고려대학교 KOREA UNIVERSITY

Optimization Grand Challenge 2024

양방향 레이블링 알고리즘을 활용한 묶음 배송 최적화

2024.10.24

하늘청 팀

김하늘

고려대학교 산업경영공학부

고려대학교
KOREA UNIVERSITY

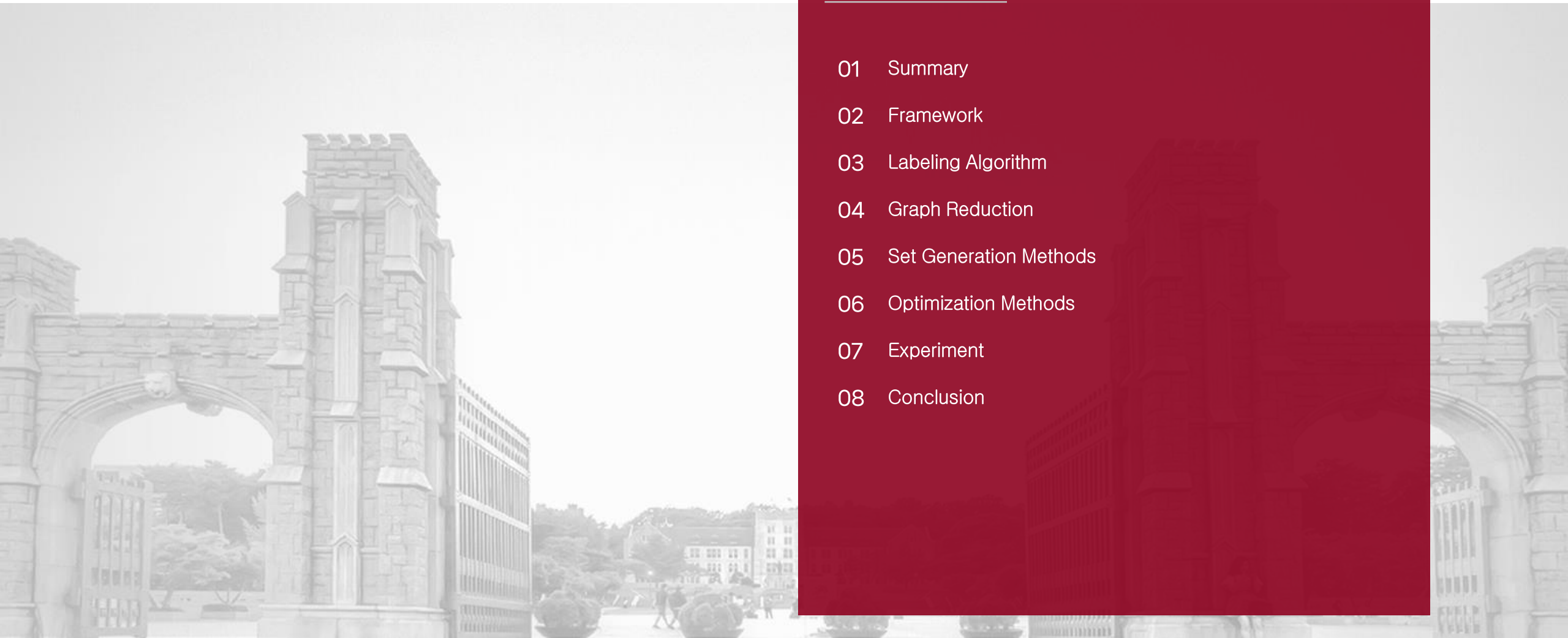


Table of Contents

01 Summary

고려대학교 KOREA UNIVERSITY

- 제안 방법론
  - 경로 단위의 탐색 및 최적화 프레임워크 제안
  - 효율적인 경로 (Useful Path) 탐색을 위한 방법론 제안
    - Network Reduction
    - Bidirectional Labeling Algorithm
    - Iterative Search Methods (군집화 방법 및 비군집화 방법)
  - Set Partitioning Constraints 기반의 경로 선택 문제 정의
    - Dual Price를 반영한 비용 행렬 기반의 반복 경로 탐색
    - 정수 계획법 및 Rounding Heuristic을 적용한 선형 계획법 접근

