Practical Applications of Neural Combinatorial Optimization

2024 . 06 . 28

최종원

자기소개

- **최종원(Choi Jong Won)**
  - 고려대학교 산업경영공학과 재학 중
  - Data Mining & Quality Analytics Lab(김성범 교수님 연구실)
  - 석사 과정(2023.03 ~)

- **연구 관심 분야**
  - Machine learning & Deep learning Algorithms
  - Multivariate Time Series Data
  - Deep Reinforcement Learning for Scheduling

목차

- **Neural Combinatorial Optimization**
  - 조합 최적화 문제란?
  - 전통적 방법론의 한계
  - NCO 관련 연구
    - Pointer Networks (2015, NIPS)
    - Neural Combinatorial Optimization with Reinforcement Learning (2017, ICLR)

- **Practical Applications of Neural Combinatorial Optimization**
  - Pathfinding Problem
    - Path Planning using Neural A* Search (2021, ICML)
  - Cluster Tool Scheduling
    - Introduction of Cluster Tools & Scheduling
    - Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search (2024, IEEE Transactions on Automation Science and Engineering)

Neural Combinatorial Optimization

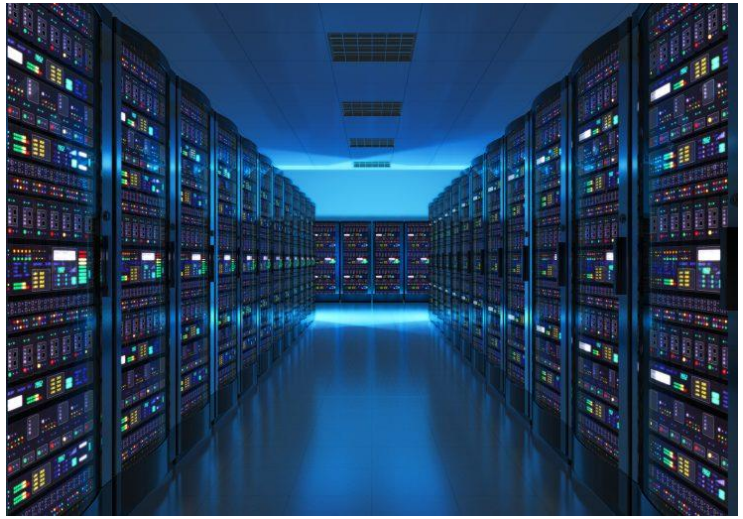
Introduction

조합 최적화 (Combinatorial Optimization)

- **조합 최적화 문제**
  - 특정 조건 하에서 최적 해를 찾는 문제
  - 다양한 실 세계 문제를 모델링 하는데 사용 됨

생산 설비 최적화

자원 할당 최적화



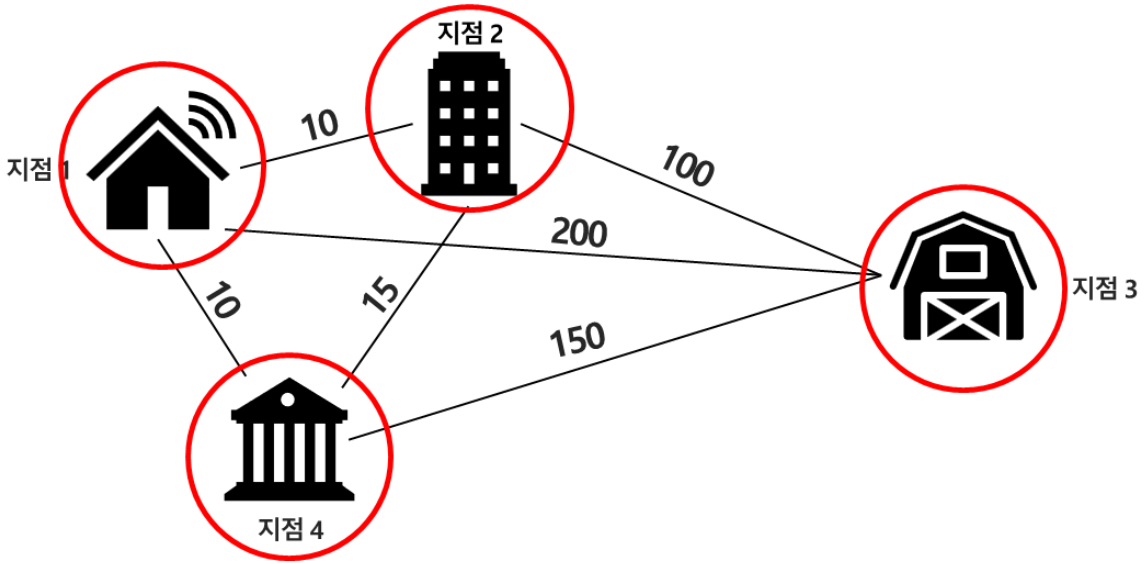
운송 경로 최적화

Introduction

Neural Combinatorial Optimization (NCO)

❖ **조합 최적화 문제**

- NP-Hard 문제로, 정확한 해를 찾는데 많은 시간이 들거나 불가능
- 대표적인 조합 최적화 문제
  - Travelling Salesman Problem (TSP) : 여러 도시를 한 번씩 방문하고 출발지로 돌아오는 가장 짧은 경로 찾기



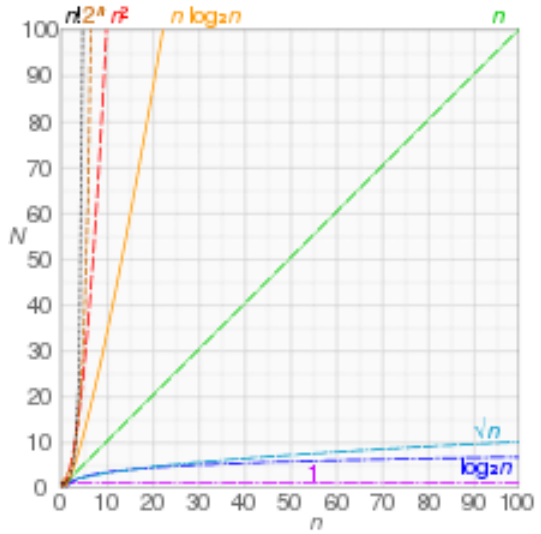
Introduction

Neural Combinatorial Optimization (NCO)

❖ **전통적 방법론의 한계**

- 조합 최적화 문제는 휴리스틱 또는 메타 휴리스틱 알고리즘으로 최적해를 찾음
- 전문 지식 필요 : 문제에 대한 깊은 이해가 필요하며 일반화가 불가능함
- 확장성 부족 : 문제의 크기가 커짐에 따라 계산 복잡도가 급격히 증가 → 대규모 문제 적용이 어려움
- 변동성 대응 어려움 : 실 세계 문제는 변화하는 폭이 잦음 → 알고리즘 수정 필요

Time Complexity



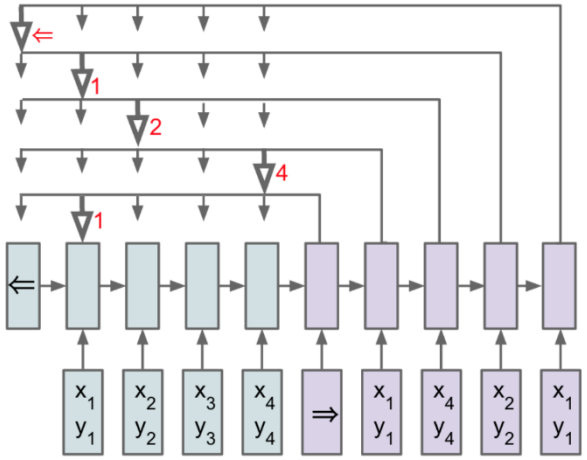
Introduction

Neural Combinatorial Optimization (NCO)

❖ **Neural Combinatorial Optimization (NCO)의 등장**

- 딥러닝과 강화학습의 발전으로 NCO 연구 증가
- 신경망을 이용하여 조합 최적화화 문제를 푸는 접근 방식
- 자동화된 학습 : 전문가 개입 없이 데이터로 문제 해결 방법을 학습
- 유연성 및 확장성 : 다양한 문제에 대해 동일한 모델을 적용 가능
- 실시간 적용 가능성 : 빠른 추론 속도를 통히 실시간으로 최적의 해를 제공

Pointer Networks [2015, NIPS]



Pointer Networks [2015, NIPS]

Background

❖ **RNN의 한계**

- 순환 신경망(RNN)은 시퀀스 데이터를 처리하는데 사용
- 전통적인 RNN은 입력과 출력이 고정된 길이로 제공될 때만 작동 함

❖ **Seq2Seq 모델**

- 고정된 출력 사전 크기 문제 존재

❖ **Attention Mechanism**

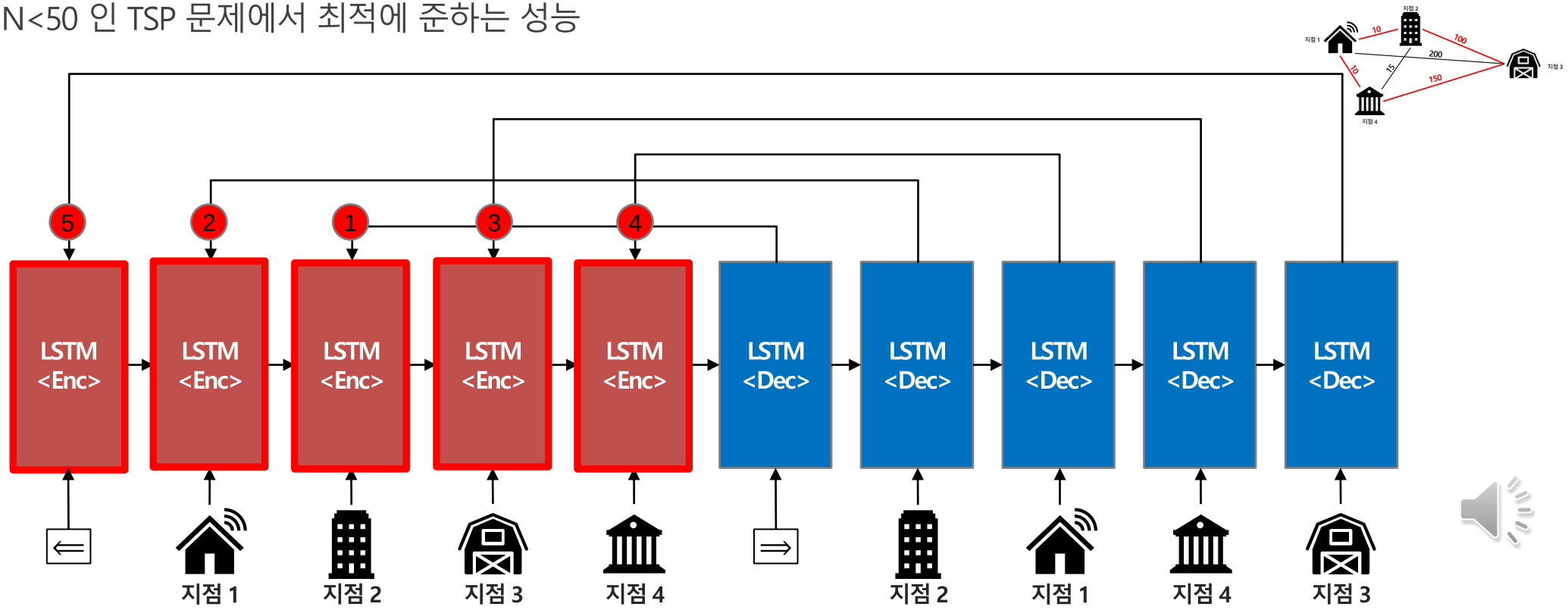
- 인코더 상태 참조로 성능 개선
- 여전히 고정된 출력 사전 크기 문제 존재

Pointer Networks [2015, NIPS]

Background

❖ **포인터 네트워크 (Ptr-Net)**

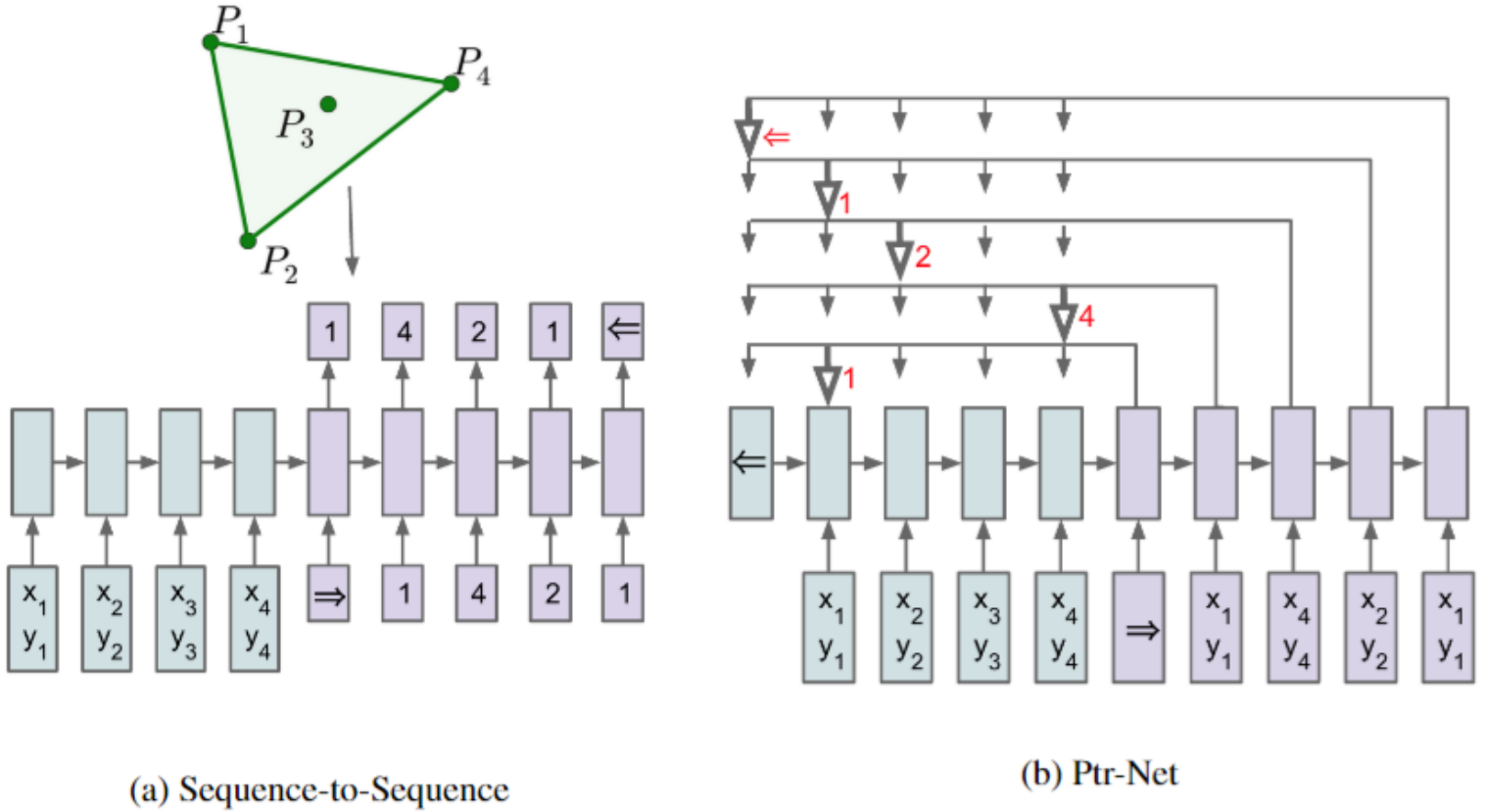
- 가변적인 입력을 가리키는 Pointer Network를 제안
- 3종류의 Combinatorial Optimization (CO)문제에 신경망을 적용
- N<50 인 TSP 문제에서 최적에 준하는 성능



Pointer Networks [2015, NIPS]

Background

❖ Seq2Seq 과 Ptr-Net



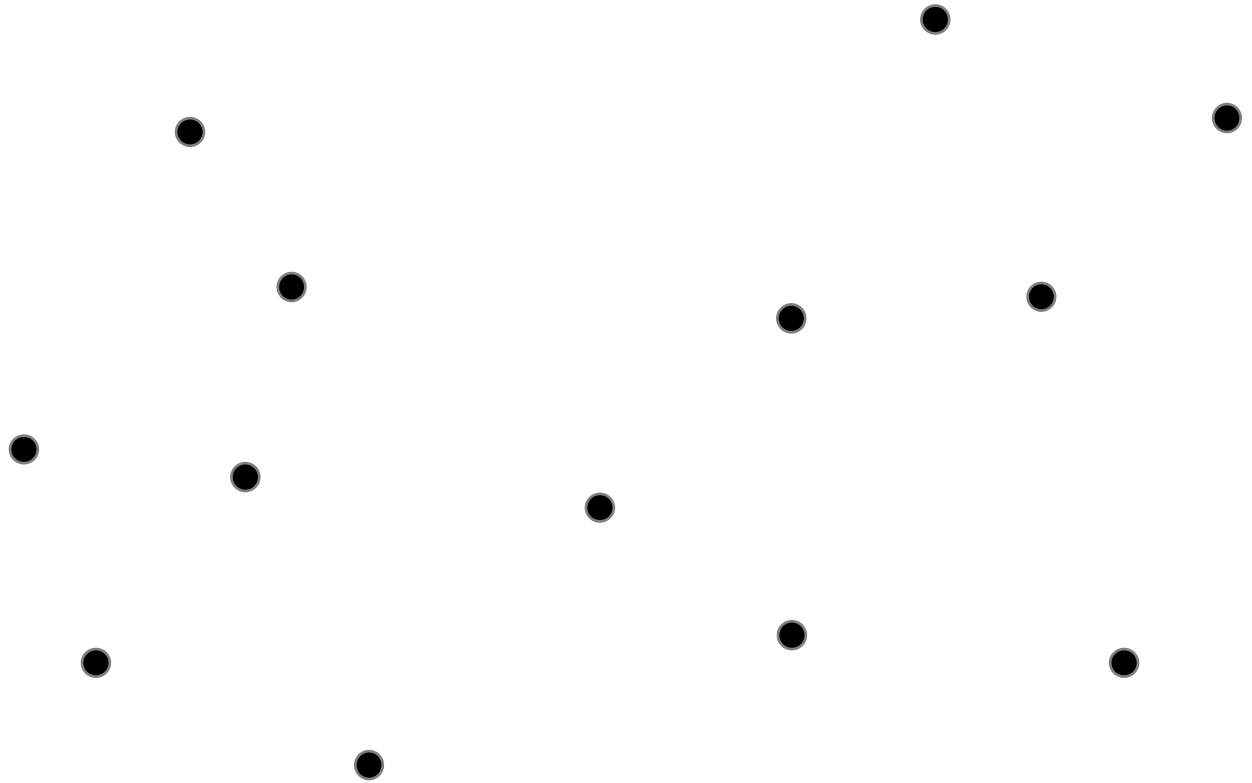
(a) Sequence-to-Sequence

(b) Ptr-Net

Pointer Networks [2015, NIPS]

Background

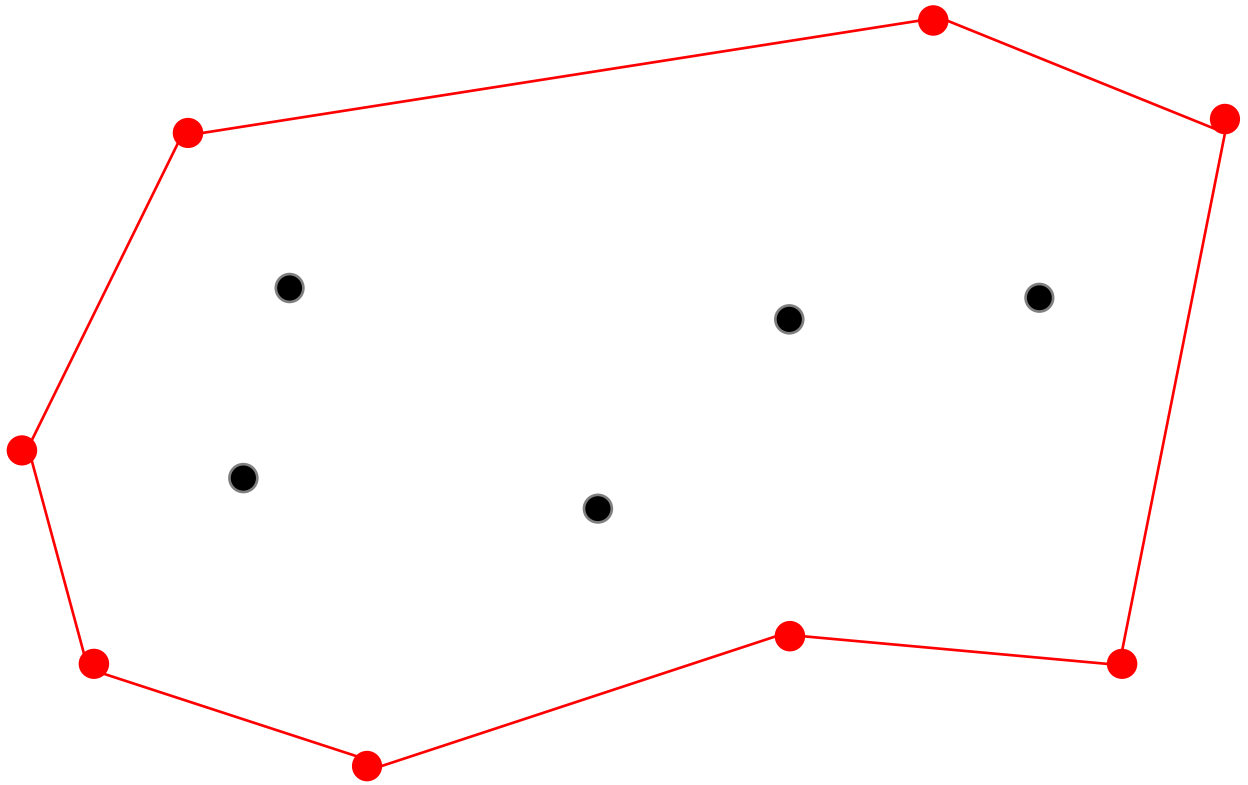
- **Convex hull problem**
  - 최외각 점 탐색문제



Pointer Networks [2015, NIPS]

Background

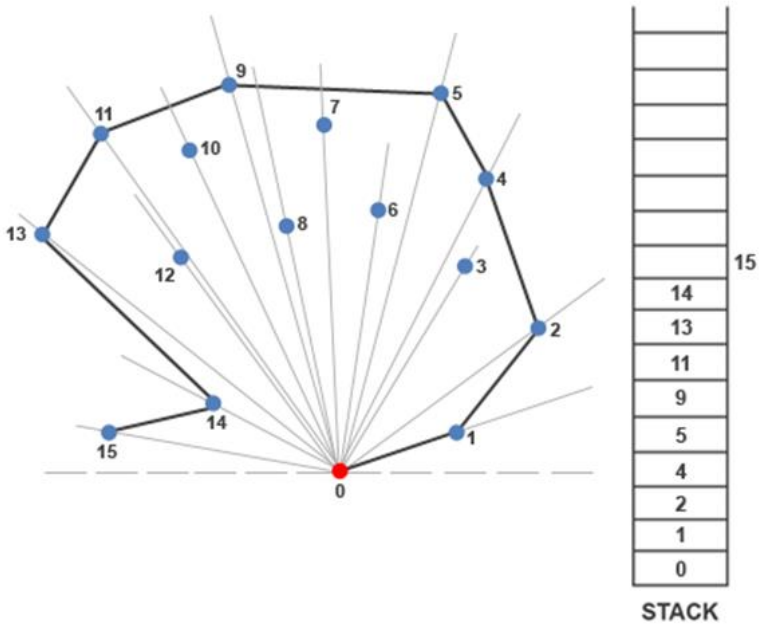
- **Convex hull problem**
  - 빨간 점들이 최외각 점

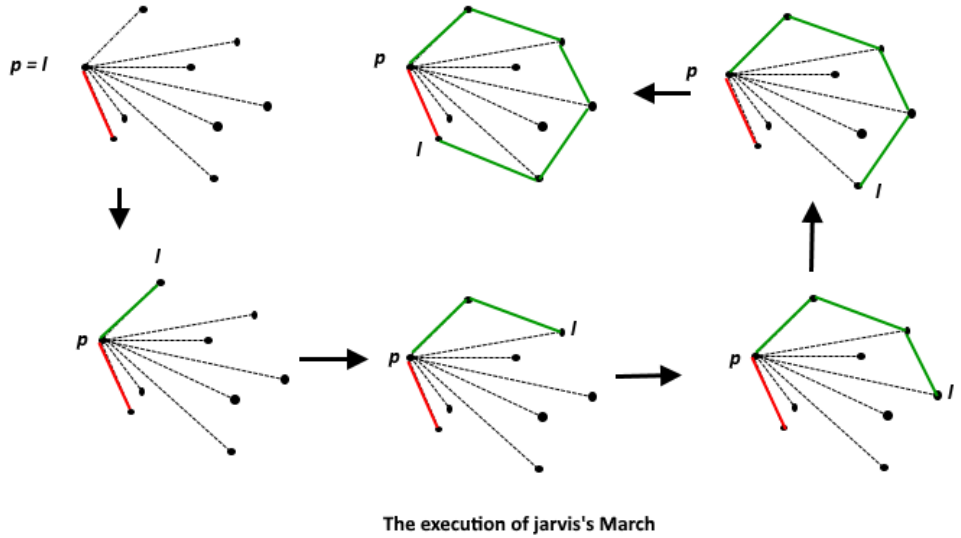


Pointer Networks [2015, NIPS]

Background

- **Algorithm for solving convex hull problem**
  - Graham's Scan : $O(nlogn)$
  - Javis's March : $O(n^2)$





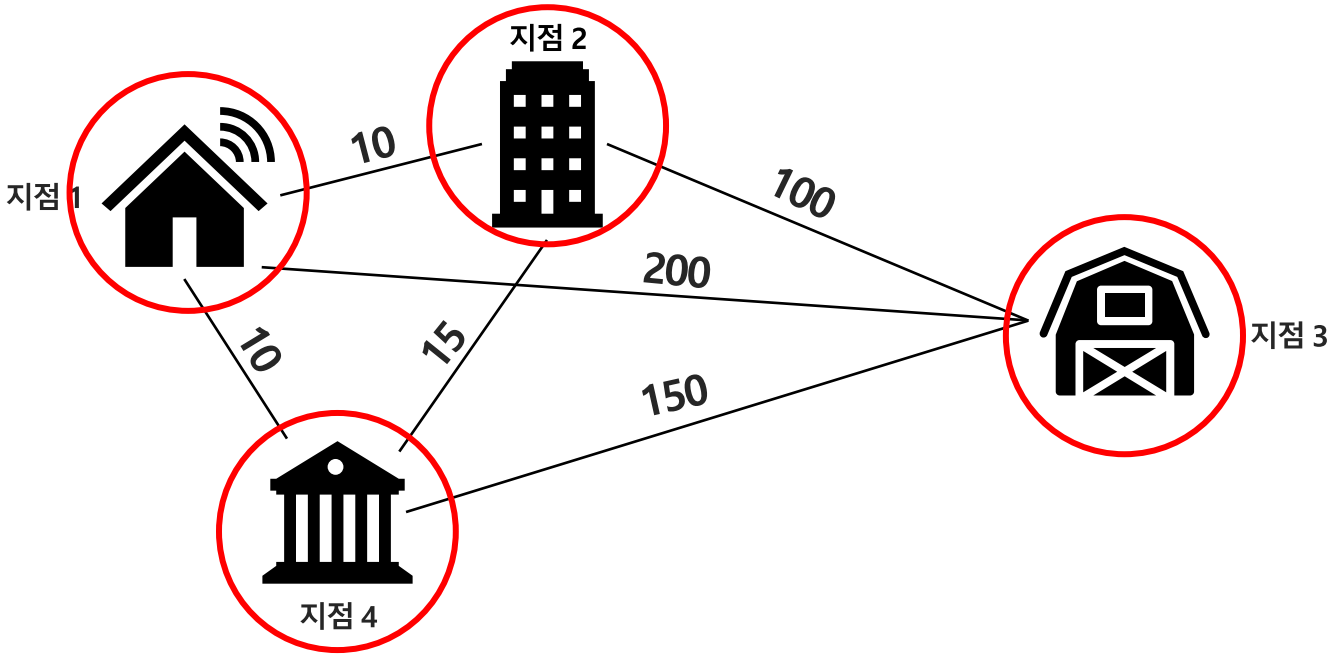
The execution of jarvis's March

Pointer Networks [2015, NIPS]

