R FOR INDUSTRIAL ENGINEERS

Operations Research with R — Graphical Method

Exploring the "lpSolve" R package

Roberto Salazar
Nov 25, 2020 · 6 min read ★

Image by Kelly Sikkema available at Unsplash

Operations Research

Operations Research is a scientific approach for decision making that seeks for the best design and operation of a system, usually under conditions requiring the allocation of scarce resources. The scientific approach for decision making requires the use of one or more mathematical/optimization models (i.e. representations of the actual situation) to make the optimum decision.

An optimization model seeks to find the values of the ***decision variables*** that optimize (maximize or minimize) an ***objective function*** among the set of all values for the decision variables that satisfy the given ***constraints***. Its three main components are:

- Objective function: a function to be optimized (maximized or minimized)
- Decision variables: controllable variables that influence the performance of the system
- Constraints: set of restrictions (i.e. linear inequalities or equalities) of decision variables. A non-negativity constraint limits the decision variables to take positive values (e.g. you cannot produce negative number of items $x1$, $x2$ and $x3$).

The solution of the optimization model is called the **optimal feasible solution**.

Modeling Steps

Modeling accurately an operations research problem represents the most significant-and sometimes the most difficult-task. A wrong model will lead to a wrong solution, and thus, will not solve the original problem. The following steps should be performed by different team members with different areas of expertise to obtain an accurate and greater view of the model:

1. **Problem definition**: defining the scope of the project and identifying that the result is the identification of three elements: description of decision variables, determination of the objective and determination of the limitations (i.e. constraints).
2. **Model construction**: translating the problem definition into mathematical relationships.
3. **Model solution**: using a standard optimization algorithm. Upon obtaining a solution, a sensitivity analysis should be performed to find out the behavior of the solution due to changes in some of the parameters.
4. **Model validity**: checking if the model works as it was supposed to.
5. **Implementation**: translating the model and the results into the recommendation of a solution.

Linear Programming

Linear programming (also referred as LP) is an operations research technique used when all the objectives and constraints are linear (in the variables) and when all the decision variables are **continuous**. In hierarchy, linear programming could be considered as the easiest operations research technique.

Graphical Method

The graphical method represents an optimization algorithm for solving linear programming problems containing two decision variables ($x1$ and $x2$). It is one of the most popular approaches for solving simple linear programming problems.

. . .

For the following example, let's consider the following simple mathematical model to be solved:

$$\max z = 4x_1 + 5x_2$$

s.t.

$$\begin{aligned} 6x_1 + 6x_2 &\leq 36 \\ x_1 + 2x_2 &\leq 10 \\ x_1 &\leq 4 \\ x_1, \quad x_2 &\geq 0 \end{aligned}$$

Let's take a look at the R code!

As a first step, in order to plot the lines corresponding to each constraint, we must transform them into equalities:

$$\begin{aligned} x_2 &= 6 - x_1 \\ x_2 &= 5 - 0.5x_1 \\ x_1 &= 4 \end{aligned}$$

$$x_1 = 0$$
$$x_2 = 0$$

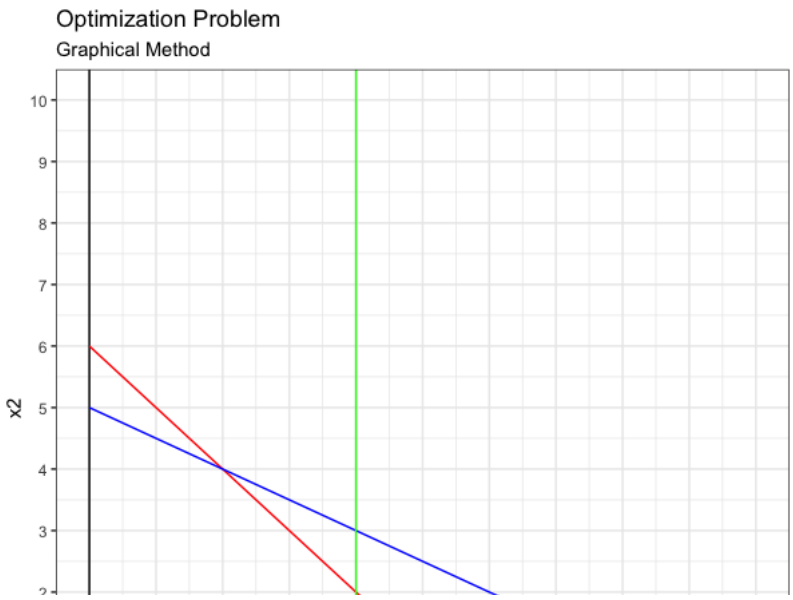
```r
# Define constraints
cons.1 <- function(x) 6 - x
cons.2 <- function(x) 5 - 0.5*x
# cons.3 x1 = 4 (plotted using geom_vline)
# cons.4 x1 = 0 (defined by setting x1 axis limits)
# cons.5 x2 = 0 (defined by setting x2 axis limits)
```

