The Out-of-Kilter Algorithm[*]

Abigail Nix

May 2024

Abstract

In this project, I explored the Out-of-Kilter Algorithm for solving a minimum cost flow problem. This algorithm works by successively augmenting flow along shortest paths in order to decrease the kilter numbers of each arc. At each iteration of the algorithm, the mass balance constraints are satisfied, and the algorithm incrementally increases and optimality. We begin by establishing the background information on optimality conditions and node potentials necessary to understand the algorithm. Then, in the next section, we introduce the algorithm itself, justify correctness, and analyze complexity.

1 Introduction

The Out-of-Kilter algorithm works by satisfying the mass balance constraints of the network in every iteration, but flow bounds and optimality conditions do not need to be satisfied. However, in this project, we focus on a simplified version of the algorithm where flow bounds are respected, and we start with a feasible flow. Each iteration of the algorithm decreases the kilter number of arcs in the network by updating node potentials and augmenting flow along some shortest path, and the algorithm terminates when every arc is in-kilter. At this point, the flow found by the algorithm satisfies the complementary slackness optimality conditions, and is thus a minimum cost flow for the network. The description of the algorithm in this project is based on [1].

Before explaining exactly how the algorithm works, we will explain a couple optimality conditions that can be used as termination criterion for algorithms solving minimum cost flow problems. Recall that for shortest path problems, we have the following optimality conditions: $d(j) \leq d(i) + c_{ij}$ for every edge (i, j) in the network. When these conditions are satisfied, we know that the distance labels d define true shortest path distances. We can define different optimality conditions for the minimum cost flow problem instead of the shortest path problem.

2 Optimality Conditions

The first optimality conditions we will introduce are the negative cycle optimality conditions. They are defined as follows:

Theorem 2.1. *A feasible flow x^* is optimal for a minimum cost flow problem if and only if the residual network $G(x^*)$ contains no negative cycle.*

Proof. We will not explicitly prove these optimality conditions, but instead will give the proof idea. The first direction is proven by contrapositive. If $G(x)$ contains a negative cycle, then flow can be augmented along this negative cycle and decrease the overall objective function value, so the original flow x was not optimal. The other direction starts with a suboptimal flow, and decomposes the difference between this and an optimal flow into cycles. This then implies that the original flow is optimal. □

For the next optimality conditions we will describe, we first establish a few definitions.

Definition 2.2. Let $\pi(i)$, the node potential for node i, be some real number associated with node i.

[*]This project follows the description of the Out-of-Kilter algorithm in chapter 9 of [1].

Definition 2.3. For a given set of node potentials π, we define the reduced cost of an arc (i,j) as $c^\pi_{ij} = c_{ij} - \pi(i) + \pi(j)$.

With reduced costs defined, we note a couple properties that we will use to prove the reduced cost optimality conditions. First, consider a directed path P from node k to node l. Then, the reduced cost of the whole path forms a telescoping sum, so we can rewrite it as

$$\sum_{(i,j)\in P} c^\pi_{ij} = \sum_{(i,j)\in P} (c_{ij} - \pi(i) + \pi(j)) = \left(\sum_{(i,j)\in P} c_{ij}\right) - \pi(k) + \pi(l).$$

For a directed cycle C, using the same argument, the reduced cost of the cycle becomes

$$\sum_{(i,j)\in C} c^\pi_{ij} = \sum_{(i,j)\in C} c_{ij}.$$

Using these properties, we can prove the following optimality conditions by showing that they are equivalent to the negative cycle optimality conditions.

Theorem 2.4. *A feasible flow x^* is optimal for a minimum cost flow problem if and only if there exists a set of node potentials π that satisfy $c^\pi_{ij} \geq 0$ for every arc (i,j) in the residual network $G(x^*)$.*

Proof. Again, we only give the proof idea. For the first direction, we assume the negative cycle optimality conditions, and find shortest path distances d between node 1 and each node k. Then, by defining $\pi = -d$, the shortest path optimality conditions imply the reduced cost optimality conditions. For the other direction, assume the reduced cost optimality conditions, and let C be a directed cycle in the residual network. Then, since $\sum_{(i,j)\in C} c^\pi_{ij} = \sum_{(i,j)\in C} c_{ij}$, the negative cycle optimality conditions hold. □

The third and final optimality conditions that we will explore can be proven using the two sets of conditions we have already introduced. While the negative cycle and reduced cost optimality conditions are good certificates to use to check if a solution is optimal, the Out-of-Kilter algorithm uses the complementary slackness conditions as its termination criteria. In particular, the complementary slackness conditions define whether an arc is ”in-kilter” or ”out-of-kilter” for a given residual network.

