OPTIMIZATION GRAND CHALLENGE 2025

25.05.25 ~ 25.06.16
25.06.23 ~ 25.07.28
25.08.04 ~ 25.08.25
25.09.01 ~ 25.09.08
25.09.19

TEAM : 081F38

MEMBER : 노 영 주

AFFILIATION : SysOpt@SNU
We optimize

LG CNS · 사단법인 대한산업공학회 · HYUNDAI GLOVIS · GUROBI OPTIMIZATION · FICO · Omelet · 서울대학교 공과대학 산업시스템혁신연구소

Table of Contents

Idea

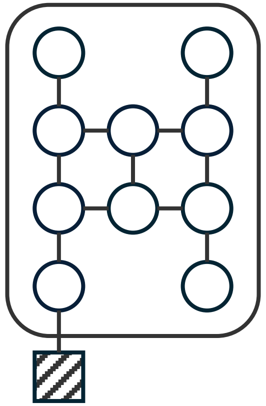


그림: 선박 그래프

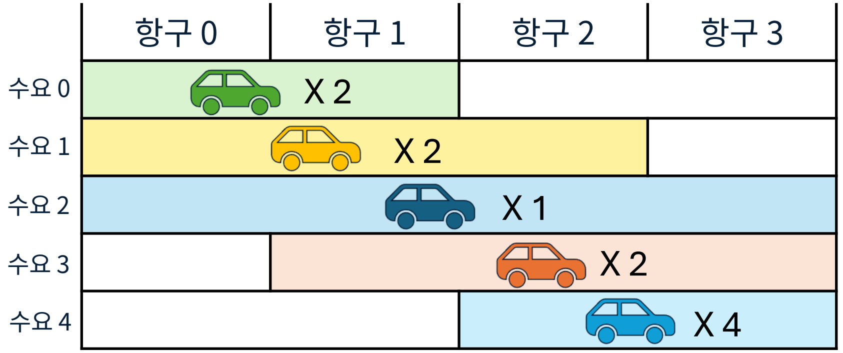
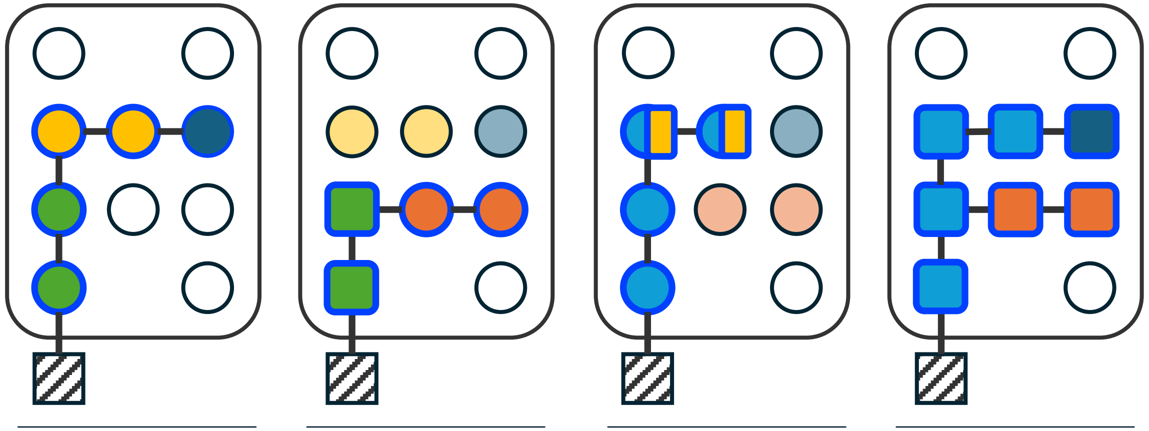


그림: 항구-수요 정보



항구 0 항구 1 항구 2 항구 3

- 각 항구에서 선적할 수요의 위치를 결정
- 각 항구에서 선적/하역 수요가 위치한 모든 노드는 게이트 노드까지 연결
- 빈 노드 혹은 경유 수요가 위치한 노드 포함하여 연결 가능
- 총 비용(고정비 + 이동거리에 따른 변동비) 최소화

