NYCU TLM

Final Project Presentation

Tabu Search

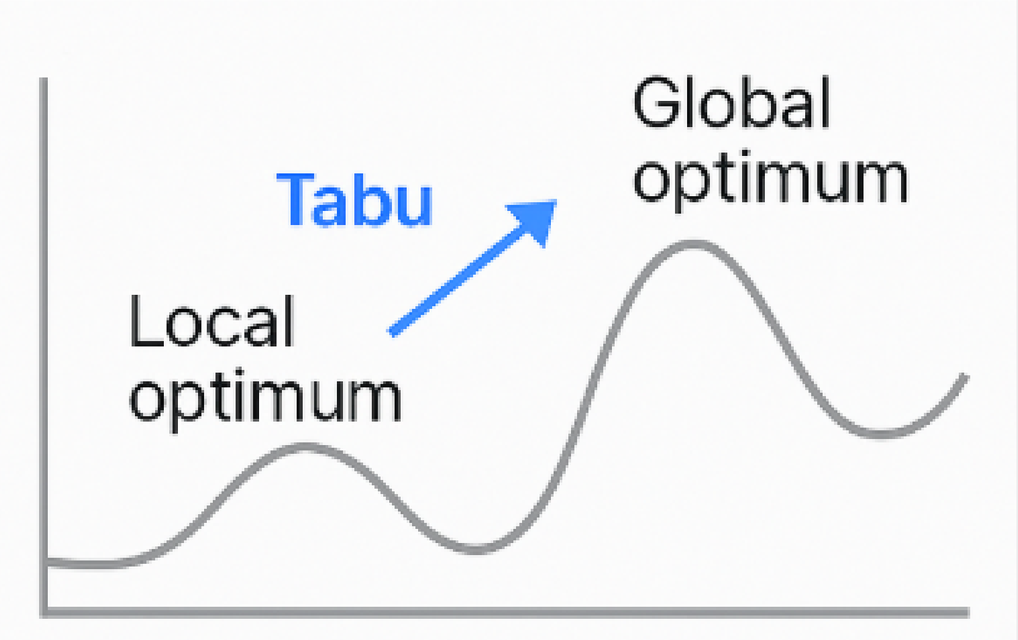
Team Members

李幸其 吳昭泓 陳俊翰

What is Tabu Search?

Definition:

- A metaheuristic algorithm for solving complex optimization problems.
- Guides a local heuristic search to explore the solution space beyond local optima.



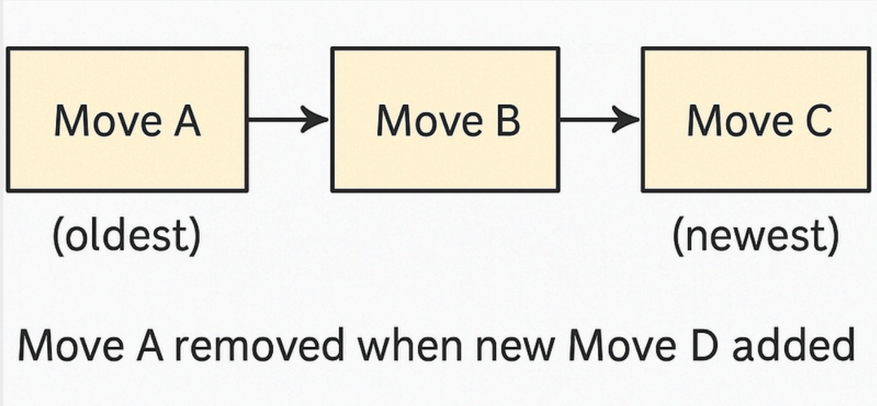
Tabu List and Tenure

What is the Tabu List?

- A short-term memory that stores recent moves or attributes.
- Prevents cycling back to recent solutions.

Tabu Tenure:

- Number of iterations a move remains tabu.
- Implemented as a FIFO (First In, First Out) queue.



Why FIFO?

- Keeps the Tabu List size fixed.
- Automatically removes oldest entries as new ones come in.

Tuning Tabu Tenure

Choosing the Right Tenure:

- Too short: Risk of cycling
- Too long: Search becomes too restricted

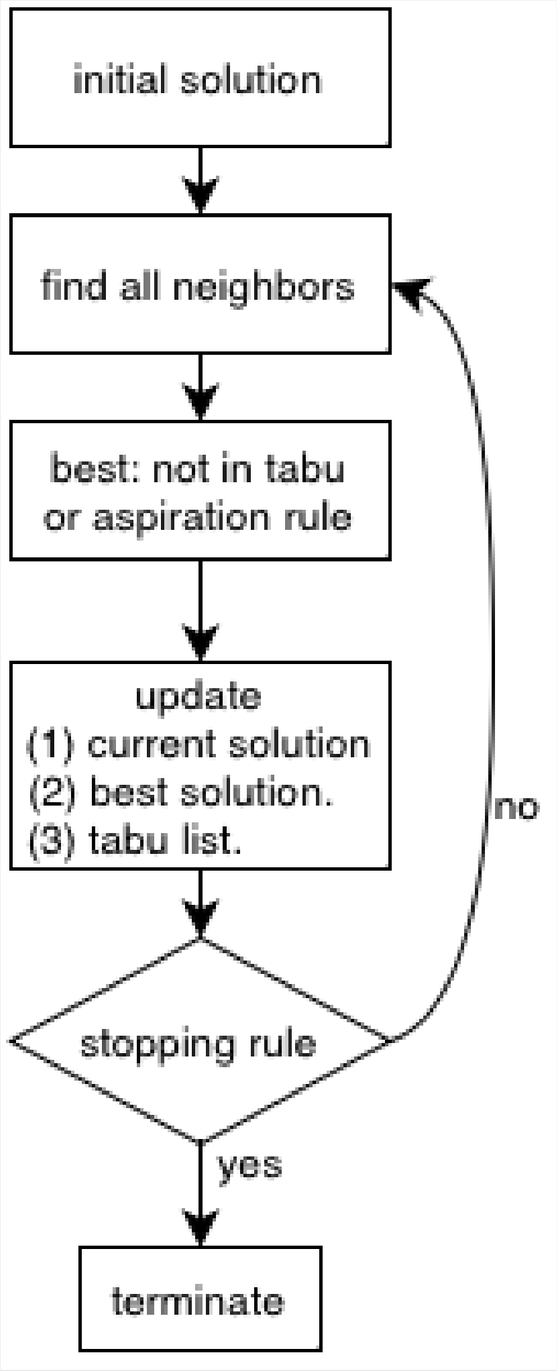
Advanced Option:

- Dynamic Tenure — Adjust tabu length during the search based on progress or stagnation.

What is Tabu Search?

Basic Algorithm Flow:

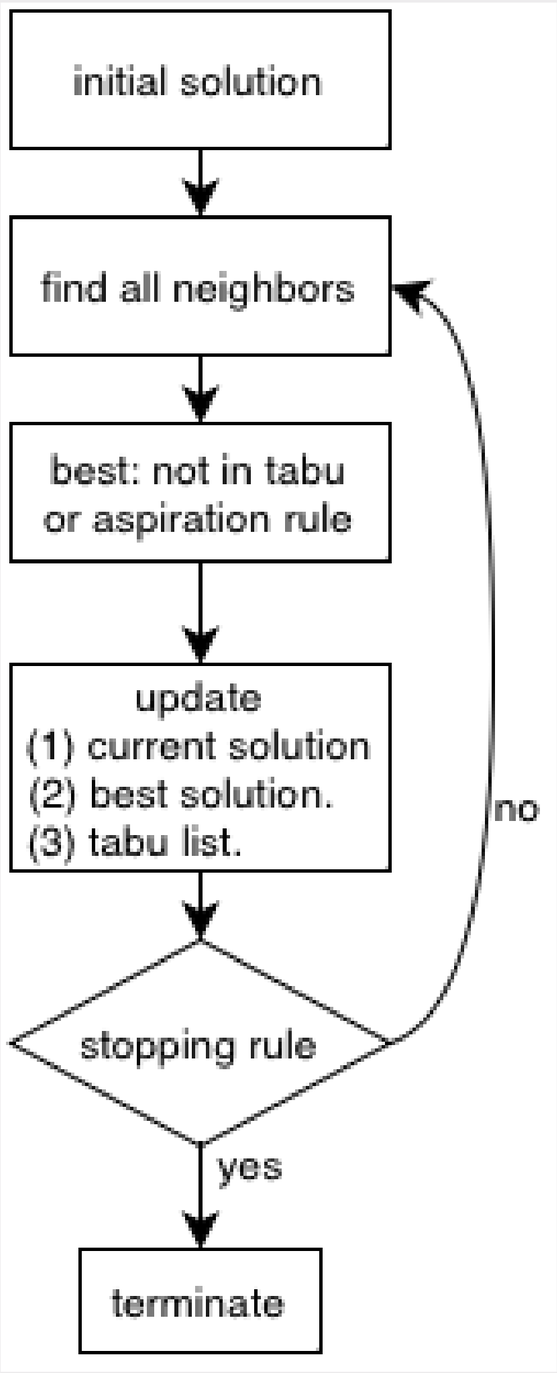
- 1. Start from an initial solution
- 2. Repeat until stopping condition:
  a. Generate neighbor solutions
    – Defines possible moves
  b. Apply Tabu Rule
    – Block moves in Tabu List
    – Aspiration Criteria: Allow tabu move if it's better than best so far



What is Tabu Search?

