Combinatorial Optimization

CS 491G

Shortest Path Problem

Babak Khorrami

Shortest Path problems are ubiquitous in real-world applications; both in their own right as well as in the modeling of certain optimization problems. There are a number of versions of the Shortest Path problem, viz. Single-Source, All-Pairs, etc. In this course, we shall focus on the Single Source Shortest Path (SSSP) problem only.

Digraph: A *directed graph* or *digraph* G consists of disjoint finite sets $V = V(G)$ of nodes and $E = E(G)$ of arcs and functions associating with each $e \in E$ a tail $t(e) \in V$ and a head $h(e) \in V$. In other words, each arc has two end nodes, and a direction from one to the other.

Shortest Path Problem

Input: A digraph G, a node $r \in V$, and a real cost vector $(c_e \in E)$.

Objective: To find, for each $v \in V$, a dipath from r to v of least cost (if one exists)

Although the problem specifies real costs $(c_e \in E)$, for practical purposes, we can assume that the costs are rational or even integral!

Consider the following example:

Example 1. Find the shortest path from source node *r* to *a* in the following network,

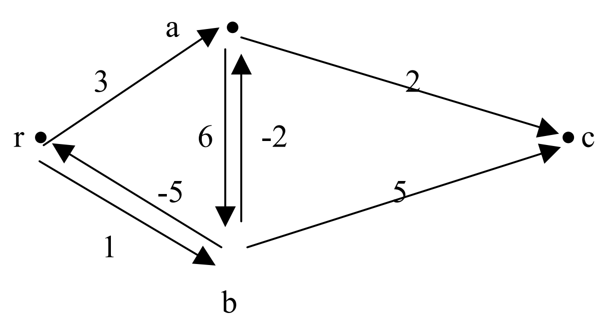


Figure 1.

There is no shortest path from *r* to *a.*

Reason: digraph depicted in Figure 1 contains negative cost cycle, the cycle between *r* and *b.* Instead of going from *r* to *a* directly, one can travel to *b* and then come back to *r* and make the cost -4 plus 3 to go to *a.* One can keep traveling from r to b and lowering the cost in each trip. Hence the shortest path problem is not defined in the presence of negative cost cycles.

Shortest Path Structure: The structure containing the shortest path from the source node *r* to every other node in the network is called the shortest path structure.

Observation: Shortest path structure should be a tree.

Reason: If there is a cycle in the shortest path structure between nodes *a* and *b*, by definition, should have a positive cost (weight). One can remove that cycle and still reach *b* from *a*, at cost which is at most the original cost!.

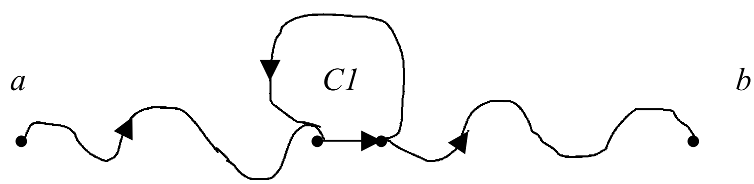


Figure 2.

Negative cost cycle can occur in a number of applications. One application is the *currency arbitrage problem.*

<u>Statement of the problem:</u>

We are given *m* currencies $c_1, c_2, \ldots c_m$. and the matrix R_{ij} of pairwise conversion rates, where r_{ij} represents the number of units of c_j that one can get from 1 unit of c_i. The question that we face is the following: Can we start with k units of some currency, say c_i, go through a series of conversions to other currencies and finally return to currency c_i, having more than k units of c_i? The phenomenon which makes such a trip possible is called arbitrage. The following relationship holds for all currencies:

$r_{ik} = r_{ij}\ r_{jk}$.

We construct a complete directed graph, having nodes $c_1, c_2, \ldots c_m$ and weight $-\log r_{ij}$ on the edge between c_i and c_j.

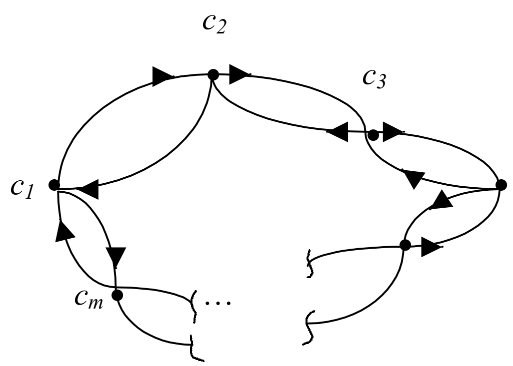


Figure 3

Claim: There exists arbitrage if and only if there exists a negative weight cycle in the above graph.

Proof: Exercise!