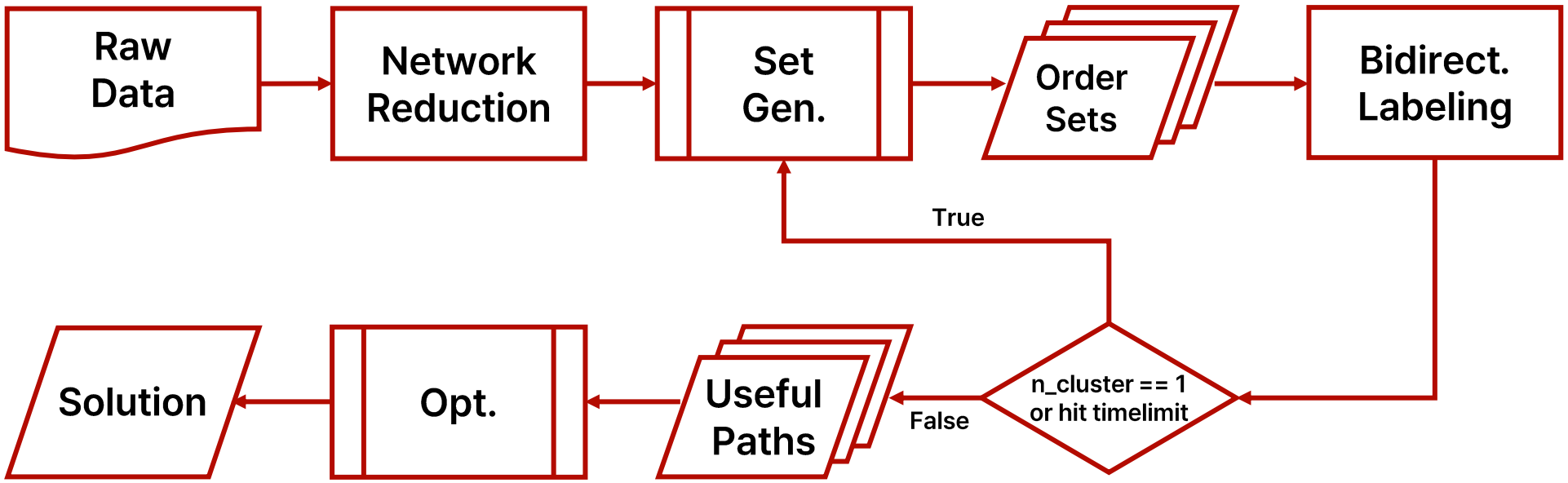
- 실험 결과
  - 전통적인 Heuristic Rule 기반의 Network Reduction Technique은 해 품질을 감소
  - 도보 (Walk) 배달원의 경우, Network Reduction 및 Bidirectional Useful Path 탐색이 효율적
  - 1st Cluster, 2nd Route 방식 中
    - 1st Cluster: 무작위 성질을 가지는 군집화 방법 (KMeans 계열), 1st 매장 위치, 2nd 배달 위치 기반의 이중 군집화가 효과적
    - 2nd Route: Dual Value를 활용하지 않은 일반적인 정수 계획법 문제 풀이가 가장 효과적

1000개 주문 인스턴스에 대한 탐색 시간: 약 2초

02 Framework

고려대학교 KOREA UNIVERSITY

- 경로 단위의 탐색 및 최적화 프레임워크 제안

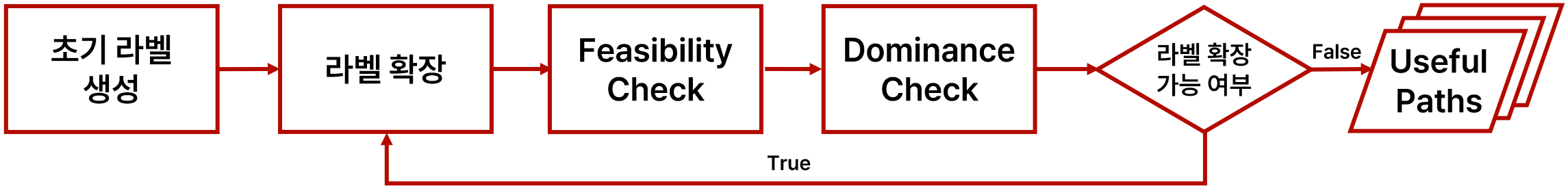


- Set Generation Methods: BiDP로 경로 탐색할 범위 (주문 집합)을 결정하는 방법론
- Optimization Methods: 문제 제약을 고려한 Useful Paths의 최적 조합 탐색

