A Note on Optimal Covering Augmentation for Graphic Polymatroids

Sachin B. Patkar * and H. Narayanan †

June 2000

Abstract

We present a simple and efficient algorithm for the problem of *optimal covering augmentation* for graphic polymatroids. We make a simple modification to the greedy algorithm for polymatroids [3] so that it terminates in fewer steps. Our algorithm is far simpler yet as efficient as the previous best algorithm of Gabow [5]. It may also be noted that our algorithm is indeed an efficient algorithm for the general problem of performing min-cost augmentation of a feasible vector to a base of a graphic polymatroid.

keywords: Algorithms, Combinatorial Problems, Analysis of Algorithms, Design of Algorithms.

*Department of Mathematics, Indian Institute of Technology, Bombay Bombay-400 076, India. e-mail: patkar@math.iitb.ernet.in

†Department of Electrical Engineering, Indian Institute of Technology, Bombay Bombay-400 076, India e-mail: hn@ee.iitb.ernet.in

1 Introduction

Let $G(V, E, w_e)$, denote an undirected graph $G(V, E)$ with vertex (node) set V, edge set E and a **positive** assignment of weights given by w_e to the edges. $\mathcal{V}(X)$ denotes the set of endpoints of the given subset X of E. $\mathcal{E}(U)$ denotes the set of edges having both the endpoints in the given subset U of V. $\delta(U)$ denotes the set of edges having exactly one of the endpoints in the given subset U of V. Let $(U, \mathcal{E}(U))$ denote the subgraph induced by the subset of nodes U. Let $(\mathcal{V}(X), X)$ denote the subgraph induced by the subset of edges X. Graphic polymatriod $GP(\lambda)$ is defined as $GP(\lambda) = \{x \in \mathbf{R}_+^E \mid x(X) \leq \lambda * r(X) \ldots X \subseteq E\}$ Fujishige [4] discussed the problem of computing a subset of edges of the given graph $G(V, E, w_e)$ that maximizes the ratio $\frac{w_e(X)}{r(X)}$ over all nonempty subsets $X \subseteq E$, where $r(.)$ is the rank function of the graph ($r(X)$ is the number of endpoints of edge subset X minus the number of connected components of the subgraph $(\mathcal{V}(X), X)$). The maximum such ratio is called *fractional arboricity* of the given edge-weighted graph. We denote it by $\sigma(G(V, E, w_e))$ (σ for short, whenever the graph is clear from the context). Equivalently, σ is the smallest real such that the edge-weight vector $w_e \in GP(\sigma)$.

Fujishige [4] showed how this problem could be solved using the results of Tomizawa and Narayanan (see [4] [7] for details) on the Principal Partition of the rank function of graph. Efficient algorithms for the computation of the principal partition are given by Patkar and Narayanan (see [6] [7] for more information). The problem of computing fractional arboricity of a graph is solved most efficiently by Gabow in [5] in $O(|V||E| \log(|V|^2/|E|))$ time.

In this paper, we shall focus on the minimum cost augmentation problem described below: Let σ be the fractional arboricity of the given graph. The problem of computing *optimal covering augmentation* for an edge-weighted graph is to compute maximum possible augmentation of the edge-weights of the graph such that the *fractional arboricity* does not change. Gabow [5] gave an efficient $O(|V|^2|E|\log(|V|^2/|E|))$ algorithm for computing *optimal covering augmentation.* However, one may note that the *prefix algorithm* of Gabow [5] involves invocations of powerful but *relatively complicated* algorithms for *parametric network flow in bipartite networks* (due to Ahuja,Orlin,Stein and Tarjan), *global minimum cut in directed graphs* (due to Hao and Orlin), and *parametric augmentation* in [5]. Indeed the *prefix algorithm* would involve $O(|V|^2)$ network flow computations, grouped in $O(|V|)$ groups each containing a pipelining of $O(|V|)$ network flow problems.

In this paper, we present a **far simpler** algorithm for *optimal covering augmentation* that has the same time complexity, *ie.* $O(|V|^2|E|\log(|V|^2/|E|))$. Our simple algorithm would involve only $O(|V|)$ straightforward network flow computations (without pipelining!), after the fractional arboricity σ has been computed by the efficient algorithm of [5] in $O(|V||E|\log(|V|^2/|E|))$.

