TR-2010-05. Published by the Egerváry Research Group, Pázmány P. sétány 1/C, H–1117, Budapest, Hungary. Web site: `www.cs.elte.hu/egres`. ISSN 1587–4451.

Simple push-relabel algorithms for matroids and submodular flows

András Frank and Zoltán Miklós

July 2011

Simple push-relabel algorithms for matroids and submodular flows

András Frank and Zoltán Miklós[⋆]

Abstract

We derive simple push-relabel algorithms for the matroid partitioning, matroid membership, and submodular flow feasibility problems. It turns out that, in order to have a strongly polynomial algorithm, the lexicographic rule used in all previous algorithms for the two latter problems can be avoided. Its proper role is that it helps speeding up the algorithm in the last problem.

1 Introduction

Push-relabel algorithms (see, for example, the first one of Goldberg and Tarjan, [16]), unlike augmenting path type algorithms, use only small, local steps. In order to make progess, in selecting the current element where the next local step is to be performed, they use a control parameter $\Theta : S \to \{0, 1, 2, \ldots\}$ called a **level** (or distance) **function**. Here S can be the node-set of a directed graph or the ground-set of a matroid. In the present work the range of the level functions is $\{0, 1, 2, \ldots, n\}$ where $n = |S|$ while the original algorithm of Goldberg and Tarjan for maximum flows must have allowed $\{0, 1, 2, \ldots, 2n - 1\}$ for the range of Θ.

The goal of the present paper is to develop simple push-relabel algorithms in submodular optimization. We exhibit versions for matroid partition, for membership in a matroid polytope, and for submodular flow feasibility. All the previous algorithms relied on a selection rule based on a consistent ordering of the elements. This rule can be considered as a counterpart of the lexicographic rule of Schönsleben [19] applied to augmenting path type algorithms. The new push-relabel algorithms do not use the consistency rule and the proof of strong polynomiality becomes much simpler. The true role of the consistency rule is that, though not needed for strong polynomiality, it improves the complexity of the algorithm by one order of magnitude.

For a given level function Θ, the sets $L_i = \{v : \Theta(v) = i\}$ $(i = 0, \ldots, n)$ are called the **level sets** of Θ. For an element s with $\Theta(s) = j$, we say that the level of s is j or that s is in L_j. For a subset $X \subseteq S$, let $\Theta_{\min}(X) := \min\{\Theta(v) : v \in X\}$. One of the local steps during the algorithm is **lifting** an element s of S with $\Theta(s) \leq n - 1$

[⋆]MTA-ELTE Egerváry Research Group, Department of Operations Research, Eötvös University, Pázmány P. s. 1/c, Budapest, Hungary, H-1117. e-mail: `frank@cs.elte.hu` and `miklosz@cs.elte.hu`

which means that we increase $\Theta(s)$ by 1. The set of operations performed between two lifting operations will be called a **phase** of the algorithm. Since the level of an element is never decreased, the number of lifting operations, and hence the number of phases, is at most n^2.

We do not distinguish between a one-element set $\{s\}$ and its only element s. For example, $B - s$ consists of the elements of B distinct from s, while $B + s$ is an abbreviation for $B \cup \{s\}$. Similarly, for a set-function b and an element s, $b(\{s\})$ is denoted by $b(s)$. For a function $m : V \to \mathbf{R}$, we define its extension $\widetilde{m}$ by $\widetilde{m}(X) := \sum[m(v) : v \in X]$ for $X \subseteq S$. Given a ground-set S, the characteristic (or indicator) vector of a subset $B \subseteq S$ will be denoted by $\underline{\chi}_B$.

Acknowledgements Many thanks are due to Tom McCormick for his valuable remarks on an early version of our manuscript. The authors received a grant (no. CK 80124) from the National Development Agency of Hungary, based on a source from the Research and Technology Innovation Fund. Part of research was done while the first author visited the Research Institute for Mathematical Sciences, Kyoto University, 2008, and the Institute of Discrete Mathematics, University of Bonn, 2011.

2 Matroid partition

Both matroid optimization problems we are considering need the following lemma. For a given basis B and element $u \in S - B$, there is a unique circuit in $B + u$ containing u, which is denoted by $C(B, u)$ and called the fundemantal circuit of u. For each element $v \in C(B, u)$, the set $B + u - v$ is also a basis of M.

Lemma 2.1. *Let B be a basis of a matroid $M = (S, \mathcal{B})$, $s \in S - B$, $t \in C(B, s) - s$, and $B' := S - t + s$. If $\Theta : S \to \{0, 1, \dots, n\}$ is a level function for which $\Theta(t) = \Theta_{\min}(C(B, s))$, then*

$$\Theta_{\min}(C(B', u)) \geq \Theta_{\min}(C(B, u)) \tag{1}$$

holds for every $u \in (S - B') - t$ and $C(B', u) = C(B, s)$ if $u = t$.

Proof. The second part of the lemma is straightforward. For the first part, if $t \notin C(B, u)$, then $C(B, u) = C(B', u)$ and (1) holds (with equality). If $t \in C(B, u)$, then $\Theta_{\min}(C(B, u)) \leq \Theta(t)$ and, by the strong circuit axiom, there is a circuit $C \subseteq (C(B, u) \cup C(B, s)) - t)$ containing u. We must have $C(B', u) = C$ from which $\Theta_{\min}(C(B', u) \geq \min\{\Theta_{\min}(C(B, s)),\ \Theta_{\min}(C(B, u))\} = \min\{\Theta(t), \Theta_{\min}(C(B, u))\} = \Theta_{\min}(C(B, u))$. •

Let $M_1 = (S, \mathcal{B}_1), M_2 = (S, \mathcal{B}_2), \dots, M_k = (S, \mathcal{B}_k)$ be k matroids on an n-element ground-set S. We say that a subset $F \subseteq S$ is **coverable** if $F \subseteq B_1 \cup \dots \cup B_k$ for some $B_i \in \mathcal{B}_i$ $(i = 1, \dots, 2)$. We construct a push-relabel algorithm for finding a largest coverable subset. Previously, Edmonds and Fulkerson [5] developed an augmenting path type algorithm for this purpose and proved the following min-max formula. Our algorithm reproves their result.

Theorem 2.2 (Matroid partition theorem, Edmonds and Fulkerson, [5]). *Let $M_1, M_2, , \dots, M_k$ be matroids on a common ground-set S. The largest cardinality of a coverable subset of S is equal to*

$$\min\left\{\sum_i r_i(Z) + |S - Z| : Z \subseteq S\right\}. \tag{2}$$

Proof. For a subset $Z \subseteq S$ and for the union F of k bases $B_i \in \mathcal{B}_i$,

$$|F| = |F \cap Z| + |F - Z| \leq \sum_i |B_i \cap Z| + |S - Z| \leq \sum_i r_i(Z) + |S - Z| \tag{3}$$

from which max $\leq$ min follows. In the estimation (3), equality holds if and only if the following optimailty criteria are met.

$$S - Z \subseteq \cup_i B_i \tag{4}$$

$$B_i \cap B_j \cap Z = \emptyset \text{ for } 1 \leq i < j \leq k \tag{5}$$

$$B_i \cap Z \text{ spans } Z \text{ in } M_i \text{ for } i = 1, \dots, k. \tag{6}$$

We show how the push-relabel technique can be used for finding a subset Z and k basis $B_i \in \mathcal{B}_i$ satisfying the three optimality criteria, completing in this way the proof of Theorem 2.2. At the beginning, Θ is identically 0. At an intermediate stage of the algorithm, we are given M_i-bases B_i for $i = 1, \dots, k$ and a level function $\Theta : S \to \{0, 1, \dots, n = |S|\}$ for which the following level properties hold.

(L1) $\Theta(u) = 0$ holds for every $u \in S$ covered by more than one of the bases B_i.
(L2) $\Theta_{\min}(C_i(B_i, u)) \geq \Theta(u) - 1$ holds for every $u \in S - B_i$.

The algorithm terminates when one of the following **stopping rules** occurs.

(A) $S = B_1 \cup \dots \cup B_k$.
(B) There is an empty level set L_j so that every element under j is covered.

Lemma 2.3. *If* **(A)** *holds, then the bases $\{B_1, \dots, B_k\}$ and $Z^* := \emptyset$ meet the optimality criteria. If* **(B)** *holds, then the bases $\{B_1, \dots, B_k\}$ and $Z^* := \{u : \Theta(u) > j\}$ meet the optimality criteria.*

Proof. The first case is obvious. To see the second, observe that $S - Z^*$ is covered by property **(B)**, Z^* contains no multiply covered element by **(L1)**, and (6) also holds since the emptiness of L_j and **(L2)** imply that $C_i(B_i, s) \subseteq Z^*$ for all $s \in Z^* - B_i$. •

There are two basic operations at an uncovered element s. Lifting s means that $\Theta(s)$ is increased by 1. A **basis-change** at s means that we take a basis B_i along with an element $t \in C_i(B_i, s) - s$ and replace B_i with $B_i - t + s$.