03 Labeling Algorithm

고려대학교 KOREA UNIVERSITY

- 소개
  - 경로 탐색 및 네트워크 흐름 문제에서 주로 사용되는 알고리즘
  - 각 노드나 경로에 대하여 다양한 레이블 (속성)을 부여하여 문제를 해결
  - VRP 계열 문제에 대해 Branch & Price, Column Generation 기반 접근 시,
    Sub Problem (Elementary Shortest Path Problem with Resource Constraints; ESPPRC) 풀이에 활용



- 장점
  - 복잡한 제약 조건을 반영하여 경로 구성 가능
  - Dominance Rule을 기반으로 열등한 경로는 탐색 (확장)하지 않음

03 Labeling Algorithm

고려대학교 KOREA UNIVERSITY

K: $Order\ set$

- **Bidirectional Labeling Algorithm**
  - 경로 생성 시, 양 방향에서 동시에 라벨을 확장하여 결합하는 방법
    - 순방향 (Forward): 시작점에서 목표 지점
    - 역방향 (Backward): 도착점에서 목표 지점
  - 단 방향 (Monodirectional) 탐색보다 일반적으로 탐색 속도가 빠른 것으로 알려져 있음 (Righini and Salani 2006)

- **라벨**
  - 라이더 유형 r에 대한 순방향 라벨 $(L_f^r) = (path\ (p), travel\ distance\ (d), time\ (t))$
  - 라이더 유형 r에 대한 역방향 라벨 $(L_b^r) = (path\ (p), travel\ distance\ (d), time\ (t), volume\ (v))$

- **라벨 초기화**
  - 주어진 주문 집합을 중 하나를 포함하는 라벨 생성
  - 순방향 라벨의 경우, 주문의 픽업 노드($[i]$), 주문의 Earliest Service Time(e_i) 할당
  - 역방향 라벨의 경우, 주문의 배달 노드($[i + |K|]$, 주문의 Latest Service Time(l_i) 할당

03 Labeling Algorithm

고려대학교 KOREA UNIVERSITY

$M: Distance\ matrix$
$T: Travel\ time\ matrix$

- **라벨 확장**
  - 경로의 마지막/처음 방문 노드에서 그와 연결된 이웃 노드로 경로를 확장 (Bellman-Ford Algorithm 방식)

  - 순방향: 라벨 (p, d, t)에서 경로의 마지막 방문 노드가 i이며, j 노드로 경로를 확장하는 경우,
    - 경로: $p' \leftarrow p + [i]$
    - 이동거리: $d' \leftarrow d + M_{ij}$
    - 현재 시간: $t' \leftarrow \max(t + T^r_{ij}, e_j)$

  - 역방향: 라벨 (p, d, t, v)에서 경로의 첫 번째 방문 노드가 j이며, i 노드로 경로를 확장하는 경우,
    - 경로: $p' \leftarrow [i + K] + p$
    - 이동거리: $d' \leftarrow d + M_{i+|K|,j+|K|}$,
    - 현재 시간: $t' \leftarrow \min(t - T^r_{i+|K|,j+|K|}, l_i)$
    - 현재 용량: $v' \leftarrow v + v_i$

