Homework 6 Solution

August 9, 2024

1 Problem 1 - Machine Scheduling

1.1 (a)

[1]:
```
using Random
seed = 425 # seed for data generation

N = 1:60 # jobs
M = 1:30 # machines

Random.seed!(seed)

# set time lengths of jobs on each machine
a = zeros(length(M),length(N))
for i in M
    for j in N
        a[i,j] = round(6+Random.rand()*Random.rand(6:25),digits=2)
    end
end

# capacity of each machine
b = Dict(zip(M,[12*sum(a[i,j] for j in N)/length(M) for i in M]))

# cost of running jobs on each machine
c = zeros(length(M),length(N))
for i in M
    for j in N
        c[i,j] = 20+round(Random.rand()*Random.rand(20:60),digits=2)
    end
end

# fixed cost of each machine
h = [30*length(M) for i in M]

# duration of each job (for question (c))
d = [Random.rand(1:10)*Random.rand()+10 for j in N];
```

[9]:
```julia
using JuMP, GLPK

# Initialize the model
model = Model(GLPK.Optimizer)

# Decision variables
@variable(model, x[M, N], Bin)  # 1 if job j is assigned to machine i
@variable(model, y[M], Bin)     # 1 if machine i is operated

# Objective: Minimize total cost
@objective(model, Min, sum(c[i, j] * x[i, j] for i in M, j in N) + sum(h[i] *␣
 ↪y[i] for i in M))

# Constraint: Each job must be assigned to exactly one machine
@constraint(model, [j in N], sum(x[i, j] for i in M) == 1)


# Constraint: Total work on each machine must not exceed its capacity
@constraint(model, [i in M], sum(a[i, j] * x[i, j] for j in N) <= b[i] * y[i])

# Solve the model
optimize!(model)

# Get the results
optimal_value = objective_value(model)
job_assignment = value.(x)
machines_used = value.(y)

println("Objective value (Total Cost): ", optimal_value)
for i in M
    if machines_used[i] == 1
        println("Machine $i is operated with jobs: ", [j for j in N if␣
 ↪job_assignment[i, j] > 0.5])
    else
        println("Machine $i is not operated")
    end
end

solve_duration = @elapsed optimize!(model)
println("Time taken to solve (seconds): ", solve_duration)
```

```
Objective value (Total Cost): 3780.9100000000003
Machine 1 is not operated
Machine 2 is not operated
Machine 3 is not operated
Machine 4 is not operated
Machine 5 is not operated
Machine 6 is operated with jobs: [4, 7, 9, 10, 11, 13, 14, 15, 17, 18, 20, 21,
```

```
22, 23, 27, 28, 29, 31, 33, 37, 45, 46, 48, 50, 51, 52, 53, 56, 58, 60]
Machine 7 is not operated
Machine 8 is not operated
Machine 9 is not operated
Machine 10 is not operated
Machine 11 is not operated
Machine 12 is not operated
Machine 13 is not operated
Machine 14 is not operated
Machine 15 is not operated
Machine 16 is not operated
Machine 17 is not operated
Machine 18 is not operated
Machine 19 is not operated
Machine 20 is not operated
Machine 21 is not operated
Machine 22 is not operated
Machine 23 is not operated
Machine 24 is not operated
Machine 25 is not operated
Machine 26 is not operated
Machine 27 is not operated
Machine 28 is operated with jobs: [1, 2, 3, 5, 6, 8, 12, 16, 19, 24, 25, 26, 30,
32, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44, 47, 49, 54, 55, 57, 59]
Machine 29 is not operated
Machine 30 is not operated
Time taken to solve (seconds): 4.846412494
```

1.2 (b)

