Data Science and Engineering (2019) 4:352–366
https://doi.org/10.1007/s41019-019-00105-0

Time-Dependent Graphs: Definitions, Applications, and Algorithms

Yishu Wang[1] · **Ye Yuan**[1] · **Yuliang Ma**[1] · **Guoren Wang**[2]

Received: 1 June 2019 / Revised: 6 September 2019 / Accepted: 16 September 2019 / Published online: 25 September 2019

Abstract
A time-dependent graph is, informally speaking, a graph structure dynamically changes with time. In such graphs, the weights associated with edges dynamically change over time, that is, the edges in such graphs are activated by sequences of time-dependent elements. Many real-life scenarios can be better modeled by time-dependent graphs, such as bioinformatics networks, transportation networks, and social networks. In particular, the time-dependent graph is a very broad concept, which is reflected in the related research with many names, including temporal graphs, evolving graphs, time-varying graphs, historical graphs, and so on. Though static graphs have been extensively studied, for their time-dependent generalizations, we are still far from a complete and mature theory of models and algorithms. In this paper, we discuss the definition and topological structure of time-dependent graphs, as well as models for their relationship to dynamic systems. In addition, we review some classic problems on time-dependent graphs, e.g., route planning, social analysis, and subgraph problem (including matching and mining). We also introduce existing time-dependent systems and summarize their advantages and limitations. We try to keep the descriptions consistent as much as possible and we hope the survey can help practitioners to understand existing time-dependent techniques.

Keywords Time-dependent network · Graph data management · Network analysis · Graph system

1 Introduction

A graph is a data structure which is widely used in network modeling. Almost every scientific domains, including mathematics, computer science, chemistry, and biology, can be modeled and studied by graphs. Moreover, graphs are extensively applied in social networks, biological networks, transportation networks, distributed systems, and so on. A static graph (we use the "static graphs" to refer to classical graphs in this review to opposite it from time-dependent graphs) consists of two sets: vertices and edges. Here, each vertex represents an object. Each edge represents a relation between each pair of vertices. In practical applications, vertices and edges of graphs often contain specific information, such as labels or particular weights (such as length and cost). Generally, for example, when we model a road transportation network into a graph, each vertex represents an intersection and the associated coordinates (latitude and longitude) are a vertex weight. Each edge represents a road segment between two adjacent intersections and the distance of the edge is an edge weight.

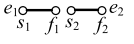
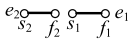
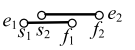
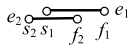
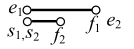
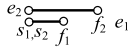
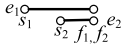
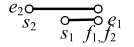
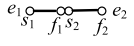
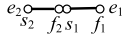
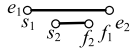
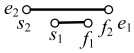
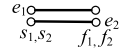
However, many real-life scenarios can be better modeled by time-dependent graphs, such as bioinformatics networks [52, 60, 63], transportation networks [25, 38, 78], social networks [56, 62, 72]. In bioinformatics networks, graphs are used to reflect the similarity and regulatory of biomolecules, such as proteins, genes, and enzymes. The connections in biological functions are not always active but change over time [28, 45]. In social networks, the topological structure of a graph represents a social relationship. When a person leaves a group or a new group is established, the topological structure needs to be updated. While time is a main reason for these changes [11, 72]. In transportation networks, there is usually a set of fixed routes on which a group of transport units moves over time [20, 61, 71, 73]. The transportation network is one of the most suitable networks that can be modeled by a time-dependent graph, where the influence of time always needs to be considered. For example, when a user takes a transfer in flight transportation networks, the

✉ Yishu Wang
yishu_w0124@163.com

[1] School of Computer Science and Engineering, Northeastern University, Shenyang, China

[2] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China

Table 1 Presentation of ALLEN's 13 time-dependent relations. Adapted from [77]

Relation code	Relation	Figure illustration	Relation code	Relation	Figure illustration
1	e_1 before e_2		8	e_2 before e_1	
2	e_1 overlaps e_2		9	e_2 overlaps e_1	
3	e_1 starts e_2		10	e_2 starts e_1	
4	e_1 finishes e_2		11	e_2 finishes e_1	
5	e_1 meets e_2		12	e_2 meets e_1	
6	e_1 contains e_2		13	e_2 contains e_1	
7	e_1 equals e_2				

departure time of the second flight must be later than the arrival time of the first flight. Generally, in such networks, the weights associated with edges dynamically change over time (time-dependency) [22]. In terms of modeling, a time-dependent graph can be thought as a special case of labeled graphs, in which labels capture some measure of time [55]. That is to say, edges in such graphs are activated by sequences of time-dependent elements.

An advantage of modeling a network as a time-dependent graph is that we can study the dynamic effect of time on the graph instead of the impact of the actual dynamics. In general, when is a graph appropriate to be modeled and analyzed as a time-dependent graph? The system needs to be modeled as a time-dependent graph only when it conforms to the time-dependent framework and involves the time scale. And if the dynamic system changes much faster than the speed of the dynamic connection, or if the edges in the graph are actively changing, then there is no need to model the dynamic system into a time-dependent graph [33]. Allen divided time-dependent relationship into 13 categories which are shown in Table 1.

Though static graphs have been extensively studied, there is still far from having a concrete set of structures and algorithm frameworks for time-dependent graphs [55]. Due to the influence of time, most query processing and mining problems are more complicated in time-dependent graphs than that in static graphs. For example, a time-dependent shortest path problem aims to find the optimal path from a source to a destination, when the starting time is selected from a user-given starting-time interval [22] instead of just finding the shortest path. It is still not clear how time affects the complexity of the optimization problem in time-dependent graphs. However, there is an evidence in [4] which shows that the concept of time can significantly increase the computational complexity. Many problems, which can be solved in a linear time or a polynomial time on static graphs, become NP-complete or NP-hard problems on time-dependent graphs, such as the connected component problem. In recent works, one important problem is to understand the complexity of classical graph problems in time-dependent graphs and proposing algorithms for them [20, 24]. Other works focus on proposing practical application problems and solving them [44, 61].

In this paper, we focus on prior works on time-dependent graphs. The rest of the paper is organized as follows. Section 2 introduces the time-dependent graph framework including definitions and models. Section 3 introduces route planning problems and existing algorithms on time-dependent graphs. Section 4 shows some special time-dependent queries in social analysis. Section 5 provides two kinds of time-dependent subgraph problems, including subgraph matching and mining. Section 6 shows existing time-dependent graph systems and summarizes their advantages and limitations. Section 7 concludes this paper.

2 Time-Dependent Graph Framework

2.1 Definition

The time-dependent graphs can be fundamentally divided into two types according to the time type of the active an edge: time instances and time intervals as illustrated in Fig. 1. In Fig. 1a, the weight on an edge represents a sequence of time instances. In this case, the interaction between each pair of vertices is activated at a sequence of time instances. The time-dependent graph is represented as a triple $G = (V, E, T)$. V is a set of vertices and $E \subseteq V \times V$ is a set of edges, and for each $e \in E$, there is a non-empty set of time $T_e = \langle t_1, \ldots, t_n \rangle$ as a weight. Such time-dependent graphs are commonly used to represent instant messaging networks where the duration does not need to be considered, such as mail networks, telephone networks, and information networks. For example, a public transportation network can

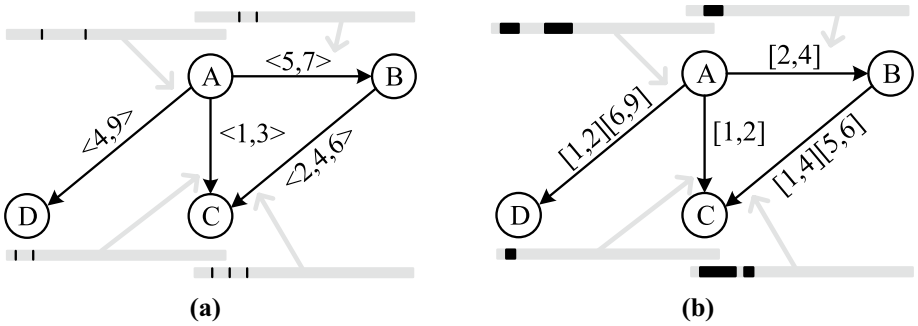


Fig. 1 Two fundamental time-dependent graphs. Here, **a** shows a time-dependent graph on which the weight on each edge is a sequence of time instances. These instances are marked by black lines in the timelines (gray bars). **b** Shows a time-dependent graph on which the weight on each edge is a set of time intervals. These intervals are marked by black bars in the timelines (gray bars). Note that these time intervals can be either continuous or discrete

be represented as this type of time-dependent graph. A vehicle on the network is operated according to a specific set of discrete times on the timetable.

