Lecture 17: March 26, 2013

1 Introduction

In the next three weeks or so, we will learn about integer programs. In particular:

1. We will model various problems as integer programming problems
2. We will explore various approaches for solving integer programming problems (namely: "Branch and Bound" and "Gomory cutting planes method")
3. We will develop other ideas that are useful for solving large-scale integer programming problem (for example, "constraint generation" and "column generation") which can also be useful for solving large-scale *linear programs*

Remark. In this course, whenever we say "integer program" what we mean is "integer *linear* program". That is: a *linear* program where *some* or *all* of its decision variables are restricted to integer values.

Example 1. A bakery bakes and sells one type of bread and one type of cake. One loaf of bread uses 3 eggs and 7 cups of flour, and one cake uses 2 eggs and 4 cups of flour. The bakery's daily supply consists of 15 eggs and 25 cups of flour. The profit for each loaf of bread is \$2 and the profit for each cake is \$5. Assume that these are the only two ingredients used in making the bread and the cake, and that everything that the bakery bakes will be sold.

How many loaves of bread and how many cakes should the bakery make so that its daily total profit is maximized? Note that the bakery cannot sell a fractional number of loaves of bread or a fractional number of cakes.

An integer (linear) programming formulation: Let x_1 = number of loaves of bread and x_2 = number of cakes that are produced in the bakery in one day.

$$\begin{aligned} \max \quad & 2x_1 + 5x_2 \\ s.t. \quad & 3x_1 + 2x_2 \leq 15 \\ & 7x_1 + 4x_2 \leq 25 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \text{ integers.} \end{aligned}$$

Note that if the bakery is allowed to sell fractional numbers of loaves of bread and cakes, then we can formulate this optimization problem as a linear program.

Example 2. The following example is NOT an integer linear program, because the objective function and the first constraint is not linear in terms of the decision variables.

$$\begin{aligned} \max \quad & 3x_1 + 4x_2^2 \\ s.t. \quad & 5x_1x_2 + x_2 \leq 10 \\ & -x_1 + 5x_2 \geq 8 \\ & x_1, x_2 \geq 0 \\ & x_1 \text{ integer.} \end{aligned}$$

It is very useful to be able to formulate problems like Example 1 as linear programs: where the problem is essentially a linear programming problem with integer feasible solutions. However, there are more interesting uses of integer programming formulation.

One such problem is the "Nonlinear Knapsack Problem" which we introduced a few weeks ago (see Lecture 14) as a problem that can be solved using a dynamic programming approach.

2 The nonlinear knapsack problem, revisited

Example 3. Consider a knapsack problem with a weight capacity of $W = 13$ pounds. There are 4 types of products, where each product is available in at most 3 units. The weight of each unit of products is as follows:

Product (k)	Weight per unit (w_k)
1	5
2	3
3	2
4	1

The weights are additive. That is, if we decide to take 2 units of product 1, then the total weight from product 1 is $x_1 \times w_1 = 2 \times 5 = 10$ pounds.

However, the values are not additive. The following table has the value if we take i units of product k, for each $i = 1, 2, 3$ and each $k = 1, 2, 3, 4$:

Product (k)	Value of i units of product k (v_{ki})		
	$i = 1$	$i = 2$	$i = 3$
1	9	15	20
2	4	8	12
3	3	4	5
4	0.5	2.5	5

(Note that one unit of product 1 has value 9, but two units of product 1 has value $15 \neq x_1 \times 9 = 2 \times 9$, etc.)

An integer programming formulation:

Step 1: First, we describe our decision variables.

(1) For each $k \in \{1, 2, 3, 4\}$, let x_k denote the number of units of product k that are taken.

(2) For each $k \in \{1, 2, 3, 4\}$ and each $i \in \{0, 1, 2, 3\}$, we have one decision variable:

$$\begin{aligned} y_{ki} &= \text{an indicator variable to indicate if } i \text{ units of product } k \text{ is taken} \\ &= \begin{cases} 1 & \text{if } i \text{ units of product } k \text{ is taken} \\ 0 & \text{if other number of units of product } k \text{ is taken} \end{cases} \end{aligned}$$

Step 2: Next, we describe our constraints.

- First, we need to make sure that the meaning of y_{ki} as described above is expressed correctly as a linear constraint. That is, we want a set of constraints that make sure that the values of $y_{k1}, y_{k2}, y_{k3}, y_{k4}$ corresponds correctly to x_k, for each product k.

