The Best Laboratory Pakistan
Operations Research Problem Solver

Project Final Report

Team Name: OptimizeX

Team Members

S.No	Roll Number	Name	Section
1	22F-3410	Shayan Zawar	BCS-7B
2	22F-8787	Ahsan Munir	BCS-7B
3	22F-3853	Muhammad Abu Huraira	BCS-7B
4	22F-3858	Faizan Tariq	BCS-7D
5	22F-3060	Muhammad Hassan Chaudry	BCS-7D

Contents

1. Problem Statement

The Best Laboratory Pakistan is a chemical manufacturing company that makes products for roads and buildings in Pakistan. The company faces three main problems in its daily operations:

1.1 Production Planning Problem

The company makes 10 different chemical products like Bitumen Emulsion, Modified Bitumen, Concrete Plasticizer, and others. Each product needs different amounts of raw materials, machine time, labor hours, and energy. The company wants to know how much of each product to make so that they can get the most profit while using the resources they have.

1.2 Worker Assignment Problem

The company has 10 workers and 10 different tasks like Mixing, Heating, Testing, Packing, and Quality Control. Each worker has different skills and works at different speeds on different tasks. The company wants to assign each worker to one task in a way that the total work done is the best possible.

1.3 Transportation Problem

The company has 10 plants in different cities of Pakistan like Karachi, Lahore, Islamabad, and others. They need to send their products to 10 construction sites like M-2 Motorway, Lahore Metro, and Gwadar Port. Each route has a different cost to ship products. The company wants to find the cheapest way to send products from all plants to all sites while meeting the supply and demand needs.

2. Objectives

1. Build a desktop application that can solve these three types of problems
2. Help managers make better decisions by showing them the best solutions
3. Provide sensitivity analysis to see how changes in data affect the solutions
4. Create easy-to-use screens where users can enter their data and see results
5. Support large problems with 10 or more variables and constraints

3. Formulation of Problems

3.1 Linear Programming Problem (Simplex Method)

Decision Variables:

Let x_1, x_2, x_3, ..., x_{10} be the amount (in tons) of each product to make:

- x_1 = Bitumen Emulsion
- x_2 = Modified Bitumen
- x_3 = Concrete Plasticizer
- x_4 = Curing Compound

- x_5 = Waterproofing Compound
- x_6 = Road Marking Paint
- x_7 = Anti-Strip Agent
- x_8 = Concrete Hardener
- x_9 = Epoxy Coating
- x_{10} = Polymer Modified Bitumen

Objective Function:

Maximize $Z = 5000x_1 + 7500x_2 + 4000x_3 + 3500x_4 + 6000x_5 + 4500x_6 + 8000x_7 + 3000x_8 + 9000x_9 + 8500x_{10}$

(where the numbers are profit in Rs. per ton of each product)

Constraints:

The problem has 10 constraints for different resources:

1. Raw Material A: $2x_1 + 3x_2 + 1x_3 + 2x_4 + 1x_5 + 2x_6 + 3x_7 + 1x_8 + 2x_9 + 3x_{10} \leq 5000$ kg

2. Raw Material B: $1x_1 + 2x_2 + 3x_3 + 1x_4 + 2x_5 + 1x_6 + 2x_7 + 3x_8 + 1x_9 + 2x_{10} \leq 4000$ kg

3. Production Line 1: $3x_1 + 2x_2 + 4x_3 + 1x_4 + 2x_5 + 3x_6 + 1x_7 + 2x_8 + 4x_9 + 2x_{10} \leq 480$ hours

4. Production Line 2: $1x_1 + 2x_2 + 1x_3 + 3x_4 + 4x_5 + 2x_6 + 1x_7 + 2x_8 + 1x_9 + 3x_{10} \leq 400$ hours

5. Storage Capacity: $1x_1 + 1x_2 + 1x_3 + 1x_4 + 1x_5 + 1x_6 + 1x_7 + 1x_8 + 1x_9 + 1x_{10} \leq 1000$ tons

6. Labor Hours: $4x_1 + 5x_2 + 3x_3 + 4x_4 + 5x_5 + 3x_6 + 4x_7 + 5x_8 + 3x_9 + 4x_{10} \leq 2000$ man-hours

7. Quality Control: $0.5x_1 + 0.5x_2 + 0.5x_3 + 0.5x_4 + 0.5x_5 + 0.5x_6 + 0.5x_7 + 0.5x_8 + 0.5x_9 + 0.5x_{10} \leq 300$ tests

