Dynamic hierarchies in temporal directed networks

Nikolaj Tatti[1,2]

[1] F-Secure, Helsinki, Finland
[2] Aalto University, Espoo, Finland `nikolaj.tatti@aalto.fi`

Abstract. The outcome of interactions in many real-world systems can be often explained by a hierarchy between the participants. Discovering hierarchy from a given directed network can be formulated as follows: partition vertices into levels such that, ideally, there are only forward edges, that is, edges from upper levels to lower levels. In practice, the ideal case is impossible, so instead we minimize some penalty function on the backward edges. One practical option for such a penalty is agony, where the penalty depends on the severity of the violation. In this paper we extend the definition of agony to temporal networks. In this setup we are given a directed network with time stamped edges, and we allow the rank assignment to vary over time. We propose 2 strategies for controlling the variation of individual ranks. In our first variant, we penalize the fluctuation of the rankings over time by adding a penalty directly to the optimization function. In our second variant we allow the rank change at most once. We show that the first variant can be solved exactly in polynomial time while the second variant is **NP**-hard, and in fact inapproximable. However, we develop an iterative method, where we first fix the change point and optimize the ranks, and then fix the ranks and optimize the change points, and reiterate until convergence. We show empirically that the algorithms are reasonably fast in practice, and that the obtained rankings are sensible.

1 Introduction

The outcome of interactions in many real-world systems can be often explained by a hierarchy between the participants. Such rankings occur in diverse domains, such as, hierarchies among athletes [3], animals [8, 14], social network behaviour [11], and browsing behaviour [10].

Discovering a hierarchy in a directed network can be defined as follows: given a directed graph $G = (V, E)$, find an integer $r(v)$, representing a rank of v, for each vertex $v \in V$, such that ideally $r(u) < r(v)$ for each edge $(u, v) \in E$. This is possible only if G is a DAG, so in practice, we penalize each edge with a penalty $q(r(u), r(v))$, and minimize the total penalty. One practical choice for a penalty is *agony* [6, 15, 16], $q(r(u), r(v)) = \max(r(u) - r(v) + 1, 0)$. If $r(u) < r(v)$, an ideal case, then the agony is 0. On the other hand, if $r(u) = r(v)$, then we penalize the edge by 1, and the penalty increases as the edge becomes more 'backward'.

The major benefit of computing agony is that we can solve it in polynomial time [6, 15, 16].

In this paper we extend the definition of agony to temporal networks: we are given a directed network with time stamped edges[3] and the idea is to allow the rank assignment to vary over time; in such a case, the penalty of an edge with a time stamp t depends only on the ranks of the adjacent vertices at time t.

We need to penalize or constrain the variation of the ranks, as otherwise the optimization problem of discovering dynamic agony reduces to computing the ranks over individual snapshots. In order to do so, we consider 2 variants. In our first variant, we compute the fluctuation of the rankings over time, and this fluctuation is added directly to the optimization function, multiplied by a parameter λ. In our second variant we allow the rank to change at most once, essentially dividing the time line of a single vertex into 2 segments.

We show that the first variant can be solved exactly in $\mathcal{O}\left(m^2 \log m\right)$ time. On the other hand, we show that the second variant is **NP**-hard, and in fact inapproximable. However, we develop a simple iterative method, where we first fix the change points and optimize the ranks, and then fix the ranks and optimize the change points, and reiterate until convergence. We show that the resulting two subproblems can be solved exactly in $\mathcal{O}\left(m^2 \log m\right)$ time.

We show empirically that, despite the pessimistic theoretical running times, the algorithms are reasonably fast in practice: we are able to compute the rankings for a graph with over 350 000 edges in 5 minutes.

The remainder of the paper is organized as follows. We introduce the notation and formalize the problem in Section 2. In Section 3 we review the technique for solving static agony, and in Section 4 we will use this technique to solve the first two variants of the dynamic agony. In Section 5, we present the iterative solution for the last variant. Related work is given in Section 6. Section 7 is devoted to experimental evaluation, and we conclude the paper with remarks in Section 8. The proofs for non-trivial theorems are given in Appendix in supplementary material.

2 Preliminaries and problem definition

We begin with establishing preliminary notation, and then continue by defining the main problem.

The main input to our problem is a *weighted temporal directed graph* which we will denote by $G = (V, E)$, where V is the set of vertices and E is a set of tuples of form $e = (u, v, w, t)$, meaning an edge e from u to v at time t with a weight w. We allow multiple edges to have the same time stamp, and we also allow two vertices u and v to have multiple edges. If w is not provided we assume that an edge has a weight of 1. To simplify the notation we will often write $w(e)$ to mean the weight of an edge e. Let T be the set of all time stamps.

[3] An edge may have several time stamps.

A *rank assignment* $r : V \times T \to \mathbb{N}$ is a function mapping a vertex and a time stamp to an integer; the value $r(u;t)$ represents the rank of a vertex u at a time point t.

