Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 2, 2026

A COMPUTATIONAL COMPARISON OF LINEAR OPTIMIZATION SOLVERS IN PYTHON FOR AGGREGATE PRODUCTION PLANNING MODELS

Waheed Ahmad Khan[1], Aamir Shahzad[*2], Sana Shabbir[3], Sidra Ashraf[4], Hassan Raza[5]

[1,3,5]*Faculty of Sciences, Superior University Lahore, Lahore 54000, Pakistan*
[*2]*Department of Mathematics, Faculty of Natural Science and Technology, Baba Guru Nanak University, Nankana Sahib 39100, Pakistan*
[4]*Department of Mathematics, Faculty of Sciences, University of Sargodha, Sargodha 40100, Pakistan*

[1]waheed.ahmadsb31@gmail.com, [*2] aamir.shahzad@bgnu.edu.pk,
[3] sanamaliksgd@gmail.com, [4]sideeera@gmail.com, [5]hassanraza9945@gmail.com

DOI: https://doi.org/10.5281/zenodo.18766104

Keywords
Aggregate Production Planning, Productivity, Python PuLP, Python SciPy, Python CVXPY, Optimization.

Article History
Received: 21 December 2025
Accepted: 05 February 2026
Published: 25 February 2026

Corresponding Author: * Aamir Shahzad

Abstract

Aggregate Production Planning (APP) plays a dynamic role in leveling production rates, manpower levels, and inventory over a planning scope based on demand prediction. This paper investigates a comparative analysis of three open-source Python solvers PuLP(using the CBC solver), SciPy(using the linprog method), and CVXPY(using the SCS solver) to solve the APP problem under a fixed workforce model in textile industry. Instead of introducing productivity loss as in some earlier models, this research keep constant workforce levels to simplify manpower-related cost and to emphasize solver efficiency. The first goal is to measure the computational efficiency, solution accuracy, and utility of each solver when applied to the same linear programming(LP) formulation of the APP problem. On a constant data set, all solvers are tested to assure fairness in comparison. The results highlight the capability and limitations of each solver, offering valuable modality for researchers and professionals intent to select proper tools for production planning project using Python-based optimization.

1. Introduction and Literature Review

Optimization is a mathematical technique which is used to find the best optimal solution to a problem by satisfying the given constraints. It plays a vital role in decision-making in different industries like finance, logistics and manufacturing. Aggregate production planning (APP) in production management based on optimization techniques that are use to balance demand and supply, by checking cost-effective resource allocation and effective scheduling. Linear program- ming(LP) is one of the most globally used optimization method for solving APP problems. It allows industries to find the optimal production plan for maximizing profit, minimizing cost and execution time by using different constraints such as raw material, labour cost, demand and supply fluctuations [1, 3]. Python, a high level programming language offers many optimization solvers for linear programming

