How to Cite:
Almosa, N. A. A., & Al-Jilawi, A. S. (2022). Solving linear programming problem using approximate optimization method. *International Journal of Health Sciences*, *6*(S1), 5258–5270.
https://doi.org/10.53730/ijhs.v6nS1.6043

Solving linear programming problem using approximate optimization method

Nadia Ali Abbas Almosa
College of Information Technology, Networking Department, University of Babylon, Iraq
Email: na438487@gmail.com

Dr. Ahmed Sabah Al-Jilawi
College of Education for Pure Sciences, Department of Mathematics, University of Babylon, Iraq.
Email: aljelawy2000@yahoo.com

***Abstract*---**In this paper, we will define optimization, linear programming, and the duality of linear programming and demonstrate them in practice through several examples in which the Python language was used to display the final outputs using codes for various libraries. We will also illustrate the method of approximation to Tyler by defining the strategy and demonstrating it in practice through an example.

***Keywords*---**Linear programming, Duality in linear programming, Approximations with Taylor Series and Python language.

Introduction

Finding the optimal answer out of a plethora of viable options is what optimization entails. The only solutions that meet all of the restrictions in the optimization problem are those that are feasible [12]. Minimizing a process's cost or increasing a system's efficiency may be the best answer. Machine allocation and diet difficulties are two easy optimization problems that spring to mind. In the machine allocation problem, one must determine how tasks should be distributed among several machines with varying capacities and running costs to fulfill the production goal at the lowest possible cost. Varied food varieties are accessible with different nutritional contents at different prices for the diet problem. The goal is to estimate various amounts of food to supply an individual's dietary needs at a low cost. Though the foundations of optimization problems can be traced back to about 300 B.C., when the Greek mathematician Euclid assessed the minimal distance between a point and a line, the roots can be traced back to around 300 B.C. In the year 200, another Greek mathematician,

Zenedorous, demonstrated that a semicircle is a figure surrounded by a line with the most significant area for a given perimeter. Pierre de Fermat, a French mathematician, set the foundations of calculus in the 17th century [19]. He demonstrated that a function's gradient disappears at its highest or lowest point. Newton and Leibniz developed the mathematical foundations for the calculus of variations as the chronology progressed. This technique deals with functional maxima and minima. Euler and Lagrange (in the 18th century) are recognized for laying the groundwork for the calculus of variations since they offered precise mathematical details on the subject. Gauss and Legendre then devised the least-squares approach, which is still widely used today. To handle unconstrained optimization problems, Cauchy employed the steepest descent approach [7].

Applications of Optimization is a topic that is being utilized more and more in the domains of science, engineering, economics, management, and industry, among others. It is concerned with determining the best of many alternative options in a real-world setting, developing computational methods for identifying optimum solutions[15], researching theoretical properties, and analyzing the computational performance of numerical algorithms implemented using computational methods. With the rapid expansion of high-performance computers and advancements in computational approaches, many large-scale optimization challenges have been examined and handled. According to Harvard University's Professor Yuqi He, a member of the U.S. National Academy of Engineering, optimization is a cornerstone of civilization's advancement [6].

The optimization formulation in question should only be treated as a rough approximation. Modeling ability, which allows you to capture the most important aspects of a situation. To reach significant conclusions, you'll need both excellent judgments in interpreting the findings and sound judgment in interpreting the results. As a result, optimization should be seen as a tool for conceptualizing and analyzing rather than a principle that produces the theoretically perfect answer. Concrete, practical experience, and a solid comprehension of relevant theory improve skill and excellent judgment when problem creation and interpreting of findings. When it comes to problem formulation, there is always a compromise between the competing goals of creating a mathematical model that is sophisticated enough to correctly represent the problem description and creating a model that is tractable [1].

The linear programming

Linear programming is an optimization technique for solving problems when the objective function and constraints are represented as linear functions of the choice variables [13]. In a linear programming problem, the constraint equations might be equalities or inequalities. Economists initially noticed the linear programming kind of optimization problem in the 1930s while creating approaches for the best resource allocation. During World War II, the United States Air Force looked for more efficient resource allocation methods and resorted to linear programming. In 1947, Air Force group member George B. Dantzig developed the general linear programming problem and created the simplex technique of solution which is mean is a method for manually solving linear programming models employing slack variables, tableaus, and pivot

variables to find the best solution to an optimization problem. Simplex tableau is used to execute row operations on the linear programming model and to ensure that it is optimum . This is a huge step forward in the adoption of linear programming. After then, there was a lot of progress in both theoretical research and practical applications of linear programming. Among all of their contributions, Kuhn and Tucker's theoretical contributions had a significant effect on the development of duality theory in L.P. [10].
An L.P. program is an optimization program of the form [5]:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t} \quad & Ax \geq b \\ & Ax = 0 \\ & x \geq 0 \end{aligned}$$

Linear programming is an optimization problem in which the goal and constraints are expressed as a linear function of the design variables. The limitations may be the equality, inequality, or both types [14]. Applications for linear programming may be found all over the place. We employ linear programming on a personal and professional level. When we wish to go from home to work as quickly as possible, we employ linear programming.