Our next step is to penalize backward edges in a ranking r. In order to do so, consider an edge $e = (u, v, w, t)$. We define the penalty as

$$p(e;r) = w \times \max(0, r(u;t) - r(v;t) + 1) \quad .$$

This penalty is equal to 0 whenever $r(v;t) > r(u;t)$, if $r(v;t) = r(u;t)$, then the $p(e;r) = w$, and the penalty increases as the difference $r(u;t) - r(v;t)$ increases.

We are now ready to define the cost of a ranking.

Definition 1. *Assume an input graph $G = (V, E)$ and a rank assignment r. We define a score for r to be*

$$q(r, G) = \sum_{e \in E} p(e;r) \quad .$$

Static ranking: Before defining the main optimization problems, let us first consider the optimization problem where we do *not* allow the ranking to vary over time.

*Problem 1 (*AGONY*).* Given a graph $G = (V, E)$, an integer k, find a ranking r minimizing $q(r, G)$, such that $0 \leq r(v;t) \leq k - 1$ and $r(v;t) = r(v;s)$, for every $v \in V$ and $t, s \in T$.

Note that AGONY does not use any temporal information, in fact, the exact optimization problem can be defined on a graph where we have stripped the edges of their time stamps. This problem can be solved exactly in polynomial time, as demonstrated by Tatti [16]. We should also point out that k is an optional parameter, and the optimization problem makes sense even if we set $k = \infty$.

Dynamic ranking: We are now ready to define our main problems. The main idea here is to allow the rank assignment to *vary* over time. However, we should penalize or constrain the variation of a ranking. Here, we consider 2 variants for imposing such a penalty.

In order to define the first variant, we need a concept of fluctuation, which is the sum of differences between the consecutive ranks of a given vertex.

Definition 2. *Let r be a rank assignment. Assume that T, the set of all time stamps, is ordered, $T = t_1, \ldots, t_\ell$. The* fluctuation *of a rank for a single vertex u is defined as*

$$\mathit{fluc}(u;r) = \sum_{i=1}^{\ell-1} |r(u, t_{i+1}) - r(u, t_i)| \quad .$$

Note that if $r(u, t)$ is a constant for a fixed u, then $\mathit{fluc}(u;r) = 0$. We can now define our first optimization problem.

*Problem 2 (*FLUC-AGONY*).* Given a graph $G = (V, E)$, an integer k, and a penalty parameter λ, find a rank assignment r minimizing

$$q(r, G) + \lambda \sum_{v \in V} \mathit{fluc}(v; r)\,,$$

such that $0 \leq r(v; t) \leq k - 1$ for every $v \in V$ and $t \in T$.

The parameter λ controls how much emphasis we would like to put in constraining *fluc*: If we set $\lambda = 0$, then the *fluc* term is completely ignored, and we allow the rank to vary freely as a function of time. In fact, solving FLUC-AGONY reduces to taking snapshots of G at each time stamp in T, and applying AGONY to these snapshots individually. On the other hand, if we set λ to be a very large number, then this forces $\mathit{fluc}(v; r) = 0$, that is the ranking is constant over time. This reduces FLUC-AGONY to the static ranking problem, AGONY.

In our second variant, we limit how many *times* we allow the rank to change. More specifically, we allow the rank to change only once.

Definition 3. *We say that a rank assignment r is a* rank segmentation *if each u changes its rank $r(u; t)$ at most once. That is, there are functions $r_1(u)$, $r_2(u)$ and $\tau(v)$ such that*

$$r(u; t) = \begin{cases} r_1(u), & t < \tau(u), \\ r_2(u), & t \geq \tau(u) \end{cases}\quad .$$

This leads to the following optimization problem.

*Problem 3 (*SEG-AGONY*).* Given a graph $G = (V, E)$ and an integer k, find a *rank segmentation* r minimizing $q(r; G)$ such that $0 \leq r(v; t) \leq k - 1$ for every $v \in V$ and $t \in T$.

Note that the obvious extension of this problem is to allow rank to change ℓ times, where $\ell > 1$. However, in this paper we focus specifically on the $\ell = 1$ case as this problem yields an intriguing algorithmic approach, given in Section 5.

3 Generalized static agony

In order to solve the dynamic ranking problems, we need to consider a minor extension of the static ranking problem.

To that end, we define a *static graph* $H = (W, A)$ to be the graph, where W is a set of vertices and A is a collection of directed edges (u, v, c, b), where $u, v \in V$, c is a positive—possibly infinite—weight, and b is an integer, negative or positive.

