Roll-on Roll-off 문제에 대한, 경로에 추가 제약을 두는 정수선형최적화 모형

tryAgain팀

서울대학교 산업공학과 시스템최적화연구실
조원우 (newsky0329@snu.ac.kr)

September 19, 2025

최적화 모형을 사용한 의사결정

최적화 모형을 사용한 의사결정 과정

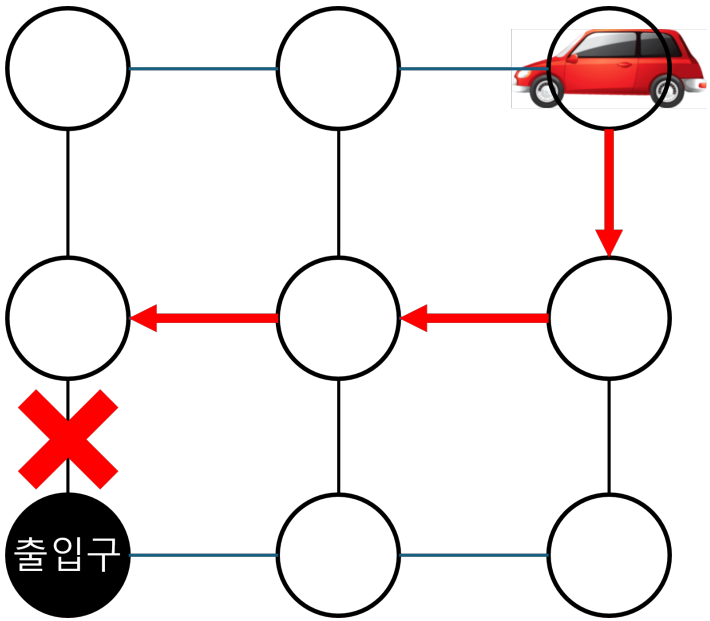
1. 의사결정 사항을 정의:
   → 의사결정의 결과를 나타내는 변수인 결정변수 정의
2. 의사결정이 만족해야 하는 조건 설정:
   → 탐색할 의사결정 집합을 **결정변수를 사용**한 **제약식**들의 교집합으로 정의
3. 의사결정의 목표 설정:
   → 의사결정을 평가하는 지표(목적함수)를 결정변수에 대한 식으로 정의
4. 목표를 달성하며 조건을 모두 만족하는 의사결정 탐색:
   → 최적화 알고리듬으로 최적해 탐색

• 해 집합을 어떻게 정의하느냐에 따라 모형화에 필요한 결정변수/제약식의 수 및 제약식의 강한 정도가 크게 달라질 수 있으며, 이는 해 탐색의 성능과 직결됨.
따라서 **계산 효율성**과 **해 집합의 정확한 묘사** 간의 trade-off를 잘 조절하는 것이 중요함.

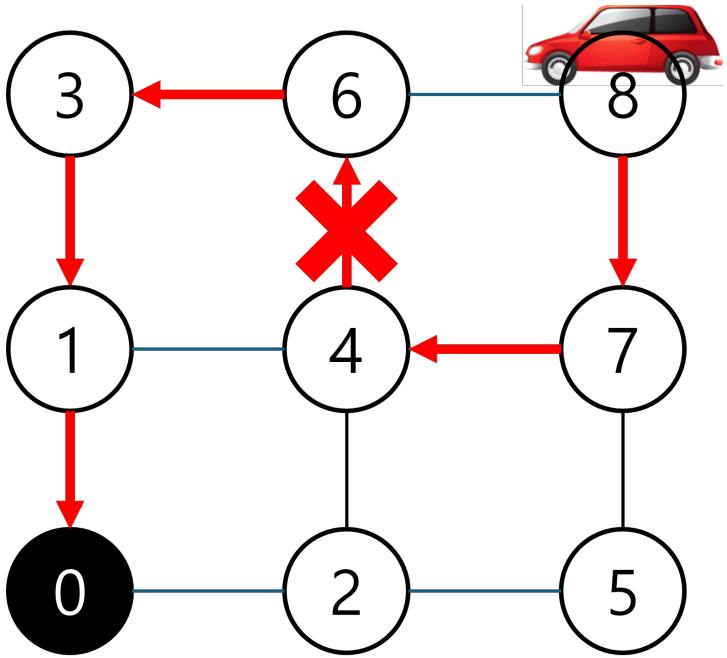
• 이러한 이유로, **차량의 경로에 추가적인 제약**을 두는 **정수선형최적화 모형**을 고안하여 사용하였음.

