CS 524 HW9 Sparsh Agarwal 9075905142

Q1. Democrats advantage is shown in solution

In [31]:
```
A = [ 80 34
      60 44
      40 44
      20 24
      40 114
      40 64
      70 14
      50 44
      70 54
      70 64];

r = A[:,1];
d = A[:,2];


using JuMP, Gurobi

m = Model(solver = GurobiSolver())
@variable(m, R[1:5,1:10], Bin)
@variable(m, z[1:5], Bin)
for j = 1:10
    @constraint(m, sum(R[i,j] for i in 1:5)==1)
end

for i = 1:5
    @constraint(m, sum(R[i,j]*d[j] - R[i,j]*r[j] for j in 1:10)>= -150*(
1-z[i]))
    @constraint(m, sum(R[i,j]*(r[j]+d[j]) for j in 1:10) >=150)
    @constraint(m, sum(R[i,j]*(r[j]+d[j]) for j in 1:10) <=250)
end

@objective(m, Max, sum(z[i] for i in 1:5))

status = solve(m)
println(status)
cities = getvalue(R)
println("cities")
for j in 1:10
    print("| ", j," ")
end
println("|")
for i in 1:5
    print("| ")
    for j in 1:10
        if cities[i,j] == 1
            print(" ",1," |")
        else
            print(" ", 0," |")
        end
    end
    println()
end
for i in 1:5
    println("Advantage for democrats in votes per city", i, ": ", sum(ci
ties[i,j]*(d[j]) -cities[i,j]*r[j] for j in 1:10))
end
```

```
Academic license - for non-commercial use only
Optimize a model with 25 rows, 55 columns and 205 nonzeros
Variable types: 0 continuous, 55 integer (55 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+02]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 2e+02]
Found heuristic solution: objective 2.0000000
Presolve time: 0.00s
Presolved: 25 rows, 55 columns, 205 nonzeros
Variable types: 0 continuous, 55 integer (55 binary)

Root relaxation: objective 4.629630e+00, 40 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Wor
k
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0    4.62963    0    9    2.00000    4.62963   131%     -
0s
H    0     0                       3.0000000    4.62963  54.3%     -
0s
     0     0    4.62963    0   14    3.00000    4.62963  54.3%     -
0s
     0     0    4.60955    0   16    3.00000    4.60955  53.7%     -
0s
     0     0    4.51852    0    9    3.00000    4.51852  50.6%     -
0s
     0     0    4.51852    0   16    3.00000    4.51852  50.6%     -
0s
     0     0    4.00000    0   20    3.00000    4.00000  33.3%     -
0s
     0     0    4.00000    0   14    3.00000    4.00000  33.3%     -
0s
     0     0    4.00000    0   17    3.00000    4.00000  33.3%     -
0s
     0     0    4.00000    0   20    3.00000    4.00000  33.3%     -
0s
     0     0    4.00000    0   19    3.00000    4.00000  33.3%     -
0s
     0     0    4.00000    0    8    3.00000    4.00000  33.3%     -
0s
     0     2    4.00000    0    4    3.00000    4.00000  33.3%     -
0s

Cutting planes:
  Clique: 8
  MIR: 1
  StrongCG: 1

Explored 8 nodes (596 simplex iterations) in 0.04 seconds
Thread count was 4 (of 4 available processors)

Solution count 2: 3 2
```

```
Optimal solution found (tolerance 1.00e-04)
Best objective 3.000000000000e+00, best bound 3.000000000000e+00, gap
0.0000%
Optimal
cities
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  0 |  0 |  1 |  1 |  0 |  0 |  0 |  1 |  0 |  0 |
|  0 |  0 |  0 |  0 |  1 |  0 |  0 |  0 |  0 |  0 |
|  0 |  1 |  0 |  0 |  0 |  1 |  0 |  0 |  0 |  0 |
|  1 |  0 |  0 |  0 |  0 |  0 |  0 |  0 |  1 |  0 |
|  0 |  0 |  0 |  0 |  0 |  0 |  1 |  0 |  0 |  1 |
Advantage for democrats in votes per city1: 2.0
Advantage for democrats in votes per city2: 74.0
Advantage for democrats in votes per city3: 8.0
Advantage for democrats in votes per city4: -62.0
Advantage for democrats in votes per city5: -62.0
```