  For instance, consider product $k = 1$. For product 1, we decide to take $x_1 = 0$, 1, 2, or 3 units of that product. So, among $y_{10}, y_{11}, y_{12}, y_{13}$, exactly one of them must take the value "1" while the rest must take the value "0", in such a way that if $x_1 = i$, then $y_{ki} = 1$.

  The following constraint express this condition:

$$\sum_{i=0}^{3} i y_{ki} = x_k, \quad \forall k \in \{1, 2, 3, 4\}.$$

- Next, we need the weight capacity constraint: at most 13 pounds can be put into the knapsack. The constraint can be written quite easily as follows:

$$\sum_{k=1}^{4} x_k w_k \leq 13.$$

Step 3: Lastly, we formulate our objective:

$$\max \sum_{k=1}^{4} \sum_{i=0}^{3} y_{ki} v_{ki}.$$

(Some explanation: To see that the objective function above really represents the total value of items that are taken:

$$\begin{aligned} \sum_{k=1}^{4} \sum_{i=0}^{3} y_{ki} v_{ki} &= \sum_{k=1}^{4} \left(\sum_{i=0}^{3} y_{ki} v_{ki} \right) \\ &= \underbrace{\left(\sum_{i=0}^{3} y_{1i} v_{1i} \right)}_{\text{value from product 1}} + \underbrace{\left(\sum_{i=0}^{3} y_{2i} v_{2i} \right)}_{\text{value from product 2}} + \underbrace{\left(\sum_{i=0}^{3} y_{3i} v_{3i} \right)}_{\text{value from product 3}} + \underbrace{\left(\sum_{i=0}^{3} y_{4i} v_{4i} \right)}_{\text{value from product 4}}. \end{aligned}$$

Hence, the objective function $\sum_{k=1}^{4} \sum_{i=0}^{3} y_{ki} v_{ki}$ does represent the total value of items that are taken.) Putting everything together, our integer program is:

$$\begin{array}{lll} \max & \sum_{k=1}^{4} \sum_{i=0}^{3} y_{ki} v_{ki} & \\ s.t. & \sum_{i=0}^{3} i y_{ki} = x_k, & \forall k \in \{1,2,3,4\} \\ & \sum_{k=1}^{4} x_k w_k \leq 13 & \\ & 0 \leq y_{ki} \leq 1, \text{ integers} & \forall k \in \{1,2,3,4\}, \forall i \in \{0,1,2,3\}. \end{array}$$

The corresponding AMPL model is:

```
param N; # number of items
param weightLimit = 13; # maximum weight that we can take
param itemLimit {k in 1..N}; # limit on number of copies of item k available
param value {k in 1..N, i in 0..itemLimit[k]);
param weight {k in 1..N} >= 0;

var x {k in 1..N} integer, >= 0, <= itemLimit[k];

var y {k in 1..N, i in 0..itemLimit[k]}, binary;

maximize Benefit:
 sum { k in 1..N, i in 0..itemLimit[k]} y[k, i]*benefit[k, i];

subject to Weight: sum{k in 1..N} weight[k] * x[k] <= weightLimit;

subject to Select {k in 1..N}: sum{i in 0..itemLimit[k]} y[k, i] = 1;

subject to Meaning {k in 1..N}:
x[k] = sum {i in 0..itemLimit[k]} y[k, i]*i;
```

3 Useful integer programming formulation tricks

In the nonlinear knapsack example above, the objective function is not a linear function of the number of units of the products that are taken. However, by using binary indicator decision variables, we were able to formulate the problem as an integer program.

3.1 Some simple integer programming constraints

1. Suppose that n activities are available, and we were to select exactly one of these activities. Let the binary decision variable x_i be an indicator of whether we select activity i. That is, if $x_i = 1$, we select activity i, but if $x_i = 0$, we don't. Hence, the constraint that force us to select exactly one out of the n available activities is:

$$x_1 + x_2 + \ldots + x_n = 1.$$

2. If instead, we want to select at least two of the activities, then we use the constraint:

$$x_1 + x_2 + \ldots + x_n \geq 2.$$

3. If we want a constraint to express that "activity 3 is possible only if both activities 1 and 2 are selected", then we add the following two constraints:

$$x_3 \leq x_1,$$

$$x_3 \leq x_2.$$

4. Suppose that you can only select activity 3 if at least one of activities 1 or 2 are selected:

$$x_3 \leq x_1 + x_2$$