Flows

Guillaume Dalle (ENPC, CERMICS)

REOP - Class 3 (13/10/2021)

Contents

1 Homework solutions **1**
1.1 Ex 5.12 1
1.2 Ex 5.19 1

2 Flow vocabulary **2**
2.1 *s*-*t* flows 2
2.2 *s*-*t* cuts 2
2.3 *b*-flows 2
2.4 Modeling examples 3

3 Flow algorithms **3**
3.1 Optimality criterion for *s*-*t* flows 3
3.1.1 Upper bound 3
3.1.2 Residual graph & augmenting paths 3
3.2 Ford-Fulkerson 4
3.2.1 Pseudocode 4
3.2.2 Complexity 4
3.3 Edmonds-Karp 4
3.3.1 Pseudocode 4
3.3.2 Complexity 4
3.4 Minimum mean cycle-canceling (for minimum cost *b*-flows) 5

4 Linear programming for flows **5**
4.1 Formulation 5
4.2 Polyhedral interpretation 5

1 Homework solutions

1.1 Ex 5.12

See lecture notes with answers on Educnet.

1.2 Ex 5.19

As usual in dynamic programming, we generalize the problem. Let us try to compute the length $\ell(i, j)$ of the longest common subword of $w_1[1:i]$ and $w_2[1:j]$. There are several cases to consider:

1. The end letters of both words are identical:

$$\ell(n_1, n_2) = 1 + \ell(n_1 - 1, n_2 - 1)$$

2. The end letters of both words are different:

$$\ell(n_1, n_2) = \max\{\ell(n_1 - 1, n_2), \ell(n_1, n_2 - 1)\}$$

That way, we get the recursive Bellman equation. The initialization is done with

$$\ell(n_1, 0) = \ell(0, n_2) = 0$$

2 Flow vocabulary

2.1 s-t flows

We consider a digraph $D = (V, A)$ with additional features:

- nonnegative upper capacities $u(a) \geq 0$ on each arc
- two special nodes: a source s and a target t

An s-t flow is a vector $f \in \mathbb{R}_+^A$ satisfying Kirchhoff's current law

$$\forall v \in V \backslash \{s, t\}, \quad \sum_{a \in \delta^-(v)} f(a) = \sum_{a \in \delta^+(v)} f(a)$$

and capacity constraints

$$\forall a \in A, \quad f(a) \leq u(a)$$

The value of an s-t flow f is the total quantity flowing out of the source:

$$\mathrm{val}(f) = \sum_{a \in \delta^+(s)} f(a) - \sum_{a \in \delta^-(s)} f(a)$$

2.2 s-t cuts

An s-t cut (S, T) is a partition $V = S \cup T$ of the vertices into disjoint subsets ($S \cap T = \emptyset$) such that $s \in S$ and $t \in T$. It can be viewed as a set of arcs $B = \delta^+(S)$ intersecting any s-t path.

The capacity of a cut is the sum of the capacities of its arcs:

$$u(S, T) = \sum_{\substack{i \in S, j \in T \\ (i,j) \in A}} u(i, j) \quad \text{or} \quad u(B) = \sum_{a \in B} u(a)$$

We will study the following problems:

- Maximum flow problem: Find an s-t flow of maximum value $\mathrm{val}(f)$ subject to u.
- Minimum cut problem: Find an s-t cut of minimum capacity $u(B)$

Theorem 6.5 (Max flow / min cut): The maximum value of an s-t flow is equal to the minimum value of an s-t cut.

2.3 b-flows

For this setting we consider a digraph $D = (V, A)$ with slightly different features :

- lower and upper capacities $0 \leq \ell(a) \leq u(a)$ on each arc
- cost values $c(a) \geq 0$ on each arc
- algebraic inflows $b(v)$ at each vertex such that $\sum_{v \in V} b(v) = 0$

A b-flow is a vector $f \in \mathbb{R}_+^A$ satisfying Kirchhoff's current law

$$\forall v \in V, \quad b(v) + \sum_{a \in \delta^-(v)} f(a) = \sum_{a \in \delta^+(v)} f(a)$$

and capacity constraints

$$\forall a \in A, \quad \ell(a) \leq f(a) \leq u(a)$$

If $b(v) = 0$ everywhere, we call f a circulation.

The cost of a b-flow f is the sum of the costs induced by f on each arc:

$$c(f) = \sum_{a \in A} c(a) f(a)$$

2.4 Modeling examples

Exercise 6.9: Monge's transportation problem

Exercise 6.13: taxi fleet

Exercise 6.10: bus trip

Exercise 6.18: battle on a network

3 Flow algorithms

3.1 Optimality criterion for s-t flows

3.1.1 Upper bound

Proposition 6.3 (Cuts as upper bounds on flows): Let $0 \leq f \leq u$ be an s-t flow and (S, T) be an s-t cut. Then $\text{val}(f) \leq u(S, T)$.

