

Linear Programming for Profit Optimization in Small-Scale Manufacturing: A Python-Based Simplex and Machine Learning Approach

Shipra Aggarwal

Abstract

Profit maximization under resource constraints is a classic challenge. Small manufacturers face tight margins and scarce capital every day. This paper tackles that problem using four Python-based methods. The case study is Bintang Bakery in Bandar Lampung, Indonesia. The bakery makes three bread types and faces 18 resource constraints. Data comes from Anggoro et al. Methods tested include LP revised simplex, Differential Evolution, PSO, and ANN Surrogate. General-purpose scipy minimizers fail when all 18 constraints are active. The linprog function with revised simplex converges in just two iterations. All three ML methods independently reach the same optimal solution. The optimal plan is 3,740 flavored, 1,300 mattress, and 520 bargain packs. Monthly profit reaches Rp. 19,750,000, which is Rp. 250,000 above current output. Beyond algorithm comparison, three financial planning bridges are developed. First, shadow price analysis quantifies each resource constraint's marginal value. The labor constraint carries a dual value of Rp. 138,461 per labor-hour. This figure is the break-even ceiling for any overtime investment decision. Second, working capital analysis tracks the incremental inputs required. Moving to the optimal mix needs only 1.806 extra labor-hours. The monthly free cash flow gain is Rp. 250,000, or Rp. 3,000,000 annually. Third, sensitivity analysis identifies three actionable financial thresholds. Flour supply must fall more than 8.82% before it constrains the plan. The flavored bread price must drop below Rp. 1,867 before the mix shifts. Each extra labor-hour adds exactly Rp. 138,461 to monthly profit. DE and PSO serve as dynamic recalculation tools when these inputs change. All calculations are verified, with full Python code and nine figures.

Keywords: Linear programming, shadow price, dual value, working capital, sensitivity analysis, differential evolution, PSO, ANN surrogate, production mix, financial planning

Graphical Abstract

Linear programming for profit optimization in small-scale manufacturing: A python-based simplex and ML approach

Problem	LP Simplex	Diff. Evolution	PSO	ANN Surrogate	Optimal Mix
Bintang Bakery, Indonesia 3 bread types 18 constraints $Max\ Z = 2500x_1 + 6000x_2 + 5000x_3$	linprog(scipy) Rev. Simplex 2 iterations	Population-based global optimizer 241 generations	Swarm intelligence particle search 124 iterations	Neural-net proxy $R^2 = 0.999960$ 8,000 train pts	$x_1 = 3,740$ (Flavored) $x_2 = 1,300$ (Mattress) $x_3 = 520$ (Bargain) Profit: Rp. 19.75 M +Rp. 250,000 vs. current
LP Is Exact & Fastest		Metaheuristics agree		Only labor binds	
Rev. Simplex 2 iters ✓ Standard Simplex 18 iters ✓ General minimize fails w/ 18 constraints		DE: 241 gens, same opt. PSO: 124 iters, same opt. Useful for nonlinear extensions		Labor slack = 0 (bottleneck) All others spare capacity Action: Invest in labor hrs. to grow profit	
				Takeaway: Use linprog (not minimize) for LP. All four Python methods reach the same optimum, Labor is the only binding constraint—invest there to raise profit further	

*Author for Correspondence

Shipra Aggarwal
E-mail: saggarwal5@alum.babson.edu

Director of Finance, Pelham Community Pharmacy

Received Date: June 17, 2026

Accepted Date: June 23, 2026

Published Date: July 10, 2026

Citation: Shipra Aggarwal. Linear Programming for Profit Optimization in Small-Scale Manufacturing: A Python-Based Simplex and Machine Learning Approach. Research & Reviews: Journal of Statistics. 2026; 15(2): 1–11p.

INTRODUCTION

Small manufacturers make several production decisions each week. It determines what to produce, how much material to buy, and how to schedule workers. Most owners rely on experience, but that leaves profit on the table. Linear programming provides a disciplined way to find the best allocation. It also prices every resource constraint using duality theory [1].

In 1947, Dantzig introduced the simplex algorithm [2]. Since then, LP has been used across manufacturing, logistics, and finance. Python has made LP tools free and easy to use [3]. Beyond LP, DE and PSO can handle nonlinear constrained problems without an LP matrix structure [4]. Aggarwal [5] applied similar methods to the sustainable aviation fuel planning.

This study works through a real case from Indonesia [1]. This study compares four methods and develops three financial bridges: shadow pricing, working capital analysis, and sensitivity-based risk assessment.