In the other case (Fig. 1b), the edges are activated in a sequence of time intervals. The weight on each edge e represents as $T_e = \{[t_1, t_2], \ldots, [t_{n-1}, t_n]\}$, where a period of activity is beginning at t_1 and ending at t_2. In such graphs, the duration is very important. This type of time-dependent graph can naturally model the systems whose edges change in time intervals, such as proximity networks, seasonal food webs, and infrastructural systems [33]. For example, a road network can be modeled as such time-dependent graph because of the traffic control (i.e., the limit line) and traffic congestion (i.e., the morning–evening rush hours).

2.2 Data Modeling

In the existing works, there are several ways of modeling a time-dependent graph with a sequence of time instances on each edge. These models are mainly divided into two categories: one is to model discrete time-dependent graphs and the other is to model continuous time-dependent graphs. Next, we introduce the two models separately.

2.2.1 Models for Discrete Time-Dependent Graphs

An edge on a discrete time-dependent graph is activated by disjoint time points. Formally, there are several ways of modeling discrete time-dependent graphs.

One is to build labels for the edges which has well-defined attributes [15, 35, 73]. A time-dependent graph with a label $\lambda : E \to 2^{\mathbb{N}}$ is denoted by $G = (V, E, \lambda(G))$. The label set of $\lambda(G)$ is a collection of natural numbers on each edge, which can be denoted by $\lambda(E)$ where $|\lambda| = \sum_{e \in E} |\lambda(e)|$. This modeling method is widely available and can build unified searching schemes for time-dependent graphs.

Another is to build snapshots for the time-dependent graph, that is, the time-dependent graph is considered as an ordered pair of disjoint sets [9, 49, 80]. A time-dependent graph is denoted by $G = (V, A)$, where A is a set of time-dependent edges. A function $A(t) = \{e : (e, t) \in A\}$ contains all edges in G at time t which is also called the tth instance of G. Accordingly, a snapshot of G at t is denoted by $G(t) = (V, A(t))$. In this case, a time-dependant graph can be considered as a sequence of static graphs $G = \{G_1, G_2, \ldots, G_t\}$. This modeling method is most commonly used for modeling discrete time-dependent graphs, which is suitable for the time-dependent graph with a specific time structure, especially in real-time networks.

The third is to transfer the time-dependent graph into a static graph by building copies of vertices [35, 75, 76]. The reason for doing this is mainly because the technology tends to be perfect on static graph. If a time-dependent graph can be transformed into a static graph without losing any time-dependent information, it can provide a good foundation for future studies on time-dependent graphs. [75] provides a transfer method which has two steps to transform a time-dependent graph $G = (V, E)$ into a static graph $\hat{G} = (\hat{V}, \hat{E})$ as illustrated in Fig. 2. Firstly, create copies of each vertex $v \in V$ in $\hat{V}$, where $\hat{V}$ composed of

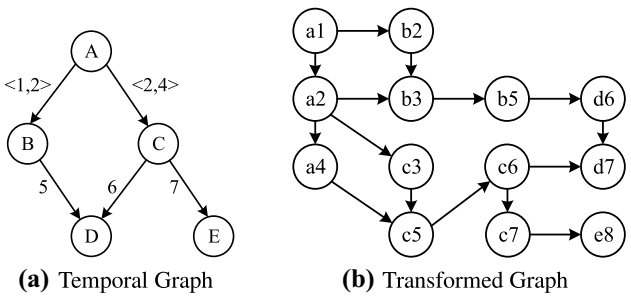


Fig. 2 Graph transformation from a time-dependent graph to a static graph

two parts $T_{in}(u,v) = \{t + \lambda : (u,v,t,\lambda) \in \prod(u,v)\}$ and $T_{out}(v,u) = \{t : (u,v,t,\lambda) \in \prod(v,u)\}$. Here, $T_{in}(u,v)$ maintains all vertices which are in-neighbors of v. Symmetrically, $T_{out}(u,v)$ maintains all vertices which are out-neighbors of v. Secondly, create edges from (v,t_{in}) to (v,t_{out}) for each vertex $v \in V$ by order vertex in $\hat{V}_{in}(v)$ according to time instances order, then create a directed edge from each v to its copes. This modeling method can transplant algorithms of static graphs to problems of time-dependant graphs. But the disadvantage is obvious that building multiple copies of the vertices increase the scale of graphs. And [75] also shows that the results obtain directly from the time-dependent graph are quite different from the results obtained from the converted static graph for queries that need to consider time order.

2.2.2 Models for Continuous Time-Dependent Graphs

There are two drawbacks of the discrete time model. First, this model cannot represent the state of the graph between two discrete time points, which might yield inaccurate results. Second, the memory and processing requirements are high. A more precise way to describe a time-dependent network is to use the continuous time-dependent function. For a continuous time-dependent graph whose edges are activated in a sequence of time intervals, there are two modeling methods.

One is to define an index function, called presence function, to determine whether a pair of vertices is connected at a given time interval [65]. If vertex u and v is connected at time t, then the presence function $f(u,v,t) = 1$. Otherwise, $f(u,v,t) = 0$. This modeling method applies to time-dependent graphs where the time intervals are discrete and disjoint and the edges are only active and inactive.

The other is to treat a time-dependent graph as a flow-dependent graph which means the weight it takes for flow to traverse an edge is different in different time intervals [19, 22, 24]. For $e \in E$, the flow function $f_e : [0,T) \to \mathbb{R}^+$, where $f_e(t)$ defines the rate of flow entering edge e at time t. For example, a road network can be modeled as a continuous time-dependent graph, where a non-negative travel time function τ_e determines the time it takes for flow to traverse an edge e [42]. The flow arrives at the end node of e at time $t + \tau_e(f_e(t))$. Accordingly, this modeling method applies to time-dependent graphs where the time intervals are continuous and the functions on edges change over time.

2.3 Topological Structure

In static graphs, the topological structure can be characterized by measures. Some measures can be applied directly to time-dependent graphs, including adjacent point, degree, etc. For example, in a time-dependent graph, the degree calculation is similar to that in a static graph. It only needs to calculate the number of active edges connected to vertices over a period of time. While other measures need rethinking and redefining to take time into account on time-dependent graphs.

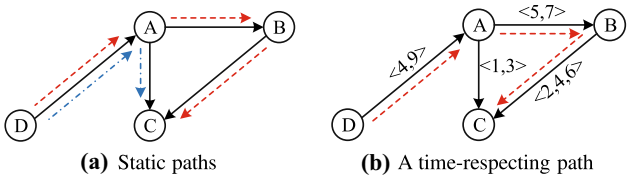


Fig. 3 Paths from D to C in a static graph and a time-dependent graph

2.3.1 Time-Respecting Paths

In the static graph, a path is simply a sequence of edges from origin vertex u to destination vertex v, such as $p_1 = \langle D, A, B, C\rangle$ (red dotted line) and $p_2 = \langle D, A, C\rangle$ (blue dotted line) in Fig. 3a. However, a time-respecting path [32, 39] has to be redefined to take time into account. For example, as shown in Fig. 3b, D and C are connected only through path $p = \langle (D,A,4),(A,B,5),(B,C,6)\rangle$ (red dotted line), assuming edge delay is 1. Accordingly, the time-respecting path is a sequence of edges that follow a time order. If there are paths from A to B and B to C, it does not mean that there is a time-respecting path between A and C. Consequently, as seen in the example above, a time-respecting path from A to C is existing only if the contact between A and B takes place before the contact between B and C.

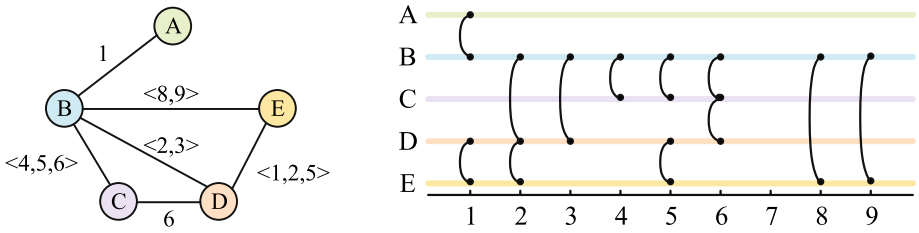
In the static graph, the most basic problem associated with the path is the shortest path problem that computes the shortest distance between two reachable vertices. In [16], the time-dependent shortest path problem is refined into three types:

Earliest Arrival Path Problem. The earliest arrival path problem asks for the corresponding route with the earliest arrival time at B from A at departure time t_a.