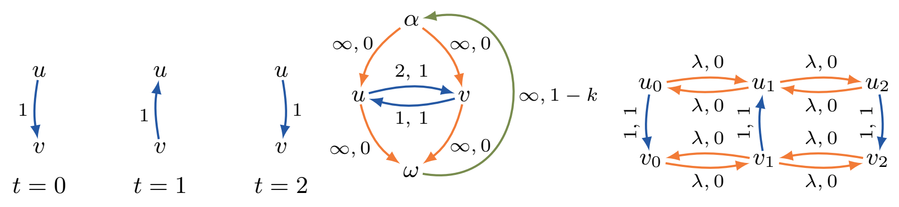
*Problem 4 (*GEN-AGONY*).* Given a static graph $H = (W, A)$ find a function $r : W \to \mathbb{Z}$ minimizing

$$\sum_{(u,v,c,b) \in A} \max(c \times (r(u) - r(v) + b), 0)\quad .$$

Note that c in (u, v, c, b) may be infinite. This implies that if the solution has a finite score, then $r(u) + b \leq r(v)$.[4]

We can formulate the static ranking problem, AGONY, as an instance of GEN-AGONY: Assume a graph $G = (V, E)$, and a(n optional) cardinality constraint k. Define a graph $H = (W, A)$ as follows. The vertex set W consists of the vertices V and two additional vertices α and ω. For each edge $(u, v, w, t) \in E$, add an edge $(u, v, c = w, b = 1)$ to A. If there are multiple edges from u to v, then we can group them and combine the weights. This guarantees that the sum in GEN-AGONY corresponds exactly to the cost function in AGONY. If k is given, then add edges $(\alpha, u, c = \infty, b = 0)$ and $(u, \omega, c = \infty, b = 0)$ for each $u \in V$. Finally, add $(\omega, \alpha, c = \infty, b = 1 - k)$. This guarantees that the for the optimal solution we must have $r(\alpha) \leq r(u) \leq r(\omega) \leq r(\alpha) + k - 1$, so now the ranking defined $r(u; t) = r(u) - r(\alpha)$ satisfies the constraints by AGONY.

Example 1. Consider a temporal network given in Figure 1a. The corresponding graph H is given in Figure 1b.



(a) Toy network, G (b) H for AGONY (c) H for FLUC-AGONY

Fig. 1: Graph G, and the corresponding graphs H used in AGONY and FLUC-AGONY. In (b), the edges with omitted parameters have $c = \infty$ and $b = 0$. In (c), vertices α and ω, and the adjacent edges, are omitted.

As argued by Tatti [16], GEN-AGONY is a dual problem of capacitated circulation, a classic variant of a max-flow optimization problem. This problem can be solved using an algorithm by Orlin [13] in $\mathcal{O}\left(|A|^2 \log |W|\right)$ time. In practice, the running time is faster.

4 Solving fluc-agony

In this section we provide a polynomial solution for FLUC-AGONY by mapping the problem to an instance of GEN-AGONY.

Assume that we are given a temporal graph $G = (V, E)$, a parameter λ and a(n optional) constraint on the number of levels, k.

[4] Here we adopt $0 \times \infty = 0$, when dealing with the case $r(u) - r(v) + b = 0$.

We will create a static graph $H = (W, A)$ for which solving GEN-AGONY is equivalent of solving FLUC-AGONY for G. First we define W: for each vertex $v \in V$ and a time stamp $t \in T$ such that there is an edge adjacent to v at time t, add a vertex v_t to W. Add also two vertices α and ω. The edges A consists of three groups A_1, A_2 and A_3:

(i) For each edge $e = (u, v, w, t) \in E$, add an edge $(u_t, v_t, c = w, b = 1)$.

(ii) Let $v_t, v_s \in W$ such that $s > t$ and there is no $v_o \in W$ with $t < o < s$, that is v_t and v_s are 'consecutive' vertices corresponding to v. Add an edge $(v_t, v_s, c = \lambda, b = 0)$, also add an edge $(v_s, v_t, c = \lambda, b = 0)$.

(iii) Assume that k is given. Connect each vertex u_t to ω with $b = 0$ and weight $c = \infty$. Connect α to each vertex u_t with $b = 0$ and weight $c = \infty$. Connect ω to α with $b = 1 - k$ and $c = \infty$. This essentially forces $r(\alpha) \leq r(u_t) \leq r(\omega) \leq r(\alpha) + k - 1$.

Example 2. Consider a temporal graph in Figure 1a. The corresponding graph, without α and ω, is given in Figure 1c.

Let r be the rank assignment for H with a finite cost, and define a rank assignment for G, $r'(v; t) = r(v_t)$. The penalty of edges in A_1 is equal to $q(r', G)$ while the penalty of edges in A_2 is equal to $\lambda \sum_{v \in V} \mathit{fluc}(v, r')$. The edges in A_3 force r' to honor the constraint k, otherwise $q(r, H) = \infty$. This leads to the following proposition.

Proposition 1. *Let r be the solution of* GEN-AGONY *for H. Then $r'(v; t) = r(v_t) - r(\alpha)$ solves* FLUC-AGONY *for G.*

We conclude with the running time analysis. Assume G with n vertices and m edges. A vertex $v_t \in W$ implies that there is an edge $(u, v, w, t) \in E$. Thus, $|W| \in \mathcal{O}(m)$. Similarly, $|A_1| + |A_2| + |A_3| \in \mathcal{O}(m)$. Thus, solving GEN-AGONY for H can be done in $\mathcal{O}\big(m^2 \log m\big)$ time.

5 Computing seg-agony

In this section we focus on SEG-AGONY. Unlike the previous problem, SEG-AGONY is very hard to solve (see Appendix for the proof).

