CS 491 Minimum-Cost Flow Problems

Zi Zhuang

1. Minimum-cost Flow Problem

We recall that the *Maximum Flow Problem* is the problem defined below.

$$\text{Maximize } f_{\vec{x}}(s) \tag{1.0}$$

Subject to

$$f_{\vec{x}}(v) = 0, \text{for all } v \quad V \setminus \{r, s\}$$

$$0 \le x_e \le u_e, \text{ for all } e \in E .$$

r is the source and *s* is the sink.

In a *Minimum-cost Flow* problem, every node has a demand and the goal is to find a feasible flow $\vec{x}$ of minimum cost .
We take as our standard minimum-cost flow model as the following problem:

$$\text{Minimize } \Sigma(c_e x_e : e \in E). \tag{1.1}$$

Subject to

$$f_{\vec{x}}(v) = b_v, \text{for all } v \in V$$

$$0 \le x_e \le u_e, \text{ for all } e \in E .$$

$f_{\vec{x}}$----Net flow into *v,* or the excess of $\vec{x}$ at *v*

b_v-----Demand of node v, If $b_v < 0$, we can think of the demand as "supply"

u_e-----Capacity of edge *e*

The *Transportation problem* is a special case of the *Minimum-cost flow* problem. Here *G* is bipartite with bipartition {*P, Q*} and we are given positive numbers $a_p, p \in P$ and $b_q, q \in Q$, as well as costs $c_{pq}, pq \in E$ as shown in Figure 1. We define the problem as

$$\text{Minimize } \Sigma(c_{pq} x_{pq} : pq \in E). \tag{1.2}$$

Subject to

$$\Sigma(x_{pq} : p \in P, pq \in E) = b_q, \text{ for all } q \in Q$$

$$\Sigma(x_{pq} : q \in Q, pq \in E) = a_p, \text{ for all } p \in P$$

$$x_{pq} \ge 0, \text{ for all } pq \in E .$$

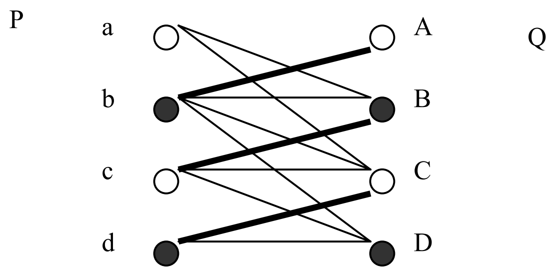

Figure 1. A bipartite graph

If we allow also constraints of the form $x_{pq} \le u_{pq}$ with u_{pq} finite, we have a *capacitated transportation problem.* In either case, multiplying each of the second group of equations by –1, and putting $b_v = -a_v$ for $v \in P$, we get a problem of type (1.1). If u_e is ∞, for all $e \in E$, such a problem is called a *transshipment problem.*

The dual of (1.1), obtained by first writing the constraints $x_{vw} \le u_{vw}$ as $-x_{vw} \ge -u_{vw}$, is

$$\text{Maximize } \Sigma(b_v y_v : v \in V) - \Sigma(u_{vw} z_{vw} : vw \in E) \qquad (1.3)$$

$$\text{Subject to}$$

$$-y_v + y_w - z_{vw} \le c_{vw}, \text{ for all } vw \in E$$

$$z_{vw} \ge 0, \text{ for all } vw \in E.$$

If $u_e = \infty$, then there is no dual variable z_{vw}, because the corresponding constraint $x_{vw} \le u_{vw}$ is missing from (1.1).

Given $\vec{y}_v\{y_v \text{ is associated with note } v\}, v \in V$, it is convenient to denote $c_{vw} + y_v - y_w$ by $\bar{c}_{vw}$, and call $\bar{c}_{vw}$ *reduced cost* of edge *vw.* Hence a feasible solution $(\vec{y}, \vec{z})$ of (1.3) satisfies $\bar{c}_e \ge 0$ if $u_e = \infty$. Moreover, if $u_e \ne \infty$, then z_e must satify only $z_e \ge 0$ and $z_e \ge -\bar{c}_e$. Therefore, $z_e = \max(0, -\bar{c}_e)$.