The algorithm runs as follows. At a general step, assuming that neither of the stopping rules holds, we select an uncovered element s for which $\Theta(s) \leq n - 1$. Note that the failure of **(A)** ensures that there is an uncovered element, and we cannot have each uncovered element in L_n since then there would be an empty level L_j and hence **(B)** would be satisfied. If there is a basis B_i $(1 \leq i \leq k)$ and an element $t \in C_i(B_i, s) - s$ for which $\Theta(t) = \Theta(s) - 1$, perform a basis-change by replacing B_i with $B_i - t + s$. By Lemma 2.1, the level properties remain intact. Note that t is the only element that may get uncovered and t is under s. If no such a B_i and t exist anymore, lift s. This operation also maintains the level properties.

The algorithm terminates when lifting s leaves an empty level set such that all elements under s is covered. In this case, **(B)** holds. The other way of termination occurs when after the current basis-change every element is covered in which case **(A)** holds.

Lemma 2.4. *The total number of lifts is at most n^2. The total number of basis-changes is at most n^2.*

Proof. Since one element is lifted at most n times, there are at most n^2 lifts. For an M_i-basis B_i, let $\widetilde{\Theta}(B_i) := \sum[\Theta(v) : v \in B_i]$ and let $\alpha := \sum_i \widetilde{\Theta}(B_i)$. At the beginning, $\alpha = 0$. A lift operation does not affect α since only uncovered elements are lifted. At a basis-change, α is increased by exactly 1. Therefore α is equal to the number of basis-changes. An uncovered element has no contribution to α and neither does an element of L_0. The contribution of a covered element $v \notin L_0$ is $\Theta(v)$. Therefore $\alpha \leq \sum \Theta(v) \leq n^2$. •

Lemma 2.4 completes the proof of the theorem. It also shows that the algorithm is polynomial provided that there is a subroutine for each M_i that determines for an uncovered element s if there is an element $t \in C_i(B_i, s) - s$ for which $\Theta(t) = \Theta(s) - 1$. By denoting the complexity of this subroutine with γ, we can conclude that the overall complexity of the algorithm is $O(\gamma k n^2)$. • •

Remarks Suppose we always select an uncovered element s with $\Theta(s) \leq n - 1$ for which $\Theta(s)$ is *maximum* (the highest level selection rule). If an uncovered element gets covered, it remains so within one phase since a push operation at s may create a newly uncovered element only under s. Hence the number of basis-changes in one phase is at most n and altogether is at most n^3. That is, in this case, though we get a bound worse than the one obtained above, but its proof is immeadiate. The highest level rule will have a more significant role in other submodular frameworks.

Interestingly, the proof of the theorem and of the polynomiality of the algorithm is even easier if we always select an uncovered element s *minimizing* $\Theta(s)$ (lowest level selection rule). In this case, the algorithm terminates either when each element gets covered or at the very first moment when an empty level set arises. After at most n basis-changes either the number of uncovered elements decreases or the level of an element is increased. Since the first event may occur at most n times while the second one at most n^2 times, the total number of basis-changes is at most $n^2 + n^3$.

3 Testing membership in a matroid polytope

Let $M = (S, r)$ be a matroid. The matroid (or independence) polytope $P(r)$ of M is the convex hull of the characteristic vectors of independent sets of M. The base polytope $B(r)$ of M is the convex hull of the characteristic vectors of bases of M. In what follows we will use the slightly sloppy but shorter expression of convex combination of bases. Edmonds [4] proved the following polyhedral descriptions:

$$P(r) = \{x \in \mathbf{R}^S : x \geq 0 \text{ and } \widetilde{x}(Z) \leq r(Z) \text{ for every } Z \subseteq S\}. \tag{7}$$

$$B(r) = \{x \in \mathbf{R}^S : x \geq 0 \text{ and } \widetilde{x}(Z) \leq r(Z) \text{ for every } Z \subseteq S,\ \widetilde{x}(S) = r(S)\}. \tag{8}$$

$P(r)$ and $B(r)$ are often called the matroid (or independence) polyhedron and the base polyhedron of M, respectively.

By Theorem 2.2, S can be covered by k bases if and only if $kr(X) \geq |X|$ holds for every subset X. By (7), this latter property is equivalent to requiring that the vector $\underline{\chi}_S/k$ is in $P(r)$. Cunningham [1] developed a strongly polynomial algorithm to test if a given vector g belongs to $P(r)$. He also solved the more general problem when g is not in $P(r)$ and one is interested in finding a subset most violating (7) along with an element $x \leq g$ of $P(r)$ for which $\widetilde{x}(S)$ is maximum. His approach uses shortest augmenting paths and also the technique of lexicographic selection rule introduced by Schönsleben [19]. The essence of this rule is that it specifies a fixed total ordering on S and in finding the shortest augmenting path in the current auxiliary digraph the algorithm breaks tie by chosing the earliest possible element of the ordering. The lexicographic selection rule of Schönsleben turned out to be an unavoidable device in all combinatorial algorithms concerning submodular frameworks. It was adapted to push-relabel algorithms as well. (See, for example, the works of Fujishige and Zhang [13], [14] and of Iwata and Fleischer [7].)

We describe a simple push-relabel algorithm for the matroid membership problem that does not use the lexicographic rule. The algorithm works for the slightly more general problem when a specified upper bound $g : S \to \mathbf{R}_+$ is given and we are interested in finding a member $x \in P(r)$ for which $x \leq g$ and $\widetilde{x}(S)$ is maximum. Clearly, g belongs to $P(r)$ if and only if this maximum is $\widetilde{g}(S)$. Since both a matroid polyhedron and a box $\{x \in \mathbf{R}^S : 0 \leq x \leq g\}$ is a polymatroid, the following min-max result is a special case of the polymatroid intersection theorem of Edmonds [3].

Theorem 3.1. *Let $M = (S, r)$ be a matroid and $g : S \to \mathbf{R}_+$ function. Then*

$$\max\{\widetilde{x}(S) : x \leq g,\ x \in P(r)\} = \min\{r(Z) + \widetilde{g}(S - Z)\}. \tag{9}$$

Proof. We call an element $x \in P(r)$ **feasible** if $x \leq g$. For a subset $Z \subseteq S$ and for a feasible $x \in \mathbf{R}$, one has

$$\widetilde{x}(S) = \widetilde{x}(Z) + \widetilde{x}(S - Z) \leq r(Z) + \widetilde{g}(S - Z) \tag{10}$$

from which max $\leq$ min follows. In the estimation (10), equality holds if and only if the following optimality criteria are met.

$$\widetilde{x}(Z) = r(Z) \tag{11}$$

$$\widetilde{x}(S-Z) = \widetilde{g}(S-Z). \tag{12}$$

We shall prove the theorem by developing an algorithm that computes a feasible x and a subset $Z \subseteq S$ satisfying the optimality criteria. A convex combination of bases of M will be described by a coefficient function $\lambda : \mathcal{B} \to \mathbf{R}_+$ for which $\sum[\lambda(B) : B \in \mathcal{B}] = 1$. The element of $B(r)$ defined by λ is $x_\lambda = \sum[\lambda(B)\underline{\chi}_B : B \in \mathcal{B}]$. Clearly, a non-negative vector x belongs to $P(r)$ if and only if there is a convex combination x_λ of bases such that x_λ covers x in the sense that $x_\lambda \geq x$. We say that a basis B is **λ-active** (or simply **active**) in the convex combination x_λ if $\lambda(B) > 0$. By a theorem of Charathodory, every element of $B(r)$ can be expressed as a convex combination of at most n bases. An element $s \in S$ is **g-larger, g-smaller** or **neutral** according to whether $g(s) > x_\lambda(s)$, $g(s) < x_\lambda(s)$ or $g(s) = x_\lambda(s)$.

Beside a convex combination x_λ of bases, the algorithm maintains a level function $\Theta : S \to \{0, 1, \ldots, n\}$. In the following **level properties** we use again the notation $\Theta_{\min}(X) := \min\{\Theta(v) : v \in X\}$ for $X \subseteq S$.

