University of Konstanz
Department of Computer & Information Science
Prof. Dr. Ulrik Brandes / Dr. Sabine Cornelsen / Melanie Badent / Natalie Indlekofer

Contents

6.2.3 The Push-Relabel Algorithm of Goldberg & Tarjan

So far a flow in a flow network

$$(D = (V, E \subset V \times V), s, t \in V, c : E \to \mathbb{R}_0^+)$$

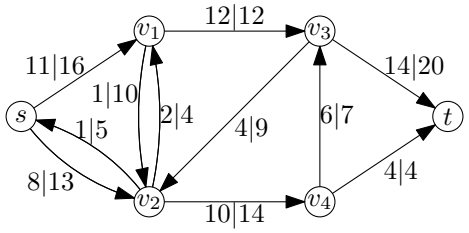
was a function

$$f : E \to \mathbb{R}_0^+$$

fulfilling the capacity constraints ($f \leq c$) and the flow conservation (incoming flow = outgoing flow – for all vertices except for the source s and the sink t). The maximum value of a flow (amount of flow reaching the sink from the source) was determined by iteratively augmenting flow on an $s-t$-path (method of Ford & Fulkerson). The algorithm of Edmonds & Karp chooses in each step a shortest augmenting path and yields a run time in $\mathcal{O}(|E|^2|V|)$.

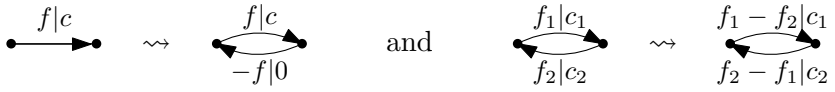
Example 1.

$w(f) = 8 + 11 - 1 = 4 + 14 = 18$
augmenting path, e.g., $\langle s, v_2, v_3, t\rangle$

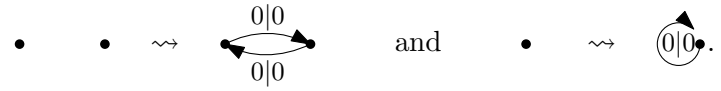


arc label $f|c$.

Instead of pushing flow over an entire augmenting path, the algorithm of Goldberg and Tarjan (1988) pushes flow only along single edges – making decisions locally. Vertices may have an overflow during the execution of the algorithm. For an easier notation, we use the following definition of flow. Idea:



as well as



Definition 1. *Let V be a finite set, $s,t \in V$ and $c : V \times V \to \mathbb{R}_0^+$. The function*

$$f : V \times V \to \mathbb{R}$$

is a flow (Fluss) *in (V,s,t,c) if*

(1) $f(v,w) \le c(v,w)$ for all $v,w \in V$ *(capacity constraint – Kapazitätsbedingung)*

(2) $f(v,w) = -f(w,v)$ for all $v,w \in V$ *(skew symmetry – Antisymmetrie)*

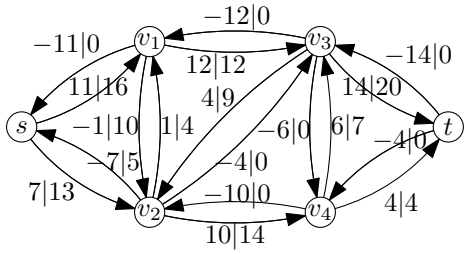
(3) $\sum_{u \in V} f(u,v) = 0$ for all $v \in V \setminus \{s,t\}$ *(flow conservation – Flusserhaltung)*

The value (Wert) *of a flow is*

$$w(f) := \sum_{v \in V} f(s,v) = \sum_{v \in V} f(v,t).$$

Example 2.

$w(f) = 7 + 11 = 4 + 14 = 18$



arc label $f|c$.

plus all loops and arcs between all other pairs of vertices with $f = c = 0$

Remark 1. *The above definition of flow corresponds to the previous definition.*

- Let f be a flow in $(D = (V,E), s,t \in V, c : E \to \mathbb{R}_o^+)$. For $u,v \in V$ let

$$c'(u,v) = \begin{cases} c(u,v) & \text{if } (u,v) \in E \\ 0 & \text{else} \end{cases}$$

$$f'(u,v) = \begin{cases} f(u,v) & \text{if } (u,v) \in E, (v,u) \notin E \\ -f(v,u) & \text{if } (u,v) \notin E, (v,u) \in E \\ f(u,v) - f(v,u) & \text{if } (u,v), (v,u) \in E \\ 0 & \text{else} \end{cases}$$

  Then f' is a flow in (V,s,t,c') and $w(f) = w(f')$.