Latest Departure Path Problem. The latest departure path problem asks for the corresponding route with the latest departure time from A and arrives at B no later than t_b.

Shortest Duration Path Problem. The shortest duration path problem asks for the corresponding route with shortest duration time that departs from A no sooner than t_a and arrives B no later than t_b.

Due to the complexity of time-dependent information, each of the above problems can not be solved in a greedy strategy which is often used to compute the shortest path problems in a static graph. Cooke et al. [16] proposed a



(a) A time-dependent graph **(b)** Time-dependent reachability

Fig. 4 A time-dependent undirected graph and the time-dependent reachability

modified version of Bellman's iteration scheme [3] to compute the shortest path with time-dependent information. Afterward, many algorithms [59, 70, 73, 75] are proposed to solve Cooke et al.'s three problems on the time-dependent graph. Since the path problem is the most foundational problem on graphs, most of these algorithms are discussed in Sect. 3.1.

2.3.2 Connectivity, Components, and Menger's Theorem

Connectivity is a fundamental concept for both static networks and time-dependent networks. We said a graph is connectivity if there are paths between every pair of vertices. A connected component is defined as a set of vertices with paths between each pair of them. In directed static graphs, connected components can be classified into weakly and strongly connected components. In particular, a graph is said to be strongly connected, if there is a path from i to j and a path from j to i, for each pair of vertices i and j. Correspondingly, a graph is said to be weakly connected, if the corresponding undirected graph which is obtained by removing all directions in the edges is a connected graph, i.e., replace all directed edges with undirected edges. In time-dependent graphs, the order of time naturally introduces a directionality of the graph. For instance, in the time-dependent graph G (shown as Fig. 4a), there exists a time-respecting path between vertex A and vertex E (i.e., edge e_{AB} at time 1 and edge e_{BE} at time 8 in Fig. 4b), but there is no path between E and A. Hence, the connectivity of the time-dependent graph is more similar to that of the static directed graph [58]. However, the concepts of weakly and strongly connected can not be generalized for time-dependent graphs if we only hold time-respecting paths instead of paths. Nicosia et al. proposed the definition of strong and weak connectivity on time-dependent graphs (called time-varying graphs in their paper [58]) as follows.

Definition 1 (*Strong Connectedness* [58]) Two nodes i and j of a time-varying graph are strongly connected, if i is temporally connected to j and also j is temporally connected to i.

Definition 2 (*Weak Connectedness* [58]) Two nodes i and j of a time-varying graph are weakly connected if i is temporally connected to j and also j is temporally connected to i in the underlying undirected time-varying graph.

According to above two definitions, the strongly connected component and the weakly connected component can be easily defined, that is, the vertices sets where each pair of vertices fulfills the criteria. And Nicosia et al. also showed that finding the strongly connected components of a time-dependent graph is equivalent to finding the maximal cliques of an affine graph that is an undirected static graph with the same vertices as the time-dependent graph.

Menger's theorem [53] is one of the most basic theorems in the theory of graph connectivity. It states that the maximum number of node-disjoint s-v paths is equal to the minimum number of nodes that must be removed in order to separate s from v [8]. However, Kempe et al. [39] proved that the Menger's theorem of static graphs does not apply to time-dependent graphs. In particular, they also proved that there is no natural analogue of Menger's Theorem for the single-label time-dependent graph and it is **NP**-hard to compute the number of node-disjoint time-respecting paths. Over Kempe et al.'s work, Mertzios et al. [54] provided a natural time-dependent (called temporal in their paper) analogue of Menger's theorem for all (both single-label and multi-label) time-dependent graphs.

Theorem 1 (Menger's Temporal Analogue [54]) *Take any temporal graph $\lambda(G)$, where $G = (V, E)$, with two distinguished nodes s and v. The maximum number of out-disjoint journeys from s to v is equal to the minimum number of node departure times needed to separate s from v.*

By symmetry, they have that the maximum number of in-disjoint journeys from s to v is equal to the minimum number of node arrival times needed to separate s from v.

2.3.3 Spanning Tree

The spanning tree is an important concept related to paths in static graphs. In particular, the minimum spanning tree problem refers to the minimal connected subgraph generated

for the original graph. The connected subgraph contains all the vertices of the original graph, whose number of edges connecting the subgraphs is the smallest. The minimum spanning tree problem can be solved in polynomial time and has widely used in graphs. Many complex queries are based on the minimum spanning tree problem [5, 31, 46, 68], and many efficient graph algorithms are based on building a minimum spanning tree, such as min-cut max-flow algorithm.[1]

Gunturi et al. [27] proposed a related concept in time-dependent graph called time-sub-interval minimum spanning tree (TSMST). TSMST refers to a collection of minimum spanning trees in an interval with minimum total cost. Further in [35], Huang et al. defined two kinds of minimum spanning trees (MST_s) in a time-dependent graph: (1) $\mathcal{MST}_a$ which is the MST_s with earliest arrival times and (2) $\mathcal{MST}_w$ which is the MST_s with the smallest total weight. Huang et al. proved that $\mathcal{MST}_a$ can be computed in linear time, but $\mathcal{MST}_w$ is a **MAX-SNP** hard. Consequently, they transformed the $\mathcal{MST}_w$ problem to the minimum Directed Steiner Tree problem.

3 Time-Dependent Route Planning

The time-dependent route planning is an important problem with application in routing on road networks [22, 24], travel planning on public transportation networks [14, 73], vehicle dispatching on vehicle networks [37, 50], as well as in some robotic and navigation systems [12, 43]. To the best of our knowledge, the time-dependent shortest path problem, as the most basic route planning problem, is first proposed by Cooke and Halsey in [16].

Before we discuss the algorithms of routing problems on the time-dependent graph, we first introduce an important property of the time-dependent graph, namely first-in-first-out (FIFO) property. The FIFO property means that, on every link, an earlier departure will lead to an earlier arrival and a later departure will result in a later arrival. It renders that every edge has a non-decreasing arrival time function in a time-dependent network. We called a network is a FIFO network if every link in the network has the FIFO property. The facts of the FIFO property can be referred in [44]. Notice that the FIFO property is a special property on time-dependent graphs, but not all time-dependent graphs are FIFO networks. For example, a source s and a destination d are reachable by two vehicles, one is faster and the other is slower. Assume that a traveler A chooses the slower vehicle to depart in advance from s, then he may arrive later than the traveler B who chooses to depart late but takes the faster vehicle. Obviously, this travel does not satisfy the FIFO property. The other example of not satisfying the FIFO property is a path that allows waiting time at vertices. The FIFO property is relevant to time-dependent routing and many routing problems are based on the FIFO attribute.

In this section, we first discuss the single-criteria time-dependent routing problem which only focuses on finding minimum travel time in time-dependent graphs. Then, we introduce the multi-criteria time-dependent routing problem which optimizes the travel time and multiple time-independent costs. Since the methods in this section are all used to solve the route planning problem on the time-dependent graph in principle, we try to keep the descriptions consistent as much as possible.

3.1 Single-Criteria Route Planning

As already noted, broadly speaking, time-dependent graphs can be divided into discrete and continuous. So, we introduce single-criteria routing algorithms on the two types of time-dependent graphs, respectively.

3.1.1 Models Based on Discrete Travel Time

In the time-dependent network with discrete travel time, certain segments can only be traversed at specific discrete time instances based on a timetable. Therefore, the cost of an edge, varying as a travel time function, represents a succinctly periodic, discrete, piecewise linear (PWL) function. The public transportation network is representative of discrete time-dependent networks. According to real traffic conditions, it is always necessary to wait for a vehicle at a station. And when transfer from one vehicle to another, the vehicle schedule must be strictly followed.

As we maintained before, Cooke and Halsey [16] proposed the most basic route planning problem. Cooke and Halsey proved that these queries could be solved with a modified version of Dijkstra's algorithm. However, it does not scale well with the size of the graph and several techniques, such as indexing have therefore been proposed to improve efficiency. Pyrga et al. [61] proposed realistic time-expanded and time-dependent models and extended their methods for a series of realistic requirements of time-dependent shortest path (TDSP) problem on discrete time-dependent graphs. Chabini [14] extended the A* algorithm to compute the minimum travel time path for one as well as for multiple departure times. They improved a lower bound on minimum travel time which is used to design an effective adaptive A*-based algorithm. To speed up the A* algorithm, Nannicini et al. [57] and Demiryurek et al. [21] presented methods based on the bidirectional A* search algorithm. Notice that the time-dependent paths must be in time order. But we cannot know the arrival time in advance during backward search.