In terms of $\vec{y}$ the complementary slackness conditions for the pair (1.1), (1.3) of the problems can be written

$$x_e > 0 \Rightarrow -\bar{c}_e = \max(0, -\bar{c}_e)$$

$$\Rightarrow -\bar{c}_e \le 0$$

$$z_e > 0 \Rightarrow x_e = u_e, \text{that is} \Rightarrow -\bar{c}_e > 0 \Rightarrow x_e = u_e$$

Thus only $\vec{y}$ is necessary to describe a feasible or optimal dual solution. The Complementary Slackness Theorem then gives the desired characterization of optimality.

Theorem 1. *A feasible solution $\vec{x}$ of (1.1) is optimal if and only if there exits a vector $\vec{y}_v, v \in V$ satisfying for each $e \in E$*

$$\bar{c}_e < 0 \Rightarrow x_e = u_e (\neq \infty)$$
$$\bar{c}_e > 0 \Rightarrow x_e = 0$$

Proof: Follows from the above discussion.

Definition 1: Given a path *P* in *G*, not necessarily directed, we define the cost of *P* as

$$c(P) = \Sigma(c_e : e \text{ forward in P}) - \Sigma(c_e : e \text{ reverse in P})$$

Definition 2: Given $\vec{x} = x_e : e \in E$, satisfying $0 \le \vec{x} \le u$, we define the auxiliary digraph $G(\vec{x})$ as we did in the context of maximum flows, except that we include a notion of arc cost.

For each $vw \in E$, $x_{vw} \le u_{vw}, c'_{vw} = c_{vw}$

For each $vw \in E$, $x_{vw} \ge 0$, we include $wv \in E(G(x))$ with cost $\text{c}'_{\text{wv}} = -c_{vw}$

Then every $\vec{x}$-incrementing path in *G* corresponds to a dipath in *G($\vec{x}$)* having the same cost.

In particular, if $\vec{x}$ is a feasible solution of (1.1), then an $\vec{x}$-incrementing circuit of negative cost gives a solution of lower cost.

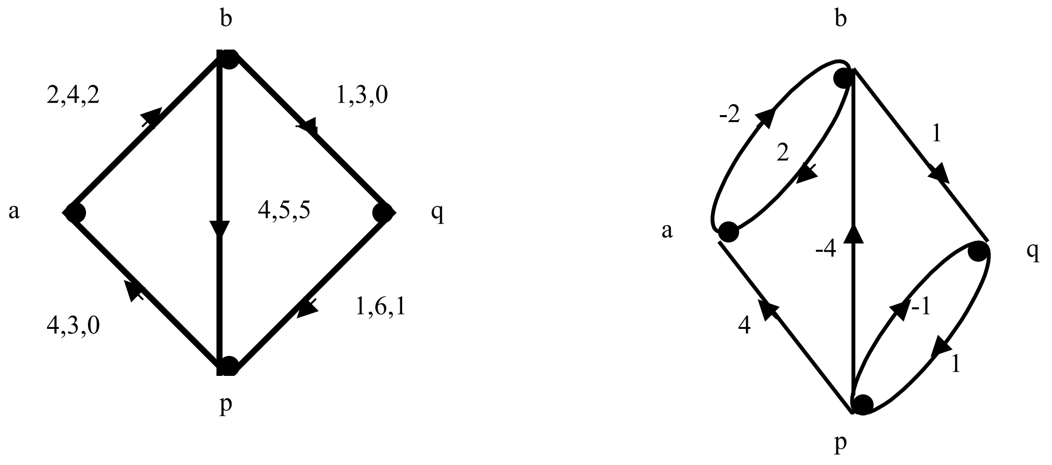

Figure 2. ($\vec{G}$, $\vec{c}$, $\vec{u}$, $\vec{x}$) and its auxiliary digraph

Figure 2 illustrates some of these ideas. On the left is G; numbers beside arc *e* are c_e, u_e, x_e. On the right is G($\vec{x}$). Notice that G($\vec{x}$) has a dicircuit of cost –2, having node-sequence p,b,q,p.

The corresponding circuit of G can be used to produce a feasible solution whose cost is lower by 6.