Basic Algorithm Flow:

- 2. Repeat until stopping condition:
  - c. Determine the New solution
    - – Update Current solution
    - – Update Best solution
    - – Update Tabu List
  - d. Stopping Criteria Met?
    - – Maximum iterations or time, or no improvement (maxStag)
- 3.Find the Final Solution



Uncapacitated Facility Location Problem

fixed cost **assignment cost**

$$Min. \quad \sum_{i=1}^{n} f_i\, y_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}\, x_{ij}$$

$$s.t. \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1, 2, \ldots, m$$

for each demand, one facility serves it

$$x_{ij} \le y_i \qquad \forall i = 1, 2, \ldots, n$$

the facility should be open

$$x_{ij} \in \{0, 1\} \qquad \forall i = 1, 2, \ldots, n \text{ and } j = 1, 2, \ldots, m$$

$$y_i \in \{0, 1\} \qquad \forall i = 1, 2, \ldots, n$$

Initial Solution and Neighbor

- Neighbor

  - type I : for each facility, open ☐ closed

  - type II : swap an open facility and a closed one

  - tabu stores the flipped facility only

- Initial Solution : by the Fixed Cost Type

  - high: open one facility with min total cost

  - medium / low : open k facilities

    facilities farther from open ones are more likely to be selected

Data Structures used in Acceleration

- Tabu

  a flip is in the tabu ☐

  the last iteration we flip it ≥ current iteration - tabu tenure

- Second Best Open Facility

  maintain this info for each customer

- Incremental Cost

  - open ☐ judge whether this facility is closer to each demand

  - close ☐ assign the "affected customer" to its second best facility

100 facilities = customers

tabu tenure = 10

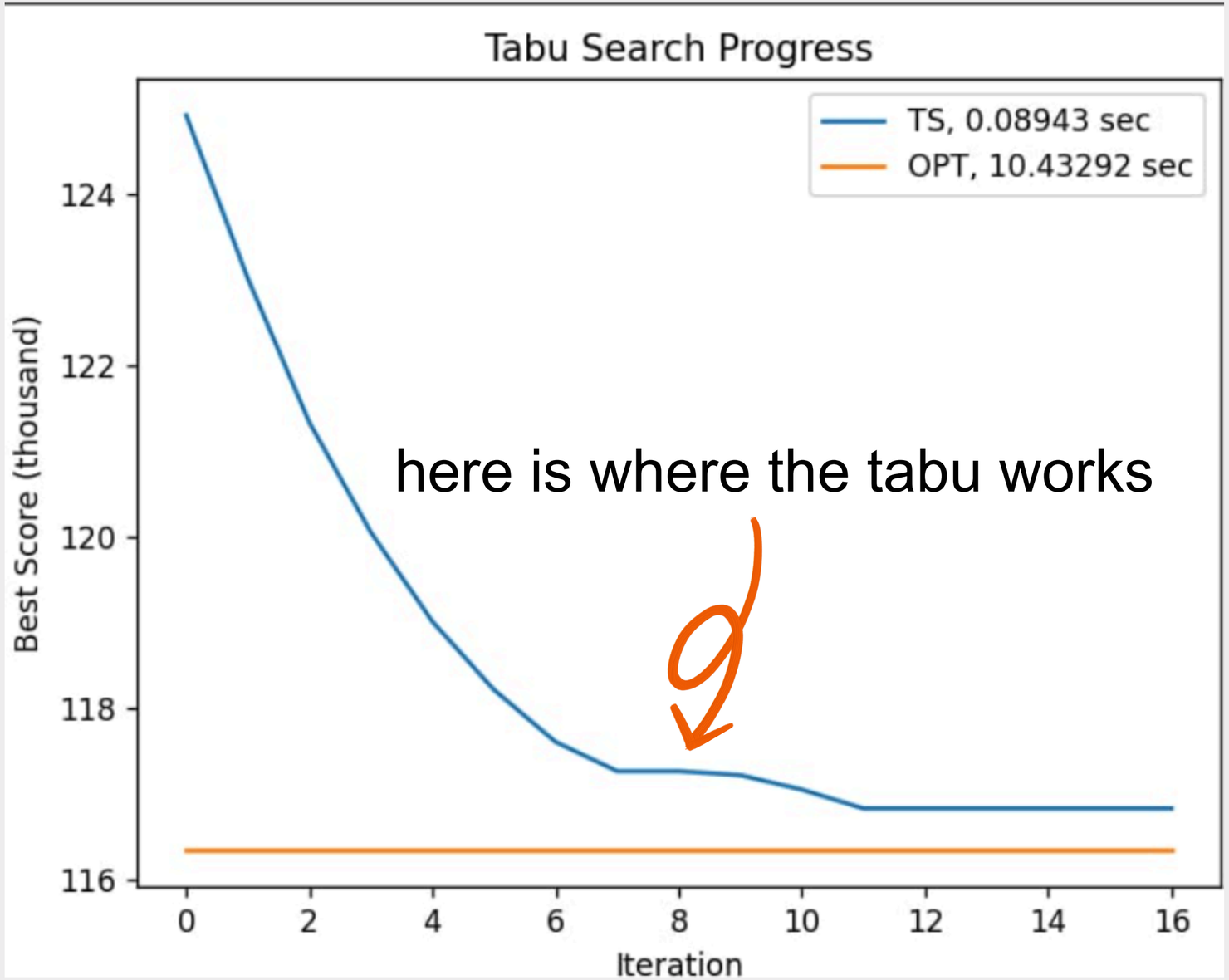
max stag = 5

fixCost : medium

initial: open 10% facilities

TS is 130 times faster

gap = 0.41%



Iteration	*Move*	Facility		
		Open	Close	Current
Initial		-	-	[3, 25, 30, 33, 45, 59, 61, 76, 77, 88]
1	*swap*	96	33	...
2	*swap*	90	3	...
3	*swap*	28	76	...
4	*flip*	-	61	...
5	*swap*	75	59	...
6	*flip*	-	88	...
7	*swap*	41	77	...
8	*swap*	7	45	...
9	*swap*	61	41	...
10	*flip*	-	30	...
11	*swap*	70	25	...
12	*swap*	31	61	...
13	*swap*	24	28	...
14	*swap*	42	24	...
15	*flip*	-	96	...
16	*flip*	65	-	Final: [7, 28, 61, 70, 75, 90, 96]
Gurobi	-	-	-	Final: [12, 31, 74, 75, 79, 96]