[1] https://brilliant.org/wiki/spanning-trees/.

Hence, both Nannicini et al. and Demiryurek et al. run the backward search by the idea of lower bound function. Wu et al. [75] proposed efficient algorithms, named as one-pass algorithms. The one-pass algorithm is based on Dijkstra's algorithm and solves the route planning queries by enumeration. For each vertex v, the one-pass algorithm builds a store list $L(v) = (s[v], a[v])$, where $s[v]$ is the departure time from source x to v, and $a[v]$ is the arrival time at v. Then, it updates the $L(v)$ until finding the earliest arrival time from the source vertex x to v at different starting time. In order to provide an alternative approach, they proposed a method of transferring a time-dependent graph into a static graph. Notice that Wu et al. proved that the results obtained directly from the time-dependent graph are quite different from the results obtained from the converted static graph. This calls the need for studying the time-dependent problem directly on time-dependent graphs. After that Wang et al. [73] presented an efficient indexing technique, named as *Timetable Labeling* (TTL), for time-dependent routing problem in public transportation networks. The TTL index is defined based on the concept of canonical paths. It builds two label sets $L_{\text{in}}(v)$ and $L_{\text{out}}(v)$ for each node v in the time-dependent graph, which contains the information about the canonical paths between v and the nodes that rank higher than v. Wang et al. showed that the TTL index enables to support three common route planning queries (described in Sect. 2.3.1).

3.1.2 Models Based on Continuous Travel Time

Route planning on a continuous time-dependent network is drastically different from that on a discrete time-dependent network. Routing in continuous time-dependent networks is to calculate the optimal path from a source s to a destination d on road network without changing vehicle. It is a key component in road network, self-driving technology, navigation systems as well as in some robotics. In a continuous time-dependent network, the cost of the edge varies as a travel time function which is always a continuous and PWL function. Notice that we are not simply calculating the optimal path with a certain departure time, which is easily calculated by Dijkstra's algorithm, but calculating the minimum travel time under certain conditions, such as the expected arrival time or the departure time interval.

Ordal and Rom [59] presented a Ford-type algorithm to solve the TDSP problem when no constraints are imposed on waiting times at the nodes. On the basis of Ordal and Rom's work, Dehne et al. [18] presented an improved method that can solve the TDSP problem in FIFO networks. They proved that the algorithm can run in polynomial time, while other methods require super-polynomial time. And they also presented an approximation method with possibly super-polynomial size output in time $O(\frac{\Delta}{\epsilon}(|E| + |V|\log|V|))$. Afterward, Foschini et al. [24] presented a method that can solve TDSP problem in super-polynomial time. The main idea of their method is to calculate primitive and minimized breakpoints and build acyclic layered graph representation of the time-dependent graph. They presented an output-sensitive algorithm whose running time depends on the number of breakpoints at the nodes and edges in the time-dependent graph. In order to speed up the technique for TDSP problem, Foschini et al. proposed $(1 + \epsilon)$-approximation schemes to compute the arrival time functions in time $O(K\frac{1}{\epsilon}\log(\frac{\text{MaxTravelTime}}{\text{MinTravelTime}}))$. Ding et al. [22] presented a query as $\text{LTT}(v_s, v_e, T)$ to ask the minimum-travel-time path from a source s to a destination d with the best departure time selected in a given time interval. They showed that their approach, named as *Two-Step-LTT*, adapts both on FIFO and NON-FIFO time-dependent networks. The first step is *Dijkstra-based Time-refinement* in which they compute the earliest arrival time function for each node. The second step is *Fast Path-selection* in which they compute the optimal path. To adapt Two-Step-LTT in non-FIFO networks, they showed how to transform a non-FIFO time-dependent graph into a FIFO one. They also proved the time complexity of their algorithm is $O((n\log n + m)\alpha(T))$.

3.2 Multi-criteria Route Planning

Multi-criteria routing problem is more complicated than the single-criteria shortest path problem. Given two paths p_i and p_j, we say that p_i dominates p_j if and only if there are no criteria for which p_i has a worse value than p_j, with at least on strict equality. A path is called Pareto-optimal if it is not dominated by any other path. Therefore, multi-criteria routing problems aim to find Pareto-optimal path sets from a source s to a destination d. The most common multi-criteria (or bi-criteria) time-dependent routing optimization problems consider the optimization between travel time and multiple (or a) time-independent costs (expressed as the weights on each edge) including an approximation of energy consumption, distance, tolls, or other penalties [1].

Pyrga et al. [61] considered a bi-criteria search to optimize the travel time and the number of transfers in discrete time-dependent graphs. In order to solve the bi-criteria optimization problem, they set a bounded number of transfers which constraints that the total number of transfers is not greater than k. Based on Pyrga et al.'s work, Khoa et al. [40] also optimized the arrival time and the number of transfers, but they allowed walking during a transfer in a bus network. They presented three speed up techniques to accelerate the multi-criteria searching algorithm. Disser et al. [23] presented a multi-criteria searching algorithm by introducing the reliability of the transfer and gave the probability of taking the train. Different from Pyrga et al.'s work, Disser et al. focused on a many-source shortest path on train networks and they considered foot-path and special transfer rules

during transfers. In the multi-criteria searching algorithm, they used multi-dimensional labels to record all the promising path that the node can reach and found the shortest path based on the Dijkstra's algorithm. And they proposed four speed up techniques to improve the performance of the searching algorithm.

Batz and Sanders [1] considered the optimization of travel time and costs in continuous time-dependent graphs. They proved that this problem is **NP**-hard and it can be computed by Dijkstra-like algorithms as prefix optimality is violated. Therefore, they presented a multi-label A* searching algorithm. However, the efficiency of the algorithm is too low in practical applications. They used the generalized heuristic time-dependent contraction hierarchies (TCHs) during preprocessing and contracted the least important nodes. The heuristic minimum cost queries with TCHs are divided into two phases. Firstly, it is a bidirectional upward search which only takes little running time. Then, it only uses the edge touched in bidirectional upward search to do a downward search. Experiments show that heuristic TCHs are very fast.

4 Time-Dependent Social Analysis

In real scenarios, time is a common and necessary dimension on online social networks (OSNs). For example, when a user logs in or logs out of an account, the background database will have a timestamp to record such operations. Many complex dynamic time-dependent interactions can be abstracted into time-dependent information. Compared with traditional static social queries that only reflect the relationship between users, time-dependent social queries can distinguish between new and old, active and inactive relationships, and so on. Additionally, query and mining problems on time-dependent social networks are more focused on the evolution over time [29, 36]. For example, the time-dependent shortest path queries in a social network can discover how close between two given users have been and how the closeness has evolved over time. Consequently, queries on time-series social networks are more complicated, and due to the concept of the time dimension, there are many query problems that are unique to time-dependent social networks.

Most of the works on time-dependent social networks are based on snapshots and time windows. Wang et al. [74] proposed a time-dependent social network advertising and they designed a learning algorithm to obtain the optimal policies between users and advertisers. Wang et al. [72] demonstrated that the influence of a user on another depends on their interactions in previous time windows. Therefore, they presented a dynamic factor graph (DFG) model to tackle the dynamic social influence analysis with time information. They modeled networks as a sequence of time-dependent graph snapshots (which called factor graph in [72]), and each graph snapshot belongs to a time window. In particular, each graph snapshot depends on the factor graph in the previous time window. This technique is optimal to time-dependent social network, but it has huge storage overhead. To efficient the queries, Huo and Tsotras [36] presented the temporal partitioning. They divided the time instance in the time-dependent graph into multiple partitions, and each partition has a time window with a fixed length.

Furthermore, Chen et al. [15] introduced primitive queries on time-dependent social networks. They divided the primitive queries into three aspects, which are aimed to find and check the user's online status, the relationships between users, and the status of the user participating in the activity during a time interval $[t_s, t_e]$. Based on three primitive queries, they also presented three special queries that are shown as follows.

Friends of Interesting Activities (FIA) query The FIA query aims to find out user's friends who are interested in something or people over a period of time.

Users of Time Filter (UTF) query The UTF query is used to find a user who is active within a certain time threshold, and the user's friends are interested in a given event. UTF queries can be used in the social network advertising and social influence analysis in practice.

Group of Users with Relationship Duration (GURD) query The GURD query is used to find a group of users that satisfy a certain time condition, and the average intimacy between users satisfies a given value within a certain period of time.

To solve three problems on time-dependent social networks, Chen et al. designed two tree structures: Temporal Users and Relationships tree (TUR-tree) and Temporal Users and Activities tree (TUA-tree). The TUR-tree is extended by MVB tree [2] to index user relationships. The TUR-tree generates keys by string concatenating. For a user v_i identified by uid_i, the key is expressed as $0|uid_i$, and the relationship between uid_i and uid_j is expressed as $1|uid_i|uid_j$. The TUA-tree is a hybrid structure of B^+-tree and Bloom Filter to represent the communication between user identifiers, time information, and keywords. For a TUA-tree, the entry of leaf node is formatted as $< key, ptr >$, where ptr points to an activity, and key is concatenation of user identifier and time associated with the activity the user is participating in. The TUA-tree splits, merges, and redistributes nodes by key.

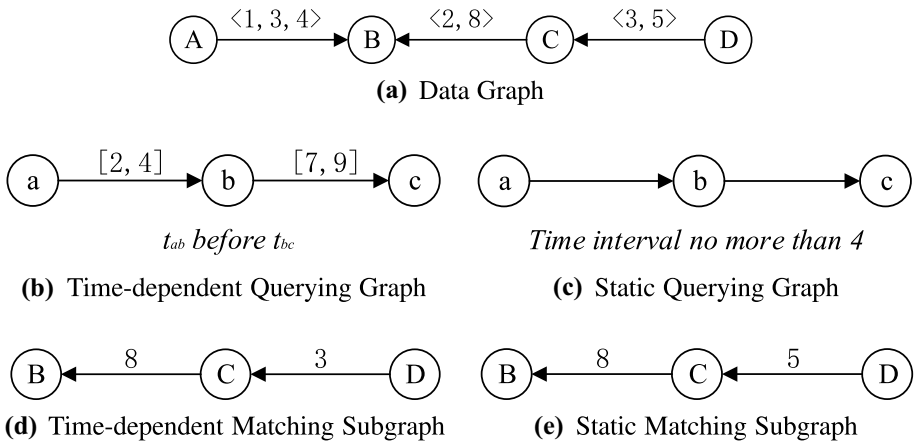


Fig. 5 Two kinds of graph pattern matching problems on the time-dependent graph. **a** Shows a data graph. **d** Shows the query result of TGP query from (**b**). The edge (D, C) only can be activated at time $t = 3$, thus there is only one matching result. **c** Shows the query result of SGP query from **c** with a limitation that the interaction occurring time interval $d \leq 4$. The edge (D, C) only can be activated at time $t = 5$, because $t_{CB} - t_{DC} = 3 < d$ at $t = 5$ and $t_{CB} - t_{DC} = 5 > d$ at $t = 3$. Only when $t = 5$ (D, C) contents the time constraint

5 Time-Dependent Subgraph Problems

5.1 Time-Dependent Subgraph Matching

The graph pattern matching is a fundamental problem in the graph data mining and has been applied in many fields. The graph pattern matching (a.k.a. subgraph isomorphism) problem is known to be NP-complete even in a static graph [30]. The existing graph pattern matching problems can be roughly divided into two types according to the pattern of query graph on time-dependent graphs.

Querying static graph pattern (SGP) in time-dependent graphs The querying static graph pattern problem is to find isomorphism subgraphs, when the query graph is a static graph (see Fig. 5c) with some time constraints (i.e., the interaction occurring time interval or the relation order) and the data graph is a time-dependent graph (see Fig. 5a).

Querying time-dependent graph pattern (TGP) in time-dependent graphs The querying time-dependent graph pattern problem is to find isomorphism subgraphs, when the query graph is a time-dependent graph (see Fig. 5b) and the data graph is a time-dependent graph (see Fig. 5a).

For the same time-dependent graph, the difference in the query graph will result in different query results, as shown in Fig. 5.

5.1.1 Querying Static Graph Pattern

Redmond and Cunningham [64] proposed three methods for subgraph matching algorithms by considering time-dependent and topological information at different stages.

Time before Topology (Ti-To) The Ti-To first abstracts all possible time-respecting subgraphs as a candidate subgraph set from the data graph by breadth-first algorithm on edges. Then, find extracted subgraphs that satisfy the conditions of the query graph from the candidate set by the VF2 [17] subgraph isomorphism algorithm. Experiments show that the candidate set may have a large overlap and time consuming.

Topology before Time (To-Ti) The To-Ti performs the VF2 subgraph isomorphism algorithm on the entire data graph and build a candidate subgraph set. At this stage, it does not pay attention to any time-dependent information. Then, To-Ti checks the time-dependent information between the query graph and the candidate subgraph set and retains the matching result.

Time and Topology Together (Ti&To) The Ti&To considers subgraph isomorphism and time constraints at same time. The isomorphic processing is described by a state space, each state s in the state space describes a local map. In a state s, the Ti&To first determines whether a candidate pair, consisting of vertices in query graph and data graph, is isomorphic and then Ti&To determines whether the candidate pair contents the time constraint. If the candidate pair satisfies both conditions, the neighbor vertices of the candidate pair are expanded downward to become new candidate pair. Loop until all the subgraphs that satisfy the conditions are found.

To solve the SGP problem with time-dependent relation order, a Hasse diagram based algorithm is proposed by Sun et al. [69]. They used Hasse diagram to express the time-dependent partial order of the edges in the query graph. Each edge corresponds to a Hasse node, and a time-dependent relationship in the query graph is represented as a directed

Hasse edge. Then, they built the Hasse-cache structure to implement the continuous time-dependent subgraph query algorithm. The algorithm uses the probability of the time-dependent graph to reduce intermediate results, while achieving topology matching and time relationship verification. Compared with searching the entire query graph, the algorithm searches whether the time-dependent relationships of the data edges match the relationship of query edges and improves the performance of the algorithm by using the Hasse-cache structure.

5.1.2 Querying Time-Dependent Graph Pattern

TCGPM-V [77] is a pattern matching method based on vertex matching. Firstly, it uses the depth-first search tree to find all matching vertex sets and then enumerates all possible time-dependent subgraphs until finding the prospective time-dependent subgraph. Unlike TCGPM-V, TCGPM-E [77] is a pattern matching method based on edge matching. TCGPM-E first selects the edge with the smallest calculation result in the query graph according to the sorting function. Then, calculate the maximum number of edges in the query graph that can be linked around by the edge. Next, find an edge set in which all edges match with this edge in the data graph and decompose the data graph into multiple subgraphs. Each subgraph contains an edge in the edge set and the number of subgraphs is equal to the size of the edge set. Finally, subgraph isomorphism is performed on each subgraph, and the prospective time-dependent subgraph is enumerated.

In addition to the basic time-dependent graph pattern query problem, there are some more complex isomorphic problems exist in the time-dependent graph. Semertzidis and Pitoura [66] presented a durable graph pattern queries that aim to find persistent matches (the top-k longest period of time) of input patterns in node-labeled time-dependent graph. The evolution of graph is expressed as a sequence of snapshots which is corresponding to graphs with different time instances. A straightforward algorithm for the top-k most durable graph pattern queries is to perform a subgraph isomorphism algorithm on each snapshot then aggregates the results. But the straightforward algorithm does not adapt to the large-scale time-dependent graphs. They merged the snapshots into a labeled version graph (LVG) which records the time interval of nodes, edges, and labels as lifespan. Then, they calculated and filtered the candidate set by neighborhood and path time indexes based on Bloom filters [6], and then they presented an algorithm to find the top-k most durable graph. The algorithm first checks if there are isomorphic matches of the query graph in the candidate sets and deletes the candidates without an isomorphic match. Then the algorithm checks if the lifespan of edges and vertices in the candidate graph is conformed to the lifespan of the query graph. The top-k most durable graphs are gotten by ordering the duration of the results.

5.2 Time-Dependent Subgraph Mining

Subgraph discovery and analysis problems are widely used in static graphs. In practice, subgraph mining is a very broad concept and essentially can be seen as a clustering problem. The problem of finding a set of vertices with certain features and closely interacting with each other in graphs is called a subgraph mining problem. A time-dependent graph mining problem seeks to mine subgraph structures with time-dependent information, such as relaxed moving object clusters [47], heavy subgraphs [7], diversified subgraphs [79], and dense subgraphs [49].

