Sharif University of Technology
Department of Industrial Engineering

Operations Research II

Instructor: Dr. Maryam Radman

Ava Sedighi

Spring 2025

Table of Content

Mathematical Model

Parameters

c_i: *Cargo load of ship i*

C_j: *Cargo capacity of berth j*

l_i: *Length of ship i*

L_j: *Length of berth j*

t_{ij}: *Time required for ship i to be unloaded at berth j*

p_i: *Priority of ship i (between 0 and 1)*

C_{EC_j}: *Cost of using extra capacity of berth j*

E_j : *Amount of extra capacity used in berth j*

A_i: *Arrival time of ship i*

ε_j: *Required time interval between consecutive unloading at berth*

Decision Variables

W_i: *Waiting time of ship i*

$$x_{ij} \begin{cases} 1 & if\ ship\ i\ is\ assigned\ to\ berth\ j \\ 0 & otherwise \end{cases}$$

$$y_{ik} \begin{cases} 1 & if\ p_i > p_k \\ 0 & if\ p_k > p_i \end{cases} \qquad i, k \in [1, 8]\ , i < k$$

u_j: *Amount of extra capacity used at berth j*

$$\delta_{1i} \begin{cases} 1 & if\ W_i \geq 10 \\ 0 & otherwise \end{cases} \qquad \delta_{2i} \begin{cases} 1 & if\ W_i \geq 40 \\ 0 & otherwise \end{cases}$$

$$d_{1i} \begin{cases} W_i & if\ W_i \geq 10 \\ 0 & otherwise \end{cases} \qquad d_{2i} \begin{cases} W_i & if\ W_i \geq 40 \\ 0 & otherwise \end{cases}$$

Objective Function

The objective function includes three components:

1. Unloading cost (proportional to unloading time)
2. Extra capacity usage cost
3. Waiting time cost (piecewise-linear function)

$$Min\ Z = C_{unload} + C_W + C_{EC}$$

$$C_{unload} = \sum_j \sum_i 5\ x_{ij} t_{ij}$$

$$C_{EC} = \sum_j u_j\ C_{EC_j}$$

$$\begin{aligned} C_W = \sum_i C_{W_i} &= \sum_i 25W_i + \delta_{1i}(-15W_i + 150) + \delta_{2i}(-5W_i + 200) \\ &= \sum_i 25W_i - 15\ \delta_{1i}W_i + 150\ \delta_{1i} - 5\ \delta_{2i}W_i + 200\ \delta_{2i} \\ &= \sum_i 25W_i - 15\ d_{1i} + 150\ \delta_{1i} - 5\ d_{2i} + 200\ \delta_{2i} \end{aligned}$$

Constraints

- Each ship is assigned to exactly one berth:

$$\sum_j x_{ij} = 1, \qquad \forall i \in [1, 8]$$

- The number of ships assigned to berths is limited by the number of available positions:

$$\sum_i x_{ij} \le 2, \qquad \forall j \in [1, 2, 4]$$

$$\sum_i x_{i3} = 3$$

- Capacity constraints, including the use of extra capacity.

$$\sum_i x_{ij}\ c_i \le C_j + u_j\ , \qquad \forall j \in [1, 4]$$

- Ship lengths cannot exceed berth length limits.

  $l_i\, x_{ij} \le L_j, \qquad \forall j \in [1,4], \qquad \forall i \in [1,8]$

- Sequencing and priority constraints among ships.

  $p_i - p_k < y_{ik}, \qquad i,k \in [1,8], \qquad i < k$

  $p_k - p_i < 1 - y_{ik}, \qquad i,k \in [1,8], \qquad i < k$

  Note: Since Pyomo does not allow strict inequalities, we use "≤". There are no equal-priority ships, so this does not affect the model.

- Constraints on ship sequencing for shared berths — controlled by an auxiliary condition to prevent redundant constraints.

  $$W_k + A_k \ge W_i + A_i + t_{ij} + \varepsilon_j - M\left(1 - y_{ik} + 1 - x_{ij} + 1 - x_{kj}\right)$$
  $$j \in [1,4], \qquad i,k \in [1,8], \qquad i < k$$

  $$W_i + A_i \ge W_k + A_k + t_{ij} + \varepsilon_j - M\left(y_{ik} + 1 - x_{ij} + 1 - x_{kj}\right)$$
  $$j \in [1,4], \qquad i,k \in [1,8], \qquad i < \mathrm{k}$$

- Upper bounds on waiting time (not strictly defined as the main focus is cost minimization).

