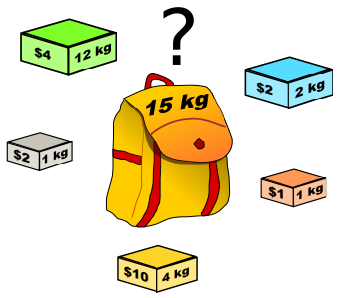
0.1 Knapsack Problem

Knapsack problem can take different forms depending on if the variables are binary or integer. The binary version means that there is only one item of each item type that can be taken.

Binary Knapsack Problem:
NP-Complete
Given an non-negative weight vector $a \in \mathbb{Q}^n_+$, a capacity $b \in \mathbb{Q}_+$, and objective coefficients $c \in \mathbb{Q}^n$,

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & a^\top x \leq b \\ & x \in \{0,1\}^n \end{aligned} \tag{0.1.1}$$



[1]

Figure 1: *Knapsack Problem: which items should we choose take in the knapsack that maximizes the value while respecting the 15kg weight limit?*

Example: Knapsack [Code: **??**]
You have a knapsack (bag) that can only hold W = 15 kgs. There are 5 items that you could possibly put into your knapsack. The items (weight, value) are given as: (12 kg, $4), (2 kg, $2), (1kg, $2), (1kg, $1), (4kg, $10). Which items should you take to maximize your value in the knapsack?

Variables:

[1] https://en.wikipedia.org/wiki/Knapsack_problem

- let $x_i = 0$ if item i is in the bag
- let $x_i = 1$ if item i is not in the bag

Model:

$$\begin{aligned} \max \quad & 4x_1 + 2x_2 + 2x_3 + 1x_4 + 10x_5 \\ \text{s.t.} \quad & 12x_1 + 2x_2 + 1x_3 + 1x_4 + 4x_5 \leq 15 \\ & x_i \in \{0,1\} \text{ for } i = 1,\ldots,5 \end{aligned} \tag{0.1.2}$$

In the integer case, we typically require the variables to be non-negative integers, hence we use the notation $x \in \mathbb{Z}^n_+$. This setting reflects the fact that instead of single individual items, you have item types of which you can take as many of each type as you like that meets the constraint.

Integer Knapsack Problem:
NP-Complete
Given an non-negative weight vector $a \in \mathbb{Q}^n_+$, a capacity $b \in \mathbb{Q}_+$, and objective coefficients $c \in \mathbb{Q}^n$,

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & a^\top x \leq b \\ & x \in \mathbb{Z}^n_+ \end{aligned} \tag{0.1.3}$$

We can also consider an equality constrained version

Equality Constrained Integer Knapsack Problem:
NP-Hard
Given an non-negative weight vector $a \in \mathbb{Q}^n_+$, a capacity $b \in \mathbb{Q}_+$, and objective coefficients $c \in \mathbb{Q}^n$,

$$\max \quad c^\top x \tag{0.1.4}$$
$$\text{s.t.} \quad a^\top x = b \tag{0.1.5}$$
$$x \in \mathbb{Z}^n_+ \tag{0.1.6}$$

Example 2: Using pennies, nickels, dimes, and quarters, how can you minimize the number of coins you need to to make up a sum of 83¢?

Variables:

- Let p be the number of pennies used
- Let n be the number of nickels used
- Let d be the number of dimes used
- Let q be the number of quarters used

Model

$$\begin{array}{lll} \min & p+n+d+q & \text{total number of coins used} \\ \text{s.t.} & p+5n+10d+25q=83 & \text{sums to } 83¢ \\ & p,d,n,q \in \mathbb{Z}_+ & \text{each is a non-negative integer} \end{array}$$

0.2 Set Covering

Set Covering:

NP-Complete

Given a set V with subsets $V_1,\ldots,V_l$, determine the smallest subset $S \subseteq V$ such that $S \cap V_i \neq \emptyset$ for all $i = 1,\ldots,l$.

The set cover problem can be modeled as

$$\begin{array}{ll} \min & \mathbb{1}^\top x \\ \text{s.t.} & \sum_{v \in V_i} x_v \geq 1 \text{ for all } i = 1,\ldots,l \\ & x_v \in \{0,1\} \text{ for all } v \in V \end{array} \tag{0.2.1}$$

where x_v is a 0/1 variable that takes the value 1 if we include item j in set S and 0 if we do not include it in the set S.