- **swap: 11 times**
- **flip : 5 times**

facility configuration differs a lot,

but with near-optimal cost

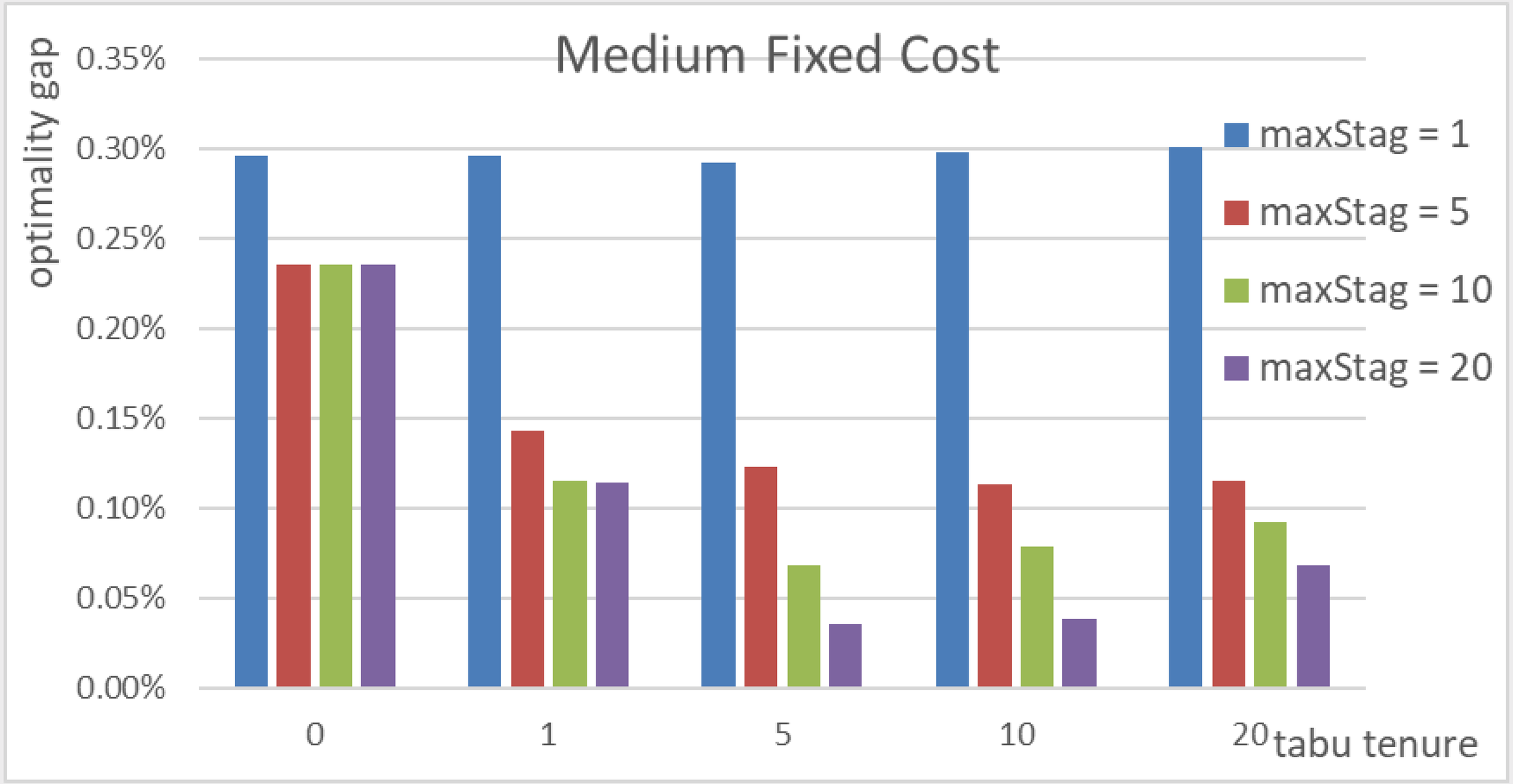
Numerical Experiment

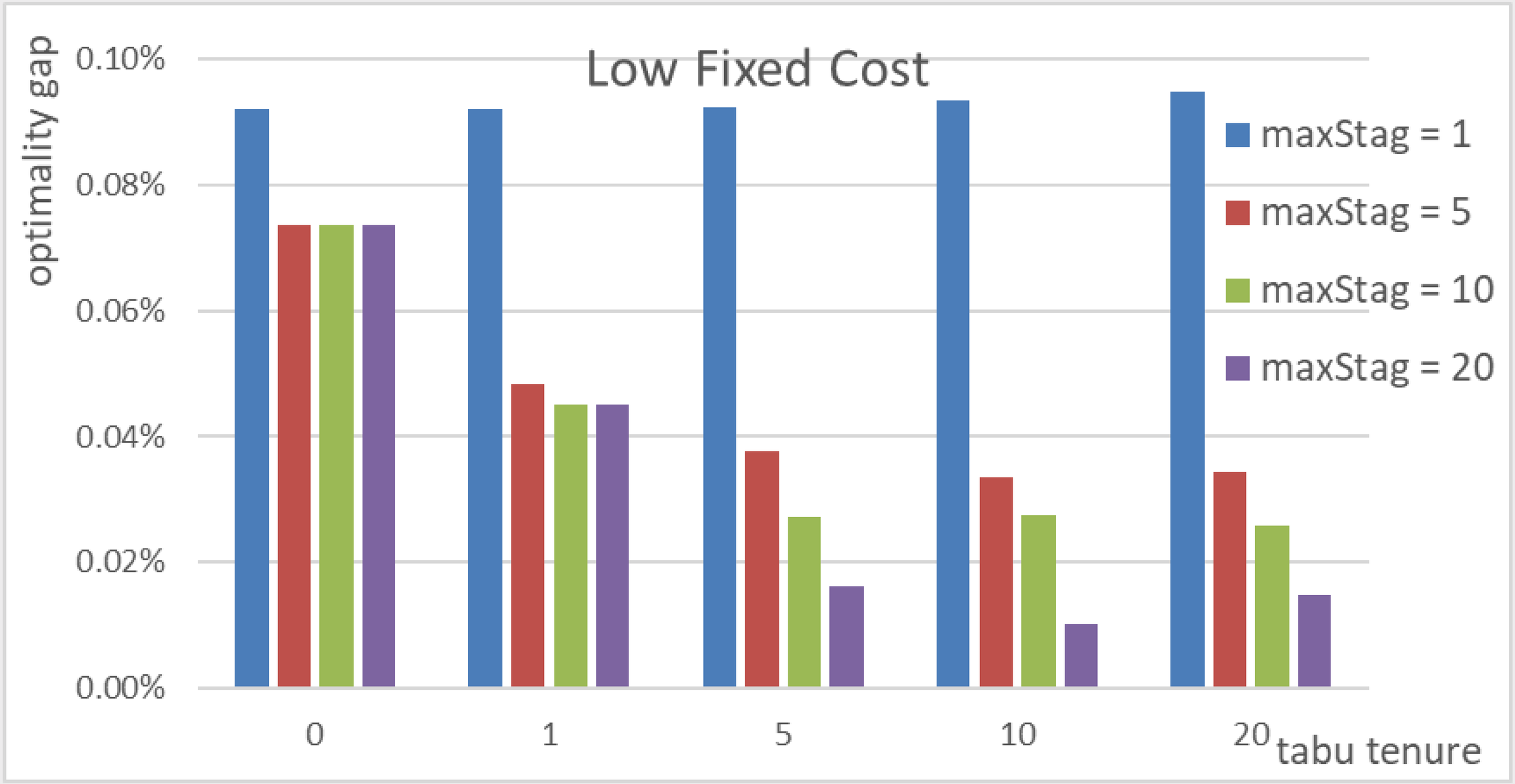
- Assignment Cost
  - Symmetric
  - Uniform [1000, 2000]
- Fixed Cost
  - Low : Uniform [100, 200]
  - Medium : Uniform [1000, 2000]
  - High : Uniform [10000, 20000]
- 10 instances for each type of test case, each repeat 10 times.
- explore the relationship between
  - maxStag, tabu tenure, type of Fixed Cost
  - optimality gap

Comparison about Tabu tenure and maxStag

Low and Medimum Fixed Cost

1. Higher maxStag leads to lower optimality gap.
2. Without tabu, the gap stays high.
3. Higher tabu tenure: the gap may rebound.
4. Medium Fixed Cost is more challenging.

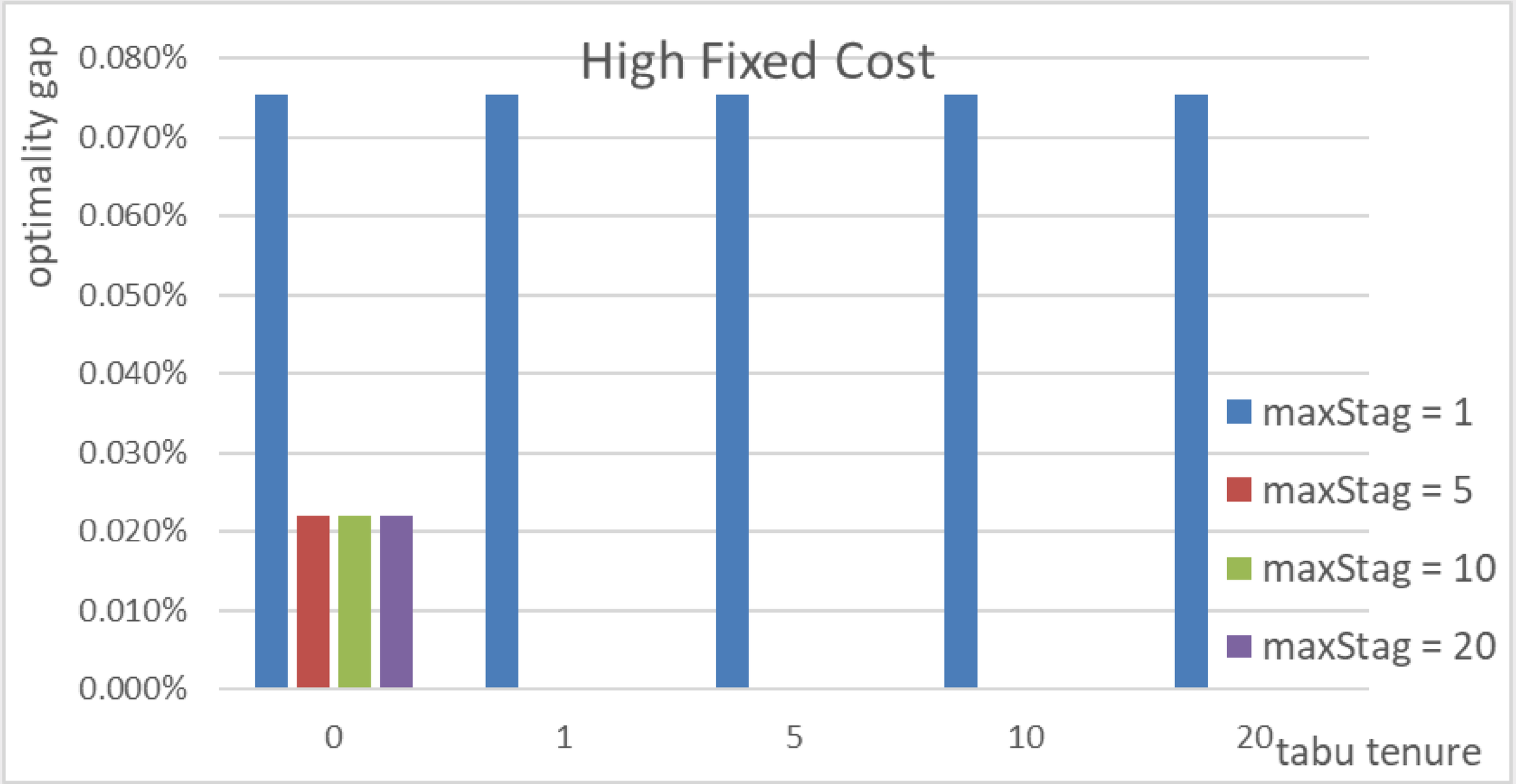

Medium Fixed Cost
optimality gap
0.35%
0.30%
0.25%
0.20%
0.15%
0.10%
0.05%
0.00%
maxStag = 1
maxStag = 5
maxStag = 10
maxStag = 20
0
1
5
10
20
tabu tenure


Low Fixed Cost
optimality gap
0.10%
0.08%
0.06%
0.04%
0.02%
0.00%
0
1
5
10
20
tabu tenure
maxStag = 1
maxStag = 5
maxStag = 10
maxStag = 20