problems. PuLP, SciPy and CVPXY are generally used to LP problems. By integrating it with linear discriminant analysis, Gangkou et al. (2003) [2] proposed a multiple criteria linear programming (MCLP) approach to data mining. They developed data mining algorithms by using MCLP, described how to use them on SAS and Linux, and evaluate their performance on experimental and commercial databases. The research demonstrated that MCLP methods can outperform induction decision tree techniques in some cases and effectively discover knowledge patterns. Stefan Vo and David L. Woodruff (2006) [3] focused on the significance of integrated and optimized decision-making within sup- ply chain management with an emphasis on tactical production planning. The book started with simple but effective models before increasingly addressing more advanced topics such as congestion and uncertainty. It pointed out how the application of advanced optimization software for production planning had risen with improvements in computing power and in- formation technology. At the intermediate level of a hierarchical decision-making process, the aggregate production planning difficulties were studied by Oscar S. Silva Filho et al. (2010) [4]. In order to predict managerial methods for future resource allocation, these difficulties were utilized. An Excel spreadsheet-based managerial decision support system was implemented in the study to address issues with aggregate production planning. The application of the tool was demonstrated using a case study from the literature, and its computational aspects were introduced. Gilvan C. Souza (2014) [5] investigated supply chain management's use of advanced analytics techniques. Based on the domains of the SCOR model plan, source, make, deliver, and return the study divided these applications into three categories: descriptive, predictive, and prescriptive analytics. Real-time supply chain insights were obtained by descriptive analytics using data from GPS, RFID chips, and visualization tools. Demand forecasting at the strategic, tactical, and operational levels was the main emphasis of predictive analytics in order to assist planning procedures. By combining descriptive and predictive models, prescriptive analytics used mathematical optimization and simulation techniques to improve decision-making. Gholamian et al. (2015) [6] developed a fuzzy multi-objective mixed-integer nonlinear programming (FMOMINLP) model for supply chain production planning in the real world. The model of their's deals with uncertainty in aggregate production planning involved multi-site, multi-period, and multi-product by considering multiple suppliers, manufacturers, and customers. Four conflicting objectives cost minimization, customer satisfaction, workforce stability, and purchasing valueare optimized in this research. A comparative evaluation and verification by way of a genuine industrial SC case study was made after the model was re- formulated as a multi-objective mixed-integer linear programming (MOMILP) and optimized using two methodologies. A mathematical model for aggregate production planning in a pump manufacturing company was created by Anand Jayakumar et al. (2017) [7]. To help planners choose industrial processes and inventory strategies, a formulation based on process selection and lot-sizing models was suggested. A case study was carried out using a mixed method that permitted adjustments to labor and inventory levels during the course of the planning horizon. Python was utilized to effectively tackle the problem, and the study showed that optimization models could identify the best mixed strategy. Three methods of decision-making for inventory management, sales, and production planning were introduced by Jorge Luiz Biazzi (2018) [8]. The analysis considered a situation with limited storage space, non-stationary probabilistic de-mand, production capacity limitations, and revenue losses due to shortages. While the second method included safety inventories without accounting for stockouts, the first method approached the problem in a linear

and predictable manner. The third method calculated short- ages caused by erratic demand. The study showed that creating more complex models could not always lead to appreciable benefits using Microsoft Excel solvers. In their study, Demirel et al.(2018) [9] studied how demand uncertainty affected production planning and suggested incorporating the Flexibility Requirements Profile (FRP) into traditional aggregate planning. To impose flexible limits on production plans, they structured the problem as a mixed-integer linear program with additional constraints. The study contrasted FRP-based aggregate plan- ning with conventional techniques using numerical experiments conducted in the textile and automotive industries. According to the results, the incorporation of FRP resulted in more consistent production schedules without significantly increasing costs. Aggregate production planning (APP) models were thoroughly reviewed by Ali Cheraghalikhania et al. (2018) [10]. Identifying gaps in the literature and rigorously classifying APP models were the objectives of their survey. By highlighting modeling structures, important problems, and approaches to solutions, the report demonstrated how APP research has evolved since Nam and Logendran's 1992 review. This study offered a more comprehensive assessment of APP characteristics than previous reviews, which were primarily methodological in nature. The authors also provided a roadmap for future research aimed at improving APP models and techniques. For aggregate production planning (APP) in a fuzzy context, Jamalnia and Soukhakian (2018) [11] created a hybrid fuzzy multi- objective nonlinear programming (H-FMONLP) model. Their methodology aimed to maximize customer satisfaction while reducing expenses associated with production, storage, backorders, and labor change. The study incorporated the limitations of inventory, demand, labor, machine capacity, and storage space through an interactive decision- making process. The relevance of the model was illustrated with a real-life case study, and the GENOCOP III algorithm was used to solve the final accurate nonlinear programming problem. Parganiha K. (2018) [12] studied how linear programming can be used to optimize resource allocation in various sectors. The main objective of the study was to use Python and the PuLP module to solve a product mix optimization problem. By creating a mathematical model and using a solver, the objective was to identify the best combination that would maximize profit. Python was used to create a graphical representation of the feasible region, which helped locate the best solution at the corner where the profit was highest. The social process of adopting Big Data and Predictive Analytics (BDPA) in retail supply chain management was studied by Sodero, Jin, and Barratt (2019) [13].According to their research, BDPA technology needed to be adjusted for social and organizational contexts before it could be used. They found that the implementation of BDPA was influenced by institutional and social issues, making it difficult to transfer between cooperating organizations. The study clearly showed that to ensure effective use of BDPA, managers must pay close attention to the institutional context. A theoretical study of manufacturing decisions in supply chain management was conducted by Shete et al. (2019) cite14, who emphasized the importance of making informed decisions in inventory management and manufacturing. Backpropagation neural networks and multi-layer perceptrons, two artificial intelligence technologies, were used in the study to develop a forecasting method. Using real data from a Pune-based industrial product manufacturing company, the effectiveness of the proposed method in forecasting demand was illustrated. Rehman et al. (2021) [15] studied how productivity loss affects conventional aggregate pro- duction planning, which normally bases labor, production, and inventory decisions on demand forecasts without considering productivity loss. To address this issue in various sectors, the study incorporated productivity loss into aggregate