  $W_i \le 80\,(1 - p_i), \qquad \forall i \in [1,8]$

- $\boldsymbol{\delta_{1i}}$ **&** $\boldsymbol{\delta_{2i}}$**:**

  $W_i - 10 \le M\delta_{1i}, \qquad i \in [1,8]$

  $W_i - 40 \le M\delta_{2i}, \qquad i \in [1,8]$

  Since the objective function minimizes total cost and coefficients of certain variables are negative, lower bounds are unnecessary, but upper bounds ensure model feasibility.

- Linearization of the objective function

  $d_{1i} \le W_i, \qquad i \in [1,8]$
  $d_{2i} \le W_i, \qquad i \in [1,8]$

  $d_{1i} \le M\delta_{1i}, \qquad i \in [1,8]$
  $d_{2i} \le M\delta_{2i}, \qquad i \in [1,8]$

Solving the Model with Pyomo

Model and Set Definition

A *ConcreteModel* is defined with two main sets:

- Ships = $\{1, \ldots, 8\}$
- Berths = $\{1, \ldots, 4\}$

Reading Data

Parameters are imported from an Excel file named **Parameters.xlsx** and converted into Python dictionaries to feed Pyomo parameters.

```
# loading excel sheets
c = pd.read_excel('Parameters.xlsx', sheet_name='Containers')
C = pd.read_excel('Parameters.xlsx', sheet_name='Capacity')
l = pd.read_excel('Parameters.xlsx', sheet_name='Length')
L = pd.read_excel('Parameters.xlsx', sheet_name='Max Length')
t = pd.read_excel('Parameters.xlsx', sheet_name='Unload Time')
p = pd.read_excel('Parameters.xlsx', sheet_name='Priority')
C_ec = pd.read_excel('Parameters.xlsx', sheet_name='Cost for Extra Capacity')
E = pd.read_excel('Parameters.xlsx', sheet_name='Max Extra Allowable Capacity')
A = pd.read_excel('Parameters.xlsx', sheet_name='Arrival Time')
epsilon = pd.read_excel('Parameters.xlsx', sheet_name='Lag')
anch = pd.read_excel('Parameters.xlsx', sheet_name='Anchorage')
```

Parameters

All parameters such as capacity, ship lengths, unloading times, arrival times, priorities, extra capacities, and costs are defined via Param. Some parameters (e.g., unloading time) are two-dimensional (ship–berth).

```
model_p.ShipBerths = Set(dimen=2, initialize=t_dict.keys())


# parameters
model_p.c = Param(model_p.Ships, initialize=c, mutable=True)
model_p.C = Param(model_p.Berths, initialize=C, mutable=True)
model_p.l = Param(model_p.Ships, initialize=l, mutable=True)
model_p.L = Param(model_p.Berths, initialize=L, mutable=True)
model_p.t = Param(model_p.ShipBerths, initialize=t_dict, mutable=True)
model_p.p = Param(model_p.Ships, initialize=p, mutable=True)
model_p.C_ec = Param(model_p.Berths, initialize=C_ec, mutable=True)
model_p.E = Param(model_p.Berths, initialize=E, mutable=True)
model_p.A = Param(model_p.Ships, initialize=A, mutable=True)
model_p.epsilon = Param(model_p.Berths, initialize=epsilon, mutable=True)
model_p.anch = Param(model_p.Berths, initialize=anch, mutable=True)
```

Decision Variables

Key variables:

- $x[i, j]$: Ship *i* assigned to berth *j*
- $y[i, k]$: Ship *i* precedes ship *k*
- $W[i]$: Waiting time of ship *i*
- $u[j]$: Extra capacity used at berth *j*
- $\delta 1[i], \delta 2[i]$: Auxiliary binary variables for piecewise cost

```
# if ship i assigned to berth j
model_p.x = Var(model_p.Ships, model_p.Berths, domain=Binary)

# if ship i is shipper than k
model_p.Y_index = Set(initialize=[(i, k) for i in model_p.Ships for k in model_p.Ships if i < k])
model_p.y = Var(model_p.Y_index, domain=Binary)

# waiting time of ship i
model_p.W = Var(model_p.Ships, domain=NonNegativeReals)

# used extra capacity
model_p.u = Var(model_p.Berths, domain=NonNegativeReals)

# if W[i] ≥ 10
model_p.delta1 = Var(model_p.Ships, domain=Binary)

# if W[i] ≥ 40
model_p.delta2 = Var(model_p.Ships, domain=Binary)

# if W[i] ≥ 10
model_p.d1 = Var(model_p.Ships, domain=NonNegativeReals)

# if W[i] ≥ 40
model_p.d2 = Var(model_p.Ships, domain=NonNegativeReals)
```

Objective Function

Sum of:

1. Unloading cost
2. Extra capacity cost
3. Waiting time cost (piecewise-linear)

```
model_p.objective = Objective(rule=objective_rule, sense=minimize)
```

Solving

The model is solved using the **GLPK solver**, and outputs include the optimal objective value, ship–berth assignments, and waiting times.
A detailed interpretation of these results is given later in the sensitivity analysis section.

