nature machine intelligence

Supplementary information

https://doi.org/10.1038/s42256-026-01307-8

Large language models discover complementary heuristics for combinatorial optimization

In the format provided by the authors and unedited

Contents

1 Port Scheduling Problem (PSP) 2
- 1.1 Problem description . . . 2
- 1.2 Notation . . . 3
  - 1.2.1 Sets and indices . . . 3
  - 1.2.2 Parameters . . . 3
  - 1.2.3 Decision variables . . . 3
- 1.3 Mathematical formulation . . . 4
  - 1.3.1 Objective . . . 4
  - 1.3.2 Constraints . . . 4
- 1.4 Instance generation . . . 5

2 Tugboat Routing and Scheduling Problems (TRS, MB-TRS, TRS-VS) 7
- 2.1 Shared problem setting and notation . . . 7
  - 2.1.1 Sets and indices . . . 7
  - 2.1.2 Parameters . . . 8
  - 2.1.3 Decision variables . . . 8
- 2.2 TRS: single-depot, fixed-speed formulation . . . 9
- 2.3 MB-TRS: multi-depot extension . . . 11
- 2.4 TRS-VS: variable-speed extension . . . 12
- 2.5 Summary and comparison . . . 14
- 2.6 Instance generation . . . 14

3 Baseline methods for the novel problems 16
- 3.1 Metaheuristic baselines . . . 16
  - 3.1.1 Common encoding and feasibility-preserving decoder . . . 16
  - 3.1.2 Search settings and per-algorithm operators . . . 16
- 3.2 Exact baseline (Gurobi) . . . 17

4 Automated Construction of the I–O–T Interface 19
- 4.1 Design principle: a verified contract, separated from algorithmic reasoning . . . 19
- 4.2 The four designer agents . . . 19
- 4.3 Execution-grounded validation and bounded repair . . . 20
- 4.4 Benchmark instantiation: what is adopted versus generated . . . 22

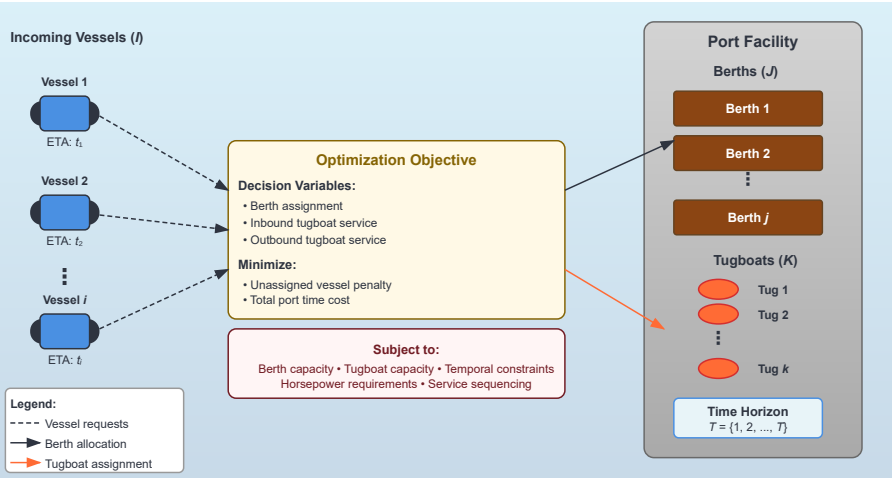
5 Ablation studies 22
- 5.1 Effect of the complementary selection strategy . . . 22
- 5.2 Effect of the evolution operators . . . 24
- 5.3 Effect of the per-instance runtime budget . . . 24
- 5.4 Effect of the number of evolution iterations . . . 25

Supplementary Note 1: Port Scheduling Problem (PSP)

1.1 Problem description

Maritime transport carries over 80% of world merchandise trade, and ports are among its most congested nodes: berth, vessel, and tugboat decisions are tightly coupled, so that poor coordination translates directly into vessel queuing, berth idling, and higher operating cost [1, 2]. The Port Scheduling Problem (PSP) is a novel combinatorial optimization problem that integrates these three decisions under a shared time horizon. A port operates $|\mathcal{J}|$ berths and $|\mathcal{K}|$ tugboats over a discrete horizon $\mathcal{T} = \{1, 2, \ldots, T\}$, during which $|\mathcal{I}|$ vessels request service (Supplementary Fig. 1). Each vessel must be assigned a berth, a berthing time, and tugboat services for both its inbound (arrival) and outbound (departure) maneuvers, so as to minimize total scheduling cost.

PSP couples spatial resources (berths), mobile resources (tugboats), and temporal resources (time slots) in a single model. Three structural features distinguish it from classical allocation problems: each vessel requires two asymmetric tugboat operations (inbound and outbound) with different durations and preparation times; several tugboats may be pooled on one operation to meet a horsepower requirement, introducing coalition-formation decisions; and berth, tugboat, and timing decisions cannot be decomposed without losing feasibility. This constraint structure does not match any standard benchmark and is therefore unlikely to appear in LLM pretraining corpora, making PSP a suitable test of zero-shot generalization. Given its scale, PSP is not intended to be solved to proven optimality; it serves as a testbed for feasibility, robustness, and quality–time trade-offs under practical runtime budgets.



Supplementary Figure 1: Schematic overview of PSP. Over a discrete time horizon, each incoming vessel must be simultaneously allocated a berth and the tugboats required for its inbound and outbound maneuvers, subject to capacity, temporal, and service-sequencing constraints. Decision variables determine berth assignments (x_{ijt}), inbound tugboat services (y^{in}_{ikt}), and outbound tugboat services (y^{out}_{ikt}); the objective minimizes unassigned-vessel penalties together with port-time, schedule-deviation, and tugboat-utilization costs.

1.2 Notation

1.2.1 Sets and indices

- $\mathcal{I} = \{1, \ldots, I\}$: vessels, indexed by i
- $\mathcal{J} = \{1, \ldots, J\}$: berths, indexed by j
- $\mathcal{K} = \{1, \ldots, K\}$: tugboats, indexed by k
- $\mathcal{T} = \{1, \ldots, T\}$: time periods, indexed by t

1.2.2 Parameters

- S_i, C_j: size class of vessel i and capacity class of berth j (vessel i fits berth j only if $C_j \geq S_i$)
- ETA_i: expected time of arrival of vessel i
- D_i: berthing duration of vessel i (number of consecutive periods)
- τ_i^{in}, τ_i^{out}: inbound and outbound tugboat service durations for vessel i
- ρ^{in}, ρ^{out}: tugboat preparation time after inbound and outbound service
- $\Delta_i^{\mathrm{early}}$, Δ_i^{late}: maximum early and late arrival deviation allowed for vessel i
- α_i, β_i, γ_i: priority weight, unit waiting cost, and unit just-in-time deviation cost of vessel i
- c_k: per-period usage cost of tugboat k
- P_k, P_i^{req}: horsepower of tugboat k and minimum horsepower required by vessel i
- $H_{\max}$: maximum number of tugboats per service operation
- ϵ_{time}: maximum allowable timing deviation in the service-sequencing constraints
- M: large penalty constant
- $\lambda_1, \lambda_2, \lambda_3, \lambda_4$: weights of the four objective components

1.2.3 Decision variables

- $x_{ijt} \in \{0, 1\}$: 1 if vessel i begins berthing at berth j at period t (defined only when $C_j \geq S_i$)
- $y_{ikt}^{\mathrm{in}} \in \{0, 1\}$: 1 if tugboat k starts inbound service for vessel i at period t
- $y_{ikt}^{\mathrm{out}} \in \{0, 1\}$: 1 if tugboat k starts outbound service for vessel i at period t
- $z_{it}^{\mathrm{in}}, z_{it}^{\mathrm{out}} \in \{0, 1\}$: auxiliary variables, 1 if vessel i begins inbound / outbound tugboat service at period t
- $u_i^{\mathrm{early}}, u_i^{\mathrm{late}} \geq 0$: early and late arrival deviation of vessel i relative to ETA_i

1.3 Mathematical formulation

1.3.1 Objective

The objective minimizes a weighted sum of four cost components,

$$\min Z = \lambda_1 Z_1 + \lambda_2 Z_2 + \lambda_3 Z_3 + \lambda_4 Z_4, \tag{1}$$

where Z_1 penalizes vessels that cannot be assigned to any berth, Z_2 measures the total time each vessel spends in the port system (from inbound service initiation to outbound service completion), Z_3 penalizes deviation from the expected arrival time, and Z_4 is the operational cost of tugboat deployment:

$$Z_1 = \sum_{i\in\mathcal{I}} M\alpha_i \Big(1 - \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} x_{ijt}\Big), \tag{2}$$

$$Z_2 = \sum_{i\in\mathcal{I}} \alpha_i\beta_i \Big[\sum_{t\in\mathcal{T}} (t+\tau_i^{\text{out}}) z_{it}^{\text{out}} - \sum_{t\in\mathcal{T}} t\, z_{it}^{\text{in}}\Big], \tag{3}$$

$$Z_3 = \sum_{i\in\mathcal{I}} \alpha_i\gamma_i \big(u_i^{\text{early}} + u_i^{\text{late}}\big), \tag{4}$$

$$Z_4 = \sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}} c_k \big(\tau_i^{\text{in}} y_{ikt}^{\text{in}} + \tau_i^{\text{out}} y_{ikt}^{\text{out}}\big). \tag{5}$$

1.3.2 Constraints

Each vessel is assigned to at most one berth at one period, and only to a berth of sufficient capacity:

$$\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} x_{ijt} \le 1, \qquad \forall i \in \mathcal{I}, \tag{6}$$

$$x_{ijt} = 0, \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T} : C_j < S_i. \tag{7}$$

Inbound and outbound tugboat services are provided if and only if the vessel is berthed:

$$\sum_{t\in\mathcal{T}} z_{it}^{\text{in}} = \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} x_{ijt}, \qquad \forall i \in \mathcal{I}, \tag{8}$$

$$\sum_{t\in\mathcal{T}} z_{it}^{\text{out}} = \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} x_{ijt}, \qquad \forall i \in \mathcal{I}. \tag{9}$$