<u>Feasible Potentials:</u>

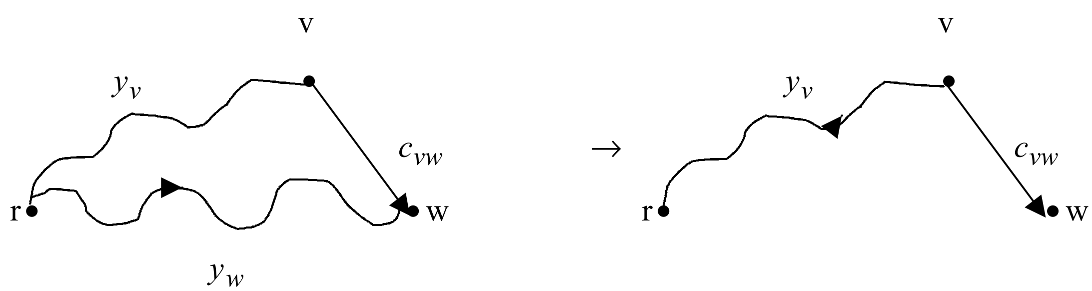
A vector $\vec{y}$ is given, $\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$ and it estimates the shortest paths from r to every other vertex in the network. We call $\vec{y}$ a *feasible potential* if it satisfies the following conditions:

(1) $y_r = 0$

(2) $y_v + c_{vw} \geq y_w$, for all $vw \in E$

Obviously the shortest path from source r to itself is zero. The second condition is the basic idea behind all methods for solving the shortest path problems. Suppose there exists a dipath from r to v of cost y_v for each $v \in V$ and we find an arc $vw \in E$ satisfying $y_v + c_{vw} < y_w$. One can improve y_w, by adding the vw to the dipath and going from r to w through v and the cost of the dipath from r to w would be $y_v + c_{vw} = y_w$.

If an assignment $\vec{y}$ is given in which $y_r = \alpha \neq 0$ and $y_v + c_{vw} \geq y_w$, one can obtain the feasible potential by subtracting y_r from all y_vs, $v \in V$. $\vec{y}' = \begin{bmatrix} y_1 - y_r \\ y_2 - y_r \\ \vdots \\ y_n - y_r \end{bmatrix}$. Hence the important condition is $y_v + c_{vw} \geq y_w$. The following figure shows the notion of appending arc *vw* to the dipath between *r* and *w* and improving the y_w.



(a) $y_v + c_{vw} < y_w$ (b) $y_v + c_{vw} = y_w$

Figure 4.

<u>Lemma 1:</u>

Let $\vec{y}$ be a feasible potential and let P be a dipath from r to v. Then, $c(P) \geq y_v$ (i.e. cost of path (P) $\geq y_v$).

Proof: Let $v_0, e_1, v_1, e_2, \ldots, e_k, v_k$, where $v_0 = r$ and $v_k = v$ be the dipath P. Then

$$c(P) = \sum_{e_i \in P} c(e_i) \geq \sum_{i=1}^{k} (y_{v_i} - y_{v_{i-1}}) = y_{v_k} - y_{v_0} = y_v .$$

Subpaths of shortest paths are shortest *paths*, for instance *v* is in the least cost dipath *P* from *r* to *w,* then *P* splits into two dipaths, P_1 from *r* to *v* and P_2 from *v* to *w.* Obviously if P_1 is not the least cost dipath from *r* to *v,* one can replace it by a better dipath and at the same time obtain a better dipath from *r* to *w*.

<u>Ford's Procedure:</u>

Lemma 1 provides a stopping condition for the shortest path problem. Suppose there exists a feasible potential $\bar{y}$ and for each $v \in V$ there is a y_v, which is the least cost path from r to v. If there exists a vertex w and an arc vw, which violate $y_v + c_{vw} \geq y_w$, we replace y_w with $y_v + c_{vw}$. This procedure can be initialized by allowing $y_r = 0$ and ($y_v = \infty$) for $v \in V$ and $v \neq r$. The least cost dipath from *r* to *w*, which contains arc *vw*, will satisfy $y_v + c_{vw} = y_w$. This dipath contains the least cost dipath from *r* to *v* plus arc *vw*. So knowing the last arc information at each node allows us to trace the least cost dipath from *r*. To do this, we need to keep the predecessor, $p(w)$, of each node $w \in V$, and set $p(w)$ to v, whenever the least cost dipath to *w*, y_w is set to be $y_v + c_{vw} = y_w$. An arc *vw* violating $y_v + c_{vw} \geq y_w$ is called incorrect. To correct *vw*, one needs to set $y_v + c_{vw} = y_w$ and $p(w) = v$.

To start Ford's procedure one needs to initialize $\bar{y}$, $\bar{p}$ which means to set $y_r = 0$, $p(r) = 0$, $y_v = \infty$ and $p(v) = -1$ for $v \in V$ and $v \neq r$. $p(v) = -1$ means that the predecessor of *v* is still not defined.

