Min cost flows

Combinatorial Optimization

Giovanni Righini

Università degli Studi di Milano

Definitions

A flow network is a digraph $\mathcal{D} = (\mathcal{N}, \mathcal{A})$ with two particular nodes s and t acting as *source* and *sink* of a flow.

The flow is a quantity that can traverse the arcs from their tails to their heads, starting from s and reaching t.

The digraph $\mathcal{D}$ is weighted with

- a capacity $u : \mathcal{A} \mapsto \Re^m_+$;
- a cost $c : \mathcal{A} \mapsto \Re^m_+$;

Arc capacity: limit to the amount of flow that can traverse the arc.
Arc cost: cost to be paid for each unit of flow traversing the arc.

- An arc with no flow is *empty*.
- An arc with a flow equal to its capacity is *saturated*.

A formulation

We use a continuous and non-negative variable x_{ij} to indicate the amount of flow on each arc $(i,j) \in \mathcal{A}$.

A mathematical model of the min-cost flow problem is:

$$\begin{aligned} \text{minimize } z = & \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{j\in\mathcal{N}:(i,j)\in\mathcal{A}} x_{ij} - \sum_{j\in\mathcal{N}:(j,i)\in\mathcal{A}} x_{ji} = b_i \qquad \forall i \in N \\ & 0 \le x_{ij} \le u_{ij} \qquad \forall (i,j) \in \mathcal{A}. \end{aligned}$$

We assume that:

- all data are integer;
- $\sum_{i\in\mathcal{N}} b_i = 0$;
- capacities and costs are non-negative.

Optimality conditions

A feasible solution x^* is optimal if and only if

1. the residual digraph $R(x)$ does not contain any negative cost cycle;
2. there is a dual vector y such that the reduced cost $\overline{c}_{ij} = c_{ij} - y_i + y_j \geq 0$ for all arcs in the residual digraph $R(x)$;
3. complementary slackness conditions hold.

All these conditions are equivalent.

Flow decomposition

The difference between two feasible flows of the same value, is a set of directed cycles.

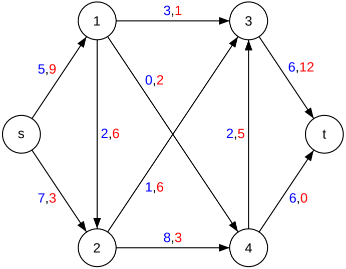


Figure: Two feasible flows, x_1 and x_2.

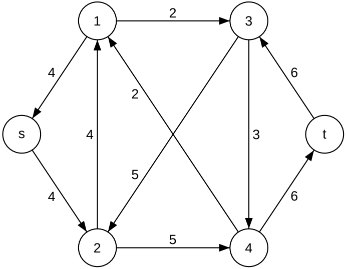


Figure: The difference $x_1 - x_2$.

Flow decomposition

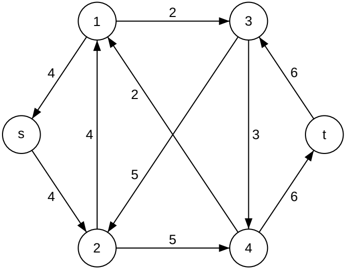


Figure: The difference $x_1 - x_2$.

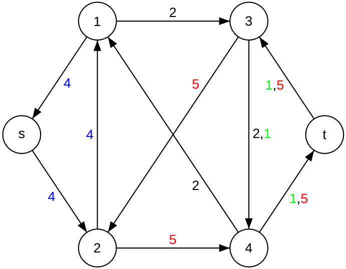


Figure: Decomposition in 4 directed cycles.

Negative cycles optimality conditions

Theorem. A feasible flow x is optimal for the min cost flow problem, if and only if the residual graph $R(x)$ does not contain any negative cost cycle.

Proof (1): x optimal $\Rightarrow$ No negative cycles in $R(x)$.

By construction of the residual digraph, any directed cycle in $R(x)$ is an augmenting cycle for x.

Then, sending a unit of flow along a negative cost cycle decreases the cost, without violating any constraint.

Therefore, if $R(x)$ contains a negative cost cycle, x cannot be optimal.

Negative cycles optimality conditions

Proof (2): No negative cycles in $R(x) \Rightarrow x$ optimal.

Assume that x^* is feasible, x^o is optimal (i.e. a min cost flow) with $x^o \neq x^*$ and $R(x^*)$ has no negative cost cycles.
The difference vector $x^o - x^*$ can be decomposed into a set of augmenting cycles with respect to x^* on $R(x^*)$ and the sum of the costs of the flows along them is equal to $cx^o - cx^*$.

Since there are no negative cost cycles, $cx^o - cx^* \geq 0$ for each augmenting cycle: hence $cx^o \geq cx^*$.

Since x^o is a min cost flow, then $cx^o \leq cx^*$.

Therefore $cx^o = cx^*$ and x^* is also optimal.