03 Labeling Algorithm

고려대학교 KOREA UNIVERSITY

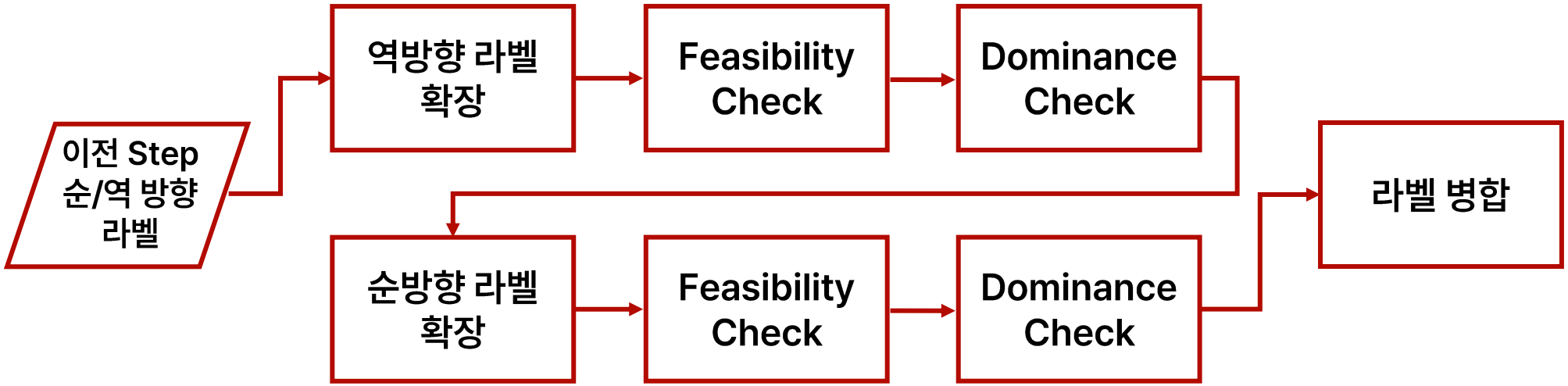
E^r: $Edge\ set\ of\ rider\ type\ r$
C^r: $Capacity\ of\ rider\ type\ r$

- **Feasibility Check**
  - **Strong Feasibility**
    - **도달 가능성:** $(i.j) \in E^r$
    - **용량 제약:** $v + v_i \leq C^r$
    - **주문 집합을 공유하는 역방향 (backward Label) 라벨 존재 여부**
  - **Weak Feasibility**
    - **순방향:** $t' \leq \min_{i \in p'}(-T_{j,i+|K|} + l_i)$
    - **역방향:** $t' \geq \max_{j \in p'}(T_{j,i+|K|} + e_i)$
- **Dominance Check**
  - **방문 노드 집합이 동일한 라벨** L^1, L^2**에 대하여,**
    - **순방향: If** $d^1 \leq d^2, t^1 \leq t^2$ **then** L^1 **dominates** L^2
    - **역방향: If** $d^1 \leq d^2, t^1 \geq t^2$ **then** L^1 **dominates** L^2

03 Labeling Algorithm

고려대학교 KOREA UNIVERSITY

- 라벨 병합
  - 방문 노드 집합이 동일한 순방향 라벨 L_f^1, 역방향 라벨 L_b^2에 대하여 Feasibility Check
    - 순방향 라벨의 마지막 노드 i, 역방향 라벨의 처음 노드 $j + |K|$에 대하여, $t_1 + T_{i,j+|K|} \leq t_2$ 만족 필요
  - Useful path: 병합된 라벨 중 가장 이동 거리가 작은 라벨
- 구현
  - 3가지 방식으로 구현(Python 기본 문법, Numpy + Numba (JIT Compile), Cython (Compile))
  - 매 Step 마다 이전 Step의 라벨에 대해 방문 가능한 모든 노드를 고려하여 라벨 확장



04 Network Reduction

고려대학교 KOREA UNIVERSITY

- **Volume 및 Time Window 기반 간선 집합 구성**
  1. 두 개의 주문 i, j의 자원 합이 라이더 유형 r의 용량을 넘지 않아야 함
  2. 두 개의 주문 i, j에 대해 한 명의 라이더가 처리하기 위해서는 아래의 경로 중 하나 이상 실행 가능해야 함