- Let $f : V \times V \to \mathbb{R}$ be a flow in $(V, s, t \in V, c : V \times V \to \mathbb{R}_0^+)$. Let

$$E = \{(u, v) \in V \times V;\ c(u, v) > 0, u \neq v\}$$

and for $(u, v) \in E$ let

$$f'(u, v) = \max(f(u, v), 0).$$

Then f' is a flow in $(D = (V, E), s, t, c|_E)$ with $w(f') = w(f)$.

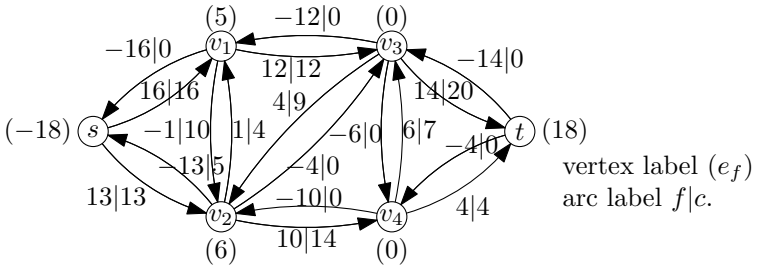
Definition 2. *Let V be a finite set, $s, t \in V$ and $c : V \times V \to \mathbb{R}_0^+$. The function $f : V \times V \to \mathbb{R}$ is a* preflow (Präfluss) *in (V, s, t, c) if f fulfills the capacity and skew symmetry constraints and*

(3') $e_f(v) := \sum_{u \in V} f(u, v) \geq 0$ for all $v \in V \setminus \{s\}$. *(flow excess – Flussüberschuss)*

The residual capacity (Restkapazität) *of a pair $(u, v) \in V \times V$ with respect to a preflow f is*

$$\Delta_f(u, v) = c(u, v) - f(u, v).$$

Example 3.



plus all loops and arcs between all other pairs of vertices with $f = c = 0$

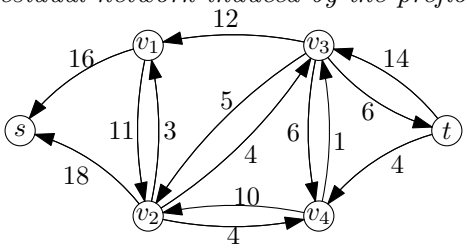
Remark 2. *If $\Delta_f(u, v) > 0$ and $e_f(u) > 0$ then the preflow f can be augmented on the pair $(u, v) \in V \times V$ by $min(\Delta_f(u, v), e_f(v))$.*

E.g., in Ex. 3 an amount of 4 could be pushed from v_2 to v_3 or to v_4. An amount of 6 could be pushed from v_2 back to s.

Definition 3. *Let f be a preflow in (V, s, t, c).*

- *A pair $(v, w) \in V \times V$ with $\Delta_f(v, w) > 0$ is a* residual arc (Residualkante) *with respect to f.*
- *$D_f = (V, E_f)$ with $E_f = \{(v, w) \in V \times V;\ \Delta_f(v, w) > 0\}$ is the* residual network (Residualgraph) *of D induced by f.*

Example 4. *The residual network induced by the preflow in Ex. 3.*



arc label Δ_f.

The idea of the algorithm is to push flow excess to the sink along "estimated" shortest paths in the residual network or – if this is not possible – back to the source, again along "estimated" shortest paths in the residual network. To decide on which edge we should push flow next, we consider the vertices on platforms with increasing height. Flow may only be pushed from a higher platform to a lower one.

Definition 4. *The* height function

$$h : V \to \mathbb{N}_0$$

is compatible *with a preflow f in (V, s, t, c) if*

(1) $h(s) = |V|, h(t) = 0$, *and*

(2) $h(v) \leq h(w) + 1$ *for* $(v, w) \in E_f$.

That means, if the height is compatible with a preflow f then the height cannot decrease by more than one on a residual arc. (It might, however, increase, remain the same or decrease by one.) This means especially that an arc with positive flow can never go uphill by more than one.

For a preflow f in (V, s, t, c) let $d_f(v, w)$ be the length of a shortest $v - w$-path in D_f.

Lemma 3. *Let h be a height function that is compatible with a preflow f in (V, s, t, c). Then*

(1) $d_f(v, w) \geq h(v) - h(w)$ *for* $v, w \in V$,

(2) $d_f(v, s) \geq h(v) - |V|$ *for* $v \in V$,

(3) $d_f(v, t) \geq h(v)$ *for* $v \in V$, *and*

(4) $d_f(s, t) \geq |V|$.