The Python language

Python is an easy to learn high level programming language. Python is a flexible language that can be used to manage the performance of many programs in general, as well as to construct independent programs with graphical interfaces and in web applications. It can also be used as a scripting language to control the performance of many programs. Python is a programming language that may be used to create basic programs for novices as well as to complete huge projects simultaneously [16]. Guido van Rossum created Python at the Institute of Mathematics and Informatics in Amsterdam in the late 1980s, and it was initially released in 1991 [3].

Optimizing Python Code

Choosing the best solution from a multitude of options accessible to developers is part of the process of optimizing Python applications. For online and A.I. development, Python is the most frequently utilized, dynamic, and adaptable programming language. Even in activities like machine learning and data mining, Python remains the best and most relevant programming language for application developers [18]. Python may be used to improve the fit of data in a model, boost the profitability of a proposed engineering design, or achieve any other sort of goal that can be stated quantitatively using variables and equations.

Example:
We have a plant that manufactures product and product, which are two different commodities. Each product requires the use of both materials and labor. Every product sold creates income.

The table below shows the required per unit material, labor input, and revenue:

Material	3	6
Labor	3	4
Revenue	5	4

32 units of material and 25 units of labor available
The challenge for a company is to devise a production plan that maximizes income by using its 30 units of resources and 20 units of work.

Solution:
This problem may be stated as follows:

$$\max_{x_1,x_2} z = 5x_1 + 4x_2$$

$$\text{s.t} \quad 3x_1 + 6x_2 \leq 32$$

$$5x_1 + 4x_2 \leq 25$$

$$x_1, x_2 \geq 0$$

To find max profit by using python code

```
from gekko import GEKKO
m = GEKKO()
x1 = m.Var(lb=0, ub=5) # Product 1
x2 = m.Var(lb=0, ub=4) # Product 2
m.Maximize(5*x1+4*x2) # Profit function
m.Equation(3*x1+6*x2<=32) # Units of A
m.Equation(5*x1+4*x2<=25) # Units of B
m.solve(disp=False)
p1 = x1.value[0]; p2 = x2.value[0]
print ('Product 1 (x1): ' + str(p1))
print ('Product 2 (x2): ' + str(p2))
print ('Profit       : ' + str(5*p1+4*p2))
```

Output

Product 1 (x_1): 3.7746175863
Product 2 (x_2): 1.5317280156
Profit : 24.9999999939

By using python code to find graphical

```
import numpy as np
from scipy.optimize import linprog
import matplotlib.pyplot as plt
from matplotlib.patches import Polygon
import inline as inline
fig, ax = plt.subplots(figsize=(8, 6))
ax.grid()
# Draw constraint lines
ax.hlines(0, -1, 17.5)
ax.vlines(0, -1, 12)
ax.plot(np.linspace(-1, 17.5, 100), 6-0.4*np.linspace(-1, 17.5, 100), color="c")
ax.plot(np.linspace(-1, 5.5, 100), 10-2*np.linspace(-1, 5.5, 100), color="c")
ax.text(1.5, 8, "$3x_1 + 6x_2 \leq 32$", size=12)
ax.text(10, 2.5, "$3x_1 + 4x_2 \leq 25$", size=12)
ax.text(-2, 2, "$x_2 \geq 0$", size=12)
ax.text(2.5, -0.7, "$x_1 \geq 0$", size=12)
feasible_set = Polygon(np.array([[0, 0],
                                 [0, 6],
                                 [2.5, 5],
                                 [5, 0]]),
                       color="cyan")
ax.add_patch(feasible_set)
ax.plot(np.linspace(-1, 5.5, 100), 3.875-0.75*np.linspace(-1, 5.5, 100), color="orange")
ax.plot(np.linspace(-1, 5.5, 100), 5.375-0.75*np.linspace(-1, 5.5, 100), color="orange")
ax.plot(np.linspace(-1, 5.5, 100), 6.875-0.75*np.linspace(-1, 5.5, 100), color="orange")
ax.arrow(-1.6, 5, 0, 2, width = 0.05, head_width=0.2, head_length=0.5, color="orange")
ax.text(5.7, 1, "$z = 5x_1 + 4x_2$", size=12
ax.plot(2.5, 5, "*", color="black")
ax.text(2.7, 5.2, "Optimal Solution", size=12)

plt.show()
```