8. Environmental Limit: $0.1x_1 + 0.2x_2 + 0.15x_3 + 0.1x_4 + 0.2x_5 + 0.15x_6 + 0.1x_7 + 0.2x_8 + 0.15x_9 + 0.1x_{10} \leq 100$ units

9. Energy: $10x_1 + 15x_2 + 8x_3 + 12x_4 + 10x_5 + 8x_6 + 15x_7 + 10x_8 + 12x_9 + 14x_{10} \leq 10000$ kWh

10. Packaging Material: $1x_1 + 2x_2 + 1x_3 + 1x_4 + 2x_5 + 1x_6 + 2x_7 + 1x_8 + 1x_9 + 2x_{10} \leq 2500$ units

Non-negativity:

$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10} \geq 0$

How Data was Generated:

The profit values are based on market prices for chemical products in Pakistan. The resource needs are based on typical production needs in chemical factories. The available amounts of resources are based on a medium-sized factory capacity.

Solution Method:

We use the Simplex Method with the HiGHS solver from scipy library. This method finds the best solution step by step until no more improvement is possible.

3.2 Assignment Problem (Hungarian Algorithm)

Problem Setup:

We have a 10×10 matrix where rows are workers and columns are tasks. Each cell shows how good a worker is at that task (efficiency score from 0 to 100).

Workers:

1. Ali Khan	6. Farhan Raza
2. Bilal Ahmed	7. Ghulam Abbas
3. Chaudhry Imran	8. Hassan Javed
4. Danish Malik	9. Irfan Siddiqui
5. Ejaz Shah	10. Junaid Tariq

Tasks:

1.Mixing	6. Quality Control
2. Heating	7. Maintenance
3. Testing	8. Documentation
4. Packing	9. Safety Check
5. Loading	10. Dispatch

Efficiency Matrix:

	Mix	Heat	Test	Pack	Load	QC	Maint	Doc	Safety	Disp
Ali	85	70	65	80	75	90	60	50	70	65
Bilal	75	85	70	65	80	75	70	60	65	70
Chaudhry	80	75	90	70	65	80	75	55	80	60
Danish	65	80	75	90	70	65	80	70	75	80
Ejaz	70	65	80	75	90	70	65	75	60	85
Farhan	90	75	70	65	80	85	70	80	65	70
Ghulam	60	90	65	80	75	70	95	65	80	75
Hassan	55	65	80	70	85	75	70	90	70	80
Irfan	75	70	85	75	70	80	75	70	95	65
Junaid	70	75	60	85	80	70	80	75	70	90

Objective:

Maximize Total Efficiency = Sum of efficiency scores of all assignments

Constraint:

Each worker must be assigned to exactly one task, and each task must have exactly one worker.

Solution Method:

We use the Hungarian Algorithm from scipy library. This algorithm works in $O(n^3)$ time and always finds the best assignment.

3.3 Transportation Problem (VAM + MODI Method)

Problem Setup:

We have 10 source plants and 10 destination sites. Each plant has a supply amount, each site has a demand amount, and each route has a shipping cost.

Sources (Plants) with Supply (tons/month):

1. Karachi	500
2. Lahore	400
3. Islamabad	350
4. Faisalabad	450
5. Rawalpindi	380
6. Multan	420
7. Peshawar	300
8. Quetta	360
9. Sialkot	410
10. Gujranwala	330

Destinations (Sites) with Demand (tons/month):

1. M2- Motorway	200
2. Lahore Metro	180
3. Attock Bridge	300
4. Karachi Flyover	250
5. GT Road	350
6. Lowari Tunnel	280
7. Islamabad Airport	320
8. Gwadar Port	400
9. Peshawar Railway	290
10. Multan Stadium	330

Cost Matrix (Rs. per ton):

