Figure 9). We can restrict attention to instances of HAMPATH of order at least $2/\varepsilon$, without affecting its **NP**-completeness. We also set $n_2 = n_1^2 + n_1$ (in fact, we can set n_2 equal to any polynomial-time computable function of n_1). If G is hamiltonian, then for the arrival time, OPT, of an optimum exploration it holds that OPT $= n_1 + n_2 - 1 = n_1^2 + 2n_1 - 1$ while if G is not hamiltonian, then OPT $\geq 2n_2 + 1 = 2(n_1^2 + n_1) + 1 > 2(n_1^2 + n_1)$, which can be shown to introduce the desired $(2 - \varepsilon)$ gap.

The above inapproximability result has been recently improved by Erlebach *et al.* [EHK15] to $O(n^{1-\varepsilon})$ for any $\varepsilon > 0$. They first present a family of continuously connected temporal graphs that require $\Omega(n^2)$ steps to be explored. Any graph in the family is a star with $n/2$ c-nodes and $n/2$ l-nodes. The c-nodes go one after the other in the center of the star in a modular way and the l-nodes are always peripherals. Whenever an agent is on an l-node and wants to move to another l-node it has first to move to the center (which is a c-node) and then wait $n/2$ steps until that c-node becomes again the center. So, it takes $\Omega(n^2)$ steps to visit all l-nodes. The inapproximability is again proved by reduction from HAMPATH. The l-nodes are replaced by the graph G of HAMPATH and shortcut time-edges are added between the l-components but in such a way that can only be
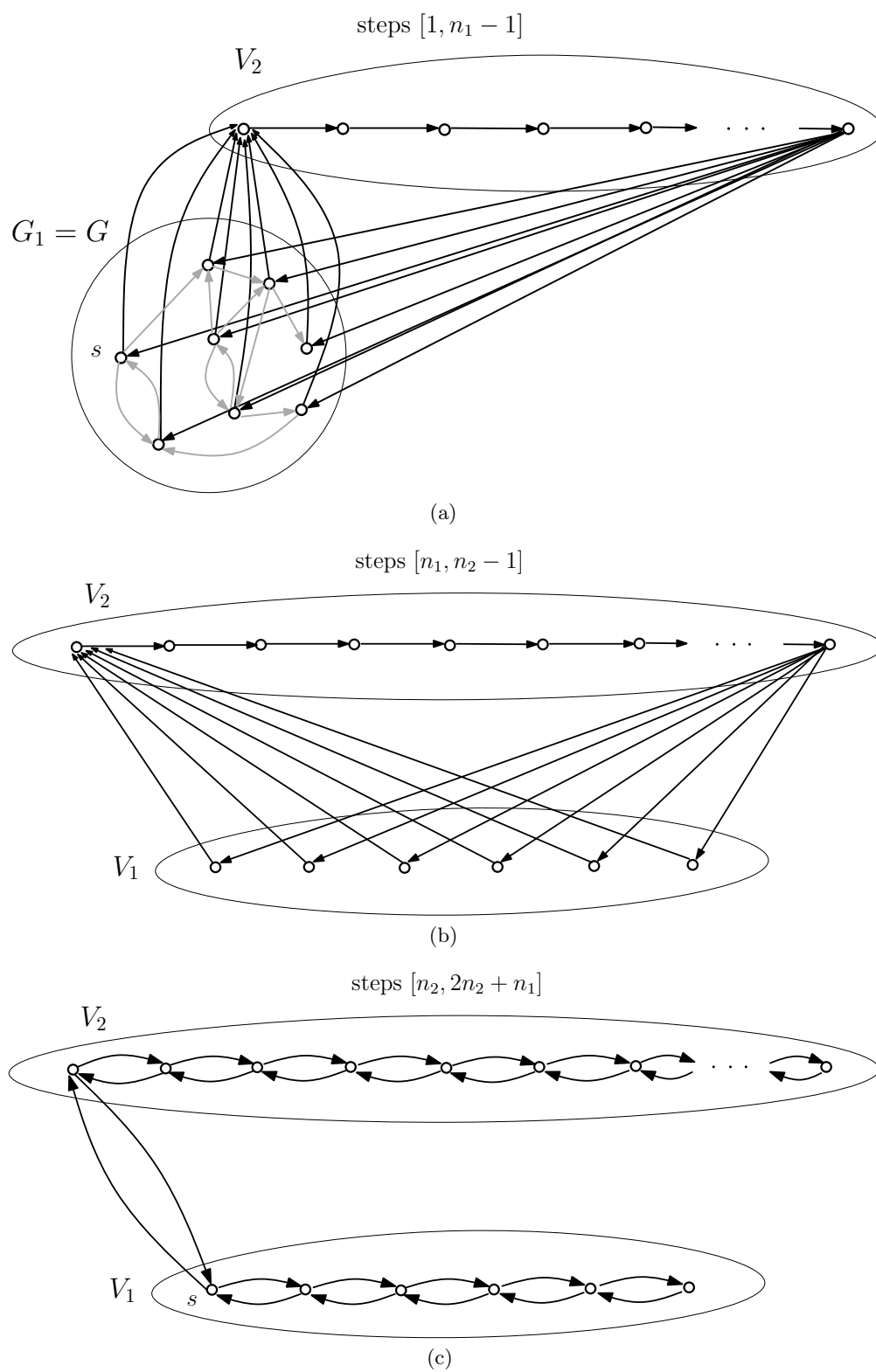
steps $[1, n_1 - 1]$

V_2

$G_1 = G$

s

(a)

steps $[n_1, n_2 - 1]$

V_2

V_1

(b)

steps $[n_2, 2n_2 + n_1]$

V_2

V_1

s

(c)

Fig. 9. The temporal graph constructed by the reduction. (a) T_1 (b) T_2 (c) T_3

23

used when G can be explored by a hamiltonian path. When G is hamiltonian, the constructed temporal graph can be explored by exploiting all shortcuts in linear time (i.e. $O(n)$). On the other hand, when G is not hamiltonian, the shortcuts cannot be used, or, if used, at least one node in each l-component will remain unvisited. In both cases, the components have to be revisited in the slow way of using the c-nodes which gives an $\Omega(k^2)$ time, where k is the number of c-nodes which is equal to the number of l-components. By making k large enough, in particular a large polynomial of the number of nodes of G, this becomes $\Omega(n^{2-\varepsilon})$. So, the introduced gap is $(O(n), \Omega(n^{2-\varepsilon}))$ which establishes the $O(n^{1-\varepsilon})$ inapproximability result. In the same work, an explicit construction of continuously connected temporal graphs that require $\Theta(n^2)$ steps to be explored was also given.

On the positive side, it is not hard to show that in continuously connected temporal graphs, Temporal Exploration can be approximated within the temporal diameter of the temporal graph [MS14b]. In [EHK15], the authors additionally studied the Temporal Exploration problem in other interesting restricted families of continuously connected temporal graphs, like those whose underlying graph has treewidth k (a work explicitly concerned with the treewidth of temporal graphs and its relation to the treewidth of static graphs is [MM13]), is a $2 \times n$ grid, a cycle, a cycle with a chord, or a bounded-degree planar graph, for which they provided upper bounds on exploration time. Several of these results were proved in the following interesting way. Instead of trying to explore by a single agent, the authors specified an exploration schedule for multiple agents and then applied the following reduction from the multi-agent case to the single-agent case. [6]

Lemma 2 ([EHK15]). *Let $G = (V, E)$ be a connected graph with n vertices. If any continuously connected $\lambda(G)$ can be explored in t steps with k agents, then any continuously connected $\lambda(G)$ can be explored in $O((t + n)k \log n)$ steps with one agent.*

Take any continuously connected $\lambda(G)$. Observe that all its temporal subgraphs of lifetime t are also continuously connected and therefore (by assumption) can be explored in t steps by k agents. Moreover, observe that all agents can return to the source node in $n - 1$ steps, because, as already discussed in Section 5, the temporal diameter of a continuously connected temporal graph is at most $n - 1$. [7] So, every $k + (n - 1)$ steps the k agents can re-explore $\lambda(G)$ and return to their origin. Call this a phase. Our goal is to have a single agent a follow in each phase the walk of one of the k agents. In particular, a implements the following greedy strategy: in the current phase it will mimic the walk of the agent that will visit the largest number of nodes not yet visited by a. The crucial observation is that, in every phase, all the k agents visit all n nodes, so the agent that visits the largest number of nodes not yet visited by a must necessarily visit at least a $1/k$ fraction of these nodes. This implies that, in every phase (lasting $t + n$ steps), a reduces the number of unexplored vertices by a factor of $1 - 1/k$. So, after $\lceil k \ln n \rceil + 1$ iterations the number of unexplored nodes is less than $n(1 - 1/k)^{k \ln n} \leq n e^{-\ln n} = 1$, therefore all nodes have been explored.