최적화 모형에서 추가적으로 가정하는 제약

- 고안한 최적화 모형에서는 차량 경로에 다음과 같은 추가 제약들들을 부과한다:
  1. 모든 경로는 출입구에서 시작하거나(적재) 출입구에서 끝난다(하역).
  2. 모든 노드에 서로 다른 '높이' 값을 부여하여, 높이가 커지기만(작아지기만) 하는 경로로 적재(하역)한다.



(a) 첫 번째 제약을 위반하는 경로



(b) 두 번째 제약을 위반하는 경로

- 사용한 최적화 모형에서는 출입구로부터 너비 우선 탐색을 할 때 방문하는 순서를 높이 값으로 사용하였다.

최적화 모형 - 결정변수 및 목적함수

결정변수

- x_{pqi}: 항구 p에서 노드 i에 적재하여 항구 q에서 하역한 차량이 있으면 1, 아니면 0.
- y_{pi}: 항구 p에서, 노드 i로 적재($\Leftrightarrow$노드 i에서 하역)가 가능하면 1, 아니면 0.
- z_{pqr}: 항구 p에서 적재하여 항구 r에서 하역해야 하는 차량을 중간 항구 q에서 하역 후 재적재한 횟수

목적함수: 총 비용 최소화
(경로의 이동 거리는 출입구와의 최단거리로 계산)

최적화 모형 - 제약식

제약식 (수식은 Appendix 참조)

- 주어진 수요를 만족해야 한다.
- 임의의 항구에서, 임의의 노드에 머무르는 차량은 최대 한 대이며, 이 경우 해당 노드는 적재/하역 경로에 포함시킬 수 없다.
- 임의의 항구에서, 임의의 노드에서 하역하는 차량은 최대 한 대이며, 이 경우 해당 노드는 출입구로 하역 가능한 상태여야 한다.
- 임의의 항구에서, 임의의 노드로 적재하는 차량은 최대 한 대이며, 이 경우 해당 노드는 출입구에서 적재 가능한 상태여야 한다.
- 임의의 항구에서, 어떤 노드에서 차량을 적재/하역시킬 수 있다면, 인접한 노드 중 하역 방향에 있는(=높이 값이 더 작은) 노드들 중 하나 이상에서 출입구로 적재/하역할 수 있어야 한다.

제약식을 통해 만족되는 조건들

- 모든 수요 만족
- 모든 시점에 모든 노드에는 최대 한 대의 차량만 존재
- 충돌 없는 적재/하역 가능

최적화 모형의 장점 및 특성

최적화 모형의 장점 및 특성

- 경로의 추가 제약을 둔 덕분에, **경로의 순서를 결정하는 Roll-on Roll-off 문제**를 **경로와 순서를 결정하지 않고 모형화**할 수 있었다!!!
  - 추가 제약으로 인한 해 품질 저하는 크지 않은데, 해 탐색 성능은 크게 증가한다.
  - 특히 내부 이동이 필요 없는 인스턴스의 경우, 원 문제의 최적 목적함수 값에 매우 근접한 해를 빠르게 얻을 수 있다.
- 항구별로 노드의 높이를 설정할 수 있다. 이를 잘 활용하면 해 품질을 더 높일 수 있을 것이라고 기대된다.
- 각 항구에서 적재되어 있는 차량을 전부 하역하고 적재하면 항상 가능해가 보장되는데, 최적화 모형이 이 해를 나타낼 수 있으므로 이 최적화 모형은 항상 가능해를 보장하는 **강건성**을 지닌다.
- 모형 크기
  - 결정변수: O(노드 개수 $\times$ **항구 개수**^{2}) $\rightarrow$ 항구의 수에 영향을 크게 받음
  - 제약식: O(노드 개수 $\times$ 항구 개수)

구현 - 사용 언어 및 소프트웨어

프로그래밍 언어 및 소프트웨어

- 최적화 소프트웨어: Gurobi
- 프로그래밍 언어: C++ (알고리듬), Python (입/출력 가공)

Gurobi의 혼합정수선형계획문제 솔버 옵션

• Gurobi, Xpress를 비롯한 대부분의 (혼합)정수선형계획문제 솔버는 선형완화문제 풀이 기반 분지한계법 (Branch-and-bound) 을 해 탐색 알고리듬으로 사용한다. 이 알고리듬은 전체 탐색 영역 중 일부 영역에 해당하는 선형완화문제를 푸는 것을 반복하는데, 최적화 모형의 크기가 커질수록 선형완화문제를 푸는 데 걸리는 계산적 부담이 커져 해 탐색의 성능이 나빠진다.

• Gurobi에서는 선형완화문제를 푸는 부담이 큰 경우, 빠른 시간에 고품질의 가능해를 제공하고자 선형완화문제를 풀지 않는 휴리스틱인 noRel 휴리스틱을 제공한다.

