Temporal Pattern Matching

Campbell, Wu

June 9, 2016

1 Problem Definition

In this section, we provide a set of formal definitions of temporal graph and temporal graph query. Then, we will discussion different semantics for interpreting the graph pattern matching problem on temporal graphs.

1.1 Temporal Dimension

Given two time intervals $T_1, T_2 \in \mathbb{N}^2$, we define the following predicates and computation:

- $\pi_1(T_1) = t_1^s$, and $\pi_2(T_1) = t_1^f$;
- $T_1 = T_2 \Leftrightarrow t_1^s = t_2^s \wedge t_1^f = t_2^f$;
- $T_1 \subseteq T_2 \Leftrightarrow t_1^s \geq t_2^s \wedge t_1^f \leq t_2^f$;
- $T_1 \cap T_2 = (max(t_1^s, t_2^s), min(t_1^f, t_2^f))$;
- $T_1 \cup T_2 = (min(t_1^s, t_2^s), max(t_1^f, t_2^f))$;

1.2 Temporal Graph

A temporal graph is a node and edge-labeled graph that is annotated with time intervals. Formally,

Definition 1.1. *A **temporal graph** is a node and edge labeled $G = (V_G, E_G, L_G)$ where V_G is the set of vertices, and $E_G \subset V_G^2 \times \mathbb{N}^2$ is the set of edges, and $L_G : V_G \cup E_G \to \mathcal{L}$ is a label function that maps each node and each edge to a label in domain $\mathcal{L}$.*

We call the time interval (t^s, t^f) associated with each edge e the **active period** *of e.*

We provide a few helper functions to obtain the end nodes of each edge and the set the outgoing and incoming edges of a node:

- given a node $u \in V_G$, $out(u) = \{e \in E_G \mid e = (u, v)$ for some $v \in V_G\}$; $in(u) = \{e \in E_G \mid e = (v, u)$ for some $v \in V_G\}$.

- given an edge $e = (u, v) \in E_G$, $\pi_1(e) = u$ and $\pi_2(e) = v$.

$\mathrm{T} : E \to N^2$ is a function retrieve the active period for an edge $e \in E_G$, $\mathrm{T}(e) = (t^s_e, t^f_e)$.

We provide two helper function T_s and T_f that return the start and finish time of the active period of an edge: $\mathrm{T}_s(e) = \pi_1(T(e))$ and $\mathrm{T}_f(e) = \pi_2(T_(e))$.

Remarks:

1. G is not necessarily connected.
2. There maybe more than one edge between a pair of nodes, bearing different active period (t^s, t^f).
3. For each pair of edges e_1 and e_2 between the same pair of nodes (u, v) that have the same edge label, we can assume that the active period of the edges do not overlap, e.g., $\mathrm{T}(e_1) \cap \mathrm{T}(e_2) = \emptyset$, since if they do overlap, we can combine the two edges to form one whose active period is $\mathrm{T}(e_1) \cup \mathrm{T}(e_2)$.
4. There can be many simplified versions of the node and edge labeling. For instance, only nodes are labeled, but edges are not, hinting that all edges are labeled the same.
5. In this definition, we only associate edges with timestamps. We can assume that nodes are always active.
6. In this definition, G is a directed graph. To make it undirected, we can
   - define $E \subseteq P^2(V) \times \mathbb{N}^2$, where $P^2(V)$ is the powerset of size two over the naturals.
   - require that $(u, v, t^s, t^f) \in E_G \to (v, u, t^s, t^f) \in E_G$

Definition 1.2. *Given a graph $G = (V_G, E_G, L_G)$ and a set of edges $S_e \subseteq E_G$, we say that $\pi_{S_e}(G) = (V_{S_e}, S_e, L_G)$, where $V_{S_e} = \bigcup_{e \in S_e} \{\pi_1(e)\} \cup \{\pi_2(e)\}$, is the edge-induced graph of S_e on G.*

1.3 Temporal Graph Pattern

We define graph patterns in a way similar to how graph patterns are defined in sub-graph isomorphism problems, but allowing users to provide additional constraints on time.

