ORIENTING GRAPHS IS SOMETIMES HARD

JENNIFER IGLESIAS AND ANTONIO OCHOA

ADVISOR: GLENCORA BORRADAILE
OREGON STATE UNIVERSITY
SECONDARY ADVISOR: THERESA MIGLER
OREGON STATE UNIVERSITY

ABSTRACT. We look at orienting graphs in a way that minimizes either the maximum in-degree, or the lexicographic order of the degree sequence. We sometimes also stipulate that the resulting graph have some property; either acyclic or strongly connected. We prove that in the strongly connected case minimizing the maximum in-degree can be done by a greedy algorithm. We then show that minimizing the lexicographic order of the degree sequence is NP-hard in the acyclic case.

1. INTRODUCTION

We investigated the problem of orienting edges of a graph so to minimize the maximum load of a vertex. In other words, we wanted to minimize the maximum in-degree of a vertex. This problem has been solved when there is no restriction on the orientation and when the orientation must be acyclic [V]. We will restrict ourselves to the case where the edges are unweighted, as [LLP] and [AMOZ] have shown the problem is NP-Hard for weighted graphs and gave various approximation results. We will give a polynomial time[1] algorithm and prove its correctness if we restrict the orientation to be strongly connected.

A natural extension of the problem of minimizing the maximum in-degree vertex is to minimize the lexicographic order of the in-degree sequence. This minimizes the number of maximum in-degree vertices, and then the number of next largest in-degree vertices etc. This problem has a polynomial-time solution in the case where there is no restriction on the orientation, and an extension of this work also minimizes the lexicographic order of the degree sequence for strongly connected [unpublished]. We will show that restricting the orientation to be acyclic, causes this problem to be NP-Hard.

1.1. Definitions.

In our problem we begin with a graph, and then we orient the edges in the graph. We are most interested in the resulting degrees in the directed graph.

Definition 1.1. *The* degree *of a vertex is the number of edges adjacent to it. The* in-degree *of a vertex is the number of edges oriented into that vertex. Likewise, the* out-degree *of a vertex is the number of edges oriented out of that vertex.*

Definition 1.2. *The* in-degree sequence *for a graph is the list of the in-degrees of the vertices in non-increasing order.*

Date: 8/11/2011.

This work was done during the Summer 2011 REU program in Mathematics at Oregon State University.

[1]Polynomial time is a marker for efficient solvability of a problem

When we orient a graph, we keep the same set of edges and just give an orientation to each edge. There are different types of orientations that we will be working with in this paper.

Definition 1.3. *A directed graph is* strongy connected *if for every ordered pair of vertices* (u,v) *we can find a path from u to v following the edges in their given directions.*

Definition 1.4. *A directed graph is* acyclic *if there exists no directed cycle in the graph.*

We will abbreviate the problem names to make things more compact. When we refer to the problem of minimizing the maximum in-degree of a graph, we will refer to it as min-max. When we refer to the problem of minimizing the lexicographic order of the in-degree sequence of a graph we will call it min-lex.

2. Strong Connectivity

We will begin by looking at the case in which we restrict the orientation to be strongly connected. It may seem at first glance the min-max strongly connected may be 'close' to the min-max without constraints. This is not case and the two can be very far apart. We found the following example where we can always have min-max in-degree be 4, but with the strongly connected condition, can be arbitrarily large.

As seen in figure 1, we can make a graph G by starting with a K_9, now choose 2 vertices in the K_9, u,v. Now add n vertices $w_1, w_2, \cdots w_n$ connected to u and v only. In the non-restricted case then we can orient the K_9 so each vertex in the K_9 has in-degree 4, and then orient all the other edges towards the w_i. Now the in-degree for each w_i is 2. So the min-max in-degree is 4. Now if we require strong connectivity, then for each w_i then one edge must be coming out of it and the other must be coming in to it. This requires now at least n edges total coming in to u and v. So the best we could do with a strongly oriented configuration is $n/2$. Since we can choose n as large as we want, then the min-max in-degree for strongly connected graphs can get arbitrarily far from the min-max in-degree without restrictions.

