Algoritmos para o cálculo da distância temporal em grafos temporais

Junot Freire, Tiago Januario e Rafael Melo
Computational Intelligence and Optimization Research Lab (CInO)
Departamento de Ciência da Computação, Universidade Federal da Bahia
Avenida Adhemar de Barros, s/n, Salvador, BA 40170-110, Brazil
{junotsantos,januario,melo}@dcc.ufba.br

RESUMO

Grafos temporais são grafos cujas arestas possuem informações relacionadas aos períodos nos quais estão ativas. Esses grafos possuem aplicações em diversos contextos, incluindo o estudo da disseminação de epidemias e a análise da transmissão de informações em redes sociais. Este trabalho propõe dois novos algoritmos para o cálculo da distância temporal entre vértices de um grafo temporal, considerando que um número limitado de *hops* pode ser percorrido em um dado instante de tempo. Resultados computacionais demonstram a superioridade do algoritmo guloso proposto quando comparado ao algoritmo baseado na busca em largura.

PALAVRAS CHAVE. Grafos temporais, distância temporal, algoritmos.

Tópicos (TAG - Teoria e Algoritmos em Grafos, OA - Outras aplicações em PO)

ABSTRACT

Temporal graphs are graphs whose edges have information regarding the periods in which they are active. These graphs encounter applications in several contexts, including the study of epidemic dissemination and social network analysis. In this work we propose two new algorithms to determine the temporal distance between vertices in a temporal graph, considering that a limited number of hops can be traversed in a given time instant. Computational results show that a proposed greedy algorithm outperforms the breadth-first search based algorithm.

KEYWORDS. Temporal graphs, temporal distance, algorithms.

Paper topics (TAG - Graph Theory and Algorithms, OA - Other applications in OR)

1. Introdução

Grafos temporais são grafos nos quais as arestas possuem informações relacionadas ao seus tempos de existência. Grafos temporais tem sido objeto de diversos estudos devido à grande quantidade de aplicações, as quais incluem: análise de redes sociais, estudo de propagação epidemiológica, análise de redes de comunicação, escalonamento de vôos, planejamento de viagens, dentre outras (Holme e Saramäki [2012]).

Redes com informações temporais podem se comportar como redes dinâmicas ou redes complexas. Em uma rede dinâmica, as conexões se tornam presentes ou não ao longo do tempo (Holme e Saramäki [2012]). Essas estruturas podem ser usadas para análise de comportamentos entre grupos de indivíduos em redes sociais, dentre outras coisas. Uma rede complexa, por sua vez, possui estrutura irregular, complexa e evolui dinamicamente ao longo do tempo (Boccaletti et al. [2006]), e podem ser utilizadas para a modelagem de relações mais caóticas, como a internet.

Neste trabalho, estuda-se a determinação da distância temporal em grafos temporais, i.e. o menor tempo de chegada a um vértice partindo de um vértice fonte no grafo. O restante deste trabalho está organizado da seguinte forma. A Seção 2 apresenta definições preliminares em grafos temporais bem como uma breve revisão de trabalhos relacionados. Na Seção 3 são propostos dois algoritmos utilizando diferentes representações dos grafos para o cálculo da distância temporal a partir de um vértice fonte, considerando-se que um número limitado de *hops* podem ser percorridos em um dado instante de tempo. Resultados computacionais preliminares são apresentados na Seção 4. Considerações finais são discutidas na Seção 5.

2. Definições preliminares sobre grafos temporais

Seja $G = (V, E)$ um *grafo temporal* com um conjunto V de vértices e um conjunto E de arestas possuindo tempos de duração contidos no intervalo discretizado $[0, T]$ (Holme e Saramäki [2013]). Cada aresta $e \in E$ é uma quádrupla $e = (u, v, t_s, t_f)$ onde: $u, v \in V$; t_s e t_f indicam, respectivamente, o tempo de ativação e o tempo de fim da aresta, $0 \leq t_s \leq t_f \leq T$. Denota-se por $t_s(e)$ e $t_f(e)$, respectivamente, o tempo de ativação e o tempo de fim da aresta e. Observe que múltiplas arestas podem existir entre dois vértices $u, v \in V$. Em um *grafo temporal valorado*, cada aresta $e \in E$ possui um pesso associado w_e. Em um *grafo temporal direcionado*, cada aresta $e = (u, v, t_s, t_f)$ é definida por um par ordenado (u, v) representando o arco direcionado de u para v. Neste trabalho, em situações nas quais as únicas informações necessárias sobre uma aresta temporal e sejam seus vértices incidentes, a mesma poderá ser denotada $e = (u, v)$.

