# Integer Programming

Jianing Yao
Department of Operations Research
Rutgers University, the State University of New Jersey
Piscataway, NJ 08854 USA

November 30, 2014

This is a "short notes" on integer programming. The main effort is made to illustrate binary optimization problem, where I abstract the knapsack problem as the foundation of all variations. For general integer programming and mixed-integer programming, we can talk about in the class if needed. Hope this note helps.

## 1 Why Integer Programming

In real life, the amount of vegetables that you buy from a supermarket can be fractional values, the quantity of gas you "inject to" your cars can be fractional values, say 14 gallons, and so on so forth. They are chosen from the real line, $\mathbb{R}$, actually, in most of cases, there are further restricted to be non-negative, i.e., $\mathbb{R}_+$. However, they are certain amount of things that are strictly integer-valued, we denote it as $\mathbb{Z}$ (Likewise, we usually deal with non-negative integers, they have they own name, *natural number*, usually denoted as $\mathbb{N}$, i.e., $0, 1, 2, ...$). For example, you have a bag with certain capacity, you want to put all kinds of fruit in but not exceeding the limit. Obviously, you won't put quarter of apple, half of the banana. Of course, you can, but it's just not that reasonable. If this is not convincing, then imagine that you need to decide how many people assigned to one task, you won't argue to divided people into half, right ?

That's why we need to introduce so called *integer programming*, where we put **extra** restriction to some or all of the variables $x$ to be integer, $x \in \mathbb{Z}$. This type of constraints is called *integrality constraints*. **It changes the model to a nonlinear program instead of linear optimization**. But, don't be scared, nothing bad will happen, you don't need to start from scratch to learn nonlinear optimization, it's just a small modification. Because non-linearity only enters to those extra integrality constraints which is very easy to formulate, I will explain in a moment. Other than that, everything still comes out to be, the objective function is linear, the constraints are linear.

Now, let's list all possible types of decision variables.

- *Continuous-valued:* fractional values are allowed;

- *Binary valued:* the decision variable can either take 0 or 1, we usually write $x$ : binary, alternatively, $x \in \{0, 1\}$;
- *General integer valued:* decision variable can take value from all integers, $x \in \mathbb{Z}$.

The model we learned before can only take variable from the first category. The integer program can take any mixture of above categories. From the lecture slides and recitation slides, I summarized some typical problems that we can analyze.

## 2 General Integer Programming

Let's recall those happy days when we started this class, product mix table, process model. They are actually all problems of the following format:

$$\begin{aligned}
\min. \; & \sum_{i=1}^{n} c_i x_i \\
\text{subject to}: \; & \sum_{j=1}^{n} a_{1j} x_j \leq b_1, \\
& \sum_{j=1}^{n} a_{2j} x_j \leq b_2, \\
& \ldots\ldots \\
& \sum_{j=1}^{n} a_{nj} x_j \leq b_n.
\end{aligned} \tag{1}$$

$c_i$, $a_{ij}$, $b_j$ are known parameters. Don't argue with me there is maximization problem, inequality can be equality, inequality can be the other way around. In some way, you can convert them into this format, right? (I will explain in the class) The characteristic is, again, objective function and constraints are all linear. However, as discussed in *section 1*, we sometimes really want decision variable to be integer valued, $x_i \in \mathbb{Z}$. Thus, we need to put one extra conditions on above model,

$$\begin{aligned}
\min. \; & \sum_{i=1}^{n} c_i x_i \\
\text{subject to}: \; & \sum_{j=1}^{n} a_{1j} x_j \leq b_1, \\
& \sum_{j=1}^{n} a_{2j} x_j \leq b_2, \\
& \ldots\ldots \\
& \sum_{j=1}^{n} a_{nj} x_j \leq b_n, \\
& x_i \in \mathbb{Z}, i = 1, ..., n.
\end{aligned} \tag{2}$$

Just that simple, maybe the difficulty is the excel part, but the lecture slide is a good guide. Follow the steps, I don't think it is a real issue.