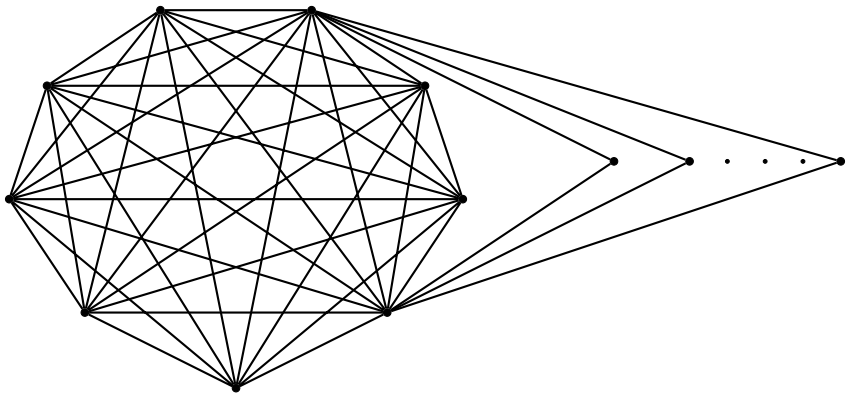

FIGURE 1. This is a graph whose min-max unrestricted is 4, and whose min-max strongly connected can be arbitrarily bad depending on how many nodes to the side we have

While the strongly connected case for min-max could be arbitrarily worse than the unconstrained case for min-max, the idea of path reversal is key to finding min-max in both cases. In the strongly

connected case, we must place a simple condition on paths which we choose to reverse to maintain strong connectivity.

Definition 2.1. *We will call a path* reversible *if we can reverse it without losing strong connectivity.*

Interestingly, the criteria for a path being reversible is only dependent on the end points of the path.

Theorem 2.2. *A path from s to t is reversible if and only if there are two edge disjoint paths from s to t.*

Proof. Consider that P is a reversible path from s to t. Then when we reverse P there must still be a path from s to t. So in the original graph there is a flow of at least 2 from s to t. If we consider any flow of at least 2, we can get two edge disjoint paths from s to t.

Now consider there are two edge disjoint paths from s to t, and any two vertices u, v in the graph. Consider any cut from u to v in the original graph. The size of this cut is at least one, since the graph is strongly connected. If s and t are on the same side of the cut, then reversing at path between them doesn't change the size of the cut. If t is on the same side as u, and s is on the same side as v then reversing a path from s to t only increases the size of the cut. If s is on the same side as u, and t is on the same sides as v then we know that reversing the path still leaves a path, so the size of the cut is at least 1. Therefore there is still a path from u to v. Therefore the graph is still strongly connected. □

Now that we have a nice criteria for when a path is reversible. We will now give a rather simple lower bound for min-max strongly connected, which we will use later to prove the optimality of our algorithm.

Theorem 2.3. *Consider any subset S of vertices. Let $n_S = |S|$, m_S be the number of edges contained within S, and c_S be the number of connected components in $G[V/S]$. Then a lower bound for the minimum-maximum in-degree is $\left\lceil \frac{m_s+c_s}{n_s} \right\rceil$.*

Proof. Consider the graph G and any strongly connected orientation. The subgraph $G[S]$ must have m_S edges within it, and at least c_S edges pointing into it. So the total in-degree of the vertices in S must be at least $m_S + c_S$. Therefore the best we can do within S is if we spread out the in-degree as much as possible. So some vertex has in-degree at least $\left\lceil \frac{m_s+c_s}{n_s} \right\rceil$. This is therefore indeed a lower bound for min-max in-degree. □

Having developed some intuition for how the strongly connected case works, we present the following algorithm which will achieve min-max strongly connected.

Algorithm 2.4. *Let k be the current maximum in-degree of a vertex. While there exists a reversible path from a vertex of in-degree $k-2$ or less to a vertex of degree k then reverse this path.*

To prove the correctness of this algorithm, we must first prove the following simple lemma.

Our algorithm reduces maximum in-degree with each iteration, and there can be at most m iterations. At each iteration, we compare the at most all pairs of vertices of our graph of which there is a quadratic number. Determining whether there are two edge disjoint paths between two nodes can be done in linear time. Therefore the whole algorithm runs in polynomial time.