[3]:
```
using JuMP, GLPK

# Define the model using the GLPK optimizer
model = Model(GLPK.Optimizer)

# Define decision variables
@variable(model, x[1:length(M), 1:length(N)], Bin)  # Binary variable for job␣
 ↪assignment to machines
@variable(model, y[1:length(M)], Bin)               # Binary variable to␣
 ↪indicate if a machine is operated

# Objective function: Minimize total cost
@objective(model, Min, sum(c[i,j] * x[i,j] for i in 1:length(M), j in 1:
 ↪length(N)) + sum(h[i] * y[i] for i in 1:length(M)))

# Constraint: Each job must be assigned to exactly one machine
@constraint(model, [j in 1:length(N)], sum(x[i,j] for i in 1:length(M)) == 1)
```

```julia
# Constraint: Total work on each machine must not exceed its capacity
@constraint(model, [i in 1:length(M)], sum(a[i,j] * x[i,j] for j in 1:length(N)) <= b[i] * y[i])

# LOGICAL CONDITION
# Binary variables to count active machines in each segment
@variable(model, z1, Bin)  # Indicator for at least 3 of the first 10 machines being active
@variable(model, z2, Bin)  # Indicator for at least 3 of the second 10 machines being active
@variable(model, z12, Bin) # Product of z1 and z2 (linearized)

# Ensure at least 3 of the first 10 machines are operated if z1 is active
@constraint(model, sum(y[i] for i in 1:10) >= 3 * z1)

# Ensure at least 3 of the second 10 machines are operated if z2 is active
@constraint(model, sum(y[i] for i in 11:20) >= 3 * z2)

# Linearization of z1 * z2
@constraint(model, z12 <= z1)
@constraint(model, z12 <= z2)
@constraint(model, z12 >= z1 + z2 - 1)

# If both z1 and z2 are active, operate at most 2 machines in the last 10 machines
@constraint(model, sum(y[i] for i in 21:30) <= 2 + 8 * (1 - z12))

# Optimize the model
optimize!(model)

# Retrieve the results
optimal_value_b = objective_value(model)
job_assignment_b = value.(x)
machines_used_b = value.(y)
solve_duration_b = @elapsed optimize!(model)

# Print the results
println("Objective value (Total Cost) for Part (b): ", optimal_value_b)
for i in 1:length(M)
    if machines_used_b[i] == 1
        println("Machine $i is operated with jobs: ", [j for j in 1:length(N) if job_assignment_b[i, j] > 0.5])
    else
        println("Machine $i is not operated")
    end
end
```

```
println("Time taken to solve Part (b) (seconds): ", solve_duration_b)
```

```
Objective value (Total Cost) for Part (b): 3780.9100000000003
Machine 1 is not operated
Machine 2 is not operated
Machine 3 is not operated
Machine 4 is not operated
Machine 5 is not operated
Machine 6 is operated with jobs: [4, 7, 9, 10, 11, 13, 14, 15, 17, 18, 20, 21,
22, 23, 27, 28, 29, 31, 33, 37, 45, 46, 48, 50, 51, 52, 53, 56, 58, 60]
Machine 7 is not operated
Machine 8 is not operated
Machine 9 is not operated
Machine 10 is not operated
Machine 11 is not operated
Machine 12 is not operated
Machine 13 is not operated
Machine 14 is not operated
Machine 15 is not operated
Machine 16 is not operated
Machine 17 is not operated
Machine 18 is not operated
Machine 19 is not operated
Machine 20 is not operated
Machine 21 is not operated
Machine 22 is not operated
Machine 23 is not operated
Machine 24 is not operated
Machine 25 is not operated
Machine 26 is not operated
Machine 27 is not operated
Machine 28 is operated with jobs: [1, 2, 3, 5, 6, 8, 12, 16, 19, 24, 25, 26, 30,
32, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44, 47, 49, 54, 55, 57, 59]
Machine 29 is not operated
Machine 30 is not operated
Time taken to solve Part (b) (seconds): 4.857854347
```

1.3 (c)