Finally, we should mention that [FMS13] is another study of the exploration problem in temporal graphs with periodic edge-availabilities, from a distributed computing perspective. We defer the description of these results until Section 8, where the focus is on temporal graphs with recurrent and periodic properties.

---

[6] In the k-agent case, all k agents begin from the same source node and each node of the graph must be visited by at least one of the agents.

[7] It is worth mentioning here that the problem of returning all agents to the origin may be also viewed as a simplified version of Lemma 1, where now we do not limit the number of tokens (here agents) that a node may transmit (allow to leave from it) per round.

Another demanding problem that becomes even more challenging in its temporal version is the famous TRAVELING SALESMAN PROBLEM, in which a graph with non-negative costs on its edge is provided and the goal is to find a tour visiting every node exactly once (called a *TSP tour*), of minimum total cost. In one version of the problem, introduced in [MS14b], the digraph remains static and complete throughout its lifetime but now each edge is assigned a cost that may change from instance to instance. So, the dynamicity has now been transferred from the topology to the costs of the edges. The goal is to find (by an offline centralized algorithm) a *temporal TSP tour* of minimum total cost, where the cost of a tour is the sum of the costs of the time-edges that it traverses. The authors of [MS14b] introduced and studied the special case of this problem in which the costs are chosen from the set $\{1, 2\}$. In particular, there is a cost function $c : A \to \{1, 2\}$ assigning a cost to every time-edge of the temporal graph (see Figure 10 for an illustration). This is called the TEMPORAL TRAVELING SALESMAN PROBLEM WITH COSTS ONE AND TWO and abbreviated TTSP(1,2). Now observe that the famous (static) ATSP(1,2) problem is a special case of TTSP(1,2) when the lifetime of the temporal graph $D = (V, A)$ is restricted to n and $c(e, t) = c(e, t')$ for all edges e and times t, t'. This immediately implies that TTSP(1,2) is also **APX**-hard [PY93] and cannot be approximated within any factor less than 207/206 [KS13] and the same holds for the interesting special case of TTSP(1,2) with lifetime restricted to n, that we will also discuss.

In the static case, one easily obtains a $(3/2)$-factor approximation for ATSP(1,2) by computing a perfect matching maximizing the number of ones and then patching the edges together arbitrarily. This works well, because such a minimum cost perfect matching can be computed in polynomial time in the static case by Edmonds' algorithm [Edm65] and its cost is at most half the cost of an optimum TSP tour, as the latter consists of two perfect matchings. The $3/2$ factor follows because the remaining $n/2$ edges that are added during the patching process cost at most n, which, in turn, is another lower bound to the cost of the optimum TSP tour. This was one of the first algorithms known for ATSP(1,2). Other approaches have improved the factor to the best currently known $5/4$ [Blä04]. Unfortunately, as we already discussed in the beginning of this section, even the apparently simple task of computing a matching maximizing the number of ones is not that easy in temporal graphs. A simple modification of those arguments yields that the problem remains **NP**-hard if we require consecutive labels (in an increasing ordering) of the matching to have a time difference of at least two. Such time-gaps are necessary for constructing a time-respecting patching of the edges of the matching. In particular, if two consecutive edges of the matching had a smaller time difference, then the patching-edge would share time with at least one of them and the resulting tour would not have strictly increasing labels.

Our inability to compute a temporal matching in polynomial time, still does not exclude the possibility to find good approximations for it and then hope to be able to use them for obtaining good approximations for TTSP(1,2). Two main approaches were followed in [MS14b]. One was to reduce the problem to MAXIMUM INDEPENDENT SET (MIS) in $(k + 1)$-claw free graphs and the other was to reduce it to k'-SET PACKING, for some k and k' to be determined. The first approach gives a $(7/4 + \varepsilon)$-approximation $(= 1.75 + \varepsilon)$ for the generic TTSP(1,2) and a $(12/7 + \varepsilon)$-approximation $(\approx 1.71 + \varepsilon)$ for the special case of TTSP(1,2) in which the lifetime is restricted to n (the latter is obtained by approximating a temporal path packing instead of a matching). The second approach improves these to $1.7 + \varepsilon$ for the general case and to $13/8 + \varepsilon = 1.625 + \varepsilon$ when the lifetime is n. In all the above cases, $\varepsilon > 0$ is a small constant (not necessarily the same in all cases) adopted from the factors of the approximation algorithms for independent set and set packing.
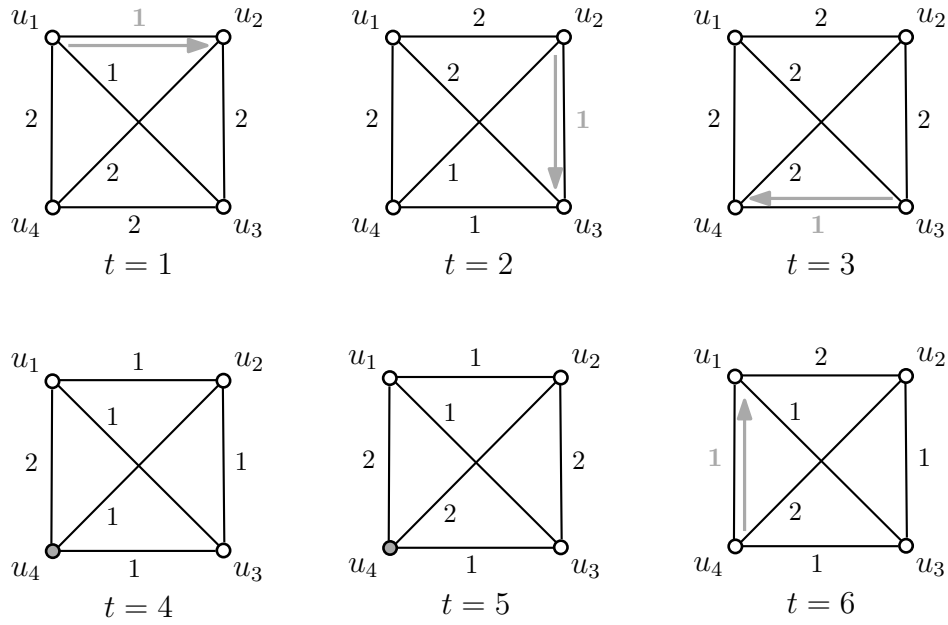
Fig. 10. An instance of TTSP(1,2) consisting of a complete temporal graph $D = (V, A)$, where $V = \{u_1, u_2, u_3, u_4\}$, and a cost function $c : A \to \{1, 2\}$ which is presented by the corresponding costs on the edges. For simplicity, D is an undirected temporal graph. Observe that the cost of an edge may change many times, e.g. the cost of u_2u_3 changes 5 times while of u_1u_4 changes only once. Here, the lifetime of the temporal graph is 6 and it is greater than $|V|$. The gray arcs and the nodes filled gray (meaning that the tour does not make a move and remains on the same node for that step) represent the TTSP tour $(u_1, 1, u_2, 2, u_3, 3, u_4, 6, u_1)$ that has cost $4 = |V|$ and therefore it is an optimum TTSP tour.

26

We summarize now how the first of these approximations works. Consider the static expansion $H = (S, E)$ of D and an edge $e = (u_{(i-1)j}, u_{ij'}) \in E$. There are three types of conflicts, each defining a set of edges that cannot be taken together with e in a temporal matching (with only unit time differences): (i) Edges of the same row as e, because these violate the unit time difference constraint (ii) edges of the same column as $u_{(i-1)j}$, because these share a node with e, thus violate the condition of constructing a matching, and (iii) edges of the same column as $u_{ij'}$, for the same reason as (ii). Next consider the graph of edge conflicts $G = (E, K)$, where $(e_1, e_2) \in K$ iff e_1 and e_2 satisfy some of the above constraints (observe that the node set of G is equal to the edge set of the static expansion H). Observe that temporal matchings of D are now equivalent to independent sets of G. Moreover, G is 4-claw free meaning that there is no 4-independent set in the neighborhood of any node. To see that it is 4-claw free, take any $e \in E$ and any set $\{e_1, e_2, e_3, e_4\}$ of four neighbors of e in G. There are only 3 constraints thus at least two of the neighbors, say e_i and e_j, must be connected to e by the same constraint. But then e_i and e_j must also satisfy the same constraint with each other thus they are also connected by an edge in G. Now, from [Hal95], there is a factor of 3/5 for MIS in 4-claw free graphs, which implies a $(3/5)$-approximation algorithm for temporal matchings. Simple modifications of the above arguments yield a $\frac{1}{2+\varepsilon}$-approximation algorithm for temporal matchings with time-differences at least two. Additionally, it can be proved that a $(1/c)$-factor approximation for the latter problem implies a $(2 - \frac{1}{2c})$-factor approximation for TTSP(1,2). All these together, yield a $(7/4 + \varepsilon)$-approximation algorithm for TTSP(1, 2) [MS14b].

An immediate question, which is currently open, is whether there is a $(3/2)$-factor approximation algorithm either for the general TTSP(1,2) or for its special case with lifetime restricted to n (the reader may have observed that in the temporal case we have not yet achieved even the simplest factor of the static case).

