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

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

The proofs are in the complete version of the paper.[1]

[1]http://research.ics.aalto.fi/dmg/

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

$$p_c(d) = \begin{cases} 1 & \text{if } d \geq 0 \\ 0 & \text{otherwise} \end{cases}, \qquad p_l(d) = \max(0, d + 1) \quad .$$

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 define the the score 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*

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

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

Example 2. *Consider the left ranking r_1 of a graph G given in Figure 1. This ranking has 5 backward edges: 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, 2 edges has agony of 2, and (d, b) has agony of 3. Hence, agony is equal to*

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

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.

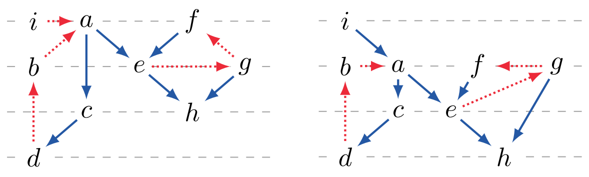

Figure 1. Toy graphs. Backward edges are represented by dotted lines, while the forward edges are represented by solid lines. Ranks are represented by dashed grey horizontal lines.

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 s and capacities c on edges. Find a flow f such that $0 \leq f(e) \leq c(e)$ for every $e \in E$ and*

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

minimizing

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

We denote the above sum as $circ(G)$.

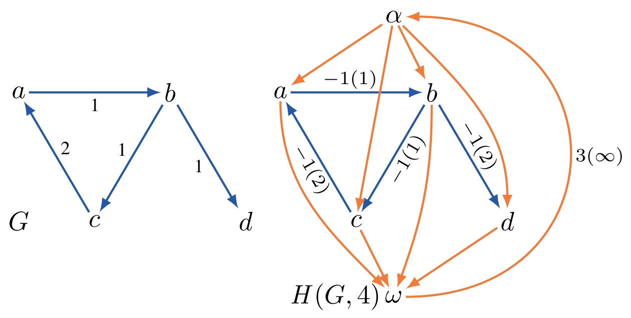

Figure 2. Toy graph G and the related circulation graph $H(G,4)$. Edge costs and capacities for (α, v) and (v, ω) are omitted to avoid clutter.

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, that is, $f(e) \leq c(e)$ 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 α and ω. 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 (v, ω) and (α, v) for every $v \in V$ with a weight of 0 and a capacity of ∞, and finally we add (ω, α) 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 2. Set cardinality constraint $k = 4$. In order to construct $H(G, k)$ we add two additional vertices α and ω 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 α and ω with a, b, c, and d, and finally we connect ω to α. 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*

$$q(G, k, p_l) = -circ(H(G, k)) \quad .$$

B. Algorithm for minimizing agony

Proposition 4 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.

Luckily, the algorithms for solving CIRCULATION by Edmonds and Karp [2] or by Orlin [13] solve the dual problem, 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 $q(G(C_i), |C_i|)$. Let $b_i = \sum_{j=1}^{i-1} |C_j|$. Then the ranking $r(v) = r_i(v) + b_i$, where C_i is the component containing v, yields the optimal score $q(G, k)$.*

IV. Alternative penalty functions

We have shown that we can find ranking minimizing edge penalties p_l 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 $x < 0$, 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_l 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. Set $n = |V|$ 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 α and ω. For each edge $e = (v, w) \in E$, we add ℓ edges $f_i = (u, v)$ to F. We set $s(f_i) = \beta_i$ and $c(f_i) = \alpha_i w(e)$. We add edges (v, ω) and (ω, v) for every $v \in V$ with 0 weight and infinite capacity. Finally we add (ω, α) with $c(t, s) = k - 1$. We denote this graph by $H(G, k, p_s) = H$.

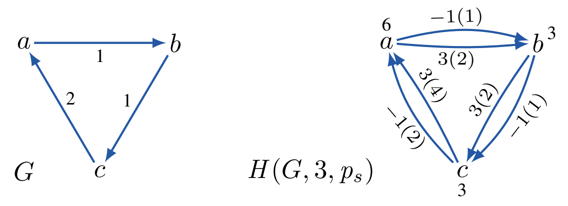

Figure 3. Toy graph G and the related circulation graph $H(G, 3, p_s)$. To avoid clutter the vertices α and ω and the adjacent edges are omitted.

Example 6. *Consider a graph G given in Figure 3 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 α or ω. The graph H without α and ω is given in Figure 3.*

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

$$q(G, k, p_s) = -circ(H(G, k, p_s)) \quad .$$

Note that this proposition is a generalization of Proposition 4. 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 4. The proof of Proposition 7 is essentially equivalent to the proof of Proposition 4, 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 $H(G, k, p_s)$ 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 $\ell = k$ components. Consequently, we will have at most $dm + n$, edges where $d = \min(k, \ell, n)$ for p_s, 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 $p : \mathbb{Z} \to \mathbb{R}$ such that $p(x) = 0$ for $x < 0$, $p(2) > p(1)$, and there is an integer t such that*

$$p(t) > \frac{p(t-1) + p(t+1)}{2} \tag{1}$$

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 σ is an **NP**-complete problem.*

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

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

This leads to the following corollary.