Vertex Cover:

NP-Complete

Given a graph $G = (V,E)$ of vertices and edges, we want to find a smallest size subset $S \subseteq V$ such that every for every $e = (v,u) \in E$, either u or v is in S.

We can write this as a mathematical program in the form:

$$\begin{aligned} \min \quad & \mathbb{1}^\top x \\ \text{s.t.} \quad & x_u + x_v \geq 1 \text{ for all } (u,v) \in E \\ & x_v \in \{0,1\} \text{ for all } v \in V. \end{aligned} \tag{0.2.2}$$

Example: Vertex Cover [Code: **??**]

Consider the graph with nodes

$$V = \text{["You", "Ginger", "Juan", "Jameis", "Bob", "Geoff", "Jane"]} \tag{0.2.3}$$

and edges

$$E = \text{[["You", "Ginger"], ["You", "Juan"], ["You", "Jameis"], ["You", "Bob"],} \tag{0.2.4}$$
$$\text{["You", "Jane"], ["You", "Geoff"], ["Ginger", "Jameis"], ["Jameis", "Bob"],} \tag{0.2.5}$$
$$\text{["Bob", "Jane"], ["Bob", "Geoff"], ["Geoff", "Jane"]]} \tag{0.2.6}$$

This problem can be modeled as

$$\begin{aligned} \min \quad & x_{\text{You}} + x_{\text{Ginger}} + x_{\text{Juan}} + x_{\text{Jameis}} + x_{\text{Bob}} + x_{\text{Geoff}} + x_{\text{Jane}} \\ \text{Subject to} \quad & x_{\text{You}} + x_{\text{Ginger}} \geq 1 \\ & x_{\text{You}} + x_{\text{Juan}} \geq 1 \\ & x_{\text{You}} + x_{\text{Jameis}} \geq 1 \\ & x_{\text{You}} + x_{\text{Bob}} \geq 1 \\ & x_{\text{You}} + x_{\text{Jane}} \geq 1 \\ & x_{\text{You}} + x_{\text{Geoff}} \geq 1 \\ & x_{\text{Ginger}} + x_{\text{Jameis}} \geq 1 \\ & x_{\text{Jameis}} + x_{\text{Bob}} \geq 1 \\ & x_{\text{Bob}} + x_{\text{Jane}} \geq 1 \\ & x_{\text{Bob}} + x_{\text{Geoff}} \geq 1 \\ & x_{\text{Geoff}} + x_{\text{Jane}} \geq 1 \\ & x_i \in \{0,1\} \quad \forall i \in \{\text{You,Ginger,}\ldots\text{,Geoff,Jane}\} \end{aligned}$$

Example: Southwestern Airways[2] [Code: **??**]

Southwestern Airways needs to assign its crews to cover all its upcoming flights. We will focus on the problem of assigning three crews based in San Francisco to the flights listed in the first column of the below table. The other 12 columns show feasible sequences of flights for a crew. (The numbers in each column indicate the order of the flights: 1 = first stop, 2 = second stop, 3 = third stop...etc.) Exactly three of the sequences need to be chosen (one per crew) in such a way that every flight is covered. (It is permissible to have more than one crew on a flight, where extra crews would fly as passengers, but union contracts require that the extra crews would still need to be paid for their time as if they were working.) The cost of assigning a crew to a particular sequence of flights is given (in thousands of dollars) in the bottom row of the table. The objective is to minimize the total cost of the three crew assignments that cover all the flights.

	Feasible Sequence of Flights											
Flight	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1. San Francisco to Los Angeles	1			1			1			1		
2. San Francisco to Denver		1			1			1			1	
3. San Francisco to Seattle			1			1			1			1
4. Los Angeles to Chicago				2			2		3	2		3
5. Los Angeles to San Francisco	2					3				5	5	
6. Chicago to Denver				3	3				4			
7. Chicago to Seattle							3	3		3	3	4
8. Denver to San Francisco		2		4	4				5			
9. Denver to Chicago					2			2			2	
10. Seattle to San Francisco			2				4	4				5
11. Seattle to Los Angeles						2			2	4	4	2
Cost, $1,000's	2	3	4	6	7	5	7	8	9	9	8	9

[3]

Sets Let $I = \{1, \ldots, 12\}$ denote flight sequences Let $F = \{1, \ldots, 11\}$ denote the flights. For each flight i, let $F_i \subseteq I$ be the set of flight sequences that intersects flight i.

