Max-flow and min-cut problems

Edmonds Karp alg

Generic reduction to MaxFlow

Dinic and Edmonds-Karp algorithm

J.Edmonds, R. Karp: *Theoretical improvements in algorithmic efficiency for network flow problems*. Journal ACM 1972.

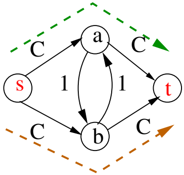
Yefim Dinic: *Algorithm for solution of a problem of maximum flow in a network with power estimation*. Doklady Ak.N. 1970

Choosing a good augmenting path can lead to a faster algorithm.
Use BFS to find an augmenting paths in G_f.

Edmonds-Karp algorithm

FF algorithm but using BFS: choose the augmenting path in G_f with the smallest length (number of edges).

Edmonds-Karp(G, c, s, t)
For all $e = (u, v) \in E$ let $f(u, v) = 0$
$G_f = G$
while there is an $s \rightsquigarrow t$ path in G_f
do
  $P = \text{BFS}(G_f, s, t)$
  $f = \text{Augment}(f, P)$
  Compute G_f
return f



The BFS in EK will choose: or

BFS paths on G_f

For $\mathcal{N} = (V, E, c, s, t)$ and a flow f in $\mathcal{N}$, assuming that G_f has an augmenting path, let f' be the next flow after executing one step of the EK algorithm.

BFS paths on G_f

For $\mathcal{N} = (V, E, c, s, t)$ and a flow f in $\mathcal{N}$, assuming that G_f has an augmenting path, let f' be the next flow after executing one step of the EK algorithm.

- The path from s to t in a BFS traversal starting at s, is a path $s \rightsquigarrow t$ with minimum number of edges, i.e., a shortest length path.

BFS paths on G_f

For $\mathcal{N} = (V, E, c, s, t)$ and a flow f in $\mathcal{N}$, assuming that G_f has an augmenting path, let f' be the next flow after executing one step of the EK algorithm.

- The path from s to t in a BFS traversal starting at s, is a path $s \rightsquigarrow t$ with minimum number of edges, i.e., a shortest length path.
- For $\in V$, let $\delta_f(s, v)$ denote length of a shortest length path from s to v in G_f.

Some properties of G_f and $G_{f'}$

Some properties of G_f and $G_{f'}$

How can we have $(u, v) \in E_{f'}$ but $(u, v) \notin E_f$?

- (u, v) is a forward edge saturated in f and not in f''.
- (u, v) is a backward edge in G_f and $f(v, u) = 0$

Some properties of G_f and $G_{f'}$

How can we have $(u, v) \in E_{f'}$ but $(u, v) \notin E_f$?

- (u, v) is a forward edge saturated in f and not in f''.
- (u, v) is a backward edge in G_f and $f(v, u) = 0$

In any of the two cases, the augmentation must have modified the flow from v to u, so (u, v) must form part of the augmenting path.

EK and the shortest length distances

Lemma

If the EK-algorithm runs on $\mathcal{N} = (V, E, c, s, t)$, for all vertices $v \neq s$, $\delta_f(s, v)$ increases monotonically with each flow augmentation.

Proof. By contradiction.

Let f be the first flow such that, for some $u \neq s$,

$$\delta_{f'}(s, u) < \delta_f(s, u).$$

EK and the shortest length distances

Proof (cont)

Let v be the vertex with the minimum $\delta_{f'}(s,v)$ whose distance was decreased.

- Let $P : s \rightsquigarrow u \rightarrow v$ be a shortest length path from s to v in $G_{f'}$
- Then, $\delta_{f'}(s,v) = \delta_{f'}(s,u) + 1$ and $\delta_{f'}(s,u) \geq \delta_f(s,u)$.
- If $(u,v) \in E_f$, $\delta_f(s,v) \leq \delta_f(s,u) + 1 \leq \delta_{f'}(s,u) + 1 = \delta_{f'}(s,v)$
- So, $(u,v) \notin E_f$

EK and the shortest length distances

Proof (cont)

How can we have ?

- $(u, v) \in E_{f'}$ but $(u, v) \notin E_f$
- If so, (v, u) appears in the augmenting path.

EK and the shortest length distances

Proof (cont)

How can we have ?

- $(u, v) \in E_{f'}$ but $(u, v) \notin E_f$
- If so, (v, u) appears in the augmenting path.
- Then, the shortest length path from s to u in G_f has (v, u) as it last edge.
  $\delta_f(s, v) \leq \delta_f(s, u) - 1 \leq \delta_{f'}(s, u) - 1 = \delta_{f'}(s, v) - 1 - 1$

EK and the shortest length distances

Proof (cont)

How can we have ?

