The min cost flow problem (part II)

Giovanni Righini

University of Milan

UNIVERSITÀ DEGLI STUDI DI MILANO

Successive shortest paths algorithm

In this case the algorithm keeps the optimality of the flow and iteratively achieves feasibility with respect to the flow constraints.

At each iteration, the current flow x is the minimum cost flow among all flows of its value.

When the flow is maximum, then the algorithm stops.

Notation

Flow constraints:

$$e_i = b_i + \sum_{(j,i)\in A} x_{ji} - \sum_{(i,j)\in A} x_{ij} \quad \forall i \in N$$

We define $E = \{i \in N : e_i > 0\}$ and $D = \{i \in N : e_i < 0\}$.

Given a dual vector y, the corresponding reduced costs are

$$c_{ij}^{y} = c_{ij} - y_i + y_j \quad \forall (i,j) \in R(x).$$

Lemma 1: optimality conditions

Lemma 1. Let x be a min cost flow and let d be the min distance vector from $s \in N$ and the other nodes in $R(x)$ according to the reduced costs c^y. Then

1. x is still a min cost flow with respect to potentials $y' = y - d$;
2. $c_{ij}^{y'} = 0 \;\; \forall (i,j) \in P(s,k) \;\; \forall k \in N$, where $P(s,k)$ indicates the shortest path from s to k.

Lemma 1: optimality conditions

Proof (1). Since x is a min cost flow, the optimality conditions hold:

$$c_{ij}^{y} \geq 0 \ \ \forall (i,j) \in R(x).$$

For the properties of shortest paths (using a c^{y} cost function)

$$d_j \leq d_i + c_{ij}^{y} \ \ \forall (i,j) \in R(x).$$

By definition

$$c_{ij}^{y} = c_{ij} - y_i + y_j.$$

Therefore

$$d_j \leq d_i + c_{ij} - y_i + y_j \Rightarrow c_{ij} - (y_i - d_i) + (y_j - d_j) \geq 0 \Rightarrow c_{ij}^{y'} \geq 0 \ \forall (i,j) \in R(x).$$

Lemma 1: optimality conditions

Proof (2). Given any shortest path $P(s,k)$, we have

$$d_j = d_i + c_{ij}^{y} \;\; \forall (i,j) \in P(s,k).$$

Therefore

$$d_j = d_i + c_{ij} - y_i + y_j,$$

i.e.

$$c_{ij}^{y'} = 0 \;\; \forall (i,j) \in P(s,k).$$

Lemma 2: optimality conservation

Optimality conditions are initially satisfied because $c^y = c \geq 0$.

Lemma 2. Let x be a min cost flow and let x' be the flow obtained from x after sending flow along a shortest path from a node $u \in E$ to a node $v \in D$. Then x' is still a min cost flow.

Proof. From Lemma 1, $c_{ij}^{y'} = 0 \;\; \forall (i,j) \in P(u,v)$.

After sending flow along $P(u,v)$, some arc (j,i) can appear in $R(x')$ corresponding to an arc $(i,j) \in P(u,v)$.

However, $c_{ij}^{y'} = 0$ implies $c_{ji}^{y'} = 0$.

Therefore all reduced costs remain non-negative.

Pseudo-code

$x \leftarrow 0$
$y \leftarrow 0$
$e_i \leftarrow b_i \ \ \forall i \in N$
$E \leftarrow \{i \in N : e_i > 0\}$
$D \leftarrow \{i \in N : e_i < 0\}$
while $E \neq \emptyset$ **do**
    Select $u \in E$ and *Select* $v \in D$
    $(P(u,v), d) \leftarrow \mathit{ShortestPath}(u, v, R(x), c^y)$
    $y \leftarrow y - d$
    $\delta \leftarrow \min\{e_u, -e_v, \min_{(i,j) \in P(u,v)}\{r_{ij}\}\}$
    $(x, R(x), E, D, c^y) \leftarrow \mathit{Update}(u, v, \delta)$

Complexity

The total excess decreases by at least one unit for each iteration.