(L1) $\Theta(u) = 0$ holds for every g-smaller $u \in S$.
(L2) $\Theta_{\min}(C(B,u)) \geq \Theta(u) - 1$ holds for every λ-active basis B and for every $u \in S - B$.

The algorithm terminates when one of the following **stopping rules** occurs.

(A) There is no g-larger element of S.
(B) There is an empty level set L_j so that there is no g-larger element under j.

Lemma 3.2. *If* **(A)** *holds, then* $g \in P(r)$. *If* **(B)** *holds, then* x^* *and* $Z^* := \{u : \Theta(u) > j\}$ *meet the optimality criteria where* x^* *is defined by* $x^*(u) := \min\{g(u), x_\lambda(u)\}$.

Proof. Suppose **(A)** holds, that is, $g(u) \leq x_\lambda(u)$ for every $u \in S$. By $x_\lambda \in B(r)$, we conclude that $g \in P(r)$.

Suppose **(B)** holds. By **(L1)**, every g-smaller element is under j and hence $x^*(u) = x_\lambda(u)$ for every $u \in Z^*$. By **(L2)**, $C(B,u) \subseteq Z^*$ holds for every active basis B and every element $u \in Z^* - B$. Hence $r(Z^*) = |B \cap Z^*|$ from which $\widetilde{x}^*(Z^*) = \widetilde{x}_\lambda(Z^*) = r(Z^*)$, that is, optimality criterion (11) holds for x^* and Z^*. Since there is no g-larger element under j, $x^*(u) = g(u)$ for every $u \in S - Z^*$ and hence $\widetilde{x}^*(S-Z^*) = \widetilde{g}(S-Z^*)$, that is, optimality criterion (12) holds for x^* and Z^*. •

Basic operations: push and lift Let s be a g-larger element for which $\Theta(z) \leq n-1$. Lifting s means again that we increase $\Theta(s)$ by 1. Push is performed at s when there is an active basis B not containing s for which $\Theta_{\min}(C(B,s)) = \Theta(s) - 1$. Let $t \in C(B,s)$ for which $\Theta(t) = \Theta(s) - 1$, let $B' = B - t + s$, and define $\Delta := \min\{g(s) - x_\lambda(s), \lambda(B)\}$. A **push** decreases $\lambda(B)$ by Δ and increases $\lambda(B')$ by Δ. A push is called **neutralizing** if $\Delta = g(s) - x_\lambda(s)$. In this case s becomes neutral. A non-neutralizing push does not change the number of active bases while a neutralizing push either preserves or increases this number by 1. Note that the only element that may become g-larger after a push operation is t and t is under s. This observation will be used in estimating the number of steps.

Treating a g-larger element s with $\Theta(s) \leq n-1$ means that we apply push operations at s as long as possible. No more push is possible at s when either the last push at s was neutralizing or else when there is no more active basis B not containing s for which $\Theta_{\min}(C(B,s)) = \Theta(s)-1$. In the latter case, lift s.

Observe that if neither of the stopping rules hold, then there is a g-larger element but it is not possible that each g-larger element is in L_n since then there would be an empty level and **(B)** would be satisfied.

The algorithms runs as follows. As long as neither of the stopping rules holds select a g-larger node s for which $\Theta(s) \leq n-1$ and $\Theta(s)$ is maximum (the highest level rule), and treat s. The algorithm terminates either when after a push there are no more g-larger elements, that is, **(A)** holds, or else, when after a lift every g-larger element is in L_n, in which case **(B)** holds.

It follows from Lemma 2.1 and from the rules of the algorithm, that the basic operations preserve the level properties. Due to the highest level rule, if an element gets neutral, it remains so in the same phase. Therefore the number of neutralizing pushes in one phase is at most n and, since the number of phases is at most n^2, altogether there are at most n^3 neutralizing pushes.

The number of non-neutralizing pushes can be estimated as follows. Within one phase, there are at most n treatments and hence the total number of treatments is at most n^3. Within one treatment, there is at most 1 neutralizing push. Since only a neutralizing push can increase the number of active bases, and only by 1, it follows that the number of active bases at the beginning of the j'th treatment is at most j. Therefore the number of neutralizing pushes at the j'th treatment is at most j, and hence the total number of neutralizing pushes is at most $1+2+\cdots+n^3 \leq n^6$.

Therefore the algorithm teminates after at most $O(n^6)$ basic operations. If the complexity of the required subroutine that determines for a basis B and an element $s \in S-B$ if there is an element $t \in C(B,s)-s$ for which $\Theta(t) = \Theta(s)-1$ is γ, then the overall complexity of the algorithm is $O(\gamma n^6)$. • •

Remark If g is rational, then the components of the coefficient vector λ can be chosen as integer multiples of $1/K$ where K is the least common denominator of the components of g.

Improvement By using Gauss elimination and relying on Caratheodory's theorem, it is possible to reduce the number of active basis to n in $O(n^3)$ time provided that the initial number of active bases is linear in n. A naive approach would be to apply Caratheodory every time when a neutralizing push results in $n+1$ active bases. But in this case we may need Caratheodory as many times as the number of treatings, $O(n^3)$, and hence we would get the same complexity $O(n^6)$ as before. It is more efficient to apply Caratheodory only at the end of each phase. If there are n active bases at the beginning of a phase, there will be at most $2n$ active bases at its end since the number of neutralizing pushes within one phase is at most n. Therefore we need altogether at most n^2 applications of Caratheodory, and hence the complexity of this version of the algorithm is $O(n^5)$.

4 Polymatroid intersection and discrete separation

The simple push-relabel algorithm for matroid partition can easily be transformed to one for matroid intersection. Instead of working out the details of this, we consider the polymatroid intersection problem. One version of the polymatroid intersection theorem of Edmonds [3] provides a necessary and sufficient condition for the intersection of two base-polyhedra to be non-empty. It was P. Schönsleben [19] who developed the first polynomial algorithm for polymatroid intersection. His work is remarkable since it introduced the lexicographic technique. This approach has been transformed to push-relabel algorithms, and the lexicographic technique was, in fact, used in all algorithms so far concerning submodular frameworks more general than matroids (like polymatroid intersection, polymatroidal flows by Lawler and Martel [17], [18], and submodular flows in [8] and by Fujishige and Zhang [14]).

Let p be a fully supermodular and b a fully submodular set-function on a ground-set S of n elements for which $b(\emptyset) = 0 = p(\emptyset)$. A base-polyhedron $B(b)$ confined by a fully submodular function b is defined by $B(b) := \{x \in \mathbf{R}^V : \widetilde{x}(Z) \leq b(Z)$ for every $Z \subset V$ and $\widetilde{x}(V) = b(V)\}$. The polyhedron $B'(b) := \{x \in \mathbf{R}^V : \widetilde{x}(Z) \geq p(Z)$ for every $Z \subset V$ and $\widetilde{x}(V) = p(V)\}$ is also a base-polyhedron, since $B'(p) = B(b^*)$ where b^* is defined by $b^*(X) = p(S) - p(S - X)$ for $X \subseteq S$ and hence b^* is submodular. Before turning to the algorithm, we prove a lemma that will be used for submodular flows as well, and plays an analogous role as Lemma 2.1 played at matroids.

Let m be an element of a base-polyhedron $B(b)$. Call a subset $X \subseteq S$ m-**tight** (or just **tight**) if $\widetilde{m}_b(X) = b(X)$. Note that S is always tight. Due to the submodularity of b, the tight sets are closed under taking union and intersection. Let $T(v)$ denote the unique smallest m-tight set containing v. Let $s \in S$ and $t \in T(s) - s$, and let $\Delta := \min\{b(X) - \widetilde{m}(X) : t \in X \subseteq S - s\}$. Define m' as follows.

$$m'(v) := \begin{cases} m(s) + \Delta & \text{if } v = s \\ m(t) - \Delta & \text{if } v = t \\ m(v) & \text{otherwise} \end{cases} \tag{13}$$

Then m' also belongs to $B(b)$. Let $T'(v)$ denote the smallest m'-tight set containing v. If the base-polyhedron $B'(p)$ is confined by a supermodular function p, then $\Delta := \min\{\widetilde{m}(X) - p(X) : t \in X \subseteq S - s\}$ and

$$m'(v) := \begin{cases} m(s) - \Delta & \text{if } v = s \\ m(t) + \Delta & \text{if } v = t \\ m(v) & \text{otherwise} \end{cases} \tag{14}$$

Recall the notation $\Theta_{\min}(X) := \min\{\Theta(u) : u \in X\}$.

Lemma 4.1. *If $\Theta(t) = \Theta_{\min}(T(s))$, then*

$$\Theta_{\min}(T'(u)) \geq \Theta_{\min}(T(u)) \tag{15}$$

for every $u \in S$.