Theorem 2: *A feasible solution $\vec{x}$ of (1.1) is optimal if and only if there is no $\vec{x}$-incrementing circuit having negative c-cost.*

Proof: Add a node r to $G(\vec{x})$ and, for each $v \in V$, an arc rv with $c'_{rv} = 0$. If we solve the shortest path problem in this new digraph G', we get either a negative-cost dicircuit or a feasible potential. A negative-cost dicircuit cannot use r, and hence corresponds to a negative-cost $\vec{x}$-incrementing circuit of G.A feasible potential $\vec{y}$ satisfies

$$\begin{aligned} & y_v + c'_{vw} \geq y_w, \text{for all } vw \in E(G(x)). \\ \Rightarrow\ & y_v + c_{vw} \geq y_w, \text{ if } x_{vw} < u_{vw} \\ \Rightarrow\ & y_v - c_{vw} \geq y_w, \text{if } x_{vw} > 0 \end{aligned}$$

which are equivalent to the conditions of Theorem 1.

An immediate consequence of the method of proof, is the following:

Theorem 3: *Suppose that (1.1) has an optimal solution and c is integral. Then $\vec{y}$ in Theorem 1 can be chosen integral; equivalently, the dual linear-programming problem of (1.1) has an optimal solution that is integral.*

Proof: Since the cost-vector is integral, the cost of the shortest path in the proof of Theorem 2 is also integral. So $\vec{y}$ can be chosen integral.

Theorem 4: *Suppose that (1.1) has a feasible solution. Then it has an optimal solution if and only if there exists no negative-cost dicircuit of G, each of whose arcs had infinite capacity.*

Proof: If there is such a circuit, then clearly there is no optimal solution. Otherwise, we can choose $z_e = \max(0, -\bar{c}_e)$ if $u_e \neq \infty$ or simply find $\vec{y}$ such that $\bar{c}_e \geq 0$ for all vw with $u_e = \infty$. Deleting all arcs of finite capacity from G, adding a new node r and an arc rv of cost 0 for each $v \in V$. The resulting digraph had no negative-cost dicircuit and hence has a feasible potential $\vec{y}$, which has the desired property.

1.1 Reduction from min-cost flow to transshipment problem

We can transform the (1.1) problem into a transshipment problem by taking the reduction. See Figure 3. Given an arc e=vw with $u_e \neq \infty$, replace e by two nodes p,q and three arcs vp,qp,qw with $c_{vp} = c_e, c_{qp} = c_{qw} = 0, b_p = u_e, b_p = -u_e$; the new arcs have infinite capacity. And the new problem has O(m+n) nodes and O(m) arcs.

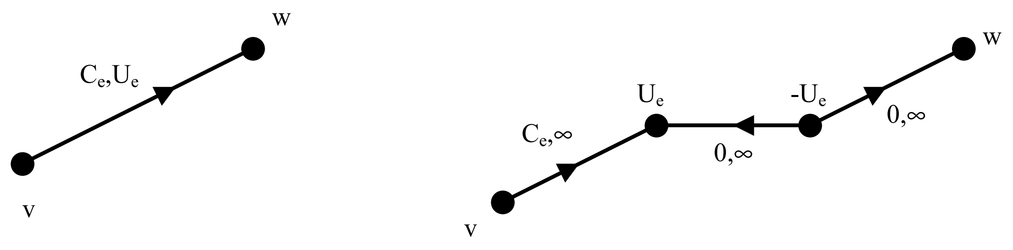

Figure 3.Getting rid of capacities

2. Primal Minimum-cost Flow Algorithm

The primal minimum-cost flow algorithm was first proposed by Kantorovich [1942]. The name comes from the fact that at every step such an algorithm has a feasible solution to the "primal" linear-programming problem, that is, the minimum-cost flow problem.

```
Augmenting Circuit Algorithm for the Min Cost Flow Problem

Find a feasible solution $\vec{x}$;
While there exists an augmenting circuit
        Find an augmenting circuit C;
        If C has no reverse arc and no forward arc of finite capacity, then stop;
        Augment $\vec{x}$ on C.
```

How to find a feasible solution? We form a digraph G' with $V' = V \quad \{r,s\}$. Each $vw \quad E$ is an arc of G', and has capacity u_{vw}. For each $v \quad V$ with $b_v < 0$, there is an arc *rv* with $u_{rv} = -b_v$. For each $v \quad V$ with $b_v > 0$, there is an arc *vs* with $u_{vs} = b_v$. Now apply the Augmenting Path Algorithm to find an (r,s)-flow in G' of value $\quad (b_v : v \quad V, b_v > 0)$. Then the restriction of the flow to G is a feasible solution to the original problem.

