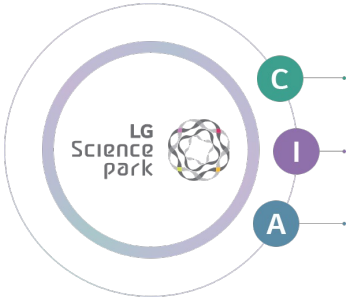
Optimization Grand Challenge 2025

최단 경로 도달 가능성 기반 선적 및 하역 최적화 모형

Roll-on/Roll-off Optimization Model based on the Shortest Path Reachability

NormalDeck

2025. 9.19



목차

1. 문제 재정의
2. 모형화 전략
3. 최적화 모형

} 알고리즘 로직

4. 구현 세부사항 – 알고리즘 구현
5. 알고리즘 장단점 – 알고리즘 특장점 / 개선점 / 후기

RoRo 문제의 제한적 정의

문제 재정의

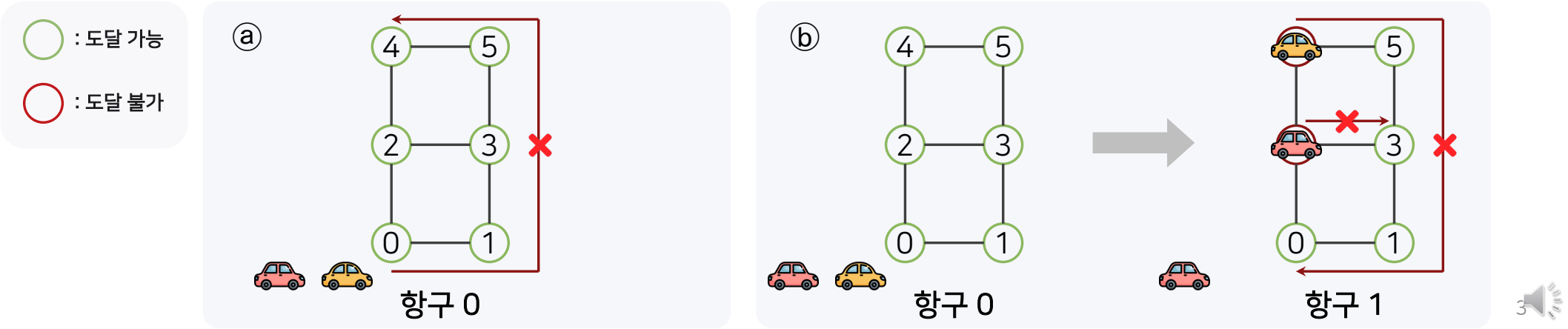
- Roll-on/Roll-off(RoRo) 최적화 문제
  - 입력 정보: ① 수요 ② 선박 내부구조 (deck graph)
  - 제약: ① 수요 제약 ② 선적/하역/재배치 경로 제약
  - 목표: 총 경로의 이동 비용 최소화 (고정비 + 변동비)

- 경로 제약
  ① 주어진 deck graph 위의 경로
  ② 이미 차량이 적재된 노드 포함 불가능

★ **모형에서 추가적으로 도입한 가정**

ⓐ 모든 경로는 주어진 선박 내부 구조 상의 **최단경로**로 이동

ⓑ 재배치가 필요할 경우 **임시 하역 후, 기존 위치**에 재적재하는 것만 허용 **(위치 변경 X)**



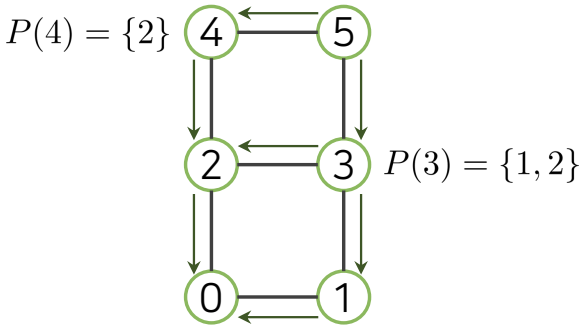
모형화 전략 I : 최단 경로의 최적성 원리

알고리즘 로직

- 각 항구마다 각 노드의 **최단 경로 도달 가능성**을 이진 변수로 표현
  - r_{pn}: 항구 p에서 노드 n까지 최단 경로로 도달 가능하면 1, 아니면 0
  - 해당 항구에서의 하역이 모두 발생한 이후를 기준으로 함
  - 선적/하역 경로 제약 → 최단 경로 도달 가능한 위치에서만 선적/하역 가능