The initial excess is bounded by nB, where B is the maximum supply of a a node:

$$B = \max_{i \in N}\{|b_i|\}.$$

Therefore no more than nB iterations are required.

Every iteration requires the computation of a shortest path.

The resulting complexity is pseudo-polynomial.

However polynomial-time versions also exist (using scaling techniques).

A practical improvement

The computation of labels d from any node $u \in E$ can be stopped as soon as any node $v \in D$ is labelled permanently.

The dual update rule is

$$y_i \leftarrow \begin{cases} y_i - d_i & \text{if } i \text{ is labelled permanently} \\ y_i - d_v & \text{if } i \text{ is not labelled permanently} \end{cases}$$

This update rule guarantees that the reduced costs remain non-negative.

Primal-dual algorithm

Consider a flow network with a single excess node s a single deficit node t (wlog).

This is obtained by connecting all nodes i with an excess $b_i > 0$ to node s with arcs (s, i) of capacity b_i and all nodes i with a deficit $b_i < 0$ to node t with arcs (i, t) of capacity $-b_i$. Let z be the sum of all excesses.

The primal-dual algorithm iteratively solves a max flow problem on an admissible graph $A(x, y)$, which depends on the current flow x and a set of potentials y.

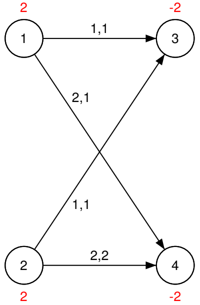
The admissible graph $A(x, y)$ contains only the arcs of the residual graph $R(x)$ that have zero reduced cost c^y.

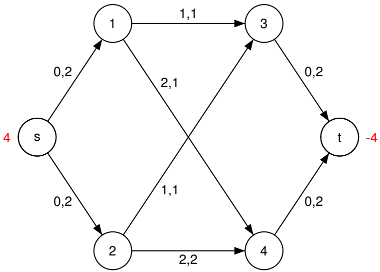
The residual capacity of each arc in $A(x, y)$ is the same as in $R(x)$.

Primal-dual algorithm: pseudo-code

$x \leftarrow 0$
$y \leftarrow 0$
$e(s) \leftarrow z$
$e(t) \leftarrow -z$
while $e(s) > 0$ **do**
  $d \leftarrow \mathit{ShortestPaths}(s, R(x), c^y)$
  $y \leftarrow y - d$
  Define $A(x, y)$
  $\mathit{ComputeMaxFlow}(s, t, A(x, y))$
  Update $e(s)$, $e(t)$, $R(x)$

An example

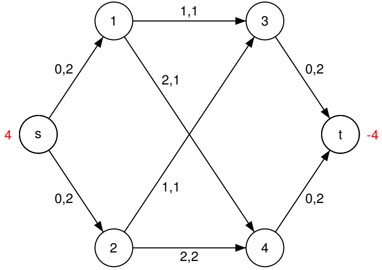




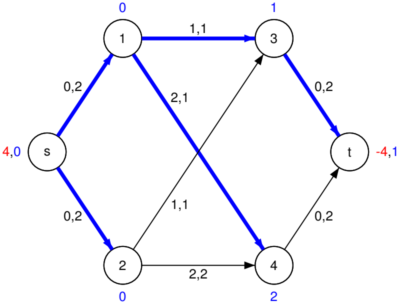
The original network with excess nodes 1 and 2 and deficit nodes 3 and 4.
Node labels: *b*.
Arc labels: (c, u).

The equivalent flow network.
Node labels: *e*.
Arc labels: (c, u).

An example

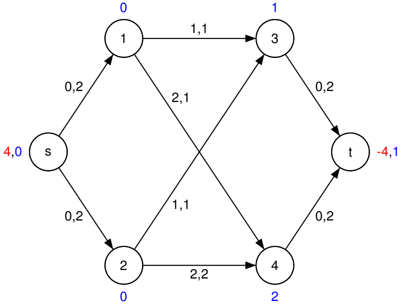


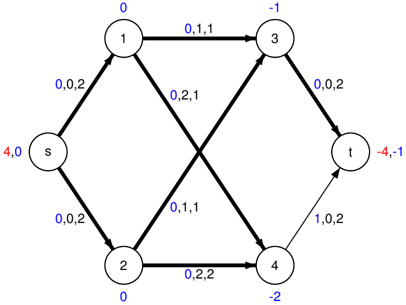
The equivalent flow network.
Node labels: e.
Arc labels: (c, u).