Ford's Procedure

Initialize y, p;
While y is not a feasible potential
    Find an incorrect arc *vw* and correct it.

Example: Consider the following network, apply the Ford's Procedure and obtain the shortest paths to each node.

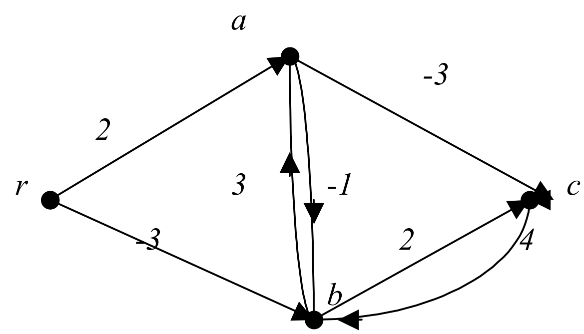
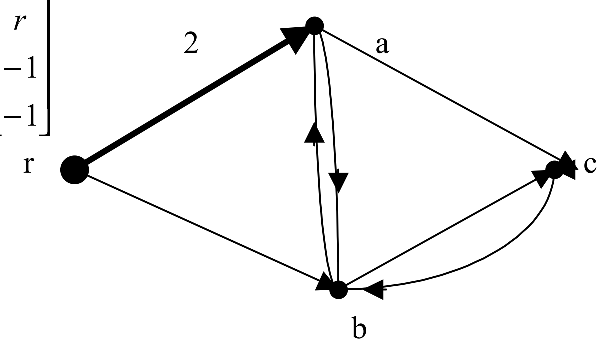


Figure 5

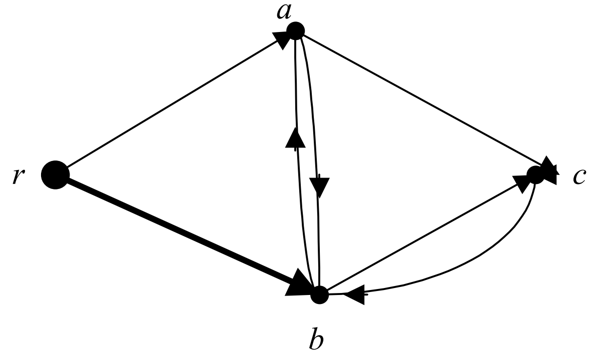
Initialize y, p;

$$y = \begin{bmatrix} y_r \\ y_a \\ y_b \\ y_c \end{bmatrix} = \begin{bmatrix} \infty \\ \infty \\ \infty \\ \infty \end{bmatrix}, p(r) = 0,\ p(a) = p(b) = p(c) = -1.$$

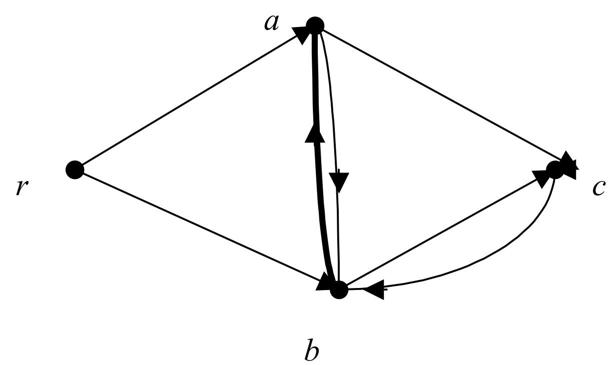
1) $vw = ra$, $y = \begin{bmatrix} y_r \\ y_a \\ y_b \\ y_c \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ \infty \\ \infty \end{bmatrix}, \begin{bmatrix} p(r) \\ p(a) \\ p(b) \\ p(c) \end{bmatrix} = \begin{bmatrix} 0 \\ r \\ -1 \\ -1 \end{bmatrix}$



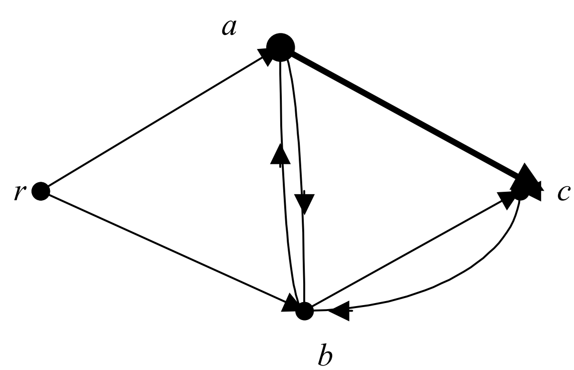
2) $vw = rb$, $y = \begin{bmatrix} y_r \\ y_a \\ y_b \\ y_c \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ -3 \\ \infty \end{bmatrix}, \begin{bmatrix} p(r) \\ p(a) \\ p(b) \\ p(c) \end{bmatrix} = \begin{bmatrix} 0 \\ r \\ r \\ -1 \end{bmatrix},$



