CPSC 500 Fundamentals of Algorithm Design and Analysis

Lecturer: Prof. Will Evans
Date: November 4, 2013
Scriber: YuanFang Chi

1. Overview

This lecture introduces linear programming. Several examples are shown in this lecture to illustrate formulation of linear program from a problem. Simplex algorithm that solves linear programs and Ford – Fulkerson's method that computes maximum flow of a flow network are also discussed in this lecture.

2. What is Linear Programming?

In general linear program problems, we are given a set of variables and we want to assign real values to variables so that:

- They satisfy given linear equations and/or inequalities.
- They maximize a given linear "objective" function.

2.1. Example: Chocolatier

Suppose a chocolatier has two kinds of chocolates, call it Box1 and Box2. Chocolate Box1 gives a profit of $1, with daily demand at most 200 boxes. Chocolate Box2 gives a profit of $6, with daily demand at most 300 boxes. The total number of boxes of chocolate the chocolatier can produce is at most 400 boxes per day.

How many boxes should the chocolatier produce of each type to maximize its profit?

Let X_1 be the number of boxes of type Box1 chocolate to produce per day, X_2 be the number of boxes of type Box2 chocolate to produce per day.

We have the following linear program that represents the problem.

$$Maximize\ X_1 + 6X_2$$
$$X_1 \leq 200$$
$$X_2 \leq 300$$
$$X_1 + X_2 \leq 400$$
$$X_1, X_2 \geq 0$$

The set of all feasible solutions of this linear program is bounded to a convex polygon by the constraints as shown in Figure 1.

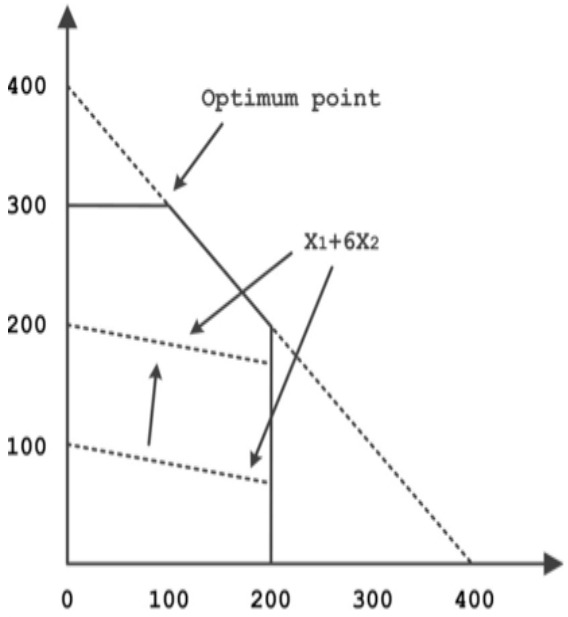

Figure 1

Contour lines of the object function $X_1 + 6X_2$ for different values of X_1 and X_2, we can find the optimum point at $X_1 = 100$ and $X_2 = 300$.

2.2. Infeasibility and Unboundedness

The optimum point exists unless one of the two circumstances occurs:

- The linear program is infeasible, which means it is impossible to satisfy all of the constraints of this linear program. For example, a linear program is infeasible if it has constraints like:

$$X_1 \leq 1 \; and \; X_1 \geq 2$$

- The problem is unbounded. For example, for a linear program

$$Maximize \; X_1 + X_2$$
$$X_1, X_2 \geq 0$$

The constraint of the linear program describes an unbounded set of solutions.

3. Simplex Algorithm (Dantzig 1947)

The simplex algorithm solves linear program according to the facts that the optimum value can be found at a vertex of a convex feasible region and a locally optimal vertex implies a globally optimal vertex in the region. The algorithm is described as follows:

- Start at a vertex v of feasible set.
- Repeatedly look for a neighbor vertex v' of v with better objective value.
- If there is no neighbor vertex for the vertex v, then the simplex algorithm declares it has found the optimum value for the linear program.