Proof:

$$\begin{aligned}
\text{val}(f) &= \sum_{a \in \delta^+(s)} f(a) - \sum_{a \in \delta^-(s)} f(a) + \sum_{v \in S \setminus \{s\}} \left(\sum_{a \in \delta^+(v)} f(a) - \sum_{a \in \delta^-(v)} f(a) \right) \\
&= \sum_{a \in \delta^+(S)} f(a) - \sum_{a \in \delta^-(S)} f(a) \leq \sum_{a \in \delta^+(S)} u(a) - \sum_{a \in \delta^-(S)} 0 \\
&= u(S)
\end{aligned}$$

3.1.2 Residual graph & augmenting paths

For every arc $a = (i, j) \in A$, we define a reversed arc $\overleftarrow{a} = (j, i)$ and the residual capacities (assuming that $\overleftarrow{a} \notin A$):

$$u_f(a) = u(a) - f(a) \quad \text{and} \quad u_f(\overleftarrow{a}) = f(a)$$

In the general case where both directions of an edge can be present, we must use

$$u_f(a) = u(a) - f(a) + f(\overleftarrow{a})$$

The residual graph is the capacitated graph $D_f = (V, A_f, u_f)$ with

$$A_f = \{a \in A \cup \overleftarrow{A} : u_f(a) > 0\}$$

An f-augmenting path is an s-t path in the residual graph D_f.

To augment f by γ along an f-augmenting path P means performing, for every $a \in P$:

- $f(a) \leftarrow f(a) + \gamma$ if $a \in A$
- $f(a) \leftarrow f(a) - \gamma$ if $a \in \overleftarrow{A}$

Theorem 6.4 (Optimality criterion): An s-t flow f is maximal iff there is no f-augmenting path.

Proof: If there is an augmenting path, the flow can be augmented along this path. If no such path exists, then s and t are separated in the residual graph. Let S and T denote the connected components of s and t in D_f: its residual capacity is $u_f(S, T) = 0$, which means $u(S, T) = \text{val}(f)$.

3.2 Ford-Fulkerson

3.2.1 Pseudocode

Input: a digraph $D = (V, A)$ with capacities u, two vertices s and t

1. Set $f(a) = 0$ for all $a \in A$;
2. While there is an f-augmenting path:
   1. Select an f-augmenting path P
   2. Augment f along P by $\min_{a \in P} u_f(a)$

Output: a maximum s-t flow f

Questions:

- How do we find / select an augmenting path?
- Does the algorithm terminate, and if so when?

3.2.2 Complexity

Each iteration of the Ford-Fulkerson loop takes $O(|A|)$ time

If the capacities u are integral, so are the flow augmentations.

Theorem: If the capacities u are integral, the Ford-Fulkerson algorithm returns a maximum s-t flow in $O(|A| \times \text{val}_{\max})$ time, where $\text{val}_{\max}$ is the maximum value of an s-t flow.

3.3 Edmonds-Karp

3.3.1 Pseudocode

Input: a digraph $D = (V, A)$ with capacities u, two vertices s and t

1. Set $f(a) = 0$ for all $a \in A$
2. While there is an f-augmenting path:
   1. Select an f-augmenting path P *with minimum number of edges*
   2. Augment f along P by $\min_{a \in P} u_f(a)$

Output: a maximum s-t flow f

Questions:

- How do we select such an augmenting path?
- Why does it improve the complexity?

3.3.2 Complexity

The Edmonds-Karp loop is crossed at most $|A| \times |V|$ times.

Proof: We can show that

- The (unweighted) distance $\text{dist}_{D_f}(s, t)$ in the residual graph is nonincreasing
- It can only remain constant for at most $|A|$ iterations

Theorem: The Edmonds-Karp algorithm returns a maximum s-t flow in $O(|A|^2 \times |V|)$ time.

3.4 Minimum mean cycle-canceling (for minimum cost b-flows)

Pseudocode

Input: a digraph $D = (V, A)$ with capacities $l \leq u$, costs c and inflows b}

1. Find an initial b-flow f
2. While there is an f-augmenting cycle with negative cost
    1. Select an f-augmenting cycle C *with minimum mean cost*
    2. Augment f along C by $\min_{a \in C} u_f(a)$

Output: a minimum cost b-flow f

Questions:

- How do we find an initial b-flow?
- How do we select an f-augmenting cycle with minimum mean cost?

Complexity

An initial b-flow can be found in $O(|A| \times |V|)$ time (see Ex 6.4).

An f-augmenting cycle of minimum mean cost can be found in $O(|A| \times |V|)$ time (see Ex 6.3).

Theorem 6.10: The cycle-canceling algorithm returns a minimum cost b-flow in $O(|A|^3|V|^2 \log |V|)$ time.

4 Linear programming for flows

4.1 Formulation

We can formulate the maximum flow problem as follows:

$$\begin{aligned} \max \quad & \sum_{a \in \delta^+(s)} x_a - \sum_{a \in \delta^-(s)} x_a \\ \text{s.t.} \quad & \sum_{a \in \delta^-(v)} x_a = \sum_{a \in \delta^+(v)} x_a \qquad \forall v \in V \backslash \{s, t\} \\ & 0 \leq x_a \leq u(a) \qquad \forall a \in A \end{aligned}$$

We will prove the following results later in the course:

- **Proposition 6.11**: The constraint matrix of the max flow LP is totally unimodular.
- **Proposition 6.12**: The minimum s-t cut is the Lagrangian dual of the maximum s-t flow.

  Exercise 6.16: prove this

4.2 Polyhedral interpretation

A general result in polyhedral geometry (the Minkowski-Weyl theorem) states that every polyhedron P can be written as

$$P = \left\{ \sum_i \lambda_i x_i + \sum_j \mu_j y_j : \lambda \geq 0, \mu \geq 0, \sum_i \lambda_i = 1 \right\}$$

The flow version of this result is Proposition 6.13: every s-t flow can be decomposed as a positive sum of flows along elementary s-t paths or elementary cycles.