The combined horsepower of assigned tugboats must meet the vessel requirement on each service:

$$\sum_{k\in\mathcal{K}} P_k y_{ikt}^{\text{in}} \ge P_i^{\text{req}} z_{it}^{\text{in}}, \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}, \tag{10}$$

$$\sum_{k\in\mathcal{K}} P_k y_{ikt}^{\text{out}} \ge P_i^{\text{req}} z_{it}^{\text{out}}, \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}. \tag{11}$$

The number of tugboats per service cannot exceed the system limit, and the auxiliary service indicators are linked to tugboat deployment:

$$\sum_{k\in\mathcal{K}} y^{\text{in}}_{ikt} \le H_{\max}\, z^{\text{in}}_{it}, \qquad \forall i\in\mathcal{I}, t\in\mathcal{T}, \tag{12}$$

$$\sum_{k\in\mathcal{K}} y^{\text{out}}_{ikt} \le H_{\max}\, z^{\text{out}}_{it}, \qquad \forall i\in\mathcal{I}, t\in\mathcal{T}, \tag{13}$$

$$z^{\text{in}}_{it} \le \sum_{k\in\mathcal{K}} y^{\text{in}}_{ikt}, \qquad \forall i\in\mathcal{I}, t\in\mathcal{T}, \tag{14}$$

$$z^{\text{out}}_{it} \le \sum_{k\in\mathcal{K}} y^{\text{out}}_{ikt}, \qquad \forall i\in\mathcal{I}, t\in\mathcal{T}. \tag{15}$$

Each berth holds at most one vessel at any time, accounting for berthing duration, and each tugboat performs at most one operation at a time, accounting for service plus preparation time:

$$\sum_{i\in\mathcal{I}} \sum_{\tau=\max(1,\,t-D_i+1)}^{t} x_{ij\tau} \le 1, \quad \forall j\in\mathcal{J}, t\in\mathcal{T}, \tag{16}$$

$$\sum_{i\in\mathcal{I}} \Big(\sum_{\tau=\max(1,\,t-\tau^{\text{in}}_i-\rho^{\text{in}}+1)}^{t} y^{\text{in}}_{ik\tau} + \sum_{\tau=\max(1,\,t-\tau^{\text{out}}_i-\rho^{\text{out}}+1)}^{t} y^{\text{out}}_{ik\tau} \Big) \le 1, \quad \forall k\in\mathcal{K}, t\in\mathcal{T}. \tag{17}$$

Inbound tugboat services may only start within the allowable window around the expected arrival time:

$$y^{\text{in}}_{ikt} = 0, \quad \forall i\in\mathcal{I}, k\in\mathcal{K},\ t < \text{ETA}_i - \Delta^{\text{early}}_i \text{ or } t > \text{ETA}_i + \Delta^{\text{late}}_i. \tag{18}$$

Service sequencing requires inbound service to complete just before berthing begins, and outbound service to begin just after berthing completes, each within a tolerance ϵ_{time}:

$$0 \le \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} t\, x_{ijt} - \sum_{t\in\mathcal{T}} (t+\tau^{\text{in}}_i)\, z^{\text{in}}_{it} \le \epsilon_{\text{time}} \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} x_{ijt}, \qquad \forall i\in\mathcal{I}, \tag{19}$$

$$0 \le \sum_{t\in\mathcal{T}} t\, z^{\text{out}}_{it} - \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} (t+D_i)\, x_{ijt} \le \epsilon_{\text{time}} \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} x_{ijt}, \qquad \forall i\in\mathcal{I}. \tag{20}$$

The arrival-time deviation is linearized into early and late components:

$$\sum_{t\in\mathcal{T}} t\, z^{\text{in}}_{it} = \text{ETA}_i + u^{\text{late}}_i - u^{\text{early}}_i, \quad \forall i\in\mathcal{I}. \tag{21}$$

Finally, the variable domains are

$$\begin{aligned} x_{ijt},\, y^{\text{in}}_{ikt},\, y^{\text{out}}_{ikt},\, z^{\text{in}}_{it},\, z^{\text{out}}_{it} \in \{0,1\}, \qquad & \forall i\in\mathcal{I}, j\in\mathcal{J}, k\in\mathcal{K}, t\in\mathcal{T}, \\ u^{\text{early}}_i,\, u^{\text{late}}_i \ge 0, \qquad & \forall i\in\mathcal{I}. \end{aligned} \tag{22}$$

1.4 Instance generation

Instances follow the naming convention `{vessels}_{berths}_{tugboats}_T{horizon}`, where the horizon sets the temporal granularity ($T = 24$: 1-hour periods; $T = 48$:

30-minute periods). Vessels are drawn from four types (container, bulk, tanker, general cargo) with size classes 1–5 and corresponding horsepower requirements of 800–6000 HP; expected arrival times follow a clustered pattern (70% in early periods, 30% spread over the remaining horizon). Berth capacities range over 2–5, with roughly half able to accommodate large vessels (capacity ≥ 4). Tugboat horsepower ranges over 800–5000 HP across four categories (small 800–1500, medium 1500–2500, medium-large 2500–3500, large 3500–5000 HP), so that high-horsepower vessels can be served by combining 2–3 tugboats. The penalty constant M and the weights $\lambda_1, \ldots, \lambda_4$ are calibrated per instance so that the four objective components contribute comparably. Instances span three scales—small (100–200 vessels), medium (250–300), and large (400–500)—with proportional resource allocation to maintain consistent capacity ratios. All instances are available in the supplementary repository.

Supplementary Note 2: Tugboat Routing and Scheduling Problems (TRS, MB-TRS, TRS-VS)

The second group of novel problems addresses tugboat dispatch in container-terminal logistics. We define three progressively complex variants that share a common constraint foundation: **TRS** (Tugboat Routing and Scheduling), the single-depot fixed-speed baseline; **MB-TRS**, which extends TRS with multiple geographically dispersed tugboat bases and mandatory home-base return; and **TRS-VS**, which extends TRS with variable tugboat speed and the resulting speed-dependent fuel coupling. MB-TRS and TRS-VS each extend TRS along a single operational direction, so we present TRS in full and then describe the two extensions incrementally. A comparison of the three variants is given in Supplementary Table 2.

2.1 Shared problem setting and notation

A port operator manages a fleet of tugboats to serve a set of scheduled vessel maneuvering tasks within a planning horizon $[0, T^{\max}]$. Each task s represents a maneuvering operation (e.g., berthing or unberthing) in which one or more tugboats escort a vessel along a service leg from an entry position $\mathrm{loc}_s^{\mathrm{start}}$ to an exit position $\mathrm{loc}_s^{\mathrm{end}}$, starting within a time window $[a_s, b_s]$ and completing no later than $T^{\max}$. Feasibility of a task is governed by two requirements on its assigned tugboats: their combined horsepower must meet a minimum threshold $H_s^{\min}$ (to guarantee safe towing power), and their number must not exceed Γ_s (to avoid congestion in confined waterways). Unserved tasks incur a penalty of weight W.

Fuel consumption has two parts. The transit fuel coefficient β_k captures fuel burn per unit power per unit time while a tugboat repositions between tasks. The service fuel coefficient depends jointly on the tugboat and the assisted vessel, since larger vessels impose greater hydrodynamic resistance; each task s therefore carries a vessel class $c(s) \in C$, and the service coefficient is $\alpha_{k,c(s)}$.

Each maneuvering operation is modeled as a single node in a routing graph (Supplementary Fig. 2). Because the positional information of a task is fully captured by the pair $(\mathrm{loc}_s^{\mathrm{start}}, \mathrm{loc}_s^{\mathrm{end}})$, the task collapses to a node whose entry attribute is $\mathrm{loc}_s^{\mathrm{start}}$ and exit attribute is $\mathrm{loc}_s^{\mathrm{end}}$. A directed arc (i, j) represents the transit leg along which a tugboat repositions from $\mathrm{loc}_i^{\mathrm{end}}$ to $\mathrm{loc}_j^{\mathrm{start}}$. The single depot is represented by a virtual departure node 0 and a virtual return node $n+1$, both at $\mathrm{loc}^{\mathrm{base}}$, giving the node set $\mathcal{V} = \{0\} \cup S \cup \{n+1\}$. Tugboat scheduling then reduces to finding, for each tugboat, a directed path from node 0 through a feasible subset of task nodes to node $n+1$ satisfying all temporal, horsepower, and fuel constraints.

