Hierarchies in directed networks

Nikolaj Tatti
HIIT, Aalto University, Espoo, Finland, nikolaj.tatti@aalto.fi

Abstract—Interactions in many real-world phenomena can be explained by a strong hierarchical structure. Typically, this structure or ranking is not known; instead we only have observed outcomes of the interactions, and the goal is to infer the hierarchy from these observations. Discovering a hierarchy in the context of directed networks can be formulated as follows: given a graph, partition vertices into levels such that, ideally, there are only edges from upper levels to lower levels. The ideal case can only happen if the graph is acyclic. Consequently, in practice we have to introduce a penalty function that penalizes edges violating the hierarchy. A practical variant for such penalty is agony, where each violating edge is penalized based on the severity of the violation. Hierarchy minimizing agony can be discovered in $O(m^2)$ time, and much faster in practice. In this paper we introduce several extensions to agony. We extend the definition for weighted graphs and allow a cardinality constraint that limits the number of levels. While, these are conceptually trivial extensions, current algorithms cannot handle them, nor they can be easily extended. We provide an exact algorithm of $O(m^2 \log n)$ time by showing the connection of agony to the capacitated circulation problem. We also show that this bound is in fact pessimistic and we can compute agony for large datasets. In addition, we show that we can compute agony in polynomial time for any convex penalty, and, to complete the picture, we show that minimizing hierarchy with any concave penalty is an NP-hard problem.

I. INTRODUCTION

Interactions in many real-world phenomena can be explained by a strong hierarchical structure. As an example, it is more likely that a line manager in a large, conservative company will write emails to her employees than the other way around. Typically, this structure or ranking is not known; instead we only have observed outcomes of the interactions, and the goal is to infer the hierarchy from these observations. Discovering hierarchies or ranking has applications in various domains, such as, ranking players [3], discovering dominant animals [8], hierarchy discovery in social networks [11], and summarizing browsing behaviour [10].

We consider the following problem of discovering hierarchy in the context of directed networks: given a directed graph, partition vertices into ranked groups such that there are only edges from upper groups to lower groups.

Unfortunately, such a partitioning is only possible when the input graph has no cycles. Consequently, a more useful problem definition is to define a penalty function $p$ on the edges. This function should penalize edges that are violating a hierarchy. Given a penalty function, we are then asked to find the hierarchy that minimizes the total penalty.

The feasibility of the optimization problem depends drastically on the choice of the penalty function. If we attach a constant penalty to any edge that violates the hierarchy, that is, the target vertex is ranked higher or equal than the source vertex, then this problem corresponds to a feedback arc set problem, a well-known NP-hard problem [1], even without a known constant-time approximation algorithm [4].

A more practical variant is to penalize the violating edges by the severity of their violation. That is, given an edge $(u, v)$ we compare the ranks of the vertices $r(u)$ and $r(v)$ and assign a penalty of $\max(r(u) - r(v) + 1, 0)$. Here, the edges that respect the hierarchy receive a penalty of 0, edges that are in the same group receive a penalty of 1, and penalty increases linearly as the violation becomes more severe, see Figure 1. This particular score is referred as agony. Minimizing agony was introduced by Gupte et al. [6] where the authors provide an exact $O(nm^2)$ algorithm, where $n$ is the number of vertices and $m$ is the number of edges. A faster discovery algorithm with the computational complexity of $O(m^2)$ was introduced by Tatti [17]. In practice, the bound $O(m^2)$ is very pessimistic and we can compute agony for large graphs in reasonable time.

In this paper we specifically focus on agony, and provide the following main extensions for discovering hierarchies in graphs.

- **weighted graphs**: We extend the notion of the agony to graphs with weighted edges. Despite being a conceptually extension, current algorithms [6, 17] for computing agony are specifically design to work with unit weights, and cannot be used directly or extended trivially. Consequently, we need a new approach to minimize the agony, and in order to do so, we demonstrate that we can transform the problem into a capacitated circulation, a classic graph task known to have a polynomial-time algorithm.

- **cardinality constraint**: The original definition of agony does not restrict the number of groups in the resulting partition. Here, we introduce a cardinality constraint $k$ and we are asking to find the optimal hierarchy with at most $k$ groups. This constraint works both with weighted and non-weighted graphs. Current algorithms for solving agony cannot handle cardinality constraints. Luckily, we can enforce the constraint when we transform the problem into a capacitated circulation problem.

- **convex edge penalties**: Minimizing agony uses linear penalty for edges. We show that if we replace the linear penalty with a convex penalty, see Figure 1 we can still solve the problem in polynomial time. However, this extension increases the computational complexity.

- **concave edge penalties**: To complete the picture, we also study concave edge penalties, see Figure 1. We show that in this case discovering the optimal hierarchy is an NP-hard problem. This provides a stark difference between concave and convex edge penalties.

- **canonical solution**: A hierarchy minimizing agony may not be unique. For example, given a DAG any topological sorting of vertices will give you an optimal agony of 0. To address this issue we propose to compute a canonical
solution, where, roughly speaking, the vertices are ranked as high as possible without compromising the optimality of the solution. We demonstrate that this solution is unique, it creates a hierarchy with the least amount of groups, and that we can compute it in $O(n \log n + m)$ time.

The rest of the paper is organized as follows. We introduce the notation and formally state the optimization problem in Section II. In Section III we transform the optimization problem into a capacitated circulation problem, allowing us a polynomial-time algorithm. In Section IV we discuss alternative edge penalties. We demonstrate how to extract a canonical optimal solution in Section V. We discuss the related work in Section VI and present experimental evaluation in Section VII. Finally, we conclude the paper with remarks in Section VIII.

II. PRELIMINARIES AND PROBLEM DEFINITION

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

The main input to our problem is a weighted directed graph which we will denote by $G = (V, E, w)$, where $w$ is a function mapping an edge to real positive number. If $w$ is not provided we assume that each edge has a weight of 1. We will often denote $n = |V|$ and $m = |E|$.

As mentioned in the introduction, our goal is to partition vertices $V$. We express this partition with a rank assignment $r$, a function mapping a vertex to an integer. To obtain the groups from the rank assignment we simply group the vertices having the same rank.