## 3 Binary Optimization Problem

### 3.1 $0-1$ Knapsack Problem

As the name indicates we only need to restrict the decision variables to be binary, 0 or 1. The most classic examples is called $0-1$ knapsack problem, the name existed in folklore, dates back to 1897, before it is defined as a mathematical problem. Officially, *Tobias Dantzig* suggested the name and carried out the research.

*Small digression, I feel so enforced to mention his son, George Dantzig, it will be a shame that you don't know this guy but you say to people you know linear programming. He is the father of operations research, developed simplex method for linear programming. That is the black box you always use in excel, remember, "checking simplex LP". He also made tremendous contributions to computer science, statistics and economics, e.t.c.. All in all, genius does what genius does.*

Sorry for the distraction, we shall continue. The simplest version of knapsack problem is the following:

**Example 3.1** a bag has certain capacity $C$ that is given along with different items (**one for each**) 1, 2, ..., $n$ that cannot be split , each of which has a selling price $p_1$, $p_2$, ..., $p_n$ and a volume $c_1$, $c_2$, ..., $c_n$, a vendor needs cleverly choosing items to put in the bag so that the profit can be maximized while maintaining undamaged.

We first sketch the integer program for this toy problem. Let's ignore the consideration of binary-valued. The vendor needs to determine how many of each items to put. Thus, we define the decision variable formally as:

$$x_i = \text{the number of item } i \text{ to put into the bag}, \ i = 1, ..., n$$

For the objective function, we need to maximize the profit, $x_i$ of item $i$ can bring profit $x_i \times p_i$. Therefore, the objective function is:

$$\max.\ p_1x_1 + p_2x_2 + \cdots + p_nx_n$$

or, simply,

$$\max.\ \sum_{i=1}^{n} p_ix_i$$

Speaking of the constraints, we have a simple one that is the total volume of items in the back cannot explode the bag, that is:

$$\left(\sum_{i=1}^{n} c_ix_i\right) = c_1x_1 + c_2x_2 + \cdots + c_nx_n \leq C,$$

Also, we have at most one of each item:

$$x_i \leq 1,\ i = 1, ..., n$$

Lastly, non-negative constraints: $x_i \geq 0$, $i = 1, ..., n$. Now, let's come back to the binary issue, in the problem statement, it is explicitly stated that items cannot be split. In other words, it can only be integer-valued. Since we have the constraints that $0 \leq x_i \leq 1$, $i = 1, ..., n$, the only two possibilities for integers are 0 and 1. Therefore, $x_i$ can either take 0 or 1. Combine the analysis above, we have the following model:

$$\begin{aligned} \max. \ & \sum_{i=1}^{n} p_i x_i \\ \text{subject to}: \ & \sum_{i=1}^{n} c_i x_i \leq C, \\ & x_i \in \{0,1\},\ i = 1, ..., n \end{aligned} \tag{3}$$

The last constraint is nonlinear, because it is certainly not in the format:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq (\ \text{,or} \geq)(\ \text{,or} =)\ b$$

Because the decision variables can only take binary value, actually the decision the vendor make can be either to put or not to put, we modify the decision variable a little bit:

$$x_i = 1,\ \text{if item } i \text{ put in the bag};\ x_i = 0,\ \text{otherwise}, i = 1, ..., n$$

Don't think it is a new thing, just an alternative to express the same idea. It's like, previously, the decision can be do, not do and do some, where do some includes all real numbers, they're uncountablely many, so we cannot list, like 0, 0.1, 0.111, .... Now, it can only be do or not do, so we can explicitly say what they are, 0 and 1, that's why we make such a small revision.

I can't say all but most of binary optimization problems are variations of $0-1$ knapsack problem. Remember the one you had in class, the investment on projects. You have a certain budget $C$, it's like the capacity of the knapsack. Different assets needs money to be invested on and they bring the net return, their analogy of volume of the items and profit of the items, respectively. The objective is to maximize the profit within the budget, so

$$\begin{aligned} \max. \ & 19,000x_1 + 21,000x_2 + 14,000x_3 + 10,000x_4 \\ \text{subject to:} \ & 7,000x_1 + 9,000x_2 + 6,000x_3 + 5,000x_4 \leq 21,000, \\ & x_1, x_2, x_3, x_4 \in \{0,1\} \end{aligned}$$

Compared to (3), exactly the same.

