POMO: Policy Optimization with Multiple Optima for Reinforcement Learning

24.08.09

김정현

❖ Introduction

- 현실 세계의 조합 최적화 (Combinatorial optimization, CO) 문제는 각기 다른 고유한 제약 조건을 가지고 있음

  → 일반적으로 전문가들이 설계한 휴리스틱 방법으로 처리해왔음

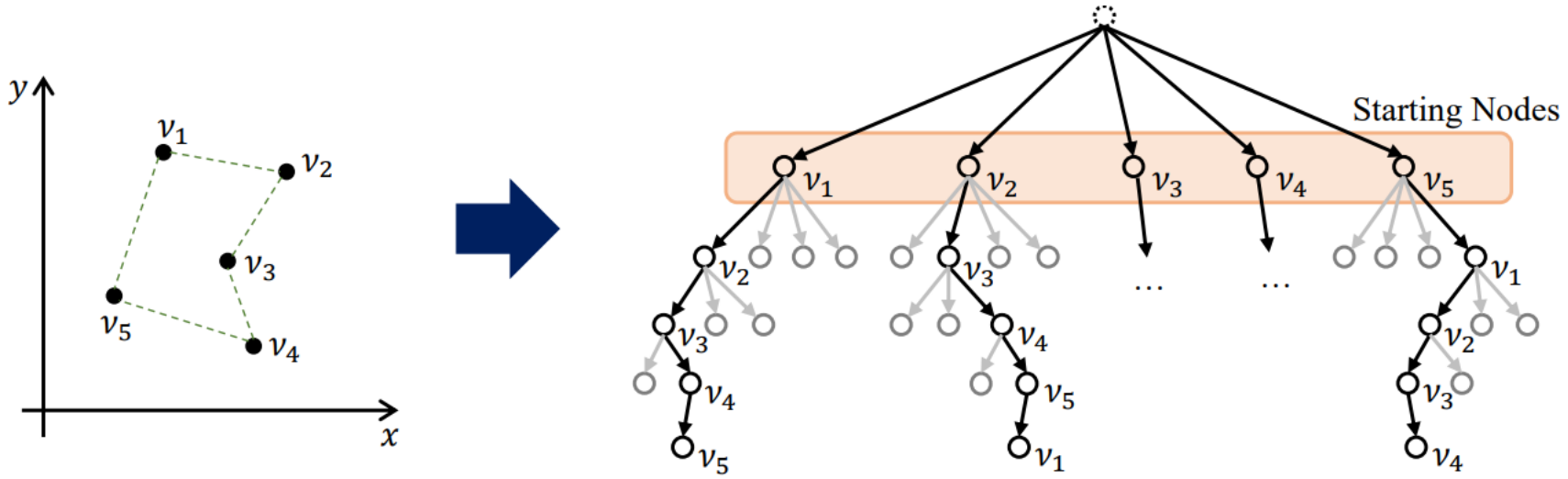
- POMO는 강화학습을 이용하여 CO문제를 해결하였고, TSP, CVRP, KP와 같은 문제에 대한 최적의 솔루션을 우수한 속도로 찾을 수 있었음
- POMO는 CO문제 솔루션의 대칭성을 활용하도록 설계되었음

❖ Contribution

- CO문제를 해결하기 위한 강화학습 방법에서 대칭성을 식별하여 병렬 다중 롤아웃을 통해 신경망 훈련에 활용함
- 다양한 경로 그룹에서 파생된 새로운 낮은 분산 베이스라인을 고안하여 학습의 local minima 취약성을 감소시킴
- 다중 그리디 롤아웃 기반의 효과적인 추론 방법과 CO 문제의 대칭성을 더욱 활용하는 instance augmentation기법을 도입