Q2.(a)

In [32]:
```
using JuMP, Gurobi

m = Model(solver = GurobiSolver())

@variable(m, x[1:8,1:8], Bin)
@variable(m, xlr[1:8,1:8], Bin)
@constraint(m, sum(x)==8 )
@constraint(m, xlr .== x[:,end:-1:1])
@constraint(m, cstrA[i in 1:8], sum(x[i,:]) <= 1)
@constraint(m, cstrB[j in 1:8], sum(x[:,j]) <= 1)
for j in 1:8
    @constraint(m, sum(x[i,i+j-1] for i in 1:8-j+1) <= 1)
    @constraint(m, sum(x[i+j-1,i] for i in 1:8-j+1) <= 1)
end
for j in 1:8
    @constraint(m, sum(xlr[i,i+j-1] for i in 1:8-j+1) <= 1)
    @constraint(m, sum(xlr[i+j-1,i] for i in 1:8-j+1) <= 1)
end

solve(m)

for j in 1:8
    println(getvalue(x)[j,:])
end
```

```
Academic license - for non-commercial use only
Optimize a model with 113 rows, 128 columns and 464 nonzeros
Variable types: 0 continuous, 128 integer (128 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [0e+00, 0e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 8e+00]
Found heuristic solution: objective 0.0000000

Explored 0 nodes (0 simplex iterations) in 0.00 seconds
Thread count was 1 (of 4 available processors)

Solution count 1: 0

Optimal solution found (tolerance 1.00e-04)
Best objective 0.000000000000e+00, best bound 0.000000000000e+00, gap
0.0000%
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0]
[0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0]
[0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0]
[0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0]
```

(b)

In [33]:
```
using JuMP, Gurobi

m = Model(solver = GurobiSolver())

@variable(m, x[1:8,1:8], Bin)
@variable(m, xlr[1:8,1:8], Bin)
@constraint(m, sum(x)==8 )
@constraint(m, xlr .== x[:,end:-1:1])
@constraint(m, x .== x[end:-1:1,end:-1:1])
@constraint(m, cstrA[i in 1:8], sum(x[i,:]) <= 1)
@constraint(m, cstrB[j in 1:8], sum(x[:,j]) <= 1)
for j in 1:8
    @constraint(m, sum(x[i,i+j-1] for i in 1:8-j+1) <= 1)
    @constraint(m, sum(x[i+j-1,i] for i in 1:8-j+1) <= 1)
end
for j in 1:8
    @constraint(m, sum(xlr[i,i+j-1] for i in 1:8-j+1) <= 1)
    @constraint(m, sum(xlr[i+j-1,i] for i in 1:8-j+1) <= 1)
end

solve(m)

for j in 1:8
    println(getvalue(x)[j,:])
end
```

```
Academic license - for non-commercial use only
Optimize a model with 177 rows, 128 columns and 592 nonzeros
Variable types: 0 continuous, 128 integer (128 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [0e+00, 0e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 8e+00]
Presolve removed 161 rows and 104 columns
Presolve time: 0.00s
Presolved: 16 rows, 24 columns, 88 nonzeros
Variable types: 0 continuous, 24 integer (24 binary)
Found heuristic solution: objective 0.0000000

Explored 0 nodes (0 simplex iterations) in 0.00 seconds
Thread count was 4 (of 4 available processors)

Solution count 1: 0

Optimal solution found (tolerance 1.00e-04)
Best objective 0.000000000000e+00, best bound 0.000000000000e+00, gap
0.0000%
[0.0, -0.0, -0.0, -0.0, -0.0, 1.0, -0.0, 0.0]
[-0.0, 0.0, -0.0, 1.0, -0.0, -0.0, 0.0, -0.0]
[-0.0, -0.0, 0.0, -0.0, -0.0, 0.0, 1.0, -0.0]
[1.0, -0.0, -0.0, 0.0, 0.0, -0.0, -0.0, -0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0]
[0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

(c)

In [34]:
```
using JuMP, Gurobi

m = Model(solver = GurobiSolver())

