Homework #6

Due **Monday, August 8** @ 11:59pm

Submission requirements

Upload a **single PDF or HTML file** of your IJulia notebook for this entire assigment. Clearly denote which question each section of your PDF corresponds to.

Problem 1 -- Facility Location + Logic

The 524 Bakery is expanding their operations beyond Wisconsin to Minnesota, Iowa, and Michigan. It is estimated that weekly demand for their most popular cake (Vanilla Buttercream) in each state will be 750 in Wisconsin, 300 in Minnesota, 450 in Iowa, and 600 in Michigan. The bakery is opening warehouses to mass produce the cakes in four cities: Madison, Minneapolis, Dubuque, and Ann Arbor. The average combined cost per cake of baking, packing, and shipping it from a warehouse to destinations in each state is given in the table below.

City	Wisconsin	Minnesota	Iowa	Michigan
Madison	75	92	84	102
Minneapolis	96	105	80	114
Dubuque	90	84	82	95
Ann Arbor	106	102	112	85

Any warehouse can produce up to 1300 cakes per week. If a warehouse is opened, it incurs a fixed cost of $20,000 when opened. Demand in a state can be met from any combination of open warehouses. However, to make the hassle of shipping "worth it," if any cakes are sent from a warehouse to a state, then at least 300 cakes must be sent from that warehouse to that state.

(a) Formulate an integer program **using math notation** that will help the bakery determine where to locate their warehouses and how many cakes should be sent from each warehouse to each state in order to minimize the total cost of meeting weekly demand.

(b) Implement and solve the model from part(a) using the JuMP package in Julia, and explain the solution in terms of the problem. You should use Gurobi or Cbc as your solver.

(c) Now suppose the bakery wishes to enforce the following constraint: if at least 600 cakes are made in Madison, then the amount shipped from Madison to Minnesota must be at least as much as the amount shipped from Madison to all other states. Extend the model to include this constraint. How does your solution change?

Solution Problem 1

(a) Formulate an integer program using math notation that will help the bakery determine where to locate their warehouses and how many cakes should be sent from each warehouse to each state in order to minimize the total cost of meeting weekly demand.

Solution

Sets

I: set of cities $\{$madison (1), minneapolis (2), dubuque (3), annarbor (4)$\}$

J: set of states $\{$wi (1), mn (2), ia (3), mi (4)$\}$

A: set of all arcs between $i \in I$ and $j \in J$.

Parameters

c_{ij}: cost of producing and shipping a cake from $i \in I$ to $j \in J$.

d_j: demand for cakes in state $j \in J$.

Variables

x_{ij}: number of cakes shipped from warehouse $i \in I$ to state $j \in J$.

y_i: Binary variable, equal to 1 if a warehouse in $i \in I$ is opened, 0 otherwise.

z_{ij}: Binary variable, equal to 1 if any cakes are shipped from $i \in I$ to state $j \in J$, 0 otherwise

Objective

Minimize the toatal cost:

$$\begin{aligned}\min \quad & 20000(y_1+y_2+y_3+y_4) + 75x_{11} + 92x_{12} + 84x_{13} + 102x_{14} \\ & + 96x_{21} + 105x_{22} + 80x_{23} + 114x_{24} + 90x_{31} + 84x_{32} + 82x_{33} \\ & + 95x_{34} + 106x_{41} + 102x_{42} + 112x_{43} + 85x_{44}\end{aligned}$$

Constraints

Demand in a state can be met by any combination of open warehouses:

$$\begin{aligned} x_{11}+x_{21}+x_{31}+x_{41} &\geq 750 \\ x_{12}+x_{22}+x_{32}+x_{42} &\geq 300 \\ x_{13}+x_{23}+x_{33}+x_{43} &\geq 450 \\ x_{14}+x_{24}+x_{34}+x_{44} &\geq 600 \end{aligned}$$