Adicionalmente, considera-se $G_{t_{min},t_{max}} = (V, E')$ como o grafo temporal restrito ao intervalo $[t_{min}, t_{max}]$, no qual $E' = \{e \in E \mid [t_s(e), t_f(e)] \cap [t_{min}, t_{max}] \neq \varnothing\}$. Dado um tempo $t \in [0, T]$, a matriz de adjacência temporal $A(t)$ contém elementos $a_{uv}(t)$ representando a adjacência de u e v no tempo t. Há diferentes representações para grafos temporais disponíveis na literatura, incluindo a *condensada*, a *decomposta por instantes de tempo* e a *rotulada*. A *representação condensada* basicamente considera o grafo $G = (V, E)$ com a informação temporal de cada aresta $e = (u, v, t_s, t_f)$ representada através do tempo de ativação t_s e do tempo de fim t_f. A *representação decomposta por instantes de tempo* considera G como uma sequência de grafos G_t, $0 \leq t \leq T$, onde cada G_t possui um conjunto de vértices V e um conjunto de arestas $E_t = \{e \in E \mid t_s(e) \leq t \leq t_f(e)\}$. O grau do vértice v no grafo G_t é dado por $d_{G_t}(v)$. A *representação rotulada* considera um mapeamento λ sobre as arestas de G no qual, para cada $e \in E$, $\lambda(e) = \{t_1, \ldots, t_k\}$ é o conjunto de tempos nos quais a aresta está disponível $0 \leq t_i \leq T$ para $1 \leq i \leq k$.

2.1. Passeios e caminhos temporais

Um *passeio temporal* $w^h_{ij}(t_{min}, t_{max})$ entre dois vértices $i, j \in V$ é uma sequência $v_0 e_0 v_1 e_1 v_2 \ldots v_{k-1} e_{k-1} v_k$ de k arestas na qual: v_l é o l-ésimo vértice alcançado no passeio temporal sendo $v_0 = i$ e $v_k = j$; e tem-se uma sequência crescente de tempos de travessia $t_{min} \leq t_{v_0} \leq t_{v_1} \leq \ldots \leq t_{v_k} \leq t_{max}$ tal que $a_{v_{l-1},v_l}(t_l) \neq 0$ para $l = 1, \ldots, k$. O valor h representa o número

máximo de *hops* (subsequência de arestas consecutivas no passeio) permitidos em um instante de tempo t. Define-se t_{v_k} como o *tempo de chegada* de um passeio temporal $w_{ij}^h(t_{min}, t_{max})$. Um *caminho temporal* $p_{ij}^h(t_{min}, t_{max})$ é um passeio temporal passando por cada vértice no máximo uma vez (Grindrod et al. [2011]; Holme e Saramäki [2013]; Tang et al. [2010]). Dados dois vértices $i, j \in V$, define-se a *distância temporal* $d_{ij}^h(t_{min}, t_{max})$ como o menor tempo de chegada considerando todos os caminhos temporais possíveis entre i e j no intervalo $[t_{min}, t_{max}]$, onde $d_{ij}^h(t_{min}, t_{max}) = \infty$ caso não exista tal caminho e a distância temporal de um vértice a ele mesmo é definida como $d_{ii}^h(t_{min}, t_{max}) = \min\{t \mid t_{min} \leq t \leq t_{max},\ t_s(e) \leq t \leq t_f(e)$ para algum $e \in E_t(i)\}$.