8 Recurrent and Periodic Temporal Graphs

A quite common property of the underlying temporal graphs of many real-world systems is *recurrence*, informally meaning *the guaranteed reappearance of some instantaneous or temporal property in a given amount of time*. Depending on the application and the available knowledge about the system, recurrence may mean reappearance of a property either in some *unknown* finite time or in some *known* finite time-bound. The most typical example of a recurrent property is the availability of an edge. For instance, a temporal graph could satisfy that whenever an edge appears, it must reappear in at most T steps. Other such properties could be the reappearance of a whole instance, or of a particular subgraph of it, or the reappearance of a particular journey (the latter is a temporal recurrent property). A special type of recurrence which is of particular interest and shows up in many applications is *periodicity*. Informally, a recurrent property of a temporal graph *is periodic if its recurrence occurs at regular intervals*. In the existing literature, periodicity usually refers to the edges of the temporal graph. A way to define it is: *If $G = (V, E)$ is the underlying graph, every edge e in a $E' \subseteq E$ has an associated period p_e and it holds that $e \in A(t)$ if and only if $e \in A(t + p_e)$, for all times t, when the lifetime is infinite, and for all times $t \leq l - p_e$, in case of a finite lifetime l.* Observe that if $E' = E$ and $p_e = p$ for all $e \in E$, then the temporal graph repeats itself after every p steps. The reader is encouraged to consult [CFQS12] for a systematic classification of different types of temporal graphs, including a classification of the recurrent and periodic ones.

A natural source of edge-periodic temporal graphs are the so called *carrier graphs* of [FMS13], where the time-edges are defined by the periodic movements of some mobile entities, called carriers,

over the edges of an underlying graph. In particular, an edge of the underlying graph becomes available whenever a carrier passes over that edge. This type of temporal graphs is a natural abstraction of several real-world systems like transportation networks with inherent periodicity, like public transports with fixed timetables (e.g. buses, trains, planes, subway), low earth orbiting (LEO) satellite systems, and security guards' tours [FMS13, CFQS12]. The focus in [FMS13] was on solving the following exploration problem: *An agent is placed on an arbitrary node and the goal is for the agent to visit all nodes of the temporal graph without ever waiting on some node* (unless possibly its carrier waits on that node, which can be modeled by self-loop time-edges). So the agent must always follow the route of a carrier and can switch from one carrier to another if it happens that the two carriers meet on the same node at the same time. It was proved that of the nodes are indistinguishable to the agent and the agent does not know the length of the longest route (which is also the maximum period $\max_{e \in E} p_e$ of the temporal graph), then the problem is impossible to solve. On the other hand, if the agent can distinguish the nodes and the agent knows either n or an upper-bound on the maximum period then the problem becomes solvable. Moreover, the authors studied the time-complexity of the problem, counted as usual by the number of time-steps required to explore the graph, which in this case coincides with the number of moves of the agent over the nodes of the graph, because the agent never waits on some node. In particular, for a graph defined by k carriers, they proved that if $p_e = p$ for all $e \in E$ (which they called the *homogeneous* case) then $\Omega(kp)$ steps are necessary, otherwise the lower bound becomes $\Omega(kp^2)$, where now $p = \max_{e \in E} p_e$. These bounds hold even if the agent knows n, k, and p and has unlimited memory. Additionally, the authors investigated the impact of the carriers' routes structure on the complexity of the exploration problem and to this end they they gave similar lower bounds for the special cases of simple and circular routes. Finally, they gave matching upper bounds (by providing optimal exploration algorithms) for all of their lower bounds. The impact of strengthening the agent with the ability to wait at nodes was studied in [IW11]. There are also papers that have been concerned with routing in periodic temporal graphs as e.g. [LW09].

Another interesting family of temporal graphs consists of those temporal graphs whose availability times are provided by some succinct representation. Generally speaking, there are two main types of such compact representations: either a function, which is the case that we will now discuss, or a probability distribution, which will be discussed in the next section.

We now present a family of temporal graphs in which a set of functions describes the availability times of the edges. The underlying graph is a complete static graph $G = (V, E)$. Each $e \in E$ has an associated linear function of the form $f_e(x) = a_e x + b_e$, where $x, a_e, b_e \in \mathbb{N}_{\geq 0}$. For example, if an edge e has $f_e(x) = 3x + 4$, then it is available at times $4, 7, 10, 13, 16, \ldots$. Clearly, the temporal graph that we obtain in this manner is $D = (V, A)$ where $A(r) = \{e \in E : f_e(x) = r$ for some $x \in \mathbb{N}\}$. If we are additionally provided with a lifetime l of the temporal graph then we just restrict $A(r)$ to $r \leq l$. This model may be viewed as a very special case of the periodic model discussed so far. Even though we are now free to delay the first appearance of an edge for any finite number of steps (by setting b_e appropriately), still, every edge e, if observed after time b_e, has period a_e, thus the temporal graph, if observed after time $\max_{e \in E} b_e$, satisfies the definition of edge-periodicity. For example, if $a_e = 1$ and $b_e = 100$, then for the first 99 steps e will be unavailable and from that point on it will be forever available. The pattern obtained after time b_e is a very special type of periodic pattern in which every edge is available once and then it is unavailable for $a_e - 1$ steps. It is not hard to see that the definition of periodicity given in the beginning of this section, allows much more than this. For example, it allows the availability pattern of an edge during its period

to be arbitrarily non-linear; the availability times of a particular edge in the interval $[1, p_e]$ could be given by $f_e(x) = x^2$ and then the periodic model just has to repeat the pattern forever. Such patterns cannot be generated by the linear functions consider here.

There is an immediate way to obtain the rth instance of the temporal graph, for any time r, based on the provided linear functions. For every $e \in E$, the rth instance contains edge e iff $(r - b_e)/a_e$ is integer. It is important to note that, in this type of temporal graphs, algorithmic solutions that depend at least linearly on the lifetime l are not acceptable. The reason is that the lifetime l is provided in binary so a linear dependence on l grows exponentially in the binary representation of l. Foremost journeys in such graphs can be easily computed by a variation of the algorithm discussed in Section 2.1.

Now consider the following problem. We are given two edges e_1 and e_2 with corresponding functions $f_{e_1}(x) = a_1 x + b_1$ and $f_{e_2}(x) = a_2 x + b_2$ and we are asked to determine whether there is some instance having both edges, that is, to determine whether there exist x_1 and x_2 such that $f_{e_1}(x_1) = f_{e_2}(x_2) \Leftrightarrow a_1 x_1 + b_1 = a_2 x_2 + b_2 \Leftrightarrow a_1 x_1 = a_2 x_2 + (b_2 - b_1)$. So, in fact, we are seeking for a x_2 such that $a_1 \mid a_2 x_2 + (b_2 - b_1)$ (where '$\mid$' reads as "divides") and we have reduced our problem to the problem of determining whether $c \mid ax + b$ for some x. Now imagine a right oriented ring of c nodes numbered $0, 1, \ldots, c - 1$. Consider a process beginning from node b (mod c) and making clockwise jumps of length a in each round (where a round corresponds to an increment of x by 1). We have that the process falls at some point on node 0 iff $c \mid ax + b$ for some x. Viewed in this way, our problem is equivalent to checking whether $ax + b \equiv 0$ (mod c) is solvable for the unknown x. This, in turn, may easily take the form $ax \equiv b'$ (mod c) (given that $-b \equiv b'$ (mod c)) for $a > 0$ and $c > 0$ (equalities to 0 correspond to trivial cases of our original problem). Clearly, we have reduced our problem to the problem of detecting whether a modular linear equation admits a solution which is well-known to be solvable in polynomial time. In particular, a modular linear equation $ax \equiv b'$ (mod c) has a solution iff $\gcd(a, c) \mid b'$ (see e.g. [CLRS01], Corollary 31.21, page 869). Additionally, by solving the equation we can find all solutions modulo c in $O(\log c + \gcd(a, c))$ arithmetic operations (see e.g. [CLRS01], page 871).

Note that in the case where $b_1 = b_2 = 0$ then the answer to the problem is always "yes" as $a_1 x_1 = a_2 x_2$ trivially holds for $x_1 = a_2$ and $x_2 = a_1$ (provided that $a_1 a_2$ does not exceed the lifetime of the network if a lifetime is specified). In particular, if we are asked to determine the foremost instance containing both edges then this reduces to the computation of $\text{lcm}(a_1, a_2)$ (where lcm is the least common multiple) which in turn reduces to the computation of $\gcd(a_1, a_2)$ by the equation $\text{lcm}(a_1, a_2) = |a_1 a_2| / \gcd(a_1, a_2)$.

