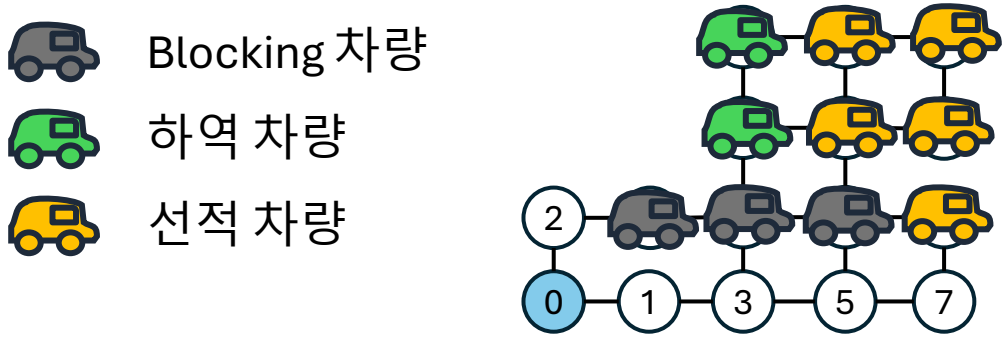
Optimization Grand Challenge 2025

2025.10.19

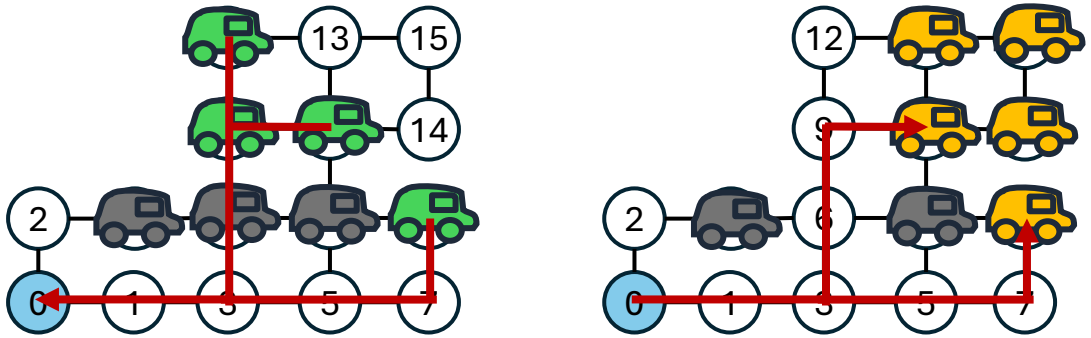
Team **Wirtz**

문제 분석

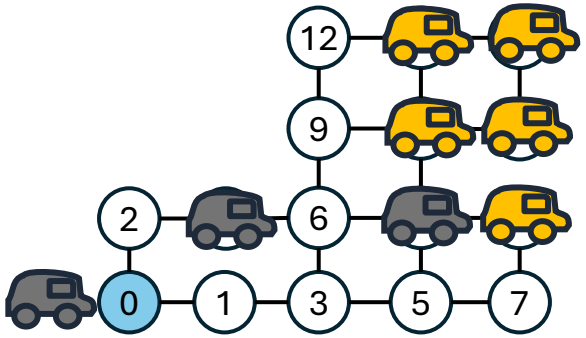
패킹(Packing) 문제
수요 차량들의 적재 노드 결정



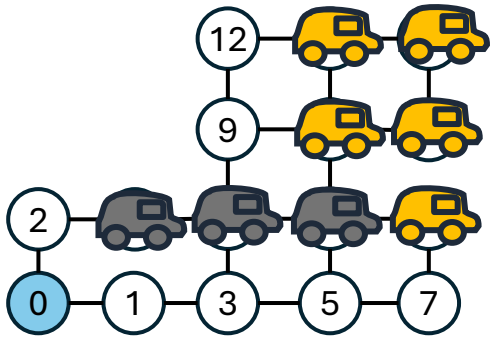
흐름(Flow) 문제
최소 비용의 선적/하역 경로 결정



재배치(rehandling) 문제
재배치 차량 적재 노드 결정



문제 단순화
임시 하역 후 **기존 위치**로 재적재



수리 모형(M1)

Sets

N : 적재 노드 집합
E : 단방향 간선 집합
E': 양방향 간선 집합 $(E' = E \cup \{(j,i)|(i,j) \in E\})$
D : 수요 집합
P : 항구 집합

Parameters

F : 고정비
Q_p : 항구 p에서 선적+하역 차량 수, $\forall p \in P$
qty_d : 수요 d의 차량 수, $\forall\, d \in D$
in_d/out_d : 수요 d의 출발/목적지 항구, $\forall d \in D$