@variable(m, x[1:8,1:8], Bin)
# @variable(m, xlr[1:8,1:8], Bin)
# @constraint(m, sum(x)==8 )
# @constraint(m, xlr .== x[:,end:-1:1])
# @constraint(m, x .== x[end:-1:1,end:-1:1])
# @constraint(m, cstrA[i in 1:8], sum(x[i,:]) => 1)
# @constraint(m, cstrB[j in 1:8], sum(x[:,j]) => 1)
# for j in 1:8
#     @constraint(m, sum(x[i,i+j-1] for i in 1:8-j+1) => 1)
#     @constraint(m, sum(x[i+j-1,i] for i in 1:8-j+1) => 1)
# end
# for j in 1:8
#     @constraint(m, sum(xlr[i,i+j-1] for i in 1:8-j+1) => 1)
#     @constraint(m, sum(xlr[i+j-1,i] for i in 1:8-j+1) => 1)
# end

for i in 1:8
    for j in 1:8
        k = j-i;
        kd = i+j;
        @constraint(m,(sum(x[i,:])+sum(x[:,j])+sum(x[l,l+k] for l in min
(8,8-k):max(1,1-k))+sum(x[l,kd-l] for l in max(1,kd-8):min(8,kd-1))) >=
1)
    end
end

@objective(m,Min, sum(x))

solve(m)

println(getvalue(sum(x)))

for j in 1:8
    println(getvalue(x)[j,:])
end
```

```
Academic license - for non-commercial use only
Optimize a model with 64 rows, 64 columns and 1240 nonzeros
Variable types: 0 continuous, 64 integer (64 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 7.0000000
Presolve time: 0.00s
Presolved: 64 rows, 64 columns, 1240 nonzeros
Variable types: 0 continuous, 64 integer (64 binary)

Root relaxation: objective 3.400000e+00, 129 iterations, 0.01 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Wor
k
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0    3.40000    0   44    7.00000    3.40000  51.4%     -
0s
H    0     0                       6.0000000    3.40000  43.3%     -
0s
     0     0    3.50302    0   48    6.00000    3.50302  41.6%     -
0s
     0     0    3.51696    0   46    6.00000    3.51696  41.4%     -
0s
     0     0    3.56498    0   45    6.00000    3.56498  40.6%     -
0s
     0     0    3.58671    0   47    6.00000    3.58671  40.2%     -
0s
     0     0    3.59403    0   50    6.00000    3.59403  40.1%     -
0s
     0     0    3.60939    0   48    6.00000    3.60939  39.8%     -
0s
     0     0    3.61198    0   50    6.00000    3.61198  39.8%     -
0s
     0     0    3.61440    0   49    6.00000    3.61440  39.8%     -
0s
     0     0    3.62118    0   45    6.00000    3.62118  39.6%     -
0s
     0     0    3.62251    0   47    6.00000    3.62251  39.6%     -
0s
     0     0    3.62275    0   47    6.00000    3.62275  39.6%     -
0s
     0     0    3.62275    0   47    6.00000    3.62275  39.6%     -
0s
     0     0    3.65261    0   42    6.00000    3.65261  39.1%     -
0s
     0     0    3.66865    0   44    6.00000    3.66865  38.9%     -
0s
     0     0    3.69971    0   42    6.00000    3.69971  38.3%     -
0s
     0     0    3.70491    0   44    6.00000    3.70491  38.3%     -
0s
     0     0    3.70787    0   43    6.00000    3.70787  38.2%     -
```

0s

0	0	3.70854	0	43	6.00000	3.70854	38.2%	-	0s
0	0	3.70896	0	44	6.00000	3.70896	38.2%	-	0s
0	0	3.70930	0	45	6.00000	3.70930	38.2%	-	0s
0	0	3.72035	0	44	6.00000	3.72035	38.0%	-	0s
0	0	3.72155	0	44	6.00000	3.72155	38.0%	-	0s
0	0	3.72349	0	45	6.00000	3.72349	37.9%	-	0s
0	0	3.72593	0	45	6.00000	3.72593	37.9%	-	0s
0	0	3.72703	0	45	6.00000	3.72703	37.9%	-	0s
0	0	3.72779	0	45	6.00000	3.72779	37.9%	-	0s
0	0	3.72803	0	45	6.00000	3.72803	37.9%	-	0s
0	0	3.73293	0	42	6.00000	3.73293	37.8%	-	0s
0	0	3.73634	0	44	6.00000	3.73634	37.7%	-	0s
0	0	3.73752	0	44	6.00000	3.73752	37.7%	-	0s
0	0	3.74100	0	44	6.00000	3.74100	37.7%	-	0s
0	0	3.74280	0	46	6.00000	3.74280	37.6%	-	0s
0	0	3.74375	0	46	6.00000	3.74375	37.6%	-	0s
0	0	3.74566	0	45	6.00000	3.74566	37.6%	-	0s
0	0	3.74975	0	43	6.00000	3.74975	37.5%	-	0s
0	0	3.74993	0	44	6.00000	3.74993	37.5%	-	0s
0	0	3.75621	0	45	6.00000	3.75621	37.4%	-	0s
0	0	3.75764	0	45	6.00000	3.75764	37.4%	-	0s
0	0	3.76021	0	45	6.00000	3.76021	37.3%	-	0s
0	0	3.76100	0	44	6.00000	3.76100	37.3%	-	0s
0	0	3.76122	0	42	6.00000	3.76122	37.3%	-	0s
0	0	3.76328	0	46	6.00000	3.76328	37.3%	-	0s
0	0	3.76443	0	44	6.00000	3.76443	37.3%	-	0s
0	0	3.76584	0	47	6.00000	3.76584	37.2%	-	0s
0	0	3.76811	0	47	6.00000	3.76811	37.2%	-	0s