Solving the Model with Gurobipy

Similar to Pyomo:

- Define parameters, variables, objective, and constraints.
- Sequencing and waiting constraints are explicitly expressed.
  The solution closely matches the Pyomo results.

Downloading DataFrame and Visualization

A DataFrame is built (as required in part 3 of the assignment) and saved in the attached zip file. A visual representation of ship assignments to berths is created.

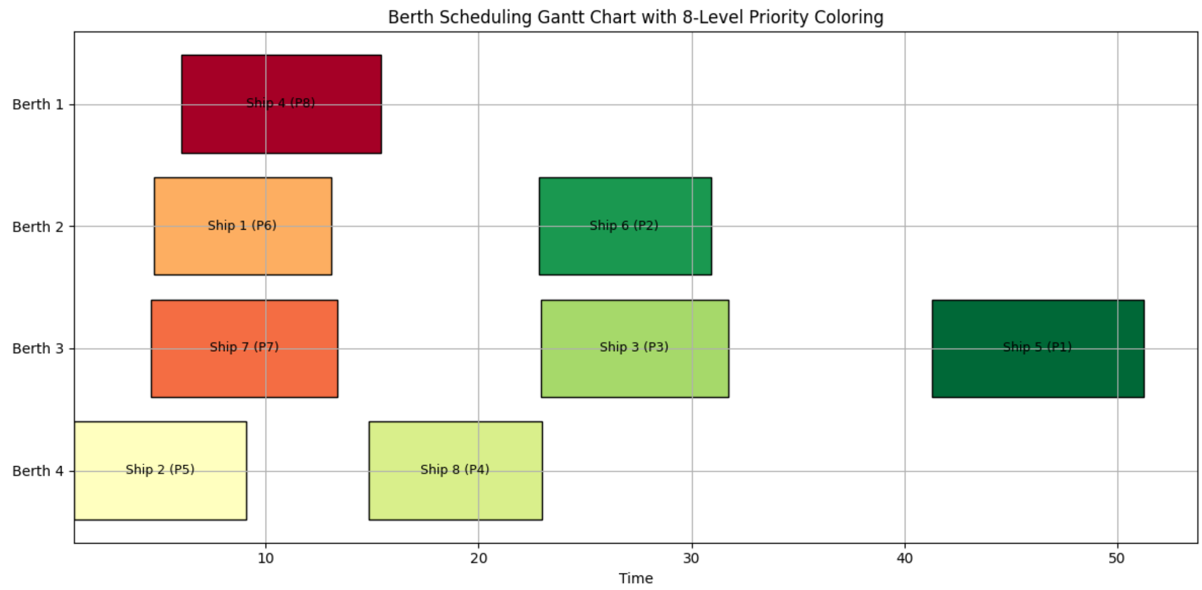


Chart 1: Ship–berth assignments and unloading times. Each rectangle represents the unloading duration of a ship. Greener colors indicate lower priority (lower waiting cost). The x-axis is time; the y-axis represents berths.

Observations:

- Ship 4 (late arrival, high priority) is assigned to a separate berth to minimize total cost.
- Berth 3, which must host 3 ships, receives the lowest-priority ships.

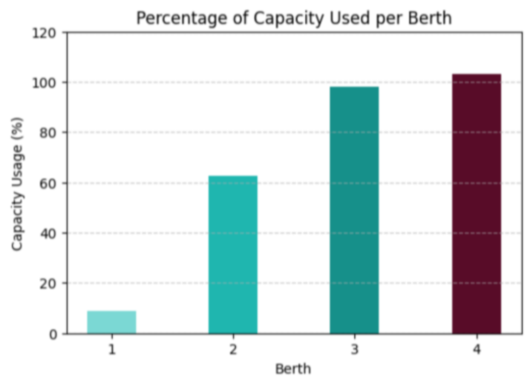


Chart 2: Berth capacity utilization.

- Berth 3 uses almost all of its capacity.
- Berth 1 uses little capacity because it hosts the most important ship (Ship 4).
- Berth 4 requires extra capacity.
- Berth 2 uses about 60% of its capacity.

