Container Loading Problem

A Search-Based Solution

Gour Krishna Dey | Roll - MT24035

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Overview

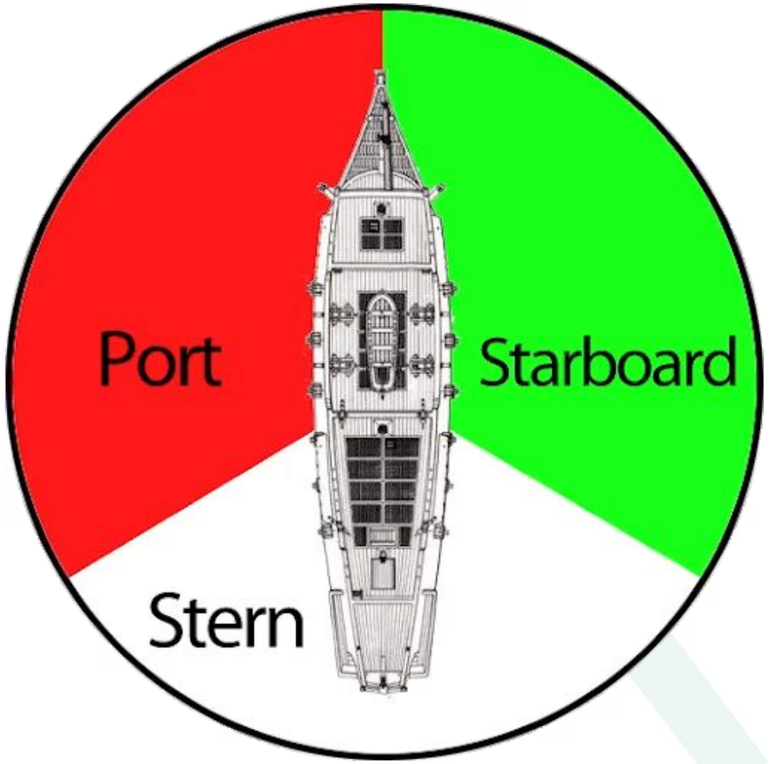
- Goal: Optimize loading of containers on ship.
- Minimize cost while respecting:
    - Balance constraints
    - Weight distribution
    - Destination order
- Implemented Algorithms: **BFS, Greedy Best-First, A***.
- Input → Config file (input.txt)
- Output → Solution plan + cost + violations

Problem Formulation

- **State (S):**
  `s=⟨stacks,loaded_mask,g,h,f⟩`
  (stacks arrangement, placed containers, cost so far, heuristic, evaluation function).
- **Initial State (s_0):**
  All stacks empty, no containers loaded.
- **Actions (A):**
  Place an unloaded container c_i on a valid stack $stack_j$.
- **Transition Model (T):**
  Update stacks, loaded_mask, and cost after each action.
- **Goal Test (G):**
  All containers loaded **while satisfying hard constraints**:
  - Stack height ≤ H
  - Balance difference ≤ B
  - No heavier above lighter

Problem Formulation 2

- **Goal**: Load containers into ship stacks while minimizing violations and cost.
- Ship model:
  - 2 sides → Port & Starboard
  - Each side → 2 stacks (total = 4 stacks)



- **Constraints**:
  - Max stack height (H)

    $|stack_j| \leq H, \forall j \in \{1,2,3,4\}$
  - Balance limit (B) between port & starboard

    $|W_{port} - W_{starboard}| \leq B$
  - Weight ordering (no heavier above lighter)

    If c_i is below c_j in the same stack, then $w_i \geq w_j$
  - Destination order (earlier ports not blocked)