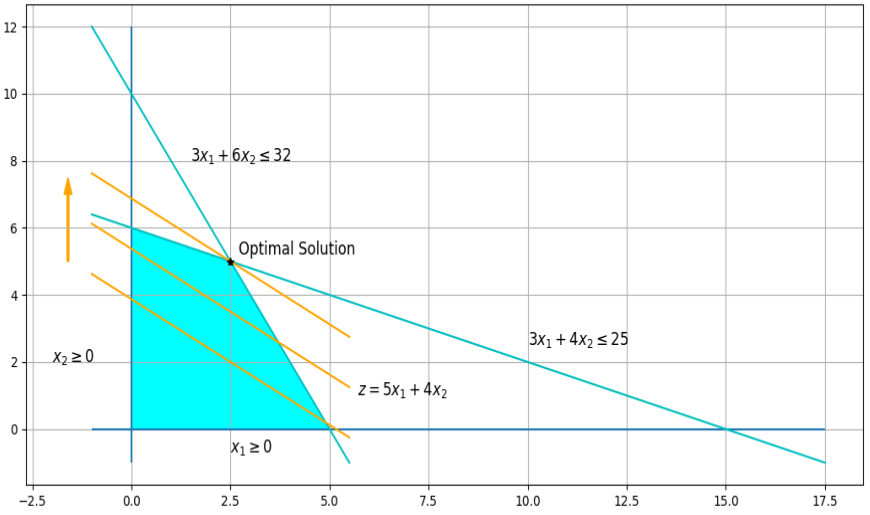

Figure (1): The feasible set within which all constraints are met is the blue area**,** and Intersection of constraints is the optimal solution, and revenue lines are parallel orange lines. The firm's goal is to locate the parallel orange lines that go from the upper border of the feasible set to the upper boundary of the feasible set. Example: discover the linear programming problem's objective value:

$$\begin{aligned} \text{Max} \quad & z = 10x_1 + 8x_2 \\ \text{s.t} \quad & 4x_1 + 3x_2 \leq 350 \\ & 5x_1 + 6x_2 \leq 497 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Solution:

by using python code

```
# import PuLP
from pulp import *

# Create the 'prob' variable to contain the problem data
prob = LpProblem("The Product maximization", LpMaximize)

# Create problem variables
x1=LpVariable(x_1",0,None,LpInteger)
x2=LpVariable("x_2",0, None, LpInteger)

# The objective function is added to 'prob' first
prob += 10*x1 + 8*x2, "Profit to be maximized"
# The two constraints are entered
prob += 4*x1 + 3*x2 <= 350, "Machine X constraint"
prob += 5*x1 + 6*x2 <= 497, "Machine Y constraint"

# The problem data is written to an .lp file
prob.writeLP("Productmax.lp")
# The problem is solved using PuLP's choice of Solver
prob.solve()
# The status of the solution is printed to the screen
print("Status:", LpStatus[prob.status])

for v in prob.variables():
   print(v.name, "=", v.varValue)

# The optimised objective function value is printed to the screen
print("Total units can to be manufactured = ", value(prob.objective))
```

Output
Status: Optimal
x_1= 68.0
x_2= 26.0
Objective value: 888

Duality in linear programming

Every primary linear programming problem is accompanied by a secondary linearprogramming problem, known as the dual. These two situations have a lot of similarities and are pretty intriguing [18]**.** If you know the best answer to one,

you can easily find the best solution to the other. It makes no difference whether problem is called the primordial since the dual of a dual is the primal. The answer to a linear programming problem may be achieved by solving either the primal or the dual, according to these features [9].

The primal linear program [11]:

$$minmize\ z = c^T x$$
$$\text{S. t} \quad Ax = b$$
$$x \geq 0$$

Where A ∈ $R^{m\times n}$, b ∈ R^{m}, and $c \in R^n$ and its dual linear program:

$$maxmize\ w = b^T y$$
$$\text{subject to } A^T y \leq c$$
$$y \geq 0$$

Example: find the dual of the primary problem using python code:

$$max \quad 3x_1 + 4x_2$$
$$\text{Subject to} \quad 2x_1 + 5x_2 \leq 30$$
$$4x_1 + 2x_2 \leq 20$$
$$x_1, x_2 \geq 0$$