[10]:
```
using JuMP
using GLPK

M = 15
N = 15


# Model creation
```

```julia
model = Model(GLPK.Optimizer)

# Decision variables
@variable(model, x[1:M, 1:N], Bin)  # 1 if job j is assigned to machine i
@variable(model, y[1:M], Bin)       # 1 if machine i is used
@variable(model, makespan)          # Makespan to be minimized

# Objective: Minimize the makespan
@objective(model, Min, makespan)

# Each job must be assigned to exactly one machine
@constraint(model, [j in 1:N], sum(x[i,j] for i in 1:M) == 1)

# Total work on each machine must not exceed its capacity
@constraint(model, [i in 1:M], sum(a[i,j] * x[i,j] for j in 1:N) <= b[i])

# Makespan constraint: Time spent on any machine cannot exceed makespan
@constraint(model, [i in 1:M], makespan >= sum(d[j] * x[i,j] for j in 1:N))

# Use no more than half of the machines (15 out of 30)
@constraint(model, sum(y[i] for i in 1:M) <= 15)

# Solve the model
optimize!(model)

# Retrieve and print results
println("Optimal Makespan: ", objective_value(model))
for i in 1:M
    if value(y[i]) == 1
        println("Machine $i is used with jobs: ", [j for j in 1:N if␣
 ↪value(x[i,j]) > 0.5])
    else
        println("Machine $i is not used")
    end
end
```

```
InterruptException:

Stacktrace:
 [1] glp_intopt
   @ ~/.julia/packages/GLPK/2y5V8/src/gen/libglpk_api.jl:342 [inlined]
 [2] _solve_mip_problem(model::GLPK.Optimizer)
   @ GLPK ~/.julia/packages/GLPK/2y5V8/src/MOI_wrapper/MOI_wrapper.jl:1399
 [3] optimize!(model::GLPK.Optimizer)
   @ GLPK ~/.julia/packages/GLPK/2y5V8/src/MOI_wrapper/MOI_wrapper.jl:1457
 [4] optimize!
```

```
  @ ~/.julia/packages/MathOptInterface/jqDoD/src/Bridges/bridge_optimizer.jl:
↪367 [inlined]
[5] optimize!
  @ ~/.julia/packages/MathOptInterface/jqDoD/src/MathOptInterface.jl:122␣
↪[inlined]
[6] optimize!(m::MathOptInterface.Utilities.CachingOptimizer{MathOptInterface.
↪Bridges.LazyBridgeOptimizer{GLPK.Optimizer}, MathOptInterface.Utilities.
↪UniversalFallback{MathOptInterface.Utilities.Model{Float64}}})
  @ MathOptInterface.Utilities ~/.julia/packages/MathOptInterface/jqDoD/src/
↪Utilities/cachingoptimizer.jl:321
[7] optimize!(model::Model; ignore_optimize_hook::Bool,␣
↪_differentiation_backend::MathOptInterface.Nonlinear.SparseReverseMode, kwargs:
↪:@Kwargs{})
  @ JuMP ~/.julia/packages/JuMP/7rBNn/src/optimizer_interface.jl:595
[8] optimize!(model::Model)
  @ JuMP ~/.julia/packages/JuMP/7rBNn/src/optimizer_interface.jl:546
```

1.4 (d)