Variables Let $x_i = 1$ if flight sequence i is chosen and 0 otherwise

Model The model is to minimize cost while respecting that flights are covered. This can be modeled as

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & \sum_{j \in F_i} x_j \geq 1 && \text{for all } i = 1, \ldots, 11 \\ & x_j \in \{0, 1\} && \text{for all } j = 1, \ldots, 12 \end{aligned}$$

Plugging in the data creates the following integer program:

$$
\begin{aligned}
\min \quad & 2x_1 + 3x_2 + 4x_3 + 6x_4 + 7x_5 + 5x_6 + 7x_7 + 8x_8 + 9x_9 + 9x_{10} + 8x_{11} + 9x_{12} \\
\text{Subject to} \quad & x_1 + x_4 + x_7 + x_{10} \geq 1 \\
& x_2 + x_5 + x_8 + x_{11} \geq 1 \\
& x_3 + x_6 + x_9 + x_{12} \geq 1 \\
& x_4 + x_7 + x_9 + x_{10} + x_{12} \geq 1 \\
& x_1 + x_6 + x_{10} + x_{11} \geq 1 \\
& x_4 + x_5 + x_9 \geq 1 \\
& x_7 + x_8 + x_{10} + x_{11} + x_{12} \geq 1 \\
& x_2 + x_4 + x_5 + x_9 \geq 1 \\
& x_5 + x_8 + x_{11} \geq 1 \\
& x_3 + x_7 + x_8 + x_{12} \geq 1 \\
& x_6 + x_9 + x_{10} + x_{11} + x_{12} \geq 1 \\
& x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} = 3 \\
& x_i \in \{0,1\} \quad \forall i \in \{1,2,\ldots,11,12\}
\end{aligned}
$$

Set Covering - Matrix description:

NP-Complete

Given a non-negative matrix $A \in \{0,1\}^{m \times n}$, a non-negative vector, and an objective vector $c \in \mathbb{R}^n$, the set cover problem is

$$
\begin{aligned}
\max \quad & c^\top x \\
\text{s.t..} \quad & Ax \geq \mathbb{1} \\
& x \in \{0,1\}^n.
\end{aligned}
\tag{0.2.7}
$$

Example: Vertex Cover with matrix [Code: **??**]

An alternate way to solve **??** is to define the adjacency matrix A of the graph. The adjacency matrix is a $|E| \times |V|$ matrix with $\{0,1\}$ entries. The each row corresponds to an edge e and each column corresponds to a node v. For an edge $e = (u,v)$, the corresponding row has a 1 in columns corresponding to the nodes u and v, and a 0 everywhere else. Hence, there are exactly two 1's per row. Applying the formulation above in **??** solves the problem.

[3]This example was taken from Hillier and Lieberman

[3]This image table was taken from Hillier and Lieberman

General Set Covering

We could also allow for a more general type of set covering where we have non-negative integer variables and a right hand side that has values other than 1.

Set Covering - Generalized:
NP-Complete
Given a non-negative matrix $A \in \mathbb{Z}_+^{m\times n}$, a non-negative vector $b \in \mathbb{Z}^m$, and an objective vector $c \in \mathbb{R}^n$, the set cover problem is

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t..} \quad & Ax \geq b \\ & x \in \mathbb{Z}_+^n. \end{aligned} \tag{0.2.8}$$

Example 6: Nurse Scheduling— Adapted from "Applied Integer Programming". exercise 2.6 and 2.7

Nurses in large hospitals usually work 3 days a week. Daily demand for nurses is summarized in Table 2.4. Determine the number of nurses required per schedule type so that the total wage cost is minimized.

1. Use the numbers (not symbols) in the table to model this problem instance.
2. Solve this instance and submit the code.
3. Let S denote the set of schedule types. How many variables of each type (continuous, binary, integer) does your model have in terms of the number of schedules $|S|$?
4. If part-time nurses are hired at the rate of \$175/day, formulate the problem to minimize the total cost.
5. If part-time nurses must be accompanied by at least three full-time nurses, how would you formulate this constraint?