Uma definição alternativa de passeio temporal considera o intervalo $[t_s(e), t_f(e)]$ como o tempo de travessia de uma aresta $e \in E$ (Wu et al. [2014, 2016]). Nesta, um passeio temporal $w_{ij}^\delta(t_{min}, t_{max})$ entre dois vértices $i, j \in V$ é uma sequência $v_0 e_0 v_1 e_1 v_2 \ldots v_{k-1} e_{k-1} v_k$ de k arestas, sendo $v_0 = i$, $v_k = j$, e uma sequência crescente de tempos de travessia $t_{min} \leq t_{v_0} \leq t_{v_1} \leq \ldots \leq t_{v_k} \leq t_{max}$ tal que $t_f(v_{l-1}, v_l) = t_{v_l} \leq t_s(v_l, v_{l+1})$ para todo $1 \leq l \leq k-1$. Consequentemente, define-se um caminho temporal $p_{ij}^\delta(t_{min}, t_{max})$ como um passeio temporal sem repetição de vértices.

2.2. Trabalhos relacionados

Problemas relacionados a passeios e caminhos temporais têm sido estudados por diversos autores nos últimos anos. Tang et al. [2010] propõem um método baseado em busca em profundidade para o cálculo de distâncias temporais considerando valores arbitrários de número máximo de *hops* por instante de tempo. O método proposto pelos autores, no entanto, não é apresentado de forma clara e os autores não fornecem uma prova de corretude para o mesmo. Considerando $h = 1$, a complexidade do método apresentado é $O(T \times (|V| + |E|))$, onde T é o número de instantes de tempo.

Mertzios et al. [2013, 2015] propõem um algoritmo para cálculo de distâncias temporais utilizando uma representação rotulada. O algoritmo apresentado assemelha-se ao descrito em Tang et al. [2010], porém considerando $h = 1$. O algoritmo proposto possui complexidade $O(|V|\lambda_{max}^3 + |E|)$, sendo λ_{max} a maior duração de uma aresta existente no grafo, o que o torna ineficiente para grafos com arestas de longa duração.

Algoritmos para cálculo de distância temporal, permitindo números ilimitados de *hops* em um instante de tempo, utilizando a representação condensada de um grafo temporal direcionado são descritos em Xuan et al. [2003] e Wu et al. [2016]. O algoritmo proposto em Xuan et al. [2003] possui complexidade $O(|E|(log\Delta_E + log|V|))$, sendo Δ_E o maior número de instantes de tempo dentre as arestas em E. Por outro lado, algoritmo proposto em Wu et al. [2016] calcula distâncias temporais levando em conta o tempo de travessia das arestas, i.e. $p_{ij}^\delta(t_{min}, t_{max})$, e possui complexidade $O(|V| + |E|)$. No entanto, os autores assumem que as arestas já estão ordenadas em relação ao tempo de início.

3. Algoritmos para o cálculo da distância temporal

Nesta seção, são propostos dois algoritmos para o cálculo da distância temporal a partir de um vértice fonte a todos os demais vértices de um grafo temporal não-direcionado. O primeiro utiliza a representação de grafo decomposto em instantes de tempo enquanto o segundo considera a representação condensada do grafo. No restante da seção, considere um grafo temporal não-direcionado $G_{t_{min}, t_{max}}$, um vértice fonte r e o número máximo de *hops* h permitidos em um mesmo instante de tempo.

Defina os seguintes atributos associados a cada vértice $u \in V$: $u.d$ determina um limite superior para o tempo de chegada em u, $u.d = \infty$ caso u não tenha sido alcançado; $u.\pi$ é o antecessor de u no caminho temporal conhecido para alcançar u no tempo $u.d$, $u.\pi =$ NIL caso u não tenha sido alcançado; $u.hops$ indica o número de *hops* percorridos para se alcançar u em um instante de tempo t, $u.hops = \infty$ caso u não tenha sido alcançado, e $u.hops = 0$ caso u tenha sido alcançado em algum instante de tempo $t' < t$.