The complementary slackness conditions are as follows:

Theorem 2.5. *A feasible flow x^* is optimal for a minimum cost flow problem if and only if there exists a set of node potentials π such that:*

1. *If $c^\pi_{ij} > 0$, then $x^*_{ij} = 0$,*
2. *If $0 < x^*_{ij} < u_{ij}$, then $c^\pi_{ij} = 0$,*
3. *If $c^\pi_{ij} < 0$, then $x^*_{ij} = u_{ij}$.*

Proof. For the first direction, suppose x^* satisfies the reduced cost optimality conditions, i.e. $c^\pi_{ij} \geq 0$ for all (i,j) in $G(x^*)$. Then consider the following three cases.

1. $c^\pi_{ij} > 0$. In this case, suppose that arc (j,i) is in $G(x^*)$, i.e., $x^*_{ij} > 0$. Then,

$$\begin{aligned} c^\pi_{ij} &= c_{ij} - \pi(i) + \pi(j) \\ &= -(-c_{ij} + \pi(i) - \pi(j)) \\ &= -(c_{ji} - \pi(j) + \pi(i)) = -c^\pi_{ji} \leq 0, \end{aligned}$$

   since $c^\pi_{ji} \geq 0$. This is a contradiction, since $c^\pi_{ij} > 0$. Thus, $x^*_{ij} = 0$.

2. $0 < x^*_{ij} < u_{ij}$. Here, both arcs (i,j) and (j,i) are in $G(x^*)$. But, both $c^\pi_{ij} \geq 0$ and $c^\pi_{ji} = -c^\pi_{ij} \geq 0$. This implies that $c^\pi_{ij} = c^\pi_{ji} = 0$.

3. $c^\pi_{ij} < 0$. Suppose arc $(i,j) \in G(x^*)$, i.e., $x^*_{ij} < u_{ij}$. Then, for this arc, since $c^\pi_{ij} < 0$, the reduced cost optimality conditions do not hold. So, (i,j) cannot be in $G(x^*)$, so we must have $x^*_{ij} = u_{ij}$.

For the other direction of this proof, assume the complementary slackness conditions hold for (x, π). Let $(i, j) \in G(x)$ and suppose for contradiction that $c_{ij}^{\pi} < 0$. Then, $x_{ij} = u_{ij}$. However, this implies that $(i, j) \notin G(x)$, a contradiction. This finishes the proof. □

In the next section, we will discuss the actual Out-of-Kilter algorithm, starting off by defining the kilter number of an arc using the complementary slackness optimality conditions.

3 The Algorithm

As we have previously stated, the Out-of-Kilter algorithm iteratively augments flow in a network along shortest paths found in the current residual network. Each augmentation's goal is to decrease the kilter number of some arc, which increases both the feasibility and optimality of the new flow. Throughout the algorithm, intermediate flows may not satisfy the upper and lower arc flow bounds or the complementary slackness optimality conditions. However, at each step, mass balance constraints are satisfied. Before displaying and explaining pseudocode for the algorithm, we describe what a kilter number for an arc is.

3.1 Kilter Numbers

Essentially, the kilter number of an arc measures how suboptimal that arc flow is. The complementary slackness optimality conditions are used to define the kilter number of an arc, and to form its kilter diagram, shown below.

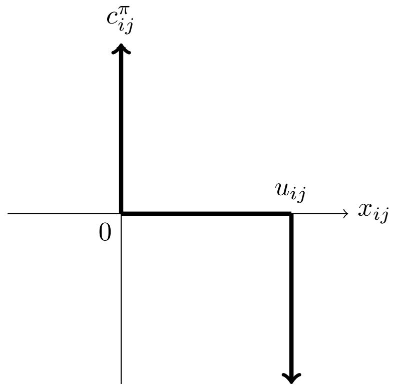

Figure 1: The kilter diagram for arc (i, j).

The kilter diagram tells you which (x_{ij}, c_{ij}^{π}) pair corresponds to the arc being "in-kilter" or "out-of-kilter". A point that is on the thick line in the diagram corresponds to an in-kilter arc, i.e., it satisfies the complementary slackness optimality conditions, while a point anywhere else on the grid corresponds to an out-of-kilter arc. As a reminder, the complementary slackness optimality conditions are:

1. If $c_{ij}^{\pi} > 0$, then $x_{ij}^{*} = 0$,
2. If $0 < x_{ij}^{*} < u_{ij}$, then $c_{ij}^{\pi} = 0$,
3. If $c_{ij}^{\pi} < 0$, then $x_{ij}^{*} = u_{ij}$.