Reduced cost optimality conditions

Theorem. A feasible flow x is optimal for the min cost flow problem, if and only if there exists a vector of node potentials y satisfying the condition

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

Proof (1): $\exists y : c_{ij}^y \geq 0 \ \forall (i,j) \in R(x) \Rightarrow x$ optimal.

If $c_{ij}^y \geq 0 \ \forall (i,j) \in R(x)$, then $\sum_{(i,j)\in W} c_{ij}^y \geq 0$ for any cycle W in $R(x)$.

For every cycle W, $\sum_{(i,j)\in W} c_{ij}^y = \sum_{(i,j)\in W} c_{ij}$, because potentials cancel out along the cycle.

Therefore for every cycle W in $R(x)$, $\sum_{(i,j)\in W} c_{ij} \geq 0$, i.e. $R(x)$ does not contain any negative cost cycle. Therefore x is optimal.

Reduced cost optimality conditions

Proof (2): x optimal $\Rightarrow \exists y : c^y_{ij} \geq 0 \ \forall (i,j) \in R(x)$.

If x is optimal, then $R(x)$ has no negative cost cycles.
Consider a feasible flow x^* such that $R(x^*)$ has no negative cost cycles.
Then the shortest path problem is well-defined on $R(x^*)$.

Compute min cost paths from s to all nodes in $R(x^*)$: let d_i be the resulting min cost $\forall i \in N$.

From optimality conditions for shortest paths

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

Now choosing $y = -d$, we obtain

$$c_{ij} - y_i + y_j \geq 0 \ \ \forall (i,j) \in R(x^*).$$

The dual problem

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

$$\text{s.t.} \sum_{j\in\mathcal{N}:(i,j)\in\mathcal{A}} x_{ij} - \sum_{j\in\mathcal{N}:(j,i)\in\mathcal{A}} x_{ji} = b_i \qquad \forall i \in \mathcal{N}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \forall (i,j) \in \mathcal{A}.$$

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

$$\text{s.t. } y_i - y_j - \lambda_{ij} \leq c_{ij} \qquad \forall (i,j) \in \mathcal{A}$$

$$y_i \text{ free} \qquad \forall i \in \mathcal{N}$$

$$\lambda_{ij} \geq 0 \qquad \forall (i,j) \in \mathcal{A}.$$

Integer capacities $\Rightarrow$ integer optimal solution.

Complementary slackness conditions

Primal C.S.C.

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

Dual C.S.C.

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

While the previous optimality conditions are formulated on the residual digraph, the c.s. optimality conditions are formulated on the original digraph.

Complementary slackness optimality conditions

Theorem. A feasible flow x is optimal for the min cost flow problem, if and only if for some node potential y, the reduced costs $\overline{c}$ and the flow values x satisfy the following c.s.c. for each arc $(i,j) \in A$:

- if $\overline{c}_{ij} > 0$ then $x_{ij} = 0$;
- if $0 < x_{ij} < u_{ij}$ then $\overline{c}_{ij} = 0$;
- if $\overline{c}_{ij} < 0$ then $x_{ij} = u_{ij}$.

Proof. From linear programming duality.

This is a notable case of LP with bounded variables: flow variables x can be non-basic in two different ways: either because they are at their lower bound (0) or because they are at their upper bound (u).

Optimal flows and optimal potentials

Question 1. Given an optimal flow x^*, how can we obtain optimal node potentials y^*?

Question 2. Given optimal node potentials y^*, how can we obtain an optimal flow x^*?

Answer 1. By computing a shortest path.

Answer 2. By computing a maximum flow.

From x^* to y^*

Let $R(x^*)$ be the residual graph corresponding to an optimal flow x^*. Since x^* is optimal, $R(x^*)$ does not contain any negative cost cycle.

Let d be the vector of shortest distances from node s to all the other nodes, using c as arc lengths.

Shortest path optimality conditions imply

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

Let $y_i = -d_i \;\; \forall i \in \mathcal{N}$. Then

$$c_{ij} - y_i + y_j \geq 0 \;\; \forall (i,j) \in R(x^*).$$

Then y is an optimal vector of node potentials.

From y^* to x^*

Let y^* be an optimal vector of node potentials.
We can compute the corresponding reduced costs:

$$\overline{c}_{ij} = c_{ij} - y_i + y_j \quad \forall (i,j) \in \mathcal{A}.$$

We examine each arc $(i,j) \in \mathcal{A}$:

- if $\overline{c}_{ij} > 0$, then $x^*_{ij} = 0$: delete (i,j).
- if $\overline{c}_{ij} < 0$, then $x^*_{ij} = u_{ij}$: set $b_i := b_i - u_{ij}$; $b_j := b_j + u_{ij}$; delete (i,j).
- if $\overline{c}_{ij} = 0$, then we have the constraint $0 \leq x^*_{ij} \leq u_{ij}$.

