Team DMS

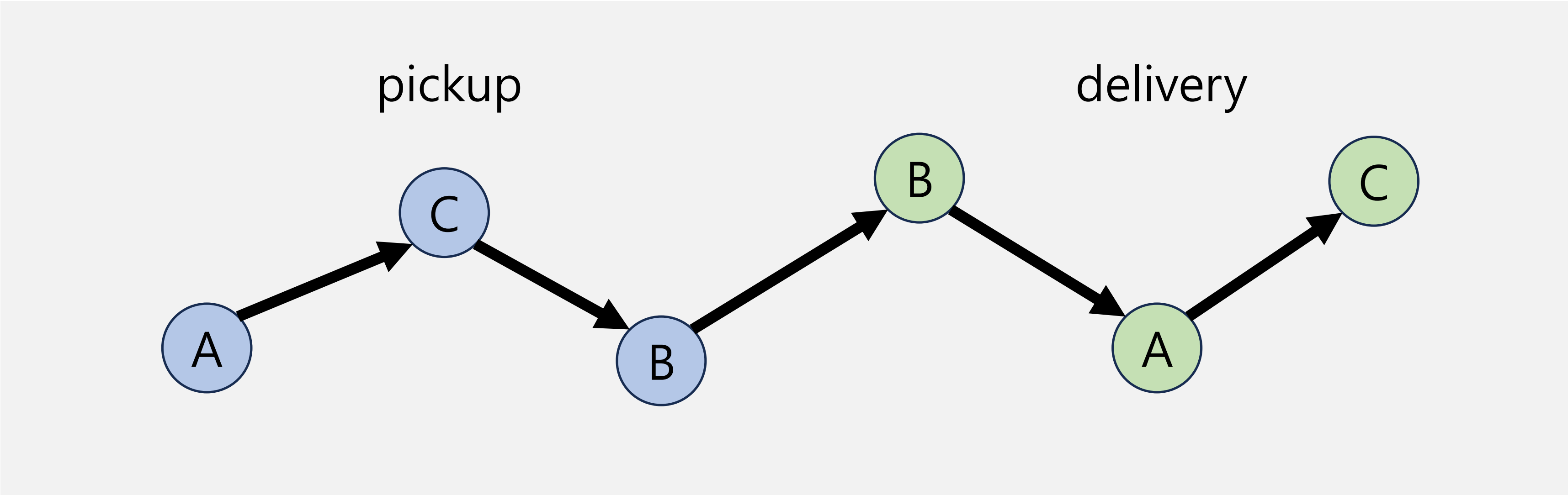
Wonjae Jang, Seoyong Lee

jangwj1026@snu.ac.kr, sylee2685@snu.ac.kr

Optimization Grand Challenge 2024

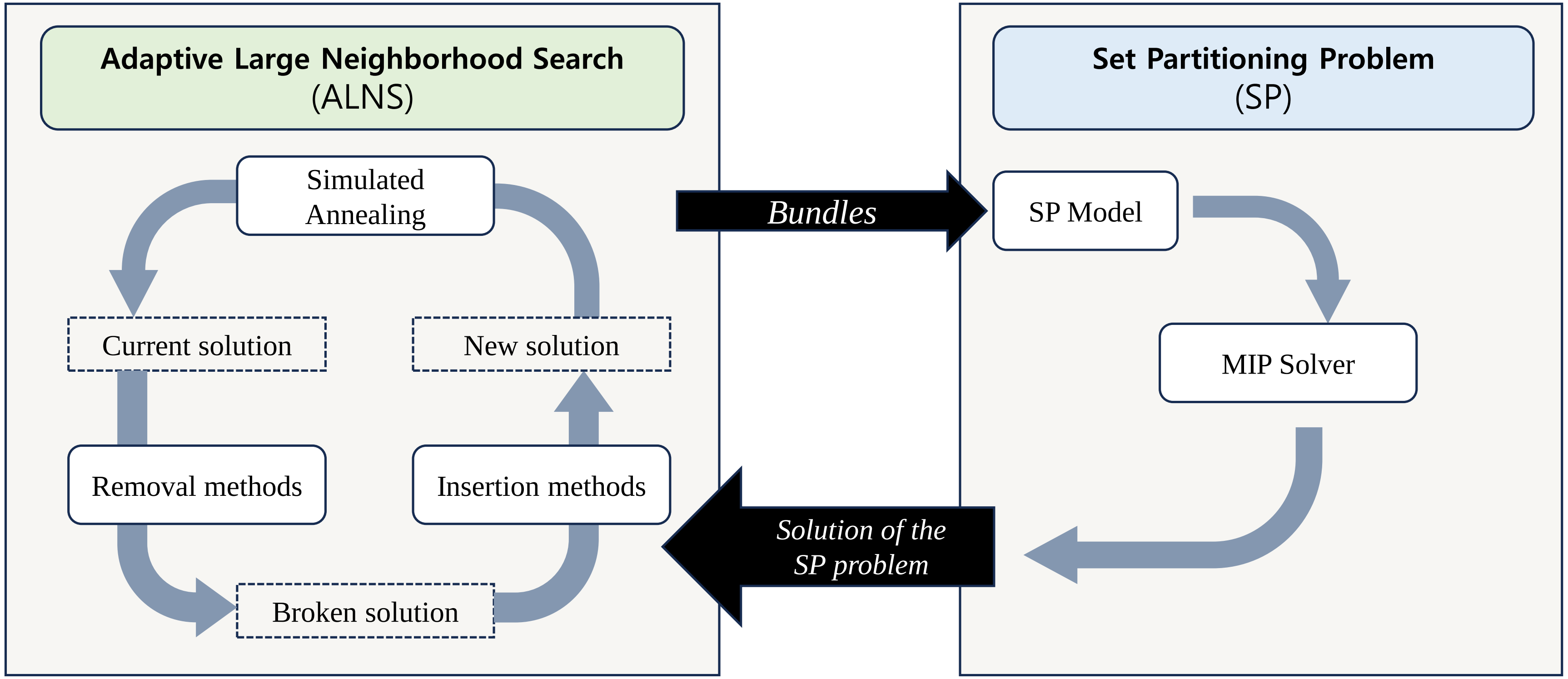
Problem Summary

A Variant of the Pickup and Delivery Problem with Time Windows for Capacitated and Heterogeneous Vehicle Fleets



Algorithm : High level View

Our approach : Iteratively apply the Adaptive Large Neighborhood Search (ALNS) and Set Partitioning (SP) models in succession.



Algorithm : High level View

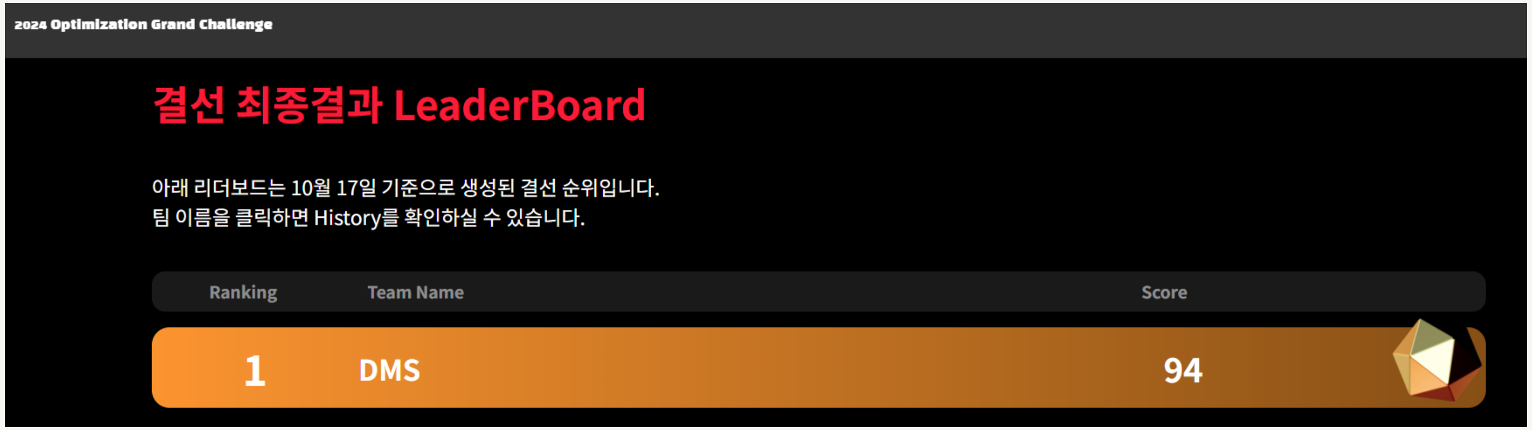
Algorithm 1 Main Algorithm

```
procedure SOLVE(L)                                  ▷ Time list L
    S ← φ                                           ▷ Solution S
    B ← φ                                           ▷ Bundles B
    for (T_ALNS, T_SP) ∈ L do
        (S, B) ← ALNS(S, B, T_ALNS)
        S ← SP(S, B, T_SP)
    end for
    return S
end procedure
```

```
procedure ALNS(S, B, T)                             ▷ Time limit T
    if S = φ then
        S ← generateInitialSolution()
    end if
    p ← Uniform                                     ▷ prob dist p for choosing removal method
    S_best ← S
    while spent time < T do
        Randomly select op from p and n_remove      ▷ destroy method op
        S̃ ← APPLYDELETION(op, n_remove, S)
        S̃ ← APPLYINSERTION(S̃ )
        Store the routes of S̃ in B
        if acceptanceCriteria(S̃ , S) then           ▷ Simulated Annealing
            S ← S̃
        end if
        if cost(S̃ ) < cost(S_best) then
            S_best ←S̃
        end if
        Update p                                    ▷ based on performance : Adaptive!
    end while
    return (S_best, B)
end procedure

procedure SP(S, B, T)
    model ← buildSetCoveringModel(S,B)
    sol_SC ← solveMIP(model, timelimit)
    sol_SC ↦ sol_SP
    return sol_SP
end procedure
```

