Hot Air

Fleet assignment is a hard and critical problem in airline management. Here we will go through the solution to a simplified version.

Problem 1

- Maximize the profit, i.e. the ticket sales subtracted the flying cost

Sets

- Cities: $c \in Cities = \{"Brownsville", "Dallas", "Austin", "ElPaso"\}$
- Time: $t \in Time = \{1, 2, 3\}$

Parameters

- Passenger demand: $D_{t,cf,cf}$: The number of passengers flying from city cf to ct in timeperiod t
- Ticket price: $P_{cf,ct}$. Paid for each passenger flying from city cf to city ct
- Take off cost: $TOC_{cf,ct}$. Cost of flying from city cf to city ct
- No. of hangar spots available: H_c. No of planes which can over-night in city c
- Plane capacity: $PlaneCapacity = 120$

Decision variables

- No of planes flying from city $c1$ to city $c2$ in timeperiod t: $y_{t,c1,c2} \in \{0, 1, 2, 3, 4\}$
- No of passengers flying from city $c1$ to city $c2$ in timeperiod t: $x_{t,c1,c2} \geq 0$

Model

Objective:

- Maximize profit:

$$\sum_{t,cf,ct} P_{cf,ct} \cdot x_{t,cf,ct} -$$

$$\sum_{t,cf,ct} TOC_{cf,ct} \cdot y_{t,cf,ct}$$

Constraints:

- Ensure that not more passengers are transported than the demand:

$$x_{t,cf,ct} \leq D_{t,cf,ct} \ \forall \ cf, ct, t$$

- Ensure that there are seas to all passengers transported:

$$x_{t,cf,ct} \leq PlaneCapacity \cdot y_{t,cf,ct} \ \forall \ cf, ct, t$$

- Ensures that the correct number of planes overnight, i.e. take off from a city in time period 1, :

$$\sum_{cf,ct} y_{1,cf,ct} = H_{cf} \ \forall \ cf, ct$$

- Balance constraint: The number of planes which arrive to a city in the previous time period **has** to take off in the next time period:

$$\sum_{c2} y_{t,c2,c1} = \sum_{c2} (y[t+1, c1, c2] | (t < T) + y[1, c1, c2] | (t = T)) \ \forall \ c1, c2, t$$

  Notice, that for a plane to **stay** in a city, it simply takes-off and lands in the same city, not carrying passengers and costing nothing. This is some-what of a trick, which makes the model elegant.

