Mderangu\_Integer

2024-11-17

##Integer Programming (IP) is a type of mathematical optimization technique used to solve decision-making problems where some or all of the decision variables are restricted to be integers. It is widely used in industries such as logistics, scheduling, finance, and manufacturing, where solutions must be whole numbers.  
  
# Load the lpSolveAPI package  
library(lpSolveAPI)

## Warning: package 'lpSolveAPI' was built under R version 4.2.3

# Initialize a linear programming model with 0 constraints and 7 decision variables  
model <- make.lp(0, 7)  
  
# Define the objective function (weekly wages for workers in each shift)  
set.objfn(model, c(775, 800, 800, 800, 800, 775, 750))  
  
# Specify that all decision variables are integers  
set.type(model, 1:7, "integer")  
  
lp.control(model, sense = 'min')

## $anti.degen  
## [1] "fixedvars" "stalling"   
##   
## $basis.crash  
## [1] "none"  
##   
## $bb.depthlimit  
## [1] -50  
##   
## $bb.floorfirst  
## [1] "automatic"  
##   
## $bb.rule  
## [1] "pseudononint" "greedy" "dynamic" "rcostfixing"   
##   
## $break.at.first  
## [1] FALSE  
##   
## $break.at.value  
## [1] -1e+30  
##   
## $epsilon  
## epsb epsd epsel epsint epsperturb epspivot   
## 1e-10 1e-09 1e-12 1e-07 1e-05 2e-07   
##   
## $improve  
## [1] "dualfeas" "thetagap"  
##   
## $infinite  
## [1] 1e+30  
##   
## $maxpivot  
## [1] 250  
##   
## $mip.gap  
## absolute relative   
## 1e-11 1e-11   
##   
## $negrange  
## [1] -1e+06  
##   
## $obj.in.basis  
## [1] TRUE  
##   
## $pivoting  
## [1] "devex" "adaptive"  
##   
## $presolve  
## [1] "none"  
##   
## $scalelimit  
## [1] 5  
##   
## $scaling  
## [1] "geometric" "equilibrate" "integers"   
##   
## $sense  
## [1] "minimize"  
##   
## $simplextype  
## [1] "dual" "primal"  
##   
## $timeout  
## [1] 0  
##   
## $verbose  
## [1] "neutral"

# Add constraints to ensure sufficient workers for each day  
# Sunday  
add.constraint(model, c(0, 1, 1, 1, 1, 1, 0), ">=", 18)  
# Monday  
add.constraint(model, c(0, 0, 1, 1, 1, 1, 1), ">=", 27)  
# Tuesday  
add.constraint(model, c(1, 0, 0, 1, 1, 1, 1), ">=", 22)  
# Wednesday  
add.constraint(model, c(1, 1, 0, 0, 1, 1, 1), ">=", 26)  
# Thursday  
add.constraint(model, c(1, 1, 1, 0, 0, 1, 1), ">=", 25)  
# Friday  
add.constraint(model, c(1, 1, 1, 1, 0, 0, 1), ">=", 21)  
# Saturday  
add.constraint(model, c(1, 1, 1, 1, 1, 0, 0), ">=", 19)  
  
# Matrix to represent daily worker requirements  
daily\_worker\_demand <- matrix(c(  
 "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",  
 18, 27, 22, 26, 25, 21, 19  
), ncol = 2, byrow = FALSE)  
  
# Assign column names for better readability  
colnames(daily\_worker\_demand) <- c("Day", "Required\_Workers")  
  
# Convert to a tabular format for easy viewing  
as.table(daily\_worker\_demand)

## Day Required\_Workers  
## A Sunday 18   
## B Monday 27   
## C Tuesday 22   
## D Wednesday 26   
## E Thursday 25   
## F Friday 21   
## G Saturday 19

# Matrix to represent shift details, days off, and weekly wages  
shift\_schedule\_and\_wages <- matrix(c(  
 1, 2, 3, 4, 5, 6, 7,  
 "Sun & Mon Off", "Mon & Tue Off", "Tue & Wed Off", "Wed & Thu Off",   
 "Thu & Fri Off", "Fri & Sat Off", "Sat & Sun Off",  
 "$775", "$800", "$800", "$800", "$800", "$775", "$750"  
), ncol = 3, byrow = FALSE)  
  
# Assign column names for clarity  
colnames(shift\_schedule\_and\_wages) <- c("Shift\_ID", "Days\_Off", "Weekly\_Wage")  
  
# Convert to a tabular format for easy viewing  
as.table(shift\_schedule\_and\_wages)

## Shift\_ID Days\_Off Weekly\_Wage  
## A 1 Sun & Mon Off $775   
## B 2 Mon & Tue Off $800   
## C 3 Tue & Wed Off $800   
## D 4 Wed & Thu Off $800   
## E 5 Thu & Fri Off $800   
## F 6 Fri & Sat Off $775   
## G 7 Sat & Sun Off $750

# Solve the optimization problem  
solve(model)

## [1] 0

# Get the total minimized cost (objective value)  
total\_cost <- get.objective(model)  
cat("Total Cost: $", total\_cost, "\n")

## Total Cost: $ 25675

# Get the optimal number of workers assigned to each shift  
workers\_per\_shift <- get.variables(model)  
cat("Number of workers assigned to each shift:\n")

## Number of workers assigned to each shift:

print(workers\_per\_shift)

## [1] 2 4 5 0 8 1 13

# Get the number of workers available on each day based on the solution  
daily\_worker\_coverage <- get.constraints(model)  
cat("Number of workers available each day:\n")

## Number of workers available each day:

print(daily\_worker\_coverage)

## [1] 18 27 24 28 25 24 19

##Conclusion:  
  
##Minimized Cost: The total weekly cost is minimized at $25,675, considering shift-specific wages and additional weekend pay.  
  
##Worker Allocation: The optimal number of workers assigned to each shift ensures that no constraints are violated.  
  
##Daily Labor Needs: The number of workers available each day satisfies or exceeds the required demand:  
  
##No shortages occur, as every day's coverage meets or exceeds the specified minimum.