★ **최단 경로의 최적성 원리**

- **v가 최단 경로 도달 가능하고, 호 (v,t)가 존재 $\Leftrightarrow$ t가 최단 경로 도달 가능**
- BFS를 통한 **최단 경로 선행 노드 (predecessor)** 활용
- 오른쪽 그림에서 **초록색 화살표**로 표현됨



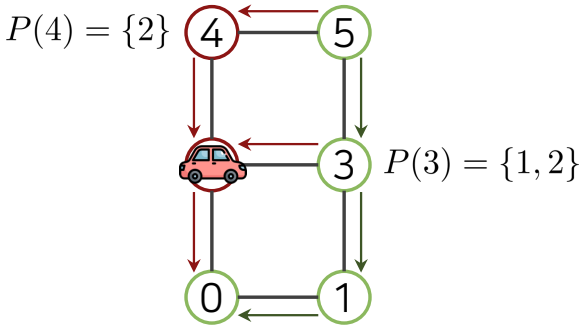
최단 경로 선행 노드

최단 경로 도달 가능 제약

$$r_{pn} \leq \sum_{n' \in P(n)} r_{pn'}, \quad \forall p \in P, n \in N$$

➢ 도달 가능하려면 ($r_{pn} = 1$) 적어도 하나의 최단 경로 선행 노드 가 도달 가능해야 함

- $P(n)$: 노드 n의 최단 경로 선행 노드 집합

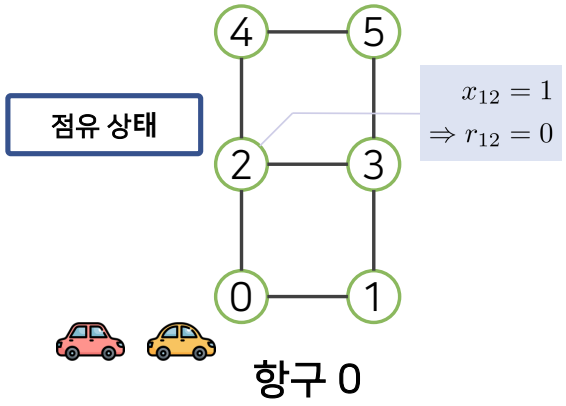


모형화 전략 II : 임시 하역 및 동일 위치 재적재

알고리즘 로직

- 각 항구마다 **점유**되어 있는 노드는 최단 경로 **도달 불가능**하도록 설정
  - x_{dn}: 수요 d 중 하나의 차량이 노드(적재 위치) n에 배정되면 1, 아니면 0
  - 해당 항구에서의 하역이 모두 발생한 이후를 기준으로 함

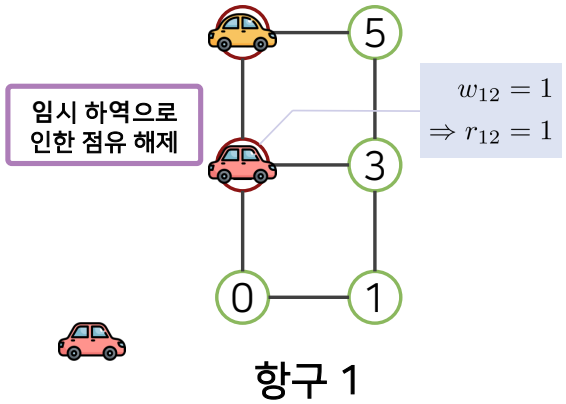
- 충돌 금지 제약과 임시하역
  - 충돌: 점유되어 있는 노드에 도달하는 현상
  - 충돌이 발생하는 노드에서는 임시 하역이 수행돼야 함
  - 충돌 미발생 또는 임시 하역 수행 필요 → Big-M 기법을 활용하여 모형화 가능
  - w_{pn}: 항구 p에서 노드 n에서의 임시 하역이 수행되면 1, 아니면 0



충돌 금지 제약

$$r_{pn} \leq 1 - \sum_{d \in D(p)} x_{dn} + M \cdot w_{pn}, \quad \forall p \in P, n \in N$$

- $D(p)$: 항구 p에서 적재된 상태에 있으며, 하역되지 않는 수요 집합
- 노드가 점유되어 있다면($\sum_{d \in D(p)} x_{dn} = 1$), 해당 노드는 도달 불가능($r_{pn} = 0$)
- 임시 하역을 진행한다면($w_{pn} = 1$), 해당 노드는 도달 가능
- $M = 1$로 설정 가능



최적화 모형

알고리즘 로직

- F: 경로 고정비
- D_n: 노드 n까지의 최단 거리
- $L(p)$: 항구 p에서 선적되는 수요 집합
- $U(p)$: 항구 p에서 하역되는 수요 집합
- K_d: 수요 d의 차량 수