Table: 분지한계법, Gurobi의 noRel 휴리스틱 비교

	분지한계법	noRel 휴리스틱
작은 인스턴스에서의 성능	✓✓	✓
큰 인스턴스에서의 성능		✓✓
Dual bound의 tightness	✓✓	✓
짧은 시간에 가능해 제공		✓✓✓

⇒ 작은 인스턴스에서는 분지한계법이, 큰 인스턴스에서는 noRel 휴리스틱이 좋다.

알고리듬

알고리듬: 전체 시간 동안 Gurobi의 솔버를 사용하여 최적화 문제의 해 탐색.
가장 개수가 많은 x_{pqi} 결정변수의 수에 따라 Gurobi 솔버 옵션 변경:

- 10000 이하: 전체 시간 동안 기본 Gurobi 솔버(분지한계법) 사용.
  (솔버 파라미터 설정: Appendix 참조)
- 10000 초과: 전체 시간 동안 Gurobi의 noRel 휴리스틱 사용.

알고리듬의 개선 방향

알고리듬의 개선 방향

- 위치 변경:
  - 최적화 모형의 해를 기반으로 위치 변경을 시도하는 후처리 알고리듬 고안
  - 위치 변경을 최적화 모형에 포함시키는 방법
- 문제 크기를 줄이기 위한 휴리스틱:
  - 인접한 노드들을 클러스터링
  - 중간 항구의 배치를 고정해서 문제를 분해
  - 사용될 가능성이 낮은 결정변수 값을 0으로 고정.
- 해 탐색 알고리듬: 최적화 모형을 기반으로 빠르게 좋은 해를 반환하는 알고리듬 고안
  - Lagrangian relaxation 기반 알고리듬
  - Branch-and-price 알고리듬

Thank you!

Appendix - 최적화 모형 관련

파라미터

- N: 노드 개수
- P: 항구 개수
- d_{pq}: 항구 p에서 적재하여 항구 q에서 하역해야 하는 차량의 수
  $(0 \leq p < q \leq P-1)$
- $pos(i)$: 노드 i의 상대적 위치를 나타내는 값. 모든 노드는 0 이상 $N-1$ 이하의 정수 position 값을 가지며, 임의의 두 노드의 position 값은 달라야 한다 (Permutation). 적재 시에는 position 값이 커지는 방향으로만 이동하며, 하역 시에는 position 값이 작아지는 방향으로만 이동한다. Position 값을 정하는 방법은 여러 가지가 있을 수 있음. 본 알고리듬의 최적화 모형에서는 출입구에서 너비 우선 탐색을 수행할 때 방문하는 노드의 순서대로 position 값을 정하였음.
- $pred(i)$: 노드 i에 인접하면서 position 값이 $pos(i)$ 보다 작은 노드들의 집합.

Appendix - 최적화 모형 관련

결정변수

- x_{pqi}: 항구 p에서 노드 i에 적재하여 항구 q에서 하역한 차량이 있으면 1, 아니면 0. $(\forall\ 0 \leq p < q \leq P-1,\ i = 1, 2, \ldots, N-1)$
- y_{pi}: 항구 p에서,
  - 출입구에서 노드 i로 적재 가능하면
  - 노드 i에서 출입구로 하역할 수 있으면
  - 노드 i를 적재/하역하는 경로에 포함시킬 수 있으면

  1, 아니면 0.
  $(\forall\ p = 1, 2, \ldots, P-2,\ i = 0, 1, \ldots, N-1)$
- z_{pqr}: 항구 p에서 적재하여 항구 r에서 하역해야 하는 차량을 항구 q에서 하역 후 재적재한 횟수 $(\forall\ 0 \leq p < q < r \leq P-1)$

목적함수: $\sum_{p=0}^{P-2} \sum_{q=p+1}^{P-1} \sum_{i=1}^{N-1} 2 \times (fixedCost + dist(i)) \times x_{pqi}$

($fixedCost$: 고정 비용, $dist(i)$: 출입구와 노드 i 간의 최단거리)

Appendix - 최적화 모형 관련

제약식 (1/2)

- $\sum_{i=1}^{N-1} x_{pqi} = d_{pq} - \sum_{r=p+1}^{q-1} z_{prq} + \sum_{r=q+1}^{P-1} z_{pqr} + \sum_{r=0}^{p-1} z_{rpq}, \quad \forall\, 0 \le p < q \le P-1$

  $\rightarrow$ 주어진 수요를 만족해야 한다.

- $\sum_{p=0}^{q-1} \sum_{r=q+1}^{P-1} x_{pqi} \le 1 - y_{qi}, \quad \forall q = 1, \ldots, P-2,\ i = 1, \ldots, N-1$

  임의의 항구에서, 어떤 노드에 내내 머무르는 차량은 최대 한 대이며, 이 경우 해당 노드는 적재/하역 경로에 포함시킬 수 없다.

