Integer Programming

Review/ Slide Correction

The Transportation Problem General Form

- Decision Variables:

  $X_{i,j} = x_{i,j}$ where x is the quantity shipped from

  - i the origin (1,n), to
  - j the destination (1,m)

- Assumes total demand is less than or equal to total supply

$$\text{Min} \sum_{i=1}^{n} \sum_{j=1}^{m} c_{i,j} x_{i,j}$$

where $c_{i,j}$ **is the cost to ship one unit from** i **to** j

Supply $\sum_{j=1}^{m} x_{i,j} \leq s_i$ **for** $i = 1, 2, \ldots, n$

Demand $\sum_{i=1}^{n} x_{i,j} = d_j$ **for** $j = 1, 2, \ldots, m$

$$x_{i,j} \geq 0 \quad \forall i, j$$

The Assignment Problem General Form

$x_{i,j}$ where x is 1 if we assign

- i the person (1,n), to
- j the job (1,m)

$$\text{Max} \sum_{i=1}^{n} \sum_{j=1}^{m} c_{i,j} x_{i,j}$$

where $c_{i,j}$ **is the affinity of person** i **to job** j

Employment $\sum_{j=1}^{m} x_{i,j} \leq 1$ **for person** $i = 1, 2, \ldots, n$

Fulfillment $\sum_{i=1}^{n} x_{i,j} = d_j$ **for job** $j = 1, 2, \ldots, m$

$x_{i,j} \geq 0 \quad \forall i, j$

The Transshipment Problem General Form

- Decision Variables:

  $X_{i,j}=x_{i,j}$ where x is the quantity

$$\sum_{\text{all arcs}} c_{i,j} x_{i,j}$$

where $c_{i,j}$ **is the cost to ship one unit from** i **to** j

Supply Nodes $\sum_{\text{arcs out}} x_{i,j} - \sum_{\text{arcs in}} x_{i,j} \leq s_i$

Transshipment Nodes $\sum_{\text{arcs out}} x_{i,j} - \sum_{\text{arcs in}} x_{i,j} = 0$

Demand Nodes $\sum_{\text{arcs in}} x_{i,j} - \sum_{\text{arcs out}} x_{i,j} = d_j$

$x_{i,j} \geq 0 \quad \forall i, j$

The Shortest Path Problem General Form

$$\text{Min} \sum_{\text{all arcs}} c_{i,j} x_{i,j}$$

where $c_{i,j}$ is the travel time/distance from i to j

where $x_{i,j} \in 0, 1$ is the decision to travel from i to j

Source Node $\sum_{\text{arcs out}} x_{i,j} = 1$

Transshipment Nodes $\sum_{\text{arcs out}} x_{i,j} - \sum_{\text{arcs in}} x_{i,j} = 0$

Destination Node $\sum_{\text{arcs in}} x_{i,j} = 1$

$x_{i,j} \geq 0 \quad \forall i, j$

Max Flow Constraints

Arc Capacities $x_{i,j} \leq c_{i,j}$

Transshipment Nodes $\sum_{\text{arcs out}} x_{i,j} - \sum_{\text{arcs in}} x_{i,j} = 0$

Match Sinks and Sources $\sum_{i \in \text{Sources}} x_{i,j} - \sum_{j \in \text{Sinks}} x_{i,j} = 0$

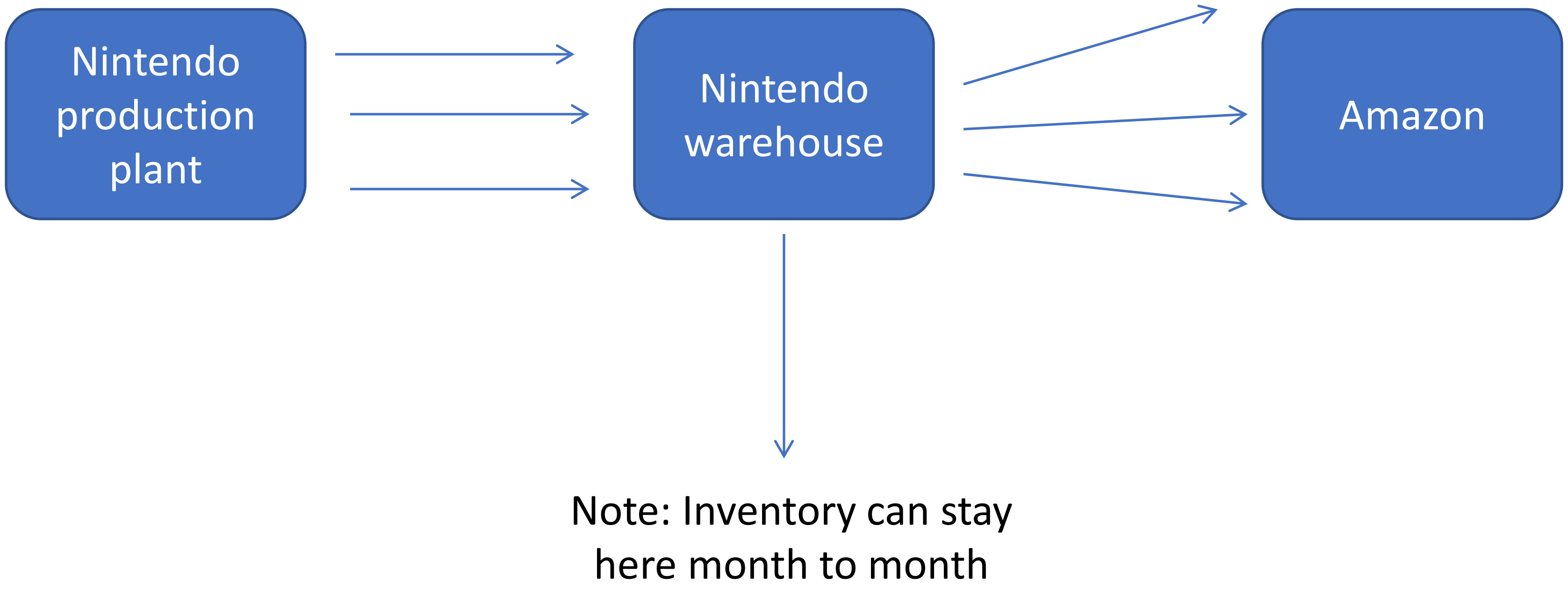
$x_{i,j} \geq 0 \quad \forall i, j$

- Arc Capacity Constraints
  - Limits the flow along arcs
- Transshipment Node Constraints
  - Ensures that outside of Sinks and Sources all flow into nodes balances flow out
- Match flow from Sources to flow into Sinks
  - Balances flow through the whole system

