The Heterogeneous Fleet Vehicle Routing Problem with Draft Limits

Mathematical Optimization - A.A. 2024-2025

Elma Huseinovic, Elisabetta Chisso

Problem Description

Context: In modern maritime transport, naval gigantism has led to increasingly large vessels with greater drafts

Operational issue: Ports have draft limits-ships that are too heavily loaded cannot enter them, which affects the sequence of port visits.

Objective: Minimize the total network cost (port access + sailing) by deciding how many and which ships (of different sizes) to use, and in what sequence to visit the ports.

Novelty:

- Incorporates load-dependent draft limits into routing decisions.
- Considers a heterogeneous fleet (ships with different capacities, costs, and drafts).
- Integrates fleet sizing and routing under draft constraints.

Formulation

SETS

- $I = [1, I_{max}]$ $\rightarrow$ set of ports
- $I0 = [0, I_{max}]$ $\rightarrow$ set of ports including the depot
- $S = [1, S_{max}]$ $\rightarrow$ set of ships

PARAMETERS

- Q_s → Capacity (tons) of the ship s
- q_i → Demand (tons) of the port i
- L_{is} → Maximum loading for ship s to access port i (tons)
- t_{ij} → Sailing time between port i and port j (h)
- c_s → Hourly sailing cost for ship s (€/h)
- r_{is} → Access cost for ship s entering port i (€)

VARIABLES

- l_{is} $\rightarrow$ Loading of ship s entering port i
- $u_i \in N^+ \quad \forall i \in I$ $\rightarrow$ Position of port i in the sequence of visited ports
- $p_s = \sum_{\{i \in I\}} q_i Y_{is} \; \forall s \in S$ $\rightarrow$ Total load for ship s

DECISION VARIABLES

- $X_{ijs} \in \{0, 1\} \quad \forall i \in I0 \quad \forall j \in I0 \quad \forall s \in S$ Takes value 1 if the arc (i, j) is traversed by ship s
- $Y_{is} \in \{0,1\} \quad \forall i \in I \quad \forall s \in S$ Takes value 1 if port i is served by ship s

OBJECTIVE FUNCTION

The goal is to minimize the total network cost

$$min \sum_{i \in I0} \sum_{j \in I0} \sum_{s \in S} c_s t_{ij} X_{ijs} + \sum_{i \in I} \sum_{s \in S} r_{is} Y_{is}$$

where the first term

$$\sum_{i \in I0} \sum_{j \in I0} \sum_{s \in S} c_s t_{ij} X_{ijs}$$

represents the sailing cost, and the second term

$$\sum_{i \in I} \sum_{s \in S} r_{is} Y_{is}$$

represents the sum of the fixed costs to access ports

CONSTRAINTS

1)

$$\sum_{s \in S} Y_{is} = 1 \quad \forall i \in I$$

imply that each port is assigned to a ship

2)

$$\sum_{i \in I} q_i Y_{is} \leq Q_s \quad \forall s \in S$$

ensure that the maximum load capacity of a ship is never exceeded

3)

$$\sum_{i \in I0} X_{ijs} = Y_{is} \quad \forall j \in I \quad \forall s \in S$$

if ship s serves port j, it must have previously visited another port including the depot

4)

$$\sum_{i \in I0} X_{ijs} = \sum_{i \in I0} X_{jis} \quad \forall j \in I \quad \forall s \in S$$

for each port and ship, the number of incoming arcs to j equals the number of outgoing arcs from j, ensuring flow conservation

5)

$$X_{0js} \leq \sum_{j \in I} Y_{js} \quad \forall s \in S$$

a ship may depart from the depot only if it serves at least one port

6)

$$X_{0js} \geq \sum_{j \in I} \frac{Y_{js}}{I_{max}} \quad \forall s \in S$$

if a ship serves any port, it must depart from the depot

7)

$$u_j \geq u_i + 1 - I_{max}(1 - \sum_{s \in S} X_{ijs}) \quad \forall i \in I \quad \forall j \in I0$$

if ship s traverses arc (i, j), then port j must appear after port i in the visit sequence

8)

$$l_{js} \geq l_{is} - q_i - Q_s\left(1 - X_{ijs}\right) \quad \forall i \in I \quad \forall j \in I0 \quad \forall s \in S$$

if ship s travels from port i to port j, then the load at j must be at least the load at i minus the demand at i.