Decision variables

$y_{d,i}$: 수요 d를 노드 i에 적치하면 1, 아니면 0, $\forall d \in D, i \in N$
$f_{i,j,p}$: 항구 p에서 간선 (i,j)를 지나는 경로 수, $\forall (i,j) \in E', p \in P$
$s_{i,p}$: 항구 p에서 노드 i에서 재배치 필요하면 1, $\forall i \in N, p \in P$

- Small scale ~ Medium scale : 시간 내에 최적에 가까운 결과 도출
- Large scale(예선 연습 문제 10번[1]) : 정수해를 빠르게 찾지 못함

1) $|N| = 363, |E| = 500, |P| = 15, |D| = 62, \sum_{d \in D} qty_d = 812$

목적 함수

$$\min \sum_{(i,j) \in E'} \sum_{p \in P} f_{i,j,p} + 2F \sum_{i \in N} \sum_{p \in P} s_{i,p}$$

패킹(Packing) 제약

$$\sum_{i \in N} y_{d,i} = qty_d \quad , \forall d \in D$$

$$\sum_{\substack{d \in D \\ in_d \le p < out_d}} y_{d,i} \le 1 \quad , \forall i \in N, p \in P$$

흐름(Flow) 제약

$$\sum_{j \in adj(0)} f_{0,j,p} = Q_p \quad , \forall p \in P$$

$$\sum_{j \in adj(i)} f_{j,i,p} - \sum_{j \in adj(i)} f_{i,j,p} = \sum_{\substack{d \in D \\ in_d = p \vee out_d = p}} y_{di} \quad , \forall i \in N, p \in P$$

$$\sum_{j \in adj(i)} f_{i,j,p} \le Q_p \left(1 - \sum_{\substack{d \in D \\ in_d \le p < out_d}} y_{d,i} + s_{i,p} \right) \quad , \forall i \in N, p \in P$$

알고리즘(M2) 로직

알고리즘 전략

M1을 활용하여 small scale ~ medium scale에서 강하고
large scale에서 괜찮은 성능을 보이는 알고리즘

Two-Stage 알고리즘(M2)

Stage1 : Rolling horizon 방법으로 빠르게 정수해 구성
Stage2 : 해당 정수해를 초기값으로 사용해 M1을 통한 최적화

Stage1에서 사용되는 보조 행렬

- $status\_mat(|N| \times |P|)$: 각 항구에서 노드의 상태 행렬(0으로 초기화)
  $\{1,2\}$: 선적+하역 차량 수, 0: 빈 상태, -1: blocking 상태

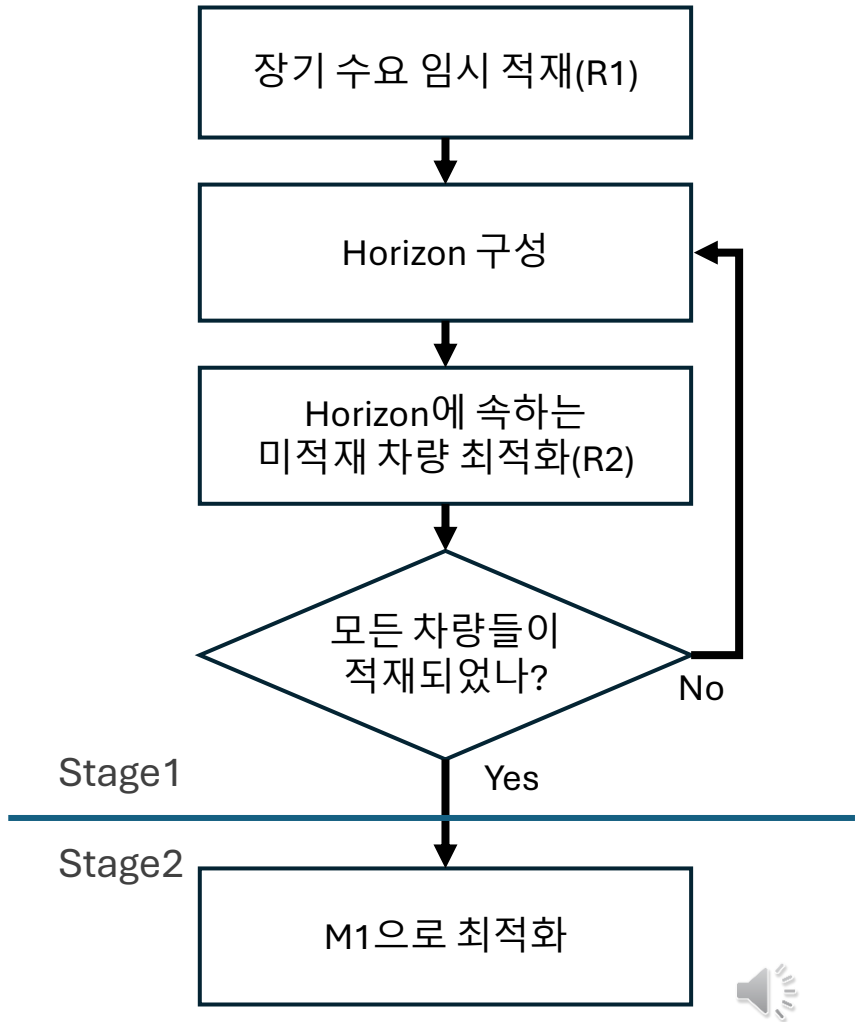
- $dur\_mat(|P| \times |P|)$: 미적재 차량의 수 행렬; 행은 적재 기간, 열은 목적지 항구
  dur_d : 수요 d가 선박에 적재되는 기간 $(out_d - in_d)$