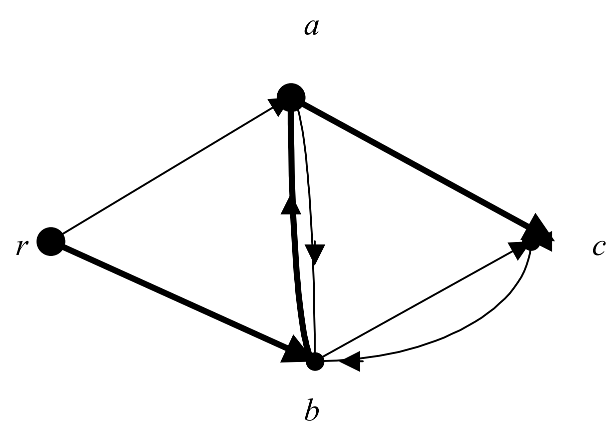
$$3)\ vw = ba,\ y = \begin{bmatrix} y_r \\ y_a \\ y_b \\ y_c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -3 \\ \infty \end{bmatrix},\ \begin{bmatrix} p(r) \\ p(a) \\ p(b) \\ p(c) \end{bmatrix} = \begin{bmatrix} 0 \\ b \\ r \\ -1 \end{bmatrix},$$



$$4)\ vw = ac,\ y = \begin{bmatrix} y_r \\ y_a \\ y_b \\ y_c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -3 \\ -3 \end{bmatrix},\ \begin{bmatrix} p(r) \\ p(a) \\ p(b) \\ p(c) \end{bmatrix} = \begin{bmatrix} 0 \\ b \\ r \\ a \end{bmatrix},$$



If we consider other arcs, *cb, bc, ab,* we will not find and incorrect arc so the shortest path structure is:



$$y = \begin{bmatrix} y_r \\ y_a \\ y_b \\ y_c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -3 \\ -3 \end{bmatrix}, \begin{bmatrix} p(r) \\ p(a) \\ p(b) \\ p(c) \end{bmatrix} = \begin{bmatrix} 0 \\ b \\ r \\ a \end{bmatrix}.$$

<u>Lemma 2:</u>

If (G, c) has no negative-cost cycle, then at any stage of the execution of *Ford's Procedure,* we have:

(i) If $y_v \neq \infty$, then it is the cost of a simple dipath from *r* to *v*.

(ii) If $p(v) \neq -1$, then *p* defines a simple dipath from *r* to *v* of cost at most y_v.

Proof:

Let y_v^j be the value of y_v, which is the cost of a dipath, at *jth* iteration of Ford's procedure. Assume that the dipath is not simple, hence there is a sequence of nodes, $v_0, v_1, v_2, ..., v_k = v_0$ and iteration numbers $q_0 < q_1 < q_2 < ... < q_k$ such that:

$$y_{v_{i-1}}^{q_{i-1}} + c(v_{i-1}, v_i) = y_{v_i}^{q_i}, \qquad 1 \leq i \leq k.$$

The cost of the resulting dicircuit is:

$$\sum c(v_{i-1}, v_i) = \sum (y_{v_i}^{q_i} - y_{v_{i-1}}^{q_{i-1}}) = y_{v_k}^{q_k} - y_{v_0}^{q_0}.$$

one should consider that the value of y_v at the last iteration, q_k, has been lowered and $y_{v_i}^{q_i} - y_{v_0}^{q_0} < 0$. The dipath has a negative cost which is a contradiction and (i) is proved.

To prove (ii), consider that *p* defines a closed path from *r* to *v*. there is a sequence, $v_0, v_1, v_2, ..., v_k = v_0$ and $p(v_i) = v_{i-1}$. Since $y_v - y_{p(v)} \geq c(p(v), v)$, the cost of the resulting closed dipath is less than or equal to zero. And consider a case in which the predecessor has been most recently assigned, which means the value of $y_{p(v)}$ has been assigned and is lowered. Then the cost is strictly less than zero, which is a contradiction, negative cost cycle.

Consider the dipath *P*, $v_0, e_1, v_1, e_2, v_2, ..., e_k, v_k = v$, $v_0 = r$ and $p(v_i) = v_{i-1}$ for $1 \leq i \leq k$.

The cost of this dipath: $c(P) \leq \sum y_{v_i} - y_{v_{i-1}}) = y_v - y_r = y_v$, so the cost of this dipath is at most y_v and that's what we need.