2.1.1 Sets and indices

- $S = \{1, \ldots, n\}$: vessel maneuvering tasks, indexed by s (and by i, j when viewed as routing nodes), with $n = |S|$

- $K = \{1, \ldots, |K|\}$: tugboats, indexed by k
- $C = \{1, \ldots, m\}$: vessel classes (by size), with $c(s) \in C$ the class of task s
- $\mathcal{V} = \{0\} \cup S \cup \{n+1\}$: routing nodes, with virtual depot nodes 0 (departure) and $n+1$ (return)

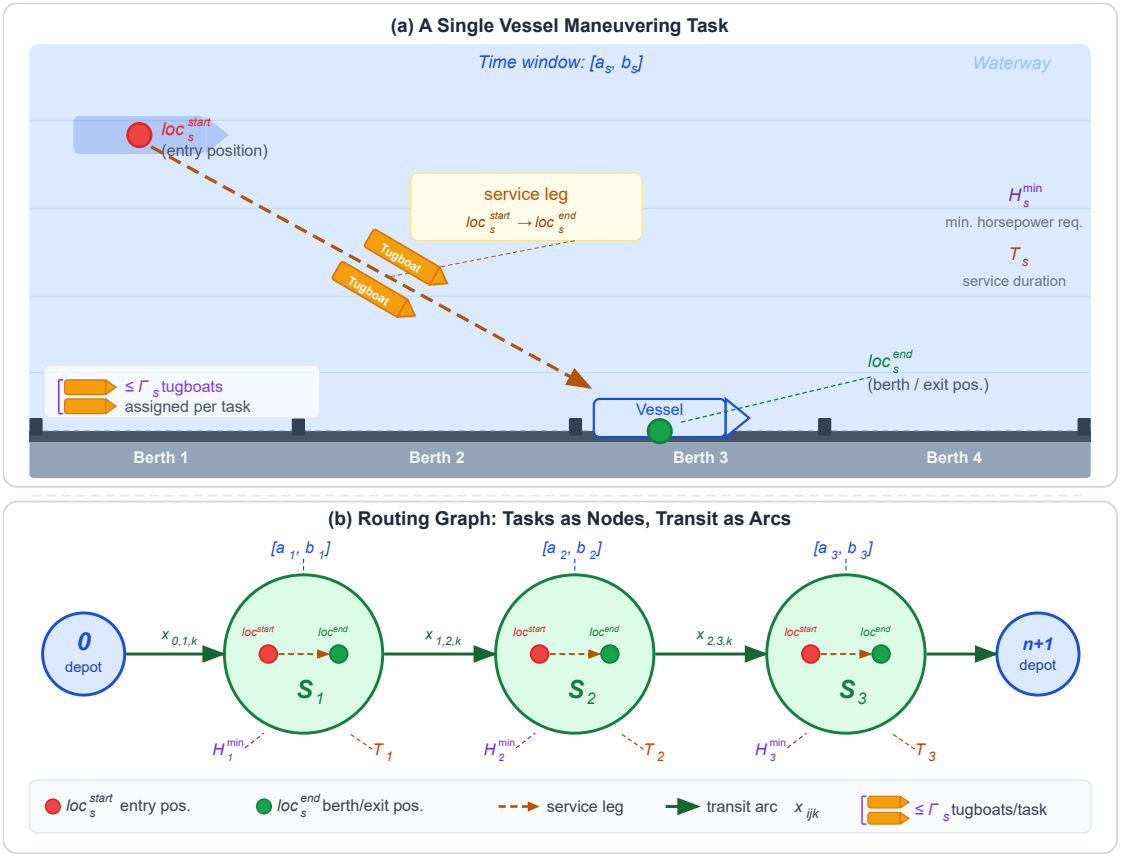
2.1.2 Parameters

- $T^{\max}$: planning horizon; $[a_s, b_s]$: service time window of task s
- $\text{loc}_s^{\text{start}}$, $\text{loc}_s^{\text{end}}$, loc^{base}: entry, exit, and depot locations
- T_s: service duration of task s (fixed in TRS and MB-TRS; speed-dependent in TRS-VS)
- $H_s^{\min}$: minimum combined horsepower required by task s; Γ_s: maximum tugboats per task
- W: penalty per unserved task
- P_k: rated horsepower of tugboat k; $F_k^{\max}$: fuel capacity of tugboat k
- $\alpha_{k,c}$: service fuel coefficient (tugboat k assisting a class-c vessel); β_k: transit fuel coefficient
- v: fixed tugboat speed (10 knots) in TRS and MB-TRS
- $d_{i,j}$, t_{ij}: inter-node distance and travel time
- M: big-M constant

2.1.3 Decision variables

- $z_s \in \{0, 1\}$: 1 if task s is served
- $y_{sk} \in \{0, 1\}$: 1 if tugboat k is assigned to task s
- $x_{ijk} \in \{0, 1\}$: 1 if tugboat k traverses arc (i, j), i.e., serves j immediately after i
- $\tau_s \geq 0$: service start time of task s

Variant-specific variables are introduced in the corresponding subsections: speed-selection variables $u_s^\ell, p_{ijk}^\ell, w_{sk}^\ell$ for TRS-VS, and home-base routing variables $x_{b,s}^k, x_{s,n+|b|}^k, x_{i,j}^k$ for MB-TRS.



Supplementary Figure 2: (a) A single vessel maneuvering task, characterized by a service leg from entry position $\mathrm{loc}_s^{\mathrm{start}}$ to exit position $\mathrm{loc}_s^{\mathrm{end}}$, a time window $[a_s, b_s]$, a minimum horsepower requirement $H_s^{\min}$, a maximum tugboat count Γ_s, and a service duration T_s. (b) The routing-graph abstraction: each task collapses to a single node with entry and exit attributes; directed arcs represent tugboat transit legs between consecutive tasks; virtual depot nodes 0 and $n+1$ anchor each tugboat's route.

2.2 TRS: single-depot, fixed-speed formulation

TRS considers a single depot at $\mathrm{loc}^{\mathrm{base}}$, from which all tugboats depart and to which they return, traveling at a fixed speed $v = 10$ knots. Three Euclidean distances capture the spatial structure—depot to task start, task end to task start, and task end back to depot—from which the service duration T_s and inter-task travel time t_{ij} are derived:

$$d_{0,s} = \left\|\mathrm{loc}^{\mathrm{base}} - \mathrm{loc}_s^{\mathrm{start}}\right\|, \quad d_{i,j} = \left\|\mathrm{loc}_i^{\mathrm{end}} - \mathrm{loc}_j^{\mathrm{start}}\right\|, \quad \forall s \in S,\ i, j \in S, \tag{23}$$

$$d_{s,n+1} = \left\|\mathrm{loc}_s^{\mathrm{end}} - \mathrm{loc}^{\mathrm{base}}\right\|, \quad d_s = \left\|\mathrm{loc}_s^{\mathrm{end}} - \mathrm{loc}_s^{\mathrm{start}}\right\|, \quad \forall s \in S, \tag{24}$$

$$T_s = \frac{d_s}{v}, \qquad t_{ij} = \frac{d_{ij}}{v}, \qquad \forall s \in S,\ i \in \{0\} \cup S,\ j \in S \cup \{n+1\}. \tag{25}$$

The objective minimizes total fuel consumption plus the unserved-task penalty:

$$\min\ Z = Z_{\mathrm{service}} + Z_{\mathrm{transit}} + W \cdot Z_{\mathrm{penalty}}, \tag{26}$$

where Z_{service} is service fuel (vessel-class-dependent coefficient $\alpha_{k,c(s)}$, horsepower P_k, duration T_s), Z_{transit} is transit fuel (coefficient β_k, travel time t_{ij}), and Z_{penalty}

counts unserved tasks:

$$Z_{\text{service}} = \sum_{s \in S} \sum_{k \in K} y_{sk} \cdot \alpha_{k,c(s)} \cdot P_k \cdot T_s, \tag{27}$$

$$Z_{\text{transit}} = \sum_{i \in \{0\} \cup S} \sum_{j \in S \cup \{n+1\}} \sum_{k \in K} x_{ijk} \cdot \beta_k \cdot P_k \cdot t_{ij}, \tag{28}$$

$$Z_{\text{penalty}} = \sum_{s \in S} (1 - z_s). \tag{29}$$

Task-execution constraints cap the number of assigned tugboats and enforce the horsepower threshold for each served task:

$$\sum_{k \in K} y_{sk} \leq \Gamma_s \cdot z_s, \qquad \forall s \in S, \tag{30}$$

$$\sum_{k \in K} y_{sk} \cdot P_k \geq H_s^{\min} \cdot z_s, \qquad \forall s \in S. \tag{31}$$

Routing constraints (32)–(33) allow each tugboat at most one departure from and one return to the depot, (34) enforces flow conservation at each task node, and (35) links assignment to routing:

$$\sum_{j \in S} x_{0jk} \leq 1, \qquad \forall k \in K, \tag{32}$$

$$\sum_{i \in S} x_{i,n+1,k} \leq 1, \qquad \forall k \in K, \tag{33}$$

$$\sum_{i \in \{0\} \cup S} x_{ijk} = \sum_{i \in S \cup \{n+1\}} x_{jik}, \qquad \forall j \in S,\ k \in K, \tag{34}$$

$$y_{sk} = \sum_{i \in \{0\} \cup S} x_{isk}, \qquad \forall s \in S,\ k \in K. \tag{35}$$

Temporal constraints enforce the time window (36), completion within the horizon (37), and time propagation from the depot (38) and between consecutive tasks (39); unserved tasks are relaxed via big-M:

$$a_s \leq \tau_s + M(1 - z_s), \quad \tau_s \leq b_s + M(1 - z_s), \qquad \forall s \in S, \tag{36}$$

$$\tau_s + T_s \leq T^{\max} + M(1 - z_s), \qquad \forall s \in S, \tag{37}$$

$$\tau_j \geq t_{0j} - M(1 - x_{0jk}), \qquad \forall j \in S,\ k \in K, \tag{38}$$

$$\tau_j \geq \tau_i + T_i + t_{ij} - M(1 - x_{ijk}), \quad \forall i, j \in S,\ k \in K. \tag{39}$$

The fuel-capacity constraint bounds each tugboat's total (service plus transit) fuel by its onboard capacity:

$$\sum_{s \in S} y_{sk} \cdot \alpha_{k,c(s)} \cdot P_k \cdot T_s + \sum_{i \in \{0\} \cup S} \sum_{j \in S \cup \{n+1\}} x_{ijk} \cdot \beta_k \cdot P_k \cdot t_{ij} \leq F_k^{\max}, \quad \forall k \in K. \tag{40}$$

Domain constraints specify variable types:

$$x_{ijk} \in \{0, 1\}, \quad y_{sk} \in \{0, 1\}, \quad z_s \in \{0, 1\}, \quad \tau_s \geq 0. \tag{41}$$

2.3 MB-TRS: multi-depot extension

In large ports, tugboat fleets are stationed across several geographically dispersed bases to reduce transit distance and improve responsiveness, with each tugboat required to depart from and return to its designated home base. MB-TRS replaces the single-depot routing structure of TRS with a multi-depot one while keeping the speed fixed at $v = 10$ knots. It retains the task-execution constraints (30)–(31), the time-window bounds (36), the inter-task time propagation (39), and the horizon constraint (37) without modification; the routing constraints (32)–(35), the temporal departure constraint (38), and the fuel-capacity constraint (40) are replaced by their multi-depot counterparts below.