Proposition 2. *Discovering whether there is a rank segmentation with a 0 score is an **NP**-complete problem.*

This result not only states that the problem is hard to solve exactly but it is also very hard to approximate: there is no polynomial-time algorithm with a multiplicative approximation guarantee, unless **NP** = **P**.

5.1 Iterative approach

Since we cannot solve the problem exactly, we have to consider a heuristic approach. Note that the rank assignment of a single vertex is characterized by 3 values: a change point, the rank before the change point, and the rank after the change point. This leads to the following iterative algorithm: (i) fix a change point for each vertex, and find the optimal ranks before and after the change point, (ii) fix the ranks for each vertex, and find the optimal change point. Repeat until convergence.

More formally, we need to solve the following two sub-problems iteratively.

*Problem 5 (*CHANGE2RANKS*).* Given a graph $G = (V, E)$ and a function τ mapping a vertex to a time stamp, find $r_1 : V \to N$ and $r_2 : V \to N$ mapping a vertex to an integer, such that the rank assignment r defined as

$$r(v; t) = \begin{cases} r_1(v), & t < \tau(v), \\ r_2(v), & t \geq \tau(v) \end{cases}$$

minimizes $q(r; G)$.

*Problem 6 (*RANKS2CHANGE*).* Given a graph $G = (V, E)$ and two functions $r_1 : V \to N$ and $r_2 : V \to N$ mapping a vertex to an integer, find a rank segmentation r minimizing $q(r; G)$ such that there is a function τ such that

$$r(v; t) = \begin{cases} r_1(v), & t < \tau(v), \\ r_2(v), & t \geq \tau(v) \end{cases} .$$

Surprisingly, we can solve both sub-problems exactly as we see in the next two subsections. This implies that during the iteration the score will always decrease. We still need a starting point for our iteration. Here, we initialize the change point of a vertex v as the median time stamp of v.

5.2 Solving change2ranks

We begin by solving the easier of the two sub-problems.

Assume that we are given a temporal network $G = (V, E)$ and a function $\tau : V \to T$. We will map CHANGE2RANKS to GEN-AGONY. In order to do so, we define a graph $H = (W, A)$. The vertex set W consists of two copies of V; for each vertex $v \in V$, we create two vertices v^1 and v^2, we also add vertices α and ω to enforce the constraint k. For each edge $e = (u, v, w, t) \in E$, we introduce an edge $(u^i, v^j, c = w, b = 1)$ to A, where

$$i = \begin{cases} 1 & \text{if } t < \tau(u), \\ 2 & \text{if } t \geq \tau(u), \end{cases} \quad \text{and} \quad j = \begin{cases} 1 & \text{if } t < \tau(v), \\ 2 & \text{if } t \geq \tau(v) \end{cases} .$$

Finally, like before, we add $(\alpha, v, c = \infty, b = 0)$, $(v, \omega, c = \infty, b = 0)$ and $(\omega, \alpha, c = \infty, b = 1 - k)$ to enforce the constraint k.

We will denote this graph by $G(\tau)$.

Example 3. Consider the toy graph given in Figure 1a. Assume $\tau(u) = 1$ and $\tau(v) = 2$. The resulting graph $G(\tau)$ is given in Figure 2a.

The following proposition shows that optimizing agony for H is equivalent of solving CHANGE2RANKS. We omit the proof as it is trivial.

Proposition 3. *Let r be a ranking for H. Define r' as*

$$r'(v;t) = \begin{cases} r(v^1) - r(\alpha), & t < \tau(v), \\ r(v^2) - r(\alpha), & t \geq \tau(v) \end{cases} .$$

Then $q(r', G) = q(r, H)$. Reversely, given a ranking r' satisfying conditions of CHANGE2RANKS*, define a ranking r for G by setting $r(v^i) = r_i(v)$. Then $q(r', G) = q(r, H)$.*

We conclude with the running time analysis. Assume G with n vertices and m edges. We have at most $2n + 2$ vertices in W and $|A| \in \mathcal{O}(m)$. Thus, solving CHANGE2RANKS for H can be done in $\mathcal{O}\big(m^2 \log n\big)$ time.

5.3 Solving ranks2change

Our next step is to solve the opposite problem, where we are given the two alternative ranks for each vertex, and we need to find the change points. Luckily, we can solve this problem in polynomial time. To solve the problem we map it to GEN-AGONY, however unlike in previous problems, the construction will be quite different.

Assume that we are given a graph $G = (V, E)$, and the two functions r_1 and r_2. To simplify the following definitions, let us first define

$$r_{min}(v) = \min(r_1(v), r_2(v)) \quad \text{and} \quad r_{max}(v) = \max(r_1(v), r_2(v)) \quad .$$

Assume an edge $e = (u, v, w, t) \in E$. A solution to RANKS2CHANGE must use ranks given by r_1 and r_2, that is the rank of u is either $r_{min}(u)$ or $r_{max}(u)$, and the rank of v is either $r_{min}(v)$ or $r_{max}(v)$, depending where we mark the change point for u and v. This means that there are only 4 possible values for the penalty of e. They are

$$\begin{aligned} p_{00}(e) &= w \times \max(0, r_{min}(u) - r_{min}(v) + 1), \\ p_{10}(e) &= w \times \max(0, r_{max}(u) - r_{min}(v) + 1), \\ p_{01}(e) &= w \times \max(0, r_{min}(u) - r_{max}(v) + 1), \\ p_{11}(e) &= w \times \max(0, r_{max}(u) - r_{max}(v) + 1) \quad . \end{aligned}$$

Among these penalties, $p_{01}(e)$ is the smallest, and ideally we would pay only $p_{01}(e)$ for each edge. This is rarely possible, so we need to design a method that takes other penalties into account.