Dual iteration 1.
Shortest paths from s on $R(x)$.
Node labels: e, d.
Arc labels: (c, u).

An example

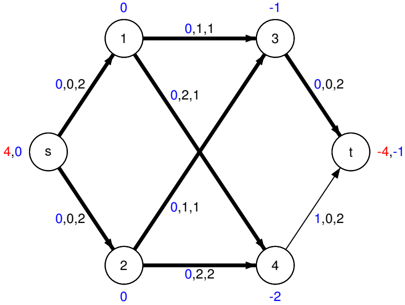


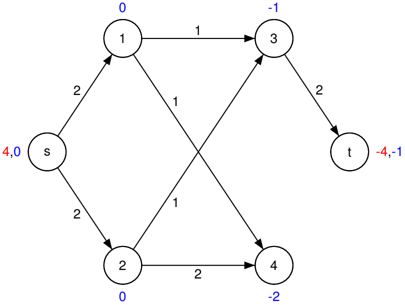


Dual iteration 1 on $R(x)$.
Node labels: e, d.
Arc labels: (c, u).

Potentials and reduced costs.
Node labels: e, y.
Arc labels: (c^y, c, u).

An example

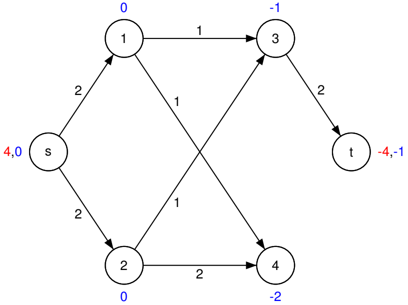




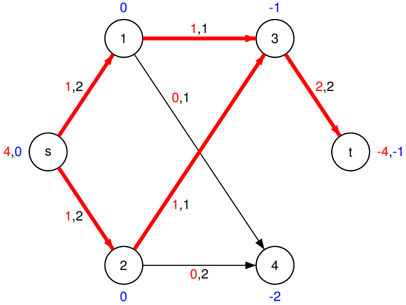
Potentials and reduced costs.
Node labels: e, y.
Arc labels: (c^y, c, u).

The admissible graph $A(x, y)$.
Node labels: e, y.
Arc labels: u.

An example

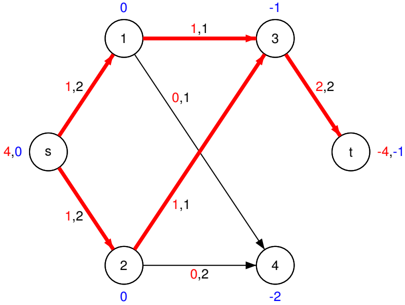


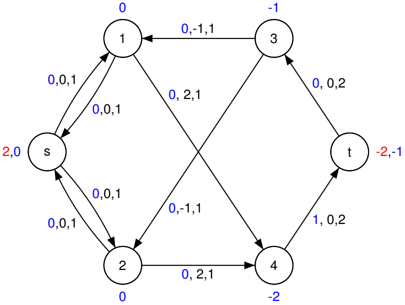
The admissible graph $A(x, y)$.
Node labels: e, y.
Arc labels: u.



Primal iteration 1.
A max flow on $A(x, y)$.
Node labels: e, y.
Arc labels: (x, u).