9)

$$l_{is} \leq L_{is} \quad \forall i \in I \quad \forall s \in S$$

the load of ship s upon entering port i must not exceed the port-specific loading limit L_{is}

10)

$$l_{0s} = \sum_{i \in I} q_i Y_{is} \quad \forall s \in S$$

the initial load of ship s equals the total demand of the ports it serves

VALID INEQUALITIES

These inequalities help the solver reduce solution time by eliminating infeasible or suboptimal routes early.

VI1)

$$X_{0js} \leq 1 - \frac{1}{TOT_q}\left(\sum_{i \in I} q_i Y_{is} - L_{js}\right) \quad \forall j \in I \quad \forall s \in S$$

for each port j, if the total load of the ship s, to which it has been assigned, is greater than the maximum allowed load for s to enter j, then j cannot be the first port visited in the route

VI2)

$$X_{ijs} = 0 \quad \forall i \in I \quad \forall j \in J \quad \forall s \in S \mid q_i + q_j > L_{is}$$

for each ship s and each pair of ports i and j, if the sum of their demand, q_i and q_j, is greater than the maximum allowed load for $s$to enter i, then j cannot be served immediately after i by ship s

VI3)

$$p_s - (u_i - 1)q_{big} \leq L_{is} + Q_{is}(1 - Y_{is}) \quad \forall i \in I \quad \forall s \in S$$

$$q_{big} = \max_{\mathrm{i} \in I} q_i$$

allows to identify the earliest position a port i can occupy in the visiting sequence, without violating draft limit constraints, given the ship s to which it has been assigned and the set of ports assigned to it

VI4)

$$u_i \leq I^* \quad \forall i \in I$$

that the latest position a port can assume in the visiting sequence is equal to the maximum number of ports that can be assigned simultaneously to the same ship, I^*.

I^* is computed by sorting ports in non-decreasing order of demand and counting how many can be assigned to the largest ship before exceeding its capacity.

Matheuristic

OVERVIEW

- The mathematical model efficiently solves only small instances
- For larger networks, the computational time grows exponentially
- To handle larger instances, two matheuristics are proposed:
  - **Large Neighborhood Search (LNS)**
  - **Iterated Local Search (ILS)**
- Both methods combine mathematical programming with local search principles

PARAMETERS TUNING

- Both the heuristics depend on two parameters
  1. m which is the number of nodes involved by the destroy operator
  2. α which is the proximity threshold
- The parameters were calibrated using the medium size instances
- The best resulting values are
  - $m = 5$
  - $\alpha = 1.5$
- The two parameters are uncorrelated

LARGE NEIGHBORHOOD SEARCH (LNS)

Main idea:

- Use a randomized operator to partially destroy the solution and exploit the mathematical model to optimally rebuild a feasible solution starting from the partial solution obtained
- The destroy operator is