    If c_i (dest d_i) is below c_j (dest d_j), then $d_i \leq d_j$

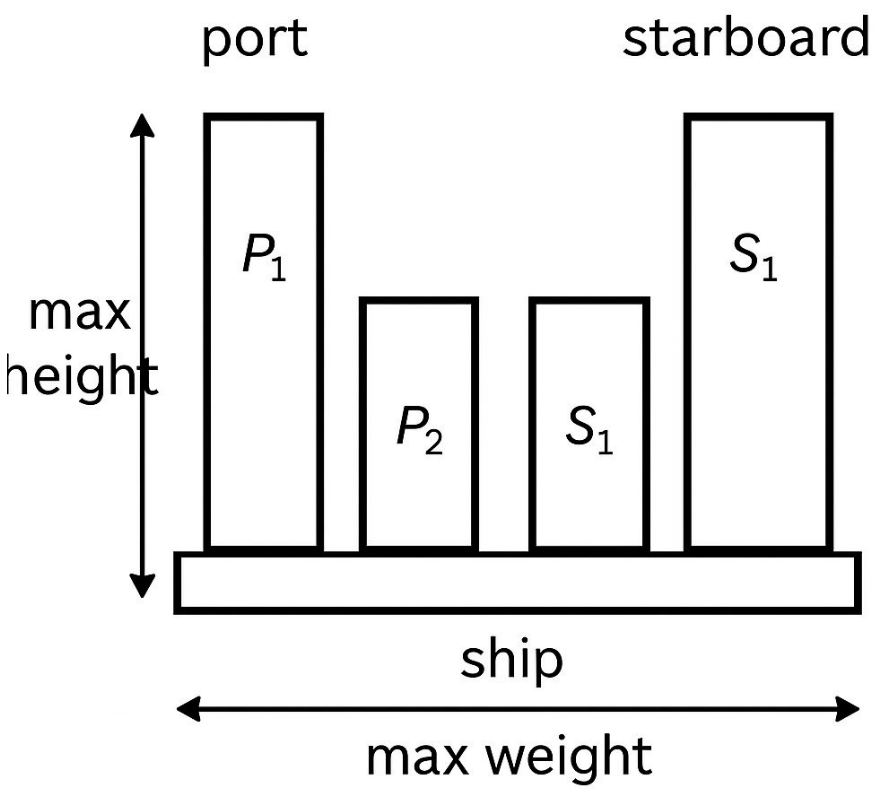
Objective Function & Assumptions

Objective Function:

- Load cost = 1 per container
- Destination violation = +2 per violation
- Weight violation = +1
- Balance violation = forbidden (hard constraint)

$$\mathrm{Cost}_{net} = f(s) = g(s) + h(s)$$

Total Cost = (Number of Containers × 2) + (Total Violations × 2)



Assumptions:

- Ship has **4 stacks** (2 Port, 2 Starboard)
- Hard constraints: stack height ≤ H, balance ≤ B, no heavier above lighter
- Soft constraint: destination order (penalty if violated)

Algorithm & DataStructures

Algorithms Used:

- **BFS**: Complete, but exponential → impractical for large cases.
- **Greedy Best-First**: Fast, relies only on heuristic.
- **A***: Combines cost (g) + heuristic (h), guarantees optimality.

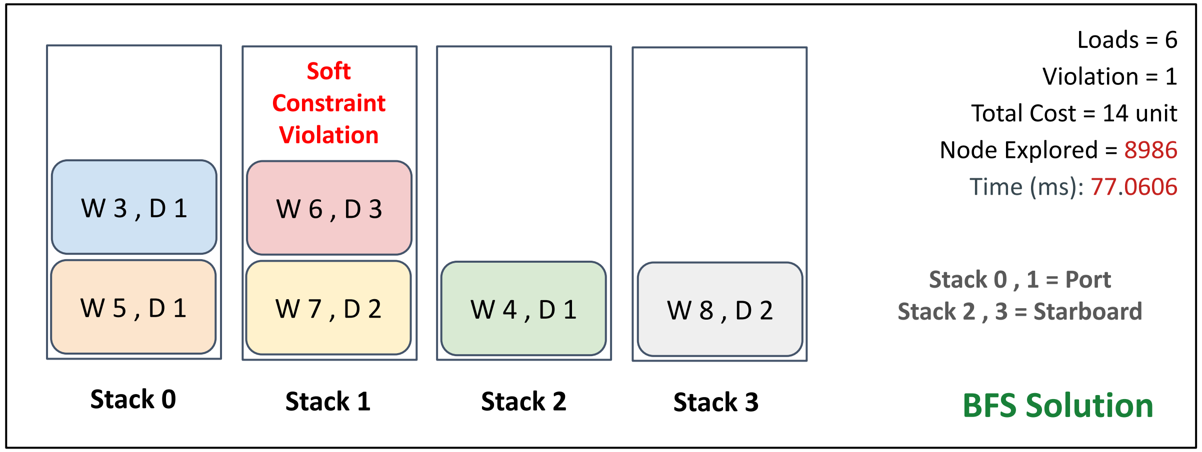
Data Structures:

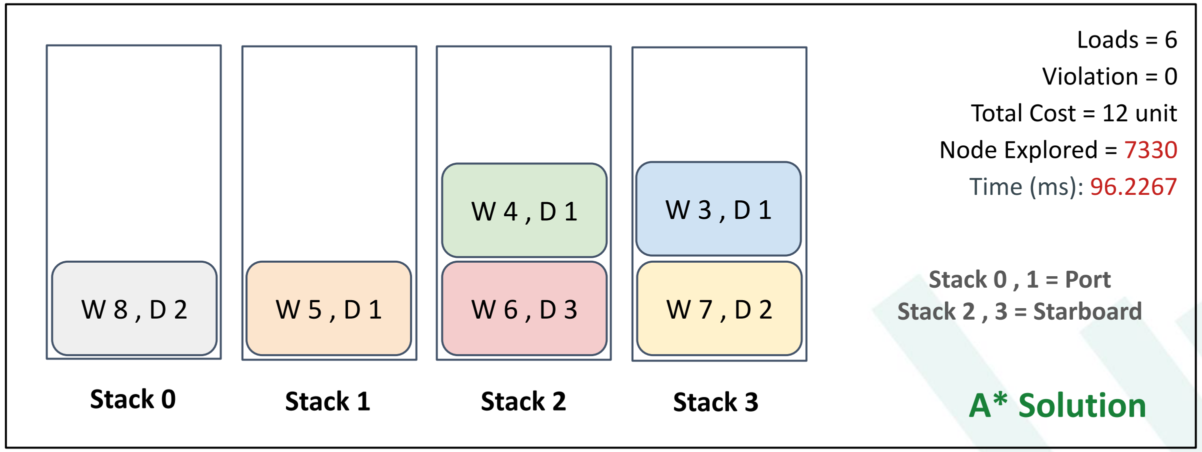
- State = ⟨stacks, loaded_mask, g, h, f⟩
- Priority queue (Greedy & A*)
- Queue (BFS)
- Parent map for path reconstruction

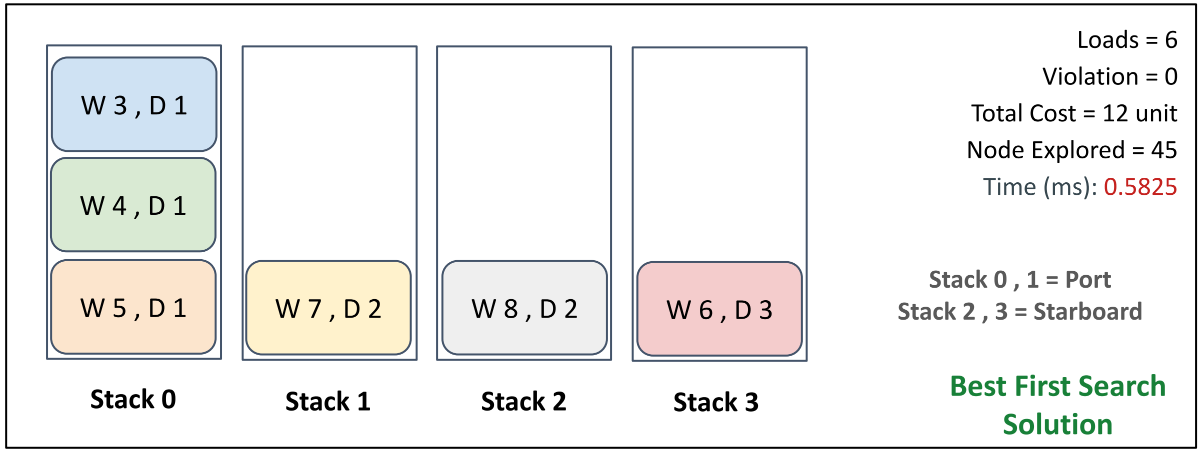
Input & Output

Input (example input.txt):

```arduino
4 3 15    ← stacks, max height, balance limit
6         ← number of containers
5 1       ← weight=5, dest=1
7 2
4 1
6 3
8 2
3 1
```







Conclusion

- **BFS**: Useful for correctness verification, but impractical beyond small cases.
- **Greedy**: Provides rapid approximate solutions; highly efficient if heuristics are accurate.
- **A***: Best practical approach—balances efficiency with guaranteed optimality.
- Demonstrates how **AI search techniques** can significantly improve **real-world logistics optimization**.

Links

Github Repo for Code , Reports & PPT - https://github.com/gkdey17cse/AI_Assignment_2025/tree/main/Assignment_1
Submission Drive Link - https://drive.google.com/drive/folders/1-Yc7cxQb_E9DRst-pCqztCKtFjkjwC8H?usp=sharing