production planning (APP), applied the executory evaluation approach, and linear programming (LP) methods. A fixed workforce and a flexible workforce were the two model variations discussed. To assess the impact of productivity loss, the PuLP library was used to solve mathematical models. Computational results showed that hiring and termination decisions were directly affected by production loss. To find the optimal solutions for inventory systems while respecting the imposed restrictions, Alridha et al. (2022) [16] studied optimization strategies. By using numerical optimization techniques for linear programming, the study sought to maximize net profit in production planning, particularly in the manufacturing of vegetable oil. Numerous methods were applied, such as objective functions, dense and sparse matrices, and equations. The GEKKO package was used to implement the optimization in Python. Strong convergence in profit and production line calculations was shown by the numerical findings, demonstrating the effectiveness and precision of the methods used to arrive at the best answers. With a focus on Kenya Power and Lighting Company, Patrick et al. (2022) [17] examined the use of predictive analytics in supply chain management within utility firms. Although the amount of data has increased due to information technology, they pointed out that predictive analytics's efficient use in supply chain decision-making has been hampered by the absence of a clear causal relationship.

The goal of the project was to improve supply chain management decision-making by using a predictive analytics framework. The researchers created an integrated framework for big data analytics, evaluated supply chain management systems in Kenyan power businesses, and evaluated current predictive analytics solutions. The study was carried out in the Western Region of Kenya Power and Lighting Company using a Design Science research methodology. The application of predictive analytics to improve inventory control and production scheduling in the US manufacturing sector was investigated by Bashar et al. in (2024) [18]. Businesses can estimate demand, optimize production, and save inventory costs by evaluating both historical and current data. In order to emphasize the advantages, difficulties, and prerequisites for successful implementation, the study examined case studies, literature, and empirical data. Resource constraints, organizational opposition, and data quality are major obstacles. The results highlight how data-driven decision-making using predictive analytic boosts productivity, shortens lead times, and gives businesses a competitive edge. In this research paper, we use a fixed work force modal for aggregate production planning in the textile industry, where hiring and firing of workers are not allowed. We develop a LP modal to examine this problem and apply three Python's optimization packages (SciPy, PuLP, CVXPY) to assess their solvers proficiency for prospective utilization.

2. Problem Discription and Mathematical Framework

The global market is observing escalating competition as businesses worldwide try to capture market share, improve and adapt to changing customer demands. In the textile industry, grow- ing competition requires efficient APP to remain leading. The goal is to develop an optimal aggregate production plan that meet production objectives, minimizing costs for suppliers, labour(regular and overtime) assuming maximum profit in spite of prospective production challenges. In this study, we assume a constant fixed workforce problem(without productively loss) i.e. hiring and firing of labour is disallowed and accorders are also not allowed.

2.1. Notation and Parameters

γ_t: Production cost per unit in time period t (excluding work expenses)

β_t: Inventory holding cost per unit in period t

ρ_t: Per hour cost in time period t(Regular Labour)

ω_t: Per hour cost in time period t(Overtime Labour)
d_t: Order forecast in time t
μ: Man-hours required to produce one unit
$\bar{U}_t$: Maximum regular labour man-hours in period t
$\bar{V}_t$: Maximum overtime labour man-hours in period t
i_0: Initial stock position
H: Number of planning periods (time horizon.