     I. $i \rightarrow j \rightarrow i + |K| \rightarrow j + |K|$

     II. $i \rightarrow j \rightarrow j + |K| \rightarrow i + |K|$

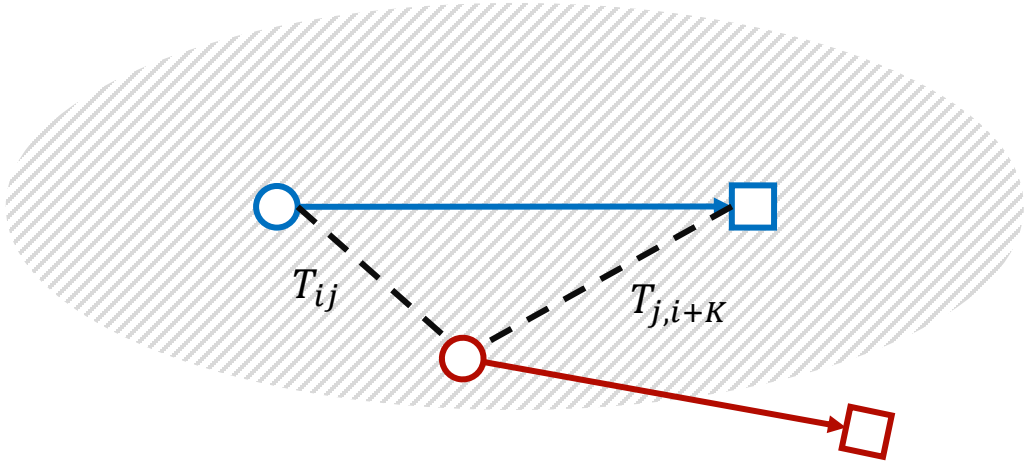
     III. $j \rightarrow i \rightarrow i + |K| \rightarrow j + |K|$

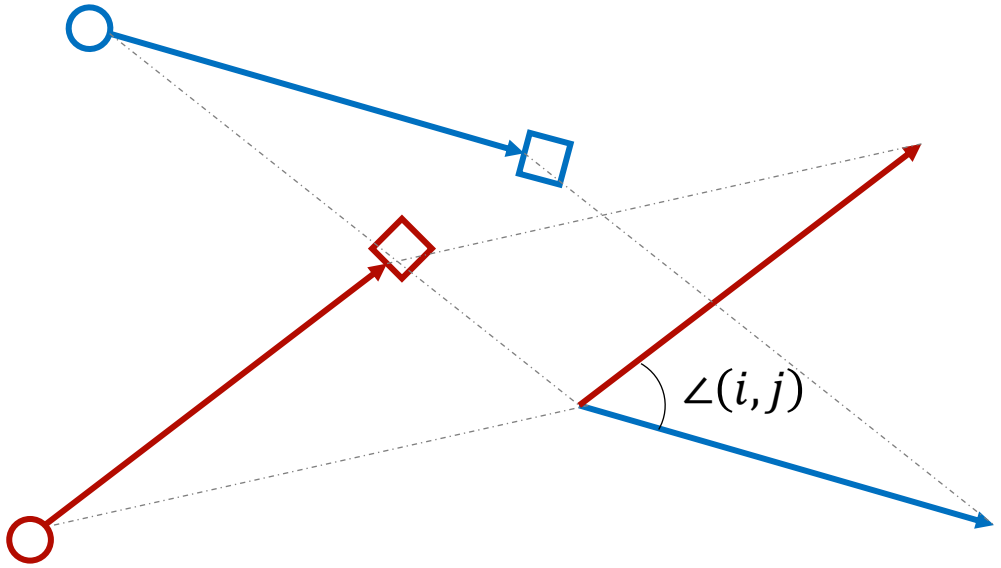
     IV. $j \rightarrow i \rightarrow j + |K| \rightarrow i + |K|$
  - 실행 가능한 경로를 구성하는 간선만으로 라이더 r 별 간선 집합 E^r 구성

- **Neighbors (Heuristic) Rule (Irina et al. 1995)**
  - 1, 2번의 기준과 후술할 3,4번의 기준을 만족할 경우, 두 개의 주문이 인접하다고 정의
  - 주문의 인접성에 대한 전이를 고려
    - 만약 주문 i, j 쌍이 인접하고, j, k 쌍이 인접할 경우, i, k 쌍은 기준을 만족하지 않더라도 인접하다고 판단

04 Network Reduction

- **Neighbors (Heuristic) Rule (Irina et al. 1995)**
  3. 두 개의 주문 i, j에 대한 Time Proximity
     - $T_{ij} + T_{j,i+|K|} \leq \rho T_{i,i+|K|}$ or $T_{ji} + T_{i,j+|K|} \leq \rho T_{j,j+|K|}, \rho > 1$
  4. 두 개의 주문 i, j에 대한 Directionality Gap
     - $\angle(i,j) \leq \beta^{\circ}$