Lemma 2.5. *If there exists, a vertex u which has edge disjoint paths P_s and P_t to vertices s and t respectively where s and t each have two edge disjoint paths to v, then u has two edge disjoint paths to v.*

Proof. Consider any cut with u in one side U and v in one side V. If either s or t is in U then the cut has size at least 2, since s and t both have two paths to v. Otherwise s and t are both in V. In this case, the cut still has size at least two since there is a path from u to s which is edge disjoint from a path from u to t. Therefore the flow from u to v is at least two. So there are two edge disjoint paths from u to v. □

We will now prove a more interesting lemma about the graph after the algorithm has run. This is the key to our proof as the optimality of our algorithm follows immediately from this lemma.

Lemma 2.6. *Let k be the maximum in-degree after our algorithm has run and let v be vertex with this in-degree. Let X contain v and all the vertices which have two disjoint paths to v. Let C be a component of $G[V \setminus X]$. There is exactly one edge from C to X.*

Proof. Let C be some maximally connected component in $G[V \setminus X]$. Let $x_1, x_2, \cdots x_\ell$ be the vertices in C which have arcs from C to X. Note that each x_i must have at most one edge to X since assuming otherwise implies there exists an x_i with two edge disjoint paths to v by lemma 2.5 which contradicts our definition of X. Let C_i be the set of points in C which have a path to x_i without leaving C. Since G is strongly connected, then every vertex in C has a path to a vertex in X, so each path must have an arc which goes from C to X at some point. Therefore every vertex in C is in one of the C_i. Suppose for the sake of contradiction that a vertex u is in two of the C_i, let it be C_α and C_β. There is a path from u to x_α and a path from u to x_β entirely within C. We can take u' to be the last vertex they have in common. Now we have two edge disjoint paths from u' to X. Thus u' has two edge disjoint paths each to a vertex in X. Since every vertex in X has two edge disjoint paths to v, it follows from lemma 2.5 that $u' \in C$ has two edge disjoint paths to v, which contradicts our definition of X. Therefore each vertex in C is in exactly one of the C_i.

Since C is connected though, there must be an arc between two different C_i's, let it be uw from C_α to C_β. This would cause u to be included in C_β. Therefore there can't be any arc between the C_i's. C is connected but partitioned by the C_i with no edge between the C_i. Therefore $C = C_1$ and there is exactly one arc from C to X. □

We now have the required lemmas to prove the correctness of our algorithm.

Theorem 2.7. *Algorithm 2.4 achieves the min-max in-degree for strongly connected orientations.*

Proof. Let v, k and X be the same as defined in lemma 2.6. All the vertices in X must have in-degree k or $k-1$ for our algorithm to have stopped.

If we let n_X be the number of vertices in X, let m_X be the number of edges in X, and let c_X be the number of connected components of X. We know the sum of the in-degrees in X must be exactly the number of edges in X plus the number of edges coming into X. The number of edges coming into X is c_X. Therefore we have

$$n_X k \geq m_X + c_X > n_X(k-1)$$

$$k \geq \frac{m_X + c_X}{n_X} > (k-1)$$

$$k = \left\lceil \frac{m_X + c_X}{n_X} \right\rceil$$

Since we know that $\left\lceil \frac{m_X+c_X}{n_X} \right\rceil$ is a lower bound for the minimum maximum in-degree, we can't do any better. So our algorithm achieves the minimum maximum in-degree. □

3. Acyclicity

While min-max acyclic probably has the simplest of the min-max algorithms, min-lex acyclic is the hardest. The min-max algorithm for the acyclic case is to repeatedly take a vertex of smallest degree among unoriented edges and orient edges that haven't yet been oriented into this vertex. Trying to minimize the lexicographic order of the degree sequence for an acyclic orientation is NP-hard. To show this, we will show an easier problem, to which min-lex reduces, is NP-Hard. To start, we first need to describe the gadgets and define some new terms.

3.1. Making the Gadgets.

We define a *k-gadget* graph H_ℓ for $0 < \ell < k$ and k odd to be such that:

(1) All the vertices, except the root vertex r, have degree k
(2) The root vertex r has degree $k-\ell$
(3) $H_\ell \backslash \{r\}$ is connected

Our construction will depend on the parity of ℓ.