Any open warehouse can produce up to 1300 cakes per week:

$$\begin{aligned} x_{11}+x_{12}+x_{13}+x_{14} &\leq 1300y_1 \\ x_{21}+x_{22}+x_{23}+x_{24} &\leq 1300y_2 \\ x_{31}+x_{32}+x_{33}+x_{34} &\leq 1300y_3 \\ x_{41}+x_{42}+x_{43}+x_{44} &\leq 1300y_4 \end{aligned}$$

Turn off flow from warehouse $i \in I$ to state $j \in J$ if $z_{ij} = 0$, and enforce minimum flow if $z_{ij} = 1$:

$$300z_{ij} \leq x_{ij} \leq 1300z_{ij} \quad \forall i \in I, j \in J$$

Here we used the constant 1300 in the upper bound constraint since we know no warehouse can produce more than 1300 cakse (and hence at most that much can be shipped from the warehouse to any state). Smaller constants could be obtained by usng the demand of each state.

Sign constraints and binay restrictions,

$$\begin{aligned} & x_{ij} \geq 0, \quad \forall (i,j) \in A \\ & z_{ij} \in \{0,1\}, \quad \forall (i,j) \in A \\ & y_i \in \{0,1\}, \quad \forall i \in I \end{aligned}$$

(b) Implement and solve the model from part(a) using the JuMP package in Julia, and explain the solution in terms of the problem. You should use Gurobi or Cbc as your solver.

Solution

In [1]:

```julia
using JuMP, Cbc

warehouse = [:madison, :minneapolis, :dubuque, :annarbor]
states = [:WI, :MN, :IA, :MI]

fixed_cost = 20000

var_costs = [75 92 84 102
             96 105 80 114
             90 84 82 95
             106 102 112 85]

using NamedArrays

var_cost_NA = NamedArray(var_costs,(warehouse,states),("warehouse","state"))

demand = Dict(zip(states,[750 300 450 600]))
max_ship = 1300

m = Model(Cbc.Optimizer)

@variable(m, x[warehouse, states] >= 0)
@variable(m, z[warehouse, states], Bin)
@variable(m, y[warehouse], Bin)

@objective(m, Min, sum(x[i,j]*var_cost_NA[i,j] for i in warehouse, j in states) + fixed_cost*sum(y))

@constraint(m, meet_demand[i in states], sum(x[j,i] for j in warehouse) >= demand[i])
@constraint(m, capacity[j in warehouse], sum(x[j,i] for i in states) <= max_ship*y[j])
@constraint(m, lb[i in states, j in warehouse], 300z[j,i] <= x[j,i] )
@constraint(m, ub[i in states, j in warehouse], x[j,i] <= 1300z[j,i])

set_silent(m)
optimize!(m)

for i in warehouse
    if value(y[i]) > 0.99
        println("open warehouse in ", i)
        for j in states
            if value(x[i,j]) > 0.001
                println("and ship ", value(x[i,j]), " cakes to ", j)
            end
        end
    end
end
```

```
open warehouse in madison
and ship 750.0 cakes to WI
and ship 450.0 cakes to IA
open warehouse in annarbor
and ship 300.0 cakes to MN
and ship 600.0 cakes to MI
```

(c) Now suppose the bakery wishes to enforce the following constraint: if at least 600 cakes are made in Madison, then the amount shipped from Madison to Minnesota must be at least as much as the amount shipped from Madison to all other states. Extend the model to include this constraint. How does your solution change?

Solution

Introduce an auxiliary binary variable δ and we model the logical chain:

