12. Maksimal flyt

Forelesning 12

Et stort skritt i retning av generell lineær optimering (såkalt lineær programmering). Her ser vi på to tilsynelatende forskjellige problemer, som viser seg å være *duale* av hverandre, noe som hjelper oss med å finne en løsning.

Pensum

- [ ] Kap. 24. Maximum flow

Læringsmål

[L_1] Kunne definere *flytnett*, *flyt* og *maks-flyt-problemet*

[L_2] Kunne håndtere *antiparallelle kanter* og *flere kilder og sluk*

[L_3] Kunne definere *restnett*

[L_4] Forstå *oppheving* av flyt

[L_5] Forstå *forøkende stier*

[L_6] Forstå *snitt*, *snitt-kapasitet* og *minimalt snitt*

[L_7] Forstå *maks-flyt/min-snitt*

[L_8] Forstå FORD-FULKERSON (FF)

[L_9] EDMONDS-KARP = FF m/BFS

[L_{10}] Finne min-snitt vha. FF

[L_{11}] Kunne finne en *maksimum bipartitt matching* vha. flyt

[L_{12}] Forstå *heltallsteoremet*

[L_{13}] Kunne redusere til maks-flyt

📚Problemet

Maksimal flyt er et problem på en rettet graf $G = (V, E)$. Et flytnett er grafen hvor flyten kan flyte igjennom, og den har følgende kriterier:

- Hver kant har en kapasitet $c(u, v) \geq 0$
- Det finnes ihvertfall en kilde, og en sluk $s, t \in V$
- For en hver $v \in V \implies s \to v \to t$
- Grafen har ingen self-loops
- $(u, v) \in E \implies (v, u) \notin E$

- $(u,v) \notin E \implies c(u,v) = 0$

Den resulterende flyten er en funksjon $f : V \times V \to \mathbb{R}$. Flyten på en kant kan aldri være større en kapasiteten på kanten, altså:

- $0 \leq f(u,v) \leq c(u,v)$
- For en kant $u \neq s,t$, gjelder $\sum_v f(u,v) = \sum_v (v,u)$

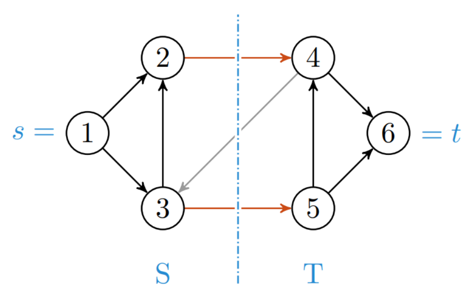
Flytverdien $|f| = \sum_v f(s,v) - \sum_v f(v,s)$ er flyten som kan nås fra source til en gitt node.

Hvis du har flere kilder og sluker, legger man til en super-kilde og en super-sluk

Input: Et flytnett G.

Output: En flyt f for G med maks. $|f|$.

Det minimale snittet er gitt etter å ha kjørt maksimal flyt på en graf. Den skaper en partisjon i flytnettet etter at sluket til grafen ikke kan nås lengre. Snittet i flytnettet er en partisjon (S,T) av V. $s \in S$ og $t \in T$.



Netto-flyten beskriver flyten som mellom partisjonen.

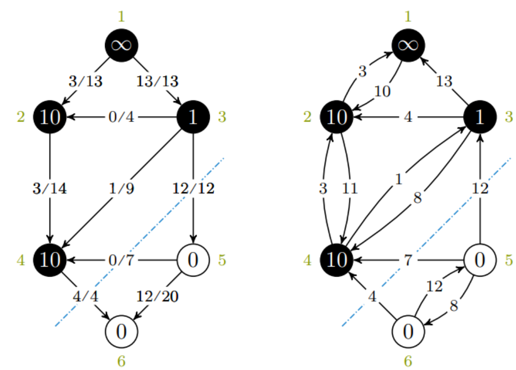
$$f(S,T) = \sum_{u \in S} \sum_{v \in S} f(u,v) - \sum_{u \in T} \sum_{v \in T} f(u,v)$$

Kapasiteten blir $c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$.

Til slutt beskriver Lemma 24.4: $f(S,T) = |f|$. Bevis for dette krever en del utregning, men er ganske rett frem. Videre Corollary 24.5: $|f| \leq c(S,T)$.

Input: Et flytnett $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ med kilde s og sluk t.

Output: Et snitt (S, T) med minst mulig kapasitet, dvs., der $c(\mathrm{S}, \mathrm{T})$ er minimal.



Flyt   Rest

Maksimal flyt fører dermed til det minste snittet. Videre forteller helltallsteoremet (24.10) at for heltallskapsiteter $c(u,v) \in \mathbb{N}$ gir $FordFulkerson$ og andre flytalgoritmer basert på $FordFulkerson$ en heltallsflyt $|f| \in \mathbb{N}$.