Proof. If $T(u)$ is m'-tight, then $T'(u) \subseteq T(u)$ from which (15) follows. If $T(u)$ is not m'-tight, then $t \in T(u)$ from which $\Theta_{\min}(T(u)) \leq \Theta(t) = \Theta_{\min}(T(s))$. This implies for $T := T(u) \cup T(s)$ that $\Theta_{\min}(T) = \min\{\Theta_{\min}(T(u)), \Theta_{\min}(T(s))\} = \Theta_{\min}(T(u))$. Now T is m-tight as it is the union of two m-tight sets. Since $s, t \in T$, it follows that T is m'-tight, too. Hence $T'(u) \subseteq T$ from which $\Theta_{\min}(T'(u)) \geq \Theta_{\min}(T) = \Theta_{\min}(T(u))$. •

The polymatroid intersection theorem can be formulated in an equivalent form which is sometimes called the discrete separation theorem [8]. Our main goal is to develop a simple push-relabel algorithm for proving this result. There will be however an important difference between this algorithm and the ones described in the previous sections for matroids. Namely, for computing Δ, here we need a subroutine for minimizing certain submodular functions related to b while the matroid algorithms required subroutines only for finding fundamental circuits.

Theorem 4.2. *For a fully supermodular p and a fully submodular b, the intersection $B(b) \cap B'(p)$ of their base-polyhedra is non-empty (that is, there exists a function $m : S \to \mathbf{R}$ separating p and b in the sense that $p \leq \widetilde{m} \leq b$) if and only if $p \leq b$. For integer-valued p and b, the separating m can also be chosen integer-valued.*

Proof. As the necessity of the condition $p \leq b$ is straightforward, we deal with sufficiency. Suppose that $m_b \in B(b)$ and $m_p \in B'(p)$. Call a subset $X \subseteq S$ ***b*-tight** (with respect to m_b) if $\widetilde{m}_b(X) = b(X)$ and ***p*-tight** (w.r. to m_p) if $\widetilde{m}_p(X) = p(X)$. Note that S is always b-tight and p-tight. Due to the sub- and supermodularity of b and p, both the b-tight sets and the p-tight sets are closed under taking union and intersection. Let $T_b(v)$ and $T_p(v)$ denote, respectively, the intersection of b-tight and p-tight sets containing v. Then $T_b(v)$ is b-tight and $T_p(v)$ is p-tight.

The algorithm needs a subroutine to compute $\Delta_p(u,v) = \min\{\widetilde{m}_p(X) - p(X) : u \in X \subseteq S - v\}$ and $\Delta_b(u,v) = \min\{b(X) - \widetilde{m}_b(X) : u \in X \subseteq S - v\}$. We denote the complexity of this subroutine by γ. Since $m_b \in B(b)$ and $m_p \in B'(p)$, these values are non-negative and $\Delta_p(u,v) > 0$, for example, holds precisely if $v \in T_p(u)$.

Call a element $v \in S$ **m_p-smaller**, **m_p-larger**, and **neutral** according to whether if $m_b(v) > m_p(v)$, or $m_b(v) < m_p(v)$, or $m_b(v) = m_p(v)$. Let $\Theta : S \to \{0, 1, \ldots, n\}$ be a level function. Suppose that Θ satisfies the following level properties.

(L1) Every m_p-smaller element is in L_0.
(L2) $\min\{\Theta_{\min}(T_b(v)), \Theta_{\min}(T_p(v))\} \geq \Theta(v) - 1$ for every $v \in S$.

Let $m_b \in B(b)$ and $m_p \in B'(p)$, and let Θ be a level function. We call a triplet (m_b, m_p, Θ) **feasible** if it satisfies the level properties. The algorithm will terminate when one of the following two stopping rules occurs.

(A) There is no more m_p-larger node.
(B) There exists a m_p-larger node s and an empty level set L_ℓ under s (that is, $\ell < \Theta(s)$).

Lemma 4.3. *Let (m_b, m_p, Θ) be a feasible triplet. Then* **(A)** *implies that both m_p and m_b separate p and b while* **(B)** *implies that the set $Z := \{v \in V : \Theta(v) > \ell\}$ violates $p \leq b$.*

Proof. If **(A)** holds, that is, if there are no m_p-larger elements, then we have $p \leq \widetilde{m}_p \leq \widetilde{m}_b \leq b$.

Suppose now that **(B)** holds. By **(L1)** and **(B)**, we have $\widetilde{m}_p(Z) > \widetilde{m}_b(Z)$. Furthermore, **(L2)** implies that $T_b(v) \subseteq Z$ and $T_p(v) \subseteq Z$ for every $v \in Z$. Since the union of b-tight sets (p-tight sets) is b-tight (p-tight), we can conclude that $p(Z) = \widetilde{m}_p(Z) > \widetilde{m}_b(Z) = b(Z)$. •

At an intermediate stage of the algorithm, a feasible triplet (m_p, m_b, Θ) is available for which neither of the two stopping rules holds. The core subroutine of the algorithm is a **treatment** of a m_p-larger element s that suitably changes m_p, m_b, and Θ. A treatment of s ends up either by making s neutral or by lifting s by 1. It consists of applying three basic operations that are used several times.

Three basic operations at an m_p-larger element s

1. Pushing m_p along at s means that we decrease $m_p(s)$ by Δ and increase $m_p(t)$ by Δ where $t \in T_p(s)$ is an element with $\Theta(t) = \Theta(s) - 1$, and $\Delta = \Delta_p(s,t)$.

2. Pushing m_b at s means that we increase $m_b(s)$ by Δ and decrease $m_b(t)$ by Δ where $t \in T_b(s)$ is an element with $\Theta(t) = \Theta(s) - 1$, and $\Delta = \Delta_b(s,t)$.

3. Lifting s.

We say that the push operation is **along** (s,t). A push operation at s is **neutralizing** if it makes s neutral.

Treating an m_p-larger element s means that we apply m_p-pushes and m_b-pushes at s as long as possible. When this is not possible anymore, lift s.

Description of the algorithm The algorithm starts with $\Theta \equiv 0$, $m_p \in B'(p)$, and $m_b \in B(p)$ where m_p and m_b can be found by applying the polymatroid greedy algorithm. At an intermediate stage, a feasible triplet (m_p, m_b, Θ) is available. Suppose that neither of the two stopping rules holds. Then there are m_p-larger elements of S. We cannot have an m_p-larger element s in L_n for otherwise at least one level set under s would be empty, and hence Stopping rule **(B)** would hold. As long as there are m_p-larger elements, the algorithm selects one with highest level (highest level rule) and treats it.

One way of termination is that the current treatment neutralizes s and no more m_p-larger node remains. In this case, Stopping rule **(A)** holds and hence the resulting m_p separates p and b by Lemma 4.3. The other way of termination is that the current treatment lifts s and leaves its (original) level set empty. In this case, Stopping rule **(B)** holds and hence the set $Z = \{v : \Theta(v) \geq \Theta(s)\}$ violates $p \leq b$ by Lemma 4.3 showing that no separating m exists.

Lemma 4.4. *The basic operations preserve the level properties.*

Proof. Since a lift of s is performed only when no more pushes are available at s, the lifting operation preserves **(L2)**. It clearly does not affect **(L1)**. A push operation cannot generate m_p-smaller elements and hence it leaves Property **(L1)** intact.

It follows from Lemma 4.1 that if an m_b-push is performed along (s,t), then $\Theta_{\min}(T'_b(u)) \geq \Theta_{\min}(T_b(u))$ where $T'_b(u)$ denotes the smallest b-tight set containing

u with respect to the revised vector m'_b, and analogously, if an m_p-push is performed along (s,t), then $\Theta_{\min}(T'_p(u)) \geq \Theta_{\min}(T_p(u))$ for every $u \in S$. Hence a push operation also preserves **(L2)**. •

The sufficiency of $p \leq b$ in the theorem follows once we show that one of stopping rules **(A)** and **(B)** occurs after a finite number of basic opertaions. In fact, we shall prove the following polynomial bound.

Theorem 4.5. *The total number of basic operations is $O(n^4)$.*

Proof. We claim that a single treatment of an element s needs $O(n)$ basic operations. Indeed, there may be one neutralizing push, one lift, and $O(n)$ non-neutralizing pushes since if one is carried out along (s,t), then $T'_b(s) \subseteq T_b(s) - t$ and $T'_p(s) \subseteq T_p(s) - t$. A treatment of s is completed when s becomes neutral or when s is lifted. Due to the highest level rule, s will not be m_p-larger again within the same phase. Therefore, in one phase there can be at most n treatments and, since the number of phases is at most n^2, the total number of basic operations is $O(n^4)$. •

This bound can be lowered to $O(n^3)$ if we introduce a consistency rule for chosing t. Also, if there is a subset X violating $p \leq b$, then the algorithm can be modified so that even a subset can be found for which $p(X) - b(X)$ is maximum. But the details will be worked out for the more general framework of submodular flows, the topic of the next sections.