Corollary 9. *Assume a monotonic penalty function $p : \mathbb{Z} \to \mathbb{R}$ 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 σ is **NP**-complete problem.*

Note that, due to proper inequality in Equation 1, p must be non-linear. This condition is needed since p_l satisfies every other requirement. Corollary 9 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. Related work

The problem of ranking 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 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 parents tend to dominate the children. To score such a hierarchy they 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 a unweighted graph is a dual problem to finding a maximal eulerian subgraph, a subgraph in which, for each vertex, the same number of outdoing 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 has weighed edges or we wish to enforce the cardinality constraint, we need to use 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 $q(G, k, p_c)$ is equivalent to FEEDBACK ARC SET (FAS), which is known to be **APX**-hard with a coefficient of $c = 1.3606$ [1]. 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. [7] studied assigning roles to vertices based on its features while McCallum et al. [12]

assigned topic distributions to individual vertices. A potential direction for a future work is to study whether we can use the obtained ranking as a feature in role discovery.

VI. Experiments

In this section we present our experimental evaluation.

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 [16]. 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 I.

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. 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.[2]

Results We begin with running times given in Table I. 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 decomposition, but a single iteration may be significantly faster when the decomposition is used.

[2]The source code is available at http://research.ics.aalto.fi/dmg/

Table I
Basic characteristics of the datasets and the experiments. The 6th column is the number of groups in the optimal ranking. The columns named *iter.* represent the number of iterations needed by the capacitated circulation solver.

			max SCC			with SCC		w/o SCC		
Name	$\lvert V \rvert$	$\lvert E \rvert$	$\lvert V' \rvert$	$\lvert E' \rvert$	$\lvert r \rvert$	iter.	time	iter.	time	base
Amazon	403k	3m	395k	3m	17	9k	6h24m	3k	7h29m	4h27m
Gnutella	63k	148k	14k	51k	24	83	8s	136	45s	45s
EmailEU	265k	419k	34k	151k	9	1k	10m	8k	3h8m	2m
Epinions	76k	509k	32k	444k	10	4k	49m	4k	1h5m	20m
Slashdot	82k	870k	71k	841k	9	4k	1h38m	4k	2h21m	1h5m
Google	876k	5m	435k	3m	31	52k	8h50m	6k	26h43m	2h32m
WikiVote	7k	104k	1k	39k	12	738	43s	1k	5m	7s
Nfl	32	205	32	205	6	157	22ms	157	22ms	–
Reef	258	4232	1	0	19	0	10ms	454	1.2s	–
HiggsRep.	37k	31k	263	569	11	3k	0.2s	7k	10m	–
HiggsRet.	425k	734k	13k	64k	22	5k	10m	45k	31h34m	–
HiggsM.	303k	445k	5k	20k	21	12k	2m	45k	19h38m	–

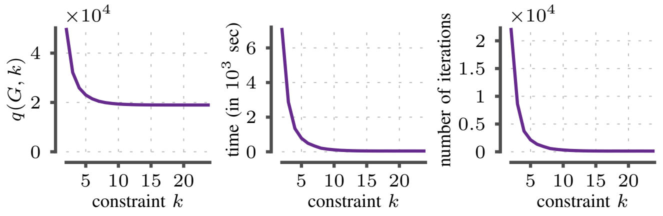

Figure 4. Agony, execution time, and number of iterations needed by capacitated circulation solver as a function of the constraint k for *Gnutella*.

The baseline Relief is almost always faster than the approach, even though the computational complexity guarantees, $O(m^2)$ for Relief and $O(m^2 \log n + mn \log^2 n)$ for canon, are practically the same. Nevertheless, the difference between the computation times is less than the order of magnitude.

Our next step is to study the effect of the constraint k, the maximum number of different rank values. We see in 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 enforced *and* the input graph has small connected components: First compute the agony quickly without the constraint using the strongly connected component decomposition. If the obtained ranking has less than k components, then we are done. Otherwise, enforce the constraint k and solve the problem without the decomposition.

Next, we consider agony as a function of k, which we have plotted in Figure 4 for *Gnutella*. We see that agony remains relatively constant as we decrease k, and starts to increase more prominently once we consider assignments with $k \leq 5$.

Enforcing the constraint k has an impact on running time. In Figure 4 we plotted the running time as a function of k. 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.

Table II
RANK ASSIGNMENT DISCOVERED FOR *Nfl* DATASET WITH $k = 3$ GROUPS

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 NYJ CHI

Let us look on the ranking that we obtained from *Nfl* dataset using $k = 3$ groups, given in Table II. 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.

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 $k = 4$ 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-detritivores* while being eaten by many other guilds. Consequently, it is ranked in the bottom group.