Given a graph $G = (V, E)$ and a rank assignment $r$, we will say that an edge $(u, v)$ is forward if $r(u) < r(v)$, otherwise edge is backward, even if $r(u) = r(v)$. Ideally, rank assignment $r$ should not have backward edges, that is, for any $(u, v) \in E$ we should have $r(u) < r(v)$. However, this is only possible when $G$ is a DAG. For a more general case, we assume that we are given a penalty function $p$, mapping an integer to a real number. The penalty for a single edge $(u, v)$ is then equal to $p(d)$, where $d = r(u) - r(v)$. If $p(d) = 0$, whenever $d < 0$, then the forward edges will receive 0 penalty.

We highlight two penalty functions. The first one assigns a constant penalty to each backward edge,

$$p_c(d) = \begin{cases} 1 & \text{if } d \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

The second penalty function assigns a linear penalty to each backward edge,

$$p_l(d) = \max(0, d + 1)$$

For example, an edge $(u, v)$ for which $r(u) = r(v)$ is penalized by $p_l(r(u) - r(v)) = 1$, the penalty is equal to 2 if $r(u) = r(v) + 1$, and so on.

Given a penalty function and a rank assignment we can now define the score for the ranking to be the sum of the weighted penalties.

**Definition 1:** Assume a weighted directed graph $G = (V, E, w)$ and a rank assignment $r$. Assume also a cost function $p$ mapping an integer to a real number. We define a score for a rank assignment to be

$$q(G, r, p) = \sum_{(u, v) \in E} w(u, v) p(r(u) - r(v)).$$

We will refer the score $q(G, r, p)$ as agony.

**Example 2:** Consider the left ranking $r_1$ of a graph $G$ given in Figure 2. This ranking has 5 backward edges, consequently, the penalty is $q(G, r_1, p_c) = 5$. On the other hand, there are with 2 edges, $(i, a)$ and $(e, g)$, with the agony of 1. Moreover, 2 edges has agony of 2 and $(d, b)$ has agony of 3. Hence, agony is equal to

$$q(G, r_1, p_l) = 2 \times 1 + 2 \times 2 + 1 \times 3 = 10.$$  

The agony for the right ranking $r_2$ is $q(G, r_2, p_l) = 7$. Consequently, $r_2$ yields a better ranking in terms of agony.

We can now state our main optimization problem.

**Problem 1:** Given a graph $G = (V, E, w)$, a cost function $p$, and an integer $k$, find a rank assignment $r$ minimizing $q(r, G)$ such that $0 \leq r(v) \leq k - 1$ for every $v \in V$. We will denote the optimal score by $q(G, k, p)$.

We should point out that we have an additional constraint by demanding that the rank assignment may have only $k$ distinct values, that is, we want to find at most $k$ groups. Note that if we assume that the penalty function is non-decreasing and does not penalize the forward edges, then setting $k = |V|$ is equivalent of ignoring the constraint. This is the case since there are at most $|V|$ groups and we can safely assume that these groups obtain consecutive ranks. However, an optimal solution may have less than $k$ groups, for example, if $G$ has no edges and we use $p_l$ (or $p_c$), then a rank assigning each vertex to 0 yields the optimal score of 0. We should also point out that unlike with $p_c$, it is possible that the ranking minimizing agony must yield a partition which contains non-singleton groups.

It is easy to see that minimizing $q(G, p_c)$ is equivalent to finding a directed acyclic subgraph with as many edges as possible. This is known as **Feedback Arc Set** (FAS) problem, which is **NP-complete** [1].
On the other hand, if we assume that $G$ has unit weights, and set $k = |V|$, then minimizing agony has a polynomial-time $O(m^2)$ algorithm [6] [17].

### III. Computing Agony

In this section we present a technique for minimizing agony, that is, solving Problem 1 using $p_l$ as a penalty. In order to do this we show that this problem is in fact a dual problem of the known graph problem, closely related to the minimum cost max-flow problem.

#### A. Agony is a dual problem of Circulation

Minimizing agony is closely related to a circulation problem, where the goal is to find a circulation with a minimal cost satisfying certain balance equations.

**Problem 2 (Circulation):** Assume a directed graph $G = (V, E, s, c)$ with weights and capacities on edges. Find a flow $f$ such that $0 \leq f(e) \leq c(e)$ for every $e \in E$ and

$$\sum_{(u,v) \in E} f(u,v) = \sum_{(u,v) \in E} f(u,v), \quad \text{for every } v \in V$$

minimizing

$$\sum_{(u,v) \in E} s(u,v)f(u,v).$$

We denote the above sum as $\text{circ}(G)$.

This problem is known as capacitated circulation problem, and can be solved in $O(m \log n (m + n \log n))$ time with an algorithm presented by Orlin [13]. We should stress that we allow $s$ to be negative, otherwise the optimal solution would be a zero flow. We also allow capacities for certain edges to be infinite, which simply means that $f(e) \leq c(e)$ is not enforced, if $c(e) = \infty$.

In order to transform the problem of minimizing agony to the capacitated circulation problem, assume that we are given a graph $G = (V, E, w)$ and an integer $k$. We define a graph $H = (W, F, s, c)$ as follows. The vertex set $W$ consists of 2 groups: (i) $|V|$ vertices, each vertex corresponding to a vertex in $G$ (ii) 2 additional vertices $\omega$ and $\alpha$. For each edge $e = (u, v) \in E$, we add an edge $f = (u, v)$ to $F$. We set $c(f) = w(e)$ and $s(e) = -1$. We add edges $(\omega, v)$ and $(\alpha, v)$ for every $v \in V$ with a weight of 0 and a capacity of $\infty$, and finally we add $(\omega, \alpha)$ with $s(\omega, \alpha) = k - 1$ and $c(\omega, \alpha) = \infty$. We will denote this graph by $H(G, k) = H$.

Example 3: Consider $G = (V, E)$, a graph with 4 vertices and 4 edges, given in Figure 3. Set cardinality constraint $k = 4$. In order to construct $H(G, k)$ we add two additional vertices $\alpha$ and $\omega$ to enforce the cardinality constraint $k$. We set edge costs to $-1$ and edges capacities to be the weights of the input graph. We connect $\alpha$ and $\omega$ with $a, b, c,$ and $d$, and finally we connect $\omega$ to $\alpha$. The resulting graph is given in Figure 2:

The following proposition shows the connection between the agony and the capacitated circulation problem.