$$\rho_i = \alpha v_i$$

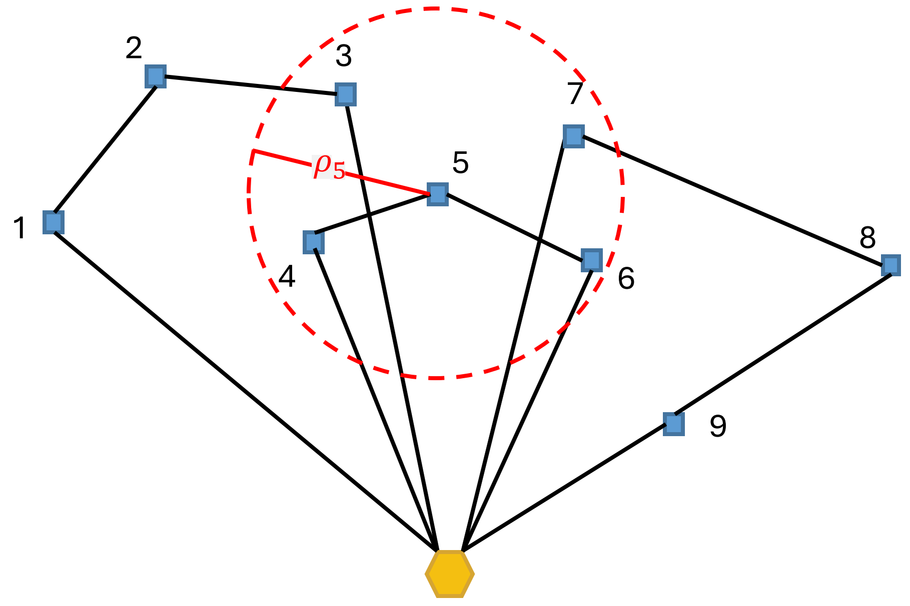
where v_i is the distance between the port i and its nearest port

Key advantage:

- A very large neighborhood can be efficiently explored at each itaration allowing to quickly move toward strongly better solution

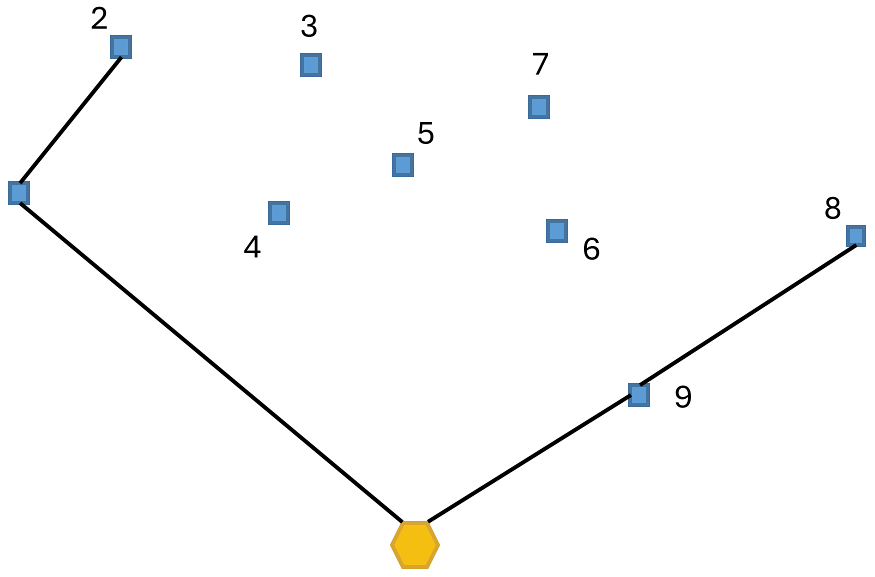
PROCEDURE

1. Run the mathematical model with a short time limit and keep the solution obtrained so far
2. At each iteration are selected m random ports and removed from the solution
3. The destroy operator removes all the ports within a certain radius
4. The partially destroyed solution is passed to the model
5. If the obained solution is better than the actual best solution, it is kept as current best, otherwise is discarded
6. The procedure is repeated until the stopping condition is reached



