Optimization Grand Challenge 2025

Roll-on / Roll-off 최적화

Team: 300

(연세대학교 지능형데이터·최적화학과 석사과정) 홍유빈
(연세대학교 산업공학과 석사과정) 장동현
(연세대학교 산업공학과 석사과정) 정동운

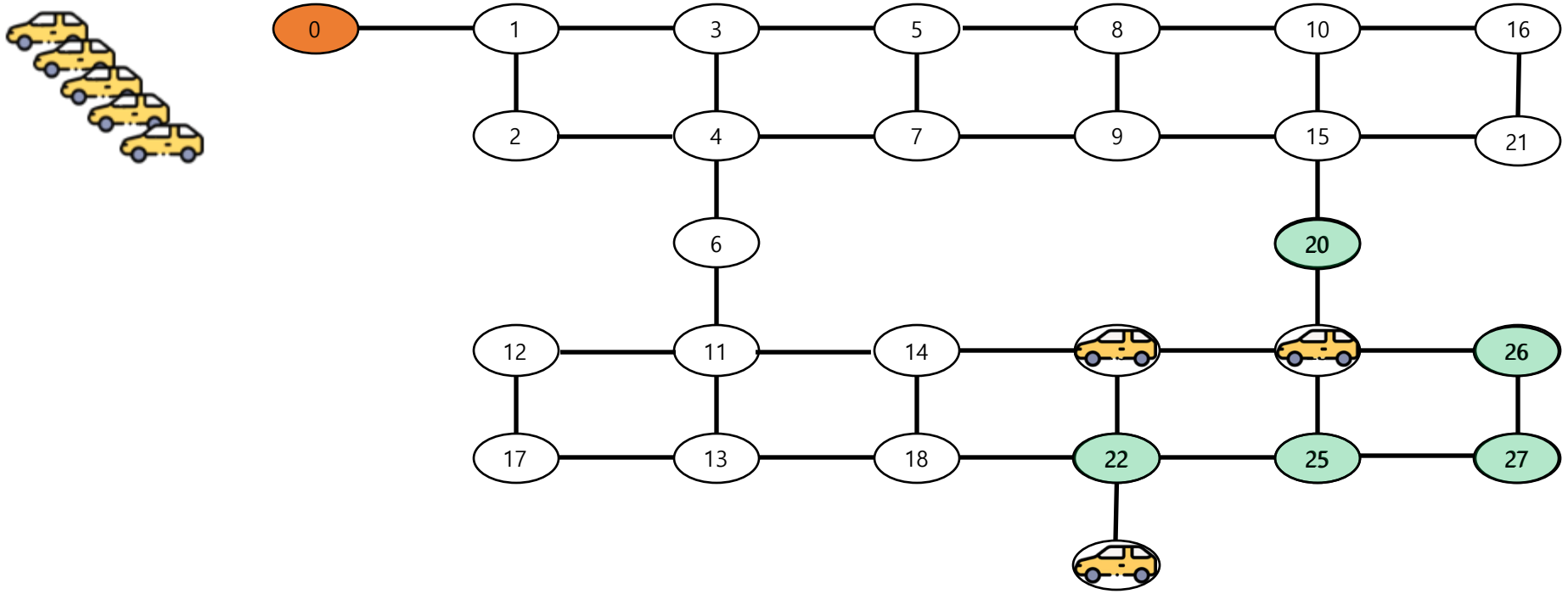
본선 & 결선 알고리즘

문제의 핵심 고려 사항

1. 문제 정보 (수요 정보, 선박 그래프 구조 등)를 바탕으로 **적절한 차량 배치 (+ 위치변경, 재적재), 경로, 순서** 결정을 통해 고정비와 변동비를 줄이는 문제
2. 이전 항구의 의사결정이 이후 상태와 의사결정에 영향을 주는 “**Multi-stage Problem**”