4. Bandwidth Allocation

Suppose we need to establish connection between three users A, B and C in a network shown in Figure 2.

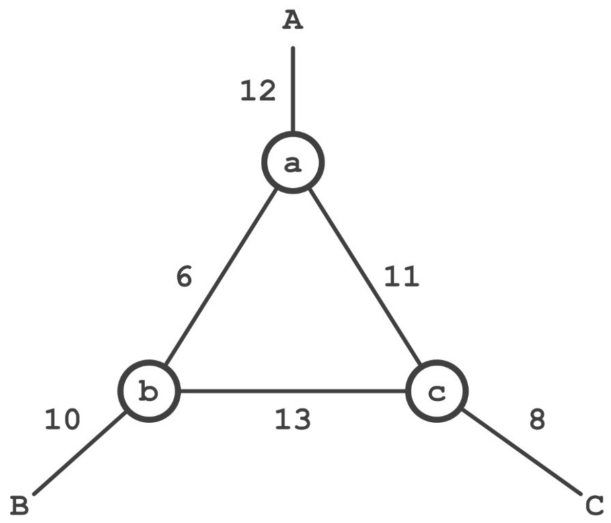

Figure 2

Each connection requires at least 2 units of bandwidth. Two paths satisfy each connection, a long path and a short path. For example, a long path from user A to user B is routed in path: A -> a -> c -> b -> B. A short path from user A to user B is routed in path: A -> a -> b -> B. The maximum bandwidth of a path should not exceed the maximum bandwidth on the edges along the path. Connection between user A and B pays $3 per unit, connection between user B and C pays $2 per unit and connection between user A and C pays $4 per unit.

How do we configure the connections to maximize the profit?

Let X_{AB} be the amount of bandwidth from user A to user B going through the long path and X'_{AB} be the amount of bandwidth from user A to user B going through the short path.

We can represent the problem in following linear program:

$$Maximize\ 3(X_{AB} + X'_{AB}) + 2(X_{BC} + X'_{BC}) + 4(X_{AC} + X'_{AC})$$
$$X_{AB} + X'_{AB} + X_{BC} + X'_{BC} \leq 10$$
$$X_{AB} + X'_{AB} + X_{AC} + X'_{AC} \leq 12$$
$$X_{BC} + X'_{BC} + X_{AC} + X'_{AC} \leq 8$$
$$X_{AB} + X'_{AC} + X'_{BC} \leq 6$$
$$X'_{AB} + X_{BC} + X'_{AC} \leq 13$$
$$X'_{AB} + X'_{BC} + X_{AC} \leq 11$$
$$X_{AB} + X'_{AB} \geq 2$$
$$X_{BC} + X'_{BC} \geq 2$$
$$X_{AC} + X'_{AC} \geq 2$$
$$X_{AB}, X'_{AB}, X_{BC}, X'_{BC}, X_{AC}, X'_{AC} \geq 0$$

We can use a solver or the simplex algorithm introduced earlier in this lecture to solve this linear program. The solution to this linear program is:

$$X_{AB} = 0$$
$$X'_{AB} = 7$$
$$X_{BC} = 1.5$$
$$X'_{BC} = 1.5$$
$$X_{AC} = 0.5$$
$$X'_{AC} = 4.5$$

5. Flow Network

Flow network is a directed graph $G = (V, E)$. There is a source vertex $s \in V$ and a sink vertex $t \in V$. Each edge $(u, v) \in E$ has a positive capacity $c(u, v)$. A flow is an assignment f of real numbers to edges of G and satisfies the following properties:

- Capacity constraints.

$$f(u, v) \leq c(u, v)$$

- Flow conservation. For all vertex $v \in V, v \neq s, t$, the amount of flow entering the vertex equals the amount of flow leaving the vertex.

$$\sum_{\substack{(u,v)\in E \\ u\in V}} f(u,v) = \sum_{\substack{(v,w)\in E \\ w\in V}} f(v,w)$$

The size of flow is the net amount of flow travelling from the source $s \in V$ to a vertex $v \in V$.

$$size(f) = \sum_{(s,v)\in E} f(s,v) - \sum_{(v,s)\in E} f(v,s)$$

How to find a flow with maximum size?