**Proposition 4:** Assume a weighted directed graph $G = (V, E, w)$ and an integer $k$. Then

$$q(G, k, p_l) = -\text{circ}(H(G, k)).$$

**Proof:** Let $H = (W, F, c, b) = H(G, k)$. To prove this result we will show that computing $\text{circ}(H)$ is a linear program, whose dual corresponds to optimizing agony. In order to do this, we first express a general circulation problem as a linear program,

minimize $\sum_{(u,v) \in F} s(u,v)f(u,v)$

subject to

$$\sum_{(v,u) \in W} f(v,u) = \sum_{(u,v) \in W} f(u,v), \quad \text{for every } v \in W,$$

$$c(u,v) \geq f(u,v) \geq 0, \quad \text{for every } (u,v) \in F.$$

This program has the following dual problem,

maximize $-\sum_{(u,v) \in F} \eta(u,v)c(u,v)$

subject to

$$\pi(u) - \pi(v) - \eta(u,v) \leq s(u,v), \quad \text{if } c(u,v) < \infty,$$

$$\pi(u) - \pi(v) \leq s(u,v), \quad \text{if } c(u,v) = \infty,$$

$$\eta(u,v) \geq 0,$$

which is optimized over the variables $\pi$ and $\eta$.

Let us write this dual specifically for $H$,

maximize $-\sum_{(u,v) \in F} \eta(u,v)c(u,v)$

subject to

$$\pi(u) - \pi(v) + 1 \leq \eta(u,v), \quad \text{for every } (u,v) \in E,$$

$$\eta(u,v) \geq 0, \quad \text{for every } (u,v) \in E,$$

$$\pi(v) \geq \pi(\alpha), \quad \text{for every } v \in V,$$

$$\pi(v) \leq \pi(\omega), \quad \text{for every } v \in V,$$

$$\pi(\omega) \leq \pi(\alpha) + k - 1.$$

Let $C$ be the optimal value of the dual problem. The standard LP dual theory, see Theorem 3.1 in [14], states that $C = \text{circ}(H)$. We need to show that $C = -q(G, k, p_l)$.

Let us write $r(v) = \pi(v) - \pi(\alpha)$. Then, since $\eta$ needs to be as small as possible, we have

$$\eta(u,v) = \max(\pi(u) - \pi(v) + 1, 0) = p_l(r(u) - r(v))$$
This leads to

$$\text{maximize} - \sum_{w \in W} w(u, v) p_u (r(u) - r(v)) \quad \text{such that}$$

$$0 \leq r(v) \leq k - 1.$$

This optimization problem is equal to solving $q(G, k, p_i)$, with the relaxation that we allow $r$ to have non-integer values. To complete the proof we need to show that there is an optimal integer-valued dual solution $\pi$ and $\eta$. This result follows from the fact that the constraints of the dual form an arc-vertex incidence matrix, which is known to be totally unimodular [14, Corollary of Theorem 13.3]. Since $s(u, v)$ are integers, Theorem 13.2 in [14] implies that there is an optimal solution with integer-valued $\pi$, which completes the proof. $\square$

B. Algorithm for minimizing agony

Proposition 3 states that we can compute agony but it does not provide direct means to discover an optimal rank assignment. However, a closer look at the proof reveals that minimizing agony is a dual problem of circulation. That is, if we were to solve the dual optimization problem given in Equation 1 then we can extract the optimal ranking from the dual parameters $\pi$ by setting $r(v) = \pi(v) - \pi(\alpha)$ for $v \in V$, where $\alpha$ is the special vertex added during the construction of $H$.

Luckily, the algorithms for solving circulation by Edmonds and Karp [2] or by Orlin [13] in fact solve Equation 1 and are guaranteed to have integer-valued solution as long as the capacities $s(u, v)$ are integers, which is the case for us.

If we are not enforcing the cardinality constraint, that is, we are solving $q(G, k)$ with $k = |V|$, we can obtain a significant speed-up by decomposing $G$ to strongly connected components, and solve ranking for individual components.

Proposition 5: Assume a graph $G$, and set $k = |V|$. Let $\{C_i\}$ be the strongly connected components of $G$, ordered in a topological order. Let $r_i$ be the ranking minimizing $q(G(C_i), |C_i|)$. Then the ranking $r(v) = r_i(v) + b_i$, where $C_i$ is the component containing $v$, yields the optimal score $q(G, k)$.

Proof: Note that $\max r(v) \leq k$, hence $r$ is a valid ranking. Let $r'$ be the ranking minimizing $q(G, k)$. Let $r'_i$ be the projection of the ranking to $C_i$. Then

$$q(G, r') \geq \sum_{i=1}^{\ell} q(G(C_i), r'_i) \geq \sum_{i=1}^{\ell} q(G(C_i), r_i) = q(G, r),$$

where the last equality holds because any cross-edge between the components is a forward edge. $\square$

IV. ALTERNATIVE PENALTY FUNCTIONS

We have shown that we can find ranking minimizing edge penalties $p_i$ in polynomial time. In this section we consider alternative penalties. More specifically, we consider convex penalties which are solvable in polynomial time, and concave penalties which are NP-complete.

A. Convex penalty function

We say that the penalty function is convex if $p(x) \leq (p(x - 1) + p(x + 1))/2$ for every $x \in \mathbb{Z}$.

Let us consider a penalty function that can be written as

$$p_s(x) = \sum_{i=1}^{\ell} \max(0, \alpha_i (x - \beta_i)),$$

where $\alpha_i > 0$ and $\beta_i \in \mathbb{Z}$ for $1 \leq i \leq \ell$. This penalty function is convex. On the other hand, if we are given a convex penalty function $p$ such that $p(x) = 0$ for $x < \ell$, then we can safely assume that an optimal rank assignment will have values between 0 and $|V| - 1$. We can define a penalty function $p_s$ with $\ell \leq |V|$ terms such that $p_s(x) = p(x)$ for $x < |V|$. Consequently, finding an optimal rank assignment using $p_s$ will also yield an optimal rank assignment with respect to $p$.

Note that $p_1$ is a special case of $p_s$. This hints that we can solve $q(G, k, p_s)$ with a technique similar to the one given in Section III In order to do this, assume a graph $G = (V, E, w)$ and an integer $k$. Let $n = |V|$ and $m = |E|$. We define a graph $H = (W; F, c, b)$ as follows. The vertex set $W$ consists of $2$ groups: (i) $n$ vertices, each vertex corresponding to a vertex in $G$ (ii) $2$ additional vertices $\alpha$ and $\omega$. For each edge $e = (v, w) \in E$, we add $\ell$ edges $f_{ij} = (u, v)$ to $F$. We set $s(f_{ij}) = \beta_i$ and $c(f_{ij}) = \alpha_i w(e)$. We add edges $(v, \omega)$ and $(\omega, v)$ for every $v \in V$ with 0 weight and infinite capacity. Finally we add $(\omega, \alpha)$ with $c(t, s) = k - 1$. We denote this graph by $H(G, k, p_s) = H$.