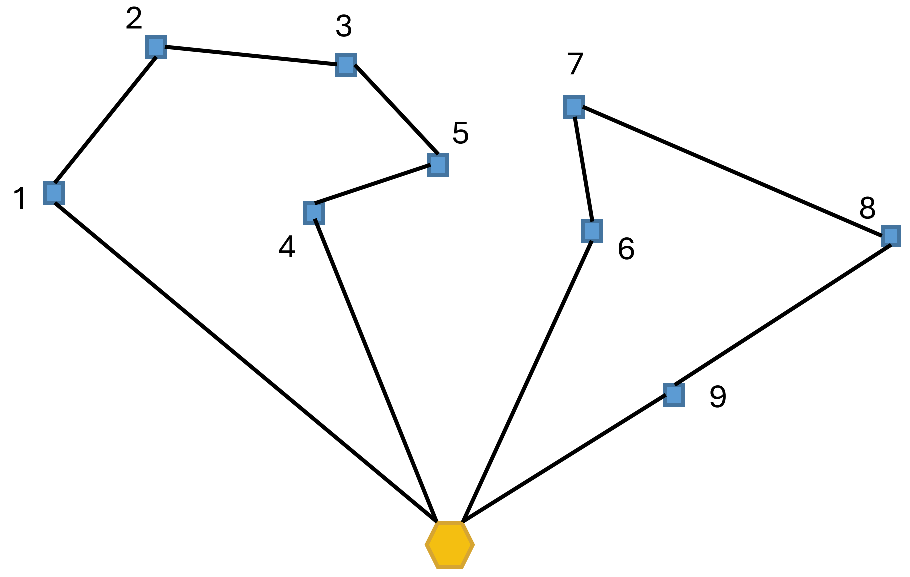
PROCEDURE

1. Run the mathematical model with a short time limit and keep the solution obtrained so far
2. At each iteration are selected m random ports and removed from the solution
3. The destroy operator removes all the ports within a certain radius
4. The partially destroyed solution is passed to the model
5. If the obained solution is better than the actual best solution, it is kept as current best, otherwise is discarded
6. The procedure is repeated until the stopping condition is reached



PROCEDURE

1. Run the mathematical model with a short time limit and keep the solution obtrained so far
2. At each iteration are selected m random ports and removed from the solution
3. The destroy operator removes all the ports within a certain radius
4. The partially destroyed solution is passed to the model
5. If the obained solution is better than the actual best solution, it is kept as current best, otherwise is discarded
6. The procedure is repeated until the stopping condition is reached



ITERATED LOCAL SEARCH (ILS)

Main idea:

- Combine a deterministic **Local Search (LS)** phase with a randomized **Diversification (DIV)** phase.

Algorithm outline:

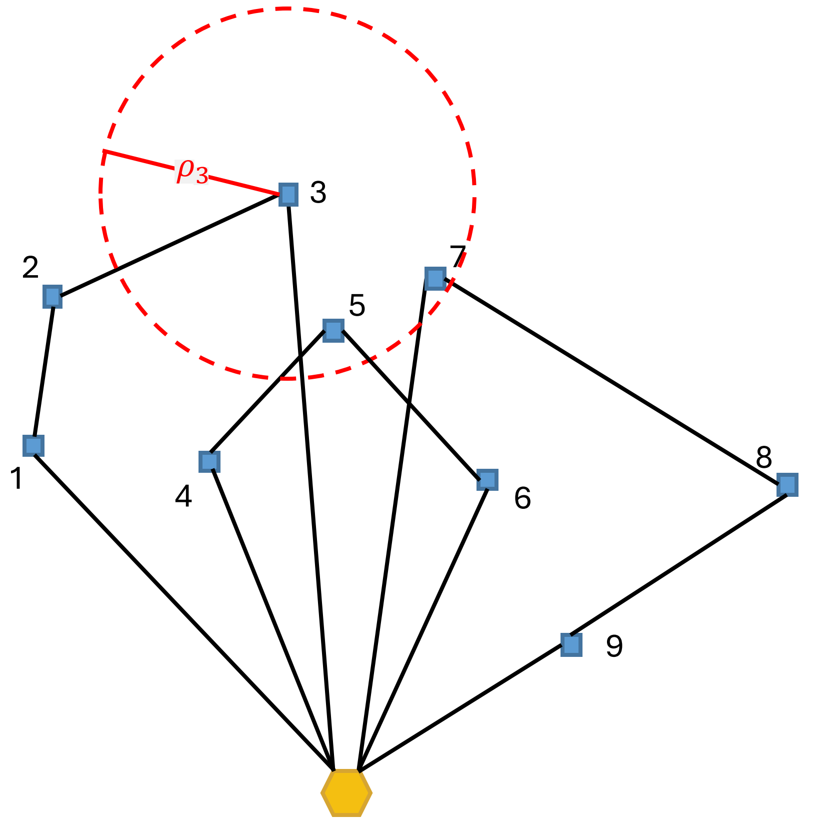
1. Run the mathematical model with a short time limit and keep the solution obtrained so far
2. Run the Local Search phase with the current solution
3. When the LS is trapped into a local minima Diversification phase is applied
4. LS and DIV are repeated until the stopping condition is reached