5.1. Max-Flow via Path Augmentation (Ford - Fulkerson 1962)

Ford and Fulkerson developed an algorithm that computes the maximum flow in 1962. The algorithm is described as follows:

- Start from zero flow.
- Choose a path from source s to sink t in the residual network $G^f = (V, E^f)$.
- Repeatedly augment flow along edges of the path up to maximum capacity of the path.

For a flow $f\ (u,v)$ in a flow network $G = (V,E)$ with a capacity $c(u,v)$, $(u,v) \in E$, the residual network $G^f = (V, E^f)$ for this flow f has a capacity $c^f(u,v)$ defined as:

$$c^f(u,v) = \begin{cases} c(u,v) - f(u,v), & f(u,v) < c(u,v) \\ f(v,u), & f(v,u) > 0 \\ 0, & otherwise \end{cases}$$

5.1.1. Example

For the flow network shown in Figure 3, let $f\ (s,t)$ be the amount of flow going from the source $s \in V$ to the sink $t \in V$.

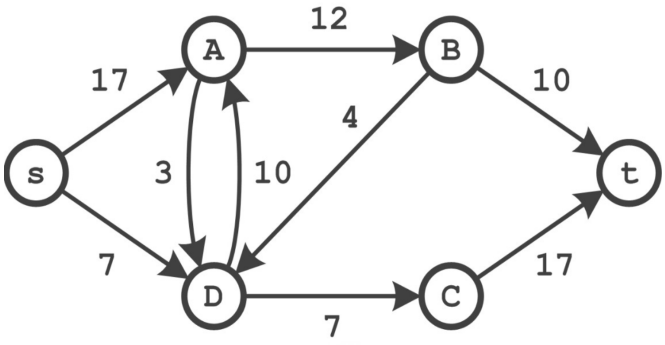

Figure 3

- Choose a path s -> A -> B -> t. The capacity on the path is 10.
- Augment a flow $f\ (s,t) = 10$ as shown in Figure 4 on this path

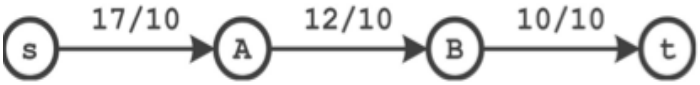

Figure 4

- The residual network for this flow is shown in Figure 5.

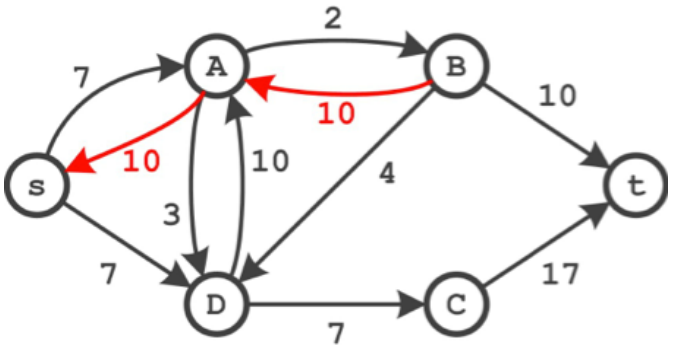

Figure 5

6. Conclusion

During this lecture, we described a simple algorithm to solve the linear program and formulated a simple Bandwidth Allocation problem as a linear program. At the end, we showed how to use Ford-Fulkerson algorithm to compute maximum flow in a flow network. In the future lecture, we will discuss how to guarantee termination of Ford-Fulkerson algorithm and how to prove the result is correct.

Reference:

[1] Dasgupta, Sanjoy, Christos H. Papadimitriou, and Umesh V. Vazirani. Algorithms. Boston: McGraw-Hill Higher Education, 2008.

[2] WIKIPEDIA. Ford-Fulkerson algorithm. http://en.wikipedia.org/wiki/Ford–Fulkerson_algorithm, November 2013.