From the kilter diagram, we can see each of these three conditions. The first condition corresponds to the leftmost line segment, where $c_{ij}^{\pi} > 0$ and $x_{ij} = 0$. The second condition corresponds to the center line segment, where $c_{ij}^{\pi} = 0$ and $0 < x_{ij} < u_{ij}$. Finally, the third condition corresponds to the rightmost line segment, where $c_{ij}^{\pi} < 0$ and $x_{ij} = u_{ij}$. Using this diagram, we can define the kilter number of an arc.

Definition 3.1. The kilter number of an arc (i, j), denoted k_{ij}, with current flow x_{ij} and reduced cost c_{ij}^{π}, is defined as how much x_{ij} needs to be changed without changing c_{ij}^{π} in order to make (i, j) in-kilter. In the diagram this is just how far the point would need to move in the x_{ij} direction in order to be on an in-kilter line segment.

Note that the kilter number can also be described analytically based on where the point (x_{ij}, c^{π}_{ij}) lies:

1. If $c^{\pi}_{ij} > 0$, then $k_{ij} = |x_{ij}|$.
2. If $c^{\pi}_{ij} = 0$ and $x_{ij} < 0$, then $k_{ij} = -x_{ij}$. If $c^{\pi}_{ij} = 0$ and $x_{ij} > u_{ij}$, then $k_{ij} = x_{ij} - u_{ij}$.
3. If $c^{\pi}_{ij} < 0$, then $k_{ij} = |u_{ij} - x_{ij}|$.

Note we can also define the kilter number using residual capacities in $G(x)$, $r_{ij} = u_{ij} - x_{ij}$. This is useful since the Out-of-Kilter algorithm works on residual networks. This definition is as follows:

$$k_{ij} = \begin{cases} 0 & \text{if } c^{\pi}_{ij} \geq 0 \\ r_{ij} & \text{if } c^{\pi}_{ij} < 0. \end{cases} \tag{3.1}$$

With the kilter numbers defined, we can describe the Out-of-Kilter algorithm.

3.2 Pseudocode

The Out-of-Kilter algorithm starts with $\pi(i) = 0$ for every node i, and a feasible flow, and then iteratively chooses an out-of-kilter arc (p, q) in the current residual network. In each iteration, the algorithm updates the node potentials π and augments flow along a shortest path from q to p if (p, q) is still an out-of-kilter arc ($c^{\pi'}_{pq} < 0$). This process does not increase the kilter number of any arc, and strictly decreases the kilter number of (p, q). Once all kilter numbers are zero, the current flow satisfies the complementary slackness optimality conditions, and thus, is optimal for the minimum cost flow problem. We provide pseudocode below.

Algorithm 1 Out-of-Kilter Algorithm for finding a Minimum Cost Flow

```
 1: procedure OutOfKilter($G$)
 2:     $\pi := 0$
 3:     find feasible flow $x$                              ▷ using a Max Flow Algorithm
 4:     define residual network $G(x)$
 5:     for $(i, j) \in G(x)$ do
 6:         $k_{ij}$ := Kilter($(i, j)$)
 7:     end for
 8:     while there exists $(i, j) \in G(x)$ with $k_{ij} > 0$ do      ▷ there is an out-of-kilter arc
 9:         select out-of-kilter arc $(p, q) \in G(x)$
10:         define length of each arc $(i, j) \in G(x)$ as $\max\{0, c^{\pi}_{ij}\}$
11:         let $d(i)$ denote the shortest path distance from $q$ to $i$ for each $i \in G(x) - \{(q, p)\}$
12:         let $P$ denote the shortest path from $q$ to $p$
13:         $\pi'(i) := \pi(i) - d(i)$ for each node $i$ in $G(x)$
14:         if $c^{\pi'}_{pq} < 0$ then
15:             $W := P \cup \{(p, q)\}$
16:             $\delta := \min\{r_{ij} : (i, j) \in W\}$
17:             augment $\delta$ units of flow along $W$
18:             update $x$ with new feasible flow, residual network $G(x)$, and reduced costs $c^{\pi'}_{ij}$
19:         end if
20:     end while
21:     return $x$                                          ▷ $x$ is the minimum cost flow
22: end procedure
```

3.3 Correctness and Complexity

The correctness of the Out-of-Kilter algorithm relies on the following two lemmas about the two parts of the algorithm where the kilter number of some arc could change: when updating node potentials or when augmenting flow.