![Example 6: A graph and its penalty function](image)

Example 6: Consider a graph $G$ given in Figure 4 and a penalty function $p_s(d) = \max(0, d+1) + 2 \max(0, d-3)$. The graph $H = H(G, 3, p_s)$ has 5 vertices, the original vertices and the two additional vertices. Each edge in $G$ results in two edges in $H$. This gives us 6 edges plus the 7 edges adjacent to $\alpha$ or $\omega$. The graph $H$ without $\alpha$ and $\omega$ is given in Figure 4.

A proposition similar to Proposition 3 holds.

Proposition 7: Assume a weighted directed graph $G = (V, E, w)$, an integer $k$, and a penalty $p_s$ cost. Then

$$q(G, k, p_s) = -\text{circ}(H(G, k, p_s)).$$

Note that this proposition is a generalization of Proposition 3. If we set $\ell = 1$, $\alpha_1 = 1$, and $\beta_1 = -1$, then $H(G, k, p_s) = H(G, k)$, and Proposition 7 is equivalent to Proposition 3. The proof of Proposition 4 is essentially equivalent to the proof of Proposition 3 and we omit the proof for brevity. More importantly, we can discover the optimal ranking by solving the capacitated circulation problem, similarly as we did when minimizing agony.
Finally, let us address the computational complexity of the problem. The circulation graph \( H(G, k, p_v) \) will have \( n + 2 \) vertices and \( \ell m + n \). If the penalty function \( p \) is convex, then we need at most \( \ell = n \) functions to represent \( p \) between the range of \([0, n - 1]\). Moreover, if we enforce the cardinality constraint \( k \), we need only \( k \) components. Consequently, we will have at most \( d m + n \) edges where \( d = \min(k, \ell, n) \) for \( p_v \), and \( d = \min(k, n) \) for a convex penalty \( p \). This gives us computational time of \( O(dm \log n(dm + n \log n)) \).

B. Concave penalty function

We have shown that we can solve Problem \(^1\) for any convex penalty. Let us consider concave penalties, that is, penalties for which \( p(x) \geq (p(x - 1) + p(x + 1))/2 \). There is a stark difference compared to the convex penalties as the minimization problem becomes computationally intractable.

Proposition 8: Assume a monotonic penalty function \( p : \mathbb{Z} \rightarrow \mathbb{R} \) such that \( p(x) = 0 \) for \( x < 0 \), \( p(2) > p(1) \), and there is an integer \( t \) such that

\[
p(t) > \frac{p(t - 1) + p(t + 1)}{2}
\]

and

\[
\frac{p(s)}{s + 1} \geq \frac{p(y)}{y + 1}
\]

for every \( 0 \leq s \leq y \) and \( y \in [t - 1, t, t + 1] \). Then, determining whether \( q(G, k, p) \leq \sigma \) for a given graph \( G \), integer \( k \), and threshold \( \sigma \) is an \( \text{NP} \)-complete problem.

We provide the proof in Appendix.

While the conditions in Proposition 8 seem overly complicated, they are quite easy to satisfy. Assume that we are given a penalty function that is concave in \([ -1, \infty) \), and \( p(-1) = 0 \). Then due to concavity we have

\[
\frac{p(x)}{x + 1} \geq \frac{p(x + 1)}{x + 2}, \quad \text{for} \quad x \geq 0
\]

This leads to the following corollary.

Corollary 9: Assume a monotonic penalty function \( p : \mathbb{Z} \rightarrow \mathbb{R} \) such that \( p(x) = 0 \) for \( x < 0 \), \( p(2) > p(1) \), and \( p \) is concave and non-linear in \([-1, s] \) for some \( s \geq 1 \). Then, determining whether \( q(G, k, p) \leq \sigma \) for a given graph \( G \), integer \( k \), and threshold \( \sigma \) is an \( \text{NP} \)-complete problem.

Note that we require \( p \) to be non-linear. This is needed so that the proper inequality in Equation 2 is satisfied. This condition is needed since \( p_1 \) satisfies every other requirement.

Corollary 8 covers many penalty functions such as \( p(x) = \sqrt{x + 1} \) or \( p(x) = \log(x + 2) \), for \( x \geq 0 \). Note that function needs to be convex only in \([-1, s] \) for some \( s \geq 1 \). At extreme, \( s = 1 \) in which case \( t = 0 \) satisfies the conditions in Proposition 8.

V. SELECTING CANONICAL SOLUTION

A rank assignment minimizing agony may not be unique. In fact, consider a graph \( G \) with no edges, then any ranking will have the optimal score of 0. Moreover, if the input graph \( G \) is a DAG, then any topological sorting of vertices will yield the optimal score of 0.

In this section we introduce a technique to select a unique optimal solution. The idea here is to make the ranks as small as possible without compromising the optimality of the solution. More specifically, let us define the following relationship between to rankings.

Definition 10: Given two rank assignments \( r \) and \( r' \), we write \( r \preceq r' \) if \( r(x) \leq r'(x) \) for every \( x \).

The following proposition states that there exists exactly one ranking with the optimal score that is minimal with respect to the \( \preceq \) relation. We will refer to this ranking as canonical ranking.

Proposition 11: Given a graph \( G \) and an integer \( k \), there exists a unique optimal rank assignment \( r \) such that \( r \preceq r' \) for every optimal rank assignment \( r' \).

The proof of this proposition is given in Appendix.

Canonical ranking has many nice properties. The canonical solution for a graph without edges assigns rank 0 to all vertices. More generally, if \( G = (V, E) \) is a DAG, then the source vertices \( S \) of \( G \) will receive a rank of 0, the source vertices of \( G(V \setminus S) \) will receive a rank of 1, and so on. For general graphs we have the following proposition.

Proposition 12: Let \( r \) be the canonical ranking. Then \( r \) has the least distinct rank values among all optimal solutions.

In other words, the partition of \( V \) corresponding to the canonical ranking has the smallest number of groups.

Our next step is to provide an algorithm for discovering canonical ranking. In order to do so we assume that we have already found one optimal, not necessarily canonical, solution. We will first define the following graph.