2 Preliminaries and Notation

Let $f : 2^S \longrightarrow \mathbf{R}$ be a given function. The function f is submodular **iff** $f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y) \ldots \forall\ X, Y \subseteq S$. The function f is supermodular **iff** $-f$ is submodular and f is modular **iff** it is both submodular and supermodular. A *polymatroid function* is a function $f : 2^S \rightarrow \mathbf{R}_+$ such that $f(\emptyset) = 0$, and f is nondecreasing and submodular. Such an f determines the polymatroid $P(f) = \{x \in \mathbf{R}_+^S \mid x(X) \leq f(X) \ldots \forall X \subseteq S\}$. A *base* of $P(f)$ is any maximal vector b (ie. $b(S)$ is maximum) which belongs to $P(f)$. Given a vector $p \in P(f)$ and a cost vector c on the elements of S, the *polymatroid greedy algorithm* (see [3] [4] [7] for details), finds a minimum cost base b of $P(f)$ such that $p \leq b$ (componentwise!), as follows: It initializes b to p. Then it examines each component i ($1 \leq i \leq |S|$) in order of nondecreasing cost. For each i, it increases b_i as much as possible while keeping $b \in P(f)$.

3 Modified Greedy Algorithm for Graphic Polymatroids

¿From the definitions it is clear that, to solve the *optimal covering augmentation* problem we need to find w'_e satisfying

System 1 $w'_e(X) \leq \sigma * r(X) \ldots X \subseteq E$, $w'_e(.) \geq w_e(.)$ componentwise*, and* $w'_e(E)$ is maximum.

As $r(.)$ is a well-known polymatroid function (see [3] [7]), and $\sigma \geq 0$, the function $\sigma * r(.) : 2^E \rightarrow \mathbf{R}$ is also a polymatroid function. We shall work with the polymatroid, ie., $P(\sigma * r(.))$, which is also denoted by $GP(\sigma)$. The following lemma is a consequence of the definitions of a feasible vector and a base of a polymatroid.

Lemma 1 *For any graph $G(V, E, w_e)$ the weight vector $w_e(.)$ is a feasible vector in $GP(\sigma)$. The system 1 indeed describes the collection of all bases w'_e of $GP(\sigma)$, which are componentwise greater than or equal to the vector $w_e \in GP(\sigma)$.*

It is well-known (see [3] [4] [7]) that any base b of the polymatroid $GP(\sigma)$ satisfies $b(E) = \sigma * r(E)$. The following min-max relation is a simple consequence of the above discussion.

Proposition 1 *The maximum total amount of augmentations permissible while maintaining the fractional arboricity (ie, the maximum ratio $\frac{w_e(X)}{r(X)}$ over all subsets $X \subseteq E$), at the same level* **is equal to** *the minimum total amount of augmentation needed to make the ratio of total edge weight and the rank of the full graph equal to its fractional arboricity (w.r.t the augmented weights).*

We will use the following equivalent description of $GP(\sigma)$, which is due to Nash-Williams (see [4] [7] for more details).

System 2

$$GP(\sigma) = \{x \in \mathbf{R}_+^E \mid x(X) \leq \sigma * (|\mathcal{V}(X)| - 1) \ldots \emptyset \neq X \subseteq E\}$$

We will use the above equivalent formulation of $GP(\sigma)$ to implement the basic iteration of the greedy algorithm, which finds to what extent j^{th} component of the feasible vector be increased while maintaining feasibility. We will use $\overline{w}_e$ to denote $w'_e - w_e$. Now we present the efficient version of the polymatroid greedy algorithm for the polymatroid $GP(\sigma)$.

Algorithm Modified-Greedy

Order the elements of E in the non-decreasing order of the costs $c(.)$

Let $S = E = \{1, 2, \ldots, |E|\}$

$\overline{w}_e = 0$ (ie. set all components of $\overline{w}_e$ to zero)

while (S not empty) do

    let j be the smallest index in S

$\epsilon = \min\{\sigma * |\mathcal{V}(X)| - \sigma - w_e(X) - \overline{w}_e(X) | j \in X \subseteq E\}$

let $\hat{X}$ be the unique largest X that solves the above minimization step

$\overline{w}_e(j) = \epsilon$

$S = S - \hat{X}$

endwhile

We now state the following theorem about its time complexity, the proof is given in the following sections.