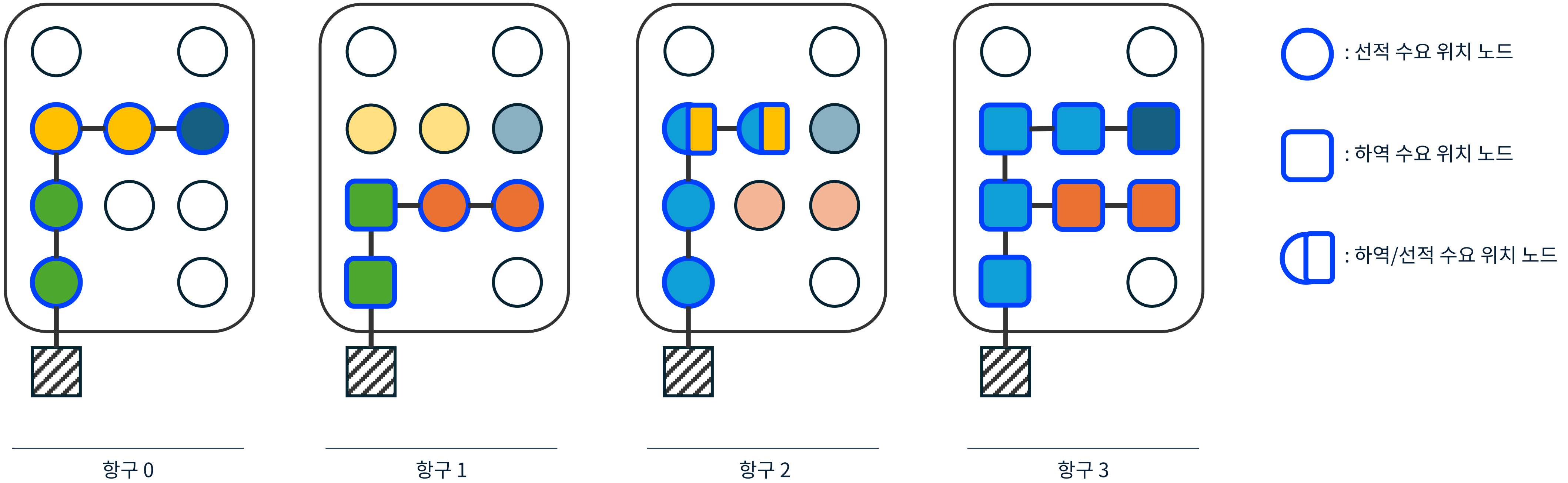
Definition (Steiner Tree)

주어진 그래프 G = (V, E) 와 terminal set $T \subseteq V$ 에 대해서 T 에 속하는 모든 terminal node 를 연결하는 edge-minimal, acyclic subgraph

Steiner tree 의 terminal node depth 총합을 최소화 !

Idea

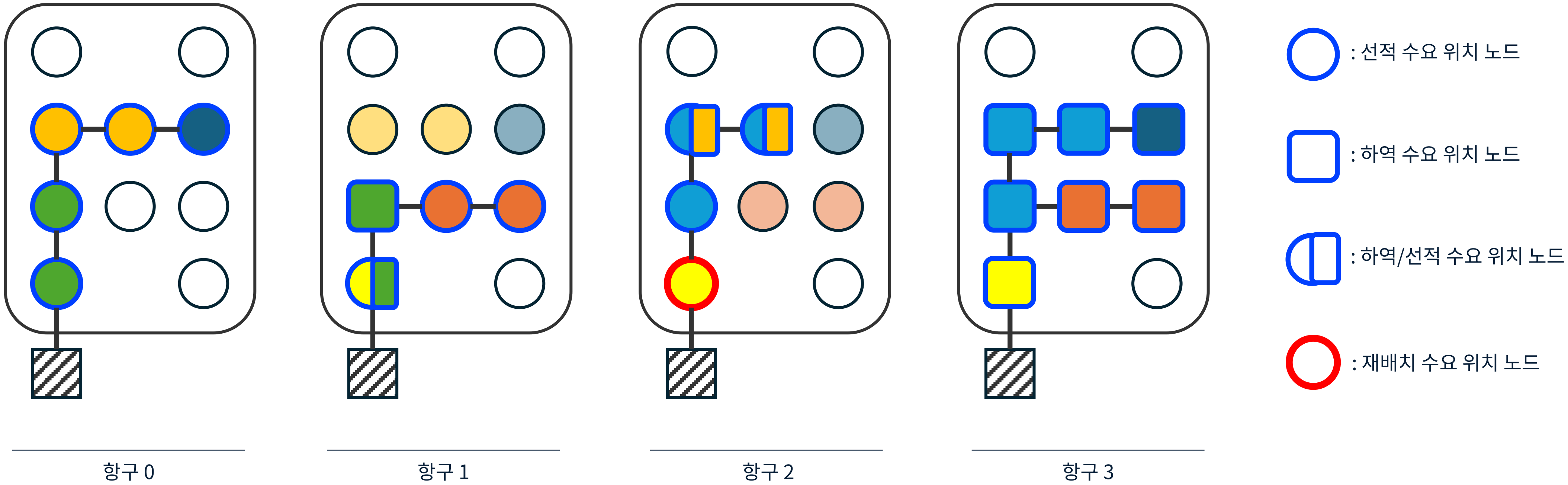
Steiner tree



선적/하역 수요가 위치한 노드에서 flow 를 1만큼 생성, 나머지 노드에서는 flow balance → flow 총합 = node depth 총합

Idea

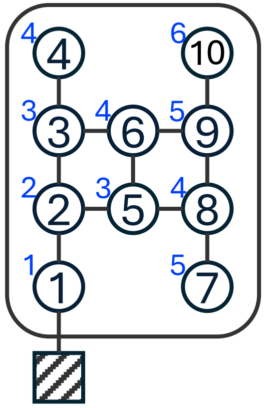
Steiner tree



해당 항구를 경유하는 수요가 위치한 노드에 in-flow 가 존재하면 재배치 대상

Idea

Node Filtering



1	1		
2	2		
3	3	5	
4	4	6	8
5	7	9	
6	10		

그림: 선박 그래프, 노드-거리 정보
검정: 노드 인덱스, 파랑: 게이트 노드로부터 최단거리)