Background

❖ **TSP (Traveling Salesman Problem)**

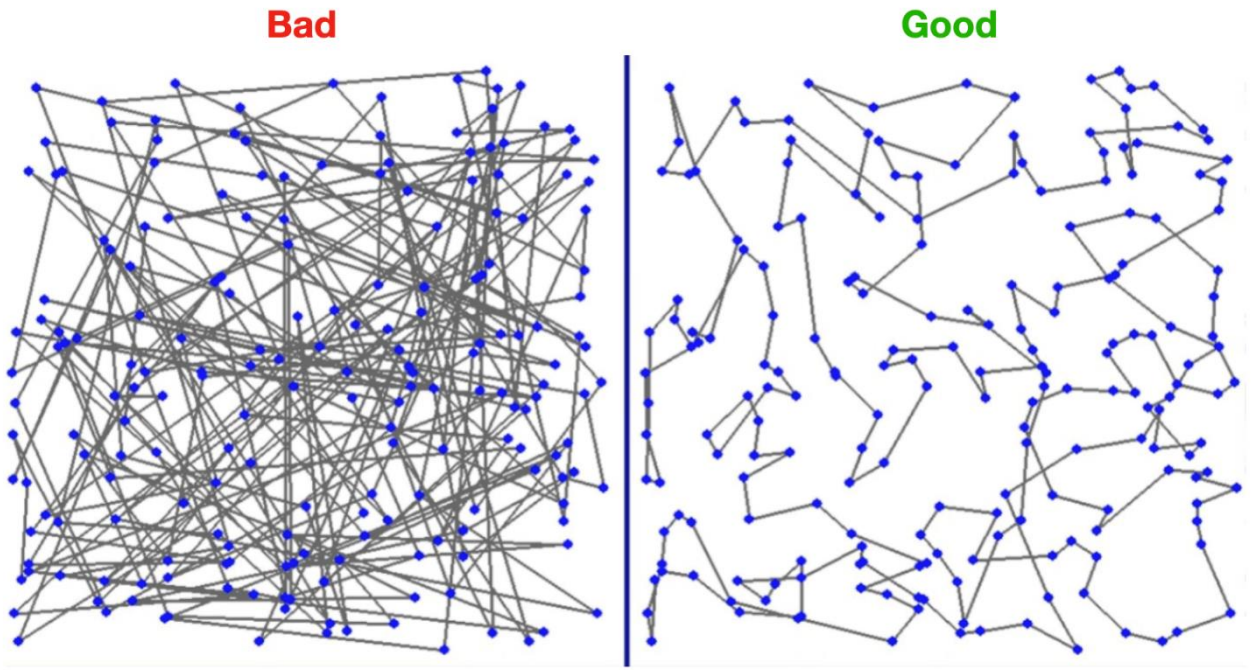
- 방문해야 하는 지점들과 각 지점들 사이의 거리가 주어졌을 때,
  주어진 지점들을 모두 한 번씩 방문하는 최소 길이의 cycle (Optimal Cycle)을 찾는 문제



Pointer Networks [2015, NIPS]

Background

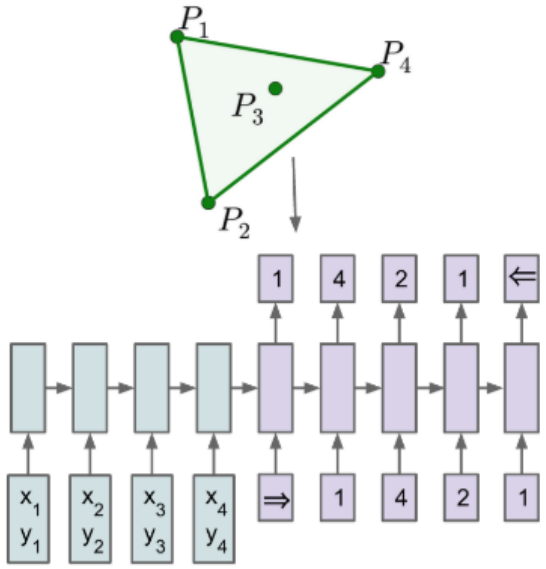
- ❖ **Algorithm for solving TSP**
  - Brute Force : $O(n!)$
  - Held-Karp Algorithm : $O(n^2 \cdot 2^n)$
  - Nearest Neighbor Algorithm (근사치) : $O(n^2)$
- ❖ **노드 개수가 20~30개 넘어가면 최적 해를 구하기 불가능**



Pointer Networks [2015, NIPS]

Background

- **Seq2seq**
  - 인코더 : 점들의 나열을 입력으로 점들의 정보를 임베딩
  - 디코더 : 학습한 정보로 최외각 점들의 인덱스 시퀀스를 출력

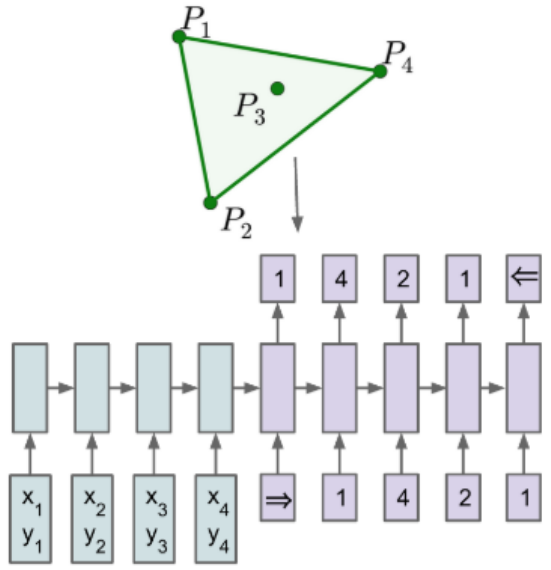


(a) Sequence-to-Sequence

Pointer Networks [2015, NIPS]

Background

- **Seq2seq**
  - 점이 4개가 아닌 5개라면?
    - 디코더의 Output Dictionary 가 고정되어 현재 구조로는 학습할 수 없음
  - 디코더의 사전 크기를 늘리면?
    - 인코더가 포착해야할 시퀀스가 길어짐
      - 장거리 관계 포착 어려워짐
    - 범위를 벗어난 점을 출력할 가능성

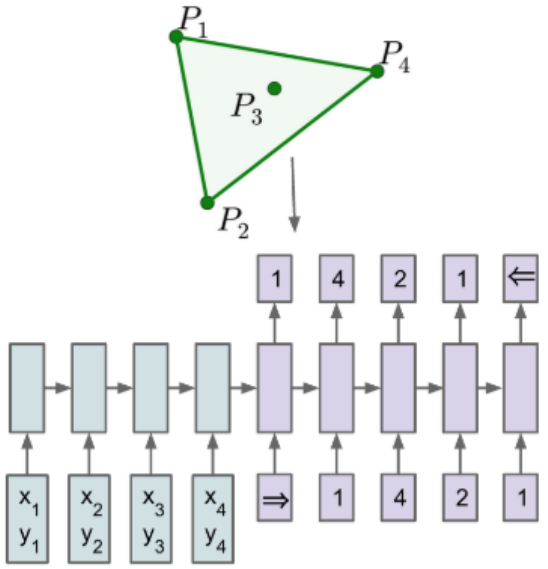


(a) Sequence-to-Sequence

Pointer Networks [2015, NIPS]

Background

- **Seq2seq**
  - 출력이 입력에 의존하는 문제에 적합하지 않음
  - Attention Mechanism을 도입하여 장거리 포착문제를 해결 가능
    - Output Dictionary 고정 문제는 해결 불가

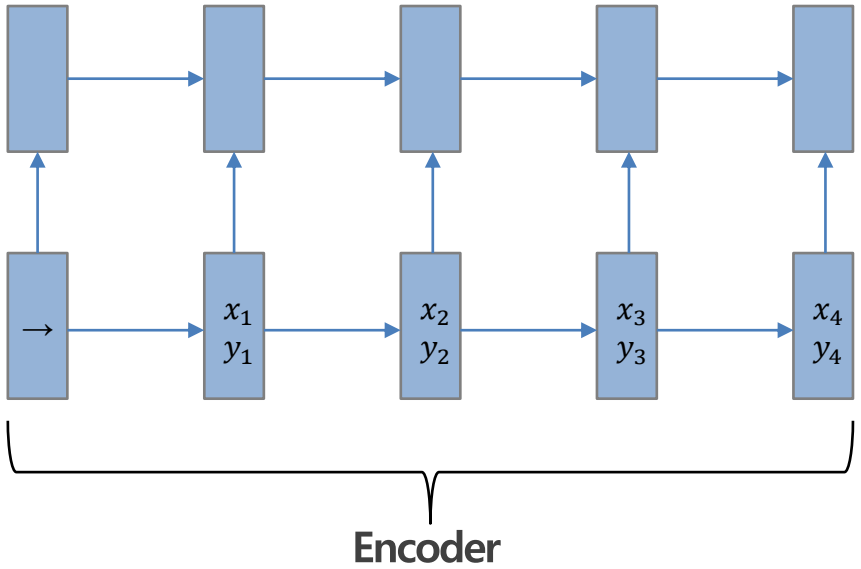


(a) Sequence-to-Sequence

Pointer Networks [2015, NIPS]

Method

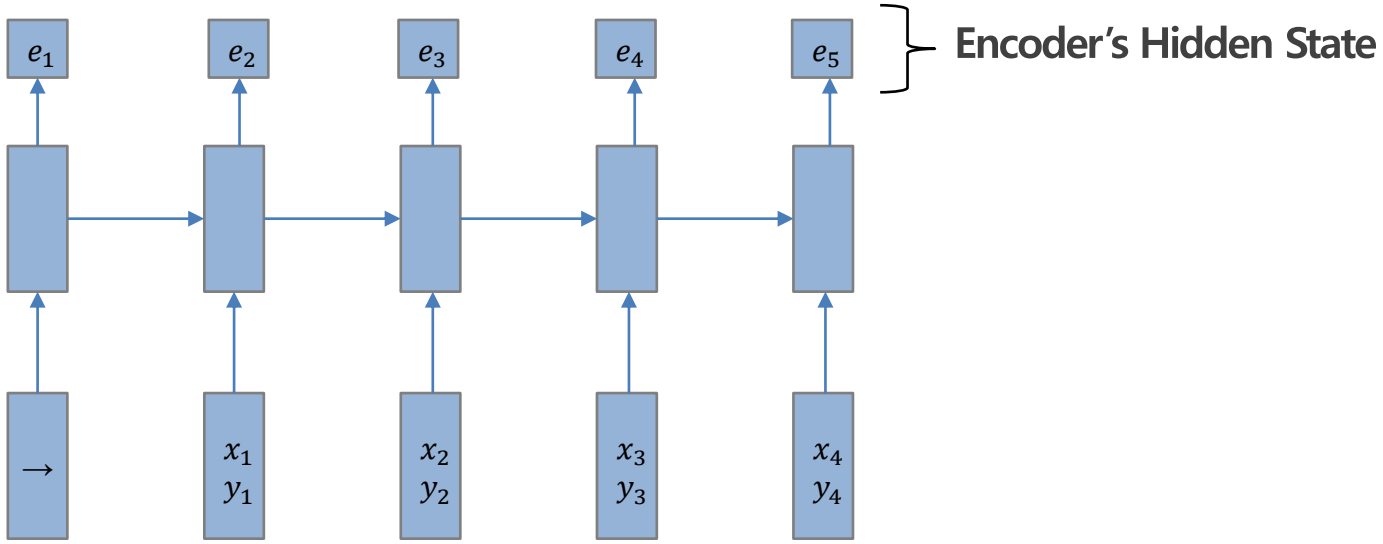
- Seq2seq



Pointer Networks [2015, NIPS]

Method

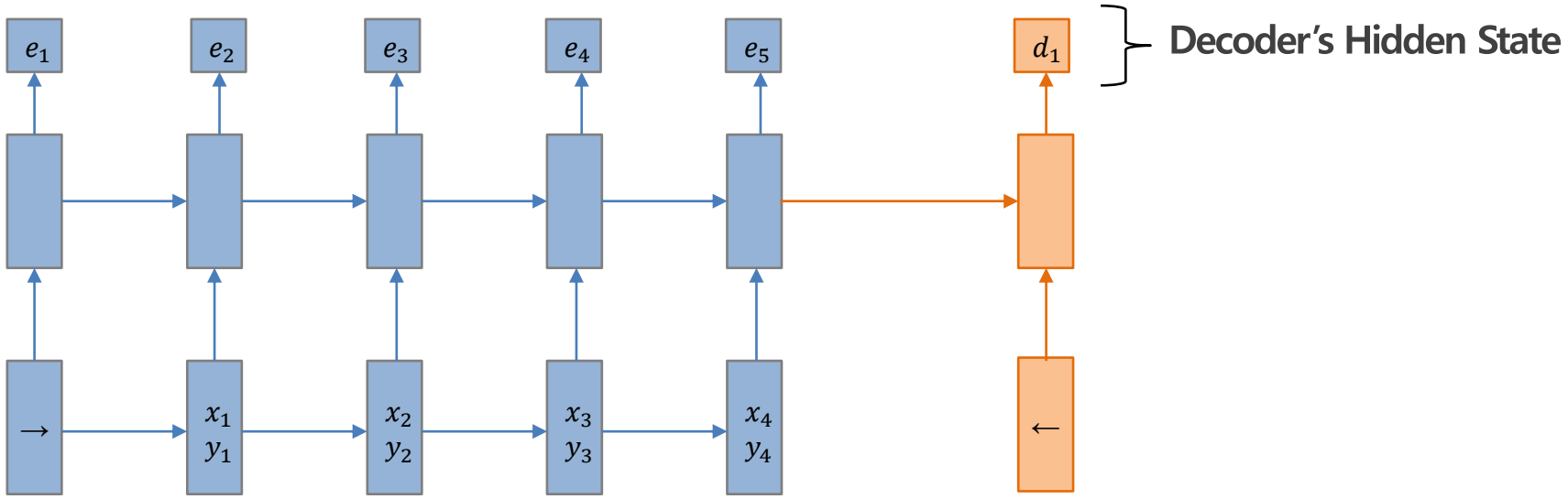
- Seq2seq



Pointer Networks [2015, NIPS]

Method

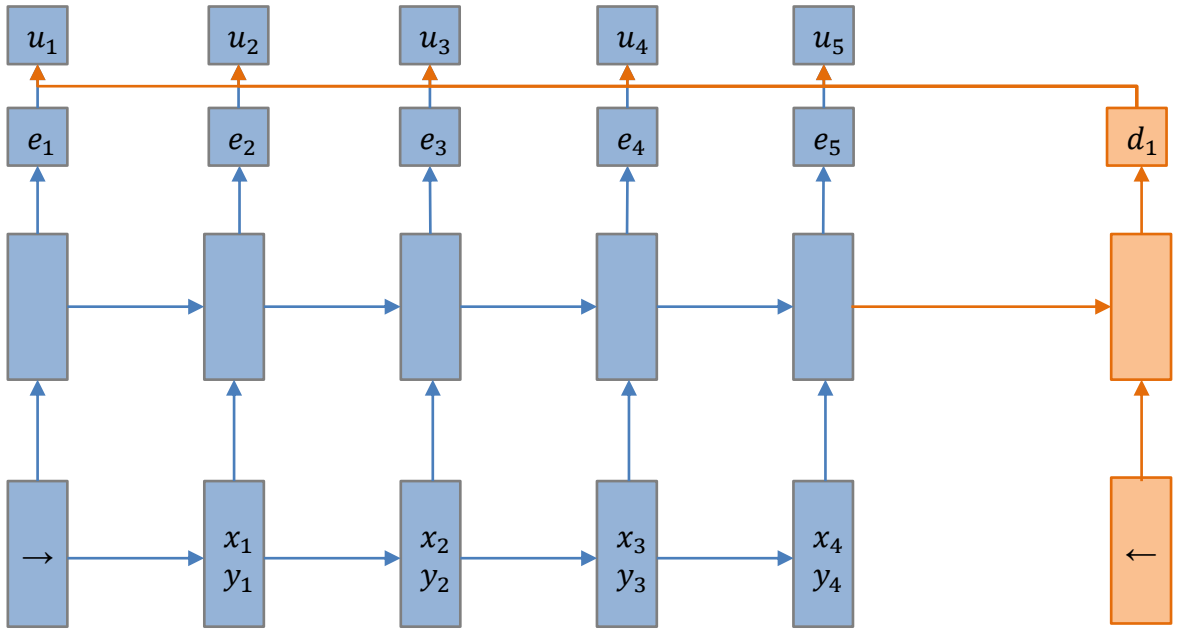
- **Seq2seq**
  - 입력이 길어질 수록 장거리 관계 포착이 어려움
  - 가변 길이의 입력을 처리할 수 없음



Pointer Networks [2015, NIPS]

Method

- **Seq2seq using Attention Mechanism**
  - Compute the attention vector at each output time i



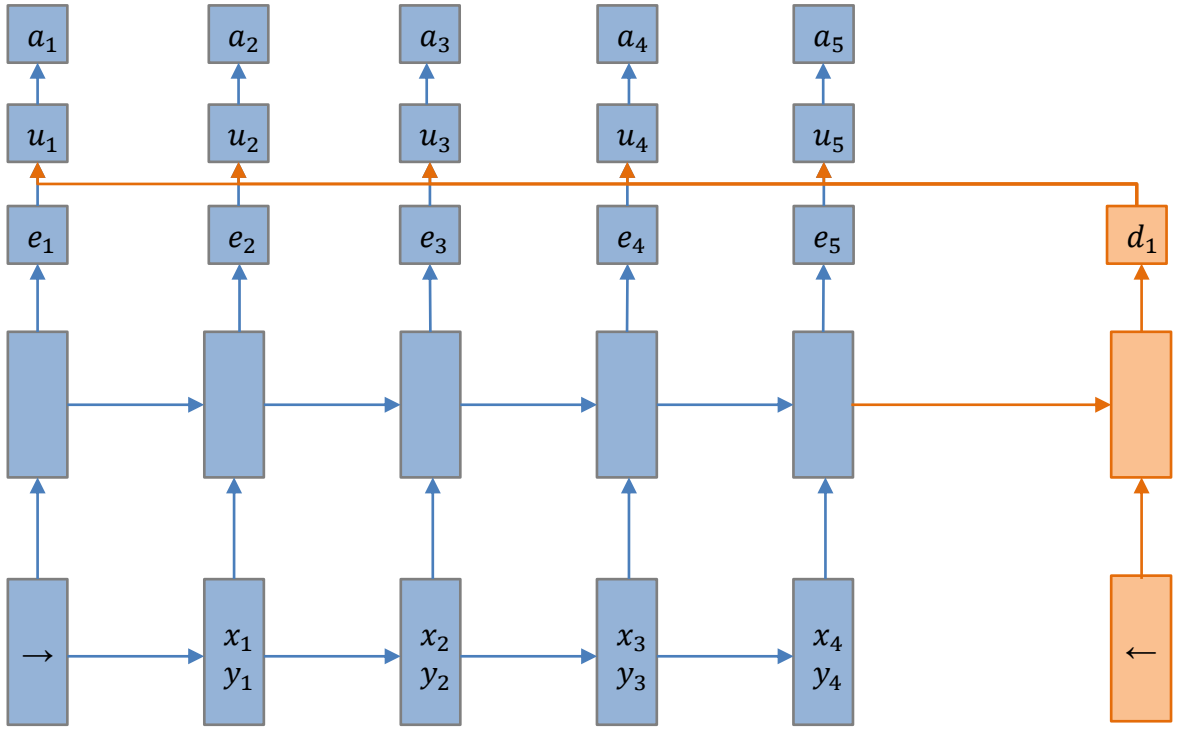
$$u_j^i = v^T \tanh(W_1 e_j + W_2 d_i) \quad j \in (1, \ldots, n)$$

Learnable parameters

Pointer Networks [2015, NIPS]

Method

- **Seq2seq using Attention Mechanism**
  - Normalizes the attention vector u_i with softmax function



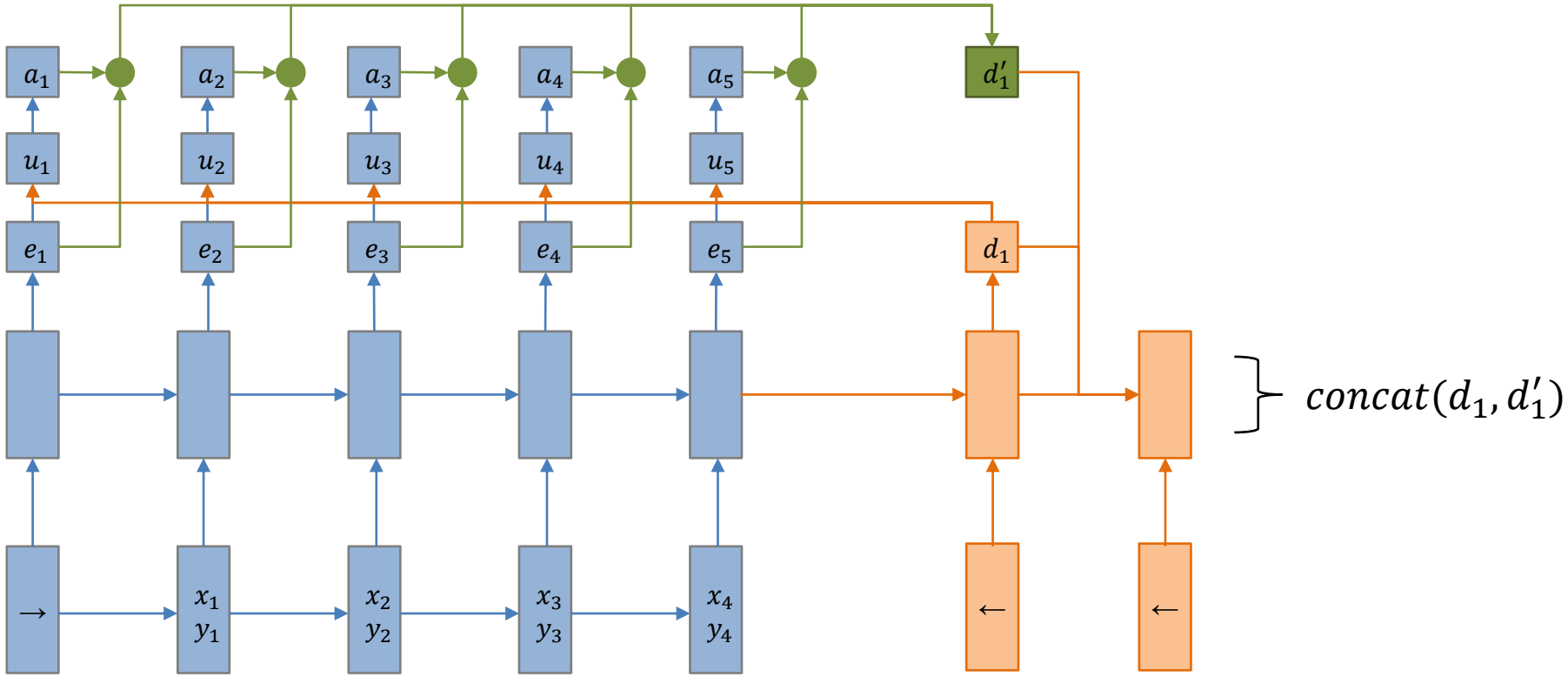
$$a_j^i = \text{softmax}(u_j^i) \qquad j \in (1, \ldots, n)$$

Pointer Networks [2015, NIPS]

Method

❖ **Seq2seq using Attention Mechanism**

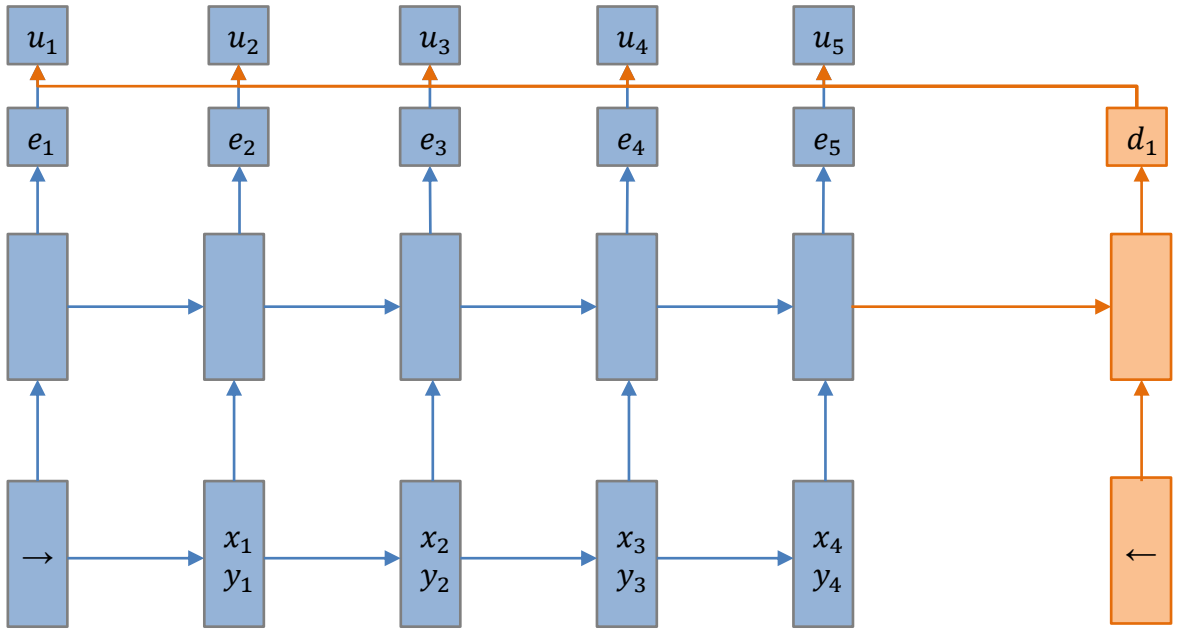
- Attention Scoring 을 통해 장거리 의존 관계를 해결했으나, Output Dictionary 때문에 가변 길이 입력 대응 불가능



Pointer Networks [2015, NIPS]

Method

- **Pointer Network**
  - Compute the attention vector



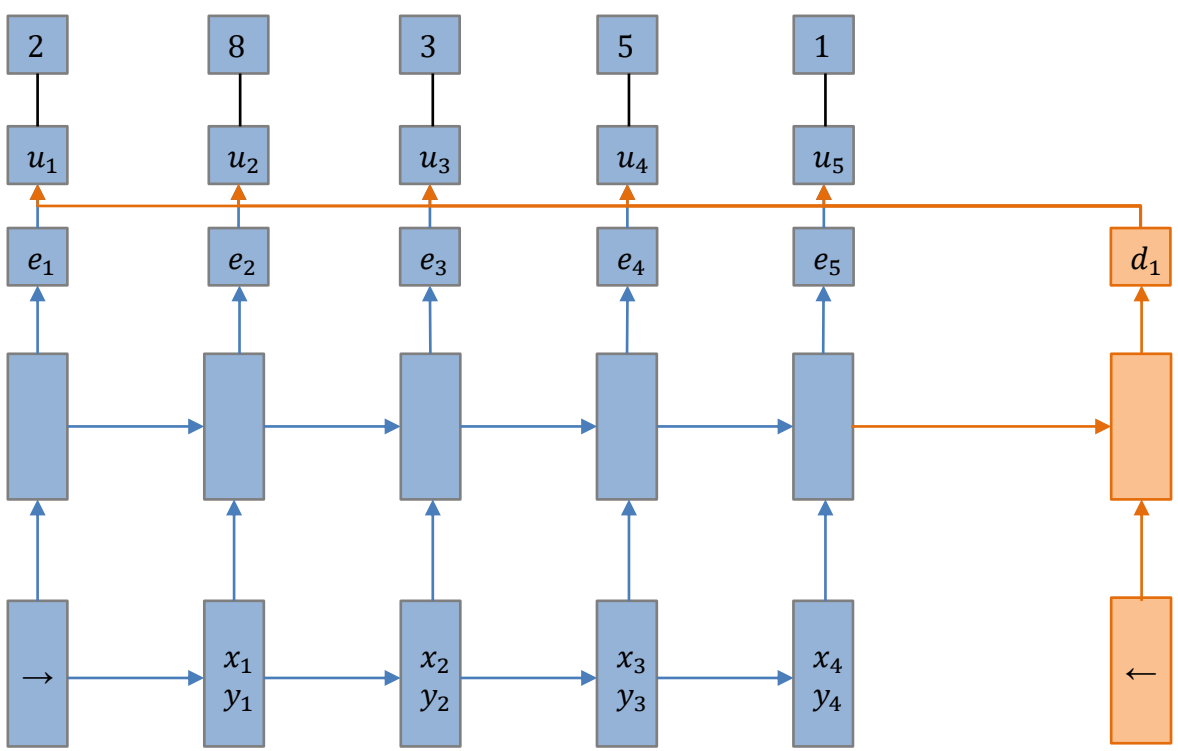
$$u_j^i = v^T \tanh(W_1 e_j + W_2 d_i) \quad j \in (1, \ldots, n)$$

Learnable parameters

Pointer Networks [2015, NIPS]