Insert a dummy source s' and a dummy sink t'.
Insert an arc (s', i) for each $i \in \mathcal{N}$ with $b'_i > 0$.
Insert an arc (i, t') for each $i \in \mathcal{N}$ with $b'_i < 0$.
Send a maximum flow x^* from s' to t'.

Algorithms

Algorithms for the min-cost flow problem can be roughly classified according to the optimality conditions they exploit.

1. Cycle-canceling algorithms find a maximum flow first and then iteratively improve its cost by detecting negative cost cycles.
2. Successive shortest path algorithms iteratively increase a min-cost flow by detecting minimum cost augmenting paths.
3. Primal-dual algorithms send an augmenting flow at each iteration instead of using a single augmenting path.
4. Out-of-kilter algorithm.

Cycle-canceling algorithms

Algorithm 1 Cycle-canceling algorithm

Compute a max flow x and the corresponding residual graph $R(x)$;
while $R(x)$ contains a negative cost cycle **do**
  Select a negative cost cycle W;
  $\delta \leftarrow \min_{(i,j) \in W} \{r_{ij}\}$;
  Send δ units of flow along W and update $R(x)$;

Cycle-canceling algorithms: complexity

Let define

- $C = \max_{(i,j) \in A} \{c_{ij}\}$;
- $U = \max_{(i,j) \in A} \{u_{ij}\}$;

Then mCU is a trivial upper bound on the cost of the initial maximum flow.

Then the algorithm terminates in at most mCU iterations, since $\delta \geq 1$ at each iteration.

If negative cost cycles are identified in $O(nm)$ (with Moore algorithm with FIFO policy), the overall complexity is $O(nm^2CU)$, which is not polynomial.

Polynomial-time implementations

Two possible polynomial-time implementations of the generic cycle-canceling algorithm select

- a negative cost cycle with maximum residual capacity: $O(m \log (mCU))$
- a negative cost cycle with minimum mean cost: $O(\min\{nm \log (nC), nm^2 \log n\})$.

Both of them yield algorithms with polynomial-time complexity.

Cycle with maximum residual capacity

Any two feasible flows on a given network can be obtained from each other by at most m augmenting cycles in the residual graph.

Let x be a feasible flow and x^* an optimal flow.
Then the cost cx^* equals cx plus the (negative) cost of at most m cycles in $R(x)$.
The improvement in cost is $cx - cx^*$.

Consequently, at least one of the augmenting cycles must produce a decrease of at least $(cx - cx^*)/m$.
Then, by selecting the cycle yielding maximum improvement, the algorithm requires $O(m \log(mCU))$ iterations.

Unfortunately, finding the maximum improvement cycle is an NP-hard problem.
However a slight modification of this approach yields an overall polynomial-time complexity.

Cycle with minimum mean cost

The mean cost of a cycle is its cost divided by the number of arcs it contains.

A cycle with minimum mean cost can be identified in $O(nm)$ or $O(\sqrt{n}m \log(nC))$.

If the cycle canceling algorithm always selects a minimum mean cost cycle, it requires $O(\min\{nm \log(nC), nm^2 \log n\})$ iterations.

Therefore it is strongly polynomial.

A basic property

Basic property. Given any flow x and its corresponding residual graph $R(x)$, for each cycle W in $R(x)$ and for each choice of the node potentials y,

$$\sum_{(i,j)\in W} c_{ij} = \sum_{(i,j)\in W} c^y_{ij}$$

where $c^y_{ij} = c_{ij} - y_j + y_i \ \ \forall (i,j) \in R(x)$, because the potentials cancel out along the cycle.

ϵ-optimality

Definition. A flow x is ϵ-optimal if $\exists y : c_{ij}^y \geq -\epsilon \ \forall (i,j) \in R(x)$.

Given a vector of potentials y, let define

$$\epsilon^y(x) = -\min_{(i,j) \in R(x)} \{c_{ij}^y\}.$$

Then

$$\begin{cases} c_{ij}^y \geq -\epsilon^y(x) \ \forall (i,j) \in R(x) \\ \exists (u,v) \in R(x) : c_{uv}^y = -\epsilon^y(x) \end{cases}$$

Therefore x is ϵ-optimal for $\epsilon = \epsilon^y(x)$.

For different choices of y, we can have different values for $\epsilon^y(x)$.
Let $\epsilon(x)$ be the minimum value of $\epsilon^y(x)$ for which x is $\epsilon^y(x)$-optimal:

$$\epsilon(x) = \min_y \{\epsilon^y(x)\}.$$

Reduced costs along cycles

Let $\mu(x)$ be the mean cost of the minimum mean cost cycle in $R(x)$.