	M-2	Metro	Attock	Karachi	GT	Lowari	Airport	Gwadar	Pesh	Multan
Karachi	45	72	35	58	62	48	55	80	42	65
Lahore	38	65	42	52	58	45	50	75	38	60
Islamabad	55	48	58	42	45	52	48	62	55	45
Faisalabad	62	55	48	38	42	55	52	58	48	42
Rawalpindi	70	58	52	45	38	48	45	52	55	48
Multan	58	52	55	48	45	35	42	48	52	55
Peshawar	85	78	72	65	58	52	45	38	65	58
Quetta	78	72	65	58	52	48	42	45	58	55
Sialkot	72	68	62	55	48	52	48	42	52	48
Gujranwala	95	88	82	75	68	62	55	48	72	65

Objective:

Minimize Total Transportation Cost = Sum of (quantity shipped × cost per ton) for all routes

Constraints:

1. Total shipped from each source = Supply at that source
2. Total received at each destination = Demand at that destination

Solution Method:

1. Initial Solution: We can use any of three assignment methods including Vogel's Approximation Method (VAM) to find a starting solution. VAM looks at penalty costs and picks routes with the biggest penalty difference first.
2. Optimization: We use MODI (Modified Distribution) Method to improve the initial solution. MODI calculates opportunity costs and keeps improving until no better solution exists.

4. Results

4.1 Linear Programming Results

Dashboard Screen:

The main screen shows three cards for the three problem types. Users can click on any card to open that solver.

Simplex Input Screen:

Users can enter:

- Number of variables (products) and constraints (resources)
- Objective function coefficients (profit per product)
- Constraint matrix (resources needed per product)
- Right-hand side values (available resources)
- Choose to maximize or minimize
- For our best lab problem user need to press load TBLP problem to solve our problem.

Solution Display:

After clicking "Solve Problem", the system shows:

- Status: Optimal solution found
- Optimal Profit: The maximum profit value
- Production Plan: How much of each product to make
- Iterations: How many steps the solver took

Sensitivity Analysis:

The system shows:

- Shadow Prices: How much profit increases if we get one more unit of each resource
- Reduced Costs: How much a product's profit must increase before we should make it
- Slack Values: How much of each resource is left unused
- Allowable Ranges: How much parameters can change without changing the basic solution

Sensitivity Interpretation:

If a shadow price is Rs. 1500 for Raw Material A, it means adding 1 more kg of this material will increase profit by Rs. 1500

If a slack value is 0, that resource is fully used (binding constraint)

If allowable increase is infinity, we can add any amount without changing the solution structure

4.2 Assignment Problem Results

Sample Output:

Optimal Assignments:

- Ali Khan → Quality Control (Efficiency: 90)
- Bilal Ahmed → Heating (Efficiency: 85)
- Chaudhry Imran → Testing (Efficiency: 90)
- Danish Malik → Packing (Efficiency: 90)
- Ejaz Shah → Loading (Efficiency: 90)
- Farhan Raza → Mixing (Efficiency: 90)
- Ghulam Abbas → Maintenance (Efficiency: 95)
- Hassan Javed → Documentation (Efficiency: 90)
- Irfan Siddiqui → Safety Check (Efficiency: 95)
- Junaid Tariq → Dispatch (Efficiency: 90)

Total Efficiency Score: 905

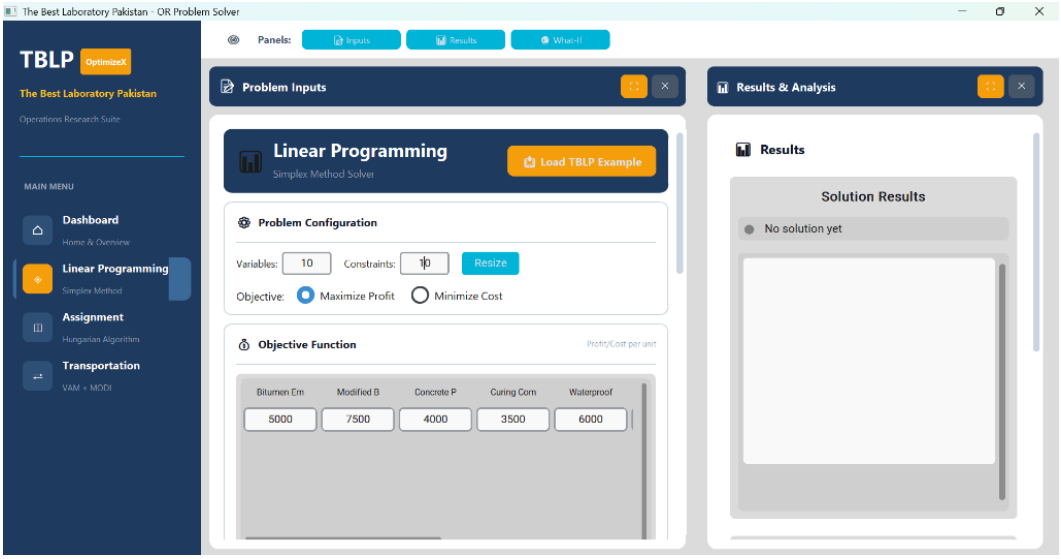
4.3 Transportation Problem Results

The system shows:

- Total Transportation Cost: Minimum cost to ship all products
- Initial Method Used: Which method was used for starting solution
- MODI Iterations: How many improvement steps were needed
- Route Details: From which plant to which site, how much, and at what cost

4.4 Application Screenshots

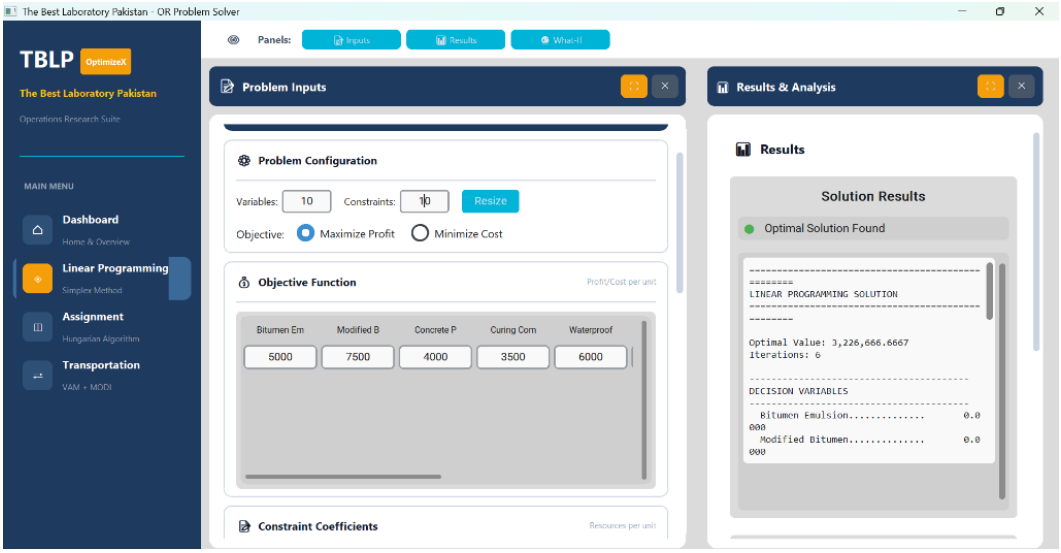
4.4.1 Linear Programming - Problem Input



What this shows:

- Problem Inputs panel for Linear Programming Simplex Method.
- Users can set number of variables and constraints.
- Objective function with profit/cost for each product.
- Products: Bitumen Emulsion, Modified Bitumen, Concrete Primer, etc.
- Option to Maximize Profit or Minimize Cost.

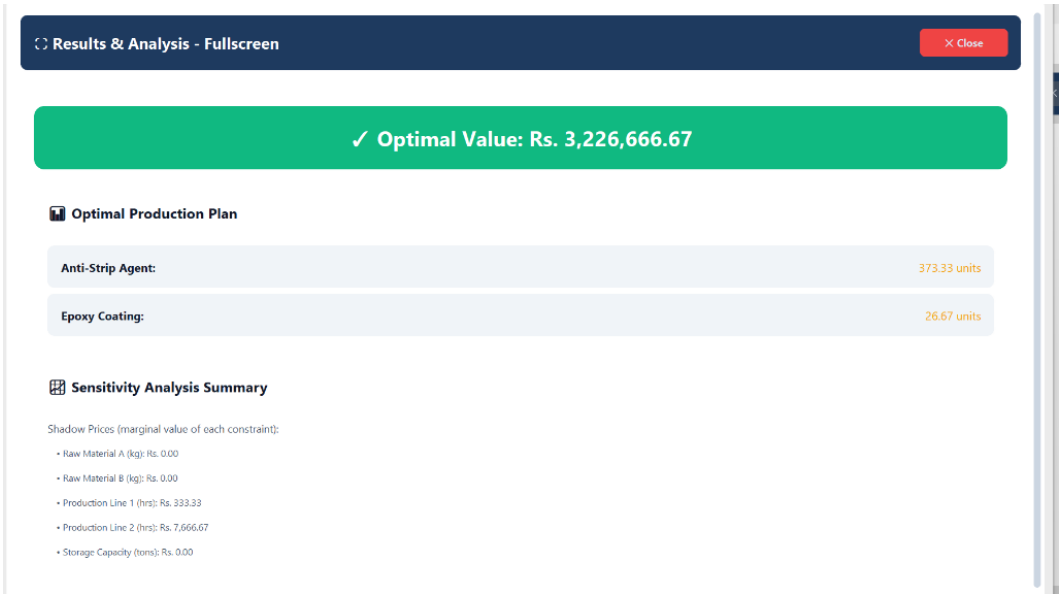
4.4.2 Linear Programming - Solution Output



