CSC 373 H 1 Y — Summer 2007

University of Toronto — St. George Campus

Lecture Summary for Week 8

This summary is not a replacement for the lecture. If you miss a class, please arrange with a friend to take note for you.

4 LINEAR PROGRAMMING [CLRS Chapter 29]

The Linear Programming (LP) problem is to optimize (either maximize or minimize) an linear objective functions subject to a set of linear constraints which are linear inequalities. In particular, we are given coefficients

$$c_1, c_2, \ldots, c_n, \qquad a_{ij} \text{ for } 1 \le i \le m, 1 \le j \le n, \qquad b_1, b_2, \ldots, b_m$$

We want to optimize the objective function

$$c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad \text{for } i = 1, 2, \ldots, m \tag{1}$$

$$x_j \ge 0 \qquad \text{for } j = 1, 2, \ldots, n \tag{2}$$

(The above setting is often referred to as the *standard form*. In general, an LP might have some *equalities* instead of inequalities, or some inequalities have $\ge$ instead of $\le$, or some nonnegativity constraints in (2) might be missing. But it is easy to convert this more general form to the standard form.)

The constraints (1) and (2) define the set of *feasible solutions* called the *feasible region*. This is the set of all points $(x_1, x_2, \ldots, x_n)$ in the n-dimensional Euclidean space that satisfy (1) and (2). The feasible region might be empty or nonempty. A nonempty feasible region might be bouned or unbounded; but it is always convex.

Example: Consider the case $n = 2$ and the following set of constraints:

$$x_1 - x_2 \le -1$$
$$4x_1 + 3x_2 \le 12$$
$$-x_1 - x_2 \le -2$$
$$x_1, x_2 \ge 0$$

The students is recommended to draw the feasible region by draw the line that define the constraints given above. They should also try to find the point(s) in the feasible region that maximize the following objective functions:

$$x_2$$
$$x_1 + x_2$$
$$3x_1 - 2x_2$$

(For an objective function $c_1x_1+c_2x_2$, one way is to start with a line of the form $c_1x_1+c_2x_2 = c$ that does not intersect the feasible region, and move the line toward the feasible region. The optimal solution(s) are either the first or last intersection of the line with the feasible region.)

Note that for each of these objective functions, the optimal solution(s) are at the vertices or on some edges of the feasible region. In general, for an LP with a bounded feasible region, the optimal value is always obtained at the boundary of the feasible region.

The above observation provides the intuition behind the *convex algorithm* that solves LP, invented by G. Dantzig in 1947. This algorithm requires exponential time in the worst case, but in practice it runs quite fast. The problem of finding a polytime algorithm for LP was open for some time, until it was resolved by L. G. Khachian in 1979 with the ellipsoid algorithm. Another polytime algorithm, called the interior-point method, is discovered later by N. Karmarkar in 1984.

Although the simplex algorithm worst case running time is exponential, specialized simplex algorithm designed for some special problem, such as network flow, runs in polynomial time. In the next section we will see how to transform the Max Flow problem to an LP.

We will not learn these algorithms in details. Instead, we will focus on how to formulate a given problem as an LP. The formulation should often be accompanied by a justification. Here are the steps to formulate a problem as an LP:

1. Specify the variables and explain their intended meaning.
2. Specify the objective function and whether it is to be maximized or minimized..
3. Give the constraints.
4. Justify that the optimal solution to the LP can be used to solve the original problem.

4.1 Max Flow Problem as an LP

Given a flow network $G = (V, E)$ with the capacity function c on the edges. We set up an LP as follows.

For each edge (u, v), there is a variable $x_{u,v}$ which is intended to be the flow value on (u, v).

The objective is to maximize

$$\sum_{v \in V \text{ and } (s,v) \in E} x_{s,v}$$

To obtain the constraints, we translate the Capacity and Conservation conditions into linear inequalities. The Capacity condition is translated into

$$0 \leq x_{u,v} \qquad \text{and} \qquad x_{u,v} \leq c(u, v) \qquad \text{for } (u, v) \in E$$

The Conservation condition is translated into

$$\text{for each } v \in V, v \neq s, v \neq t : \sum_{(u,v) \in E} x_{u,v} = \sum_{(v,w) \in E} x_{v,w}$$

The justification is traightforward in this case: For each flow f, the values $x_{u,v} = f(u, v)$ satisfies all the constrains. Also, if the variables $x_{u,v}$ satisfy all the constraints then define $f(u, v) = x_{u,v}$. The function f is a valid flow.