Method

❖ **Pointer Network**

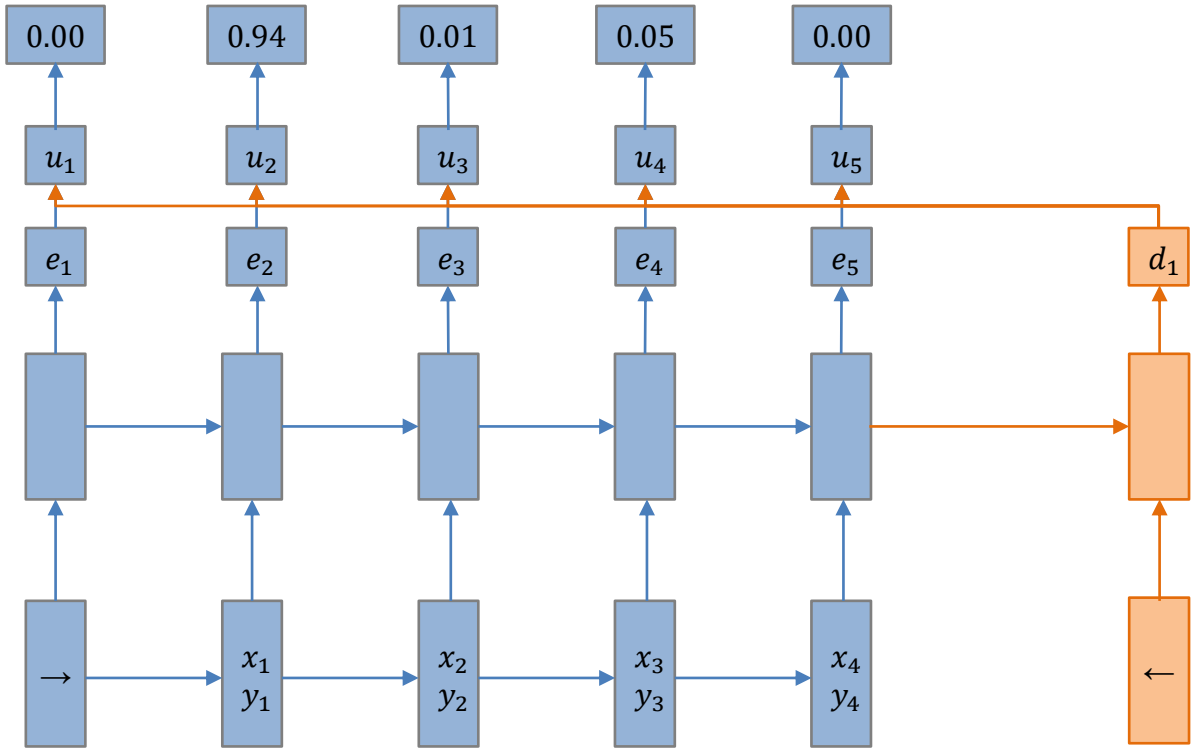


Example value of u_j

Pointer Networks [2015, NIPS]

Method

❖ **Pointer Network**

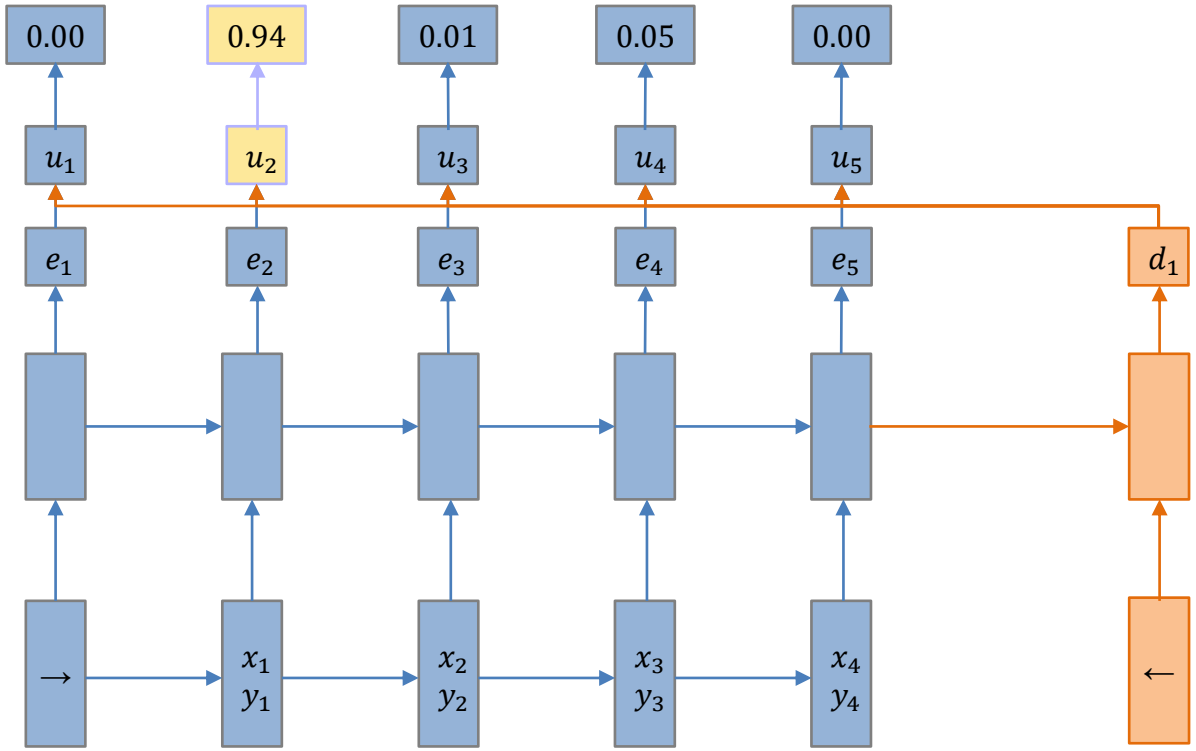


$$a_j^i = \text{softmax}(u_j^i) \qquad j \in (1, \ldots, n)$$

Pointer Networks [2015, NIPS]

Method

❖ **Pointer Network**



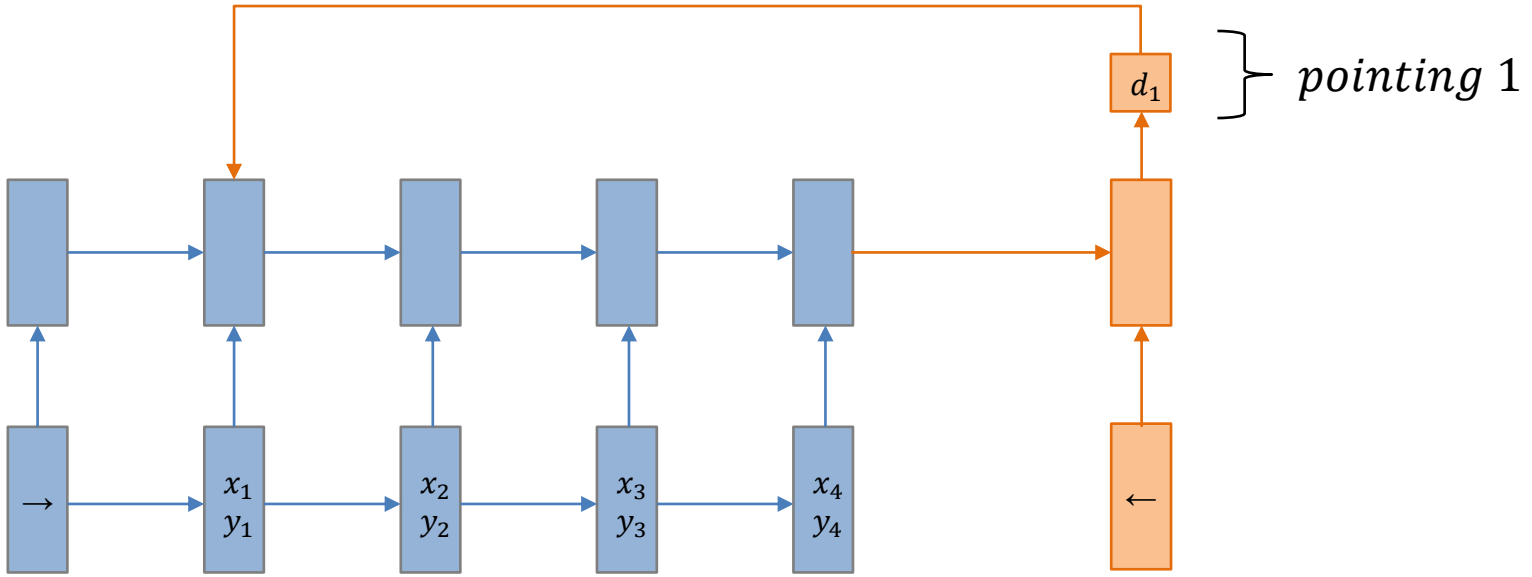
$$argmax(u^1) = 1$$

Pointer Networks [2015, NIPS]

Background

❖ **Pointer Network**

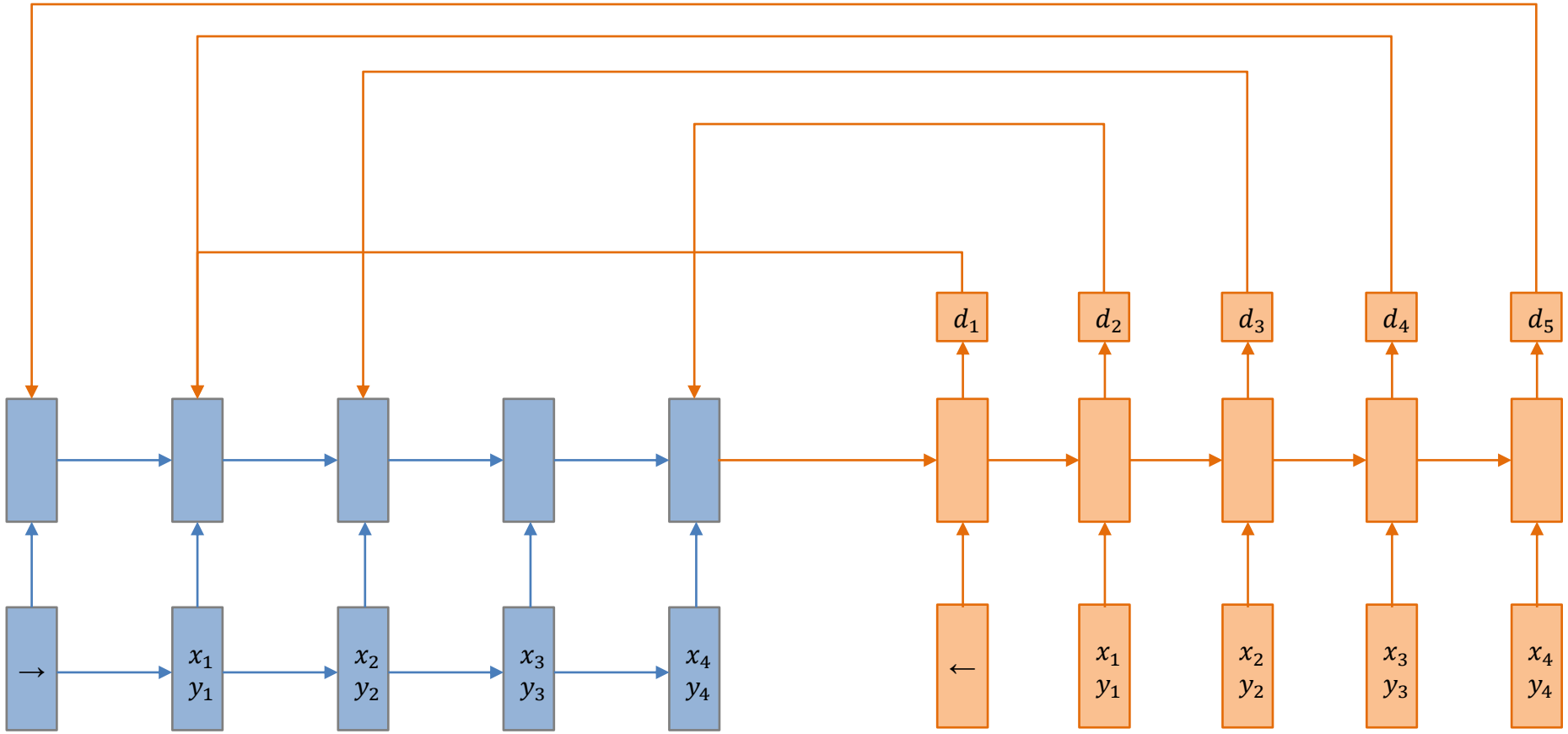
- Seq2seq 과 다르게 Input element를 가리키도록 함



Pointer Networks [2015, NIPS]

Background

❖ Pointer Network



Pointer Networks [2015, NIPS]

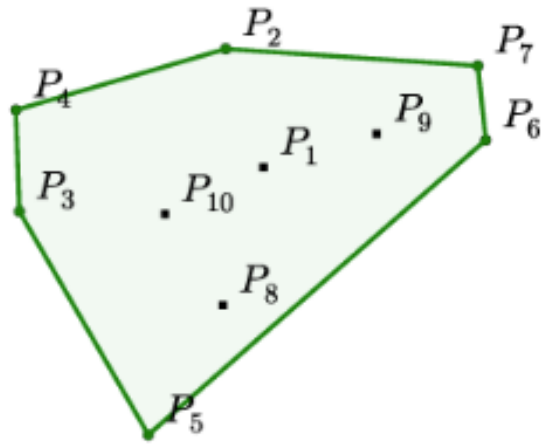
Experiments

- **Convex Hull**
  - 최외곽 점을 탐색하는 문제
  - $O(nlogn)$
- **Delaunay Triangulation (들로네 삼각분할)**
  - 평면위의 점들을 삼각형으로 공간을 분할
  - 제약 : 삼각형의 외접원이 세 꼭지점을 제외한 어떤 점을 포함하지 않음
  - $O(nlogn)$
- **TSP**
  - 평면위의 점들 간의 비용이 존재할 때, 각 점을 한번씩만 방문하고 시작점으로 돌아오는 최소 비용의 순서 탐색
  - Optimal solution : $O(n!)$
  - Approximated Solution : $O(n^2)$

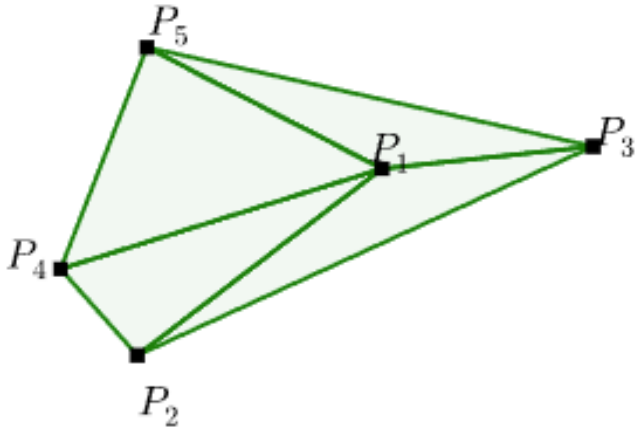
Pointer Networks [2015, NIPS]

Experiments

❖ **Example**



(a) Input $\mathcal{P} = \{P_1, \ldots, P_{10}\}$, and the output sequence $\mathcal{C}^{\mathcal{P}} = \{\Rightarrow, 2, 4, 3, 5, 6, 7, 2, \Leftarrow\}$ representing its convex hull.



(b) Input $\mathcal{P} = \{P_1, \ldots, P_5\}$, and the output $\mathcal{C}^{\mathcal{P}} = \{\Rightarrow, (1, 2, 4), (1, 4, 5), (1, 3, 5), (1, 2, 3), \Leftarrow\}$ representing its Delaunay Triangulation.

Pointer Networks [2015, NIPS]

Experiments

❖ Model Architecture & Hyper parameter

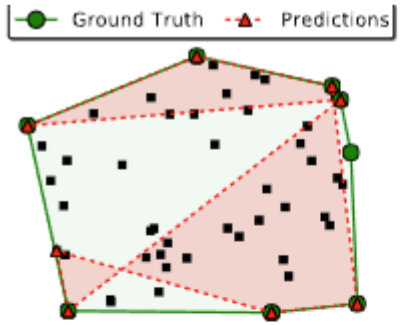
- 모델 일관성 유지를 위해 모든 문제에 대해 같은 아키텍처와 하이퍼 파라미터 사용

하이퍼파라미터	값
모델 아키텍처	
은닉 유닛 수	256 또는 512
LSTM 레이어 수	1
학습 하이퍼파라미터	
학습률	1.0
배치 크기	128
가중치 초기화	균등 분포 (-0.08 ~ 0.08)
L2 그라디언트 클리핑	2.0
학습 데이터	1,000,000 예제 쌍
학습 에포크 수	10 ~ 20

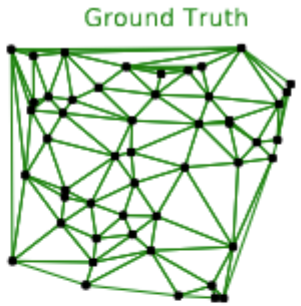
Pointer Networks [2015, NIPS]

Experiments

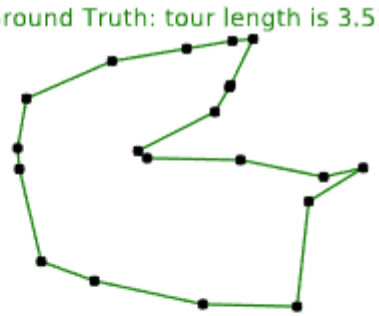
❖ **Result**



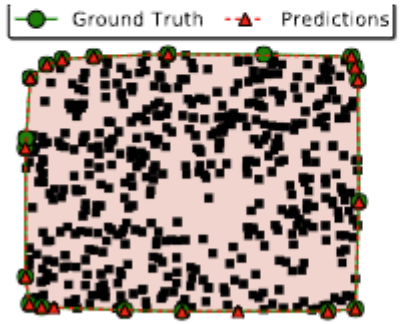
(a) LSTM, m=50, n=50



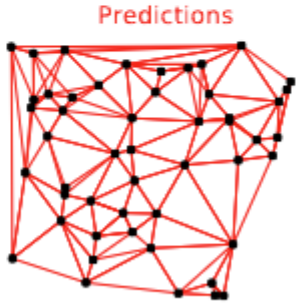
(b) Truth, n=50



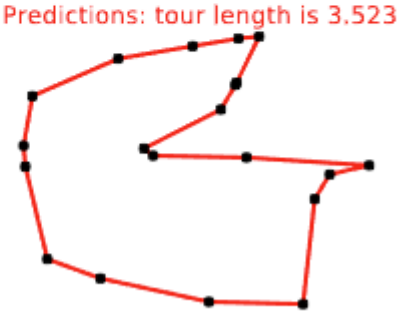
(c) Truth, n=20



(d) Ptr-Net, m=5-50, n=500



(e) Ptr-Net , m=50, n=50



(f) Ptr-Net , m=5-20, n=20

Pointer Networks [2015, NIPS]

Experiments

❖ **Result : Convex Hull**

- Accuracy : 생성된 Convex Hull point 와 정답 Convex Hull point 가 얼마나 동일한지
- Area : 생성된 Convex Hull 면적과 정답 Convex Hull 면적이 얼마나 일치하는지

METHOD	TRAINED n	n	ACCURACY	AREA
LSTM [1]	50	50	1.9%	FAIL
+ATTENTION [5]	50	50	38.9%	99.7%
PTR-NET	50	50	72.6%	99.9%
LSTM [1]	5	5	87.7%	99.6%
PTR-NET	5-50	5	92.0%	99.6%
LSTM [1]	10	10	29.9%	FAIL
PTR-NET	5-50	10	87.0%	99.8%
PTR-NET	5-50	50	69.6%	99.9%
PTR-NET	5-50	100	50.3%	99.9%
PTR-NET	5-50	200	22.1%	99.9%
PTR-NET	5-50	500	1.3%	99.2%

Pointer Networks [2015, NIPS]

Experiments

❖ **Result : Convex Hull**

- Input : [0,1] x [0,1] 범위에서 균등 분포로 샘플링된 점 집합
- Output : Convex Hull 을 형성하는 점의 인덱스 순서

METHOD	TRAINED n	n	ACCURACY	AREA
LSTM [1]	50	50	1.9%	FAIL
+ATTENTION [5]	50	50	38.9%	99.7%
PTR-NET	50	50	72.6%	99.9%
LSTM [1]	5	5	87.7%	99.6%
PTR-NET	5-50	5	92.0%	99.6%
LSTM [1]	10	10	29.9%	FAIL
PTR-NET	5-50	10	87.0%	99.8%
PTR-NET	5-50	50	69.6%	99.9%
PTR-NET	5-50	100	50.3%	99.9%
PTR-NET	5-50	200	22.1%	99.9%
PTR-NET	5-50	500	1.3%	99.2%

Pointer Networks [2015, NIPS]

Experiments

❖ **Result : Convex Hull**

- Ptr-Net은 LSTM, LSTM+Attention 대비 높은 정확도와 면적 커버리지를 보임
- 다양한 길이의 입력에 대해 높은 일반화 성능을 보이며, 학습 범위를 초과하는 경우에도 어느정도 성능을 보임

METHOD	TRAINED n	n	ACCURACY	AREA
LSTM [1]	50	50	1.9%	FAIL
+ATTENTION [5]	50	50	38.9%	99.7%
PTR-NET	50	50	72.6%	99.9%
LSTM [1]	5	5	87.7%	99.6%
PTR-NET	5-50	5	92.0%	99.6%
LSTM [1]	10	10	29.9%	FAIL
PTR-NET	5-50	10	87.0%	99.8%
PTR-NET	5-50	50	69.6%	99.9%
PTR-NET	5-50	100	50.3%	99.9%
PTR-NET	5-50	200	22.1%	99.9%
PTR-NET	5-50	500	1.3%	99.2%

Pointer Networks [2015, NIPS]

Experiments

❖ **Result : Delaunay Triangulation**

- Input : Convex Hull 실험과 동일한 샘플링 된 점 집합
- Output : Delaunay 삼각형의 인덱스 집합

테스트 n	정확도	삼각형 커버리지
5	80.7%	93.0%
10	22.6%	81.3%
50	0%	52.8%

Pointer Networks [2015, NIPS]

Experiments

❖ **Result : Delaunay Triangulation**

- 정확도 : 정답과 동일한 삼각형을 생성한 비율
- 삼각형 커버리지 : 모델이 예측한 삼각형이 실제 삼각형과 일치하는 비율

테스트 n	정확도	삼각형 커버리지
5	80.7%	93.0%
10	22.6%	81.3%
50	0%	52.8%

Pointer Networks [2015, NIPS]

Experiments

❖ **Result : Delaunay Triangulation**

- 작은 n에 대해 높은 정확도와 삼각형 커버리지를 보였지만, n이 증가할수록 성능이 저하

테스트 n	정확도	삼각형 커버리지
5	80.7%	93.0%
10	22.6%	81.3%
50	0%	52.8%

Pointer Networks [2015, NIPS]

Experiments

❖ **Result : Travelling Salesman Problem (TSP)**

- 입력 : [0,1] x [0,1] 범위에서 샘플링된 도시 좌표
- 출력 : 최단 경로의 도시 순서
- **결과 : 경로 길이(생성된 경로의 총 길이)**

n	OPTIMAL	A1	A2	A3	PTR-NET
5	2.12	2.18	2.12	2.12	2.12
10	2.87	3.07	2.87	2.87	2.88
50 (A1 TRAINED)	N/A	6.46	5.84	5.79	6.42
50 (A3 TRAINED)	N/A	6.46	5.84	5.79	6.09
5 (5-20 TRAINED)	2.12	2.18	2.12	2.12	2.12
10 (5-20 TRAINED)	2.87	3.07	2.87	2.87	2.87
20 (5-20 TRAINED)	3.83	4.24	3.86	3.85	3.88
25 (5-20 TRAINED)	N/A	4.71	4.27	4.24	4.30
30 (5-20 TRAINED)	N/A	5.11	4.63	4.60	4.72
40 (5-20 TRAINED)	N/A	5.82	5.27	5.23	5.91
50 (5-20 TRAINED)	N/A	6.46	5.84	5.79	7.66

Pointer Networks [2015, NIPS]

Experiments

❖ **Result : Travelling Salesman Problem (TSP)**

- Ptr-Net 모델은 작은 n에 대해 최적 경로와 매우 가까운 성능
- n=20까지는 비교적 좋은 일반화 성능을 보였으나, 그 이상에서는 성능이 저하
- 학습 데이터로 사용된 알고리즘(A1, A3)에 따라 Ptr-Net의 성능이 달라짐

n	OPTIMAL	A1	A2	A3	PTR-NET
5	2.12	2.18	2.12	2.12	2.12
10	2.87	3.07	2.87	2.87	2.88
50 (A1 TRAINED)	N/A	6.46	5.84	5.79	6.42
50 (A3 TRAINED)	N/A	6.46	5.84	5.79	6.09
5 (5-20 TRAINED)	2.12	2.18	2.12	2.12	2.12
10 (5-20 TRAINED)	2.87	3.07	2.87	2.87	2.87
20 (5-20 TRAINED)	3.83	4.24	3.86	3.85	3.88
25 (5-20 TRAINED)	N/A	4.71	4.27	4.24	4.30
30 (5-20 TRAINED)	N/A	5.11	4.63	4.60	4.72
40 (5-20 TRAINED)	N/A	5.82	5.27	5.23	5.91
50 (5-20 TRAINED)	N/A	6.46	5.84	5.79	7.66

Pointer Networks [2015, NIPS]

Conclusion

❖ **새로운 아키텍처 제안**

- 가변 길이의 출력 사전을 처리하는 Pointer Networks (Ptr-Nets) 제안
- 입력 시퀀스의 위치를 가리키는 attention 메커니즘 사용

❖ **세 가지 CO문제 해결**

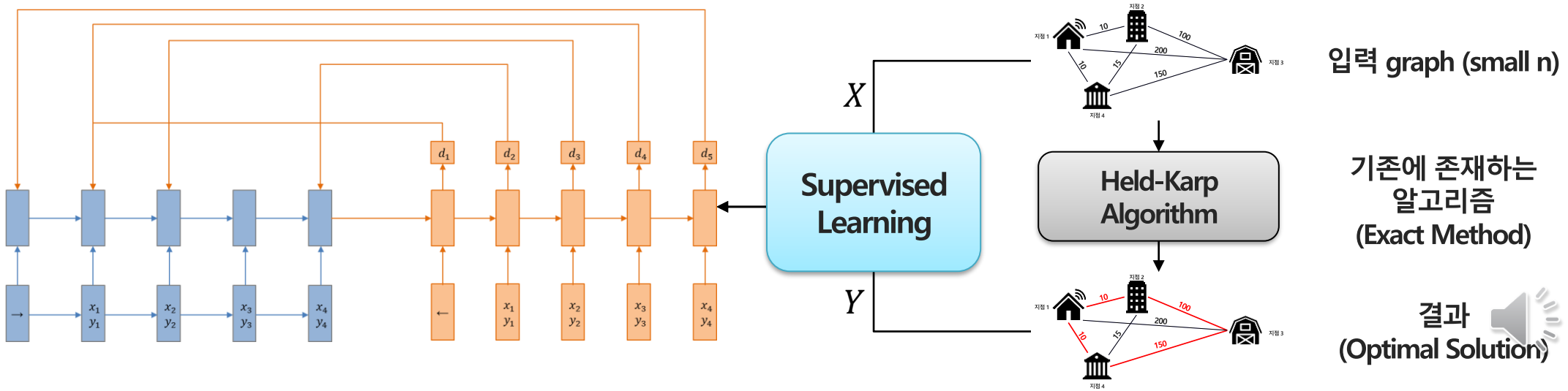
- Convex Hull, Delaunay Triangulation, Travelling Salesman Problem (TSP) 해결
- 데이터만을 사용하여 근사 솔루션 학습

Neural Combinatorial Optimization with Reinforcement Learning[2017, ICLR]

Background

❖ **Pointer Network의 한계**

- 지도학습 방식으로 네트워크 파라미터 훈련이 필요함
- 학습 시 정답이 필요함
- 따라서, 기존 존재하는 알고리즘으로 결과를 구하고 이를 정답 데이터로 제공
- Ptr-Net은 알고리즘이 제시하는 답을 기반으로 학습되어, 기존 알고리즘 보다 높은 성능 기대하기 어려움
  - 알고리즘이 입력 그래프에 대한 Optimal Solution을 구할 수 없는 경우 (예 : TSP 문제 n>30) 활용 불가

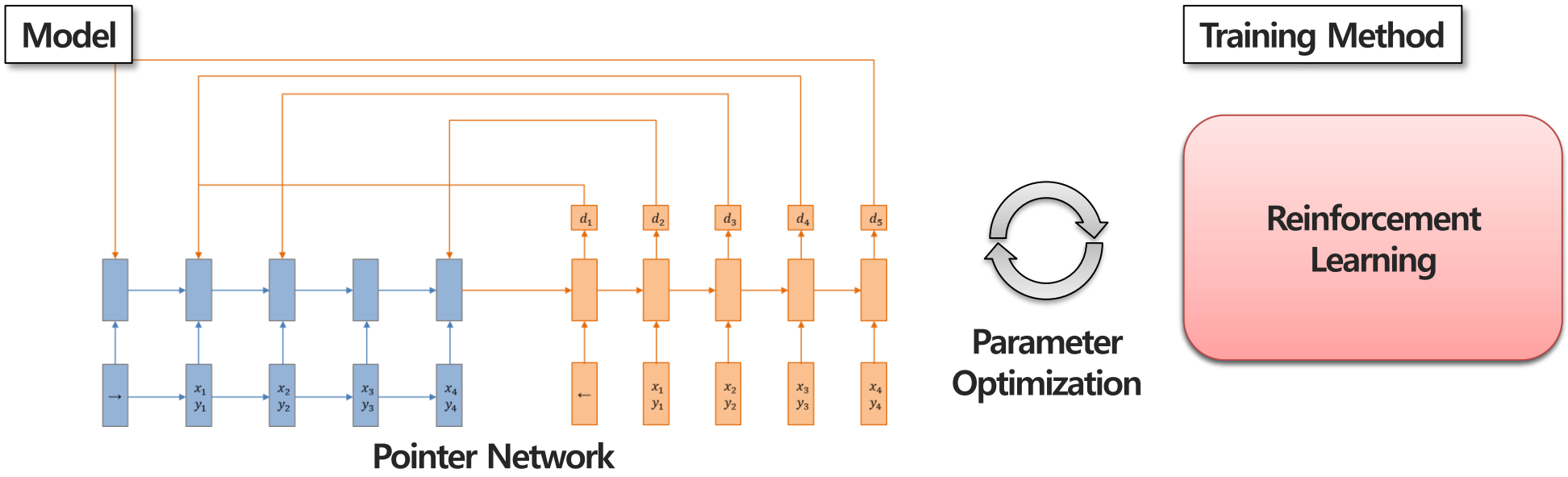


Neural Combinatorial Optimization with Reinforcement Learning [2017, ICLR]

Method

❖ Concept

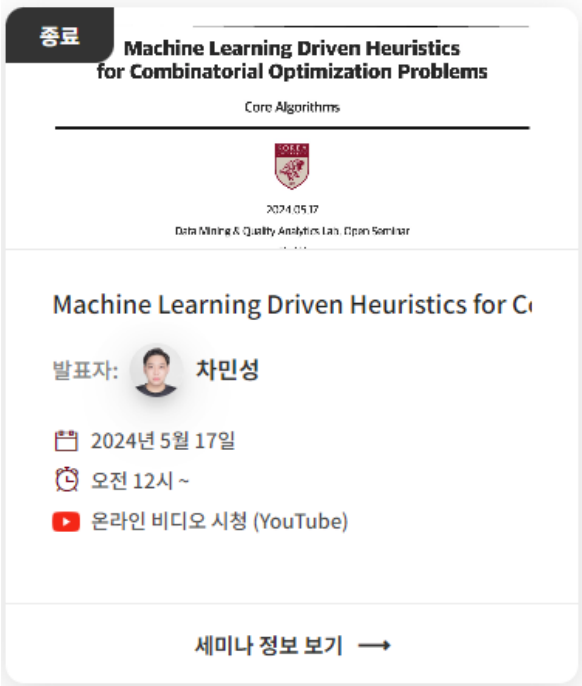
- Pointer Network (2015, NIPS) 구조 그대로 활용
- 예를들어, N=100 인 TSP 문제는 정확한 해를 구할 수 없어 지도학습 기반인 Ptr-Net을 사용 할 수 없음
- 강화학습을 적용



Neural Combinatorial Optimization 관련 세미나

❖ 연구실 Neural Combinatorial Optimization(NCO) 세미나

- Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. Advances in neural information processing systems, 28.
- Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940.
- Kool, W., Van Hoof, H., & Welling, M. (2018). Attention, learn to solve routing problems!. arXiv preprint arXiv:1803.08475.