5 Submodular flows

Let $D = (V, A)$ be a directed graph with no parallel edges, and let $f : A \to \mathbf{R} \cup \{-\infty\}$ and $g : A \to \mathbf{R} \cup \{\infty\}$ be two bounding functions for which $f \leq g$. For a function $x : A \to \mathbf{R}$, let $\varrho_x(Z) := \sum[x(e) : e \in A \text{ enters } Z]$, $\delta_x(Z) := \sum[x(e) : e \in A \text{ leaves } Z]$, and

$$\Psi_x(Z) := \varrho_x(Z) - \delta_x(Z).$$

Note that Ψ_x is modular in the sense that $\Psi(Z) = \sum[\Psi_x(v) : v \in Z]$ for $Z \subseteq V$. Moreover, we are given a crossing submodular set-function $b : 2^V \to \mathbf{Z} \cup \{\infty\}$ for which $b(\emptyset) = 0$ and $b(V)$ is finite. A function (or vector) $x : A \to \mathbf{R}$ is a **submodular flow** or a **subflow** if

$$\Psi_x(Z) \leq b(Z) \text{ for every } Z \subseteq V. \tag{16}$$

Since $\Psi_x(V) = 0$ for an arbitrary x, the $b(V)$ must be non-negative and, in fact, $b(V)$ can be rduced to zero. Therefore we assume throughout that $b(V) = 0$. A subflow x is **feasible** if

$$f \leq x \leq g. \tag{17}$$

The set $Q(f, g; b)$ of feasible subflows is called a **submodular flow** (or **subflow**) **polyhedron**. We will say that the subflow or the subflow polyhedron is **confined** by function b. A feasibililty theorem for the non-emptiness of $Q(f, g; b)$ was described in [8]. In the special case of fully submodular functions, it is as follows.

Theorem 5.1. *Let f and g be functions on the edge-set of a digraph $D = (V, A)$ for which $f \leq g$ and let b be a fully submodular function. There is a feasible subflow if and only if $\varrho_f - \delta_g \leq b$, that is, if*

$$\varrho_f(Z) - \delta_g(Z) \leq b(Z) \text{ for every } Z \subseteq V. \tag{18}$$

If each of f, g, b is integral, then (18) *implies the existence of an integral feasible subflow.*

For a feasible subflow z, we have $\varrho_f(Z) - \delta_g(Z) \leq \varrho_z(Z) - \delta_z(Z) \leq b(Z)$, and hence (18) is necessary. For proving sufficiency, an augmenting path type algorithm was constructed in [8]. Fujishige and Zhang [14] developed a push-relabel algorithm for finding a feasible submodular flow. Here we exhibit a largely simplified push-relabel algorithm that either finds a feasible subflow or finds a set Z violating (18), completing this way the proof ot Theorem 5.1.

Due to the modularity of Ψ_x, there is a feasible subflow precisely if there is such an element m of the base-polyhedron $B(b) := \{x \in \mathbf{R}^V : \widetilde{x}(Z) \leq b(Z)$ for every $Z \subset V$ and $\widetilde{x}(V) = b(V)\}$ for which there exists a feasible m-flow. Here an m-flow $x : A \to \mathbf{R}$ is a function for which $\Psi_x(v) = m(v)$ for every $v \in V$.

The algorithm maintains a feasible vector x and a member m of $B(b)$. During the course of the algorithm, we revise gradually both x and m so as to make $\Psi_x(v) \leq m(v)$ at every node v. Once this goal is achieved, the algorithm terminates by returning the final x^* as a feasible subflow. We refer to a node v as **Ψ-larger**, **Ψ-smaller**, or **neutral** according to whether $\Psi_x(v) - m(v)$ is positive, negative, or zero, respectively.

First we concentrate on the special case when the confining function b is fully submodular. Later we will outline how the general case of crossing submodular functions can be reduced to this special case. We can assume that $b(V) = 0$ for if $b(V) < 0$, then Q is empty while if $b(V) > 0$, then lowering $b(V)$ to zero does not change Q and preserves submodularity.

An edge $e \in A$ is **decreasable** or **increasable** when $x(e) > f(e)$ or $x(e) < g(e)$, respectively. Given an $m \in B(b)$, a subset $X \subseteq V$ is **m-tight** (or just **tight**) if $\widetilde{m}(X) = b(X)$. The ground-set V is tight as $m \in B(b)$. Since b is fully submodular, the set of tight sets is closed under taking intersection and union. Therefore there is a unique smallest tight set $T(u)$ containing u for every node $u \in V$. For any two distinct elements u and v of V, define

$$\Delta(u, v) := \min\{b(X) - \widetilde{m}(X) : u \in X \subseteq V - v\}. \tag{19}$$

Then Δ is non-negative since $m \in B(b)$, moreover, $\Delta(u, v) > 0$ if and only if $v \in T(u) - u$. The algorithm below can be used when a subroutine is available for computing $\Delta(u, v)$. The complexity of this subroutine is denoted by γ. Note that a general-purpose subroutine for computing $\Delta(u, v)$ is available via submodular function minimization, but in applications it is often the case that $\Delta(u, v)$ can be computed with the help of an MFMC computation.

5.1 Level properties and stopping rules

At a general stage of the algorithm, a triplet (x, m, Θ) is available where x is a feasible vector, m is a member of $B(b)$, and $\Theta : V \to \{0, 1, \ldots, n\}$ is a level function. As before, by the level $\Theta_{\min}(X)$ of a (non-empty) subset $X \subseteq V$, we mean $\Theta_{\min}(X) = \min\{\Theta(v) : v \in X\}$. Consider the following level properties.

(L1) Every Ψ-smaller node is on the lowest level, that is, in L_0.

(L2′) $\Theta(v) \geq \Theta(u) - 1$ for every increasable edge uv (each increasable edge steps down at most one level).
(L2″) $\Theta(v) \leq \Theta(u) + 1$ for every decreasable edge uv (each decreasable edge steps up at most one level).

(L3) $\Theta_{\min}(T(v)) \geq \Theta(v) - 1$ for every node v.

A triplet (x, m, Θ) is **feasible** if $f \leq x \leq g$, $m \in B(b)$, and Θ fulfils the level properties.

Lemma 5.2. *Let (x, m, Θ) be a feasible triplet. Suppose that L_ℓ is an empty level and $Z := \{v : \Theta(v) \geq \ell\}$ is non-empty. Then Z is m-tight and $\Psi_x(Z) = \varrho_f(Z) - \delta_g(Z)$.*

Proof. The emptiness of L_ℓ and Property **(L3)** imply that $T(v) \subseteq Z$ for every $v \in Z$. It follows that the union $\cup(T(u) : u \in Z)$ is equal to Z. Since the union of tight sets is tight, we can conclude that $\widetilde{m}(Z) = b(Z)$.

The emptiness of L_ℓ implies that every edge e leaving Z steps down at least two levels and hence **(L2′)** implies $x(e) = g(e)$, that is, $\delta_x(Z) = \delta_g(Z)$. Similarly, every edge e entering Z steps up at least two levels and hence **(L2″)** implies $x(e) = f(e)$, that is, $\varrho_x(Z) = \varrho_f(Z)$. Therefore $\Psi_x(Z) = \varrho_f(Z) - \delta_g(Z)$. •

The algorithm terminates when one of the following two stopping rules occurs.

(A) There is no more Ψ-larger node.
(B) There exists a Ψ-larger node s and an empty level set L_ℓ under s (where $\ell < \Theta(s)$).

Lemma 5.3. *Let (x, m, Θ) be a feasible triplet. Then* **(A)** *implies that x is a feasible subflow while* **(B)** *implies that the set $Z := \{v \in V : \Theta(v) > \ell\}$ violates* (18).

Proof. If there is no Ψ-larger node, then $\Psi_x(Y) = \sum[\Psi_x(v) : v \in Y] \leq \sum[m(v) : v \in Y] = \widetilde{m}(Y) \leq b(Y)$ for every $Y \subseteq V$, and hence x is a feasible subflow.

