Operations Research Lab
Workshop

Speaker : Min Joon Kim

Aug 10, 2022

Papers

① **딥러닝 기반 반도체 물류 반송 시스템 다중 시점 처리 능력 예측 연구**
최정우(고려대학교)

② **심층신경망이 적용된 반송시간 추정을 통한 OHT의 동적 라우팅**
최지원(포항공대)

③ **OHT 차량 건강 상태 및 물동량을 고려한 작업 할당**
조문기(KAIST)

④ **모방학습을 활용한 동적 Job Shop 스케줄링**
이제훈(KAIST)

⑤ **강화학습 및 GNN 기반 가상발전소의 Renewable Energy Certificate(REC) 시장 최적 중개 의사결정 모형 개발 연구**
김대호(포항공대)

Keywords : 반도체, 물류, 강화학습

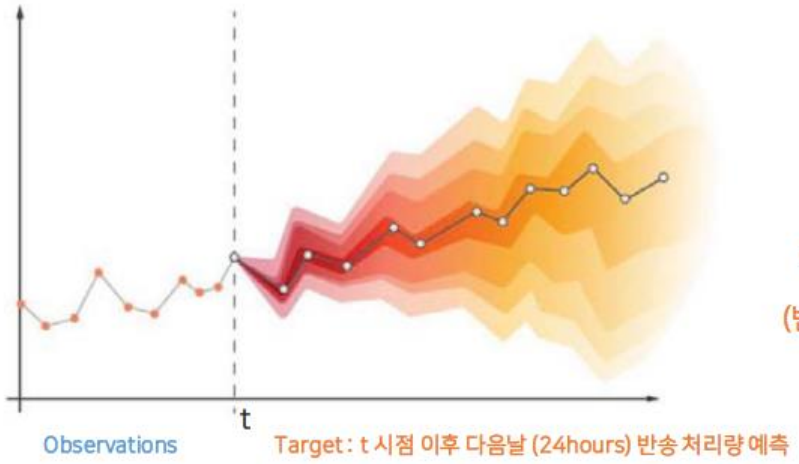
딥러닝 기반 반도체 물류 반송 시스템 다중 시점 처리 능력 예측 연구

최정우, 강형원, 김정섭, 강필성
고려대학교 산업경영공학부, 삼성전자 혁신센터

반송 처리량 예측 모델 연구

▪ 문제 정의

- **Time-series Multi-horizon Forecasting**
  - 과거 데이터와 미래에 알고 있는 데이터를 바탕으로 다중 시점(24hours) 시스템 처리량 예측
  - 예측을 통해 생산 라인 운영 최적화(시스템 사전 대응, 작업 일정 할당 최적화)



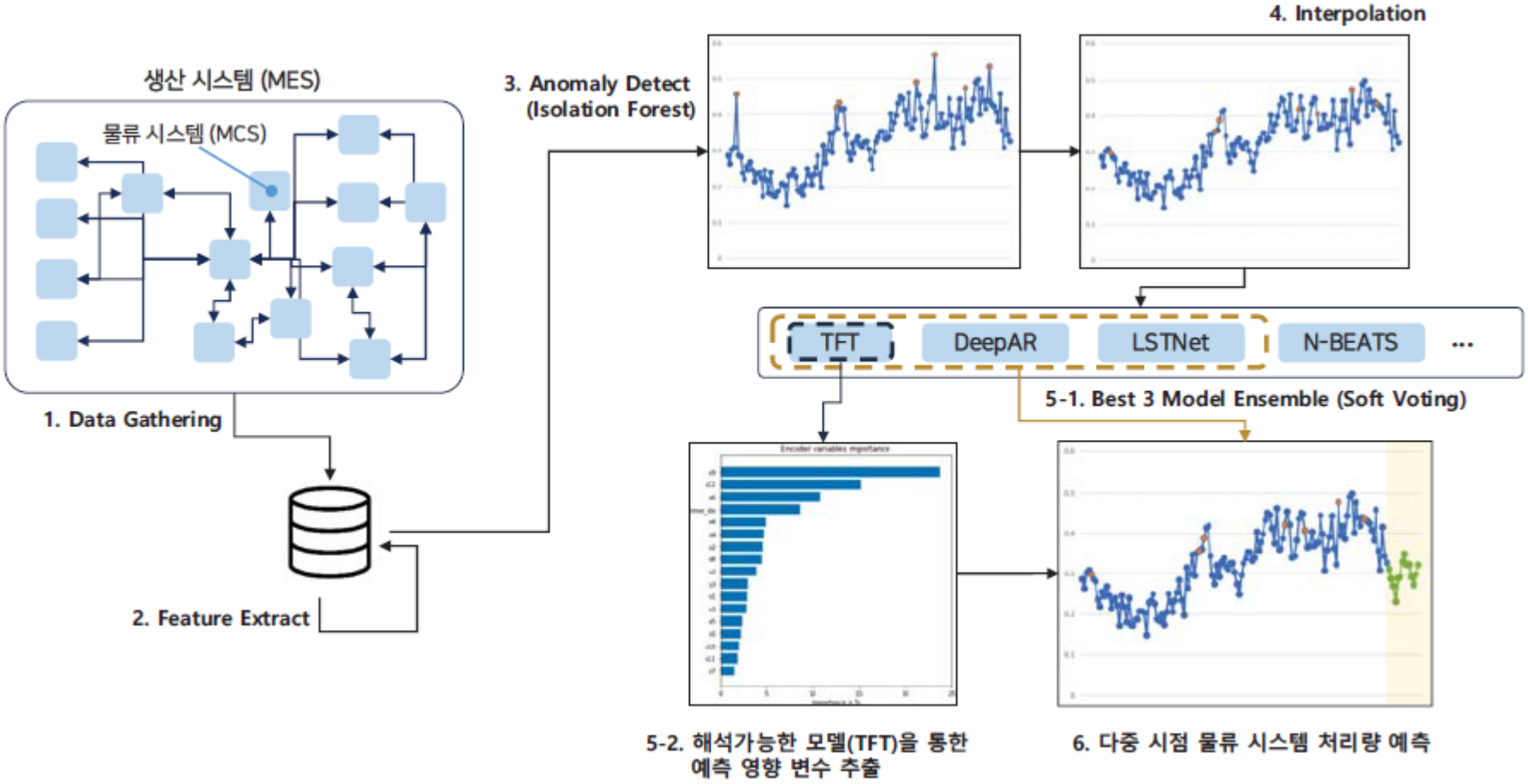
Known future inputs (미리 알 수 있는 변수)　Discrete forecast horizon (예측 기간 = 24hours)

$$\hat{y}_{t+\tau} = f(y_{t-k:t}, \boldsymbol{x}_{t-k:t}, \boldsymbol{u}_{t-k:y+\tau}, \boldsymbol{s}_i, \tau)$$