05 Set Generation Methods

고려대학교 KOREA UNIVERSITY

$A: Adjacent\ matrix$

- **Data Selection**
  - 군집화하기 위한 기준 정보 (Coords vs Similarity (RBF Kernel))
  - Similarity (RBF Kernel): 클러스터 내 인접하지 않은 주문 쌍 수를 줄이기 위해 인접성과 거리 정보를 모두 반영
    - $A' = A_{:|K|,:|K|} + A^T_{:|K|,:|K|}$, $A_{:|K|,:|K|} + A^T_{:|K|,:|K|} = A_{|K|:,|K|:} + A^T_{|K|:,|K|:}$
    - $W = \{A'_{ij} M_{ij}\}_{i,j \in \mathrm{K}}$
    - $K(i,j) = \exp\left(-\gamma \left|\left|W_{ij}\right|\right|^2\right), \gamma > 0$
    - 1에 가까울수록 유사하며, 0에 가까울수록 유사하지 않음

- **군집화 알고리즘**
  - 매 Set Generation마다 클러스터 내 노드 수를 확장하여 라벨링 알고리즘의 탐색 범위를 넓힘
  - 무작위 성질이 있는 KMeans 계열 알고리즘을 활용
    - KMeans Clustering
    - Bi-sectoring KMeans Clustering
    - Spectral Clustering

05 Set Generation Methods

고려대학교 KOREA UNIVERSITY

A: $Adjacent\ matrix$

- 군집화 방법
  - 1st 매장 위치 기반 클러스터링 후, 각 클러스터에 대해 2nd 배달 위치 기반 클러스터링
  - 1st 배달 위치 기반 클러스터링 후, 각 클러스터에 대해 2nd 매장 위치 기반 클러스터링
  - Time Window 기반 클러스터링
    1. 구간 $[min(e_i), max(e_i)]$ 에 대해 부분구간 시작점 $e = min(e_i)$, $Interval, Shift$ 선언
    2. 노드 집합 $\{i \in K | \mathrm{e} \leq \mathrm{e_i} \leq \mathrm{e} + Interval\}$ 구성
    3. 변수 $e \leftarrow \mathrm{e} + \mathrm{shift}$
    4. 2,3 번을 $e > max(e_i) - Interval$ 까지 반복

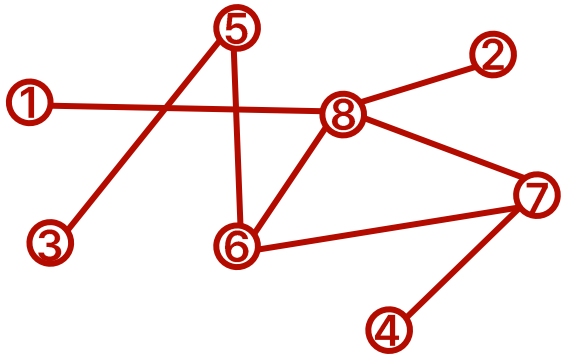
- 비군집화 방법 (인접 주문 소거법)
  - 모든 주문의 인접 주문 수는 상이
  - 픽업을 모두 완료한 뒤, 배송이 가능하다는 강한 제약으로 인해, 어떤 주문을 포함한 가능 경로는 항상 그 인접 주문으로 구성됨
  - 라벨링 알고리즘은 고려하는 네트워크의 크기가 커질수록 연산 비용이 지수적으로 증가

05 Set Generation Methods

고려대학교 KOREA UNIVERSITY

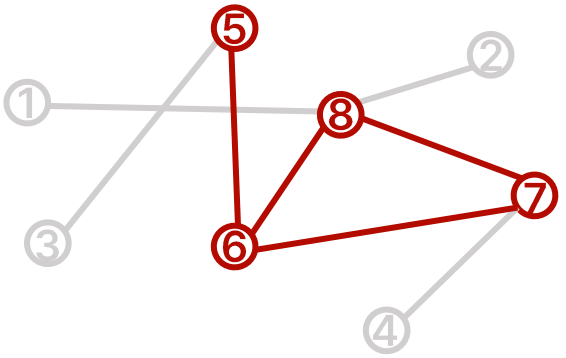
- **비군집화 방법 (인접 주문 소거법)**
  1. 인접 주문 수가 가장 작은 주문 i 선택
  2. 주문 i와 그 인접 주문으로 구성된 주문 집합 형성
  3. 주문 집합에 대해 라벨링 알고리즘 수행
  4. 주문 i의 인접 주문 j와 인접한 주문 집합에서 주문 i 제거