- **Key advantage:**
  More robust than LNS, better solution quality with similar computational time

LOCAL SEARCH

PROCEDURE

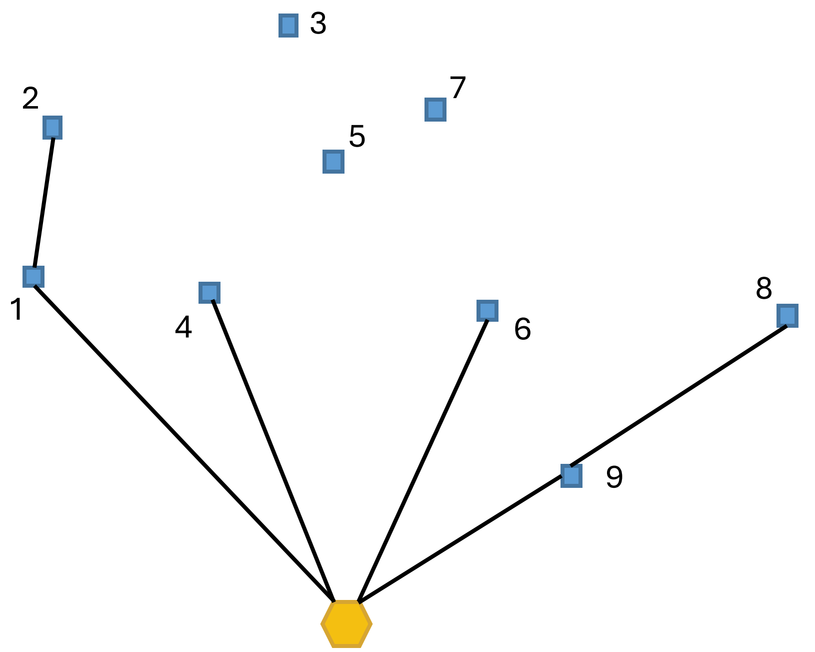
1. Every port is given a score $\sigma_i = \frac{\lambda_i}{v_i}$ where λ_i is the longest between the arc entering anche the arc exiting the port i
2. The m ports with the highest score are removed together with their neighborhood within a certain radius ρ_i
3. Use the mathematical model to optimally rebuild the partial destroyed one
4. Accept the new solution if it improves the objective function



LOCAL SEARCH

PROCEDURE

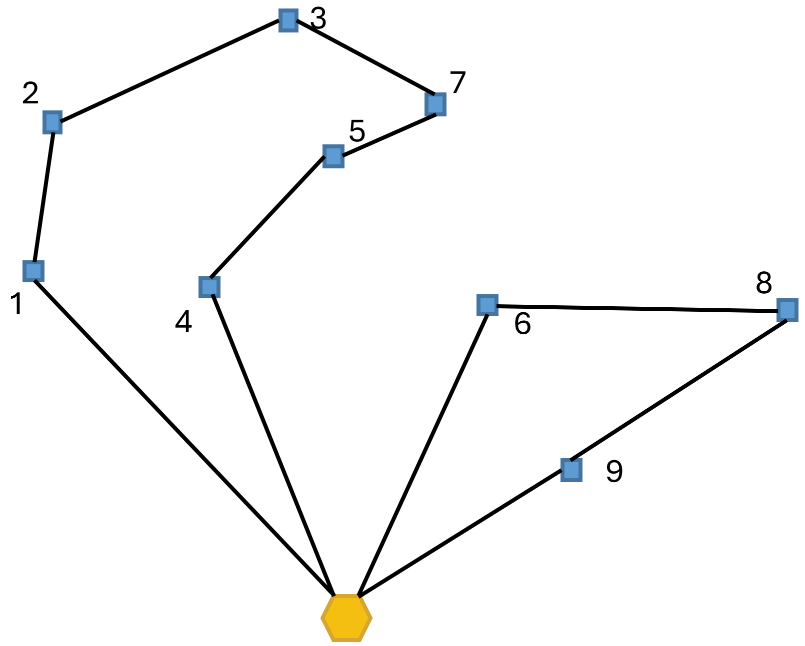
1. Every port is given a score $\sigma_i = \frac{\lambda_i}{v_i}$ where λ_i is the longest between the arc entering anche the arc exiting the port i
2. The m ports with the highest score are removed together with their neighborhood within a certain radius ρ_i
3. Use the mathematical model to optimally rebuild the partial destroyed one
4. Accept the new solution if it improves the objective function