The moving object cluster is a loosely defined and general task to find a group of moving objects that are traveling together sporadically [47], which can be widely used in the studies of animal behaviors, routes planning, and vehicle control. There are two elements which are geographically close to each other when moving together in a certain time interval. To record the moving object cluster, *swarm* was proposed to contain a number of objects who are in the same cluster in a time interval. For a group of moving objects as O and a set of timestamps as T, a swarm is a pair (O, T) containing at least $\min_o$ individuals who are in the same cluster at least $\min_t$ timestamp snapshots, that is, $|O| \geqslant \min_o$ and $|T| \geqslant \min_t$. The search space of moving an object cluster is very huge, because the size of all the possible combinations is exponential (i.e., $2^{O_{DB}} \times 2^{O_{TB}}$). And since the discovery of the moving object cluster is based on multiple timestamps, it has a polynomial solution. Therefore, the efficient method called ObjectGrowth [47] was proposed to remove redundant and hopeless candidates through two pruning rules.

The heaviest dynamic subgraph (HDS) seeks to find the highest-scoring time-dependent subgraph in a weighted dynamic network whose edge weights evolve over time. The score of a subgraph is defined by calculating the sum of the edge weights and the score has to be maximized over all possible subgraphs and all possible sub-intervals. For each time snapshot, if the edge exists, the score gets 1 or the score gets -1. Given a time-dependent graph with T timestamps, there are $T \cdot (T+1)/2$ time intervals to consider. Bogdanov et al. [7] proved that the HDS problem is NP-hard even if the score is 1 or -1 and they gave a filter-and-verify algorithm to solve the HDS problem, named as MEDEM. To prune the irrelevant sub-interval space and quickly verifying candidate sub-intervals, MEDEM calculates the upper bound of the solution. A simple upper bound can be obtained by summing all the positive edges. And a tighter upper bound is calculated by score of a connected component of positive edges P and the lowest score among negative edges N. The tighter upper bound is calculated as follows:

$$\sum \max(0, P + N) - \min(N) \geqslant \text{score}.$$

To avoid the quadratic enumeration, MEDEM gives an efficient and effective group filtering phase with time complexity in $O(t \cdot \log^2(t) \cdot |E|)$. MEDEM is based on a filter-and-verification framework, but it not adapted to the time-dependent graph with large number of vertices, edges, and timestamps. Because there are a large number of time intervals that need to be filtered and verified, Ma et al. [49] proposed a highly efficient data-driven approach named as FIDES. They showed that all the edges evolve in a convergent manner and defined a cohesive density curve to find the dense subgraph. Accordingly, the dense subgraphs only exist in time intervals in which the cohesive density curve has a local maximum. They also proved that finding dense graphs problem is equivalent to the net worth maximization problem, a variant of the Prize Collecting Steiner Tree problem. FIDES first computes k time intervals and then finds and returns dense subgraphs with the largest possible cohesive density. The experiments show that both quality of the dense subgraphs and the running time of the FIDES are better than MEDEM.

Since it is important to find dense subgraph patterns with close vertices interacting, many definitions of dense subgraph patterns have been proposed, such k-core, k-truss, γ -dense subgraph, and γ-quasi-clique [79] . Most dense subgraph pattern problems are NP-hard problems. A diversified subgraph is a dense graph based on the definition of γ-quasi-clique to calculate the γ-quasi-clique during a time interval in a time-dependent graph.

Definition 3 (γ-*Quasi-Clique* [79]) Given a time-dependent graph $G = (V, E)$ and a parameter γ, for all $v \in V$ and $t \in I$, we say that G is a γ-quasi-clique during the time interval I iff

$$d_v(t) \geqslant \gamma \cdot (|V| - 1).$$

According to Difinition 3, a diverseifed time-dependent graph mining problem is to find k dense time-pendent subgraphs in a time-dependent graph, that the k is large and diversified (i.e., maximizing the coverage). Here the coverage set of a time-dependent graph G' is denoted as $C(G') = \{(v, t) \mid v \in V', t \in I'\}$, whose size is $|C(G')| = |V'| \cdot |I'|$. For finding qualified patterns, effective pruning rules are required. Yang et al. presented five pruning rules for time-dependent subgraph mining, to prune the vertices with low degree and short duration, the vertices far away form selected vertex, the snapshot graphs over bound, the vertices over bound, and the remaining snapshot graphs with a consecutive interval, respectively.

[2] http://neo4j.com/.

[3] http://thinkaurelius.github.io/titan/.

6 Time-Dependent Graph Systems

At present, a large number of graph management systems have been proposed, including Neo4j[2] and Titan[3] as well as large graph processing frameworks such as Pregel [51], GraphLab [48], and GraphX [26]. But these graph systems are all designed for static graphs and they can be extended to process time-dependent graphs.

6.1 Snapshot-Based System

Existing works of the time-dependent graph system are mainly based on snapshot-based approaches. These systems use snapshot-based approaches to solve problems on time-dependent graphs, so they are called snapshot-based approach systems. The snapshot-based approach is to convert the time-dependent graph into a sequence of static graphs over time and each static graph is called a snapshot. The snapshot-based approach faces a huge challenge of generating a large number of snapshots even if there is only one time-dependent graph, which consumes a lot of storage space and query time. Therefore, many corresponding techniques have been proposed for efficient snapshot processing.

Cattuto et al. [13] proposed a time-dependent social network management system with a multi-layer index structure based on Neo4j. They used wearable sensors to collect time-dependent data and built a time-resolved behavioral social network. Then, they defined a frame to contain the status of the social network during each time interval. For each frame, they built a proximity graph whose nodes represent individuals and edges represent proximity relations between individuals recorded in the corresponding frame. The TGraph [34] is also a time-dependent network management system based on Neo4j. The TGraph uses Neo4j storage nodes, relationships and static properties of nodes/relationships, while the time-dependent properties of nodes/relationships are stored by a dynamic properties storage (DPS) component. The TGraph first writes the time-dependent properties to the memory data structure MemTable. Then, DPS stores the data in the MemTable into the disk files UnStableFile and StableFile. Each UnStableFile and StableFile holds the data within a time interval. Finally, DPS uses MetaFile to record the filename and the corresponding time interval of each file. The TGraph records all time-dependent information in memory, which causes many data to be logged repeatedly.

To support the large-scale graph, distributed time-dependent graph management systems with low-cost storage and efficient query time-dependent information are proposed. DynamoGraph [67] exploits graph partitioning to store each individual vertex as a map containing key-value pairs tagged with time or JSON document. The edges are stored within vertex documents in a list that contains

the edges alongside with their attributes. DeltaGraph [41] performs snapshot retrieval queries in parallel by recording graph data over time through a hierarchical index structure. In DeltaGraph, the leaf nodes correspond to equispaced snapshots, and the edges between the nodes preserve the difference between the corresponding snapshots. Snapshots and edges can be distributed to a group of computers in parallel by using horizontal partitions for distributed storage. DeltaGraph records an atomic activity in the network by an event. An event can be used for creation, deletion, and changing attributes of vertices and edges. Moreover, an event represents the occurrence of an edge or vertex at a time instance. Hence, an event always corresponds to a single timepoint. Therefore, for the snapshot S_t and S_{t+1} at time t and $t+1$, respectively, we have that

$$S_{t+1} = S_t + E, \qquad S_t = S_{t+1} - E.$$

where E is the set of all events at time $t+1$. Khurana and Deshpande showed that the DeltaGraph can solve both single-point queries as well as multi-point snapshot queries, that is, to retrieve a graph structure/a subset of vertex or edge attributed/all attributed in a time instance and a time interval, respectively.

6.2 Traversal-Based System

Snapshot-based systems are useful for analyzing the evolution of networks. However, as pointed out in [75], the query results obtained by converting the time-dependent graph into a static graph are completely different from the results directly obtained from time-dependent graph. Therefore, these snapshot-based systems only adapt to settle snapshot-based queries, but not suitable for traversal-based queries such as information diffusion analysis [10]. Compared to snapshot-based systems, traversal-based systems are suitable for analyzing information dissemination that follows time limits.

The ChronoGraph [10] is a novel system enabling time-dependent graph traversals. The ChronoGraph supports three types of graphs, including static property graphs, time-instant property graphs, and time-period property graphs. It splits time-dependent graph into a collection of events that are instances of graph elements that are valid for a specific time-instant or a non-negligible period of time. In a citation network, the time-instant property graph consists of $\bigcup_{id_1,id_2,t} e(id_1|isCitedBy|id_2)_t$, representing $v(id_1)$ cites $v(id_2)$ at time t. And in a phone call network, the time-period property graph consists of $\bigcup_{id_1,id_2,(t_1,t_2)} e(id_1|isCalling|id_2)_{(t_1,t_2)}$, representing $v(id_1)$ has a phone call with $v(id_2)$ starting at t_1 and ending at t_2. The ChronoGraph divides the time period from the time instances into a time period by a property filter, a property skip parameter, a singular constraint filter, and a singular constraint skip parameter, and converts all time-instant events into time-period events. Then, the ChronoGraph uses a path management scheme, formed as $Map < Object, Set < Lsit < Object >>>$, to manage a pair of current graph elements and their relevant path set (i.e., lists of objects). The ChronoGraph allows convenient and efficient time-dependent graph traversal and has a good effect on time-dependent breadth-first search, depth-first search, and single source shortest path.