Theorem 1 *The algorithm* **Modified-Greedy** *solves the the problem of optimal covering augmentation using* $O(|V|)$ *invocations of the minimization step.*

3.1 Correctness of the Modified Greedy Algorithm

We begin with noting that the *Greedy Algorithm* is correct [3] [4] [7]. Since weights are positive, $w'_e(\hat{X}) < w'_e(\mathcal{E}(\mathcal{V}(\hat{X})))$, unless $\hat{X} = \mathcal{E}(\mathcal{V}(\hat{X}))$. Thus,

Lemma 2 *Any solution* $\hat{X}$ *for* X *that solves* $\min_{j \in X \subseteq E}\{\sigma * |\mathcal{V}(X)| - \sigma - (w_e + \overline{w}_e)(X)\}$ *is of the form* $\mathcal{E}(\mathcal{V}(\hat{X}))$. *Thus it is sufficient to solve* $\min_{U \subseteq V, j \in \mathcal{E}(U)}\{\sigma * |U| - (w_e + \overline{w}_e)(\mathcal{E}(U))\}$

Noting that if X_1 and X_2 are two solutions to the minimization step then $X_1 \cup X_2$ is also a solution. This follows from submodularity and the fact that $j \in X_1 \cap X_2$ (thus ensuring that $X_1 \cap X_2 \neq \emptyset$, so the submodularity inequality can be employed to prove that $X_1 \cup X_2$ also solves the minimization step). Thus,

Lemma 3 *There exists unique largest $\hat{X}$ for X that solves* $\min_{j \in X \subseteq E}\{\sigma * |\mathcal{V}(X)| - \sigma - (w_e + \overline{w}_e)(X)\}$

We are justified in letting the weights of edges in $\hat{X}$ remain at the original values with the exception of the weight of the edge j which is increased by the amount ϵ found in the iteration. Indeed, if any of the edges other than j in $\hat{X}$ is also to be augmented by non-zero weight then the subset $\hat{X}$ will violate the feasibility requirement.

3.2 Bound on Number of Iterations in Modified Greedy Algorithm

Now we establish that the number of iterations in the modified greedy algorithm is $O(|V|)$. The following crucial lemma establishes the claimed bound.

Lemma 4 *There are at most $O(|V|)$ iterations of the while loop of the* **Modified-Greedy** *algorithm.*

Clearly the above lemma establishes theorem 1.

Proof of Lemma 4: Let $j_1, j_2, \ldots, j_k$ be the sequence of edges whose edge-weights are augmented (strictly!) in the while-loop of the algorithm *Modified Greedy*. Let $\hat{X}_p$ denote the (unique largest) solution of the minimization step that computes augmentation to the edge j_p. Let V_p denote $\mathcal{V}(\hat{X}_p)$. We will show that for all $i < j$, either $V_i \subset V_j$ (note properness!) or $V_i \cap V_j = \emptyset$. Clearly for $i < j$, if V_i, V_j are to satisfy containment then it must be $V_i \subseteq V_j$. Furthermore, proper inclusion also follows as we can not have $\hat{X}_i = \mathcal{E}(V_i)$ and $\hat{X}_j = \mathcal{E}(V_j)$ identical. Now it remains to prove that, for $i < j$, whenever $V_i \cap V_j \neq \emptyset$, one must be contained in the other. Consider $V_i \cup V_j$. By supermodularity of $w'_e(\mathcal{E}(.))$ (which can be checked easily or may be found in [7]), $w'_e(\mathcal{E}(V_i \cup V_j)) + w'_e(\mathcal{E}(V_i \cap V_j)) \geq w'_e(\mathcal{E}(V_i)) + w'_e(\mathcal{E}(V_j))$. Thus $w'_e(\mathcal{E}(V_i \cup V_j)) + w'_e(\mathcal{E}(V_i \cap V_j)) \geq \sigma * ((|V_i|-1)+(|V_j|-1)) = \sigma * ((|V_i \cup V_j|-1) + (|V_i \cap V_j|-1))$ implying $w'_e(\mathcal{E}(V_i \cup V_j)) \geq \sigma * (|V_i \cup V_j|-1)$, as $w'_e(\mathcal{E}(V_i \cap V_j)) \leq \sigma * (|V_i \cap V_j| - 1)$. One can easily see that this would violate maximality of $\hat{X}_j$ unless $V_i \subseteq V_j$. Thus at the end of iteration p, the maximal subsets in the collection $V_1, V_2, \ldots, V_p$ are disjoint. Now consider the iteration $p+1$. If V_{p+1} has some vertex that is not in any of V_l $(l = 1, .., p)$ then $\bigcup_{l=1}^{p+1} V_l \supset \bigcup_{l=1}^{p} V_l$. Now suppose $V_{p+1} \subseteq \bigcup_{l=1}^{p} V_l$ then, at least two of the maximal subsets from among $V_1, V_2, \ldots, V_p$ (namely those containing the endpoints of j_{p+1}) must have been merged inside V_{p+1}. Thus the number of maximal subsets in the collection $V_1, \ldots, V_{p+1}$ has decreased. These observations bound the number of iterations to $O(|V|)$. **q.e.d**

Now we note that we can use a simple modification of a network model used in [2, 1] to solve the minimization step using one max-flow invocation on a network with $O(|V|)$ nodes and $O(|E|)$ arcs.