If x is ϵ-optimal, then for each cycle W of $R(x)$ and for each vector of potentials y

$$\sum_{(i,j)\in W} c_{ij} = \sum_{(i,j)\in W} c^y_{ij} \geq -\epsilon^y(x)|W|.$$

If W^* is the minimum mean cost cycle in $R(x)$, then

$$\mu(x) \geq -\epsilon^y(x)$$

and

$$\exists y : c^y_{ij} = -\epsilon(x) \;\; \forall (i,j) \in W^*.$$

Lemma 1: relationship between $\mu(x)$ and $\epsilon(x)$

Lemma 1. Consider a sub-optimal flow $x \neq x^*$. Then $\epsilon(x) = -\mu(x)$.

Proof. Let modify the costs c into c' as follows:

$$c'_{ij} = c_{ij} - \mu(x) \;\; \forall (i,j) \in A.$$

The resulting digraph $R'(x)$ has the same arcs as $R(x)$.

The cost modification reduces the mean cost of all cycles by $\mu(x)$ (which is negative).

The mean cost of W^* is zero in $R'(x)$.

Therefore $R'(x)$ does not contain cycles with negative cost.

Lemma 1: relationship between $\mu(x)$ and $\epsilon(x)$

Select a node $s \in N$ and consider the shortest paths arborescence from s in $R'(x)$.

Let d' be the shortest distances.

$$d'_j \leq d'_i + c'_{ij} = d'_i + c_{ij} - \mu(x) \;\; \forall (i,j) \in R'(x).$$

Setting $y_j = d'_j \;\; \forall j \in N$ we have

$$y_j \leq y_i + c_{ij} - \mu(x) \;\; \forall (i,j) \in R(x)$$

$$c^y_{ij} \geq \mu(x) \;\; \forall (i,j) \in R(x)$$

Therefore x is $(-\mu(x))$-optimal.

Since $\mu(x)$ does not depend on y, then $\epsilon(x) = -\mu(x)$.

Lemma 2: relationship between c^y and $\mu(x)$ and $\epsilon(x)$

Lemma 2. Consider a sub-optimal flow $x \neq x^*$. Then
$\exists y : c^y_{ij} = -\epsilon(x) = \mu(x) \;\; \forall (i,j) \in W^*$.

Proof. Selecting y as before, $c^y_{ij} \geq \mu(x) \;\; \forall (i,j) \in R(x)$.
By definition

$$c(W^*) = \sum_{(i,j)\in W^*} c_{ij} = \sum_{(i,j)\in W^*} c^y_{ij} = \mu(x)|W^*|.$$

So, the mean value of c^y_{ij} along W^* is $\mu(x)$ and all values of c^y_{ij} are at least $\mu(x)$. Therefore

$$c^y_{ij} = \mu(x) \quad \forall (i,j) \in W^*$$

and from Lemma 1

$$c^y_{ij} = -\epsilon(x) \quad \forall (i,j) \in W^*.$$

Lemma 3: monotonicity of $\epsilon(x)$

Lemma 3. Consider a sub-optimal flow $x \neq x^*$. After deleting W^*, $\epsilon(x)$ does not increase and $\mu(x)$ does not decrease.

Proof. Consider a dual vector y such that

$$\begin{cases} c^y_{ij} = -\epsilon(x) & \forall (i,j) \in W^* \\ c^y_{ij} \geq -\epsilon(x) & \forall (i,j) \in R(x) \end{cases}$$

Let x' be the flow and $R'(x')$ the residual graph after the cancellation of W^*.

At least one arc of $R(x)$ does not belong to $R'(x')$ (because it has been saturated).

Lemma 3: monotonicity of $\epsilon(x)$

Some new arcs may appear in $R'(x')$ that were not in $R(x)$.
For all $(i,j) \in R'(x')$:

$$\begin{cases} \text{if } (i,j) \in R(x) & c^y_{ij} \geq -\epsilon(x) \\ \text{if } (i,j) \notin R(x) & c^y_{ji} = -\epsilon(x)((j,i) \in W^*) \end{cases}$$

In the latter case $c^y_{ij} = -c^y_{ji} = \epsilon(x) > 0 > -\epsilon(x)$.

Lemma 3: monotonicity of $\epsilon(x)$

Therefore, in both cases

$$c_{ij}^y \geq -\epsilon(x) \;\; \forall (i,j) \in R'(x').$$

Then x' is still $\epsilon(x)$-optimal: $\epsilon(x') \leq \epsilon(x)$.

$$\mu(x') = \sum_{(i,j) \in W^{*\prime}} \frac{c_{ij}}{|W^{*\prime}|} = \sum_{(i,j) \in W^{*\prime}} \frac{c_{ij}^y}{|W^{*\prime}|} \geq \min_{(i,j) \in W^{*\prime}} \{c_{ij}^y\} \geq -\epsilon(x) = \mu(x).$$

Therefore $\mu(x') \geq \mu(x)$.

Lemma 4: decrease rate of $\epsilon(x)$

Lemma 4. Within at most m iterations, ϵ decreases by a factor at least $(1 - \frac{1}{n})$.

Proof. We have already proven that

$$\exists y : c_{ij}^{y} \geq -\epsilon(x) \;\; \forall (i,j) \in R(x).$$

Type-1 iterations: $c_{ij}^{y} < 0 \;\; \forall (i,j) \in W^*$
Type-2 iterations: otherwise.

