Minimum Cost Flow: By Zealint– TopCoder Member

In Part 1, we looked at the basics of minimum cost flow. In this section, we'll look at three algorithms that can be applied to minimum cost flow problems.

Working with Residual Networks

Let's consider the concept of residual networks from the perspective of min-cost flow theory. You should be familiar with this concept thanks to maximum flow theory, so we'll just extend it to minimum cost flow theory.
We start with the following intuitive idea. Let G be a network and x be a feasible solution of the minimum cost flow problem. Suppose that an edge (i,j) in E carries x_{ij} units of flow. We define the residual capacity of the edge (i,j) as $r_{ij} = u_{ij} – x_{ij}$. This means that we can send an additional r_{ij} units of flow from vertex i to vertex j. We can also cancel the existing flow x_{ij} on the arc if we send up x_{ij} units of flow from j to i over the arc (i,j). Now note that sending a unit of flow from i to j along the arc (i,j) increases the objective function by c_{ij}, while sending a unit of flow from j to i on the same arc decreases the flow cost by c_{ij}.

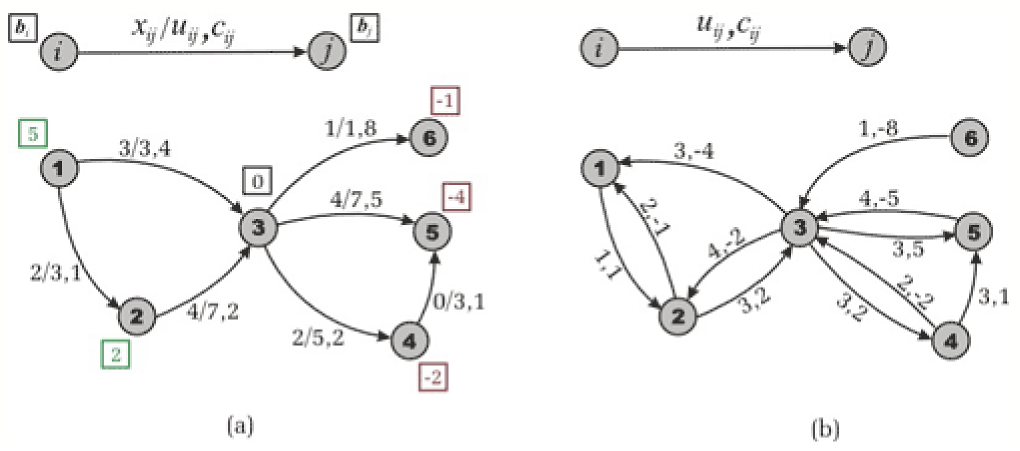

Figure 1. The transportation network from Part 1. (a) A feasible solution. (b) The residual network with respect to the found feasible solution.

Based on these ideas we define the residual network with respect to the given flow x as follows. Suppose we have a transportation network G = (V,E). A feasible solution x engenders a new (residual) transportation network, which we are used to defining by G_x = (V,E_x), where E_x is a set of residual edges corresponding to the feasible solution x.

What is E_x? We replace each arc (i,j) in E by two arcs (i,j), (j,i): the arc (i,j) has cost c_{ij} and (residual) capacity $r_{ij} = u_{ij} – x_{ij}$, and the arc (j,i) has cost $-c_{ij}$ and (residual) capacity $r_{ji}=x_{ij}$. Then we construct the set E_x from the new edges with a positive residual capacity. Look at Figure 1 to make sure that you understand the construction of the residual network.

You can notice immediately that such a definition of the residual network has some technical difficulties. Let's sum them up:

- If G contains both the edges (i,j) and (j,i) (remember assumption 2) the residual network may contain four edges between i and j (two parallel arcs from i to j and two contrary). To avoid this situation we have two options. First, transform the original network to one in which the network contains either edge (i,j) or edge (j,i), but not both, by splitting the vertexes i and j. Second, represent our network by the adjacency list, which is handling parallel arcs. We could even use two adjacency matrixes if it were more convenient.
- Let's imagine now that we have a lot of parallel edges from i to j with different costs. Unfortunately, we can't merge them by summarizing their capacities, as we could do while we were finding the maximum flow. So, we need to keep each of the parallel edges in our data structure separate.

The proof of the fact that there is a one-to-one correspondence between the original and residual networks is out the scope of this article, but you could prove all the necessary theorems as it was done within the maximum flow theory, or by reading [1].