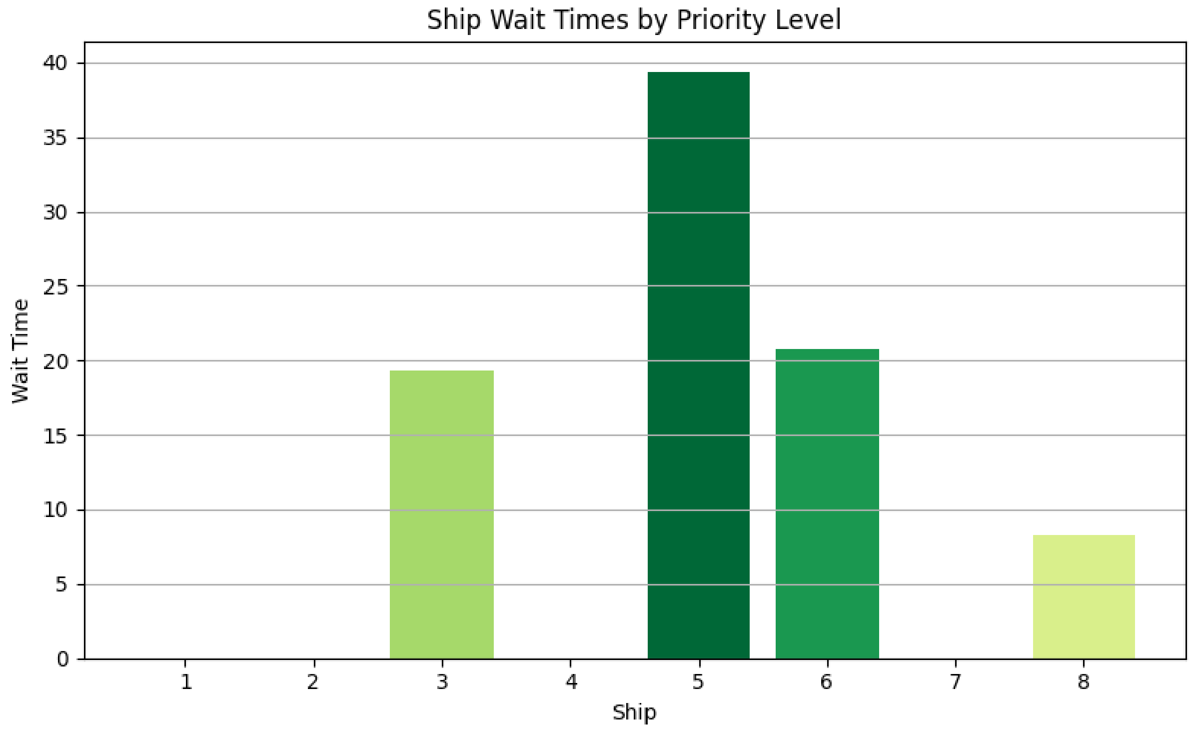
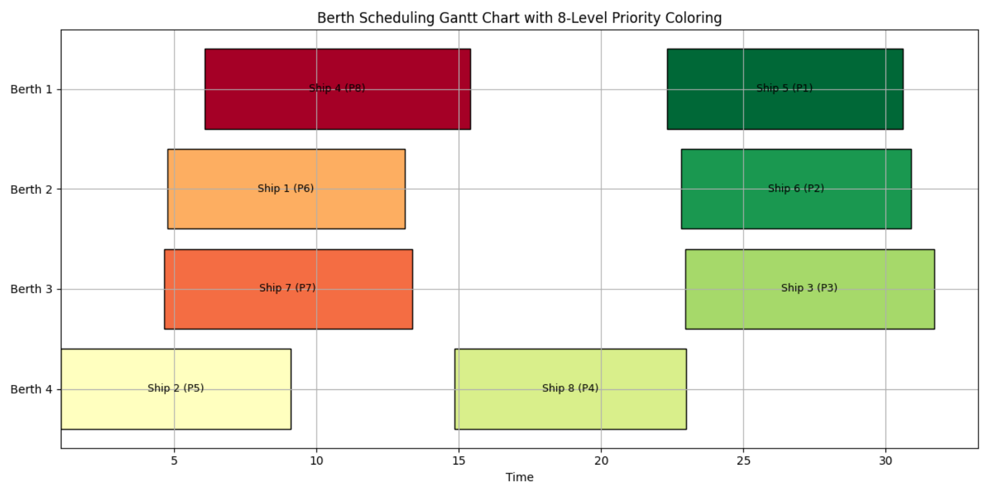


Chart 3: Waiting times of ships.