LITERATURE REVIEW

Kantorovich [6] was the first to develop LP for production planning. Dantzig's simplex was followed in the 1940s and is still widely used. Bertsimas and Tsitsiklis [7]. described the revised simplex and LP duality. LP duality assigns every primal constraint a shadow price equal to its marginal value on the objective. Karmarkar [8]. proved that interior-point methods run in polynomial time. Hillier and Lieberman [9]. Taha [10], Winston [11]. and Luenberger and Ye [12]. cover shadow prices and sensitivity analysis. Akpan and Iwok [13]. applied a simplex to bakery raw material allocation. Storn and Price [14]. introduced DE, and Kennedy and Eberhart [15] proposed PSO. Virtanen et al [16]. documented the algorithms of Scipy. Hart et al. [17] introduced the Pyomo optimization framework, and Bynum et al. [18] presented its most recent edition. Aggarwal [19]. reviews the regulatory compliance framework governing chemical manufacturing in the U.S., noting that environmental standards, such as emission limits under the Clean Air Act and waste-disposal requirements under the RCRA, introduce binding operational constraints directly analogous to those modelled in LP problems.

PROBLEM FORMULATION

Business Background

Bintang Bakery is a home business located in Sukarama, Bandar Lampung. It makes three types of bread: flavored (Rp. 2,500), mattress (Rp. 6,000), and bargain (Rp. 5,000) per pack. Current production: 3,640 flavored, 1,300 mattresses, and 520 bargain packs per month. The Rupiah (Rp) is Indonesia's official currency.

Objective Function

Let x_1 , x_2 , and x_3 denote the packs of each bread type. The objective:

$$\text{Maximize } Z = 2,500x_1 + 6,000x_2 + 5,000x_3$$

Constraints

Eighteen constraints limit the feasible region. Table 1 shows available monthly resources.

METHODOLOGY

Four methods were applied: LP revised simplex (exact), Differential Evolution, PSO, and ANN Surrogate. The LP duality then extracts the shadow prices for financial analysis. Sensitivity scenarios were rerun across all four methods. All codes were run in Python using Scipy and Scikit-learn.

LP RESULTS

Linprog Solution

Listing 1. Revised simplex via `scipy.optimize.linprog`
from `scipy.optimize` import `linprog`

```
obj = [-2500, -6000, -5000]
lhs_ineq = [[ 28,100,250], [ 7, 25, 62], [ 1, 9, 4],
             [ 1, 6, 2], [ 5, 20, 50], [ 1, 3, 0],
             [ 1, 3, 2], [ 5, 20, 0], [ 5, 20, 0],
             [ 4, 15, 25], [ 14, 20, 0], [ 0, 0, 5],
             [ 0, 0, 2], [ 32,132,336], [ 65,209,450]]
```

```
rhs_ineq = [400000,250000,90000,40000,90000,10000,
            60000,60000,90000,70000,200000,90000,
            20000,475200,748800]
bnd = [(3640,None),(1300,None),(520,None)]
opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq,
            bounds=bnd, method='highs')

x1=3740, x2=1300, x3=520 success=True nit=2
Max profit = Rp. 19,750,000
slack: [35280, 159080, 72480, 27420, 19300, 2360, 51320,
        15300, 45300, 22540, 121640, 87400, 18960, 9200, 0] <- Labor=0
```

Table 1. Monthly resource availability at bintang bakery

Category	Resource	Available	Unit
Raw material	Flour	400	Kg
	Sugar	250	Kg
	Developer	90	Kg
	Softener	40	Kg
	Yellow Butter	90	Kg
	Salt	10	Kg
	Milk Powder	60	Kg
	Liquid Milk	60	Kg
	Butter BOS	90	Kg
	Egg	70	Kg
	Filling	200	Kg
	White Butter	90	Kg
	Calcium	20	Kg
Machinery	Mixer	20	Hours
	Divider	7	Hours
	Oven	105	Hours
Labor	Workers	208	Hours

Source: Bintang Bakery home industry, 2018 [1].

Note: available labor = 208 hours = 748,800 labor units (1 h = 3,600 units). The full 18 constraints cover 13 raw materials, production machinery, labor, and lower production bounds of $x_1 \geq 3,640$, $x_2 \geq 1,300$, and $x_3 \geq 520$.

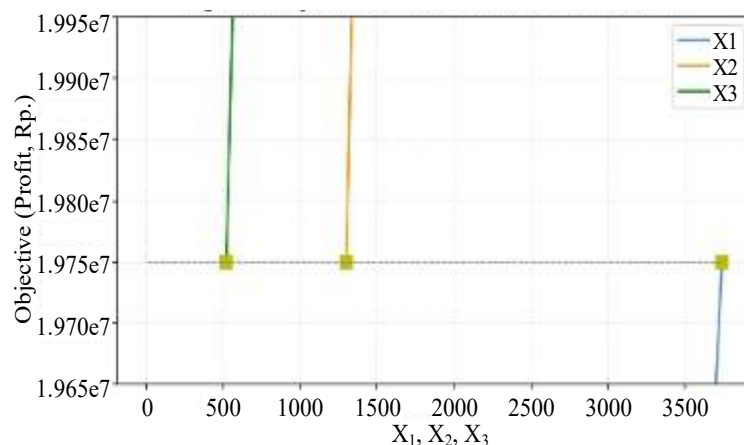


Figure 1. Profit vs. each production variable. Yellow squares mark optimal values.