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



1,2,3,4 주문에 대해 인접 주문 소거법 수행 후

1	2	3	4	5	6	7	8
0	0	0	0	1	2	2	2



- 만약 인접 주문 수가 큰 경우, 거리 및 추출된 횟수를 기반으로 분포를 형성하고, 분포 기반의 샘플링 수행

06 Optimization Methods

고려대학교 KOREA UNIVERSITY

- **Master Problem (MP)**
  - **Sets**

    K : 주문 집합

    R : 라이더 유형 집합

    P_r : 라이더 유형 r 별 Useful Path 집합, $\forall r \in R$

  - **Parameters**

    $c_{r,p}$: 라이더 유형 r 이 경로 p 를 따라 배달하는데 발생하는 비용, $\forall p \in P_r, r \in R$

    $a_{i,r,p}$: 라이더 유형 r 이 경로 p 를 따라 처리하는 주문 내에 주문 i 가 있을 경우 1, 아니면 0, $\forall i \in K, p \in P_r, r \in R$

    n_r : 라이더 유형 r 의 가용 가능 수, $\forall r \in R$

  - **Decision Variable**

    $x_{r,p}$: 라이더 유형 r 이 경로 p 를 따라 이동 할 경우 1, 아니면 0, $\forall p \in P_r, r \in R$

06 Optimization Methods

고려대학교 KOREA UNIVERSITY

- **Master Problem (MP)**
  - Objective: 비용 최소화

$$min \quad \Sigma_{r\in R}\Sigma_{p\in P_r} c_{r,p}x_{r,p}$$

  - Constraints
    - Set Partitioning Constraints: 모든 주문은 하나의 라이더에만 할당

$$s.t. \quad \Sigma_{r\in R}\Sigma_{p\in P_r} a_{i,r,p}x_{r,p} = 1 \quad ,\forall i \in K$$

    - 유형 별로 경로 할당되는 라이더 수는 유형 별 라이더 가용 가능 수를 넘을 수 없음

$$\Sigma_{p\in P_r} x_{r,p} \leq n_r \quad ,\forall r \in R$$

    - Binary Constraints

$$x_{r,p} \in \{0,1\} \quad ,\forall p \in P_r, r \in R$$

- **구현**
  - $Ax = b$ 형태로 변형하여 Gurobi 모델 선언 (A는 Sparse Matrix)

06 Optimization Methods

고려대학교 KOREA UNIVERSITY

- **접근법**
  - Master Problem에 대해 풀이
  - Restricted Master Problem으로 풀이
    - Dual Values (λ)를 반영한 Edge Cost Matrix (C) 구성
      - $C_{i,j} \leftarrow C_{i,j} - \frac{1}{4}\lambda_i - \frac{1}{4}\lambda_j$
      - $C_{i,j+|K|} \leftarrow C_{i,j+|K|} - \frac{1}{4}\lambda_i - \frac{1}{4}\lambda_j$
      - $C_{i+|K|,j+|K|} \leftarrow C_{i+|K|,j+|K|} - \frac{1}{4}\lambda_i - \frac{1}{4}\lambda_j$
    - Edge Cost Matrix를 기반으로 주문 집합을 생성, 라벨링 알고리즘을 통해 Useful Path 수집
  - Rounding Heuristic
    - 매 RMP 풀이 이후, 아래의 기준에 의거하여 RMP 상에서 의사결정 변수를 소거하고 제약식을 업데이트
      - 의사결정 변수 값이 1인 경우
      - 의사결정 변수 값이 분수로 나타나는 변수 중, 아래의 손실 값이 가장 작은 의사결정 변수
        - $\left(1 - \lambda_{r,p}\right) \cdot \left(\frac{c_{r,p}}{\sum_{i \in K} a_{i,r,p}}\right)$