7 Conclusions and Open Questions

In this paper, we reviewed extensive studies on time-dependent graphs. We showed how dynamic systems benefit from the modeling of time-dependent graphs and discussed methods for discovering and analyzing dynamic network structures in the time domain. Meanwhile, we explained the great significance of the time-dependent graph structure in network researches, which has a good application in many fields. Therefore, the studies of the time-dependent graph have great theoretical significance and broad application prospects for studying the query processing and mining problems. As the data scale continues to expand and the data structure becomes more and more complex, these bring more challenges to the query processing and data mining problems on time-dependent graphs, which also bring more opportunities for researchers.

The study of the time-dependent graph is still a rather young field with many open questions and unexplored directions. We will list some of these issues as follows:

Generative models for time-dependent networks. There are very few models building for time-dependent networks. At the same time, existing models also have shortcomings, especially the snapshot-based model which ignores the impact of time information between snapshots. Therefore, an important open issue is to clearly construct and study parameterized, generative models for time-dependent networks.

Measures for time-dependent networks. Although a large number of measures of time-dependent graphs have been discussed in this review, we believe there is much room for improvement in this measure. The existing studies of time-dependent graphs are mostly based on time instances which are discrete. However, the discrete time-dependent graph is rarely used in practical applications, and more is the time-dependent graph with continuous time. Therefore, how to extend the existing time-dependent algorithms to not only discrete time-dependent graphs but also continuous time-dependent graphs is one of the focuses of future research works. At the same time, there are many unique time-dependent queries but most of the existing measures are generalizations of static network measures. Accordingly,

other important open issues are quantifying and characterizing the structural features of real-time-dependent systems.

Dynamical systems for time-dependent networks As we mentioned in Sect. 6, most of the time-dependent systems are snapshot based which only adapt to snapshot-based queries. These systems are less sensitive to time-dependent effects and only focus on the structure changes over time. However, a maturity and integrity time-dependent system does not only benefit to the questions about structure contacts, but also about the nature of acting over contacts. Consequently, there should be more systems that are sensitive to time-dependent effects like the order of events.

Acknowledgements This work is supported by the National Natural Science Foundation of China for Excellent Young Scientists (61622202), the National Natural Science Foundation of China (61732003, 61572119), and the Fundamental Research Funds for the Central Universities (N150402005).

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

Informed consent Informed consent was obtained from all individual participants.

References

1. Batz GV, Sanders P (2012) Time-dependent route planning with generalized objective functions. In: European symposium on algorithms. Springer, pp 169–180
2. Becker B, Gschwind S, Ohler T, Seeger B, Widmayer P (1996) An asymptotically optimal multiversion B-tree. VLDB J Int J Very Large Data Bases 5(4):264–275
3. Bellman R (1958) On a routing problem. Q Appl Math 16(1):87–90
4. Bhadra S, Ferreira A (2003) Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks. In: International conference on ad-hoc networks and wireless. Springer, pp 259–270
5. Bhavsar SP, Splinter RJ (1996) The superiority of the minimal spanning tree in percolation analyses of cosmological data sets. Mon Not R Astron Soc 282(4):1461–1466
6. Bloom BH (1970) Space/time trade-offs in hash coding with allowable errors. Commun ACM 13(7):422–426
7. Bogdanov P, Mongiovì M, Singh AK (2011) Mining heavy subgraphs in time-evolving networks. In: 2011 IEEE 11th international conference on data mining. IEEE, pp 81–90
8. Bollobás B (2013) Modern graph theory, vol 184. Springer, Berlin
9. Braha D, Bar-Yam Y (2009) Time-dependent complex networks: dynamic centrality, dynamic motifs, and cycles of social interactions. In: Adaptive networks. Springer, pp 39–50
10. Byun J, Woo S, Kim D (2019) ChronoGraph: enabling temporal graph traversals for efficient information diffusion analysis over time. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2019.2891565
11. Carrasco B, Lu Y, da Trindade JM (2011) Partitioning social networks for time-dependent queries. In: Proceedings of the 4th workshop on social network systems. ACM, p 2
12. Carroll KP, McClaran SR, Nelson EL, Barnett DM, Friesen DK, William GN (1992) AUV path planning: an A* approach to path planning with consideration of variable vehicle speeds and multiple, overlapping, time-dependent exclusion zones. In: Proceedings of the 1992 symposium on autonomous underwater vehicle technology. IEEE, pp 79–84
13. Cattuto C, Quaggiotto M, Panisson A, Averbuch A (2013) Time-varying social networks in a graph database: a Neo4j use case. In: First international workshop on graph data management experiences and systems. ACM, p 11
14. Chabini I, Lan S (2002) Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks. IEEE Trans Intell Transp Syst 3(1):60–74
15. Chen X, Zhang C, Ge B, Xiao W (2017) Temporal query processing in social network. J Intell Inf Syst 49(2):147–166
16. Cooke KL, Halsey E (1966) The shortest route through a network with time-dependent internodal transit times. J Math Anal Appl 14(3):493–498
17. Cordella LP, Foggia P, Sansone C, Vento M (1999) Performance evaluation of the VF graph matching algorithm. In: Proceedings 10th international conference on image analysis and processing. IEEE, pp 1172–1177
18. Dehne F, Omran MT, Sack JR (2009) Shortest paths in time-dependent FIFO networks using edge load forecasts. In: Proceedings of the second international workshop on computational transportation science. ACM, pp 1–6
19. Dehne F, Omran MT, Sack JR (2012) Shortest paths in time-dependent FIFO networks. Algorithmica 62(1–2):416–435
20. Delling D, Wagner D (2009) Time-dependent route planning. In: Robust and online large-scale optimization. Springer, pp 207–230
21. Demiryurek U, Banaei-Kashani F, Shahabi C, Ranganathan A (2011) Online computation of fastest path in time-dependent spatial networks. In: International symposium on spatial and temporal databases. Springer, pp 92–111
22. Ding B, Yu JX, Qin L (2008) Finding time-dependent shortest paths over large graphs. In: Proceedings of the 11th international conference on extending database technology: advances in database technology. ACM, pp 205–216
23. Disser Y, Müller-Hannemann M, Schnee M (2008) Multi-criteria shortest paths in time-dependent train networks. In: International workshop on experimental and efficient algorithms. Springer, pp 347–361
24. Foschini L, Hershberger J, Suri S (2011) On the complexity of time-dependent shortest paths. In: Proceedings of the twenty-second annual ACM-SIAM symposium on discrete algorithms. SIAM, pp 327–341
25. Gendreau M, Ghiani G, Guerriero E (2015) Time-dependent routing problems: a review. Comput Oper Res 64:189–197
26. Gonzalez JE, Xin RS, Dave A, Crankshaw D, Franklin MJ, Stoica I (2014) GraphX: graph processing in a distributed dataflow framework. In: 11th USENIX symposium on operating systems design and implementation (OSDI 14), pp 599–613

