

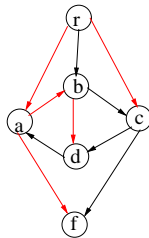
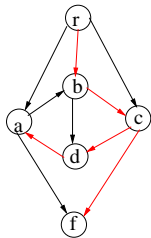
Min-Cost Arborescence

Edmonds Algorithm

R. Inkulu

<http://www.iitg.ac.in/rinkulu/>

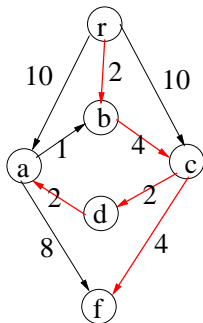
Defintion: arborescence (a.k.a. connected branching)



An *arborescence* w.r.t. $r, T(V, F)$, of a directed graph $G(V, E)$ is a spanning tree of G if we ignore the direction of edges; and there is a path in T from r to each node in $V - \{r\}$ if we take the direction of edges into account.

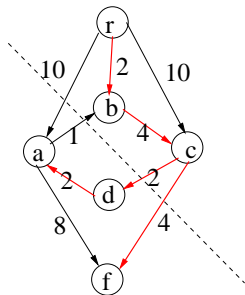
- A subgraph $T(V, F)$ is an arborescence w.r.t. root r iff T has no cycles, and for each node $v \neq r$, there is exactly one edge in F that enters v .
- A directed graph G has an arborescence rooted at r iff there is a directed path from r to each of $V - \{r\}$.

Problem: Min-Cost Arborescence (MCA)



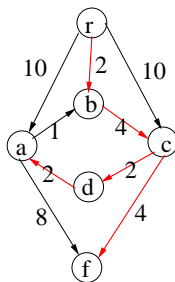
Given a weighted directed graph $G(V, E)$ with a distinguished root node r and with a positive cost w_e on each edge e , compute an arborescence rooted at r of minimum total cost.

MST cut property (of undirected graphs) not applicable



ab is not part of MCA

MST cycle property (of undirected graphs) not applicable



max weighted edge *bc* of cycle *abcd* is part of MCA

Outline

1 Ellipsoid Method

2 Edmonds Branching Algorithm

LP for the problem (P)

$$\min \sum_{(i,j) \in E} c_{ij} x_{ij}$$

$$\sum_{(i,j) \in E, i \in S, j \in V-S} x_{ij} \geq 1 \quad \text{for every } S \subseteq V, r \in S$$

$$x_{ij} \in \{0, 1\} \quad \text{for every } (i,j) \in E$$

Relaxed LP

$$\min \sum_{(i,j) \in E} c_{ij} x_{ij}$$

$$\sum_{(i,j) \in E, i \in S, j \in V-S} x_{ij} \geq 1 \quad \text{for every } S \subseteq V, r \in S \quad \text{————— (1)}$$

$$0 \leq x_{ij} \leq 1 \quad \text{for every } (i,j) \in E \quad \text{————— (2)}$$

(1) essentially ensures that for every $r \in S, S \subset V$, $\delta^+(S)$ has positive weight

Edmonds has proven that the relaxed LP polytope is same as P — **not proved in class**

Separation Oracle

- explicitly detecting a violated one in (2) \longrightarrow takes linear time
- detecting a violated one in (1) is equivalent to finding a r - t min-cut for some $t \in V - \{r\}$ such that the cut capacity is less than one \longrightarrow takes $O((n-1)n^3)$ time

i.e., although the number of constraints in the LP are exponential, (weakly) polynomial algorithm exists through Ellipsoid method!

Outline

1 Ellipsoid Method

2 Edmonds Branching Algorithm

Properties

For each vertex $v \in V - \{r\}$, let S_v be the arc set entering v ; suppose choosing a least cost arc in S_v for every vertex v , caused an arborescence $T(V, F)$, then T is a min-cost arborescence.

- consider G' with modified costs: subtract w_{e_v} from each arc in S_v for any/every vertex v of $G(V, E)$ wherein e_v is the least cost arc entering v
 - $T(V, F)$ is an optimal arborescence for G iff $T'(V, F)$ is an optimal arborescence for G'
 - further, the cost of T' is zero

Properties (cont)

In other words,

- when the direction of arcs is ignored, if T' is a spanning tree of G' , then T' is a min-cost arborescence.
- otherwise, T' must be having at least one zero cost cycle.

Properties (cont)

Let C be a zero cost cycle in G' , then there is an optimal arborescence T'' rooted at r that has exactly one arc entering C in G' .

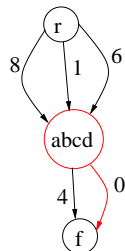
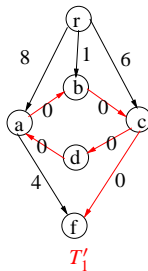
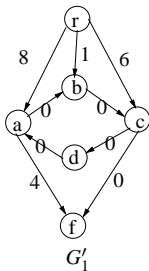
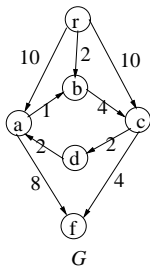
- among all paths that lead to C from r in G' , choose the shortest one, say P , in forming T'' ; let P incident to $v \in C$; include all arcs of C except the one that has v as the head
- a T that has two paths to C from r is an optimal arborescence $\Rightarrow T''$ is an optimal arborescence

(Greedy) Algorithm

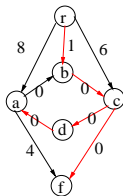
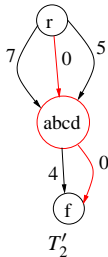
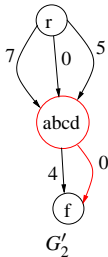
FindMincostArbo(G)

- 1 obtain G' from G (as detailed above)
- 2 obtain not necessarily connected T' from G' (as detailed above)
- 3 if T' is an arborescence, then return T'
- 4 contract vertices of G' that correspond to some cycle C in T' , and let the resultant graph be G''
- 5 $(V'', F'') \leftarrow \text{FindMincostArbo}(G'')$
- 6 extend (V'', F'') to an arborescence (V', F') in G' by including all arcs of $F'' \cup C$ except one arc in C whose head is a head of some arc in F''

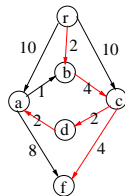
Algorithm in Execution



G'_2 : G'_1 after contracting $abcd$ cycle in T'_1



expand $abcd$ and remove ab to obtain a MCA in G'_1



corresponding MCA in G

Correctness

induction on the number of nodes in G

Analysis

$O(nm)$ time