Flow Problems as Linear Programming

At this point, it would be interesting to revisit a problem discussed earlier in this course, namely finding maximum flows in a flow network. Specifically, when givien a flow network $G = (V, E)$ with two special vertices $s \in V$ (the source) and $t \in V$ (the sink) where each edge $(u, v) \in E$ has a non-negative capacity $c(u, v) \geq 0$, the objective was to find a flow $f(s, t)$ of maximum value that does not violate any of the constraints.

Notice that we can reformulate this problem as a linear programming problem. Specifically, we seek to

$$\text{maximize} \sum_{\forall v \in V} f(s, v) - \sum_{\forall v \in V} f(v, s)$$

subject to

$$\begin{array}{rcll} f(u,v) & \leq & c(u,v) & \forall u, v, \in V \\ \sum_{\forall v \in V} f(u,v) & = & \sum_{\forall v \in V} f(v,u) & \forall u \in V - \{s,t\} \\ f(u,v) & \geq & 0 & \forall u, v, \in V \end{array}$$

Shortly, we will examine the simplex algorithm as a strategy for solving linear programs, and as we will see, this will introduce an alternative to the Ford-Fulkerson method for finding maximum flows. What is interesting, however, is that we can relate the simplex process directly to the path augmentation process from Ford-Fulkerson.

As a precursor to the discussion on the simplex algorithm, we will describe the basic process here. First, we note that we can apply a geometric interpretation to the simplex algorithm. Specifically, once we have converted a linear programming problem to slack form, we see that an optimality solution must satisfy a set of equality constraints. The equality constraints define an n-dimensional convex polyhedron that actually encloses all of the feasible points. Indeed, we call this polyhedron, the "convex hull" of the set of feasible points. Furthermore, this convex hull is also called a "simplex," thus the name "simplex algorithm."

As a side note, we also point out that we may have situations where there is no optimal solution to the linear program. This can occur in one of two cases:

1. The linear program is "infeasible," meaning the constraints are so tight that no points exist satisfying all constraints. A simple example of this is if we have inconsistent constraints such as $x \leq 1$ and $x \geq 2$.

2. The set of constraints are so loose as to make the feasible region unbounded. As a simple example, suppose we wish to maximize $x_1 + x_2$ and the only constraints are $x_1 \geq 0$ and $x_2 \geq 0$.

The simplex algorithm proceeds under the assumption the feasible region is bounded and non-empty.

The basic idea behind the simplex algorithm is to start at one of the vertices defined by the set of constraints and examine adjacent vertices. When considering the adjacent vertices, the objective values of these vertices are calculated, and the algorithm moves, in a "hillclimbing" fashion, to the vertex with the greatest improvement in objective value. If there is no neighboring vertex with a higher objective value, the algorithm terminates.

Let's consider what happens with this approach in the context of the maximum flow problem. The problem begins with no flow being introduced into the network. This corresponds to a minimum value since the flow cannot be negative. Since it is an extreme value, it also corresponds to a vertex in the simplex. When considering how to move to an adjacent vertex in the simplex, we look for another extreme point by introducing as much flow as possible without violating some constraint. We note that the equality constraints and the non-negativity constraints are handled directly by flow conservation and a process of increasing flow. Thus the constraints of interest are the capacity constraints. Recall that the approach taken was to identify an "augmenting path" and apply additional flow equal to the most restrictive residual capacity on that path. This is directly analogous to traversing to a neighboring vertex in the simplex. In other words, the path augmentation process corresponds directly to the hillclimbing process of the simplex algorithm.