We can construct such a our graph H_ℓ as with two copies of K_k, r our root vertex, and s if ℓ is even. If ℓ is odd, connect the root to $(k-\ell)/2$ of the vertices in each of the complete graphs. If ℓ is even, connect r to $k-\ell$ vertices in one of the K_k, and connect s to $\ell/2$ vertices in this same K_k, and $k-\ell/2$ of the vertices in the other complete graph. Now with the non-root vertices of degree $k-1$ just put an edge from one to another vertex of degree $k-1$ in the other K_k.

These k-gadgets have the desired qualities. By construction, all the vertices but the root vertex have degree k, and the root vertex has degree $k-\ell$. Since $0 < \ell < k$, we will always have at least one edge between the two complete graphs in the k-gadget. Therefore $H_\ell \backslash \{r\}$ is connected.

3.2. t-Strippable.

We will define a graph to be t-strippable if there exists an ordering of the vertices, such that we can remove the vertices in this order and at each step the vertex we are removing has degree at most t.

If a graph has Minimum Maximum in-degree t, then the graph is t-strippable. Another specific type of graph which is t-strippable, is when a vertex is removed from a graph where every vertex has degree at most t. In particular, once we remove a vertex from any of the gadgets, the entire gadget becomes $(k-1)$ strippable.

Lemma 3.1. *For any v a vertex in a k-gadget H_ℓ then $H_\ell \backslash \{v\}$ is $(k-1)$ strippable.*

Proof. We will show that whenever we have removed some positive number of vertices from H_ℓ, there is some vertex with degree less than k. Let S be the set of vertices we have removed so far. Consider $N(S)$ the neighbors of S, which aren't in S. Since removing the root vertex from H_ℓ does not disconnect H_ℓ then $N(S)$ always contains a non-root vertex unless $S = V$ or $S = V \backslash \{r\}$. If $N(S)$ contains a non-root vertex, then we can take this vertex next in our order. If $S = V$, then we have stripped all the vertices from the gadget and are done. If $S = V \backslash \{r\}$ then we can remove r since its degree is now at most $k-1$ when all the vertices in the gadget are removed. So any k-gadget, with a vertex removed is $(k-1)$ strippable. □

3.3. **Min number of max acyclic is NP HARD.** We know that finding min max acyclic takes just polynomial time and can be done by stripping. We show that the problem of Min Lex Acyclic is NP hard by a reduction from Set Cover to the related problem of Number of Maximum In-degree vertices in an Acyclic orientation [GJ]. The problem Number of Maximum in-degree vertices in an Acyclic orientation takes a graph G and a number ℓ, and says whether there exists an acyclic orientation of G where at most ℓ vertices have the maximum in-degree. The problem of finding acyclic minimum lexicographic order solves the problem of minimizing the number of maximum in-degree vertices.

The reduction will be from Set Cover to Number of Maximum in-degree vertices. Let our instance of Set Cover be $S = x_1, x_2, \cdots, x_n$, $\mathcal{C} = S_1, S_2, \cdots S_m$, and ℓ. Set Cover asks if we can choose at most ℓ sets from $\mathcal{C}$ which cover S. We will now create our instance of Number of Maximum in-degree acyclic.

Let k be the smallest odd number which is greater than all $|S_i|$ and f_i where f_i is the frequency of the element i in the sets in C. Now let k be the smallest odd number which is at least x. We will now construct our graph for Number of Maximum in-degree vertices as follows. For each S_i create a k-gadget H_1, call the root vertex for this gadget u_i. Now for each $x_i \in S$, create a k-gadget H_{f_i}, and call the root vertex for this gadget v_i. We will now connect u_i to v_j if and only if $x_j \in S_i$. In the resulting graph, all the vertices except for the u_i have degree k. The u_i have degree $k + |S_i| - 1$.

Note that G is k-strippable. We can strip a vertex of degree k from each gadget, and then the remainder of the each gadget is $(k-1)$-strippable. So G is k strippable. We also know that originally all the vertices in the graph have degree at least k. Therefore Minimum Maximum In-degree acyclic for G is k.