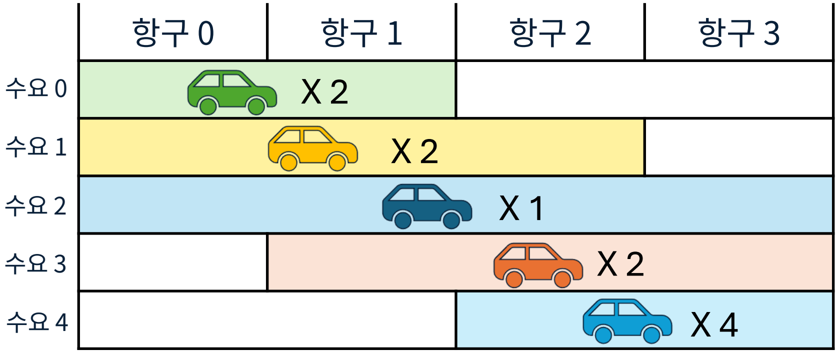
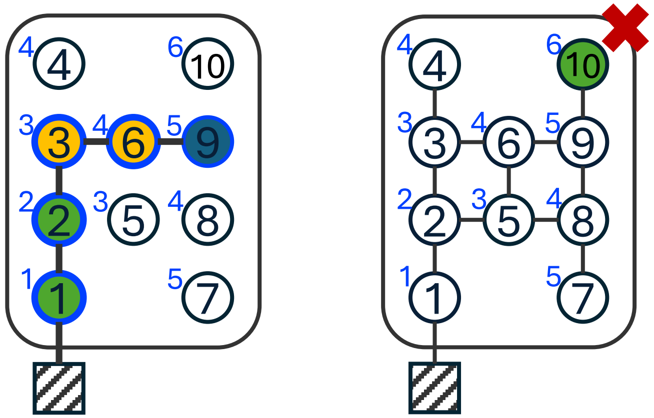


그림: 항구-수요 정보



항구 0

주어진 정보

- 게이트 노드로부터 각 노드까지의 최단 경로
- 각 수요별 항해 거리 (경유 항구 수)
- 항해 기간이 겹치는 타 수요의 항해 거리

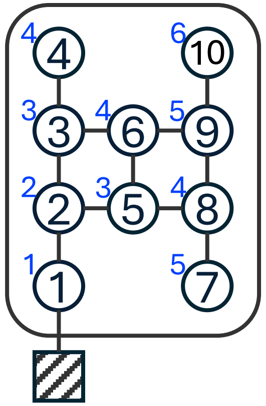
항해 거리가 비교적 짧은 수요는 항해 거리가 긴 수요에 비해, 게이트 노드로부터 먼 노드에 위치할 가능성이 적음

- 각 수요별로 경유 항구마다(하역 항구 제외) 선적 되어있는 수요들에 대해 다음을 계산:

 전체 노드 개수 * (항해 거리가 같거나 짧은 수요 개수)/총 수요 개수

Algorithm

Node Filtering



1	1		
2	2		
3	3	5	
4	4	6	8
5	7	9	
6	10		

그림: 선박 그래프, 노드-거리 정보
검정: 노드 인덱스, 파랑: 게이트 노드로부터 최단거리)

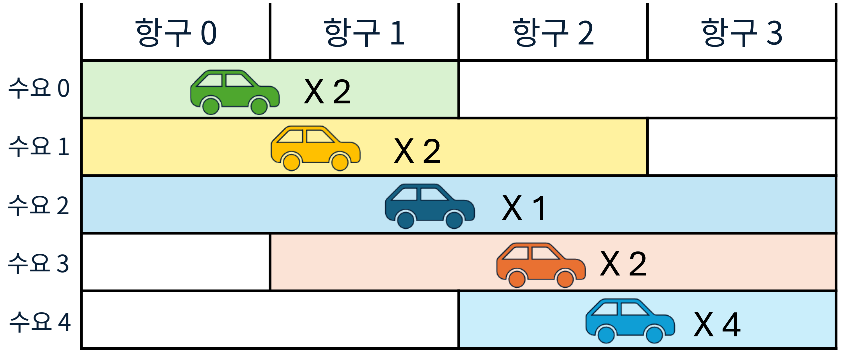
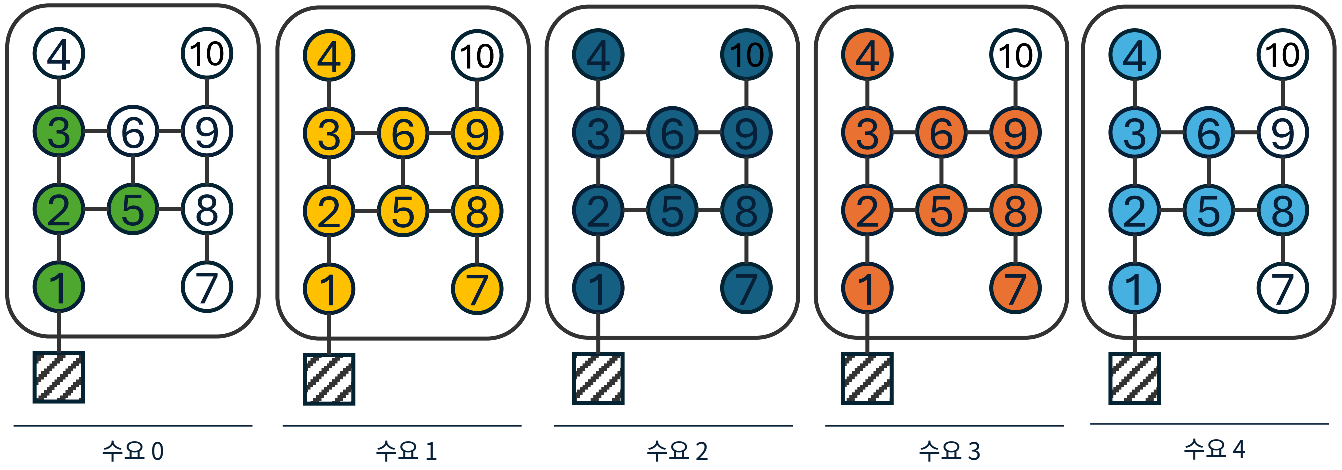