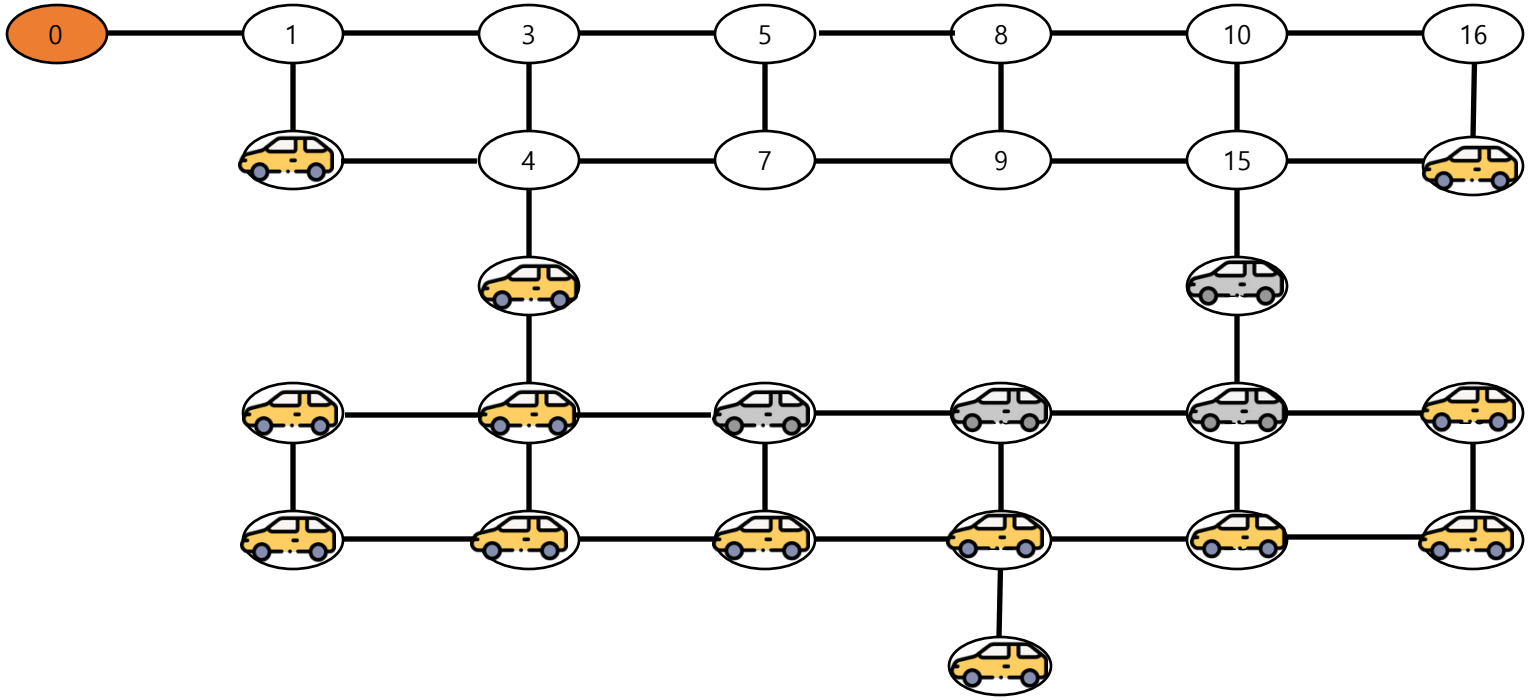
본선 & 결선 알고리즘

- 위치 변경을 제외하면 각 항구에서 **선적 차량 배치 및 하역 차량 결정이 “적절”하다면 상하역 과정에서 차량 이동의 “순서”는 고려 대상 X**
  - **Why?: 최단 이동 경로를 보장하는 Greedy Rule 존재**
  - **선적:** 배치된 정점들 중 출입구와의 최단 이동 거리가 먼 정점부터 순서대로 선적하면, **최단 경로로 선적 가능**
  - **하역:** 출입구와의 최단 이동 거리가 가까운 차량부터 순서대로 하역한다면, **최단 경로로 하역 가능**



본선 & 결선 알고리즘

- 위치 변경을 제외하면 각 항구에서 **선적 차량 배치 및 하역 차량 결정이 “적절”하다면 상하역 과정에서 차량 이동의 “순서”는 고려 대상 X**
  - **Why?: 최단 이동 경로를 보장하는 Greedy Rule**
  - **선적:** 배치된 정점들 중 출입구와의 최단 이동 거리가 먼 차량부터 순서대로 선적하면, 최단 경로로 선적 가능
  - **하역:** 출입구와의 최단 이동 거리가 가까운 차량부터 순서대로 하역한다면, **최단 경로로 하역 가능**



본선 & 결선 알고리즘

문제의 핵심 고려 사항

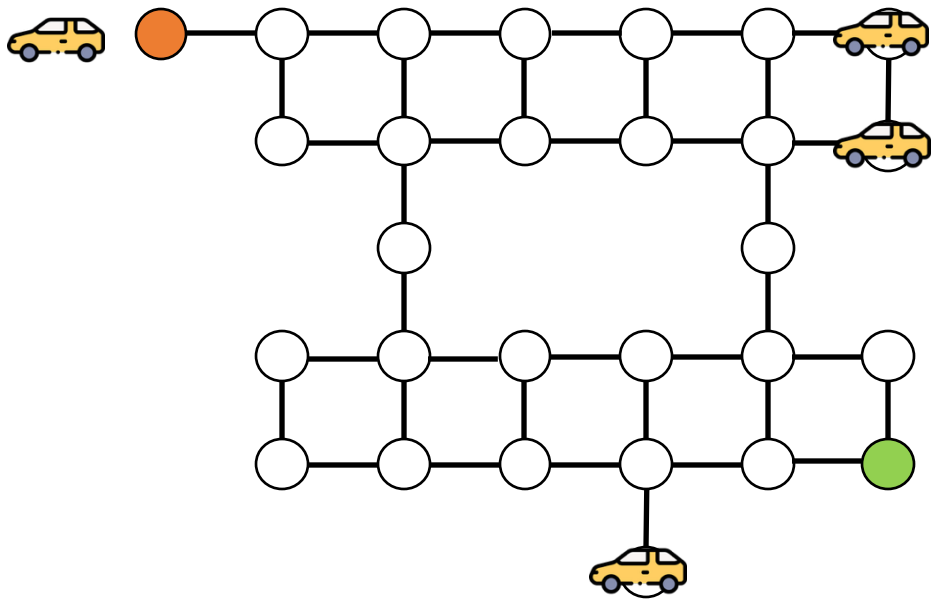
1. 문제 정보 (수요 정보, 선박 그래프 구조 등)를 바탕으로 **적절한 차량 배치 (+ ~~위치변경~~, 재적재), 경로,** ~~순서~~ 결정을 통해 고정비와 변동비를 줄이는 문제
2. 이전 항구의 의사결정이 이후 상태와 의사결정에 영향을 주는 "**Multi-stage Problem**"