Now let us slightly simplify our model in order to obtain a solution to a more generic version of the above problem. We restrict the edge functions $a_i x + b_i$ so that $b_i < a_i$, e.g. $7x + 4$. Then clearly, each such function corresponds to the whole equivalence subclass of $\mathbb{N}$ modulo a_i containing b_i, that is, $[b_i]_{a_i} = \{b_i + xa_i : x \in \mathbb{N}\}$. So, for example, $7x + 4$ corresponds to $\{4, 11, 18, 25, \ldots\}$ in contrast to $7x + 11$ that was allowed before and would just give the subset $\{11, 18, 25, \ldots\}$ of the actual class. Consider now the following problem: "We are given a subset E' of the edge set E and we want to determine whether there is some instance of the temporal graph containing all edges in E'". For simplicity, number the edges in E' from 1 to k. Formally, we want to determine the existence of some time t such that, for all $i \in \{1, 2, \ldots, k\}$, there exists x_i such that $t = a_i x_i + b_i$, or equivalently, $t \equiv b_i$ (mod a_i). Clearly, we have arrived at a set of simultaneous linear congruences and we can now apply the following known results.

Theorem 3 (see e.g. [BS96], Theorem 5.5.5, pg 106). *The system of congruences* $t \equiv b_i$ *(mod a_i), $1 \le i \le k$, has a solution iff $b_i \equiv b_j$ (mod $\gcd(a_i, a_j)$) for all $i \ne j$. If the solution exists, it is unique modulo $\mathrm{lcm}(a_1, a_2, \ldots, a_k)$.*

Corollary 1 (see e.g. [BS96], Corollary 5.5.6, pg 106). *Let $a_1, a_2, \ldots, a_k$ be integers, each ≥ 2, and define $a = a_1 a_2 \cdots a_k$, and $a' = \mathrm{lcm}(a_1, a_2, \ldots, a_k)$. Given the system S of congruences $t \equiv b_i$ (mod a_i), $1 \le i \le k$, we can determine if S has a solution, using $O(\lg^2 a)$ bit operations, and if so, we can find the unique solution modulo a', using $O(\lg^2 a)$ bit operations.*

We may now return to the original formulation of our model in which $a_i x + b_i$ does not necessarily satisfy $b_i < a_i$. First keep in mind that $t_{min} = \max_{i \in E'}\{b_i\}$ is the minimum time for every edge from E' to appear at least once (in fact, at that time, the last edge of E' appears). So we cannot hope to have them all in one instance sooner than this. Now notice that $a_i x + b_i$ is equivalent to $a_i x' + (b_i \bmod a_i)$ for $x' \ge \lfloor b_i/a_i \rfloor$; for example, $7x + 15$ is equivalent to $7x' + 1$ for $x' \ge 2$. In this manner, we obtain an equivalent setting in which again $b_i < a_i$ for all i but additionally for every i we have a constraint on x of the form $x \ge q_i$. We may now ignore the constraints and apply Theorem 3 to determine whether there is a solution to the new set of congruences as there is a solution that satisfies the constraints iff there is one if we ignore the constraints (the reason being that the constraints together form a finite lower bound while there is an infinite number of solutions). If there is a solution it will be a unique solution modulo $\mathrm{lcm}(a_1, a_2, \ldots, a_k)$ corresponding to an infinite number of solutions if expanded. From these solutions we just have to keep those that are not less than t_{min} (in case we want to find the actual solutions to the system).

Consider now the following type of bounded edge recurrence. Assume an undirected underlying graph $G = (V, E)$, which is additionally sparse, having only $O(n)$ edges. For every edge $e \in E$ there is a constant integer R_e such that the maximum number of consecutive steps in which e is unavailable is at most R_e and at least R_e/c, for some constant $c > 1$. Assume additionally that the resulting temporal graph has connected instances. It was proved in [EHK15] that such a temporal graph can be explored in $O(n)$ steps. The idea is to first round down each R_e to the nearest power of 2 and obtain a new set of values R'_e. Then calculate a minimum spanning tree T of G w.r.t. to the edge costs R'_e and explore G by following an Euler tour of T. Such a tour visits every $e \in T$ at most twice and the walk has to wait at most R_e steps for an $e \in T$ to become available. Consequently, if the walk begins at time 1, it must have visited all nodes at most by time $\sum_{e \in T} 2R_e = 2 \sum_{e \in T} R_e < 4 \sum_{e \in T} R'_e$ (because $R_e < 2R'_e$, for all $e \in E$). It remains to prove that $\sum_{e \in T} R'_e = O(n)$. The proof strategy is interesting and it is worth describing. Take any $k \ge 0$ such that T contains at least one edge e with $R'_e = 2^k$. Temporarily removing the edges of T with $R'_e = 2^k$ partitions T into some components C_i. Now add the removed edges and also all inter-component edges of G. The idea is to take the 2^k cost of an edge leaving a component C_i and distribute it to all edges (set E_i) leaving C_i (and joining it to other components), in such a way that each edge will receive a sufficiently small charge. The above partitioning process will be repeated for all $k \ge 0$ for which there is an $e \in T$ with cost 2^k and we must ensure that in the end of all repetitions every edge of G will have received only a constant charge. This will imply that the total cost of T has been distributed to the edges of G in such a way that every edge of G has received a charge upper bounded by a constant H and as G has $O(n)$ edges, the total cost of T must have been at most $H \cdot O(n) = O(n)$, which is what we wanted to prove. In fact, H turns out to be equal to $4c$, where c is the constant assumed in the R_e/c minimum consecutive unavailability of an edge e. For the precise charging mechanism, which is quite technical, relying on the facts that every

inter-component edge must have cost at least 2^k (otherwise it could be used as a swapping edge to obtain a tree cheaper than T) and that in every step at least one of the edges of each E_i must be present (due to the continuous connectivity assumption), the reader is referred to [EHK15].

Finally, we should say a few things about the temporal graph model considered by Orlin in [Orl81, Orl84] that we briefly mentioned in Section 1. In that model, an underlying digraph $G = (V, E)$ is provided with each $e \in E$ having a single integer label (possibly negative). The labels in this case represent *transit times*, that is if a walk at a node u chooses to cross edge $e = (u, v)$ at time t, then it will arrive at node v at time $t + \lambda(e)$ (where now $\lambda : E \to \mathbf{Z}$). Then G induces the following type of temporal graph: Consider an infinite static expansion as those that we have already seen, without yet any edges. For every time i and every edge $e = (u_j, u_k) \in E$ with transit time $t = \lambda(e)$ add edge $(u_{ij}, u_{(i+t)k})$ in the static expansion, unless $i + t < 0$ (in case of a negative t) in which case the edge is not added. Observe that if $t = 0$, then the edge is an intra-row one, if $t < 0$ then the edge is from one row to a previous one, and if $|t| > 1$ then an edge connects non-consecutive rows of the static expansion. None of these cases can be produced by the temporal graph model with which we are concerned in this article. Note also that in the above model the temporal graph repeats itself in every step, so we could say that it is a 1-periodic temporal graph. Though being very different from the models considered in the modern treatment of the subject, still it is a periodic temporal model and also the results proved for it are resounding and possibly give some first indications of what to expect when adding to combinatorial optimization problems a time dimension. In particular, it was proved in [Orl84] that connected components, eulerian paths, odd length circuits, and minimum average cost spanning trees can be solved in polynomial time. It was observed that in each case the problem reduces to a static-graph problem which, however, is *in no case the same* as the temporal problem. For example, determining the strongly connected components of a temporal graph does not reduce to determining the strongly connected components of a static graph unless one allows the static graph to have an exponential number of nodes. Also it was proved that the (apparently simple) problem of determining if there is a directed path from u to v in such a temporal graph is **NP**-complete. Finally, in an earlier paper [Orl81], Orlin had focused on the complexity of classical difficult graph-problems and had showed that many of them that are **NP**-complete actually become **PSPACE**-hard when defined on such temporal graphs.

9 Random Temporal Graphs

Another model of temporal graphs with succinct representation, is the model of random temporal graphs. Consider the case in which each edge (of an underlying clique) just picks independently and uniformly at random *a single time-label* from $[r] = \{1, 2, \ldots, r\}$. So it gets label $t \in [r]$ with probability $p = r^{-1}$. We mainly present here a set of unpublished results concerning this model, jointly developed by the author of the present article and Paul Spirakis in 2012. We also discuss results from [AGMS14], which is a very recent paper concerned with the same issue.

We first calculate the probability that given a specific path $(u_1, u_2, \ldots, u_{k+1})$ of length k a journey appears on this path. We begin with the directed case. First, let us obtain a weak but elegant upper bound. Partition $[r]$ into $R_1 = \{1, \ldots, \lfloor r/2 \rfloor\}$ and $R_2 = \{\lfloor r/2 \rfloor + 1, \ldots, r\}$. Clearly, $\mathrm{P}(journey) \leq \mathrm{P}(\text{no } R_2R_1 \text{ occurs})$ as any journey assignment cannot have two consecutive selections such that the first one is from R_2 and the second from R_1. So, it suffices to calculate $\mathrm{P}(\text{no } R_2R_1 \text{ occurs})$. Notice that the assignments in which no R_2R_1 occurs are of the form $(R_1)^i(R_2)^j$ for $i + j = k$, e.g. $R_1R_1R_2R_2R_2$ and there are $k + 1$ of them. In contrast, all possible assignments

are 2^k corresponding to all possible ways to choose k times with repetition from $\{R_1, R_2\}$. So, P(no R_2R_1 occurs) $= k/2^k$ (as all assignments are equiprobable, with probability 2^{-k}) and we conclude that P($journey$) $\le k/2^k$, which, interestingly, is independent of r; e.g. for $k = 6$ we get a probability of at most 0.09375 for a journey of length 6 to appear.