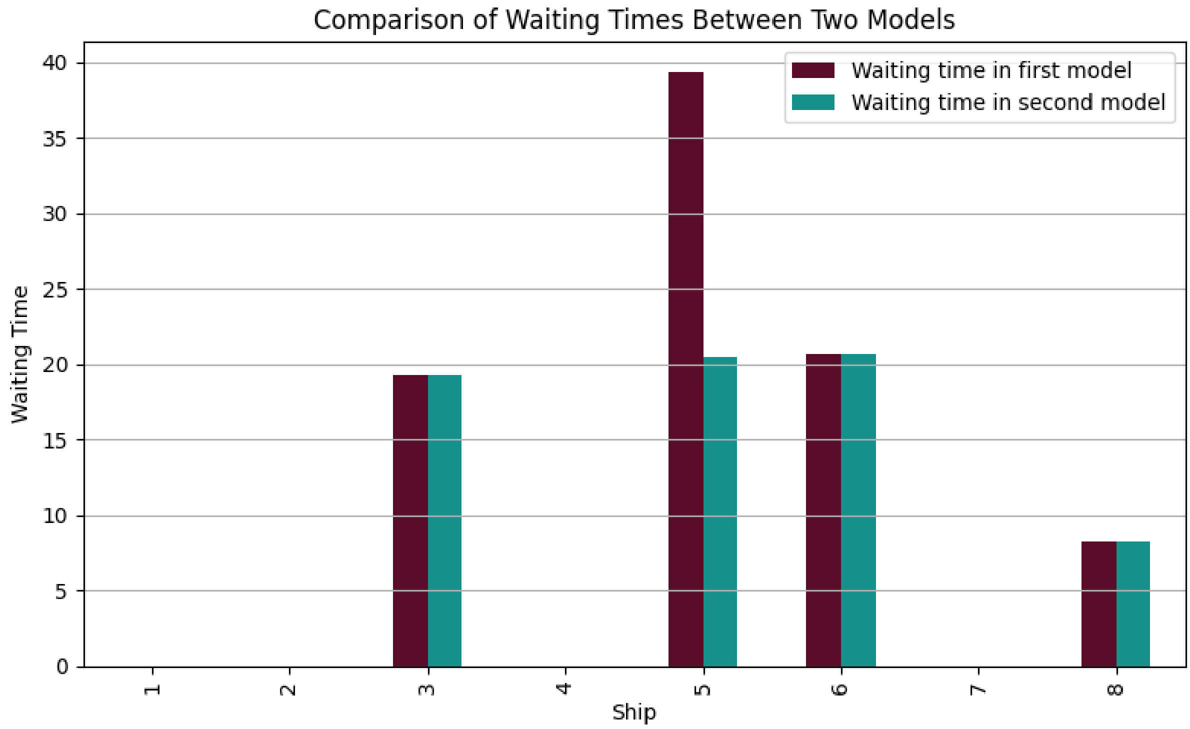
- The lowest-priority ship has the highest waiting time.
- Higher-priority ships wait less or not at all.

Relaxing the Constraint of Assigning Three Ships to Berth 3

When this restriction is removed, the objective function improves significantly:



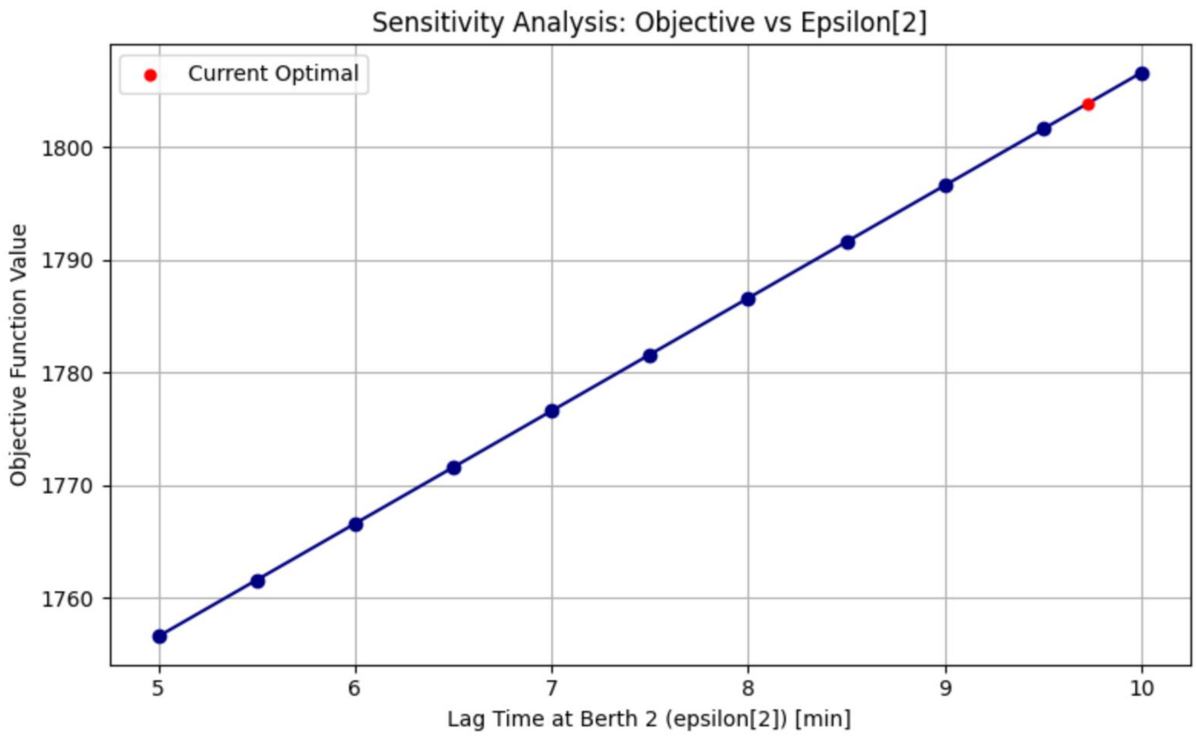
Ship 5 is reassigned from Berth 3 to Berth 1, reducing its waiting time and total cost.