$$dur\_mat[i,j] = \begin{cases} qty_d & \text{if } \exists d \in D \text{ such that } (i = dur_d) \wedge (j = out_d) \\ 0 & \text{otherwise} \end{cases}$$

0	0	0	0	0	0	0	0	0
0	**3**	0	**1**	0	0	0	**9**	0
0	0	**6**	0	**7**	0	0	**3**	0
0	0	0	**19**	0	0	0	0	**20**
0	0	0	0	0	**25**	0	0	0
0	0	0	0	0	**18**	0	**11**	0
0	0	0	0	0	0	0	0	**23**
0	0	0	0	0	0	0	**8**	0
0	0	0	0	0	0	0	0	**4**

적재 기간 = 3,
목적지 = 마지막 항구
미적재 차량 수

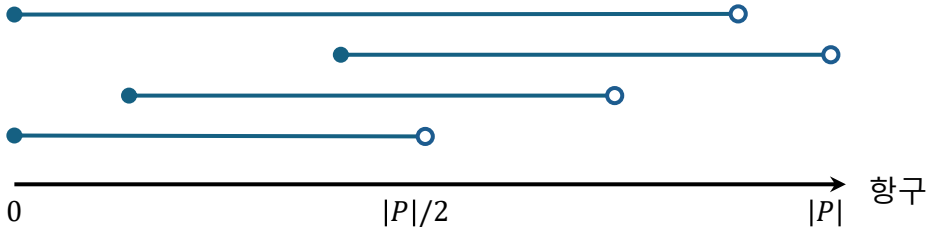
9×9 dur_mat 예시



알고리즘(M2) 로직 – 장기 수요 임시 적재

장기 수요

장기 수요 : 적재 기간(dur_d)가 $|P|/2$ 보다 큰 수요
장기 수요들은 서로 다른 노드에 적재됨
임시 적재했을 때 이후 문제들의 사이즈를 줄일 수 있음



장기 수요들의 적재 기간 도식화

$status\_mat$ 업데이트

For each $(d,i) \in \{(d,i) | \, y_{d,i} = 1, d \in D, i \in I\}$

$status\_mat[i, in_d] \leftarrow status\_mat[i, in_d] + 1$

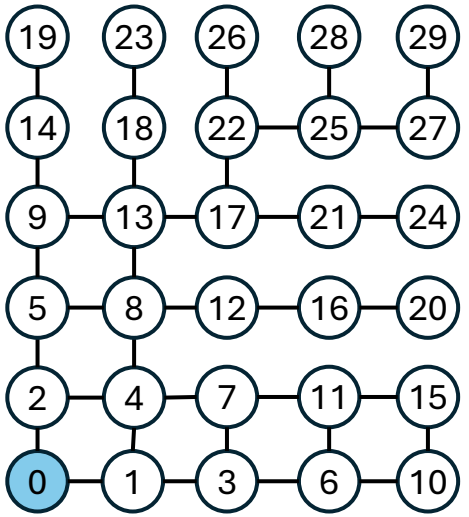
$status\_mat[i, out_d] \leftarrow status\_mat[i, out_d] + 1$