그림: 항구-수요 정보

	항구 0	항구 1	항구 2	항구 3		
수요 0	2/5				2/5	10*2/5 = 4
수요 1	4/5	4/5			4/5	10*4/5 = 8
수요 2	5/5	5/5	7/7		7/7	10*1 = 10
수요 3		4/5	< 6/7		6/7	⌊10*6/7⌋ = 9
수요 4			4/7		4/7	⌊10*4/7⌋ = 6

그림: 노드 필터링 데이터



Algorithm

Mixed Integer Programming Model

- **결정변수:**
  - $x_{q,r}^{p,i}$: 항구 q 에서 적재한 수요 r 이 항구 p 의 노드 i 에 위치해 있으면 1, 아니면 0
  - $y^{p,i}$: 항구 p 의 노드 i 에 위치한 수요가 있으면 1, 아니면 0
  - $z^{p,i}$: 항구 p 의 노드 i 에 in-flow 가 있으면 1, 아니면 0
  - $w^{p,i}$: 항구 p 의 노드 i 에 위치한 수요의 재배치가 필요하면 1, 아니면 0
  - $l_{i,j}^{p}$: 항구 p 에서 edge (i, j) 에 흐르는 흐름의 양
- **목적함수:** Flow 총합 + 총 고정비용 최소화
- **제약식:**
  - 수요 만족
  - 노드에는 하나의 수요만 위치
  - $y^{p,i}, z^{p,i}, w^{p,i}$ 결정 제약
  - Flow 제약 (선적/하역/재배치 flow 생성 + flow balance)

모형의 크기

- 결정 변수: O(노드 개수 × 항구 개수 × 수요 o-d pair 개수)
- 제약식: O(노드 개수 × 항구 개수 × 수요 o-d pair 개수)