Target (반송처리량)　Observations (관측 치)　static metadata (정적 메타 정보)

Framework

- **딥러닝 기반 물류 반송 시스템 처리량 예측 모델 Framework**



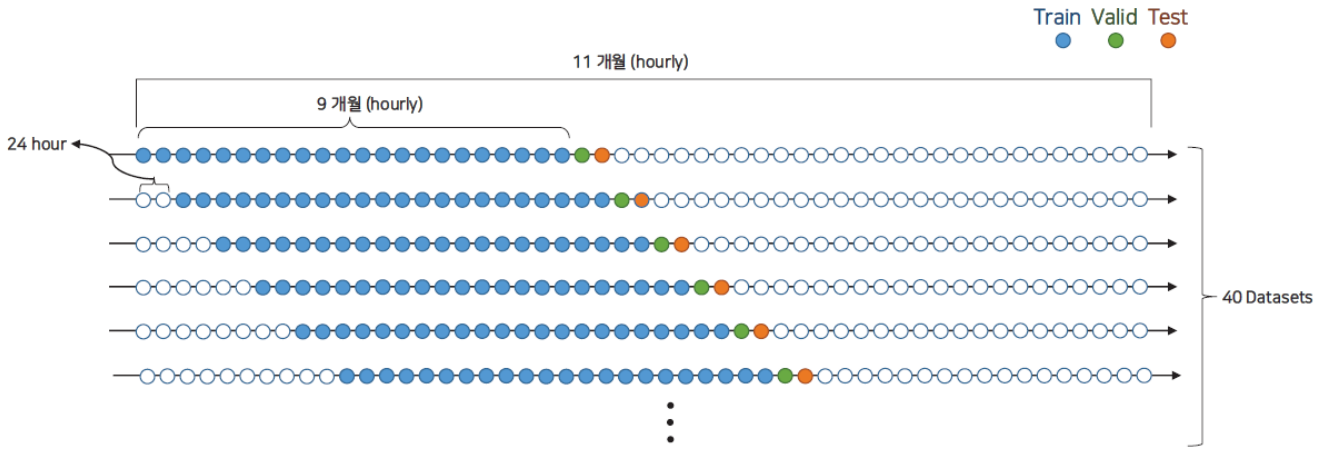
About inputs

▪ Feature engineering

- 수집한 데이터를 물류 시스템의 상태를 나타낼 수 있는 지표화(Feature exraction)
- Time interval(Hourly)을 갖는 통계값으로 추출(Multivariate time series data)
- Observed past inputs (물류반송량, 재공 현황, OHT 사용률 등)
- Known future inputs (Shift 교대시간, 공휴일, 계획된 라인 작업 일정 등)

▪ Time series cross validation

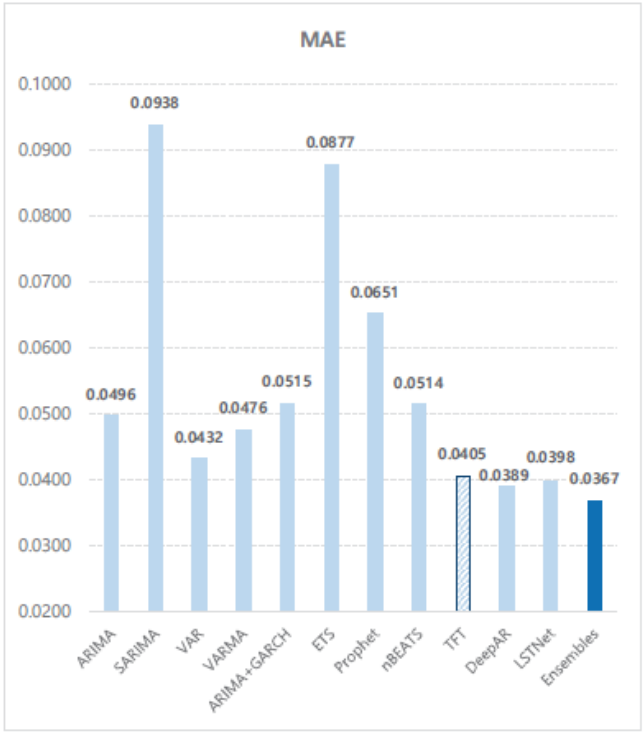
- 9개월치 데이터로 모델 학습, 7일 간 Input값을 사용하여 다음날 예측을 수행함.

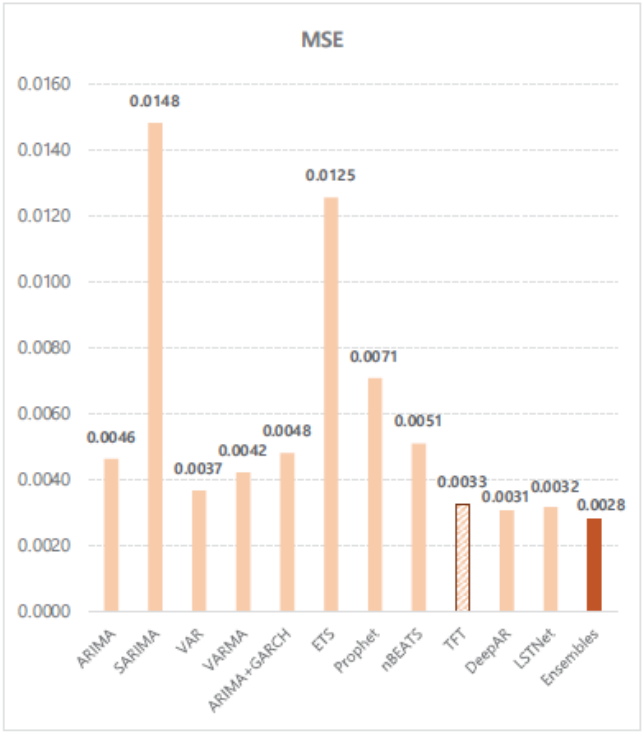


Results

▪ 예측 모델별 실험 결과 비교

Method	MAE	MSE
ARIMA	0.0496	0.0046
SARIMA	0.0938	0.0148
VAR	0.0432	0.0037
VARMA	0.0476	0.0042
ARIMA+GARCH	0.0515	0.0048
Exponential Smoothing	0.0877	0.0125
Prophet	0.0651	0.0071
nBEATS	0.0514	0.0051
TFT (Anomaly Interpolation)	0.0405	0.0033
DeepAR (Anomaly Interpolation)	0.0389	0.0031
LSTNet (Anomaly Interpolation)	0.0398	0.0032
Ensembles (Anomaly Interpolation)	0.0367	0.0028