2.2. Mathematical Modal: Fixed Work Force Decision Variable:

q_t: Quantity of units produced in time t
s_t: Inventory level in time t (at last)
u_t: Routine working man-hours utilized in time t
v_t: Overtime man-hours utilized in time t

Objective Function

$$\min \sum_{t=1}^{H} (\gamma tqt + \beta tst + \rho tut + \omega tvt) \quad (1)$$

The main objective is to maximize the profit and minimize the overall cost, which includes production costs, inventory holding costs, and labor costs (regular and overtime).

Model Constraints

$$q_t + s_{t-1} - s_t = d_t, \quad \forall t = 1, 2, \ldots, H \quad \text{(Inventory balancing constraint)} \quad (2)$$

$$\mu q_t = u_t + v_t, \quad \forall t = 1, 2, \ldots, H \quad \text{(Labour time requirement)} \quad (3)$$

$$0 \le u_t \le \bar{U}_t, \quad \forall t \quad \text{(Regular labor Supply)} \quad (4)$$

$$0 \le v_t \le \bar{V}_t, \quad \forall t \quad \text{(Overtime Supply)} \quad (5)$$

The Inventory Balancing constraint (2) ensure that need should be fulfill in period t. The constraint (3) ensure that total production time needs to be same as given in time period t. Regular and overtime labour supply is restricted by constraints (4) and (5).

3. Algorithm Approach for solving the problem

Three open-source Python solvers PuLP, SciPy, and CVXPY are used to perform the computational experiments on the numerical dataset shown in Table 1 [15]. Every simulation is run on a Windows 10 HP Elitebook with 4 GB of RAM using the Python Community Edition 3.12 domain. An Intel Core i5 processor with a base frequency of 2.50 GHz and a maximum turbo frequency of 2.71 GHz are among the technical characteristics. This layout proved sufficient to solve the fixed workforce aggregate production planning model and effectively compare solver performances.

TABLE 1. Data Set

Parameter	Jan	Feb	Mar	Apr	May	Jun
Demand (units)	110	110	120	210	160	90
Production cost per unit	8	7	7	9	6	9
Carrying cost per unit	2	5	5	3	4	3
Regular labor cost	18	17	19	17	15	17
Overtime labor cost	23.5	24.5	28	28	23.5	23.5
Available man-hours (Regular)	130	140	150	160	110	110
Available man-hours (Overtime)	40	50	40	40	40	40

3.1. Python Code

PuLp Code

The PuLP code which is used in this research is based on the coding [15], which is a key reference for applying LP to APP problems.

```
# Import PuLP library with all essential files
from pulp import *

# List (time period)
t = [0, 1, 2, 3, 4, 5, 6]

# Assignment of variables
qt = LpVariable.dicts("Amount_Manufactured ", t, 0)
st = LpVariable.dicts("Stock.Level", t, 0)
ut = LpVariable.dicts("Regular.Labor_Used", t, 0)
vt = LpVariable.dicts("Overtime .Labor_Used ", t, 0)

# Dataset
D = {1: 110, 2: 110, 3: 120, 4: 210, 5: 160, 6: 90}   # Demand(units)
UPC = {1: 8, 2: 7, 3: 7, 4: 9, 5: 6, 6: 9}                  # production
    expenditure
UCC = {1: 2, 2: 5, 3: 5, 4: 3, 5: 4, 6: 3}                  # Per item storage
    expense
RLC = {1: 18, 2: 17, 3: 19, 4: 17, 5: 15, 6: 17}           # labor Cost(
    conventional hours)
OLC = {1: 23.5, 2: 24.5, 3: 28, 4: 28, 5: 23.5, 6: 23.5} # labor Cost(work
    beyond scheduled hours)
RMH = {1: 130, 2: 140, 3: 150, 4: 160, 5: 110, 6: 110}    # total Man-hours(
    Regular)
OMH = {1: 40, 2: 50, 3: 50, 4: 40, 5: 40, 6: 40}          # total Man-hours(
    Overtime)
```

```
# Define the LP problem \\
resq = LpProblem("APP_Model", LpMinimize)\\
# Function to be optimized \\
resq += ( lpSum(UPC[k] * qt[k] for k in t[1:]) +
     lpSum(UCC[k] * st[k] for k in t[1:]) +
     lpSum(RLC[k] * ut[k] for k in t[1:]) +
     lpSum(OLC[k] * vt[k] for k in t[1:])
)

# Constraints to be used \\
# Initial Inventory\\
resq  +=st[0]==  4   # December stock Status

# Balancing Constraints
for k in t[1:]:
     resq  += (qt[k] + st[k-1] - st[k] == D[k])

# Required time for production
for k in t[1:]:
     resq  += (qt[k] - 1 * (ut[k] + vt[k]) == 0)

# Required (regular) time
for k in t[1:]:
     resq  += (ut[k] <= RMH[k])

# Required (overtime) time
for k in t[1:]:
     resq += (vt[k] <= OMH[k]

# Simulated Result
resq.solve()
print("Solution ={LpStatus[resq.status]}")

# Result
for w in resq.variables():
     if w.varValue > 0:
          print(w.name, "=", w.varValue)

# Ideal solution of APP problem
print("Final.Cost=", value(resq.objective))
```