Definition 13: Assume a weighted graph \( G = (V, E, w) \). Let \( H = H(G, k) \) be the circulation graph, let \( f \) be the optimal circulation flow and let \( r \) be the optimal ranking. We define a residual graph \( R = (W, F, t) \) with weighted edges as follows. The vertices \( W \) consists of \( V \) and an additional vertex \( q \). Let \( (u, v) \in E \). If \( f(u, v) > 0 \), we add an edge \((u, v)\) to \( F \) with a cost of \( t(u, v) = r(v) - r(u) + 1 \). If \( f(u, v) < c(u, v) \), we add an edge \((q, v)\) to \( F \) with a cost of \( t(q, v) = r(v) - r(q) - 1 \). Finally, we add edges \((q, v)\), weighted by \( t(q, v) = r(v) \), for each \( v \in V \). However, if \( f(v, t) > 0 \), we set \( t(q, v) = 0 \). We denote this graph by \( R(G, f, r) \).

Once, we have computed the residual graph, we simply compute the shortest path distance from \( q \) and subtract the distance from the optimal ranking, see Algorithm 1.

Algorithm 1: CANON(G), computes canonical optimal solution

\[
r, f \leftarrow \text{optimal ranking and flow of CIRCULATION};
R \leftarrow R(G, f, r);
d(v) \leftarrow \text{shortest weighted distance from } s \text{ to } v \text{ in } R;
\text{foreach } v \in V \text{ do } r'(v) \leftarrow r(v) - d(v);
\text{return } r';
\]

Proposition 14: Algorithm CANON returns canonical solution with optimal score.

We give the proof of this proposition in Appendix.
Proposition states that to compute the canonical ranking it is enough to form the residual graph, compute the shortest edge distances \(d(v)\) from the vertex \(v\), and subtract them from the input ranking. The computational complexity of these steps is \(O(m + n \log n)\). Moreover, this proposition holds for a more general convex penalty function, described in Section V-A.

VI. RELATED WORK

The problem of discovering the rank of an object based on its dominating relationships to other objects is a classic problem. Perhaps the most known ranking method is Elo rating devised by Elo [3], used to rank chess players. In similar fashion, Jameson et al. [8] introduced a statistical model, where the likelihood of the vertex dominating other is based on the difference of their ranks, to animal dominance data.

Maiya and Berger-Wolf [11] suggested an approach for discovering hierarchies, directed trees from weighted graphs such that parent vertices tend to dominate the children. To score such a hierarchy the authors propose a statistical model where the probability of an edge is high between a parent and a child. To find a good hierarchy the authors employ a greedy heuristic.

The technical relationship between our approach and the previous studies on agony by Gupte et al. [6] and Tatti [17] is a very natural one. The authors of both papers demonstrate that minimizing agony in an unweighted graph is a dual problem to finding a maximal eulerian subgraph, a subgraph in which, for each vertex, the same number of outgoing edges and the number of incoming edges is the same. Discovering the maximum eulerian subgraph is a special case of the capacitated circulation problem, where the capacities are set to 1. However, the algorithms in [6, 17] are specifically designed to work with unweighted edges. Consequently, if our input graph edges or we wish to enforce the cardinality constraint, we need to solve the problem using the capacitated circulation solver.

The stark difference of computational complexities for different edge penalties is intriguing: while we can compute agony and any other convex score in polynomial-time, minimizing the concave penalties is NP-hard. Minimizing the score \( q(G, k, p_r) \) is equivalent to FEEDBACK ARC SET (FAS), which is known to be APX-hard with a coefficient of \( c \approx 1.3606 \) [11]. 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 this paper we have shown that minimizing concave penalty is NP-hard. An interesting theoretical question is whether this optimization problem is also APX-hard, and is it possible to develop an approximation algorithm.

Role mining, where vertices are assigned different roles based on their adjacent edges, and other features, has received some attention. Henderson et al. [2] studied assigning roles to vertices based on its features while McCullam et al. [12] assigned topic distributions to individual vertices. A potential direction for a future work is to study whether the rank obtained from minimizing agony can be applied as a feature in role discovery.

VII. EXPERIMENTS

In this section we present our experimental evaluation. Our main focus of the experiments is practical computability of the weighted agony.

A. Datasets and setup

For our experiments we took 10 large networks from SNAP repository [9]. In addition, for illustrative purposes, we used two small datasets: NFL, consisting of National Football League teams. We created an edge \((x, y)\) if team \(x\) has scored more points against team \(y\) during 2014 regular season, we assign the weight to be the difference between the points. Since not every team plays against every team, the graph is not a tournament. Reef, a food web of guilds of species [15], available at [10]. The dataset consisted of 3 food webs of coral reef systems: The Cayman Islands, Jaimaica, and Cuba. An edge \((x, y)\) appears if a guild \(x\) is known to prey on a guild \(y\). Since the guilds are common among all 3 graphs, we combined the food webs into one graph, and weighted the edges accordingly, that is, each edge received a weight between 1 and 3.

The sizes of the graphs, along with the sizes of the largest strongly connected component, are given in the first 4 columns of Table 1.

The 3 Higgs and NFL graphs had weighted edges, and for the remaining graphs we assigned a weight of 1 for each edge. We removed any self-loops as they have no effect on the ranking, as well as any singleton vertices.

For each dataset we computed the agony using CANON, given in Algorithm 1. To solve the CIRCULATION problem we used algorithm by Edmonds and Karp [2]. While the algorithm by Orlin [13] has better theoretical bounds, the pathological case that is solved by Orlin [13] does not occur in practice for our datasets. For the unweighted graphs we also computed the agony using RELIEF, an algorithm suggested by Tatti [17]. Note that this algorithm, nor the algorithm by Gupte et al. [6], does not work for weighted graphs nor when the cardinality constraint \(k\) given in Problem 1 is enforced. We implement both algorithms in C++ and performed experiments using a Linux-desktop equipped with a Opteron 2220 SE processor.

B. Results

Let us begin by studying running times given in Table 1. We report the running times of our approach with and without the strongly connected component decomposition as suggested by Proposition 5 and compare it against the baseline RELIEF, whenever possible. Note that we can use the decomposition only if we do not enforce the cardinality constraint.

Our first observation is that the decomposition always helps to speed up the algorithm. In fact, this speed-up may be dramatic, if the size of the strongly connected component is significantly smaller than the size of the input graph, for example, with HiggsRetweet. The number of iterations of iterations needed to solve the circulation problem is significantly smaller than the worst case scenario of \(O(m \log n)\), making this algorithm feasible even for large graphs. Interestingly enough, this number increases in some cases, when using the

\(^1\)The source code is available at [http://users.ics.aalto.fi/ntatti/agony.zip]
decomposition, but a single iteration may be significantly faster when the decomposition is used.