Solution: the dual problem

$$min \ \ 30y_1 + 20y_2$$
$$\text{Subject to} \quad 2y_1 + 4y_2 \geq 3$$
$$5y_1 + 2y_2 \geq 4$$
$$y_1, y_2 \geq 0$$

Solve the dual problem by using python code

```
# import the library pulp as p
import pulp as p

# Create a dual LP Minimization problem
Dual_Lp_prob = p.LpProblem('Problem', p.LpMinimize)

# Create problem Variables
y_1 = p.LpVariable("y_1", lowBound = 0) # Create a variable x >= 0
y_2 = p.LpVariable("y_2", lowBound = 0) # Create a variable y >= 0

# Objective Function
Dual_Lp_prob += 30 * y_1 + 20 * y_2

# Constraints:
Dual_Lp_prob += 2*y_1 + 4 * y_2 >= 3
Dual_Lp_prob += 5*y_1 + 2*y_2 >= 4

# Display the problem
print(Dual_Lp_prob )

status = Dual_Lp_prob .solve() # Solver
print(p.LpStatus[status]) # The solution status

# Printing the final solution
print(p.value(y_1), p.value(y_2), p.value(Dual_Lp_prob.objective))
```

Output
Optimal
0.625 0.4375 27.5

Approximation Method

For minimizing a convex function $f: R^n \rightarrow R$ over a convex set X,[**8**]approximation approaches are based on substituting f and X by approximations $\mathrm{F_k}$ *and* X_k respectively, at each iteration k, and finding

$$x \in arg \min_{x \in X_k} F_k(x)$$

F_{k+1} and X_{k+1} are formed in the following iteration by refining the approximation, which is based on the new point x_{k+1} and perhaps on the previous points Xk,…, xo. Of course, this strategy only makes sense if the approximation problems are easier than the original[17]. There are several approximation techniques available, each with its own set of goals and applications.

Approximations with Taylor Series

A Taylor series approximation[2] is when a number is represented as a polynomial that has a very similar value to the number in a neighborhood around a given x value:

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^3(a)}{3!}(x-a)^3 + o(h^3)$$

Taylor series are incredibly useful tools for approximating functions that would otherwise be difficult to calculate, as well as assessing infinite sums and integrals[18].