Next we define a static graph $H = (W, A)$ that will eventually solve RANKS2CHANGE. For each vertex $v \in V$ and a time stamp $t \in T$ such that there is an edge adjacent to v at time t, add a vertex v_t to W. Add also two additional vertices α and ω. We will define the edges A in groups. The first two sets of edges in A

essentially force $r(u_t) = 0, 1$, and that the ranking is monotonic as a function of t. Consequenty, there will be at most only one time stamp for each vertex u, where the ranking changes. This will be the eventual change point for u. The edges are:

(i) Connect each vertex u_t to ω with $b = 0$ and weight $c = \infty$. Connect α to each vertex u_t with $b = 0$ and weight $c = \infty$. Connect ω to α with $b = -1$ and $c = \infty$. Connect α to ω with $b = 1$ and $c = \infty$. This forces $r(\alpha) \leq r(u_t) \leq r(\omega) = r(\alpha) + 1$.

(ii) Let $v_t, v_s \in W$ such that $s > t$ and there is no $v_o \in W$ with $t < o < s$. If $r_2(v) \geq r_1(v)$, then connect v_t to v_s with $b = 0$ and $c = \infty$. This forces $r(v_s) \geq r(v_t)$. If $r_2(v) < r_1(v)$, then connect v_s to v_t with $b = 0$ and $c = \infty$. This forces $r(v_s) \leq r(v_t)$.

For notational simplicity, let us assume that $r(\alpha) = 0$. The idea is then that once we have obtained the ranking for H, we can define the ranking for G as

$$r'(v; t) = r_{min}(v) + (r_{max}(v) - r_{min}(v))r(v_t) \quad .$$

Our next step is to define the edges that correspond to the penalties in the original graph. We will show later in Appendix that the agony of r' is equal to $P_1 + P_2 + P_3 + \mathit{const}$, where

$$\begin{aligned} P_1 &= \sum_{v_t \mid r(v_t)=0} \sum_{e=(u,v,w,t)\in E} p_{00}(e) - p_{01}(e), \\ P_2 &= \sum_{u_t \mid r(u_t)=1} \sum_{e=(u,v,w,t)\in E} p_{11}(e) - p_{01}(e), \\ P_3 &= \sum_{\substack{e=(u,v,w,t)\in E \\ r(v_t)=0, r(u_t)=1}} p_{10}(e) - p_{00}(e) - p_{11}(e) + p_{01}(e) \quad . \end{aligned}$$

Let us first define the edges that lead to these penalties .

(i) Connect ω to each vertex v_t with $b = 0$ and weight

$$c = \sum_{e=(u,v,w,t)\in E} p_{00}(e) - p_{01}(e) \quad .$$

In the sum v and t are fixed, and correspond to v_t. This edge penalizes vertices with $r(v_t) = 0$ with a weight of c. Summing these penalties yields P_1.

(ii) Connect each vertex u_t to α with $b = 0$ and weight

$$c = \sum_{e=(u,v,w,t)\in E} p_{11}(e) - p_{01}(e) \quad .$$

In the sum u and t are fixed, and correspond to u_t. This edge penalizes vertices with $r(u_t) = 1$ with a weight of c. Summing these penalties yields P_2.

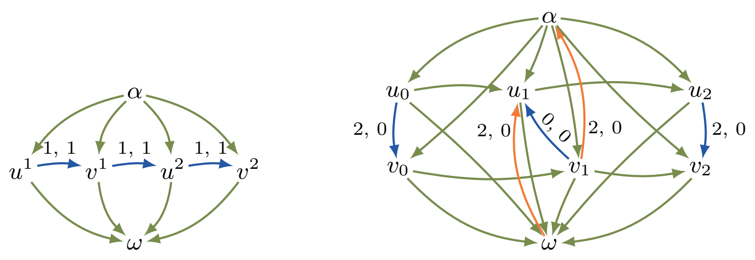
(iii) For each edge $e = (u, v, w, t) \in E$, connect u_t and v_t with $b = 0$ and

$$c = p_{10}(e) - p_{00}(e) - p_{11}(e) + p_{01}(e) \quad .$$

This edge penalizes cases when $r(u_t) = 1$ and $r(v_t) = 0$, and constitute P_3.

We will denote the resulting H by $G(r_1, r_2)$.

Example 4. Consider the toy graph given in Figure 1a. Assume that the rank assignments are $r_1(u) = 0$, $r_1(v) = 1$, $r_2(u) = 2$, $r_2(v) = 3$. The resulting graph $G(r_1, r_2)$ is given in Figure 2b. The optimal ranking for $G(r_1, r_2)$ assigns 0 to α, u_0, v_0, and v_1; the rank for the remaining vertices is 1.