Figure 4 shows an bad example for the Augmenting Circuit Algorithm. Add an arc *sr* with $c_{sr} = -1$ and $u_{sr} = \infty$, give all original arcs a cost of zero, and put all demands equal to zero. Since we know that the number of steps in the basic augmenting path algorithm can be unacceptably high, we can get the same conclusion for the Augmenting Circuit Algorithm. So the algorithm may not terminate.

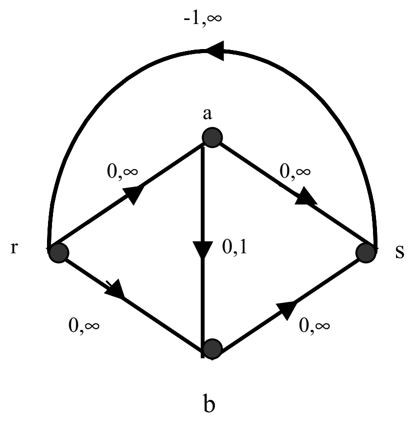

Figure 4. A bad example for the Augmenting Circuit Algorithm

How to find "good" augmenting circuits? One idea is to find a most-negative augmenting circuit at each step. However, in the above maximum flow example, every negative cost augmenting circuit is most negative, since each has cost –1, so these may not provide a good choice. A better idea is to hind an augmenting circuit whose" average arc cost" is small. The mean cost of a circuit *C* of *k* arcs is its cost divided by *k*. Notice that in the maximum flow setting, a minimum-mean-cost circuit is a good choice--- it corresponds to a shortest augmenting path!

3. The Network Simplex Method

The *Network Simplex Method* is an interpretation of the linear-programming simplex method applied to the minimum cost flow problem.

We assume that the digraph *G* has a spanning tree. It is convenient to present the algorithm first for the special case of the transshipment problem, that is, to assume that $u_e = \infty$ for every arc *e*. A tree solution for the transshipment problem is

$$f_x(v) = b_v, \text{ for all } v \in V$$
$$x_e = 0 \text{ for all } e \notin T$$

Proposition 1: Let *v*, *w* be nodes of a tree *T*. Then there is a unique simple path from *v* to *w* in *T*.

A tree *T* and arc *h*=*pq* of *T* determine a partition of the nodes into two sets, *R*(*T*,*h*) and *V**R*(*T*,*h*). See Figure 5.

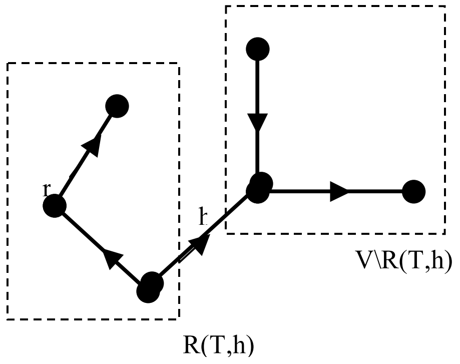

Figure 5. Partition induced by a tree and an arc

R(*T,h*) is the set of those nodes *v* such that the simple path in *T* from *r* to *v* does not use *h*. Obviously, $r \in R(T,h)$. And *h* is the only arc of *T* having one end in *R*(*T,h*) and one end not in *R*(*T,h*). Thus in any tree solution *x* associated with *T* the net flow into *R*(*T,h*) must be entirely carried by *h*. That is,

$$x_h = b(R(T,h)), \text{ if } q \in R(T,h)$$
$$x_h = -b(R(T,h)), \text{Otherwise}$$

Proposition 2: A tree *T* uniquely determines its tree solution.

Theorem 5: *If (G, $\vec{b}$) has a feasible solution, then it has a feasible tree solution. If it has an optimal solution, then it has an optimal tree solution.*

Proof: Let $\vec{x}$ be a feasible solution. If $\vec{x}$ is not a tree solution, then there is a circuit C, each of whose arcs carries positive flow. We may assume that C has at least one reverse arc, since otherwise we can replace C by the circuit defined by the same sequence taken in reverse. Now let $\varepsilon = \min(x_e : \text{e a reverse arc of C})$. We replace x_e by $x_e + \varepsilon$ if e is a forward arc of C, and by $x_e - \varepsilon$ if e is a reverse arc of C. The new *x* is feasible and has fewer arcs carrying positive flow. Continuing this procedure, we get a feasible tree solution.

If $\vec{x}$ is an optimal solution, observe that the cycle C must have cost zero. This is because neither C nor its reverse can have positive cost as per Theorem 2! Thus we can use the same construction that we used for the feasible case to conclude that if there is an optimal solution, there must be an optimal tree solution.