SciPy code

Same data given in table 1 is used to perform APP problem but here SciPy is the solver.

```
import numpy as np
from scipy.optimize import linprog

# Time period
t = [1, 2, 3, 4, 5, 6]

D = np.array([110, 110, 120, 210, 160, 90]) #demand
UPC = np.array([8, 7, 7, 9, 6, 9])   # Unit Production Cost
UCC = np.array([2, 5, 5, 3, 4, 3])   # Unit Carrying Cost
RLC = np.array([18, 17, 19, 17, 15, 17])   # Regular Labor Cost
OLC = np.array([23.5, 24.5, 28, 28, 23.5, 23.5])   # Overtime Labor Cost
RMH = np.array([130, 140, 150, 160, 110, 110])    # Available Regular Man-Hours
OMH = np.array([40, 50, 50, 40, 40, 40])   # Available Overtime Man-Hours
i = 4     #initial inventory

# Decision Variables: q1 to q6 | s1 to s6 | u1 to u6 | v1 to v6
n = len(t)
q_start, s_start, u_start, v_start = 0, n, 2 * n, 3 * n

# Objective Function Coefficients
c = np.concatenate([UPC, UCC, RLC, OLC])

# Equality Constraints
B_eq = []
a_eq = []

# Inventory Balance Constraints: q_t - s_t + s_(t-1) = D_t
for k in range(n):
    row = np.zeros(4 * n)
    row[q_start + k] = 1   # Production quantity
    row[s_start + k] = -1 # Inventory level at t
    if k > 0:
        row[s_start + k - 1] = 1     # Inventory level at t-1
        a_eq.append(D[k])
    else:
        a_eq.append(D[k] - i) # Adjust for initial inventory level
    B_eq.append(row)

# Production of (Regular and Overtime) Constraints
for k in range(n):
    row = np.zeros(4 * n)
    row[q_start + k] = 1
    row[u_start + k] = -1
    row[v_start + k] = -1
    B_eq.append(row)
    a_eq.append(0)
```

```
# Convertion NumPy arrays arrangment
B_eq = np.array(B_eq)
a_eq = np.array(a_eq)

# Inequality Constraints
B_ub = []
a_ub = []

# Regular Man Hour Limits
for k in range(n):
    row = np.zeros(4 * n)
    row[u_start + k] = 1
    B_ub.append(row)
    a_ub.append(RMH[k])

# Overtime Man Hour Limits
for k in range(n):
    row = np.zeros(4 * n)
    row[v_start + k] = 1
    B_ub.append(row)
    a_ub.append(OMH[k])

# Convertion to NumPy arrays
B_ub = np.array(B_ub)
a_ub = np.array(a_ub)

# Bounds:decision variables >= 0
bounds = [(0, None)] * (4 * n)

# Solution using SciPy's linprog (HiGHS solver)
result = linprog(
    c,
    B_eq=B_eq,
    a_eq=a_eq,
    B_ub=B_ub,
    a_ub=a_ub,
    bounds=bounds,
    method='highs'
)

# Display final Results
if result.success:
    print(" Solution: Optimal")
    print("Decision Variables:")
    for k in range(n):
        print(f" Production at t={k+1}: {result.x[q_start + k]:.4f}")
    for k in range (n):
        print(f" Inventory at t={k+1}: {result.x[s_start + k]:.4f}")
    for k in range (n):
        print(f" Regular Labor  at t={k+1}: {result.x[u_start + k]:.4f}")
    for k in range (n):
        print(f" Overtime Labor  at t={k+1}: {result.x[v_start + k]:.4f}")
    print(f"\n Total  Cost: {result.fun:,.4f}")
else:
    print(" No Optimal Solution")
```