Proof. Let $P : v = v_0, \ldots, v_\ell = w$ be a shortest $v - w$-path in D_f. Then

$$h(w) = h(v_\ell) \geq h(v_{\ell-1}) - 1 \geq \cdots \geq h(v_0) - \ell = h(v) - d_f(v, u).$$

This proves the first statement. The other three follow immediately. □

Corollary 4. *If a flow has a compatible height function it is maximum.*

Proof. By the fourth condition of the previous lemma it follows that there is no $s-t$-path in the residual network. Hence, if f is a flow, then there is no augmenting path with respect to f and, thus, f is a maximum flow. □

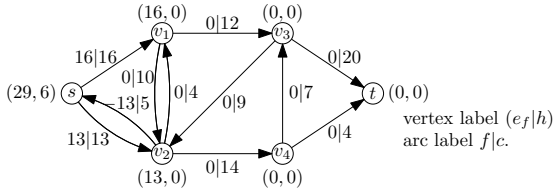
A vertex $v \in V$ is called *active* if $v \notin \{s,t\}$ and $e_f(v) > 0$. The idea of the algorithm is to change a preflow f and a compatible height function h until f becomes a flow, i.e., until there are no more active vertices. The pseudocode of the algorithm can be found in Algorithm 1. An illustration of the algorithm is given in Fig. 1+2.

Algorithm 1: Goldberg & Tarjan

Input : finite set V, $s,t \in V$, $c : V \times V \to \mathbb{R}_0^+$
Output: maximum flow $f : V \times V \to \mathbb{R}$ in (V,s,t,c)
Data : height function $h : V \to \mathbb{N}_0$, compatible with f

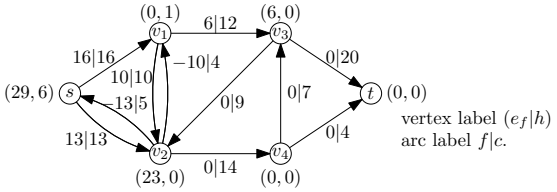
$$\left.\begin{array}{rcll} \Delta_f(v,w) & = & c(v,w) - f(v,w), & v,w \in V \\ e_f(v) & = & \sum_{u \in V} f(u,v), & v \in V \end{array}\right\} \text{ to be updated after each change of } f.$$

```
GOLDBERG-TARJAN(V, s, t, c)
    for v, w ∈ V do
        f(v, w) ← 0;
    for v ∈ V \ {s} do
        f(s, v) ← c(s, v); f(v, s) ← −f(s, v);
        h(v) ← 0;
    h(s) ← |V|;
    while there is a v ∈ V \ {s, t} with e_f(v) > 0 do
        if there is a (v, w) ∈ E_f with h(v) > h(w) then
            PUSH(v, w);
        else
            RELABEL(v);
    return f;

PUSH(v, w)
    Δ ← min{e_f(v), Δ_f(v, w)};
    f(v, w) ← f(v, w) + Δ; f(w, v) ← −f(v, w);

RELABEL(v)
    h(v) ← min{h(w) + 1; Δ_f(v, w) > 0};
```