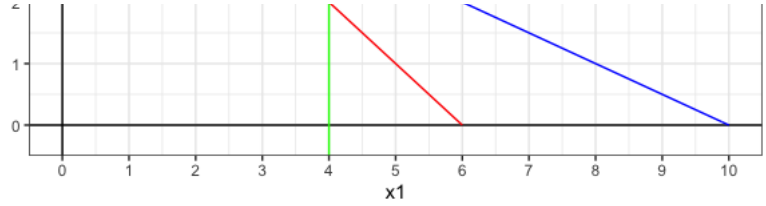
Graphical Method - Constraints.R hosted with ♡ by GitHub view raw

Next, the transformed constraints must be plotted in a two-dimensional space plot:

```r
# Import ggplot2 package
library(ggplot2)

# Build plot
p <- ggplot(data = data.frame(x = 0), aes(x = x)) +

  # Add axes
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = 0) +

  # Add constraints lines
  stat_function(colour = "Red", fun = cons.1) +
  stat_function(colour = "Blue", fun = cons.2) +
  geom_vline(xintercept = 4, colour = "Green") +

  # Specify axes breaks and limits
  scale_x_continuous(breaks = seq(0, 10, 1), lim = c(0, 10)) +
  scale_y_continuous(breaks = seq(0, 10, 1), lim = c(0, 10)) +

  # Define labels
  labs(title = "Optimization Problem",
       subtitle = "Graphical Method",
       x = "x1",
       y = "x2") +

  # Add black and white theme
  theme_bw()

# Print plot
print(p)
```