❖ Motivation

- 기존의 Baseline을 사용하던 방법: 같은 문제를 여러 번 반복하여 얻은 결과의 평균을 사용함
  → Local Optima에 빠지는 문제 발생 (인공지능이 반복해 풀어 얻는 솔루션은 보통 서로 크게 다르지 않기 때문)
- 이를 해결하기 위해 같은 문제를 다양한 각도로 대칭 변형하여 인공지능이 다른 문제로 인식하도록 함
- 많은 CO문제의 해는 노드 시퀀스로 여러 가지 형태를 가질 수 있음
- TSP의 최적해가 τ = (v1,v2,v3,v4,v5)라면, τ' = (v2,v3,v4,v5, v1)도 동일한 최적 솔루션임
- 즉, 어떤 노드를 먼저 출력하는지에 관계없이 동일한 솔루션을 도출하기 위해 대칭을 활용하여 Policy를 학습하도록 함



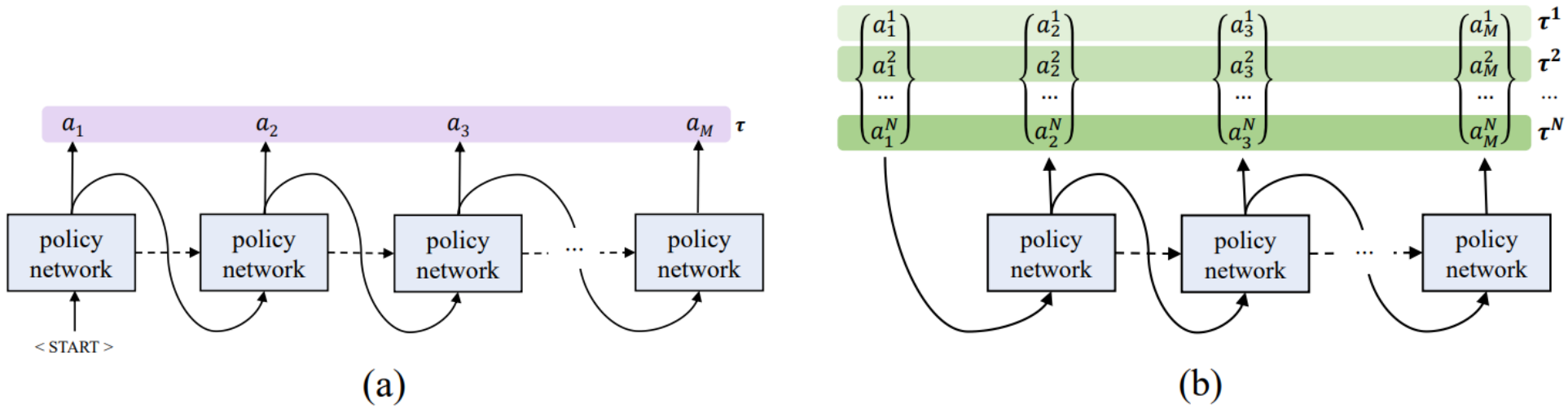
Ⅲ POMO Algorithm - Explorations from multiple starting nodes

- **일반적인 조합최적화를 푸는 방법 – 1개의 솔루션**
  - 어떤 Policy Network에 start token을 넣어 첫번째 action을 뽑고, 이 action이 다음에 어떤 노드를 방문할지 결정함
  - 이전의 action인 output값을 다음 단계의 network에 넣어주고, 이 단계를 반복함

- **POMO 방법 – N개의 솔루션**
  - 서로 다른 starting node를 start token 대신 넣어줌
  - 서로 다른 starting node에서 출발하기 때문에, 각 node에서 시작한 거리를 최소화하도록 구할 수 있음

→ 이러한 접근법은 동일한 문제를 여러 각도에서 반복적으로 바라보도록 함

→ Multiple trajectories를 구하고, 모든 trajectories를 종합하여 baseline을 구하기 때문에 학습 효율을 높일 수 있음