LOCAL SEARCH

PROCEDURE

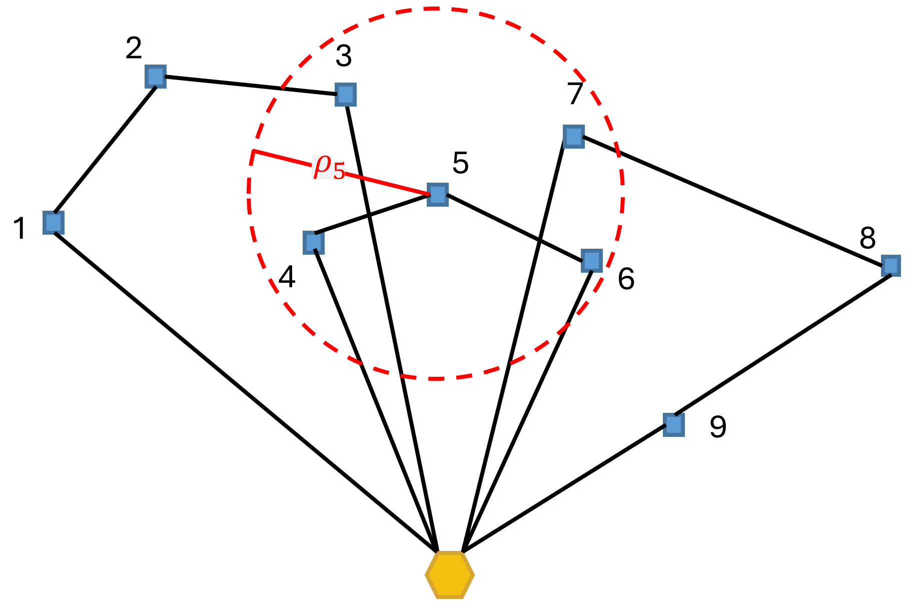
1. Every port is given a score $\sigma_i = \frac{\lambda_i}{v_i}$ where λ_i is the longest between the arc entering anche the arc exiting the port i
2. The m ports with the highest score are removed together with their neighborhood within a certain radius ρ_i
3. Use the mathematical model to optimally rebuild the partial destroyed one
4. Accept the new solution if it improves the objective function



