# Operation Research Journal

## M.Sc Part II Computer Science

Mithun Parab 509

October 11, 2023

R.J. College of Arts, Science & Commerce
Operation Research
Seat number: 509

# RAMNIRANJAN JHUNJHUNWALA COLLEGE

**Ghatkopar (W), Mumbai-400 086**

## CERTIFICATE

M.Sc Part II

Computer Science

2023-2024

This is to certify that **Mithun Sahdev Parab** of M.Sc Prat II (Sem-III) Computer Science, Seat No **509** of satisfactorily completed the practicals of **OPERATION RESEARCH (PAPER III)** during the academic year **2023 - 2024** as specified by the **MUMBAI UNIVERSITY**.

No. of Experiments completed **10** out of **10**

**Sign of Incharge:**
**Date: October 11, 2023** **Seat Number: 509**

**Sign of Examiner:**
**Date: October 11, 2023** **Course Co-ordinator**

# Contents



Link for GitHub

# 1 PRACTICAL 01: GRAPHICAL METHOD USING R PROGRAMMING

## 1.1 Find a geometrical interpretation and solution as well for the following LP problem

$Max\ z = 3x1 + 5x2$

subject to constraints:

$x1 + 2x2 <= 2000$

$x1 + x2 <= 1500$

$x2 <= 600$

$x1, x2 >= 0$

[ ]:
```
install.packages("lpSolve")
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```

[ ]:
```
# R Program
#Find a geometrical interpretation and solution as well for the following LP␣
↪problem
#Max z= 3x1 + 5x2
#subject to constraints:
#x1+2x2<=2000
#x1+x2<=1500
#x2<=600
#x1,x2>=0
# Load lpSolve
require(lpSolve)
```

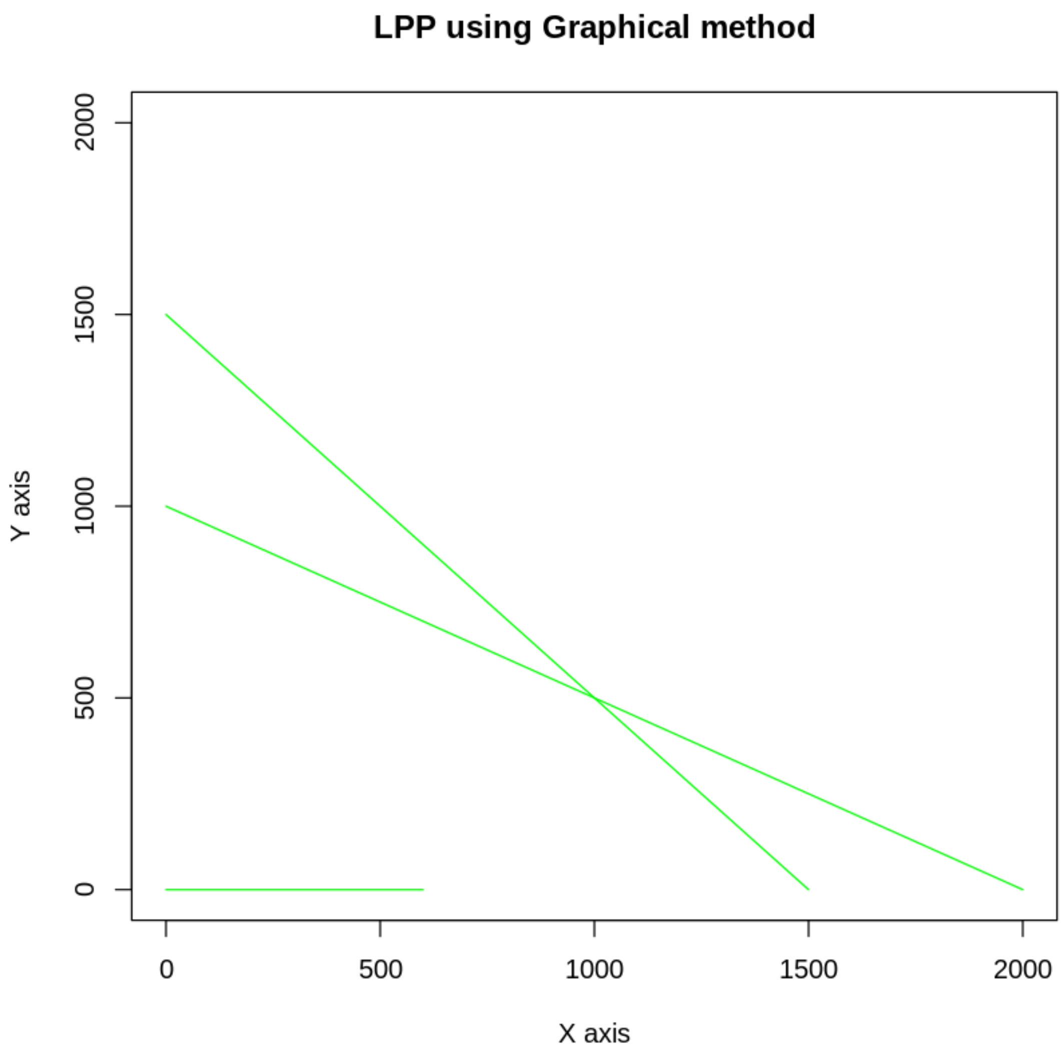
```
Loading required package: lpSolve
```

[ ]:
```
## Set the coefficients of the decision variables -> C of objective function
C <- c(3,5)
# Create constraint martix B
A <- matrix(c(1, 2,
              1, 1,
              0, 1
              ), nrow=3, byrow=TRUE)
# Right hand side for the constraints
B <- c(2000,1500,600)
```

[ ]:
```
# Direction of the constraints
sconstranints_direction <- c("<=", "<=", "<=")
```

[ ]:

```r
# Create empty example plot
plot.new()
plot.window(xlim=c(0,2000), ylim=c(0,2000))
axis(1)
axis(2)
title(main="LPP using Graphical method")
title(xlab="X axis")
title(ylab="Y axis")
box()
# Draw one line
segments(x0 = 2000, y0 = 0, x1 = 0, y1 = 1000, col = "green")
segments(x0 = 1500, y0 = 0, x1 = 0, y1 = 1500, col = "green")
segments(x0 = 0, y0 = 0, x1 = 600, y1 = 0, col = "green")
```



[ ]:
```
# Find the optimal solution
optimum <- lp(direction="max",
objective.in = C,
const.mat = A,
const.dir = sconstranints_direction,
const.rhs = B,
all.int = T)
# Print status: 0 = success, 2 = no feasible solution
print(optimum$status)
```

```
[1] 0
```

[ ]:
```
# Display the optimum values for x1,x2
best_sol <- optimum$solution
names(best_sol) <- c("x1", "x2")
print(best_sol)
# Check the value of objective function at optimal point
print(paste("Total cost: ", optimum$objval, sep=""))
```

```
  x1   x2
1000  500
[1] "Total cost: 5500"
```

[ ]:

## 2 PRACTICAL 02: SIMPLEX METHOD WITH 2 VARIABLES USING PYTHON

$ Max  z=3x1+2x2$

subject to

$x1 + x2 <= 4$

$x1 - x2 <= 2$

$x1, x2 >= 0$

[ ]:
```
from scipy.optimize import linprog
#Max z=3x1+2x2
#subject to
#x1 + x2 <=4
#x1 - x2 <=2
#x1,x2>=0
obj = [-3, -2]
lhs_ineq = [[ 1, 1], # Red constraint left side
            [1, -1]] # Blue constraint left side
rhs_ineq = [4, 2] # Blue constraint right side
bnd = [(0, float("inf")), # Bounds of x
       (0, float("inf"))] # Bounds of y
```

```
opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq,
              bounds=bnd,method="revised simplex")
```

```
<ipython-input-1-70a4ba51b253>:13: DeprecationWarning: `method='revised
simplex'` is deprecated and will be removed in SciPy 1.11.0. Please use one of
the HiGHS solvers (e.g. `method='highs'`) in new code.
  opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq,
```

[ ]:
```
opt
```

[ ]:
```
 message: Optimization terminated successfully.
 success: True
  status: 0
     fun: -11.0
       x: [ 3.000e+00  1.000e+00]
     nit: 2
```

[ ]:
```
opt.fun
```

[ ]:
```
-11.0
```

[ ]:
```
opt.success
```

[ ]:
```
True
```

[ ]:
```
opt.x
```

[ ]:
```
array([3., 1.])
```

# 3 PRACTICAL 03: SIMPLEX METHOD WITH 3 VARIABLES USING PYTHON

$Min\ z = x1 - 3x2 + 2x3$

subject to

$3x1 - x2 + 3x3 <= 7$

$-2x1 + 4x2 <= 12$

$-4x1 + 3x2 + 8x3 <= 10$

$x1, x2, x3 >= 0$

[ ]:
```
from scipy.optimize import linprog
#Min z= x1-3x2+2x3
#subject to
#3x1-x2+3x3<=7
#-2x1+4x2<=12
#-4x1+3x2+8x3<=10
```