Comparison about Tabu tenure and maxStag

High Fixed Cost

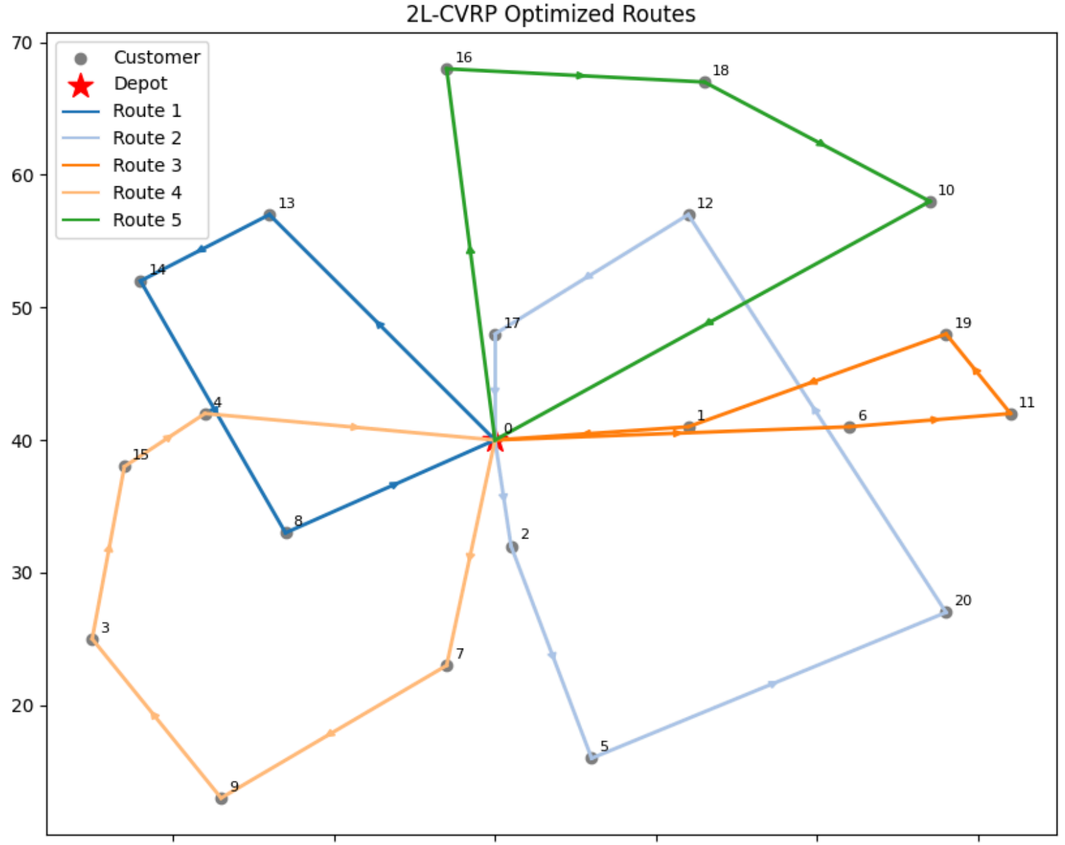
1. High Fixed Cost is the least challenging.
2. When maxStag > 1 and tabu tenure > 0, optimality gap = 0.
3. Increasing maxStag significantly improves solution quality.
4. maxStag = 1, no tabu setting helps □ insufficient search depth.

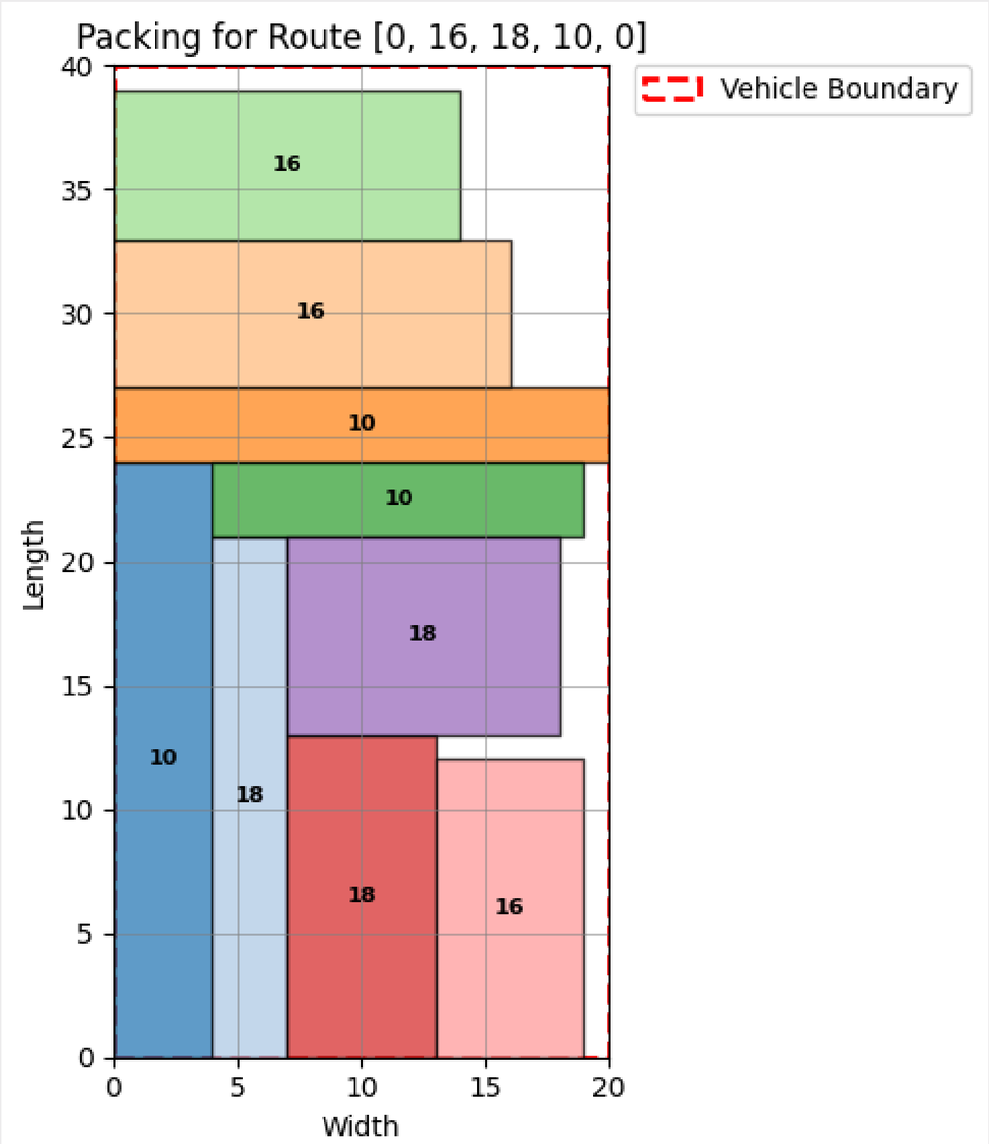

High Fixed Cost
optimality gap
0.080%
0.070%
0.060%
0.050%
0.040%
0.030%
0.020%
0.010%
0.000%
0
1
5
10
20
tabu tenure
maxStag = 1
maxStag = 5
maxStag = 10
maxStag = 20

2L-CVRP

A Guided Tabu Search for the Vehicle Routing Problem with two-dimensional loading constraints

Visualization





Background & Objectives

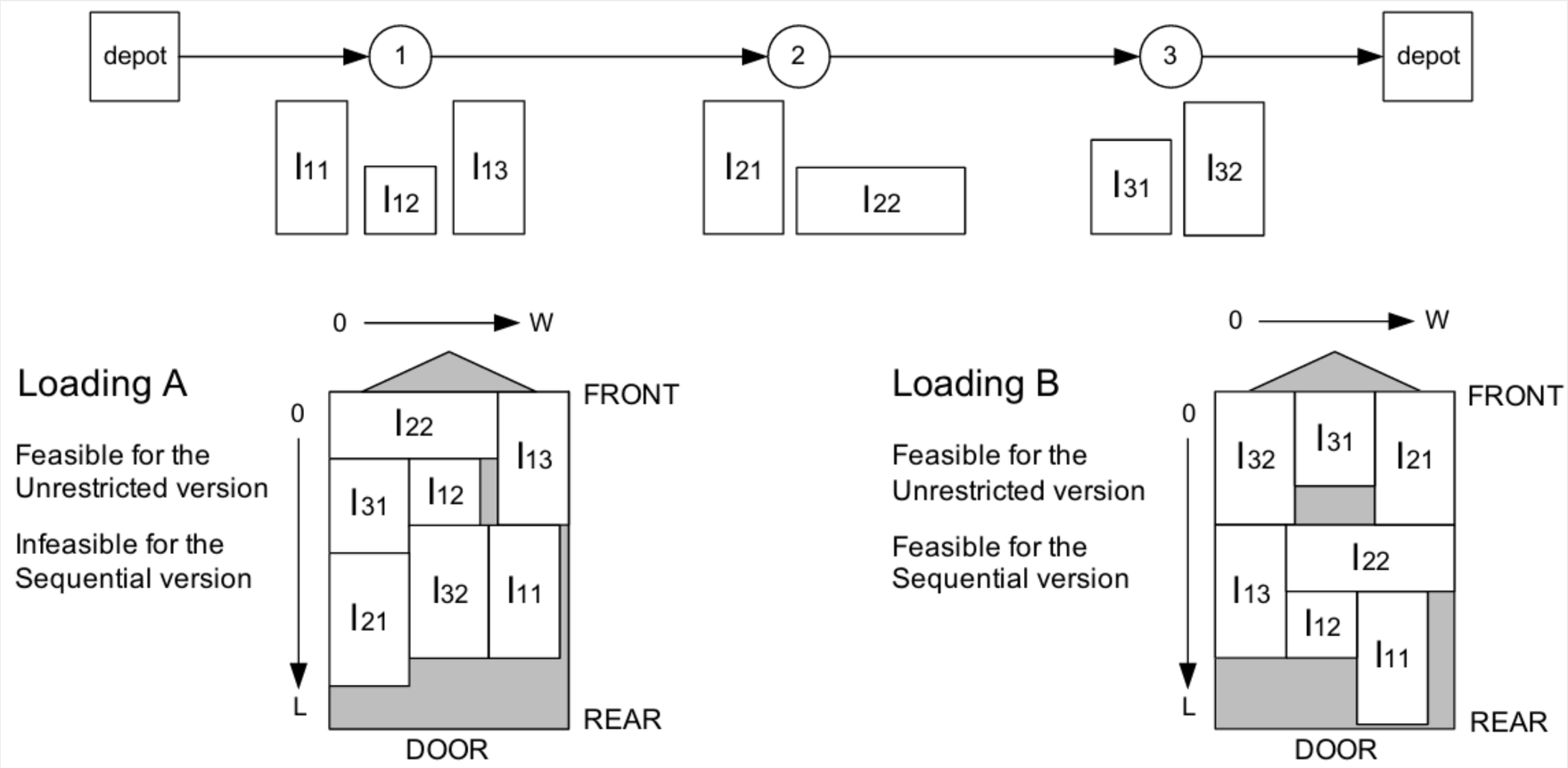
- Problem: Vehicle Routing Problem with two-dimensional loading constraints
  - Customers demand multiple rectangular items
  - Must optimize both routing and 2D packing
- Motivation:
  - Guided Tabu Search (GTS) can escape local optima and handle routing with packing feasibility checks
- Objectives:
  a. Implement full GTS for 2L-CVRP
  b. Verify feasibility of both routes & packings
  c. Test on given benchmark instances