3.1. Algoritmo utilizando representação decomposta por instantes de tempo

O algoritmo EARLIEST-ARRIVAL-BFS utiliza uma variação da busca em largura no grafo decomposto por instantes de tempo, aprimorando as abordagens encontradas em Mertzios et al. [2013] e Tang et al. [2010]. Observe que a representação decomposta por instantes de tempo necessita de um espaço em $O((t_{max} - t_{min})(|V| + |E|))$ para armazenamento da estrutura. Defina L como a lista de vértices não alcançados.

Algoritmo 1 descreve o funcionamento de EARLIEST-ARRIVAL-BFS. Nas linhas 1-4, os atributos de todos os vértices, exceto a raiz r, são inicializados. Linhas 5-7 determinam o primeiro instante de tempo t no intervalo $[t_{min}, t_{max}]$ no qual existe uma aresta incidente a r e atribui t a $r.d$ na linha 8. A lista de vértices não visitados inicializa com todos os vértices, exceto a raiz, na linha 10. Cada iteração do laço nas linhas 11-13 realiza uma chamada a EARLIEST-ARRIVAL-BFS-EXPLORE, descrito no Algoritmo 2, o qual efetua uma busca em largura modificada para um dado instante de tempo t. Em EARLIEST-ARRIVAL-BFS-EXPLORE, todo vértice u já alcançado é adicionado a uma fila Q e atribui-se $u.hops = 0$ nas linhas 1-3. O laço das linhas 4-15 retira o primeiro vértice u da fila e, para todo vértice v adjacente a u, verifica se ele pode ser alcançado à partir de u, atualizando então os atributos de v. O vértice v só será adicionado a Q na linha 13 caso ainda não se tenha atingido o limite de *hops* no instante de tempo correspondente ao momento do seu alcance. EARLIEST-ARRIVAL-BFS retorna, na linha 14, o vetor $V.d$ contendo as distâncias temporais à partir de r para todo $v \in V$.

Algoritmo 1: EARLIEST-ARRIVAL-BFS($G, r, h, t_{min}, t_{max}$)

```
1  foreach u ∈ V(G) \ {r} do
2      u.d ← ∞;
3      u.π ← NIL;
4      u.hops ← ∞;
5  t ← t_min;
6  while d_{G_t}(r) = 0 and t ≤ t_max do
7      t++;
8  r.d ← t;
9  r.π ← NIL;
10 L ← V(G) \ {r} ;
11 while L ≠ ∅ and t ≤ t_max do
12     EARLIEST-ARRIVAL-BFS-EXPLORE(G_t, r, h, t);
13     t++;
14 return V.d;
```

Algoritmo 2: EARLIEST-ARRIVAL-BFS-EXPLORE(G_t,r,h,t)

```
1  Q ← V(G) \ L;
2  foreach u ∈ V(G) \ L do
3      u.hops ← 0;
4  while Q ≠ ∅ do
5      u ← DEQUEUE(Q);
6      foreach e = (u, v) ∈ E_t(u) do
7          if v ∈ L then
8              v.hops ← u.hops + 1;
9              v.π ← u {com incidência em e};
10             v.d ← t;
11             L ← L \ {v};
12             if v.hops < h then
13                 ENQUEUE(Q, v);
```

Teorema 1. *Algoritmo 1 calcula a distância temporal do vértice raiz r para todos os vértices em $V(G)$ e seu tempo de execução é $O((t_{max} - t_{min}) \times (|V| + |E|))$.*

Demonstração. Seja $\bar{L}$, com $L \cap \bar{L} = \varnothing$ e $L \cup \bar{L} = V(G)$, o conjunto de vértices cuja distância temporal é conhecida e seja $C = (\bar{L}, L)$ o corte que particiona o conjunto de vértices em $\bar{L}$ e L. A partir do instante de tempo $t = t_{min}$, o valor de t é incrementado até que alguma aresta do corte C esteja ativa e, inicialmente tem-se $\bar{L} = \{r\}$ e $L = V(G) \setminus \{r\}$. Em todo tempo t no qual há arestas ativas no corte C, realiza-se um percurso em largura no grafo G_t a partir dos vértices do conjunto $\bar{L}$ que sejam terminais de arestas em C. Todos os vértices de L que forem alcançáveis utilizando-se no máximo h *hops* terão suas distâncias temporais atualizadas na linha 10 do Algoritmo 2. Em seguida, tais vértices serão removidos do conjunto L e inseridos no conjunto $\bar{L}$. Esse procedimento é repetido sempre que houver alguma aresta ativa no corte C enquanto $t \leq t_{max}$ e $L \neq \varnothing$. Portanto, no fim da execução todo vértice $v \in \bar{L}$ alcançado à partir de r possuirá $v.d \neq \infty$ enquanto todo vértice $v \in L$ permanecerá com $v.d = \infty$.