Conclusion

Summary

- 높은 예측 성능을 갖는 딥러닝 모델을 통해 사전 대응 가능한 물류 처리량 예측 방법을 제시
- 예측에 기여한 인자 추출을 통해 시스템 모니터링 및 분석 시간 단축 및 자동화
- 비지도 학습 이상탐지 모델을 활용한 이상치 보간으로 딥러닝 예측 성능 향상

Future works

- 정확한 반도체 생산 라인 환경을 반영(Digital Twin) 하기 위한 변수 다각화
- 생산 라인의 변동성을 반영하기 위하여 분포 변화에도 강건한 예측을 달성할 수 있는 학습 방법론

심층신경망이 적용된 반송시간 추정을 통한 OHT의 동적 라우팅

최재원, 유태영, 최동구(포항공과대학교 산업경영공학과)

동적 환경에서 OHT 동적 라우팅

▪ Overhead Hoisting Transport(OHT)

- OHT는 생산라인에서 DNP이퍼를 운반하는 이송 장비 시스템
- 복잡한 설비에서 대규모 운행량을 처리할 수 있도록 OHT의 효과적인 경로 선택이 필요

Semiconductor Fab

Overhead Hoisting Transport

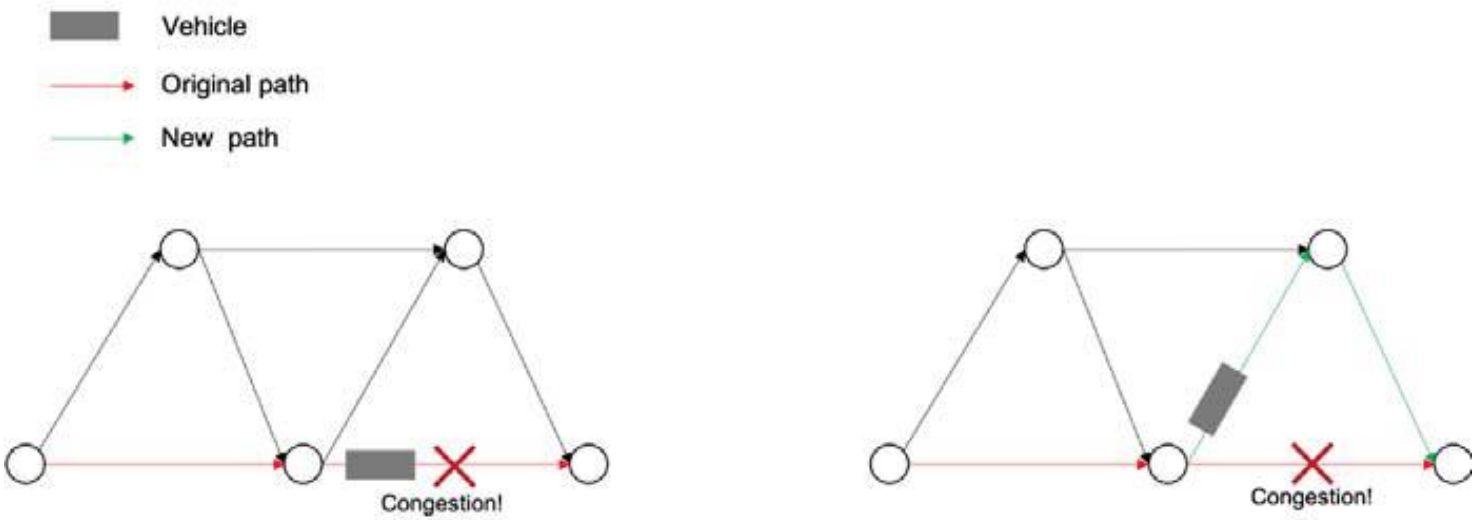
▪ 동적 라우팅

- 경로를 진행하면서 경로를 수시로 수정하는 방법
- 환경을 관찰하고 정체를 예상하여 경로를 결정

▪ 동적 환경

- OHT의 위치 및 레이아웃 상황이 수시로 변화하며 복잡한 환경을 조성함
- 각자 이동하는 다른 OHT들로 인해 실제 반송 시간이 변경될 수 있음