Packing Heuristics

- Five Heuristics (applied in order)
  - Heur1 – Bottom-Left(W-axis)
  - Heur2 – Bottom-Left(L-axis)
  - Heur3 – Maximize touching perimeter (incl. walls)
  - Heur4 – Maximize touching perimeter (excl. walls)
  - Heur5 – Minimize leftover free area
- Feasibility Check
  - Total weight $\leq$ vehicle capacity
  - All items placed (else route is infeasible)

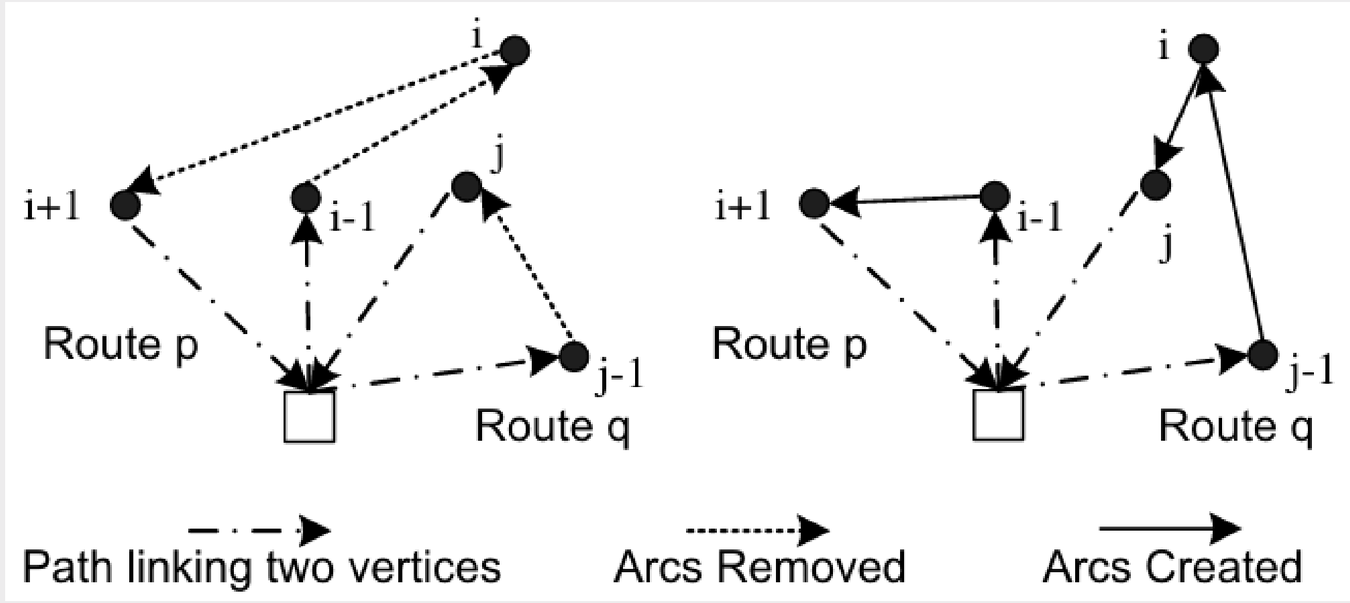
Packing Heuristics

- Ordering of Items
  - Unrestricted mode: Vehicle load ≤ capacity Q, all items can be placed on the L×W loading surface without overlap
  - Sequential mode: Meets Unrestricted constraints and items must be unloaded in delivery order via straight-line movement