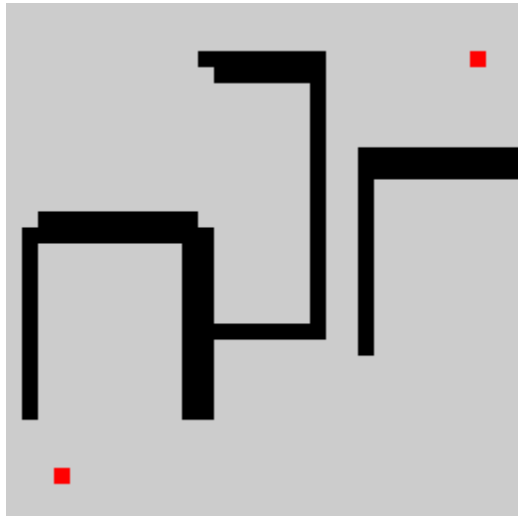
https://youtu.be/k_khfw2jA2I

Practical Applications of Neural Combinatorial Optimization

Path Planning using Neural A* Search (2021, ICML)

Introduction

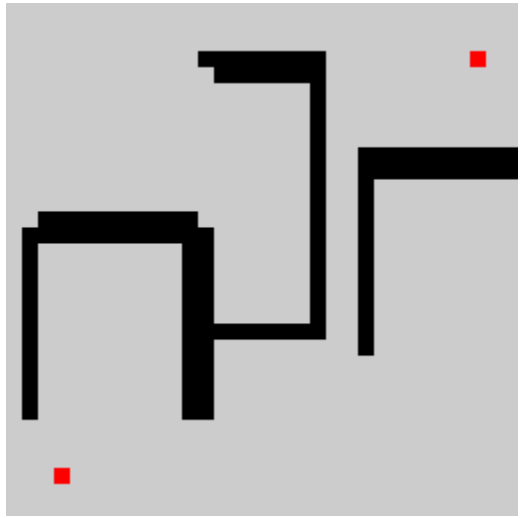
- **A* Search**
  - 주어진 출발 지점에서 목적지까지 도달할 수 있는 최적 경로를 찾는 알고리즘
  - 자동차 네비게이션 등 일상 생활 속에서 많이 쓰임
  - Heuristic 함수를 통해 탐색의 정확도와 효율을 높힘



Path Planning using Neural A* Search (2021, ICML)

Introduction

- **A* Search 예제**
  - 왼쪽 아래 빨간점 → 오른쪽 위 빨간 점 까지 도달하는 최단 경로 찾기
  - 검은색 벽은 통과할 수 없음
  - 탐색 이력(Search History)인 초록색 영역은 경로 탐색을 위해 소모한 시간 또는 비용
  - 최종 경로 : 빨간색 선



Path Planning using Neural A* Search (2021, ICML)

Introduction

❖ **이미지 기반 경로 탐색 예제**

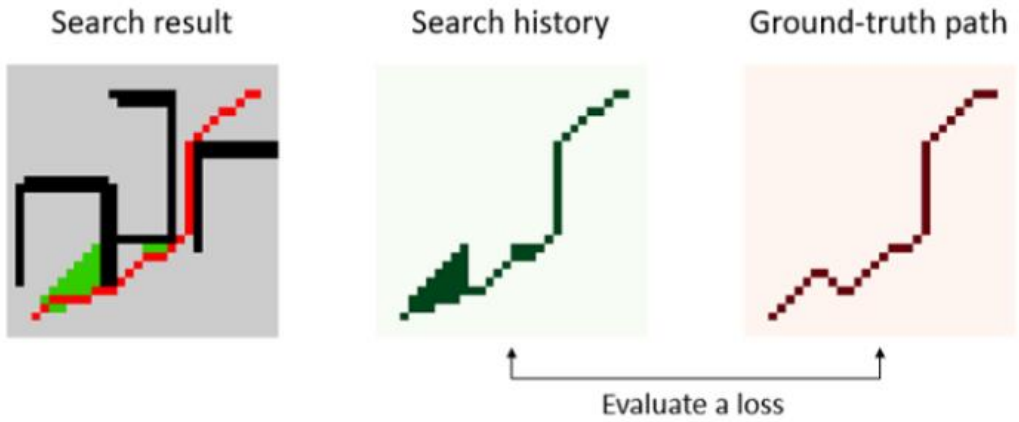
- 사람은 이미지를 보고 경로를 찾아 낼 수 있음
- 실제 사진에서 이동경로를 예측하는데 A* Search 가 동작 할 수 있는 Neural A* Search 제안

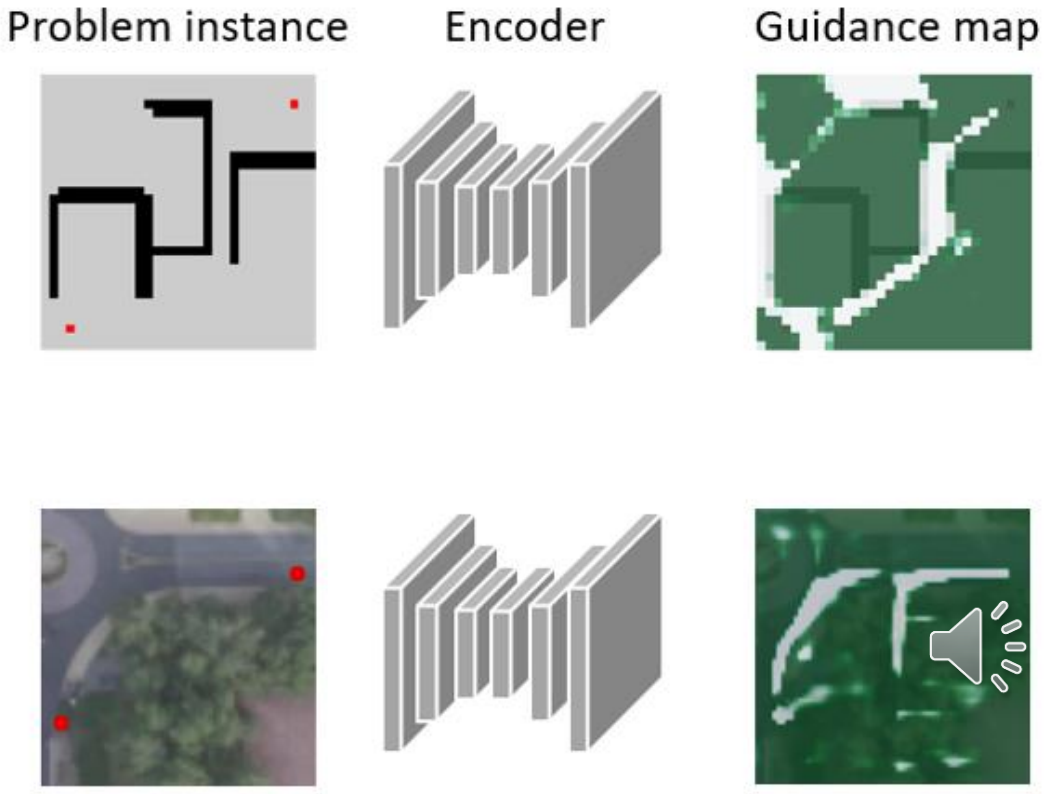
Path Planning using Neural A* Search (2021, ICML)

Method

❖ **Guidance Map**

- 경로 탐색 시 우선순위가 높은 후보 경로들에 정보를 제공
- Guidance Map과 Ground-truth 간 차이를 최소화 하는 방향으로 학습이 이루어짐
- A* Search 의 Heuristics 함수 대체



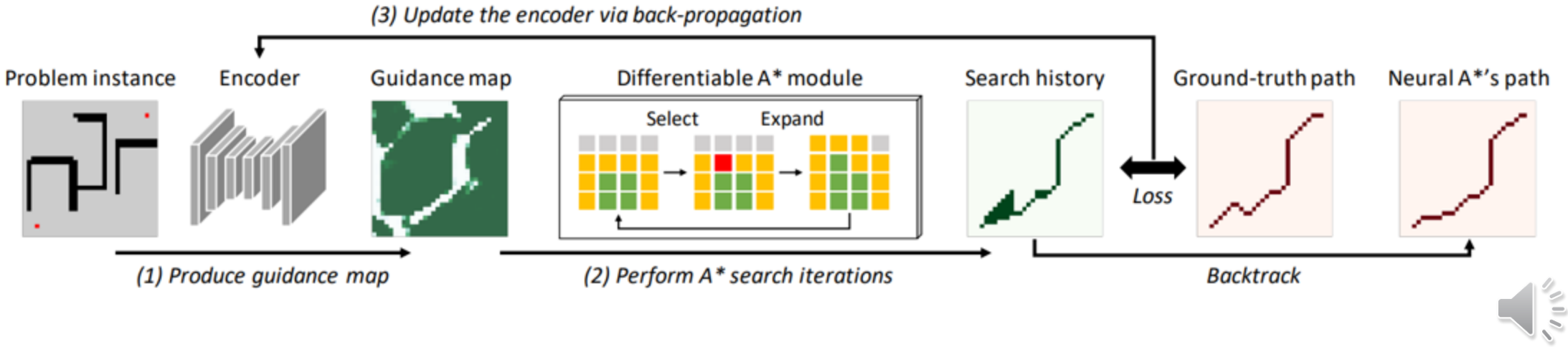


Path Planning using Neural A* Search (2021, ICML)

Method

❖ **Overall Flow of Neural A* Search**

- Input 을 Encoder 를 통해 Guidance Map (A Heuristic function) 생성
- A* 트리 탐색
- 실제 정답 경로와의 차이를 줄이는 방향으로 Encoder 네트워크 학습



Path Planning using Neural A* Search (2021, ICML)

Method

❖ 미분 가능한 A* 모듈

- A* 탐색은 이진 탐색 알고리즘으로 미분 불가함
- 그래디언트가 흐르지 않아 Encoder 네트워크 업데이트가 불가
- A* 의 연산을 행렬연산으로 치환한 미분 가능한 A* 모듈 고안

Node selection

- Finding nodes for constructing a shortest path
- **Soft-max + discretized activation**

$$V^* = \mathcal{I}_{\max}\left(\frac{\exp(-(G+H)/\tau) \odot O}{\langle \exp(-(G+H)/\tau), O \rangle}\right)$$

Total cost so far + Estimated cost to go

of nodes in the open list

Node expansion

- Adding neighboring nodes to the list of next selection candidates
- **Fixed convolution + binary masking**

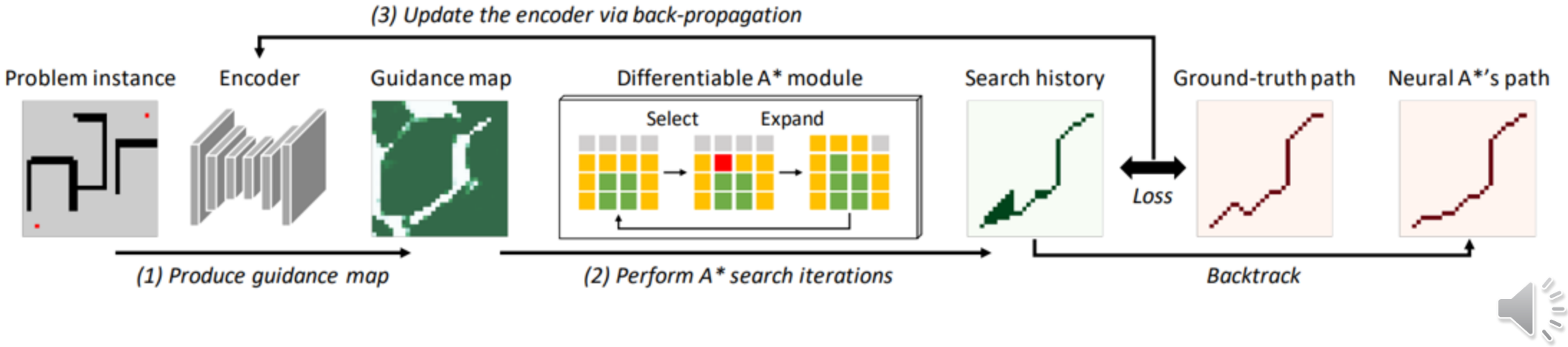
$$V_{\text{nbr}} = (V^* * K) \odot X \odot (\mathbb{1} - O) \odot (\mathbb{1} - C)$$

Neighbor Obstacle Not opend Nor selected

Path Planning using Neural A* Search (2021, ICML)

Method

- **Back-propagation**
  - 2번 A* Module 부분이 행렬기반 미분 가능한 A* 모듈로 대체 됨
  - 따라서, Encoder 까지 Gradient 가 적용 될 수 있는 구조

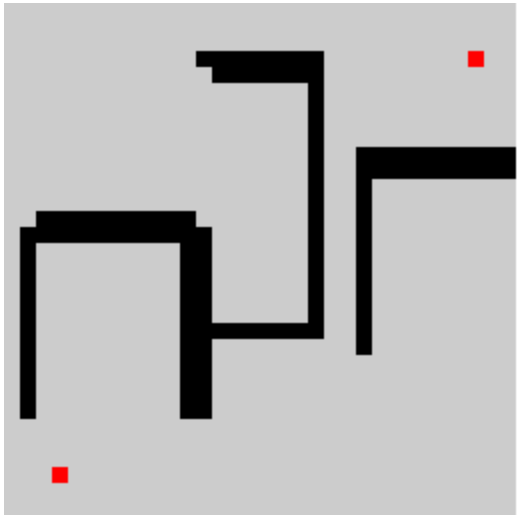


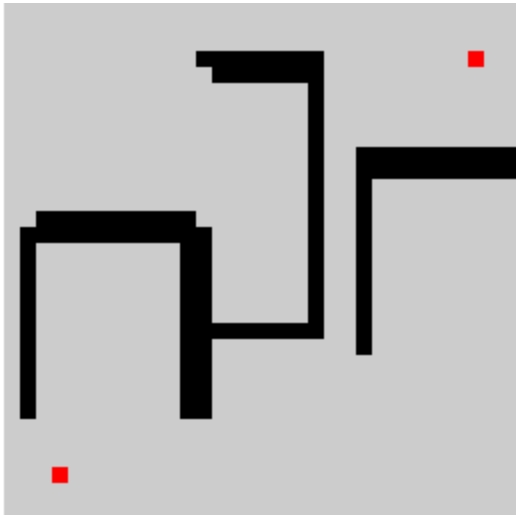
Path Planning using Neural A* Search (2021, ICML)

Experiment

❖ 정성적 분석

- Vanilla A* Search 대비 Neural A* Search가 더 빨리 최적의 경로를 탐색 가능

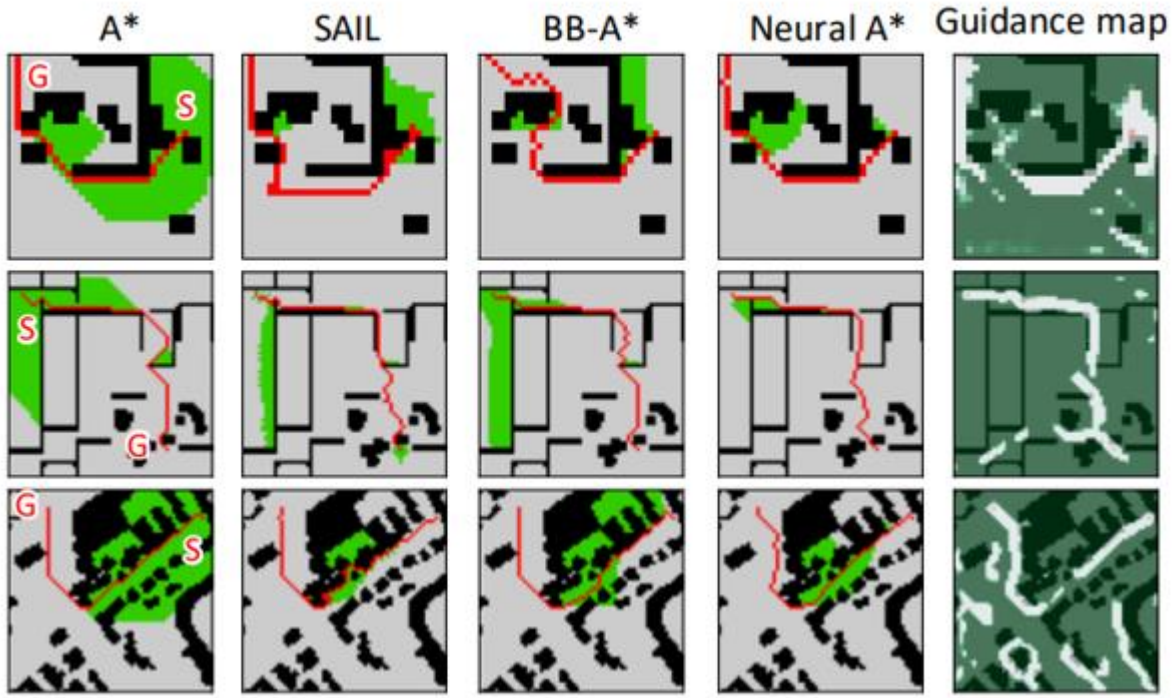




Path Planning using Neural A* Search (2021, ICML)

Experiment

❖ 데이터 기반 알고리즘 간 비교



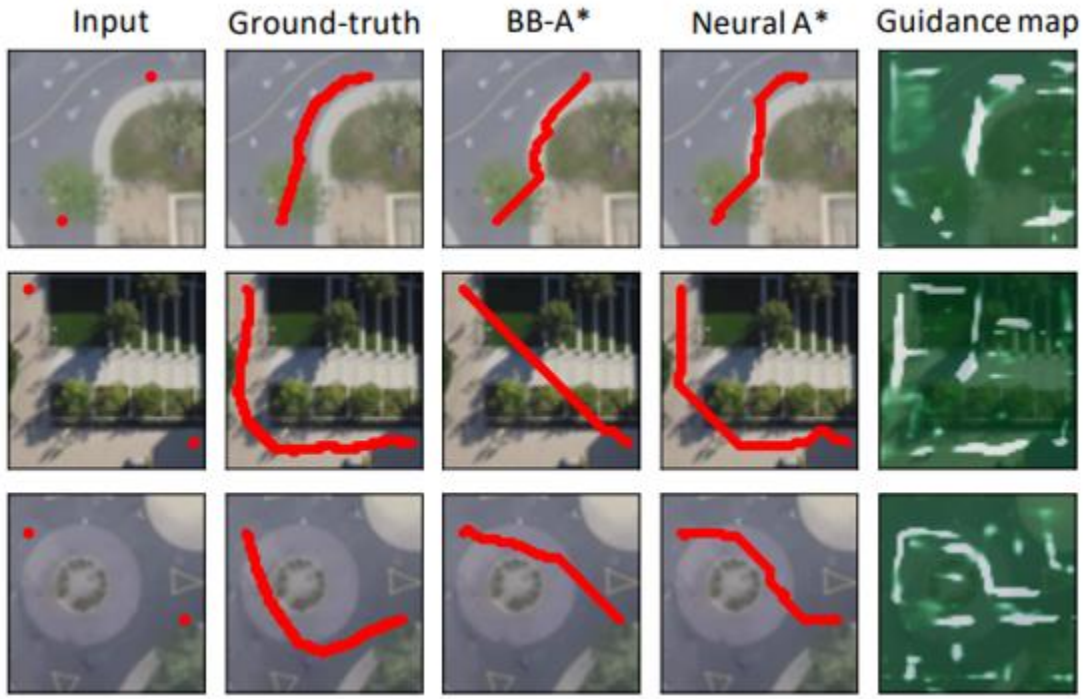
* The higher the better

	(a) Search optimality	(b) Search efficiency	Hmean of (a) and (b)
SAIL [Choudhury+, 2018]	34.6	48.6	26.3
BB-A* [Vlastelica+, 2020]	62.7	42.0	42.1
Neural A*	**87.7**	40.1	**52.0**

Path Planning using Neural A* Search (2021, ICML)

Experiment

❖ 이미지 기반 이동경로 실험 결과



Experimental setup

- Surveillance images + pedestrian trajectories as demonstrations
- Task: Predicting realistic trajectories consistent with those of pedestrians when start and goal locations are provided

	BB-A* [Vlastelica+, 2020]	Neural A*
Chamfer distance (the lower the better)	152.2	**16.1**

Path Planning using Neural A* Search (2021, ICML)

Conclusion

❖ **Contribution**

- A* Search 를 기반으로 학습 가능한 최적 경로 탐색 모델 제안
- A* 의 휴리스틱 함수를 Guidance Map 형태로 대체하여 인공적 Path 찾기 뿐만 아니라 이미지 기반 Path 찾기에도 적용 가능 한 점
- 미분가능한 A* 모듈 고안하여 A* Search 를 사용하는 네트워크를 학습 가능 한 구조 만든 점

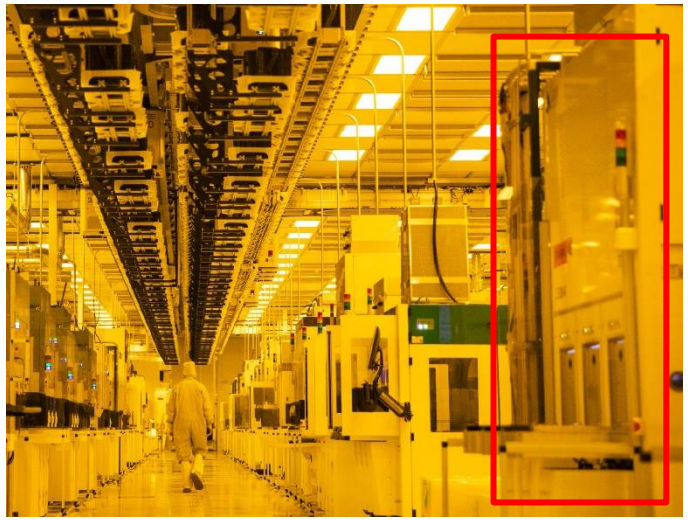
❖ **한계**

- Guidance Map 학습 실패 Case
- Guidance Map 이 최적 경로라는 이론적인 보장이 안됨

Introduction of Cluster Tools & Scheduling

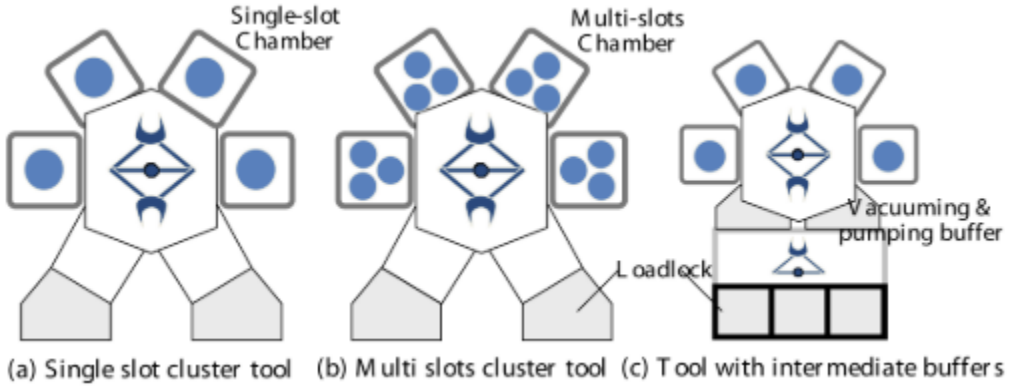
클러스터 설비 란

- ❖ Wafer 1장 단위로 공정을 수행하기 위한 설비 플랫폼
- ❖ 반도체 제조 장비의 70~80% 가 클러스터 장비로 구성



반도체 라인 내부

클러스터 설비



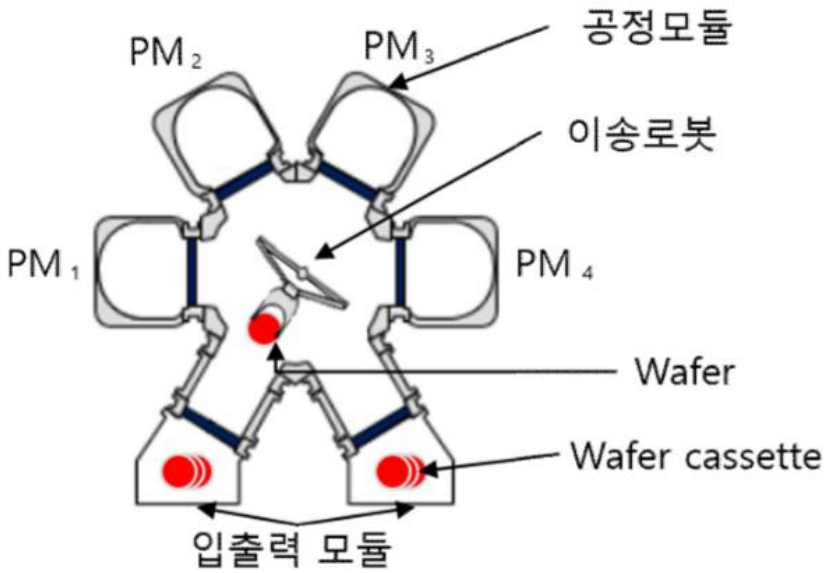
다양한 클러스터 장비 구조[1]

Introduction of Cluster Tools & Scheduling

What is cluster tool?