The baseline RELIEF is almost always faster than the approach, even though the computational complexity guarantees, $O(m^2)$ for RELIEF and $O(m^2 \log n + mn \log n)$ for CANON, are practically the same. Nevertheless, the difference between the computation times is less than the order of magnitude. Moreover, we should stress that RELIEF is specifically designed to work with unweighted graphs, and without the cardinality constraints.

Our next step is to study the effect of the constraint $k$, the maximum number of different rank values. We see from the 6th column in Table I that despite having large number of vertices, that the optimal rank assignment has low number of groups, typically around 10–20 groups, even if the cardinality constraint is not enforced. This suggests the following strategy to compute the agony when the cardinality constraint is not enforced. We should stress that RELIEF is specifically designed to work with unweighted graphs, and without the cardinality constraints.

Let us now consider agony as a function of $k$, which we have plotted in Figure 5 for Gnutella and WikiVote graphs. We see that for these datasets that agony remains relatively constant as we decrease $k$, and starts to increase more prominently once we consider assignments with $k \leq 5$.

As we can see lower values of $k$ are computationally more taxing to solve even though they have the same theoretical bounds on running time. The main reason for more additional computational burden is that solving agony with small values of $k$ requires more iterations, as we can see in Figure 6.

![Figure 5: Agony as a function of the constraint $k$ for Gnutella and WikiVote datasets.](image)

| Name          | $|V|$ | $|E|$ | $|V'|$ | $|E'|$ | $|\text{ranks}|$ | with SCC decomp. | w/o SCC decomp. |
|---------------|------|------|-------|-------|----------------|-----------------|------------------|
|               | iter | time | iter  | time  | base           | iter           | time            |
| Amazon        | 9375 | 6h24m| 3.156 | 7h29m| 4h27m         | 45s            | 45s             | 45s             |
| Gnutella      | 83   | 8    | 136   | 45    |                |                 |                 |                 |
| EmailEU       | 1.380| 10m  | 7.876 | 3h8m | 2m             |                 |                 |                 |
| Epinions      | 4.460| 49m  | 3.552 | 1h5m | 20m            |                 |                 |                 |
| Dashdot       | 3.527| 1h38m| 3.717 | 2h21m| 1h5m           |                 |                 |                 |
| WebGoogle     | 52.119| 8h50m| 6.464 | 26h43m| 2h32m         |                 |                 |                 |
| WikiVote      | 738  | 43s  | 1.079 | 5m    | 7x             |                 |                 |                 |
| Nl            | 32   | 205  | 157   | 22ms |                |                 |                 |                 |
| Reef          | 2974 | 0.2s | 6.990 | 10m   |                |                 |                 |                 |
| HiggsReply    | 5.488| 10m  | 44.820| 3h34m|                |                 |                 |                 |
| HiggsRetweet  | 11.846| 2m   | 44.560| 19h38m|              |                 |                 |                 |
| HiggsMention  | 7.38 | 43s  | 1.079 | 5m    | 7x             |                 |                 |                 |

Figure 6. Execution time as a function of the constraint $k$ for Gnutella and WikiVote datasets.

| Name       | $|V|$ | $|E|$ | $|V'|$ | $|E'|$ | $|\text{ranks}|$ | with SCC decomp. | w/o SCC decomp. |
|------------|------|------|-------|-------|----------------|-----------------|------------------|
| Nfl        | 32   | 205  | 157   | 22ms |                |                 |                 |                 |
| Reef       | 2974 | 0.2s | 6.990 | 10m  |                |                 |                 |                 |
| HiggsReply | 5.488| 10m  | 44.820| 3h34m|                |                 |                 |                 |
| HiggsMention| 11.846| 2m   | 44.560| 19h38m|              |                 |                 |                 |

Figure 7. Number of iterations needed by capacitated circulation solver as a function of the constraint $k$ for Gnutella and WikiVote datasets.

| Rank | Teams                                      |
|------|--------------------------------------------|
| 1    | DEN BAL NE DAL SEA PHI KC GB PIT           |
| 2    | STL NYG MIA CAR NO SD MIN CIN BUF DET IND HOU SF ARI |
| 3    | WSH OAK TB JAX TEN CLE ATL NYI CHI         |

Let us look on the ranking that we obtained from Nfl dataset using $k = 3$ groups, given in Table II. We see from the results that the obtained ranking is very sensible. 7 of 8 teams in the top group consists of playoff teams of 2014 season, while the bottom group consists of teams that have a significant losing record.

![Figure 7: Number of iterations needed by capacitated circulation solver as a function of the constraint $k$ for Gnutella and WikiVote datasets.](image)
Finally, let us look on rankings obtained Reef dataset. The graph is in fact a DAG with 19 groups. To reduce the number of groups we rank the guilds into 4 groups. The condensed results are given in Table III. We see that the top group consists of large fishes and sharks, the second group contains mostly smaller fishes, a large portion of the third group are crustacea, while the last group contains the bottom of the food chain, planktons and algae. We should point out that this ranking is done purely on food web, and not on type of species. For example, cleaner crustacea is obviously very different than plankton. Yet cleaner crustacea only eats planktonic bacteria and micro-detrivores while being eaten by many other guilds. Consequently, it is ranked in the bottom group.

VIII. CONCLUDING REMARKS

In this paper we studied the problem of discovering a hierarchy in a directed graph that minimizes agony. We introduced several natural extensions: (i) we demonstrated how to compute the agony for weighted edges, and (ii) how to limit the number of groups in a hierarchy. Both extensions cannot be handled as pointed out in Section IV-A we demonstrated how to compute agony for such large graphs is sensible only if it needs to be done few times, ideally only once. A fruitful direction is to develop an algorithm for such cases. Ideally, such an algorithm is fast, preferably linear-time, and yields an approximation guarantee.