1.4.1 Using HiGHS

[4]:
```julia
using JuMP, HiGHS

# Part (a): Least Cost Assignment with HiGHS
model_highs_a = Model(HiGHS.Optimizer)

@variable(model_highs_a, x[1:length(M), 1:length(N)], Bin)
@variable(model_highs_a, y[1:length(M)], Bin)

@objective(model_highs_a, Min, sum(c[i,j] * x[i,j] for i in 1:length(M), j in 1:
↪length(N)) + sum(h[i] * y[i] for i in 1:length(M)))

@constraint(model_highs_a, [j in 1:length(N)], sum(x[i,j] for i in 1:length(M))␣
↪== 1)
@constraint(model_highs_a, [i in 1:length(M)], sum(a[i,j] * x[i,j] for j in 1:
↪length(N)) <= b[i] * y[i])

# Solve the model
optimize!(model_highs_a)

# Retrieve and print the results
optimal_value_a = objective_value(model_highs_a)
job_assignment_a = value.(x)
machines_used_a = value.(y)

println("Objective value (Total Cost) for Part (a) with HiGHS: ",␣
↪optimal_value_a)
```

```
for i in M
    if machines_used_a[i] == 1
        println("Machine $i is operated with jobs: ", [j for j in N if␣
 ↪job_assignment_a[i, j] > 0.5])
    else
        println("Machine $i is not operated")
    end
end
```

```
Running HiGHS 1.7.2 (git hash: 5ce7a2753): Copyright (c) 2024 HiGHS under MIT
licence terms
Coefficient ranges:
  Matrix [1e+00, 4e+02]
  Cost   [2e+01, 9e+02]
  Bound  [1e+00, 1e+00]
  RHS    [1e+00, 1e+00]
Presolving model
90 rows, 1830 cols, 3630 nonzeros  0s
90 rows, 1830 cols, 3630 nonzeros  0s
Objective function is integral with scale 100

Solving MIP model with:
   90 rows
   1830 cols (1830 binary, 0 integer, 0 implied int., 0 continuous)
   3630 nonzeros

        Nodes      |    B&B Tree     |            Objective Bounds
|  Dynamic Constraints |       Work
     Proc. InQueue |  Leaves   Expl. | BestBound       BestSol              Gap
|   Cuts   InLp Confl. | LpIters     Time

         0       0         0   0.00%   0               inf                  inf
0       0       0         0     0.0s
 S       0       0         0   0.00%   0               23041.32         100.00%
0       0       0         0     0.0s
 R       0       0         0   0.00%   2670.720633     23038.07          88.41%
0       0       0       190     0.0s
 C       0       0         0   0.00%   3102.121994     23037.19          86.53%
418      95       3       365     0.1s
 L       0       0         0   0.00%   3631.77821      4357.83           16.66%
1542     775       3      2746     1.5s
        10       1         1   0.20%   3631.77821      4357.83           16.66%
1529     733      19    121583    17.9s
 B     102      48        27   0.21%   3631.77821      4328.69           16.10%
1517     763     528    140000    23.0s
 L     202      86        58   0.21%   3632.131854     4319.75           15.92%
1351     816    1215    151328    26.1s
 L     302      90       104   0.21%   3632.131854     4057.49           10.48%
```

```
1370    625   1618    178665    29.3s
 L     415     55       156   0.38%   3632.131854     3871.71            6.19%
1241    492   2412    192972    32.9s
 L     416     42       157  12.67%   3632.131854     3855.33            5.79%
1344    496   2814    195676    34.1s
       918     98       373  33.96%   3632.131854     3855.33            5.79%
1468    239   9628    225587    39.1s
      1394    159       574  36.75%   3632.131854     3855.33            5.79%
1401    195   9839    256056    44.2s
      1887    217       788  36.99%   3632.131854     3855.33            5.79%
1161    211   9717    284781    49.2s
 T    2153    229       902  49.66%   3632.131854     3849.06            5.64%
1168    225   9735    303776    53.5s
 T    2155    201       903  49.83%   3632.131854     3840.74            5.43%
1169    225   9742    303788    53.5s
 T    2185    199       912  49.84%   3632.131854     3839.12            5.39%
1170    225   9904    304021    53.6s
 T    2193    176       917  49.88%   3632.131854     3829.68            5.16%
1170    225   9983    304109    53.6s
 T    2298    163       958  49.99%   3632.131854     3816.86            4.84%
1170    251   8737    307635    54.2s
 T    2310    111       966  50.15%   3632.131854     3803.02            4.49%
1173    251   8778    307711    54.2s

        Nodes      |    B&B Tree     |            Objective Bounds
|  Dynamic Constraints |       Work
     Proc. InQueue |  Leaves   Expl. | BestBound       BestSol             Gap
|   Cuts   InLp Confl. | LpIters     Time

 T    2362    106       986  50.15%   3632.131854     3797.95            4.37%
1177    251   9129    308009    54.3s
 T    2366     93       990  50.15%   3632.131854     3793.81            4.26%
1178    251   9258    308036    54.4s
 T    2380     96       994  50.15%   3632.131854     3793.41            4.25%
1087    255   9366    308149    54.4s
 T    2389     87       999  50.16%   3632.131854     3790.68            4.18%
1090    255   9416    308211    54.5s
 T    2421     74      1011  50.16%   3632.131854     3782.76            3.98%
1092    255   9661    308374    54.5s
 T    2440     64      1022  50.21%   3632.131854     3780.91            3.93%
1082     75   9937    308498    54.6s
      2894    105      1224  50.56%   3632.131854     3780.91            3.93%
1714    290   9968    340808    59.7s
      3392    124      1456  50.59%   3632.131854     3780.91            3.93%
1570    123   9308    370617    64.8s
      3794    131      1649  50.61%   3632.131854     3780.91            3.93%
2161    279   9982    400496    69.8s
```

```
Restarting search from the root node
Model after restart has 90 rows, 1830 cols (1830 bin., 0 int., 0 impl., 0
cont.), and 3630 nonzeros

      3933        0          0   0.00%   3632.131854     3780.91            3.93%
169      0       0    410854    71.6s
      3933        0          0   0.00%   3632.131854     3780.91            3.93%
169     98       4    411033    71.7s
      4242       22        137  88.80%   3636.358202     3780.91            3.82%
1567   251    2970    438367    76.7s
      4760       33        383  92.14%   3636.358202     3780.91            3.82%
1467    95    6764    470428    81.7s
      5248       20        628  92.14%   3636.358202     3780.91            3.82%
2947   265    9230    503152    86.8s
      5505       15        756  92.14%   3636.358202     3780.91            3.82%
2761   322    9758    536208    91.8s
      5867        2        940  98.44%   3730.620451     3780.91            1.33%
1790   336    9860    564053    96.9s

Solving report
  Status            Optimal
  Primal bound      3780.91
  Dual bound        3780.54
  Gap               0.00979% (tolerance: 0.01%)
  Solution status   feasible
                    3780.91 (objective)
                    0 (bound viol.)
                    1.00808250636e-13 (int. viol.)
                    0 (row viol.)
  Timing            97.00 (total)
                    0.04 (presolve)
                    0.00 (postsolve)
  Nodes             5872
  LP iterations     564947 (total)
                    228598 (strong br.)
                    45024 (separation)
                    43762 (heuristics)
Objective value (Total Cost) for Part (a) with HiGHS: 3780.9099999999994
Machine 1 is not operated
Machine 2 is not operated
Machine 3 is not operated
Machine 4 is not operated
Machine 5 is not operated
Machine 6 is operated with jobs: [4, 7, 9, 10, 11, 13, 14, 15, 17, 18, 20, 21,
22, 23, 27, 28, 29, 31, 33, 37, 45, 46, 48, 50, 51, 52, 53, 56, 58, 60]
Machine 7 is not operated
Machine 8 is not operated
Machine 9 is not operated
```

```
Machine 10 is not operated
Machine 11 is not operated
Machine 12 is not operated
Machine 13 is not operated
Machine 14 is not operated
Machine 15 is not operated
Machine 16 is not operated
Machine 17 is not operated
Machine 18 is not operated
Machine 19 is not operated
Machine 20 is not operated
Machine 21 is not operated
Machine 22 is not operated
Machine 23 is not operated
Machine 24 is not operated
Machine 25 is not operated
Machine 26 is not operated
Machine 27 is not operated
Machine 28 is operated with jobs: [1, 2, 3, 5, 6, 8, 12, 16, 19, 24, 25, 26, 30,
32, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44, 47, 49, 54, 55, 57, 59]
Machine 29 is not operated
Machine 30 is not operated
```