❖ **클러스터 장비란?**

- 공정모듈(PM)과 이송로봇(VTR)을 단일장비에 집약한 반도체 제조 장비
- 진공/대기 환경 분리
- 웨이퍼 1매 씩 처리하는 매엽식 클러스터 장비가 일반적



Introduction of Cluster Tools & Scheduling

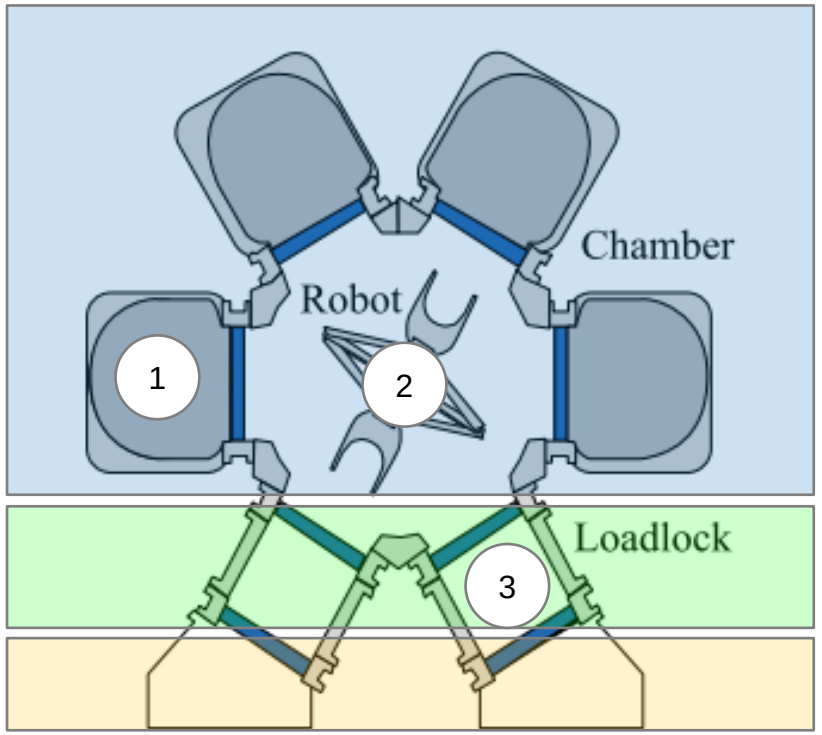
What is cluster tool?

❖ 용어

① Process module (PM) : 웨이퍼 가공이 이루어지는 챔버(chamber)

② Vacuum transfer robot (VTR) : 웨이퍼의 운반을 담당하는 로봇

③ Load lock (LL) : 웨이퍼 투입/배출 시 외부 대기압 환경과 진공 환경을 분리하는 역할



진공

진공 ↔ 대기압

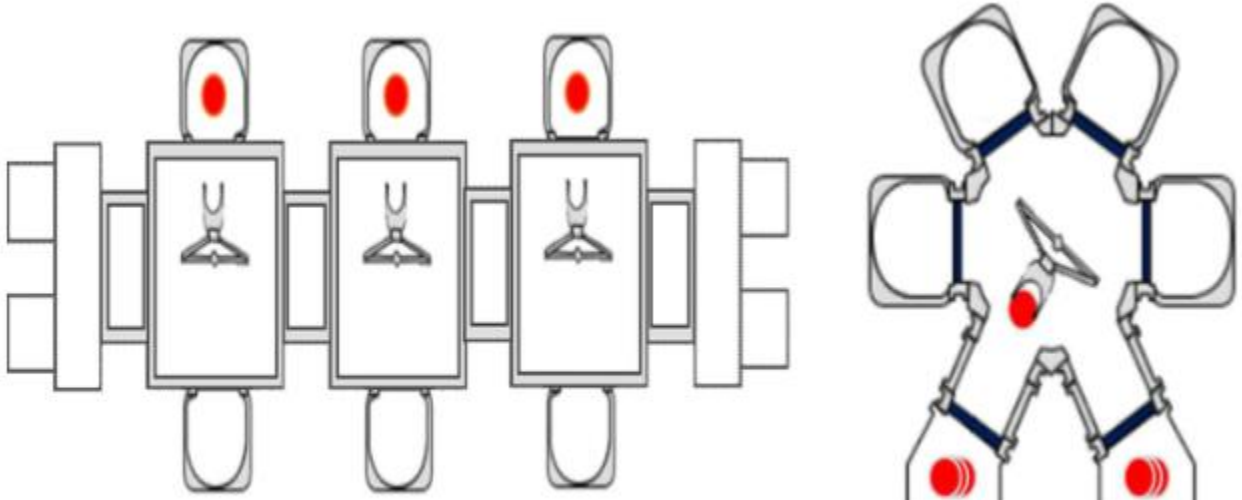
대기압

Introduction of Cluster Tools & Scheduling

What is cluster tool?

❖ **다양한 형태의 클러스터 장비**

- 선형 클러스터 장비 : PM이 일렬로 배치되어 공간 활용성 높으나, VTR 이동 동선이 김
- 원형 클러스터 장비 : PM이 VTR을 중심으로 배치되어 공간 활용성은 떨어지나 VTR 이동시간 측면 유리
- 원형 클러스터 장비가 대부분 사용 됨



(a) 선형 클러스터 장비　　(b) 원형 클러스터 장비

Introduction of Cluster Tools & Scheduling

What is cluster tool?

❖ **이송로봇(VTR) 의 종류**

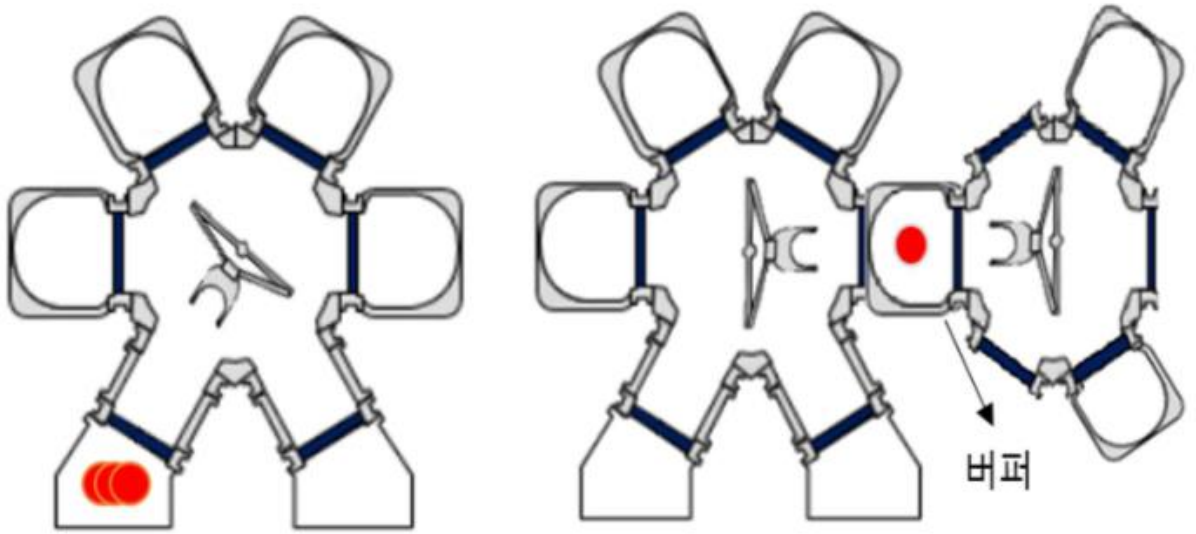
- Single Arm Robot : 웨이퍼를 집을 수 있는 로봇의 팔이 1개인 형태로 최신 장비에는 자주 쓰이지 않음
- Dual Arm Robot : 웨이퍼를 집을 수 있는 로봇의 팔이 2개인 형태

Introduction of Cluster Tools & Scheduling

What is cluster tool?

❖ **단일 클러스터 장비와 다중 클러스터 장비**

- 단일 클러스터 장비 : 일반적인 형태로, 하나의 VTR을 중심으로 여러 PM이 원형으로 배치되어 있는 형태
- 다중 클러스터 장비 : 특정 공정을 위해 여러대의 클러스터를 버퍼를 통해 연결 한 형태



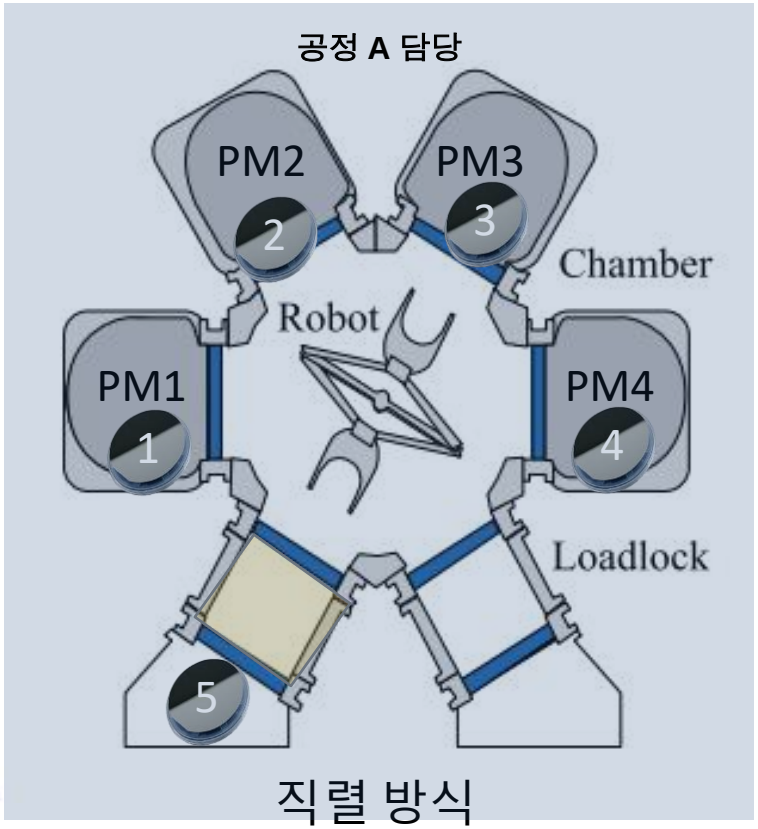
(a) 단일 클러스터 장비 (b) 다중 클러스터 장비

Introduction of Cluster Tools & Scheduling

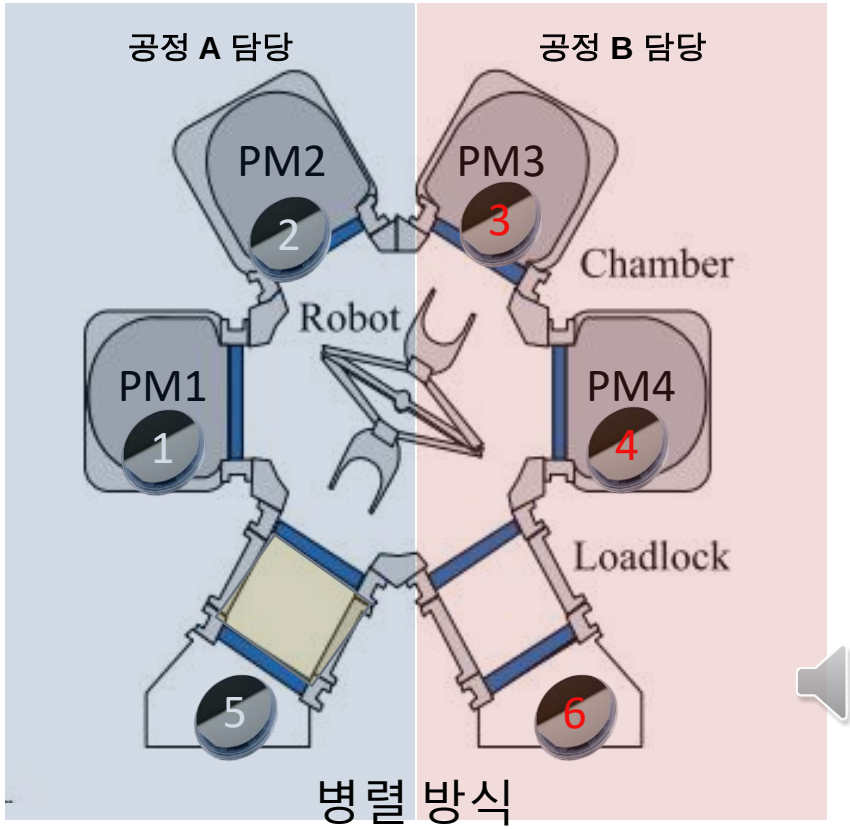
What is cluster tool?

❖ **클러스터 장비의 운용**

- 직렬방식 : 전체 PM이 하나의 공정을 다루는 형태로 현재 공정이 끝나야 다음 공정이 시작 되는 형태
- 병렬방식 : 각 PM 이 서로 다른 공정을 수행할 수 있어 동시에 여러 공정 처리 가능



직렬 방식



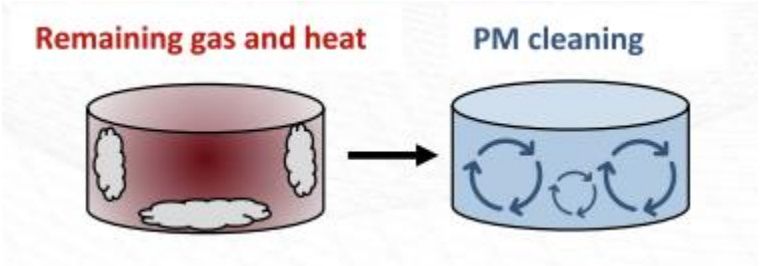
병렬 방식

Introduction of Cluster Tools & Scheduling

What is cluster tool?

❖ **클러스터 장비의 클리닝**

- 공정 중 발생한 클러스터 내 불순물을 제거하기위해 수행
- 주기적 또는 일정 수 의 공정 후 클리닝 하는 형태로 운용
- 클리닝 과정 중에는 웨이퍼 가공이 불가능 함



(a) Purge

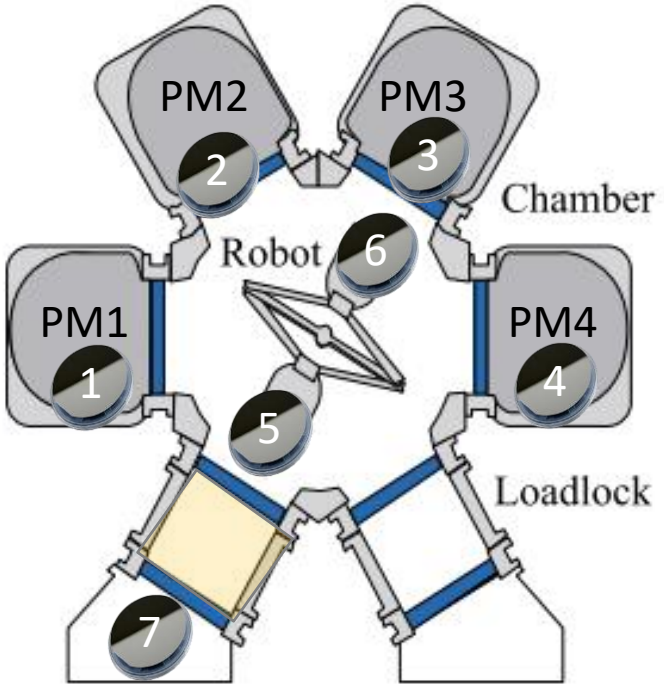
(b) 클리닝 주기가 3인 경우

Introduction of Cluster Tools & Scheduling

What is scheduling?

❖ **클러스터 장비 스케줄링**

- 여러 개의 PM과 운송 로봇으로 구성되어있어 스케줄링이 복잡
- 클러스터 장비를 운용하기위한 요구사항이 충족 되어야 함
  - 웨이퍼 지연
  - 챔버 클리닝
  - Deadlock 조건
  - 로봇의 유형 (Single-Arm / Dual-Arm)
  - 공정 흐름 (직렬 / 병렬) 등
- 요구사항을 충족하며 장비 생산성을 최대로 하는 것이 중요

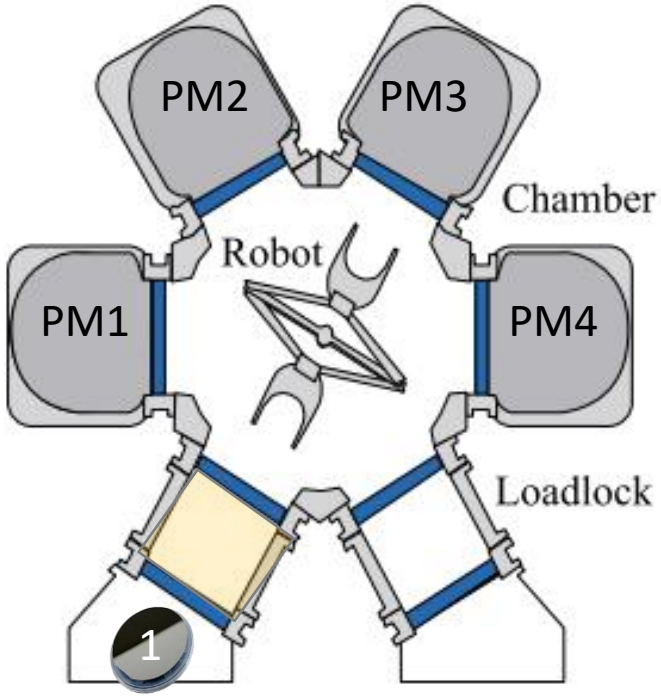


Deadlock state

Introduction of Cluster Tools & Scheduling

What is scheduling?

- **단순 스케줄링 시나리오**
  - 클러스터 내 모든 모듈 (PM, VTR, LL) 정상 동작 상태
  - "웨이퍼 1을 PM1에서 Process A 로 가공해라"

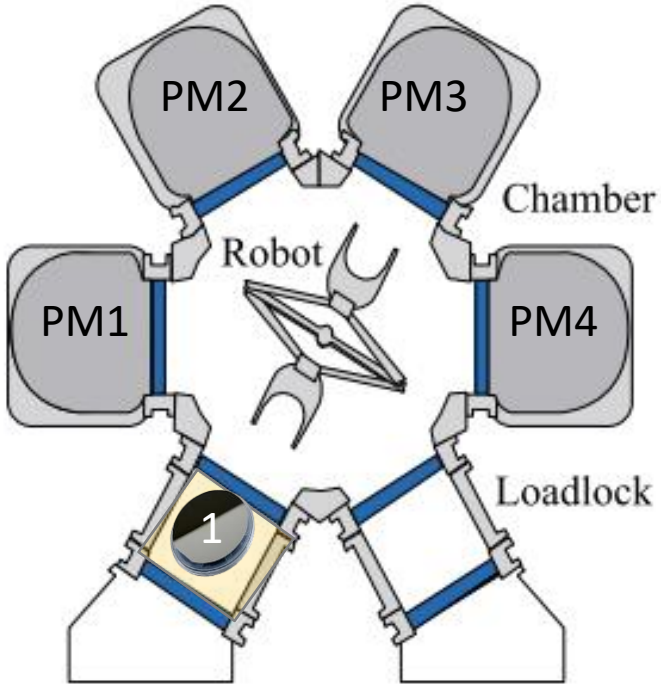


Introduction of Cluster Tools & Scheduling

What is scheduling?

❖ **단순 스케줄링 시나리오**

1. 웨이퍼 1을 LL 에 PUT

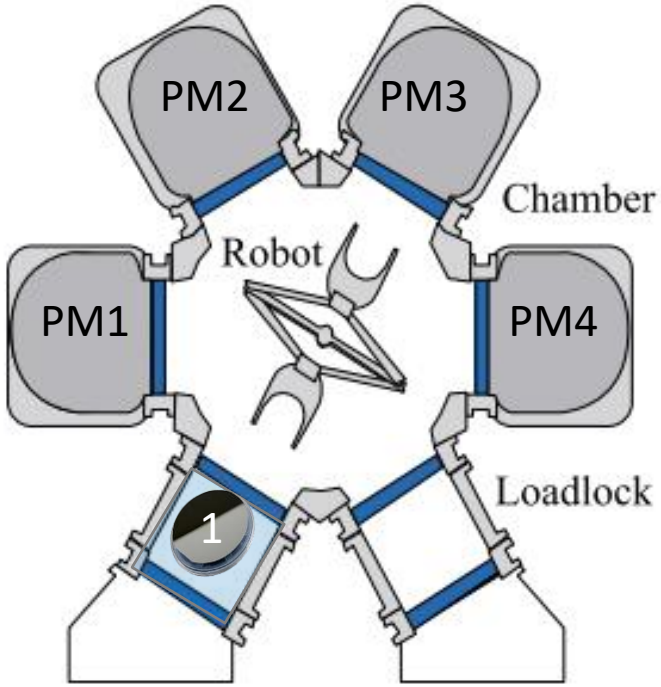


Introduction of Cluster Tools & Scheduling

What is scheduling?

❖ **단순 스케줄링 시나리오**

2. LL을 진공 상태로 전환

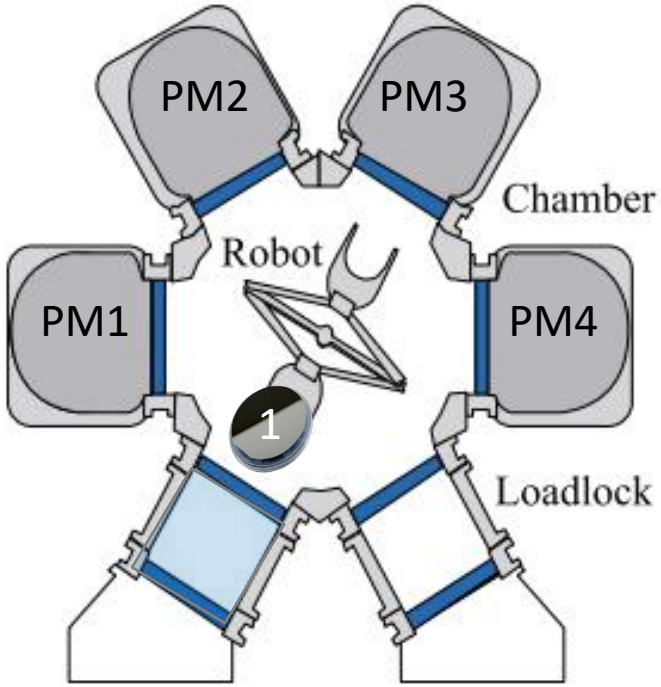


Introduction of Cluster Tools & Scheduling

What is scheduling?

❖ **단순 스케줄링 시나리오**

3. VTR 이 웨이퍼 1을 PICK

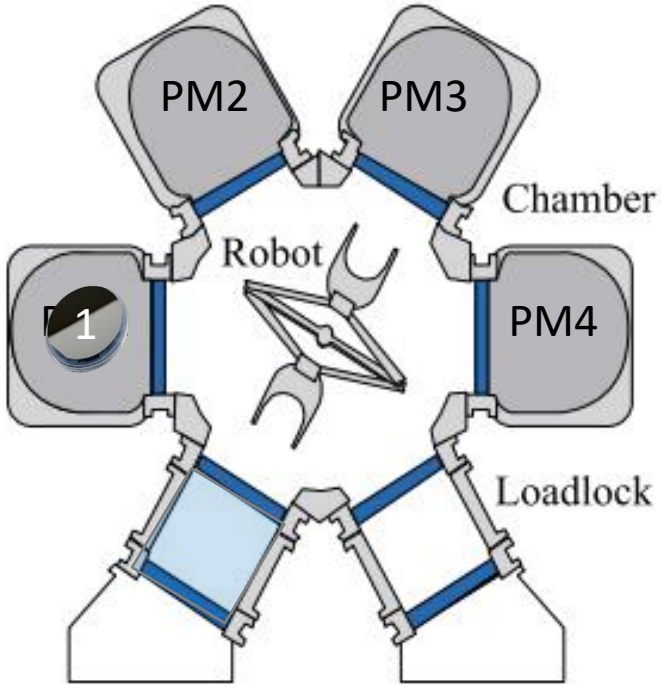


Introduction of Cluster Tools & Scheduling

What is scheduling?

❖ **단순 스케줄링 시나리오**

4. VTR 이 웨이퍼 1을 PM1 에 PUT

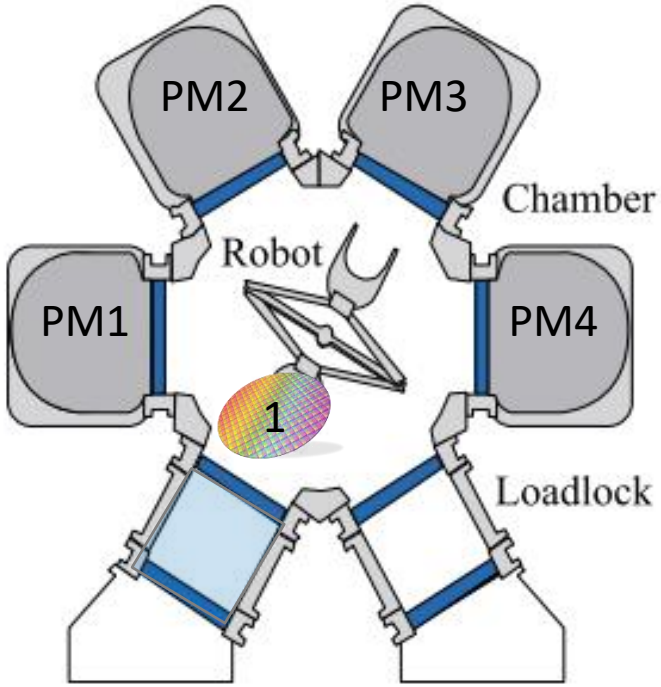


Introduction of Cluster Tools & Scheduling

What is scheduling?

❖ **단순 스케줄링 시나리오**

5. PM1 가공 완료 되었으면, VTR 이 웨이퍼 1을 PM1 으로부터 PICK

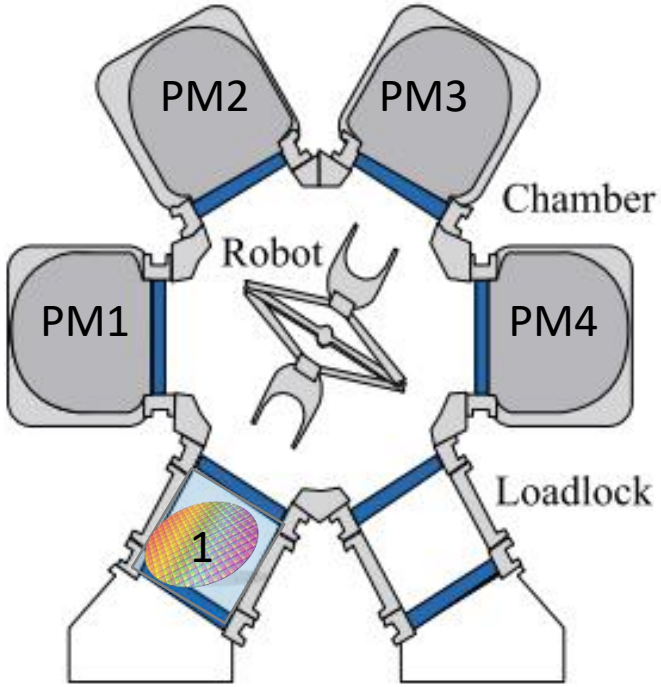


Introduction of Cluster Tools & Scheduling

What is scheduling?

❖ **단순 스케줄링 시나리오**

6. VTR 이 웨이퍼 1을 LL에 PUT

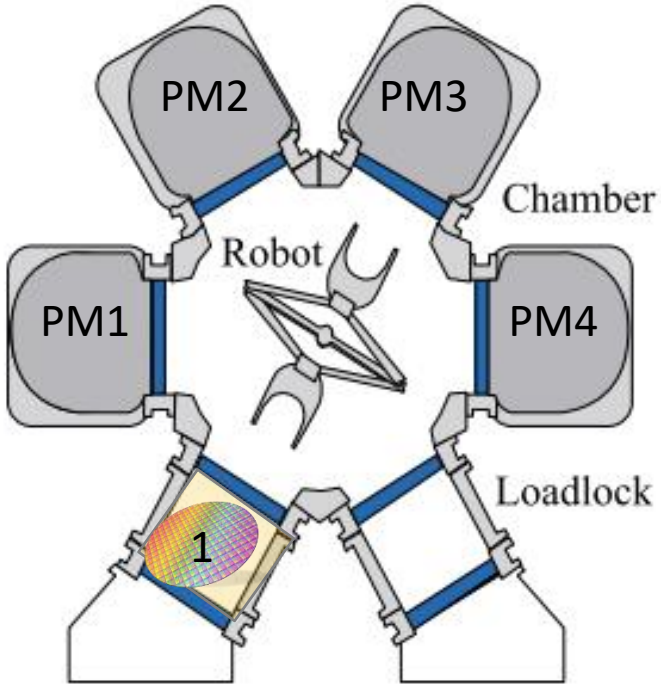


Introduction of Cluster Tools & Scheduling

What is scheduling?