REFERENCES

[1] I. Dinur and S. Safra. On the hardness of approximating vertex cover. Annals of Mathematics, 162(1):439–485, 2005.
[2] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. Journal of ACM, 19(2):248–264, 1972.
[3] A. E. Elo. The rating of chessplayers, past and present. Arco Pub., 1978.
[4] G. Even, J. (Seffi) Naor, B. Schieber, and M. Sudan. Approximating minimum feedback sets and multicut in directed graphs. Algorithmica, 20(2):151–174, 1998.
[5] M. Garey and D. Johnson. Computers and intractability: a guide to the theory of NP-completeness. WH Freeman & Co., 1979.
[6] M. Gupte, P. Shankar, J. Li, S. Muthukrishnan, and L. Ifode. Finding hierarchy in directed online social networks. In Proceedings of the 20th International Conference on World Wide Web, pages 557–566, 2011.
[7] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li. Rolx: Structural role extraction & mining in large graphs. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1231–1239, 2012.
[8] K. A. Jameson, M. C. Appleby, and L. C. Freeman. Finding an appropriate order for a hierarchy based on probabilistic dominance. Animal Behaviour, 57:991–998, 1999.
[9] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data Jan. 2015.
[10] L. Macchia, F. Bonchi, F. Gullo, and L. Chiaramondi. Mining summaries of propagations. In IEEE 13th International Conference on Data Mining, pages 498–507, 2013.
[11] A. S. Maiya and T. Y. Berger-Wolf. Inferring the maximum likelihood hierarchy in social networks. In Proceedings IEEE CSE ’09, 12th IEEE International Conference on Computational Science and Engineering, pages 245–250, 2009.
[12] A. McCallum, X. Wang, and A. Corrada-Emmanuel. Topic and role discovery in social networks with experiments on enron and academic email. J. Artif. Int. Res., 30(1):249–272, 2007.
[13] J. B. Orlin. A faster strongly polynomial minimum cost flow algorithm. Operations Research, 41(2), 1993.
APPENDIX

A. Proof of Proposition \[\text{[15]}\]

Proof: The problem is obviously in \textbf{NP}. To prove the completeness we will provide reduction from \textsc{Maximum Cut} \[\text{[14]}\]. An instance of \textsc{Maximum Cut} consists of an undirected graph, and we are asked to partition vertices into two sets such that the number of cross edges is larger or equal than the given threshold \(\sigma\).

Note that the conditions of the proposition guarantee that \(p(0) > 0\).

Assume that we are given an instance of \textsc{Maximum Cut}, that is, an undirected graph \(G = (V, E)\) and a threshold \(\sigma\). Let \(m = |E|\). Define a weighted directed graph \(H = (W, F, w)\) as follows. Add \(V\) to \(W\). For each edge \((u, v) \in E\), add a path with \(t\) intermediate vertices from \(u\) to \(v\), the length of the path is \(t + 2\). Add also a path in reversed direction, from \(v\) to \(u\). Set edge weights to be 1. Add 4 special vertices \(\alpha_1, \ldots, \alpha_4\). Add edges \((\alpha_i, \alpha_{i+1})\), for \(i = 1, \ldots, 3\) with a weight of

\[
C = 2B\dfrac{p(0)}{p(2) - p(1)}, \quad \text{where} \quad B = 2(t + 1)m.
\]

Add edges \((\alpha_i, \alpha_{i+1})\), for \(i = 1, \ldots, 3\) with a weight of

\[
D = 4Cp(1)/p(0) + B.
\]

Add edges \((\alpha_0, v)\) and \((v, \alpha_4)\), for each \(v \in V\), with a weight of \(D\).

Let \(r\) be the optimal ranking for \(H\). We claim that \(r(\alpha_i) = i - 1\), and \(r(v) = 1, 2\) for each \(v \in V\). To see this, consider a ranking \(r'\) such that \(r'(\alpha_i) = i - 1\) and the rank for the remaining vertices is 2. The score of this rank is 3.

\[
q(H, r') = 3Cp(1) + 2(t + 1)m p(0) = 3Cp(1) + Bp(0).
\]

Let \((u, v) \in F\) with the weight of \(D\). If \(r(u) \geq r(v)\), then the score of \(r\) is at least \(Dp(0) = 4Cp(1) + Bp(0)\) which is more than \(q(H, r')\). Hence, \(r'(u) < r'(v)\). Let \((u, v) \in F\) with the weight of \(C\). Note that \(r(u) \geq r(v) + 1\). Assume that \(r(u) \geq r(v) + 2\). Then the score is at least 3.

\[
3Cp(1) + C(p(2) - p(1)) = 3Cp(1) + 2Bp(0),
\]

which is a contradiction. This guarantees that \(r(\alpha_i) = i - 1\), and \(r(v) = 1, 2\) for each \(v \in V\).

Consider \((u, v) \in E\) and let \(u = x_0, \ldots, x_{t+1} = v\) be the corresponding path in \(H\). Let \(d_i = r(x_i) - r(x_{i+1})\) and set \(\ell = t + r(u) - r(v)\). Let \(P = \sum p(d_i)\) be the penalty contributed by this path. Note that \(P \leq p(\ell)\), a penalty that we achieve by setting \(r(x_i) = r(x_{i+1}) + 1\) for \(i = 1, \ldots, t\). This implies that \(d_i \leq \ell\). The condition of the proposition now implies

\[
P = \sum_{i=0}^{t} p(d_i) = \sum_{i=0, d_i \geq 0}^{t} p(d_i) \geq \sum_{i=0}^{t} \max(d_i + 1, 0) - p(\ell) \geq \sum_{i=0}^{t} d_i + 1 - p(\ell) \geq \frac{p(\ell)}{\ell + 1} \left(\sum_{i=0}^{t} 1 + r(x_i) - r(x_{i+1})\right) = \frac{p(\ell)}{\ell + 1} (t + 1 + r(u) - r(v)) = p(\ell).
\]

This guarantees that \(P = p(\ell)\).

Partition edges \(E\) into two groups,

\[
X = \{(u, v) \in E \mid r(u) = r(v)\}
\]

and

\[
Y = \{(u, v) \in E \mid r(u) \neq r(v)\}.
\]

Let \(\Delta = p(t - 1) - p(t + 1) - 2p(t)\). Note that concavity implies that \(\Delta < 0\). Then

\[
q(H, r) = 3C + |X|2p(t) + |Y|(p(t - 1) - p(t + 1)) = 3C + m2p(t) + |Y|(p(t - 1) - p(t + 1) - 2p(t)) = 3C + m2p(t) + |Y|\Delta.
\]

The first two terms are constant. Consequently, \(q(H, r)\) is optimal if and only if \(|Y|\), the number of cross-edges is maximal.

Given a threshold \(\sigma\), define \(\sigma' = 3C + m2p(t) + \Delta\sigma\). Then \(q(H, r) \leq \sigma'\) if and only if there is a cut of \(G\) with at least \(\sigma\) cross-edges, which completes the reduction.