Sometimes, your knapsack may have other constraints, like, the tolerance of the weight $W$. It is very reasonable to assume that each item has a weight $w_i$, $i = 1, ..., n$, so if they are added up to be very heavy, the knapsack may also be destroyed. The model looks like:

$$\begin{aligned}
\max. \quad & \sum_{i=1}^{n} p_i x_i \\
\text{subject to}: \quad & \sum_{i=1}^{n} c_i x_i \le C, \\
& \sum_{i=1}^{n} w_i x_i \le W, \\
& x_i \in \{0,1\},\ i = 1, ..., n
\end{aligned} \tag{4}$$

Now, imagine you have a very vulnerable bag, then more constraints are required:

$$\begin{aligned}
\max. \quad & \sum_{i=1}^{n} p_i x_i \\
\text{subject to}: \quad & \sum_{j=1}^{n} a_{1j} x_j \le b_1, \\
& \sum_{j=1}^{n} a_{2j} x_j \le b_2, \\
& \cdots\cdots \\
& \sum_{j=1}^{n} a_{nj} x_j \le b_n, \\
& x_i \in \{0,1\},\ i = 1, ..., n
\end{aligned} \tag{5}$$

where $a_{ij}$ is the attributes $i$ of the item $j$, $b_j$ corresponds to the tolerance of the bag in terms of attributes $j$. Let's look at a concrete example:

**Example 3.2** You are given a group of possible investment projects for your companys capital. For each project, you are given the NPV(*Net Present Value*) the project would add to the firm, as well as the cash outflow required by each project during each year. Given the information in the table below (Table 1), determine the investments that maximize the firm's NPV. Note: the firm has \$30 million available during each of the next 5 years. All numbers are in millions of dollars.

The decision variables are

$$x_i = 1,\ \text{if the company invests in project } i = 1, ..., 12;\ x_i = 0, \text{ otherwise}$$

Table 1: Cash outflows in various years, NPV of projects

|  | Proj. 1 | Proj. 2 | Proj. 3 | Proj. 4 | Proj. 5 | Proj. 6 | Proj. 7 | Proj. 8 | Proj. 9 | Proj. 10 | Proj. 11 | Proj. 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Yr 1 | 1 | 3 | 4 | 6 | 5 | 4 | 2 | 0 | 1 | 3 | 9 | 8 |
| Yr 2 | 3 | 4 | 4 | 5 | 1 | 5 | 3 | 0 | 1 | 2 | 2 | 7 |
| Yr 3 | 4 | 2 | 3 | 3 | 2 | 2 | 1 | 3 | 4 | 4 | 4 | 1 |
| Yr 4 | 1 | 1 | 2 | 2 | 3 | 5 | 4 | 6 | 8 | 1 | 1 | 1 |
| Yr 5 | 1 | 2 | 1 | 3 | 8 | 5 | 6 | 7 | 3 | 6 | 1 | 1 |
| NPV | 20 | 25 | 30 | 35 | 40 | 42 | 31 | 33 | 35 | 37 | 38 | 39 |

The formulation is:

$$\begin{aligned}
\max. \quad & 20x_1 + 25x_2 + 30x_3 + \cdots + 39x_{12} \\
\text{subject to:} \quad & x_1 + 3_x2 + \cdots + 8x_{12} \leq 30 \\
& 3x_1 + 4x_2 + 5x_4 + \cdots + 7x_{12} \leq 30 \\
& \cdots\cdots \\
& x_1 + 2x_2 + 3x_3 + \cdots + x_{12} \leq 30 \\
& x_i \in \{0,1\},\ i = 1, ..., 12.
\end{aligned}$$

This is the "vulnerable" bag problem mentioned above, (5).

## 3.2 Binary Grid Problem

This kind of problem is actually built on $0-1$ knapsack problem as well. You may still remember the first day you encountered double subscripts in transportation problem, we now face the same thing but in the setting of binary problem. Imagine you now have multiple knapsacks, ordered as knapsack 1, knapsack 2, ..., knapsack $m$, they all have different capacities $C_1, C_2, ..., C_m$, again, you have different items (one for each) that cannot be split, but the knapsacks are magical that if you put different items to different bags, they have different volume $v_{ij}$ and can be sold at different prices $p_{ij}$, for example, item 1 in knapsack 3 has volume $v_{13} = 1$ and selling price $p_{31} = \$5$, your objective is to maximize the profit. It may sound a little bit artificial, but let's play along.

