University “Alexandru Ioan Cuza” of Iaşi
Faculty of Computer Science

T
E
C
H
N
I
C
A
L

R
E
P
O
R
T

A new strategy for push-relabel algorithm framework for matroid optimization

Emanuel Florentin Olariu

TR 13-01, June 2013

ISSN 1224-9327

Universitatea “Alexandru Ioan Cuza” Iaşi
Facultatea de Informatică
Str. Berthelot 16, 6600-Iaşi, Romania
Tel. +40-32-201090, email: bibl@infoiasi.ro

Abstract

In this paper we present a combinatorial push-relabel algorithm based on lowest level/bfs traversal for matroid optimization. In contrast with other known algorithms our procedure uses lowest level rule and needs no lexicographic order of the elements. Combined with a reduction of the number of active basis our strategy gives a time complexity of $\mathcal{O}(n^6)$.

Keywords: push-relabel algorithms, combinatorial optimization, matroid optimization.

1 Introduction

Push-relabel algorithms are often used in the submodular optimization especially for the submodular flow problem ([3], [4]). This framework includes some interesting combinatorial optimization problems as polymatroid intersection, minimum cost problem, or minimizing submodular functions ([5], [8], [10]). Push-relabel algorithms are an alternative to the procedures using augmenting path technique (see [7] in the context of minimizing submodular functions).

Suppose E is the ground-set of our structure (say, a matroid), the procedure uses a *level function* $\varphi : E \to \{0, 1, \ldots, n\}$. In the push relabel framework (see [6]) we perform a local search, and, if a progress (i.e. a *push*) is not possible, then the level of the current element is increased (this means a *relabel*). The algorithm will terminate when a specific condition (related to the optimization context or the level sets) holds.

The algorithm maintains a level function $\varphi : E \to \{0, 1, \ldots, n\}$ (where $|E| = n$), the level sets are $\Lambda_k = \varphi^{-1}(k)$, $0 \leqslant k \leqslant n$. For a given set $S \subseteq E$, we denote

$$\varphi_{min}(S) = \min_{u \in S} \varphi(u)$$

Let $M = (E, \mathcal{I}, \mathbf{r})$ be a matroid over E, $\mathcal{I}$ being its independence family and $\mathbf{r}$ its rank function(for an exhaustive introduction in matroid theory see [9]). We denote by $\mathcal{B} \in \mathcal{B}$ the basis family, i.e., the collection of maximal inclusionwise independent sets. For a given basis B, and an element $s \in E \setminus B$, we denote by $C(B, s)$ the unique circuit (a maximal non independent set of E) included in $B \cup \{s\}$.

Lemma 1. *([2]) Let $M(E, \mathcal{B})$ a matroid, $B \in \mathcal{B}$, $s \in E \setminus B$, $t \in C(B, s)$, and $B' = B \setminus \{t\} \cup \{s\} \in \mathcal{B}$. If $\varphi(t) = \varphi_{min}(C(B, s))$, then, for each $u \in S - \setminus(B' \cup \{t\})$, we have*

$$\varphi_{min}(C(B', u)) \geqslant \varphi_{min}(C(B, u))$$

Proof. If $t \in C(B, u)$, then $\varphi_{min}(C(B, u)) \leqslant \varphi(t)$; as $u \in C(B, u) \setminus C(B, s)$, there exists a circuit

$$C \subseteq (C(B, u) \cup C(B, u)) \setminus \{t\}, u \in C,$$

by strong circuit elimination axiom - see [9]. $C \subseteq B' \cup \{u\}$, therefore we must have $C = C(B', u)$, and $\varphi_{min}(C(B', u)) \geqslant \min\{t, \varphi_{min}(C(B, u))\} = \varphi_{min}(C(B, u))$.

If $t \notin C(B, u)$, the $C(B, u) = C(B', u)$. □

Remark 1. *Obviously $\varphi_{min}(C(B', t)) = \varphi_{min}(C(B, s))$ as $C(B', t) = C(B, s)$.*

This lemma will be useful for proving that some condition are algorithm invariant.

