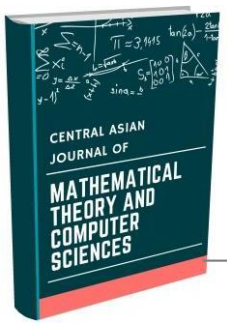

CENTRAL ASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES

http://cajmtcs.centralasianstudies.org/index.php/CAJMTCS

Volume: 03 Issue: 06 | June 2022 ISSN: 2660-5309

Numerical Optimization Approach for Solving Production Planning Problem Using Python language

Ahmed Hasan ALRIDHA

Ministry of Education, General Directorate of Education in Babylon, Iraq.

amqa92@yahoo.com

Abbas Musleh Salman

Ministry of Education, General Directorate of Education in Babylon, Iraq.

abbas.mmm2019@yahoo.com

Ahmed Sabah Al-Jilawi

Mathematics department, University of Babylon, Iraq..

aljelawy2000@yahoo.com

Abstract:

Optimization Technique aims to find the best possible option that meet all the imposed constraints. In fact, the goal in this paper is to find the best values for the decision variables for various inventory systems in order to maximize net profit by the field of numerical optimization for linear programming using software. Besides, the optimize procedure was applied on the production planning problem (vegetable oil production Problem) models in many methods which is equations and objective, dense matrices and sparse matrices. The optimization was implemented within the Python by GEKKO package. Finally, the numerical results of the techniques used showed a strong convergence in calculating the amount of profit and production lines, which indicates their efficiency and accuracy in obtaining optimal solutions.

ARTICLEINFO

Article history:

Received 15 Mar 2022

Revised form 13 Apr 2022

Accepted 12 May 2022

Keywords: Optimization Technique, linear programming, production planning problem, Python (Gekko) package.

***-

1. INTRODUCTION

One of the most significant managerial roles for manufacturing managers is the planning and usage of production. Such judgments must be taken in the face of uncertainty in a number of key criteria, the most important of which is market demand for the products being produced. From business to engineering design, from vacation planning to daily routine, optimization is ubiquitous. Business enterprises must maximize profits while minimizing costs. Engineering design must maximize the performance of the proposed product

while also minimizing the expense. Even when we plan vacations, we want to maximize our experience while minimizing our expenses. As a result, optimization studies are of scientific interest as well as practical ramifications, and the methodology will have various applications as a result. Production planning is a strategy plan that businesses follow whenever they intend to manufacture things [1,2,3] . Production planning entails confirming the product to be manufactured, the production volume, capacity planning, the materials required, the scheduling schedules, and so on. This is critical for firms who want to maximize efficiency, cut costs, and have a long-term production cycle. For production planning procedures there are some of the major milestones in the manufacturing plan's progression:

1. Examine consumer demand
2. Analyzing production capacity and schedules
3. Assess raw materials
4. Production and quality control, as well as accounting
5. Production system evaluation and optimization
6. Complete final production of finalized goods.

Figure 1. Steps in Production Planning

In case of vegetable oil production manufacture factory, the following will be included in the production planning:

1- Factory level planning

Planning the activity sequence (purchasing raw materials for processing vegetable oils until production, packaging, etc.) and this is represented as an objective function according to the mathematical model of the optimization problem. In both the objective function and constraints, we will deal with the production planning problem with interval data as uncertainty.

2- Process and Operation level planning

Planning operations on the inputs to convert them to the desired outputs (the amount of product concentration, the amount of mixing raw materials, the amount of packaging needed for the product, etc.) and these are represented according to the mathematical model as constraints for the optimization function [4,5].