The full model in Julia/JuMP, available with the name

`HotAir1.jl`

from the book web-site, is given below:

```
#*************************************************************************
# HotAir Assignment, Question 1, "Mathematical Programming Modelling" (42112)
using JuMP
using HiGHS
#*************************************************************************
```

```
#*********************************************************************
# PARAMETERS
cities = ["Brownsville", "Dallas", "Austin", "El Paso"]
C=length(cities)
time = [1,2,3]
T=length(time)

# ticket-prices ticket_price[cf,ct]
P=[
  0  99  89 139;
109   0  99 169;
109 104   0 129;
159 149 119   0]

# takeoff-cost takeoff_cost[cf,ct]
TOC=[
   0  5100  4400  8000;
5100     0 11200  6900;
4400 11200     0  5700;
8000  6900  5700     0]

# passenger demand passengers[t,cf,ct]
D=zeros(Int16,T,C,C)
D[1,:,:]=
[  # timeperiod 1
   0  50  53  14 ;
  84   0  80  21 ;
  17  58   0  40 ;
  31  79  34   0
   ]
D[2,:,:]=
[  # timeperiod 2
   0  15  53  52 ;
  17   0 134  29 ;
  24 128   0  99 ;
  23  15  30   0
]
D[3,:,:]=
[  # timeperiod 3
   0   3  16   9 ;
  48   0 104  48 ;
  62  92   0  68 ;
  13  15  21   0
   ]
# plane starting places
```

```
H=[2 1 1 0]
PlaneCapacity=120
#*************************************************************************

#*************************************************************************
# Model
hotair = Model(HiGHS.Optimizer)

# passenters transported in time-period t from city cf to city ct
@variable(hotair, x[t=1:T,cf=1:C,ct=1:C] >= 0)

# planes flying in timeperiod t from city cf to city ct
@variable(hotair, 0 <= y[1:T,cf=1:C,ct=1:C] <= 4, Int)

# Maximize profit
@objective(hotair, Max,
           sum( P[cf,ct]*x[t,cf,ct] for t=1:T, cf=1:C, ct=1:C if cf != ct) -
           sum( TOC[cf,ct]*y[t,cf,ct] for t=1:T, cf=1:C, ct=1:C if cf != ct) )

# demand limit
@constraint(hotair, [t=1:T,cf=1:C,ct=1:C],
            x[t,cf,ct] <= D[t,cf,ct] )

# plain capacity limit
@constraint(hotair, [t=1:T,cf=1:C,ct=1:C],
            x[t,cf,ct] <= PlaneCapacity*y[t,cf,ct] )


# where the planes over-night
@constraint(hotair, [cf=1:C],
            sum( y[1,cf,ct] for ct=1:C) == H[cf] )

# plane balance, what goes in must come out
@constraint(hotair, [t=1:T,c1=1:C],
sum( y[t,c2,c1] for c2=1:C) == sum( (t<T ? y[t+1,c1,c2] : y[1,c1,c2]) for c2=1:C) )
#*************************************************************************

#*************************************************************************
# solve
optimize!(hotair)
println("Termination status: $(termination_status(hotair))")
#*************************************************************************

#*************************************************************************
# Report results
println("------------------------------------");
```

```
if termination_status(hotair) == MOI.OPTIMAL
    println("RESULTS:")
    println("objective = $(objective_value(hotair))")
    yres=round.(Int64,value.(y))
    xres=round.(Int64,value.(x))
    for t=1:T
        println("Time step: $(t)")
        for cf=1:C # from
            for ct=1:C # to
                if yres[t,cf,ct]>0
                    println(" Plane: $(yres[t,cf,ct])    Passengers: $(xres[t,cf,ct])  $(cit
                end
            end
        end
        println("")
    end
else
  println("  No solution")
end
println("---------------------------------------");
#*****************************************************************************
```

Comments to the Julia/JuMP program:

- Notice: Again we use the conditional if in a constraint to differentiate between time being less than T and being equal to T

Hot Air considers to build hangars in other cities. Which cities should they choose and how much can they earn more ?

Problem 2

- Maximize the profit, when allowing the 4 planes to overnight in the most profitable cities

New Decision variables

- No. of hangar spots available: $v_c \in \{0, 1, 2, 3, 4\}$

New and updated constraints:

- Ensures that the correct number of planes overnight, i.e. take off from a city in time period 1:

$$\sum_{cf,ct} y_{1,cf,ct} = v_{cf} \ \forall \ cf, ct$$

- Ensure that exactly 4 hangars are built:

$$sum_c v_c = 4$$

Notice, this is actually a relaxation and we can achieve a higher profit, but we of course do not consider the costs of establishing he hangars.