Although Berth 4 still needs extra capacity, removing the constraint decreases unloading and waiting costs — two of the three components of the objective function.

Sensitivity Analysis

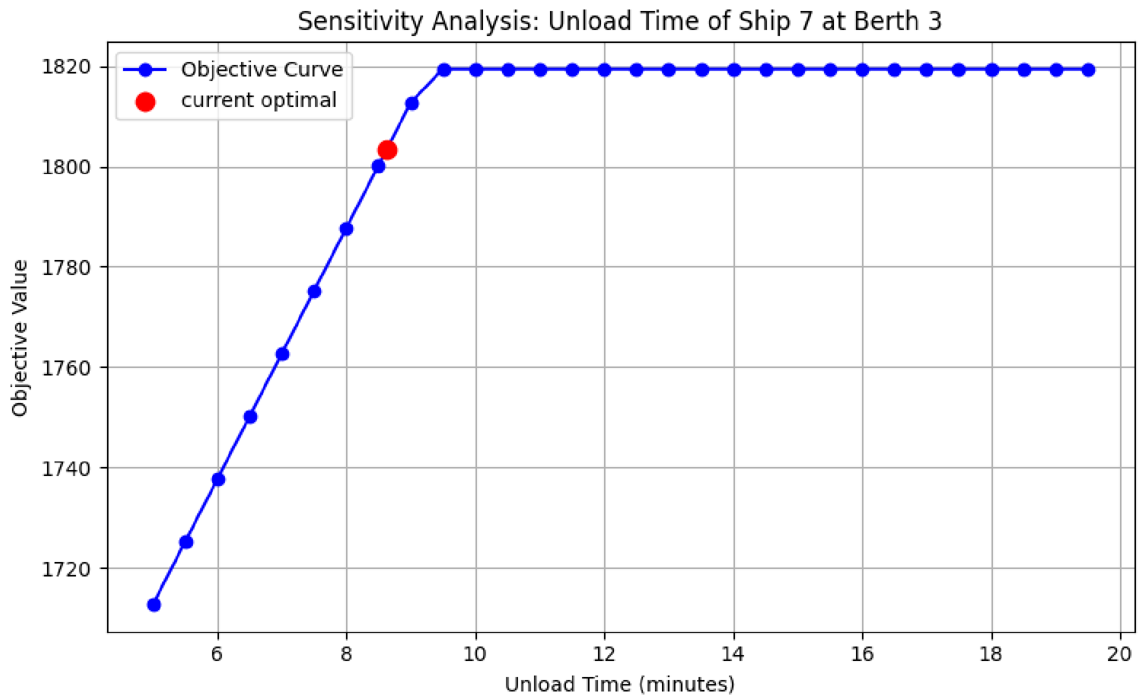
Time Between Two Ship Unloadings at Berth 2



The chart shows the effect of changing ε (time gap) between unloadings. Increasing ε from 5 to 10 minutes raises the objective function linearly by 5 units for every 0.5-minute increase.

→ The system is highly sensitive to unloading intervals; shorter intervals improve efficiency but may require more resources.