A análise da complexidade do Algoritmo 1 depende da complexidade do Algoritmo 2. No Algoritmo 2, a cada iteração do laço de repetição das linhas 4-13 retira-se um vértice u da fila Q e examinam-se as arestas de $E_t(u)$. Uma vez que cada vértice entra e sai da fila no máximo uma vez, cada aresta do grafo G_t é examinada no máximo duas vezes. Assim, a execução de todas as iterações do while consome tempo proporcional a E_t no pior caso. O restante do Algoritmo 2 consome tempo proporcional a $V(G)$. Assim, a complexidade do Algoritmo 2 pode ser expressa como $O(|V| + |E|)$. No Algoritmo 1, o laço de repetição das linhas 1-4 consome tempo proporcional a $V(G)$. O laço de repetição das linhas 6-7 consome tempo $O(t_{max} - t_{min})$. No laço de repetição das linhas 11-13, a função EARLIEST-ARRIVAL-BFS-EXPLORE definida em Algoritmo 2 é executada $O(t_{max} - t_{min})$ vezes. Portanto, a complexidade do Algoritmo 1 pode finalmente ser expressa como $O((t_{max} - t_{min}) \times (|V| + |E|))$.

□

3.2. Algoritmo utilizando representação condensada

O algoritmo guloso EARLIEST-ARRIVAL-GREEDY calcula as distâncias temporais à partir da fonte utilizando a representação condensada do grafo temporal, possuindo princípios de funcionamento similares ao do algoritmo proposto em Xuan et al. [2003]. Observe que a representação condensada necessita de um espaço em $O(|V| + |E|)$ para armazenamento da estrutura. Define-se Q' como o conjunto de vértices não alcançados. A função MINIMUM(Q') retorna $\min_{u \in Q'} u.d$, enquanto EXTRACT-MIN(Q') retorna $\operatorname{argmin}_{u \in Q' : u.d = \text{MINIMUM}(Q')} u.hops$, i.e. o vértice u com menor $u.d$. Caso haja vértices $u, u' \in Q'$ tal que $u.d = u'.d$, EXTRACT-MIN(Q') retorna o vértice u com menor valor $u.hops$.

Algoritmo 3 descreve EARLIEST-ARRIVAL-GREEDY. O laço das linhas 1-4 inicializa os atributos de todos os vértices do grafo. Na linha 5, determina-se o primeiro instante de tempo no intervalo $[t_{min}, t_{max}]$ no qual existe uma aresta incidente a r. A linha 6 define a fila Q' contendo os vértices em $V \setminus \{r\}$ como não-alcançados. O laço das linhas 7-22 é executado enquanto houver elementos não alcançados em Q' e MINIMUM(Q') for diferente de ∞. Na linha 8, é extraído da fila Q' o vértice u que pode ser alcançado em menor tempo ou, em caso de empate, com menor quantidade de *hops*. O laço de repetição das linhas 9-22 itera para cada aresta $e = (u, v)$ incidente a u e caso v ainda esteja em Q' verifica se o mesmo pode ser alcançado em momento anterior à combinação $v.d$ e $v.hops$ da seguinte forma: (a) linhas 11-14 consideram o caso em que a aresta e existe no tempo de chegada de u, o número máximo de hops h ainda não foi alcançado no momento de chegada em u, e este tempo é menor do que aquele previamente conhecido para v; (b) linhas 15-18 consideram a situação em que a aresta e existe no tempo de chegada de u (com $u.d < t_f(e)$), porém o número máximo de hops h já foi alcançado no momento de chegada em u, e este tempo é menor do que aquele previamente conhecido para v; (c) linhas 19-22 consideram o caso em que a aresta e torna-se ativa somente após o tempo de chegada a u e o tempo de ativação da mesma é menor do que o menor tempo previamente conhecido para v.