Let $B = \{-1, -2, \ldots, -p\}$ be the set of depots, indexed by negative integers. Each depot $b \in B$ has a location loc_b and a capacity δ_b (maximum tugboats departing from or returning to it within the horizon). The fleet is partitioned as $K = \bigcup_{b \in B} K_b$ with $K_b \cap K_{b'} = \emptyset$ for $b \neq b'$; tugboat $k \in K_b$ belongs to depot b and must start and end its shift there. The departure node of depot b is node b itself, and its return node is node $n + |b|$; with $B_d = \{n+1, \ldots, n+p\}$ the set of return nodes, the node set becomes $V = B \cup S \cup B_d$. Depot-related distances and travel times are

$$d_{b,s} = \|\mathrm{loc}_b - \mathrm{loc}_s^{\mathrm{start}}\|, \qquad \forall b \in B,\ s \in S, \tag{42}$$

$$d_{s,n+|b|} = \|\mathrm{loc}_s^{\mathrm{end}} - \mathrm{loc}_b\|, \qquad \forall s \in S,\ b \in B, \tag{43}$$

with $t_{b,s} = d_{b,s}/v$ and $t_{s,n+|b|} = d_{s,n+|b|}/v$. For tugboat $k \in K_b$, three routing variables make depot membership explicit: $x^k_{b,s} \in \{0,1\}$ (departs home depot b to task s), $x^k_{s,n+|b|} \in \{0,1\}$ (returns from task s to home depot b), and $x^k_{i,j} \in \{0,1\}$ for $i, j \in S$, $i \neq j$ (proceeds from task i to task j).

The transit fuel term decomposes into outbound, inter-task, and inbound components:

$$Z_{\mathrm{transit}} = Z_{\mathrm{out}} + Z_{\mathrm{mid}} + Z_{\mathrm{in}}, \tag{44}$$

$$Z_{\mathrm{out}} = \sum_{b \in B} \sum_{k \in K_b} \sum_{s \in S} x^k_{b,s} \cdot \beta_k \cdot P_k \cdot t_{b,s}, \tag{45}$$

$$Z_{\mathrm{mid}} = \sum_{i \in S} \sum_{j \in S} \sum_{k \in K} x^k_{i,j} \cdot \beta_k \cdot P_k \cdot t_{i,j}, \tag{46}$$

$$Z_{\mathrm{in}} = \sum_{b \in B} \sum_{k \in K_b} \sum_{s \in S} x^k_{s,n+|b|} \cdot \beta_k \cdot P_k \cdot t_{s,n+|b|}. \tag{47}$$

Home-base departure/return constraints limit each tugboat to at most one departure and one return with respect to its home depot and forbid use of any other depot's nodes:

$$\sum_{s \in S} x^k_{b,s} \leq 1, \qquad \forall b \in B,\ k \in K_b, \tag{48}$$

$$\sum_{s \in S} x^k_{s,n+|b|} \leq 1, \qquad \forall b \in B,\ k \in K_b, \tag{49}$$

$$x^k_{b,s} = 0 \ \text{ and } \ x^k_{s,n+|b|} = 0, \qquad \forall b \in B,\ k \notin K_b,\ s \in S. \tag{50}$$

Flow conservation and the service-linkage constraint connect routing to assignment

for each tugboat $k \in K_b$ (with $b(k)$ its home depot):

$$\sum_{i \in \{b(k)\} \cup S} x^k_{i,j} = \sum_{i \in S \cup \{n+|b(k)|\}} x^k_{j,i}, \qquad \forall j \in S,\ k \in K, \tag{51}$$

$$y_{sk} = \sum_{i \in \{b(k)\} \cup S} x^k_{i,s}, \qquad \forall s \in S,\ k \in K. \tag{52}$$

Depot-capacity constraints limit departures and returns at each depot:

$$\sum_{k \in K_b} \sum_{s \in S} x^k_{b,s} \le \delta_b, \qquad \forall b \in B, \tag{53}$$

$$\sum_{k \in K_b} \sum_{s \in S} x^k_{s,n+|b|} \le \delta_b, \qquad \forall b \in B. \tag{54}$$

The temporal departure constraint uses depot-specific travel times:

$$\tau_j \ge t_{b(k),j} - M\big(1 - x^k_{b(k),j}\big), \quad \forall j \in S,\ k \in K. \tag{55}$$

The fuel-capacity constraint aggregates consumption over all three arc types:

$$\sum_{s \in S} y_{sk}\, \alpha_{k,c(s)} P_k T_s + \sum_{s \in S} x^k_{b(k),s}\, \beta_k P_k t_{b(k),s} + \sum_{i,j \in S} x^k_{i,j}\, \beta_k P_k t_{i,j} + \sum_{s \in S} x^k_{s,n+|b(k)|}\, \beta_k P_k t_{s,n+|b(k)|} \le F^{\max}_k, \quad \forall k \in K. \tag{56}$$

MB-TRS expands the node set from $n+2$ to $n+2p$ nodes, and the mandatory-return constraint (49) together with the cross-depot prohibition (50) forces each tugboat's route to form a closed tour anchored at its home depot, substantially tightening the feasible region relative to TRS.

2.4 TRS-VS: variable-speed extension

TRS-VS targets port decarbonization, a binding priority under the IMO 2023 GHG Strategy [3]: because propulsive power scales approximately with the cube of speed [4], a modest speed reduction yields a disproportionately large fuel saving, but slower service extends durations and may violate time windows or the horizon. A continuous speed variable would couple nonlinearly with routing and assignment, yielding an intractable non-convex MINLP [5, 6]; real tugboat engines also operate at a few discrete load settings [7]. We therefore discretize speed into a finite set $L = \{\text{slow}, \text{medium}, \text{fast}\}$, which keeps all speed-dependent quantities precomputable as constants and preserves the MILP structure [8]. TRS-VS keeps the speed fixed-depot structure of TRS and retains its task-execution constraints (30)–(31), routing constraints (32)–(35), time-window bounds (36), and domain constraints (41); constraints (37)–(39) and (40) are updated for speed dependence, as detailed below.

The speeds $v^\ell \in \{8, 10, 12\}$ knots span the economically meaningful range of harbour tugboats, with the medium level matching the fixed speed of TRS and serving as the normalization reference [6, 9]. The power coefficient at level ℓ follows the cubic speed–power law,

$$\rho^\ell = \left(\frac{v^\ell}{v^{\text{medium}}} \right)^3, \tag{57}$$

Supplementary Table 1: Speed levels and power coefficients for TRS-VS.

Level ℓ	**Speed** v^ℓ (knots)	**Power coefficient** ρ^ℓ
Slow	8	0.512
Medium	10	1.000
Fast	12	1.728

giving the values in Supplementary Table 1.

Service durations, transit times, and per-leg fuel consumption are pre-computed for each speed level:

$$T_s^\ell = \frac{d_s}{v^\ell}, \quad t_{ij}^\ell = \frac{d_{ij}}{v^\ell}, \qquad \forall s \in S,\ \forall i,j,\ \forall \ell \in L, \tag{58}$$

$$\psi_s^{k\ell} = \alpha_{k,c(s)} \rho^\ell P_k T_s^\ell = \alpha_{k,c(s)} P_k d_s \frac{(v^\ell)^2}{(v^{\text{medium}})^3}, \qquad \forall s,k,\ell, \tag{59}$$

$$\phi_{ij}^{k\ell} = \beta_k \rho^\ell P_k t_{ij}^\ell = \beta_k P_k d_{ij} \frac{(v^\ell)^2}{(v^{\text{medium}})^3}, \qquad \forall i,j,k,\ell. \tag{60}$$

Two binary variables capture speed selection: $u_s^\ell = 1$ if task s is served at level ℓ, and $p_{ijk}^\ell = 1$ if tugboat k traverses (i,j) at level ℓ. Auxiliary variables $w_{sk}^\ell = y_{sk}\, u_s^\ell$ linearize the bilinear product appearing in the objective and fuel constraint. The objective retains the form of (26), with speed-weighted fuel terms:

$$Z_{\text{service}} = \sum_{s \in S} \sum_{k \in K} \sum_{\ell \in L} w_{sk}^\ell \cdot \psi_s^{k\ell}, \tag{61}$$

$$Z_{\text{transit}} = \sum_{i \in \{0\} \cup S} \sum_{j \in S \cup \{n+1\}} \sum_{k \in K} \sum_{\ell \in L} p_{ijk}^\ell \cdot \phi_{ij}^{k\ell}. \tag{62}$$

Speed-selection constraints assign exactly one level to each traversed arc and each served task:

$$\sum_{\ell \in L} p_{ijk}^\ell = x_{ijk}, \qquad \forall i \in \{0\} \cup S,\ j \in S \cup \{n+1\},\ k \in K, \tag{63}$$

$$\sum_{\ell \in L} u_s^\ell = z_s, \qquad \forall s \in S. \tag{64}$$

McCormick constraints enforce $w_{sk}^\ell = y_{sk}\, u_s^\ell$:

$$w_{sk}^\ell \le y_{sk}, \qquad \forall s \in S,\ k \in K,\ \ell \in L, \tag{65}$$

$$w_{sk}^\ell \le u_s^\ell, \qquad \forall s \in S,\ k \in K,\ \ell \in L, \tag{66}$$

$$w_{sk}^\ell \ge y_{sk} + u_s^\ell - 1, \qquad \forall s \in S,\ k \in K,\ \ell \in L. \tag{67}$$

The time-propagation constraints (37)–(39) are updated for speed-dependent du-

rations:

$$\tau_s + \sum_{\ell \in L} u_s^\ell \, T_s^\ell \le T^{\max} + M(1 - z_s), \qquad \forall s \in S, \tag{68}$$

$$\tau_j \ge \sum_{\ell \in L} p_{0jk}^\ell \, t_{0j}^\ell - M(1 - x_{0jk}), \qquad \forall j \in S,\ k \in K, \tag{69}$$

$$\tau_j \ge \tau_i + \sum_{\ell \in L} u_i^\ell \, T_i^\ell + \sum_{\ell \in L} p_{ijk}^\ell \, t_{ij}^\ell - M(1 - x_{ijk}), \quad \forall i, j \in S,\ k \in K. \tag{70}$$

The fuel-capacity constraint aggregates speed-dependent consumption:

$$\sum_{s \in S} \sum_{\ell \in L} w_{sk}^\ell \cdot \psi_s^{k\ell} + \sum_{i,j} \sum_{\ell \in L} p_{ijk}^\ell \cdot \phi_{ij}^{k\ell} \le F_k^{\max}, \quad \forall k \in K. \tag{71}$$

After linearization, TRS-VS is an MILP with $O(n \cdot |L|)$ additional binary variables relative to TRS, encoding the central trade-off: faster service helps meet tight time windows but raises fuel consumption cubically, whereas slower service conserves fuel at the risk of time-window violations.