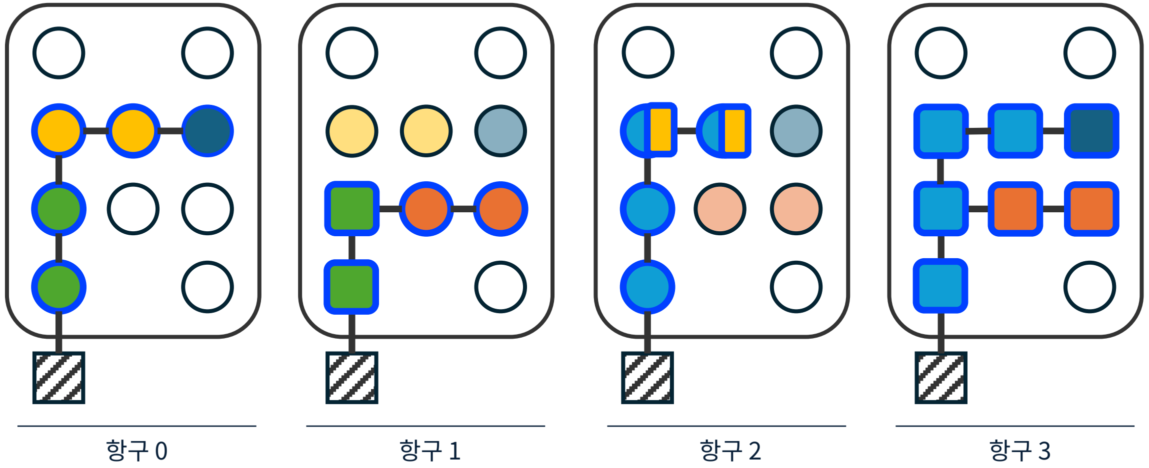
Full formulation 은 Appendix 참조

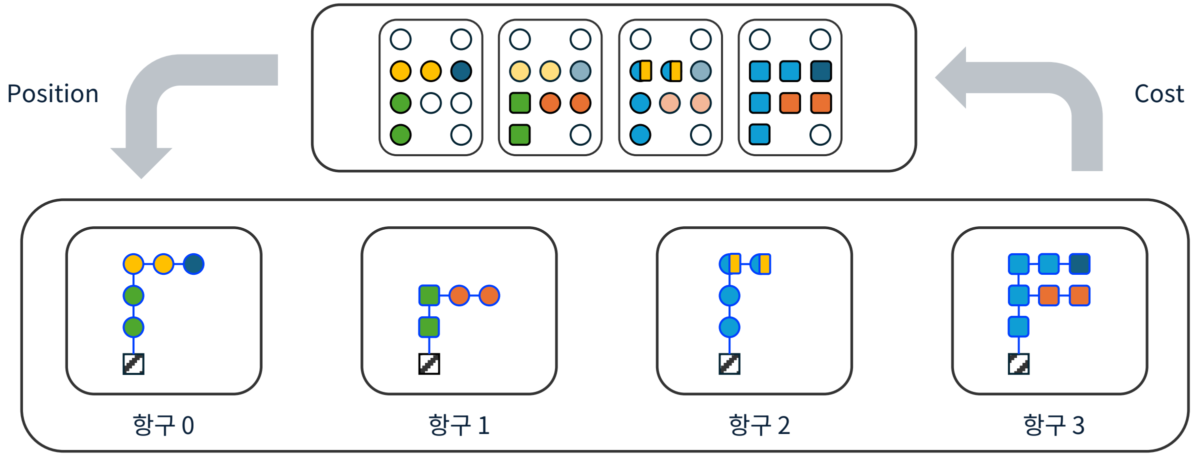
Algorithm

Implementation

- **최적화 소프트웨어:** Gurobi 12.0.1
- **프로그래밍 언어:** Python (데이터 입/출력), C++ (데이터 전처리 및 알고리즘)
- **주요 Gurobi solver parameter :**
    - Branching priority : $y \gg z > w > x$ (선적 항구) > x (그 외)

Distinctive Features & Future Works

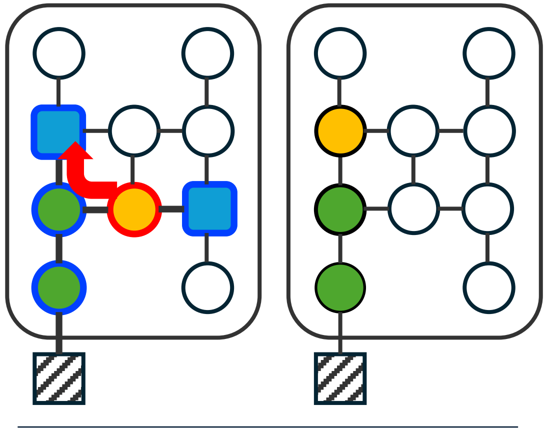




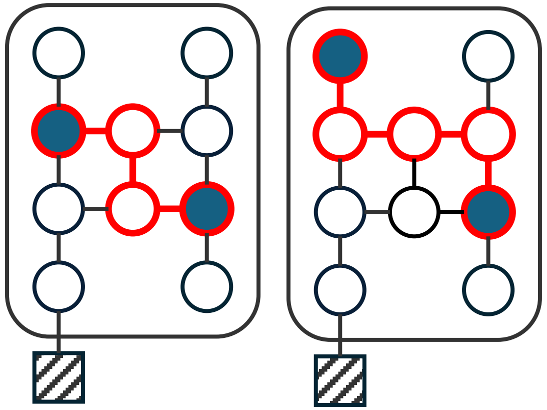
- **Distinctive Features**
  - 선적/하역 위치와 경로를 최적화 모형에서 함께 고려
  - 확장 가능성
- **Benders Decomposition approach**
  - Master problem: 각 항구마다 수요 위치 결정
    - Gate 노드로부터 최단 경로 길이로 bound 계산
  - Subproblem: 선적/하역/재배치 경로 결정
    - 항구 별로 병렬 처리 가능 !