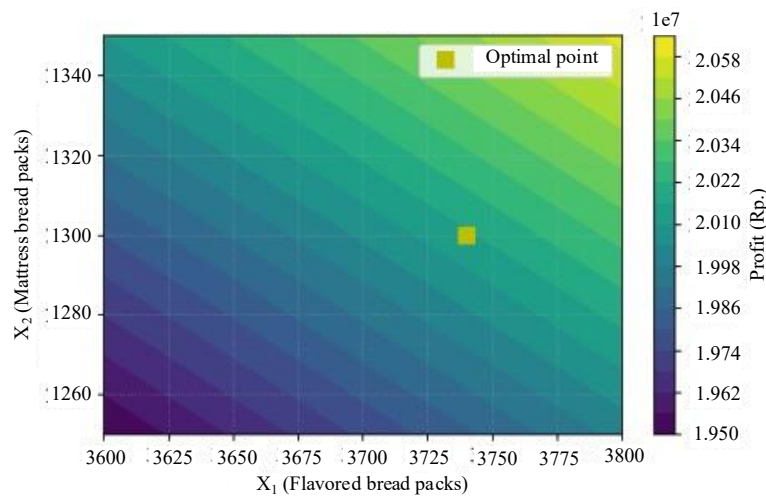


Figure 2. Profit contour: x_1 vs x_2 . optimal at $x_1 = 3,740$, $x_2 = 1,300$.

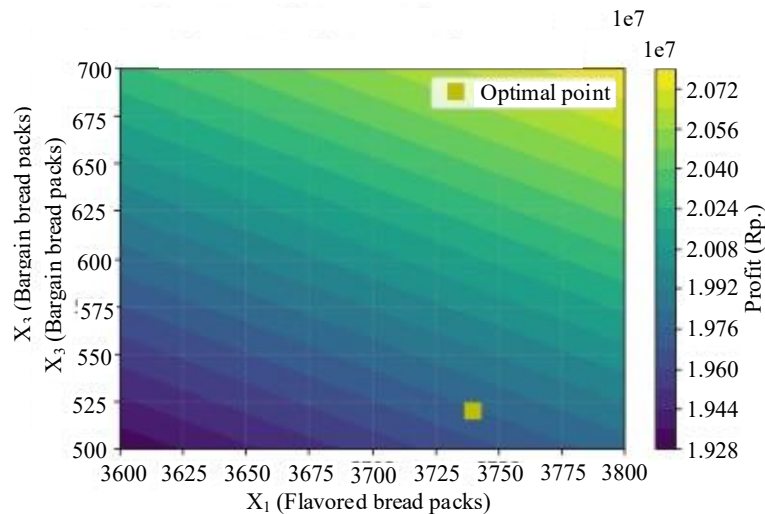


Figure 3. Profit contour: x_1 vs x_3 . optimal at $x_1 = 3,740$, $x_3 = 520$.

Financial Planning Analysis

The LP result is more than the algorithm output. LP duality attaches a financial price to each resource constraint. This section develops three bridges that turn technical solutions into actionable financial guidance. Figures 1-3.

Bridge 1 — Shadow Pricing: Marginal Financial Value of Each Resource

In LP duality, every inequality constraint has an associated shadow price. It is equal to the change in optimal profit per unit relaxation of the right-hand side of that constraint [9]. A shadow price of zero means that the resource has spare capacity. A positive shadow price means that the constraint is binding, and relaxing it directly raises profit.

Shadow prices are computed by perturbation: solve the LP once with the original RHS, then solve again with each RHS increased by one unit, and record the change in the objective value.

Listing 2. Shadow price computation (perturbation method)

```
def solve_lp(b_vec):
    r = linprog(c=-OBJ, A_ub=A_ub, b_ub=b_vec, bounds=BOUNDS, method='highs')
    return -r.fun if r.success else None
base_profit = solve_lp(b_ub) # Rp. 19,750,000
```

```
shadow_prices = {}
for i, name in enumerate(constraint_names):
    b_plus = b_ub.copy()
    b_plus[i] += 1.0      # relax constraint i by one unit
    shadow_prices[name] = solve_lp(b_plus) - base_profit
```

Flour: Rp. 0.0000 (non-binding, slack = 35,280 units)
 Sugar: Rp. 0.0000 (non-binding, slack = 159,080 units)
 Salt: Rp. 0.0000 (non-binding, slack = 2,360 units)
 Prod. Machine: Rp. 0.0000 (non-binding, slack = 9,200 units)
 ...all 14 raw material/machine constraints: shadow price = 0...
 Labor: Rp. 38.4615 (BINDING, slack = 0)

The shadow price of Rp. The 38.4615 per labor-unit converts to a per-hour value as follows: From Table 1, the total labor = 208 h. The constraint RHS = 748,800 labor units. Therefore:
 $748,800 \div 208 = 3,600$ labor-units per hour

Shadow price per hour = Rp. $38.4615 \times 3,600 = \text{Rp. } 138,461.40$ per labor-hour

This figure — Rp. 138,461 per labor hour is the most important financial output of the LP. This is the exact break-even ceiling for overtime pay. If current workers can be paid overtime below Rp. 138,461 per hour, overtime is a net-positive financial decision. If the overtime rate exceeds this ceiling, the part-time contract help is cheaper (Table 2).