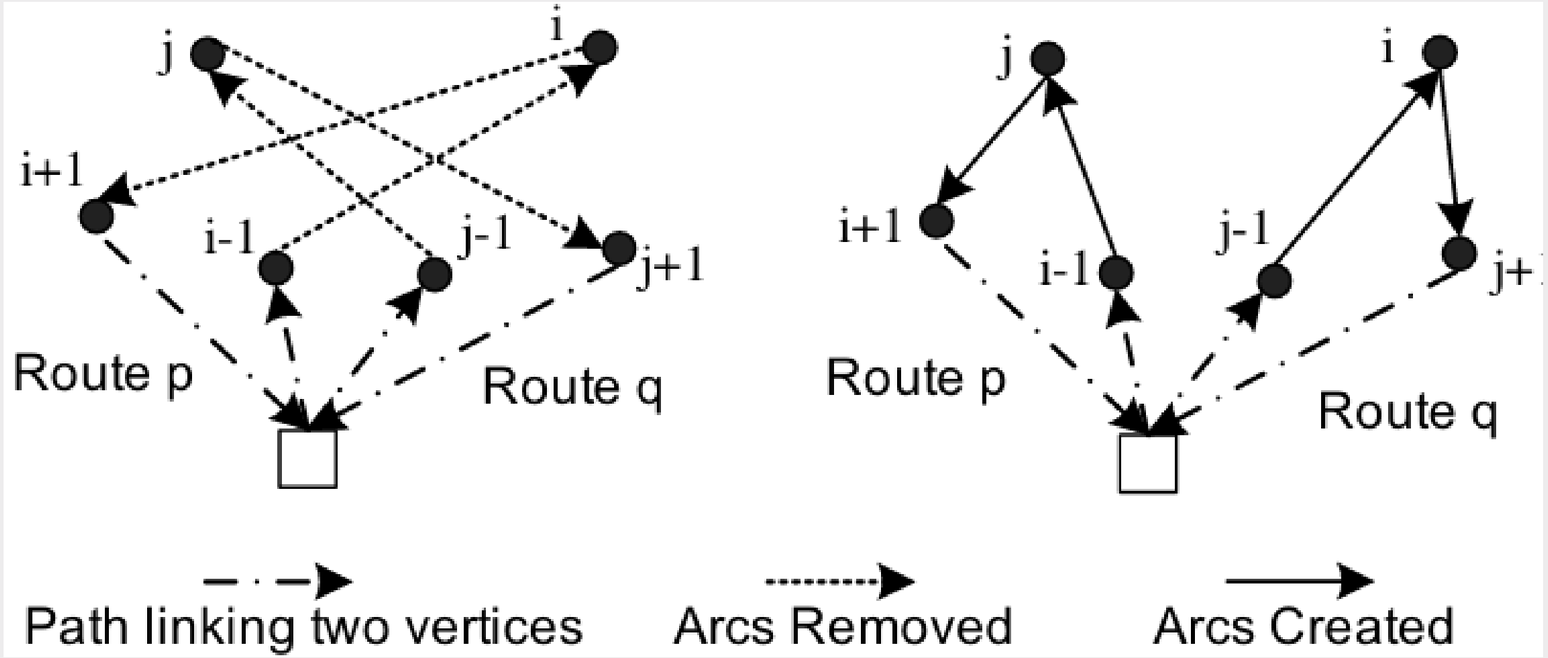


Initial solution & Neighborhood Operators

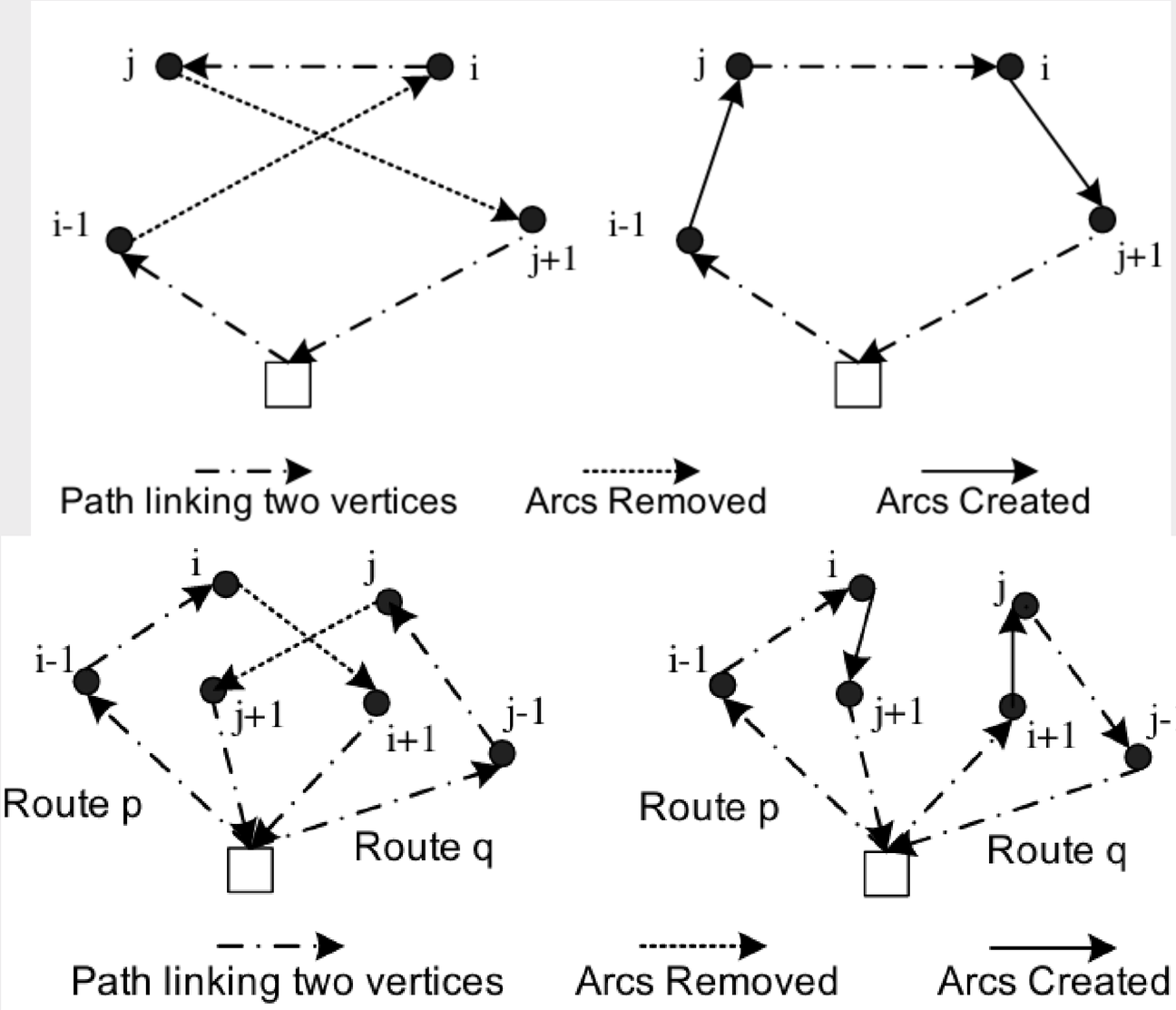
- Best-Fit-Decreasing:
  - Customers are sorted by decreasing item area and inserted into vehicle routes to minimize unused loading space and added travel cost
- NS1 (Relocation)
  - Remove one customer from a route and insert into another



- NS2 (Route exchange)
  - Exchange one customer from route A with one from route B



- NS3 (2-opt)
  - Intra-route segment reversal or cross-exchange tails of two routes



Guided Tabu Search Flow

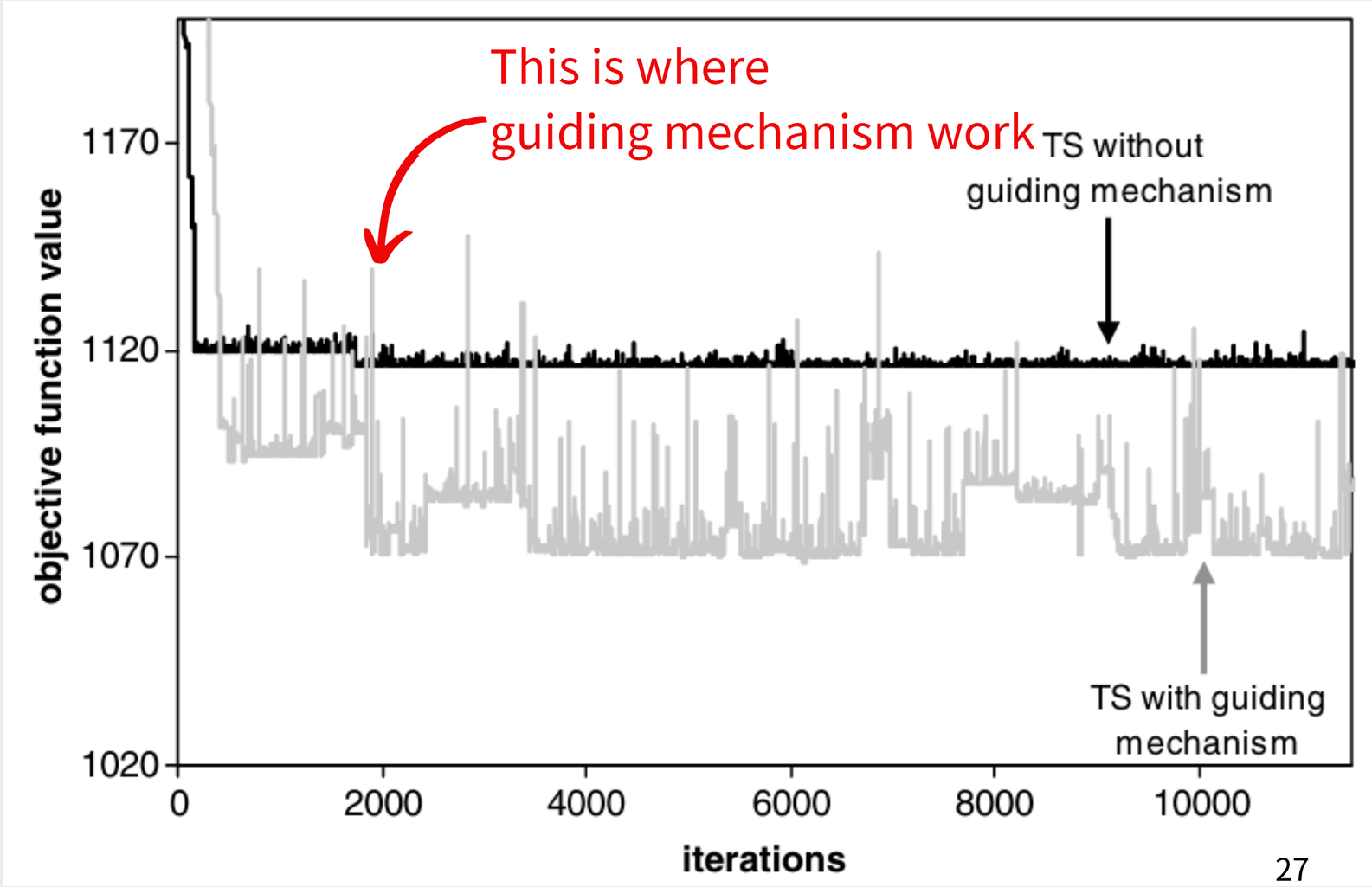
- Initial Solution
  - Greedy Best-Fit-Decreasing
- Iterate until stop
  - Generate neighbors via NS1/NS2/NS3
  - Apply penalties every guidFreq iterations on the “worst” edge
  - Tabu list prevents reversing recent moves
  - Update current solution, best solution, tabu list
- Stopping criteria
  - No improvement after several iterations

What is “Guided” Tabu Search

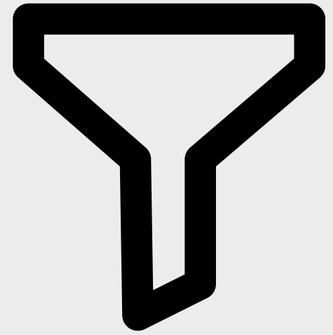
Using utility function to penalize the longest arc present in the solution

$$U(i,j) = \frac{\frac{c_{ij}}{\text{avg}_{ij}}}{1 + p_{ij}}$$

The cost of the penalized arc is doubled for the next guidFreq/2 iterations of the TS body.



Strategy to accelerate searching



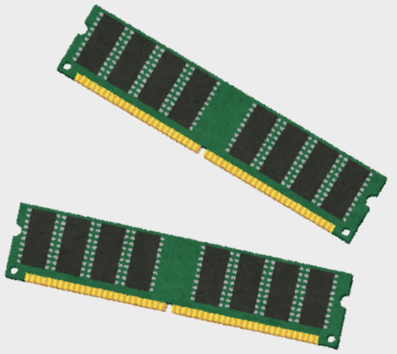
1. Neighborhood Reduction via “Average-Cost” Filter
   - Concept: Avoid evaluating every possible move by only considering “reasonably cheap” edges.
   - Compute for each node i the average cost of its incident edges:

$$\text{avg}_i = \frac{1}{|N(i)|} \sum_{k \in N(i)} c_{i,k}$$

   - Candidate Neighbors NV_i : only include node j if

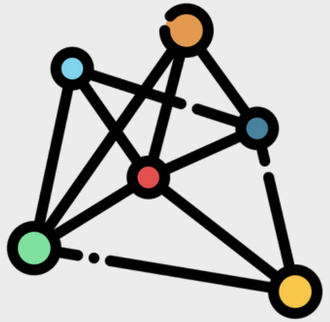
$$c_{i,j} \leq \text{avg}_i \quad \text{and} \quad c_{i,j} \leq \text{avg}_j$$

Strategy to accelerate searching



2. Caching Two‑Dimensional Loading Feasibility
   - Maintain a **memory structure** keyed by the exact sequence of customers in a route.
     i. When a new route sequence appears, run the five‑heuristic packing check → store “feasible/infeasible” in cache.
     ii. If the same route sequence reappears, skip re‑running packing heuristics and use the cached result.
   - Periodic Clearing: Every 2000 iterations, clear old cache entries to free memory (since older routes become unlikely to reoccur).
   - Benefit: Reduces overall CPU time by roughly 33–54%, with no loss of solution quality.