27. Gunturi V, Shekhar S, Bhattacharya A (2010) Minimum spanning tree on spatio-temporal networks. In: International conference on database and expert systems applications. Springer, pp 149–158
28. Han JDJ, Bertin N, Hao T, Goldberg DS, Berriz GF, Zhang LV, Dupuy D, Walhout AJ, Cusick ME, Roth FP et al (2004) Evidence for dynamically organized modularity in the yeast protein–protein interaction network. Nature 430(6995):88
29. Hanneke S, Fu W, Xing EP et al (2010) Discrete temporal models of social networks. Electron J Stat 4:585–605
30. Hartmanis J (1982) Computers and intractability: a guide to the theory of NP-completeness. SIAM Rev 24(1):90
31. Held M, Karp RM (1970) The traveling-salesman problem and minimum spanning trees. Oper Res 18(6):1138–1162
32. Holme P, Edling CR, Liljeros F (2004) Structure and time evolution of an internet dating community. Soc Netw 26(2):155–174
33. Holme P, Saramäki J (2012) Temporal networks. Phys Rep 519(3):97–125
34. Huang H, Song J, Lin X, Ma S, Huai J (2016) TGraph: a temporal graph data management system. In: Proceedings of the 25th ACM international on conference on information and knowledge management. ACM, pp 2469–2472
35. Huang S, Fu AWC, Liu R (2015) Minimum spanning trees in temporal graphs. In: Proceedings of the 2015 ACM SIGMOD international conference on management of data. ACM, pp 419–430
36. Huo W, Tsotras VJ (2014) Efficient temporal shortest path queries on evolving social graphs. In: Proceedings of the 26th international conference on scientific and statistical database management. ACM, p 38
37. Ichoua S, Gendreau M, Potvin JY (2003) Vehicle dispatching with time-dependent travel times. Eur J Oper Res 144(2):379–396
38. Idri A, Oukarfi M, Boulmakoul A, Zeitouni K, Masri A (2017) A new time-dependent shortest path algorithm for multimodal transportation network. Procedia Comput Sci 109:692–697
39. Kempe D, Kleinberg J, Kumar A (2002) Connectivity and inference problems for temporal networks. J Comput Syst Sci 64(4):820–842
40. Khoa VD, Pham TV, Nguyen HT, Van Hoai T (2015) Multi–criteria route planning in bus network. In: IFIP international conference on computer information systems and industrial management. Springer, pp 535–546
41. Khurana U, Deshpande A (2013) Efficient snapshot retrieval over historical graph data. In: 2013 IEEE 29th international conference on data engineering (ICDE). IEEE, pp 997–1008
42. Köhler E, Langkau K, Skutella M (2002) Time-expanded graphs for flow-dependent transit times. In: European symposium on algorithms. Springer, pp 599–611
43. Kollmitz M, Hsiao K, Gaa J, Burgard W (2015) Time dependent planning on a layered social cost map for human-aware robot navigation. In: 2015 European conference on mobile robots (ECMR). IEEE, pp 1–6
44. Kontogiannis S, Zaroliagis C (2016) Distance oracles for time-dependent networks. Algorithmica 74(4):1404–1434
45. Lebre S, Becq J, Devaux F, Stumpf MP, Lelandais G (2010) Statistical inference of the time-varying structure of gene-regulation networks. BMC Syst Biol 4(1):130
46. Li D, Jia X, Liu H (2004) Energy efficient broadcast routing in static ad hoc wireless networks. IEEE Trans Mob Comput 3(2):144–151
47. Li Z, Ding B, Han J, Kays R (2010) Swarm: mining relaxed temporal moving object clusters. Proc VLDB Endow 3(1–2):723–734
48. Low Y, Bickson D, Gonzalez J, Guestrin C, Kyrola A, Hellerstein JM (2012) Distributed GraphLab: a framework for machine learning and data mining in the cloud. Proc VLDB Endow 5(8):716–727
49. Ma S, Hu R, Wang L, Lin X, Huai J (2017) Fast computation of dense temporal subgraphs. In: 2017 IEEE 33rd international conference on data engineering (ICDE). IEEE, pp 361–372
50. Malandraki C, Daskin MS (1992) Time dependent vehicle routing problems: formulations, properties and heuristic algorithms. Transp Sci 26(3):185–200
51. Malewicz G, Austern MH, Bik AJ, Dehnert JC, Horn I, Leiser N, Czajkowski G (2010) Pregel: a system for large-scale graph processing. In: Proceedings of the 2010 ACM SIGMOD international conference on management of data. ACM pp 135–146
52. Marchetti-Bowick M, Yin J, Howrylak JA, Xing EP (2016) A time-varying group sparse additive model for genome-wide association studies of dynamic complex traits. Bioinformatics 32(19):2903–2910
53. Menger K (1927) Zur allgemeinen kurventheorie. Fundam Math 10(1):96–115
54. Mertzios GB, Michail O, Spirakis PG (2019) Temporal network optimization subject to connectivity constraints. Algorithmica 81(4):1416–1449
55. Michail O (2016) An introduction to temporal graphs: an algorithmic perspective. Internet Math 12(4):239–280
56. Moinet A, Starnini M, Pastor-Satorras R (2015) Burstiness and aging in social temporal networks. Phys Rev Lett 114(10):108701
57. Nannicini G, Delling D, Liberti L, Schultes D (2008) Bidirectional A* search for time-dependent fast paths. In: International workshop on experimental and efficient algorithms. Springer, pp 334–346
58. Nicosia V, Tang J, Musolesi M, Russo G, Mascolo C, Latora V (2012) Components in time-varying graphs. Chaos Interdiscip J Nonlinear Sci 22(2):023101
59. Orda A, Rom R (1990) Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. J ACM (JACM) 37(3):607–625
60. Przytycka TM, Singh M, Slonim DK (2010) Toward the dynamic interactome: it's about time. Brief Bioinform 11(1):15–29
61. Pyrga E, Schulz F, Wagner D, Zaroliagis C (2008) Efficient models for timetable information in public transportation systems. J Exp Algorithmics (JEA) 12:2–4
62. Qiu X, Zhao L, Wang J, Wang X, Wang Q (2016) Effects of time-dependent diffusion behaviors on the rumor spreading in social networks. Phys Lett A 380(24):2054–2063
63. Rao A, Hero AO III, Engel JD et al (2007) Inferring time-varying network topologies from gene expression data. EURASIP J Bioinform Syst Biol 2007:7
64. Redmond U, Cunningham P (2016) Subgraph isomorphism in temporal networks. arXiv preprint arXiv:1605.02174
65. Riolo CS, Koopman JS, Chick SE (2001) Methods and measures for the description of epidemiologic contact networks. J Urban Health 78(3):446–457
66. Semertzidis K, Pitoura E (2019) Top-k durable graph pattern queries on temporal graphs. IEEE Trans Knowl Data Eng 31(1):181–194
67. Steinbauer M, Anderst-Kotsis G (2016) Dynamograph: a distributed system for large-scale, temporal graph processing, its implementation and first observations. In: Proceedings of the 25th international conference companion on world wide web. International World Wide Web Conferences Steering Committee, pp 861–866
68. Subramaniam S, Pope S (1998) A mixing model for turbulent reactive flows based on euclidean minimum spanning trees. Combust Flame 115(4):487–514
69. Sun X, Tan Y, Wu Q, Wang J (2017) Hasse diagram based algorithm for continuous temporal subgraph query in graph stream. In: 2017 6th international conference on computer science and network technology (ICCSNT). IEEE, pp 241–246

70. Tang J, Musolesi M, Mascolo C, Latora V (2009) Temporal distance metrics for social network analysis. In: Proceedings of the 2nd ACM workshop on online social networks. ACM, pp 31–36
71. van der Tuin MS, de Weerdt M, Batz GV (2018) Route planning with breaks and truck driving bans using time-dependent contraction hierarchies. In: Proceedings of the twenty-eighth international conference on automated planning and scheduling, ICAPS 2018, Delft, The Netherlands, June 24–29, 2018, pp 356–365
72. Wang C, Tang J, Sun J, Han J (2011) Dynamic social influence analysis through time-dependent factor graphs. In: 2011 International conference on advances in social networks analysis and mining. IEEE, pp 239–246
73. Wang S, Lin W, Yang Y, Xiao X, Zhou S (2015) Efficient route planning on public transportation networks: a labelling approach. In: Proceedings of the 2015 ACM SIGMOD international conference on management of data. ACM, pp 967–982
74. Wang W, Yang L, Liao Q, Zhu X, Zhang Q (2015) TiSA: time-dependent social network advertising. In: 2015 IEEE international conference on communications (ICC). IEEE, pp 1188–1193
75. Wu H, Cheng J, Huang S, Ke Y, Lu Y, Xu Y (2014) Path problems in temporal graphs. Proc VLDB Endow 7(9):721–732
76. Wu H, Cheng J, Ke Y, Huang S, Huang Y, Wu H (2016) Efficient algorithms for temporal path computation. IEEE Trans Knowl Data Eng 28(11):2927–2942
77. Xu Y, Huang J, Liu A, Li Z, Yin H, Zhao L (2017) Time-constrained graph pattern matching in a large temporal graph. In: Asia-Pacific web (APWeb) and web-age information management (WAIM) joint conference on web and big data. Springer, pp 100–115
78. Yang L, Zhou X (2017) Optimizing on-time arrival probability and percentile travel time for elementary path finding in time-dependent transportation networks: linear mixed integer programming reformulations. Transp Res Part B Methodol 96:68–91
79. Yang Y, Yan D, Wu H, Cheng J, Zhou S, Lui J (2016) Diversified temporal subgraph pattern mining. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 1965–1974
80. Zhao Q, Tian Y, He Q, Oliver N, Jin R, Lee WC (2010) Communication motifs: a tool to characterize social communications. In: Proceedings of the 19th ACM international conference on Information and knowledge management. ACM, pp 1645–1648