Theorem 3.2. *G has an acyclic orientation with at most ℓ vertices of in-degree k if and only if there is a set cover of size at most ℓ.*

Proof. First, we will prove the forward direction. Let X be the set of ℓ vertices which have in-degree k in our solution to Number of Maximum In-degree. Let $\mathcal{C}'$ be formed by all the sets whose gadget have an in-degree k vertex, and for each element gadget that contains a in-degree k vertex, take any set which contains that element. Clearly we have $|\mathcal{C}'| \leq \ell$. We need to show that every element in S is covered by a set in $\mathcal{C}'$. We know that $G \backslash X$ is $(k-1)$ strippable. Consider any $x_i \in S$. Within the stripping process of $G \backslash X$, the only ways for the gadget for v_i to become $k-1$ strippable, is for one of the vertices to have originally been in X or an edge from a u_j to be removed. If one of the vertices was originally in X, we picked a set covering x_i. If none of the vertices in the v_i gadget were removed that means we were able to $k-1$ strip one of the gadgets at a u_j, where S_j contains x_i, before we $k-1$ stripped the gadget at v_i. A gadget at an u_j becomes $k-1$ strippable only if one of its vertices are removed or all the gadgets at v_r it is connected to became $k-1$ strippable. The second case can't occur for all such u_j. So there was a vertex removed from the gadget at some u_j. Therefore we took S_j as one of the sets covering S, and x_i gets covered. So the sets induced by X cover S. We have at most ℓ sets which cover S.

Now we will prove the reverse direction. Let $\mathcal{C}'$ be the collection of at most ℓ sets from C which cover S. If for every $S_i \in \mathcal{C}'$ we take a non-root vertex in the gadget at u_i, then then these ℓ vertices can all be taken to be sinks and the remaining graph becomes $k-1$-strippable. Each of the gadgets at a u_i where a vertex is removed becomes $k-1$-strippable. Now every v_j is next to one of these u_i. The gadgets at each of the v_j become $k-1$-strippable. So we are now just left with the gadgets

at the u_i which weren't chosen. The remaining u_i all have degree $k-1$ though. So the remaining gadgets are $k-1$ strippable. So we have found an orientation which has ℓ vertices of degree k. □

We can solve the problem of minimizing the number of maximum degree vertices if we could solve min-lex. Since the problem of minimizing the number of maximum degree vertices is NP-hard for acyclic orientations, then min-lex must be NP-hard for acyclic orientations.

4. Conclusion

We were able to provide a polynomial time algorithm for min-max strongly connected and prove its correctness. It turns out that only a slight modification to the algorithm allows us to also find min-lex strongly connected. The results for this will be provided in a later paper. Even though we have simple algorithms for min-lex unconstrained, and min-lex strongly connected, min-lex acyclic is NP-hard. Another problem which is related to minimizing the lexicographic degree sequence is minimizing the sum of a convex function of the in-degrees. The problem of min-lex and minimum sum of a convex function are the same for the unconstrained case and the strongly connected case. One problem that remains open though, is the problem of minimizing the sum of a convex function of the in-degrees with restriction to an acyclic orientation.

References

[GJ] Garey, and Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness* W. H. Freeman & Co., New York, NY, USA, 1979

[LLP] Kangbok Lee, Joseph Y.-T. Leung, and Michael L. Pinedo *A Note on Graph Balancing Problems with Restrictions* Information Processing Letters 110 (2009) 24-29.

[AMOZ] Yuichi Asahiro, Eiji Miyano, Hirotaka Ono, and Kouhei Zenmyo *Graph Orientation Algorithms to Minimize the Maximum Outdegree* CATS '06 Proceedings of the 12th Computing: The Australasian Theroy Symposium - Volume 51

[CGT] Fan R. K. Chung, Michael R. Garey, Robert E. Tarjan. *Strongly Connected Orientations of Mixed Multigraphs* Networks, Volume 15, Issue 4, (Winter 1985), 477-484.

[V] V. Venkateswaran *Minimiziming Maximum Indegree* Discrete Applied Mathematics, 143 (2004) 374-378.

Harvey Mudd College
E-mail address: jiglesias@hmc.edu

California State Polytechnic, Pomona
E-mail address: adjochoa@gmail.com