The *Network Simplex Method* maintains feasible tree solutions and looks for negative-cost circuits of a special kind. For each arc $e = vw \notin T$, there is a unique circuit C(*T,e*) having the following properties:

(a) Each arc of C(*T,e*) is an element of $T \cup \{e\}$;
(b) *e* is a forward arc of C(*T,e*);
(c) The initial node *s* of C(*T,e*) is the first common node of the simple paths in *T* from *v* and *w* to *r*.

Figure 6 illustrates these conditions.

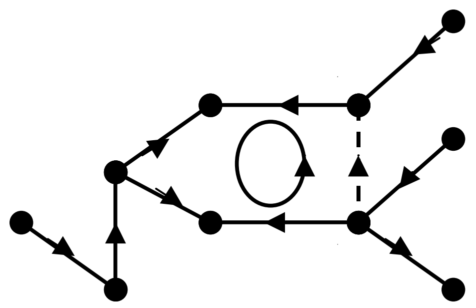

Figure 6. Example of C(T,e)

Consider $\vec{y}$, where y_v is defined as the cost of the simple path in T from r to v. Notice that, for any two nodes v, w, the cost of the simple path in T from v to w is just $y_w - y_v$. Hence the reduced costs $\bar{c}_{vw}$ defined by

$$\bar{c}_{vw} = c_{vw} + y_v - y_w \text{ satisfy}$$
$$\bar{c}_{vw} = 0 \text{ for all } vw \in T\,;$$
$$\bar{c}_{vw} \text{ is the cost of C}(T,e) \text{ for all } vw \in E \setminus T$$

It follows immediately from this that, if every C(T,e) has nonnegative cost, then the tree solution $\vec{x}$ determined by T satisfies the conditions of Theorem 1, so we have the following result.

Proposition 3: If the tree T determines the feasible tree solution $\vec{x}$ and C(T,e) has nonnegative cost for every $e \notin T$, the $\vec{x}$ is optimal.

Testing whether T satisfies this optimality condition is relatively easy.

1. Compute $\vec{y} \to O(n)$ time
2. Check $\bar{c}_e \geq 0$ for all $e \to O(m)$ time

Are we sure a circuit of this form is actually an augmenting circuit? We cannot, because it may have a reverse arc having zero flow. This difficult is illustrated in Figure 7. Here the numbers at the nodes are the demands, the pair c_e, x_e is on arc e, and the tree arcs are the unbroken ones. Then C(T,wr) has positive cost, and C(T,vq) has a reverse arc vw having zero flow. But $\vec{x}$ is not optimal—sending one unit of flow on the circuit with node-sequence r,v,q,w,r gives a better solution.

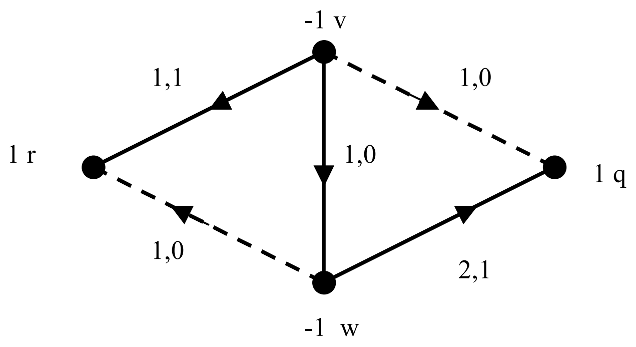

Figure 7. No augmenting circuit of type C(T,e)

Using the Network Simplex Method, the basic idea is: Use C(T,e) to find a different tree $\hat{T}$ having the same tree solution $\vec{x}$. Like the Figure 6 example, $\hat{T} = \{vr, wq, vq\}$. We can state the preliminary form of the algorithm as below.