For any specific assignment of labels $t_1, t_2, \ldots, t_k$ of this path, where $t_i \in [r]$ ($[r] = \{1, 2, \ldots, r\}$), the probability that this specific assignment occurs is simply p^k. So, all possible assignments are equiprobable and we get

$$\text{P}(journey) = \frac{\#\text{ strictly increasing assignments}}{\#\text{ all possible assignments}} = \frac{\binom{r}{k}}{r^k},$$

where $\binom{r}{k}$ follows from the fact that any strictly increasing assignment is just a unique selection of k labels from the r available and any such selection corresponds to a unique strictly increasing assignment. So, for example, for $k = 2$ and $r = 10$ we get a probability of 9/20 which is a little smaller than $1/2$ as expected, due to the fact that there is an equal number of strictly increasing and strictly decreasing assignments but we also loose all remaining assignments which in this case are only the ties (that is, those for which $t_1 = t_2$).

Now it is easy to compute the expected number of journeys of length k. Let S be the set of all directed paths of length k and let Y_p be an indicator random variable which is 1 if a journey appears on a specific $p \in S$ and 0 otherwise. Let also X_k be a random variable giving the number of journeys of length k. Clearly, $\text{E}(X_k) = \text{E}(\sum_{p \in S} Y_p) = \sum_{p \in S} \text{E}(Y_p) = |S| \cdot \text{P}(\text{a journey appears on a specific path of length } k) = n(n-1) \cdots (n-k)\binom{r}{k}r^{-k} \ge (n-k)^k\binom{r}{k}r^{-k}$. Now, if we set $n \ge r/\binom{r}{k}^{(1/k)} + k$, we get $E(X) \ge 1$. A simpler, but weaker, formula can be obtained by requiring $n \ge r + k$. In this case, we get $E(X) \ge \binom{r}{k}$. So, for example, a long journey of size $k = n/2$ that uses all available labels is expected to appear provided that $n \ge 2r$ (to see this, simply set $k = r$).

We will now try to obtain bounds on the probability that a journey of length k appears on a random temporal graph. Let us begin from a simple case, namely the one in which $k = 4$, that is, we want to calculate the probability that a journey of length 4 appears. Let the r.v. X be the number of journeys of length 4 and let X_p be an indicator for path $p \in S$, where S is the set of all paths of length 4. Denote $n(n-1) \cdots (n-k)$ by $(n)_{k+1}$ First note that $E(X) = (n)_5\binom{r}{4}r^{-4} = \Theta(n^5)$ and clearly goes to ∞ for every r. However, we cannot yet conclude that $\text{P}(4 - journey)$ is also large. To show this we shall apply the second moment method. We will make use of Chebyshev's inequality $\text{P}(X = 0) \le \text{Var}(X)/[\text{E}(X)]^2$ and of the following well-known theorem:

Theorem 4 ([MS12]). *Suppose $X = \sum_{i=1}^n X_i$, where X_i is an indicator for event A_i. Then,*

$$\text{Var}(X) \le \text{E}(X) + \sum_i \text{P}[A_i] \underbrace{\sum_{j:j \sim i} \text{P}(A_j \mid A_i)}_{\Delta_i},$$

where $i \sim j$ denotes that i depends on j. Moreover, if $\Delta_i \le \Delta$ for all i, then

$$\text{Var}(X) \le \text{E}(X)(1 + \Delta).$$

So, in our case, we need to estimate $\Delta_p = \sum_{p' \sim p} \text{P}(A_{p'} \mid A_p)$. If we show that $\Delta_p \le \Delta$ for all $p \in S$ then we will have that $\text{Var}(X) \le \text{E}(X)(1 + \Delta)$. If we additionally manage to show that

$\Delta/E(X) = o(1)$, then $\Delta = o(\mathrm{E}(X))$ which tells us that $\mathrm{Var}(X) = o([\mathrm{E}(X)]^2)$. Putting this back to Chebyshev's inequality we get that $\mathrm{P}(X = 0) = o(1)$ as needed.

So, let us try to bound Δ_p appropriately. Clearly, p' cannot be a journey if it visits some edges of p in inverse order (than the one they have on p). Intuitively, the two paths must have the same orientation. We distinguish cases based on the number of edges shared by the two paths. First of all, note that if p' and p have precisely i edges in common then $\mathrm{P}(A_{p'} \mid A_p) \leq \binom{r}{k-i}/r^{k-i}$ which becomes $\binom{r}{4-i}/r^{4-i}$ in our case. The reason is that the $k - i$ edges of p' that are not shared with p must at least obtain an increasing labeling. If we also had taken into account that that labeling should be consistent to the labels of the shared edges then this would decrease the probability. So we just use an upper bound which is sufficient for our purposes.

Case 1: 1 shared edge. If a single edge is shared then there are $k\binom{n-k+1}{k-1}(k-1)!4 = 16 \cdot 3!\binom{n-5}{3}$ different paths p' achieving this as there are k ways to choose the shared edge, $\binom{n-k+1}{k-1}$ to choose the missing nodes (nodes of p' not shared with p), $(k-1)!$ ways to order those nodes, and, in this particular example, 4 ways to arrange the nodes w.r.t. the shared edge. In particular, we can put all nodes before the shared edge, all nodes after, 2 nodes before and 1 node after, or 1 node before and 2 nodes after. We conclude that the probability that $\sum_{|p' \cap p|=1} \mathrm{P}(A_{p'} \mid A_p) \leq 16 \cdot 3!\binom{n-5}{3}\binom{r}{3}/r^3 = O(n^3)$.

Case 2: 2 shared edges. In this case, we can have all possible $\binom{k}{2} = \binom{4}{2}$ 2-sharings. Let us denote by e_1, e_2, e_3, e_4 the edges of p. For the sharings (e_1, e_2), (e_2, e_3), and (e_3, e_4) we get in total $3\binom{n-k-1}{k-2}(k-2)!4 = 24\binom{n-5}{2}$ paths. For (e_1, e_3), (e_2, e_4) we get $2(n-k-1) = 2(n-5)$. For (e_1, e_4) we get $(n-5)$ in case we connect the 2 edges by an intermediate node (i.e. go from the head of e_1 to some u not in p and then form u to the tail of e_4) and $2(n-5)$ in case we connect e_1 directly to e_4 and use an external node either before or after, so in total $3(n-5)$ paths. Putting these all together we get $\sum_{|p' \cap p|=2} \mathrm{P}(A_{p'} \mid A_p) \leq [24\binom{n-5}{2} + 5(n-5)]\binom{r}{2}/r^2 = O(n^2)$.

Case 3: 3 shared edges. Here there are just 2 choices for the 3 shared edges, namely (e_1, e_2, e_3) and (e_2, e_3, e_4), the reason being that if the edges are not consecutive then a fourth edge must be necessarily shared and the 2 paths would coincide. As there are $(n - k - 1)$ ways to choose the missing node and 2 ways to arrange that node we get $2(n - k - 1)2 = 4(n - 5)$ and consequently $\sum_{|p' \cap p|=3} \mathrm{P}(A_{p'} \mid A_p) \leq 4(n-5)\binom{r}{1}/r^1 = O(n)$.