Outline for Class

- Two examples using Network Framing
- What an Integer Program is
  - Terminology associated with different formulations involving integers
- A two-variable integer programming example
  - Graphical depiction
- Rounding non-integer solutions
- A binary integer programming example
- Example

Example 1: Time as a Network



Amazon’s Quarterly Order

System	January	February	March
Switch	1000	3000	5000
Switch lite	1000	500	3000

Production Control Department

- Task:
  - Create a production schedule for the next three months that minimizes overall costs to Nintendo
- What should we consider?
  - Total production costs per item
  - Inventory holding costs
  - Change-in-production-level costs

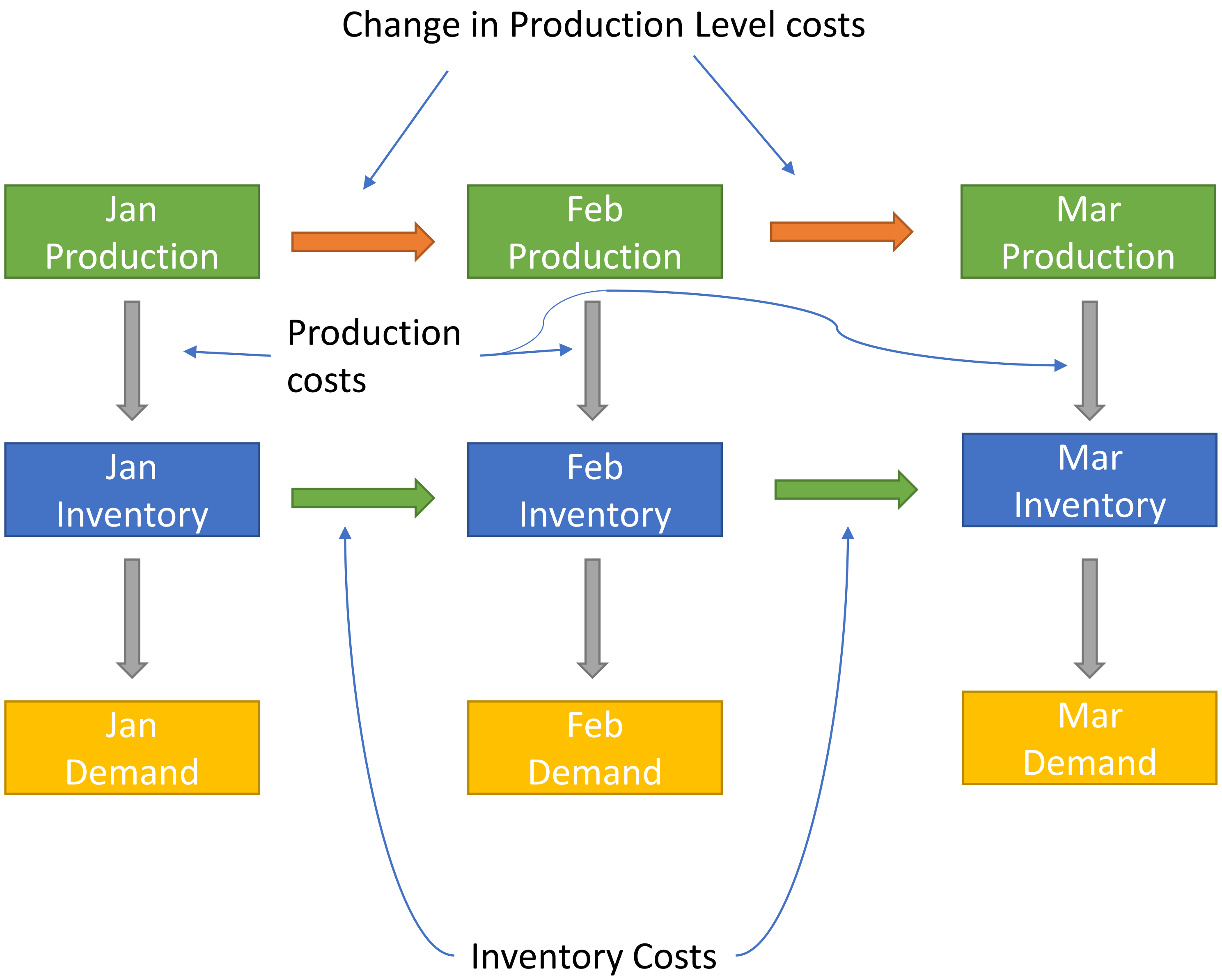
Costs of Producing and Holding Goods

Production

- Variables needed:
  - $x_{S,1}$=# of Switch units to make in month 1 (January)
  - $x_{S,2}$=# of Switch units to make in month 2 (February)
  - $x_{S,3}$=# of Switch units to make in month 3 (March)
  - $x_{L,1}$=# of Switch lite units to make in month 1
  - $x_{L,2}$=# of Switch lite units to make in month 2
  - $x_{L,3}$=# of Switch lite units to make in month 3
- Total Cost:
  - \$80 per S unit
  - \$40 per L unit
  - $80(x_{S,1}+x_{S,2}+x_{S,3}) + 40(x_{L,1}+x_{L,2}+x_{L,3})$