```
#x1,x2,x3>=0
obj = [1, -3, 2]
lhs_ineq = [[ 3, -1, 3], # Red constraint left side
            [-2, 4, 0], # Blue constraint left side
            [ -4, 3, 8]] # Yellow constraint left side
```

[ ]:
```
rhs_ineq = [7, # Red constraint right side
            12, # Blue constraint right side
            10] # Yellow constraint right side
bnd = [(0, float("inf")), # Bounds of x
       (0, float("inf")),
       (0, float("inf"))] # Bounds of y
```

[ ]:
```
opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq,
              bounds=bnd, method="revised simplex")
```

```
<ipython-input-3-188681a0ccf8>:1: DeprecationWarning: `method='revised simplex'`
is deprecated and will be removed in SciPy 1.11.0. Please use one of the HiGHS
solvers (e.g. `method='highs'`) in new code.
  opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq,
```

[ ]:
```
opt
```

[ ]:
```
 message: Optimization terminated successfully.
 success: True
  status: 0
     fun: -11.0
       x: [ 4.000e+00  5.000e+00  0.000e+00]
     nit: 2
```

# 4 Practical 04: SIMPLEX METHOD WITH EQUALITY CONSTRAINTS USING PYTHON

$Max\ z = x + 2y$

subject to

$2x + y <= 20$

$ -4x+5y<=10$

$ -x+2y>=-2$

$ -x+5y=15$

$ x,y>=0$

[ ]:
```
from scipy.optimize import linprog
#Max z=x+2y
#subject to
```

```
#2x+y<=20
#-4x+5y<=10
#-x+2y>=-2
#-x+5y=15
#x,y>=0

obj = [-1, -2]
lhs_ineq = [[ 2, 1], # Red constraint left side
            [-4, 5], # Blue constraint left side
            [ 1, -2]] # Yellow constraint left side
rhs_ineq = [20, # Red constraint right side
            10, # Blue constraint right side
            2] # Yellow constraint right side
```

[ ]:
```
lhs_eq = [[-1, 5]] # Green constraint left side
rhs_eq = [15] # Green constraint right side

bnd = [(0, float("inf")), # Bounds of x
       (0, float("inf"))] # Bounds of y
```

[ ]:
```
opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq,
        A_eq=lhs_eq, b_eq=rhs_eq, bounds=bnd, method="revised simplex")
```

```
<ipython-input-2-44a20d41c7cb>:1: DeprecationWarning: `method='revised simplex'`
is deprecated and will be removed in SciPy 1.11.0. Please use one of the HiGHS
solvers (e.g. `method='highs'`) in new code.
  opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq,
```

[ ]:
```
opt
```

[ ]:
```
 message: Optimization terminated successfully.
 success: True
  status: 0
     fun: -16.818181818181817
       x: [ 7.727e+00  4.545e+00]
     nit: 3
```

# 5 PRACTICAL 05:SOLVE FOLLOWING LINEAR PROGRAMMING PROBLEM USING BIG M SIMPLEX METHOD.

Min $z = 4x1 + x2$

subjected to:

$3x1 + 4x2 >= 20$

$x1 + 5x2 >= 15$

$x1, x2 >= 0$

[1]:
```python
from scipy.optimize import linprog

obj = [4, 1]
```

[2]:
```python
lhs_ineq = [[ -3, -4], # left side of first constraint
            [-1, -5]] # right side of first constraint
rhs_ineq = [-20, # right side of first constraint
            -15] # right side of Second constraint
```

[3]:
```python
bnd = [(0, float("inf")), # Bounds of x1
       (0, float("inf"))] # Bounds of x2
```

[4]:
```python
opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq,
              bounds=bnd,method="interior-point")
```