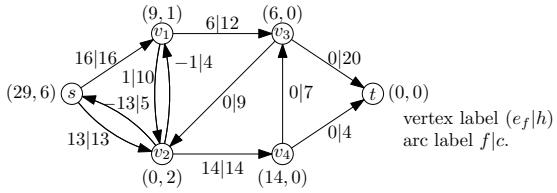
active vertices: v_1, v_2

(a) After initialization



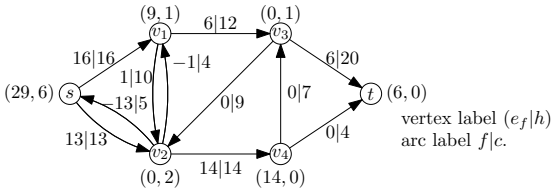
active vertices: v_2, v_3

(b) $R(v_1), P(v_1, v_2), P(v_1, v_3)$



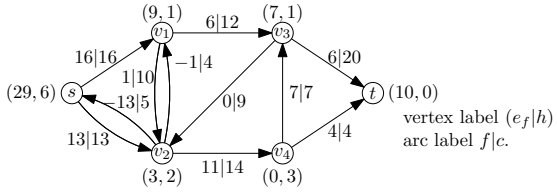
active vertices: v_3, v_4, v_1

(c) $R(v_2), P(v_2, v_4), R(v_2), P(v_2, v_1)$



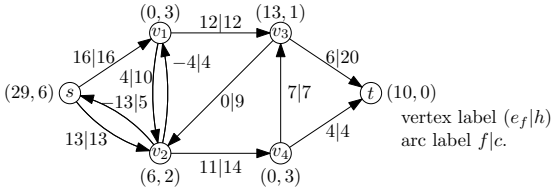
active vertices: v_4, v_1

(d) $R(v_3), P(v_3, t)$



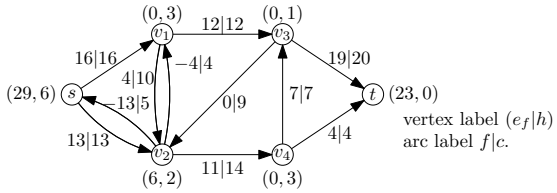
active vertices: v_1, v_3, v_2

(e) $R(v_4), P(v_4, t), R(v_4), P(v_4, v_3), R(v_4), P(v_4, v_2)$



active vertices: v_3, v_2

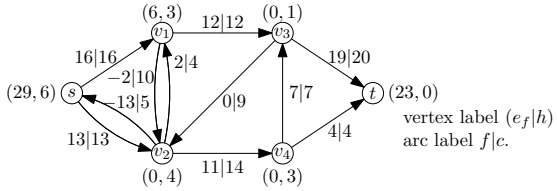
(f) $R(v_1), P(v_1, v_3), R(v_1), P(v_1, v_2)$



active vertices: v_2

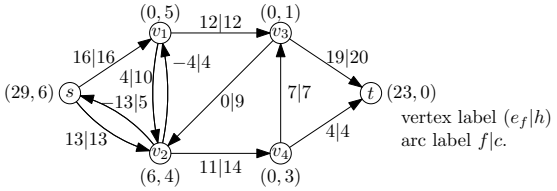
(g) $P(v_3, t)$

Figure 1: Illustration of the algorithm of Goldberg & Tarjan. First part: pushing flow to the sink. Arcs with zero capacity are implicit. $R = \textsc{Relabel}$, $P = \textsc{Push}$.



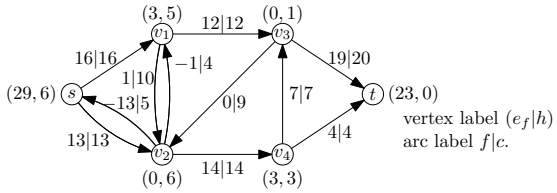
active vertices: v_1

(a) $R(v_2), P(v_2, v_1)$



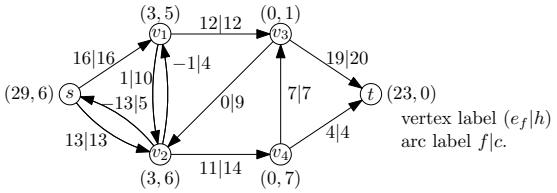
active vertices: v_2

(b) $R(v_1), P(v_1, v_3)$



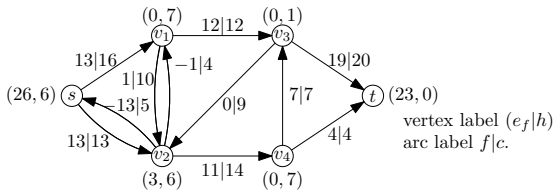
active vertices: v_4, v_1

(c) $P(v_2, v_4), R(v_2), P(v_2, v_1)$



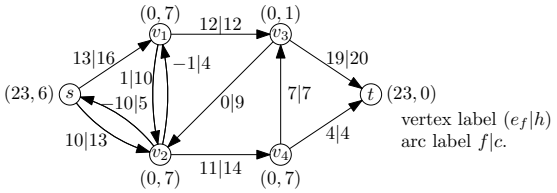
active vertices: v_1, v_2

(d) $R(v_4), P(v_4, v_2)$



active vertices: v_2

(e) $R(v_1), P(v_1, s)$



(f) $R(v_2), P(v_2, s)$

Figure 2: Illustration of the algorithm of Goldberg & Tarjan. Second part: pushing flow back to the source. Arcs with zero capacity are implicit. $R = \textsc{Relabel}$, $P = \textsc{Push}$.

6.2.3.1 Proof of Correctness

Note that RELABEL(v) is well defined if

$$v \text{ active} \quad \Longrightarrow \quad \{w \in V;\ \Delta_f(v,w) > 0\} \neq \emptyset$$

This is guaranteed by the following lemma.