- Let
$$x_i = \text{The number of nurses to assign to schedule } i \;\forall\; i \in \{1,2,...,5\}.$$

The model is

$$\text{Minimize} \quad 525x_1 + 470x_2 + 550x_3 + 500x_4 + 425x_5$$

	Schedule Type					
Day	1	2	3	4	5	Nurses Required
Monday	X		X			20
Tuesday	X				X	25
Wednesday			X	X		26
Thursday		X			X	26
Friday		X		X	X	30
Saturday		X		X		30
Sunday	X		X			35
Weekly wage	525	470	550	500	425	

$$
\begin{aligned}
\text{s.t.} \quad & x_1 + x_3 \geq 20 \\
& x_1 + x_5 \geq 25 \\
& x_3 + x_4 \geq 26 \\
& x_2 + x_5 \geq 26 \\
& x_2 + x_4 + x_5 \geq 30 \\
& x_2 + x_4 \geq 30 \\
& x_1 + x_3 \geq 35 \\
& x_i \in \mathbb{Z}_+ \qquad \forall\, i \in \{1, 2, ..., 5\}
\end{aligned}
$$

The table tells us which schedule types contribute to each day's staff; i.e. Mondays are staffed entirely by nurses assigned to schedule types 1 and 3, Tuesdays by those assigned to type 1 and 5, and so on. At least 20 nurses are required to be on duty on Mondays; so the number of nurses assigned to schedules 1 and 3 should be at least 20. That is,

$$x_1 + x_3 \geq 20.$$

Extending this logic to each day gives us all seven constraints.

Alternately, by replacing the X's in the table with 1's we are left with a constraint matrix S. Let $\vec{x}^\top = [x_1, x_2, x_3, x_4, x_5]$ be the decision variables and $\vec{r}^\top = [20, 25, 26, 26, 30, 30, 35]$ describe the nurses required for each day, so that

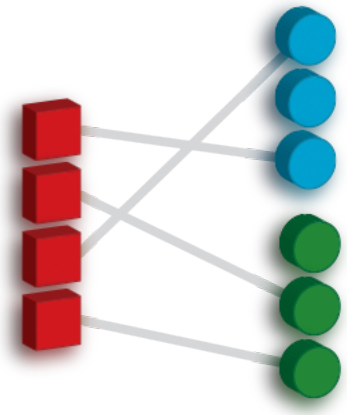
$$S\ \vec{x} \geq \vec{r}$$

describes all seven constraints.

0.3 Machine Assignment

Example Model: Machine assignment

- Given m machines and n jobs, find a least cost assignment of jobs to machines not exceeding the machine capacities.
- Each job j requires a_{ij} units of machine i's capacity b_i.
- Cost of assigning job j to machine i is c_{ij}.



Here is a model.... **To be added....**

0.4 Facility Location

https://en.wikipedia.org/wiki/Facility_location_problem

Integer programming formulations

Facility location problems are often solved as integer programs. In this context, facility location problems are often posed as follows: suppose there are n facilities and m customers. We wish to choose (1) which of the n facilities to open, and (2) which (open) facilities to use to supply the m customers, in order to satisfy some fixed demand at minimum cost. We introduce the following notation: let f_i denote the (fixed) cost of opening facility i, for $i = 1, \ldots, n$. Let c_{ij} denote the cost to ship a product from facility i to customer j for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Let d_j denote the demand of customer j for $j = 1, \ldots, m$. Further suppose that each facility has a maximum output. Let u_i denote the maximum amount of product that can be produced by facility i, that is, let u_i denote the *capacity* of facility i. The remainder of this section follows[14]

Capacitated facility location

In our initial formulation, introduce a binary variable x_i for $i = 1, \ldots, n$, where $x_i = 1$ if facility i is open, and $x_i = 0$ otherwise. Further introduce the variable y_{ij} for $i = 1, \ldots, n$ and $j = 1, \ldots, m$ which represents the fraction of the demand d_j filled by facility i. The so-called **capacitated facility location problem** is then given by

0.4.1 Capacitated Facility Location

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} y_{ij} + \sum_{i=1}^{n} f_i x_i \\
\text{s.t.} \quad & \sum_{i=1}^{n} y_{ij} = 1 \text{ for all } j = 1, \ldots, m \\
& \sum_{j=1}^{m} d_j y_{ij} \leqslant u_i x_i \text{ for all } i = 1 \ldots, n \\
& y_{ij} \geqslant 0 \text{ for all } i = 1, \ldots, n \text{ and } j = 1, \ldots, m \\
& x_i \in \{0, 1\} \text{ for all } i = 1, \ldots, n
\end{aligned}
\tag{0.4.1}
$$