Graphical Method - Plot.R hosted with ♡ by GitHub view raw

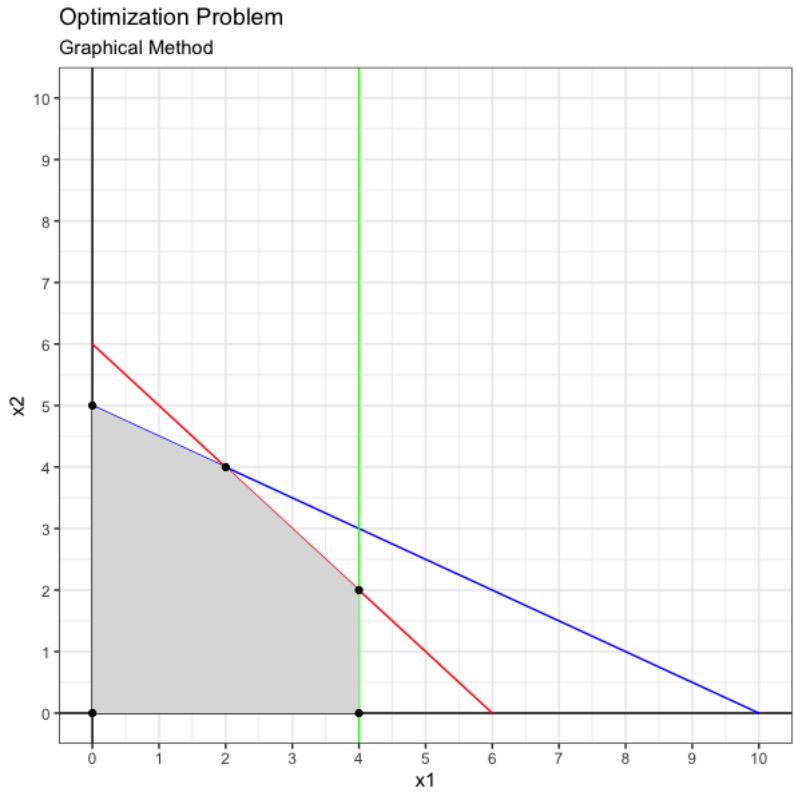




Now that the constraints equations have been plotted, the next step consists in defining the feasible region, which is the polygon (i.e. area plot) where all constrains original inequalities are met:

```r
# Define feasible region polygon
feasible_region = data.frame(x = c(0, 4, 4, 2, 0), y = c(0, 0, 2, 4, 5))

# Add feasible region to current plot
p <- p + geom_polygon(data = feasible_region, mapping = aes(x = x, y = y), fill = "#dddddd") +
  geom_point(data = feasible_region, aes(x = x, y = y), color = "Black")

# Print plot
print(p)
```

Graphical Method - Feasible Region.R hosted with ♡ by GitHub view raw



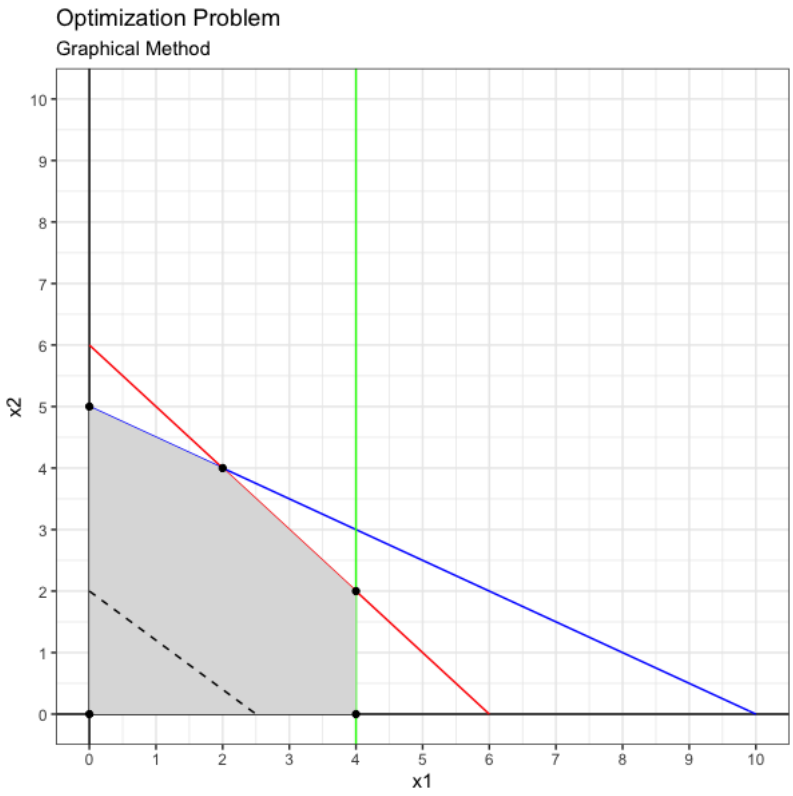
The feasible region above contains all the possible combinations of $x1$ and $x2$ that satisfy the given constraints. However, the optimum solution of the problem will be located in one of its corner points, either (0,0), (4,0), (4,2), (2,4) or (0,5).

To identify the corner point that represents the optimum solution, the objective function needs be plotted next. Once plotted, the objective function line will be gradually displaced to the right all the way before exiting the feasible region. The last point the objective function line touches before exiting the feasible region represents the optimum solution of the problem (i.e. the combination of $x1$ and $x2$ values that maximize the value of z).

```r
# Define objective function
z1 <- function(x) -0.8*x + 2 # (the + 2 constant is just for displacement visualization purposes)

```

```r
# Visualize objective function displacement
p <- p + stat_function(colour = "Black", fun = z1, lty=2)

# Print plot
print(p)
```