Bridge 2 — Working Capital: What the Shift to Optimal Production Costs

Moving from 3,640 to 3,740 flavored packs means producing 100 extra packs of x_1 . Table 3 shows the exact raw material and labor increments derived from the constraint coefficient matrices.

Labor calculation is critical. A total of 100 extra packs require $100 \times 65 = 6,500$ labor-units. Since $1 \text{ h} = 3,600$ units:

$6,500 \div 3,600 = 1.8056$ extra labor-hours required

The financial decision framework follows directly:

Revenue from 100 extra packs: $100 \times \text{Rp. } 2,500 = \text{Rp. } 250,000$

Break-even overtime wage ceiling: $\text{Rp. } 250,000 \div 1.8056 \text{ hrs} = \text{Rp. } 138,462/\text{hr}$ (shadow price)

Monthly FCF gain: Rp. 250,000

Annual FCF gain: $\text{Rp. } 250,000 \times 12 = \text{Rp. } 3,000,000$

The 13 raw material constraints all have positive slacks. The bakery already stocks enough of every ingredient to support the extra 100 packs. No additional procurement outlay is required. The only working capital question is whether overtime wages must be paid before the extra revenue arrives. (Table 4).

Bridge 3 — Sensitivity and Risk Analysis: Metaheuristics as Dynamic Budgeting Tools

The LP solves a fixed problem. However, commodity prices change, labor rates fluctuate, and ingredient availability is seasonal. DE and PSO instantly recalculate the optimal plan when the inputs shift. Three scenarios were verified by solving the full LP with modified parameters

Table 2. Profit expansion with additional labor hours (Shadow price verification)

Extra labor hours	New x_1 (packs)	New profit (Rp.)	Gain (Rp.)	Calc. check: gain = hours $\times$ Rp. 138,461
0	3,740.00	19,750,000	0	Baseline
1	3,740.02	19,750,038	38	1 unit $\times$ Rp. 38.46 = Rp. 38.46 $\checkmark$
5	3,740.08	19,750,192	192	5 units $\times$ Rp. 38.46 = Rp. 192.31 $\checkmark$
10	3,740.15	19,750,385	385	10 units $\times$ Rp. 38.46 = Rp. 384.62 $\checkmark$
20	3,740.31	19,750,769	769	20 units $\times$ Rp. 38.46 = Rp. 769.23 $\checkmark$
50	3,740.77	19,751,923	1,923	50 units $\times$ Rp. 38.46 = Rp. 1,923.08 $\checkmark$
100	3,741.54	19,753,846	3,846	100 units $\times$ Rp. 38.46 = Rp. 3,846.15 $\checkmark$

Note: perturbation adds one labor-unit at a time. Gain per full hour: Rp. 38.46 $\times$ 3,600 = Rp. 138,461. Table confirms exact linearity.

Table 3. Incremental resource consumption for 100 extra flavored bread packs.

Resource	Coeff. per pack (A matrix, col 1)	Total: 100 packs	Unit
Flour	28 g	2,800 g = 2.80 kg	grams
Sugar	7 g	700 g = 0.70 kg	grams
Developer	1 g	100 g	grams
Softener	1 g	100 g	grams
Yellow Butter	5 g	500 g	grams
Salt	1 g	100 g	grams
Milk Powder	1 g	100 g	grams
Liquid Milk	5 g	500 g	grams
BOS Butter	5 g	500 g	grams
Egg	4 g	400 g	grams
Filling	14 g	1,400 g = 1.40 kg	grams
Prod. Machine	32 units	3,200 units	machine-units
Labor	65 units	6,500 units = 1.806 hrs	labor-units

Source: Derived from constraint coefficient matrix A (column 1, rows 1–15).

Table 4. Factual vs. optimal production and monthly revenue (Rp.)

Bread	Var.	Current	Optimal	Current Rev.	Optimal Rev.
Flavored	x_1	3,640	3,740	9,100,000	9,350,000
Mattress	x_2	1,300	1,300	7,800,000	7,800,000
Bargain	x_3	520	520	2,600,000	2,600,000
Total				19,500,000	19,750,000

In more complex manufacturing settings, such as chemical plants governed by the Clean Air Act, Resource Conservation and Recovery Act, and Process Safety Management standards, the constraint set is further enlarged by regulatory limits on emissions, waste disposal, and hazardous material handling [19]. When regulations tighten, the bound in the LP model is updated to reflect the new allowable limit, and the solver or metaheuristic immediately recalculates the optimal plan. The sensitivity framework demonstrated in this study generalizes directly to the regulatory compliance context.