- $\sum_{j=1}^4 x_{1j} \geq 600 \Rightarrow \delta = 1 \Rightarrow x_{12} \geq \sum_{j=2}^4 x_{1j}$
- For the first implication we use the last trick in the slide of tricks: $\sum_{j=1}^4 x_{1j} \leq 600 - \epsilon + (M+\epsilon)\delta$
- If we assume cakes can only be sent in integer quantities, we can use $\epsilon = 1$, otherwise we use $\epsilon = 0.01$. For M we need an upper bound on $\sum_{j=1}^4 x_{1j} - 600$. Since the total that can be made in Madison is limited to 1300 cakes, we can use $M = 1300 - 600 = 700$. Thus, the first constraint is modeled with the linear constraint:$\sum_{j=1}^4 x_{1j} \leq 699.99 + 700.01\delta$.
- For the second logical condition, we re-write it as: $\delta = 1 \Rightarrow x_{11} - \sum_{j=2}^4 x_{1j} \geq 0$
- Using the third trick on the slide of tricks we have: $x_{11} - \sum_{j=2}^4 x_{1j} \geq m(1-\delta)$ where we need m to be a lower bound on $x_{11} - \sum_{j=2}^4 x_{1j}$. To derive this lower bound we know $x_{11} \geq 0$, and $\sum_{j=2}^4 x_{1j} \leq 1300$ (the maximum possible production from Madison), thus we can use $m = -1300$.
- Hence, the last constraint we add is: $x_{11} - \sum_{j=2}^4 x_{1j} \geq -1300(1-\delta)$.

In summary, the extended model introduces the binary variable δ and the two constraints:

$$\begin{aligned}\sum_{j=1}^4 x_{1j} \leq 699.99 + 700.01\delta, \\ x_{12} - x_{11} - x13 - x14 \geq -1300(1-\delta).\end{aligned}$$

In [2]:

```julia
using JuMP, Cbc

states = [:WI, :MN, :IA, :MI]
warehouse = [:madison, :minneapolis, :dubuque, :annarbor]

fixed_cost = 20000

var_costs = [75 92 84 102
             96 105 80 114
             90 84 82 95
             106 102 112 85]

using NamedArrays

var_cost_NA = NamedArray(var_costs,(warehouse,states),("warehouse","state"))

demand = Dict(zip(states,[750 300 450 600]))
max_ship = 1300

m = Model(Cbc.Optimizer)

@variable(m, x[warehouse, states] >= 0)
@variable(m, z[warehouse, states], Bin)
@variable(m, y[warehouse], Bin)
@variable(m, δ, Bin)

@objective(m, Min, sum(x[i,j]*var_cost_NA[i,j] for i in warehouse, j in states) + fixed_cost*sum(y))

@constraint(m, meet_demand[i in states], sum(x[j,i] for j in warehouse) >= demand[i])
@constraint(m, capacity[j in warehouse], sum(x[j,i] for i in states) <= max_ship*y[j])
@constraint(m, lb[i in states, j in warehouse], 300z[j,i] <= x[j,i] )
@constraint(m, ub[i in states, j in warehouse], x[j,i] <= 1300z[j,i])

@constraint(m, sum(x[:madison,j] for j in states) <= 699.99 + 700.01δ)
@constraint(m, x[:madison, :MN] - x[:madison, :WI] - x[:madison, :IA] - x[:madison, :MI] >= -1300(1-δ))

set_silent(m)
optimize!(m)

for i in warehouse
    if value(y[i]) > 0.99
        println("open warehouse in ", i)
        for j in states
            if value(x[i,j]) > 0.001
                println("and ship ", value(x[i,j]), " cakes to ", j)
            end
        end
    end
end
```

```
open warehouse in dubuque
and ship 450.00000000000006 cakes to WI
and ship 300.0 cakes to MN
and ship 450.0 cakes to IA
open warehouse in annarbor
and ship 300.0 cakes to WI
and ship 600.0 cakes to MI
```

Problem 2 -- Even *math* is political?

Governor Gary Mander of the state of "Win"sconsin is redrawing the state's congressional districts (gerrymandering) so that he can ensure his preferred congressional candidates win every election. The state consists of ten cities, and the numbers of registered members of Party 1 and Party 2 (in thousands) in each city are shown below

City	Party 1	Party 2
1	80	34
2	60	44
3	40	44
4	20	24
5	40	114
6	40	64
7	70	14
8	50	44
9	70	54
10	70	64

Winsconsin has 5 congressional representatives, meaning these 10 cities are divided up into 5 different districts, and each district elects one representative. There are some restrictions on how these districts can be formed:

- All voters in a city must belong to the same district
- Each district must constain between 150,000 and 250,000 voters

Governer Mander is a member of Party 1. Make the simplifying assumption that every voter will vote for the party they are registered to, regardless of the candidate. In other words, there are no "independent" voters in Winsconsin.