Numerical Experiment



- 10 benchmark problems of Iori et al. (2007) and Gendreau et al. (in press), for the Unrestricted version.
- Our implementation obtained feasible solutions for 5 out of 10 instances, revealing implementation challenges that require further refinement.

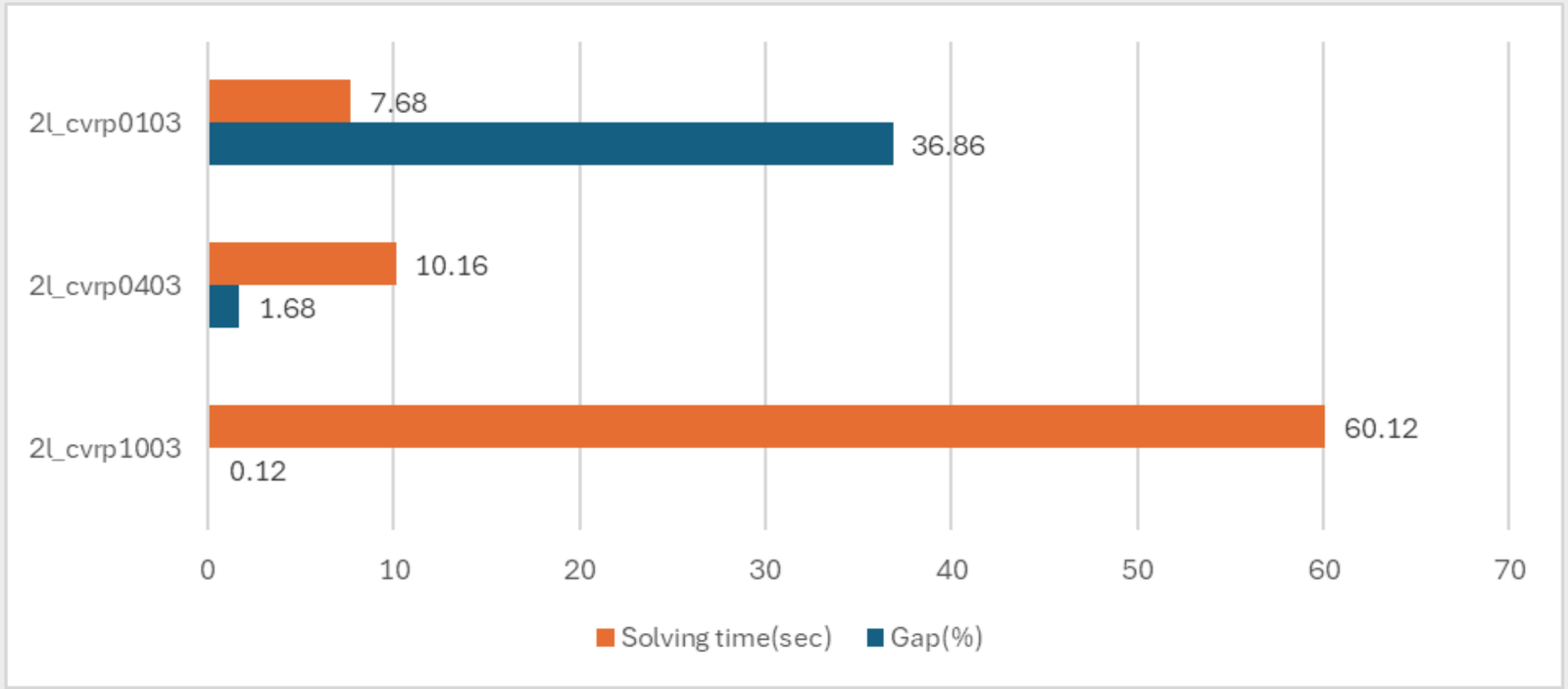
- Potential Issues Identified:
  - Initial solution generation precision and vehicle assignment logic
  - Packing heuristics implementation with coordinate accuracy requirements
  - Constraint validation completeness

Parameter Sensitivity Analysis

CPU: Ryzen 9 5900HS @ 3.28GHz
RAM: 32GB
OS: Windows 11 Home
Development Environment: VS Code
Programming Language: Python 3.10.11

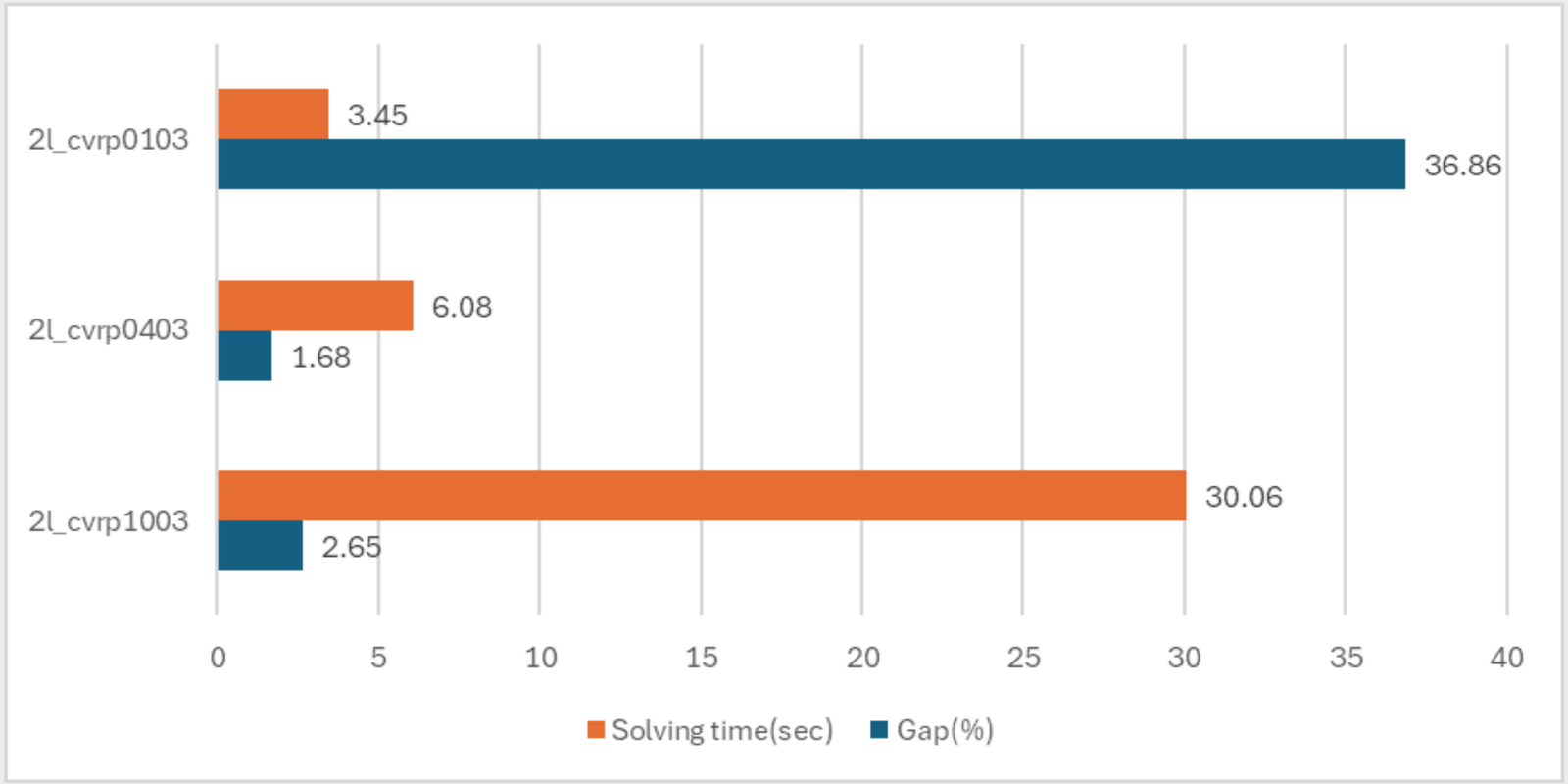
Balance

```
tabu_tenure = 18
tabu_iter_max = 7000
guid_freq = 20
max_time = 60
```