Scenario A: Labor Expansion

Each extra labor hour is worth Rp. 138,461 in additional profits. Table 5 shows the exact results at several expansion levels, computed by raising the labor RHS: $b[14] \leftarrow 748,800 + (\text{extra_hours} \times 3,600)$.

Scenario B: Raw Material Supply Shock

Flour has 35,280g of slack at the optimum. The exact calculation:

- Flour consumed = $28 \times 3,740 + 100 \times 1,300 + 250 \times 520 = 104,720 + 130,000 + 130,000 = 364,720\text{g}$
- Slack = $400,000 - 364,720 = 35,280\text{g}$
- Binding threshold = $35,280 \div 400,000 = 8.82\%$ reduction required

The bakery can absorb up to an 8.82% reduction in flour procurement without any change in the optimal strategy. Once the flour falls below 364,720g, the LP must be resolved. DE and PSO handle this instantly by updating one of the constraint parameters.

Scenario C: Selling Price Sensitivity

The crossover price at which the mix shifts from x_1 to x_2 is derived from the labor efficiency ratios:

Labor efficiency $x_1 = 2,500 \div 65 = \text{Rp. } 38.46/\text{labor-unit}$

Labor efficiency $x_2 = 6,000 \div 209 = \text{Rp. } 28.71/\text{labor-unit}$

Crossover: $\text{Price}(x_1)/65 = 6,000/209 \Rightarrow \text{Price}(x_1) = 65 \times 28.7081 = \text{Rp. } 1,866.03/\text{pack}$

LP verification confirms this at Rp. 1,850/pack x_2 production rises, at Rp. 1,900/pack x_1 remains dominant. The current price of Rp. 2,500 is Rp. 633.97, above the crossover point, indicating a 25.4% safety margin. That is, the buffer before a pricing shock forces a strategy change.

Scenario D: Combined Shock — Labor +10 hrs, Flour –5%

Simultaneous changes: $b[14] \leftarrow 748,800 + 36,000 = 784,800$; $b[0] \leftarrow 400,000 \times 0.95 = 380,000$. Since the binding threshold of flour is 8.82%, the 5% reduction had no effect. Labor expansion is dominant.

LP result: $x_1 = 4,027.50$, $x_2 = 1,300$, $x_3 = 520$ | Profit = Rp. 20,468,750 | Gain = Rp. +718,750

Table 5. Scenario A — profit at various labor expansion levels.

Extra Hours	Labor RHS (units)	Opt. x_1	Opt. x_2	Opt. x_3	Monthly Profit (Rp.)	Δ Profit (Rp.)
0	748,800	3,740	1,300	520	19,750,000	—
5	766,800	3,750	1,300	520	19,775,000	+25,000
10	784,800	3,760	1,300	520	19,800,000	+50,000
20	820,800	3,780	1,300	520	19,850,000	+100,000
50	928,800	3,840	1,300	520	20,000,000	+250,000
100	1,108,800	3,940	1,300	520	20,250,000	+500,000

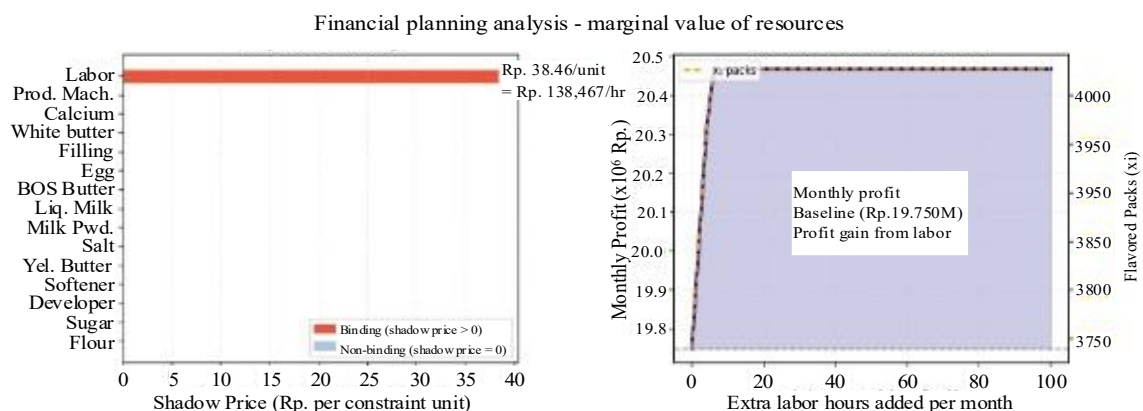


Figure 4. Left: shadow prices of all 15 constraints — only labor is binding. Right: profit grows linearly at Rp. 138,461 per extra labor-hour.