So, we have $\Delta_p \leq \Delta = O(n^3)$ and $\Delta/E(X) = O(n^3)/\Theta(n^5) = o(1)$ which applied to Theorem 4 gives $\mathrm{Var}(X) \leq \mathrm{E}(X)(1 + \Delta) = o([\mathrm{E}(X)]^2)$ and this in turn applied to Chebyshev's inequality gives the desired $\mathrm{P}(X = 0) \leq \mathrm{Var}(X)/[\mathrm{E}(X)]^2 = o(1)$. We conclude that:

Theorem 5 ([MS12]). *For all $r \geq 4$, almost all random temporal graphs contain a journey of length 4.*

Now let us turn back to our initial $(n)_{k+1}\binom{r}{k}r^{-k}$ formula of $\mathrm{E}(X)$ (which holds for all k). This gives $\mathrm{E}(X) \geq (n)_{k+1}/k^k$, which, for all $k = o(n)$ and all $r \geq k$, goes to ∞ as n grows. We will now try to generalize the ideas developed in the $k = 4$ case to show that for any not too large k almost all random temporal graphs contain a journey of length k. Take again a path p of length k and another path p' of length k that shares i edges with p. We will count rather crudely but in a sufficient way for our purposes. As again the shared edges can be uniquely oriented in the order they appear on p, there are at most $\binom{k}{i}$ ways to choose the shared edges (at most because some selections force more than i sharings to occur). Counting the tail of the first edge and the head of every edge, these i edges occupy at least $i + 1$ nodes, so at most $k + 1 - i - 1 = k - i$ nodes are missing from p' and thus there are at most $\binom{n-k-1}{k-i}$ ways to choose those nodes. Moreover there are

at most $(k-i)!$ ways to permute them on p'. Finally, we have to place those nodes relative to the i shared edges. In the worst case, the i edges define $i+1$ slots that can be occupied by the nodes in $\binom{k-i+(i+1)-1}{(i+1)-1} = \binom{k}{i}$ ways. In total, we have $N = \binom{k}{i}^2 \binom{n-k-1}{k-i}(k-i)!\binom{k}{i}$ different paths and the corresponding probability is

$$\sum_{|p'\cap p|=i} \mathrm{P}(A_{p'} \mid A_p) \le N\binom{r}{k-i}/r^{k-i} \le \binom{k}{i}^2\binom{n-k-1}{k-i}$$

$$\le \binom{k^2}{i}\binom{n-k-1}{k-i}.$$

So we have that

$$\Delta_p = \sum_{i=1}^{k-1} \sum_{|p'\cap p|=i} \mathrm{P}(A_{p'} \mid A_p) \le \sum_{i=0}^{k}\binom{k^2}{i}\binom{n-k-1}{k-i}$$

$$= \binom{n+k^2-k-1}{k} \le \binom{n+k^2}{k} = \Delta.$$

The first equality follows from the Chu-Vandermonde identity $\sum_{i=0}^{k}\binom{m}{i}\binom{z}{k-i} = \binom{m+z}{k}$ by setting $z = n-k-1$ and $m = k^2$ as needed in our case.

Thus, we have $\Delta = \binom{n+k^2}{k}$ and for $k^2 = o(n)$ we have $\Delta \sim (n)_k/k!$. At the same time we have $\mathrm{E}(X) = (n)_{k+1}\binom{r}{k}/r^k \sim (n)_{k+1}/k!$ (for large r), thus $\Delta/\mathrm{E}(X) \sim (n)_k/(n)_{k+1} = o(1)$ as needed. So we have $\mathrm{Var}(X) = o([\mathrm{E}(X)]^2)$ and we again get that $\mathrm{P}(X = 0) \le \mathrm{Var}(X)/[\mathrm{E}(X)]^2 = o(1)$. Captured in a theorem:

Theorem 6 ([MS12]). *For all $k = o(\sqrt{n})$ and all $r = \Omega(n)$, almost all random temporal graphs contain a journey of length k.*

However, there seems to be some room for improvements if one counts more carefully.

Now take any two nodes s and t in V. We want to estimate the arrival time of a foremost journey from s to t. Let X be the random variable of the arrival time of the foremost s-t journey. Let us focus on $\mathrm{P}(X \le 2)$. Denote by $l(u,v)$ the label chosen by edge (u,v). Given a specific node $u \in V \setminus \{s,t\}$ we have that $\mathrm{P}(l(s,u) \ne 1 \text{ or } l(u,t) \ne 2) = 1-\mathrm{P}(l(s,u) = 1 \text{ and } l(u,t) = 2) = 1-r^{-2}$. Thus, $\mathrm{P}(\forall u \in V \setminus \{s,t\} : l(s,u) \ne 1 \text{ or } l(u,t) \ne 2) = (1-r^{-2})^{n-2}$ We have:

$$\mathrm{P}(X \le 2) = 1 - \mathrm{P}(X > 2)$$

$$= 1 - \mathrm{P}(l(s,t) \notin \{1,2\})\mathrm{P}(\forall u \in V \setminus \{s,t\} : l(s,u) \ne 1 \text{ or } l(u,t) \ne 2)$$

$$= 1 - \frac{r-2}{r}(1-r^{-2})^{n-2}$$

$$\ge 1 - (1-r^{-2})^{n-2}$$

$$\ge 1 - e^{-(n-2)/r^2}, \text{ for } n \ge 2 \text{ and } r > \sqrt{n-1}.$$

So, even if $r = \Theta(\sqrt{n})$ we have that $\mathrm{P}(X \le 2) \to 1-1/e^c$ (for some constant $c \le 1$) as n goes to infinity, so we have a constant probability of arriving by time 2 at t. Clearly, for smaller values of r (smaller w.r.t. n) we get even better chances of arriving early. For another example, let $n = 10^4$

and $r = \sqrt{n}/\log n = 25$. As $P(X \leq 2)$ is almost equal to $1 - (1 - r^{-2})^{n-2}$ we get that it is almost equal to 1 in this particular case. For even greater r, e.g. $r = \sqrt{n} = 100$, we still go very close to 1.

The following proposition gives a bound on the temporal diameter of undirected random temporal graphs, by exploiting well-known results of the Erdös-Renyi ($\mathcal{G}(n, p)$) model (cf. [Bol01]).

Proposition 1 ([MS12]). *Almost no temporal graph has temporal diameter less than* $[(\ln n + c + o(1))/n]r$.

To see this, observe that if $k < [(\ln n + c + o(1))/n]r$ then $p = k/r < (\ln n + c + o(1))/n$. Consider now the temporal subgraph consisting only of the first k labels $[k] = \{1, 2, \ldots, k\}$. By the connectivity threshold of the static $\mathcal{G}(n, p)$ model this subgraph is almost surely disconnected implying that almost surely the temporal diameter is greater than k.

So, for example, if $r = O(n)$ almost no temporal graph has temporal diameter $o(\log n)$. Note, however, that the above argument is not sufficient to show that almost every temporal graph has temporal diameter at least $[(\ln n + c + o(1))/n]r$. Though it shows that in almost every graph the subgraph consisting of the labels $[k]$, for $k \geq \lceil r(\ln n + c + o(1))/n \rceil$ is connected, it does not tell us whether that connectivity also implies temporal connectivity (that is, the existence of journeys).

We should also mention that [AGMS14] studied the temporal diameter of the directed random temporal graph model for the case of $r = n$, and proved that it is $\Theta(\log n)$ w.h.p. and in expectation. In fact, they showed that information dissemination is very fast w.h.p. even in this hostile network with regard to availability. Moreover, they showed that the temporal diameter of the clique is crucially affected by the clique's lifetime, α, e.g., when α is asymptotically larger than the number of vertices, n, then the temporal diameter must be $\Omega(\frac{\alpha}{n} \log n)$. They also defined the *Price of Randomness* metric in order to capture the cost to pay per link and guarantee temporal reachability of all node-pairs by local random available times w.h.p..

The idea of [AGMS14] to establish that the temporal diameter is $O(\log n)$ is as follows. Given an instance of such a random temporal clique, the authors pick any source node s and any sink node t and present an algorithm trying to construct a journey from s arriving at t at most by time $O(\log n)$. The algorithm expands two fronts, one beginning from s and moving forward (in fact, an out-tree rooted at s) and one from t moving backward (an in-tree rooted at t). Beginning from s, all neighbors that can be reached in one step in the interval $(0, c_1 \log n]$, are visited. Next the front moves on to all neighbors of the previous front that can be reached in one step in the interval $(c_1 \log n, c_1 \log n + c_2]$. The process continues in the same way, every time replacing the current front by all its neighbors that can be reached in the next c_2 steps. A similar backward process is executed from t. These processes are executed for $d = \Theta(\log n)$ steps resulting in the final front of s and the final front of t. Note that the front of t begins from the interval $(2c_1 \log n + (2d-1)c_2, 2c_1 \log n + 2dc_2]$ and every time subtracts a c_2. Finally, the algorithm tries to find an edge from the final front of s to the final front of t with the appropriate label in order to connect the journey from s to the journey to t in a time-respecting way and obtain the desired s-t journey of duration $\Theta(\log n)$ (determined by the interval of the first front of t, and in particular by $2c_1 \log n + 2dc_2$). Via probabilistic analysis it can be proved that, with probability at least $1 - 1/n^3$, the final front of s consists of $\Theta(\sqrt{n})$ nodes and that the same holds for the front of t. Moreover, it can be proved that again with probability at least $1 - 1/n^3$ the desired edge for the final front of s to the final front of t exists, and thus we can conclude that there is a probability of at least $1 - 3/n^3$ of getting from s to t by a journey arriving at most by time $\Theta(\log n)$. Finally, it suffices to observe that the probability that there exists a pair of nodes $s, t \in V$ for which the algorithm fails is less than $n^2(3/n^3) = 3/n$, thus with probability at least $(1 - 3/n)$ the temporal diameter is $O(\log n)$, as required.