0.4.2 Uncapacitated Facility Location

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}z_{ij} + \sum_{i=1}^{n} f_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} z_{ij} = 1 \text{ for all } j = 1,\dots,m \\ & \sum_{j=1}^{m} z_{ij} \leqslant M x_i \text{ for all } i = 1\dots,n \\ & z_{ij} \in \{0,1\} \text{ for all } i = 1,\dots,n \text{ and } j = 1,\dots,m \\ & x_i \in \{0,1\} \text{ for all } i = 1,\dots,n \end{aligned} \tag{0.4.2}$$

0.5 Capital Budgeting

A firm has n projects it could undertake to maximize revenue, but budget limitations require that not all can be completed.
Project j expects to produce revenue c_j
Project j requires investment a_{ij} in time period i for $i = 1,\dots,m$
In time period i, capital b_i is available

Let x_i be a binary variable such that $x_i = 1$ if we choose investment i and $x_i = 0$ otherwise. The the model can be given as:

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j \\ s.t. \quad & \sum_{j=1}^{n} a_{ij}x_j \leq b_i, \quad i = 1,\dots m \\ & x_j \in \{0,1\},\ j = 1,\dots,n \end{aligned}$$

Consider the example given in the following table.

Project	$\mathbb{E}$[Revenue]	Resources required in week 1	Resources required in week 2
1	10	3	4
2	8	1	2
3	6	2	1
Resources available		5	6

$$\max \quad 10x_1 + 8x_2 + 6x_3$$

subject to

$$\begin{aligned} 3x_1 + 1x_2 + 2x_3 &\leq 5 \\ 4x_1 + 2x_2 + 1x_3 &\leq 6 \\ x_j \in \{0,1\},\ j &= 1,2,3 \end{aligned}$$

0.6 Network Flow

0.7 Transportation Problem

https://www.youtube.com/watch?v=Jr7LI-sUEmo

0.7.1 Modeling Tricks

In this section, we describe ways to model a variety of constraints that commonly appear in practice. The goal is changing constraints described in words to constraints defined by math.

0.7.2 Either Or Constraints

"At least one of these constraints holds" is what we would like to model. Equivalently, we can phrase this as an *inclusive or* constraint.

Either Or:

$$\text{Either} \quad a^\top x \leq b \quad \text{or} \quad c^\top x \leq d \quad \text{holds} \tag{0.7.1}$$

can be modeled as

$$\begin{aligned} a^\top x - b &\leq M_1 \delta \\ c^\top x - d &\leq M_2(1-\delta) \\ \delta &\in \{0,1\}, \end{aligned} \tag{0.7.2}$$

where M_1 is an upper bound on $a^\top x - b$ and M_2 is an upper bound on $c^\top x - d$.

Example 7: Either 2 buses or 10 cars are needed shuttle students to the football game.

- Let x be the number of buses we have and
- let y be the number of cars that we have.

Suppose that there are at most $M_1 = 5$ buses that could be rented an at most $M_2 = 20$ cars that could be available.

This constraint can be modeled as

$$\begin{aligned} x - 2 &\leq 5\delta \\ y - 10 &\leq 20(1-\delta) \\ \delta &\in \{0,1\}, \end{aligned} \tag{0.7.3}$$

If then implications

If then implications are extremely useful in models. For instance, if we have more than 5 passengers, then we need to take two cars. Most if then statements can be modeled with by a constraint and an on/off flag. For example

$$\text{If} \quad \delta = 1, \text{ then} \quad a^\top x \leq b. \tag{0.7.4}$$

By letting M be an upper bound on the quantity $a^\top x - b$, we can model this condition as

$$\begin{gathered} a^\top x - b \leq M(1-\delta) \\ \delta \in \{0,1\} \end{gathered} \tag{0.7.5}$$

On the other hand, if we want to model the reverse implication, we have to be slightly more careful. We let m be a lower bound on the quantity $a^\top x - b$ and we let ϵ be a tiny number that is an error bound in verifying if an inequality is violated. **If the data a, b are integer and x is an integer, then we can take $\epsilon = 1$.**