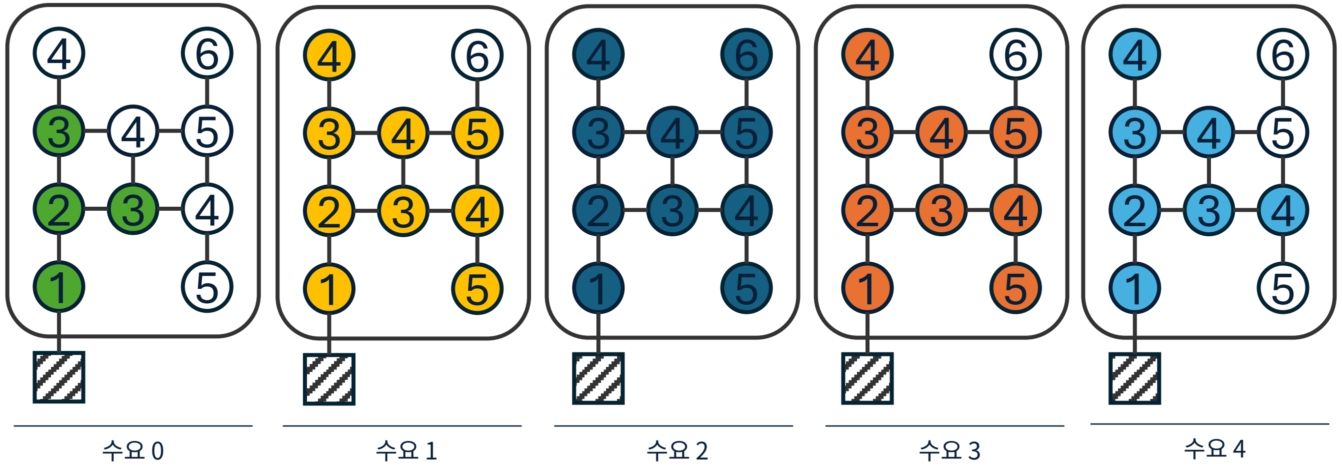
Distinctive Features & Future Works

1.



2.

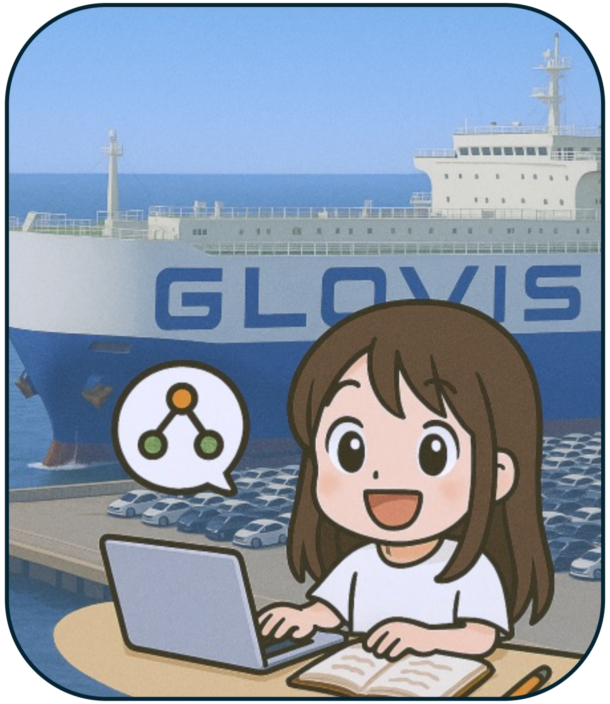




- **다양한 rehandling 반영**
  - 솔루션을 구성할 때 선박 내 재배치 고려
  - 선박 내 재배치를 최적화 모형 내에서 고려

- **Data preprocessing 개선**
  - 수요, 선박 노드 배치 특성을 고려한 수요별 노드 필터링 전략 고안

Concluding Remark



- 현실의 문제에 대해서 다양한 알고리즘을 설계 및 구현

- 예선/본선에서도 제출 마감 후 random instance 점수를 추가 반영하여 최종 순위 산출

- Time limit 범위를 늘려서 다양한 알고리즘 개발 독려

Illustration generated with OpenAI ChatGPT/DALL·E (2025)

THANK YOU

TEAM : 081F38

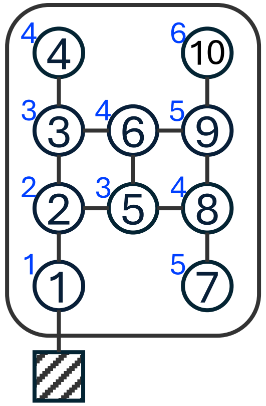
MEMBER : 노 영 주

youngjooroh@snu.ac.kr

SysOpt@SNU
We optimize

Appendix

Node Filtering



1	1		
2	2		
3	3	5	
4	4	6	8
5	7	9	
6	10		

그림: 선박 그래프, 노드-거리 정보
검정: 노드 인덱스, 파랑: 게이트 노드로부터 최단거리)

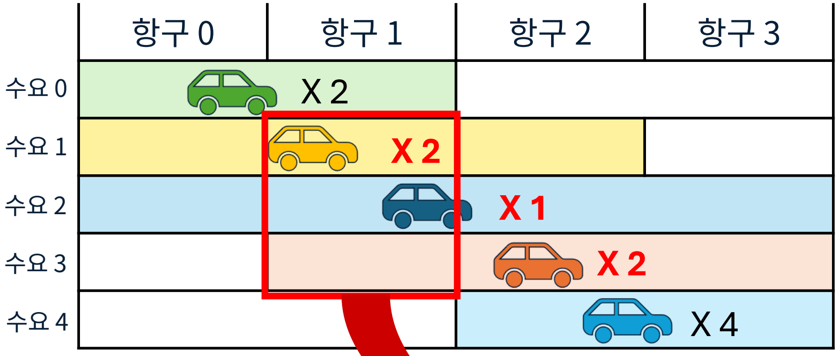


그림: 항구-수요 정보

	항구 0	항구 1	항구 2	항구 3		
수요 0	2/5				2/5	10*2/5 = 4
수요 1	4/5	4/5			4/5	10*4/5 = 8
수요 2	5/5	5/5	7/7		7/7	10*1 = 10
수요 3		4/5	< 6/7		6/7	⌈10*6/7⌉ = 9
수요 4			4/7		4/7	⌈10*4/7⌉ = 6