동적 환경에서 OHT 동적 라우팅

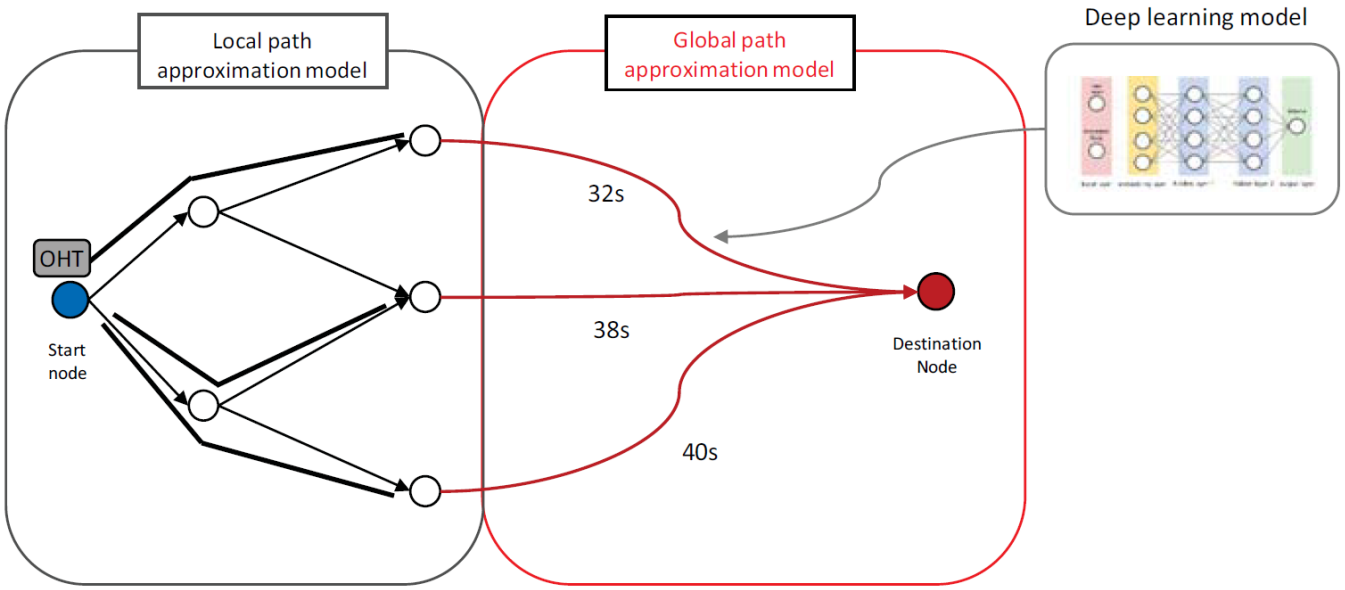


- **동적 라우팅**
  - 경로를 진행하면서 경로를 수시로 수정하는 방법
  - 환경을 관찰하고 정체를 예상하여 경로를 결정

- **동적 환경**
  - OHT의 위치 및 레이아웃 상황이 수시로 변화하며 복잡한 환경을 조성함
  - 각자 이동하는 다른 OHT들로 인해 실제 반송 시간이 변경될 수 있음

Key idea

- **경로의 근거리 이동 시간과 원기리 이동 시간 추정에 서로 다른 접근 방식을 적용**
  - 근거리 이동시간 추정 : Dijkstra 알고리즘을 활용한 지역경로탐색 모형(Local path approximation)
  - 원거리 이동시간 추정 : 심층신경망을 활용한 광역거리추정 모형(Global path approximation)



Local Path Approximation

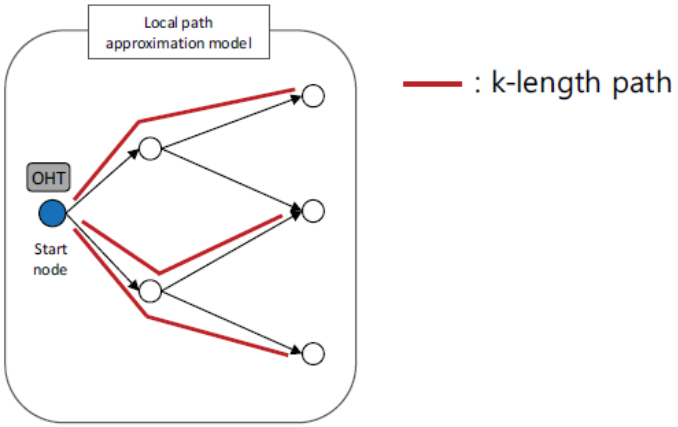
▪ 혼잡도를 반영한 Rail의 이동시간 예상

- 빈 Rail을 이동하는 시간을 기반으로 OHT 대수에 비례하여 Congestion penalty를 부여

$$\text{edge travel time} = \text{empty edge travel time} * (1 + \text{congestion penalty})$$

▪ 부분적인 Dijkstra 경로 탐색 수행

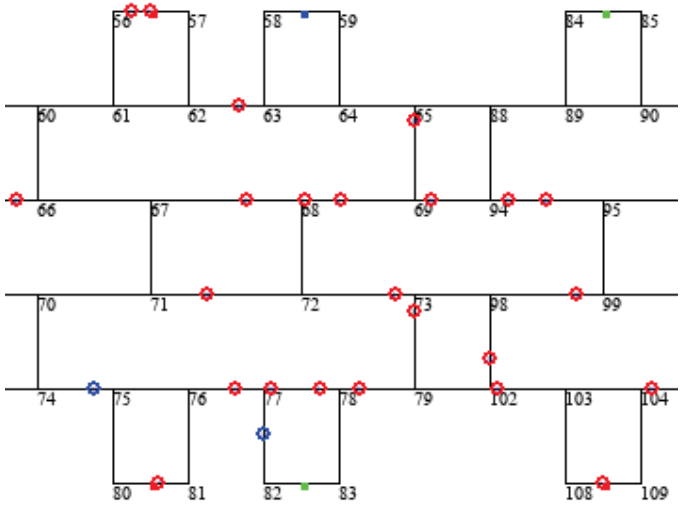
- 시작 노드부터 depth k 만큼 알고리즘 수행
- k-length path를 생성함으로써 search space를 시작 노드부터 k-depth로 한정할 수 있게 됨.



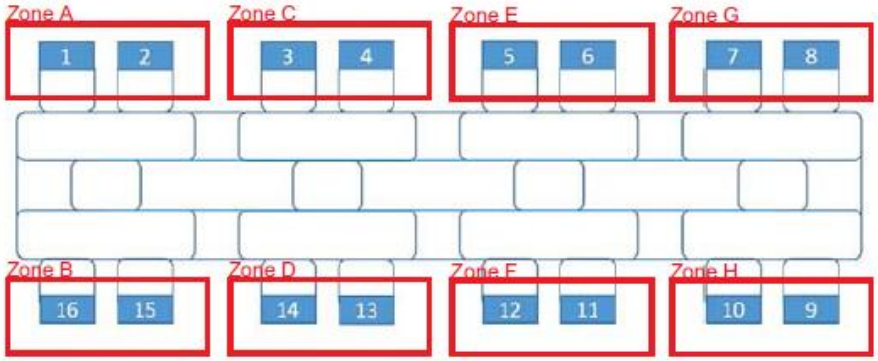
Global Path Approximation

▪ DNN을 활용하여 이동시간을 추정

- Layout features
  - Edge information : Rail 위의 OHT 대수(혼잡도)
  - Load information : 머신의 작업량 수와 관련한 정보(OHT 흐름)



Edge information



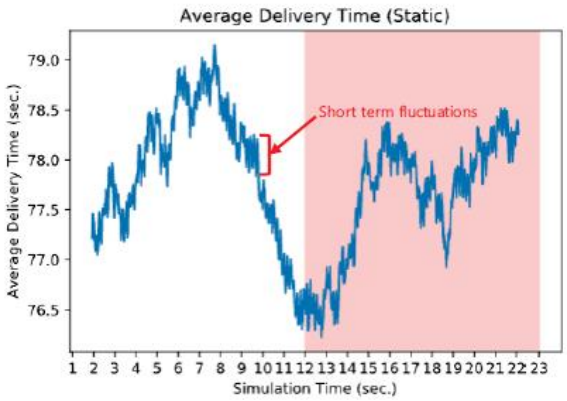
각 머신 지역에 할당된 작업의 개수(Retrieve, Deliver, Call)