2. MATHEMATICAL PROGRAMMING

A Mathematical Programming (or Mathematical Optimization) problem can be formulated as:

$$\begin{cases} m & f(a) \\ a \in A \end{cases}$$

where $A \subseteq \Re^n$ is the set of feasible solutions and $f: A \to \Re$ is the objective function. The convention is to formulate each problem as a minimization problem and, where appropriate, operate the substitution

$$\max\{f(a): a \in A\} = -\min\{-f(a): a \in A\}.$$

A problem for which there is no feasible solution ($X = \emptyset$) is said to be infeasible; in this case, we agree to write $\min\{f(a): a \in A\} = +\infty$. A problem for which f is not bounded from below in X is hence said to be an unbounded problem; in this case, we write $\min\{f(a): a \in A\} = -\infty$. Solving problem consists in identifying an optimal solution, if any, i.e., a solution $x^* \in X$ such that $f(a^*) \leq f(a)$ for all $a \in A$. This solution is not necessarily unique. A solution $\bar{a} \in A$ is said to be locally optimal if $\varepsilon > 0$ exists such that $f(\bar{a}) \leq f(a)$ for all $a \in A$ with $\| a - \bar{a} \| \leq \varepsilon$.

Linear Programming with GEKKO Optimization

Optimization is concerned with choosing the optimal option from a set of alternative possibilities that are either feasible or do not break the constraints. Actually, python can be used to optimize model parameters to better suit the data, boost the profitability of a potential engineering design, or achieve another type of goal that can be stated quantitatively with variables and equations. For example, GEKKO is a Python tool that allows you to do machine learning and optimization on mixed-integer and differential algebraic problems. Moreover, It is used in conjunction with large-scale solvers for linear, quadratic, nonlinear, and mixed integer programming (LP, QP, NLP, MILP, MINLP). Parameter regression, data reconciliation, real-time optimization, dynamic simulation, and nonlinear predictive control are all modes of operation. GEKKO is a Python object-oriented module that allows for local execution, [6]. A Linear Programming (LP), or Linear Optimization, problem is a mathematical programming problem of the kind:

$$\min f(x) \Rightarrow \min c^T x$$
$$g_i(x) \leq 0 \Rightarrow \begin{cases} -a_i^T x + b_i \leq 0\,, & i \in \{1, \dots, m\} \\ -x_j \leq 0\,, & j \in \{1, \dots, n\} \end{cases}$$

Grouping the row vectors $a_1^T, \dots, a_m^T$ into a $m \times n$ matrix A we obtain the compact representation

$$\min\{c^T x: Ax \geq b, x \geq 0\}$$

3. PRODUCTION PLANNING PROBLEM

The goal of the production planning problem is to maximize revenue. The demand and resource levels are considered to be fixed and given in order to solve it. However, there is a production planning issue because there are finite production resources that cannot be stored from period to period. Similarly, the planning problem begins with a specification of the client demand to be met by the production plan. Future demand is only partially known in most cases. As a result, one relies on a prognosis for future demand, yet the forecast is incorrect. As a result, demand that cannot be met in a given period is lost, diminishing revenue. Stephen C. Graves proposes a production planning problem to optimize revenues net of production, inventory, and missed sales costs. The objective function coefficients, inequalities, and equalities restrictions are expected to be known numbers in order to solve it, [7,8].

linear-Programming (LP) Formulation for Production Planning Problem

In order to describing the general formula of this problem, there must be three basic components of the optimization function:

1- The Decision Variables

The goal is to determine the level of production for each month. So we have twelve decision variables:

= production level for month

2- The Objective Function

Take a look at the first month . As a result we have:

The cost of production is equal to c_1x_1. The inventory-holding cost equals $h_1(x_1 - p_1)$, assuming that the ending inventory level, $x_1 - p_1$, is nonnegative. If the ending inventory level is $x_1 - p_1$,the inventory-holding cost is $h_1(x_1 - p_1)$. As a result, the first month's total cost equals to $c_1x_1 + h_1(x_1 - p_1)$. We have the following for the second month:

The cost of production is equal to c_2x_2. If the ending inventory is $x_1 - p_1 + x_2 - p_2$, is non-negative. The inventory-holding cost is $h_2(x_1 - p_1 + x_2 - p_2)$. This is due to the fact that the beginning .This month's inventory level is $x_1 - d_1$, while this month's production level is is x_2 and p_2 is the demand for this month. As a result, the second month's total cost is c_2x_2+$h_2(x_1 - p_1 + x_2 - p_2)$. Continuation of this argument yields that:

The total production cost for the full planning horizon equals

$$\sum_{j=1}^{12} c_j x_j \equiv c_1x_1 + c_2x_2 + \cdots + c_{12}x_{12}$$

where we have added the customary summation notation ("" indicates by definition).The Constraints.