Every type-1 iteration deletes an arc with negative reduced cost from the residual graph.

All arcs inserted by type-1 iterations have positive reduced cost.

Therefore the algorithm can execute at most m consecutive type-1 iterations.

Lemma 4: decrease rate of $\epsilon(x)$

When a type-2 iteration is done, the eliminated cycle W^* contains at least one arc with non-negative reduced cost.
Therefore it contains at most $|W^*| - 1$ arcs with negative reduced cost.
Let x' and x'' be the flows before and after the iteration.

$$c^y_{ij} \geq -\epsilon(x') \; \forall (i,j) \in W^*$$

$$c(W^*) = \sum_{(i,j) \in W^*} c^y_{ij}$$

$$c(W^*) \geq (|W^*| - 1)(-\epsilon(x'))$$

$$\mu(x') = c(W^*)/|W^*|$$

Then

$$\mu(x') \geq \frac{|W^*| - 1}{|W^*|}(-\epsilon(x')).$$

Lemma 4: decrease rate of $\epsilon(x)$

$$\mu(x') \geq \frac{|W^*| - 1}{|W^*|}(-\epsilon(x')).$$

From Lemma 3, $\mu(x'') \geq \mu(x')$.

Then

$$-\epsilon(x'') = \mu(x'') \geq \mu(x') \geq \left(1 - \frac{1}{|W^*|}\right)(-\epsilon(x')) \geq \left(1 - \frac{1}{n}\right)(-\epsilon(x')).$$

Therefore

$$\epsilon(x'') \leq \left(1 - \frac{1}{n}\right)\epsilon(x').$$

Lemma 5: stop criterion

Lemma 5. If $\epsilon < \frac{1}{n}$, every ϵ-optimal flow is also optimal.

Proof. If x is ϵ-optimal, then a dual vector y exists such that $c^y_{ij} \geq -\epsilon$ for all arcs in $R(x)$.

Let W be a cycle in $R(x)$. Then

$$c(W) = \sum_{(i,j) \in W} c^y_{ij} \geq -\epsilon |W| \geq -\epsilon n > -1.$$

Since $c(W)$ is integer, $c(W) > -1$ implies $c(W) \geq 0$.

Then $R(x)$ contains no negative cost cycle, and x is optimal.

Lemma 6: exponential decrease rate

Lemma 6. Consider an integer $\alpha > 1$ and a series of real numbers such that $z_{k+1} \leq (1 - \frac{1}{\alpha})z_k$ for each k. Then $z_{k+\alpha} \leq \frac{1}{2}z_k$ for any k.

Proof. From $z_{k+1} \leq (1 - \frac{1}{\alpha})z_k$ we obtain

$$z_k \geq z_{k+1} + \frac{z_{k+1}}{\alpha - 1}.$$

The same holds replacing k with $k+1$:

$$z_{k+1} \geq z_{k+2} + \frac{z_{k+2}}{\alpha - 1}.$$

Combining the two inequalities:

$$z_k \geq z_{k+2} + \frac{z_{k+2}}{\alpha - 1} + \frac{z_{k+1}}{\alpha - 1} > z_{k+2} + 2\frac{z_{k+2}}{\alpha - 1}$$

because $z_{k+2} < z_{k+1}$.

Lemma 6: exponential decrease rate

Repeating the same procedure we get

$$z_k > z_{k+3} + 3\frac{z_{k+3}}{\alpha - 1}$$

$$z_k > z_{k+4} + 4\frac{z_{k+4}}{\alpha - 1}$$

and so on. In general

$$z_k > z_{k+\alpha} + \alpha\frac{z_{k+\alpha}}{\alpha - 1}.$$

This inequality can be rewritten as

$$z_k > z_{k+\alpha}\left(1 + \frac{\alpha}{\alpha - 1}\right) > 2\, z_{k+\alpha}.$$

Proof of complexity

Let C be the maximum cost of an arc in the original digraph.

Initially the trivial bound $\epsilon(x) \leq C$ holds: every flow is C-optimal.

For every m consecutive iterations $\epsilon(x)$ decreases by a factor $(1 - \frac{1}{n})$ at least.

When $\epsilon < \frac{1}{n}$ the algorithm stops.

Therefore ϵ must decrease by a factor of nC in the worst case.

Proof of complexity

Selecting $\alpha = n$ and letting k be the index of type-2 iterations we know that $\epsilon(x)_{k+1} \leq (1 - \frac{1}{n})\epsilon(x)_k$.

For Lemma 6 we have $\epsilon(x)_{k+n} \leq \frac{1}{2}\epsilon(x)_k$.

Using an index h to count *all* iterations, since there can be up to m type-1 iterations for each single type-2 iteration, $\epsilon(x)_{h+mn} \leq \frac{1}{2}\epsilon(x)_h$.