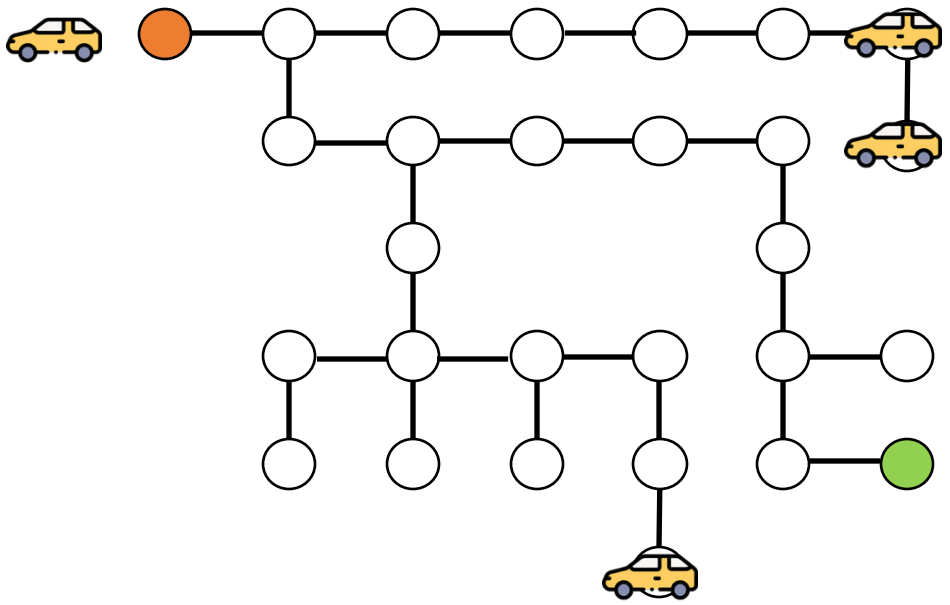
➢ 수리적 최적화 모형 (혼합 정수 계획법, MILP)

- **목적지가 다른 차량들을 어떤 경로로 어느 정점에 배치할 지, 그리고 어떤 차량들을 어느 경로로 하역 (혹은 임시 하역) 할 지 결정** (자세한 수리식은 Appendix A 참고)
- Gurobi Solver 등을 활용하여 최적해를 찾을 수 있으나, 문제 규모가 커질 경우 **오랜 시간이 소요될 수 있음**
- 대회에서 주어지는 **"제한 시간"** 내에 좋은 해를 찾지 못하는 경우 발생

본선 & 결선 알고리즘

- **그래프를 트리로 재구성**
  - **경로 선택의 복잡성 축소**
  - **기존 구조**: 출입구(or 정점)에서 정점(or 출입구)으로 이동하는 **복수의 경로 존재 가능**
  - **트리 구조**: 경로가 하나로 **유일**





본선 & 결선 알고리즘

문제의 핵심 고려 사항 (트리)

1. 문제 정보 (수요 정보, 선박 그래프 구조 등)를 바탕으로 **적절한 차량 배치 (+ ~~위치변경~~, 재적재), ~~경로~~-~~순서~~**-결정을 통해 고정비와 변동비를 줄이는 문제
2. 이전 항구의 의사결정이 이후 상태와 의사결정에 영향을 주는 “**Multi-stage Problem**”

➢ 수리적 최적화 모형 (정수 계획법, IP)

- ➢ **목적지가 다른 차량들을 ~~어떤 경로로~~ 어느 정점에 배치할 지, 그리고 어떤 차량들을 ~~어느 경로로~~ 하역 (혹은 임시 하역)할 지 결정**
- **장점**: 경로를 탐색하지 않아 해공간이 줄어들어 **제한된 시간에 이점**
- **단점**: 최적해를 포함하는 해공간이 줄어들어 **지역해에 빠질 가능성**

본선 & 결선 알고리즘

수리적 최적화 모형 (트리)

❖ 수식에서 회색으로 삭제된 부분은 트리 형태로 문제를 재구성하면서 제거되는 변수와 제약조건을 의미

Indices
$\boldsymbol{P} = [0,1,\ldots,P-1]$: Port Index | $\boldsymbol{N} = [1,2,\ldots,N-1]$: Node Index

Decision Variables
$x_{pnd} \in \{0,1\}$: 1 if node n is occupied by a vehicle with destination d at the end of port p loading phase
$u_{pnd} \in \{0,1\}$: 1 if node n is occupied by a vehicle with destination d at the end of port p unloading phase
$y_{pnd} \in \{0,1\}$: 1 if vehicle with destination d is loaded to node n at port p
$v_{pnd} \in \{0,1\}$: 1 if vehicle at node n with destination d is unloaded at port p
~~$fl_{pn_1n_2} \in \mathbb{R}^+$: Amount of vehicle flow from node n_1 to node n_2 in loading phase at port p~~
~~$fu_{pn_1n_2} \in \mathbb{R}^+$: Amount of vehicle flow from node n_1 to node n_2 in unloading phase at port p~~

Parameters
F: Fixed cost for loading / unloading / rehandling
LB: Lower bound of problem (given)
C_{pd}: Cumulative total demand to destination d at port p
D_n: Shortest distance from origin to node n (or from node n to origin)
$\boldsymbol{S_n}$: Nodes in shortest paths from origin to node n (or from node n to origin)

Objective Function

$$\min\ (F + D_n) * \left[\sum_{p\in\boldsymbol{P}}\sum_{n\in\boldsymbol{N}}\sum_{d\in\boldsymbol{P}}[y_{pnd}+v_{pnd}]\right] - LB$$

수리적 최적화 모형 (트리)

