Erick Lin

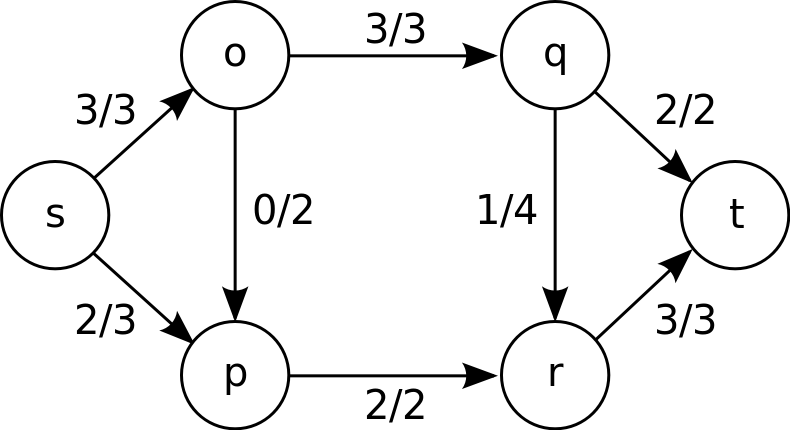
Dr. Baker

MATH 4802

12/02/14

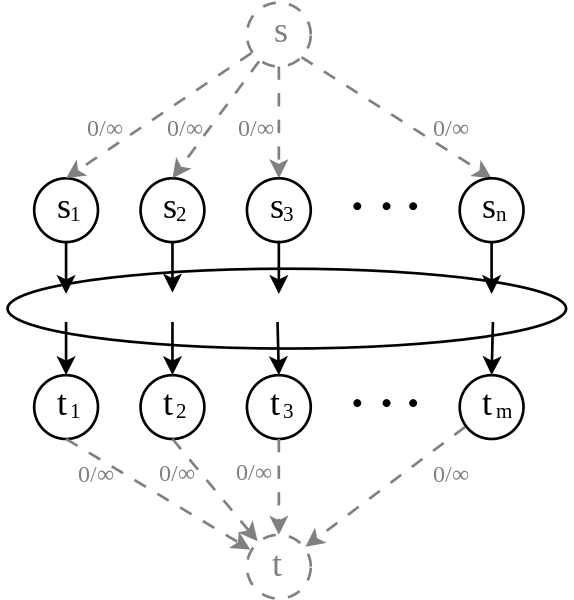
Maximum Flow as a Linear Program

Imagine a locale in the desert of Tatooine that has one moisture farm constantly producing a quantity of water and a distant village that has a demand for this water. Transport vehicles transport the water from the farm to the village using a large number of connected passageways throughout the desert, each of which has a specific capacity for the maximum number of transport vehicles it can hold. This situation can be represented by a **flow network**, or a directed graph with one node designated as the **source** and another designated as the **sink** , a **capacity** and a **flow variable** assigned to the edge from to for each edge, and an unknown flow from to . In addition, flow networks follow a principle of conservation in which the flow leading into a node equals the flow leading out of the node. Then the question of maximizing , or the rate at which transport vehicles arrive at the village, by deciding what number of vehicles should use each passageway at any given time is an example of the **maximum flow problem**.



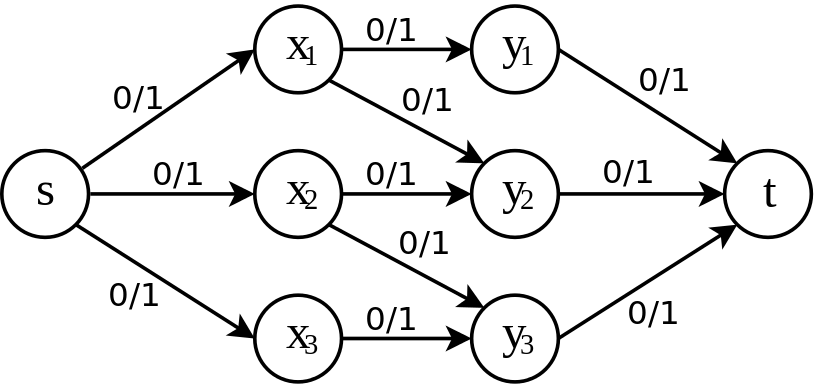
**Figure 1**. An example of a flow network. For each edge, the number on the right represents its capacity and the number on the left represents the value of its flow variable. The maximum flow is .