Inventory

- Variables needed:
  - $s_{S,1}$=# of Switch units leftover in month 1 (January)
  - $s_{S,2}$=# of Switch units to leftover month 2 (February)
  - $s_{S,3}$=# of Switch units to leftover in month 3 (March)
  - $s_{L,1}$=# of Switch lite units to leftover in month 1
  - $s_{L,2}$=# of Switch lite units to leftover in month 2
  - $s_{L,3}$=# of Switch lite units to leftover in month 3
- Inventory Holding Cost:
  - Fraction of the of the production cost:
    - \$0.30 per Switch unit per month
    - \$0.15 per Switch lite unit per month
  - $0.30(s_{S,1}+s_{S,2}+s_{S,3}) + 0.15(s_{L,1}+s_{L,2}+s_{L,3})$


Change in Production Level costs
Jan Production
Feb Production
Mar Production
Production costs
Jan Inventory
Feb Inventory
Mar Inventory
Jan Demand
Feb Demand
Mar Demand
Inventory Costs

Change-in-production-level

- Variables needed:
  - I_1=increase in production level month 1
  - I_2=increase in production level month 2
  - I_3=increase in production level month 3
  - D_1=decrease in production level month 1
  - D_2=decrease in production level month 2
  - D_3=decrease in production level month 3
- Change Cost:
  - \$0.50 per unit increase per month
  - \$0.20 per unit decrease per month
  - $0.50(I_1+I_2+I_3) + 0.20(D_1+D_2+D_3)$

We want to minimize overall cost:

- Production + Inventory + Change-in-level

  - $80(x_{S,1}+x_{S,2}+x_{S,3}) + 40(x_{L,1}+x_{L,2}+x_{L,3})$
  - $+0.30(s_{S,1}+s_{S,2}+s_{S,3}) + 0.15(s_{L,1}+s_{L,2}+s_{L,3})$
  - $+0.50(I_1+I_2+I_3) + 0.20(D_1+D_2+D_3)$

What about constraints?

- We need to meet demand:
  - Production in the month
  - Inventory leftover from the month before
  - Inventory that will be left this month
- Meeting demand for Switch in February:
  - $s_{S,1}+x_{S,2}-s_{S,2}=3000$
- Let's assume that we have a starting inventory of 500 Switch and 200 Switch lite systems in stock before we begin production…

Meeting Demand

- $500+x_{S,1}-s_{S,1}=1000$
- $200+x_{L,1}-s_{L,1}=1000$
- $s_{S,1}+x_{S,2}-s_{S,2}=3000$
- $s_{L,1}+x_{L,2}-s_{L,2}=500$
- $s_{S,2}+x_{S,3}-s_{S,2}=5000$
- $s_{L,2}+x_{L,3}-s_{L,2}=3000$

- Let us also assume that we want to have a buffer left at the end of 400 Switch and 200 Switch lite systems:
  - $s_{S,3}>=400$
  - $s_{L,3}>=200$

Time, Labor, and Storage

System	Time	Labor	Storage
Switch	0.10	0.05	2
Switch lite	0.08	0.07	3
Available	(400,500,600)	(300,300,300)	(10k, 10k, 10k)

- Time:
  - $0.10x_{S,1}+0.08x_{L,1}=400$
  - $0.10x_{S,2}+0.08x_{L,2}=500$
  - $0.10x_{S,3}+0.08x_{L,3}=600$
- Labor
  - $0.05x_{S,1}+0.07x_{L,1}=300$
  - $0.05x_{S,2}+0.07x_{L,2}=300$
  - $0.05x_{S,3}+0.07x_{L,3}=300$
- Storage
  - $2x_{S,1}+3x_{L,1}=400$
  - $2x_{S,2}+3x_{L,2}=500$
  - $2x_{S,3}+3x_{L,3}=600$

Change-in-level

- Let’s say that in December we produced a total of 2,500 units
- How do we figure out the change in production level from December to January?
  - Jan production of Switch + Jan production of Switch lite – December production=Change
  - $x_{S,1}+x_{L,1}-2500=I_1-D_1$

Change-in-level

- How do we get the change from one month to the next after the first month?
  - Current month production-previous month production=Current month change
  - $(x_{S,2}+x_{L,2})-(x_{S,1}+x_{L,1})=I_2-D_2$
  - $(x_{S,3}+x_{L,3})-(x_{S,2}+x_{L,2})=I_3-D_3$

Example 2: A finance-related Linear Program

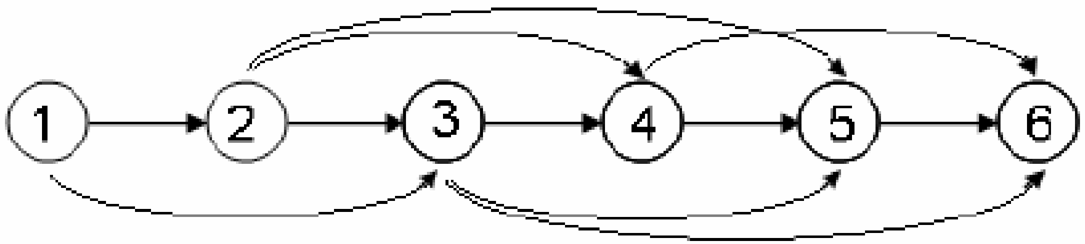
- Alice has $2200 to invest over the next five years
- At the beginning of each year, she can invest money in one or two-year time deposits
  - The bank pays 8% interest on one-year time deposits
  - The bank pays 17% total on two-year time deposits
  - Also, three-year certificates returning 27% will be offered starting at the beginning of the second year
- Alice reinvests her available money every year
- Formulate a linear program to maximize total cash on hand at the end of the fifth year

Linear Program

$$
\begin{array}{llllll}
\text{Max} & 1.27X_{33}+ & 1.17X_{42}+ & 1.08X_{51} & & \\
\text{s. t.} & X_{11}+ & X_{12} & & = & 2200 \\
 & X_{21}+ & X_{22}+ & X_{23} & = & 1.08X_{11} \\
 & X_{31}+ & X_{32}+ & X_{33} & = & 1.17X_{12}+1.08X_{21} \\
 & X_{41}+ & X_{42} & & = & 1.17X_{22}+1.08X_{31} \\
 & X_{51} & & & = & 1.27X_{23}+1.17X_{32}+1.08X_{41} \\
 & & & X_{ij} & \geq & 0
\end{array}
$$

where X_{ij} = amount invested in period i for j years

Network representation



What is an Integer Program?