❖ **단순 스케줄링 시나리오**

7. LL을 대기압 상태로 전환

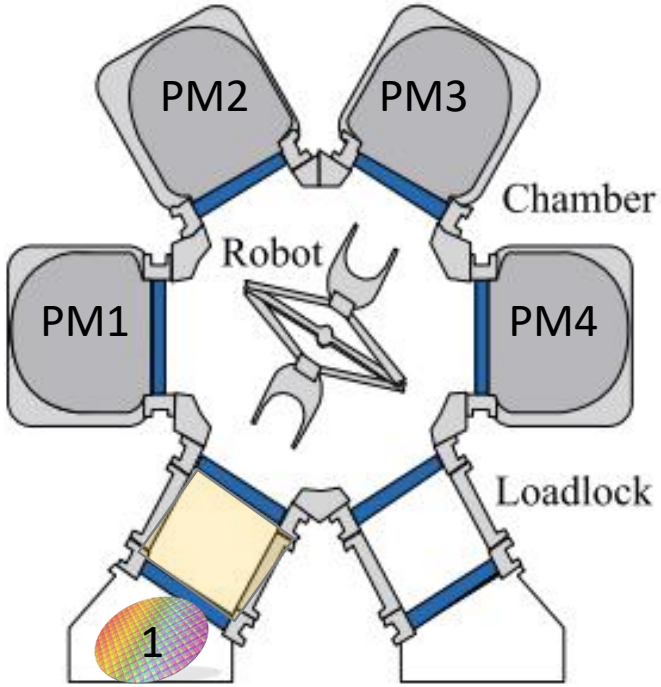


Introduction of Cluster Tools & Scheduling

What is scheduling?

❖ **단순 스케줄링 시나리오**

8. 웨이퍼 1을 LL으로 부터 PICK [작업 완료]

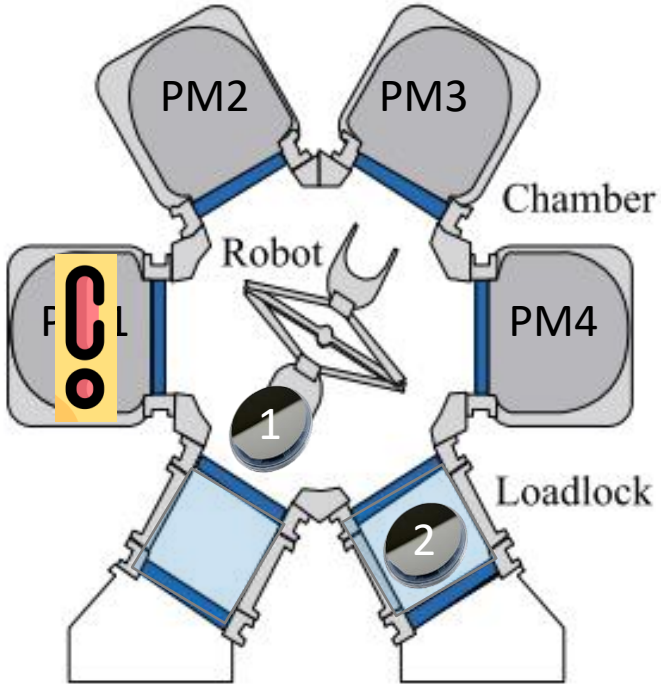


Introduction of Cluster Tools & Scheduling

What is scheduling?

❖ **스케줄링 예외 시나리오**

- 웨이퍼 1을 PM1 에 투입하려는데 PM1에 문제 발생
- PM2에서 가공 대기중인 웨이퍼 2가 LL에 있음

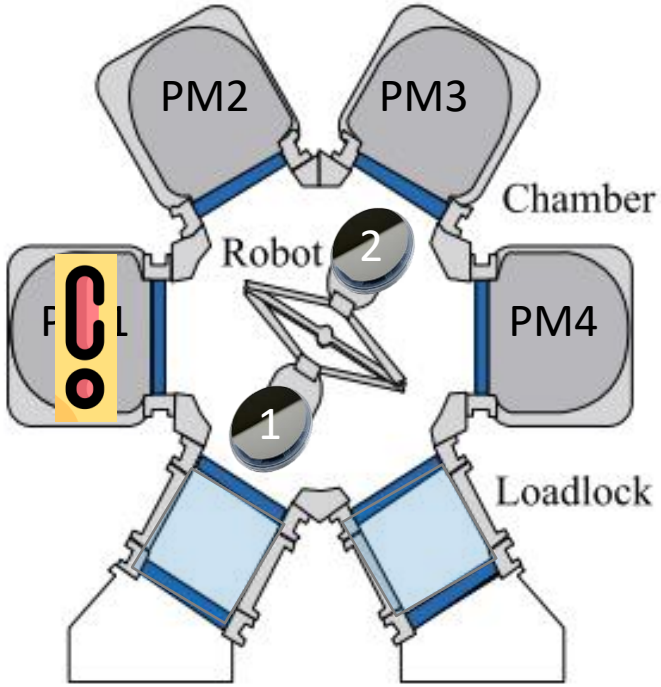


Introduction of Cluster Tools & Scheduling

What is scheduling?

❖ **스케줄링 예외 시나리오**

- VTR이 웨이퍼 2를 LL으로 부터 PICK

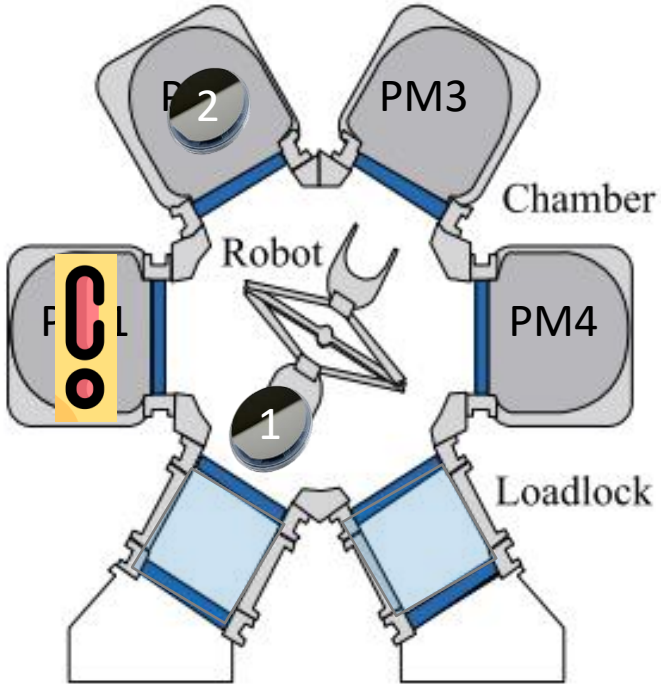


Introduction of Cluster Tools & Scheduling

What is scheduling?

❖ **스케줄링 예외 시나리오**

- VTR이 웨이퍼 2를 PM2로 PUT

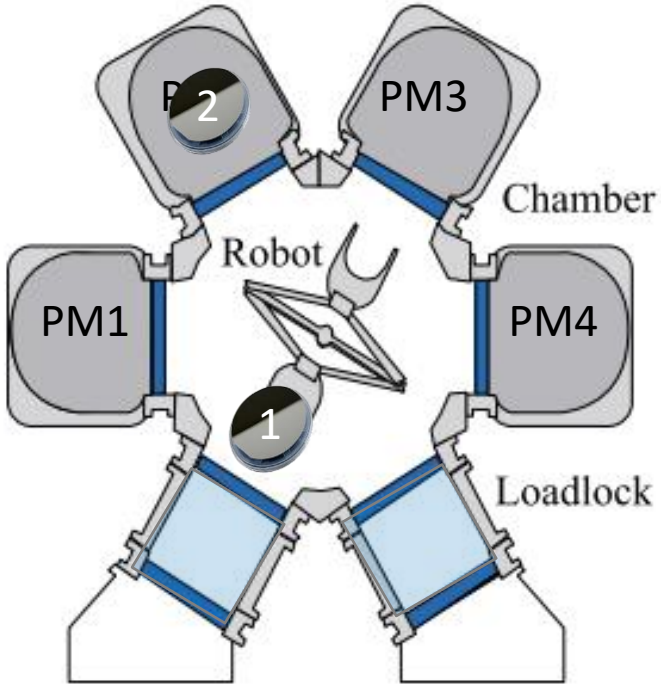


Introduction of Cluster Tools & Scheduling

What is scheduling?

❖ **스케줄링 예외 시나리오**

- PM1이 정상 상태로 전환 됨

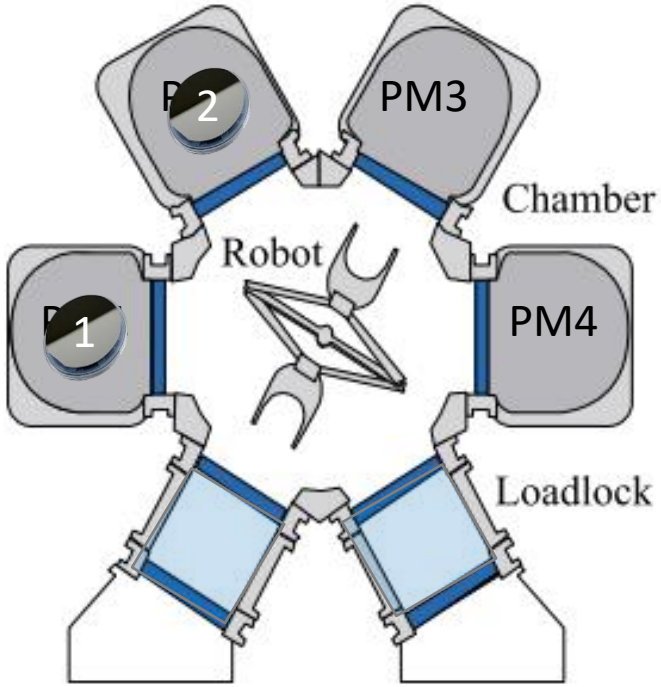


Introduction of Cluster Tools & Scheduling

What is scheduling?

❖ **스케줄링 예외 시나리오**

- PM1이 정상 상태로 전환 됨
- VTR이 웨이퍼 1을 PM1에 PUT

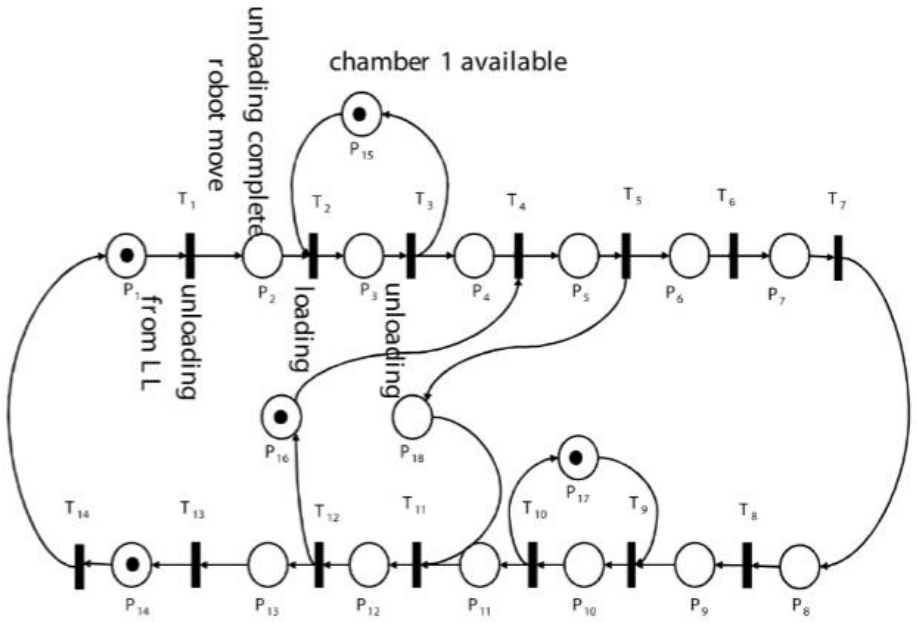


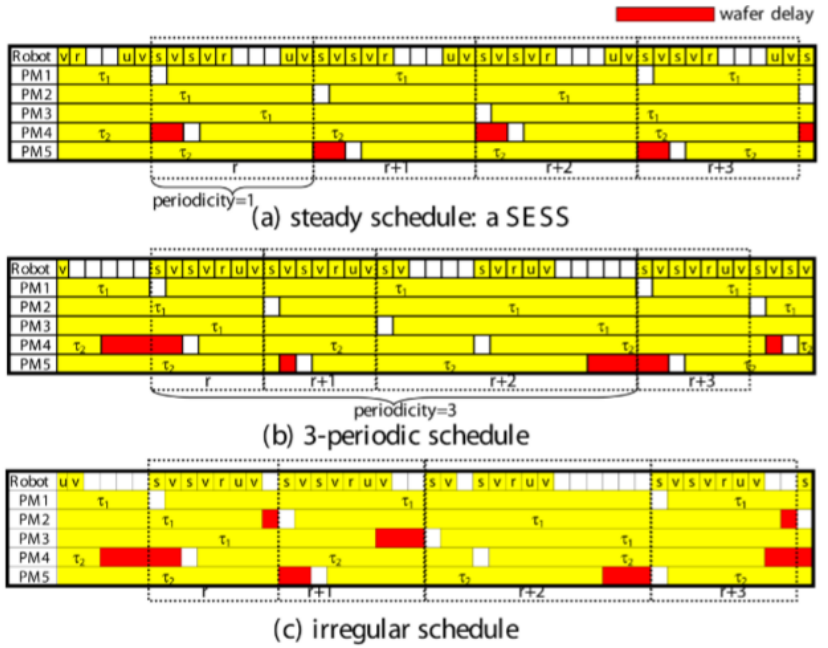
Introduction of Cluster Tools & Scheduling

Studies on Cluster Tool Scheduling

❖ Cycle Scheduling

- 미리 정해진 Cycle을 반복 수행하는 시나리오로 한정
- 스케줄링에 대한 높은 설명력
- TEG(Timed Event Graph)를 통한 수리적 최적 시나리오를 도출 가능
- 다양한 상황(제약조건)에 최적 Cycle time을 달성하기위한 많은 연구가 이루어졌음.



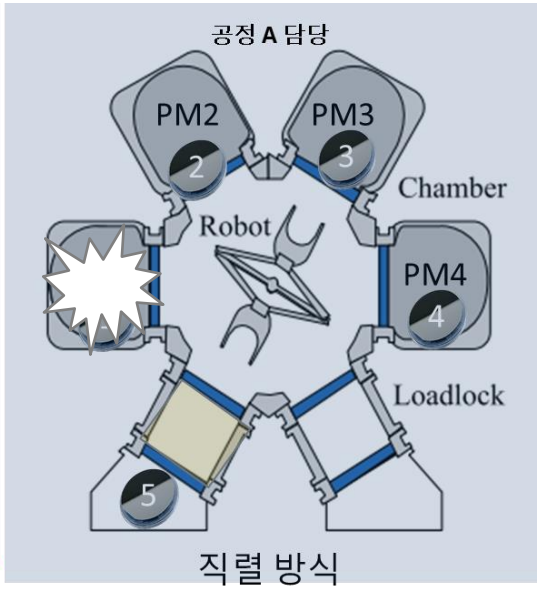


Introduction of Cluster Tools & Scheduling

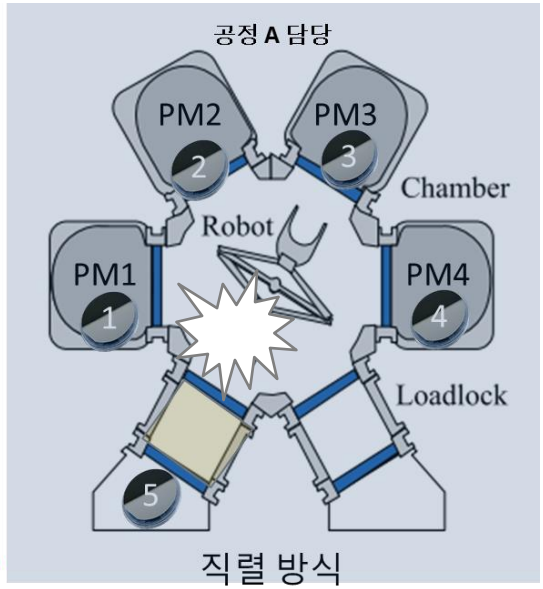
Studies on Cluster Tool Scheduling

❖ **Non-cyclic Scheduling : Dispatching rule**

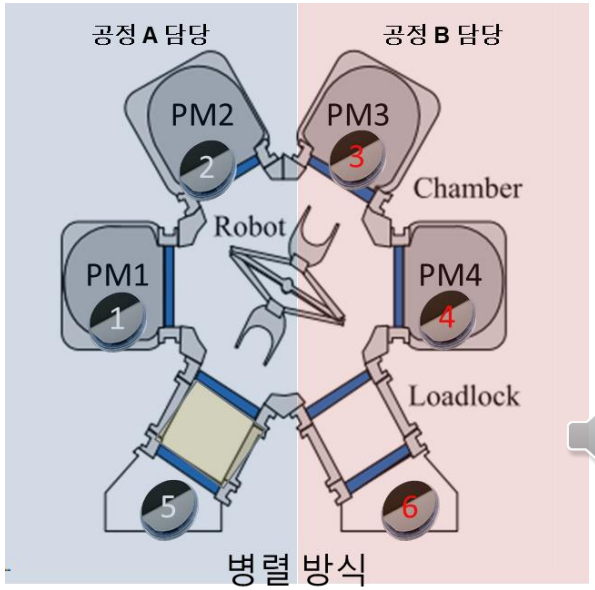
- 클러스터 장비 운용 시 발생 가능한 다양한 예외상황
  - PM1 이 Shutdown 됨
  - Dual-Arm 로봇 팔 하나가 고장 남
  - PM4 의 클리닝이 필요한 상황
  - 다른 공정 Wafer 를 같이 투입해야 할 경우 (병렬 방식)



직렬 방식



직렬 방식



병렬 방식

Introduction of Cluster Tools & Scheduling

Studies on Cluster Tool Scheduling

❖ **Non-cyclic Scheduling : Dispatching rule**

- 실제 클러스터 장비는 Cyclic Scheduling 방식으로 운용이 거의 불가능함 → 실제 클러스터 장비에 사용되는 스케줄링
- 로봇, 챔버, 웨이퍼의 상황에 따라 다음 Action을 결정 함
- 기존 Heuristic, Trial & Error 방식으로 구현 함 (if-else)

❖ **Dispatching rule 의 한계**

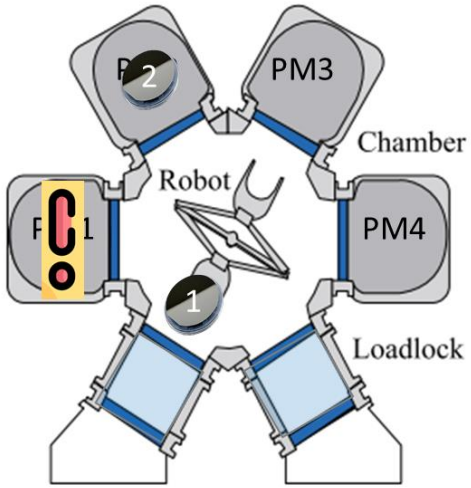
- 기존 수학적인 모델로 설명할 수 없거나, Complexity가 매우 높아 최적 해를 찾기 어려움
- 클러스터 장비의 성능 지표인 Throughput 검증하기위해 상황별로 시뮬레이션을 돌려서 평가 필요

Introduction of Cluster Tools & Scheduling

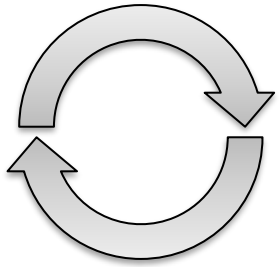
Studies on Cluster Tool Scheduling

❖ **Non-cyclic Scheduling : 강화학습 기반**

- 결국 상황에 따른 복잡한 조합 최적화 문제(NP-Hard)
- 유사한 Neural combinatorial optimization 관련 연구 존재
- 설비의 상태를 State로 하여 Throughput을 최대로 하는 Action 강화학습 모델로 Dispatching rule 대체 가능



$$Chamber\ status$$
$$Wafer\ location$$
$$Process\ status\ \ \ldots$$



$$Next\ Action, Rewards$$

Reinforcement Learning

Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Introduction

❖ **기존 연구들은 Cyclic Scheduling 을 가정**

- 실제 현장 적용과는 괴리감이 존재

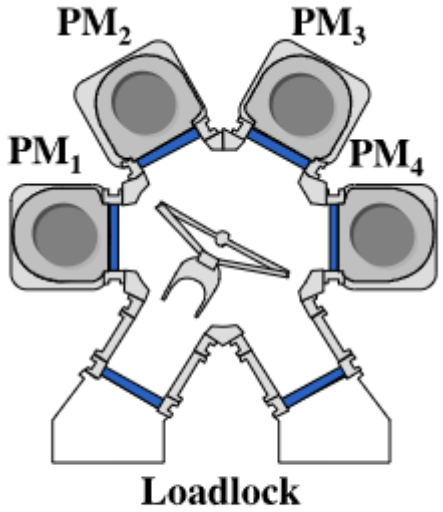
❖ **Non-cyclic Scheduling 연구의 필요성**

- 고정된 웨이퍼 유형의 수와 순서를 지정하는 Cyclic Scheduling 에서는 불필요한 장비의 Idle Time 이 발생하게 됨

❖ **병렬 처리 (Concurrent Processing) 상황 가정**

- 예) 하나의 Cluster tool 에서 2가지 유형의 Wafer를 처리 하는 상황

  Type 1 Wafer For PM1,2 / Type 2 Wafer For PM 3, 4



(a) Single-armed cluster tool

❖ **Goal : 주어진 수의 처리된 웨이퍼를 완료하는 데 소요된 시간으로 나눈 값을 통해 전체 처리량을 최대화**

Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Introduction

❖ **Gantt chart 비교**

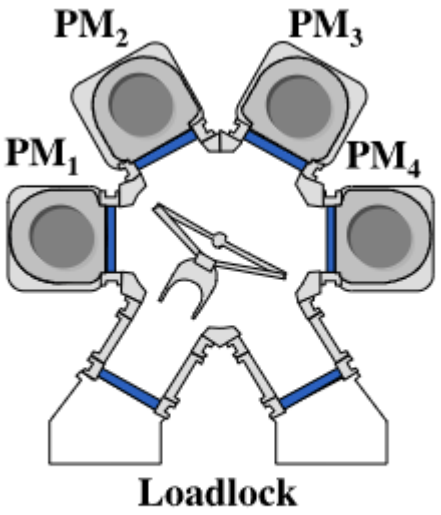
- 상단 (Cyclic Scheduling) : 고정된 순서로 웨이퍼를 처리, 주기마다 동일한 작업 반복 → 최적의 Throughput 보장 X
- 하단 (Non-cyclic Scheduling) : 동적으로 장비의 작업 순서를 조정하여 처리량을 극대화

Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

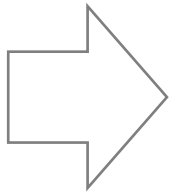
Method

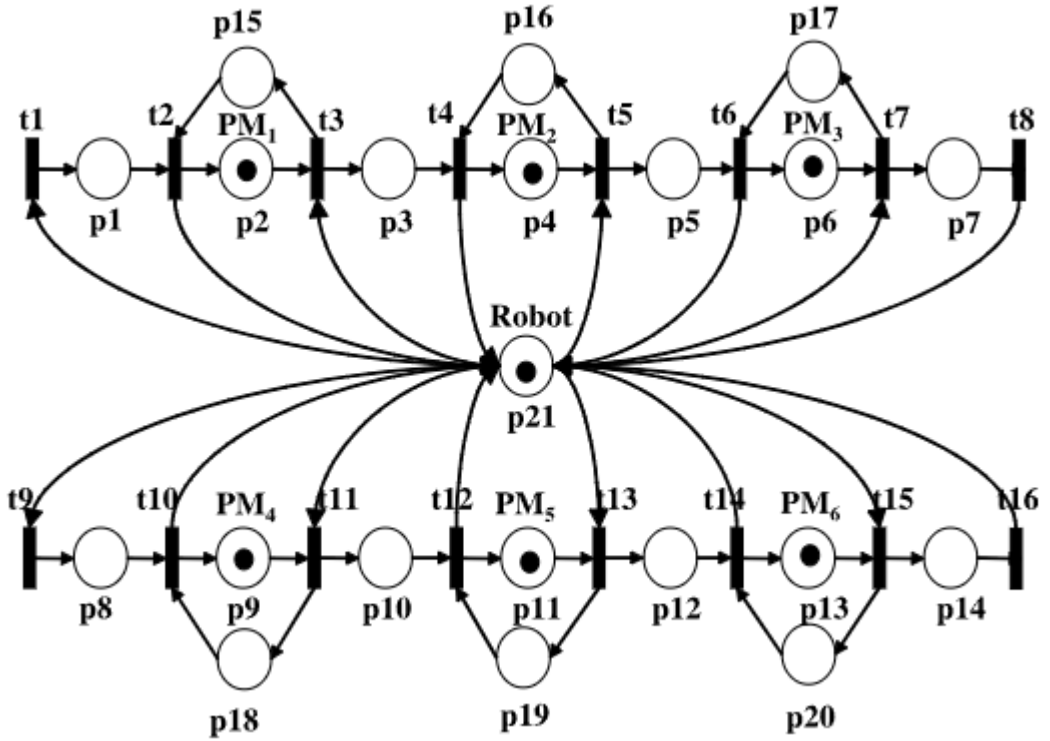
❖ **RL Environment : Timed Petri-Net (TPN)**

- 2개의 서로 다른 공정을 동시에 수행하는 Single Arm Cluster tool을 TPN의 형태로 모델링 함



(a) Single-armed cluster tool

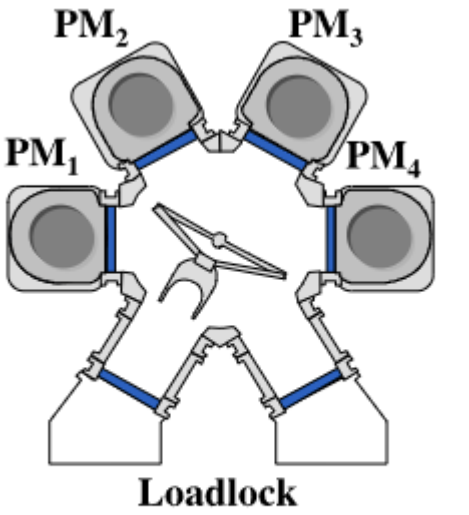




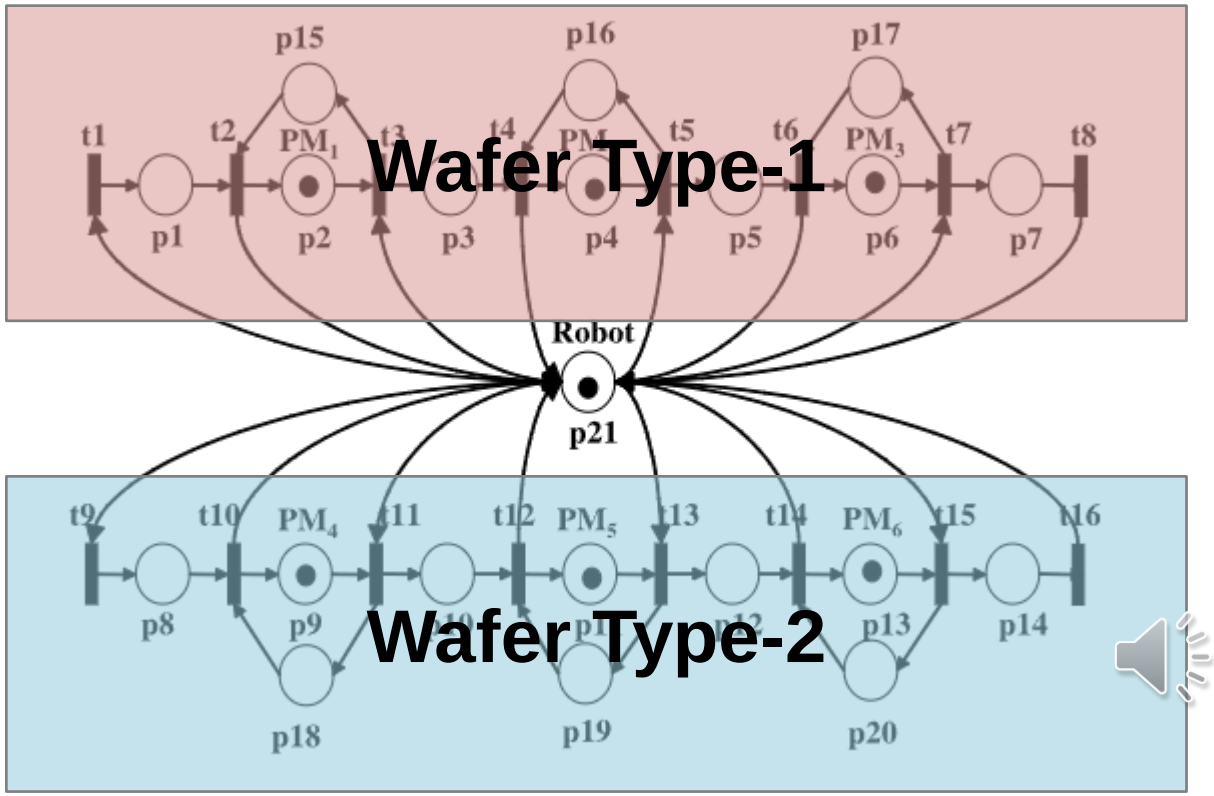
Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