Therefore $\epsilon(x)$ is halved after at most nm iterations.

Hence the number of iterations is bounded by $nm \log_2 (nC)$.

Proof of complexity

Detecting the minimum mean cost cycle requires $O(nm)$.

Therefore the overall worst-case time complexity of the cycle cancelling algorithm is $O(n^2m^2 \log(nC))$.

Strongly polynomial complexity can be also proven (see *Network flows*, chapter 10).

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:

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

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

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

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

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

Therefore

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

i.e.

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

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* $v \in D$
    $(P(u,v), d) \leftarrow ShortestPath(u, v, R(x), c^y)$
    $y \leftarrow y - d$
    $\delta \leftarrow \min\{e_u, -e_v, \min_{(i,j)\in P(u,v)}\{r_{ij}\}\}$
    $(x, R(x), E, D, c^y) \leftarrow Update(u, v, \delta)$

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.

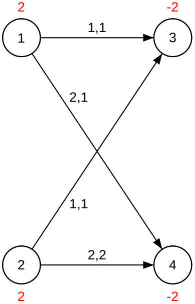
The admissible graph $A(x, y)$ contains only the arcs of the residual graph $R(x)$ that have zero reduced cost c^y.

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

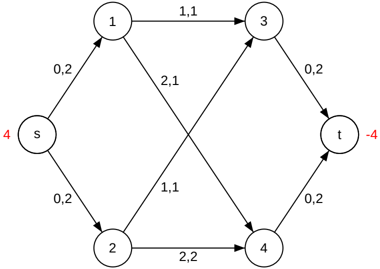
Primal-dual algorithm: pseudo-code

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

An example

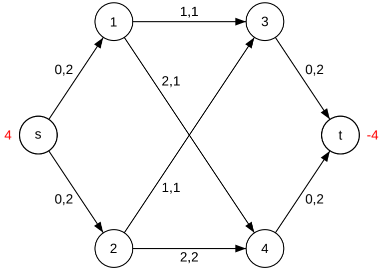


The original network with excess nodes 1 and 2 and deficit nodes 3 and 4.
Node labels: b.
Arc labels: (c, u).

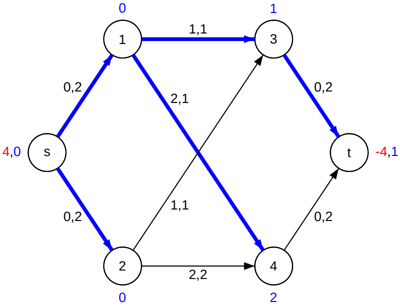


The equivalent flow network.
Node labels: e.
Arc labels: (c, u).

An example

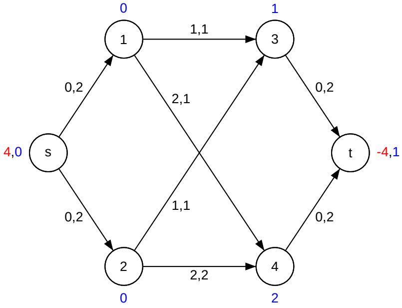


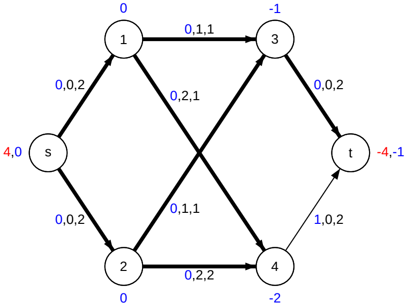
The equivalent flow network.
Node labels: e.
Arc labels: (c, u).



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

An example

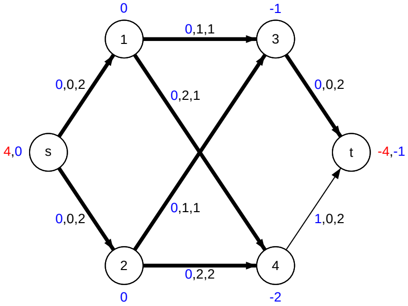


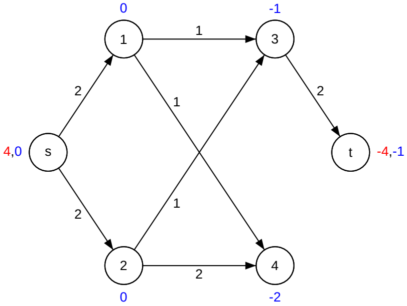


Dual iteration 1 on $R(x)$.
Node labels: e, d.
Arc labels: (c, u).