“A linear program with the additional requirement that one or more variables must be integer”

Pro

- Provides increased flexibility in types of applications as many real-world problems do not deal with strictly continuous decision variables

Con

- Solving capabilities are severely compromised when this assumption of linear programming is relaxed

Variations of Integer Programming

- All Integer Program:
  - An integer program in which all of the variables are required to be integers
- Mixed Integer Program
  - An integer program in which only some of the variables must be integers
- Binary Integer Program
  - An integer program in which variables can only take on values of 0 or 1

Important Terminology

- The Linear Programming Relaxation (LP Relaxation)
  - An integer program with the integer requirement removed
  - A linear program as we have seen them before
- LP Relaxations provide a bound for the objective function, since an IP will never be able to do better (given that an integer solution is also a valid solution to the LP)

LP Relaxations

Integer Program

- Decision Variables:
  - x_1 and x_2
- Objective Function:
  - Maximize $2x_1+3x_2$
- Constraints:
  - x_1+x_2 <=10
  - $4x_1+5x_2$ <=18
  - x_1, x_2 >=0
  - x_1, x_2 are integer

Relaxation

LP Relaxation

- Decision Variables:
  - x_1 and x_2
- Objective Function:
  - Maximize $2x_1+3x_2$
- Constraints:
  - x_1+x_2 <=10
  - $4x_1+5x_2$ <=18
  - x_1, x_2 >=0

An Integer Programming Example

- Let's say that you work for a property management company. Your boss has asked you to research investing in townhomes and apartment buildings in a specific part of town
- You have found out that a townhome costs $282,000 and is expected to generate $10,000 in annual income. An apartment building will cost $400,000 and can earn an annual profit of $15,000.
- You only have 140 hours per month to dedicate to managing these properties. A townhome requires 4 hours/month while an apartment building requires 40.
- Lastly, you know that you have a budget of $2,000,000 and there are only 5 townhomes in the area.

Model Formulation

- Decision Variables:
  - x_1 = number of townhomes to purchase
  - x_2 = number of apartment buildings to purchase
- Objective Function:
  - Maximize $10\ x_1 + 15\ x_2$
- Constraints:
  - $282\ x_1 + 400\ x_2 <= 2000$
  - $4\ x_1 + 40\ x_2 <= 140$
  - $x_1 <= 5$
  - $x_1, x_2 >= 0$
  - x_1, x_2 are integer

Let’s solve the LP Relaxation

```
obj <- c(10,15)
mat <- matrix(c(282,400,
              4,40,
              1,0), nrow = 3, byrow=TRUE)
dir <- c("<=", "<=", "<=")
rhs <- c(2000, 140, 5)
```

```
lp_lpr <- lp("max", obj, mat, dir, rhs, compute.sens = 1)
sensitivity_report(lp_lpr)
```

```
Objective Function Value:
73.57438

 Objective Function
  variable final_value reduced_cost coefficient min_coef max_coef
1      x 1    2.479339            0          10   1.5000   10.575
2      x 2    3.252066            0          15  14.1844  100.000

Constraints
     constraint         LHS  RHS shadow_price min_RHS max_RHS
1 constraint_ 1 2000.000000 2000   0.03512397    1400    2610
2 constraint_ 2  140.000000  140   0.02376033      79     200
3 constraint_ 3    2.479339    5   0.00000000    -Inf     Inf
```

- Solving this we get:
  - T=2.479
  - A=3.252
- With an objective function value of:
  - $73, 574 in annual income