"So, How Did We Clinch the Victory?"



	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Ours	2574.61	2597.28	1826.98	1803.32	2761.76	2124.65	2977.45	2140.61	2176.57	2692.06
Others	2577.69	2608.17	1862.10	1908.47	2754.71	2118.11	2972.85	2136.69	2231.24	2692.34
Diff	**-0.12%**	**-0.42%**	**-1.89%**	**-5.51%**	0.26%	0.31%	0.15%	0.18%	**-2.45%**	**-0.01%**

“We need to build the best system, not just the best algorithm!”

- Python to C++ ~*10x* **speedup**
- Multithreading (via OpenMP) ~*2x* **speedup**
- Novel Insertion Method ~*260x* **speedup**[1]
- Performance Engineering ~*3x* **speedup**

[1] Compared with the well-known Regret Repair on large instances of K = 2000

ALNS : Insertion Method

The common insertion methods widely used in literature are

- k-regret insertion
- Greedy insertion

However, they fail to scale for large instances due to their $\mathrm{O}(K^3)$ complexity.

- Calculating regret/best value of a single order takes $O(K)$.
- To insert a single order, need to calculate for all $O(K)$ orders that is removed.
- We have $O(K)$ orders to insert.

Thus, their effectiveness is significantly limited

ALNS : Insertion Method

The next method we can think of is…

- Random Order best insertion
  - Randomly choose the order to be inserted
  - Insert the order in the best location

At first glance, the algorithm seems more foolish. However, due to the $O(K^2)$ nature, it allows more iterations leading to better solutions.

An improvement over before, but still not great.

ALNS : Insertion Method

Intuition : "Evaluating for all bundles during insertion is unnecessary"

Idea : "Evaluate only for the *promising* bundles"

0. Preprocess **score[i][j]** array before ALNS

- The value increases as the difference in time and distance becomes greater[1]

$$\underbrace{c_{ij}}_{\text{distance}} + \gamma^{WT} \underbrace{\max\{e_j - \tau_i - \delta_{ij} - l_i, 0\}}_{\text{minimum waiting time}} + \gamma^{TW} \underbrace{\max\{e_i + \tau_i + \delta_{ij} - l_j, 0\}}_{\text{minimum penalty}}$$