❖ 수식에서 회색으로 삭제된 부분은 트리 형태로 문제를 재구성하면서 제거되는 변수와 제약조건을 의미

Constraints

1) Node occupancy constraints

$$\sum_{d\in \boldsymbol{P}} x_{pnd} \le 1\,, \forall p \in \boldsymbol{P}, n \in \boldsymbol{N} \qquad (1-1)$$

$$\sum_{d\in \boldsymbol{P}} u_{pnd} \le 1, \forall p \in \boldsymbol{P}, n \in \boldsymbol{N} \qquad (1-2)$$

2) Demand satisfaction constraints

$$\sum_{n\in \boldsymbol{N}} x_{pnd} = C_{pd}, \forall p \in \boldsymbol{P}, d \in \boldsymbol{P} \qquad (2-1)$$

$$\sum_{n\in \boldsymbol{N}} u_{pnd} = 0, \forall p \ge d \qquad (2-2)$$

3) Initial & Terminal condition constraints

$$u_{0nd} = 0, \forall n \in \boldsymbol{N}, d \in \boldsymbol{P} \qquad (3-1)$$

$$x_{(P-1)nd} = 0\,, \forall n \in \boldsymbol{N}, d \in \boldsymbol{P} \qquad (3-2)$$

4) Vehicle conservation constraints (Linking Constraints)

$$u_{pnd} + y_{pnd} = x_{pnd}, \forall p \in \boldsymbol{P}\backslash(P-1), n \in \boldsymbol{N}, d \in \boldsymbol{P} \qquad (4-1)$$

$$x_{(p-1)nd} - v_{pnd} = u_{pnd}, \forall p \in \boldsymbol{P}\backslash 0, n \in \boldsymbol{N}, d \in \boldsymbol{P} \qquad (4-2)$$

5) Flow conservation constraints

$$\sum_{n_1\in N_n} fl_{pn_1n} - \sum_{n_2\in N_n} fl_{pnn_2} = \sum_{d\in P} y_{pnd}\,, \forall p \in P, n \in N \qquad (5-1)$$

$$\sum_{n_1\in N_n} fu_{pn_1n} - \sum_{n_2\in N_n} fu_{pnn_2} = \sum_{d\in P} v_{pnd}\,, \forall p \in P, n \in N \qquad (5-2)$$

6) Node reachability constraints

$$\sum_{n_1\in N_n} fl_{pn_1n} \le |N| * \left(1 - \sum_{d\in P} u_{pnd}\right), \forall p \in P, n \in N \qquad (6-1)$$

$$\sum_{n_1\in N_n} fu_{pn_1n} \le |N| * \left(1 - \sum_{d\in P} u_{pnd}\right), \forall p \in P, n \in N \qquad (6-2)$$

$$\sum_{d\in P} y_{pnd} \le 1 - \sum_{d\in P} u_{pnd}\,, \forall p \in P, n \in N \qquad (6-3)$$

$$\sum_{d\in P} v_{pnd} \le 1 - \sum_{d\in P} u_{pnd}\,, \forall p \in P, n \in N \qquad (6-4)$$

7) Node reachability constraints (D_n work as big M)

$$D_n\left(1 - \sum_{d\in \boldsymbol{P}} y_{pnd}\right) \ge \sum_{n'\in \boldsymbol{S_n}} \sum_{d\in \boldsymbol{P}} u_{pn'd}\,, \forall p \in \boldsymbol{P}\backslash(P-1), n \in \boldsymbol{N} \qquad (7-1)$$

$$D_n\left(1 - \sum_{d\in \boldsymbol{P}} v_{pnd}\right) \ge \sum_{n'\in \boldsymbol{S_n}} \sum_{d\in \boldsymbol{P}} u_{pn'd}\,, \forall p \in \boldsymbol{P}\backslash 0, n \in \boldsymbol{N} \qquad (7-2)$$