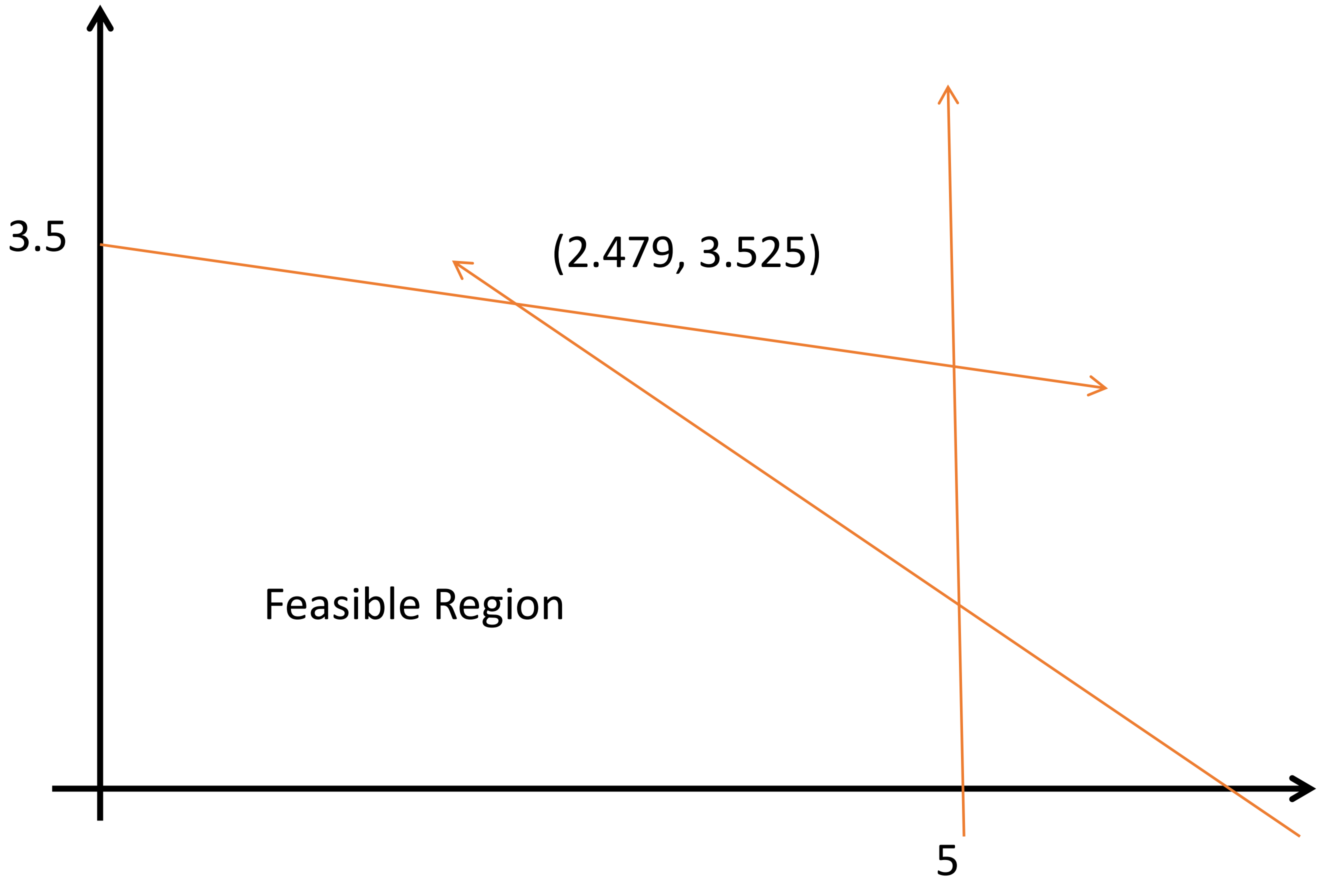
Why don’t we try rounding

- Round up:
  - T=3
  - A=4
  - Infeasible
- Round down (same as general rounding)
  - T=2
  - A=3
  - Feasible
  - Objective Function: $65,000

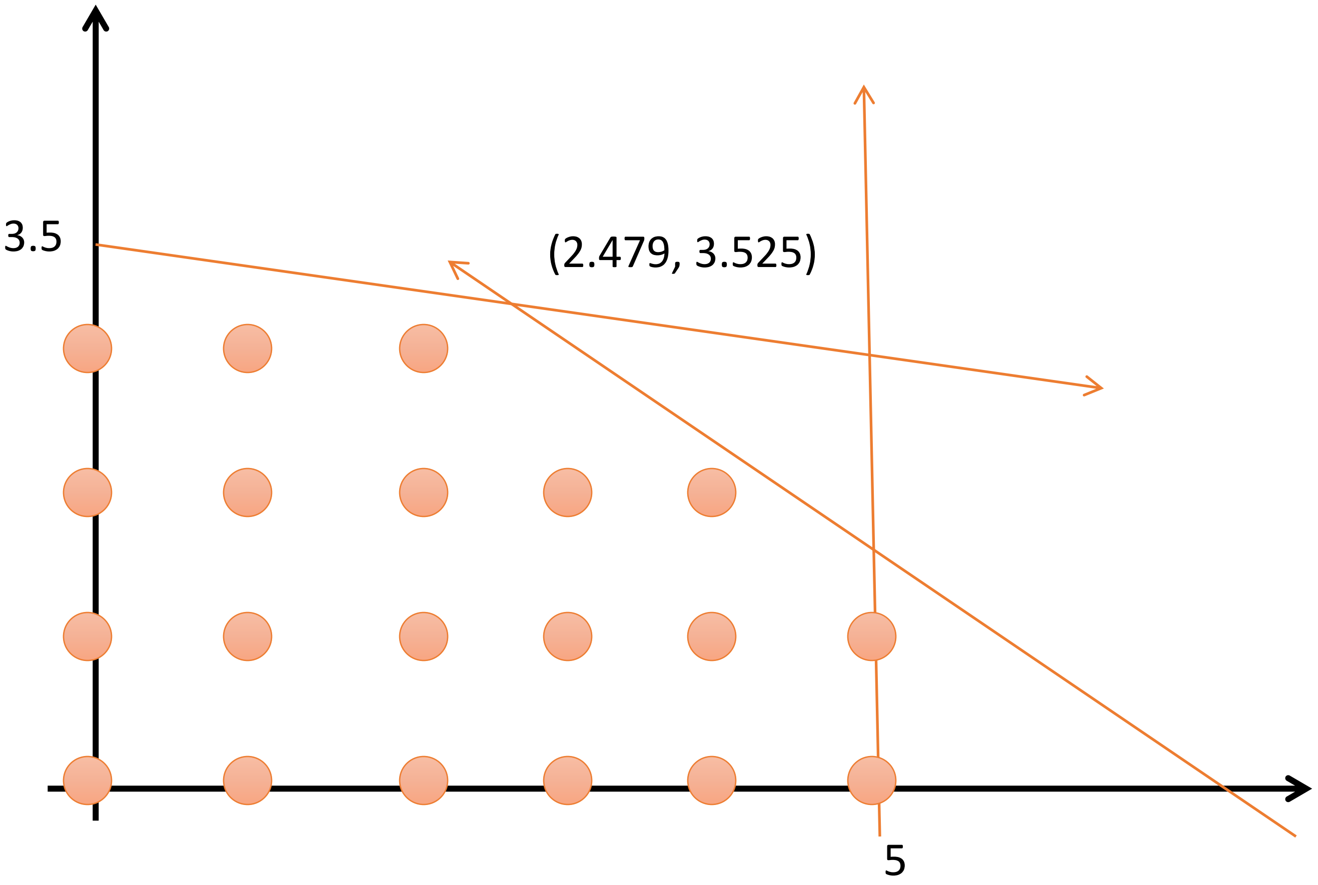
Comparison to LP Relaxation

- Difference in annual income:
  - $8,574
- The rounded integer solution is only 88% of the optimal solution
- Could we do better?