Lemma 5. *Let f be a preflow in (V,s,t,c). Then there is a $v-s$-path in D_f for each $v \in V$ with $e_f(v) > 0$.*

Proof. Let $v \in V$ with $e_f(v) > 0$ and let

$$S_v = \{w \in V;\ \text{ there is a } v-w\text{-path in } D_f\}.$$

Assume $s \notin S_v$. Then $e_f(w) \geq 0$ for all $w \in S_v$. Hence, since $v \in S_v$ and $e_f(v) > 0$ we have on one hand

$$\sum_{w \in S_v} e_f(w) > 0.$$

On the other hand, we have

$$\begin{aligned}
\sum_{w \in S_v} e_f(w) &= \sum_{\substack{u \in V \\ w \in S_v}} f(u,w) \\
&= \sum_{\substack{u \in V \setminus S_v \\ w \in S_v}} \underbrace{f(u,w)}_{\substack{\leq 0 \\ \text{(see (!) below)}}} + \underbrace{\sum_{u,w \in S_v} f(u,w)}_{\substack{=0 \\ \text{(by skew symmetry)}}} \\
&\leq 0
\end{aligned}$$

Hence the assumption was wrong and, thus, $s \in S_v$. It remains to show (!): Let $u \in V \setminus S_v$ and $w \in S_v$. Then $(w,u) \notin E_f$. Hence

$$0 = \Delta_f(w,u) = \underbrace{c(w,u)}_{\geq 0} \underbrace{-f(w,u)}_{=f(u,w)} \geq f(u,w). \quad \square$$

To show that during the algorithm of Goldberg & Tarjan indeed f is a preflow and h is a compatible height function, we need the following lemma.

Lemma 6. *Each operation* RELABEL(v) *increases the height of v. Thus, the height of a vertex never decreases.*

Proof. RELABEL(v) is only executed if $h(w) \geq h(v)$ for all $w \in V$ with $\Delta_f(v,w) > 0$. Since $h(v) \leftarrow h(w) + 1$ for some $w \in V$ with $\Delta_f(v,w) > 0$ it follows that RELABEL(v) increases $h(v)$ by at least 1. $\square$

Lemma 7. *During the algorithm of Goldberg & Tarjan*

(1) f is a preflow and

(2) h is a height function that is compatible with f.

Proof. Clearly, at any time during the algorithm of Goldberg & Tarjan, f is a preflow and h is a height function. We show by induction on the number of operations that h is always compatible with f.

After the Initialization: $h(s) = |V|, h(t) = 0$, and for each $(v, w) \in E_f$ we have

$$h(v) = 0 \leq 1 \leq h(w) + 1.$$

After a Push(v, w)**:** The only possibly new edge in E_f is (w, v). By construction, we have

$$h(w) < h(v) < h(v) + 1.$$

After a Relabel(v)**:** The vertices s and t are not relabeled. Hence, $h(s) = |V|$ and $h(t) = 0$ remain true. Further, if there is a $w \in E$ with

- $(v, w) \in E_f$ then $h(v) \leq h(w) + 1$ by construction.
- $(w, v) \in E_f$ then we had by the inductive hypothesis that $h(w) \leq h(v) + 1$ before the operation RELABEL(v). Since the operation RELABEL(v) increases the height of v it follows that we now have even $h(w) \leq h(v)$. □

Corollary 8. *If the algorithm of Goldberg & Tarjan terminates, it constructs a maximum flow.*

Proof. If the algorithm terminates, only the source and the sink may have positive flow excess. Hence, f is a flow. Since h is compatible with f, it follows that f is maximum. □

6.2.3.2 Run Time Analysis

Let $E = \{(v, w) \in V \times V;\ c(v, w) > 0; v \neq w\}, n = |V|$, and $m = |E|$. Since the algorithm operates only in the connected component of $D = (V, E)$ that contains s we assume that D is connected and that, hence, $m \geq n - 1$.

Lemma 9. *Let the height function h be compatible with a preflow f. Then*

$$h(v) \leq 2n - 1 \text{ for all } v \in V.$$

Proof. Recall that $d_f(v, s) \geq h(v) - h(s)$. Hence

$$h(v) \leq d_f(v, s) + h(s) \leq n - 1 + n. \quad \square$$

Corollary 10. *For each $v \in V$ there are at most $2n-1$ operations* Relabel(v)*. Hence, the total number of* Relabel*-operations is in $\mathcal{O}(n^2)$.*

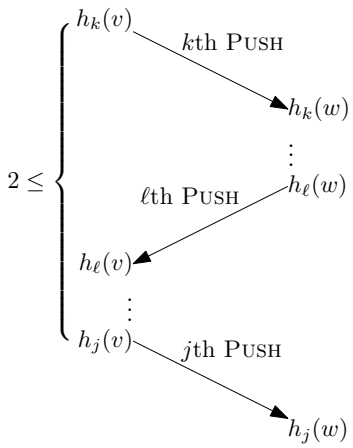
We distinguish two types of Push-operations.