The full model in Julia/JuMP, available with the name

`HotAir2.jl`

from the book web-site, is given below:

```
#*****************************************************************************
# HotAir Assignment, Question 2, "Mathematical Programming Modelling" (42112)
using JuMP
using HiGHS
#*****************************************************************************

#*****************************************************************************
# PARAMETERS
cities = ["Brownsville", "Dallas", "Austin", "El Paso"]
C=length(cities)
time = [1,2,3]
T=length(time)

# ticket-prices P[cf,ct]
P=[
  0  99  89 139;
109   0  99 169;
109 104   0 129;
159 149 119   0]

# takeoff cost TOC[cf,ct]
TOC=[
   0  5100  4400  8000;
5100     0 11200  6900;
4400 11200     0  5700;
8000  6900  5700     0]

# passenger demand D[t,cf,ct]
D=zeros(Int16,T,C,C)
D[1,:,:]=
[  # timeperiod 1
   0  50  53  14 ;
  84   0  80  21 ;
```

```
  17  58   0  40 ;
  31  79  34   0
   ]
D[2,:,:]=
[  # timeperiod 2
   0  15  53  52 ;
  17   0 134  29 ;
  24 128   0  99 ;
  23  15  30   0
]
D[3,:,:]=
[  # timeperiod 3
   0   3  16   9 ;
  48   0 104  48 ;
  62  92   0  68 ;
  13  15  21   0
   ]
# plane starting places
H=[2 1 1 0]
PlaneCapacity=120
#*****************************************************************************

#*****************************************************************************
# Model
hotair = Model(HiGHS.Optimizer)

# passangers transported in time-period t from city cf to city ct
@variable(hotair, x[t=1:T,cf=1:C,ct=1:C] >= 0)

# planes flying in timeperiod t from city cf to city ct
@variable(hotair, 0 <= y[t=1:T,cf=1:C,ct=1:C] <= 4, Int)

# planes overnighting in city c
@variable(hotair, 0 <= v[c=1:C] <= 4, Int)

# Maximize profit
@objective(hotair, Max,
           sum( P[cf,ct]*x[t,cf,ct] for
                t=1:T, cf=1:C, ct=1:C if cf != ct) -
           sum(  TOC[cf,ct]*y[t,cf,ct] for
                 t=1:T, cf=1:C, ct=1:C if cf != ct) )

# demand limit
@constraint(hotair, [t=1:T,cf=1:C,ct=1:C],
            x[t,cf,ct] <= D[t,cf,ct] )
```

```
# plain capacity limit
@constraint(hotair, [t=1:T,cf=1:C,ct=1:C],
            x[t,cf,ct] <= PlaneCapacity*y[t,cf,ct] )


# where the planes over-night
@constraint(hotair, [cf=1:C],
            sum( y[1,cf,ct] for ct=1:C) == v[cf] )

# limit the number of overnights
@constraint(hotair, sum( v[c] for c=1:C) == 4 )

# plane balance, what goes in must come out
@constraint(hotair, plane_balance[t=1:T,c1=1:C],
sum( y[t,c2,c1] for c2=1:C) ==
sum( (t<T ? y[t+1,c1,c2] : y[1,c1,c2]) for c2=1:C) )
#*****************************************************************************

#*****************************************************************************
# solve
optimize!(hotair)
println("Termination status: $(termination_status(hotair))")
#*****************************************************************************

#*****************************************************************************
# Report results
println("-------------------------------------");
if termination_status(hotair) == MOI.OPTIMAL
    println("RESULTS:")
    println("objective = $(objective_value(hotair))")
    yres=round.(Int64,value.(y))
    xres=round.(Int64,value.(x))
    for t=1:T
        println("Time step: $(t)")
        for cf=1:C # from
            for ct=1:C # to
                if yres[t,cf,ct]>0
                    println(" Plane: $(yres[t,cf,ct])    Passengers: $(xres[t,cf,ct])  $(cit
                end
            end
        end
        println("")
    end
else
  println("  No solution")
end
```

```
println("-------------------------------------");
#*****************************************************************************
```

Management wants a print-out of the profit on each route. This is actually **not** a mathematical problem, but a Julia programming problem, when presenting the solution.