Table III
RANKED GUILDS OF *Reef* DATASET, WITH $k = 4$. FOR SIMPLICITY, WE REMOVED THE DUPLICATE GUILDS IN THE SAME GROUP, AND GROUPED SIMILAR GUILDS (MARKED AS ITALIC, THE NUMBER IN PARENTHESES INDICATING THE NUMBER OF GUILDS).

Sharks (6), Amberjack, Barracuda, Bigeye, Coney grouper, Flounder, Frogfish, Grouper, Grunt, Hind, Lizardfish, Mackerel, Margate, Palometa, Red hind, Red snapper, Remora, Scorpionfish, Sheepshead, Snapper, Spotted eagle ray
Angelfish, Atlantic spadefish, Ballyhoo, Barracuda, Bass Batfish, Blenny Butterflyfish, Caribbean Reef Octopus, Caribbean Reef Squid, Carnivorous fish II-V, Cornetfish, Cowfish, Damselfish, Filefish, Flamefish, Flounder, Goatfish, Grunt, Halfbeak, Hamlet, Hawkfish, Hawksbill turtle, Herring, Hogfish, Jack, Jacknife fish, Jawfish, Loggerhead sea turtle, Margate, Moray, Needlefish Porcupinefish I-II, Porkfish, Pufferfish, Scorpionfish, Seabream, Sergeant major, Sharptail eel, Slender Inshore Squid, Slippery dick, Snapper, Soldierfish, Spotted drum, Squirrelfish, Stomatopods II, Triggerfish, Trumpetfish, Trunkfish, Wrasse, Yellowfin mojarra
Crustacea (31), Ahermatypic benthic corals, Ahermatypic gorgonians, Anchovy, Angelfish, Benthic carnivores II, Blenny, Carnivorous fish I, Common Octopus, Corallivorous gastropods IV, Deep infaunal soft substrate suspension feeders, Diadema, Echinometra, Goby, Green sea turtle, Herbivorous fish I-IV, Herbivorous gastropods I, Hermatypic benthic carnivores I, Hermatypic corals, Hermatypic gorgonians, Herring, Infaunal hard substrate suspension feeders, Lytechinus, Macroplanktonic carnivores II-IV, Macroplanktonic herbivores I, Molluscivores I, Omnivorous gastropod, Parrotfish, Pilotfish, Silverside, Stomatopods I, Tripneustes, Zooplanktivorous fish I-II,
Planktons (7), *Algae (6)*, *Sponges (2)*, *Feeders (11)*, Benthic carnivores I, Carnivorous ophiuroids, Cleaner crustacea I, Corallivorous polychaetes, Detritivorous gastropods I, Echinoid carnivores I, Endolithic polychaetes, Epiphyte grazer I, Epiphytic autotrophs, Eucidaris, Gorgonian carnivores I, Herbivorous gastropod carnivores I, Herbivorous gastropods II-IV, Holothurian detritivores, Macroplanktonic carnivores I, Micro-detritivores, Molluscivores II-III, Planktonic bacteria, Polychaete predators (gastropods), Seagrasses, Sponge-anemone carnivores I, Spongivorous nudibranchs

VII. 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 with current algorithms, hence we provide a new technique by demonstrating that minimizing agony can be solved by solving a capacitated circulation problem, a well-known graph problem with a polynomial solution.

We can further generalize the setup by allowing each edge to have its own individual penalty function. As long as the penalty functions are convex, the construction done in Section IV-A can be used to solve the optimization problem. We can further generalize the cardinality constraint by requiring that only a subset of vertices must have ranks within some range. We can have multiple such constraints.

REFERENCES

[1] I. Dinur and S. Safra. On the hardness of approximating vertex cover. *Ann. Math.*, 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 multicuts 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. Iftode. Finding hierarchy in directed online social networks. In *WWW*, 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 *ACM SIGKDD*, 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. *Anim. Behav.*, 57:991–998, 1999.

[9] J. Leskovec and A. Krevl. SNAP Datasets. http://snap.stanford.edu/data, Jan. 2015.

[10] L. Macchia, F. Bonchi, F. Gullo, and L. Chiarandini. Mining summaries of propagations. In *ICDM*, pages 498–507, 2013.

[11] A. S. Maiya and T. Y. Berger-Wolf. Inferring the maximum likelihood hierarchy in social networks. In *ICSE*, 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.

[14] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., 1982.

[15] P. D. Roopnarine and R. Hertog. Detailed food web networks of three Greater Antillean Coral Reef systems: The Cayman Islands, Cuba, and Jamaica. *Dataset Papers in Ecology*, 2013, 2013.

[16] H. R. Roopnarine PD. Data from: Detailed food web networks of three Greater Antillean Coral Reef systems: The Cayman Islands, Cuba, and Jamaica, 2012. Dryad Digital Repository, http://dx.doi.org/10.5061/dryad.c213h.

[17] N. Tatti. Faster way to agony—discovering hierarchies in directed graphs. In *ECML PKDD*, pages 163–178, 2014.