[5]:
```julia
# Part (b): Logical Constraints with HiGHS
model_highs_b = Model(HiGHS.Optimizer)

@variable(model_highs_b, x[1:length(M), 1:length(N)], Bin)
@variable(model_highs_b, y[1:length(M)], Bin)

@objective(model_highs_b, Min, sum(c[i,j] * x[i,j] for i in 1:length(M), j in 1:
↪length(N)) + sum(h[i] * y[i] for i in 1:length(M)))

@constraint(model_highs_b, [j in 1:length(N)], sum(x[i,j] for i in 1:length(M))␣
↪== 1)
@constraint(model_highs_b, [i in 1:length(M)], sum(a[i,j] * x[i,j] for j in 1:
↪length(N)) <= b[i] * y[i])

# Logical condition
@variable(model_highs_b, z1, Bin)
@variable(model_highs_b, z2, Bin)
@variable(model_highs_b, z12, Bin)

@constraint(model_highs_b, sum(y[i] for i in 1:10) >= 3 * z1)
@constraint(model_highs_b, sum(y[i] for i in 11:20) >= 3 * z2)
@constraint(model_highs_b, z12 <= z1)
@constraint(model_highs_b, z12 <= z2)
@constraint(model_highs_b, z12 >= z1 + z2 - 1)
```

```julia
@constraint(model_highs_b, sum(y[i] for i in 21:30) <= 2 + 8 * (1 - z12))

# Solve the model
optimize!(model_highs_b)

# Retrieve and print the results
optimal_value_b = objective_value(model_highs_b)
job_assignment_b = value.(x)
machines_used_b = value.(y)

println("Objective value (Total Cost) for Part (b) with HiGHS: ",␣
↪optimal_value_b)
for i in M
    if machines_used_b[i] == 1
        println("Machine $i is operated with jobs: ", [j for j in N if␣
↪job_assignment_b[i, j] > 0.5])
    else
        println("Machine $i is not operated")
    end
end
```

```
Running HiGHS 1.7.2 (git hash: 5ce7a2753): Copyright (c) 2024 HiGHS under MIT
licence terms
Coefficient ranges:
  Matrix [1e+00, 4e+02]
  Cost   [2e+01, 9e+02]
  Bound  [1e+00, 1e+00]
  RHS    [1e+00, 1e+01]
Presolving model
96 rows, 1833 cols, 3670 nonzeros  0s
90 rows, 1830 cols, 3630 nonzeros  0s
Objective function is integral with scale 100

Solving MIP model with:
   90 rows
   1830 cols (1830 binary, 0 integer, 0 implied int., 0 continuous)
   3630 nonzeros

        Nodes      |    B&B Tree     |            Objective Bounds
|  Dynamic Constraints |       Work
     Proc. InQueue |  Leaves   Expl. | BestBound       BestSol              Gap
|   Cuts   InLp Confl. | LpIters     Time

         0       0         0   0.00%   0               inf                  inf
0      0      0         0     0.0s
 S       0       0         0   0.00%   0               23041.32          100.00%
0      0      0         0     0.0s
```

```
 R       0       0         0   0.00%  2670.720633     23038.07           88.41%
0      0      0       190     0.0s
 C       0       0         0   0.00%  3102.121994     23037.19           86.53%
418    95      3       365     0.1s
 L       0       0         0   0.00%  3631.77821      4357.83            16.66%
1542   775     3      2746     1.8s
        10       1         1   0.20%  3631.77821      4357.83            16.66%
1529   733     19    121583    18.6s
 B     102      48        27   0.21%  3631.77821      4328.69            16.10%
1517   763    528    140000    23.8s
 L     202      86        58   0.21%  3632.131854     4319.75            15.92%
1351   816   1215    151328    27.3s
 L     302      90       104   0.21%  3632.131854     4057.49            10.48%
1370   625   1618    178665    30.4s
 L     415      55       156   0.38%  3632.131854     3871.71             6.19%
1241   492   2412    192972    34.5s
 L     416      42       157  12.67%  3632.131854     3855.33             5.79%
1344   496   2814    195676    35.7s
       897      92       363  33.95%  3632.131854     3855.33             5.79%
1449   239   9653    224666    40.7s
      1351     152       557  36.54%  3632.131854     3855.33             5.79%
1456   160   9260    251822    45.7s
      1869     217       780  36.99%  3632.131854     3855.33             5.79%
1065   229   9504    282476    50.7s
      2153     237       902  37.16%  3632.131854     3855.33             5.79%
1168   225   9726    303776    55.7s
 T    2153     229       902  49.66%  3632.131854     3849.06             5.64%
1168   225   9735    303776    55.7s
 T    2155     201       903  49.83%  3632.131854     3840.74             5.43%
1169   225   9742    303788    55.7s
 T    2185     199       912  49.84%  3632.131854     3839.12             5.39%
1170   225   9904    304021    55.8s
 T    2193     176       917  49.88%  3632.131854     3829.68             5.16%