```
<ipython-input-4-42faacf79545>:1: DeprecationWarning: `method='interior-point'`
is deprecated and will be removed in SciPy 1.11.0. Please use one of the HiGHS
solvers (e.g. `method='highs'`) in new code.
  opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq,
```

[5]:
```python
opt
```

[5]:
```
 message: Optimization terminated successfully.
 success: True
  status: 0
     fun: 5.000000000236444
       x: [ 6.012e-11  5.000e+00]
     nit: 5
```

# 6 PRACTICAL 06: RESOURCE ALLOCATION PROBLEM BY SIMPLEX METHOD

Use SciPy to solve the resource allocation problem stated as follows:

$Maxz = 20x1 + 12x2 + 40x3 + 25x4$ .............(profit)

subjected to:

$x1 + x2 + x3 + x4 <= 50$ ———–-(manpower)

$3x1 + 2x2 + x3 <= 100$ ———–-(material A)

$x2 + 2x3 <= 90$ ———–-(material B)

$x1, x2, x3, x4 >= 0$

[1]:
```python
from scipy.optimize import linprog
obj = [-20, -12, -40, -25] #profit objective function
```

[2]:
```
lhs_ineq = [[1, 1, 1, 1], # Manpower
            [3, 2, 1, 0], # Material A
             [0, 1, 2, 3]] # Material B
rhs_ineq = [ 50, # Manpower
             100, # Material A
             90] # Material B
```

[3]:
```
opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq,
              method="revised simplex")
```

```
<ipython-input-3-7085e87a9e94>:1: DeprecationWarning: `method='revised simplex'`
is deprecated and will be removed in SciPy 1.11.0. Please use one of the HiGHS
solvers (e.g. `method='highs'`) in new code.
  opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq,
```

[4]:
```
opt
```

[4]:
```
 message: Optimization terminated successfully.
 success: True
  status: 0
     fun: -1900.0
       x: [ 5.000e+00  0.000e+00  4.500e+01  0.000e+00]
     nit: 2
```

# 7 PRACTICAL 7: INFEASIBILITY IN SIMPLEX METHOD

## 7.1 Solve following linear programming problem using simplex method

### 7.1.1 While solving linear programming problem using simplex method, if one or more artificial variables remain in the basis at positive level at the end of phase 1 computation , the problem has no feasible solution( infeasible solution).

Example:

$ Max  z= 200x - 300y $

subject to

$2x + 3y >= 1200$

$x + y <= 400$

$2x + 3/2y >= 900$

$x, y >= 0$

[1]:
```
from scipy.optimize import linprog
obj = [-200, 300]
```

[2]:
```
lhs_ineq = [[ -2, -3], # Red constraint left side
            [1, 1], # Blue constraint left side
```

```
           [ -2, -1.5]] # Yellow constraint left side
rhs_ineq = [-1200, # Red constraint right side
              400, # Blue constraint right side
             -900] # Yellow constraint right side
bnd = [(0, float("inf")), # Bounds of x
       (0, float("inf"))] # Bounds of y
```

[3]:
```
opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq,
              method="revised simplex")
```

```
<ipython-input-3-7085e87a9e94>:1: DeprecationWarning: `method='revised simplex'`
is deprecated and will be removed in SciPy 1.11.0. Please use one of the HiGHS
solvers (e.g. `method='highs'`) in new code.
  opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq,
```

[4]:
```
opt
```

[4]:
```
 message: The problem appears infeasible, as the phase one auxiliary problem
terminated successfully with a residual of 3.0e+02, greater than the tolerance
1e-12 required for the solution to be considered feasible. Consider increasing
the tolerance to be greater than 3.0e+02. If this tolerance is unnaceptably
large, the problem is likely infeasible.
 success: False
  status: 2
     fun: 120000.0
       x: [ 0.000e+00  4.000e+02]
     nit: 1
```

# 8 PRACTICAL 8: DUAL SIMPLEX METHOD

## 8.1 Solve following linear programming problem using dual simplex method using r programming

$Max\ z = 40x1 + 50x2$

subject to

$2x1 + 3x2 <= 3$

$8x1 + 4x2 <= 5$

$x1, x2 >= 0$

[2]:
```
install.packages("lpSolve")
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```