Let’s look at the Graph



We only want integer solutions



Integer Point	Objective Function Value
(1,0)	10,000
(2,0)	20,000
(3,0)	30,000
(4,0)	40,000
(5,0)	50,000
(0,1)	15,000
(1,1)	25,000
(2,1)	35,000
(3,1)	45,000
(4,1)	55,000
(5,1)	65,000
(0,2)	30,000
(1,2)	40,000
(2,2)	50,000
(3,2)	60,000
(4,2)**	**70,000**
(0,3)	45,000
(1,3)	55,000
(2,3)	65,000

Comparison to other solutions

- Optimal Linear Programming solution:
  - Objective Function Value: $73,574
- Rounded solution:
  - Objective Function Value: $65,000
  - 88% of optimal LP
- Optimal Integer Solution:
  - Objective Function Value: $70,000
  - 95% of optimal LP

```r
lp_lpr <- lp("max", obj, mat, dir, rhs, compute.sens = 1)
sensitivity_report(lp_lpr)
```

```
Objective Function Value:
73.57438

 Objective Function
  variable final_value reduced_cost coefficient min_coef max_coef
1      x 1    2.479339            0          10   1.5000   10.575
2      x 2    3.252066            0          15  14.1844  100.000
```

```r
lp_lpr <- lp("max", obj, mat, dir, rhs, all.int = TRUE, compute.sens = 1)
sensitivity_report(lp_lpr)
```

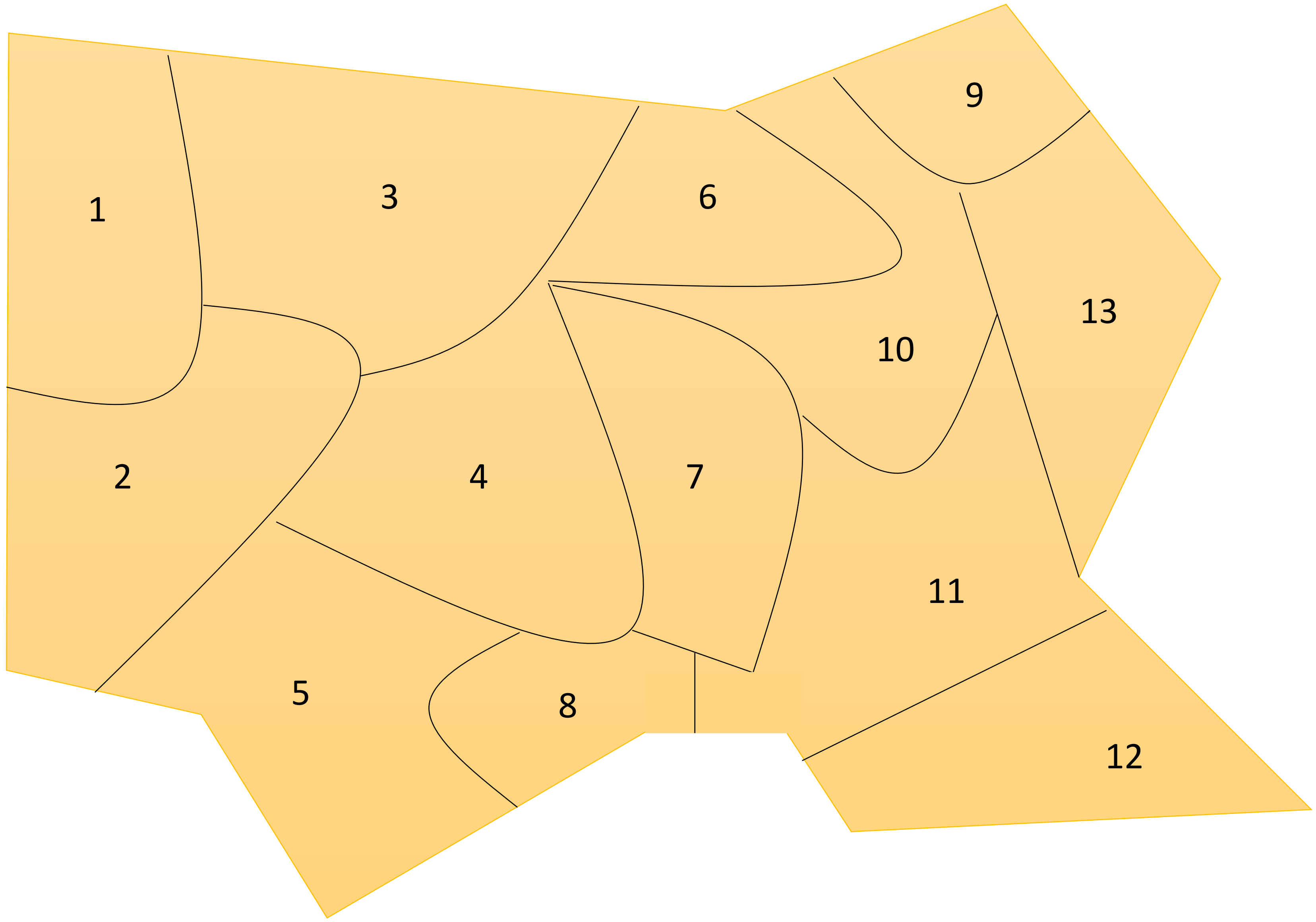
```
Objective Function Value:
70

 Objective Function
  variable final_value reduced_cost coefficient min_coef max_coef
1      x 1           4           10          10      1.5      Inf
2      x 2           2           15          15      0.0      100
```

When is it ok to round?