```
      0     0    3.77086    0   47    6.00000    3.77086  37.2%     -
0s
      0     0    3.77183    0   47    6.00000    3.77183  37.1%     -
0s
      0     0    3.77243    0   48    6.00000    3.77243  37.1%     -
0s
      0     0    3.77307    0   49    6.00000    3.77307  37.1%     -
0s
      0     0    3.77464    0   47    6.00000    3.77464  37.1%     -
0s
      0     0    3.77479    0   49    6.00000    3.77479  37.1%     -
0s
      0     0    3.77513    0   49    6.00000    3.77513  37.1%     -
0s
      0     2    3.77513    0   49    6.00000    3.77513  37.1%     -
0s
H   69    11                        5.0000000    3.95222  21.0%  25.3
0s

Cutting planes:
  MIR: 43

Explored 90 nodes (3104 simplex iterations) in 0.26 seconds
Thread count was 4 (of 4 available processors)

Solution count 3: 5 6 7

Optimal solution found (tolerance 1.00e-04)
Best objective 5.000000000000e+00, best bound 5.000000000000e+00, gap
0.0000%
5.0
[0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0]
[0.0, 0.0, 0.0, 0.0, 1.0, -0.0, 0.0, -0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0]
[-0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[-0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[-0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

(d) Yes the solution changes to 6 queens

In [35]:
```
using JuMP, Gurobi

m = Model(solver = GurobiSolver())

@variable(m, x[1:8,1:8], Bin)
@variable(m, xlr[1:8,1:8], Bin)
# @constraint(m, sum(x)==8 )
# @constraint(m, xlr .== x[:,end:-1:1])
@constraint(m, x .== x[end:-1:1,end:-1:1])
# @constraint(m, cstrA[i in 1:8], sum(x[i,:]) => 1)
# @constraint(m, cstrB[j in 1:8], sum(x[:,j]) => 1)
# for j in 1:8
#     @constraint(m, sum(x[i,i+j-1] for i in 1:8-j+1) => 1)
#     @constraint(m, sum(x[i+j-1,i] for i in 1:8-j+1) => 1)
# end
# for j in 1:8
#     @constraint(m, sum(xlr[i,i+j-1] for i in 1:8-j+1) => 1)
#     @constraint(m, sum(xlr[i+j-1,i] for i in 1:8-j+1) => 1)
# end

for i in 1:8
    for j in 1:8
        k = j-i;
        kd = i+j;
        @constraint(m,(sum(x[i,:])+sum(x[:,j])+sum(x[l,l+k] for l in min
(8,8-k):max(1,1-k))+sum(x[l,kd-l] for l in max(1,kd-8):min(8,kd-1))) >=
1)
    end
end

@objective(m,Min, sum(x))

solve(m)

println(getvalue(sum(x)))

for j in 1:8
    println(getvalue(x)[j,:])
end
```

```
Academic license - for non-commercial use only
Optimize a model with 128 rows, 128 columns and 1368 nonzeros
Variable types: 0 continuous, 128 integer (128 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 64.0000000
Found heuristic solution: objective 12.0000000
Presolve removed 97 rows and 96 columns
Presolve time: 0.00s
Presolved: 31 rows, 32 columns, 531 nonzeros
Variable types: 0 continuous, 32 integer (32 binary)

Root relaxation: objective 3.793724e+00, 56 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Wor
k
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0    3.79372    0   24   12.00000    3.79372  68.4%     -
0s
H    0     0                       8.0000000    3.79372  52.6%     -
0s