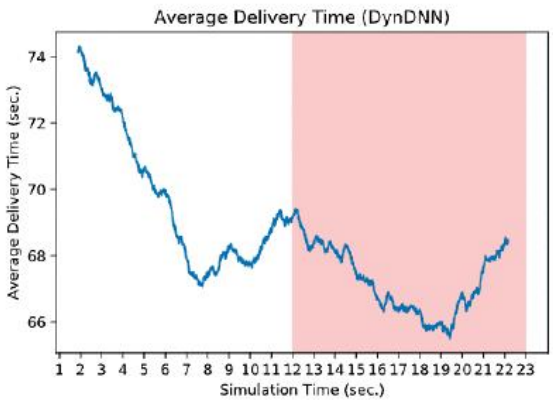
Load information

Results

- **혼잡 시나리오에서 안정적으로 OHT를 운영하여 반송시간을 감소시킴**

  - Static Dijkstra
  - QLWBR(Kaist)
  - Proposed method





Conclusion

Summary

- 정체구간을 회피하며 동적으로 경로 수정을 하여 반송시간을 줄임.
- 시스템 계산 복잡도를 완화한 방법론을 개발함.

Future works

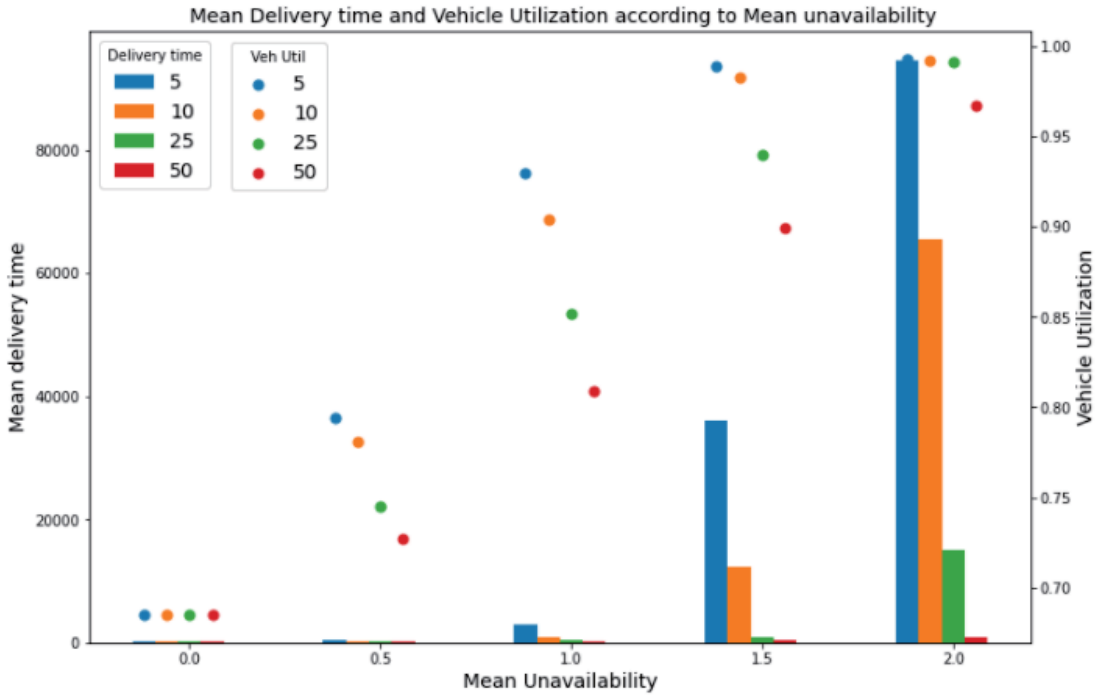
- 큰 레이아웃에서 학습 시간이 오래걸리고 계산 부담이 증가하는 문제 해결
- 레이아웃의 그래프 관계를 활용한 GNN을 활용한 접근
- Agent 사이에 정보를 공유할 수 있는 Multi-agent 관점에서의 접근

OHT 차량 건강 상태 및 물동량을 고려한 작업 할당

조문기, 장영재(KAIST)

OHT 건강상태와 물동량이 시스템에 미치는 영향

- **건강 상태에 따라 Delivery time이 크게 증가함.**
  - 특히, 물동량이 많은 Scenario에서 시스템 성능이 악화
  - 이를 해결하기 위해 Heuristic 방법론을 제안하였음.