Algoritmo 3: EARLIEST-ARRIVAL-GREEDY(G,r,h,t_{min},t_{max})

```
1  foreach u ∈ V(G) do
2      u.d ← ∞;
3      u.π ← NIL;
4      u.hops ← 0;
5  r.d ← min{t | t_min ≤ t ≤ t_max, t_s(e) ≤ t ≤ t_f(e) para algum e ∈ E_t(r)};
6  Q' ← V(G);
7  while Q' ≠ ∅ and MINIMUM(Q') ≠ ∞ do
8      u ← EXTRACT-MIN(Q');
9      foreach e = (u, v) ∈ E(u) do
10         if v ∈ Q' then
11             if t_s(e) ≤ u.d ≤ t_f(e) and u.hops < h and u.d < v.d then
12                 v.d ← u.d;
13                 v.hops ← u.hops + 1;
14                 v.π ← u {com incidência em e};
15             else if t_s(e) ≤ u.d < t_f(e) and u.d + 1 < v.d then
16                 v.d ← u.d + 1;
17                 v.hops ← 1;
18                 v.π ← u {com incidência em e};
19             else if u.d < t_s(e) and t_s(e) < v.d then
20                 v.d ← t_s(e);
21                 v.hops ← 1;
22                 v.π ← u {com incidência em e};
```

Teorema 2. *Algoritmo 3 calcula a distância temporal da raiz r para todos os vértices em $V(G)$ e pode ser implementado para executar em $O(|V|\log|V|+|E|)$.*

Demonstração. Seja $\bar{Q}'$, com $Q'\cap\bar{Q}'=\varnothing$ e $Q'\cup\bar{Q}'=V(G)$, o conjunto de vértices cuja distância temporal é conhecida e seja $C=(\bar{Q}',Q')$ o corte que particiona o conjunto de vértices em $\bar{Q}'$ e Q'. O valor $r.d$ determina o menor tempo de chegada a r de acordo com a definição (linha 5) fazendo com que inicialmente tenha-se $\bar{Q}'=\{r\}$ e $Q'=V(G)\setminus\{r\}$. O algoritmo mantém, a cada iteração do laço das linhas 7-22, uma partição $(\bar{Q}',Q')$ dos vértices. É então selecionado a cada iteração um vértice $u\in Q'$ para ser adicionado a $\bar{Q}'$ utilizando-se uma aresta leve $e=(u.\pi,u)\in C$, i.e. que faz com que u seja alcançado não mais tarde do que qualquer outro vértice em Q'. Suponha que exista uma solução na qual u não seja alcançado por $u.\pi$ e seja, portanto, alcançado usando uma aresta partindo de algum outro vértice u_0. A chegada a u partindo de u_0 não pode ter tempo de chegada menor que $u.d$ e, caso seja igual a $u.d$ o número de *hops* não pode ser menor do que $u.hops$, o que implicaria uma contradição já que a chegada a u partindo de $u.\pi$ não teria sido a escolha gulosa. Portanto, o tempo de chegada a u partindo de u_0 é maior ou igual a $u.d$, e caso seja igual, o número de *hops* é maior ou igual a $u.hops$. No entanto, o tempo de chegada a u partindo de r não pode ser maior do que $u.d$ pois nesse caso não seria a distância temporal e, portanto deve ser igual a $u.d$. Isso implica que considerando-se a chegada a u partindo de $u.\pi$ obtém-se o mesmo tempo de chegada quando considerando o alcance partindo de u_0 (e com número de *hops* menor ou igual), implicando que todo vértice alcançado à partir de u quando chega-se a u partindo de u_0 também pode ser alcançado no mesmo tempo quando considera-se a chegada a u partindo de $u.\pi$.