2.5 Summary and comparison

Supplementary Table 2 summarizes the structural relationship among the three variants. MB-TRS and TRS-VS each extend TRS in a single operational direction while sharing its task-execution and core temporal feasibility constraints, introducing new variables and constraints only where the operational reality demands. This incremental structure makes the family a natural testbed for automated heuristic design under evolving problem specifications.

Supplementary Table 2: Structural comparison of the three tugboat variants. Inherited constraints refer to TRS constraint numbers retained without modification.

Feature	**TRS**	**MB-TRS**	**TRS-VS**
Depot structure	Single	Multi	Single
Speed	Fixed	Fixed	Variable
Inherited constraints	Baseline	(30)–(31), (36)–(37), (39)	(30)–(36), (41)
New constraints	—	(48)–(56)	(63)–(71)
Additional variables	—	$x_{b,s}^k,\ x_{s,n+\lvert b\rvert}^k,\ x_{i,j}^k$	$u_s^\ell,\ p_{ijk}^\ell,\ w_{sk}^\ell$
Complexity increase	Baseline	$O(n \cdot p)$ more nodes	$O(n \cdot \lvert L\rvert)$ more variables

2.6 Instance generation

TRS. Instances span n = 10–200 tasks with K = 5–55 tugboats over a 24-h horizon at fixed speed v = 10 knots, with task entry/exit points sampled in a square port area of side 30–130 nautical miles (n.m.) and service legs of 3–8 n.m. Time windows are of width ∼2.5 h staggered across the horizon; most tasks need

one tugboat and ∼20% need 2–3 to meet $H_s^{\min}$ (∼3000–13000 by team size), against tugboat horsepower 4500–6000 with $\alpha_{k,c} \in [0.15, 0.18]$ and $\beta_k \in [0.06, 0.08]$.

MB-TRS. Instances span $n = 10$–290 tasks with $|K| \approx n/5$ tugboats distributed over $p = 2$–10 bases placed uniformly in a 10×10 n.m. area ($v = 10$ knots); each tugboat has a home base, and base capacity δ_b equals its assigned-tug count. Service times are 0.5–2.0 h and time-window widths 1–4 h, with ∼40% of tasks needing 2–3 tugboats; tiered tugboat horsepower (800–5000) is set so the largest task can be served by some base.

TRS-VS. Instances span $n = 10$–100 tasks with $|K| = 3$–20 tugboats in a square port area (side 30–80 n.m.) over a 24-h horizon, with speeds discretized into the three levels of Supplementary Table 1; service distances (3–8 n.m.) together with depot and inter-task distances are stored so all speed-dependent durations and fuel terms are pre-computed per level. Tasks need 1–3 tugboats with $H_s^{\min} \sim$ 1000–7000, against tugboat horsepower 1500–5000 with $\alpha_{k,c} \in [0.14, 0.18]$ and $\beta_k \in [0.05, 0.09]$.

Across all three variants, each instance passes a feasibility screen (every task can be served by some tugboat combination, and a sufficient fraction admit a feasible time window within the horizon) and is regenerated otherwise.

Supplementary Note 3: Baseline methods for the novel problems

3.1 Metaheuristic baselines

For each of the four novel problems we benchmark six representative metaheuristics covering swarm-intelligence and evolutionary families: a Genetic Algorithm (GA) [10], Particle Swarm Optimization (PSO) [11], Ant Colony Optimization (ACO) [12], the Grey Wolf Optimizer (GWO) [13], the Kepler Optimization Algorithm (KOA) [14], and the Equilibrium Optimizer (EO) [15]. All six share a common solution interface, so that performance differences reflect the search operators rather than the problem encoding.

3.1.1 Common encoding and feasibility-preserving decoder

Each metaheuristic searches a continuous chromosome $\mathbf{x} \in [0,1]^L$ and never manipulates the discrete schedule directly. A problem-specific *decoder* maps $\mathbf{x}$ to a feasible solution and returns the problem objective Z (lower is better); it is the only problem-specific component, and all six algorithms call it as a black box. The chromosome length is $L = 2n$ for TRS and MB-TRS, $L = 3n$ for TRS-VS, and $L = 5n$ for PSP, where n is the number of tasks (vessels). The genes encode, per task: a *priority* key that sets the decode order; a *start fraction* that places the service start within its time window; and, where applicable, a discrete *speed level* (TRS-VS) or a berth choice and inbound/outbound tugboat-ordering keys (PSP).

The decoder is *guarded greedy*: it processes tasks in descending priority order and, for each, attempts to commit a feasible assignment (a horsepower-feasible tugboat subset reachable within the time window, a start time within that window, and, for MB-TRS, a home-base-consistent route) while checking the horsepower, time-propagation, fuel-capacity, berth/tugboat-conflict, and horizon constraints. Any task that cannot be committed without violating a constraint is left unserved and incurs the model's own penalty (W per task, or $\lambda_1 M \alpha_i$ for PSP). Every chromosome therefore decodes to a feasible solution, in the worst case the trivial solution that serves no task, so the metaheuristics optimize feasibility-preserved objective values directly, without external penalty terms or post-hoc repair. ACO uses the continuous $\mathrm{ACO}_{\mathbb{R}}$ variant of Socha and Dorigo [12], the standard adaptation of ant colony optimization to continuous domains, operating on the same $[0,1]^L$ chromosome.

3.1.2 Search settings and per-algorithm operators

All six algorithms use a population (or solution archive) of 30, a per-instance wall-clock budget of 60 s (matching the runtime budget $T_{\max}$ used for LACE on these problems), and five independent seeds $\{0,1,2,3,4\}$; each algorithm checks the elapsed time before every objective evaluation, stops strictly at the budget, and returns the best feasible solution found. Algorithm-specific hyperparameters were tuned per problem on the development set, initialized from the standard

literature-recommended values reported in Supplementary Table 3; the continuous decoder (and hence the chromosome length L) is itself problem-specific.

- **GA** [10]: real-coded; tournament selection ($k = 3$), uniform crossover ($p_c = 0.8$), per-gene Gaussian mutation ($p_m = 0.1$, $\sigma = 0.1$) with clipping to $[0, 1]$, and elitism of the single best individual.
- **PSO** [11]: global-best, with inertia $w = 0.7$, cognitive $c_1 = 1.5$, social $c_2 = 1.5$, velocity clamped to $[-0.5, 0.5]$ and position to $[0, 1]$.
- **ACO$_\mathbb{R}$** [12]: a solution archive of size 30 with rank-based Gaussian kernel weights, locality $q = 0.5$, and deviation scale $\xi = 0.85$.
- **GWO** [13]: three leaders (α, β, δ); the encircling coefficient a is annealed linearly from 2 to 0 over the (adaptively estimated) run length.
- **KOA** [14]: each "planet" is pulled toward the incumbent "Sun" by a gravitational term modulated by a random eccentricity $e \in [0, 0.9]$, plus an orbital perturbation whose amplitude decays over the budget.
- **EO** [15]: an equilibrium pool of four candidates (the three best plus their centroid), with exponential-decay coefficient $a_1 = 2.0$, generation-rate control $a_2 = 1.0$, and generation probability GP $= 0.5$.

Supplementary Table 3: Hyperparameters of the six metaheuristic baselines, tuned per problem on the development set; the standard literature-recommended starting values are shown. The continuous decoder (chromosome length $L = 2n$, $3n$, or $5n$) is problem-specific.

Method	**Hyperparameters**
GA	pop $= 30$; tournament $k = 3$; $p_c = 0.8$; $p_m = 0.1$, $\sigma = 0.1$; elitism $= 1$
PSO	pop $= 30$; $w = 0.7$; $c_1 = c_2 = 1.5$; $v_{\max} = 0.5$
ACO$_\mathbb{R}$	archive $= 30$; locality $q = 0.5$; deviation $\xi = 0.85$
GWO	pop $= 30$; leaders α, β, δ; $a : 2 \to 0$
KOA	pop $= 30$; eccentricity $e \in [0, 0.9]$; annealed orbital noise
EO	pop $= 30$; $a_1 = 2.0$; $a_2 = 1.0$; GP $= 0.5$

Common to all methods: per-instance wall-clock budget 60 s; five seeds $\{0, 1, 2, 3, 4\}$.

3.2 Exact baseline (Gurobi)

As an exact reference, the mixed-integer programs defined in Supplementary Notes 1 and 2 are solved directly with Gurobi 11.0. PSP, TRS, and MB-TRS are already MILPs; for TRS-VS the speed–fuel coupling is linearized through the McCormick variables w_{sk}^{ℓ} (Eqs. (65)–(67)) together with the per-level pre-computed coefficients $\psi_s^{k\ell}$ and $\phi_{ij}^{k\ell}$, so all four models are passed to Gurobi as MILPs without modification.