10   Conclusion and Open Problems

A wide range of existing and emerging communication and computing systems are characterized by an inherent degree of dynamicity. Traditionally, this degree was low, e.g. as the result of system failures, like a crash of a process in a distributed system, and therefore it was naturally treated as an exception. Only recently have researchers started to no longer consider the dynamicity of a system as a rare event but as the law. This has been driven by the need to model systems (and problems for them) in which dynamicity is always present and usually at a very high rate. Graphs have been proved an invaluable tool for representing and enabling the formal treatment of any possible discrete set of objects and relations between them. However, if these relations, or even also the objects themselves, change constantly then a single graph can no longer capture these changes. Systems of this type can be naturally modeled by a sequence of graphs, each one of them capturing the "state" of the system at a given time. Even though we are still resorting to a graph as our basic representation tool, we now have an ordered (in time) sequence of graphs and, furthermore, all non-instantaneous events or properties of the system (such as communication between two nodes, reachability, paths, etc) can no longer be defined on a single graph but on a whole subsequence of graphs; in other words, they have to respect the evolution of the system in time. This simple fact has two important consequences. The first is that many of the existing methods and techniques for graphs (including fundamental theorems and algorithms) cannot be directly applied and others become totally inadequate, even when trying to solve an appropriately adapted version of a standard graph problem. Typically the problem becomes much harder to solve and in many cases becomes totally different from the original problem, requiring radically different approaches. The second, and most important, is that such a time-sequence of graphs is a new mathematical entity, called a *temporal graph*, which generalizes graphs and is a rich source of *new problems* with an inherent temporal nature that could not have shown up without clearly representing the evolution of the graph in a distinct time dimension.

Though it is still quite early to anticipate the full range of potential applications, there is already strong evidence that there is room for the development of a rich theory. As is always the case, the groundwork will be laid by our ability to identify and formulate radically new problems and not just by studying adjusted versions of existing graph-problems. Real dynamic systems, such as the mobile Internet, transportation networks, social networks, ad-hoc sensor networks, and mobile robotic systems, to name a few, is the natural place to look for such problems. Still, the existing literature has already identified some first challenging research directions and technical problems whose further investigation has the potential to push forward the area of temporal graphs. In this article, we gave a brief overview of these developments. The following paragraph summarizes some interesting (according to the author's personal perspective) open problems. For more details and open problems the interested reader is encouraged to consult the referenced papers.

First, is there a general rule underlying the complexity increase of a graph-problem when that problem is extended in time? Of course, this depends on how we reformulate the problem. Then, can the different possible reformulations be partitioned into classes each with its own effect on complexity? Moreover, many natural applications require an algorithm to operate on a temporal graph without knowing or being able to accurately predict the future instances of the graph. This is, for example, the case in a system of interacting mobile entities where each single entity cannot tell how the other entities will behave in future steps. It might be the case that the right treatment of such settings is via online algorithms and analysis, however little effort has been devoted to this. There is also, already, a large set of more specific and more technical questions, of which we

will only give some indicative examples; many more can be found in the referenced papers. We saw that the max-flow min-cut theorem still holds in temporal graphs and that there is a natural reformulation of Menger's theorem. It would be very valuable to check the validity of many other fundamental results of graph theory. For example, is there some analogue of Kuratowski's planarity theorem (according to which a graph is planar iff it does not contain a subdivision of K_5 or $K_{3,3}$) for temporal graphs? A possible temporal analogue of planarity here could be the following: A temporal graph is time-planar if there is a way to draw it so that no two intersecting edges share a time-label. Given such a definition we can also ask "what is the minimum number of time-labels guaranteeing time-planarity of a non-planar graph?" (coloring methods might help in attacking this and similar problems). Another interesting problem comes from extremal graph theory. It is well-known, due to a beautiful theorem of Mantel from 1907, that every graph of order n and size greater than $\lfloor n^2/4 \rfloor$ contains a triangle. An interesting observation for temporal graphs is that any triangle assigned different time-labels is a time-triangle. Thus, if we insist on different time-labels, the appearance of two adjacent edges excludes the appearance of some third edge if one wants to keep the temporal graph time-triangle free. On the other hand, if one picks some matching, then those edges can appear any possible number of times and at any possible order. These indicate that if one wants to keep a temporal graph time-triangle free then he/she should sacrifice either dynamicity or connectivity (similar things hold in the more general requirement of keeping the graph time-acyclic). This trade-off needs a precise characterization. Moreover, in the temporal-graph design problem (discussed in Section 6) there is great room for approximation algorithms (or even randomized algorithms) for all combinations of optimization parameters and connectivity constraints, or even exact polynomial-time algorithms for specific graph families. Also, though it has turned out to be a generic lower-bounding technique related to the existence of a large edge-kernel in the underlying graph G, we still do not know whether there are other structural properties of the underlying graph that could cause a growth of the temporality (i.e. the absence of a large edge-kernel does not necessarily imply small temporality). Another thing that we do not know is whether a $(3/2)$-factor is within reach either for the general TTSP(1,2) or for the special case with lifetime restricted to n. What we are looking for here is a new direct approximation algorithm, some better reduction, or the harder way of improving the known approximations for MAXIMUM INDEPENDENT SET in k-claw free graphs or for SET PACKING. It would also be interesting to know how the generic metric TSP problem behaves in temporal graphs. Is there some temporal analogue of triangle inequality (even a different assumption) that would make the problem approximable? Is there some temporal analogue of symmetry (e.g. periodicity) that does not trivialize the temporal dimension of the problem? Consider also another model in which the lifetime is n and every edge that changes "state" (e.g. cost/availability) remains in the same "state" for at least k steps. For example, $k = 1$ allows for a fully dynamic temporal graph while $k = n$ gives a static graph. So, k in some sense expresses the degree of dynamicity of the graph. We expect here to have interesting trade-offs between k and the approximation factors obtained for temporal problems. There is also a wide-open road for investigation towards (temporal) properties of random temporal graphs. Are some of the most natural temporal properties, like temporal diameter and temporal connectivity, characterized by a thresholding behavior, as is the case with the phase transition in the Erdös-Renyi random graph model? We should finally note that distributed computation in dynamic networks is a very active and steadily growing sub-area of Distributed Computing, with its own already identified non-trivial questions and goals.

References

[AAD+06] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18[4]:235–253, 2006.

[AGM+10] A. Asadpour, M. X. Goemans, A. Madry, S. O. Gharan, and A. Saberi. An $O(\log n/\log\log n)$-approximation algorithm for the asymmetric traveling salesman problem. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 379–389. SIAM, 2010.

[AGMS14] E. C. Akrida, L. Gasieniec, G. B. Mertzios, and P. G. Spirakis. Ephemeral networks with random availability of links: Diameter and connectivity. In *Proceedings of the 26th ACM symposium on Parallelism in algorithms and architectures (SPAA)*, pages 267–276. ACM, 2014.

[AKL08] C. Avin, M. Koucký, and Z. Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *Proceedings of the 35th international colloquium on Automata, Languages and Programming (ICALP), Part I*, pages 121–132. Springer-Verlag, 2008.

[AKM14] E. Aaron, D. Krizanc, and E. Meyerson. DMVP: foremost waypoint coverage of time-varying graphs. In *Graph-Theoretic Concepts in Computer Science*, pages 29–41. Springer, 2014.

[APRU12] J. Augustine, G. Pandurangan, P. Robinson, and E. Upfal. Towards robust and efficient computation in dynamic peer-to-peer networks. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 551–569. SIAM, 2012.

[Ber96] K. A. Berman. Vulnerability of scheduled networks and a generalization of Menger's theorem. *Networks*, 28[3]:125–134, 1996.

[BF03] S. Bhadra and A. Ferreira. Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks. In S. Pierre, M. Barbeau, and E. Kranakis, editors, *Ad-Hoc, Mobile, and Wireless Networks*, volume 2865 of *Lecture Notes in Computer Science*, pages 259–270. Springer Berlin Heidelberg, 2003.

[BL97] H. Broersma and X. Li. Spanning trees with many or few colors in edge-colored graphs. *Discussiones Mathematicae Graph Theory*, 17[2]:259–269, 1997.

[Blä04] M. Bläser. A 3/4-approximation algorithm for maximum atsp with weights zero and one. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 61–71. Springer, 2004.

[BLWZ05] H. Broersma, X. Li, G. Woeginger, and S. Zhang. Paths and cycles in colored graphs. *Australasian Journal on Combinatorics*, 31:299–311, 2005.

[Bol98] B. Bollobás. *Modern Graph Theory*. Graduate Texts in Mathematics. Springer; Corrected edition (July 1, 1998), 1998.

[Bol01] B. Bollobás. *Random graphs*. Cambridge Studies in Advanced Mathematics. Cambridge University Press; 2nd edition, 2001.

[BS72] B. Baker and R. Shostak. Gossips and telephones. *Discrete Mathematics*, 2[3]:191–193, 1972.

[BS96] E. Bach and J. Shallit. *Algorithmic number theory, volume 1: efficient algorithms*, volume 1. MIT press, 1996.

[CFQS12] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27[5]:387–408, 2012.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001.