H    0     0                       6.0000000    3.79372  36.8%     -
0s
     0     0    3.97712    0   22    6.00000    3.97712  33.7%     -
0s
     0     0     cutoff    0         6.00000    6.00000  0.00%     -
0s

Cutting planes:
  MIR: 7

Explored 1 nodes (85 simplex iterations) in 0.01 seconds
Thread count was 4 (of 4 available processors)

Solution count 4: 6 8 12 64

Optimal solution found (tolerance 1.00e-04)
Best objective 6.000000000000e+00, best bound 6.000000000000e+00, gap
0.0000%
6.0
[0.0, 0.0, -0.0, -0.0, -0.0, -0.0, -0.0, 0.0]
[1.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0]
[-0.0, -0.0, -0.0, 1.0, 0.0, -0.0, -0.0, -0.0]
[-0.0, -0.0, 0.0, -0.0, 0.0, 1.0, -0.0, -0.0]
[0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

Q3. The code does not use the blending times because the total required for the entire process will be just the addition of all blending times(which is constant) and minimum cleaning time. Therefore, the best order found is 1, 2, 5, 3, 4, 1

In [36]:
```
function getAllSubtours(x)
    nodesRemaining = paints
    subtours = []
    while length(nodesRemaining) > 0
        subtour = getSubtour(x,nodesRemaining[1])
        push!(subtours, subtour)
        nodesRemaining = setdiff(nodesRemaining,subtour)
    end
    return subtours
end

function getSubtour(x,start)
    subtour = [start]
    while true
        j = subtour[end]
        for k in paints
            if x[k,j] == 1
                push!(subtour,k)
                break
            end
        end
        if subtour[end] == start
            break
        end
    end
    return subtour
end


using JuMP, Cbc

c =     [ 0 11 7 13 11
          5 0 13 15 15
          13 15 0 23 11
          9 13 5 0 3
          3 7 7 7 0 ];
paints = [:1,:2,:3,:4,:5];

m = Model(solver = CbcSolver())
@variable(m, x[paints,paints], Bin)
  # must formulate as IP this time
@constraint(m, c1[j in paints], sum( x[i,j] for i in paints ) == 1)
# one out-edge
@constraint(m, c2[i in paints], sum( x[i,j] for j in paints ) == 1)
# one in-edge
@constraint(m, c3[i in paints], x[i,i] == 0 )                       # n
o self-loops
@objective(m, Min, sum( x[i,j]*c[i,j] for i in paints, j in paints ))
# minimize total cost

sols = []

for iters = 1:30
    solve(m)
    println("Tour length: ", getobjectivevalue(m))
    xx = getvalue(x)
```

```
    push!(sols,xx)
    subtours = getAllSubtours(xx)  # get all the subtours
    display(subtours)
    sleep(1)
    len = length(subtours)
    if len == 1                    # solution is just a single tour!
        println("SOLVED!")
        break
    else
        for subtour in subtours
            L = length(subtour)
            @constraint(m, sum( x[subtour[k+1],subtour[k]] for k = 1:L-1
) <= L-2)
            @constraint(m, sum( x[subtour[k],subtour[k+1]] for k = 1:L-1
) <= L-2)
        end
    end
end
```

```
2-element Array{Any,1}:
 [1, 2, 3, 1]
 [4, 5, 4]

Tour length: 37.0

2-element Array{Any,1}:
 [1, 2, 1]
 [3, 4, 5, 3]

Tour length: 39.0

1-element Array{Any,1}:
 [1, 2, 5, 3, 4, 1]

Tour length: 41.0
SOLVED!
```