A análise da complexidade do Algoritmo 3 pode ser realizada de maneira agregada. Observe que cada vértice $v\in V(G)$ é removido de Q' no máximo uma vez, quando é percorrida toda sua lista de adjacência. Isso implica que cada aresta é verificada no máximo duas vezes e portanto, todas as operações correspondentes ao laço das linhas 9-22 são executadas em $O(|V|+|E|)$. Deve-se, no entanto, determinar o custo das extrações de vértices de Q' na linha 8. No pior caso, todos os vértices seriam removidos de Q', e utilizando-se um *heap* de Fibonacci, todas essas remoções podem ser realizadas em tempo $O(|V|\log|V|)$. Portanto, Algoritmo 3 é executado em $O(|V|\log|V|+|E|)$. □

4. Resultados preliminares

Nesta seção são apresentados os experimentos computacionais realizados para analisar a performance dos algoritmos apresentados na Seção 3. Todos os experimentos foram executadas em uma máquina executando Xubuntu x86 64 GNU/Linux, com processador Intel Core i7-4770S 3.10GHz, 8Gb de RAM. Os algoritmos foram implementados utilizando a linguagem C++.

Dois tipos de instâncias foram utilizados nos experimentos. **Instâncias aleatórias:** 12 grupos contendo 10 instâncias cada com arestas possuindo tempos de ativação e fim determinados aleatoriamente, sendo cinco com arestas de duração máxima $T/2$ e cinco com arestas de duração máxima T. **Instâncias reais:** obtidas à partir de três bases de dados reais disponíveis em *Koblenz Large Network Collection* (http://konect.uni-koblenz.de/) contendo: (a) *wikipedia-conflict*, representando conflitos positivos e negativos entre usuários do *Wikipedia*; (b) *facebook-wosn-links*, da rede social *Facebook*; e (c) *arxiv-HepPh*, das redes *arxiv*. A informação temporal das bases reais foi discretizada para o intervalo $[0; 500]$ e foram considerados dois casos, no primeiro os tempos de ativação e de fim de cada aresta são iguais; já no segundo, o tempo de término de todas as arestas é igual a T.

Os resultados para as instâncias aleatórias estão sumarizados na Tabela 1. Na tabela, a primeira coluna identifica o grupo de instâncias, as colunas 2-4 indicam as dimensões de cada grupo, e as colunas seguintes apresentam, para cada quantidade máxima de *hops* $h \in \{1, 10, 100\}$, o tempo médio de execução de cada algoritmo, EARLIEST-ARRIVAL-BFS (A_BFS) e EARLIEST-ARRIVAL-GREEDY (A_GREEDY), para cada grupo de instâncias. Os resultados computacionais mostram que o tempo médio de execução de A_GREEDY é muito menor em comparação com o de A_BFS. Enquanto A_GREEDY executou cada instância em menos de um segundo, houve casos em que A_BFS demorou mais de um minuto para finalizar o processamento. Adicionalmente, pode-se observar que quando $h = 10$ ou $h = 100$ A_BFS demorou menos tempo do que quando $h = 1$. A_GREEDY teve um aumento do tempo de execução para $h = 10$ e $h = 100$. Vale observar que, no geral, houve um aumento do tempo de execução para A_BFS ao aumentar h de 10 para 100, o que não foi observado para A_GREEDY.

Tabela 1: Tempos de execução utilizando instâncias aleatórias

				h=1		h=10		h=100	
Grupo	V	E	T	A_BFS	A_GREEDY	A_BFS	A_GREEDY	A_BFS	A_GREEDY
1	50000	500000	10	1,35	0,13	0,47	0,19	0,50	0,20
2	50000	1000000	10	1,35	0,20	0,66	0,28	0,65	0,27
3	100000	500000	10	1,90	0,12	0,88	0,28	1,05	0,29
4	100000	1000000	10	2,77	0,27	0,97	0,41	1,04	0,42
5	50000	500000	50	16,77	0,18	3,26	0,22	3,92	0,22
6	50000	1000000	50	12,23	0,26	3,22	0,32	3,41	0,33
7	100000	500000	50	40,99	0,28	8,32	0,35	15,66	0,33
8	100000	1000000	50	36,43	0,39	6,28	0,47	6,84	0,47
9	50000	500000	100	58,42	0,20	8,77	0,23	15,60	0,23
10	50000	1000000	100	43,11	0,28	10,43	0,34	12,03	0,33
11	100000	500000	100	153,86	0,31	28,04	0,34	52,30	0,33
12	100000	1000000	100	128,72	0,44	18,39	0,49	28,12	0,50

Os resultados para as instâncias reais são apresentados na Tabela 2. Pode-se observar que similarmente ao que aconteceu com as instâncias aleatórias, o algoritmo A_GREEDY apresentou tempos de execução muito menores em todos os casos. Para a base *arxiv*, considerando a duração máxima da aresta como sendo igual a T, o algoritmo A_BFS não finalizou a execução em 7200 segundos para nenhum valor de h.

5. Comentários finais

Neste trabalho foi estudado o cálculo da distância temporal partindo de um vértice fonte em grafos temporais, considerando-se que um número limitado de *hops* pode ser percorrido em cada instante de tempo. Dois algoritmos utilizando diferentes representações dos grafos foram propostos e comparados computacionalmente.

Tabela 2: Tempos de execução utilizando instâncias reais

					h=1		h=10		h=100	
Base	DurMax	V	E	T	A_BFS	A_GREEDY	A_BFS	A_GREEDY	A_BFS	A_GREEDY
wikipedia	1	7118	103675	500	137,70	0,03	128,98	0,04	128,84	0,04
	500				405,98	0,04	376,48	0,04	376,14	0,04
facebook	1	63731	335708	500	238,62	0,06	238,87	0,07	239,25	0,06
	500				439,91	0,11	439,73	0,11	444,74	0,11
arxiv	1	28093	4596803	500	516,18	0,51	514,70	0,51	515,17	0,51
	500				> 7200	0,51	> 7200	0,52	> 7200	0,52

O algoritmo EARLIEST-ARRIVAL-BFS utiliza uma representação do grafo temporal decomposta por instantes de tempo e possui tempo de execução $O((t_{max} - t_{min}) \times (|V| + |E|))$, enquanto o EARLIEST-ARRIVAL-GREEDY utiliza uma representação condensada e é executado em $O(|V| \log |V| + |E|)$. Resultados computacionais demonstraram que EARLIEST-ARRIVAL-GREEDY obteve uma performance muito melhor, sendo capaz de calcular a distância temporal de todas as instâncias testadas em menos de 1 segundo.

Agradecimentos: Os autores agradecem ao PIBIC/UFBA-CNPQ e ao CNPQ pelo apoio financeiro.

Referências

Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., e Hwang, D.-H. (2006). Complex networks: Structure and dynamics. *Physics Reports*, 424:175–308.

Grindrod, P., Parsons, M., Higham, D., e Estrada, E. (2011). Communicability across evolving networks. *Physical Review E*, 83(4):046–120.

Holme, P. e Saramäki, J. (2013). *Temporal Networks*. Understanding Complex Systems. Springer Berlin Heidelberg.

Holme, P. e Saramäki, J. (2012). Temporal networks. *Physics Reports*, 519(3):97 – 125. Temporal Networks.

Mertzios, G., Michail, O., Chatzigiannakis, I., e Spirakis, P. (2013). Temporal network optimization subject to connectivity constraints. In *International Colloquium on Automata, Languages, and Programming*, p. 657–668. Springer.

Mertzios, G., Michail, O., e Spirakis, P. (2015). Temporal network optimization subject to connectivity constraints. *arXiv preprint arXiv:1502.04382*.

Tang, J., Musolesi, M., Mascolo, C., e Latora, V. (2010). Characterising temporal distance and reachability in mobile and online social networks. *ACM SIGCOMM Computer Communication Review*, 40(1):118–124.

Wu, H., Cheng, J., Huang, S., Ke, Y., Lu, Y., e Xu, Y. (2014). Path problems in temporal graphs. *Proceedings of the VLDB Endowment*, 7(9):721–732.

Wu, H., Cheng, J., Ke, Y., Huang, S., Huang, Y., e Wu, H. (2016). Efficient algorithms for temporal path computation. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2927–2942.

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