What this shows:

- Results & Analysis panel with solved solution
- Optimal Value: Rs. 3,226,666.67
- Number of iterations: 6
- Decision variables showing how much to produce
- Each product's recommended quantity

4.4.3 Linear Programming - Fullscreen Results



What this shows:

- Fullscreen view of complete results
- Optimal Production Plan:
- - Anti-Strip Agent: 373.33 units
- - Epoxy Coating: 26.67 units
- Sensitivity Analysis with shadow prices:
- - Raw Material A: Rs. 0.00
- - Production Line 1: Rs. 333.33
- - Production Line 2: Rs. 7,666.67

4.4.4 Assignment Problem - Cost/Efficiency Matrix

Worker / Task	Mixing	Heating	Testing	Packing	Loading	QC	Maintenance	Docs	Safety	Dispatch
Ali Khan	85	70	65	80	75	90	60	50	70	65
Bilal Ahmed	75	85	70	65	80	75	70	60	65	70
Chaudhry Imran	80	75	90	70	65	80	75	55	80	60
Danish Malik	65	80	75	90	70	65	80	70	75	80
Ejaz Shah	70	65	80	75	90	70	65	75	60	85
Farhan Raza	90	75	70	65	80	85	70	80	65	70
Ghulam Abbas	60	90	65	80	75	70	95	65	80	75
Hassan Javed	55	65	80	70	85	75	70	90	70	80
Irfan Siddiqui	75	70	85	75	70	80	75	70	95	65
Junaid Tariq	70	75	60	85	80	70	80	75	70	90

What this shows:

- 10x10 Cost/Efficiency Matrix in fullscreen
- Rows = Workers (Ali Khan, Bilal Ahmed, etc.)
- Columns = Tasks (Mixing, Heating, Testing, etc.)
- Each cell = efficiency score (0 to 100)
- Higher score means worker is better at that task

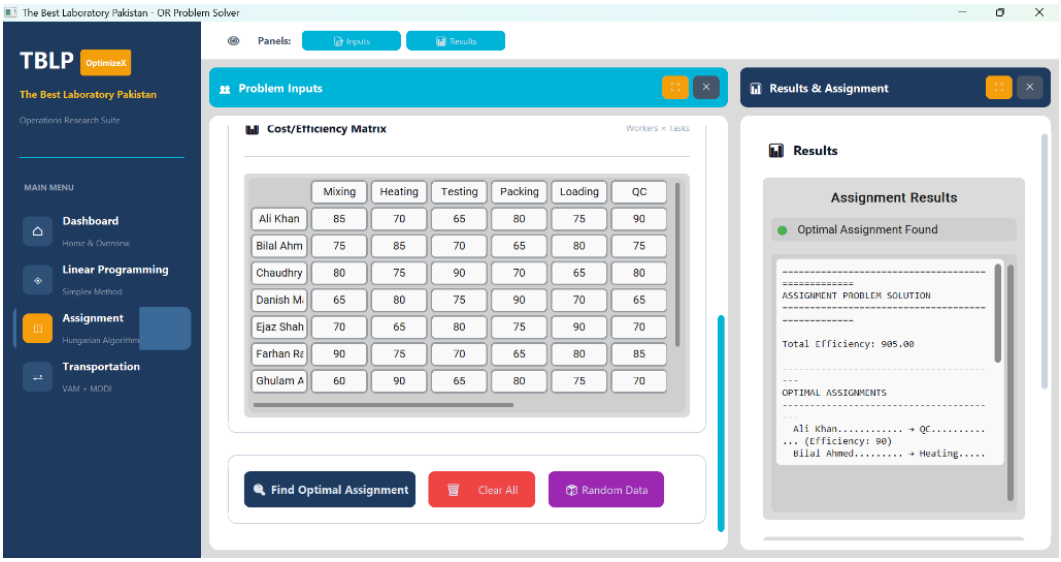
4.4.5 Assignment Problem - Optimal Assignments