Now

$$\text{If} \quad a^\top x \leq b \quad \text{then} \quad \delta = 1 \tag{0.7.6}$$

can be modeled as

$$a^\top x - b \geq \epsilon(1-\delta) + m\delta. \tag{0.7.7}$$

A simple way to understand this constraint is to consider the *contrapositive* of the if then statement that we want to model. The contrapositive says that

$$\text{If } \delta = 0, \text{ then } a^\top x - b > 0. \tag{0.7.8}$$

To show the contrapositive, we set $\delta = 0$. Then the inequality becomes

$$a^\top x - b \geq \epsilon(1-0) + m0 = \epsilon > 0.$$

Thus, the contrapositive holds.

If instead we wanted a direct proof:

Case 1: Suppose $a^\top x \leq b$. Then $0 \geq a^\top x - b$, which implies that

$$\delta(a^\top x - b) \geq a^\top x - b$$

Therefore

$$\delta(a^\top x - b) \geq \epsilon(1-\delta) + m\delta$$

After rearranging

$$\delta(a^\top x - b - m) \geq \epsilon(1-\delta)$$

Since $a^\top x - b - m \geq 0$ and $\epsilon > 0$, the only feasible choice is $\delta = 1$.

Case 2: Suppose $a^\top x > b$. Then $a^\top x - b \geq \epsilon$. Since $a^\top x - b \geq m$, both choices $\delta = 0$ and $\delta = 1$ are feasible.

By the choice of ϵ, we know that $a^\top x - b > 0$ implies that $a^\top x - b \geq \epsilon$.

—-

Since we don't like strict inequalities, we write the strict inequality as $a^\top x - b \geq \epsilon$ where ϵ is a small positive number that is a smallest difference between $a^\top x - b$ and 0 that we would typically observe. As mentioned above, if a, b, x are all integer, then we can use $\epsilon = 1$.

Now we want an inequality with left hand side $a^\top x - b \geq$ and right hand side to take the value

Logic statement	Constraint
if $z = 0$ then $a^T x \leq b$	$a^T x - b \leq Mz$
if $z = 0$ then $a^T x \geq b$	$a^T x - b \geq mz$
if $z = 1$ then $a^T x \leq b$	$a^T x - b \leq M(1 - z)$
if $z = 1$ then $a^T x \geq b$	$a^T x - b \geq m(1 - z)$
if $a^T x \leq b$ then $z = 1$	$a^T x - b \geq mz + \varepsilon(1 - z)$
if $a^T x \geq b$ then $z = 1$	$a^T x - b \leq Mz - \varepsilon(1 - z)$
if $a^T x \leq b$ then $z = 0$	$a^T x - b \geq m(1 - z) + \varepsilon z$
if $a^T x \geq b$ then $z = 0$	$a^T x - b \leq M(1 - z) - \varepsilon z$

Where M and m are upper and lower bounds on $a^T x - b$.

[4]

Table 1: *If/then models with a constraint and a binary variable.*

- ϵ if $\delta = 0$,
- m if $\delta = 1$.

This is accomplished with right hand side $\epsilon(1 - \delta) + m\delta$.

Many other combinations of if then statements are summarized in the following table:

0.7.3 Binary reformulation of integer variables

If an integer variable has small upper and lower bounds, it can sometimes be advantageous to recast it as a sequence of binary variables - for either modeling, the solver, or both. Although there are technically many ways to do this, here are the two most common ways.

Full reformulation:
u many binary variables
For a non-negative integer variable x with upper bound u, modeled as

$$0 \leq x \leq u, \quad x \in \mathbb{Z}, \tag{0.7.9}$$

[4] This table was taken from notes by Laurent Lassard.

this can be reformulated with u binary variables $z_1, \ldots, z_u$ as

$$\begin{aligned} x &= \sum_{i=1}^{u} i z_i = z_1 + 2z_2 + \cdots + u z_u \\ 1 &\geq \sum_{i=1}^{u} z_i = z_1 + z_2 + \cdots + z_u \\ z_i &\in \{0,1\} \quad \text{for } i = 1, \ldots, u \end{aligned} \tag{0.7.10}$$

We call this the *full reformulation* because there is a binary variable z_i associated with every value i that x could take. That is, if $z_3 = 1$, then the second constraint forces $z_i = 0$ for all $i \neq 3$ (that is, z_3 is the only non-zero binary variable), and hence by the first constraint, $x = 3$.