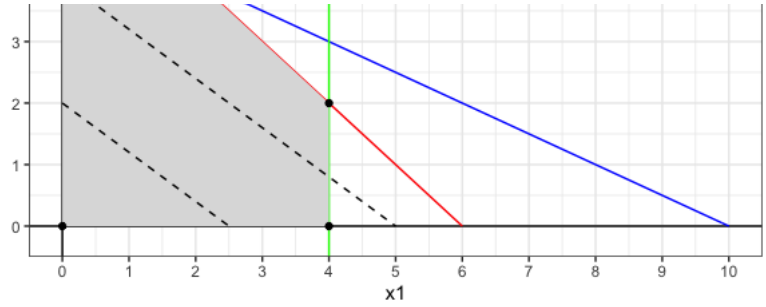
Graphical Method - Objective Function #1.R hosted with ♡ by GitHub view raw



Based on the plot above, the objective function line can be displaced even more to the right.

```r
# Redefine objective function
z2 <- function(x) -0.8*x + 4 # (the + 4 constant is just for displacement visualization purposes)

# Visualize objective function displacement again
p <- p + stat_function(colour = "Black", fun = z2, lty=2)

# Print plot
print(p)
```

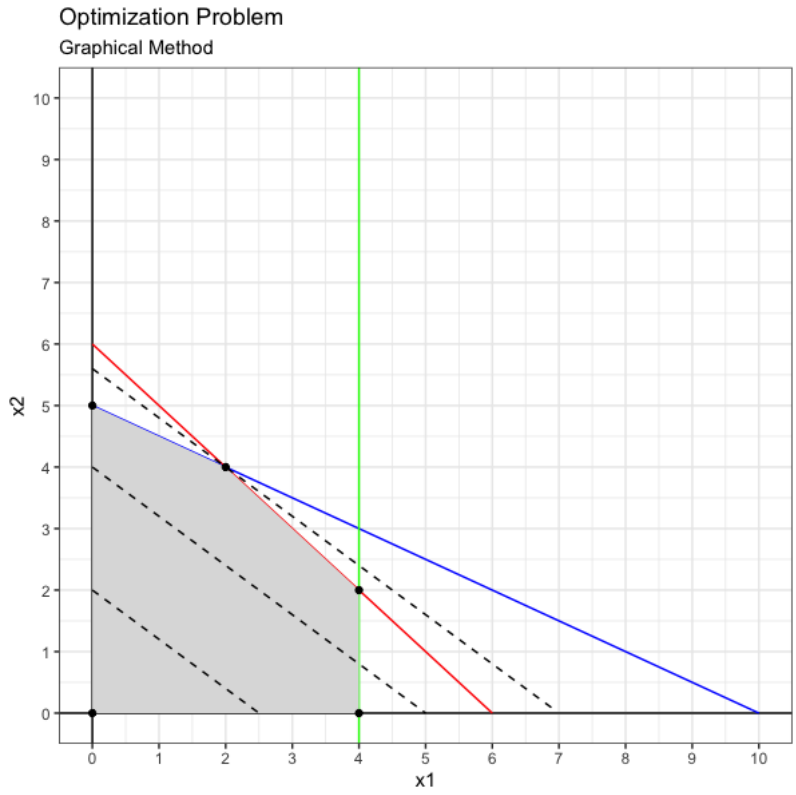
Graphical Method - Objective Function #2.R hosted with ♡ by GitHub view raw



One again, based on the plot above, the objective function line can be displaced even more to the right.

```r
# Visualize objective function displacement again
z.solution <- function(x) -0.8*x + 5.6   # (the + 5.6 constant is just for displacement visualiza

# Visualize objective function displacement - solution
p <- p + stat_function(colour = "Black", fun = z.solution, lty=2)

# Print plot
print(p)
```

Graphical Method - Objective Function #3.R hosted with ♡ by GitHub view raw



At this point, the objective function line cannot be further displaced to the right because it would exit the feasible region, thus, the optimum solution has been reached at (2,4).