Suppose now that **(B)** holds. By Lemma 5.2, we have $\widetilde{m}(Z) = b(Z)$ and $\Psi_x(Z) = \varrho_f(Z) - \delta_g(Z)$. Since all the Ψ-smaller nodes are in L_0 by **(L1)**, Z contains no Ψ-smaller node. But Z does contain the Ψ-larger s and therefore $\Psi_x(Z) = \sum[\Psi_x(v) : v \in Z] > \widetilde{m}(Z) = b(Z)$, implying that $\varrho_f(Z) - \delta_g(Z) = \Psi_x(Z) > b(Z)$, and hence (18) is violated. •

5.2 The algorithm for fully submodular b

At an intermediate stage, a feasible triplet (x, m, Θ) is available for which neither of the two stopping rules holds. The core subroutine of the algorithm is a **treatment** of a Ψ-larger node s that suitably changes x, m, and Θ. A treatment of s ends up either by making s neutral or by lifting s. It consists of applying three basic operations that are used several times.

5.2.1 Three basic operations at a Ψ-larger node s

1. Edge-push at s changes $x(e)$ on an edge e entering or leaving s, as follows.

(Increasing) If $e = su$ is an increasable edge stepping down from s, then increase $x(e)$ by α where $\alpha := \min\{g(e) - x(e),\ \Psi_x(s) - m(s)\}$.
(Decreasing) If $e = us$ is a decreasable edge stepping up to s, then decrease $x(e)$ by α where let $x' := x - \alpha\underline{\chi}_e$ where $\alpha := \min\{x(e) - f(e),\ \Psi_x(s) - m(s)\}$.

2. Node-push at s to change m applies only when $\Theta_{\min}(T(s)) = \Theta(s) - 1$. It selects an *arbitrary* element $t \in T(s)$ for which $\Theta(t) = \Theta(s) - 1$, increases $m(s)$ by α, and decreases $m(t)$ by α where $\alpha := \min\{\Psi_x(s) - m(s),\ \Delta(s,t)\}$. We say that the node-push is carried out **along** (s,t). Also, this choice of t will be said to be **generic**, to be distinguished from a specific choice introduced and analysed later.

3. Lifting s.

An edge- or node-push at s is **neutralizing** if it converts s neutral. Otherwise (when s remains Ψ-larger), the push is **non-neutralizing**. The core subroutine of the algorithm is as follows.

5.2.2 Treating a Ψ-larger node s

Step 1 (edge-pushes at s) As long as s stays Ψ-larger and there is an increasable edge stepping down from s or a decreasable edge stepping up to s, apply an edge-push at s.

Step 2 (node-pushes at s) Suppose that no more edge-push is possible at s. As long as s stays Ψ-larger and $\Theta_{\min}(T(s)) = \Theta(s) - 1$, apply a node-push at s.

Step 3 (lifting s) When s is still Ψ-larger but no more edge- or node-push is possible at s, apply a node-lift at s.

As mentioned above, a treatment of s terminates when either s becomes neutral or s is lifted.

5.2.3 Description of the algorithm

The algorithm starts with an arbitrary feasible triplet (x, m, Θ). This can be obtained by choosing an arbitrary x with $f \leq x \leq g$, by taking any arbitrary element m of $B(b)$ (which can be determined by the greedy algorithm for base-polyhedra), and by taking Θ to be identically 0.

At an intermediate stage, a feasible triplet (x, m, Θ) is available. Suppose that neither of the two stopping rules holds. Then there are Ψ-larger nodes but none of them can be in L_n since $\Theta(s) = n$ for a Ψ-larger node s would imply that at least one level set under s is empty, and hence Stopping rule **(B)** would occur. As long as there are Ψ-larger nodes, the algorithm selects one with highest level (highest level rule) and treats it.

One way of termination is that the current treatment neutralizes s and no more Ψ-larger node remains. In this case, Stopping rule **(A)** holds and hence the resulting x is a feasible subflow by Lemma 5.3. The other way of termination is that the current treatment lifts s and leaves its (original) level set empty. In this case, Stopping rule **(B)** holds and hence the set $Z = \{v : \Theta(v) \geq \Theta(s)\}$ violates (18) by Lemma 5.3 showing that no feasible subflow exists.

5.3 Correctness and complexity

Consider a transition from a stage to the subsequent one which transforms the current functions x, m, Θ, T, Ψ, respectively, into $x', m', \Theta', T', \Psi'$. It is evident from the definition of the corresponding α that the vector x' arising from x by an edge-push is feasible and the vector m' arising from m by a node-push is in $B(b)$.

Lemma 5.4. *The basic operations preserve the level properties.*

Proof. An edge-push has no effect on **(L3)**. It creates no new Ψ-smaller nodes, so Property **(L1)** is also preserved. Because it operates on an edge uv for which $|\Theta(u) - \Theta(v)| = 1$, Property **(L2)** cannot break down either.

A node-push along (s, t) does not affect x (that is, $x' = x$). Hence it has no effect on **(L2)**. The definition of α at a node-push implies that $\alpha \leq \Psi_x(s) - m(s)$. Therefore s cannot become Ψ-smaller and thus Property **(L1)** is also preserved. Property **(L3)** for m' in place of m follows from Lemma 4.1.

A node-lift of s keeps **(L1)** intact since $\Theta'(s) > \Theta(s)$ only if s is Ψ-larger. It preserves **(L2)**, since node-lift was applied only when there was no increasable edge stepping down from s and no decreasable edge stepping up to s. It preserves **(L3)**, since a node-lift was applied at s only when $\Theta_{\min}(T(s)) = \Theta(s)$. • •

The non-trivial direction of Theorem 5.1 follows once we show that one of stopping rules **(A)** and **(B)** occurs after a finite number of operations. In fact, we shell prove the following polynomial bound.

Theorem 5.5. *The total number of basic operations is $O(n^4)$.*

Proof. We claim that a single treatment of a node s needs $O(n)$ basic operations. Indeed, there may be just one neutralizing push, one node-lift, $O(n)$ non-neutralizing edge-pushes since we excluded parallel edges in D, and finally $O(n)$ non-neutralizing node-pushes since if one is carried out along (s, t), then $T'(s) \subseteq T(s) - t$. A treatment of a node s is completed when s becomes neutral or when s is lifted. Due to the highest level rule, s will not be Ψ-larger again within the same phase. Therefore, in one phase there can be at most n treatments and, since the number of phases is at most n^2, the total number of basic operations is $O(n^4)$. •

5.3.1 Improving the complexity

We describe now a tiny modification of the algorithm above that gives rise to a $O(n^3)$ bound. This is the place where the lexicographic idea of Schönsleben does play a role. The proof, however, will need a little more care. The idea is that we make the generic choice of t during a node-push specific. To this end, we choose a fixed linear ordering of the nodes of D in advance. This is arbitrary but fixed and is specified by a one-to-one function $\varphi : V \to \{1, \dots, n\}$. We will say that u is **earlier than** v if $\varphi(u) < \varphi(v)$.

Recall that a node-push at s is generic when only

$$t \in T(s) \text{ and } \Theta(t) = \Theta(s) - 1 \tag{20}$$

were required for t, meaning that when more than one such node existed it did not matter which of them has been selected. A node-push operation is **φ-consistent** if t is selected to be the earliest node meeting (20). In addition to the highest level rule, the revised algorithm selects t throughout φ-consistently.

Theorem 5.6. *The total number of basic operations, when φ-consistent node-pushes are applied throughout, is $O(n^3)$.*

Proof. In the foregoing proof, we have already pointed out that within one phase there are at most n neutralizing pushes and hence their total number is at most n^3.

Claim 5.7. *The total number of non-neutralizing edge-pushes is at most n^3.*

Proof. A non-neutralizing edge-push on e either increases $x(e)$ to $g(e)$ making e non-increasable or decreases $x(e)$ to $f(e)$ making e non-decreasable. Therefore, after a non-neutralizing edge-push on e, the next edge-push on e can occur only when the sign of $\Theta(s) - \Theta(u)$ has changed. By that time the sum $\Theta(s) + \Theta(u)$ must have been increased by at least two. Hence the number of non-neutralizing edge-pushes on a single edge is at most n and thus the total number of non-neutralizing edge-pushes is at most $|A|n \le n^3$. •

So far we have not used the φ-consistent selection rule. It is needed only for the next lemma which will complete the proof.

Lemma 5.8. *The total number of non-neutralizing node-pushes is at most n^3.*

Proof. Given a feasible triplet (x, m, Θ) at a stage of the algorithm, let

$$B(v) := \{u : u \in T(v),\ \Theta(u) = \Theta(v) - 1\} \quad \text{and} \quad \beta(v) := \min\{\varphi(u) : u \in B(v)\}$$

where $\beta(v)$ is meant to be ∞ if $B(v) = \emptyset$. We use the notational convention that the corresponding functions after performing a basic operation will be denoted by primed letters.