DIVERSIFICATION

PROCEDURE

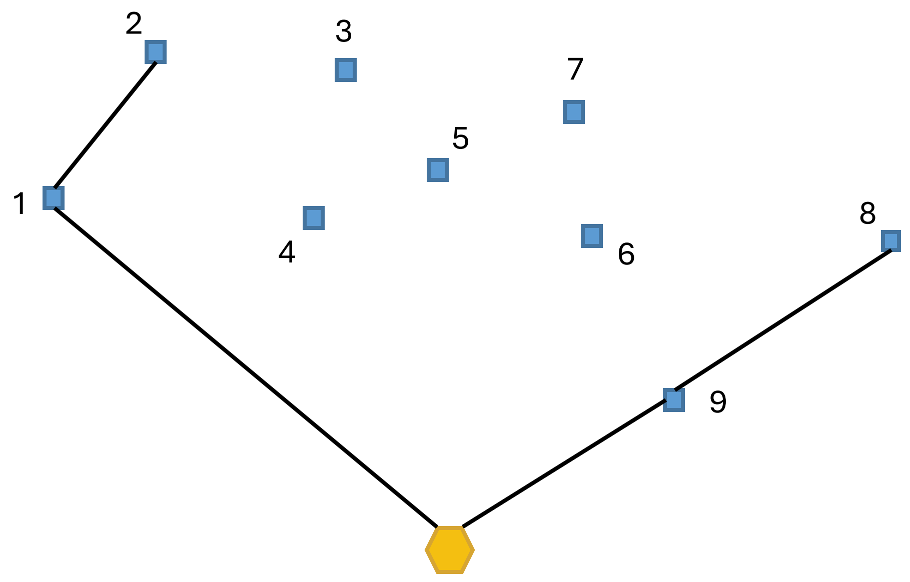
1. Randomly select m ports to remove from the best current solution
2. Include also nearby ports within a distance ρ_i
3. Use the mathematical model to optimally rebuild the partial destroyed one
4. The obtained solution, even if it is worse, is given as input to the LS



DIVERSIFICATION

PROCEDURE

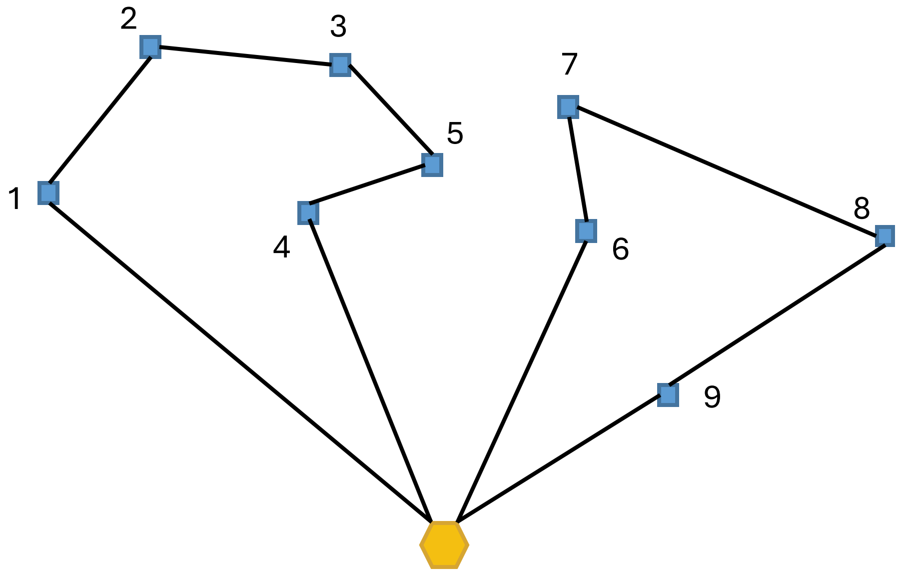
1. Randomly select m ports to remove from the best current solution
2. Include also nearby ports within a distance ρ_i
3. Use the mathematical model to optimally rebuild the partial destroyed one
4. The obtained solution, even if it is worse, is given as input to the LS



DIVERSIFICATION

PROCEDURE

1. Randomly select m ports to remove from the best current solution
2. Include also nearby ports within a distance ρ_i
3. Use the mathematical model to optimally rebuild the partial destroyed one
4. The obtained solution, even if it is worse, is given as input to the LS



Dataset

- Parameters
  - Number of ships $s \in [3, 5, 6, 10]$
  - Number of ports $\mathrm{i} \in [15, 25, 50]$
  - 2 scenarios for Draft Restriction (DR):
    - $LOW_{DR} = 30\%$ of ports affected by draft limits
    - $HIGH_{DR} = 70\%$ of ports affected by draft limits
  - 2 scenarios for Capacity Tightness ($CT = \frac{TOT_q}{TOT_Q}$)
    - $LOW_{CT} = 30\%$ of ship capacity corresponds to the total demand
    - $HIGH_{CT} = 70\%$ of ship capacity corresponds to the total demand