그림: 노드 필터링 데이터

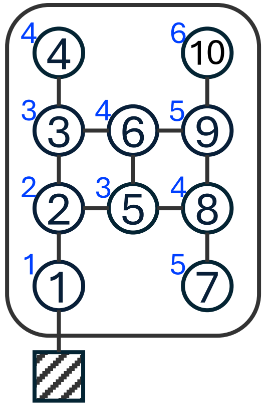
1. **각 수요별로 경유 항구마다(<u>하역 항구 제외</u>) 선적 되어있는 수요들에 대해 다음을 계산:**

   (항해 거리가 "같거나 짧은" 수요 개수)/총 수요 개수

2. 각 수요별로 1.의 값 중 최대 값에 '총 노드 개수' 를 곱한 값을 계산
3. 게이트 노드에서 거리가 2.의 값 번째 가까운 노드의 거리 이하의 노드 까지만 고려

Appendix

Node Filtering



1	1		
2	2		
3	3	5	
4	4	6	8
5	7	9	
6	10		

그림: 선박 그래프, 노드-거리 정보
검정: 노드 인덱스, 파랑: 게이트 노드로부터 최단거리)

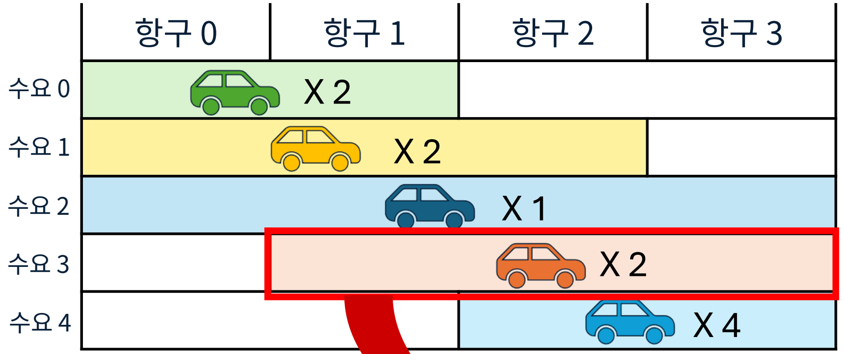


그림: 항구-수요 정보

	항구 0	항구 1	항구 2	항구 3		
수요 0	2/5				2/5	10*2/5 = 4
수요 1	4/5	4/5			4/5	10*4/5 = 8
수요 2	5/5	5/5	7/7		7/7	10*1 = 10
수요 3		4/5 <	6/7		6/7	$\lceil 10*6/7 \rceil = 9$
수요 4			4/7		4/7	$\lceil 10*4/7 \rceil = 6$

그림: 노드 필터링 데이터

1. 각 수요별로 경유 항구마다(하역 항구 제외) 선적 되어있는 수요들에 대해 다음을 계산:

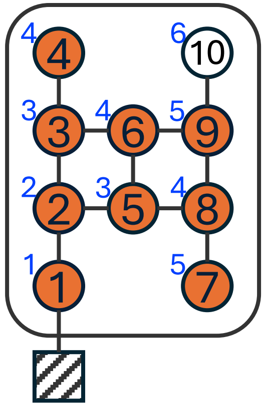
   (항해 거리가 "같거나 짧은" 수요 개수)/총 수요 개수

2. **각 수요별로 1.의 값 중 최대 값에 '총 노드 개수' 를 곱한 값을 계산**

3. 게이트 노드에서 거리가 2.의 값 번째 가까운 노드의 거리 이하의 노드 까지만 고려

Appendix

Node Filtering



1	1		
2	2		
3	3	5	
4	4	6	8
5	7	9	
6	10		

그림: 선박 그래프, 노드-거리 정보

검정: 노드 인덱스, 파랑: 게이트 노드로부터 최단거리)

<table>
<tr><th></th><th>항구 0</th><th>항구 1</th><th>항구 2</th><th>항구 3</th></tr>
<tr><td>수요 0</td><td colspan="2">X 2</td><td></td><td></td></tr>
<tr><td>수요 1</td><td colspan="3">X 2</td><td></td></tr>
<tr><td>수요 2</td><td colspan="4">X 1</td></tr>
<tr><td>수요 3</td><td></td><td colspan="3">X 2</td></tr>
<tr><td>수요 4</td><td></td><td></td><td colspan="2">X 4</td></tr>
</table>

그림: 항구-수요 정보

	항구 0	항구 1	항구 2	항구 3		
수요 0	2/5				2/5	10*2/5 = 4
수요 1	4/5	4/5			4/5	10*4/5 = 8
수요 2	5/5	5/5	7/7		7/7	10*1 = 10
수요 3		4/5 <	6/7		6/7	⌈10*6/7⌉ = 9
수요 4			4/7		4/7	⌈10*4/7⌉ = 6

그림: 노드 필터링 데이터

1. 각 수요별로 경유 항구마다(<u>하역 항구 제외</u>) 선적 되어있는 수요들에 대해 다음을 계산:

   (항해 거리가 “같거나 짧은” 수요 개수)/총 수요 개수

2. 각 수요별로 1.의 값 중 최대 값에 ‘총 노드 개수’ 를 곱한 값을 계산

3. **게이트 노드에서 거리가 2.의 값 번째 가까운 노드의 거리 이하의 노드 까지만 고려**

Appendix

Mixed Integer Programming Model

- **Set & Parameters :**
  - $N := \{0, 1, \cdots, n\}$: 선박 노드 집합 (0 : 게이트 노드, n : 선박에서 수요가 위치할 수 있는 노드 개수)
  - $A := \{(i, j) : i \in N \setminus \{0\}, j \in N \text{ s.t.} (i, j) \in E \text{ or } (j, i) \in E\}$: 선박 아크 집합 (E: 선박의 edge 집합)
  - $P := \{0, \cdots, p-1\}$: 항구 집합 (p: 선박이 경유하는 총 항구 개수)
  - $D := \{0, 1, \cdots, d-1\}$: 수요 집합 ($d$: 처리해야 하는 demand type 수; demand type : 수요의 출발지와 도착지 (s_d, t_d) pair 수)
  - D^q : 항구 q 에서 출발하는 수요 집합
  - D^{qr} : 항구 q 에서 출발하는 수요의 개수
  - D_m^{qr} : 항구 q 에서 출발하는 수요가 경유하는 항구 개수
  - D_s^p : 항구 p 에서 선적하는 수요의 개수 총합
  - H_t^p : 항구 p 에서 하역하는 수요의 개수 총합

Appendix

Mixed Integer Programming Model

- **Decision variables :**
  - $x_{q,r}^{p,i}$: 항구 q 에서 적재한 수요 r 이 항구 p 의 노드 i 에 위치해 있으면 1, 아니면 0
  - $y^{p,i}$: 항구 p 의 노드 i 에 위치한 수요가 있으면 1, 아니면 0
  - $z^{p,i}$: 항구 p 의 노드 i 에 in-flow 가 있으면 1, 아니면 0
  - $w^{p,i}$: 항구 p 의 노드 i 에 위치한 수요의 재배치가 필요하면 1, 아니면 0
  - $l_{i,j}^{p}$: 항구 p 에서 edge (i, j) 에 흐르는 흐름의 양
- **Objective function :** Flow 총합 + 총 고정비용 최소화

Appendix

Mixed Integer Programming Model

$$\min_{x,y,z,w,l} \quad \sum_{p\in P}\sum_{(i,j)\in A} l_{ij}^{p} + 2F\sum_{p\in P}\sum_{i\in N} w^{p,i} \tag{1}$$

(1) 총 비용(Flow +고정비) 최소화

$$\text{s.t.} \quad \sum_{i\in N\setminus\{0\}} x_{q,r}^{p,i} = D^{qr}, \quad \forall p \in P, q \in P, r \in D^{q}, s_{d}^{r} \le p \le t_{d}^{r}, \tag{2}$$

(2) 수요 만족 제약

$$\sum_{q\in P:q\le p}\ \sum_{r\in D^{q}:p<t_{d}^{r}} x_{q,r}^{p,i} \le 1, \quad \forall p \in P, i \in N\setminus\{0\}, \tag{3}$$

(3) 노드에는 최대 하나의 수요(하역하는 수요는 제외) 위치 가능

$$\sum_{p\in P:s_{d}^{r}\le p\le t_{d}^{r}} x_{q,r}^{p,i} = D_{m}^{qr} x_{q,r}^{q,i}, \quad \forall i \in N\setminus\{0\}, q \in P, r \in D^{q}, \tag{4}$$

(4) 위치 고정 제약

$$\sum_{q\in P:q\le p}\ \sum_{r\in D^{q}:p<t_{d}^{r}} x_{q,r}^{p,i} \le y^{p,i}, \quad p \in P, i \in N\setminus\{0\}, \tag{5}$$

$$\sum_{j\in N:(j,i)\in A} l_{i,j}^{p} \le (N-1)z^{p,i}, \quad \forall p \in P, i \in N\setminus\{0\}, \tag{6}$$

$$w^{p,i} \ge y^{p,i} + z^{p,i} - 1, \quad \forall p \in P, i \in N\setminus\{0\} \tag{7}$$

(5)–(7) *y, z, w* 값 설정 제약

$$\sum_{j\in N:(i,j)\in A} l_{i,j}^{p} - \sum_{j\in N\setminus\{0\}:(j,i)\in A} l_{j,i}^{p} \ge \sum_{r\in D^{p}} x_{p,r}^{p,i} + \sum_{q\in P:q<p}\ \sum_{r\in D^{q}:p=s_{t}^{r}} x_{q,r}^{p,i} + 2w^{p,i}, \quad \forall p \in P, i \in N\setminus\{0\}, \tag{8}$$

(8) Flow 제약

$$\sum_{i\in N:(i,0)\in A} l_{i,0}^{p} \ge D_{s}^{p} + H_{t}^{p} + 2\sum_{i\in N\setminus\{0\}} w^{p,i}, \quad \forall p \in P \tag{9}$$

(9) Tree 제약 (제출한 알고리즘에서는 해당 제약 relax)

$$l_{i,j}^{p} \le N-1, \quad \forall p \in P, (i,j) \in A. \tag{10}$$

(10) Flow capacity