Problem 3

The full model in Julia/JuMP, available with the name

`HotAir3.jl`

from the book web-site, is given below:

```
#*****************************************************************************
# HotAir Assignment, Question 3, "Mathematical Programming Modelling" (42112)
using JuMP
using HiGHS
#*****************************************************************************

#*****************************************************************************
# PARAMETERS
cities = ["Brownsville", "Dallas", "Austin", "El Paso"]
C=length(cities)
time = [1,2,3]
T=length(time)

# ticket-prices P[cf,ct]
P=[
  0  99  89 139;
109   0  99 169;
109 104   0 129;
159 149 119   0]

# takeoff cost TOC[cf,ct]
TOC=[
   0  5100  4400  8000;
5100     0 11200  6900;
4400 11200     0  5700;
8000  6900  5700     0]

# passenger demand D[t,cf,ct]
D=zeros(Int16,T,C,C)
```

```
D[1,:,:]=
[  # timeperiod 1
   0  50  53  14 ;
  84   0  80  21 ;
  17  58   0  40 ;
  31  79  34   0
   ]
D[2,:,:]=
[  # timeperiod 2
   0  15  53  52 ;
  17   0 134  29 ;
  24 128   0  99 ;
  23  15  30   0
]
D[3,:,:]=
[  # timeperiod 3
   0   3  16   9 ;
  48   0 104  48 ;
  62  92   0  68 ;
  13  15  21   0
   ]
# plane starting places
H=[2 1 1 0]
PlaneCapacity=120
#*****************************************************************************

#*****************************************************************************
# Model
hotair = Model(HiGHS.Optimizer)

# passangers transported in time-period t from city cf to city ct
@variable(hotair, x[t=1:T,cf=1:C,ct=1:C] >= 0)

# planes flying in timeperiod t from city cf to city ct
@variable(hotair, 0 <= y[1:T,1:C,1:C] <= 4, Int)

# planes overnighting in city c
@variable(hotair, 0 <= v[1:C] <= 4, Int)

# Maximize profit
@objective(hotair, Max,
           sum( P[cf,ct]*x[t,cf,ct] for
                t=1:T, cf=1:C, ct=1:C if cf != ct) -
           sum(  TOC[cf,ct]*y[t,cf,ct] for
                 t=1:T, cf=1:C, ct=1:C if cf != ct) )
```

```
# demand limit
@constraint(hotair, [t=1:T,cf=1:C,ct=1:C],
            x[t,cf,ct] <= D[t,cf,ct] )

# plain capacity limit
@constraint(hotair, [t=1:T,cf=1:C,ct=1:C],
            x[t,cf,ct] <= PlaneCapacity*y[t,cf,ct] )


# where the planes over-night
@constraint(hotair, [cf=1:C],
            sum( y[1,cf,ct] for ct=1:C) == v[cf] )

# limit the number of overnights
@constraint(hotair, sum( v[c] for c=1:C) == 4 )

# plane balance, what goes in must come out
@constraint(hotair, [t=1:T,c1=1:C],
sum( y[t,c2,c1] for c2=1:C) ==
sum( (t<T ? y[t+1,c1,c2] : y[1,c1,c2]) for c2=1:C) )
#*****************************************************************************

#*****************************************************************************
# solve
optimize!(hotair)
println("Termination status: $(termination_status(hotair))")
#*****************************************************************************

#*****************************************************************************
# Report results
println("------------------------------------");
if termination_status(hotair) == MOI.OPTIMAL
    println("RESULTS:")
    println("objective = $(objective_value(hotair))")
    yres=round.(Int64,value.(y))
    xres=round.(Int64,value.(x))
    for t=1:T
        println("Time step: $(t)")
        for i=1:C # from
            for j=1:C # to
                if yres[t,i,j]>0

print( " Plane: $(yres[t,i,j])\t Passengers: $(xres[t,i,j])\t")
print( "$(cities[i])  - > $(cities[j])")
print( "\t Profit: $((xres[t,i,j]*P[i,j])-TOC[i,j])\t")
print( "Take off Cost: $(TOC[i,j])")
```

```
println( "\t Total Ticket sale: $(xres[t,i,j]*P[i,j])")
                end
            end
        end
    end
else
  println("  No solution")
end
println("---------------------------------------");
#*****************************************************************************
```

The management now forces you to ensure that only profitable flights are flown. This ignores the network effect, ie. that we may fly a plane at a loss from one city to another to make a larger profit from that other city in the next time period. The only thing necessary is to add an extra constraint for each city pair and time period.