CVXPY Code

Similarly for CVXPY the following code is given.

```
import cvxpy as cp
import numpy as np

# Time period
t = [1, 2, 3, 4, 5, 6]

D = [110 ,110 , 120, 210, 160, 90]
UPC = [8, 7, 7, 9, 6, 9]        # Unit Production Cost
UCC = [2, 5, 5, 3, 4, 3]        # Unit Carrying Cost
RLC = [18, 17, 19, 17, 15, 17]   # Regular Labor Cost
OLC = [23.5, 24.5, 28, 28, 23.5, 23.5]  # Overtime Labor Cost
RMH = [130, 140, 150, 160, 110, 110]  # Available Regular Man Hours
OMH = [40, 50, 50, 40, 40, 40]        # Available Overtime Man Hours
i = 4    #initial inventory

# Number of time periods
n = len(t)

# Decision Variables
qt = cp.Variable(n, nonneg=True)# produced quantity
st = cp.Variable(n, nonneg=True)# inventory level
ut = cp.Variable(n, nonneg=True)   # Regular Labor Used
vt = cp.Variable(n, nonneg=True)   # Overtime Labor Used

# Objective Function for Minimize Total Cost
total_cost = (
    cp.sum(cp.multiply(UPC, qt)) +      # Production Cost
    cp.sum(cp.multiply(UCC, st)) +      # Carrying Cost
    cp.sum(cp.multiply(RLC, ut)) +      # Regular Labor Cost
    cp.sum(cp.multiply(OLC, vt))        # Overtime Labor Cost
)
```

```
# Constraints
C = []

# Initial Inventory Constraint
C.append(st[0] == i + qt[0] - D[0])

# Inventory Balancing Constraints:
for k in range(1, n):
    C.append(qt[k] + st[k-1] - st[i] == D[i])

# Production of (Regular and Overtime) Constraints:
for k in range(n):
    C.append(qt[i] == ut[k] + vt[k])

# Regular Man Hours Limit
for k in range(n):
    C.append(ut[k] <= RMH[k])

# Overtime Man Hours Limit
for k in range(n):
    C.append(vt[k] <= OMH[k])

# Define and Solution of Optimization Problem
O = cp.Minimize(total_cost) #objective
problem = cp.Problem(O,C)
problem.solve(solver=cp.SCS)  # By using SCS solver

# Print Results
if problem.status == cp.OPTIMAL or problem.status == cp.OPTIMAL_INACCURATE:
    print("Solution Status = Optimal")
    print(" Decision Variables:")
    for k in range(n):
        print(f"Quantity_Produced_{k+1} = {qt.value[k]:.4f}")
    for k in range(n):
        print(f"Inventory_{k+1} = {st.value[k]:.4f}")
    for k in range(n):
        print(f"Regular_Labor_Used_{k+1} = {ut.value[k]:.4f}")
    for k in range(n):
        print(f"Overtime_Labor_Used_{k+1} = {vt.value[k]:.4f}")
    print(f"\nTotal Production Plan Cost = {problem.value:,.4f}")
else:
    print("No Optimal Solution")
```

4. Results and Discussion

The performance of three open source solvers PuLP, SciPy, and CVXPY on the fixed work-force APP model in which hiring and firing are no allowed was examined in this research. The execution times differ even though every solver reaches the same optimal cost of 20486. With an execution time of only 0.02693 seconds, SciPy appear as the fastest solver, followed by CVXPY at 0.06882 seconds. With processing time of 0.09974 seconds, PuLP emerged as the slowest solver to solve the problem despite it produced the same optimal cost. These findings suggest that SciPy is the more effective in respect of computing speed for this specific model, even though the solvers are equally successful to attain the best optimal solution.