Theorem 2 *The minimization step can be solved in time $O(|V||E| \log(|V|^2/|E|))$ time using the preflow-push algorithm of Goldberg and Tarjan. The algorithm* **Modified-Greedy** *solves the the problem of optimal covering augmentation in $O(|V|^2|E| \log(|V|^2/|E|))$ time.*

4 Conclusions

We have presented a simple and efficient algorithm for solving the problem of *optimal covering augmentation* for graphic polymatroids.

References

[1] E. Cheng, W. Cunningham, A faster algorithm for computing the strength of a network, *Information Processing Letters*, vol. 49, (1994), pp.209-212.

[2] W. Cunningham, Optimal attack and reinforcement of a network, *JACM*, vol. 32, no. 3, (1985), pp.549-561.

[3] J. Edmonds, Submodular functions, matroids and certain polyhedra, in: *Proc. Calgary Intl. conference on Combinatorial Structures*, 1970, pp.69-87.

[4] S. Fujishige, *Submodular Functions and Optimization, Annals of Discrete Mathematics*, North Holland, 1991.

[5] H.N. Gabow, Algorithms for graphic polymatroids and parametric s-sets, *J. Algorithms*, vol. 26, (1998), pp.48-86.

[6] H. Narayanan, S. Roy, S.B. Patkar, Approximation algorithms for min-k-overlap problems, using the principal lattice of partitions approach, *J. Algorithms*, vol. 21, (1996), pp. 306-330.

[7] H. Narayanan, *Submodular Functions and Electrical Networks, Annals of Discrete Mathematics-54*, North Holland, 1997.

5 Appendix

5.1 Correctness of the Greedy Algorithm to Perform Min-cost Augmentation of a Feasible Vector to a Base of a Polymatroid

Recall that a *polymatroid function* is a function $f : 2^S \rightarrow \mathbf{R}_+$ such that $f(\emptyset) = 0$, and f is nondecreasing and submodular. Such an f determines the polymatroid $P(f) = \{x \in \mathbf{R}_+^S \mid x(X) \leq f(X) \ldots \forall X \subseteq S\}$. Let $y \in P(f)$. Given a cost vector c on the set S, we want to find $z \in \mathbf{R}_+^S$ such that $y + z$ is a base of $P(f)$ and cost $\sum_{e \in S} c(e) * z(e)$ is minimum. Equivalently, we need to find a min-cost vector (w.r.t. costs c) $z \in \mathbf{R}_+^S$ such that $z(S) = (f - y)(S)$ and $z(X) \leq (f - y)(X) \ldots X \subseteq S$. Note that $(f - y)$ is normalized and submodular, but need not be a polymatroid function. On the other hand it is well-known [2, 4] that, the function g defined as: $g(X) = \min\{(f-y)(Y) | X \subseteq Y \subseteq S\}$, is a polymatroid function. Furthermore,
$\{x \in \mathbf{R}_+^S \mid x(X) \leq (f - y)(X) \ldots X \subseteq S\} = \{x \in \mathbf{R}_+^S \mid x(X) \leq g(X) \ldots X \subseteq S\}$. Thus our problem is equivalent to finding a minimum cost base (w.r.t. c) z in the polymatroid $P(g)$. The standard polymatroid greedy algorithm [3] would start with initializing z with a zero vector, would then scan the components in the non-decreasing order of the costs. And then increase the components as far as possible while maintaining the feasibility in $P(g)$. That is, while scanning j^{th} component of z, the greedy algorithm performs $z(j) = \min\{g(X) - z(X) | j \in X \subseteq S\}$. It is easy to see that the above is equivalent to, $z(j) = \min\{(f - y)(X) - z(X) | j \in X \subseteq S\}$. Thus the polymatroid greedy algorithm may equivalently, increase j^{th} component as

far as possible maintaining feasibility w.r.t. $(f - y)$ rather than g itself. Equivalently, we could increase j^{th} component of $y + z$ as far as possible, maintaining feasibility in the polymatroid $P(f)$. Thus the following version of the greedy algorithm (as stated in this paper) for performing min-cost augmentation of a feasible vector to a base of a polymatroid, is correct.