Problem 4

New constraints:

- Calculate the profit for **each**, flight and force profitability of that flight:

$$P_{i,j} \cdot x_{t,i,j} - TOC_{i,j} \cdot y_{t,i,j} \geq 0 \; \forall \; i, j, t$$

The full model in Julia/JuMP, available with the name

`HotAir4.jl`

from the book web-site, is given below:

```
#*****************************************************************************
# HotAir Assignment, Question 4, "Mathematical Programming Modelling" (42112)
using JuMP
using HiGHS
#*****************************************************************************

#*****************************************************************************
# PARAMETERS
cities = ["Brownsville", "Dallas", "Austin", "El Paso"]
C=length(cities)
time = [1,2,3]
T=length(time)
```

```
# ticket-prices P[cf,ct]
P=[
  0  99  89 139;
109   0  99 169;
109 104   0 129;
159 149 119   0]

# takeoff cost TOC[cf,ct]
TOC=[
   0  5100  4400  8000;
5100     0 11200  6900;
4400 11200     0  5700;
8000  6900  5700     0]

# passenger demand D[t,cf,ct]
D=zeros(Int16,T,C,C)
D[1,:,:]=
[  # timeperiod 1
   0  50  53  14 ;
  84   0  80  21 ;
  17  58   0  40 ;
  31  79  34   0
   ]
D[2,:,:]=
[  # timeperiod 2
   0  15  53  52 ;
  17   0 134  29 ;
  24 128   0  99 ;
  23  15  30   0
]
D[3,:,:]=
[  # timeperiod 3
   0   3  16   9 ;
  48   0 104  48 ;
  62  92   0  68 ;
  13  15  21   0
   ]
# plane starting places
H=[2 1 1 0]
PlaneCapacity=120
#*************************************************************************

#*************************************************************************
# Model
hotair = Model(HiGHS.Optimizer)
```

```
# passangers transported in time-period t from city cf to city ct
@variable(hotair, x[t=1:T,cf=1:C,ct=1:C] >= 0)

# planes flying in timeperiod t from city cf to city ct
@variable(hotair, 0 <= y[t=1:T,cf=1:C,ct=1:C] <= 4, Int)

# planes overnighting in city c
@variable(hotair, 0 <= v[1:C] <= 4, Int)

# Maximize profit
@objective(hotair, Max,
           sum( P[cf,ct]*x[t,cf,ct] for
                t=1:T, cf=1:C, ct=1:C if cf != ct) -
           sum(  TOC[cf,ct]*y[t,cf,ct] for
                 t=1:T, cf=1:C, ct=1:C if cf != ct) )

# demand limit
@constraint(hotair, [t=1:T,cf=1:C,ct=1:C],
            x[t,cf,ct] <= D[t,cf,ct] )

# plain capacity limit
@constraint(hotair, [t=1:T,cf=1:C,ct=1:C],
            x[t,cf,ct] <= PlaneCapacity*y[t,cf,ct] )


# where the planes over-night
@constraint(hotair, [cf=1:C],
            sum( y[1,cf,ct] for ct=1:C) == v[cf] )

# limit the number of overnights
@constraint(hotair, sum( v[c] for c=1:C) == 4 )

# plane balance, what goes in must come out
@constraint(hotair, [t=1:T,c1=1:C],
sum( y[t,c2,c1] for c2=1:C) ==
sum( (t<T ? y[t+1,c1,c2] : y[1,c1,c2]) for c2=1:C) )

# only fly profitable tours
@constraint(hotair, [t=1:T,i=1:C,j=1:C],
            P[i,j]*x[t,i,j] -
            TOC[i,j]*y[t,i,j] >= 0)
#*****************************************************************************

#*****************************************************************************
# solve
optimize!(hotair)
```

```
println("Termination status: $(termination_status(hotair))")
#*************************************************************************

#*************************************************************************
# Report results
println("---------------------------------------");
if termination_status(hotair) == MOI.OPTIMAL
    println("RESULTS:")
    println("objective = $(objective_value(hotair))")
    yres=round.(Int64,value.(y))
    xres=round.(Int64,value.(x))
    for t=1:T
        println("Time step: $(t)")
        for i=1:C # from
            for j=1:C # to
                if yres[t,i,j]>0

print( " Plane: $(yres[t,i,j])\t Passengers: $(xres[t,i,j])\t")
print( "$(cities[i])  - > $(cities[j])")
print( "\t Profit: $((xres[t,i,j]*P[i,j])-TOC[i,j])\t")
print( "Take off Cost: $(TOC[i,j])")
println( "\t Total Ticket sale: $(xres[t,i,j]*P[i,j])")
                end
            end
        end
    end
else
  println("  No solution")
end
println("---------------------------------------");
#*************************************************************************
```