$status\_mat[i, in_d + 1 : out_d] \leftarrow -1$

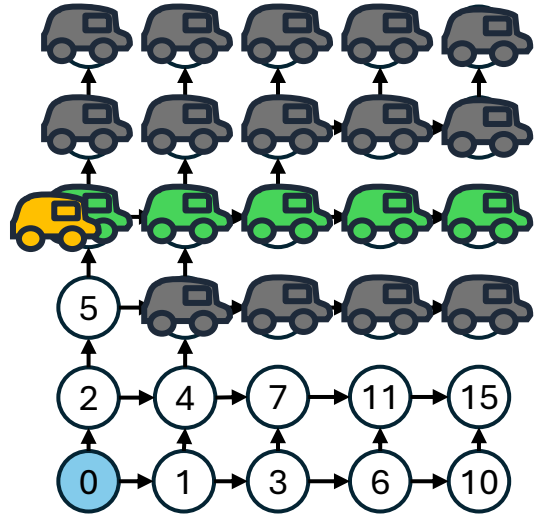
임시 적재 위치 결정

적재 전략 : 출입구에서 거리가 먼 곳에 임시 적재
M1을 경량화한 R1을 사용해 적재 위치 결정
R1 : 경로 비용 최대화, 단방향 그래프로 완화

$$\max \sum_{(i,j)\in E} \sum_{p\in P} f_{i,j,p} - 2F \sum_{i\in N} \sum_{p\in P} s_{i,p}$$



양방향 그래프



단방향 그래프

알고리즘(M2) 로직 – Horizon 구성

Horizon 초기화

Rolling period : 적재 결정 구간 | $h \times w$

1. 너비(w)를 h와 같게 설정
2. 구간$(h \times w)$ 내 미적재 차량 수가 σ를 초과할 때까지 $w \leftarrow w + 1$
   만약 w가 $|P|$에 도달했을 때 조건을 만족하지 못하면,
   $h \leftarrow h + 1$ 이후 위 과정을 다시 반복

Look-ahead period : 함께 고려하는 구간

Rolling period가 결정되면 높이는 δ, 너비는 h만큼 함께 고려

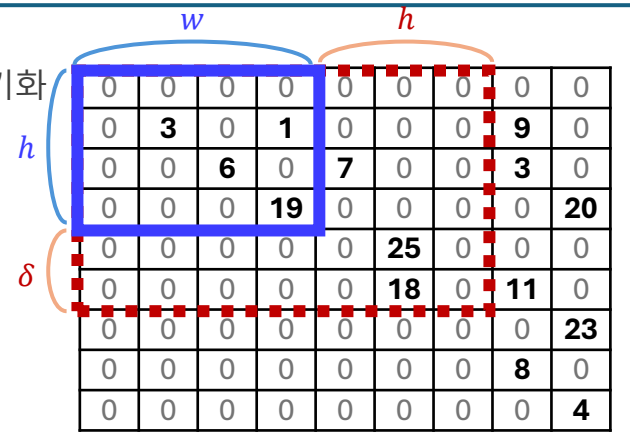
Horizon = Rolling period + Look-ahead period | $(h + \delta) \times (w + h)$[1)]

Horizon에 속하는 수요 집합 D' 생성

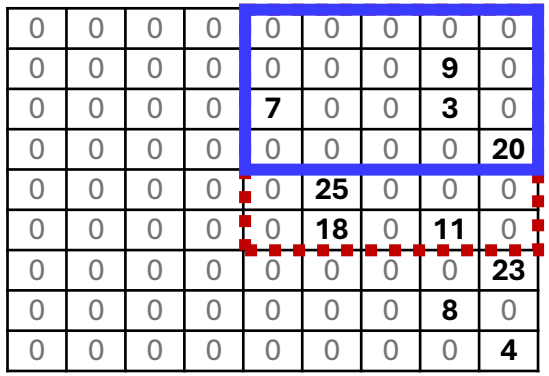
Horizon 업데이트

1. 직전 Rolling period의 마지막 항구 이후부터 Horizon 생성
2. 만약 마지막 항구가 $|P|$에 도달했을 때 미적재 차량 수가 σ를 초과 못하면
   $h \leftarrow h + \delta$ 이후 첫 항구$(p = 0)$부터 Horizon 생성