An example

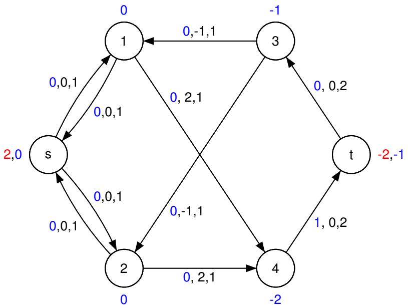




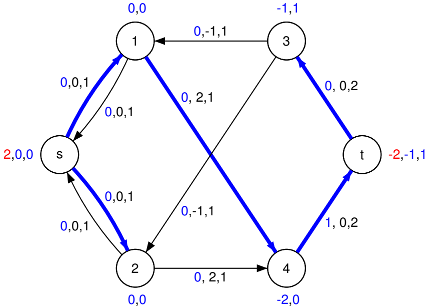
Primal iteration 1.
A max flow on $A(x, y)$.
Node labels: e, y.
Arc labels: (x, u).

The updated residual graph.
Node labels: e, y.
Arc labels: (c^y, c, u).

An example

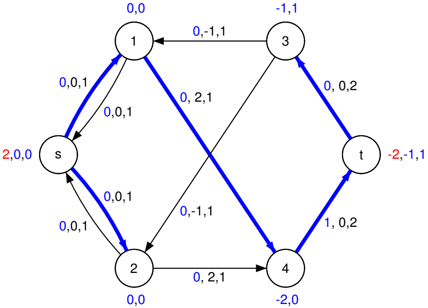


The updated residual graph.
Node labels: e, y.
Arc labels: (c^y, c, u).

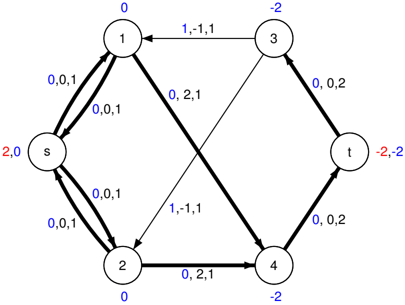


Dual iteration 2.
Shortest paths on $R(x)$.
Node labels: e, y, d.
Arc labels: (c^y, c, u).

An example

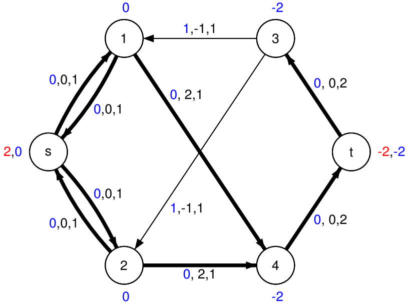


Dual iteration 2.
Shortest paths on $R(x)$.
Node labels: e, y, d.
Arc labels: (c^y, c, u).

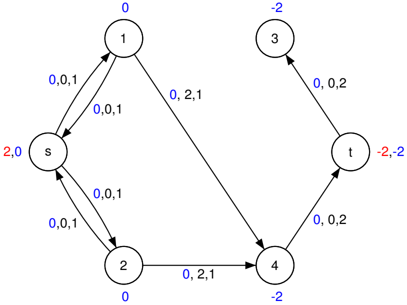


Updated potentials and reduced costs.
Node labels: e, y.
Arc labels: (c^y, c, u).

An example

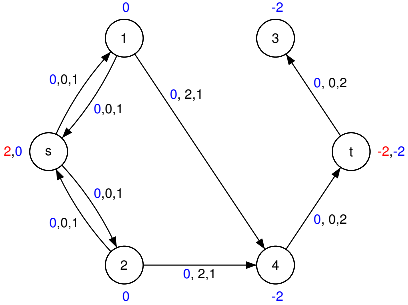


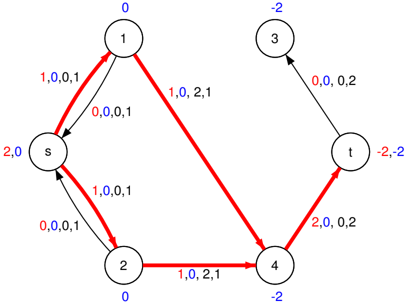
Updated potentials and reduced costs.
Node labels: e, y.
Arc labels: (c^y, c, u).



The admissible graph $A(x, y)$.
Node labels: e, y.
Arc labels: (c^y, c, u).