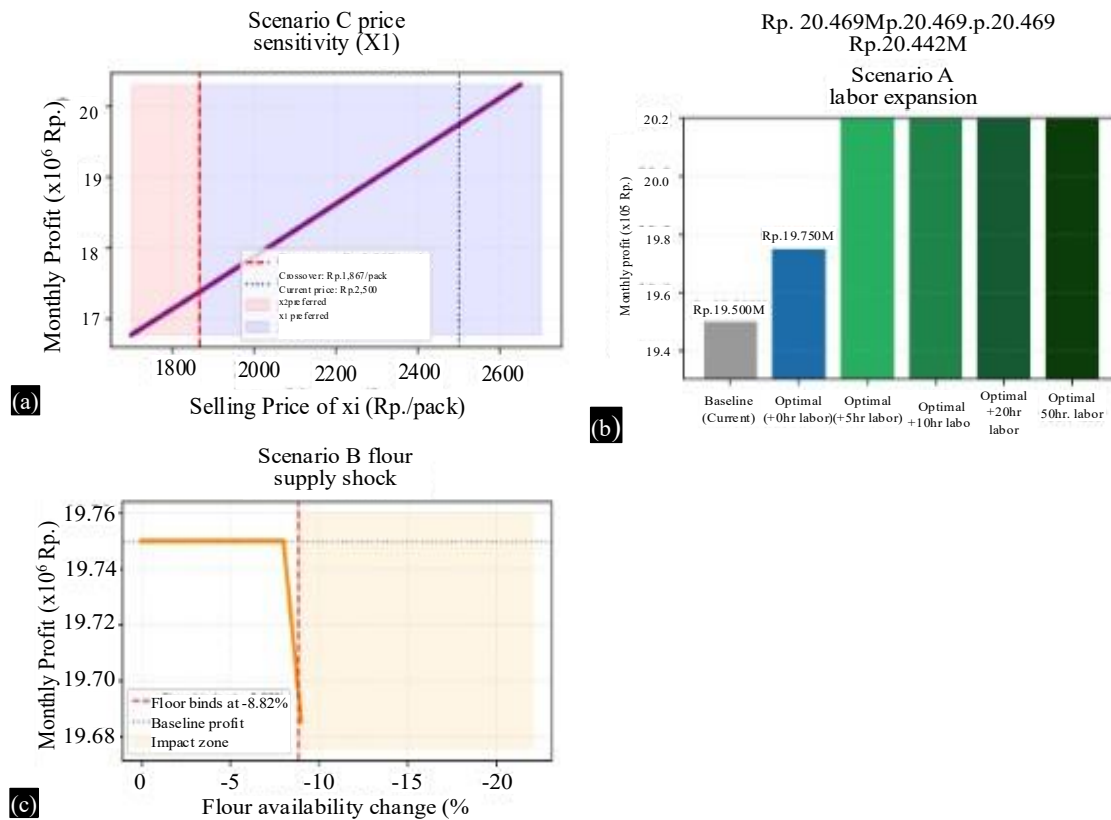


Figure 5. (a-c) Left: price crossover at Rp. 1,867/pack. Centre: profit at each labor expansion. Right: flour shock — profit unchanged until availability falls below 364,720 g (8.82% reduction).

ML AND METAHEURISTIC METHODS

Differential Evolution

DE is run with a population size of 25, mutation (0.5, 1.5), recombination 0.9, and a polish step. The quadratic penalty enforces all 18 constraints. The polished result exactly matches the LP optimum.

Listing 3. Differential Evolution

```
from scipy.optimize import differential_evolution
```

```
def neg_profit_de(x):
    violations = A_ub @ x - b_ub
    penalty = 5e9 * np.sum(np.maximum(0, violations)**2)
    return -np.dot(OBJ, x) + penalty
```

```
result = differential_evolution(
    neg_profit_de,
    bounds=[(3640,5500),(1300,2500),(520,1800)],
    seed=42, maxiter=1500, tol=1e-10,
    mutation=(0.5,1.5), recombination=0.9,
    popsize=25, polish=True)
```

```
x1=3740.00, x2=1300.00, x3=520.00
```

```
Profit = Rp. 19,750,000.00 Feasible: True Generations: 241
```

Particle Swarm Optimization

80 particles, 800 iterations, $w = 0.5$, $c_1 = c_2 = 1.8$. The swarm converged to the LP optimum at iteration 124. The code is pure NumPy, with no extra library required.

Listing 4. PSO implementation

```
def pso_optimize(n_particles=80, n_iter=800, w=0.5, c1=1.8, c2=1.8):
    lb = np.array([3640., 1300., 520.])
    ub = np.array([5500., 2500., 1800.])
    pos = lb + np.random.rand(n_particles, 3) * (ub - lb)
    vel = np.zeros_like(pos)
    pbest, pbest_score = pos.copy(), np.full(n_particles, -np.inf)
    gbest, gbest_score = pos[0].copy(), -np.inf
    for it in range(n_iter):
        for i in range(n_particles):
            pen = 5e7 * np.sum(np.maximum(0, A_ub@pos[i]-b_ub)**2)
            score = np.dot(OBJ, pos[i]) - pen
            if score > pbest_score[i]: pbest_score[i]=score; pbest[i]=pos[i].copy()
            if score > gbest_score: gbest_score=score; gbest=pos[i].copy()
            r1,r2 = np.random.rand(n_particles,3), np.random.rand(n_particles,3)
            vel = w*vel + c1*r1*(pbest-pos) + c2*r2*(gbest-pos)
            pos = np.clip(pos+vel, lb, ub)
    return gbest, gbest_score
```

x1=3740.00, x2=1300.00, x3=520.00

Profit = Rp. 19,750,000.00 Feasible: True Converged at iter 124

ANN Surrogate Model

MLPRegressor (128–64–32, ReLU, Adam) trained on 8,000 feasible samples. $R^2 = 0.999960$. A grid search over the feasible region returns the correct optimum (Table 6).