Heuristic method

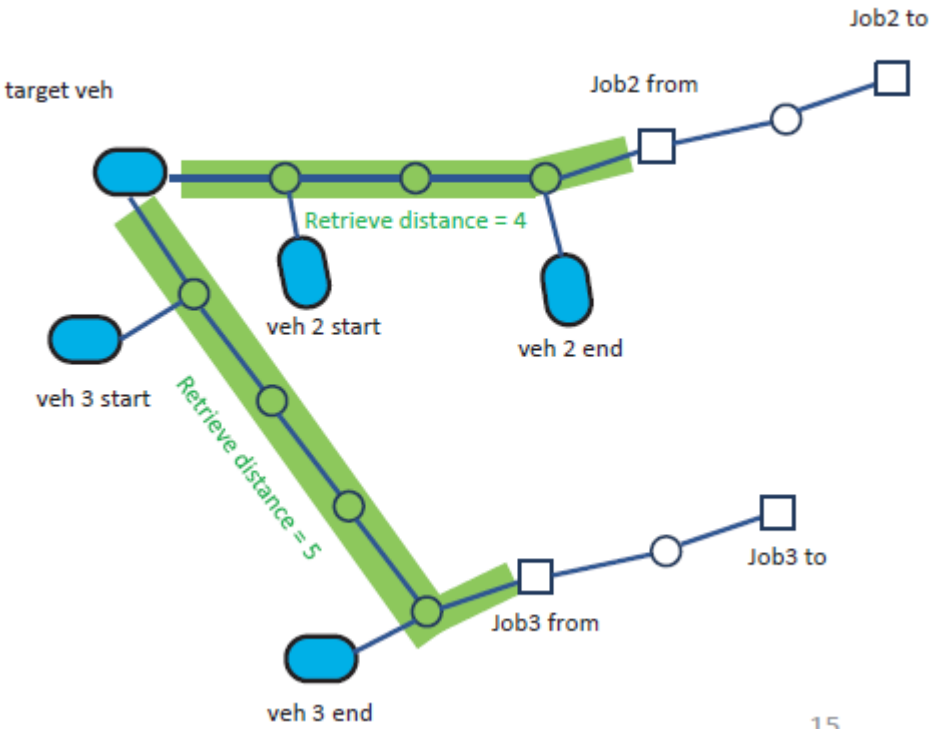
▪ Heuristic 작동과정

- **OHT 차량이 Idle 상태일 때 휴리스틱 동작**
  - 해당 idle OHT vehicle이 target vehicle
- **Waiting job을 수행하는 경로 중 고장 차량이 존재하면 해당 waiting job에 비할당**
  - 모든 waiting job을 수행하는 경로에 failure veh 존재하면 작업할당 중지
- **Target vehicle이 각 waiting job을 수행할 때 거치는 edge들의 물동량 측정**
  - Edge의 길이와 해당 Edge를 지나는 OHT 차량의 Unavailability 고려하여 연산
- **(Retrieve distance) * (Overlap edge length) 가 가장 작은 작업에 할당**
  - Unavailability 및 Overlap edge length가 Penalty 역할
  - Ui를 지수연산하여 unavailability가 높은 소수 차량에 더 높은 Penalty 부여

$$RD_j * \left(\sum_{i \in \{x | (V_x \neq TV) \cap (V_x\ status \neq idle)\}} \left(OEL_{ij}\right)^{U_i}\right)$$

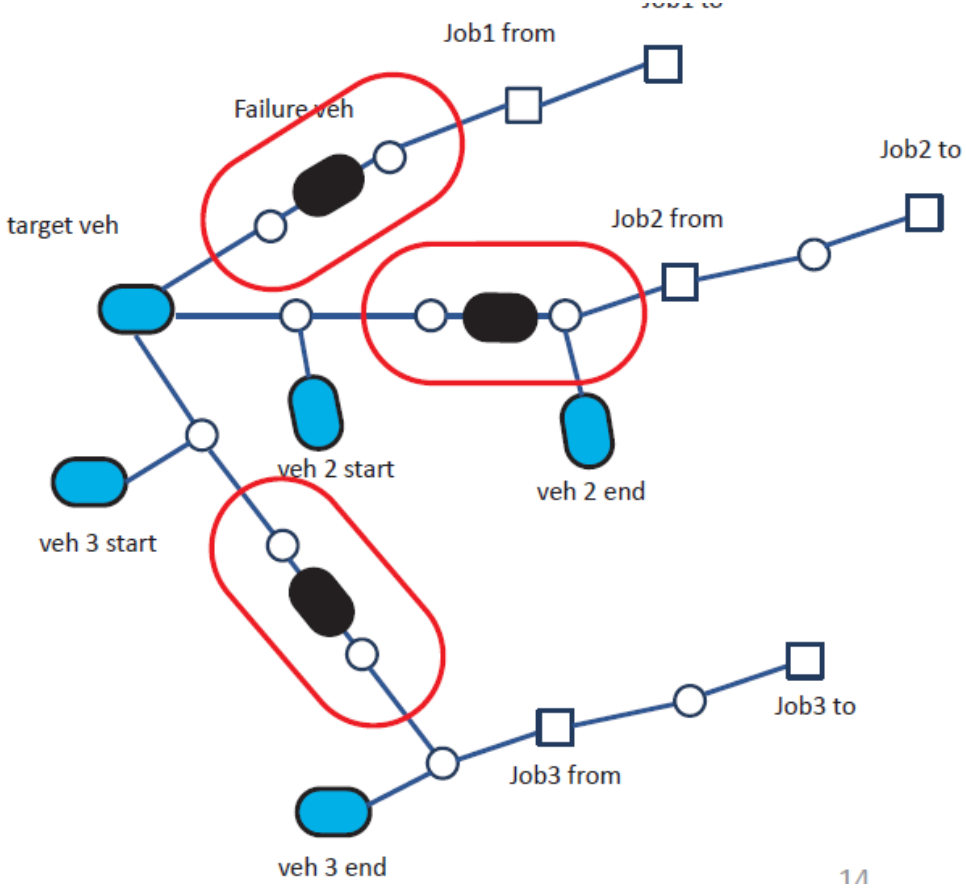
Heuristic method

- **Target vehicle에서 각 waiting job에 대한 Retrieving distance 계산**



Heuristic method

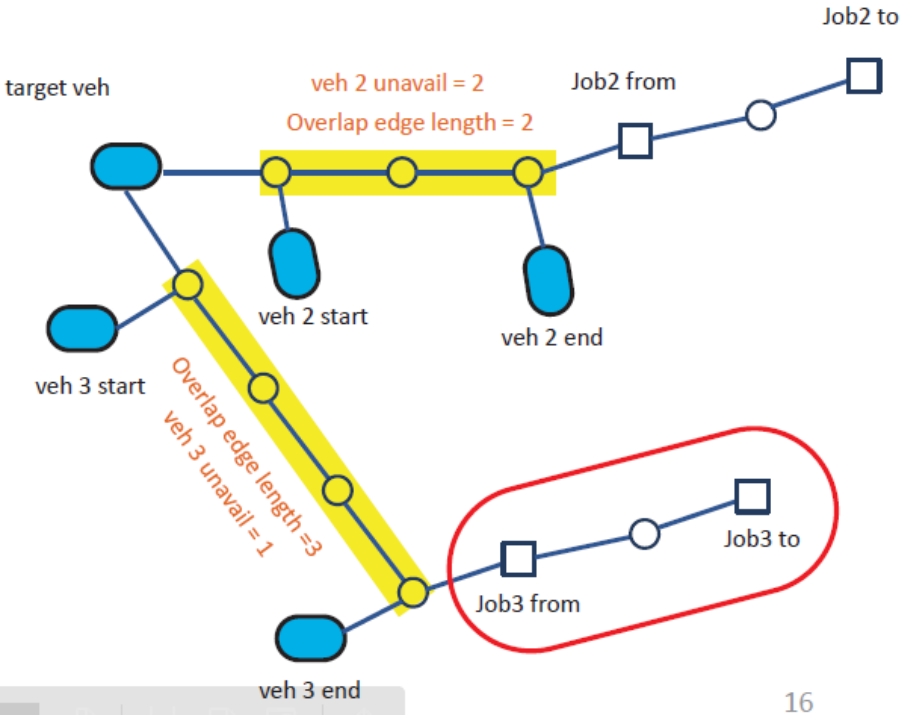
- 각 작업 경로 중 고장 차량이 있으면 해당 작업 비할당



Heuristic method

- **Target veh과 veh 들이 작업 수행 때 거치는 경로 중 겹치는 경로들을 구한 뒤, 그 경로들의 길이에 veh의 unavailability 연산 수행**