Introducing discount tickets is now studied. This is an advanced problem which is quite hard to solve. A new variable is necessary

Problem 5

- Maximize the profit including both the income from the normal tickets and the discount tickets.

New Decision variables

- Number of discount passengers transported from city cf over city cm to city ct: $0 \leq u_{t,cf,cm,ct} \leq 480 \in Z+$

Model

sum(0.7*P[cf,ct]*u[t,cf,cm,ct] for t=1:T, cf=1:C, cm=1:C, ct=1:C) **Objective:**

- Maximize profit:

$$\sum_{t,cf,ct} P_{cf,ct} \cdot x_{t,cf,ct} + \sum_{t,cf,cm,ct} 0.7 \cdot P_{cf,ct} \cdot u_{t,cf,cm,ct} - \sum_{t,cf,ct} TOC_{cf,ct} \cdot y_{t,cf,ct}$$

Constraints:

- Ensure that there are seas to all passengers transported:

$$x_{t,cf,ct} + \sum_{c} u_{t,cf,ct,c} + \sum_{c} u_{t-1,c,cf,ct} | (t > 1) \leq PlaneCapacity \cdot y_{t,cf,ct} \ \forall \ cf, ct, t$$

- Ensure that not more discount passengers are transported than the demand:

$$x_{t,cf,ct} \leq 0.5 \cdot D_{t,cf,ct} \ \forall \ cf, ct, t$$

Notice the way the discount passengers are ensured seats on the planes.