- $\sum_{p=0}^{q-1} x_{pqi} \le y_{qi}, \quad \forall q = 1, \ldots, P-2,\ i = 1, \ldots, N-1$

  $\rightarrow$ 임의의 항구에서, 어떤 노드에서 하역하는 차량은 최대 한 대이며, 이 경우 해당 노드는 출입구로 하역 가능한 상태여야 한다.

- $\sum_{q=p+1}^{P-1} x_{pqi} \le y_{pi}, \quad \forall p = 1, \ldots, P-2,\ i = 1, \ldots, N-1$

  $\rightarrow$ 임의의 항구에서, 어떤 노드로 적재하는 차량은 최대 한 대이며, 이 경우 해당 노드는 출입구에서 적재 가능한 상태여야 한다.

Appendix - 최적화 모형 관련

제약식 (2/2)

- $y_{pi} \leq \sum_{j \in pred(i)} y_{pj}, \quad \forall\, i = 1, \ldots, N-1,\ p = 1, \ldots, P-2$

  $\rightarrow$ 임의의 항구에서, 어떤 노드에서 차량을 적재/하역시킬 수 있다면, 인접한 노드 중 하역 방향에 있는 노드들 중 하나 이상에서 출입구로 적재/하역할 수 있어야 한다.

- $y_{p0} = 1, \quad p = 1, \ldots, P-2$
  $\rightarrow$ 출입구는 항상 접근 가능하다.

- $x_{pqi} \in \{0, 1\}, \quad \forall\, 0 \leq p < q \leq P-1,\ i = 1, 2, \ldots, N-1$

  $\rightarrow$ x 결정변수는 이진 결정변수

- $y_{pi} \in \{0, 1\}, \quad \forall\, p = 1, 2, \ldots, P-2,\ i = 0, 1, \ldots, N-1$

  $\rightarrow$ y 결정변수는 이진 결정변수

- $z_{pqr} \in Z_+, \quad \forall\, 0 \leq p < q < r \leq P-1$

  $\rightarrow$ z 결정변수는 비음 정수

Gurobi 솔버 파라미터

Gurobi 솔버 파라미터

- 분지한계법 알고리듬에서 휴리스틱 비율: 15% (default: 5%)
- 뿌리 노드에서 선형완화문제 푸는 알고리듬: Barrier
- Barrier 알고리듬에서 crossover: 끔

Appendix - 예선 결과

- 예선 알고리듬: 전체 시간 동안 Gurobi의 noRel 휴리스틱 사용.
- 개인적으로 실험했을 때, 하역 후 재적재가 없다고 가정하는 최적화 모형을 Gurobi default solver로 풀면 모든 인스턴스에서 최적해(모든 인스턴스에서 최적 목적함수 값은 BEST 목적함수 값과 같았습니다)를 1분 내에 얻을 수 있었습니다.

Table: 예선 결과

	P1	P2	P3	P4	P5
목적함수 값	231	822	808	1383	1723
BEST 목적함수 값	231	822	808	1383	1711
(최종 제출 기준) 더 좋은 결과 낸 팀 수	0	0	0	0	5

Appendix - 본선 결과

Table: 본선 결과 (1 ∼ 5)

	P1	P2	P3	P4	P5
목적함수 값	1,556	15,039	8,709	969	2,328
BEST 목적함수 값	1,122	11,993	8,511	969	2,100
(최종 제출 기준) 더 좋은 결과 낸 팀 수	1	1	1	0	2

Table: 본선 결과 (6 ∼ 10)

	P6	P7	P8	P9	P10
목적함수 값	6,761	1,750	1,283	757	8,455
BEST 목적함수 값	6,661	1,724	1,253	563	7,711
(최종 제출 기준) 더 좋은 결과 낸 팀 수	0	1	1	4	1

Appendix - 결선 결과

Table: 결선 결과 (1 ∼ 7)

	P1	P2	P3	P4	P5	P6	P7
목적함수 값	5,945	10,984	1,462	2,386	2,386	1,223	2362
BEST 목적함수 값	5,945	10,984	1,438	2,386	2,246	1,223	2360
더 좋은 결과 낸 팀 수	0	0	3	0	1	0	1

Table: 결선 결과 (8 ∼ 14)

	P8	P9	P10	P11	P12	P13	P14
목적함수 값	5460	7906	1255	5528	6473	915	4119
BEST 목적함수 값	5460	3502	1255	5528	6473	764	4110
더 좋은 결과 낸 팀 수	0	9	0	0	0	2	1