- Rounding is a guess and check process and never guarantees any kind of optimality
- Sometimes it is ok when our variable values are large (e.g. $x1=100435.4$)
  - Here, the contribution to the objective function may not matter to managers

A Binary Integer Programming Example

- You work for a cell phone company that is currently looking to place towers in a new area.
- Each cell phone tower costs $125,000 to build.
- You would like to minimize the total cost of setting up these towers
- You have divided up the new region into the following sections


9
1
3
6
13
10
2
4
7
11
5
8
12

Binary Program Formulation

- x_i = 1 if a tower is placed in area i, 0 otherwise
- Min $\sum_i 125{,}000x_i$
- $x_1 + x_2 + x_3 \geq 1$ [constraint that a tower must be placed in or adjacent to area 1]
- $x_2 + x_1 + x_3 + x_4 + x_5 \geq 1$ [constraint that a tower must be placed in or adjacent to area 2]
- …

```r
obj <- rep(1,13)
mat <- matrix(c(1,1,1,0,0,0,0,0,0,0,0,0,0, # 1
                1,1,1,1,1,0,0,0,0,0,0,0,0, # 2
                1,1,1,1,0,1,0,0,0,0,0,0,0, # 3
                0,1,1,1,1,0,1,1,0,0,0,0,0, # 4
                0,1,0,1,1,0,0,1,0,0,0,0,0, # 5
                0,0,1,0,0,1,0,0,0,1,0,0,0, # 6
                0,0,0,1,0,0,1,1,0,1,1,0,0, # 7
                0,0,0,1,1,0,1,1,0,0,1,0,0, # 8
                0,0,0,0,0,0,0,0,1,1,0,0,1, # 9
                0,0,0,0,0,1,1,0,1,1,1,0,1, # 10
                0,0,0,0,0,0,1,1,0,1,1,1,1, # 11
                0,0,0,0,0,0,0,0,0,0,1,1,0, # 12
                0,0,0,0,0,0,0,0,1,1,1,0,1), nrow = 13, byrow=TRUE) # 13
dir <- rep(">=", 13)
rhs <- rep(1, 13)
```

```r
lp_cel <- lp("min", obj, mat, dir, rhs, all.int = TRUE, compute.sens = 1)
sensitivity_report(lp_cel)
```

```
Objective Function Value:
3

 Objective Function
   variable final_value reduced_cost coefficient min_coef max_coef
1       x 1           0            0           1        0      Inf
2       x 2           1            0           1        1        1
3       x 3           0            0           1        1      Inf
4       x 4           0            0           1        0      Inf
5       x 5           0            0           1        1      Inf
6       x 6           0            1           1        0      Inf
7       x 7           0            0           1        1      Inf
8       x 8           0            0           1        1      Inf
9       x 9           0            0           1        1      Inf
10     x 10           1            0           1        0        1
11     x 11           1            0           1        1        2
12     x 12           0            1           1        0      Inf
13     x 13           0            0           1        1      Inf
```

Solution:

Only 3 towers are needed in sections 2,10,11

Investment	Net Present Value	Cost Year 1	Cost Year 2	Cost Year 3
Limited Warehouse Expansion	4,000	3,000	1,000	4,000
Extensive Warehouse Expansion	6,000	2,500	3,500	3,500
Test Market New Product	10,500	6,000	4,000	5,000
Advertising Campaign	4,000	2,000	1,500	1,800
Basic Research	8,000	5,000	1,000	4,000
Purchase New Equipment	3,000	1,000	500	900

Year 1 Budget	Year 2 Budget	Year 3 Budget
10,500	7,000	8,750

Mixing Variable Types

- Last Class:
  - Binary Integer Programs
  - All Integer Programs
  - Mixed Integer Programs
  - Linear Programming Relaxation

An Example From Marketing:

- Salem Foods new sausage pizza

The Current Pizzas

- Antonio’s
  - Thick Crust
  - Mozzarella Cheese
  - Chunky Sauce
  - Medium Sausage
- King’s
  - Thin Crust
  - Cheese Blend
  - Smooth Sauce
  - Mild Sausage

Data from Market Research

TABLE 7.4 PART-WORTHS FOR THE SALEM FOODS PROBLEM

	Crust		Cheese		Sauce		Sausage Flavor		
Consumer	Thin	Thick	Mozzarella	Blend	Smooth	Chunky	Mild	Medium	Hot
1	11	2	6	7	3	17	26	27	8
2	11	7	15	17	16	26	14	1	10
3	7	5	8	14	16	7	29	16	19
4	13	20	20	17	17	14	25	29	10
5	2	8	6	11	30	20	15	5	12
6	12	17	11	9	2	30	22	12	20
7	9	19	12	16	16	25	30	23	19
8	5	9	4	14	23	16	16	30	3

What are we trying to answer?

- How should Salem Foods design their new pizza in order to achieve the highest market share?

How can we achieve this goal?

- Selecting the appropriate attributes for Salem Foods’ pizza
- Converting as many consumers as possible

What is Consumer 1’s Preference?

- Antonio’s Pizza Utility:
  - Thick Crust = 2
  - Mozzarella Cheese = 6
  - Chunky Sauce=17
  - Medium Sausage=27
  - Total=52
- King’s Pizza Utility
  - Thin Crust=11
  - Cheese Blend=7
  - Smooth Sauce=3
  - Mild Sausage=26
  - Total=47

Where do the 8 consumers stand now?