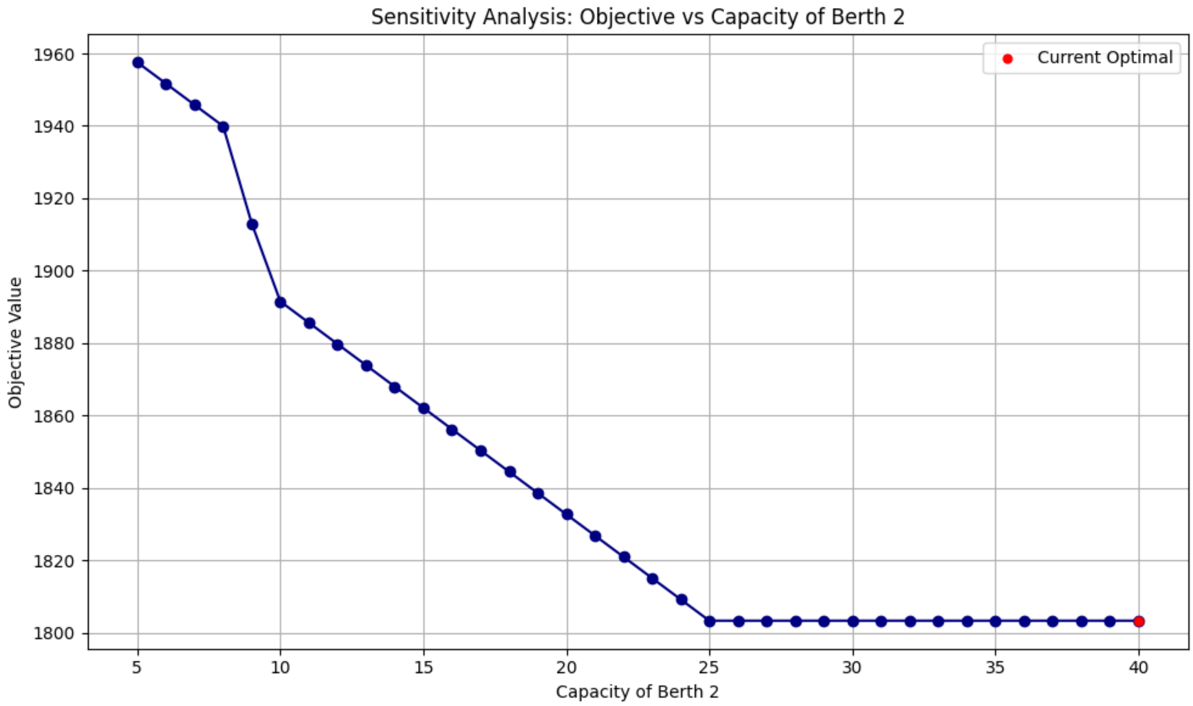
Unloading Time of Ship 7 at Berth 3



As unloading time increases from 5 to 15 minutes, the objective function grows correspondingly.
After around 9.5 minutes, the curve flattens — indicating that beyond this point, longer unloading times have minimal effect.

→ Optimizing ship 7's unloading time is critical for efficiency.