- $(u, v) \in E_{f'}$ but $(u, v) \notin E_f$
- If so, (v, u) appears in the augmenting path.
- Then, the shortest length path from s to u in G_f has (v, u) as it last edge.
  $\delta_f(s, v) \leq \delta_f(s, u) - 1 \leq \delta_{f'}(s, u) - 1 = \delta_{f'}(s, v) - 1 - 1$
- which contradicts $\delta_{f'}(s, v) < \delta_f(u, v)$. $\square$

Some properties of G_f and $G_{f'}$

Some properties of G_f and $G_{f'}$

Let P be an augmenting path in G_f.

$(u, v) \in P$ is critical if $b(P) = c_f(u, v)$.

Some properties of G_f and $G_{f'}$

Let P be an augmenting path in G_f.

$(u, v) \in P$ is critical if $b(P) = c_f(u, v)$.

Critical edges do not appear in $G_{f'}$.

- (u, v) forward, $f'(u, v) = c(u, v)$
- (u, v) backward, $f'(v, u) = 0$

EK and critical edges

Lemma

In the EK algorithm, each one of the edges can become critical at most $|V|/2$ times.

Proof:

- Let $(u, v) \in E$, when (u, v) is critical for the first time, $\delta_f(s, v) = \delta_f(s, u) + 1$

EK and critical edges

Lemma

In the EK algorithm, each one of the edges can become critical at most $|V|/2$ times.

Proof:

- Let $(u, v) \in E$, when (u, v) is critical for the first time, $\delta_f(s, v) = \delta_f(s, u) + 1$
- After this step (u, v) disappears from the residual graph until after the flow in (u, v) changes.

EK and critical edges

Lemma

In the EK algorithm, each one of the edges can become critical at most $|V|/2$ times.

Proof:

- Let $(u, v) \in E$, when (u, v) is critical for the first time, $\delta_f(s, v) = \delta_f(s, u) + 1$
- After this step (u, v) disappears from the residual graph until after the flow in (u, v) changes.
- At this point, (v, u) forms part of the augmenting path in $G_{f'}$, and $\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1$,

$$\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1 \geq \delta_f(s, v) + 1 \geq \delta_f(s, u) + 2$$

EK and critical edges

Lemma

In the EK algorithm, each one of the edges can become critical at most $|V|/2$ times.

Proof:

- Let $(u, v) \in E$, when (u, v) is critical for the first time, $\delta_f(s, v) = \delta_f(s, u) + 1$
- After this step (u, v) disappears from the residual graph until after the flow in (u, v) changes.
- At this point, (v, u) forms part of the augmenting path in $G_{f'}$, and $\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1$,

$$\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1 \geq \delta_f(s, v) + 1 \geq \delta_f(s, u) + 2$$

- So, the distance has increased by at least 2.

Complexity of Edmonds-Karp algorithm

Theorem

The EK algorithms runs in $O(mn(n+m))$ steps. Therefore it is a polynomial time algorithm.

Complexity of Edmonds-Karp algorithm

Theorem

The EK algorithms runs in $O(mn(n+m))$ steps. Therefore it is a polynomial time algorithm.

Proof:

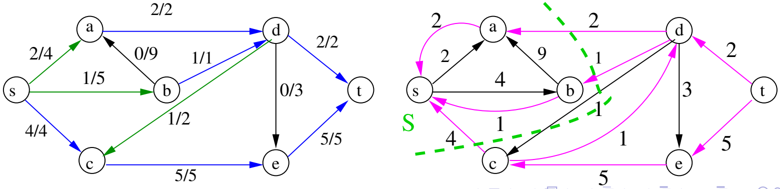
- Need time $O(m+n)$ to find the augmenting path using BFS.
- By the previous Lemma, there are $O(mn)$ augmentations. $\square$

Finding a min-cut

Given (G, s, t, c) to find a min-cut:

1. Compute the max-flow f^* in G.
2. Obtain G_{f^*}.
3. Find the set $S = \{v \in V | s \rightsquigarrow v\}$ in G_{f^*}.
4. Output the cut
   $(S, V - \{S\}) = \{(v, u) | v \in S and u \in V - \{S\}\}$ in G.

The running time is the same than the algorithm to find the max-flow.