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

An example

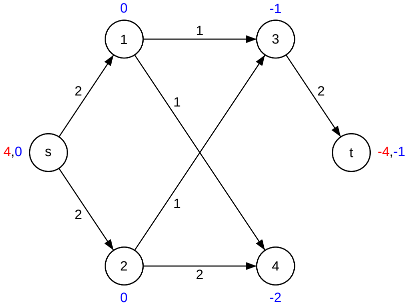


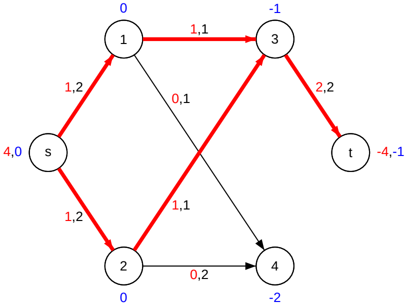


Potentials and reduced costs.
Node labels: e, y.
Arc labels: (c^y, c, u).

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

An example

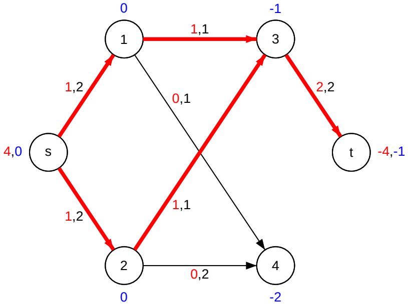




The admissible graph $A(x, y)$.
Node labels: e, y.
Arc labels: u.

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

An example

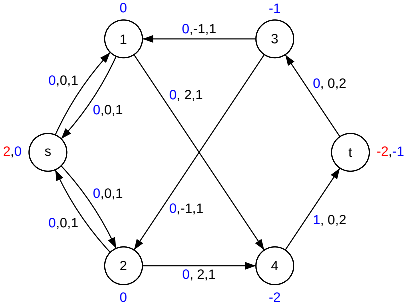


Primal iteration 1.

A max flow on $A(x, y)$.

Node labels: e, y.

Arc labels: (x, u).

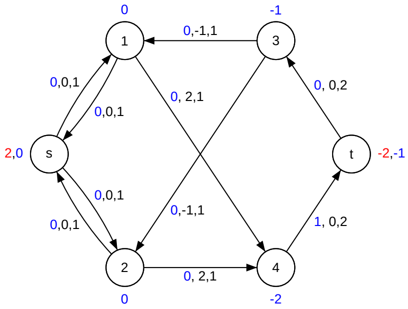


The updated residual graph.

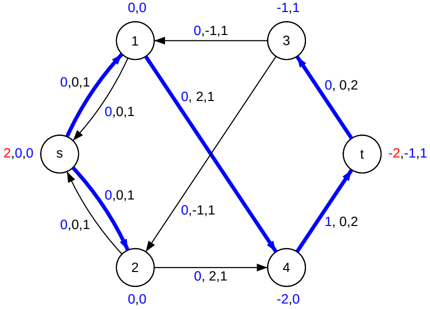
Node labels: e, y.

Arc labels: (c^y, c, u).

An example

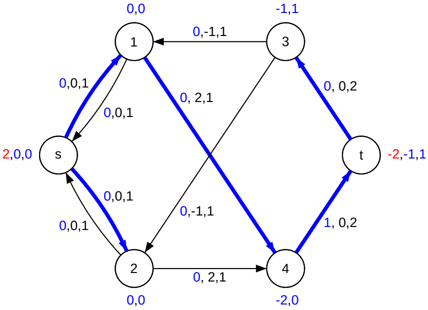


The updated residual graph.
Node labels: e, y.
Arc labels: (c^y, c, u).

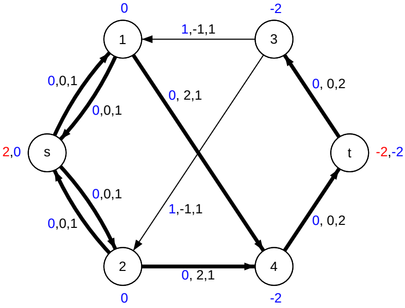


Dual iteration 2.
Shortest paths on $R(x)$.
Node labels: e, y, d.
Arc labels: (c^y, c, u).

An example

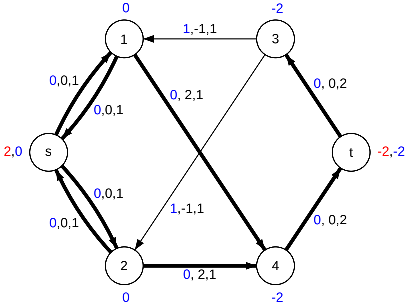


Dual iteration 2.
Shortest paths on $R(x)$.
Node labels: e, y, d.
Arc labels: (c^y, c, u).

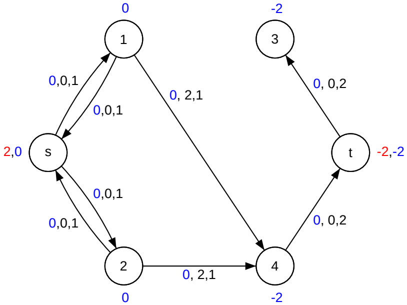


Updated potentials and reduced costs.
Node labels: e, y.
Arc labels: (c^y, c, u).