An example





The admissible graph $A(x, y)$.
Node labels: e, y.
Arc labels: (c^y, c, u).

Primal iteration 2.
A max flow on $A(x, y)$.
Node labels: e, y.
Arc labels: (x, c^y, c, u).

Complexity

The algorithm guarantees that at each iteration

- the excess of node s decreases by at least 1 unit.
  Proof: a strictly positive amount of flow is sent from s to t.
- the potential of node t decreases by at least 1 unit.
  Proof: no more (s, t)-paths of zero reduced cost can exist in the residual graph.

Initially $e(s) \leq nB$ and at the end $e(s) = 0$.
Initially $y(t) = 0$ and at the end $y(t) \geq -nC$.

Therefore, the number of iterations is bounded by $O(\min\{nB, nC\})$.

This bound must be multiplied by the complexity for solving a shortest path problem and a max flow problem at each iteration.

The out-of-kilter algorithm

The out-of-kilter algorithm is a primal-dual algorithm in which

- flow balance constraints are kept satisfied, while flow bounds constraints can be violated;
- flows and potentials are iteratively modified to move the solution towards feasiblity and optimality.

Since flow bounds constraints can be violated before the algorithm stops, the out-of-kilter algorithm can be used to solve the min cost flow problem when lower bounds are imposed on arc flows.

Circulation problem

A circulation problem is a special case of the min cost flow problem, in which $b_i = 0 \ \forall i \in \mathcal{N}$.

The flow is forced to be non-zero, although costs are positive, by the lower bounds.

Every min cost flow problem instance can be reformulated as an equivalent circulation problem instance:

- add a node s and arcs $(s, i) \ \forall i \in \mathcal{N} : b_i > 0$, with $l_{si} = u_{si} = b_i$ and $c_{si} = 0$;
- add a node t and arcs $(j, t) \ \forall j \in \mathcal{N} : b_j < 0$, with $l_{jt} = u_{jt} = b_j$ and $c_{jt} = 0$;
- add an arc (t, s) with $l_{ts} = u_{ts} = B$ and $c_{ts} = 0$,

where $B = \sum_{i \in \mathcal{N} : b_i > 0} b_i = -\sum_{i \in \mathcal{N} : b_i < 0} b_i$.
Now, setting all b to zero a circulation problem instance is obtained.

Primal and dual problems

$$\text{minimize } z = \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij}$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{j\in\mathcal{N}:(i,j)\in\mathcal{A}} x_{ij} - \sum_{j\in\mathcal{N}:(j,i)\in\mathcal{A}} x_{ji} = 0 && \forall i \in \mathcal{N} \\
& x_{ij} \geq l_{ij} && \forall (i,j) \in \mathcal{A} \\
& -x_{ij} \geq -u_{ij} && \forall (i,j) \in \mathcal{A} \\
& (x_{ij} \text{ integer}) && \forall (i,j) \in \mathcal{A}.
\end{aligned}$$

$$\text{maximize } w = \sum_{(i,j)\in\mathcal{A}} l_{ij}\mu_{ij} - \sum_{(i,j)\in\mathcal{A}} u_{ij}\lambda_{ij}$$

$$\begin{aligned}
\text{s.t.} \quad & y_i - y_j + \mu_{ij} - \lambda_{ij} \leq c_{ij} && \forall (i,j) \in \mathcal{A} \\
& y_i \text{ free} && \forall i \in \mathcal{N} \\
& \lambda_{ij} \geq 0 && \forall (i,j) \in \mathcal{A} \\
& \mu_{ij} \geq 0 && \forall (i,j) \in \mathcal{A}.
\end{aligned}$$

Complementary slackness conditions

$$
\begin{aligned}
\text{maximize } w = \sum_{(i,j)\in\mathcal{A}} l_{ij}\mu_{ij} - \sum_{(i,j)\in\mathcal{A}} u_{ij}\lambda_{ij} & \\
\text{s.t. } y_i - y_j + \mu_{ij} - \lambda_{ij} \leq c_{ij} & \qquad \forall (i,j) \in \mathcal{A} \\
y_i \text{ free} & \qquad \forall i \in \mathcal{N} \\
\lambda_{ij} \geq 0 & \qquad \forall (i,j) \in \mathcal{A} \\
\mu_{ij} \geq 0 & \qquad \forall (i,j) \in \mathcal{A}.
\end{aligned}
$$

Defining the reduced costs $c^y_{ij} = c_{ij} - y_i + y_j$ for any given vector y, dual optimality requires $\mu_{ij} - \lambda_{ij} = c^y_{ij}$ for each arc, because $u_{ij} \geq l_{ij}$.