Network Simplex Method for the Transshipment Problem

```
Find a tree T whose associated flow $\vec{x}$ is feasible;
Compute $y_v$, the $(r,v)$ path cost in $T$, for each node $v$;
While there exists an arc $e = vw$ such
        That $\bar{c}_{vw} = c_{vw} + y_v - y_w < 0$
    Find such an arc $e$;
    If C($T,e$) has no reverse arc, then stop;
    Compute $\theta = \min(x_j : j \text{ a reverse arc of } C(T,e))$;
    Find a reverse arc $h$ of C($T,e$) with $x_h = \theta$;
    Augment $\vec{x}$ by $\theta$ on C($T,e$);
    Replace $T$ by $(T \cup \{e\}) \setminus \{h\}$;
    Update $\vec{y}$.
```

How to find the initial tree and flow? We use the tree T whose arcs are $\{rv : v \in V \setminus \{r\}, b_v \geq 0\} \cup \{vr : v \in V \setminus \{r\} b_v < 0\}$. If such an arc does not exist in G, add it to G, with a large enough cost. These extra arcs are called *artificial arcs*. If the original problem is feasible, then in the optimal solution, no artificial arc can carry positive flow!

Proposition 4: In an iteration of the Network Simplex Method, let T be the old tree, $\hat{T} = (T \cup \{e\}) \setminus \{h\}$ be the new tree, y be the old path costs, and $\hat{y}$ be the new path costs. Then, where $e = vw$,

$$\hat{y}_q = y_q, \text{ for all } q \in R(T,h)$$
$$\hat{y}_q = y_q + \bar{c}_e, \text{ for all } q \notin R(T,h) \text{ if } v \in R(T,h)$$
$$\hat{y}_q = y_q - \bar{c}_e, \text{ for all } q \notin R(T,h) \text{ if } w \in R(T,h).$$

Proof: If $q \in R(T,h)$ then the *(r,q)*-path in the new tree is the same as that in the old tree, so the cost is the same. Now suppose that $q \notin R(T,h)$, and that $v \in R(T,h)$. Then, where $e = vw$, the *(r,q)*-path in $\hat{T}$ consists of the *(r,v)*-path in T, together with *e*, together with the *(w,q)*-path in *T*. (See Figure 7.) Therefore, $\hat{y}_q = y_v + c_e + (y_q - y_w) = y_q + \bar{c}_e$. For the last case, where $e = vw$, the *(r,q)*-path in $\hat{T}$ consists of the *(r,w)*-path in T, together with *e*, together with the *(v,q)*-path in *T*. We can tell that $\hat{y}_q = y_w - c_e + (y_q - y_v) = y_q - \bar{c}_e$.(See Figure 9.)

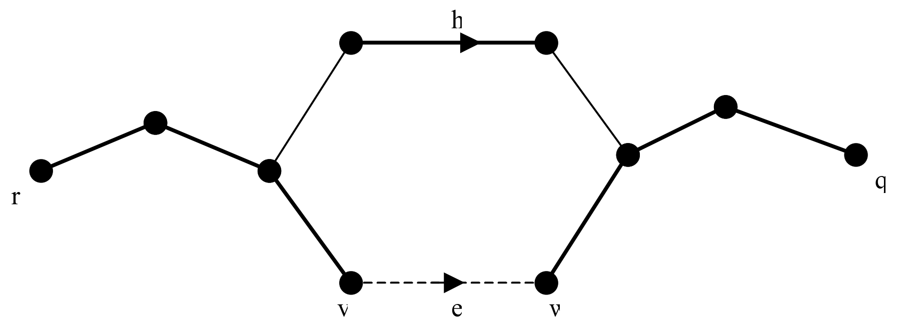

Figure 8. The (r,q)-path in $\hat{T}$,where $v \in R(T,h)$

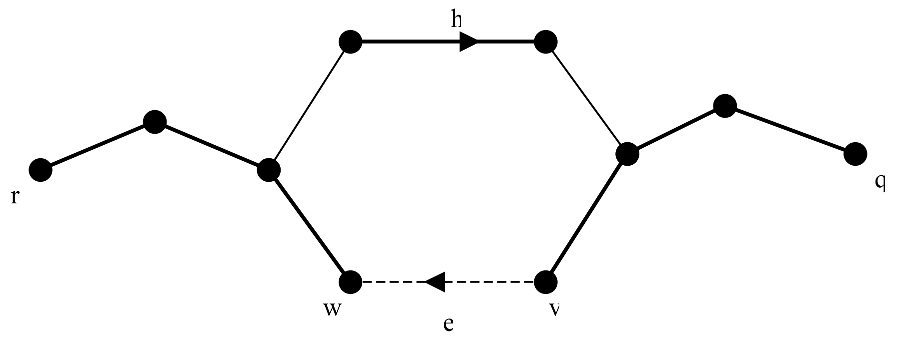

Figure 9. The *(r,q)*-path ,where $w \in R(T,h)$