Definition 1.3. *A* **temporal graph query** *$q =< G_q, T_q >$ consists of a graph pattern G_q and a time interval T_q.*

$T_q \in (\mathbb{N} \cup \{?\})^2$ is called the **global temporal constraint** *of q.*

The graph pattern is a connected graph $G_q = (V_q, E_q, L_q)$, where V_q is the set of nodes and $E_q \in V_q^2 \times (\mathbb{N} \cup \{?\})^2$ is the set of edges, and $L_q : V_q \cup E_q \to \mathcal{L} \cup \{?\}$ is a label function that maps each node/edge to a label in L or ?.

Associated with each edge in E_q, user can also provide a temporal constraint in the form of a time interval, again, both the start and finish time can be a constant or ?. We call $T(e_q)$ for each $e_q \in E_q$ the **local temporal constraints***.*

We overload the helper functions introduced earlier to apply to graph pattern and time intervals that serve as temporal constraints. In the computations and operations associated with time intervals, ? is interpreted as $-\infty$ when used as start time, and ∞ when used as finish time.
Remark: the definition above can be incorporated easily into SPARQL. We can investigate the details when we settle on the definition.

1.4 Temporal Graph Pattern Matching

Definition 1.4. *A* **match** *of a graph pattern G_q in G is a total mapping $h : \{e_q : e_q \in E_q\} \rightarrow \{e_G : e_G \in E_G\}$ such that:*

- *for each edge $e_q \in E_q$, the edge label predicate associated with e_q is satisfied by $h(e_q) \in E_G$.*
- *for each node $v_q \in V_q$, the mapping of the outgoing and incoming edges of v_q share the same end node $v_G \in V_G$ and the node label predicate associated with v_q is satisfied by v_G. Formally, for any two edges $e_1, e_2 \in E_q$, if $\pi_i(e_1) = \pi_j(e_2)$, where i, j can be 0 or 1, the following must hold: $\pi_i(h(e_1)) = \pi_j(h(e_2))$.*

Please note that the definition of matching is the same as pattern matching defined for conjunctive queries on graphs. The new problem is how we can take the temporal constraints into consideration, which will be defined next.

Note that we allow users to provide temporal constraints in the form of a time window for the whole pattern and for each edge, but we also provide the flexibility for them not to provide any specific temporal constraints via the "?" option. Hence, users' temporal specification can be very strict, or very relaxed, or anywhere in between. Here are some scenarios:

- most strict: user specifies explicit global and local temporal constraints, and for each constraint specified, the start time is the same as the end time.
- most relaxed: user specifies temporal constraints with all ?'s, which means infinity.
- anywhere in between: including the cases in which some temporal constraints contains ?.
- conflicted: the intersection of the global temporal constraint and at least one of the local temporal constraint is empty. **remark:** we can easily identify conflict cases and return empty results without query evaluation.

We first define a few semantics that explicitly address user-specified temporal constraints:

Definition 1.5. *Given a graph G, a temporal graph query $q =< G_q, T_q >$, we say that a graph pattern matching h explicitly satisfies the temporal constraint of q if*

- *under the* **exact** *semantics, h satisfies that:*
  - *for all $e_q \in E_q$, $T(h(e_q)) = T(e_q)$ and $T(h(e_q)) \subseteq T_q$.*
- *under the* **contain** *semantics, h satisfies that:*
  - *for all $e_q \in E_q$, $T(h(e_q)) \subseteq T(e_q)$ and $T(h(e_q)) \subseteq T_q$.*
- *under the* **contained** *semantics, h must satisfy that:*
  - *for all $e_q \in E_q$, $T(h(e_q)) \supseteq T(e_q)$ and $T(h(e_q)) \supseteq T_q$.*
- *under the* **intersection** *semantics, h must satisfy that:*
  - *for all $e_q \in E_q$, $T(h(e_q)) \cap T(e_q) \neq \emptyset$ and $T(h(e_q)) \cap T_q \neq \emptyset$.*

Remark: We need to think more carefully about how global temporal constraint T_q is interpreted in these semantics. Best way is to come up with some example queries.