Therefore

- $\mu_{ij} = \max\{0, c^y_{ij}\}$: if $c^y_{ij} > 0$, then $\mu_{ij} > 0$;
- $\lambda_{ij} = \max\{0, -c^y_{ij}\}$: if $c^y_{ij} < 0$, then $\lambda_{ij} > 0$.

Complementary slackness conditions

Primal C.S.C.

$$x_{ij}(c_{ij} + y_j - y_i - \mu_{ij} + \lambda_{ij}) = 0 \quad \forall (i,j) \in \mathcal{A}.$$

Dual C.S.C.

$$\lambda_{ij}(u_{ij} - x_{ij}) = 0 \quad \forall (i,j) \in \mathcal{A}$$

$$\mu_{ij}(x_{ij} - l_{ij}) = 0 \quad \forall (i,j) \in \mathcal{A}.$$

Therefore

$$x_{ij} = l_{ij} \Rightarrow c^y_{ij} \geq 0$$

$$l_{ij} < x_{ij} < u_{ij} \Rightarrow c^y_{ij} = 0$$

$$x_{ij} = u_{ij} \Rightarrow c^y_{ij} \leq 0.$$

The restricted residual graph

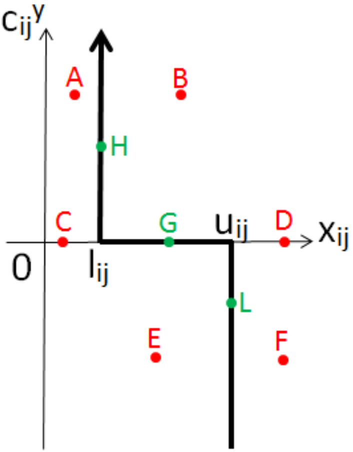
The out-of-kilter algorithm works on a restricted residual graph, because

- not all arcs with residual capacity are allowed to carry additional flow;
- the residual capacity of an arc (i, j) does not depend only on x_{ij}, u_{ij} and l_{ij}, but also on c_{ij}^{y}.

Only **admissible arcs** are allowed to receive additional flow.
Only admissible arcs are included in $R(x, y)$, which then depends both on x and y.

To measure how far the pair of solutions (x, y) is from optimality, a **kilter number** is defined for each arc $(i, j) \in \mathcal{A}$.

Kilter numbers and residual capacities

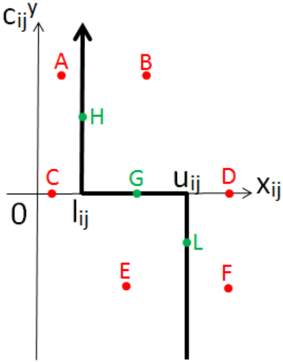


The kilter diagram for $(i,j) \in \mathcal{A}$.

Kilter numbers for each original arc.
Residual capacities for each arc in $R(x, y)$.

A : $k_{ij} = l_{ij} - x_{ij}$	$r_{ij} = l_{ij} - x_{ij}$
B : $k_{ij} = x_{ij} - l_{ij}$	$r_{ji} = x_{ij} - l_{ij}$
C : $k_{ij} = l_{ij} - x_{ij}$	$r_{ij} = u_{ij} - x_{ij}$
D : $k_{ij} = x_{ij} - u_{ij}$	$r_{ji} = x_{ij} - l_{ij}$
E : $k_{ij} = u_{ij} - x_{ij}$	$r_{ij} = u_{ij} - x_{ij}$
F : $k_{ij} = x_{ij} - u_{ij}$	$r_{ji} = x_{ij} - u_{ij}$
G : $k_{ij} = 0$	$r_{ij} = u_{ij} - x_{ij}$
	$r_{ji} = x_{ij} - l_{ij}$
H : $k_{ij} = 0$	$r_{ij} = 0$
L : $k_{ij} = 0$	$r_{ji} = 0$

Admissible arcs



The kilter diagram.