B. Proof of Proposition \[\text{[16]}\]

Proof: Let \(R\) be the set of all rank assignments with optimal score. Define

\[
r(v) = \min \{r'(v) \mid r' \in R\}.
\]

Obviously \(r \preceq r'\) for every optimal rank assignment \(r'.\) We need to show that \(r\) produces optimal score. In order to do this, let us adopt all the variables given in the proof of Proposition \[\text{[15]}\] and consider the linear program and its dual. Let \(f\) be a solution the linear program corresponding to \textsc{Circulation} and let \(\Pi\) be the set of all \textsc{integer-valued} solutions to the dual program for which \(\pi(\alpha) = 0\). Define \(\pi(y) = \min \{\pi'(y) \mid \pi' \in \Pi\}\). First note that \(\pi\) is a feasible solution to the dual program. To see this, select \((u, v) \in F\) and let \(\theta \in \Pi\) such that \(\pi(v) = \theta(v)\). Then

\[
\pi(u) \leq \theta(u) \leq c(u, v) + \theta(v) = c(u, v) + \pi(v).
\]
which proves the feasibility of $\pi$. Our next step is to prove that $\pi \in \Pi$. The complementary slackness conditions, see Theorem 3.4 in [14] or Equation 3 in [13], state that a feasible solution $\theta$ is optimal if and only if $\theta(u) - \theta(v) = c(u, v)$ whenever $f(u, v) > 0$. Select $(u, v) \in F$ such that $f(u, v) > 0$. We have

$$\pi(u) = \min_{\theta \in \Pi} \theta(u) = \min_{\theta \in \Pi} c(u, v) + \theta(v) = c(u, v) + \pi(v) .$$

This proves the optimality of $\pi$. The proof of Proposition 4 shows that every solution in $\pi' \in \Pi$ corresponds to a rank in $r' \in R$ such that $r'(u) = \pi'(u)$ for every $u \in V$. Moreover, this correspondence is surjective. Hence, $\min_{r \in R} r(u) = \min_{\pi' \in \Pi} \pi'(u)$ for every $u \in V$. This implies that $r(u) = \pi(u)$, for $u \in V$, and shows that $r \in R$, yielding the optimal score.

C. Proof of Proposition 14

**Lemma 15:** Assume graph $G = (V, E, w)$ and let $f$ be the optimal flow for $H = H(G, k) = (W, F, c, s)$. Rank assignment $r$ satisfying $0 \leq r(v) \leq k - 1$ is optimal if and only if (i) $r(v) \geq r(u) + 1$ for every $(u, v) \in E$ with $f(u, v) < c(u, v)$, (ii) $r(v) \leq r(u) + 1$ for every $(u, v) \in E$ with $f(u, v) > 0$, (iii) $r(v) = 0$ if $f(\alpha, v) > 0$, and (iv) $r(v) = k - 1$ if $f(v, \omega) > 0$.

**Proof:** We will prove the lemma by using complementary slackness conditions, a classic tool from LP theory.

In order to do that, define $\pi(v) = r(v)$ for $v \in V$, $\pi(\alpha) = 0$ and $\pi(\omega) = k - 1$. Let $\eta(u, v) = p_1(r(u) - r(v))$. Let $C$ be the value of the objective of the dual program given in Equation 12 using $\pi$ and $\eta$.

Note that $C = -q(G, r, p_1)$. Consequently, in order for $r$ to be optimal, $\pi$ and $\eta$ needs to be optimal. Complementary slackness conditions, see Theorem 3.4 in [14], state that $C$ is optimal if and only if

1. $(c(u, v) - f(u, v))\eta(u, v) = 0$
2. $(\pi(u) - \pi(v) + 1 - \eta(u, v))f(u, v) = 0$
3. $(\pi(v) - \pi(\alpha))f(\alpha, v) = 0$
4. $(\pi(v) - \pi(\omega))f(v, \omega) = 0$

Eq. 5 is equivalent to Condition (iii) while Eq. 6 is equivalent to Condition (iv). Eq. 5 states that if $f(u, v) < c(u, v)$, then $\eta = 0$, which is equivalent to $r(v) \geq r(u) + 1$, which is equivalent to Condition (i). Eq. 6 states that if $f(u, v) > 0$, then $\eta(u, v) = 0$, which is equivalent to $r(u) + 1$, which is equivalent to Condition (ii).

**Proof of Proposition 14** Let $f$ be the optimal flow. Let $R = R(G, f, r) = (U, X, t)$ be the residual network.

Assume that $f(u, v) < c(u, v)$ for $e = (u, v) \in E$. Then $(u, v) \in X$ with $t(u, v) = r(v) - r(u) - 1$. As $d$ is the shortest edge distance, we have $d(u) \leq d(v) + t(u, v)$, which implies

$$r'(u) = r(v) - d(v) \geq r(v) - d(u) - t(u, v) = r(u) - d(u) + 1 = r'(u) + 1 .$$

Assume that $f(u, v) > 0$ for $e = (u, v) \in E$. Then $(u, v) \in X$ with $t(u, v) = r(u) - r(v) + 1$. As $d$ is the shortest edge distance, we have $d(u) \leq d(v) + t(u, v)$, which implies

$$r'(u) = r(v) - d(v) \geq r(v) - d(u) - t(u, v) = r(u) - d(u) + 1 = r'(u) - 1 .$$

Lemma 15 implies that $r'$ is optimal. This implies that $r^* \leq r'$, where $r^*$ is the canonical ranking.

To prove that $r' \leq r$, we will use induction over the parenthesis of shortest-path tree $T$. Select $v$ and let $p$ be its parent in $T$. If $p = q$, then $d(v) = r(v)$ and $r'(v) = 0 \leq r^*(v)$. Assume that $p \neq q$. By the induction assumption we have $r'(p) = r^*(p)$. By definition of $T$, we have

$$d(v) - d(p) = t(p, v) - r(p) + r(v).$$

If $(p, v)$ was added due to $f(p, v) < c(p, v)$, then $t(p, v) = r(v) - r(p) - 1$, which implies $r'(v) = r^*(p) + 1 = r^*(p) + 1$. Since $r^*$ is optimal, Lemma 15 implies that $r^*(v) \geq r^*(p) + 1 = r'(v)$.

If $(p, v)$ was added due to $f(p, v) > 0$, then $t(p, v) = r(p) - r(v) + 1$, which implies $r'(v) = r^*(p) - 1 = r^*(p) - 1$. Since $r^*$ is optimal, Lemma 15 implies that $r^*(p) \leq r^*(v) + 1$ from which follows that $r^*(v) \geq r'(v)$.

This proves that $r' \leq r^*$, and $r' = r^*$, which completes the proof.