Definition 5. *An operation* Push(v,w) *is* saturating *if thereafter $\Delta_f(v,w) = 0$ and* non-saturating *otherwise.*

Lemma 11. *For each $(v,w) \in V \times V$ an operation* Push(v,w) *can be at most n times saturating.*

Proof. Let $(v,w) \in V \times V, v \neq w$. Let h_i be the height function before the ith Push. Let $j < k$ such that the jth and the kth Push operation is a saturating operation Push(v,w). Note that (v,w) is no more a residual arc after a saturating operation Push(v,w) and it can only become a residual arc again after an operation Push(v,w). Hence, there is a $j < \ell < k$ such that the ℓth Push-operation is an operation Push(w,v). Hence,

$$h_k(v) = 1 + h_k(w) \geq 1 + h_\ell(w) = h_\ell(v) + 2 \geq h_j(v) + 2$$



Further

$$1 \leq h_i(v) \leq 2n-1$$

for all i such that the ith Push operation is an operation Push(v,w). Hence there are at most n saturating Push(v,w). □

Corollary 12. *There are at most $2mn$ saturating* Push*-operations.*

Proof. For each of the m arcs $(v,w) \in E$ both, (v,w) and (w,v) can be a residual arc. Hence, there are at most $2mn$ saturating Push operations. □

Lemma 13. *There are at most $4n^2m$ non-saturating* Push*-operations.*

Proof. Consider the potential function

$$\phi := \sum_{v \text{ active}} h(v)$$

after the following operations.

Initialization: $\phi = 0$.

Relabel(v)**:** All operations Relabel(v) together increase ϕ by the maximal height of v, i.e., by at most $2n-1$.

saturating Push(v,w): One operation PUSH(v,w) increases ϕ by at most $h(w)$, i.e., by at most $2n-1$.

non-saturating Push(v,w): v has been deactivated and w might have been activated. Hence, ϕ decreases by at least $h(v)-h(w)=1$.

Hence, the total increase of ϕ is at most

$$(2n-1)(n-2)+(2n-1)2nm \overset{m\geq n-1\geq 0}{\leq} 4n^2m.$$

Since ϕ is never negative it follows that the total decrease of ϕ is at most its total increase. Hence, the total number of non-saturating PUSH-operations is at most $4n^2m$. □

Theorem 14. *The algorithm of Goldberg & Tarjan terminates after $\mathcal{O}(n^2m)$ operations* PUSH *and* RELABEL.

Choice of Operation Push / Relabel. Store for $v \in V$ the possible neighbors

$$\{w \in V;\ c(v,w)>0 \text{ or } c(w,v)>0\}$$

of v in D_f as a list $N(v)$. Further, let the last element of $N(v)$ be the dummy vertex NIL. Let $p(v)$ be a pointer to the next element in $N(v)$ that has to be considered. $p(v)$ is used and updated as follows.

Initialization: $p(v) \leftarrow$ NIL

Push(v,w) or Relabel(v)? Perform the following operation.

(1) While ($p(v) \neq$ NIL and ($\Delta_f(v,p(v))=0$ or $h(v) \leq h(p(v))$)) move $p(v)$ to the next vertex in $N(v)$.

(2) If $p(v) \neq$ NIL then PUSH$(v,p(v))$.

(3) Else RELABEL(v) and move $p(v)$ to the first vertex of $N(v)$.

Lemma 15. *If $p(v)=$ NIL then there is no $w \in V$ for which* PUSH(v,w) *can be performed.*

Proof. First, note that $h(v)$ is not increased during a scan of $N(v)$. Consider the cases in which $p(v)$ is moved from w.

$h(w) \geq h(v)$: No operation PUSH(v,w) can be performed before $h(v)$ isn't increased.

$\Delta_f(v,w)=0$: An operation PUSH(w,v) has to be executed before the next operation PUSH(w,v) can be performed. At that time we will have $h(w)>h(v)$. Hence, an operation PUSH(v,w) can only be performed after $h(v)$ is increased. □

Corollary 16. *The algorithm of Goldberg & Tarjan can be implemented to run in*

$$\mathcal{O}(n(n+m) + \#\textit{non-saturating } \textsc{Push}) \subseteq \mathcal{O}(n^2 m) \textit{ time}$$

(plus the time needed to choose an active vertex).