Ⅲ POMO Algorithm - A shared baseline for policy gradients

- POMO는 한번에 n개의 솔루션을 도출하게 됨
- 이 n개의 솔루션에서 Reward를 계산하고, 그 값들의 평균값을 Baseline으로 사용함
- 각 n개의 솔루션의 Reward 가 이 평균값보다 낮은지, 높은지를 통해 평가받음
- N개의 솔루션에 같은 Baseline을 공유하면 local minima를 방지할 수 있음

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}(R(\boldsymbol{\tau}^i) - b^i(s))\nabla_\theta \log p_\theta(\boldsymbol{\tau}^i|s)$$

$$p_\theta(\boldsymbol{\tau}^i|s) \equiv \prod_{t=2}^{M} p_\theta(a_t^i|s, a_{1:t-1}^i)$$

$$b^i(s) = b_{\text{shared}}(s) = \frac{1}{N}\sum_{j=1}^{N} R(\boldsymbol{\tau}^j) \quad \text{for all } \ i.$$

Algorithm 1 POMO Training

procedure TRAINING(training set S, number of starting nodes per sample N, number of training steps T, batch size B)
    initialize policy network parameter θ
    for $step = 1, \ldots, T$ **do**
        $s_i \leftarrow \text{SAMPLEINPUT}(S) \quad \forall i \in \{1, \ldots, B\}$
        $\{\alpha_i^1, \alpha_i^2, \ldots, \alpha_i^N\} \leftarrow \text{SELECTSTARTNODES}(s_i) \quad \forall i \in \{1, \ldots, B\}$
        $\boldsymbol{\tau}_i^j \leftarrow \text{SAMPLEROLLOUT}(\alpha_i^j, s_i, \pi_\theta) \quad \forall i \in \{1, \ldots, B\}, \forall j \in \{1, \ldots, N\}$
        $b_i \leftarrow \frac{1}{N} \sum_{j=1}^{N} R(\boldsymbol{\tau}_i^j) \quad \forall i \in \{1, \ldots, B\}$
        $\nabla_\theta J(\theta) \leftarrow \frac{1}{BN} \sum_{i=1}^{B} \sum_{j=1}^{N} (R(\boldsymbol{\tau}_i^j) - b_i) \nabla_\theta \log p_\theta(\boldsymbol{\tau}_i^j)$
        $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
    end for
end procedure

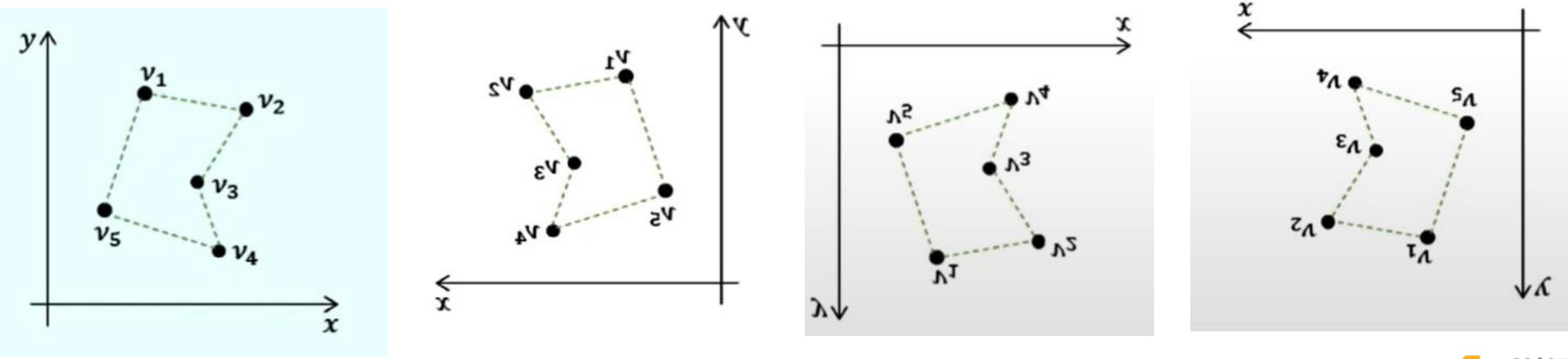
Ⅲ POMO Algorithm - Multiple greedy trajectories for inference