The ***explicit temporal semantics*** defined above address only the issue of interpreting temporal constraints specified by users. Matching returned under all these semantics will include subgraphs that are not temporally traversable.

We next define a few ***implicit temporal semantics*** as remedy.

Definition 1.6. *Given a temporal graph G, we say that the graph is* **concurrent** *(denoted* CONCUR*) if $\bigcap_{e \in E_G} T(e) \neq \emptyset$.*

Definition 1.7. *Given a temporal graph G, we say that the graph is* ***weakly consecutive*** *(denoted* WCONSEC*) if for every $e_1 = (w, u, t_1^s, t_1^f), e_2 = (u, v, t_2^s, t_2^f \in E_G$, $t_2^f \geq t_1^s$, where we say it is* **strongly consecutive** *(denoted* SCONSEC *) if for any $e_1 = (w, u), e_2 = (u, v) \in E_G$, $T(e_1) \cap T(e_2) \neq \emptyset$.*

Lemma 1.1. *If a temporal graph G is strongly consecutive, it must be weakly consecutive.*

We use $\mathcal{C}_{imp}$ to represent the set of implicit temporal constraints. $\mathcal{C}_{imp} = \{\text{CONCUR}, \text{SCONSEC}, \text{WCONSEC}\}$.

Now, we can define ***implicit temporal semantics*** to demand that resultant matching sub-graph be **concurrent**, **weakly consecutive**, or **strongly consecutive**.

Conjectures:

1. If the graph G is of star shape, e.g., there exist a center node u such that for any other nodes v there exists an edge between u and v, and if the in-degree of $u > 0$, then, G is concurrent iff G is consecutive.
2. If the graph G is a clique, then, G is concurrent iff G is consecutive.

3. If there exists a circle that traverse through all nodes in G, then, there exist an edge on this circle whose active period dominates those of all other edges on the circle.

Remarks:

- Whether these conjectures are true or not may depend on whether the graph is directed or undirected.
  If a graph is connected, it is a start shape, but all edges going out, e.g., there is no pairs of edges that have tail-to-head connection, is it consecutive? apparently it satisfy our definition.
- Either we need to find some work that has proved them, or to prove them, which should not be hard.
- These conjectures, if proved true, can be used to speed up query evaluation process. It should be easy to identify star shape, clique and circle in query patterns, which are usually very small. This will allow us to strength the temporal constraints based on the properties described in the conjectures to enhance filtering.

We now define a few functions that apply the implicit temporal semantics.

Definition 1.8. *Given a temporal graph G the function τ_B is a Boolean function that takes an implicit temporal constraint c and a set of edges $S_e \subseteq E_G$ as input and returns true if the edge-induced graph $\pi_{S_e}(G)$ satisfies constraint c, and false otherwise.*

Conjectures:

1. Given a temporal graph G, a set of edges $S_e \subseteq E_G$ and a temporal constraint c, $\tau_B(c, S_e) \Rightarrow \forall S \subseteq S_e(tau_B(c, S)$

2 Edge Isomorphism via Co-incidence Interval Graph

In this section we want to enumerate the basics of the interval graph method by expanding on the matches h we defined above in Definition 1.4.