Full reformulation:
$O(\log u)$ many binary variables
For a non-negative integer variable x with upper bound u, modeled as

$$0 \leq x \leq u, \quad x \in \mathbb{Z}, \tag{0.7.11}$$

this can be reformulated with u binary variables $z_1, \ldots, z_{\log(\lfloor u \rfloor)+1}$ as

$$\begin{aligned} x &= \sum_{i=0}^{\log(\lfloor u \rfloor)+1} 2^i z_i = z_0 + 2z_1 + 4z_2 + 8z_3 + \cdots + 2^{\log(\lfloor u \rfloor)+1} z_{\log(\lfloor u \rfloor)+1} \\ z_i &\in \{0,1\} \quad \text{for } i = 1, \ldots, \log(\lfloor u \rfloor) + 1 \end{aligned} \tag{0.7.12}$$

We call this the *log reformulation* because this requires only logarithmically many binary variables in terms of the upper bound u. This reformulation is particularly better than the full reformulation when the upper bound u is a “larger" number, although we will leave it ambiguous as to how larger a number need to be in order to be described as a “larger" number.

0.7.4 SOS1 Constraints

Definition 1. *A Special Ordered Sets of type 1 (SOS1) constraint on a vector indicates that* at most one element of the vector can non-zero.

We next give an example of how to use binary variables to model this and then show how much simpler it can be coded using the SOS1 constraint.

Example: SOS1 Constraints [Code: **??**]
Solve the following optimization problem:

$$\begin{aligned} \text{maximize} \quad & 3x_1 + 4x_2 + x_3 + 5x_4 \\ \text{subject to} \quad & 0 \leq x_i \leq 5 \\ & \text{at most one of the } x_i \text{ can be nonzero} \end{aligned}$$

0.7.5 SOS2 Constraints

Definition 2. *A Special Ordered Sets of type 2 (SOS2) constraint on a vector indicates that* at most two elements of the vector can non-zero AND the non-zero elements must appear consecutively.

We next give an example of how to use binary variables to model this and then show how much simpler it can be coded using the SOS2 constraint.

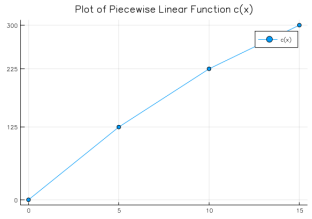
Example: SOS2 [Code: **??**]
Solve the following optimization problem:

$$\begin{aligned} \text{maximize} \quad & 3x_1 + 4x_2 + x_3 + 5x_4 \\ \text{subject to} \quad & 0 \leq x_i \leq 5 \\ & \text{at most two of the } x_i \text{ can be nonzero} \\ & \text{and the nonzero } x_i \text{ must be consecutive} \end{aligned}$$

0.7.6 Piecewise linear functions with SOS2 constraint

Example: Piecewise Linear Function [Code: **??**]
Consider the piecewise linear function $c(x)$ given by

$$c(x) = \begin{cases} 25x & \text{if } 0 \leq x \leq 5 \\ 20x + 25 & \text{if } 5 \leq x \leq 10 \\ 15x + 75 & \text{if } 10 \leq x \leq 15 \end{cases}$$



We will use integer programming to describe this function. We will fix $x = a$ and then the integer program will set the value y to $c(a)$.

$$
\begin{aligned}
\min \quad & 0 \\
\text{Subject to} \quad & x - 5z_2 - 10z_3 - 15z_4 = 0 \\
& y - 125z_2 - 225z_3 - 300z_4 = 0 \\
& z_1 + z_2 + z_3 + z_4 = 1 \\
& z_1 - w_1 \leq 0 \\
& z_2 - w_2 \leq 0 \\
& z_3 - w_3 \leq 0 \\
& z_4 - w_4 \leq 0 \\
& SOS2 : \{w_1, w_2, w_3, w_4\} \\
& 0 \leq z_i \leq 1 \quad \forall i \in \{1,2,3,4\} \\
& w_i \in \{0,1\} \quad \forall i \in \{1,2,3,4\} \\
& x = a
\end{aligned}
$$

Example: Piecewise Linear Function Application [Code: **??**]