(a) $G(\tau)$ (b) $G(r_1, r_2)$

Fig. 2: Graphs used for solving SEG-AGONY. In both figures, the edges with omitted parameters have $c = \infty$ and $b = 0$. For clarity, we omit edges between α and ω in both figures, in addition, in (b) we omit parameters for the edges (x, α) and (ω, x) with $c = 0$.

Before we show the connection between the ranks in G and $H = G(r_1, r_2)$, we first need to show that the edge weights are non-negative. This is needed to guarantee that we can find the optimal ranking of H using GEN-AGONY.

Proposition 4. *The weights of edges in H are non-negative.*

The proof is given in Appendix.

We will state our main result: we can obtain the solution for RANKS2CHANGE using the optimal ranking for H; see Appendix for the proof.

Proposition 5. *Let r be the optimal ranking for H. Then*

$$r'(v; t) = r_{min}(v) + (r_{max}(v) - r_{min}(v))(r(v_t) - r(\alpha))$$

solves RANKS2CHANGE.

We conclude this section with the running time analysis. Assume G with n vertices and m edges. A vertex $v_t \in W$ implies that there is an edge $(u, v, w, t) \in E$. Thus, $|W| \in \mathcal{O}(m)$. Similarly, $|A| \in \mathcal{O}(m)$. Thus, solving RANKS2CHANGE for H can be done in $\mathcal{O}\left(m^2 \log m\right)$ time.

6 Related work

Perhaps the most classic way of ranking objects based on pair-wise interactions is Elo rating proposed by Elo [3], used to rank chess players. A similar approach was proposed by Jameson et al. [8] to model animal dominance.

Maiya and Berger-Wolf [11] proposed discovering directed trees from weighted graphs such that parent vertices tend to dominate the children. A hierarchy is evaluated by a statistical model where the probability of an edge is high between a parent and a child. A good hierarchy is then found by a greedy heuristic.

Penalizing edges using agony was first considered by Gupte et al. [6], and a faster algorithm was proposed by Tatti [15]. The setup was further extended to handle the weighted edges, which was not possible with the existing methods, by Tatti [16], as well to be able to limit the number of distinct ranks (parameter k in the problem definitions).

An alternative to agony is a penalty that penalizes an edge (u, v) with $r(u) \geq r(v)$ with a constant penalty. In such a case, optimizing the cost is equal to FEEDBACK ARC SET (FAS), an **APX**-hard problem with a coefficient of $c = 1.3606$ [2]. Moreover, there is no known constant-ratio approximation algorithm for FAS, and the best known approximation algorithm has ratio $O(\log n \log \log n)$ [4]. In addition, Tatti [16] demonstrated that minimizing agony is **NP**-hard for any concave penalties while remains polynomial for any convex penalty function.

An interesting direction for future work is to study whether the rank obtained from minimizing agony can be applied as a feature in role mining tasks, where the goal is to cluster vertices based on similar features [7, 12].

SEG-AGONY essentially tries to detect a change point for each vertex. Change point detection in general is a classic problem and has been studied extensively, see excellent survey by Gama et al. [5]. However, these techniques cannot be applied directly for solving SEG-AGONY since we would need to have the ranks for individual time points.

The difficulty of solving SEG-AGONY stems from the fact that we allow vertices to have different change points. If we require that the change point must be the equal for all vertices, then the problem is polynomial. Moreover, we can easily extend such a setup for having ℓ segments. Discovering change points then becomes an instance of a classic segmentation problem which can be optimized by a dynamic program [1].

7 Experiments

In this section we present our experimental evaluation.

Datasets and setup: We considered 5 datasets. The first 3 datasets, *Mention*, *Retweet*, and *Reply*, obtained from SNAP repository [9], are the twitter interaction networks related to Higgs boson discovery. The 4th dataset, *Enron* consists of the email interactions between the core members of Enron. In addition, for illustrative purposes, we used a small dataset: *NHL*, consisting of

National Hockey League teams during the 2015–2016 regular season. We created an edge (x, y) if team x has scored more goals against team y in a single game during the 2014 regular season. We assign the weight to be the difference between the points and the time stamp to be the date the game was played. We used hours as time stamps for Higgs datasets, days for *Enron*. The sizes of the graphs are given in Table 1.

Table 1: Basic characteristics of the datasets and the experiments. The third data column, $|T|$, represents the number of unique time stamps, while the last column is the number of unique (v, t) pairs such that the vertex v is adjacent to an edge at time t, $\left|\bigcup_{(u,v,w,t)\in E} \{(v,t),(u,t)\}\right|$.

Name	$\|V\|$	$\|E\|$	$\|T\|$	$\|\{(v,t)\}\|$
Enron	146	105 522	964	24 921
Reply	38 683	36 395	168	54 892
Retweet	256 491	354 930	168	390 583
Mention	115 684	164 156	168	183 693
NHL	30	1 230	178	2 460

For each dataset we applied FLUC-AGONY, SEG-AGONY, and the static variant, AGONY. For FLUC-AGONY we set $\lambda = 1$ for the Higgs datasets, $\lambda = 2$ for *NHL* and *Enron*.