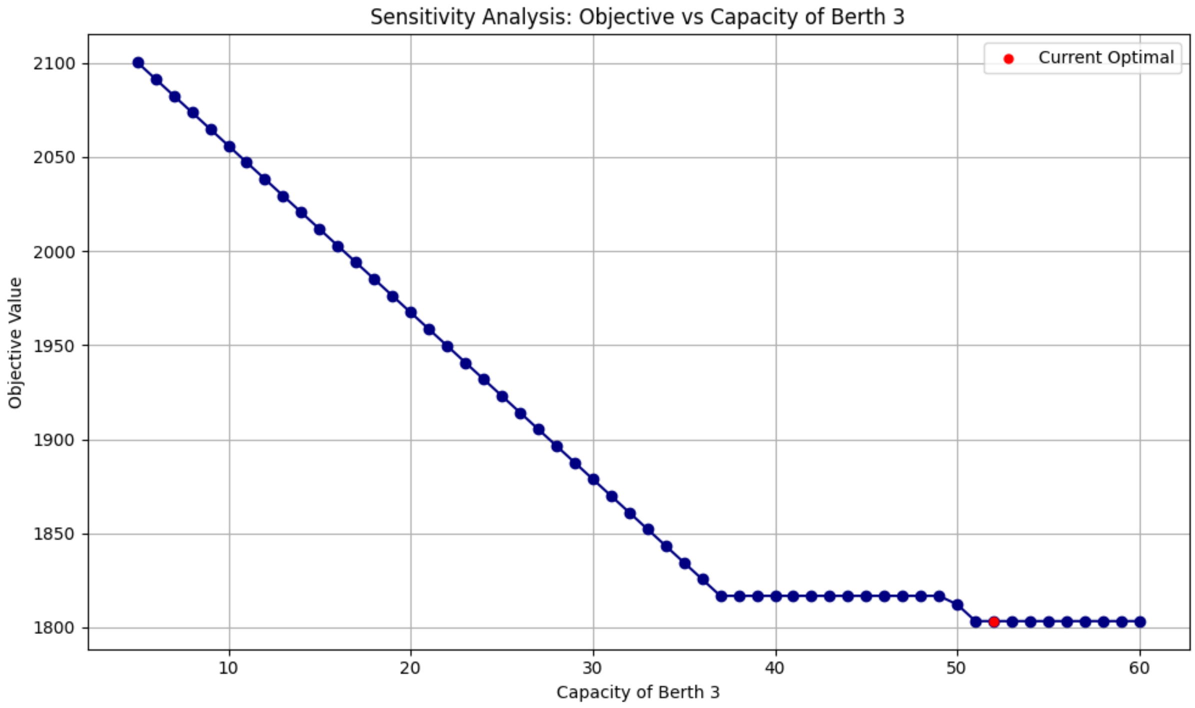
Capacity of Berth 2



As capacity increases from 5 to 40 units, the objective value decreases sharply, then stabilizes around 25 units.

→ Beyond this point, capacity is no longer the limiting factor; other parameters dominate system performance.

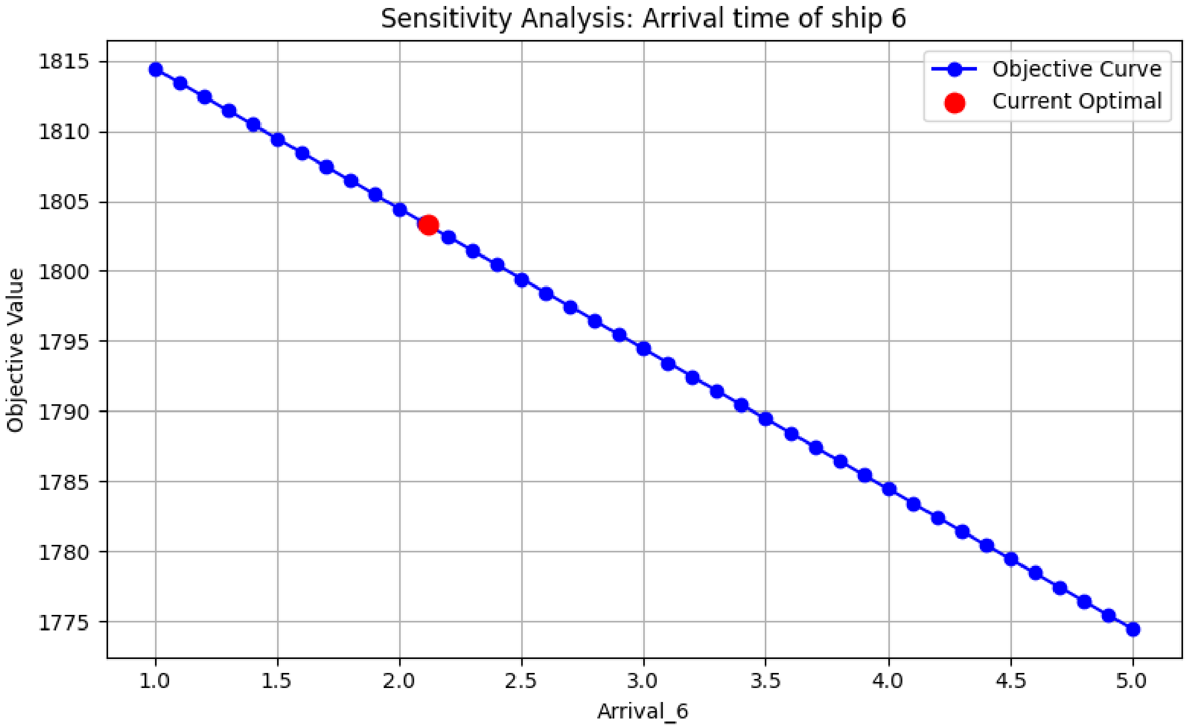
Capacity of Berth 3



Increasing capacity from 5 to 60 units first reduces the objective value significantly, then levels off, and slightly decreases again near 52 units — the optimal capacity.

→ Higher capacity initially improves efficiency but has diminishing returns after a certain point.

Arrival Time of Ship 6



As arrival time increases from 1 to 5 minutes, the objective value decreases linearly (1 unit per 0.1-minute delay).

→ A controlled delay in Ship 6's arrival can improve system performance.

Thank you for your attention!