TABLE 2. Comparison of Solver Performance for Fixed Workforce APP Model

Solver	Optimal Cost	Execution Time (seconds)
PuLP (CBC)	20486	0.09974s
SciPy	20486	0.02693s
CVXPY (SCS)	20486	0.06882s

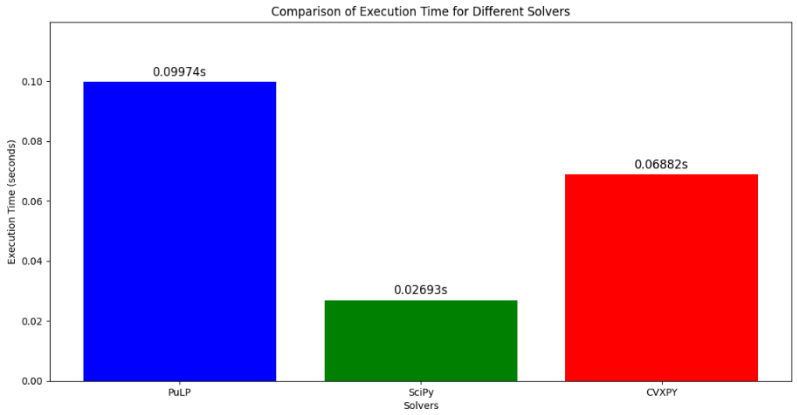

FIGURE 1. Comparison of Execution Time for SciPy, PuLP and CVXPY

Using a bar graph for visual transparency, Figure 1 compares the execution times of the three solvers SciPy, CVXPY, and PuLP. By solving the fixed workforce APP model in a far shorter amount of time than the other two solvers, this visualization tells SciPy's better computational effectiveness. In real-world production planning plot, this kind of analysis is necessary for choosing the fastest answer.

In figure 2, Line graph represents the execution time of SciPy, CVXPY, and PuLP packages. Each data point is labeled with its respective execution time value and joined by a visible

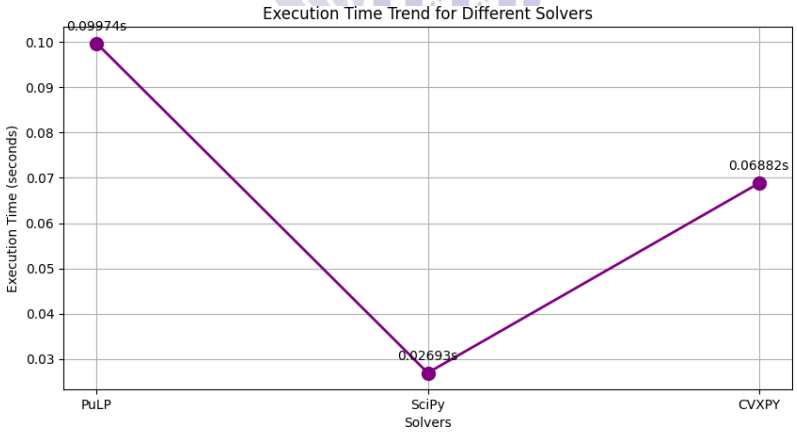

FIGURE 2. Execution Time Trend for SciPy, PuLP and CVXPY

trend line. This figure shows the corresponding speed differences among the solvers in solving the fixed workforce APP model.

5. Conclusion

It is evident from the research computational findings and critical examination that the computational proficiency of Aggregate Production Planning (APP) model is profoundly influenced based on the chosen optimization solvers . In spite of producing the same optimal cost, the execution times of the three Python solvers PuLP, SciPy, and CVXPY varied greatly, which was expressive of their appropriate computational

techniques. SciPy has the fastest execution time among the tested solvers, followed by CVXPY, while PuLP has the slowest execution time. According to the study's results, SciPy is the most computationally impressive option for determining fixed workforce LP models. Simultaneously, CVXPY and PuLP prove to be helpful because to their adjustability and simplicity in model formulation, particularly in circumstances where solver clarity or more immense modeling capabilities are needed. The results allow researchers and professionals with useful information for selecting ideal solvers for linear optimization tasks, mostly with respect to production and operations management.