1170   225   9983    304109    55.9s
 T    2298     163       958  49.99%  3632.131854     3816.86             4.84%
1170   251   8737    307635    56.4s

        Nodes      |    B&B Tree     |            Objective Bounds
|  Dynamic Constraints |       Work
     Proc. InQueue |  Leaves   Expl. | BestBound       BestSol              Gap
|   Cuts   InLp Confl. | LpIters     Time

 T    2310     111       966  50.15%  3632.131854     3803.02             4.49%
1173   251   8778    307711    56.4s
 T    2362     106       986  50.15%  3632.131854     3797.95             4.37%
1177   251   9129    308009    56.6s
 T    2366      93       990  50.15%  3632.131854     3793.81             4.26%
1178   251   9258    308036    56.6s
```

```
 T    2380      96        994  50.15%   3632.131854     3793.41            4.25%
1087    255   9366    308149   56.7s
 T    2389      87        999  50.16%   3632.131854     3790.68            4.18%
1090    255   9416    308211   56.7s
 T    2421      74       1011  50.16%   3632.131854     3782.76            3.98%
1092    255   9661    308374   56.8s
 T    2440      64       1022  50.21%   3632.131854     3780.91            3.93%
1082     75   9937    308498   56.9s
      2874      95       1217  50.56%   3632.131854     3780.91            3.93%
1701    285   9701    339825   61.9s
      3350     117       1437  50.59%   3632.131854     3780.91            3.93%
1884    135   9747    368974   66.9s
      3801     130       1653  50.61%   3632.131854     3780.91            3.93%
2169    225   9775    401778   72.0s

Restarting search from the root node
Model after restart has 90 rows, 1830 cols (1830 bin., 0 int., 0 impl., 0
cont.), and 3630 nonzeros

      3933       0          0   0.00%   3632.131854     3780.91            3.93%
169      0      0    410854    73.7s
      3933       0          0   0.00%   3632.131854     3780.91            3.93%
169     98      4    411033    73.7s
      4236      21        135  88.79%   3636.358202     3780.91            3.82%
1564    251   2965    438265   78.8s
      4705      40        356  92.14%   3636.358202     3780.91            3.82%
1482    176   6477    468292   83.8s
      5221      22        614  92.14%   3636.358202     3780.91            3.82%
2871    299   9008    500191   88.8s
      5440      12        725  92.14%   3636.358202     3780.91            3.82%
3032    290   9811    530070   93.8s
      5861       7        935  96.87%   3655.587939     3780.91            3.31%
1893    334   9865    559899   98.8s

Solving report
  Status            Optimal
  Primal bound      3780.91
  Dual bound        3780.54
  Gap               0.00979% (tolerance: 0.01%)
  Solution status   feasible
                    3780.91 (objective)
                    0 (bound viol.)
                    1.00808250636e-13 (int. viol.)
                    0 (row viol.)
  Timing            99.73 (total)
                    0.03 (presolve)
                    0.00 (postsolve)
  Nodes             5872
```

```
  LP iterations     564947 (total)
                    228598 (strong br.)
                    45024 (separation)
                    43762 (heuristics)
Objective value (Total Cost) for Part (b) with HiGHS: 3780.9099999999994
Machine 1 is not operated
Machine 2 is not operated
Machine 3 is not operated
Machine 4 is not operated
Machine 5 is not operated
Machine 6 is operated with jobs: [4, 7, 9, 10, 11, 13, 14, 15, 17, 18, 20, 21,
22, 23, 27, 28, 29, 31, 33, 37, 45, 46, 48, 50, 51, 52, 53, 56, 58, 60]
Machine 7 is not operated
Machine 8 is not operated
Machine 9 is not operated
Machine 10 is not operated
Machine 11 is not operated
Machine 12 is not operated
Machine 13 is not operated
Machine 14 is not operated
Machine 15 is not operated
Machine 16 is not operated
Machine 17 is not operated
Machine 18 is not operated
Machine 19 is not operated
Machine 20 is not operated
Machine 21 is not operated
Machine 22 is not operated
Machine 23 is not operated
Machine 24 is not operated
Machine 25 is not operated
Machine 26 is not operated
Machine 27 is not operated
Machine 28 is operated with jobs: [1, 2, 3, 5, 6, 8, 12, 16, 19, 24, 25, 26, 30,
32, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44, 47, 49, 54, 55, 57, 59]
Machine 29 is not operated
Machine 30 is not operated
```