본선 & 결선 알고리즘

문제 1. **어떤 트리 구조**로 문제를 정의할 지

문제 2.(Gurobi Solver) **초기해를 찾는데 많은 시간이 필요**

➢ **초기해 도출 및 트리 구조 선정을 위한 알고리즘**

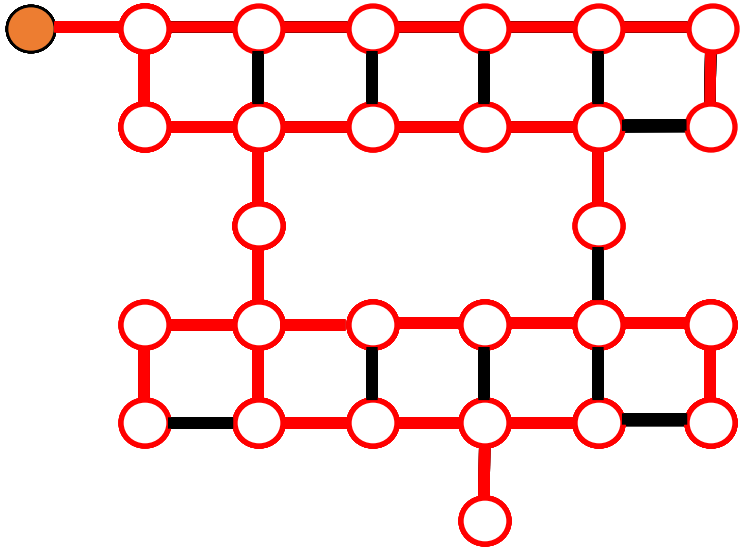
- 알고리즘 구성

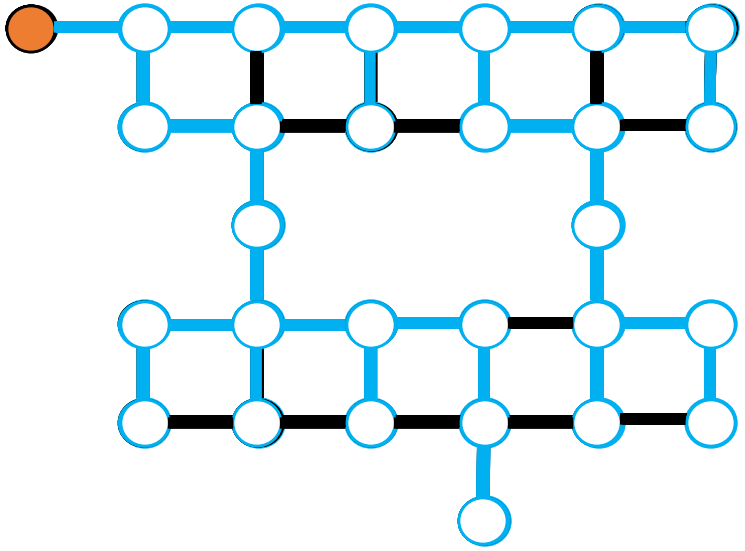
  1. 트리 생성 단계

  2. 초기해 도출 및 트리구조 선정 단계

본선 & 결선 알고리즘

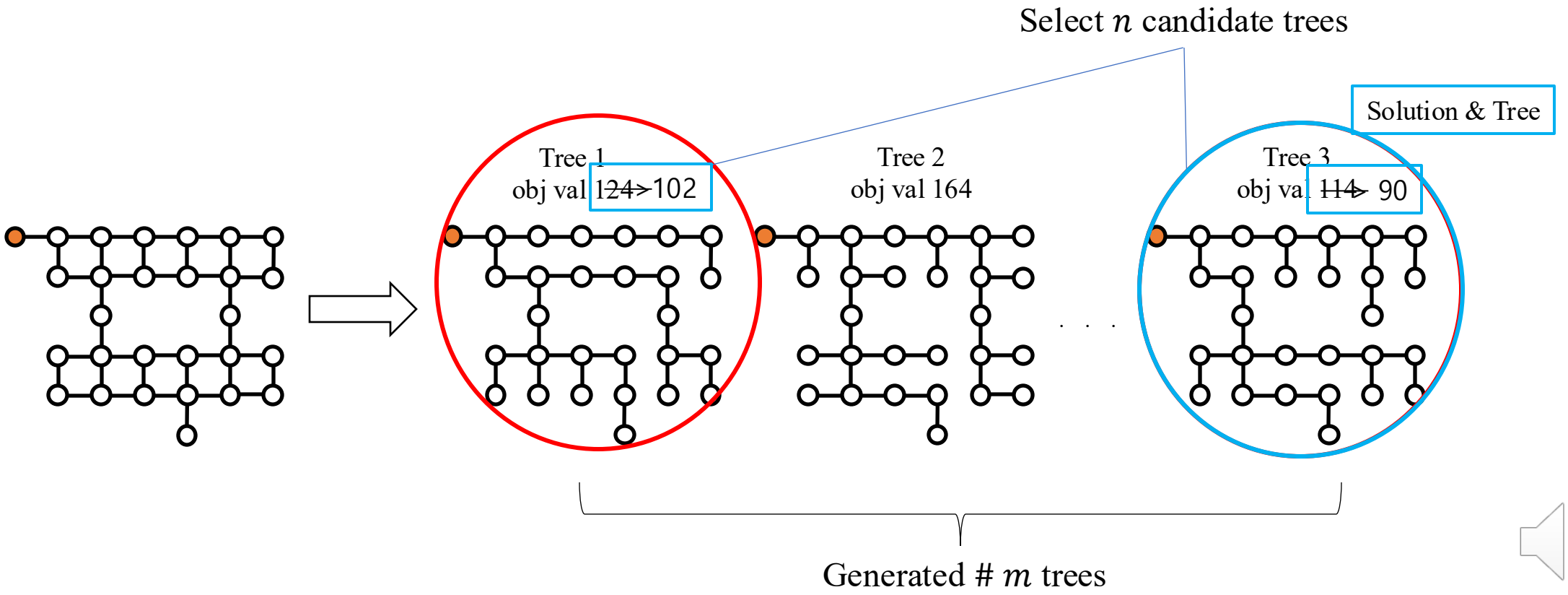
- **트리 생성 단계**
  - 출입구에서 가장 먼 정점부터 차례대로 최단경로를 하나씩 선택
  - 최단경로에 포함된 정점들은 자동으로 **부분경로를 최단경로**로 가짐
    - 예: 1-2-4-7-10 선택 시, 중간 정점들은 자동으로 1-2, 1-2-4, 1-2-4-7와 같은 부분 경로를 최단경로로 갖게 됨
  - 모든 정점이 최소 한 번 포함될 때까지 반복하여 트리 생성
  - 그래프에서 선택한 최단경로에 따라 트리 구조가 달라짐 -> 하나의 그래프에서 많은 트리 구조 생성 가능





본선 & 결선 알고리즘

- **초기해 도출 및 트리 선정 단계**
  - 각 트리에 동일한 규칙 기반 휴리스틱 적용 후 가장 목적 함수 값이 좋은 n개의 트리를 후보로 선정
  - n개의 트리에 랜덤성을 부여한 휴리스틱을 반복 적용해 **가장 낮은 비용을 도출한 해와 해당 트리 구조를 채택**
  - 휴리스틱 알고리즘에 대한 자세한 정보는 Appendix C 확인