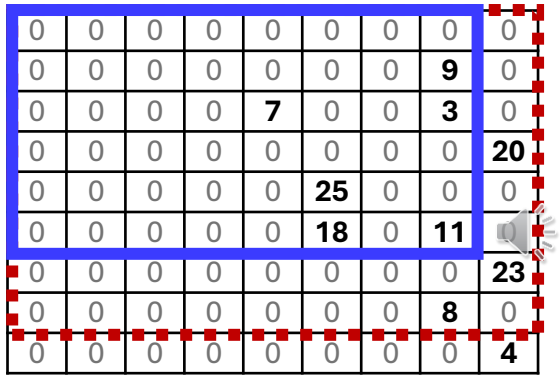
Horizon 초기화



Horizon 업데이트 1



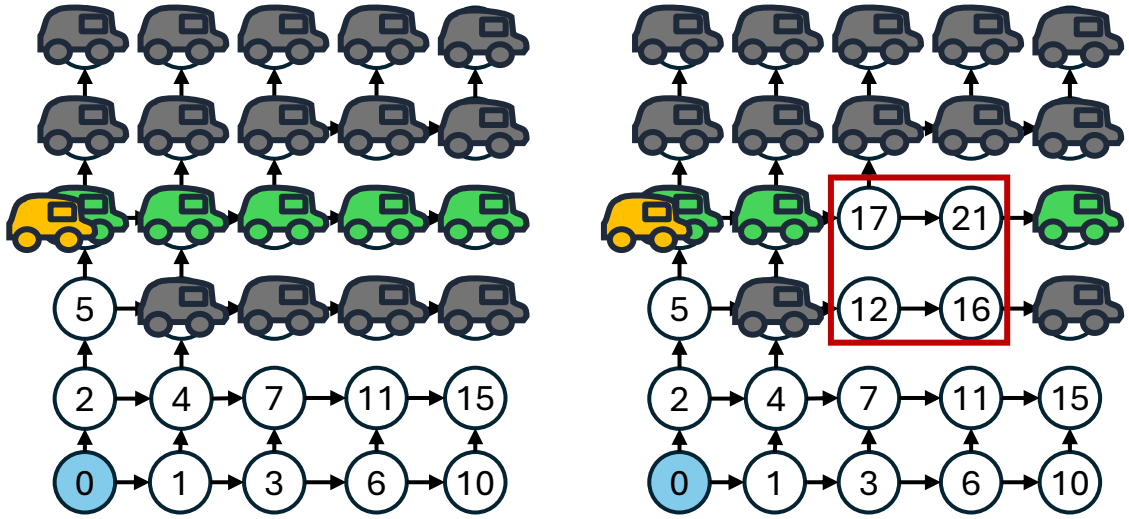
Horizon 업데이트 2



1) 제출한 코드에서는 $\sigma = 40,\ \delta = 2$ 사용

알고리즘(M2) 로직 – Horizon 미적재 차량 최적화

1. D'에 속하는 장기 수요가 임시 적재 상태라면, 적재 취소

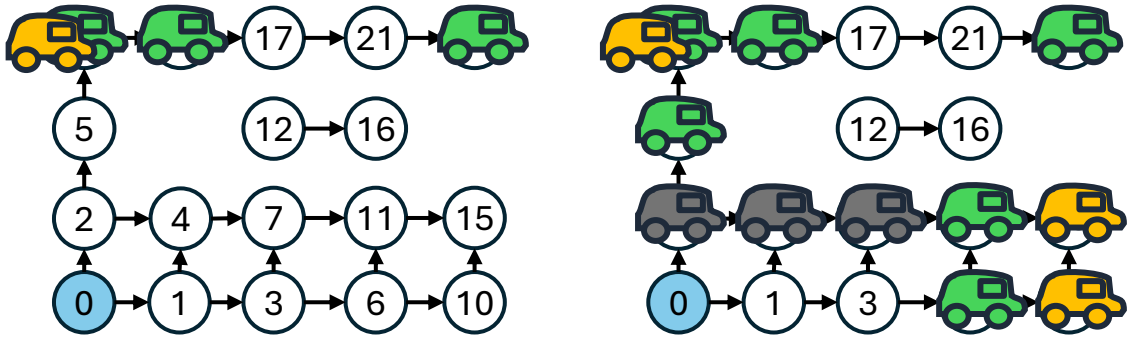


적재 취소 전 / 적재 취소 후

2. M1을 경량화한 R2를 사용해 적재 위치 결정

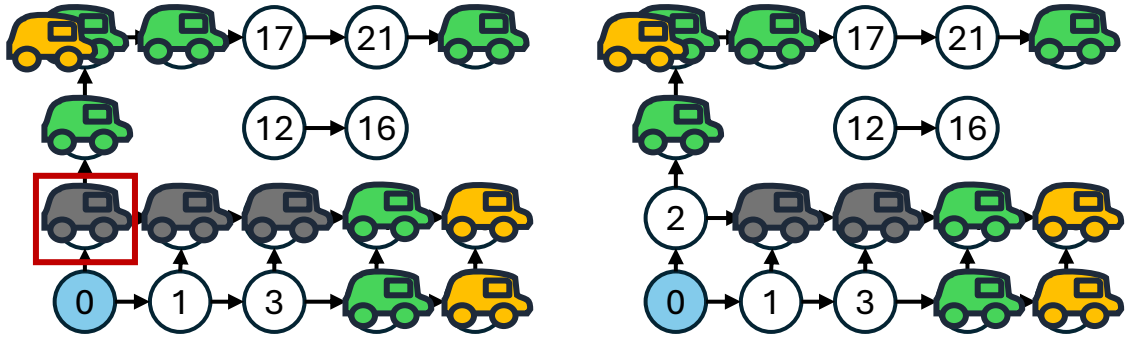
R2 : 단방향 그래프, $status_mat$을 통한 그래프 축소

$$\sum_{j\in in(i)} f_{j,i,p} - \sum_{j\in out(i)} f_{i,j,p} = \sum_{\substack{d\in D(i) \\ in_d = p \vee out_d = p}} y_{di} + \boldsymbol{staus\_mat[i,p]} \quad ,\forall i \in N(p), p \in P'$$



그래프 축소 / 적재 결과

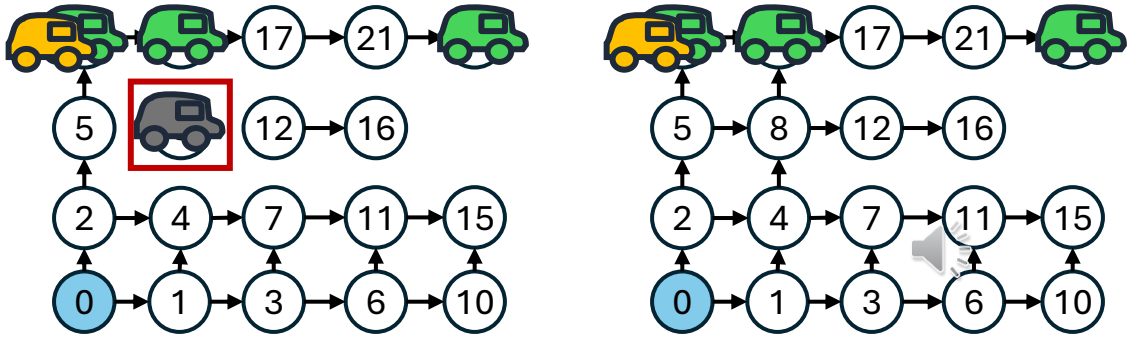
3. R2 결과에서 재적치(rehandling)가 필요한 차량은 적재 취소



노드 2에서 재적치 필요 / Blocking 적재 취소 후

4. R2가 축소한 그래프에서 Infeasible인 경우

그래프가 연결되도록 blocking 차량의 적재 취소



노드 8에 의해 단절 / 연결 그래프

결과 분석