구현된 최적화 모형

$$\textbf{min} \quad 2 \times \sum_{n \in N} (F + D_n) \left(\sum_{d \in D} x_{dn} + \sum_{p \in P} w_{pn} \right) \quad \cdots \text{ 목적함수: 경로 총 비용}$$

$$\textbf{s.t.} \quad r_{pn} \leq \sum_{n' \in P(n)} r_{pn'}, \quad \forall p \in P, n \in N, \quad \cdots \text{ 제약 ①: 최단 경로 도달 가능 제약}$$

$$r_{pn} \leq 1 - \sum_{d \in D(p)} x_{dn} + w_{pn}, \quad \forall p \in P, n \in N, \quad \cdots \text{ 제약 ②: 충돌 금지 제약}$$

$$\sum_{n \in N} x_{dn} = K_d, \quad \forall d \in D, \quad \cdots \text{ 제약 ③: 수요 제약}$$

$$w_{pn} + \sum_{d \in L(p)} x_{dn} \leq r_{pn}, \quad \forall p \in P, n \in N, \quad \cdots \text{ 제약 ④: 선적 위치 선택 및 경로 제약}$$

$$w_{pn} + \sum_{d \in U(p)} x_{dn} \leq r_{pn}, \quad \forall p \in P, n \in N, \quad \cdots \text{ 제약 ⑤: 하역 위치 선택 및 경로 제약}$$

$$x_{dn}, w_{pn}, r_{pn} \in \{0, 1\}, \quad \forall d \in D, p \in P, n \in N$$

구현 세부사항

알고리즘 구현

- 대회에서 제공된 Python 환경에서 개발

데이터 입력

- **수요 데이터 저장**
  - Boolean array에 저장
  - 수요별 선적/하역 항구
  - 수요별 차량 대수
  - 항구별 점유 수요

모형화

- **BFS 수행**
  - nx.predecessor
  - 최단 경로 선행 노드 반환
  - 최단 경로 거리 반환

해 탐색

- **NoRel 휴리스틱**
  - Gurobi에서 최초 개발
  - 선형계획완화에 의존 X
  - 제한시간 내 해 품질 우수

✓ **휴리스틱 Effort 최대 활용**

해 출력

- **적재 및 하역 노드 확인**
  - np.logical_and(or)
  - Boolean array 연산 활용
- **최단 경로 구성**
  - 거리 순서대로 적재/하역
  - path_backtracking

✓ **해 출력에 1초 할당**

알고리즘 장단점 및 후기

알고리즘 특장점 및 개선방향

😊 알고리즘 특장점

1. 노드 및 항구 수에 비례하는 모형 크기 →Scalability
2. 최적화 Solver만을 활용하는 단순한 접근법

$$\textbf{min} \quad 2 \times \sum_{n \in N} (F + D_n) \left(\sum_{d \in D} x_{dn} + \sum_{p \in P} w_{pn} \right)$$

$$\textbf{s.t.} \quad \sum_{n \in N} x_{dn} = K_d, \quad \forall d \in D,$$

$$r_{pn} \leq \sum_{n' \in P(n)} r_{pn'}, \quad \forall p \in P, n \in N,$$

$$r_{pn} \leq 1 - \sum_{d \in D(p)} x_{dn} + w_{pn}, \quad \forall p \in P, n \in N,$$

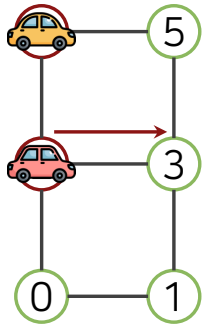
$$w_{pn} + \sum_{d \in L(p)} x_{dn} \leq r_{pn}, \quad \forall p \in P, n \in N,$$

$$w_{pn} + \sum_{d \in U(p)} x_{dn} \leq r_{pn}, \quad \forall p \in P, n \in N,$$

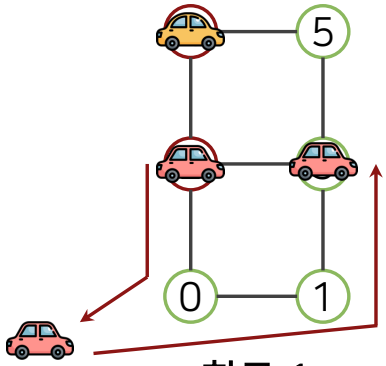
$$x_{dn}, w_{pn}, r_{pn} \in \{0, 1\}, \quad \forall d \in D, p \in P, n \in N$$

🧐 알고리즘 개선 방향

1. 위치 변경을 통한 해 품질 개선
2. 재적재 시 위치 변경을 허용하도록 모형 개선
3. Solver의 휴리스틱에 의존하지 않는 해법 고안



항구 1



항구 1

Q & A

감사합니다.