📚Reduksjon til flyt

Flyt kan nesten regnes som en designmetode siden vi kan redusere mange problemer til flytproblemet. Pensum ser på et par måter å bryte ned større problemer til flytproblemer. Blant annet:

- Biparitt matching, gjør om hver donor og mottaker til sink/source hvor kantene har en kapasitet på 1, deretter lag en supersource/supersink som vanlig. Greit å notere seg at $EdmondsKarp$ er ikke den mest effektive måte å gjøre dette på. I rekonstruksjonen representerer kanter med flyt matchingen.

Instans (i) En bipartitt graf $\mathrm{G} = (\mathrm{L} \cup \mathrm{R}, \mathrm{E})$ og et heltall k	Spørsmål (i) Finnes en matching for G med kardinalitet k?
Reduksjon Legg til kilde og sluk, med kanter til L og fra R	Rekonstruksjon ↑
Instans (ii) Flytnett G' med kapasiteter $c(u,v) = 1$	Spørsmål (ii) Finnes en flyt for G' med verdi k?

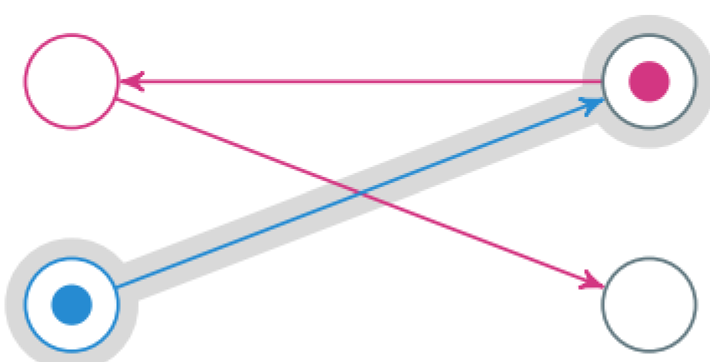
- Legekontor hvor man ønsker minst én lege på vakt hver dag i feriene, men hver lege skal jobbe maks c feriedager totalt, og maks en gang per ferie. Denne løses ved å gjøre legene om til sources med kapasitet c, og lag en source med kapasitet 1 for hver feriedag.
- Bildesegmentering for å separere forgrunn og bakgrunn kan løses ved å gjøre hver piksel til en node, og kapasiteten fra kilde tilsvarende "forgrunnsaktighet" og kapasitet til sluk "bakgrunnsaktighet". Kapaisteten mellom noder tilsvarer hvor like de er. Her vil det minimale snittet skille forgrunn fra bakgrunn.
- Veisperring kan løses ved å gjøre åstedet til source, og hvert sted man kan sette opp sperringer til sinks med kapasitet basert på hvor fort forbryterne kan bevege seg.

💻Biparitt matching

Designmetoden som brukes for biparitt matching baserer seg på å gjentatte ganger konstruere et nytt, enklere problem, som gir en forbedring av den opprinnelige instansen.

Vi øker matchingen ved å finne stier fra venstre til høyre (i en liggende graf). Kanter som brukes i matchingen blir snudd baklengs slik at det er mulig å endre på de hvis det viser seg at det er mer optimalt senere.

Hvis vi for eksempel finner ut at en donor kan kun matches med første resipient (men vi har allerede gitt første resipent en donor), kan vi bruke dette til å oppheve matchingen.



📚Ideer

Det samme gjelder for flyt. Vi har kanskje fått frem én enhet, men hvis vi har en forøkende sti (sti som gir bedre resultat en nåværende), har vi lyst til å sende tilbake enheten. Den originale enheten (eller flyten) sendes tilbake hvor den kom fra, og så finner vi en ny vei frem. Da får vi igjen en forøkende sti, og flyter fremover, og opphever den bakover til flyten når der den kom fra.

Flytoppheving

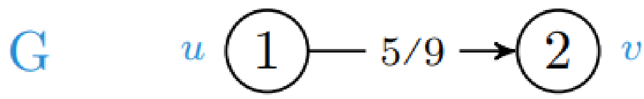
- **Vi kan «sende» flyt baklengs langs kanter der det allerede går flyt**
- **Vi opphever da flyten, så den kan omdirigeres til et annet sted**

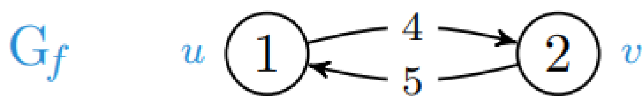
Et restnett (residual network) er et nettverk likt flytnettet, bortsett fra at kantene mellom nodene er fremoverkanter ved ledig kapasitet, og bakoverkanter ved flyt.

En forøkende sti (augmenting path) er en sti fra kilde til sluk i restnettet. Langs fremoverkanter kan flyten økes, og langs bakoverkanter kan flyten omdirigeres. Det er altså en sti hvor den totale flyten kan økes.