- Consumer 1
  - Antonio’s Utility:52
- Consumer 2
  - King’s Utility:58
- Consumer 3
  - King’s Utility:66
- Consumer 4
  - Antonio’s Utility:83
- Consumer 5
  - King’s Utility:58
- Consumer 6
  - Antonio’s Utility:70
- Consumer 7
  - Antonio’s Utility:79
- Consumer 8
  - Antonio’s Utility:59

Decision Variables

- $A_{i,j}$=1 if Salem uses choice i for attribute j, 0 if not
- y_i=1 if Consumer i prefers Salem’s pizza, 0 if not

Objective Function

- Maximize total # of consumer's that will prefer Salem Foods' sausage pizza=
  - $y_1+y_2+y_3+y_4+y_5+y_6+y_7+y_8$

Constraints:

- Attributes:
  - Use exactly 1 type of crust:
    - $A_{1,1}+ A_{2,1}=1$
  - Use exactly 1 type of cheese:
    - $A_{1,2}+ A_{2,2}=1$
  - Use exactly 1 type of sauce:
    - $A_{1,3}+ A_{2,3}=1$
  - Use exactly 1 type of sausage:
    - $A_{1,4}+ A_{2,4}+ A_{3,4}1$
- Binary
  - All variables are binary

Constraints:

- Consumer Preferences:
  - $11A_{1,1}+2A_{2,1}+6A_{1,2}+7A_{2,2}+3A_{1,3}+17A_{2,3}+26A_{1,4}+27A_{2,4}+8A_{3,4}>52y_1$

Constraints:

- Consumer Preferences:
  - $11A_{1,1}+2A_{2,1}+6A_{1,2}+7A_{2,2}+3A_{1,3}+17A_{2,3}+26A_{1,4}+27A_{2,4}+8A_{3,4}>52y_1$
  - Since preferences are in terms of whole numbers, we can say:
  - $11A_{1,1}+2A_{2,1}+6A_{1,2}+7A_{2,2}+3A_{1,3}+17A_{2,3}+26A_{1,4}+27A_{2,4}+8A_{3,4}>=1+52y_1$
  - Which is the same as:
  - $11A_{1,1}+2A_{2,1}+6A_{1,2}+7A_{2,2}+3A_{1,3}+17A_{2,3}+26A_{1,4}+27A_{2,4}+8A_{3,4}-52y_1>=1$

Constraints:

- Consumer Preferences:
  - $11A_{1,1}+2A_{2,1}+6A_{1,2}+7A_{2,2}+3A_{1,3}+17A_{2,3}+26A_{1,4}+27A_{2,4}+8A_{3,4}-52y_1>=1$
  - $11A_{1,1}+7A_{2,1}+15A_{1,2}+17A_{2,2}+16A_{1,3}+26A_{2,3}+14A_{1,4}+1A_{2,4}+10A_{3,4}-58y_2>=1$
  - $7A_{1,1}+5A_{2,1}+8A_{1,2}+14A_{2,2}+16A_{1,3}+7A_{2,3}+29A_{1,4}+16A_{2,4}+19A_{3,4}-66y_3>=1$
  - $13A_{1,1}+20A_{2,1}+20A_{1,2}+17A_{2,2}+17A_{1,3}+14A_{2,3}+25A_{1,4}+29A_{2,4}+10A_{3,4}-83y_4>=1$
  - $2A_{1,1}+8A_{2,1}+6A_{1,2}+11A_{2,2}+30A_{1,3}+20A_{2,3}+15A_{1,4}+5A_{2,4}+12A_{3,4}-58y_5>=1$
  - $12A_{1,1}+17A_{2,1}+11A_{1,2}+9A_{2,2}+2A_{1,3}+30A_{2,3}+22A_{1,4}+12A_{2,4}+20A_{3,4}-70y_6>=1$
  - $9A_{1,1}+19A_{2,1}+12A_{1,2}+16A_{2,2}+16A_{1,3}+25A_{2,3}+30A_{1,4}+23A_{2,4}+19A_{3,4}-79y_7>=1$
  - $5A_{1,1}+9A_{2,1}+4A_{1,2}+14A_{2,2}+23A_{1,3}+16A_{2,3}+16A_{1,4}+30A_{2,4}+3A_{3,4}-59y_8>=1$

Solution

- $A_{1,1}=A_{2,2}=A_{2,3}=A_{1,4}=1$
- This means:
  - Thin crust, cheese blend, smooth sauce and mild sausage
- $y_1=y_2=y_6=y_7=1$
  - Consumers 1, 2, 6, and 7 will prefer Salem’s pizza
- The optimal objective function value is 4

Binary Variable Constraints:

- Multiple Choice/Mutually Exclusive
- K out of N Alternatives
- Conditional
- Co-requisite

Example Situation:

- Suppose that you are trying to remodel your house. You have listed 10 projects that on your wish-list
  - 1:New kitchen cabinets
  - 2:New bathroom floors
  - 3:New paint in the living room
  - 4:New couch
  - 5:New floor in the basement
  - 6:Put up dry wall in the basement
  - 7:New windows for the house
  - 8:Landscape the front walkway
  - 9:Get a new stove/oven
  - 10:New counter tops in the kitchen

- Given this list of 10 projects, you have assigned a personal level of importance to each and have estimated a cost for all projects.
- You are working on a budget and have decided to set up an integer program to help you choose which of the 10 projects you should attempt now.
- Your variables:
    - x_i=1 if we will complete project i, and 0 if not

Multiple Choice/Mutually Exclusive

- We know that new floor in the basement, new kitchen cabinets, and painting the living room will be the largest hassles. Therefore, we only want to complete one of these three projects.

- $x_1 + x_3 + x_5 = 1$
- $x_1 + x_3 + x_5 =< 1$

K out of N Alternatives

- You would like to complete at least 6 of the possible 10 projects
  - $x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8+x_9+x_{10}>=6$

- Additionally, you want to complete at least two of the projects involving the kitchen/living room.
  - $x_1+x_3+x_4+x_9+x_{10}>=2$

Conditional

- You have been very frustrated with your current stove and have decided that you do not want to replace the kitchen cabinets unless you have also replaced the stove/oven.

Conditional

- You have been very frustrated with your current stove and have decided that you do not want to replace the kitchen cabinets unless you have also replaced the stove/oven.
  - $x_1 <= x_9$
- Can also be used with either or conditions

Co-requisites

- Since we are trying to renovate the basement eventually, we either want to put up dry wall in the basement and finish the floors, or do nothing in the basement.

Co-requisites

- Since we are trying to renovate the basement eventually, we either want to put up dry wall in the basement and finish the floors, or do nothing in the basement.

  - $x_5 - x_6 = 0$