07 Experiments

고려대학교 KOREA UNIVERSITY

- 실험 환경
  - CPU: 12th Gen Intel® Core™ i9-12900KS, 5.00GHZ
  - GPU: RTX 2080 TI
  - RAM: 32GB, 4800MHZ

- 양방향 라벨링 알고리즘 구현에 따른 탐색 시간 비교
  - 알고리즘에 입력되는 주문 집합이 클수록 Numpy + Numba 기반의 성능이 우세하였음 (Warm-up 고려)

Instance	# of Orders	Search Time (s)		
		Python 기본 문법	Numpy + Numba	Cython
Problem1_Test3	100	2.49	22.37	2.32
Problem1_Test1	300	203.18	825.69	247.93

07 Experiments

고려대학교 KOREA UNIVERSITY

- 군집화 알고리즘에 따른 해 품질 비교
  - 인접성을 반영한 Similarity 보다 Coords를 사용하는 것이 효과적으로 나타남

Instance	Time Limit	# of Orders	Objective Value			
			(1) KMeans	(2) Bi-sectoring KMeans	(3) Spectral Clustering	(1)+(2)
Problem1_Test1	60	100	2205.62	2200.58	2280.07	2204.31
Problem1_Test2	60	200	1994.97	1989.61	2069.16	1990.75
Problem1_Test3	60	300	3576.24	3576.24	3592.02	3576.24

- 군집화 방식에 따른 해 품질 비교
  - 방법 (1), (2)를 같이 사용하는 것이 효과적으로 나타남

Instance	Time Limit	# of Orders	Objective Value				
			(1) 1st Pick-up 2nd Delivery	(2) 1st Delivery 2nd Pick-up	(1) + (2)	인접 주문 소거법	Time Window + (1) + (2)
Problem2_Test1	60	1000	2601.95	2632.47	2594.94	2609.76	2627.25
Problem2_Test2	60	1000	2604.95	2647.72	2578.01	2649.02	2648.61
Problem2_Test3	60	300	2096.53	2112.58	2074.38	2093.90	2079.28
Problem2_Test4	60	500	2253.98	2285.89	2255.82	2392.92	2290.26

07 Experiments

고려대학교 KOREA UNIVERSITY

- **Network Reduction 방법 비교**
  - Time Window만을 사용하여 Network Reduction하는 것이 가장 효과적으로 나타남

Instance	Time Limit	# of Orders	Objective Value			
			$\beta: 90°, \rho: \sqrt{2}$	$\beta: 138°, \rho: \sqrt{2}$	$\beta: 138°, \rho: 4$	$\beta: 180°, \rho: \infty$
Problem2_Test1	60	1000	2790.35	2632.47	2644.25	2594.94
Problem2_Test2	60	1000	2768.77	2647.72	2661.73	2578.01
Problem2_Test3	60	300	2227.41	2112.58	2137.64	2074.38
Problem2_Test4	60	500	2495.42	2285.89	2318.47	2255.82

- **Optimization Methods에 대한 비교**
  - MP를 푸는 것이 가장 효과적으로 나타남

Instance	Time Limit	# of Orders	Objective Value		
			RMP + IP	RMP + Rouding	MP
Problem1_Test1	60	100	2209.21	2246.32	2204.31
Problem1_Test2	60	200	2204.64	2026.25	1990.75
Problem1_Test3	60	300	3576.24	3621.95	3576.24

08 Conclusion

고려대학교 KOREA UNIVERSITY

- 알고리즘
  - 양방향 Labeling Algorithm을 통해 Useful Path를 효율적으로 탐색하고자 하였음
  - 프레임 워크 내에서 다양한 Order Set Generation 방법을 고려하였음

- 구현 및 실험 이슈
  - Python으로 작성한 알고리즘의 경우, Large Instance에서 algorithm() Return시 변수 삭제에 30~40초의 시간이 요구
    → Memoization X, Cython 사용
  - 마지막 업로드 기준 Memoization X, Cython 버전을 업로드하였기에 구현 단계에서 이슈를 개선한 코드를 테스트 필요
  - 제안한 방법론에 대하여 다양한 조합을 구성 및 실험, 그에 대한 분석 미비

- 참여 후기
  - 알고리즘과 더불어 구현 언어 및 기술에 대한 많은 공부와 노력이 필요함을 느낌

THANK YOU