Each instance is solved under two wall-clock budgets, 1 min and 30 min, reported in the main text as Gurobi (1 min) and Gurobi (30 min), and, matching the other methods, under five random seeds. Within each budget Gurobi runs with

default optimality-gap and threading settings; on termination the best incumbent feasible solution is returned and scored with the same objective Z used by all other methods, and an instance for which no feasible incumbent is found within the budget is recorded as a failure. The two budgets bracket the quality–time trade-off of exact search: at the tested scales the integer programs do not close the optimality gap, so Gurobi returns improving incumbents rather than certified optima, and the 30-min budget yields only modest gains over the 1-min budget.

Supplementary Note 4: Automated Construction of the I–O–T Interface

Stage One of LACE turns a natural-language problem description $\mathcal{P}$ into a verified problem contract $(\mathcal{I}, \mathcal{O}, \mathcal{T})$ and an initial heuristic portfolio $\mathcal{H}_0$, without per-problem human algorithm design. The contract is produced by four dedicated LLM *designer agents*—an Input Designer ($\mathcal{I}$), an Output Designer ($\mathcal{O}$), a Tool Designer ($\mathcal{T}$), and a Heuristic Generator ($\mathcal{H}$)—invoked in sequence so that each agent is conditioned on the validated output of the previous one (Supplementary Fig. 3). The defining feature of this stage is that no artifact is accepted on the strength of an LLM's say-so: every generated module passes through a *generate–validate–repair* loop whose acceptance test is *execution-grounded*, i.e. the module must demonstrably run on real instance data before it is admitted. This section documents the four agents (Supplementary Note 4.2), the validation and repair mechanism that makes the contract reliable (Supplementary Note 4.3), and which parts of the contract are generated versus adopted in our benchmark evaluation (Supplementary Note 4.4). The complete implementation, including the exact prompt of each agent, is openly available in the code repository.

4.1 Design principle: a verified contract, separated from algorithmic reasoning

The contract $(\mathcal{I}, \mathcal{O}, \mathcal{T})$ encodes only the *problem-defining* part of a solver—what data are given ($\mathcal{I}$), what a solution is ($\mathcal{O}$), and how feasibility and cost are computed ($\mathcal{T}$)—which every combinatorial optimization problem possesses regardless of its structure. The algorithmic reasoning is deferred entirely to $\mathcal{H}$, where each heuristic is a free-form `solve` function. Separating the two means the same four contract slots instantiate for every problem, so the LLM is never asked to invent problem-specific input parsing, output assembly, and feasibility logic at the same time as the algorithm. Generation is *bounded*: the smallest one or two instance files are parsed once and reused as the ground truth against which every downstream module is validated, and the actual parsed `instance` dictionary (its keys, types, and shapes) is fed verbatim to the Output, Tool, and Helper designers so that they reference real field names rather than guessing them.

4.2 The four designer agents

Input Designer ($\mathcal{I}$). The Input Designer is prompted with $\mathcal{P}$ and the raw bytes of the smallest instance files, and returns an instance parser together with a schema that names every input field and its type. Because file-format errors are the most common failure mode, the agent is required to determine the exact byte layout from the sample—how many values each line carries and whether a single logical record wraps across physical lines—rather than assuming the format from the problem semantics. The acceptance test is purely mechanical: the parser must read *every* validation instance into a dictionary carrying all declared keys, without raising.

Output Designer ($\mathcal{O}$). Conditioned on the locked $\mathcal{I}$, the Output Designer specifies the exact solution structure a heuristic must return (keys, value types, indexing conventions), together with a trivial *feasible-by-construction* placeholder solution built only from instance data. The placeholder serves a dual role: it documents the solution shape and, crucially, acts as a deterministic known-feasible witness used later to validate $\mathcal{T}$.

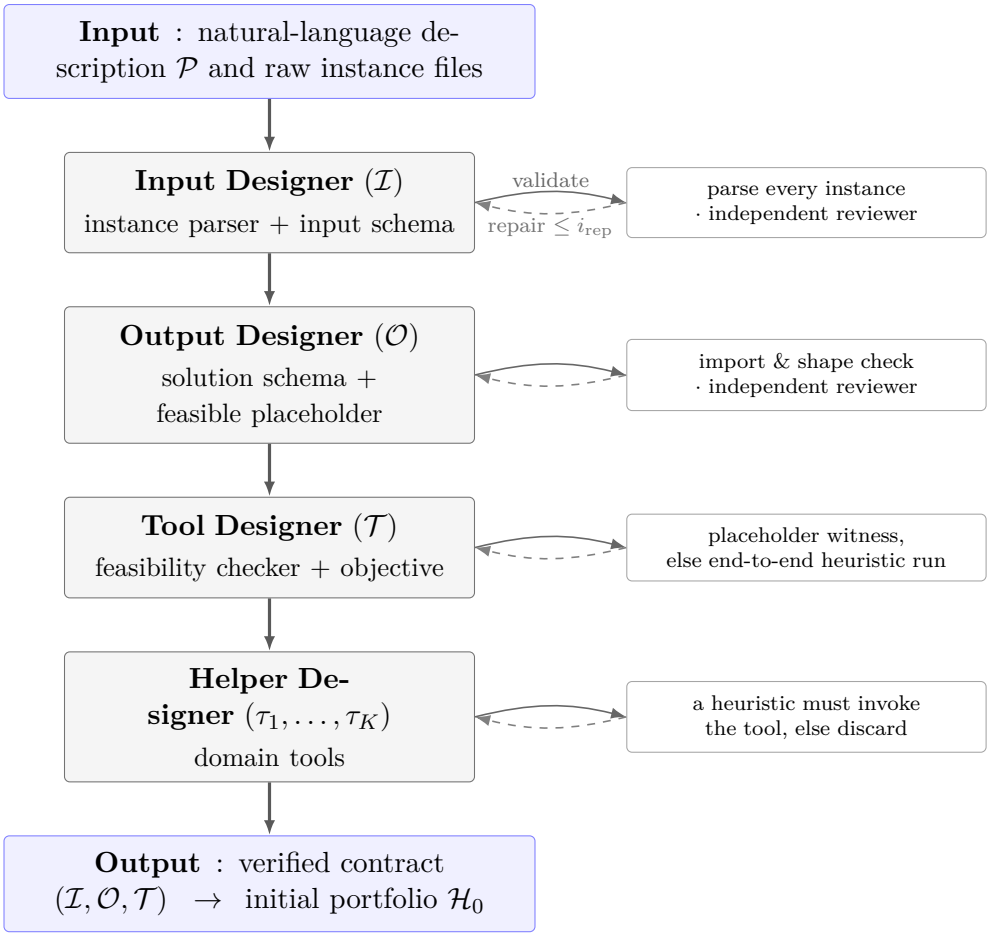
Tool Designer ($\mathcal{T}$). Conditioned on $\mathcal{I}$ and $\mathcal{O}$, the Tool Designer authors the core of the tool library: a feasibility checker and an objective evaluator (expressed in lower-is-better form). There is no pre-existing evaluator—the agent authors the formal definition of "feasible" and "cost" for the problem directly from $\mathcal{P}$. Two design choices are responsible for the reliability gains reported in the main text. First, the checker does not return a bare Boolean: it reports the *first* violated constraint with an informative message and annotates each check with the specific constraint it enforces, exposing the full step-by-step constraint-validation process to downstream heuristic generation—which the main text identifies as more useful than a declarative pass/fail verdict. Second, the checker is required to accept the $\mathcal{O}$ placeholder, giving a guaranteed feasible point on which the assembled contract can be exercised.

Helper Designer (domain tools τ_k). The remaining tools $\mathcal{T} = \{\tau_1, \ldots, \tau_K\}$—construction routines, neighbourhood moves, cost-delta evaluators, exact subroutines—are produced by a two-phase Helper Designer. A single planning call first lists a small, non-overlapping set of helpers (names and purposes only), which avoids the duplicates that independent per-helper generations would produce. Each planned helper is then implemented and validated individually by generating a heuristic that is *required* to call it, with an instrumented call counter confirming the invocation. Helpers are optional: any helper that cannot be made to validate within the repair budget is silently discarded and never blocks the pipeline, in contrast to the mandatory feasibility checker and objective evaluator.

4.3 Execution-grounded validation and bounded repair

Every designer is wrapped in the same generate–validate–repair loop, which runs for up to i_{rep} attempts (default 3) before the stage is declared failed (Supplementary Fig. 3). What differs between stages is only the acceptance test, and the central idea is that acceptance is grounded in execution rather than in an LLM judgment. Three kinds of gate are used. (i) A *machine smoke test*: the artifact must import and run on real data—the parser must read every validation instance, and the tool module must compile and define both core functions. (ii) A *heuristic witness*: for $\mathcal{T}$ and the helpers, correctness is established by assembling a temporary specification from $(\mathcal{I}, \mathcal{O}, \mathcal{T})$ and either checking that the $\mathcal{O}$ placeholder is accepted by the feasibility checker and scored by the objective (a deterministic feasible witness, the fast path), or having the LLM write a real solver that must read an instance, return a correctly shaped solution, and have it scored end to end through $\mathcal{I} \to \mathcal{O} \to \mathcal{T}$; if some heuristic can traverse the whole contract and obtain a feasible, scored solution, the contract provably composes. (iii) An *independent reviewer*: for $\mathcal{I}$ and $\mathcal{O}$, a separate reviewer LLM audits whether the artifact captures the

problem, instructed to reject *only* for substantive defects—a missing or mistyped field, an invented key absent from the data, a solution shape that cannot represent a valid solution, or a hard constraint that is simply not represented—and never for wording, style, or defensive-coding concerns, so that it acts as a low-false-positive backstop on top of the machine checks rather than a stylistic gate. When a check fails, the failure report (a parser traceback, an infeasibility message, or the reviewer's concrete objection) is fed back to the designer for a bounded repair attempt. After all four modules pass, the fully assembled contract is exercised once more with the placeholder witness as a final assembly sanity check before promotion.