The max-flow problems: History

- Ford-Fulkerson (1956) $O(mC)$, where C is the max flow.
- Dinic (1970) (blocking flow) $O(n^2 m)$
- Edmond-Karp (1972) (shortest augmenting path) $O(nm^2)$
- Karzanov (1974), $O(n^2 m)$ Goldberg-Tarjant (1986) (push re-label preflow + dynamic trees) $O(nm \lg(n^2/m))$ (uses parallel implementation)
- King-Rao-Tarjan (1998) $O(nm \log_{m/n \lg n} n)$.
- J. Orlin (2013) $O(nm)$ (clever follow up to KRT-98)

So: Maximum flows can be computed in $O(nm)$ time!

Applications: Generalized assignment problems

- Consider a generalized assignment problem $\mathcal{GP}$ where, we have as input d finite sets $X_1, \ldots, X_d$, each representing a different set of resources.

Applications: Generalized assignment problems

- Consider a generalized assignment problem $\mathcal{GP}$ where, we have as input d finite sets $X_1, \ldots, X_d$, each representing a different set of resources.
- Our goal is to chose the "largest" number of d-tuples, each d-tuple containing exactly one element from each X_i, subject to the constrains:

Applications: Generalized assignment problems

- Consider a generalized assignment problem $\mathcal{GP}$ where, we have as input d finite sets $X_1, \ldots, X_d$, each representing a different set of resources.
- Our goal is to chose the "largest" number of d-tuples, each d-tuple containing exactly one element from each X_i, subject to the constrains:
  - For each $i \in [d]$, each $x \in X_i$ can appears in at most $c(x)$ selected tuples.
  - For each $i \in [d]$, any two $x \in X_i$ and $y \in X_{i+1}$ can appear in at most $c(x, y)$ selected tuples.
  - The values for $c(x)$ and $c(x, y)$ are either in $\mathbb{Z}^+$ or ∞.

Applications: Generalized assignment problems

- Consider a generalized assignment problem $\mathcal{GP}$ where, we have as input d finite sets $X_1, \ldots, X_d$, each representing a different set of resources.
- Our goal is to chose the "largest" number of d-tuples, each d-tuple containing exactly one element from each X_i, subject to the constrains:
  - For each $i \in [d]$, each $x \in X_i$ can appears in at most $c(x)$ selected tuples.
  - For each $i \in [d]$, any two $x \in X_i$ and $y \in X_{i+1}$ can appear in at most $c(x, y)$ selected tuples.
  - The values for $c(x)$ and $c(x, y)$ are either in $\mathbb{Z}^+$ or ∞.
- Notice that only pairs of objects between adjacent X_i and X_{i+1} are constrained.

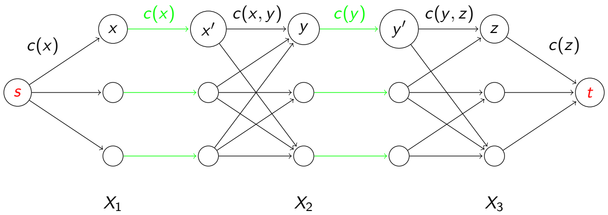
Applications: Generic reduction to Max-Flow

Make the reduction from $\mathcal{GP}$ to the following network $\mathcal{N}$:

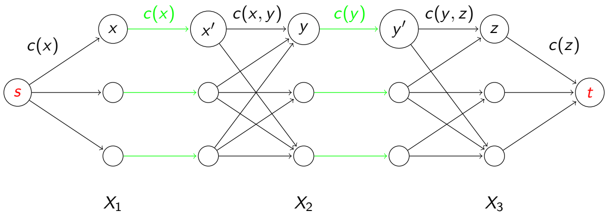
- V contains a vertex x, for each element x in each X_i, and a copy x', for each element $x \in X_i$ for $1 \leq i < d$.
- We add vertex s and vertex t.
- Add an edge $s \to x$ for each $x \in X_1$ and add an edge $y \to t$ for every $y \in X_d$. Give capacities $c(s, x) = c(x)$ and $c(y, t) = c(y)$.
- Add an edge $x' \to y$ for every pair $x \in X_i$ and $y \in X_{i+1}$. Give a capacity $c(x, y)$. Omit the edges with capacity 0.
- For every $x \in X_i$ for $1 \leq i < d$, add an edge $x \to x'$ with $c(x, x') = c(x)$.

Every path $s \rightsquigarrow t$ in $\mathcal{N}$ identifies a feasible d-tuple, conversely every d-tuple determines a a path $s \rightsquigarrow t$.