Job 2 : Retrieving distance * (Overlap edge length $^{\text{veh unavail}}$) = 4 * 2^2 = 16
Job 3 : Retrieving distance * (Overlap edge length $^{\text{veh unavail}}$) = 5 * 3^1 = 15
➔ Dispatch target veh to Job 3



16

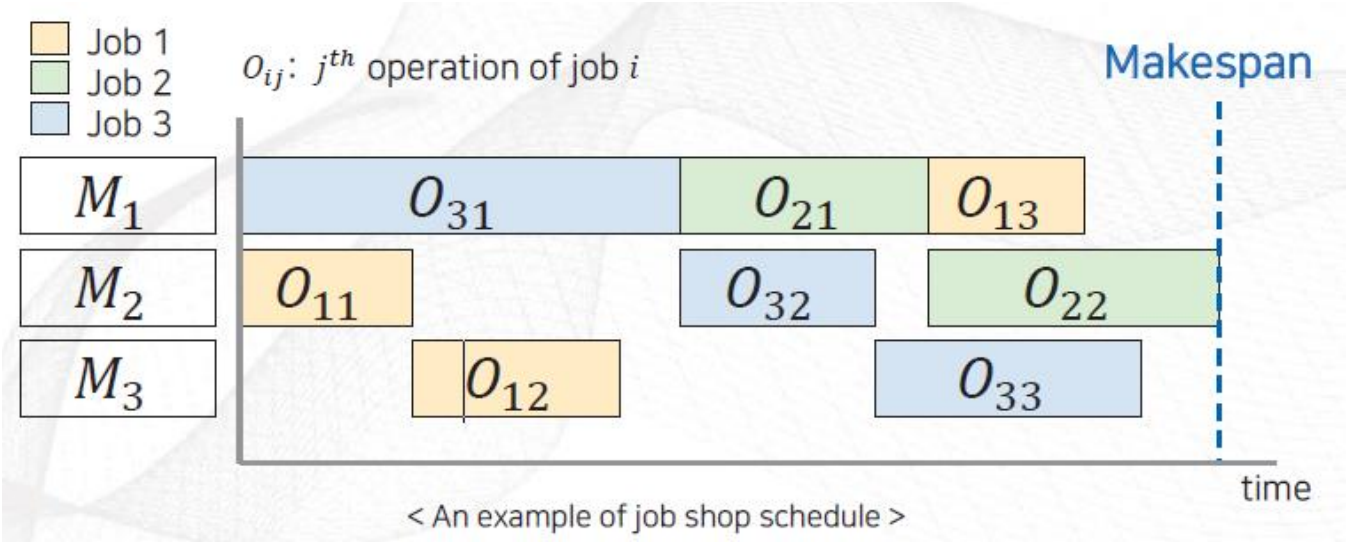
모방학습을 활용한 동적 Job shop 스케줄링

이제훈, 김현정(KAIST)

Background

▪ Job shop scheduling problem(JSSP)

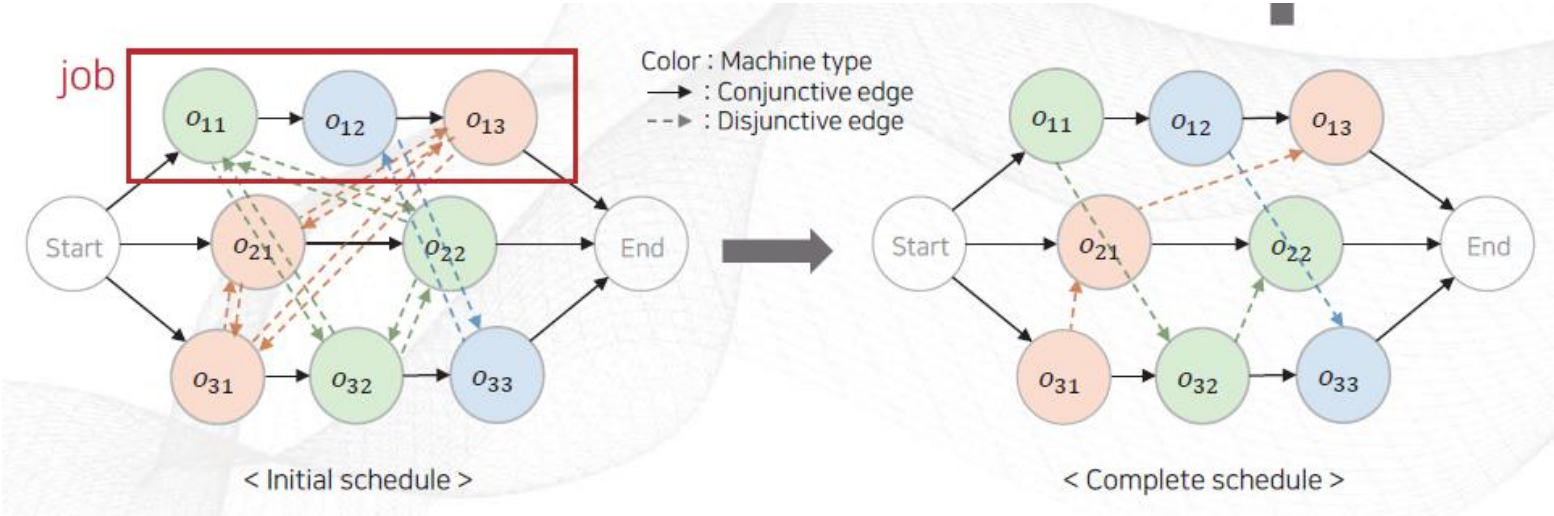
- It is NP-hard(Garey and Johnson, 1976).
- Multiple jobs have to be processed on multiple machines.
- Each job consists of a sequence of operations, which have to be processed to complete the job.
- Each operation must be processed on a predetermined machine.
- We need to determine the sequence of operations on each machine.
- General objective : Minimize makespan(= maximum compltion time of jobs)



