In [5]:

```
#Problem 1-2 alternate
using JuMP, Cbc

m=Model()

sites = [:1,:2,:3,:4,:5,:6,:7]
time = Dict(zip(sites, [50, 38, 45, 53, 42, 59, 48]))
gas = Dict(zip(sites, [100,130,250,310,220,350,190]))
# gas is calculated by adding the gas required for each plane in each site, example
# site 1 covers planes 1 2 and 4 and 1 2 and 4 require 20, 20 and 60. 20+20+60 = 100, and do for each site

#aircrafts = Dict(zip(sites, [[1,2,4],[2,3,5],[4,7,8,10],[5,6,8,9],[8,9,12],[7,10,11,12,15],[12,13,14,15]]))
#gas = Dict(zip(sites, [[20,20,60],[20,30,80],[60,90,90,110],[80,40,90,100],[90,100,30],[90,110,60,30,60], [30,70,30,60_

@variable(m, x[sites], Bin)

@objective(m, Max, sum(gas[i]*x[i] for i in sites))

@constraint(m, x[1] >= 1)          #aircraft 1 coverage
@constraint(m, x[1] + x[2] >= 1)   #aircraft 2 coverage
@constraint(m, x[2] >= 1)          #aircraft 3 coverage
@constraint(m, x[1] + x[3] >= 1)   #aircraft 4 coverage
@constraint(m, x[2] + x[4] >= 1)   #aircraft 5 coverage
@constraint(m, x[4] >= 1)          #aircraft 6 coverage
@constraint(m, x[3] + x[6] >= 1)   #aircraft 7 coverage
@constraint(m, x[3] + x[4] + x[5] >= 1) #aircraft 8 coverage
@constraint(m, x[4] + x[5] >= 1)   #aircraft 9 coverage
@constraint(m, x[3] + x[6] >= 1)   #aircraft 10 coverage
@constraint(m, x[6] >= 1)          #aircraft 11 coverage
@constraint(m, x[6] + x[7] >= 1)   #aircraft 12 coverage
@constraint(m, x[7] >= 1)          #aircraft 13 coverage
@constraint(m, x[7] >= 1)          #aircraft 14 coverage
@constraint(m, x[6] + x[7] >= 1)   #aircraft 15 coverage

@constraint(m, limit[i in sites], time[i]*x[i] <= 175) #time constraint to fill gas

set_optimizer(m, Cbc.Optimizer)
optimize!(m)
```

```
Welcome to the CBC MILP Solver
Version: 2.10.8
Build Date: Jan  1 1970

command line - Cbc_C_Interface -solve -quit (default strategy 1)
Continuous objective value is 1550 - 0.00 seconds
Cgl0004I processed model has 0 rows, 0 columns (0 integer (0 of which binary)) and 0 elements
Cbc3007W No integer variables - nothing to do
Cuts at root node changed objective from -1550 to -1.79769e+308
Probing was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
Gomory was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
Knapsack was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
Clique was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
MixedIntegerRounding2 was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 se
conds)
FlowCover was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
TwoMirCuts was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
ZeroHalf was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)

Result - Optimal solution found

Objective value:                1550.00000000
Enumerated nodes:               0
Total iterations:               0
Time (CPU seconds):             0.00
Time (Wallclock seconds):       0.01

Total time (CPU seconds):       0.00   (Wallclock seconds):       0.01
```

In [4]:

```
#Problem 1-2
#using JuMP, Cbc

#m=Model()

#aircrafts = [:1,:2,:3,:4,:5,:6,:7,:8, :9, :10, :11, :12, :13, :14, :15]
#gas = Dict(zip(aircrafts, [20, 20,30,60,80,40,90,90,100,110,60,30,70,30,60]))
#time = Dict(zip(aircrafts, [50, 88, 38, 95, 91, 53, 104, 150, 95, 104, 59, 107, 48, 48, 107]))


#@variable(m, x[aircrafts], Bin)


#@objective(m, Max, sum(gas[i]*x[i] for i in aircrafts))
```

```
#@constraint(m, x[1] >= 1)          #aircraft 1 coverage
#@constraint(m, x[1] + x[2] >= 1)   #aircraft 2 coverage
#@constraint(m, x[2] >= 1)          #aircraft 3 coverage
#@constraint(m, x[1] + x[3] >= 1)   #aircraft 4 coverage
#@constraint(m, x[2] + x[4] >= 1)   #aircraft 5 coverage
#@constraint(m, x[4] >= 1)          #aircraft 6 coverage
#@constraint(m, x[3] + x[6] >= 1)   #aircraft 7 coverage
#@constraint(m, x[3] + x[4] + x[5] >= 1) #aircraft 8 coverage
#@constraint(m, x[4] + x[5] >= 1)   #aircraft 9 coverage
#@constraint(m, x[3] + x[6] >= 1)   #aircraft 10 coverage
#@constraint(m, x[6] >= 1)          #aircraft 11 coverage
#@constraint(m, x[6] + x[7] >= 1)   #aircraft 12 coverage
#@constraint(m, x[7] >= 1)          #aircraft 13 coverage
#@constraint(m, x[7] >= 1)          #aircraft 14 coverage
#@constraint(m, x[6] + x[7] >= 1)   #aircraft 15 coverage

#@constraint(m, limit[i in aircrafts], time[i]*x[i] <= 175)

#set_optimizer(m, Cbc.Optimizer)
#optimize!(m)
```

In [ ]: