Network Flow

- **Network Design**
  - G=(N,A) diretto con s,t
  - u_ij capacità
  - Assunzioni su G
    - Non ha (s,t)–path con solo u_ij=infinito
    - Non ha Archi Paralleli
    - Inserbili (i,j) t.c. u_ij=0
- **Path Packing**
  - Dato G con u cap. vector
  - trova famiglia P di k simple DIpath t.c.
    - ogni (i,j) di G è in al più u_ij DIpath
    - k max
  - ILP (flow formulation)
    - $$\max \ f_x(s) \quad \text{s.t.} \quad \sum_{j|(i,j)\in A} x_{ij} - \sum_{j|(j,i)\in A} x_{ji} = 0; \quad 0 \le x_{ij} \le u_{ij}, \ \forall (i,j)\in A; \quad x_{ij} \text{ INTEGER}, \ \forall (i,j)\in A$$
    - Balance Constr.
    - Capacity Constr.
  - (s,t)–flow — Vett. X che soddisfa Balance Constr.
  - Feasible (s,t)–flow — flow che soddisfa Capacity Constr.
  - Flow Value — $f_x(v) = \sum_{j|(v,j)\in A} x_{vj} - \sum_{j|(j,v)\in A} x_{jv}$
  - Theorem — Path Packing <=> FF
    - (=>) Trivial
    - (<=) Decomposition Theorem
      - proof
        - X FF trasformato in X' Acyclic FF
        - Partendo da t, aggiungiamo al path un arco con X_it>0 proseguendo fino a s. Abbiamo il path, sottraiamo 1 a tutti gli X_ij sul path trovato. Ripeti finché possibile.
- **Cut**
  - Insieme di archi t.c.
    - $\delta(R) = \{u,v \mid (v,w)\in A,\ v\in R,\ w\in \bar{R}\} \mid R \subseteq N$
  - Capacity del cut — Somma delle Capacity degli archi del taglio
  - Theorem 1
    - Per ogni (s,t)–cut e ogni feasible (s,t)–flow: $x(\delta(R)) - x(\delta(\bar{R})) = f_x(s)$
      - Proof — Dividiamo gli archi in 6 sottoinsiemi...
  - Weak duality
    - Opt(min Capacity cut)=opt(max flow): $f_x(s) \le u(\delta(R))$
      - Proof — $u(\delta(R)) \overset{\text{DA DEF.}}{\ge} x(\delta(R)) \ge x(\delta(R)) - x(\delta(\bar{R})) \overset{\text{WEAK DUALITY}}{=} f_x(s)$
    - Usata come Opt. Certificate
  - Max–Flow Min–Cut Theorem
    - Se G ammette max–Flow => $\max\{f_x(s) \mid x \text{ feas. } (s,t)\text{-flow}\} = \min\{u(\delta(R)) \mid \delta(R)\ (s,t)\text{-cut}\}$
      - Proof
        - Augmentin Path
          - Forward (s–>t) and reverse (s<–t) arcs
          - Un (s,t)–path P t.c.: $\forall$ FORWARD $(i,j) \Rightarrow x_{ij} < u_{ij}$; $\forall$ REVERSE $(i,j) \Rightarrow x_{ij} > 0$
            - Augmentin path Algo: Se incontriamo un forward incrementiamo di Epsilon1 sennò di Epsilon2
              - $\varepsilon_1 = \min_{(i,j)\in P \mid (i,j) \text{ FORWARD}} \{u_{ij} - x_{ij}\}$; $\varepsilon_2 = \min_{(i,j)\in P \mid (i,j) \text{ REVERSE}} \{x_{ij}\}$
              - Ci permette di muoverci tra feasible. Sols.
        - $f_x(s) = u(\delta(R))$ — ~~Mostriamo che in G un flow è un taglio t.c.~~
          - Dato un max flow creiamoci un taglio....
    - Conseguenze del th
      - Theorem2 — Feas. FloW X ottimo <=> non c'è un Augmentin path
        - Proof — Se non c'è l'augmenting path, da MFMC possiamo costruirci un taglio t.c. $f_x(s) = u(\delta(R))$
      - Corollario 1 — Dato un (s,t)–feasible_Flow e un (s,t)–cut: il primo e massimo è il secondo minimo <=> $x_{ij} = u_{ij}\ \forall (i,j)\in \delta(R)$ e $x_{ij} = 0\ \forall (i,j)\in \delta(\bar{R})$
    - MFMC algo
      - X = 0; DO trova (s,t)-AUGMENTIN_PATH P; incrementa il flow X su P di $\min\{\varepsilon_1, \varepsilon_2\}$; WHILE $P \ne 0$
      - Termina?
      - La soluzione è ottima — Si–>Th2
      - Complessità?

XMind