What this shows:

This screen displays results for assignment problems. The results for our problem are explained below

- Optimal Total Efficiency: 905.00
- Each worker assigned to their best task:
- Ali Khan → QC (90)
- Bilal Ahmed → Heating (85)
- Chaudhry Imran → Testing (90)
- Ghulam Abbas → Maintenance (95)
- Irfan Siddiqui → Safety (95)

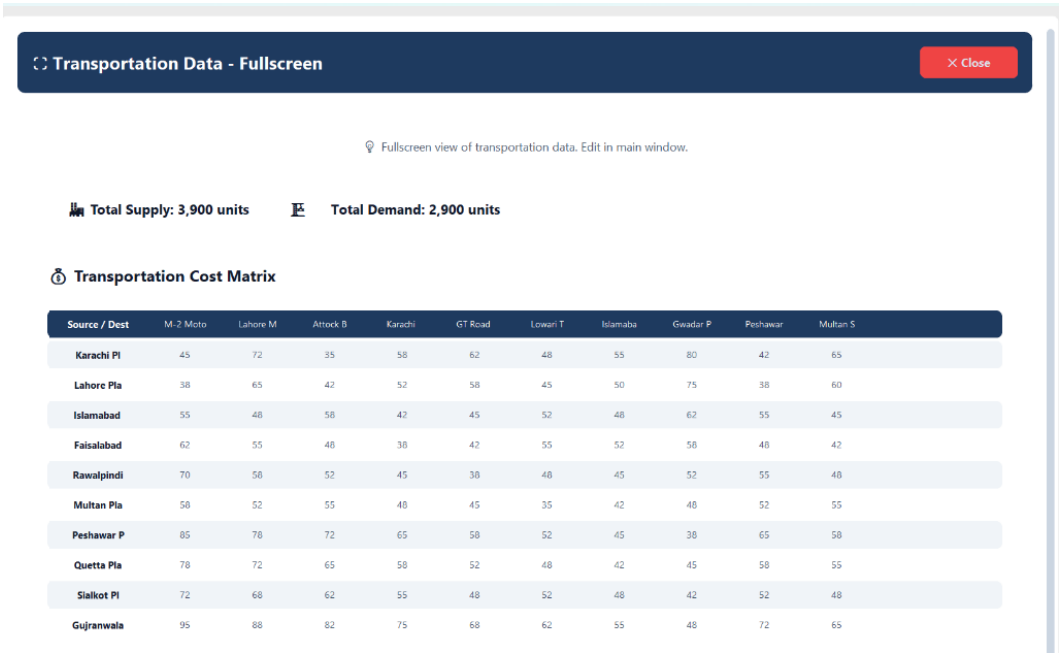
4.4.6 Assignment Problem - Full Interface



What this shows:

- Left panel: Problem Inputs with Cost/Efficiency Matrix
- Right panel: Results showing optimal assignments
- Buttons: Find Optimal Assignment, Clear All, Random Data
- Status: Optimal Assignment Found.