2 Matroid polytope membership

Let $M = (E, \mathcal{I}, \mathbf{r})$ be a matroid over E. The *independence polytope* of M, $I(\mathbf{r})$, is the convex hull of the characteristic vectors from $\mathcal{I}$, and the *base polytope* of M, $B(\mathbf{r})$, is the convex hull of the characteristic vectors from $\mathcal{B}$. These polytopes have the following characterization (see [1]):

$$I(\mathbf{r}) = \left\{\mathbf{x} \in \mathbb{R}_+^E \;:\; \mathbf{x}(S) \leqslant \mathbf{r}(S), \forall S \subseteq E\right\}$$

$$B(\mathbf{r}) = \{\mathbf{x} \in I(\mathbf{r}) \;:\; \mathbf{x}(E) = \mathbf{r}(E)\}$$

Let $\mathbf{g} : E \to \mathbb{R}_+$ a non-negative vector; we devise a push-relabel algorithm for the following problem: determine a member, $\mathbf{x}$, of the independence polytope $I(\mathbf{r})$ such that $\mathbf{x} \leqslant \mathbf{g}$ and $\mathbf{x}(E)$ is maximum. We recall here a known result followed by its proof:

Theorem 1. *([2]) We have*

$$\max_{\mathbf{x} \leqslant \mathbf{g}, \mathbf{x} \in I(\mathbf{r})} \mathbf{x}(E) = \min_{S \subseteq E} \left[\mathbf{r}(S) + \mathbf{g}(E \setminus S)\right]$$

Proof. If $\mathbf{x} \in I(r)$ and $\mathbf{x}(u) \leqslant \mathbf{g}(u)$, $\forall u \in E$, then

$$\mathbf{x}(E) = \mathbf{x}(S) + \mathbf{x}(E \setminus S) \leqslant \mathbf{r}(S) + \mathbf{g}(E \setminus S)$$

therefore

$$\max_{\mathbf{x} \leqslant \mathbf{g}, \mathbf{x} \in I(\mathbf{r})} \mathbf{x}(E) \leqslant \min_{S \subseteq E} \left[\mathbf{r}(S) + \mathbf{g}(E \setminus S)\right]$$

with equality if and only if

$$\mathbf{x}(S) = \mathbf{r}(S), \text{and} \mathbf{x}(E \setminus S) = \mathbf{g}(E \setminus S),$$

$\mathbf{x}$ being a *feasible* vector ($\mathbf{x} \leqslant \mathbf{g}$, componentwise).

The proof is completed by describing an algorithm which finds a feasible vector $\mathbf{x}$ and a subset $S \subseteq E$ satisfying the above criteria. The algorithm maintains a member $\overline{\mathbf{x}}$ of $B(\mathbf{r})$, i.e., a convex combinations of the characteristic vectors of bases of M:

$$\overline{\mathbf{x}} = \sum_{B \in \mathcal{B}} \lambda_B \cdot \chi_B$$

Following notations from [2] we say that a basis $B \in \mathcal{B}$ is *active* if $\lambda_B > 0$, a element $u \in E$ is $\mathbf{g}$-*larger* ($\mathbf{g}$-*smaller* or $\mathbf{g}$-*neutral*) if $\mathbf{g}(u) > \overline{\mathbf{x}}(u)$ ($\mathbf{g}(u) < \overline{\mathbf{x}}(u)$ or $\mathbf{g}(u) = \overline{\mathbf{x}}(u)$). We will say that a triple (B, s, t) is active if B is an active basis, $s \notin B$, $t \in C(B, s)$, and $\varphi(t) = \varphi_{min}\left(C(B, s)\right) = \varphi(s) - 1$.

Throughout the execution the following two properties are invariant (see lemma 1):

$$(L1) \quad \varphi(u) = 0, \text{ for every } \mathbf{g}\text{-smaller } u \in E$$

$$(L2) \quad \varphi_{min}(C(B, u)) \geqslant \varphi(u) - 1, \text{ for all active basis } B \text{ and } u \in E \setminus B$$

□

The algorithm ends when either (I): there is no $\mathbf{g}$-larger elements or (II): there is a $\Lambda_k = \varnothing$ and all $\mathbf{g}$-larger elements are over k.