❖ Instance Augmentation

- CO 문제에서의 Inference Mode는 일반적으로 “Greedy Mode”와 “Sampling Mode”가 있음
- POMO는 여러 시작 노드를 사용하여 여러 개의 그리디 경로를 생성할 수 있음 (Multi-Greedy Mode)

  → 그리디 경로의 수가 시작 노드의 수(N)로 제한되어 있음
- 문제를 재구성하여 더 많은 그리디 경로를 생성할 수 있는 방법
- 예를 들어, 2D routing optimization문제에서 모든 노드의 좌표를 뒤집거나 회전시켜 새로운 인스턴스를 생성함

  → 동일한 최적해를 가지면서 더 많은 그리디 경로를 얻을 수 있음



POMO Algorithm - Multiple greedy trajectories for inference

Algorithm 2 POMO Inference

procedure INFERENCE(input s, policy π_θ, number of starting nodes N, number of transforms K)

$\{s_1, s_2, \ldots, s_K\} \leftarrow \text{AUGMENT}(s)$

$\{\alpha_k^1, \alpha_k^2, \ldots, \alpha_k^N\} \leftarrow \text{SELECTSTARTNODES}(s_k) \quad \forall k \in \{1, \ldots, K\}$

$\boldsymbol{\tau}_k^j \leftarrow \text{GREEDYROLLOUT}(\alpha_k^j, s, \pi_\theta) \quad \forall j \in \{1, \ldots, N\}, \forall k \in \{1, \ldots, K\}$

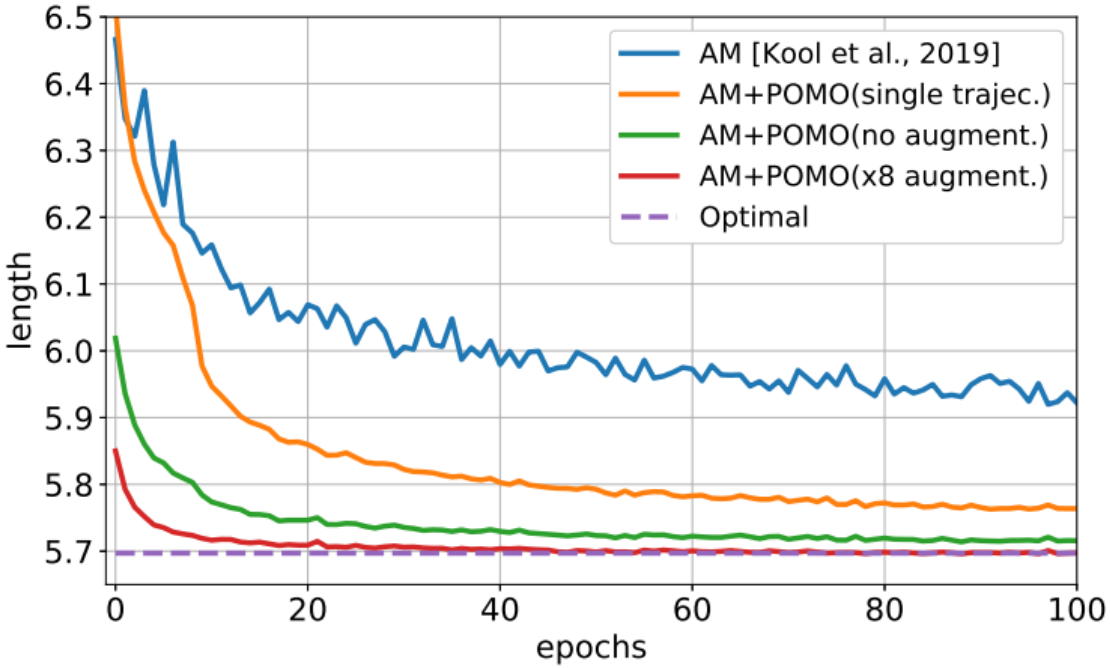
$k_{\max}, j_{\max} \leftarrow \text{argmax}_{k,j}\ R(\boldsymbol{\tau}_k^j)$