1. M1, M2 performance

Problem (TL)	P1 (600)	P2 (300)	P3 (180)	P4 (120)	P5 (180)	P6 (60)	P7 (240)	P8 (180)	P9 (30)	P10 (120)	P11 (240)	P12 (120)	P13 (180)	P14 (360)
Best	5945	10984	1438	2386	2246	1223	2360	5460	3502	1255	5528	6483	764	4110
제출 [point]	21097 [4]	19638 [6]	1438 [10]	2904 [6]	2468 [7]	1658 [7]	2372 [8]	9860 [1]	6530 [2]	1650 [8]	9313 [3]	11169 [2]	1114 [5]	5552 [3]
M2 (time)	21069 (600)	33394 (300)	1438 (5)	2902 (20)	2253 (126)	1658 (5)	**2356** (240)	6994 (180)	4604 (30)	1650 (3)	6595 (240)	12519 (120)	1114 (0.5)	5552 (57)
M1 (time)	105141 (600)	47052 (300)	1438 (6)	2902 (25)	2253 (90)	1658 (4)	**2356** (240)	6994 (180)	**3297** (30)	1650 (2)	10634 (240)	6777 (120)	1114 (0.1)	5552 (121)

- M1과 M2는 7문제에서 제한시간 내에 단순화한 문제의 최적값 도출
- 제한시간 240 이상인 문제에서 M2는 M1보다 좋은 성능을 보임
- 제한시간 180 이하인 문제에서 M2가 M1보다 나쁜 경우도 존재(P5, P9, P12, P13)