$R(x, y)$ includes only admissible arcs:

- arcs $(i, j) \in R(x, y)$ corresponding to arcs $(i, j) \in \mathcal{A}$ of type A, C and E (x_{ij} can be increased);
- arcs $(j, i) \in R(x, y)$ corresponding to arcs $(i, j) \in \mathcal{A}$ of type B, D and F (x_{ij} can be decreased);
- arcs (i, j) and $(j, i) \in R(x, y)$ corresponding to arcs $(i, j) \in \mathcal{A}$ of type G (x_{ij} can be increased/decreased);

In-kilter arcs of type H and L are not admissible.

The algorithm

Primal initialization. The flow starts from 0 on all arcs.

Dual initialization. The potential starts from 0 on all nodes.

Primal iteration. A maximum flow is sent along a circuit in a restricted residual graph $R(x)$, including only admissible arcs.
The circuit must include at least one out-of-kilter arc.

Dual iteration. A shortest path is computed to modify the potentials and the restricted residual graph.

Primal iteration

An out-of-kilter arc $(i, j) \in \mathcal{A}$ is selected.

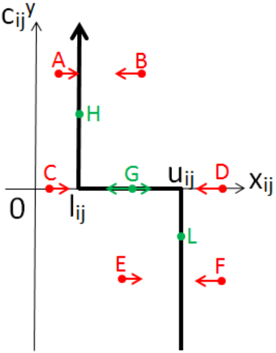
The corresponding arc $(p, q) \in R(x, y)$ is considered:

- $R(x, y)$ includes arc (i, j) if x_{ij} is of type A, C or E ($(p, q) = (i, j)$);
- $R(x, y)$ includes arc (j, i) if x_{ij} is of type B, D or F ($(p, q) = (j, i)$).

A path $P(q, p)$ from q to p in $R(x, y)$ is searched by labelling nodes from q.

Different labelling strategies can be used to select $P(q, p)$.

Primal iteration



The effects of a primal iteration.

$P(q, p)$ can only use admissible arcs in $R(x, y)$.

Sending flow along admissible arcs can only decrease their kilter number.

When a (q, p)-path is found (breakthrough), the maximum residual capacity along the circuit $P(q, p) \cup (p, q)$ determines the amount of flow.
$R(x, y)$ is updated.
If out-of-kilter arcs still exist, a new primal iteration is started.

Primal iteration

If no (q, p)-path exists in $R(x, y)$, a dual iteration is executed.

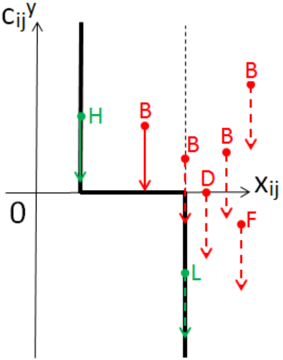
Let Q be the set of nodes reachable from q in $R(x, y)$ and $\overline{Q}$ be its complement.

There are no arcs with positive residual capacity in $R(x, y)$ across the $(Q, \overline{Q})$ cut.
Therefore there are no out-of-kilter arcs (i, j) such that

- (i, j) is of type A, C or E ($(i, j) \in R(x, y)$) and $i \in Q$ and $j \in \overline{Q}$;
- (i, j) is of type B, D or F ($(j, i) \in R(x, y)$) and $j \in Q$ and $i \in \overline{Q}$.

There are no in-kilter arcs (i, j) of type G with an endpoint in Q and the other in $\overline{Q}$.

Dual iteration



The effects of a dual iteration (direct arcs from Q to $\overline{Q}$).