return $\boldsymbol{\tau}_{k_{\max}}^{j_{\max}}$

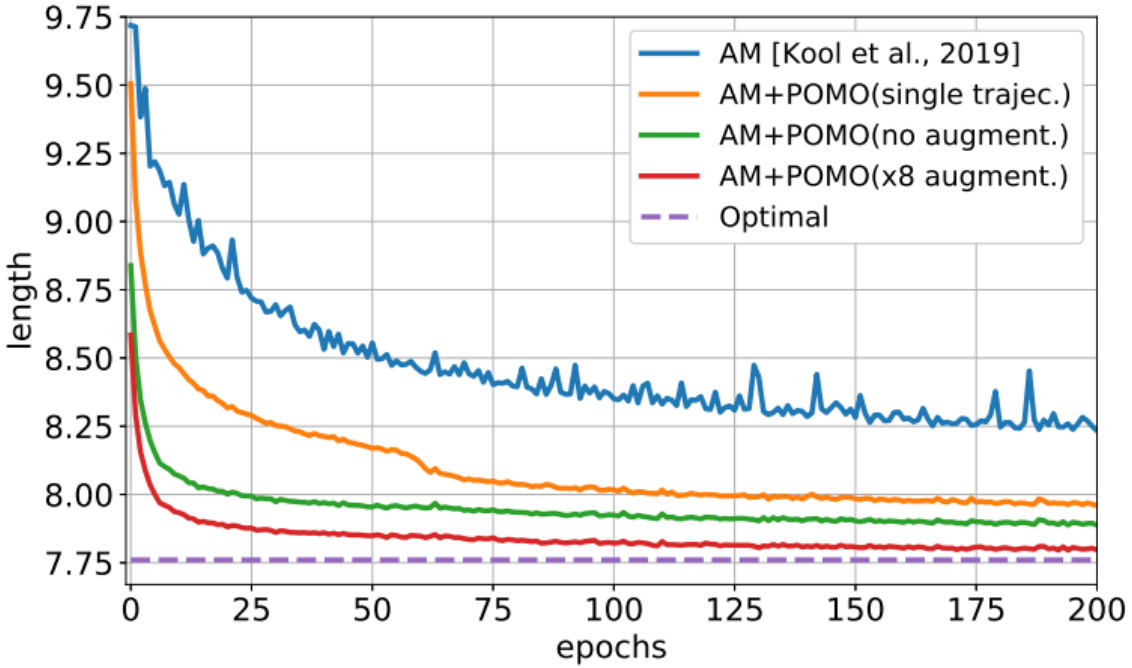
end procedure

❖ Learning Curves

- 10,000개의 랜덤 인스턴스
- 인스턴스 증강을 하고 POMO를 사용한 경우가 가장 학습이 잘되었음



(a) TSP50



(b) TSP100

❖ Experiment Results

Table 2: Experiment results on TSP

Method	TSP20			TSP50			TSP100		
	Len.	Gap	Time	Len.	Gap	Time	Len.	Gap	Time
Concorde	3.83	-	(5m)	5.69	-	(13m)	7.76	-	(1h)
LKH3	3.83	0.00%	(42s)	5.69	0.00%	(6m)	7.76	0.00%	(25m)
Gurobi	3.83	0.00%	(7s)	5.69	0.00%	(2m)	7.76	0.00%	(17m)
OR Tools	3.86	0.94%	(1m)	5.85	2.87%	(5m)	8.06	3.86%	(23m)
Farthest Insertion	3.92	2.36%	(1s)	6.00	5.53%	(2s)	8.35	7.59%	(7s)
GCN [9], beam search	3.83	0.01%	(12m)	5.69	0.01%	(18m)	7.87	1.39%	(40m)
Improv. [11], {5000}	3.83	0.00%	(1h)	5.70	0.20%	(1h)	7.87	1.42%	(2h)
Improv. [12], {2000}	3.83	0.00%	(15m)	5.70	0.12%	(29m)	7.83	0.87%	(41m)
AM [10], greedy	3.84	0.19%	($\ll$1s)	5.76	1.21%	(1s)	8.03	3.51%	(2s)
AM [10], sampling	3.83	0.07%	(1m)	5.71	0.39%	(5m)	7.92	1.98%	(22m)
POMO, single trajec.	3.83	0.12%	($\ll$1s)	5.73	0.64%	(1s)	7.84	1.07%	(2s)
POMO, no augment.	3.83	0.04%	($\ll$1s)	5.70	0.21%	(2s)	7.80	0.46%	(11s)
POMO, $\times$8 augment.	3.83	0.00%	(3s)	5.69	0.03%	(16s)	7.77	0.14%	(1m)

❖ Experiment Results

Table 3: Experiment results on CVRP

Method	CVRP20 Len.	CVRP20 Gap	CVRP20 Time	CVRP50 Len.	CVRP50 Gap	CVRP50 Time	CVRP100 Len.	CVRP100 Gap	CVRP100 Time