4.2 0-1 Integer Programming

0-1 Integer Programming (0-1 IP) is defined similar to LP, except for the coefficients $a_{i,j}, b_i, c_j$ are all integers, and the variables x_j can take value either 0 or 1.

Note that 0-1 IP is an NP-complete problem, and is not known to have a polytime algorithm. This might sound paradoxical, because at the first sight 0-1 IP might look like a special instance of LP. Notice that the requirement that x_j can take value only 0 or 1 cannot be expressed as a linear constraint, so indeed 0-1 IP is not an instance of LP.

We introduce 0-1 IP here because many (NP-complete) problems can be formulated as 0-1 IP, which in turns can be used to design an approximation algorithm for the original problem. (Approximation is a topic to be learned later.)

Notice also that the 0-1 IP formulations of the NP-complete problems (such as CNF-SAT in the next section) are actually the reduction from these problems to 0-1 IP.

4.3 CNF-SAT as a 0-1 IP

CNF-SAT: Given a propositional formula in CNF. The CNF-SAT problem is to decide whether the formula is satisfiable.

We will now explain the terminologies:

Propositional formulas are built from propositional variables $p_1, p_2, \ldots$ using connective $\neg, \wedge, \vee$ and parentheses (,).

For example

$$(((p_1 \vee p_2) \wedge (\neg p_1 \vee \neg p_2)) \vee \neg(p_3 \wedge p_4)) \tag{3}$$

is a propositional formula.

CNF formula: A propositional formula is in CNF (Conjunctive Normal Form) if it is a conjunction of clauses:

$$C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

Here a **clause** is a disjunction of literals:

$$\ell_1 \vee \ell_2 \ldots \ell_k$$

where a **literal** is either a propositional variable p_i or its negation $\neg p_i$.

The formula (3) is NOT in CNF. The formula below is an example of a CNF formula:

$$(p_1 \vee \neg p_2 \vee p_5) \wedge (p_2 \vee \neg p_4 \vee p_5 \vee p_7)$$

A truth assignment τ is a mapping that assign a Boolean value to each propositional variable:

$$\tau : \{p_1, p_2, \ldots\} \longrightarrow \{True, False\}$$

A truth assignment τ **satisfies** a formula F if F is evaluated True when all variables get the values given by τ.

Note that τ satisfies a CNF formula F if and only if τ satisfies all clauses of F.

The CNF-SAT is the first problem shown to be NP-complete (Cook's Theorem). In fact, 3CNF-SAT is already NP-complete, where 3CNF-SAT is defined as CNF-SAT but every clause contains at most 3 literals.

Given a CNF formula F:

$$F = C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

To formulate CNF-SAT as a 0-1 IP, we introduce a variable y_j for each propositional variable p_j. The intention is that $y_j = 1$ if and only if p_j is True. Also, for each clause C_i introduce a variable x_i so that $x_i = 1$ if and only if C_i True.

The objective is to maximize

$$x_1 + x_2 + \ldots + x_m$$

(Then F is satisfiable if and only if $max(x_1 + x_2 + \ldots + x_m) = m$.)

To express the constraints, we define for each clause C_i a term t_i so that $t_i = 0$ if and only if C_i is False. The term t_i is defined as follows. For each literal p_j in C_i we associate the term y_j, and for each literal $\neg p_j$ in C_i we associate the term $(1 - y_j)$. Now t_i is the sum of all terms associated with the literals in C_i. The constraints are:

$$x_i \leq t_i \qquad \text{for each clause } C_i$$

Example: Consider the CNF formula

$$F \equiv (p_1 \vee \neg p_2 \vee p_5) \wedge (\neg p_1 \vee \neg p_2) \wedge (p_2 \vee \neg p_3 \vee p_4 \vee \neg p_5)$$

The variables are: $y_1, y_2, \ldots, y_5, x_1, x_2, x_3$. The objective is to maximize

$$x_1 + x_2 + x_3$$

The constraints are:

$$x_1 \leq y_1 + (1 - y_2) + y_5$$
$$x_2 \leq (1 - y_1) + (1 - y_2)$$
$$x_3 \leq y_2 + (1 - y_3) + y_4 + (1 - y_5)$$

Exercise Verify that the above 0-1 IP is correct by showing that

$$max(x_1 + x_2 + \ldots + x_m) = m \qquad \text{if and only if} \qquad F \text{ is satisfiable}$$