Proof. Recall that there are $\mathcal{O}(n^2)$ Relabel operations and $\mathcal{O}(nm)$ saturating Push operations. Let $v \in V$. Before each traversal of the list $N(v)$ there is an operation Relabel(v). Recall that an operation Relabel(v) is performed at most $2n-1$ times. Hence the total time needed to update $p(v)$ is bounded by

$$(2n-1) \qquad \underbrace{\sum_{v \in V} |N(v)|}_{\substack{\leq 2m \\ \text{(Handshaking Lemma)}}} \qquad \in \mathcal{O}(nm). \quad \square$$

In each step of the algorithm of Goldberg & Tarjan there is a freedom of choosing a suitable active vertex (and a suitable incident residual arc). By an intelligent choice the upper bound for the number of non-saturating Push and, hence, the run time of the algorithm of Goldberg & Tarjan can be improved.

6.2.3.3 Some Vertex Selection Rules

FIFO. Organize the active vertices in a queue. Apply all possible Push-operations on a vertex before considering the next one. Run time in $\mathcal{O}(n^3)$ – Shiloach and Vishkin (1982). ($\mathcal{O}(nm \log(n^2/m))$ – Goldberg and Tarjan (1988) – with dynamic trees and pushing flow over more than one edge at once.)

Excess-Scaling. Choose v with "high enough" flow excess and among them with minimum height. Run time in $\mathcal{O}(nm + n^2 \log \max_{(u,v)\in E} c(u,v))$ – Ahuja and Orlin (1989).

Highest-Level. Choose among all active vertices one with the maximum height. Run time in $\mathcal{O}(n^2 m^{1/2})$ – Cheriyan and Maheshwari (1989). See next section.

6.2.3.4 Highest-Level Implementation

In the following, we show how to implement the algorithm of Goldberg & Tarjan with the highest-level vertex selection rule. We store the active vertices in buckets $b_i, i = 0, \ldots, 2n-1$ with

$$v \in b_i \quad \Longleftrightarrow \quad v \text{ active and } h(v) = i.$$

Let

$$H = \max\{h(v);\ v \text{ active}\}.$$

The following updates on the data structure have to be performed after

Initialization: $H \leftarrow 0$ and $b_H \leftarrow \{v \in V \setminus \{s,t\};\ c(s,v) > 0\}$.

Relabel(v)**:** Move v to $b_{h(v)}$ and set $H \leftarrow h(v)$.

Push(v,w)**:** (1) If $w \notin \{s,t\}$ and $e_f(w)$ was 0 before the last PUSH then add w to b_{H-1}.

(2) If $e_f(v) = 0$ then remove v from b_H.

(3) While $b_H = \emptyset$ and $H > 0$ do $H \leftarrow H - 1$.

Note that if $w \neq s$ then H has to be decreased by at most one: If $w \neq t$ then $w \in b_{H-1}$ and if $w = t$ then $H = 1$. However, if $w = s$ then H might have to be decreased by more than one.

Lemma 17. *The algorithm of Goldberg & Tarjan with the highest-level selection rule can be implemented to run in* $\mathcal{O}(n(n+m) + \#$*non-saturating* PUSH$)$ *time.*

Proof. By Corollary 16, it remains to examine the additional cost for choosing in each step an active vertex with the maximum height. This is in the order of the number of the performed operations PUSH or RELABEL plus the time needed to update H. Note that H is only increased by RELABEL operations. For each $v \in V$ all operations RELABEL increase H by at most $2n - 1$. Hence, the total increase of H and thus, its total decrease is in $\mathcal{O}(n^2)$. □

Lemma 18. *The number of non-saturating* PUSH *in the algorithm of Goldberg & Tarjan with the highest-level selection rule is in* $\mathcal{O}(n^2 m^{1/2})$.

Proof. Consider the phases of the algorithm between two consecutive changes of the maximal height H.

$$R\,|\,PPPPR\,|\,PPP\,|\,PPR\,|\,PP\,|\,P \quad R = \text{ RELABEL}, P = \text{ PUSH}$$

Hence, a phase ends after

- each RELABEL (H increases).
- a PUSH deactivating the last active vertex of height H (H decreases).

There are at most $4n^2$ phases: H is increased by at most $(n-2)(2n-1) \leq 2n^2$ by RELABEL operations. Hence, H can also only be decreased at most $2n^2$ times.