본선 & 결선 알고리즘

1. 초기해 생성 및 트리 선정 알고리즘

- 초기해 생성
- 트리구조 선정

2. 결정변수 변환

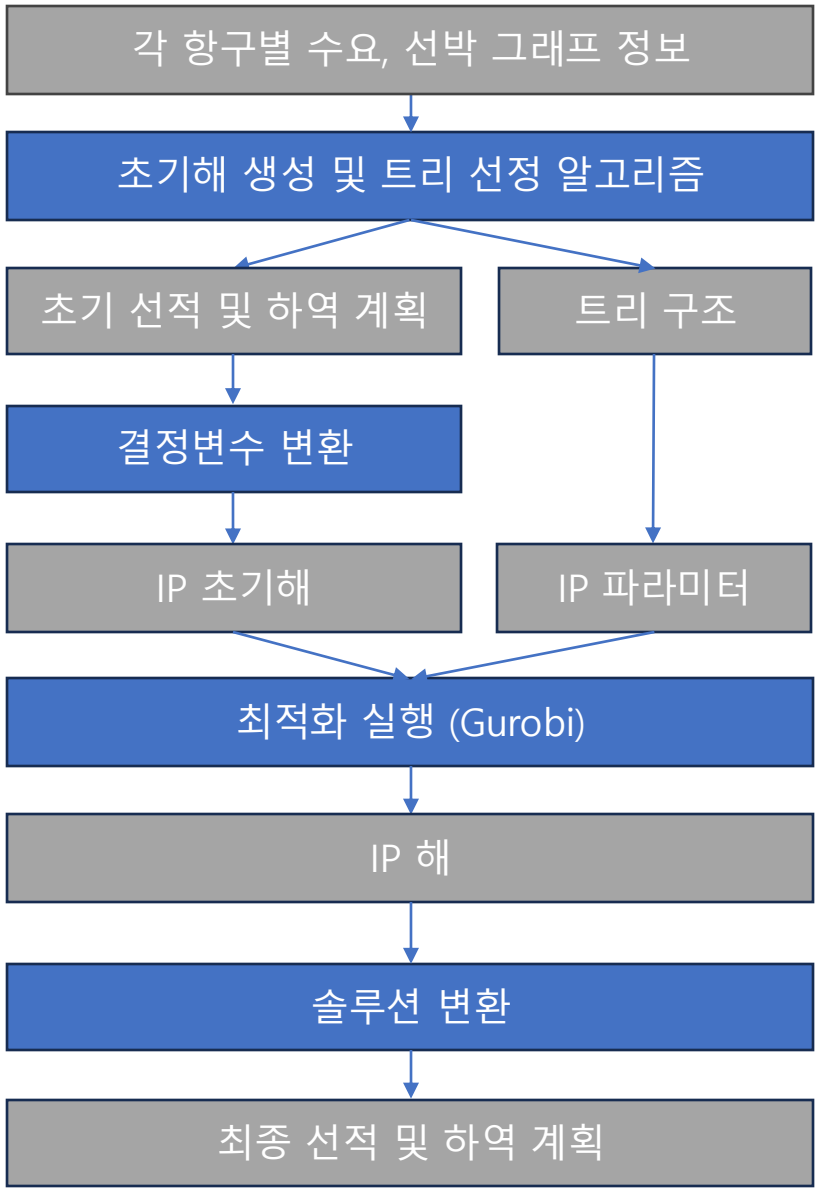
- IP 초기해

3. 최적화 실행 (Gurobi)

- Solver에서 제한 시간 동안 해 개선

4. 솔루션 변환

- IP 해를 대회 제출 형식으로 변환



❖ 모든 코드는 Python으로 작성되었으며, 대회 환경에서 제공하는 Python 라이브러리 외에 추가로 사용한 라이브러리는 없음

본선 & 결선 알고리즘

Trade-off between Optimality (Graph) vs Time (Tree)

- 예선 문제 실험 결과
  - 제한 시간 = 60초
  - Numpy random seed = 300
  - Gurobi Seed = 300
  - ImproveStartTime (Gurobi Parameter) = (Solver 제한 시간) / 2

예선 문제 결과	Graph	Tree	Best Score
Prob 1	231 (20)	233 (3)	231
Prob 2	906 (60)	826 (60)	822
Prob 3	808 (16)	1010 (5)	808
Prob 4	1643 (60)	1403 (60)	1383
Prob 5 (Hidden)	2415 (60)	1711 (60)	1711

❖ 이 외에도 예제 문제들과 본선, 결선 문제에서도 유사한 경향성

알고리즘 장점 및 개선점

장점

1. **모든 포트의 수요를 동시에 고려하는 수리적 최적화 모형**을 기반
2. **트리 형태로 문제를 변환**하고 휴리스틱으로 **얻은 초기해를 Gurobi Solver에 대입**하여 **제한된 시간내에 좋은 해**를 빠르게 탐색
3. 현실 문제에 도입한다면 발생할 **추가적인 제약조건 (eg. 차량별 이종적인 주차 가능 제약, 선박의 무게 균형 제약 등)을 수리적 최적화 모형에 추가**하여 반영할 수 있음

개선점

- 트리 선정 로직 개선
- 문제 정보와 제한시간에 따라 Graph와 Tree 모델 중 하나를 선택하는 로직 추가
- 솔루션 변환 과정에서 위치변경 가능성을 확인하는 로직 추가

대회 후기

- 최적화 관련 수업이나 교과서, 혹은 학술 연구에서 찾아보기 힘든 새로운 형태의 **실제 산업 현장 문제를 해결해보는 좋은 경험**
- 같은 목표를 가진 **다른 참가자들과의 경쟁**을 통해 짧은 시간동안 알고리즘 및 최적화 분야의 실력 비약적 성장
- “2026 OGC”

Thank You!