Supplementary Figure 3: **Stage-One construction of the $\mathcal{I}$-$\mathcal{O}$-$\mathcal{T}$ contract.** Four LLM designer agents (centre) are invoked in sequence on a natural-language description $\mathcal{P}$, each conditioned on the validated output of the previous one. Each agent runs a generate–validate–repair cycle: its draft is checked against an execution-grounded gate (right), and on failure the agent regenerates for up to i_{rep} attempts (dashed). The gates differ by stage—a machine smoke test together with an independent, substantive-defect-only reviewer for $\mathcal{I}$ and $\mathcal{O}$, and a heuristic witness running end to end through $\mathcal{I} \to \mathcal{O} \to \mathcal{T}$ for the tools—so no module is accepted on the strength of an LLM judgement alone. The output is a verified contract $(\mathcal{I}, \mathcal{O}, \mathcal{T})$ and the initial portfolio $\mathcal{H}_0$ that Stage Two evolves.

4.4 Benchmark instantiation: what is adopted versus generated

To keep the comparison on CO-Bench [16] strictly fair, we do not let LACE define its own scoring. For the 36 CO-Bench problems we therefore adopt the input layer and the scoring oracle from CO-Bench, while generating the rest of the contract:

- **Input $\mathcal{I}$ — adopted.** The instance parser is taken directly from the CO-Bench problem package, so that LACE reads the *identical* parsed instances as every baseline. The Input Designer is bypassed for these problems; its generation path is exercised only on the novel problems below.
- **Scoring oracle — adopted.** Final test scores are computed by CO-Bench's official evaluator, and the same evaluator drives objective values during Stage-Two development, so LACE is optimized and scored against the same third-party metric as all baselines rather than a self-authored one.
- **Output $\mathcal{O}$, tool library $\mathcal{T}$, heuristics $\mathcal{H}$ — generated.** The solution representation, the domain tool library (construction routines, neighbourhood moves, and the step-by-step feasibility checker exposed to the LLM), and the heuristic portfolio are all produced by LACE from the problem description, with no per-problem human authoring.

For the four novel container-terminal problems (PSP, TRS, MB-TRS, TRS-VS) no CO-Bench package exists, so the *entire* contract $(\mathcal{I}, \mathcal{O}, \mathcal{T})$ and the portfolio $\mathcal{H}$ are generated from the natural-language description through the pipeline above. This is the regime in which the full $\mathcal{I}$-$\mathcal{O}$-$\mathcal{T}$ automation is exercised end to end, and on which the zero-shot results in the main text are obtained.

Supplementary Note 5: Ablation studies

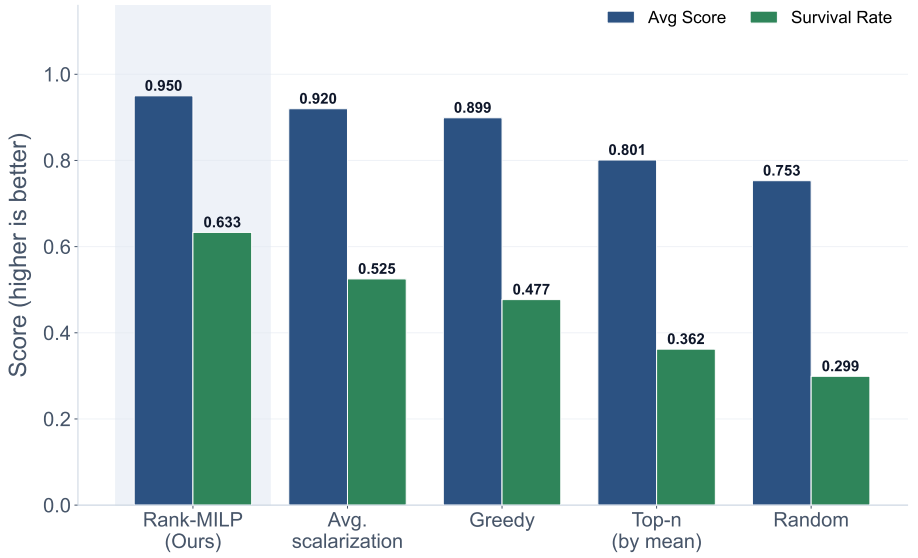
We isolate the contribution of each of LACE's core design choices through four ablations, in each case varying a single component while holding the o3-mini backbone, the development/test split, and all other components fixed across the 40 problems. All aggregate scores reported in this section are taken over the full set of 40 problems (36 CO-Bench problems plus the 4 novel problems); the corresponding aggregate over the 36 CO-Bench problems alone is an Avg Score of 0.945, matching the main text.

5.1 Effect of the complementary selection strategy

In Stage Two, LACE reduces the pool of $2n$ candidate heuristics to a portfolio of n using the rank-based complementary selection objective introduced in the main text, which minimizes the mean best-achieved rank across development instances. As argued there, a linear-averaging objective F_{avg} is dominated by large-scale instances and therefore fails to enforce balanced coverage across a heterogeneous

instance set. To test this empirically, we hold the candidate generation, the development/test split, and the backbone fixed, and replace only the selection rule applied at each Stage-Two iteration with one of five alternatives: **Rank-MILP** (ours), **Avg. scalarization** (the F_{avg} baseline), **Greedy** forward selection (iteratively adding the heuristic that most improves the best-of-set score), **Top-n by mean** (the n heuristics with the best mean rank across development instances, ignoring complementarity), and **Random** selection. Each portfolio is scored on the held-out test set, at the default per-instance budget, using Avg Score and Survival Rate (both defined in the main text; Supplementary Fig. 4).

Results. Aggregated over all 40 problems, Avg Score degrades only moderately across the five rules (0.950, 0.920, 0.899, 0.801, 0.753), whereas Survival Rate collapses far more sharply (0.633, 0.525, 0.477, 0.362, 0.299): Rank-MILP exceeds the next-best rule (Avg. scalarization) by roughly 11 percentage points on Survival Rate but by about 3 percentage points on Avg Score. This divergence is the key observation—a high average can coexist with systematic failure on the hardest instances, exactly the regime that the rank-based, scale-invariant objective is designed to protect. The Top-n rule, which selects the strongest individual heuristics without regard to complementarity, performs markedly worse than Rank-MILP (0.801 vs. 0.950 Avg Score; 0.362 vs. 0.633 Survival Rate), indicating that the gain comes from assembling a *complementary set* rather than from collecting strong solvers. Complementary rank-based selection is thus an important component of LACE, and its benefit is concentrated in tail reliability on heterogeneous instances.



Supplementary Figure 4: **Effect of the portfolio selection strategy.** Avg Score and Survival Rate on the held-out test set (40 problems) for five rules used in the Stage-Two complementary selection step: rank-based MILP (ours), average-objective scalarization (F_{avg}), greedy forward selection, top-n by mean, and random selection, with all other components held fixed. Survival Rate degrades far more steeply than Avg Score across the rules, showing that the benefit of rank-based complementary selection is concentrated in tail coverage on hard instances. Values are from a single run per rule, aggregated over all $n = 40$ problems.

5.2 Effect of the evolution operators

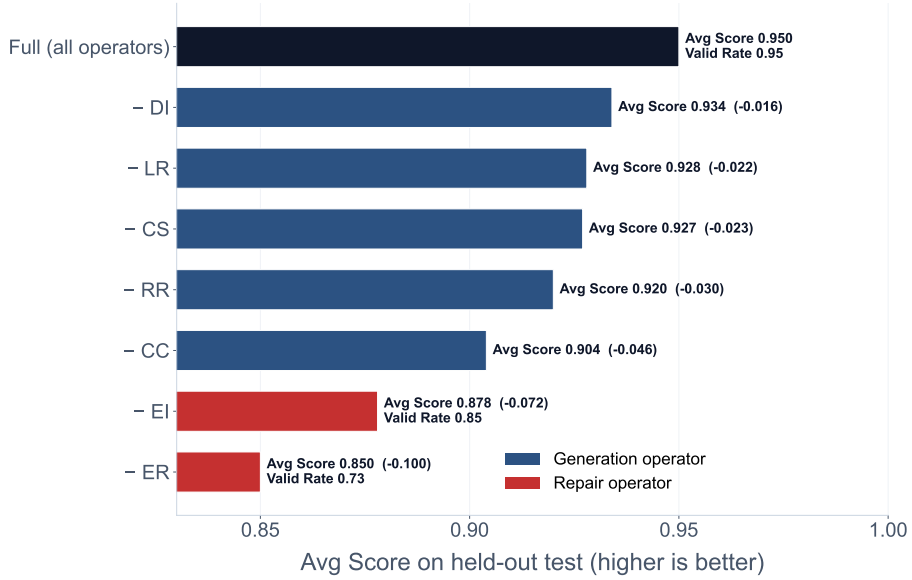
Stage Two refines the portfolio through five generation operators—Local Refinement (LR), Reflective Redesign (RR), Complementary Crossover (CC), Comparative Synthesis (CS), and Diversity Injection (DI)—together with two reactive repair operators, Error Repair (ER) and Efficiency Improvement (EI). To quantify the contribution of each, we perform a leave-one-out ablation: a single operator is disabled while all other components, the backbone, and the development/test split are held fixed across the 40 problems at the default per-instance budget. We report Avg Score for every variant and, for the two repair operators, additionally Valid Rate, since their function is to restore feasibility (ER) and enforce the runtime budget (EI) rather than to improve solution quality directly (Supplementary Fig. 5).

Results. Removing any operator lowers performance relative to the full system (Avg Score 0.950). Among the generation operators the effect is moderate and graded—−DI 0.934, −LR 0.928, −CS 0.927, −RR 0.920—and largest for Complementary Crossover (−CC 0.904, a drop of 0.046), consistent with the central role of complementarity in LACE: the operator that explicitly combines specialists is the most valuable generator. The two reactive repair operators dominate the ablation: disabling EI lowers Avg Score to 0.878 and Valid Rate to 0.85, and disabling ER lowers them further to 0.850 and 0.73, the two largest degradations on both metrics. This indicates a division of labor between the two operator classes—generation operators drive solution quality, with complementary crossover contributing most, whereas the repair operators are what keep LLM-generated heuristics feasible and within budget. Every operator is therefore retained, but reliability rests primarily on reactive repair.