Det er dermed nyttig å finne ut hva potensialet (eller restkapasiteten) mellom to noder er. Hvor mye kan vi øke flyten fra u til v?

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in \mathrm{E}\,, \\ f(v,u) & \text{if } (v,u) \in \mathrm{E}\,, \\ 0 & \text{otherwise}\,. \end{cases}$$





$$(u,v) \in \mathrm{E}_f \iff c_f(u,v) > 0$$

💻Ford-Fulkerson

$FordFulkerson$ er en generell metode for å finne den maksimale flyten gjennom et nettverk. En versjon av $FordFulkerson$ er $EdmondKarp$ som bruker en BFS. Ford-Fulkersom fungerer ved å:

1. Finn forøkende stier så lenge det går
2. Deretter er flyten så stor den kan bli.

Dette gjøres som regel ved å finne den forøkende stien, deretter finner man flaskehalsen i stien, og oppdaterer flyt langs stien med denne verdien.

FORD-FULKERSON-METHOD(G, s, t)

initialize flow f to 0
while there is an augm. path p in G_f
  augment flow f along p
return f

En mer konkret beskrivelse av algoritmen finnes i pensum:

FORD-FULKERSON(G, s, t)

for each edge $(u, v) \in G.E$
  $(u, v).f = 0$
while there is a path p from s to t in G_f
  $c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is in } p\}$
  for each edge (u, v) in p
    if $(u, v) \in E$
      $(u, v).f = (u, v).f + c_f(p)$
    else $(v, u).f = (v, u).f - c_f(p)$

Alternativt kan man flette inn BFS ved å finne flaskehalser underveis. Hold deretter styr på hvor mye flyt vi får frem til hver node, og traverser kun noder vi ikke har nådd frem til enda.

Denne implementasjonen står ikke i boka, og det er ikke krav å kunne denne i detalj.

```
EDMONDS-KARP(G, s, t)
for each edge (u, v) ∈ G.E
    (u, v).f = 0
while BFS-LABELING(G, s, t)
    c_f(p) = t.f
    u, v = t.π, t
    while u ≠ NIL
        if (u, v) ∈ G.E
            (u, v).f = (u, v).f + c_f(p)
        else (v, u).f = (v, u).f − c_f(p)
        u, v = u.π, u
```

G	flytnett
s	kilde
t	sluk
u	node
v	node
$e.f$	flyt
$c_f(p)$	flaskehals
$v.\pi$	forgjenger

```
BFS-LABELING(G, s, t)
for each vertex u ∈ G.V
    u.f = 0
    u.π = NIL
s.f = ∞
Q = ∅
ENQUEUE(Q, s)
```

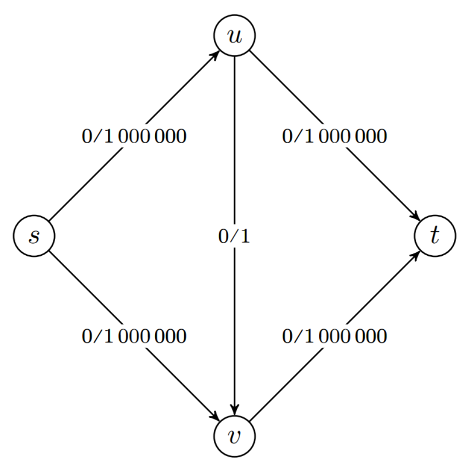
G	flytnett
s	kilde
t	sluk
u	node
$v.f$	potensial
$v.\pi$	forgjenger
Q	kø

```
while Q ≠ ∅ and t.f == 0
    u = DEQUEUE(Q)
    for all edges (u, v), (v, u) ∈ G.E
        if (u, v) ∈ G.E
            c_f(u, v) = c(u, v) − (u, v).f
        else c_f(u, v) = (v, u).f
        if c_f(u, v) > 0 and v.f == 0
            v.f = min(u.f, c_f(u, v))
            v.π = u
            ENQUEUE(Q, v)
return t.f ≠ 0
```

Q	kø
u	node
v	nabo
c	kapasitet
c_f	restkapasitet
$e.f$	flyt
$v.f$	potensial
$v.\pi$	forgjenger

Når vi hr kjørt Ford-Fulkerson kan vi finne det minimale snittet direkte fra resultatet. Forelesningsnotatet inneholder mer om bevis på korrekthet.

Kjøretiden på Ford-Fulkerson kan gå ille hvis vi ikke bruker BFS. En graf som ser slik ut, vil kunne sende flyten fra og tilbake på noden med 1 kapasitet flere millioner ganger.



Med irrasjonale kapasiteter kan vi ikke garantere at Ford-Fulkerson terminerer,

Hvis vi bruker BFS, vil får vi en kjøretid på $O(VE^2)$ for å finne forøkende stier.

Pensum har en lengre forklaring på hvorfor vi får $O(VE)$ i indre løkke.

https://folk.idi.ntnu.no/mlh/algkon/flow.pdf