[3]:
```
# Import lpSolve package
library(lpSolve)
# Set coefficients of the objective function
f.obj <- c(40, 50)
```

[4]:
```
# Set matrix corresponding to coefficients of constraints by rows
# Do not consider the non-negative constraint; it is automatically assumed
f.con <- matrix(c(2, 3, 8, 4), nrow = 2, byrow = TRUE)
```

[5]:
```
# Set unequality signs
f.dir <- c("<=", "<=")
# Set right hand side coefficients
f.rhs <- c(3, 5)
```

[7]:
```
# Final value (z)
lp("max", f.obj, f.con, f.dir, f.rhs)
# Variables final values
lp("max", f.obj, f.con, f.dir, f.rhs)$solution
# Sensitivities
lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$sens.coef.from
lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$sens.coef.to
```

```
Success: the objective function is 51.25
```

1. 0.1875 2. 0.875

1. 33.3333333333333 2. 20

1. 100 2. 60

[8]:
```
# Dual Values (first dual of the constraints and then dual of the variables)
# Duals of the constraints and variables are mixed
lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$duals
# Duals lower and upper limits
lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$duals.from
lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$duals.to
```

1. 15 2. 1.25 3. 0 4. 0

1. 1.25 2. 4 3. -1e+30 4. -1e+30

1. 3.75 2. 12 3. 1e+30 4. 1e+30

# 9 PRACTICAL 9: TRANSPORTATION PROBLEM

## 9.1 Solve following transportation problem in which cell entries represent unit costs using R programming.

```
"Customer 1", "Customer 2", "Customer 3", "Customer 4" SUPPLY

sUPPLIER 1 10 2 20 11 15
```

sUPPLIER 1 12 7 9 20 25

sUPPLIER 1 4 14 16 18 10

DEMAND 5 15 15 15

[1]:
```
install.packages("lpSolve")
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```

[2]:
```
# Import lpSolve package
library(lpSolve)
# Set transportation costs matrix
costs <- matrix(c(10, 2, 20, 11,
                  12, 7, 9, 20,
                  4, 14 , 16, 18), nrow = 3, byrow = TRUE)

# Set customers and suppliers' names
colnames(costs) <- c("Customer 1", "Customer 2", "Customer 3", "Customer 4")
rownames(costs) <- c("Supplier 1", "Supplier 2", "Supplier 3")
```

[3]:
```
# Set unequality/equality signs for suppliers
row.signs <- rep("<=", 3)
# Set right hand side coefficients for suppliers
row.rhs <- c(15, 25, 10)
# Set unequality/equality signs for customers
col.signs <- rep(">=", 4)
# Set right hand side coefficients for customers
col.rhs <- c(5, 15, 15, 15)
```

[4]:
```
# Final value (z)
TotalCost <- lp.transport(costs, "min", row.signs, row.rhs, col.signs, col.rhs)
# Variables final values
lp.transport(costs, "min", row.signs, row.rhs, col.signs, col.rhs)$solution
```

A matrix: $3 \times 4$ of type dbl

| | | | |
|---|---|---|---|
| 0 | 5 | 0 | 10 |
| 0 | 10 | 15 | 0 |
| 5 | 0 | 0 | 5 |

```
Success: the objective function is 435
```

[5]:
```
print(TotalCost)
```

```
Success: the objective function is 435
```

# 10 Practical 10: ASSIGNMENT PROBLEM

## 10.1 Solve following assignment problem represented in following matrix using r programming

```
   JOB1 JOB2 JOB3
W1  15   10    9
W2  9    15   10
W3  10   12    8
```

[1]:
```
install.packages("lpSolve")
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```

[2]:
```
# Import lpSolve package
library(lpSolve)
# Set assignment costs matrix
costs <- matrix(c(15, 10, 9,
9, 15, 10,
10, 12 ,8), nrow = 3, byrow = TRUE)
```

[3]:
```
# Print assignment costs matrix
costs
```

A matrix: 3 × 3 of type dbl

| | | |
|---|---|---|
| 15 | 10 | 9 |
| 9 | 15 | 10 |
| 10 | 12 | 8 |

[4]:
```
# Final value (z)
lp.assign(costs)
```

```
Success: the objective function is 27
```

[5]:
```
# Variables final values
lp.assign(costs)$solution
```

A matrix: 3 × 3 of type dbl

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 1 |