Lemma 2. *([2]) If (I) holds, then* $\mathbf{g} \in I(\mathbf{r})$*; if (II) holds, then* $\min\{\overline{\mathbf{x}}, \mathbf{g}\}$ *and* $S = \{u \in E : \varphi(u) \leqslant k+1\}$ *satisfy conditions (L1) and (L2) from above.*

It is important to emphasize that, if there are $\mathbf{g}$-larger elements and (II) does not hold, then we can always pick a $\mathbf{g}$-larger element on a level $\leqslant n-1$. On the current $\mathbf{g}$-larger element s, the algorithm performs one of the following two operations: *lift* s which means an increment of $\varphi(s)$ or *push* at s, if exists an active triple (B, s, t) which consists in a decrease of λ_B and an increase of $\lambda_{B'}$ by the same amount $\Delta = \min\{\mathbf{g}(s) - \overline{\mathbf{x}}(s), \lambda_B\}$, where $B' = B \setminus \{t\} \cup \{s\}$. We call the last operation a *push from s to t*.

There are two types of push at an element s: a *neutralizing* one if $\Delta = \mathbf{g}(s) - \overline{\mathbf{x}}(s)$ and a *common* one otherwise. A *treat* of an element s means push at s while it is possible; when it is no more possible to push, if s was not neutralized, lift s.

After a push from s to t, λ_B becomes $(\lambda_B - \Delta)$, and $\lambda_{B'}$ becomes $(\lambda_{B'} - \Delta)$, therefore

(a) if the push is neutralizing, the number of active basis can increase by one - if B remains active and B' steps out as active;

(b) if the push is a common one ($\Delta = \lambda_B$) B steps out inactive, and B' can become active.

Remark 2. *A neutralizing push can increase the number of active basis by at most one; on the other hand a common push maintains or decreases the number of active basis by one.*

After a push at s with $\varphi_{min}(C(B,s)) = \varphi(t) = \varphi(s) - 1$, and $t \in C(B,s)$, $\overline{\mathbf{x}}(t)$ decreases by Δ, therefore there are three cases related to t:

(a) t was $\mathbf{g}$-larger and remains so;

(b) t was $\mathbf{g}$-smaller and remains so or becomes larger or neutral;

(c) t was $\mathbf{g}$-neutral and becomes larger.

Remark 3. *If the $\mathbf{g}$-larger element s is chosen to be on the lowest level, then t cannot be $\mathbf{g}$-larger, and if it is $\mathbf{g}$-smaller, then $\varphi(s) = 1$ and $\varphi(t) = 0$.*

The time complexity of this procedure heavily depends on the strategy of choosing the element to be treated. Our approach uses a *bfs* like procedure starting with a $\mathbf{g}$-larger element s from a minimum possible level. Such an element is chosen whenever the queue becomes empty and the stopping rules are not yet fulfilled.

The execution of the algorithm can be decomposed in phases: a *phase* is the segment of execution between two liftings. Each phase consists of a variable number of waves: a *wave* starts when the queue is empty and a new $\mathbf{g}$-larger element is added to it and will end when the queue becomes again empty or a lifting arises - whichever comes first.

Lemma 3. *There are at most n^2 phases.*

Proof. This is obvious as each element starts from the zero level and can reach, at most, the nth level, hence it supports at most n liftings. □

Lemma 4. *Each phase consists of at most n waves.*

Proof. Each wave starts with a new $\mathbf{g}$-larger element; we will prove that the starting element is neutralized during its wave and remains so through the entire phase except perhaps at the last wave - in this case the starting element supports a lifting. Hence the starting elements of distinct waves are different.

Suppose that W_i is the set of elements corresponding to the ith wave, $1 \leqslant i \leqslant q$ and s_i is the corresponding starting element. During this wave Algorithm 1 builds a *bfs* tree T_i (following *parent* vector, its root is s_i), all its interior vertices being neutralized elements - if $i < q$; obviously the leaves

Algorithm 1 A push-relabel algorithm

```
$\mathcal{Q} \leftarrow \varnothing$;
for $u \in E$ do
    $visited[u] \leftarrow 0$; $parent[u] \leftarrow 0$;
end for
while (not(A) and not(B)) do
    let $s$ be a $g$-larger, $\Theta$-minimum element; $\mathcal{Q}.push(s)$;        ▷ when $\mathcal{Q}$ becomes empty
    while $\mathcal{Q} \neq \varnothing$ do
        $s = \mathcal{Q}.pop()$; $visited[s] \leftarrow 0$;
        if ($\exists B$ an active basis, $s \notin B \ni t$, $\Theta(t) = \Theta(s) - 1$) then        ▷ $\Theta_{min}(B) = \Theta(t)$
            $push(B, s, t)$;
            if ($x(t) < g(t)$) and $visited[t] = 0$ then        ▷ if $t$ becomes $g$-larger
                $\mathcal{Q}.push(t)$; $visited[t] \leftarrow 1$; $parent[t] = s$;
            end if
        else if $g(s) > x(s)$ then        ▷ if $s$ is still $g$-larger
            $\Theta(s) + +$; $\mathcal{Q} \leftarrow \varnothing$;
            for $u \in E$ do
                $visited[u] \leftarrow 0$; $parent[u] \leftarrow 0$;
            end for
        end if
    end while
end while
```