- **RL Environment : Timed Petri-Net (TPN)**
  - PM 1, 2, 3 For Wafer Type-1
  - PM 4, 5, 6 For Wafer Type-2



(a) Single-armed cluster tool



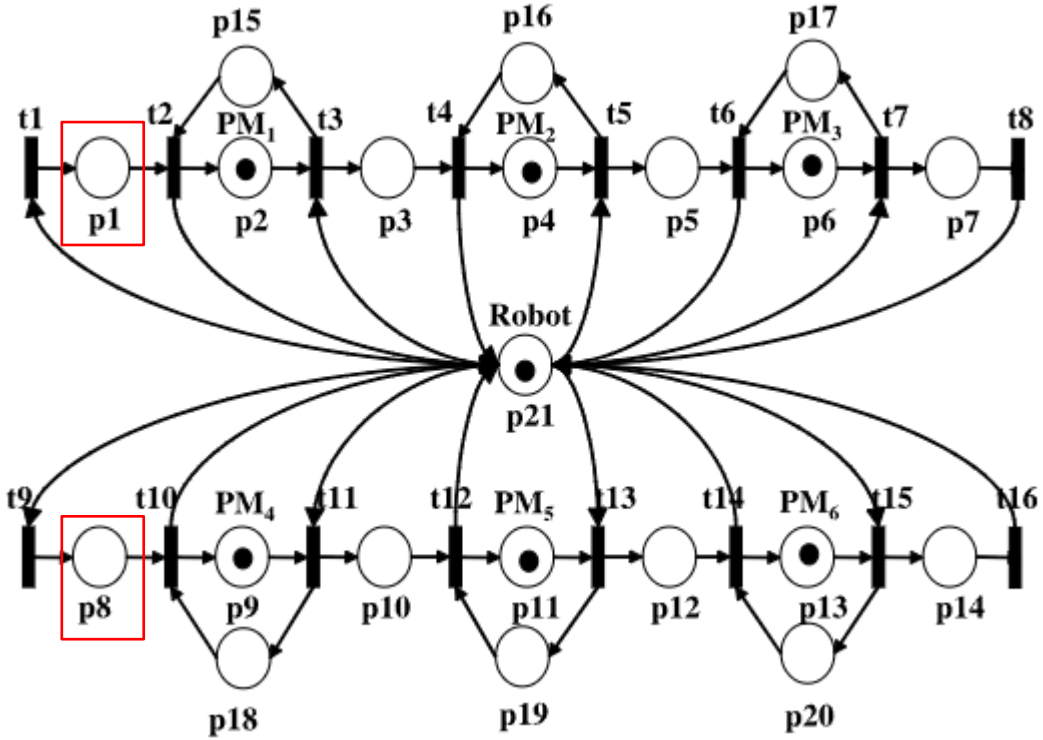
Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ TPN for a Cluster Tool Handling Two Wafer Types

- Place : 시스템의 상태나 조건

  예) 웨이퍼가 특정 PM에 있는 상태

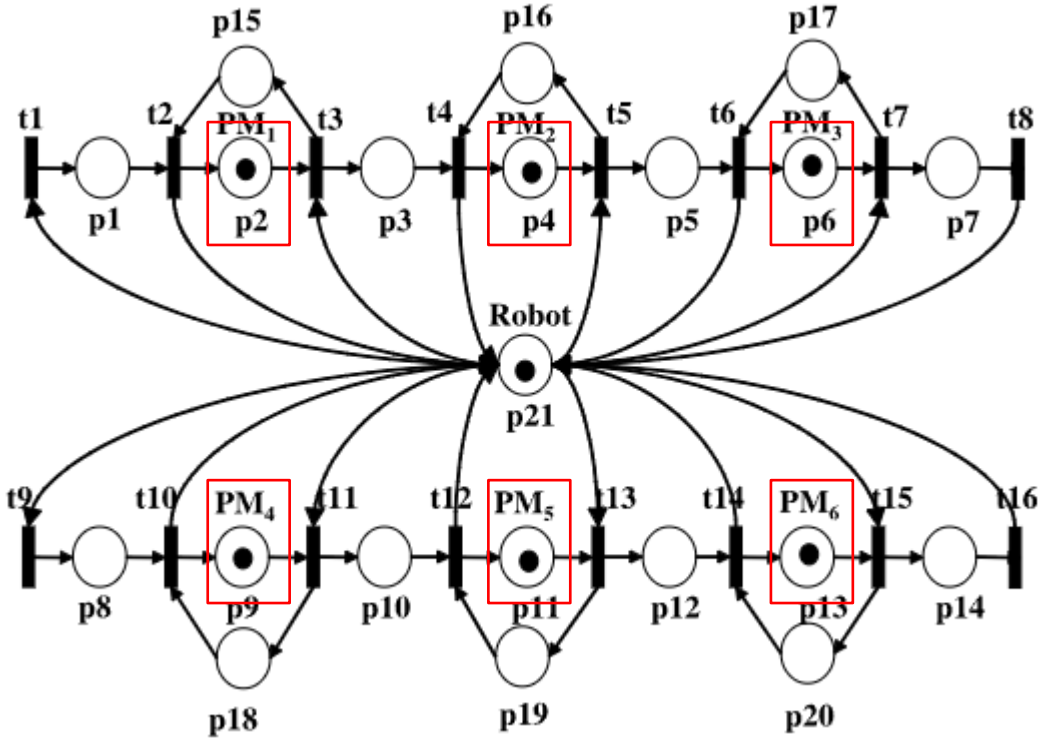


Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ TPN for a Cluster Tool Handling Two Wafer Types

- p2, p4, p6, p9, p11, p13: PM1, PM2, PM3, PM4, PM5, PM6에서의 웨이퍼 처리

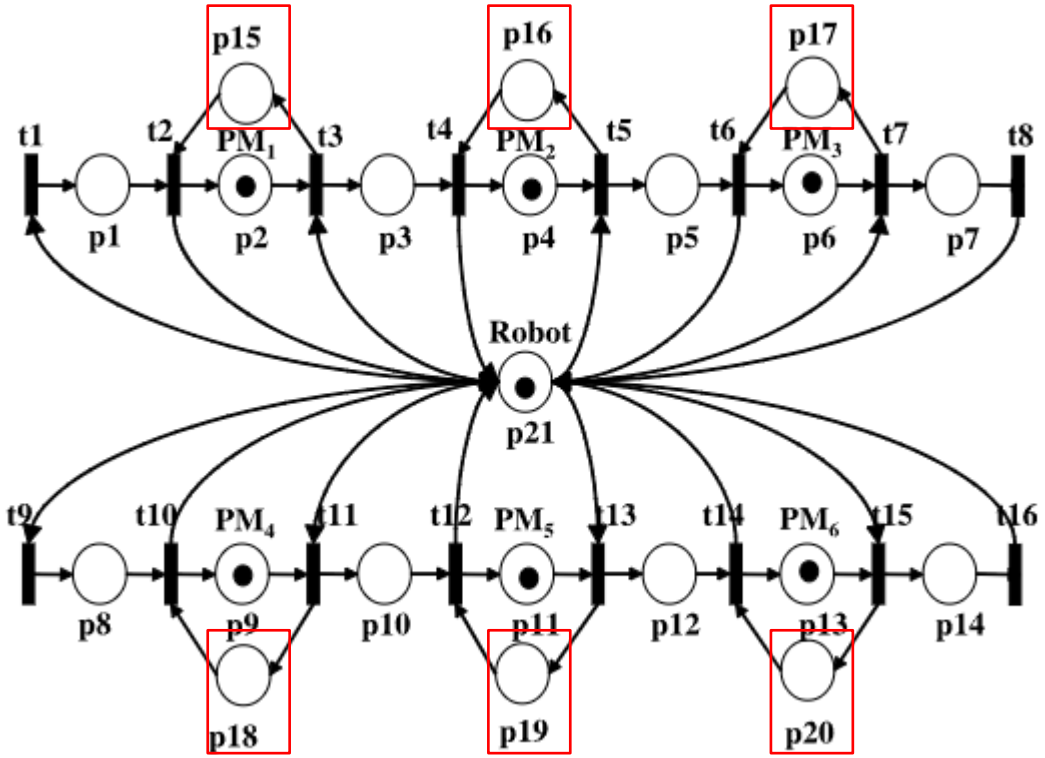


Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ **TPN for a Cluster Tool Handling Two Wafer Types**

- p15 - p20: PM1부터 PM6까지의 사용 가능 상태

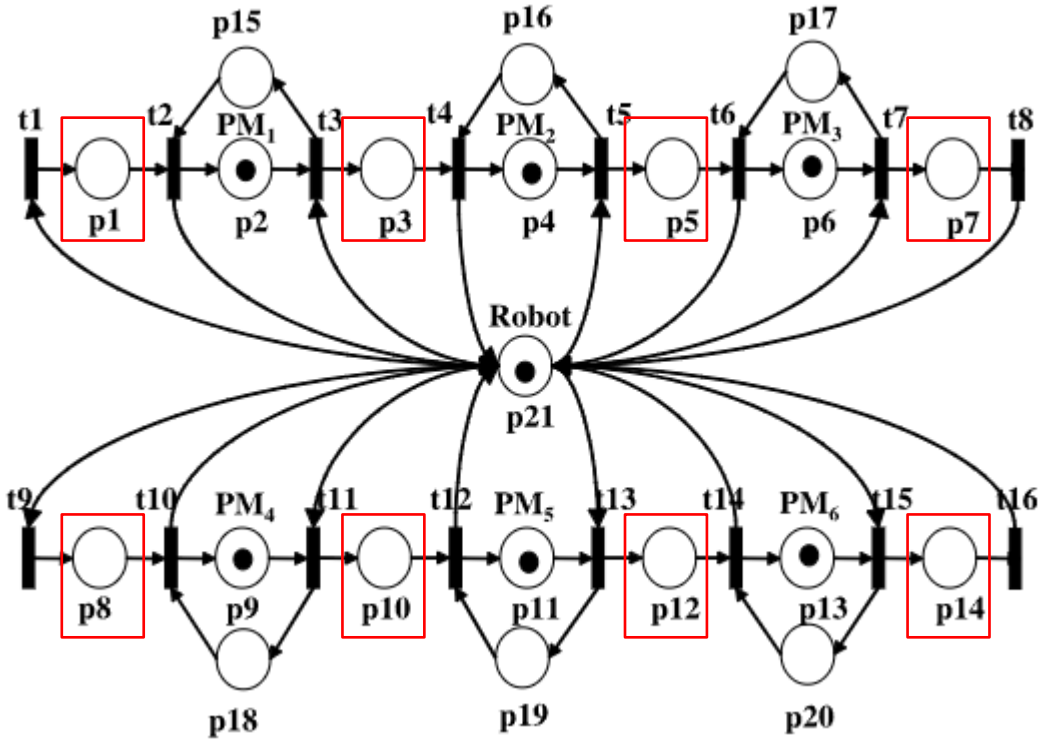


Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ **TPN for a Cluster Tool Handling Two Wafer Types**

- p1, p3, p5, p7, p8, p10, p12, p14: 로봇이 웨이퍼를 각 PM으로 운송하는 과정

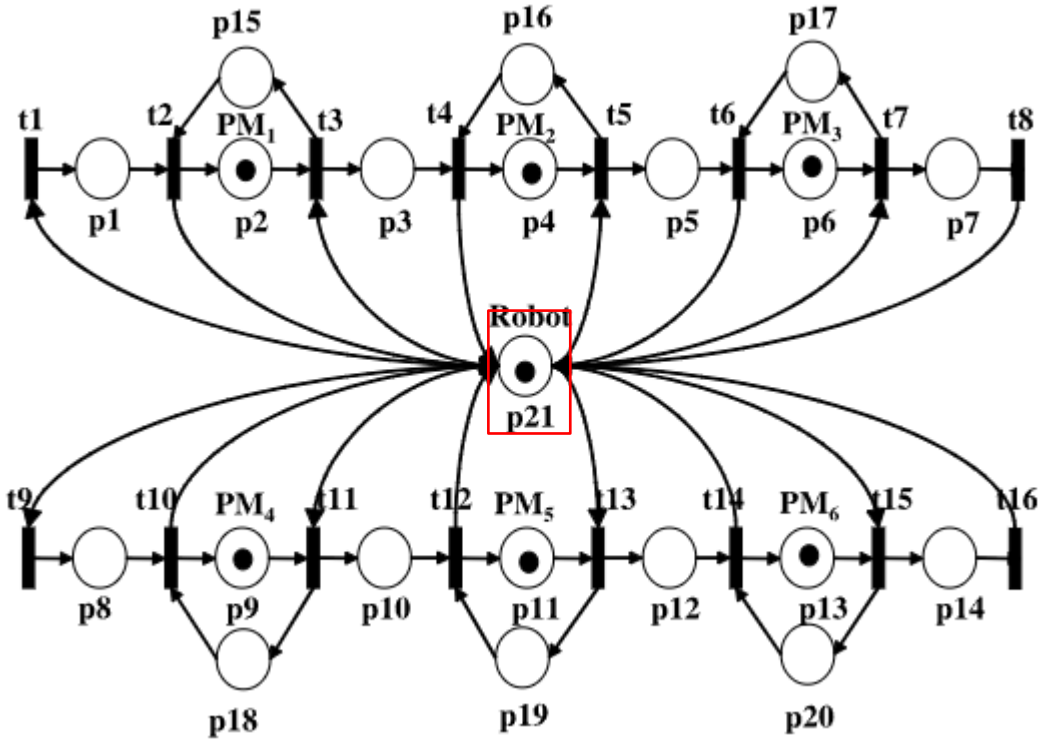


Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ **TPN for a Cluster Tool Handling Two Wafer Types**

- p21: 로봇의 사용 가능 상태



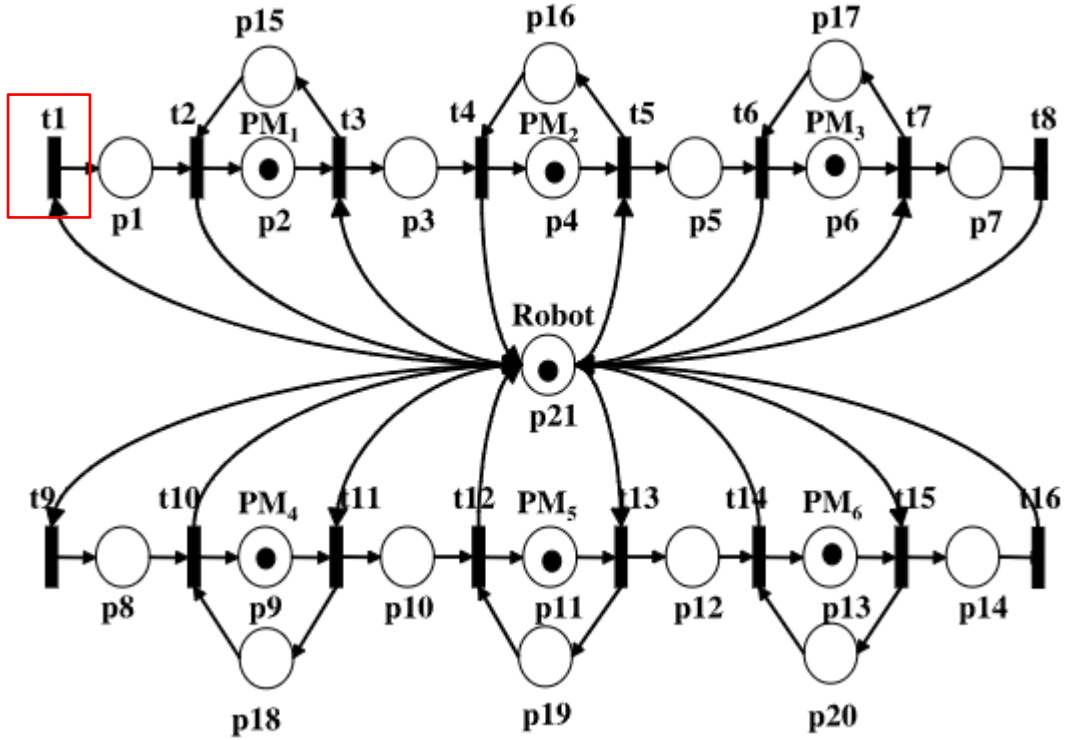
Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ TPN for a Cluster Tool Handling Two Wafer Types

- Transition : 이벤트나 Action을 나타내며, 하나의 상태에서 다른 상태로의 전환을 의미

  예) 웨이퍼를 PM1에서 Unload 후 PM2에 Load 하는 작업

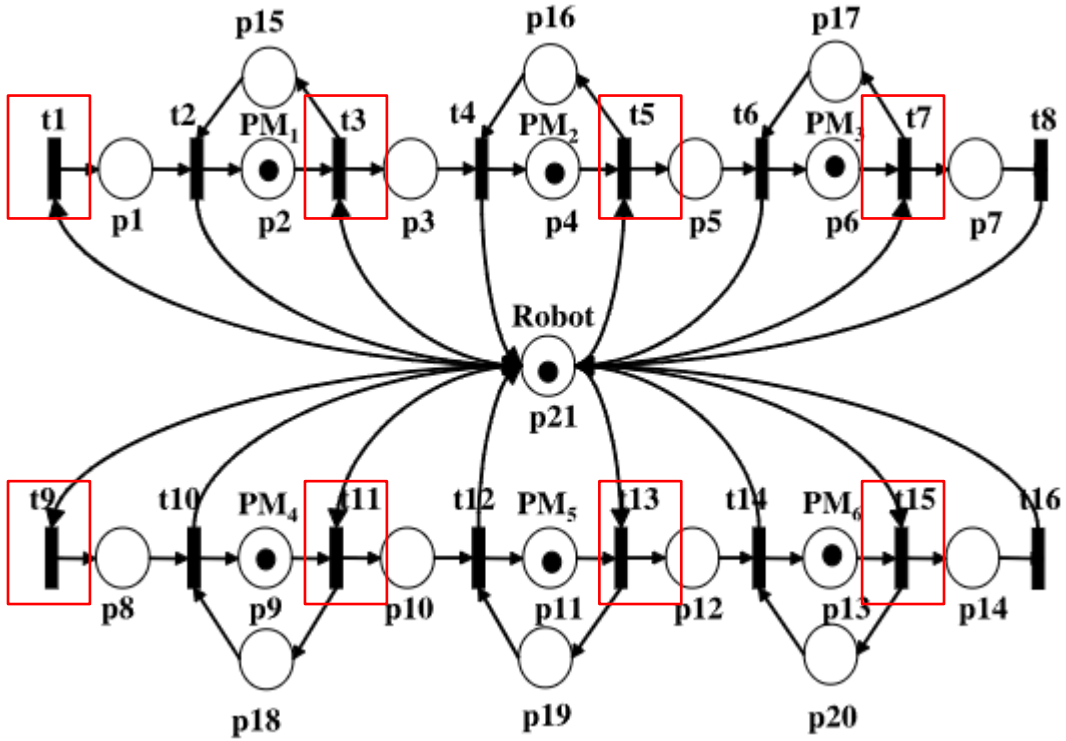


Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ TPN for a Cluster Tool Handling Two Wafer Types

- t1, t3, t5, t7, t9, t11, t13, t15 : 웨이퍼가 특정 모듈 (Loadlock, PM) 로 부터 Unload 되는 Action

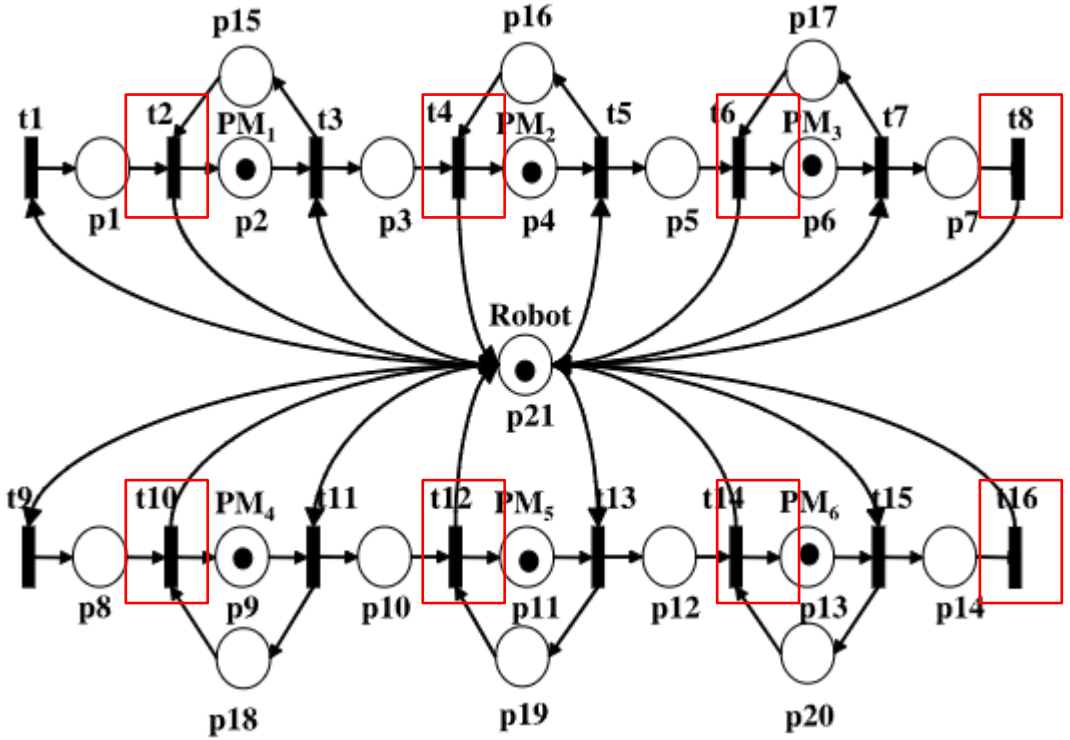


Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ TPN for a Cluster Tool Handling Two Wafer Types

- t2, t4, t6, t8, t10, t12, t14, t16 웨이퍼가 특정 모듈 (Loadlock, PM) 로 부터 Load 되는 Action



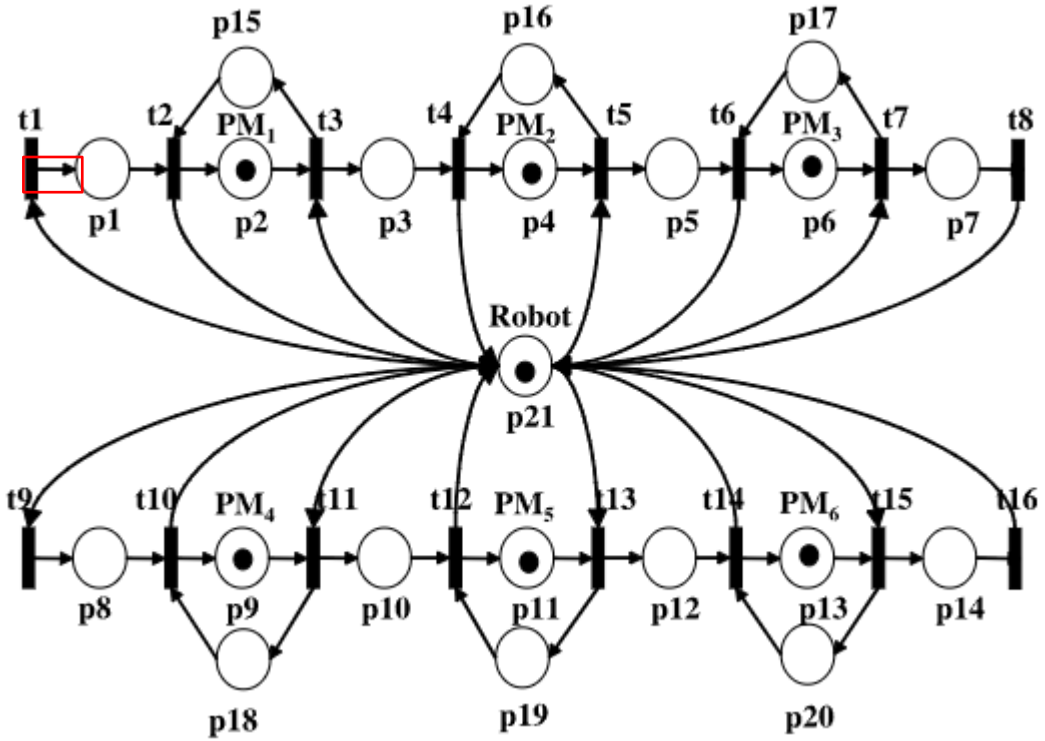
Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ TPN for a Cluster Tool Handling Two Wafer Types

- Arcs : Place와 Transition을 연결, 토큰의 이동 경로를 나타냄

  예) PM에서 로봇으로 웨이퍼가 이동하는 경로



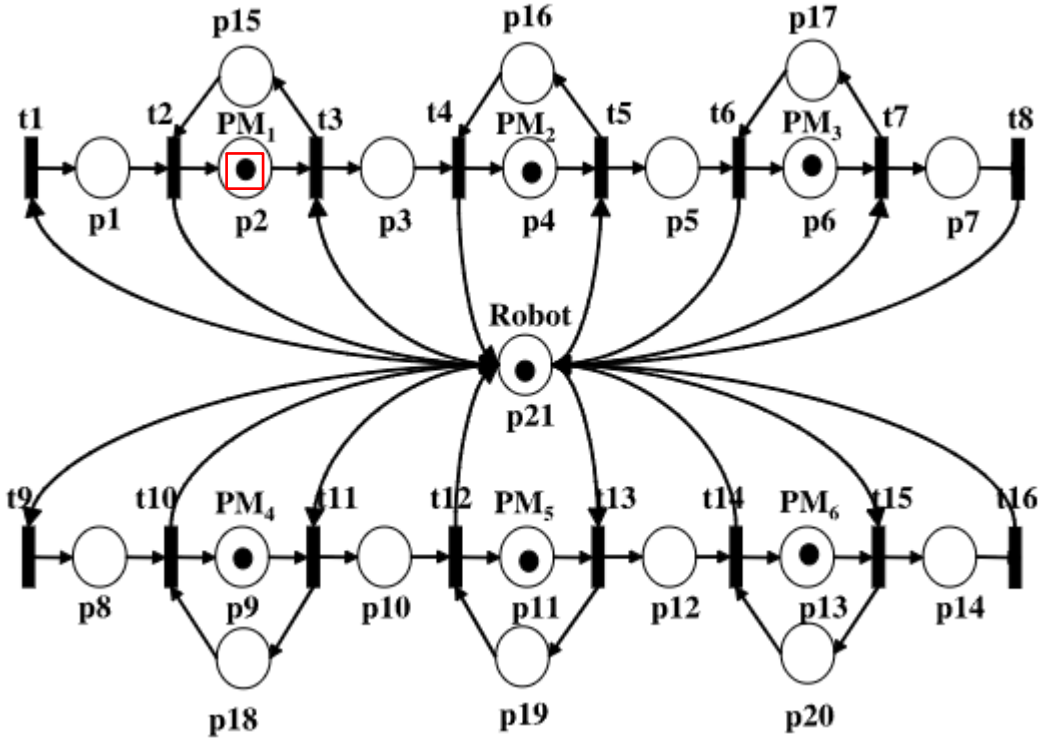
Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ TPN for a Cluster Tool Handling Two Wafer Types