[CMM+08] A. E. Clementi, C. Macci, A. Monti, F. Pasquale, and R. Silvestri. Flooding time in edge-markovian dynamic graphs. In *Proceedings of the 27th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 213–222, 2008.

[CPMS07] A. E. Clementi, F. Pasquale, A. Monti, and R. Silvestri. Communication in dynamic radio networks. In *Proceedings of the 26th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 205–214. ACM, 2007.

[DGH+87] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 1–12. ACM, 1987.

[DPR+13] C. Dutta, G. Pandurangan, R. Rajaraman, Z. Sun, and E. Viola. On the complexity of information spreading in dynamic networks. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 717–736. SIAM, 2013.

[Edm65] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17[3]:449–467, 1965.

[EHK15]   T. Erlebach, M. Hoffmann, and F. Kammer. On temporal graph exploration. In *42nd International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science, pages 444–455. Springer, 2015.

[Fer04]   A. Ferreira. Building a reference combinatorial model for manets. *Network, IEEE*, 18[5]:24–29, 2004.

[FMS13]   P. Flocchini, B. Mans, and N. Santoro. On the exploration of time-varying networks. *Theoretical Computer Science*, 469:53–68, 2013.

[FT98]   L. Fleischer and É. Tardos. Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, 23[3]:71–80, 1998.

[GSS11]   S. O. Gharan, A. Saberi, and M. Singh. A randomized rounding approach to the traveling salesman problem. In *Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 550–559, Washington, DC, USA, 2011. IEEE Computer Society.

[Hal95]   M. M. Halldórsson. Approximating discrete collections via local improvements. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 160–169. SIAM, 1995.

[HG97]   F. Harary and G. Gupta. Dynamic graph models. *Mathematical and Computer Modelling*, 25[7]:79–87, 1997.

[HHL88]   S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18[4]:319–349, 1988.

[HK12]   B. Haeupler and F. Kuhn. Lower bounds on information dissemination in dynamic networks. In *Proceedings of the 26th International Symposium on Distributed Computing (DISC)*, volume 7611 of *Lecture Notes in Computer Science*, pages 166–180. Springer Berlin Heidelberg, 2012.

[Hol15]   P. Holme. Modern temporal network theory: A colloquium. *The European Physical Journal B (EPJ B)*, 2015. To appear. Also as an arXiv preprint arXiv:1508.01303.

[HS12]   P. Holme and J. Saramäki. Temporal networks. *Physics reports*, 519[3]:97–125, 2012.

[IW11]   D. Ilcinkas and A. M. Wade. On the power of waiting when exploring public transportation systems. In *15th International Conference on Principles of Distributed Systems (OPODIS)*, pages 451–464. Springer, 2011.

[KK02]   D. Kempe and J. Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms. In *Proceedings of the IEEE 43rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 471–480. IEEE, 2002.

[KKK00]   D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. In *Proceedings of the 32nd annual ACM symposium on Theory of computing (STOC)*, pages 504–513, 2000.

[KLO10]   F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM symposium on Theory of computing (STOC)*, pages 513–522, New York, NY, USA, 2010. ACM.

[KO11]   F. Kuhn and R. Oshman. Dynamic networks: models and algorithms. *SIGACT News*, 42:82–96, March 2011. Distributed Computing Column, Editor: Idit Keidar.

[KOM11]   F. Kuhn, R. Oshman, and Y. Moses. Coordinated consensus in dynamic networks. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing (PODC)*, pages 1–10, 2011.

[Kos09]   V. Kostakos. Temporal graphs. *Physica A: Statistical Mechanics and its Applications*, 388[6]:1007–1023, 2009.

[KS13]   M. Karpinski and R. Schmied. On improved inapproximability results for the shortest superstring and related problems. *Proc. 19th CATS*, pages 27–36, 2013.

[KSSV00]   R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *Proceedings of the IEEE 41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 565–574. IEEE, 2000.

[KW98]   S. O. Krumke and H.-C. Wirth. On the minimum label spanning tree problem. *Information Processing Letters*, 66[2]:81–85, 1998.

[KZ14]   S. Kontogiannis and C. Zaroliagis. Distance oracles for time-dependent networks. In *41st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 713–725. Springer, 2014.

[Lei92]   F. T. Leighton. *Introduction to parallel algorithms and architectures*, volume 188. Morgan Kaufmann San Francisco, 1992.

[LW09]   C. Liu and J. Wu. Scalable routing in cyclic mobile networks. *Parallel and Distributed Systems, IEEE Transactions on*, 20[9]:1325–1338, 2009.

[MCS11a]   O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Mediated population protocols. *Theoretical Computer Science*, 412[22]:2434–2450, May 2011.

[MCS11b]   O. Michail, I. Chatzigiannakis, and P. G. Spirakis. *New Models for Population Protocols*. N. A. Lynch (Ed), Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool, 2011.

[MCS13]   O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Naming and counting in anonymous unknown dynamic networks. In *15th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 281–295. Springer, 2013.

[MCS14]   O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Causality, influence, and computation in possibly disconnected synchronous dynamic networks. *Journal of Parallel and Distributed Computing*, 74[1]:2016–2026, 2014.

[Men27]   K. Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10[1]:96–115, 1927.

[Mic15a]   O. Michail. An introduction to temporal graphs: An algorithmic perspective. In *Algorithms, Probability, Networks, and Games - Scientific Papers and Essays Dedicated to Paul G. Spirakis on the Occasion of His 60th Birthday*, Lecture Notes in Computer Science, pages 308–343. Springer, 2015.

[Mic15b]   O. Michail. Terminating distributed construction of shapes and patterns in a fair solution of automata. In *Proceedings of the 34th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 37–46, 2015.

[MM13]   B. Mans and L. Mathieson. On the treewidth of dynamic graphs. In D.-Z. Du and G. Zhang, editors, *Computing and Combinatorics*, volume 7936 of *Lecture Notes in Computer Science*, pages 349–360. Springer Berlin Heidelberg, 2013.

[MMCS13]   G. B. Mertzios, O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Temporal network optimization subject to connectivity constraints. In *40th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 7966 of *Lecture Notes in Computer Science*, pages 657–668. Springer Berlin Heidelberg, 2013.

[MMS15]   G. B. Mertzios, O. Michail, and P. G. Spirakis. Temporal network optimization subject to connectivity constraints. *CoRR*, abs/1502.04382, 2015. Full version of [MMCS13].

[Mon05]   J. Monnot. The labeled perfect matching in bipartite graphs. *Information Processing Letters*, 96[3]:81–88, 2005.

[MR02]   M. Molloy and B. Reed. *Graph colouring and the probabilistic method*, volume 23. Springer, 2002.

[MS12]   O. Michail and P. G. Spirakis. Unpublished work on random temporal graphs, 2012.

[MS14a]   O. Michail and P. G. Spirakis. Simple and efficient local codes for distributed stable network construction. In *Proceedings of the 33rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 76–85. ACM, 2014.

[MS14b]   O. Michail and P. G. Spirakis. Traveling salesman problems in temporal graphs. In *39th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 553–564. Springer, 2014. Also in *Theoretical Computer Science*, doi: 10.1016/j.tcs.2016.04.006, 2016.

[MV80]   S. Micali and V. V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the IEEE 21st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 17–27. IEEE, 1980.

[Orl81]   J. B. Orlin. The complexity of dynamic languages and dynamic optimization problems. In *Proceedings of the 13th annual ACM symposium on Theory of computing (STOC)*, pages 218–227. ACM, 1981.

[Orl84]   J. B. Orlin. Some problems on dynamic/periodic graphs. In W. R. Pulleybank, editor, *Progress in Combinatorial Optimization*, pages 273–293. Academic Press Canada, 1984.

[OW05]   R. O'Dell and R. Wattenhofer. Information dissemination in highly dynamic graphs. In *Proceedings of the 2005 joint workshop on Foundations of mobile computing (DIALM-POMC)*, pages 104–110, 2005.

[Pel00]   D. Peleg. Distributed computing: A locality-sensitive approach. *SIAM Monographs on discrete mathematics and applications*, 5, 2000.

[Pit87]   B. Pittel. On spreading a rumor. *SIAM Journal on Applied Mathematics*, 47[1]:213–223, 1987.

[PY93]   C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18[1]:1–11, 1993.

[Rav94]   R. Ravi. Rapid rumor ramification: Approximating the minimum broadcast time. In *Proceedings of the IEEE 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 202–213. IEEE, 1994.

[Sch02]   C. Scheideler. Models and techniques for communication in dynamic networks. In *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 27–49, 2002.

[Soi09]   A. Soifer. *The Mathematical Coloring Book: Mathematics of Coloring and the Colorful Life of its Creators*. Springer; 1st edition, 2009.

[TIR78]   S. L. Tanimoto, A. Itai, and M. Rodeh. Some matching problems for bipartite graphs. *Journal of the ACM (JACM)*, 25[4]:517–525, 1978.

[XFJ03]   B. Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14[02]:267–285, 2003.