[ ]:
```
# Part (c): Minimizing Makespan with HiGHS
model_highs_c = Model(HiGHS.Optimizer)

@variable(model_highs_c, x[M, N], Bin)
@variable(model_highs_c, y[M], Bin)
@variable(model_highs_c, makespan)

@objective(model_highs_c, Min, makespan)
```

```
@constraint(model_highs_c, [j in N], sum(x[i,j] for i in M) == 1)
@constraint(model_highs_c, [i in M], sum(a[i,j] * x[i,j] for j in N) <= b[i] *␣
 ↪y[i])
@constraint(model_highs_c, [i in M, j in N], x[i,j] <= y[i])

for i in M
    @constraint(model_highs_c, makespan >= sum(d[j] * x[i,j] for j in N))
end

@constraint(model_highs_c, sum(y[i] for i in M) <= 15)

# Solve the model
optimize!(model_highs_c)

# Retrieve and print the results
optimal_makespan = objective_value(model_highs_c)
job_assignment_c = value.(x)
machines_used_c = value.(y)

println("Optimal Makespan for Part (c) with HiGHS: ", optimal_makespan)
for i in M
    if machines_used_c[i] == 1
        println("Machine $i is operated with jobs: ", [j for j in N if␣
 ↪job_assignment_c[i, j] > 0.5])
    else
        println("Machine $i is not operated")
    end
end
```