If the example above had multiple farms and multiple villages instead, then it can be transformed into the maximum flow problem by creating two new vertices, one with edges of infinite capacity leading to each farm, and the other with edges of infinite capacity leading to it from each village. These new vertices would be the source and the sink, respectively.



**Figure 2**. Transforming the multiple-source multiple-sink problem into a regular maximum flow problem.

This same technique can be applied to the bipartite matching problem to transform it into the maximum flow problem, after assigning a capacity of 1 to each edge leading from an element of one set to an element of the other.



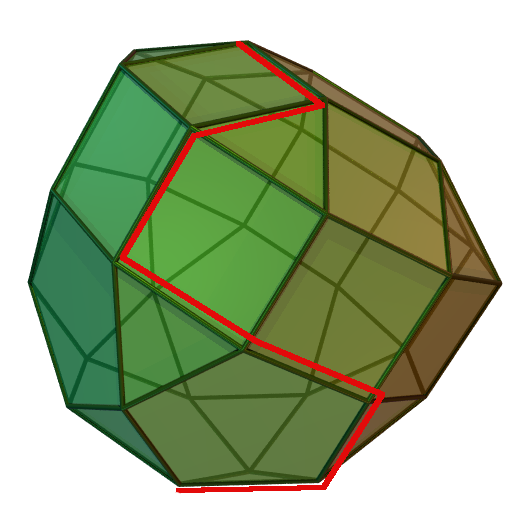
**Figure 3**. Transforming a bipartite graph problem into a maximum flow problem. One set consists of , , and , and the other of , , and .

A number of algorithms exist for solving the maximum flow problem, such as Ford-Fulkerson and Edmonds-Karp. However, this report will focus on the method of solving the maximum flow problem by formulating it as a **linear optimization problem**, more commonly known as a linear program. Linear programs are characterized by a set of linear inequalities called **constraints**, and a linear expression to be maximized called the **objective function**. In the case of maximum flow, the objective function is the sum of the flow variables of the edges leaving , or the sum of the flow variables of the edges entering . The constraints include the flow variables being non-negative and not exceeding their corresponding capacities (), as well as the principle of conservation for each node ().

Once the linear program has been formulated, the **simplex method** is the most commonly used algorithm for solving the problem. The first step of the simplex method is to introduce **slack variables** to convert all of the inequalities into equations. For example, the slack variable converts the inequality into the equation . The objective function is also converted into an equation by setting it equal to a variable and subtracting the objective function from both sides, making the right side equal to 0. Then, these equations containing the constraints and the objective function can be represented by an augmented matrix, with the variables on the left side and the constant term on the right. Then the recursion is as follows:

1. Determine if the row containing the objective function has any negative entries on the left side. If not, then the entry on the right side of the same row is the solution .
2. If it does, find the most negative entry in the row containing the objective function and choose its column as the **pivot column**.
3. For each row corresponding to the constraints, assign a ratio that equals the entry on the right side divided by the entry sharing the same column as the pivot column. Choose the row with the least non-negative ratio as the **pivot row**.
4. Choose the entry where the pivot column and pivot row intersect as the **pivot element**.
5. Row-reduce until all of the entries above and below the pivot element equal 0.
6. Return to step 1.

Geometrically, the constraints when graphed can be seen as forming the edges of a convex polytope, which is always bounded for maximum flow problems because the capacities are finite. The simplex method is an **exterior points method**, and can be visualized as moving from vertex to vertex of the polytope (which are local maxima) as long as it finds a neighboring vertex that increases the objective function. If there are no neighbors that have a higher value for the objective function, then the current vertex is at the absolute maximum.



**Figure 3.** The simplex method visualized.

In the worst case, the simplex method runs in exponential time due to it having to make an exponential number of pivots. However, simplex is often preferred in practical applications such as maximum flow because it almost always runs in linear time.