Claim 5.9. *If a φ-consistent node-push is carried out along (s, t), then*

$$\beta'(v) \ge \beta(v) \tag{21}$$

for every node v. Moreover, the inequality is strict when $v = s$ and the node-push is non-neutralizing.

Proof. The definition of φ-consistency shows that $\beta(s) = \varphi(t)$. When $v = s$ and the node-push is non-neutralizing, we have $B'(v) \subseteq B(v) - t$. This and the φ-consistent choice of t imply that $\beta'(v) > \varphi(t) = \beta(s) = \beta(v)$.

Suppose now that $v \neq s$. If $B'(v) \subseteq B(v)$, then (21) follows immediately so we can assume that

$$B'(v) \not\subseteq B(v) \tag{22}$$

and, in paricular, $B'(v) \neq \emptyset$. Then $T'(v) \not\subseteq T(v)$ and hence $T(v)$ is not m'-tight implying that $t \in T(v)$. The set $T := T(v) \cup T(s)$ is m-tight and it is m'-tight, too, since $s, t \in T$. Therefore $T'(v) \subseteq T$. We cannot have $\Theta(t) \geq \Theta(v)$ since then $\Theta_{\min}(T(s)) = \Theta(t)$ would imply $B'(v) \subseteq B(v)$ contradicting (22). Hence $t \in B(v)$, $\Theta(v) = \Theta(s)$, and $B'(v) \subseteq B(v) \cup B(s)$ from which $\beta'(v) \geq \min\{\beta(v), \beta(s)\} = \min\{\beta(v), \varphi(t)\} = \beta(v)$. •

Observe that lifting a node s results in $T'(v) = T(v)$ and $B'(v) \subseteq B(v)$ for every node $v \neq s$ from which $\beta'(v) \geq \beta(v)$ follows. This and Claim 5.9 imply that, as long as the level of a node is fixed, the number of non-neutralizing node-pushes at this node is at most n. Since one node can be lifted at most n times and there are n nodes, the total number of non-neutralizing node-pushes is at most n^3, completing the proof of the lemma and the theorem. • •

Remark The technique of φ-consistent changes was originally introduced by Schönsleben [19] under the name of lexicographic selection rule in an augmenting path type polynomial time algorithm for finding a maximum element of the intersection of two polymatroids. This approach has later been applied to other submodular frameworks such as submodular flow feasibility in [8], testing membership in a matroid polyhedron [1] by Cunningham, optimization over polymatroidal flows [17, 18] by Lawler and Martel, or submodular function minimization [20] by Schrijver. It was also used in a push-ralabel algorithm for submodular flows by Fujishige and Zhang [14] and for submodular function minimization by Fleischer and Iwata [7].

The φ-consistent node-push operation above can be considered as a lexicographic rule for a push-relabel algorithm. In this environment, as we pointed out above, the φ-consitent selection rule is avoidable if our goal is only to have a polynomial time algorithm. Its main role was to help improving on the running time. Interestingly, each polynomial time augmenting paths type algorithm known so far for polymatroid intersection, say, did use the lexicographic selection rule.

6 Extensions

6.1 Most violating set

In the case when no feasible subflow existed, the algorithm described above found a subset Z violating (18). With a slight modification of the procedure even a most violating subset can be computed, where X is most violating if $\varrho_f(X) - \delta_g(X) - b(X)$ is as large as possible.

To this end, the algorithm does not terminate when a node-lift operation leaves a level set empty. Therefore there can be Ψ-larger nodes in level set L_n and this motivates the other modification: s is to be chosen a Ψ-larger node of highest level *under* n. The algorithm terminates when there are no more Ψ-larger nodes under level n. Let x^* denote the feasible vector x and m^* the element of $B(b)$ available at termination. If there are no Ψ-larger nodes at all, then x^* is a feasible subflow. Suppose that the there are Ψ-larger nodes at termination. Then these are in L_0 and there is an empty level set L_j.

Theorem 6.1. *When there are Ψ-larger nodes at termination of the revised algorithm, the set $Z^* := \{v : \Theta(v) > j\}$ violates* (18) *most, and we have the following min-max relation.*

$$\min\Big\{\sum[(\Psi_x(v) - m(v))^+ : v \in V] : \ f \leq x \leq g, \ m \in B(b)\Big\} = \tag{23}$$

$$\max\{\varrho_f(X) - \delta_g(X) - b(X) : X \subseteq V\} \tag{24}$$

If f, g, b are integral, the optimal x in the minimum is attained by an integral vector. Moreover, Z^* (defined above) *is a maximizer in* (24) *while the couple* (x^*, m^*) *is a minimizer in* (23).

Proof. For a feasible vector x, $m \in B(b)$, and subset $Z \subseteq V$, we have

$$\varrho_f(Z) - \delta_g(Z) - b(Z) \leq \varrho_f(Z) - \delta_g(Z) - \widetilde{m}(Z) \leq \varrho_x(Z) - \delta_x(Z) - \widetilde{m}(Z) =$$

$$\sum[\Psi_x(v) - m(v) : v \in Z] \leq \sum[(\Psi_x(v) - m(v))^+ : v \in V]$$

from which max $\leq$ min follows. Here equality holds if and only if (i) $\widetilde{m}(Z) = b(Z)$, (ii) $\varrho_f(Z) = \varrho_x(Z)$, $\delta_g(Z) = \delta_x(Z)$, and (iii) Z contains every Ψ-larger node but no Ψ-smaller nodes.

We are going to show that the triplet (x^*, m^*, Θ) fulfils these optimality conditions. Indeed, by Lemma 5.2, Z^* is m^*-tight, that is, (i) holds. Since no increasable edge leaves Z^* and no decreasable edge enters Z^*, we have $\varrho_f(Z^*) = \varrho_{x^*}(Z^*)$, $\delta_g(Z) = \delta_{x^*}(Z^*)$, that is, (ii) holds. Since every Ψ-larger node is in L_n and every Ψ-smaller node is in L_0, we conclude that Z^* meets (iii), as well. •

Remark Theorem 6.1 can be used to compute a feasible submodular flow x, when one exists, for which $x(e_0)$ is maximum for a specified edge $e_0 = st$ of the digraph.

6.2 Crossing submodular functions

How can one find a member of a subflow polyhedron $Q = Q(f, g; b)$ confined by a crossing submodular function b? We can assume that $b(V) = 0$ for if $b(V) < 0$, then Q is empty while if $b(V) > 0$, then lowering $b(V)$ to zero does not change Q and preserves crossing submodularity. For crossing submodular functions, the feasibility theorem becomes more complicated. Let $\{Z_1, \dots, Z_t\}$ be a partition of Z, and for each Z_i, let $\{Z_i^1, \dots, Z_i^{t_i}\}$ be a partition of $S - Z_i$ ($t_i \geq 1$). Then the set-system $\mathcal{D} = \{S - Z_i^j\}$ is called a **double-partition** of Z. In other words, a double-partition consists of the sets occurring in the co-partitions of the members of a partition of Z. Note that a partition of Z is a special double partition. The following result appeared in [8].

Theorem 6.2. *Let b be crossing submodular for which $b(V) = 0$. There is a feasible subflow if and only if*

$$\varrho_f(Z) - \delta_g(Z) \leq \sum[b(X) : X \in \mathcal{F}]. \tag{25}$$

for every subset $Z \subseteq V$ and every double partition $\mathcal{F}$ of Z. When b is intersecting submodular, $\mathcal{F}$ can be restricted to partitions of Z. If each of f, g, b is integer-valued and there is a feasible subflow, then there is one which is integer-valued. •

First we check whether $B(b)$ is empty or not. This can be done by the bi-truncation algorithm described in [12]. If $B(b)$ turns out to be empty, then the bi-truncation algorithm produces a partition or a co-partition $\mathcal{F}$ of V for which $0 > \sum[b(X) : X \in \mathcal{F}]$. In this case, the algorithm for testing $Q = Q(f, g; b)$ for emptiness concludes that Q is empty and returns $\mathcal{F}$ as a double partition of V which violates (25) since $\varrho_f(V) - \delta_g(V) = 0$.

Suppose now that $B(b)$ is non-empty and a member m of $B(b)$ has already been computed. In this case, there is a unique fully submodular function $b^{\downarrow}$, called the the full truncation of b for which $B(b) = B(b^{\downarrow})$ (see [15] or [10]). Obviously $Q(f, g; b) = Q(f, g; b^{\downarrow})$ and therefore the algorithm described above for fully submodular border functions can, in principle, be applied to $b^{\downarrow}$ in place of b. There are two issues here to be settled.