Given a vector $p \in P(f)$ and a cost vector c on the elements of S, the *polymatroid greedy algorithm* finds a minimum cost base b of $P(f)$ such that $p \leq b$ (componentwise!), as follows: It initializes b to p. Then it examines each component i $(1 \leq i \leq |S|)$ in order of nondecreasing cost. For each i, it increases b_i as much as possible while keeping $b \in P(f)$.

5.2 Network Flow Model for Solving the Minimization Problem

We describe briefly the well-known network model used for solving the minimization step of the modified greedy algorithm. The network model was used by Cunningham (see [2, 1] for more information).

We recall that $w'_e(.) = w_e(.) + \overline{w}_e(.)$.

We shall now define a flow network which will be used to find a set $\hat{U}$ that is U which solves $\min_{U \subseteq V, j \in \mathcal{E}(U)} \{\sigma * |U| - w'_e(\mathcal{E}(U))\}$. We make a subtle change in the formulation as follows: the condition $j \in \mathcal{E}(U)$ is replaced by the condition that $\alpha, \beta \in U$ where α, β are the endpoints of $j \in E$. So we try to find $\min_{U \subseteq V,\, \alpha, \beta \in U} \{\sigma * |U| - w'_e(\mathcal{E}(U))\}$.

Our flow network model is as follows: Given a graph $G(V, E, w'_e)$, real number σ, and two distinguished nodes $\alpha, \beta \in V$, we define a flow network $FL(V, E, w'_e, \sigma, \alpha, \beta)$ as follows: $FL(V, E, w'_e, \sigma, \alpha, \beta)$ consists of nodes representing source s, sink t (with $s, t \notin V$) and nodes in V.

The arcs of the above network are as follows:

$\{(s, u) \mid u \in V\}$ $\{(u, t) \mid u \in V\}$ and

$$\{(u, v) \mid u, v \in V \text{ there exists an edge between } u, v \text{ in either direction in } G\}$$

The capacities on arcs are as follows,

$cap(s, u) = \frac{1}{2} * w'_e(\delta(\{u\})) \quad \ldots \text{if } u \in V - \{\alpha, \beta\}$

$cap(s, \alpha) = \infty$

$cap(s, \beta) = \infty$

$cap(u, t) = \sigma \ \forall u \in V$

$cap(u, v) = \frac{1}{2} * w'_e((u, v))$ or $\frac{1}{2} * w'_e((v, u)) \quad \forall u, v \in V$.

Any cut partitions the node set of the flow network in two parts one of which has the source as one of its members and the other, the sink as one of its members. The set of nodes containing the source (sink) is called s-part (t-part) of the cut.

Canonical cut: Given $U \subseteq V$, in $FL(V, E, w'_e, \sigma, \alpha, \beta)$ we define $canonicalCut(U)$ as the cut whose s-part is $\{s\} \cup U$. Note that, for $U \subseteq V$ such that $\alpha, \beta \in U$, $capacity(canonicalCut(U)) = \sigma * |U| + w'_e(E) - w'_e(\mathcal{E}(U))$. As α, β are connected to s using infinite capacity arcs, every minimum cut must contain α, β in its *s-part*. Thus the minimization problem is equivalent to finding $\hat{U}$ such that $canonicalCut(\hat{U})$ is a minimum cut. In addition, we also require that $\hat{U}$ is largest such.

To find a set $\hat{U}$ that is U which solves $\min_{U \subseteq V,\, \alpha, \beta \in U}\{\sigma * |U| - w'_e(\mathcal{E}(U))\}$, a maximum flow in the network $FL(V, E, w'_e, \sigma, \alpha, \beta)$ is constructed. The canonicalCut with the largest *s-part* is constructed using appropriate breadth first search from the sink (for more details the reader is referred to the papers of Picard and Queyeranne which are cited in [7]).

5.3 Additional Relevant References

R.K. Ahuja, J.B. Orlin, C. Stein, R.E. Tarjan, Improved algorithms for bipartite network flow, SIAM J. Comput., vol. 23, no. 5, (1994), pp. 906-933.

G. Gallo, M. Grigoriadis, R.E. Tarjan, A fast parametric network flow algorithm, *SIAM J. of Computing*, vol. 18, (1989), pp.30-55.

A.V. Goldberg, R.E. Tarjan, A new approach to the maximum flow problem, *J. Assoc. Computing Mach.*, vol. 35, (1988).

J. Hao, J.B. Orlin, A faster algorithm for finding the minimum cut in a graph, in: *Proc. of the 3rd ACM-SIAM SODA*, 1992, pp. 165-174.