< An example of job shop schedule >

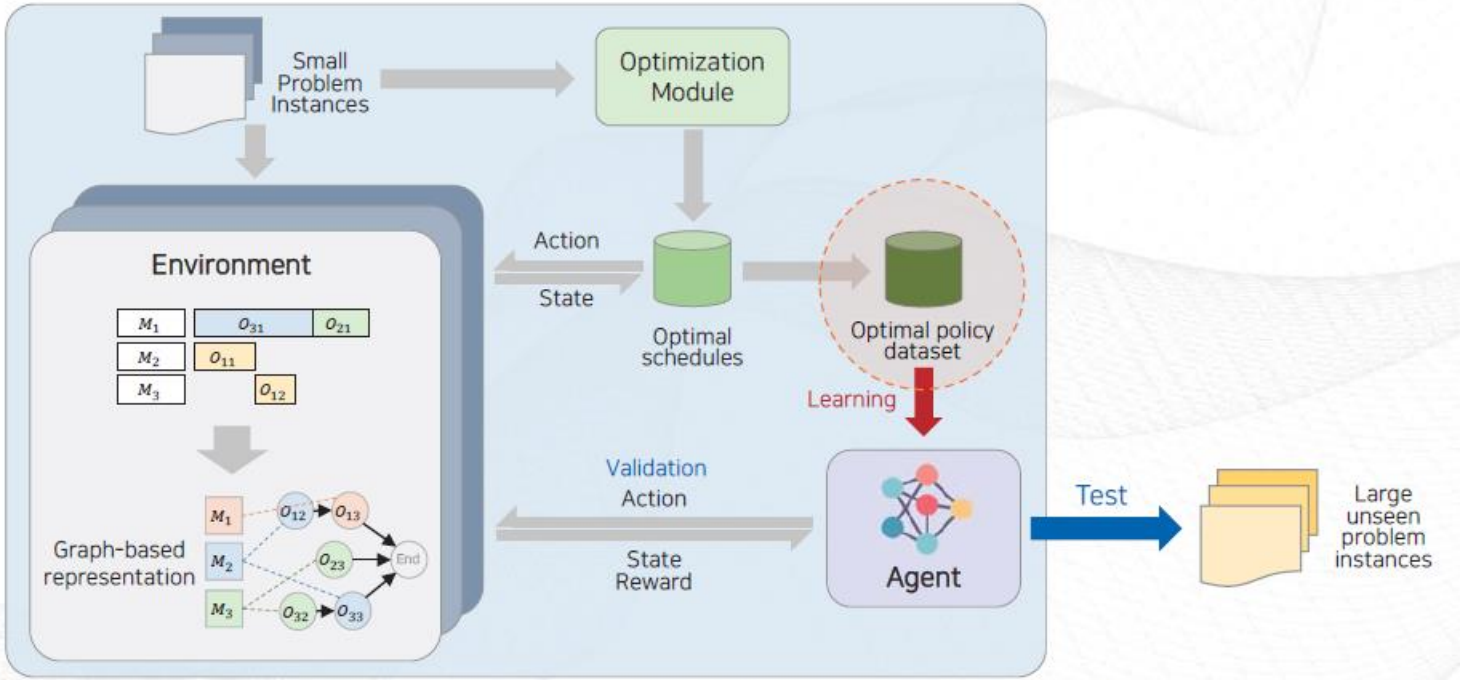
Background

- **Disjunctive graph**
  - Useful modeling tool for JSSP
    - We only need to choose a direction of disjunctive arcs.



Framework

- **GNN-based scheduler using imitation learning**
  - The proposed GNN-based scheduler can be applied whenever the problem size is changed.
  - It uses optimal solutions instead of exploration.



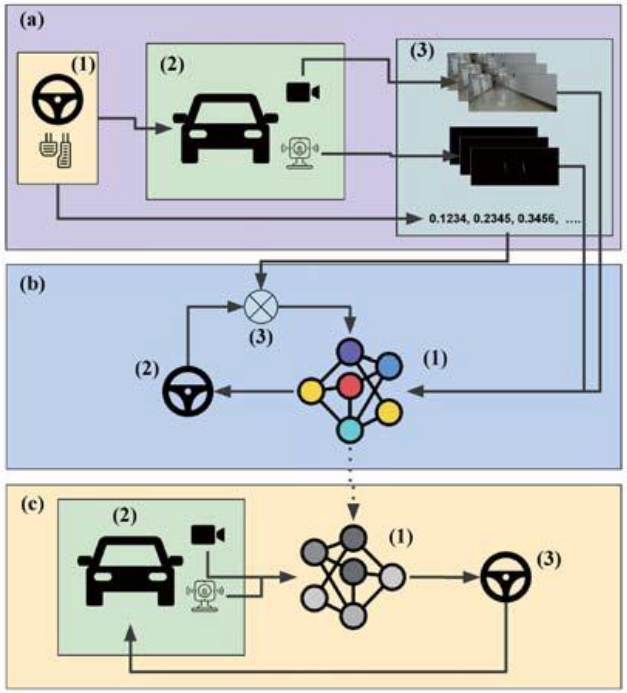
< The framework of a GNN-based scheduler using imitation learning >

Imitation learning

- **GNN-based scheduler using imitation learning**

  Behavior cloning

  - One of the imitation learning methods(Hussein et al. 2017, Attia and Dayan 2018).
  - It does not expect any reward function.
  - It just learns the expert's trajectories.
  - We use the optimal schedules derived by constraint programming of JSSP as an expert dataset



Results

- **Comparing with dispatching rules**

< Optimality gap(%) for JSSP instances according to methods >

	Test set		
Method	tai 15x15	tai 50x15	tai 50x20
random	31.37	25.03	31.40
SPT	25.89	24.11	26.86
LPT	41.22	41.21	45.15
SRPT	47.68	38.47	48.12
LRPT	19.15	16.86	19.15
STT	44.6	41.1	47.24
LTT	19.49	14.28	16.36
LOR	40.93	35.56	45.00
MOR	20.53	17.37	18.88
IL_rsv_dyn_2	**18.84**	**11.46**	**14.86**

< Makespan for dynamic JSSP instances according to methods >

	Test set	
Method	dynamic tai 50x15	dynamic tai 50x20
random	3628.1	4237.1
SPT	3585.3	4096.7
LPT	4054.4	4546.2
SRPT	3993.5	4610.9
LRPT	3410.4	3713.4
STT	3999.5	4672.6
LTT	3357	3679.8
LOR	3889.7	4527.1
MOR	3389.4	3738.2
IL_rsv_dyn_2	**3291.1**	**3646.1**

Conclusion

- **Design policy imitation method for JSSP**
  - Use GNN-based structure in order to make the proposed method size-agnostic
  - Define a more efficnet action set by considering the reservation of operations
  - Propose a policy generation method
  - Propose a GNN model and node features

- **Future works**
  - Considering another dynamic environments(machine breakdown)
  - Using a look-ahead planning method such as Monte Carlo tree search to improve the performance

Thank you

Any questions?