Consider the following optimization problem where the objective function includes the term $c(x)$, where $c(x)$ is the piecewise linear function described in **??**:

$$
\begin{aligned}
\max \quad & z = 12x_{11} + 12x_{21} + 14x_{12} + 14x_{22} - c(x) \\
\text{s.t.} \quad & x_{11} + x_{12} \leq x + 5 \\
& x_{21} + x_{22} \leq 10 \\
& 0.5x_{11} - 0.5x_{21} \geq 0 \\
& 0.4x_{12} - 0.6x_{22} \geq 0 \\
& x_{ij} \geq 0 \\
& 0 \leq x \leq 15
\end{aligned}
$$

Given the piecewise linear, we can model the whole problem explicitly as a mixed-integer linear program.

$$
\begin{aligned}
\max \quad & 12X_{1,1} + 12X_{2,1} + 14X_{1,2} + 14X_{2,2} - y \\
\text{Subject to} \quad & x - 5z_2 - 10z_3 - 15z_4 = 0 \\
& y - 125z_2 - 225z_3 - 300z_4 = 0 \\
& z_1 + z_2 + z_3 + z_4 = 1 \\
& z_1 - w_1 \leq 0 \\
& z_2 - w_2 \leq 0 \\
& z_3 - w_3 \leq 0 \\
& z_4 - w_4 \leq 0 \\
& X_{1,1} + X_{1,2} - x \leq 5 \\
& X_{2,1} + X_{2,2} \leq 10 \\
& 0.5X_{1,1} - 0.5X_{2,1} \geq 0 \\
& 0.4X_{1,2} - 0.6X_{2,2} \geq 0 \\
& SOS2 : \{w_1, w_2, w_3, w_4\} \\
& X_{i,j} \geq 0 \quad \forall i \in \{1,2\}, j \in \{1,2\} \\
& 0 \leq z_i \leq 1 \quad \forall i \in \{1,2,3,4\} \\
& w_i \in \{0,1\} \quad \forall i \in \{1,2,3,4\} \\
& 0 \leq x \leq 15 \\
& y
\end{aligned}
$$

0.7.7 Maximizing a minimum

When the constraints could be general, we will write $x \in X$ to define general constraints. For instance, we could have $X = \{x \in \mathbb{R}^n : Ax \leq b\}$ of $X = \{x \in \mathbb{R}^n : Ax \leq b, x \in \mathbb{Z}^n\}$ or many other possibilities.

Consider the problem

$$
\begin{aligned}
\max \quad & \min\{x_1, \ldots, x_n\} \\
\text{such that} \quad & x \in X
\end{aligned}
$$

Having the minimum on the inside is inconvenient. To remove this, we just define a new variable y and enforce that $y \leq x_i$ and then we maximize y. Since we are maximizing y, it will take the value of the smallest x_i. Thus, we can recast the problem as

$$
\begin{aligned}
\max \quad & y \\
\text{such that} \quad & y \leq x_i \text{ for } i = 1, \ldots, n \\
& x \in X
\end{aligned}
$$

0.7.8 Relaxing (nonlinear) equality constraints

There are a number of scenarios where the constraints can be relaxed without sacrificing optimal solutions to your problem. In a similar vein of the maximizing a minimum, if because of the objective we know that certain constraints will be tight at optimal solutions, we can relax the equality to an inequality. For example,

$$\begin{aligned} \max \quad & x_1 + x_2 + \cdots + x_n \\ \text{such that} \quad & x_i = y_i^2 + z_i^2 \text{ for } i = 1, \ldots, n \end{aligned}$$

0.8 Notes from AIMMS modeling book.

0.8.1 Guidelines for Integer Programming Modeling

Practical guidelines for solving difficult MILPs http://inside.mines.edu/~anewman/MIP_practice120212.pdf

0.8.2 Linear Programming Modeling

https://download.aimms.com/aimms/download/manuals/AIMMS3OM_LinearProgrammingTricks.pdf

0.8.3 From AIMMS

https://download.aimms.com/aimms/download/manuals/AIMMS3OM_FormulatingOptimizationModels.pdf

Linear Programming

Practical guidelines for solving difficult linear programs https://pdfs.semanticscholar.org/b01f/ad44c20c372fdda95cbfb980c0d37302de07.pdf

0.8.4 Further Topics

Precedence Constraints

https://or.stackexchange.com/questions/1319/best-model-for-precedence-constraints-within-schedul