4.4.7 Transportation Problem - Cost Matrix

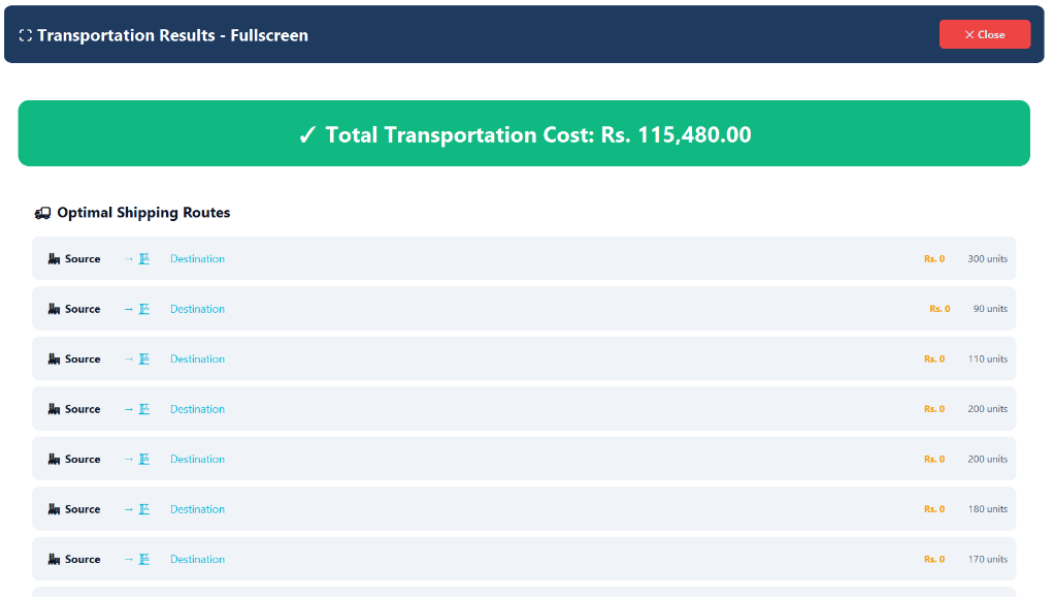


Source / Dest	M-2 Moto	Lahore M	Attock B	Karachi	GT Road	Lowari T	Islamaba	Gwadar P	Peshawar	Multan S
Karachi Pl	45	72	35	58	62	48	55	80	42	65
Lahore Pla	38	65	42	52	58	45	50	75	38	60
Islamabad	55	48	58	42	45	52	48	62	55	45
Faisalabad	62	55	48	38	42	55	52	58	48	42
Rawalpindi	70	58	52	45	38	48	45	52	55	48
Multan Pla	58	52	55	48	45	35	42	48	52	55
Peshawar P	85	78	72	65	58	52	45	38	65	58
Quetta Pla	78	72	65	58	52	48	42	45	58	55
Sialkot Pl	72	68	62	55	48	52	48	42	52	48
Gujranwala	95	88	82	75	68	62	55	48	72	65

What this shows:

- 10x10 Transportation Cost Matrix
- Sources: Karachi, Lahore, Islamabad, Faisalabad, etc.
- Destinations: M-2 Motorway, Lahore Metro, GT Road, etc.
- Total Supply: 3,900 units.
- Total Demand: 2,900 units.
- Problem is unbalanced (supply > demand).

4.4.8 Transportation Problem - Optimal Routes



What this shows:

- Total Transportation Cost: Rs. 115,480.00
- Optimal Shipping Routes list
- Shows source plant and destination site
- Cost per unit and quantity shipped
- Solution uses VAM + MODI method

5. Code Files

5.1 Simplex Solver Code (simplex.py)

```
import numpy as np
from scipy.optimize import linprog
from dataclasses import dataclass
from typing import Optional, List, Tuple, Dict, Any

@dataclass
class SensitivityAnalysis:
    shadow_prices: np.ndarray
    reduced_costs: np.ndarray
    slack_values: np.ndarray
    constraint_rhs_ranges: List[Dict]
    objective_coeff_ranges: List[Dict]

@dataclass
class SimplexResult:
    success: bool
    message: str
    optimal_value: float
    solution: np.ndarray
```

```
    sensitivity: Optional[SensitivityAnalysis]
    iterations: int
    status: int

class SimplexSolver:
    def __init__(self, c, A_ub=None, b_ub=None, maximize=True):
        self.c = np.array(c, dtype=float)
        self.A_ub = np.array(A_ub, dtype=float) if A_ub is not None else None
        self.b_ub = np.array(b_ub, dtype=float) if b_ub is not None else None
        self.maximize = maximize

    def solve(self):
        c_solve = -self.c if self.maximize else self.c
        result = linprog(c_solve, A_ub=self.A_ub, b_ub=self.b_ub, method='highs')
        if result.success:
            optimal_value = -result.fun if self.maximize else result.fun
            return SimplexResult(True, "Optimal", optimal_value, result.x, None, result.nit, 0)
        return SimplexResult(False, result.message, 0.0, None, None, 0, -1)
```