Lemma 3.2. *Updating node potentials, i.e., setting $\pi'(i) := \pi(i) - d(i)$ for all nodes i, does not increase the kilter number of any arc in $G(x)$.*

Proof. Consistent with notation in the algorithm, let π be the node potentials before updating, and π' be the updated node potentials. Suppose for contradiction that this update increases the kilter number of some arc $(i,j) \in G(x)$. Since changing π does not change r_{ij}, the definition 3.1 implies that the only way this can happen is if $c^{\pi}_{ij} \geq 0$ and $c^{\pi'}_{ij} < 0$. Note that since $c^{\pi}_{pq} < 0$, and $c^{\pi}_{ij} \geq 0$, we have $(p,q) \neq (i,j)$. Recall that in the algorithm, we defined d as shortest path distances, with edge lengths $\max\{0, c^{\pi}_{ij}\}$. Therefore, d satisfies the shortest path optimality constraints, so

$$d(j) \leq d(i) + \max\{0, c^{\pi}_{ij}\} = d(i) + c^{\pi}_{ij},$$

since $c^{\pi}_{ij} \geq 0$. Thus,

$$c^{\pi'}_{ij} = c_{ij} - (\pi(i) - d(i)) + (\pi(j) - d(j)) = c^{\pi}_{ij} + d(i) - d(j) \geq 0.$$

This is a contradiction, so updating node potentials does not increase the kilter number of any arc. ☐

Lemma 3.3. *Augmenting flow along $W = P \cup \{(q,p)\}$ does not increase the kilter number of any arc in $G(x)$, and strictly decreases k_{pq}, the kilter number of arc (p,q).*

Proof. Since we only augment flow along W, the only arcs whose kilter numbers can change are those along this cycle (or their reversed arcs). Since P is a shortest path with arc lengths $\max\{0, c^{\pi}_{ij}\}$, then for $(i,j) \in P$,

$$d(j) = d(i) + \max\{0, c^{\pi}_{ij}\} \geq d(i) + c^{\pi}_{ij}.$$

We defined $\pi' = \pi - d$, so we have

$$c^{\pi'}_{ij} = c_{ij} - (\pi(i) - d(i)) + (\pi(j) - d(j)) = c^{\pi}_{ij} + d(i) - d(j) \leq 0.$$

Now, since we augment flow along W by $\delta = \min\{r_{ij} : (i,j) \in W\}$ units, the new flow does not go over the upper bound on any arc. Thus, if $c^{\pi'}_{ij} = 0$, the kilter number of arc (i,j) stays 0, and if $c^{\pi'}_{ij} < 0$, then k_{ij} will decrease because r_{ij} decreases when the flow on that arc (i,j) increases. Also note that augmenting flow along (i,j) may introduce the arc (j,i) into the new residual network, but since $c^{\pi'}_{ij} \leq 0$, then $c^{\pi'}_{ji} \geq 0$, meaning $k_{ji} = 0$, and (j,i) is in-kilter.

Finally, consider the arc (p,q). Note that we only augment flow along this arc if $c^{\pi'}_{pq} < 0$. Since $c^{\pi'}_{pq} < 0$, we know that $k_{pq} = r_{pq}$. Thus, increasing the flow value on this arc strictly decreases its kilter number because it decreases the arc's residual capacity. Note that $c^{\pi'}_{qp} = -c^{\pi'}_{pq} > 0$, so (q,p) is an in-kilter arc.

Therefore, the flow augmentation step of the algorithm does not increase the kilter number of any arc, strictly decreases the kilter number of (p,q), and only adds in-kilter arcs to the updated residual network. ☐

With these two lemmas proven, we can show both correctness and finite termination of this algorithm. By our definition of the kilter number in 3.1, the maximum possible kilter number is $U = \max\{u_{ij}\}$. So, the sum of the kilter numbers at the start of the algorithm is at most mU, where m is the number of arcs in the network. As proven in lemmas 3.2 and 3.3, each iteration of the algorithm does not increase the kilter number of any arc, and strictly decreases the kilter number of one arc (by at least 1). Thus, the algorithm takes at most mU iterations. In each iteration, the main operation is solving a shortest path problem. Using Dijkstra's algorithm, this takes $O(n^2)$, but the running time depends on the algorithm used. So, the Out-of-Kilter algorithm takes at most $O(mUn^2)$ time, i.e., runs in pseudopolynomial time.

Note that in this project, we focus on a simpler version of the Out-of-Kilter algorithm, where we start off with a feasible flow. However, this algorithm can start with a flow of $x = 0$, and in this case, feasibility is not maintained at each step. Here, we form the residual network arcs differently for arcs that have flow below the lower bound or above the upper bound. More information on this form of the algorithm can be found in [1].

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993.

[2] E. P. Durbin and David Kroenke. *The Out-of-Kilter Algorithm: A Primer.* Santa Monica, CA: RAND Corporation, 1967.