Flow Network: The reduction

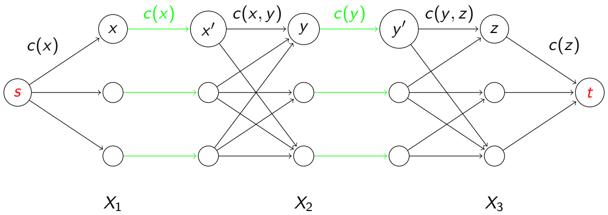


Flow Network: The reduction



- To solve $\mathcal{GP}$, we construct $\mathcal{N}$, and then we find an integer maximum flow f^*.
- In the subgraph formed by edges with $f^*(e) > 0$, we find a (s, t) path P (a d-tuple), decrease in 1 the flow in each edge of P, remove edges with 0 flow.

Flow Network: The reduction



- To solve $\mathcal{GP}$, we construct $\mathcal{N}$, and then we find an integer maximum flow f^*.
- In the subgraph formed by edges with $f^*(e) > 0$, we find a (s, t) path P (a d-tuple), decrease in 1 the flow in each edge of P, remove edges with 0 flow.
- We repeat the procedure for $|f^*|$times. In this way we obtain a set of d-tuples with maximum size verifying all the restrictions.

Final's Scheduling

We have as input:

- n courses, each one with a final. Each exam must be given in one room. Each course c_i has $E[i]$ students.
- r rooms. Each r_j has a capacity $S[j]$,
- τ time slots. For each room and time slot, we only can schedule one final.
- p professors to watch exams. Each exam needs one professor in each class and time. Each professor has its own restrictions of availability and no professor can oversee more than 6 finals. For each p_ℓ and τ_k define a Boolean variable $A[k, \ell] = T$ if p_ℓ is available at τ_k.

Design an efficient algorithm that correctly schedules a room, a time slot and a professor to every final, or report that not such schedule is possible.

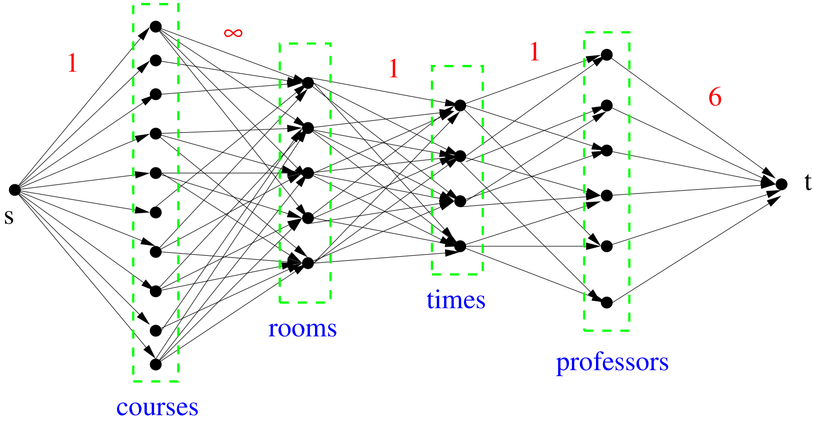
Construction of the network

Construct the network $\mathcal{N}$ with vertices $\{s, t, \{c_i\}, \{r_j\}, \{\tau_k\}, \{p_\ell\}\}$. Edges and capacities:

- (s, c_i) with capacity 1 (each course has one final)
- (c_i, r_j), if $E[i] \leq S[j]$, with capacity ∞
- $\forall j, k$, (r_j, τ_k), with capacity 1 (one final per room and time slot).
- (τ_k, p_ℓ), if $A[k, \ell] = T$, capacity 1
  (p can watch one final, if p is available at τ_k).
- (p_ℓ, t), capacity 6 (each p can watch ≤ 6 finals)

Notice that neither rooms nor time slots have individual restrictions.

Final's Scheduling: Flow Network



FINAL'S SCHEDULING

- Notice the input size to the problem is $N = n + r + \tau + p + 2$. and size of the network is $O(N)$ vertices and $O(N^2)$ edges, why?
- Every path $s \rightsquigarrow t$ is an assignment of room-time-professor to a final, and any assignment room-time-professor to a final can be represented by a path $s \rightsquigarrow t$.
- Every integral flow identifies a collection of $|f|$ (s, t)-paths leading to a valid assignment for $|f|$ finals and viceversa.

Final's Scheduling

- To maximize the number of finals to be given, we compute the max-flow f^* from s to t.
- If $|f^*| = n$, then we can schedule all finals, otherwise we can not.
- To recover the assignment we have to consider the edges with positive flow and extract assignment from the n (s,t)-paths
- Complexity:
    - To construct $\mathcal{N}$, we need $O(N^2)$.
    - As $|f^*| \leq n$ integral, we can use Ford-Fulkerson to compute f^*, with cost $O(nN^2)$.
    - The second part requires $O(N^2)$ time.
    - So, the cost of the algorithm is $O(nN^2) = O(n(n+r+\tau+p)^2)$.