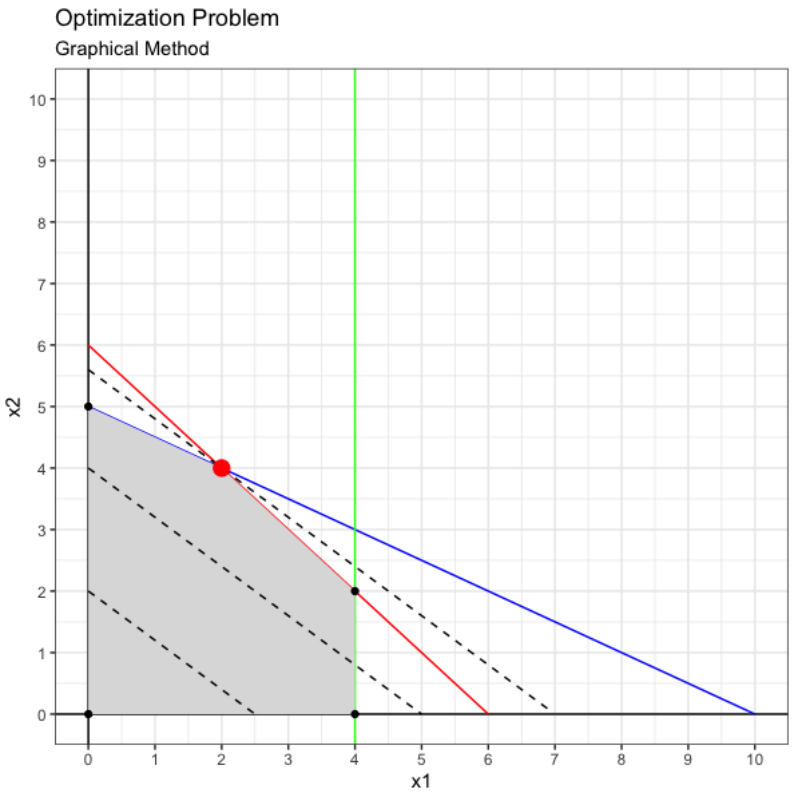
Since the point touching of the objective function line belongs to the first and second constraints lines (i.e. red and blue lines), these constraints represent binding constraint, meaning that there is no surplus or slack in them (i.e. they have reached the maximum/minimum acceptable value).

The solution can be confirmed by applying the simplex algorithm using the *lpSolve* R package:

```r
# Import lpSolve package
```

```r
library(lpSolve)

# Set coefficients of the objective function
f.obj <- c(4, 5)

# Set matrix corresponding to coefficients of constraints by rows
f.con <- matrix(c(1, 2,
                  6, 6,
                  1, 0), nrow = 3, byrow = TRUE)

# Set unequality/equality signs
f.dir <- c("<=", "<=", "<=")

# Set right hand side coefficients
f.rhs <- c(10, 36, 4)

# Final value (z)
lp("max", f.obj, f.con, f.dir, f.rhs)

# Variables final values
solution <- lp("max", f.obj, f.con, f.dir, f.rhs)$solution

# Highlight optimum solution in plot
p <- p + geom_point(aes(x = solution[1], y = solution[2]), color = "red", size = 4)

# Print plot
print(p)
```

Graphical Method - Simplex Method.R hosted with ♡ by GitHub view raw



As seen on the plot above, the simplex algorithm final outcome was (2,4), same as the outcome reached using the graphical method. The red dot (i.e. simplex solution) matches the last point the objective function line (i.e. black dashed line) touches before exiting the feasible region, thus, confirming that $x1 = 2$ and $x2 = 4$ leads the maximum possible value of z, 28.

Concluding Thoughts

The graphical method represents one approach for solving linear programming problems. However, this approach is limited to optimization problems containing just

two decision variables since constraints are plotted in a two-dimensional space. Every additional decision variable would add and additional space in the plot and would complicate its visualization.

In reality, most optimization problems will contain more than two decision variables of different types (e.g. linear, integer, binary) with more complex constraints, limiting the applicability of the graphical method for solving them.

In addition, building a two-dimensional space plot and displacing the objective function line along the feasible region to find the optimum extreme point is time consuming and not the most effective approach. The application of other optimization algorithms in multiple computer programs (e.g. R, GAMS, AMPL, LINDO) reduces significantly the time spent solving an optimization problem while obtaining additional valuable information (e.g. sensitivity analysis).

— —

If you found this article useful, feel welcome to download my personal codes on GitHub. You can also email me directly at rsalaza4@binghamton.edu and find me on LinkedIn. Interested in learning more about data analytics, data science and machine learning applications in the engineering field? Explore my previous articles by visiting my Medium profile. Thanks for reading.

- Robert

Industrial Engineering | Optimization | Operations Research | Engineering | Programming