3- The Constraints:

According to the month's production capacity j is m_j, we need $x_j \le m_j$ for $j = 1,2,\dots,12$;

Since deficiency is not allowed (by the third assumption), we need $\sum_{k=1}^{j} (x_k - p_k) \ge 0$ for $j = 1,2,\dots,12$.

As a result,

a collection of 24 functional constraints is created. The $x_j's$, should be of course nonnegative because they are production levels, [9,10].

The Final LP Formulation

In conclusion, we've come up with the following formula:

$$\text{Minimize } \sum_{j=1}^{12} c_j x_j + \sum_{j=1}^{12} h_j\left[\sum_{k=1}^{j} (x_k - p_k)\right]$$

Subject to:

$$\begin{aligned} x_j &\le m_j \text{ for } j = 1,2,\dots,12 \\ \sum_{k=1}^{j} (x_k - p_k) &\ge 0 \text{ for } j = 1,2,\dots,12 \\ x_j \ge 0 \text{ for } j &= 1,2,\dots,12. \end{aligned}$$

There are 12 decision variables, 24 functional constraints, and 12 non-negativity constraints in this linear program. The c_j 's, the h_j 's, the p_j 's, and the m_j 's must all be replaced with explicit numerical values in a real implementation, [11,12].

Algorithm Approach for solving the problem

Step 1: construct the problem of production planning using interval numbers in the objective function and constraints .

Step 2: Determine the interval values according to the required intervals.

Step 3: Convert the non-deterministic problem to the deterministic form.

Step 4: solve the problem by GEKKO software package.

Step 5: Stop.

A simple production planning problem is (vegetable oil production) which given by the use of two ingredients M and N that produce products 1 and 2. The available supply is M=40 units and N=54 units. For production it requires:

4 units of M and 10 units of N to produce Product 1,

8 units of M and 5 units of N to produce Product 2.

There are at most 7 units of Product 1 and 6 units of Product 2. Product 1 can be sold for 150 and Product 2 can be sold for 200. The objective is to maximize the profit for this production problem. To solve this problem and in order to a potential feasible solution we must determine the constraints on the contour plot and mark the contour plot with the set of viable options. Also, identify the maximum and minimum objective feasible solution. The solution to this problem will be offered in many cases by as follows:

Case 1: Equations and Objective

By using equations and an objective function is good for small problems because it is a readable optimization problem and is thereby easy to modify.

Python (Gekko) code for case 1

```
from gekko import GEKKO
m = GEKKO()
x1 = m.Var(lb=0, ub=7) # Product 1
x2 = m.Var(lb=0, ub=6) # Product 2
m.Maximize(150*x1+200*x2)  # Profit function
m.Equation(4*x1+8*x2<=40)  # Units of M
m.Equation(10*x1+5*x2<=54)  # Units of N
m.solve(disp=False)
print ('Profit      : ' + str(150*p1+200*p2))
```

After implementing the above code, the results we obtain are:

Product 1 (x_1): 3.8666666672

Product 2 (x_2): 3.0666666677

Profit : 1193.33333362

Case 2: Dense Matrices

Dense matrix form is also available in Gekko. In this case, two model functions qobj (quadratic objective) and axb (Ax<b) objects are used to create the model. Integer variables for discrete optimization are possible with the APOPT solver (option 1) when the variable is specified with integer=True.

Python (Gekko) code for case 2

```
from gekko import GEKKO
m = GEKKO(remote=False)
c = [150, 200]
M = [[4, 8], [10, 5]]
b = [40, 54]
x = m.qobj(c,otype='max')
m.axb(M,b,x=x,etype='<')
x[0].lower=0; x[0].upper=7
x[1].lower=0; x[1].upper=6
m.options.solver = 1
m.solve(disp=True)
print ('Product 1 (x1): ' + str(x[0].value[0]))
print ('Product 2 (x2): ' + str(x[1].value[0]))
print ('Profit      : ' + str(-m.options.objfcnval))
```

After implementing the above code, the results we obtain are:

Product 1 (x1): 3.8666666667

Product 2 (x2): 3.0666666667

Profit : 1193.3333333

Case 3: Sparse Matrices

Sparse matrices are faster and use less memory for very large-scale problems with many zeros in M, b, and c.