2. Stage1 performance

(1) : $\text{ratio} = \frac{\text{Stage1 time}}{\text{M2 time}} \times 100$

Problem (TL)	P1 (600)	P2 (300)	P3 (180)	P4 (120)	P5 (180)	P6 (60)	P7 (240)	P8 (180)	P9 (30)	P10 (120)	P11 (240)	P12 (120)	P13 (180)	P14 (360)
Stage1 time	104	26	1	3	3	0.3	4	7	11	1	6	9	0.3	3
ratio(%)(1)	17.3	8.7	20	15	2.4	6	1.7	3.9	36.7	33.3	2.5	7.5	60	5.3
LP gap(%)	73.5	72.4	17.8	52.6	71	48.2	37.3	57.8	70	50.4	42.8	59.5	21.8	51

- Stage1은 전체 실행시간에서 평균적으로 15.7% 소요, 평균 51.9% LP gap의 초기해 생성

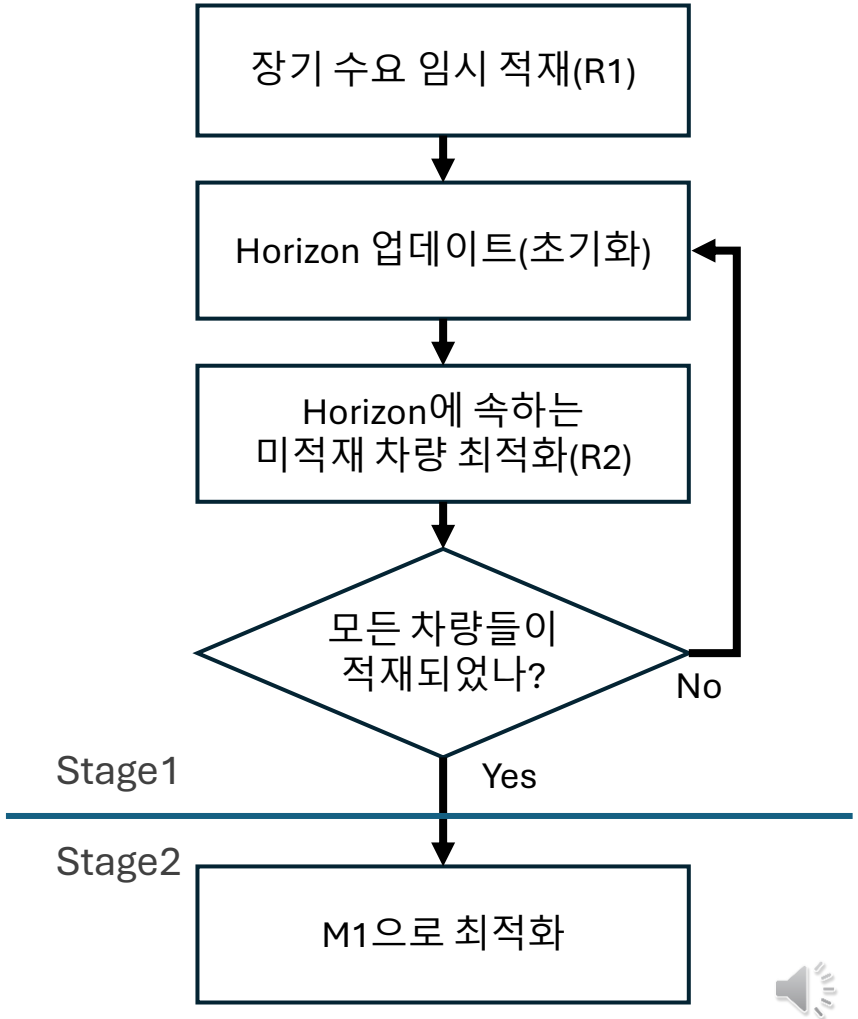
알고리즘 구현 및 특장점

알고리즘 구현

- **구현 언어** : Python
- **사용 패키지** : Gurobi(solver), Numpy
- **구현 기술** :
  - Graph reduction
  - Rolling horizon

알고리즘 특장점

- **단순성** : Python만으로 간단하게 구현 가능한 알고리즘
- **문제 분석** : 문제 특성을 분석 후 복잡도 완화
- **수리 모형** : Benchmark로 사용 가능한 수리 모형