- Tokens : 현재 시스템의 상태를 나타내는 Marker

  예: 특정 PM에서 처리 중인 Wafer를 나타냄

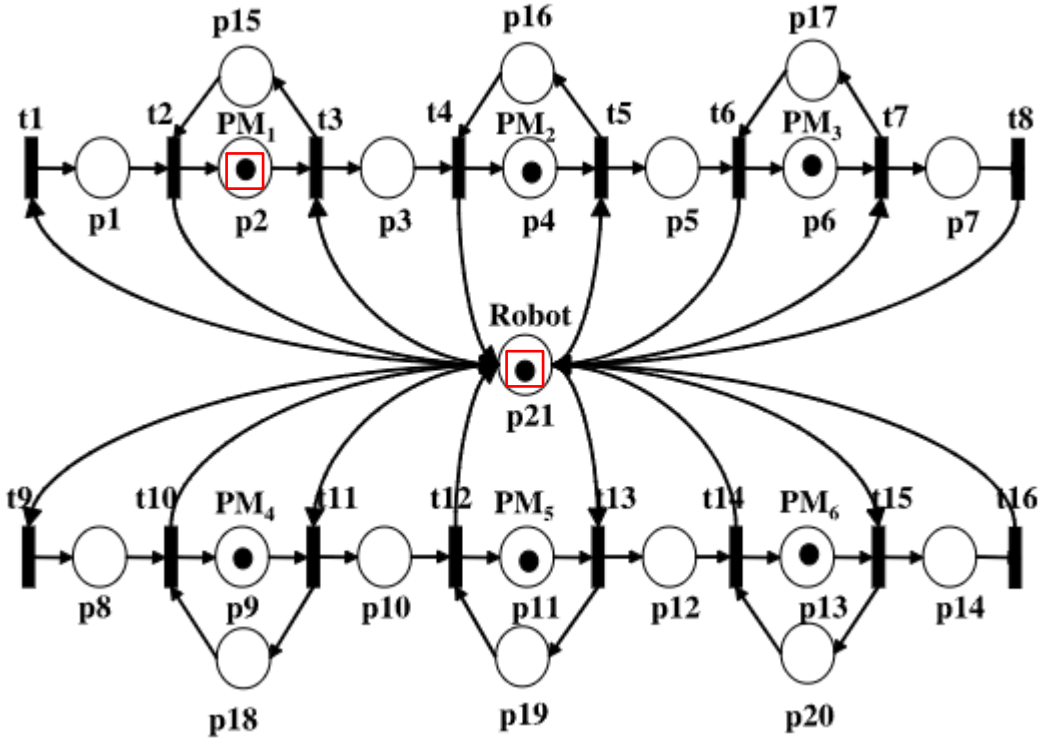


Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ TPN for a Cluster Tool Handling Two Wafer Types

- p2에 있는 token : PM1 에서 Wafer 가 가공중인 상태를 나타냄
- p21에 있는 토큰 : 로봇이 현재 사용 중임

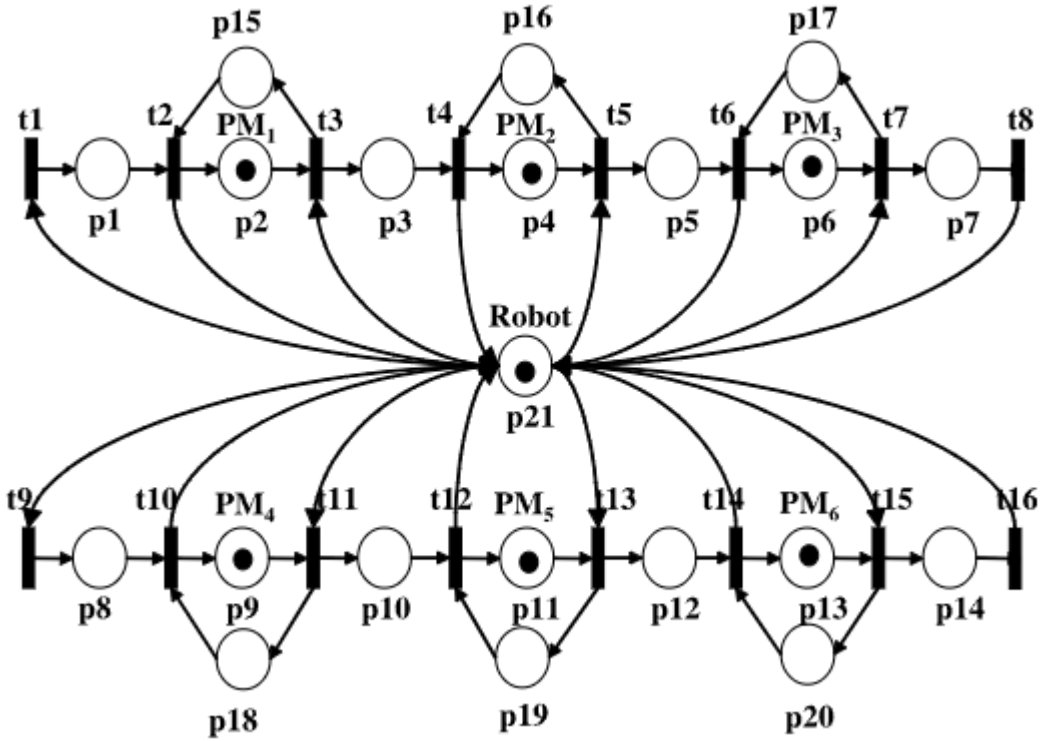


Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ TPN for a Cluster Tool Handling Two Wafer Types

- 시간에 따라 시스템 (a cluster tool for two wafer types) 의 상태를 시뮬레이션 가능

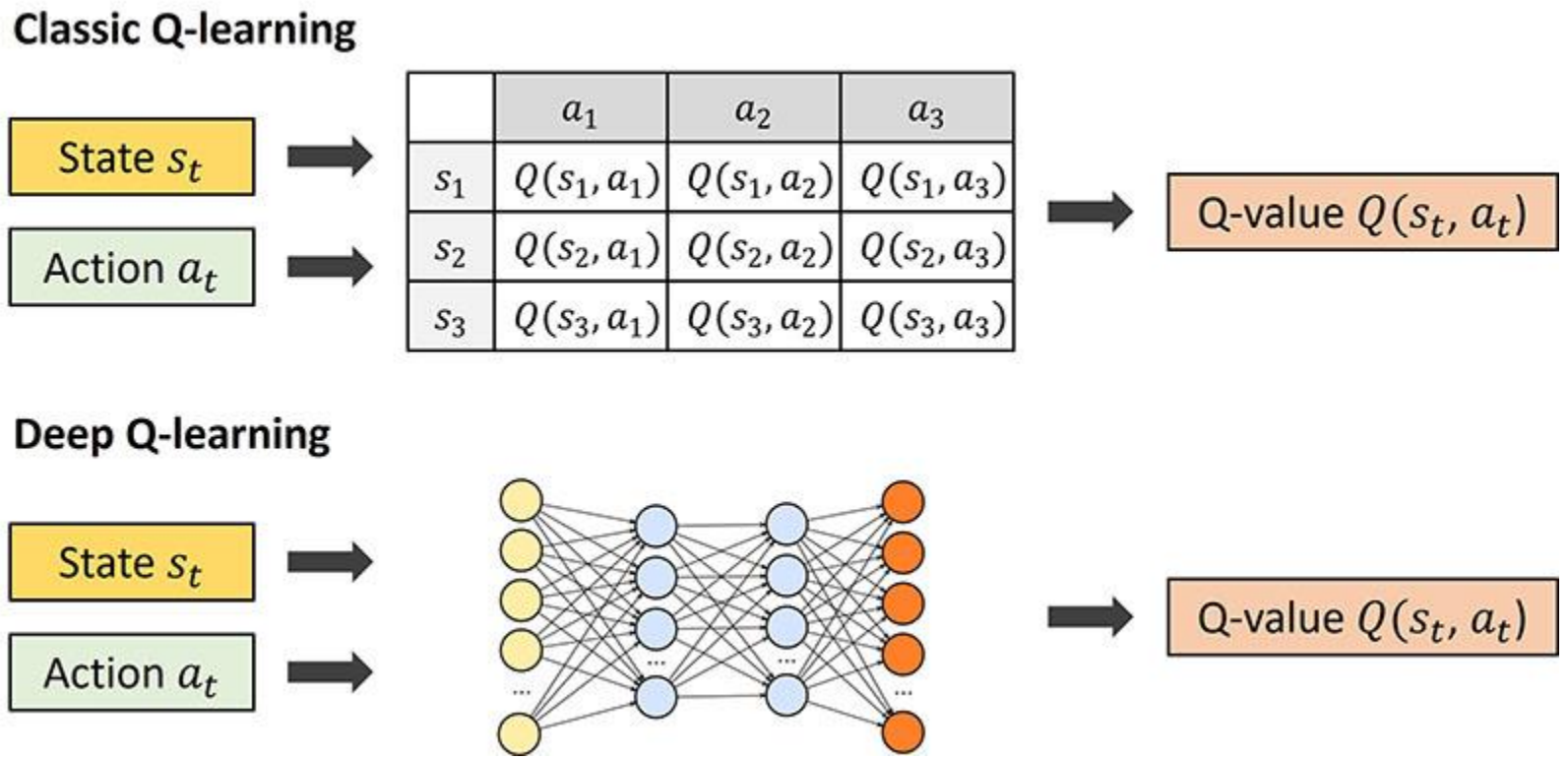


Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ RL Framework

- State-Action Pair의 복잡성으로 인해 Deep Q-Network (DQN) 을 Base model 로 사용
- DQN : Q-value 값을 신경망을 통해 근사하는 방법론



Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ **RL Framework : State**

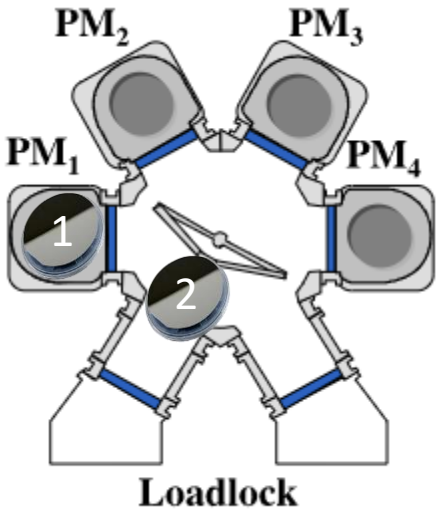
- 장비 상태 (M_h):
  - 각 PM의 상태 (빈 상태, 처리 중, 완료)
- 잔여 처리 시간 비율 (R_h):
  - 각 PM의 잔여 처리 시간 비율 (남은 시간 / 총 처리 시간)
  - 완료 된 경우 완료 후 경과 시간을 음수로 관리
- 로봇 위치 (L_h):
  - 로봇의 현재 위치
- 유휴 시간 (I_h):
  - Bottleneck PM의 평균 유휴 시간
- 작업 부하 비율 (W_h):
  - PM 간 작업 부하 비율

Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ **RL Framework : Action Selection Process**

- 상태 관찰:
  - 현재 상태 관찰 (PM 상태, 로봇 위치 등)
- 가능한 행동 결정:
  - 데드락 방지를 위해 가능한 행동 결정
- 행동 선택:
  - ϵ-greedy 정책 사용
  - $1-\epsilon$ 확률로 최대 Q 값 행동 선택, ϵ 확률로 무작위 선택
- Deadlock 방지
  - Action Masking : 데드락을 유발할 수 있는 행동을 마스킹하여 선택하지 못하게 함



Dead lock status

Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ RL Framework : Reward

- Reward 함수 구성
  - 목적 : PM의 Idle time 이 최소화 되도록
  - 에이전트가 Q 값을 최대화하려고 할 때 유휴 시간을 최소화하도록 유도하기위해 보상을 음수로 설정

$$r_h = -(\text{PM1 유휴 시간} + \text{PM2 유휴 시간})$$

Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

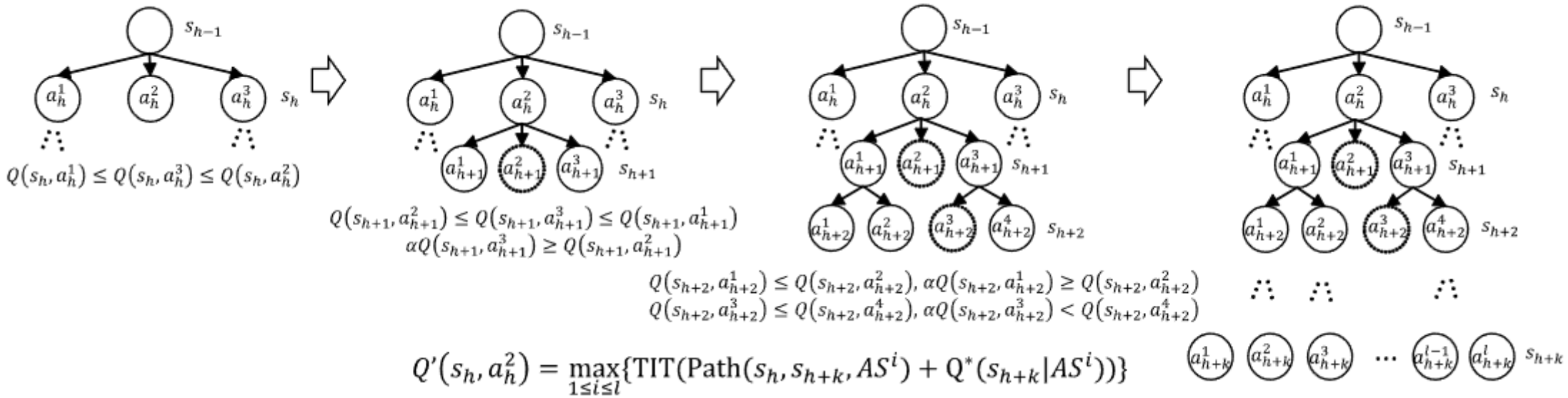
Method

- **RL Framework : Adaptive Search**
  - 추론 시 미래 상태를 미리 탐색하여 행동 결정
  - 최대 Q 값과 두 번째로 큰 Q 값의 차이 고려:
    - 각 상태에서 가능한 행동들에 대해 Q 네트워크를 통해 Q 값을 계산
    - 행동들의 Q 값 중 최대 Q 값과 두 번째로 큰 Q 값의 차이를 평가
    - 만약 이 차이가 작다면, 두 번째로 큰 Q 값을 가지는 행동도 탐색에 포함
    - 이는 새로운 인스턴스에서 최적의 행동을 찾는 데 도움

Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ **RL Framework : Adaptive Search**



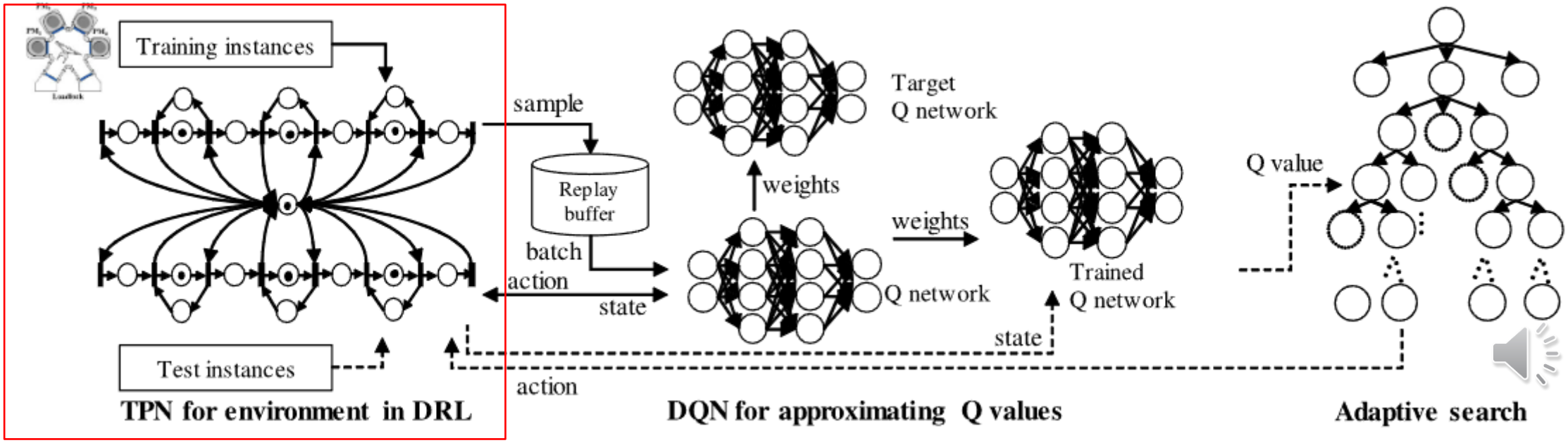
$$Q'(s_h, a_h^2) = \max_{1 \le i \le l} \{\mathrm{TIT}(\mathrm{Path}(s_h, s_{h+k}, AS^i) + \mathrm{Q}^*(s_{h+k}|AS^i))\}$$

Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ RL Framework Review

- RL Environment (TPN)
  - 훈련 인스턴스 생성 (다양한 PM구성, Process time 분포)
  - 상태 관측: TPN 환경에서 로봇은 현재 상태를 관찰하고 가능한 행동을 선택

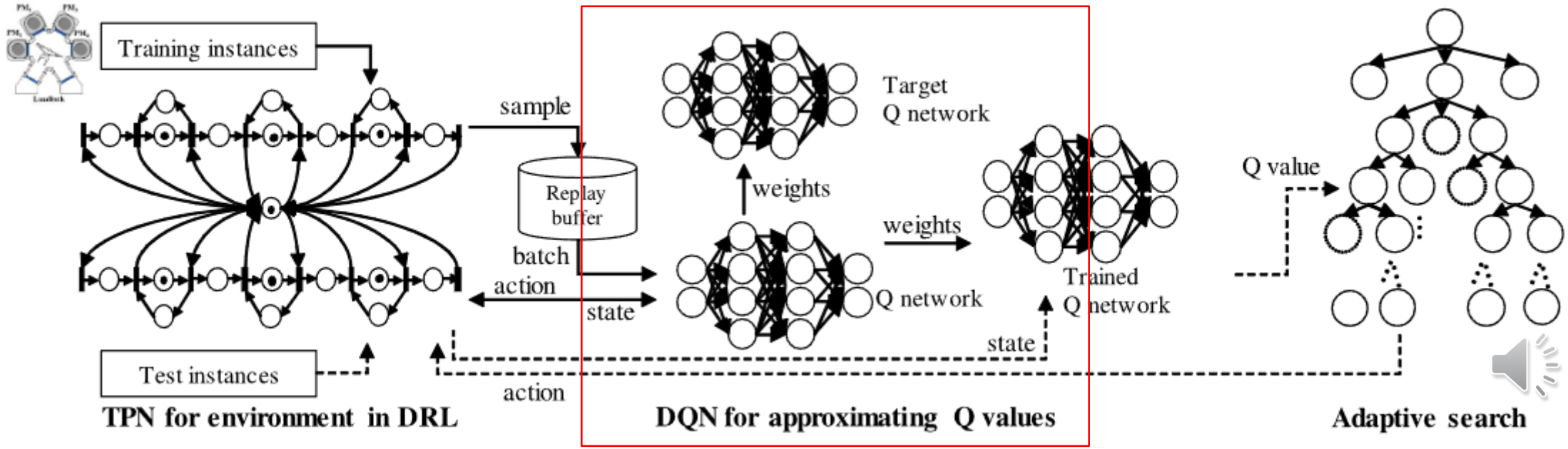


Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ RL Framework Review

- DQN
  - 샘플링 및 리플레이 버퍼
  - Q 네트워크와 타겟 Q 네트워크
  - 가중치 업데이트: 타겟 Q 네트워크는 일정 간격으로 Q 네트워크의 가중치를 받아 업데이트

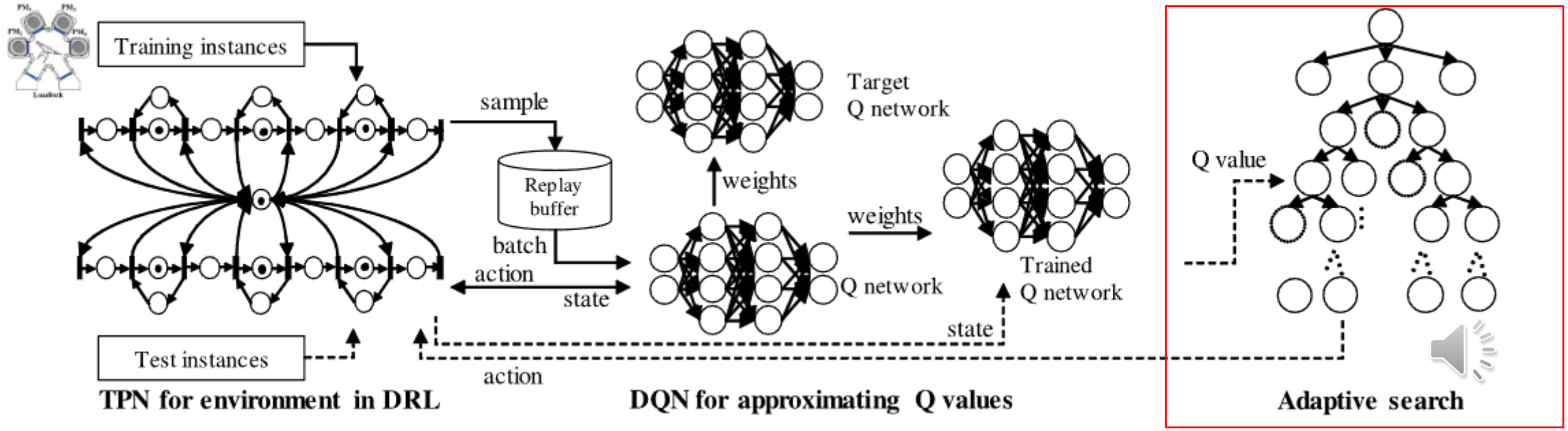


Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Method

❖ RL Framework Review

- Adaptive Search
  - 최대 Q 값과 두 번째로 큰 Q 값의 차이를 고려하여 탐색
  - 여러 경로를 탐색하고 각 경로의 총 유휴 시간을 계산하여 최적의 경로를 선택



Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Experiment

❖ 실험환경

- Training Instance
  - Well-Balanced Instances : 두 Wafer Type의 처리시간이 동일한 범위에서 생성되는 Instance
  - Unbalanced Instances : 두 Wafer Type의 처리시간이 다른 범위에서 생성되는 경우
- PM 구성
  - (2, 2), (3, 3), (2, 3), (3, 2)

    예) (2, 2) : Wafer Type 1은 PM1→PM2 사용 / Wafer Type 2는 PM3→PM4 사용
- Process time range
  - 10-30 부터 80-100 까지 다양한 조합으로 실험
- 비교 모델
  - CBS (Concurrent Backward Sequence)
  - DQN (Deep Q-Network)
  - DQN-LS (DQN with Look-ahead Search)
  - DQN-AS (DQN with Adaptive Search)

Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Experiment

❖ 실험결과 : Well-Balanced Instances

- 성능 Metric : GAP, CT (추론 시간)
- 모든 테스트에서 베이스라인인 CBS 알고리즘 대비 높은 성능
- Adaptive Search 적용 등 으로 인해 평균 추론 시간이 CBS 보다 높으나 실제 사용하는데 문제 없는 수준

$$\text{Gap} = \frac{\text{CBS 생산 웨이퍼 수} - \text{DQN-AS 생산 웨이퍼 수}}{\text{CBS 생산 웨이퍼 수}} \times 100\%$$

EXPERIMENTS WITH WELL-BALANCED INSTANCES

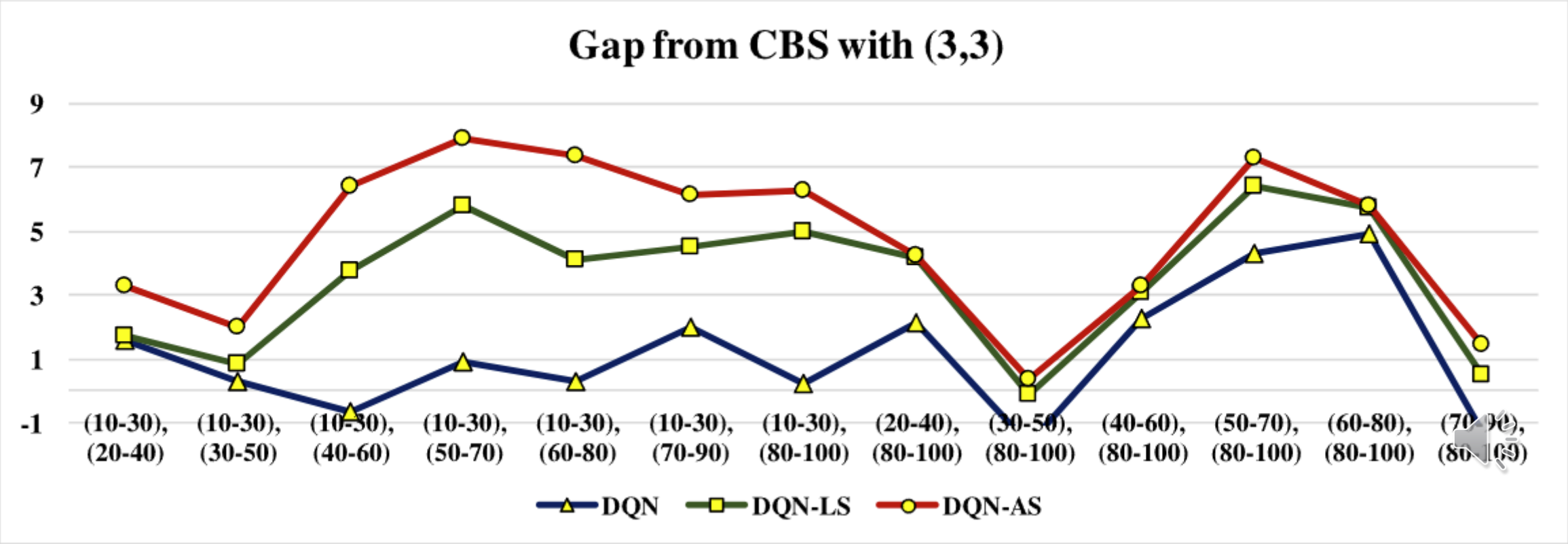
(n_1, n_2)	$(p_l^1 - p_u^1), (p_l^2 - p_u^2)$	CBS vs DQN		CBS vs DQN-LS		CBS vs DQN-AS	
		GAP^{DQN}	CT^{DQN}	GAP^{DQN-LS}	CT^{DQN-LS}	GAP^{DQN-AS}	CT^{DQN-AS}
(2,2)	(10-30), (10-30)	-0.52	0.00	0.82	0.04	2.77	0.33
	(80-100), (80-100)	-0.22	0.00	0.84	0.03	1.10	0.07
	(20-20), (20-20)	4.56	0.00	4.83	0.03	5.10	0.53
	(90-90), (90-90)	0.00	0.00	0.00	0.03	0.00	0.04
(3,3)	(10 30), (10 30)	0.33	0.00	2.62	0.07	4.32	1.59
	(80-100), (80-100)	-0.16	0.00	0.38	0.05	0.48	0.13
	(20-20), (20-20)	5.21	0.00	4.94	0.07	5.21	3.52
	(90-90), (90-90)	0.00	0.00	0.00	0.05	0.00	0.12

Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Experiment

❖ **실험결과 : Unbalanced Instances**

- 이전 실험결과와 유사하게 모든 테스트에서 CBS, DQN-LS 대비 높은 성능



Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search

Conclusion

❖ Contribution

- Single arm cluster tool의 비주기적(Non-cyclic) 스케줄링을 강화학습으로 최적화
- DQN과 Adaptive Search의 결합을 통한 효율성 증대

❖ Limitation

- Single Arm Cluster tool 시나리오에 국한
- 실제 설비 운용 상황과는 다른 시나리오로 연구 수행

요약

- **Neural Combinatorial Optimization**
  - 조합 최적화 문제란?
  - 전통적 방법론의 한계
  - NCO 관련 연구
    - Pointer Networks (2015, NIPS)
    - Neural Combinatorial Optimization with Reinforcement Learning (2017, ICLR)

- **Practical Applications of Neural Combinatorial Optimization**
  - Pathfinding Problem
    - Path Planning using Neural A* Search (2021, ICML)
  - Cluster Tool Scheduling
    - Introduction of Cluster Tools & Scheduling
    - Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search (2024, IEEE Transactions on Automation Science and Engineering)

Reference

[1] Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. Advances in neural information processing systems, 28.

[2] Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940.

[3] Lee, T. E. (2008, December). A review of scheduling theory and methods for semiconductor manufacturing cluster tools. In 2008 Winter simulation conference (pp. 2127-2135). IEEE.

[4] Yonetani, R., Taniai, T., Barekatain, M., Nishimura, M., & Kanezaki, A. (2021, July). Path planning using neural a* search. In International conference on machine learning (pp. 12029-12039). PMLR.

[5] Kim, H. J., & Lee, J. H. (2024). Scheduling Cluster Tools for Concurrent Processing: Deep Reinforcement Learning With Adaptive Search. IEEE Transactions on Automation Science and Engineering.

[6] https://medium.com/sinicx/path-planning-using-neural-a-search-icml-2021-ecc6f2e71b1f

[7] https://icml.cc/media/icml-2021/Slides/9055.pdf

[8] https://www.slideshare.net/gyuhyeonNam/study-pointer-networks

[9] https://ai-com.tistory.com/entry/RL-%EA%B0%95%ED%99%94%ED%95%99%EC%8A%B5-%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98-1-DQN-Deep-Q-Network