The full model in Julia/JuMP, available with the name

`HotAir5.jl`

from the book web-site, is given below:

```
#*************************************************************************
# HotAir Assignment, Question 5, "Mathematical Programming Modelling" (42112)
using JuMP
using HiGHS
#*************************************************************************

#*************************************************************************
# PARAMETERS
```

```
cities = ["Brownsville", "Dallas", "Austin", "El Paso"]
C=length(cities)
time = [1,2,3]
T=length(time)

# ticket-prices P[cf,ct]
P=[
  0  99  89 139;
109   0  99 169;
109 104   0 129;
159 149 119   0]

# takeoff cost TOC[cf,ct]
TOC=[
   0  5100  4400  8000;
5100     0 11200  6900;
4400 11200     0  5700;
8000  6900  5700     0]

# passenger demand D[t,cf,ct]
D=zeros(Int16,T,C,C)
D[1,:,:]=
[  # timeperiod 1
   0  50  53  14 ;
  84   0  80  21 ;
  17  58   0  40 ;
  31  79  34   0
   ]
D[2,:,:]=
[  # timeperiod 2
   0  15  53  52 ;
  17   0 134  29 ;
  24 128   0  99 ;
  23  15  30   0
]
D[3,:,:]=
[  # timeperiod 3
   0   3  16   9 ;
  48   0 104  48 ;
  62  92   0  68 ;
  13  15  21   0
   ]
# plane starting places
H=[2 1 1 0]
PlaneCapacity=120
#*****************************************************************************
```

```
#**************************************************************************
# Model
hotair = Model(HiGHS.Optimizer)

# passengers transported in time-period t from city i to city j
@variable(hotair, x[t=1:T,cf=1:C,ct=1:C] >= 0)
for t=1:T
    for c=1:C
        set_upper_bound(x[t,c,c],0)
    end
end

# planes flying in timeperiod t from city i to city j
@variable(hotair, 0 <= y[t=1:T,cf=1:C,ct=1:C] <= 4, Int)

# planes overnighting in city c
@variable(hotair, 0 <= v[c=1:C] <= 4, Int)

# passengers transported in time-period t from city cf to city cm
# (city midt) and in timepeiod t+1 from city cm to city ct
@variable(hotair, 0 <= u[t=1:T,cf=1:C,cm=1:C,ct=1:C] <= 480, Int)
for t=1:T
    for c1=1:C
        for c2=1:C
                set_upper_bound(u[t,c1,c1,c2],0)
                set_upper_bound(u[t,c1,c2,c2],0)
        end
    end
end
for cf=1:C
    for cm=1:C
        for ct=1:C
            set_upper_bound(u[3,cf,cm,ct],0)
        end
    end
end

# Maximize profit
@objective(hotair, Max,
           sum( P[cf,ct]*x[t,cf,ct] for t=1:T, cf=1:C, ct=1:C) +
           sum( 0.7*P[cf,ct]*u[t,cf,cm,ct]
                for t=1:T, cf=1:C, cm=1:C, ct=1:C) -
           sum( TOC[cf,ct]*y[t,cf,ct] for t=1:T, cf=1:C, ct=1:C)
           )
```

```
# demand limit
@constraint(hotair, [t=1:T,cf=1:C,ct=1:C],
            x[t,cf,ct] <= D[t,cf,ct] )

# discount demand limit
@constraint(hotair, [t=1:T,cf=1:C,ct=1:C],
            sum(u[t,cf,cm,ct] for cm=1:C) <= 0.5*D[t,cf,ct] )

# plain capacity limit
@constraint(hotair, [t=1:T,cf=1:C,ct=1:C],
            x[t,cf,ct] + sum( u[t,cf,ct,c] for c=1:C) +
            (t>1 ? (sum( u[t-1,c,cf,ct] for c=1:C)) : 0) <=
            PlaneCapacity*y[t,cf,ct] )

# where the planes over-night, i.e. starting from in the morning
@constraint(hotair, [cf=1:C],
            sum( y[1,cf,c] for c=1:C) == v[cf] )

# limit the number of overnights
@constraint(hotair, sum( v[c] for c=1:C) == 4 )

# plane balance, what goes in must come out
@constraint(hotair, [t=1:T,c1=1:C],
sum( y[t,c2,c1] for c2=1:C) ==
sum( (t<T ? y[t+1,c1,c2] : y[1,c1,c2]) for c2=1:C) )
#**************************************************************************

#**************************************************************************
# solve
optimize!(hotair)
println("Termination status: $(termination_status(hotair))")
#**************************************************************************

#**************************************************************************
# Report results
println("-------------------------------------");
if termination_status(hotair) == MOI.OPTIMAL
    println("RESULTS:")
    println("objective = $(objective_value(hotair))")
    yres=round.(Int64,value.(y))
    xres=round.(Int64,value.(x))
    ures=round.(Int64,value.(u))
    for t=1:T
        println("Time step: $(t)")
        for cf=1:C # from
            for ct=1:C # to
```

```
                if yres[t,cf,ct]>0
                    first=sum( ures[t,cf,ct,k] for k=1:C)
                    second=sum( ures[t,k,cf,ct] for k=1:C)

                    print( " Plane: $(yres[t,cf,ct])    \t")
                    print( " Passengers: $(xres[t,cf,ct]) first: $(first)   second: $(second
                    print( " $(cities[cf])  - > $(cities[ct] )   \t ")
                    print( " Profit: $((xres[t,cf,ct]*P[cf,ct]) - TOC[cf,ct]) \t Take off Co
                    println( " Total Ticket sale: $((xres[t,cf,ct]*P[cf,ct]) )     ")
                end
            end
        end
    end

else
  println("  No solution")
end
println("---------------------------------------");
#*****************************************************************************

#*****************************************************************************
println("Successfull end of $(PROGRAM_FILE)")
#*****************************************************************************
```