We implemented the algorithms in C++, and performed experiments using a Linux-desktop equipped with a Opteron 2220 SE processor.[5]

Computational complexity: First, we consider the running times, reported in Table 2. We see that even though the theoretical running time is $\mathcal{O}(m^2 \log n)$ for FLUC-AGONY and for a single iteration of SEG-AGONY, the algorithms perform well in practice. We are able to process graphs with 300 000 edges in 5 minutes. Naturally, SEG-AGONY is the slowest as it requires multiple iterations—in our experiments 3–5 rounds—to converge.

Table 2: Agony, running time, and number of unique ranks in the ranking.

	score			number of ranks			time		
Name	AGONY	FLUC	SEG	AGONY	FLUC	SEG	AGONY	FLUC	SEG
Enron	57 054	21 434	50 393	6	9	7	3s	4s	26s
Reply	6 017	5 401	4 147	13	12	16	0.4s	10s	15s
Retweet	2 629	1 384	1070	23	21	18	8s	4m	5m
Mention	12 756	10 082	8 219	20	19	18	4s	1m	2m
NHL	2 090	1 414	1 883	2	4	4	0.6s	0.3s	1s

[5] See `https://bitbucket.org/orlyanalytics/temporalagony` for the code.

Table 3: Statistics measuring fluctuation of the resulting rankings: *fluc* is equal to the fluctuation $\mathit{fluc}(u; r)$ averaged over u, *maxdiff* is the maximum difference between the ranks of a single vertex u, averaged over u, *change* is the number of times rank is changed for a single vertex u, averaged over u. Note that $\mathit{fluc} = \mathit{maxdiff}$ for SEG-AGONY as the assignment is allowed to change only once.

	fluc		*maxdiff*		*change*	
Name	FLUC	SEG	FLUC	SEG	FLUC	SEG
Enron	28.2	1	3.2	1	21.8	0.66
Reply	0.013	0.43	0.012	0.43	0.01	0.36
Retweet	0.003	0.17	0.003	0.17	0.002	0.13
Mention	0.016	0.3	0.014	0.3	0.011	0.2
NHL	2.7	0.73	1.5	0.73	2.6	0.5

Statistics of obtained rankings: Next, we look at the statistics of the obtained rankings, given in Table 2. We first observe that the agony of the dynamic variants is always lower than the static agony, as expected.

Let us compare the constraint statistics, given in Table 3. First, we see that FLUC-AGONY yields the smallest *fluc* in Higgs databases. SEG-AGONY produces smaller *fluc* in the other two datasets but it also produces a higher agony.

Interestingly enough, FLUC-AGONY yields a surprisingly low average number of change points for Higgs datasets. The low average is mainly due to most resulting ranks being constant, and only a minority of vertices changing ranks over time. However, this minority changes its rank more often than just once.

Agony vs fluctuation: The parameter λ of FLUC-AGONY provides a flexible way of controlling the fluctuation: smaller values of λ leads to smaller agony but larger fluctuation while larger values of λ leads to larger agony but smaller fluctuation. This can be seen in Table 2, where relatively large λ forces small fluctuation for the Higgs datasets, while relatively small λ allows variation and a low agony for *Enron* dataset. This flexibility comes at a cost: we need to have a sensible way of selecting λ. One approach to select this value is to study the joint behavior of the agony and the fluctuation as we vary λ. This is demonstrated in Figure 3 for *Enron* data, where we scatter plot the agony versus the average fluctuation, and vary λ. We see that agony decreases steeply as we allow some fluctuation over time but the obtained benefits decrease as we allow more variation.

Use case: Finally, let us look on the rankings by SEG-AGONY of *NHL* given in Figure 4. We limit the number of possible rank levels to $k = 3$.

The results are sensible: the top teams are playoff teams while the bottom teams have a significant losing record. Let us highlight some change points that reflect significant changes in teams: for example, the collapse of *Montreal Canadiens* (MTL) from the top rank to the bottom rank coincides with the injury of their star goaltender. Similarly, the rise of the *Pittsburgh Penguins* (PIT)

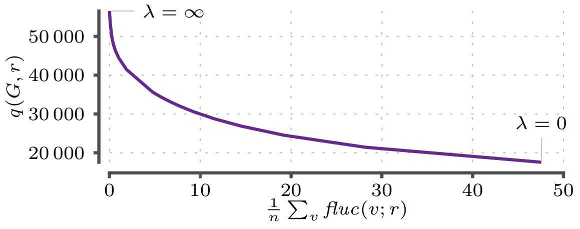


Fig. 3: Agony plotted against *fluc* of the optimal ranking for FLUC-AGONY by varying the parameter λ (*Enron*).