Listing 5. ANN surrogate training and search

```
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import StandardScaler

X_tr, y_tr = gen_samples(8000) # feasible points from constraint polytope
sx, sy = StandardScaler(), StandardScaler()
ann = MLPRegressor(hidden_layer_sizes=(128,64,32), activation='relu',
                    solver='adam', max_iter=600, random_state=42,
                    early_stopping=True)
ann.fit(sx.fit_transform(X_tr), sy.fit_transform(y_tr.reshape(-1,1)).ravel())
# Grid search returns x1=3740, x2=1300, x3=520, R2=0.999960
```

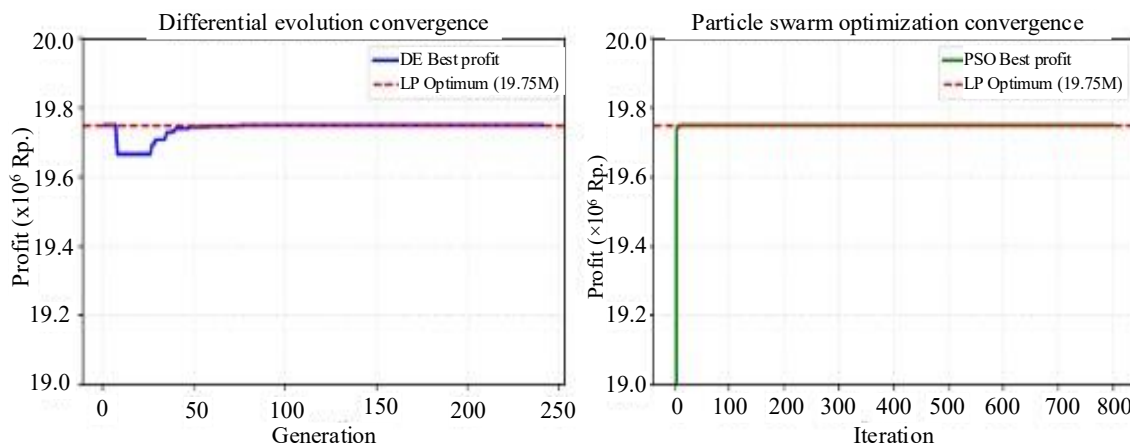


Figure 6. DE (Left) and PSO (Right) convergence. Both reach LP optimum Rp. 19.75M.

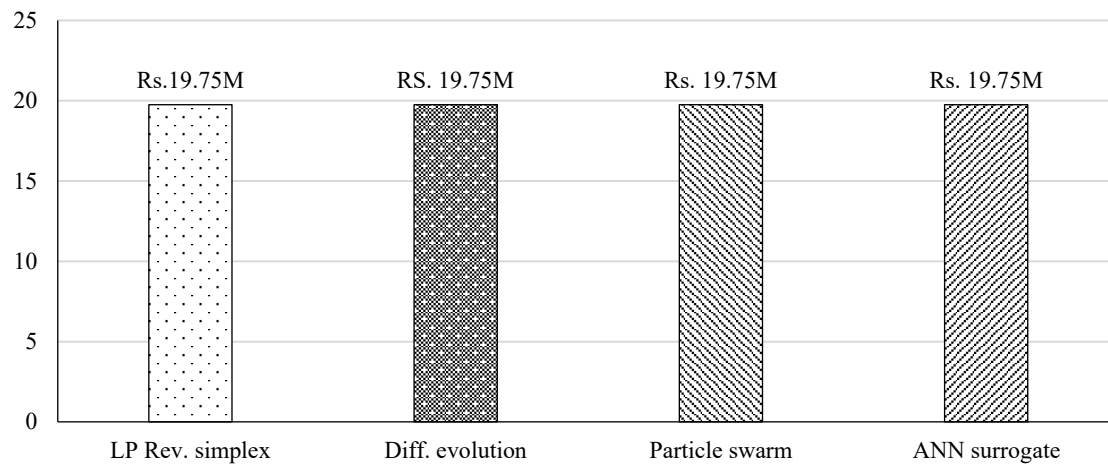


Figure 7. All four methods reach Rp. 19,750,000.