Obviously, no only you need to decide to put into bags or not, you also need to decide which bag to put. Thus, the decision variable is now as follows:

$$x_{ij} = 1,\ \text{if item } i \text{ put into knapsack } j,\ i = 1, ..., n,\ j = 1, ..., m;\ x_{ij} = 0, \text{ otherwise}$$

Let's postpone the objective function to think about a rather important issue. As stated in the problem, we have item one for each, this means that if we put item $i$ into one of the bag, say knapsack 1, we cannot put it into any other knapsacks. If you can just write down the mathematical formulation, excellent ! if you can't, don't worry, let's draw a network flow to illustrate. Whenever we have double subscripts decision variable, we can always appeal to the *bipartite graph* (Figure 1), if you still have it in your memory.

What's next ? The representative analysis, pick one of the node, say item $i$, what happened on this item is that it can be "shipped" to different knapsacks, but they are added

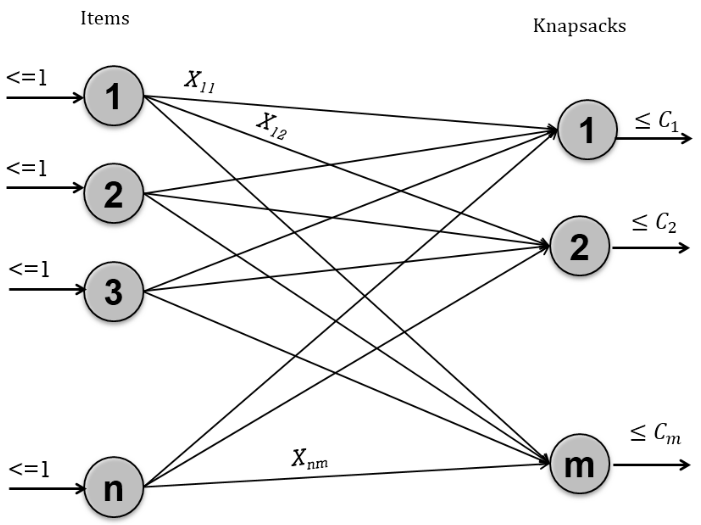

Figure 1: Binary Grid Problem

up to 1, because the poor factory only gives you one item to distribute. Now, you know how to formulate it,

$$\sum_{j=1}^{m} x_{ij} = x_{i1} + x_{i2} + \cdots + x_{im} \leq 1,\ i = 1, ..., n$$

Isn't that familiar to you, isn't it easy ? Let's continue, we didn't analyze the knapsack side. Pick one representative knapsack, knapsack $j$. It obtains different items, but every item has its volume $v_{ij}$ depending on which knapsack you choose. But all items piled up in knapsack $j$ can not exceed $C_j$, that is,

$$\sum_{i=1}^{n} v_{ij} x_{ij} = v_{1j} x_{1j} + \cdots + v_{nj} x_{nj} \leq C_j,\ j = 1, ..., m$$

How about the objective function, it is easy. Each arc means that you put item $i$ to knapsack $j$, thus it has a selling price $p_{ij}$, the total profit will be summing them up:

$$\max. \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij} x_{ij} = p_{11} x_{11} + p_{12} x_{12} + \cdots + p_{nm} x_{nm}$$

Note, there are many zeros in the end because of the way you put the constraints. Those "paths" makes no contribution to the objective function. Let's re-write the optimization problem:

$$\begin{aligned} \max . \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}x_{ij} \\ \text{subject to:} \quad & \sum_{i=1}^{n} v_{ij}x_{ij}, \ j=1,...,m \\ & x_{ij} \in \{0,1\}, \forall i,j \end{aligned} \tag{6}$$

"$\forall$" means for all, i.e., $i = 1, ..., n$, $j = 1, ..., m$.