We introduce the concept of the ***co-incidence interval graph*****, a derived static graph representing the temporal and incidence relationships between adjacent edges of a temporal graph. This interval graph is a lossless encoding (shown to be bijective) of the full temporal graph G, which allows us to run existing algorithms to find appropriate patterns.**

Definition 2.1. *The co-incidence interval graph, $\mathcal{I}_G^c$ of a graph $G = (V_G, E_G, L_G)$ under implicit temporal constrain $c \in \mathcal{C}_{imp}$, is a graph $\mathcal{I}_G^c = (E, \mathcal{E})$, where its node set is E, the edge set of graph G, and it is edge set $\mathcal{E} \subseteq E^2$ is the set of "meta-edges" between elements in E. For any pair $e_1, e_e \in E$, $(e_1, e_2) \in \mathcal{E}$ if*

- *e_1 and e_2 share an endpoint, e.g.,* $\{\pi_1(e_1), \pi_2(e_1) \cap \pi_1(e_2), \pi_2(e_2) \neq \emptyset$*, and*
- $\tau_B(c, \{e_1, e_2\}) = True$

The generic nature of this definition of the co-incidence interval graph accommodate any implicit temporal semantics defined earlier.

example

Let I_c be the function that computes this co-incidence interval graph, given an implicit temporal constraint c, e.g., $I_c(G)\mathcal{I}_G^c$.

Lemma 2.1. *Given an implicit temporal constraint c, the function I_c is a bijection over graphs with $\delta(G) \geq 1$.*

Proof. Surjectivity follows the definition of function I_c, so, we will only show Injectivity.

Injectivity. All of the information necessary to restore the temporal graph G is stored in the edge-set E, since there is no vertex that is not an endpoint of an edge. Thus, if $I_c(G) = I_c(H)$, then $V(I_c(G)) = E(G) = E(H) = V(I_c(H))$, hence $G = H$. □

[[[**how do we store the node label of the original graph G in $I_c(G)$? we can restore all the identification of the nodes, but can we also restore the node labels?**]]] <==

Remarks:

- **Note that we didn't use the edge set of $\mathcal{I}_G^c$ at all in the proof above. This is because all of the structural information needed to describe the graph G is stored in the edge set (labels are handled by an external map L).**

- **Note that Lemma 2.1 only holds for graphs with $\delta(G) \geq 1$. This is because if there is a vertex that has degree zero, there is no edge that knows about it. This could be solved if you wanted to keep track of these vertices in $\mathcal{I}_G^c$. (Its also very unlikely for interesting large graphs for singletons to be of any use or importance. They will only be returned in trivial queries such as the empty graph or singletons).**

Corollary 2.1. *Given temproal graphs G and H and a implia condition c, if there exists some isomorphism $f_c : \mathcal{I}_G^c \to \mathcal{I}_H^c$, then $I_c^{-1} \circ f_c \circ I_c : G \to H$ is also an isomorphism.*

[[[need proof.]]] <==

Algorithm 1 outlines the construction of a coincidence interval graph. Its complexity is $O(|E| d_{\max}(G))$ in the edge-relational representation of the graph).

Algorithm 2 depicts the procedure for restoring the original temporal graph given a co-incidence interval graph.

[[[it is not clear how the node labels are restored.]]] <==

Algorithm 1: MakeCoincidenceInterval(G, c), equivalently $I_C(G)$

```
Input: A temporal graph $G = (V_G, E_G, L_G)$, a implicit temporal
       constraint $c$
Output: The coincidence interval graph $\mathcal{I}_G^c$
1 $\mathcal{E} = \emptyset$;
2 foreach edge pair $e, f \in E_G$ do
3     if $\tau_B(c)(\{T(e), T(f)\})$ then
4         Add meta-edge $(e, f)$ to $\mathcal{E}$;
5     end
6 end
7 return $(E_G, \mathcal{E})$;
```

Algorithm 2: UNMAKECOINCIDENCEINTERVAL(I_G^c, c), equivalently $I_c^{-1}(\mathcal{I}_G^c)$

```
  Input: The co-incidince interval graph $\mathcal{I}_G^c = (E, \mathcal{E})$
  Output: Temproal graph $G$
1 $E_G = E$;
2 $V_G = \emptyset$;
3 foreach $e \in E_G$ do
4     $V = V \cup \{\pi_1(e), \pi_2(e)\}$;
5 end
6 return $(V_G, E_G)$;
```