(a) Formulate and solve an integer programming model to help Governer Mander maximize the number of representatives elected from Party 1. Assume that representatives are elected by simple majority, meaning that if a district contains at more Party 1 voters than Party 2 voters, a Party 1 representative is elected.

Hints to get you started:

- Create binary variables that assign each city $i = 1, 2, \ldots, 10$ to one of the five districts $j = 1, 2, \ldots, 5$
- Create variables to track the number of Party 1 and Party 2 voters in each district $j = 1, 2, \ldots, 5$
- Create binary variables that represent party 1 winning in a given district $j = 1, 2, \ldots, 5$. We'll want to maximize the total number of these that are 1 in the solution.
- Use the Slide of Tricks to enforce that if the above binary variable is 1, then there must be more Party 1 voters than Party 2 voters in that district

(b) Modify your model to maximize the number of party 2 voters who win elections.

Solution Problem 2

(a) Formulate and solve an integer programming model to help Governer Mander maximize the number of representatives elected from Party 1. Assume that representatives are elected by simple majority, meaning that if a district contains at more Party 1 voters than Party 2 voters, a Party 1 representative is elected.

In [6]:

```julia
using JuMP, Cbc

C = 1:10
D = 1:5

ρ = Dict(zip(C,[80 60 40 20 40 40 70 50 70 70]))
δ = Dict(zip(C,[34 44 44 24 114 64 14 44 54 64]))

m = Model(Cbc.Optimizer)

@variable(m, x[C,D], Bin)
@variable(m, p[D] >= 0)
@variable(m, q[D] >= 0)
@variable(m, z[D], Bin)

@objective(m, Max, sum(z))

@constraint(m, tot_p1[j in D], sum(ρ[i]*x[i,j] for i in C) == p[j])
@constraint(m, tot_p2[j in D], sum(δ[i]*x[i,j] for i in C) == q[j])
@constraint(m, assign[i in C], sum(x[i,j] for j in D) == 1)
@constraint(m, min_vote[j in D], p[j] + q[j] >= 150)
@constraint(m, max_vote[j in D], p[j] + q[j] <= 250)
@constraint(m ,trix[j in D], p[j] - q[j] + (-250-1)z[j] >= -250)

set_silent(m)
optimize!(m)

println("Total # of party 1 wins: ", sum(value.(z)))
```

```
Total # of party 1 wins: 4.0
```

In [5]:

```julia
using JuMP, Cbc

C = 1:10
D = 1:5

ρ = Dict(zip(C,[80 60 40 20 40 40 70 50 70 70]))
δ = Dict(zip(C,[34 44 44 24 114 64 14 44 54 64]))

m = Model(Cbc.Optimizer)

@variable(m, x[C,D], Bin)
@variable(m, p[D] >= 0)
@variable(m, q[D] >= 0)
@variable(m, y[D], Bin)

@objective(m, Max, sum(y))

@constraint(m, tot_p1[j in D], sum(ρ[i]*x[i,j] for i in C) == p[j])
@constraint(m, tot_p2[j in D], sum(δ[i]*x[i,j] for i in C) == q[j])
@constraint(m, assign[i in C], sum(x[i,j] for j in D) == 1)
@constraint(m, min_vote[j in D], p[j] + q[j] >= 150)
@constraint(m, max_vote[j in D], p[j] + q[j] <= 250)
@constraint(m ,trix[j in D], q[j] - p[j] + (-250-1)y[j] >= -250)

set_silent(m)
optimize!(m)

println("Total # of party 2 wins: ", sum(value.(y)))
```

```
Total # of party 2 wins: 3.0
```