$R(x, y)$ does not include any arc $(i, j) \in (Q, \overline{Q})$ such that

- (i, j) is of type A, C or E;
- (i, j) is of type G and $x_{ij} < u_{ij}$.

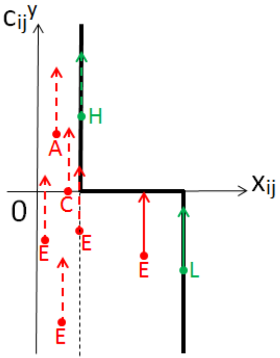
Consider the set Fw of forward arcs in $\mathcal{A}$ across the $(Q, \overline{Q})$ cut:

$$Fw = \{(i, j) \in \mathcal{A} : i \in Q, j \in \overline{Q}, c_{ij}^y > 0, x_{ij} < u_{ij}\}.$$

They all correspond to arcs of type B and H.

Compute $\alpha = \min_{(i,j)\in Fw}\{c_{ij}^y\}$.

Dual iteration



The effects of a dual iteration (reverse arcs from Q to $\overline{Q}$).

$R(x, y)$ does not include any arc $(j, i) \in (Q, \overline{Q})$ such that

- (i, j) is of type B, D or F;
- (i, j) is of type G and $x_{ij} > l_{ij}$.

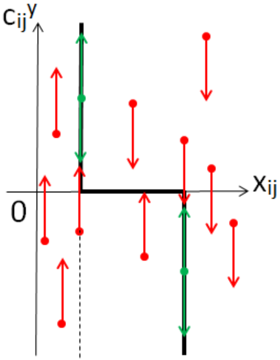
Consider the set Bw of backward arcs in $\mathcal{A}$ across the $(Q, \overline{Q})$ cut:

$$Bw = \{(i, j) \in \mathcal{A} : j \in Q, i \in \overline{Q}, c_{ij}^y < 0, x_{ij} > l_{ij}\}.$$

They all correspond to arcs of type E and L.

Compute $\beta = \max_{(i,j) \in Bw}\{c_{ij}^y\}$.

Dual iteration



The effects of a dual iteration.

Define $\theta = \min\{\alpha, -\beta\}$.

Update $y_i \leftarrow y_i + \theta \; \forall i \in Q$.

For all arcs across the $(Q, \overline{Q})$ cut:

- positive reduced costs are reduced by θ,
- negative reduced costs are increased by θ

and kilter numbers can only decrease.

The other arcs are unaffected.

At least one more arc gets zero reduced cost and becomes admissible (type G).

Update $R(x, y)$ and resume the primal iteration.

Pseudo-code

$x \leftarrow 0$; $y \leftarrow 0$
for all $(i,j) \in \mathcal{A}$ **do**
  Compute k_{ij}; Compute r_{ij} or r_{ji} in $R(x,y)$
while $\exists (i,j) \in \mathcal{A} : k_{ij} > 0$ **do**
  if $x_{ij} < l_{ij} \vee (c^y_{ij} < 0 \wedge x_{ij} < u_{ij})$ **then**
    $(p,q) \leftarrow (i,j)$
  else
    $(p,q) \leftarrow (j,i)$
  $label(i) \leftarrow null \ \forall i \in \mathcal{N}$
  repeat
    $PropagateLabels(q)$
    if $label(p) = null$ **then**
      Dual Iteration
  until $label(p) \neq null$
  Reconstruct the $(q-p)$-path P; $C \leftarrow P \cup (p,q)$
  $\delta \leftarrow \min_{(u,v)\in C}\{r_{uv}\}$
  Send δ units of flow along C and update x, k and $R(x,y)$

Correctness and complexity

Correctness.
Lemma 1. Updating the node potentials y does not increase the kilter number of any arc.

Lemma 2. Sending flow along C does not increase the kilter number of any arc.

Complexity.
Initially the kilter number of any arc is bounded by U.
Hence the sum of all kilter numbers is at most mU.

The sum of the kilter numbers decreases by at least 1 unit for each primal or dual iteration.

Therefore the algorithm requires $O(mU)$ primal or dual iterations.