The steps to approximate values are as follow [4]:
1.Compare the operation on the number in question to a function
2.Assign a to a value that makes f(a) simple to calculate
3. Select x to make f(x) the estimated number.
Example: by using Python to display the exp(x) function together with the Taylor series approximations of the first, third, fifth, and seventh
Solution: Code python Approximations with Taylor Series

```
import numpy as np
import matplotlib.pyplot as plt

plt.style.use('seaborn-poster')
x = np.linspace(-np.pi, np.pi, 200)
y = np.zeros(len(x))

labels = ['Third Order', 'fouth Order', 'Fifth Order', 'Seventh Order']

plt.figure(figsize = (10,8))
for n, label in zip(range(4), labels):
    y = y + ((-1)**n * (x)**(2*n+1)) / np.math.factorial(2*n+1)
    plt.plot(x,y, label = label)

plt.plot(x, np.cos(x), 'k', label = 'Analytic')
plt.grid()
plt.title('Taylor Series Approximations of Various Orders')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

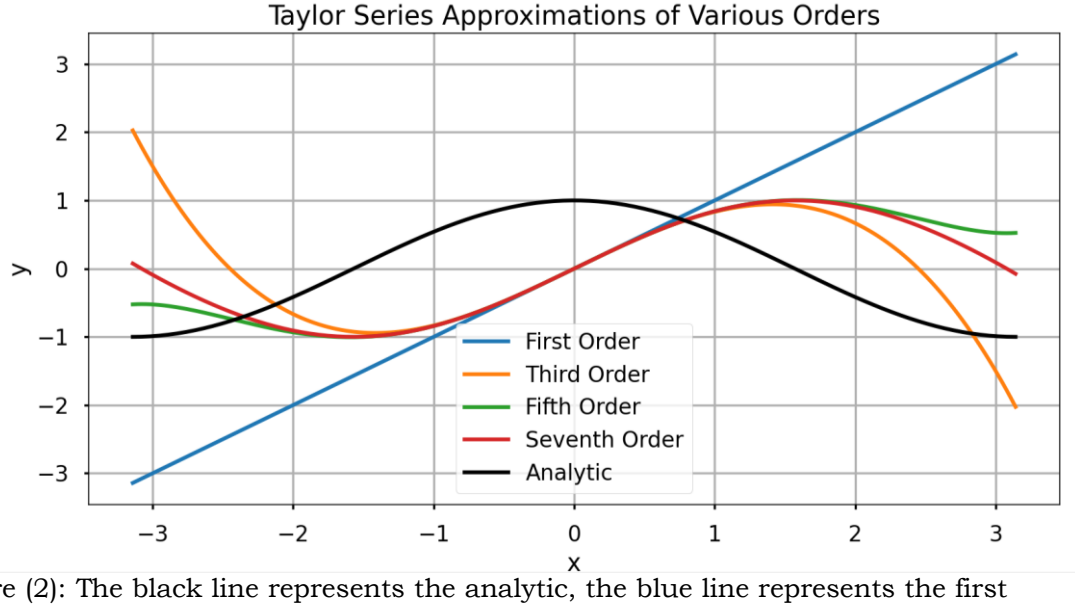

Figure (2): The black line represents the analytic, the blue line represents the first order of the Taylor Series, the orange line represents the third order of Taylor, the

green line represents the fifth order of the Taylor Series, and The red line represents the seventh order of Taylor Series

Conclusion

The core of this research is the use of Python to solve mathematical problems, where the best solution and end products were discovered via the use of Python libraries. We realized that each mathematical problem has a unique code when solved using Python codes

References

[1] Luenberger, D. G., & Ye, Y. (1984). Linear and nonlinear programming (Vol. 2). Reading, MA: Addison-Wesley.

[2] Hlawitschka, W. (1994). The empirical nature of Taylor-series approximations to expected utility. The American Economic Review, 84(3), 713-719.

[3] Sanner, M. F. (1999). Python: a programming language for software integration and development. J Mol Graph Model, 17(1), 57-61.

[4] Herrera, F., & Herrera-Viedma, E. (2000). Linguistic decision analysis: steps for solving decision problems under linguistic information. Fuzzy Sets and systems, 115(1), 67-82.

[5] Ben-Tal, A., & Nemirovski, A. (2001). Lectures on modern convex optimization: analysis, algorithms, and engineering applications. Society for industrial and applied mathematics.

[6] Sun, W., & Yuan, Y. X. (2006). Optimization theory and methods: nonlinear programming (Vol. 1). Springer Science & Business Media.

[7] Arora, R. K. (2015). Optimization: algorithms and applications. CRC Press

[8] Bertsekas, D. (2015). Convex optimization algorithms. Athena Scientific.

[9] Rao, S. S. (2019). Engineering optimization: theory and practice. John Wiley & Sons.

[10] Rao, S. S. (2019). Engineering optimization: theory and practice. John Wiley & Sons

[11] Andréasson, N., Evgrafov, A., & Patriksson, M. (2020). *An introduction to continuous optimization: Foundations and fundamental algorithms*. Courier Dover Publications

[12] Osaba, E., Villar-Rodriguez, E., Del Ser, J., Nebro, A. J., Molina, D., LaTorre, A., ... & Herrera, F. (2021)

[13] Alridha, A., Wahbi, F. A., & Kadhim, M. K. (2021). Training analysis of optimization models in machine learning. International Journal of Nonlinear Analysis and Applications, 12(2), 1453-1461.

[14] Alridha, A., Salman, A. M., & Al-Jilawi, A. S. (2021, March). The Applications of NP-hardness optimizations problem. In Journal of Physics: Conference Series (Vol. 1818, No. 1, p. 012179). IOP Publishing.

[15] Salman, A. M., Alridha, A., & Hussain, A. H. (2021, March). Some Topics on Convex Optimization. In Journal of Physics: Conference Series (Vol. 1818, No. 1, p. 012171). IOP Publishing.

[16] Alridha, A., & Al-Jilawi, A. S. (2021, March). Mathematical Programming Computational for Solving NP-Hardness Problem. In Journal of Physics: Conference Series (Vol. 1818, No. 1, p. 012137). IOP Publishing

[17] Al-Jilawi, A. S., & Abd Alsharify, F. H. (2022). Review of Mathematical Modelling Techniques with Applications in Biosciences. Iraqi Journal For Computer Science and Mathematics, 3(1), 135-144.

[18] Kadhim, M. K., Wahbi, F. A., & Hasan Alridha, A. (2022). Mathematical optimization modeling for estimating the incidence of clinical diseases. International Journal of Nonlinear Analysis and Applications, 13(1), 185-195.

[19] Alridha, A. H., & Al-Jilawi, A. S. (2022). Solving NP-hard problems using a new relaxation of approximate methods. International Journal of Health Sciences, 6(S3), 523–536. https://doi.org/10.53730/ijhs.v6nS3.5375