5.3 Effect of the per-instance runtime budget

A central design feature of LACE is that Stage Two is executed *independently* under a chosen per-instance budget $T_{\max}$, so that the same framework yields portfolios spanning a quality–time spectrum from emergency decisions to strategic planning (Fig. 1 in the main text). To characterize this trade-off, we re-run Stage Two at several budgets and report the resulting held-out Avg Score per problem family. Because the two regimes operate at different instance scales, we sweep $T_{\max} \in \{5, 10, 15\}$ s for the 36 CO-Bench families (Supplementary Fig. 6a) and $T_{\max} \in \{30, 60, 90\}$ s for the four novel problems (Supplementary Fig. 6b); the budgets actually used in the main experiments (10 s and 60 s, respectively) are marked in each panel.

Results. Avg Score increases monotonically with $T_{\max}$ in every family and then saturates, with the steepest gains at the short-budget end. The effect is largest for the families that are hardest under tight budgets: Location & Assignment rises from 0.874 at 5 s to 0.990 at 10 s, and Routing & Graphs from 0.822 to 0.896, whereas already-easy families such as Packing & Cutting move only modestly (0.945 to 0.982). By the operating budget used in the main text, most of the attainable gain has already been realized—each family shows clearly diminishing returns, with only marginal further improvement from 10 to 15 s (typically $\leq$ 0.01)—so 10 s is a deliberate, efficient choice rather than an arbitrary one; the



Supplementary Figure 5: **Leave-one-out ablation of the evolution operators.** Held-out Avg Score when each of the five generation operators (LR, RR, CC, CS, DI) and two reactive repair operators (ER, EI) is removed in turn; the full system (top bar) attains 0.950. For the two repair operators the Valid Rate of the resulting portfolio is also annotated. Removing the repair operators causes the largest drops in both Avg Score and Valid Rate, while among the generation operators Complementary Crossover (CC) is the most impactful. Values are from a single run per configuration, aggregated over all $n = 40$ problems.

four novel problems show the same monotone-then-saturating behavior on a larger budget scale (0.918 at 30 s, 0.980 at 60 s, 0.989 at 90 s), reflecting their larger instances. These curves indicate that $T_{\max}$ acts as a single, predictable control knob: increasing it trades wall-clock time for solution quality with diminishing returns, and the budgets adopted in the main experiments sit at the efficient point of that trade-off.

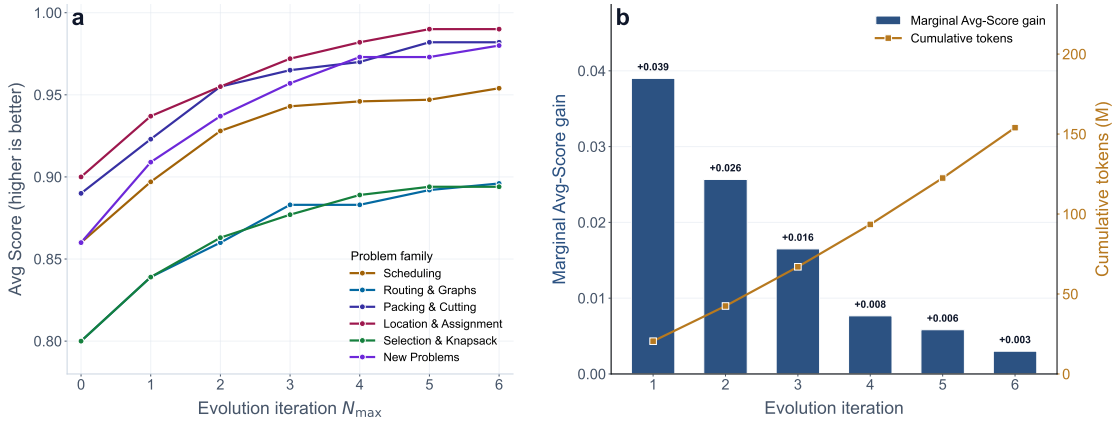
5.4 Effect of the number of evolution iterations

Stage Two refines the portfolio over $N_{\max}$ iterations, each generating n new candidates that are pooled with the current portfolio before complementary selection. To determine how solution quality accumulates and at what cost, we evaluate the held-out portfolio after every iteration (iteration 0 denotes the initial portfolio $\mathcal{H}_0$ produced in Stage One). Supplementary Fig. 7a reports Avg Score per problem family as a function of iteration, and Supplementary Fig. 7b reports the marginal Avg-Score gain of each iteration against the cumulative token cost incurred, both on the o3-mini backbone across the 40 problems.

Results. Avg Score rises monotonically in every family, from a family-mean of 0.852 at $\mathcal{H}_0$ to 0.949 after six iterations (the unweighted mean over the six families; the corresponding 40-problem mean is 0.950), but with sharply diminishing marginal returns: the per-iteration gain falls from +0.039 at the first iteration to +0.016, +0.008, and finally +0.003 by the sixth (Supplementary Fig. 7b). Meanwhile the per-iteration token cost grows rather than holds constant (20.5 to 31.5 M tokens, a cumulative ∼154 M over the run), because each new candidate is con-

Supplementary Figure 6: **Effect of the per-instance runtime budget $T_{\max}$.** Held-out Avg Score as a function of $T_{\max}$, obtained by executing Stage Two independently at each budget. **a**, The 36 CO-Bench problems grouped into five families, swept over $T_{\max} \in \{5, 10, 15\}$ s. **b**, The four novel problems (New Problems family), swept over $T_{\max} \in \{30, 60, 90\}$ s. Arrows mark the budget used in the main experiments (10 s and 60 s). Quality increases monotonically with budget and saturates near the operating point in both regimes. Each plotted value is the mean Avg Score over the problems in a family (single run per budget; $n = 36$ problems across five families in **a**, $n = 4$ in **b**).

ditioned on an increasingly long portfolio context. The two trends cross in the expected way—gains shrink while cost climbs—so that beyond roughly six iterations additional search buys negligible quality at rising expense. This identifies $N_{\max} = 6$ as the efficient operating point adopted in the main experiments: it captures essentially all of the attainable improvement while keeping discovery cost bounded.



Supplementary Figure 7: **Effect of the number of evolution iterations $N_{\max}$.** **a**, Held-out Avg Score per problem family as a function of iteration; iteration 0 is the initial portfolio $\mathcal{H}_0$ produced in Stage One. **b**, Marginal Avg-Score gain of each iteration (bars, left axis) and cumulative token cost (line, right axis), averaged over the 40 problems on the o3-mini backbone. Gains diminish sharply while per-iteration cost grows, identifying $N_{\max} = 6$ as the efficient operating point. Curves in **a** are family-mean Avg Score over the six problem families (the five CO-Bench families plus the novel-problem family); values are from a single run ($n = 40$ problems).

References

[1] UNCTAD, Review of Maritime Transport 2023, Technical Report, United Nations Conference on Trade and Development, Geneva, 2023.

[2] C. Bierwirth, F. Meisel, A survey of berth allocation and quay crane scheduling problems in container terminals, European Journal of Operational Research 202 (2010) 615–627.

[3] IMO, 2023 IMO Strategy on Reduction of GHG Emissions from Ships, Technical Report, International Maritime Organization, 2023. URL: https://www.imo.org/en/ourwork/environment/pages/2023-imo-strategy-on-reduction-of-ghg-emissions-from-ships.aspx.

[4] T. I. Fossen, Handbook of Marine Craft Hydrodynamics and Motion Control, Wiley, 2011.

[5] T. E. Notteboom, B. Vernimmen, The effect of high fuel costs on liner service configuration in container shipping, Journal of Transport Geography 17 (2009) 325–337.

[6] H. N. Psaraftis, C. A. Kontovas, Speed models for energy-efficient maritime transportation: A taxonomy and survey, Transportation Research Part C: Emerging Technologies 26 (2013) 331–351.

[7] H. Lindstad, B. E. Asbjørnslett, A. H. Strømman, Reductions in greenhouse gas emissions and cost by shipping at lower speeds, Energy Policy 39 (2011) 3456–3464.

[8] H. Zhong, Y. Zhang, Y. Gu, A bi-objective green tugboat scheduling problem with the tidal port time windows, Transportation Research Part D: Transport and Environment 110 (2022) 103409.

[9] X. Wei, S. Jia, Q. Meng, K. C. Tan, Tugboat scheduling for container ports, Transportation Research Part E: Logistics and Transportation Review 142 (2020) 102071.

[10] J. H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, MIT Press, Cambridge, MA, 1992.

[11] J. Kennedy, R. Eberhart, Particle swarm optimization, Proceedings of ICNN'95 – International Conference on Neural Networks 4 (1995) 1942–1948.

[12] K. Socha, M. Dorigo, Ant colony optimization for continuous domains, European Journal of Operational Research 185 (2008) 1155–1173.

[13] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey Wolf Optimizer, Advances in Engineering Software 69 (2014) 46–61.

[14] M. Abdel-Basset, R. Mohamed, S. A. A. Azeem, M. Jameel, M. Abouhawwash, Kepler optimization algorithm: A new metaheuristic algorithm inspired by Kepler's laws of planetary motion, Knowledge-Based Systems 268 (2023) 110454.

[15] A. Faramarzi, M. Heidarinejad, B. Stephens, S. Mirjalili, Equilibrium optimizer: A novel optimization algorithm, Knowledge-Based Systems 191 (2020) 105190.

[16] W. Sun, S. Feng, S. Li, Y. Yang, CO-Bench: Benchmarking language model agents in algorithm search for combinatorial optimization, Proceedings of the AAAI Conference on Artificial Intelligence 40 (2026) 33126–33134.