Ranking before the change:

1. MTL WIN BOS MIN DAL FLO WSH
2. VAN NYR CHI SJ TOR COL PIT NSH STL TBL DET NJ NYI
3. CAL LAK OTT BUF CAR EDM PHI CBJ ARI ANA

Ranking after the change:

1. NYR SJ LAK PIT NSH STL TBL PHI ANA WSH
2. CAL CHI BOS OTT BUF MIN DAL CAR CBJ DET ARI NJ NYI FLO
3. VAN MTL TOR WIN COL EDM

Change points:

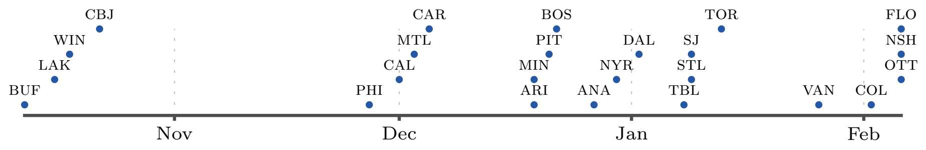


Fig. 4: Rank segmentations for *NHL* with $k = 3$. The bottom figure shows only the teams whose rank changed. The y-axis is used only to reduce the clutter.

from the middle rank to the top rank reflects firing of the head coach as well as retooling their strategy, *Penguins* eventually won the Stanley Cup.

8 Concluding remarks

In this paper we propose a problem of discovering a dynamic hierarchy in a directed temporal network. To that end, we propose two different optimization problems: FLUC-AGONY and SEG-AGONY. These problems vary in the way we control the variation of the rank of single vertices. We show that FLUC-AGONY can be solved in polynomial time while SEG-AGONY is **NP**-hard. We also developed an iterative heuristic for SEG-AGONY. Our experimental validation showed that the algorithms are practical, and the obtained rankings are sensible.

FLUC-AGONY is the more flexible of the two methods as the parameter λ allows user to smoothly control how much rank is allowed to vary. This comes at a price as the user is required to select an appropriate λ. One way to select λ is to vary the parameter and monitor the trade-off between the agony and the fluctuation. An interesting variant of FLUC-AGONY—and potential future line of work—is to minimize agony while requiring that the fluctuation should not increase over some given threshold.

The relation between SEG-AGONY and the sub-problems RANKS2CHANGE and CHANGE2RANKS is intriguing: while the joint problem SEG-AGONY is **NP**-hard not only the sub-problems are solvable in polynomial time, they are solved with the same mechanism.

A straightforward extension for SEG-AGONY is to allow more than just one change point, that is, in such a case we are asked to partition the time line of each vertex into ℓ segments. However, we can no longer apply the same iterative algorithm. More specifically, the solver for RANKS2CHANGE relies on the fact that we need to make only one change. Developing a solver that can handle the more general case is an interesting direction for future work.

References

1. Bellman, R.: On the approximation of curves by line segments using dynamic programming. Communications of the ACM 4(6) (1961)
2. Dinur, I., Safra, S.: On the hardness of approximating vertex cover. Annals of Mathematics 162(1), 439–485 (2005)
3. Elo, A.E.: The rating of chessplayers, past and present. Arco Pub. (1978)
4. Even, G., (Seffi) Naor, J., Schieber, B., Sudan, M.: Approximating minimum feedback sets and multicuts in directed graphs. Algorithmica 20(2), 151–174 (1998)
5. Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. 46(4), 44:1–44:37 (2014)
6. Gupte, M., Shankar, P., Li, J., Muthukrishnan, S., Iftode, L.: Finding hierarchy in directed online social networks. In: WWW. pp. 557–566 (2011)
7. Henderson, K., Gallagher, B., Eliassi-Rad, T., Tong, H., Basu, S., Akoglu, L., Koutra, D., Faloutsos, C., Li, L.: RolX: Structural role extraction & mining in large graphs. In: KDD. pp. 1231–1239 (2012)
8. Jameson, K.A., Appleby, M.C., Freeman, L.C.: Finding an appropriate order for a hierarchy based on probabilistic dominance. Animal Behaviour 57, 991–998 (1999)
9. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data` (Jan 2015)
10. Macchia, L., Bonchi, F., Gullo, F., Chiarandini, L.: Mining summaries of propagations. In: ICDM. pp. 498–507 (2013)
11. Maiya, A.S., Berger-Wolf, T.Y.: Inferring the maximum likelihood hierarchy in social networks. In: ICSE. pp. 245–250 (2009)

12. McCallum, A., Wang, X., Corrada-Emmanuel, A.: Topic and role discovery in social networks with experiments on enron and academic email. J. Artif. Int. Res. 30(1), 249–272 (2007)
13. Orlin, J.B.: A faster strongly polynomial minimum cost flow algorithm. Operations Research 41(2) (1993)
14. Roopnarine, P.D., Hertog, R.: Detailed food web networks of three Greater Antillean Coral Reef systems: The Cayman Islands, Cuba, and Jamaica. Dataset Papers in Ecology 2013 (2013)
15. Tatti, N.: Faster way to agony—discovering hierarchies in directed graphs. In: ECML PKDD. pp. 163–178 (2014)
16. Tatti, N.: Hierarchies in directed networks. In: ICDM. pp. 991–996 (2015)