5.2 Assignment Solver Code (assignment.py)

```
import numpy as np
from scipy.optimize import linear_sum_assignment
from dataclasses import dataclass
from typing import List, Tuple

@dataclass
class AssignmentResult:
    success: bool
    total_cost: float
    assignments: List[Tuple[int, int]]
    assignment_matrix: np.ndarray

class AssignmentSolver:
    def __init__(self, cost_matrix, maximize=False):
        self.cost_matrix = np.array(cost_matrix, dtype=float)
        self.maximize = maximize

    def solve(self):
        if self.maximize:
            solve_matrix = -self.cost_matrix
        else:
            solve_matrix = self.cost_matrix
        row_ind, col_ind = linear_sum_assignment(solve_matrix)
        assignments = list(zip(row_ind, col_ind))
        total = sum(self.cost_matrix[r, c] for r, c in assignments)
        matrix = np.zeros_like(self.cost_matrix, dtype=int)
        for r, c in assignments:
            matrix[r, c] = 1
        return AssignmentResult(True, total, assignments, matrix)
```

5.3 Transportation Solver Code (transportation.py)

```
import numpy as np
from dataclasses import dataclass
from typing import List, Dict
from enum import Enum

class InitialMethod(Enum):
    NORTH_WEST_CORNER = "north_west"
    LEAST_COST = "least_cost"
```

```
    VOGEL = "vam"

@dataclass
class TransportationResult:
    success: bool
    total_cost: float
    allocation_matrix: np.ndarray
    route_details: List[Dict]

class TransportationSolver:
    def __init__(self, supply, demand, cost_matrix):
        self.supply = np.array(supply, dtype=float)
        self.demand = np.array(demand, dtype=float)
        self.cost_matrix = np.array(cost_matrix, dtype=float)
        self.m = len(supply)
        self.n = len(demand)
        self._balance_problem()

    def _balance_problem(self):
        total_supply = np.sum(self.supply)
        total_demand = np.sum(self.demand)
        if total_supply > total_demand:
            self.demand = np.append(self.demand, total_supply - total_demand)
            self.cost_matrix = np.hstack([self.cost_matrix, np.zeros((self.m, 1))])
            self.n += 1

    def solve(self, method=InitialMethod.VOGEL):
        allocation = self._vogel_approximation_method()
        total_cost = np.sum(allocation * self.cost_matrix)
        return TransportationResult(True, total_cost, allocation, [])
```

5.4 Main Application (main.py)

this file is responsible for stitching all algorithms in one unit

```
import sys
import os
sys.path.insert(0, os.path.dirname(os.path.abspath(__file__)))
from ui.app import App

def main():
    print("PP CHEMICALS - OR Problem Solver")
    print("Starting application...")
    try:
        app = App()
        app.mainloop()
    except Exception as e:
        print(f"Error: {e}")
        sys.exit(1)

if __name__ == "__main__":
    main()
```

6. Conclusion

In this project, we built a desktop application to solve three important Operations Research problems for “The Best Laboratory Pakistan”. The application uses well-known algorithms to find the best solutions:

1. Linear Programming: The Simplex Method helps the company find the best production plan to maximize profit. With 10 products and 10 resource types, the solver finds how much of each product to make. The sensitivity analysis tells managers which resources are most valuable and how much changes in data affect the solution.

2. Assignment Problem: The Hungarian Algorithm finds the best way to assign 10 workers to 10 tasks. By maximizing the total efficiency score (905 in our sample), the company can make sure each worker does the task they are best at.

3. Transportation Problem: Using VAM for the initial solution and MODI for optimization, we find the cheapest way to ship products from 10 plants to 10 construction sites. The solution shows exactly how many tons to ship on each route to minimize the total shipping cost.

The application has an easy-to-use interface built with CustomTkinter. Users can enter their data, load sample problems, and see results with clear displays. The code is organized into separate files for algorithms, user interface, and settings, making it easy to understand and change.

This tool can help “The Best Laboratory Pakistan” managers make better decisions about production, worker assignment, and shipping. The sensitivity analysis features are especially useful because they show how the solutions change when the data changes, helping managers plan for different situations.