알고리즘 발전 방향 및 후기

알고리즘 발전 방향

- **파라미터 최적화 :** 파라미터(σ, δ)를 튜닝했을 때 약간의 개선이 있을 것으로 기대됨
- **고급 최적화 기법 적용 :** 대회 기간 중 Benders decomposition과 dual method로도 접근하였으나, 충분히 정교하지 못해 성능이 좋지 않았다. 고도화한다면 좋은 성능을 보일 수 있을 것 같다.
- **재배치 알고리즘 :** 재배치 알고리즘은 다음과 같이 구상하였으나 시간이 부족하여 개발하지 못함
  1. 재배치가 필요한 시점을 기준으로 해당 차량을 나누어, 더미 수요를 생성하고 최적화
  2. parallel beam search를 n번 시도
- **프로그래밍 언어 :** 수리 모형과 휴리스틱이 결합된 알고리즘을 C++에서 개발하려 했지만 license를 인식하지 못해 실패

경진대회 참여 후기

- 같은 문제를 두고 여러 팀이 경쟁하는 것이 정말 재밌었고 좋은 경험이라 생각합니다.
- 결선에서 실행 결과를 하루에 한 번 공개하면 좋을 것 같습니다.
- 경진대회 진행을 위해 애써주신 운영진분들께 깊이 감사드립니다.

THANK YOU

Appendix – R1

Sets

N : 적재 노드 집합
E : 단방향 간선 집합
E': 양방향 간선 집합 $(E' = E \cup \{(j,i)|(i,j) \in E\})$
D : 수요 집합
P : 항구 집합

Parameters

F : 고정비
Q_p : 항구 p에서 선적+하역 차량 수, $\forall p \in P$
qty_d : 수요 d의 차량 수, $\forall\, d \in D$
in_d/out_d : 수요 d의 출발/목적지 항구, $\forall d \in D$

Decision variables

$y_{d,i}$: 수요 d를 노드 i에 적치하면 1, 아니면 0, $\forall d \in D, i \in N$
$f_{i,j,p}$: 항구 p에서 간선 (i,j)를 지나는 경로 수, $\forall (i,j) \in E', p \in P$
$s_{i,p}$: 항구 p에서 노드 i에서 재배치 필요하면 1, $\forall i \in N, p \in P$

$$\max \sum_{(i,j)\in E}\sum_{p\in P} f_{i,j,p} - 2F\sum_{i\in N}\sum_{p\in P} s_{i,p}$$

$$\sum_{i\in N} y_{d,i} = qty_d \quad ,\forall d \in D$$

$$\sum_{\substack{d\in D \\ in_d \le p < out_d}} y_{d,i} \le 1 \quad ,\forall i \in N, p \in P$$

$$\sum_{j\in out(0)} f_{0,j,p} = Q_p \quad ,\forall p \in P$$

$$\sum_{j\in in(i)} f_{j,i,p} - \sum_{j\in out(i)} f_{i,j,p} = \sum_{\substack{d\in D \\ in_d = p \vee out_d = p}} y_{di} \quad ,\forall i \in N, p \in P$$

$$\sum_{j\in out(i)} f_{i,j,p} \le Q_p \left(1 - \sum_{\substack{d\in D \\ in_d < p < out_d}} y_{d,i} + s_{i,p}\right) \quad ,\forall i \in N, p \in P$$

Appendix – R2

$$\min \sum_{p \in P'} \sum_{(i,j) \in E(p)} f_{i,j,p} + 2F \sum_{p \in P'} \sum_{i \in N(p)} s_{i,p}$$

$$\sum_{i \in N(d)} y_{d,i} = num_d \quad , \forall d \in D'$$

$$\sum_{\substack{d \in D(i) \\ in_d \le p < out_d}} y_{d,i} \le 1 \quad , \forall i \in N(p), p \in P'$$

$$\sum_{j \in out(0)} f_{0,j,p} = Q'_p \quad , \forall p \in P'$$

$$\sum_{j \in in(i)} f_{j,i,p} - \sum_{j \in out(i)} f_{i,j,p} = \sum_{\substack{d \in D(i) \\ in_d = p \vee out_d = p}} y_{di} + staus\_mat[i,p] \quad , \forall i \in N(p), p \in P'$$

$$\sum_{j \in out(i)} f_{i,j,p} \le Q'_p \left(1 - \sum_{\substack{d \in D(i) \\ in_d < p < out_d}} y_{d,i} + s_{i,p} \right) \quad , \forall i \in N(p), p \in P'$$

Sets and parameters

$N(p)$: 항구 $p \in P'$에서 blocking 아닌 노드 집합

$N(d)$: 수요 $d \in D'$ 적재 가능한 노드 집합

$D(i)$: 노드 $i \in N$에 적재 가능한 수요 $d \in D'$ 집합

num_d : 수요 $d \in D'$의 적재되지 않은 차량 수