of this tree are all **g**-smaller or -neutral elements, hence $s_{i+1} \notin W_i$ (in particular $\varphi(s_i) \leqslant \varphi(s_{i+1})$). It is worthly to note that all these leaves are in Λ_0, as an element becomes **g**-neutral only if was -smaller - see Remark 2.

□

Remark 4. *In the last wave of a phase the element which remains* **g***-larger after its treatment is a leave or a parent of a leave in the subsequent tree.*

Lemma 5. *If each wave starts with n active basis, then every phase contains at most $2n^3$ common pushes.*

Proof. In a wave, after the ith neutralization the number of active basis is at most $n + i$, hence the number of common pushes during the ith treatment is less than $n + i$. It follows that the total number of common pushes within a wave is less than

$$n + (n+1) + \ldots + (n + n - 1) < 2n^2$$

□

Our final results concerns the total time complexity of the above algorithm.

Theorem 2. *Suppose γ is the time used for finding an active triple (B, s, t) for a given s. If we reduce the number of the active basis, after each wave, to at most n, the time complexity of the algorithm is $\mathcal{O}(\gamma n^6)$.*

Proof. In a wave which starts with at most n active basis, at the end the number of active basis is $\leqslant 2n$. Using a classic result of Caratheodory (see [7], [10], [2]) we can reduce, in $\mathcal{O}(n^3)$, this number again to at most n. If we include this reduction in the corresponding wave, the complexity of each wave becomes $\mathcal{O}(\gamma n^3)$, hence a phase has a complexity of $\mathcal{O}(\gamma n^4)$ (see lemmata 4 and 5). Now, using lemma 3, we get a total time complexity of $\mathcal{O}(\gamma n^6)$.

□

3 Conclusion

In this paper we show that the lowest level rule for choosing the curent element to be treated gives a good strategy for the push-relabel algorithm framework. In this study we use this rule in combination with a bfs traversal, the pure lowest level rule would give a *dfs* traversal which, for now, does not give the same good results.

The algorithm complexity is the same obtained in [2] using highest level rule, although in combination with active basis reduction this rule can give better results. As our rule offers more interesting properties of the treated elements, a future work can include the adaptation of this rule to the submodular flow algorithm.

Acknowledgments We thank professor C. Croitoru for this useful remarks.

References

[1] Edmonds, J., *Matroids and the greedy algorithm*, Math. Progr. 1, pp. 127-136, 1971.

[2] Frank, A., Miklós, Z., *Simple push-relabel algorithms for matroids and submodular flows*, Japan Journal of Industrial and Applied Mathematics, vol. 29, Issue 3, pp, 419-439, 2012

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

[4] Fujishige, S., S. Iwata, *Algorithms for Submodular Flows.* IEICE TRANS. Inform. Syst., 2000.

[5] Fujishige, S., *Submodular Functions and Optimization*, 2nd edition,Annals of Discrete Mathematics, vol. 58, Elsevier, 2005.

[6] Goldberg, A. V., Tarjan, E. R., *A new approach to the maximum-flow problem*, Jornal of the ACM, vol. 35, issue 4, pp. 921-940, 1988.

[7] Iwata, S., L. Fleischer, S. Fujishige, *A Combinatorial, Strongly Polynomial-Time Algorithm for Minimizing Submodular Functions*, Journal of ACM, vol. 48, pp. 761-777, 2001.

[8] Iwata, S., J. B. Orlin, *A simple combinatorial algorithm for submodular function minimization*, Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2009.

[9] Oxley, J., G., *Matroid Theory*, Oxford University Press, 1992.

[10] Schrijver, A., *A combinatorial algorithm minimizing submodular functions in strongly polynomial time*, Journal of Combinatorial Theory Series B, vol. 80, Issue 2, pp. 346-355, 2000.