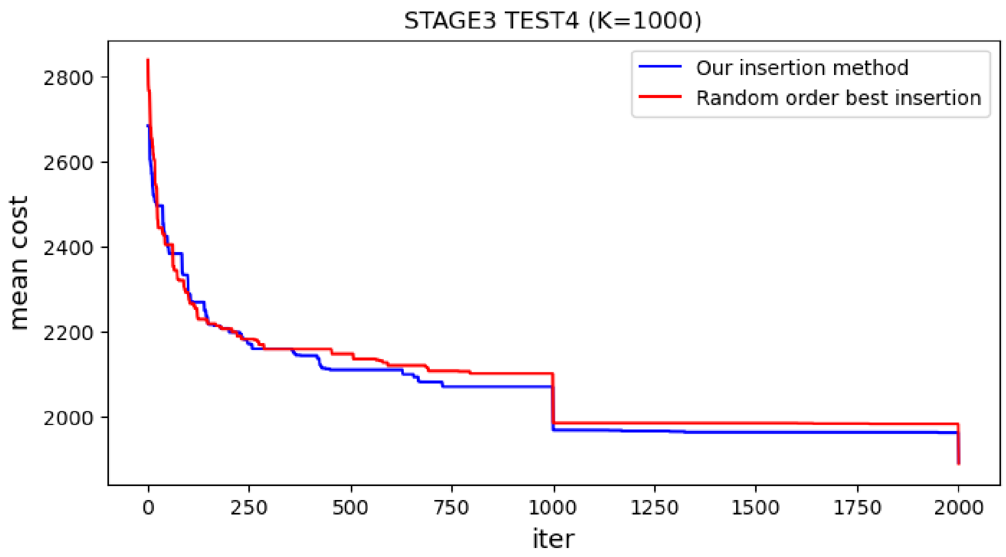
1. For a given order to insert, calculate how promising the bundle is

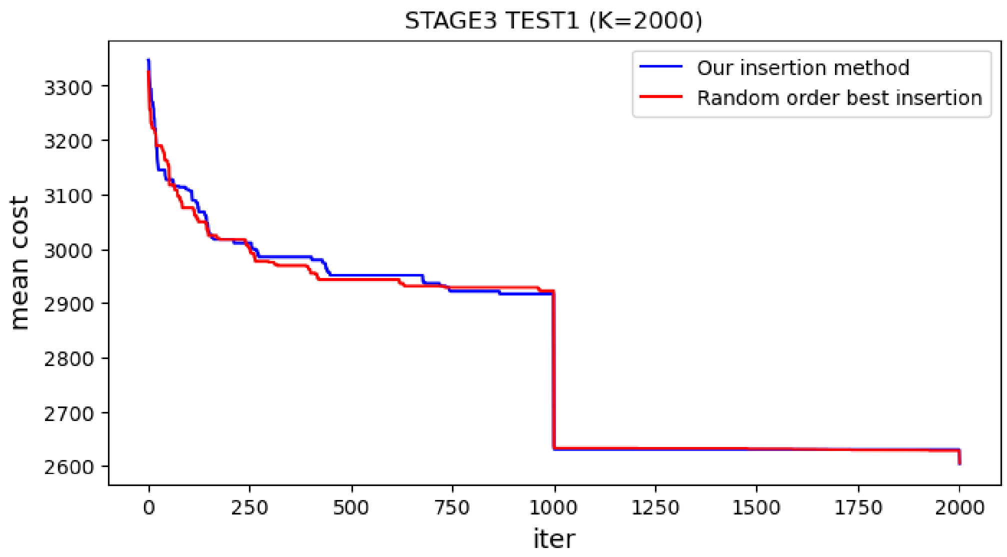
- $\text{promising}(order_i, bundle_B) \coloneqq -\frac{\sum_{order_j \in B}(score[i][j])}{|bundle_B|}$

2. Select the 10-20 most promising bundles and evaluate only those.

[2] Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, *40*(1), 475-489. https://doi.org/10.1016/j.cor.2012.07.018

ALNS : Insertion Method





No deduction in score for the same number of iterations!

	Regret	Random order best	Ours
Avg time per iteration (K=2000)	7.007s (**260x**)	0.1229s (**4.6x**)	0.0269s

Speedup gained ➔ More iterations ➔ Better ALNS result + More bundles ➔ Better SP results

ALNS : Removal Methods

orders to remove = $(0.05 \sim 0.15) \times K$

1. Random removal
2. Distance oriented removal (pickup, delivery & pickup+delivery)
3. Route removal
4. Shaw removal
5. Historical action pair removal
6. Worst removal
7. Semi-worst removal

"Even a suboptimal removal method improves the solution,
and using more diverse methods yields better results."

ALNS-SP : Additional Insights

- **Time distribution** between ALNS and SP plays a critical role.
- **Annealing factor** (temperature decay) is a crucial component.
- **Gurobi NoRel heuristics** significantly impact performance.
- **Acceptance criteria** need to be adjusted

	First ALNS & SP	Latter ALNS & SP
Acceptance Criteria	Simulated Annealing	Hill Climbing Search
Solver Search Method	Full Exact	Full NoRel Heuristic

Proposal for Next Year…

1. Unexplored problems in the literature

2. Commercial solver