Competing Interests

The authors declare that they have no competing interests.

Funding Information

The author(s) received no financial support for the research, authorship, and/or publication of this article.

Data Availability Statement

All model codes generated during the current study are available from the corresponding author on reasonable request.

Research Involving Human and /or Animal

This study did not involve human participants, human data, human tissue, or animals. The work is a computational/theoretical study of optimization solvers and aggregate production planning models and therefore did not require ethical approval.

Informed Consent

Not Applicable.

REFERENCES

[1] Diwekar UM. Introduction to applied optimization. Springer Nature; 2020 Oct 29.

[2] Kou G, Liu X, Peng Y, Shi Y, Wise M, Xu W. Multiple criteria linear programming approach to data mining: Models, algorithm designs and software development. Optimization Methods and Software. 2003 Aug 1;18(4):453-73.

[3] Vo S, Woodruff DL. Introduction to computational optimization models for production planning in a supply chain. Springer Science & Business Media; 2006 Mar 20.

[4] Silva Filho OS, Cezarino W, Ratto J. Aggregate production planning: Modeling and solution via Excel spreadsheet and solver. IFAC Proceedings Volumes. 2010 Jan 1;43(17):89-94.

[5] Souza GC. Supply chain analytics. Business Horizons. 2014 Sep 1;57(5):595-605.

[6] Gholamian N, Mahdavi I, Tavakkoli-Moghaddam R. Multi-objective multi-product multi-site aggregate production planning in a supply chain under uncertainty: fuzzy multi-objective optimisation. International Journal of Computer Integrated Manufacturing. 2016 Feb 1;29(2):149-65.

[7] Anand Jayakumar A, Krishnaraj C, Nachimuthu A. Aggregate production planning: Mixed strategy. Pak. J. Biotechnol. 2017;14(3):487-90.

[8] Biazzi JL. Aggregate planning for probabilistic demand with internal and external storage. Journal of Operations & Supply Chain Management (JOSCM). 2018 Jun 15;11(1):37-52.

[9] Demirel E, zelkan EC, Lim C. Aggregate planning with flexibility requirements profile. International Journal of Production Economics. 2018 Aug 1;202:45-58.

[10] Cheraghalikhani A, Khoshalhan F, Mokhtari H. Aggregate production planning: A literature review and future research directions. International Journal of Industrial Engineering Computations. 2019 Apr 1;10(2):309-30.

[11] Jamalnia A, Soukhakian MA. A hybrid fuzzy goal programming approach with different goal priorities to aggregate production planning. Computers & Industrial Engineering. 2009 May 1;56(4):1474-86.

[12] Parganiha K. Linear programming with python and pulp. International Journal of Industrial Engineering Research and Development. 2018 Sep;9(3):01-8.

[13] Sodero A, Jin YH, Barratt M. The social process of Big Data and predictive analytics use for logistic & supply chain management. International Journal of Physical Distribution & Logistics Management. 2019 Aug 30;49(7):706-26.

[14] Shete K, Shinde P, Tanpure P, Gunjal A. Analytics for Manufacturing Decisions in Supply Chain Management. International Journal for Research in Applied Science & Engineering Technology. 2019;7(V):861-7.

[15] Rehman HU, Ahmad A, Ali Z, Baig SA, Manzoor U. Optimization of Aggregate Production Planning Problems with and without Productivity Loss using Python Pulp Package. Management and Production Engineering Review. 2021;12(4).

[16] ALRIDHA AH, Salman AM, Al-Jilawi AS. Numerical Optimization Approach for Solving Production Planning Problem Using Python language. Central Asian Journal of Mathematical Theory and Computer Sciences. 2022;3(6):6-15.

[17] Patrick WM, Anselemo PI, Ronoh R, Mbugua S. Impact of predictive analytics of big data in supply chain management on decision-making. Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol. 2022 Jul;3307:225-38.

[18] Al Bashar M, Taher A, Johura FT. Utilizing Predictive Analytics for Enhanced Production Planning and Inventory Control in the US Manufacturing Sector. International Research Journal of Modernization in Engineering Technology and Science. 2024 Jun;6:8332-9.