SMALL INSTANCES

- 40 small instaces divided in 4 sets of 10 instances each

  - **Set 1**:
    - 15 ports
    - 3 ships
    - $(LOW_{DR} - LOW_{CT})$
  - **Set 2**:
    - 15 ports
    - 3 ships
    - $(HIGH_{DR} - LOW_{CT})$
  - **Set 3**:
    - 15 ports
    - 3 ships
    - $(LOW_{DR} - HIGH_{CT})$
  - **Set 4**:
    - 15 ports
    - 3 ships
    - $(HIGH_{DR} - HIGH_{CT})$

MEDIUM & LARGE INSTANCES

- 22 medium instances divided in 2 sets of 11 instances each
  - **Set 5**:
    - 25 ports
    - 5 ships
    - $(HIGH_{DR} - HIGH_{CT})$
  - **Set 6**:
    - 25 ports
    - 6 ships
    - $(HIGH_{DR} - HIGH_{CT})$
- 10 large instances in 1 set
  - **Set 7**:
    - 50 ports
    - 10 ships
    - $(HIGH_{DR} - HIGH_{CT})$

Mathematical model – Small instances

			AVG COMPUTATIONAL TIME (s)								
SET	BEST OF	BEST GAP	NO VI	VI1	VI2	VI3	VI4	VI1+VI4	VI2+VI4	VI3+VI4	ALL VI
1	306,995	0,003	16,757	14,233	17,611	13,907	9,526	9,447	9,367	21,613	21,942
2	306,995	0,003	14,753	9,924	14,529	11,435	9,839	7,302	9,300	22,872	17,779
3	439,852	0,006	57,709	43,258	56,489	128,282	150,548	57,017	28,059	54,539	48,892
4	439,852	0,005	48,722	23,758	74,850	41,460	33,429	13,799	34,007	49,130	21,937

Mathematical model – Medium instances

			AVG COMPUTATIONAL TIME (s)								
SET	BEST OF	BEST GAP	NO VI	1	2	3	4	1+4	2+4	3+4	ALL VI
5	738,777	0,182	TL	TL	TL	TL	TL	TL	TL	TL	TL
6	692,710	0,221	TL	TL	TL	TL	TL	TL	TL	TL	TL

- The medium size instances were run with a 600s timelimit

Mathematical model – Large instances

			AVG COMPUTATIONAL TIME (s)								
SET	BEST OF	BEST GAP	NO VI	1	2	3	4	1+4	2+4	3+4	ALL VI
7	840,011	0,351	TL	TL	TL	TL	TL	TL	TL	TL	TL

- The large size instances were run with a 600s timelimit

Matheuristic

	LNS		ILS	
SET	AVG GAP	AVG TIME	AVG GAP	AVG TIME
1	0,030	20,754	0,019	23,063
2	0,019	16,386	0,018	18,991
3	0,012	31,604	0,020	35,950
4	0,027	28,875	0,020	45,278
5	0,047	172,255	0,013	125,760
6	0,005	137,114	0,031	286,325
7	0,012	245,695	0,015	339,918

Matheuristic VS Mathematical Model

	MODEL 1 + 4		LNS		ILS	
	AVG GAP	AVG TIME	AVG GAP	AVG TIME	AVG GAP	AVG TIME
1	0,004	9,447	0,030	20,754	0,019	23,063
2	0,004	7,302	0,019	16,386	0,018	18,991
3	0,006	57,017	0,012	31,604	0,020	35,950
4	0,005	13,799	0,027	28,875	0,020	45,278
5	0,182	TL	0,047	172,255	0,013	125,760
6	0,221	TL	0,005	137,114	0,031	286,325
7	0,351	TL	0,012	245,695	0,015	339,918

Conclusions