**Remark 3.3** This is just analogy. There are too many variations that I can list them all, such as minimization instead of maximization, $v_{ij}$ is universal, $C_j$ is universal and so on so forth. Nevertheless, if you truly understand the above artificial example, you will have confidence to treat different versions.

Let's connect with the one you have in class–*MachinCo problem*. There the jobs are knapsacks and the machines are items, so to decide whether putting item $i$ to knapsack $j$ means machine $i$ performs job $j$, where $i = 1, ..., 5$, $j = 1, ..., 4$. And different machine performs different job incur different cost, $c_{ij}$, values are in that table, e.g., \$14, \$5, ..., \$8. Since they are cost, we need to minimize instead to maximize, but the idea is the same:

$$\min . \ 14x_{11} + 5x_{12} + \cdots + 8x_{54}$$

For the constraints, certainly, one machine can do one of the job, just like one item goes to one of the knapsack. Thus,

$$\sum_{j=1}^{4} x_{ij} \leq 1, \ i = 1, ..., 5$$

On the other hand, each job needs to be done, it is like each knapsack has capacity 1 and needs to be saturate by exactly one of the item, regardless its volume. Thus, we don't need to consider the volume of the item, $v_{ij} = 1$,

$$\sum_{i=1}^{5} x_{ij} = 1, \ j = 1, ..., 4$$

Therefore, the complete model reads:

$$\begin{aligned} \min . \ & 14x_{11} + 5x_{12} + \cdots + 8x_{54} \\ \text{subjet to:} \quad & \sum_{j=1}^{4} x_{ij} \leq 1, \ i = 1, ..., 5 \\ & \sum_{i=1}^{5} x_{ij} = 1, \ j = 1, ..., 4, \\ & x_{ij} \in \{0,1\}, \ i = 1, ..., 5, j = 1, ..., 4. \end{aligned}$$

One more example to practice your imagination:

Table 2: Boris Milkem

| | Year 1 | Year 2 | Year 3 |
|---|---|---|---|
| Asset 1 | 15 | 20 | 24 |
| Asset 2 | 16 | 18 | 21 |
| Asset 3 | 22 | 30 | 26 |
| Asset 4 | 10 | 20 | 30 |
| Asset 5 | 17 | 19 | 22 |
| Asset 6 | 19 | 25 | 29 |

**Example 3.4** Boris Milkem Owns six assets that he must sell over the next three years. He expects that the sales will generate the revenue in table 2 (millions of \$). For example, if he sells asset 3 in year 2, he receives \$30 million. Each asset can be sold at most once, of course. He must sell at least \$20 million of assets in year 1, \$30 millions in year 2 and \$35 million in year 3. How can he maximize his total revenue from the sales ?

The integer program looks like the following:

$$x_{ij} = 1, \text{ if we sell aasset } i = 1, ..., 6 \text{ in year } 1, 2, 3;\ x_{ij} = 0, \text{ otherwise.}$$

$$\begin{aligned}
\max.\ & 15x_{11} + 20x_{12} + \cdots + 29x_{36} \\
\text{subject to: } & \sum_{j=1}^{3} x_{ij} = 1,\ i = 1, ..., 6, \\
& 15x_{11} + 16x_{21} + 20x_{31} + 10x_{41} + 17x_{51} + 19x_{61} \geq 20, \\
& 20x_{12} + 18x_{22} + 30x_{32} + 20x_{42} + 19x_{52} + 25x_{62} \geq 30, \\
& 24x_{13} + 21x_{32} + 26x_{33} + 30x_{43} + 22x_{53} + 29x_{63} \geq 35 \\
& x_{ij} \in \{0, 1\}, i = 1, ..., 6, j = 1, ..., 3.
\end{aligned}$$

The difference is from the 'item side', we have to put the item into some knapsack, that's why we have equality constraints. From the knapsack side, we have "$\geq$" inequality and the interpretation of the volume becomes selling price. But still, it is a variation of (6).

## 4 Mixed-integer Program & Excel

The only material I have for this topic is the example you have in lecture slide. Maybe in the class, if you need, I can go over that example again. Or if we have time, we shall talk a little bit about excel in the last recitation.