2 Problem 2

[7]:
```
using JuMP
using GLPK

# Number of batches
n = 10

# Time matrix representing cleaning times between batches
time_matrix = [
    0 11 7 13 11 12 4 9 7 11;
    5 0 13 15 15 6 8 10 9 8;
    13 15 0 23 11 11 16 18 5 7;
    9 13 5 0 8 10 12 14 5 3;
    3 7 7 7 0 9 10 11 12 13;
    10 6 3 4 14 0 8 5 11 12;
    4 6 7 3 13 7 0 10 4 6;
```

```
    7 8 9 9 12 11 10 0 10 9;
    9 9 14 8 4 9 6 10 0 12;
    11 17 11 6 10 4 7 9 11 0;
]

# Mixing times for each batch
mixing_times = [40, 35, 45, 32, 50, 42, 44, 30, 33, 55]

# Create the model
model = Model(GLPK.Optimizer)

# Decision Variables
@variable(model, x[1:n, 1:n], Bin)    # x[i,j] = 1 if we go from batch i to␣
 ↪batch j, 0 otherwise
@variable(model, u[2:n] >= 0)         # Subtour elimination variables (MTZ␣
 ↪formulation)

# Objective: Minimize the total time (mixing + cleaning)
@objective(model, Min,
    sum(time_matrix[i,j] * x[i,j] for i in 1:n, j in 1:n) +  # Cleaning time
    sum(mixing_times[i] * sum(x[i,j] for j in 1:n) for i in 1:n)  # Mixing time
)

# Constraints: Ensure each batch is entered and left exactly once
@constraint(model, [i=1:n], sum(x[i,j] for j in 1:n if i != j) == 1)
@constraint(model, [j=1:n], sum(x[i,j] for i in 1:n if i != j) == 1)

# Subtour elimination constraints (MTZ formulation)
@constraint(model, [i=2:n, j=2:n], u[i] - u[j] + n*x[i,j] <= n-1)

# Solve the model
optimize!(model)

# Extract the optimal order of paint batches
optimal_order = []
current_batch = 1
push!(optimal_order, current_batch)

while length(optimal_order) < n
    for j in 1:n
        if value(x[current_batch, j]) > 0.5 && !(j in optimal_order)
            push!(optimal_order, j)
            current_batch = j
            break
        end
    end
end
```

```julia
# Print the optimal order and total time
println("Optimal Order of Paint Batches: ", optimal_order)
println("Total Time (Mixing + Cleaning): ", objective_value(model))
```

```
Optimal Order of Paint Batches: Any[1, 7, 4, 10, 8, 2, 6, 3, 9, 5]
Total Time (Mixing + Cleaning): 454.0
```

[ ]:

```julia

```