Python (Gekko) code for case 3

```
# solve with GEKKO and sparse matrices
import numpy as np
from gekko import GEKKO
m = GEKKO(remote=False)
# [[row indices],[column indices],[values]]
A_sparse = [[1,1,2,2],[1,2,1,2],[4,8,10,5]]
# [[row indices],[values]]
b_sparse = [[1,2],[40,54]]
x = m.axb(A_sparse,b_sparse,etype='<',sparse=True)
# [[row indices],[values]]
c_sparse = [[1,2],[150,200]]
m.qobj(c_sparse,x=x,otype='max',sparse=True)
x[0].lower=0; x[0].upper=7
x[1].lower=0; x[1].upper=6
m.solve(disp=True)
print ('Product 1 (x1): ' + str(x[0].value[0]))
print ('Product 2 (x2): ' + str(x[1].value[0]))
print ('Profit      : ' + str(-m.options.objfcnval))
```

The results we obtain are:

Product 1 (x1): 3.8666666672

Product 2 (x2): 3.0666666677

Profit : 1193.3333336

4. NUMERICAL RESULTS WITH CONTOUR PLOT

A contour plot can be used to find the best solution. The black lines in this case represent the upper and lower bounds on the output of 1 and 2. In this scenario, the production of 1 must be larger than 0 but less than 7 and the production of 2 must be greater than 0 but less than 6. There are no more than 40 units of M and 54 units of N ingredients available to make items 1 and 2. We obtain the optimal time to solve the problem by applying the contour plot from the previous objective function, as seen in Figure (2) and Table (1):

	scaled	unsealed
Objective	-1.1933333336079327e+03	-1.1933333336079327e+03
Dual infeasibility	2.0925451414764273e-13	2.0925451414764273e-13
Constraint violation	3.6343604551357638e-15	3.6343604551357638e-15
Complementarity	2.0566530598901004e-10	2.0566530598901004e-10
Overall NLP error	2.0566530598901004e-10	2.0566530598901004e-10

Table (1): Numerical Results

The results of the optimization problem

Product 1 (x1): 3.8666666672

Product 2 (x2): 3.0666666677

Profit: 1193.3333336

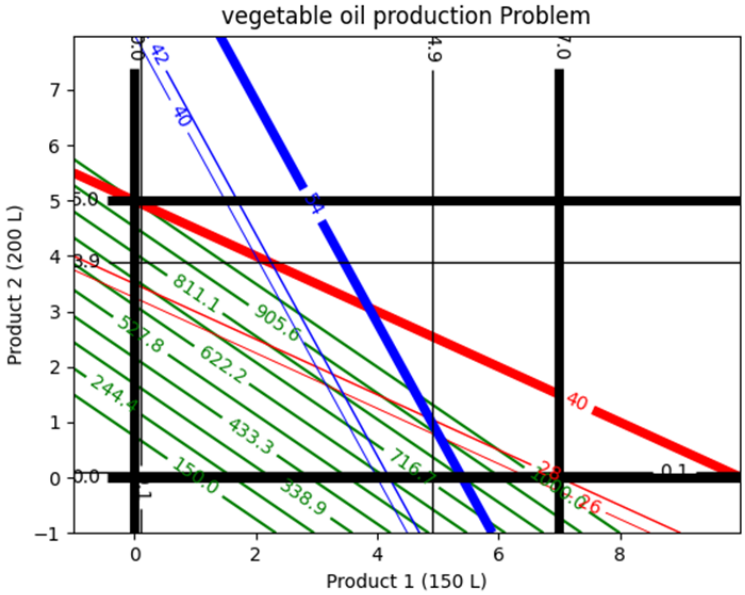

Figure 2. Contour Plot for vegetable oil production Problem

The contour plot was development by the following code for python GEKKO:

Contour Plot Python (Gekko) code

```
# Create a contour plot
plt.figure()
# Weight contours
lines = np.linspace(100.0,800.0,8)
CS = plt.contour(x1,x2,profit,lines,colors='g')
plt.clabel(CS, inline=1, fontsize=10)
# A usage < 40
CS = plt.contour(x1,x2,A_usage,[26.0, 28.0, 40.0],colors='r',linewidths=[0.5,1.0,4.0])
plt.clabel(CS, inline=1, fontsize=10)
# B usage < 54
CS = plt.contour(x1, x2,B_usage,[40.0,42.0,54.0],colors='b',linewidths=[0.5,1.0,4.0])
plt.clabel(CS, inline=1, fontsize=10)
# Container for 0 <= Product 1 <= 500 L
CS = plt.contour(x1, x2,x1 ,[0.0, 0.1, 4.9, 5.0],colors='k',linewidths=[4.0,1.0,1.0,4.0])
plt.clabel(CS, inline=1, fontsize=10)
# Container for 0 <= Product 2 <= 400 L
CS = plt.contour(x1, x2,x2 ,[0.0, 0.1, 3.9, 4.0],colors='k',linewidths=[4.0,1.0,1.0,4.0])
plt.clabel(CS, inline=1, fontsize=10)
# Add some labels
plt.title('Vegetable Oil Production Problem')
plt.xlabel('Product 1 (150 L)')
plt.ylabel('Product 2 (200 L)')
# Save the figure as a PNG
plt.savefig('contour.png')

# Show the plots
plt.show()
```

5. CONCLUSION

The goal of the optimization technique is to discover the best available alternative that meets all of the restrictions. The optimization approach was used on models of the production planning problem (vegetable oil production problem) in a variety of ways, including equations and objectives, dense matrices, and sparse matrices. The GEKKO module in Python was used to implement the optimization. Finally, the numerical results of the methodologies used demonstrated a high level of convergence in estimating the amount of profit and production lines, indicating their efficiency and accuracy in obtaining optimal solutions.

REFERENCES

1. Diwekar, U. M. (2020). Introduction to applied optimization (Vol. 22). Springer Nature.
2. Andersson, J. A., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2019). CasADi: a software framework for nonlinear optimization and optimal control. Mathematical Programming Computation, 11(1), 1-36.
3. Wu, T., Huang, L., Liang, Z., Zhang, X., & Zhang, C. (2021). A supervised learning-driven heuristic for solving the facility location and production planning problem. European Journal of Operational Research.
4. Chakrabortty, R. K., & Hasin, M. A. A. (2013). Solving an aggregate production planning problem by fuzzy based genetic algorithm (FBGA) approach. International Journal of Fuzzy Logic Systems (IJFLS), 3(1), 1-16.

5. Haq, A., Kamal, M., Gupta, S., & Ali, I. (2020). Multi-objective production planning problem: a case study for optimal production. International Journal of Operational Research, 39(4), 459-493.
6. Gladkov, L. A., Gladkova, N. V., & Gromov, S. A. (2017, April). Hybrid fuzzy algorithm for solving operational production planning problems. In Computer Science On-line Conference (pp. 444-455). Springer, Cham.
7. Torkaman, S., Ghomi, S. F., & Karimi, B. (2018). Hybrid simulated annealing and genetic approach for solving a multi-stage production planning with sequence-dependent setups in a closed-loop supply chain. Applied Soft Computing, 71, 1085-1104.
8. Brown, C., Abdelfattah, A., Tomov, S., & Dongarra, J. (2020, September). Design, optimization, and benchmarking of dense linear algebra algorithms on AMD GPUs. In 2020 IEEE High Performance Extreme Computing Conference (HPEC) (pp. 1-7). IEEE.
9. Chen, S., Fang, J., Chen, D., Xu, C., & Wang, Z. (2018, June). Adaptive optimization of sparse matrix-vector multiplication on emerging many-core architectures. In 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS) (pp. 649-658). IEEE.
10. Beal, L. D., Hill, D. C., Martin, R. A., & Hedengren, J. D. (2018). Gekko optimization suite. Processes, 6(8), 106.
11. Gates, N. S., Hill, D. C., Billings, B. W., Powell, K. M., & Hedengren, J. D. (2021, December). Benchmarks for Grid Energy Management with Python Gekko. In 2021 60th IEEE Conference on Decision and Control (CDC) (pp. 4868-4874). IEEE.
12. Bashier, E. B. (2020). Practical Numerical and Scientific Computing with MATLAB® and Python. CRC Press.