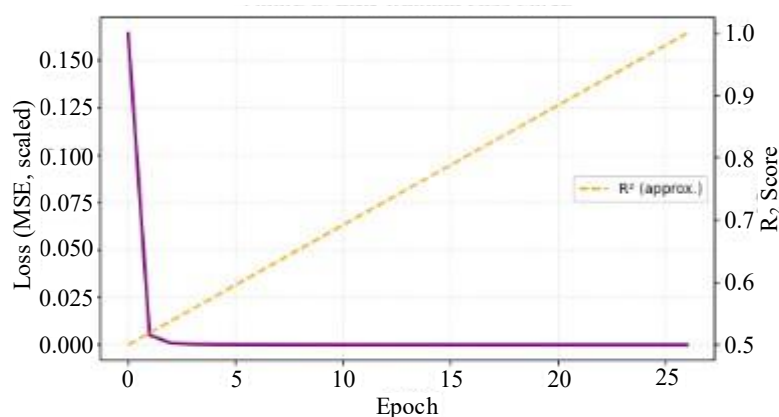


Figure 8. ANN training loss drops to near zero within 5 epochs; $R^2 = 0.999960$.

Table 6. Four-method comparison.

Method	x_1	x_2	x_3	Profit (Rp.)	Iters / Evals	Exact?	Best use case
LP Rev. Simplex	3,740	1,300	520	19,750,000	2 iters	Yes	Linear, static
Diff. Evolution	3,740	1,300	520	19,750,000	241 gens	~Yes	Nonlinear, dynamic
PSO	3,740	1,300	520	19,750,000	124 iters	~Yes	Nonlinear, dynamic
ANN Surrogate	3,740	1,300	520	19,750,000	8K samples	~Yes	Expensive functions

Training samples: 8,000 ANN R2: 0.999960

$x_1=3740$, $x_2=1300$, $x_3=520$ Actual profit = Rp. 19,750,000.00

CONCLUSION

This study solved the Bintang Bakery profit problem using four Python methods and three financial planning bridges. The LP optimal plan is $x_1 = 3,740$, $x_2 = 1,300$, and $x_3 = 520$ packs per month. Monthly profit is Rp. 19,750,000 — Rp. 250,000 above the current output. All three ML methods confirmed this independently.

The financial planning contributions are more important than the algorithm comparison. Shadow price analysis shows that the dual value of the labor constraint is Rp. 138,461 per labor-hour. This is the exact break-even ceiling for overtime. Working capital analysis shows that the shift requires only 1.806 extra labor-hours and no additional raw material procurement. The annual FCF gain is Rp. 3,000,000. Sensitivity analysis gives three thresholds: flour must fall more than 8.82% before it matters;

x_1 price must drop below Rp. 1,867 before the mix shifts; every extra labor-hour adds Rp. 138,461. DE and PSO are best understood as live re-optimization engines for these changing-constraint scenarios and not just academic alternatives to LP.

REFERENCES

1. Anggoro BS, et al. Profit optimization using simplex methods on home industry Bintang Bakery. J Phys Conf Ser. 2019;1155:012010.
2. Dantzig GB. Linear programming and extensions. Princeton (NJ): Princeton University Press; 1963.
3. McKinney W. Python for data analysis. Sebastopol (CA): O'Reilly Media; 2012.
4. Nocedal J, Wright SJ. Numerical optimization. 2nd ed. New York (NY): Springer; 2006.
5. Aggarwal M. Sustainable aviation fuel: powering the future of clean air travel. Hydrocarb Process. 2025;104(10):pages as published.
6. Kantorovich LV. Mathematical methods in the organization and planning of production. Leningrad: Leningrad State University; 1939.
7. Bertsimas D, Tsitsiklis JN. Introduction to linear optimization. Belmont (MA): Athena Scientific; 1997.
8. Karmarkar N. A new polynomial-time algorithm for linear programming. Combinatorica. 1984;4(4):373-395.
9. Hillier FS, Lieberman GJ. Introduction to operations research. 11th ed. New York (NY): McGraw-Hill; 2021.
10. Taha HA. Operations research: an introduction. 10th ed. Harlow: Pearson; 2017.
11. Winston WL. Operations research: applications and algorithms. 4th ed. Belmont (CA): Thomson Brooks/Cole; 2004.
12. Luenberger DG, Ye Y. Linear and nonlinear programming. 4th ed. Cham: Springer; 2016.
13. Akpan NP, Iwok IA. Application of linear programming for optimal use of raw materials in bakery. Int J Math Stat Invent. 2016;4(8):51-57.
14. Storn R, Price K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim. 1997;11(4):341-359.
15. Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks (ICNN); 1995. p. 1942-1948.
16. Virtanen P, Gommers R, Oliphant TE, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods. 2020;17(3):261-272.
17. Hart WE, Watson JP, Woodruff DL. Pyomo: modeling and solving mathematical programs in Python. Math Program Comput. 2011;3(3):219-260.
18. Bynum ML, Hackebeil GA, Hart WE, Laird CD, Nicholson BL, Sirola JD, et al. Pyomo—optimization modeling in Python. 3rd ed. Cham: Springer; 2021.
19. Aggarwal M. Laws and regulations governing the chemical process industries in the U.S. Chem Eng Prog. 2025;121(5):pages as published.