A phase is *cheap* if it contains at most $m^{1/2}$ non-saturating PUSH and *expensive* otherwise. Clearly, the total number of non-saturating PUSH operations in cheap phases is in $\mathcal{O}(n^2 m^{1/2})$. It remains to show that the total number of non-saturating PUSH operations in expensive phases is also in $\mathcal{O}(n^2 m^{1/2})$. Let

$$D(v) = \{w \in V;\ h(w) \leq h(v)\}.$$

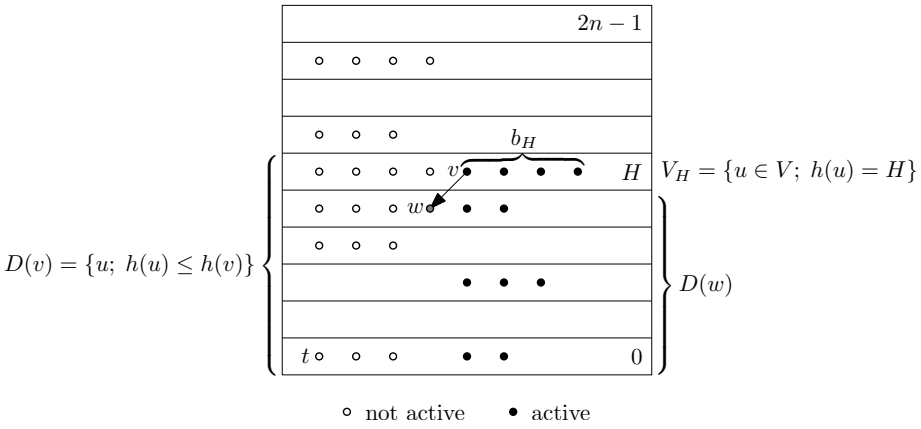


Figure 3: Illustration of the proof of Lemma 18.

We consider the potential function

$$\phi = \sum_{v \text{ active}} |D(v)|$$

after the following operations (see Fig. 3 for an illustration).

Initialization: $h(s) = n$ and $h(v) = 0, v \neq s$. Hence, $D(v) = V \setminus \{s\}$ for $v \neq s$ and

$$\phi = |\underbrace{\{v \in V \setminus \{s,t\};\ c(s,v) > 0\}}_{=\{v \in V;\ v \text{ active}\}}| \cdot |V \setminus \{s\}| < n^2.$$

Relabel(v)**:** Only $|D(v)|$ might increase. Hence, ϕ increases by at most n.

saturating Push(v,w)**:** Only w might become a new active vertex. Hence, ϕ increases by at most n.

non-saturating Push(v,w)**:** v is deactivated and w might be activated. Since $D(v)$ is the disjoint union of $D(w)$ and $V_H := \{u \in V;\ h(u) = H\}$ it follows that ϕ decreases by at least $|V_H|$. Further, each non-saturating PUSH deactivates a vertex in V_H. A vertex that was deactivated within one phase will not be activated before the end of the same phase. Hence, the number of non-saturating PUSH in a phase is at most $|b_H| \leq |V_H|$. Thus, in an expensive phase $|V_H| > m^{1/2}$.

Summarizing, starting from less than n^2, the potential function ϕ increases by at most $2n^3 + 2mn^2$. Hence, there are at most

$$\frac{n^2(1+2m+2n)}{m^{1/2}} \overset{m \geq n-1}{\in} \mathcal{O}(n^2 m^{1/2})$$

non-saturating PUSH in expensive phases. □

Theorem 19. *The algorithm of Goldberg & Tarjan with the highest-level selection rule can be implemented to run in* $\mathcal{O}(n^2 m^{1/2})$ *time.*

6.2.3.5 Literature

Among others, the algorithm of Goldberg & Tarjan and its analysis is also described in Cormen et al. (2001). The proof in these lecture notes is mainly based on Kleinberg and Tardos (2004). The proof of Theorem 18 is according to Cheriyan and Mehlhorn (1999). For further reading on network flows, please consider Ahuja et al. (1993).

References

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows*. Prentice Hall.

Ahuja, R. K. and Orlin, J. B. (1989). A fast and simple algorithm for the maximum flow problem. *Operations Research*, 37:748–759.

Cheriyan, J. and Maheshwari, S. N. (1989). Analysis of preflow push algorithms for maximum network flow. *SIAM Journal on Computing*, 18:1057–1086.

Cheriyan, J. and Mehlhorn, K. (1999). An analysis of the highest-level selection rule in the preflow-push max-flow algorithm. *Information Processing Letters*, 69:239–242.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, S. (2001). *Introduction to Algorithms*. MIT Press.

Goldberg, A. V. and Tarjan, R. E. (1988). A new approach to the maximum flow problem. *Journal of the Association for Computing Mashinery*, 35:921–940. (also STOC'86).

Kleinberg, J. and Tardos, É. (2004). *Algorithm Design*. Addison-Wesley.

Shiloach, Y. and Vishkin, U. (1982). An $\mathcal{O}(n^2 \log n)$ parallel max-flow algorithm. *Journal of Algorithms*, 3:128–146.