Appendix (A) – 수리적 최적화 모형 (그래프)

수리적 최적화 모형

Indices

$\boldsymbol{P} = [0,1,\dots,P-1]$: Port Index | $\boldsymbol{N} = [1,2,\dots,N-1]$: Node Index | $\boldsymbol{E}$: Edge Index

D.V.s

$x_{pnd} \in \{0,1\}$: 1 if node n is occupied by a vehicle with destination d at the end of port p loading phase
$u_{pnd} \in \{0,1\}$: 1 if node n is occupied by a vehicle with destination d at the end of port p unloading phase
$y_{pnd} \in \{0,1\}$: 1 if vehicle with destination d is loaded to node n at port p
$v_{pnd} \in \{0,1\}$: 1 if vehicle at node n with destination d is unloaded at port p
$fl_{pn_1n_2} \in \mathbb{R}^+$: Amount of vehicle flow from node n_1 to node n_2 in loading phase at port p
$fu_{pn_1n_2} \in \mathbb{R}^+$: Amount of vehicle flow from node n_1 to node n_2 in unloading phase at port p

Parameters

F: Fixed cost for loading / unloading / rehandling
LB: Lower bound of problem (given)
C_{pd}: Cumulative total demand to destination d at port p
N_n: Set of neighbor nodes of node n

Obj Func.

$$\min F * \left[\sum_{p\in\boldsymbol{P}}\sum_{n\in\boldsymbol{N}}\sum_{d\in\boldsymbol{P}}[y_{pnd}+v_{pnd}]\right] + \sum_{p\in\boldsymbol{P}}\sum_{(n_1,n_2)\in\boldsymbol{E}}[fl_{pn_1n_2}+fu_{pn_1n_2}] - LB$$

Appendix (A) – 수리적 최적화 모형 (그래프)

수리적 최적화 모형

Constraints

1) Node occupancy constraints

$$\sum_{d\in \boldsymbol{P}} x_{pnd} \le 1\,, \forall p \in \boldsymbol{P}, n \in \boldsymbol{N} \qquad (1-1)$$

$$\sum_{d\in \boldsymbol{P}} u_{pnd} \le 1, \forall p \in \boldsymbol{P}, n \in \boldsymbol{N} \qquad (1-2)$$

2) Demand satisfaction constraints

$$\sum_{n\in \boldsymbol{N}} x_{pnd} = C_{pd}, \forall p \in \boldsymbol{P}, d \in \boldsymbol{P} \qquad (2-1)$$

$$\sum_{n\in \boldsymbol{N}} u_{pnd} = 0, \forall p \ge d \qquad (2-2)$$

3) Initial & Terminal condition constraints

$$u_{0nd} = 0, \forall n \in \boldsymbol{N}, d \in \boldsymbol{P} \qquad (3-1)$$

$$x_{(P-1)nd} = 0\,, \forall n \in \boldsymbol{N}, d \in \boldsymbol{P} \qquad (3-2)$$

4) Vehicle conservation constraints (Linking Constraints)

$$u_{pnd} + y_{pnd} = x_{pnd}, \forall p \in \boldsymbol{P}\backslash(P-1), n \in \boldsymbol{N}, d \in \boldsymbol{P} \qquad (4-1)$$

$$x_{(p-1)nd} - v_{pnd} = u_{pnd}, \forall p \in \boldsymbol{P}\backslash 0, n \in \boldsymbol{N}, d \in \boldsymbol{P} \qquad (4-2)$$

5) Flow conservation constraints

$$\sum_{n_1\in N_n} fl_{pn_1n} - \sum_{n_2\in N_n} fl_{pnn_2} = \sum_{d\in \boldsymbol{P}} y_{pnd}\,, \forall p \in P, n \in N \qquad (5-1)$$

$$\sum_{n_1\in N_n} fu_{pn_1n} - \sum_{n_2\in N_n} fu_{pnn_2} = \sum_{d\in \boldsymbol{P}} v_{pnd}\,, \forall p \in P, n \in N \qquad (5-2)$$

6) Node reachability constraints

$$\sum_{n_1\in N_n} fl_{pn_1n} \le |N| * \left(1 - \sum_{d\in \boldsymbol{P}} u_{pnd}\right), \forall p \in P, n \in N \qquad (6-1)$$