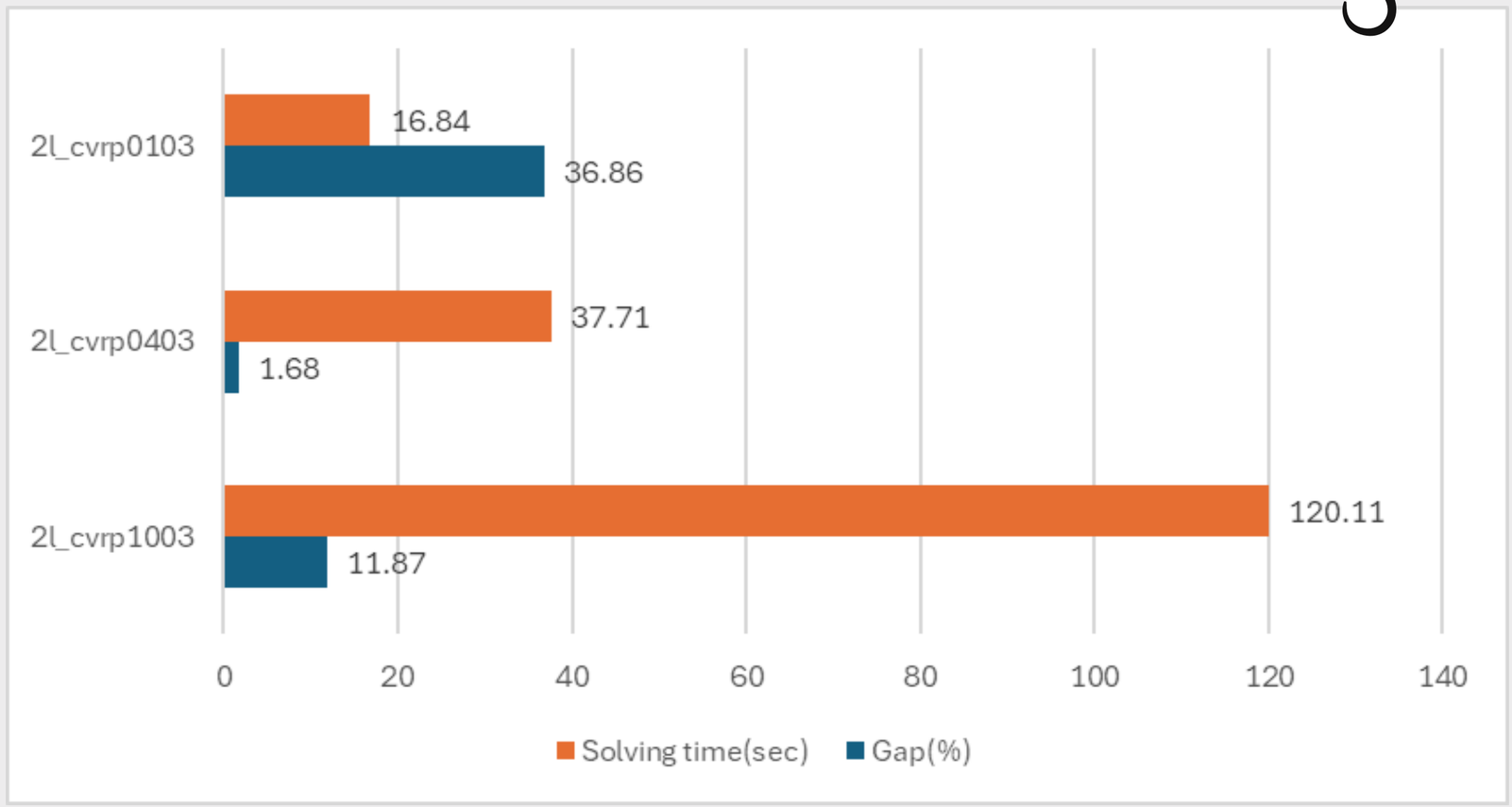
Fast

```
tabu_tenure = 12
tabu_iter_max = 3000
guid_freq = 30
max_time = 30
```



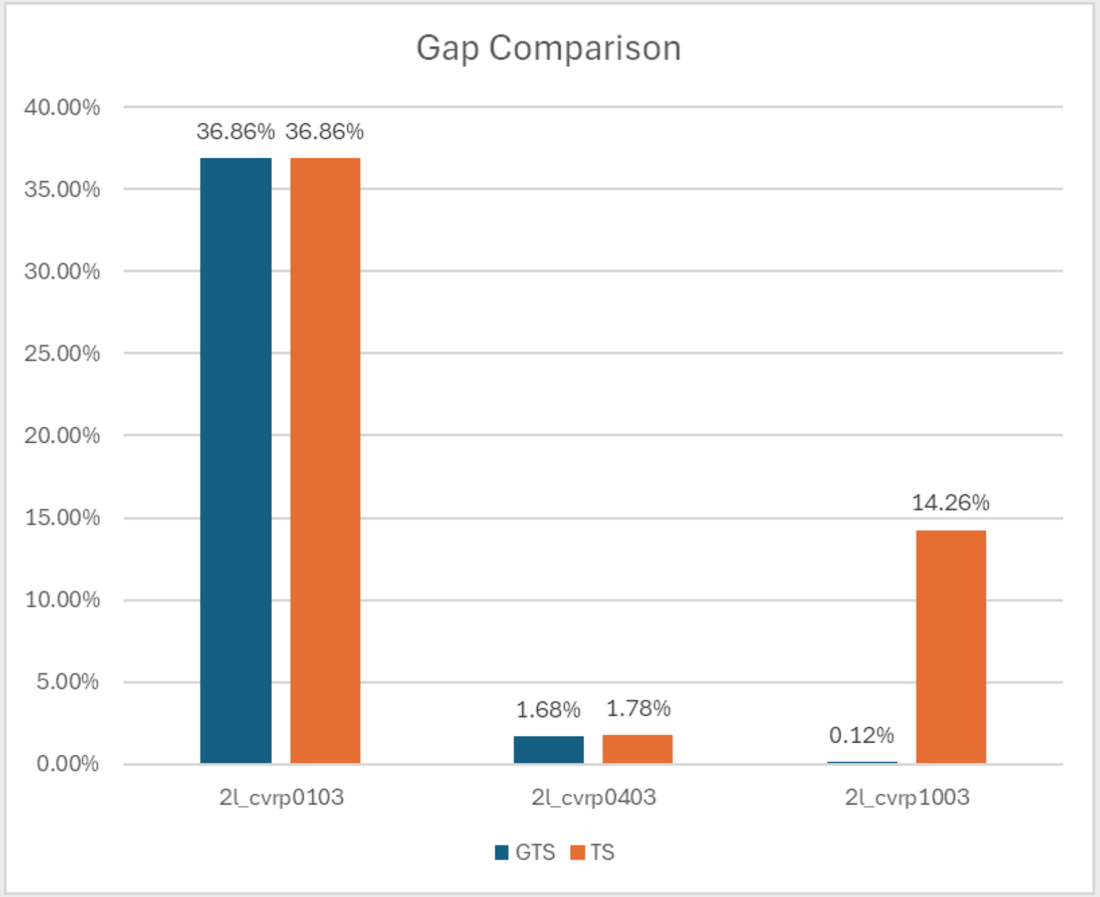
Deep

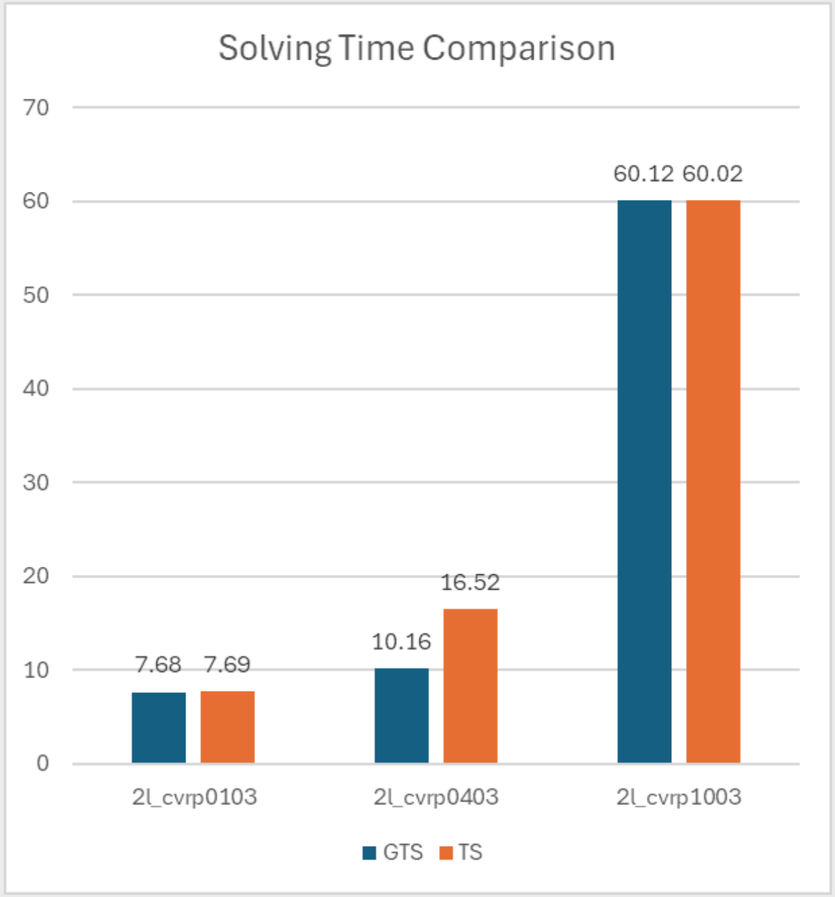
tabu_tenure = 25
tabu_iter_max = 15000
guid_freq = 15
max_time = 120



Guided vs NO Guiding

tabu_tenure = 18
tabu_iter_max = 7000
guid_freq = 100000
max_time = 60





Comparison of strategies

Strategy	Avg. Cost	Avg. Gap(%)	Avg. Solving time(sec)
Fast	512.46	13.73 / 2.17	13.2
Balance	506.94	12.89 / 0.93	25.99
Deep	532.61	16.8 / 6.78	58.22
No Guiding	537.97	17.63 / 8.02	28.08

Conclusion

- We solved 2L-CVRP by combining routing and 2D packing in one Guided Tabu Search.
- Penalty on the worst arc plus a Tabu list helps escape local optima, outperforming plain TS.
- “Balance” parameters achieved the best trade-off: cost = 506.94, gap = 12.89%, time ≈ 26 s.
- Future work: refine routing and packing accuracy to solve all 10 or more benchmark problems, incorporate additional constraints for further performance gains.

Reference

- D. Ghosh, Neighborhood search heuristics for the uncapacitated facility location problem, European Journal of Operational Research 150 (2003) 150–162.
- L. Michel, P. Van Hentenryck, A simple tabu search for warehouse location, European Journal of Operational Research 157 (2004) 576–591.
- E.E. Zachariadis, C.D. Tarantilis, C.T. Kiranoudis, A guided tabu search for the vehicle routing problem with two-dimensional loading constraints, European Journal of Operational Research 195 (2009) 729–743.

- Declaration: We have used ChatGPT, Github Copilot and Grok to write codes

The End