First, the algorithm above required a subroutine for computing $\Delta_{(b^{\downarrow} - \widetilde{m})}(u, v) := \min\{b^{\downarrow}(X) - \widetilde{m}(X) : u \in X \subseteq V - v\}$. The next lemma shows that it suffices to have a subroutine for computing

$$\Delta_{(b - \widetilde{m})}(u, v) := \min\{b(X) - \widetilde{m}(X) : u \in X \subseteq V - v\}. \tag{26}$$

which is simpler since $\Delta_{(b - \widetilde{m})}(u, v)$ is a function depending directly on the original b and not on its full truncation $b^{\downarrow}$. In applications to connectivity problems, (26) is typically available via a max-flow min-cut computation.

Lemma 6.3. *Let b be a crossing submodular function for which $b(V) = 0$ and m a member of $B(b)$. Then $\Delta_{(b^{\downarrow} - \widetilde{m})}(u, v) = \Delta_{(b - \widetilde{m})}(u, v)$.*

Proof. We need two observations.

Claim 6.4. *$B(b - \widetilde{m}) = B(b^{\downarrow} - \widetilde{m})$ and*

$$(b - \widetilde{m})^{\downarrow} = b^{\downarrow} - \widetilde{m}. \tag{27}$$

Proof. For a vector x with $\widetilde{x}(V) = 0$, we have the following sequence of equivalences. $x \in B(b - \widetilde{m}) \iff \widetilde{x} \leq b - \widetilde{m} \iff \widetilde{x} + \widetilde{m} \leq b \iff x + m \in B(b) \iff x + m \in B(b^{\downarrow}) \iff \widetilde{x} + \widetilde{m} \leq b^{\downarrow} \iff \widetilde{x} \leq b^{\downarrow} - \widetilde{m} \iff x \in B(b^{\downarrow} - \widetilde{m})$. Hence $B((b - \widetilde{m})^{\downarrow}) = B(b^{\downarrow} - \widetilde{m})$.

Now $B(b - \widetilde{m})$ is non-empty since the origin belongs to it. Therefore the full truncation $(b - \widetilde{m})^{\downarrow}$ of $b - \widetilde{m}$ esists and then $B(b - \widetilde{m}) = B((b - \widetilde{m})^{\downarrow})$. This and the firs part imply that $B((b - \widetilde{m})^{\downarrow}) = B(b^{\downarrow} - \widetilde{m})$. Since both $(b - \widetilde{m})^{\downarrow}$ and $b^{\downarrow} - \widetilde{m}$ are fully submodular for which $B(b - \widetilde{m}) = B(b^{\downarrow} - \widetilde{m})$, we conclude that (27) holds. (Here we use the consequence of the polymatroid greedy algorithm of Edmonds that a base-polyhedron uniquely determines its fully submodular bounding function. •

Claim 6.5. *Let h be a non-negative crossing submodular function on ground-set V for which $h(V) = 0$. Then the full truncation $h^{\downarrow}$ of h exists and $\Delta_{h^{\downarrow}} = \Delta_h$.*

Proof. Note that $B(h)$ is non-empty since the non-negativity of h implies that the origin is in $B(h)$. Hence the full truncation of h indeed exists. Since $h^{\downarrow} \leq h$, we have $\Delta_{h^{\downarrow}} \leq \Delta_h$. Let u and v be two elements of V and let Z be a $u\bar{v}$-set for which $\Delta_{h^{\downarrow}}(u, v) = h^{\downarrow}(Z)$.

It was shown in [9] that there exists a double partition $\mathcal{D}$ [9] of Z such that $h^{\downarrow}(Z) = \sum[h(X) : X \in \mathcal{D}]$. It follows from this definition that $d_{\mathcal{D}}(u) = d_{\mathcal{D}}(v) + 1$. Therefore there is a $u\bar{v}$-member Z' of $\mathcal{D}$ and the non-negativity of h implies that $h(Z') \leq h^{\downarrow}(Z)$ from which $\Delta_h(u, v) \leq \Delta_{h^{\downarrow}}(u, v)$ follows implying $\Delta_{h^{\downarrow}}(u, v) = \Delta_h(u, v)$. •

The lemma follows by applying Claim 6.5 to $h := b - \widetilde{m}$. • •

The second problem to be overcome arises when the subflow polyhedron is empty and the algorithm terminates by returning a subset Z for which $\varrho_f(Z) - \delta_g(Z) > b^{\downarrow}(Z)$. In this situation, an algorithm described in [8] for computing a double partition $\mathcal{F}$ of Z for which $b^{\downarrow}(Z) = \sum[b(X) : X \in \mathcal{F}]$ is to be used. A greatly simplified approach is suggested in a recent work [11].

References

[1] W.H. Cunningham, *Testing membership in matroid polyhedra,* Journal of Combinatorial Theory, Series B 36 (1984) 161-188.

[2] J. Edmonds, *Minimum partition of a matroid into independent sets,* J. Res. Nat. Bur. Standards, B69 (1965) 67-72.

[3] J. Edmonds, *Submodular functions, matroids, and certain polyhedra*, in: Combinatorial Structures and their Applications (R. Guy, H. Hanani, N. Sauer, and J. Schönheim, eds.), Gordon and Breach, New York (1970) pp. 69-87.

[4] J. Edmonds, *Matroids and the greedy algorithm*, Math. Programming, 1 (1971) 127-136.

[5] J. Edmonds and D.R. Fulkerson, *Transversals and matroid partition*, Journal of Research of the National Bureau of Standards (B) **69** (1965), 147-153.

[6] J. Edmonds and R. Giles, *A min-max relation for submodular functions on graphs*, Annals of Discrete Mathematics 1, (1977), 185-204.

[7] L.K. Fleischer and S. Iwata, *A push-relabel framework for submodular function minimization and applications to parametric optimization*, Discrete Applied Mathematics, 131, 2 (September 2003) 311-322.

[8] A. Frank, *Finding feasible vectors of Edmonds-Giles polyhedra,* J. Combinatorial Theory Ser. B, Vol. 36, No. 4 (1984) 221-239.

[9] A. Frank, *Applications of submodular functions,* in: Surveys in Combinatorics, London Mathematical Society Lecture Note Series 187, Cambridge Univ. Press, (Ed. K. Walker) 1993, pp. 85-136.

[10] A. Frank, Connections in Combinatorial Optimization, Oxford University Press, 2011 (ISBN 978-0-19-920527-1). Oxford Lecture Series in Mathematics and its Applications, 38.

[11] A. Frank and C. Király, *Tree-compositions and submodular flows*, Technical Reports, 2011- (2011).

[12] A. Frank and É. Tardos, *Generalized polymatroids and submodular flows*, Mathematical Programming, Ser. B. 42 (1988) 489-563.

[13] S. Fujishige and X. Zhang, *New algorithms for the intersection problem of submodular systems*, Japan Journal of Industrial and Applied Mathematics 9 (1992) 369-382.

[14] S. Fujishige and X. Zhang, *A push/relabel framework for submodular flows and its refinement for* $0-1$ *submodular flows*, Opimization, Vol. 38 (1996) 113–154.

[15] S. Fujishige, Submodular functions and optimization, (Second Edition) Annals of Discrete Mathematics, Vol 58, Elsevier (2005).

[16] A.V. Goldberg and E.R. Tarjan, *A new approach to the maximum-flow problem*, J. Association for Computing Machinery, 35 (1988) 921-940.

[17] E.L. Lawler and C.U. Martel, *Flow network formulations of polymatroid optimization problems*, in: Bonn Workshop on Combinatorial Optimization (Bonn, 1980; A. Bachem, M. Grötschel, B. Korte, eds.) Annals of Discrete Mathematics, 16, North-Holland, Amsterdam, (1982) pp. 189-200.

[18] E.L. Lawler and C.U. Martel, *Computing maximal 'polymatroidal' network flows*, Mathematics of Operations Research, 7 (1982) 334-347.

[19] P. Schönsleben, Ganzzahlige Polymatroid Intersections Algorithmen, Ph.D. Thesis, Eidgenössischen Techn. Hochschule, Zürich, (1980).

[20] A. Schrijver, *A combinatorial algorithm minimizing submodular functions in strongly polynomial time*, Journal of Combinatorial Theory, Ser. B. Vol 80 (2000) pp. 575-588.

[21] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, Springer, 2003. Vol. 24 of the series Algorithms and Combinatorics.