LKH3	6.12	-	(2h)	10.38	-	(7h)	15.68	-	(12h)
OR Tools	6.42	4.84%	(2m)	11.22	8.12%	(12m)	17.14	9.34%	(1h)
NeuRewriter [13]	6.16		(22m)	10.51		(18m)	16.10		(1h)
NLNS [14]	6.19		(7m)	10.54		(24m)	15.99		(1h)
L2I [15]	6.12		(12m)	10.35		(17m)	15.57		(24m)
AM [10], greedy	6.40	4.45%	($\ll$1s)	10.93	5.34%	(1s)	16.73	6.72%	(3s)
AM [10], sampling	6.24	1.97%	(3m)	10.59	2.11%	(7m)	16.16	3.09%	(30m)
POMO, single trajec.	6.35	3.72%	($\ll$1s)	10.74	3.52%	(1s)	16.15	3.00%	(3s)
POMO, no augment.	6.17	0.82%	(1s)	10.49	1.14%	(4s)	15.83	0.98%	(19s)
POMO, $\times$8 augment.	6.14	0.21%	(5s)	10.42	0.45%	(26s)	15.73	0.32%	(2m)

❖ Experiment Results

Table 4: Experiment results on KP

Method	KP50		KP100		KP200	
	Score	Gap	Score	Gap	Score	Gap
Optimal	20.127	-	40.460	-	57.605	-
Greedy Heuristics	19.917	0.210	40.225	0.235	57.267	0.338
Pointer Net [7], greedy	19.914	0.213	40.217	0.243	57.271	0.334
AM [10], greedy	19.957	0.173	40.249	0.211	57.280	0.325
POMO, single trajec.	19.997	0.130	40.335	0.125	57.345	0.260
POMO, no augment.	20.120	0.007	40.454	0.006	57.597	0.008

❖ Conclusion

- POMO는 전문가가 설계한 수작업 휴리스틱을 피하고, 심층 강화 학습을 기반으로 하는 데이터 기반의 조합 최적 접근 방식임
- 조합 최적화 문제의 Multiple Optima를 활용하여 훈련과 추론 단계 모두에서 최적해에 효율적으로 도달할 수 있게 함
- TSP, CVRP, KP 모든 문제에서 최적성 갭을 줄이고, 추론 시간도 단축시켜 SOTA를 달성함