An example

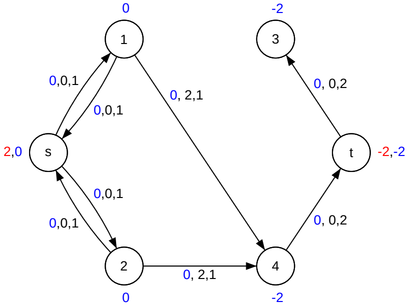


Updated potentials and reduced costs.
Node labels: e, y.
Arc labels: (c^y, c, u).



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

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

$$\text{s.t.} \sum_{j\in\mathcal{N}:(i,j)\in\mathcal{A}} x_{ij} - \sum_{j\in\mathcal{N}:(j,i)\in\mathcal{A}} x_{ji} = 0 \qquad \forall i \in \mathcal{N}$$

$$x_{ij} \geq l_{ij} \qquad \forall (i,j) \in \mathcal{A}$$

$$-x_{ij} \geq -u_{ij} \qquad \forall (i,j) \in \mathcal{A}$$

$$(x_{ij} \text{ integer}) \qquad \forall (i,j) \in \mathcal{A}.$$

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

$$\text{s.t. } y_i - y_j + \mu_{ij} - \lambda_{ij} \leq c_{ij} \qquad \forall (i,j) \in \mathcal{A}$$

$$y_i \text{ free} \qquad \forall i \in \mathcal{N}$$

$$\lambda_{ij} \geq 0 \qquad \forall (i,j) \in \mathcal{A}$$

$$\mu_{ij} \geq 0 \qquad \forall (i,j) \in \mathcal{A}.$$

Complementary slackness conditions

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

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

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

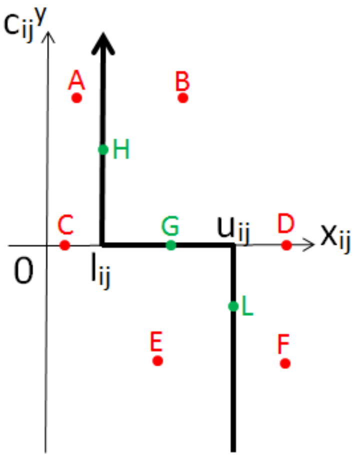
The out-of-kilter algorithm works on a restricted residual graph, because

- 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^y_{ij}.

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

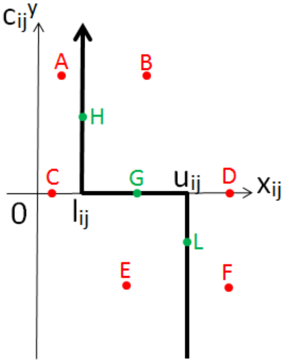


The kilter diagram for $(i,j) \in \mathcal{A}$.

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 : $k_{ij} = 0$	$r_{ij} = u_{ij} - l_{ij}$
L : $k_{ij} = 0$	$r_{ji} = u_{ij} - l_{ij}$

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.

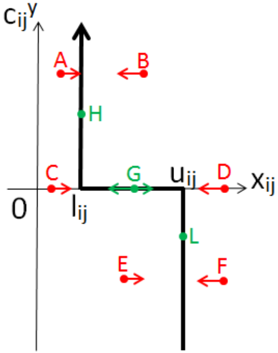
The corresponding arc $(p,q) \in R(x,y)$ is considered:

- $R(x,y)$ 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.

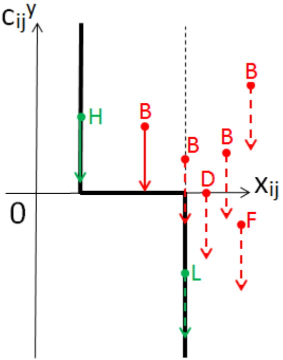
Let Q be the set of nodes reachable from q in $R(x, y)$ and $\overline{Q}$ be its complement.

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 $x_{ij} < u_{ij}$.

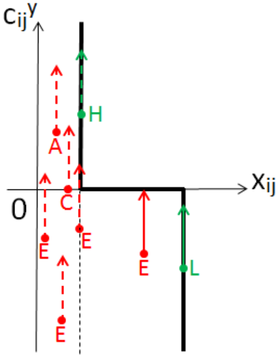
Consider the set Fw of forward arcs in $\mathcal{A}$ across the $(Q, \overline{Q})$ cut:

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

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 $x_{ij} > l_{ij}$.

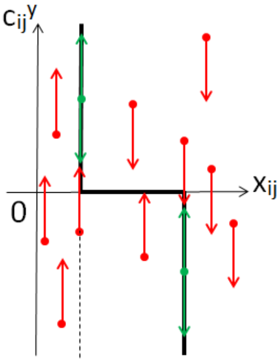
Consider the set Bw of backward arcs in $\mathcal{A}$ across the $(Q, \overline{Q})$ cut:

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

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 most 1 unit for each primal or dual iteration.

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