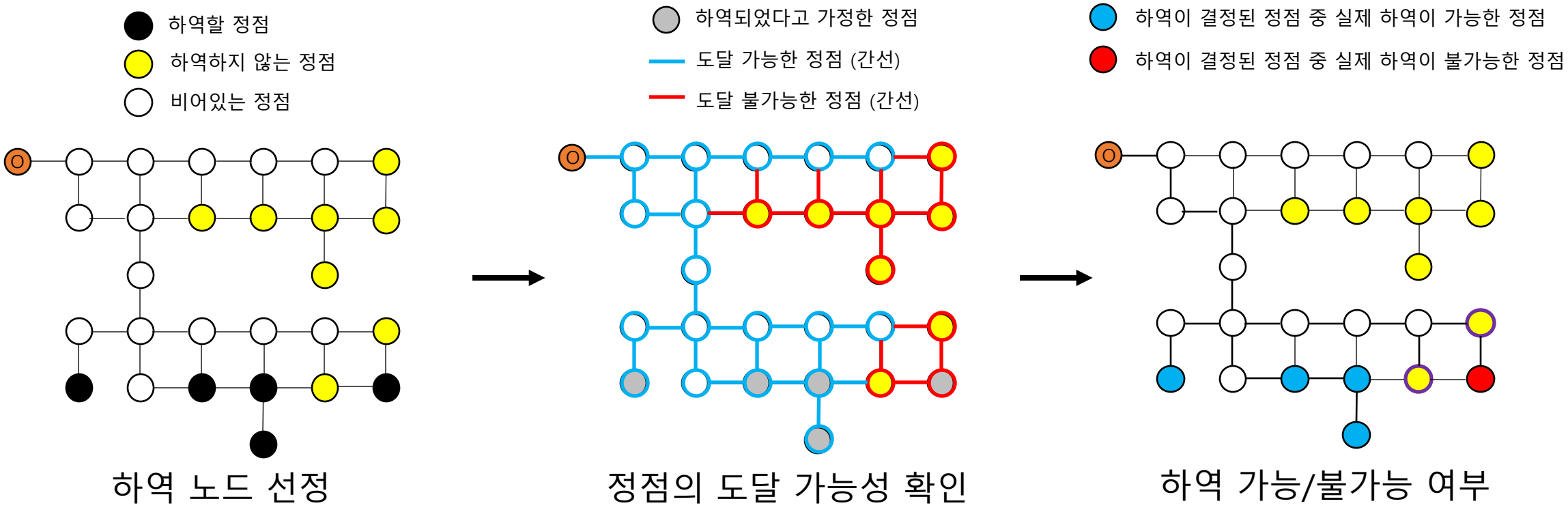
$$\sum_{n_1\in N_n} fu_{pn_1n} \le |N| * \left(1 - \sum_{d\in \boldsymbol{P}} u_{pnd}\right), \forall p \in P, n \in N \qquad (6-2)$$

$$\sum_{d\in P} y_{pnd} \le 1 - \sum_{d\in \boldsymbol{P}} u_{pnd}\,, \forall p \in P, n \in N \qquad (6-3)$$

$$\sum_{d\in P} v_{pnd} \le 1 - \sum_{d\in \boldsymbol{P}} u_{pnd}\,, \forall p \in P, n \in N \qquad (6-4)$$

Appendix (B) – 하역 가능 제약조건 추가 설명

- 하역 가능성 제약조건 설명 (Graph 모형 제약조건 5-2, 6-2, Tree 모형 제약조건 7-2)
  - 하역할 차량들이 모두 하역이 되었다고 가정하였을 때, **출입구에서 해당 정점에 도달할 수 있다면 실제로 하역 가능**
  - 해당 제약 조건을 통해 **블로킹을 해소하기 위한 임시 하역 유도**



Appendix (C) – 해 탐색 및 개선 단계

- Myopic Heuristic (규칙 기반)
  - **Psuedo Code**:

for $p = 0$ to P do

if $p \neq 0$ then do **// 하역 과정**

- 목적지가 p인 수요들과, 이들을 막고 있는 수요들을 모두 하역한다.

- 이때 내린 수요들 중 목적지가 p가 아닌 수요들은 즉시 다음 단계(적재 과정)에서 재적재 대상으로 분류한다.

end if

//적재 과정

- 출발지가 p인 수요와 재적재해야 하는 수요의 합을 k라 하자.

- 적재 대상 k개 수요를 목적지의 역순(먼 미래 항차 우선)으로 정렬한다.

for $i = 1$ to k do // **blocking threshold = 0**

- 현재 트리에서 이미 할당된 노드를 제외한 가용 서브트리를 구한다.

- 서브트리의 리프 노드에서 i번째 수요보다 하차가 빠른 차량을 막지않는 노드들의 집합을 candidate set이라 정의

while $|candidate\ set| = 0$ do

- blocking threshold += 1

- blocking threshold만큼 일찍 내리는 차량들을 막는 노드들의 집합을 candidate set으로 새로 정의

end while

if i번째 수요와 동일한 목적지를 가진 차량이 이미 트리에 존재한다면

- 동일 목적지 차량들과의 거리가 가장 가까운 candidate set의 원소 노드에 i번째 수요를 적재한다.

else

- **candidate set 중 수요 번호의 비율에 해당하는 정점을 선택**하여 i번째 수요를 적재한다.

end if

end for

end for

Appendix (C) – 해 탐색 및 개선 단계

- Myopic Heuristic (랜덤)
  - **Psuedo Code**:

for $p = 0$ to P do

if $p \neq 0$ then do **// 하역 과정**

- 목적지가 p인 수요들과, 이들을 막고 있는 수요들을 모두 하역한다.
- 이때 내린 수요들 중 목적지가 p가 아닌 수요들은 즉시 다음 단계(적재 과정)에서 재적재 대상으로 분류한다.

end if

//적재 과정

- 출발지가 p인 수요와 재적재해야 하는 수요의 합을 k라 하자.
- 적재 대상 k개 수요를 목적지의 역순(먼 미래 항차 우선)으로 정렬한다.

for $i = 1$ to k do // **blocking threshold = 0**

- 현재 트리에서 이미 할당된 노드를 제외한 가용 서브트리를 구한다.
- 서브트리의 리프 노드에서 i번째 수요보다 하차가 빠른 차량을 막지않는 노드들의 집합을 candidate set이라 정의

while $|candidate\ set| = 0$ do

- blocking threshold += 1
- blocking threshold만큼 일찍 내리는 차량들을 막는 노드들의 집합을 candidate set으로 새로 정의

end while

if i번째 수요와 동일한 목적지를 가진 차량이 이미 트리에 존재한다면

- 동일 목적지 차량들과의 거리가 가장 가까운 candidate set의 원소 노드에 i번째 수요를 적재한다.

else

- **candidate set 중 하나를 무작위로 선택하여** i번째 수요를 적재한다.

end if

end for

end for