Deep Reinforcement Learning for Traveling Purchaser Problems

2025. 08. 25

AI&OPT
김정현

01 TPP vs TSP

❖ 문제 설명

- ✓ TSP (Traveling Salesman Problem)
  - 주어진 모든 도시를 한 번씩 방문하고 시작점으로 돌아오는 최소 거리 경로 탐색
  - 목표: 이동 비용 최소화
  - 의사결정 = 도시 방문 순서

- ✓ **"TPP (Traveling Purchaser Problem)"**
  - 여러 상품을 구매해야 하고, 각 시장에서 가격·재고가 다름
  - 주어진 제품 수요를 충족하기 위해 일부 시장만 방문
  - 목표: 총 비용(이동 비용 + 구매 비용) 최소화
  - 의사결정 = 방문할 시장 순서 + 어떤 시장에서 어떤 상품 구매

- ✓ 특징
  - TSP는 TPP의 서브케이스
  - 만약 각 시장이 모든 상품을 동일 가격·재고로 제공한다면, → TSP와 동일
  - TPP는 TSP보다 복잡한 구조의 문제

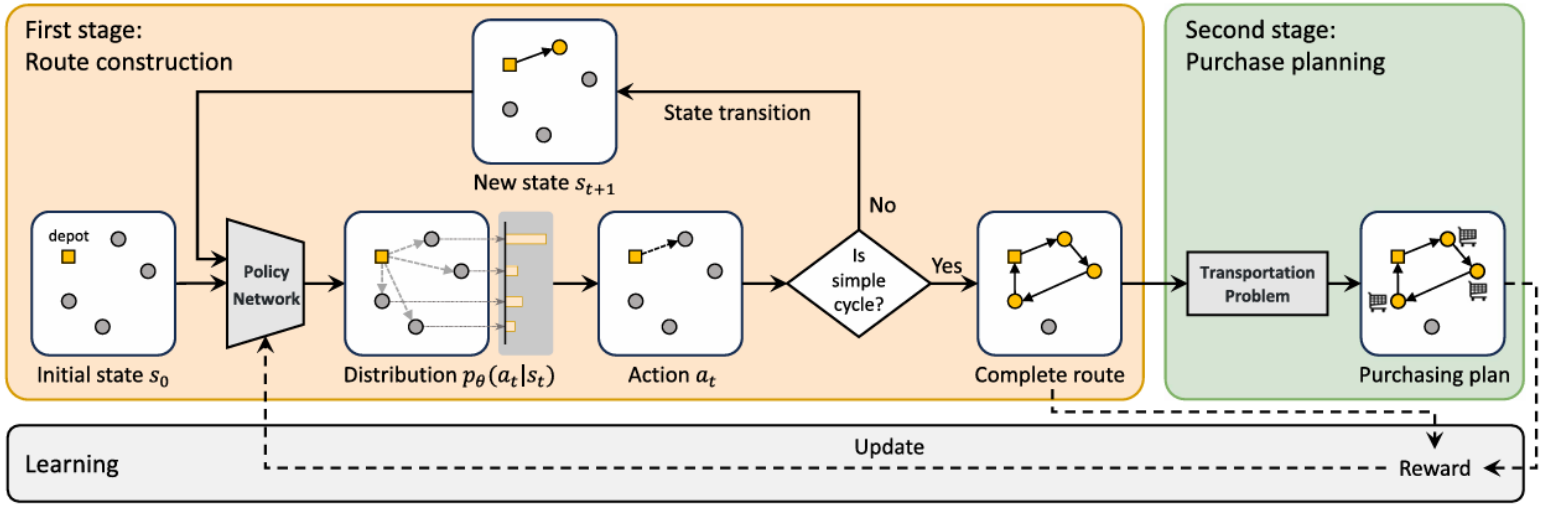
02 Related Work

❖ 기존 접근법의 한계

- ✓ 이동 비용 vs 구매 비용 시장 간의 상충 관계 → 방대한 탐색 공간
- ✓ Exact Method: 현실적인 크기의 문제에서 너무 높은 계산 비용
- ✓ 휴리스틱: 특정 상황에만 효과적이며, 복잡한 설계 및 높은 전문 지식을 요구, 일반화 성능 부족

❖ 본 논문의 핵심 아이디어

- ✓ 두 단계 분리 접근(Solve Separately, Learn Globally)
  - I. 경로 구성: DRL 기반으로 시장 방문 순서 결정
  - II. 구매 계획: 경로가 결정된 후, 선형계획법(LP)으로 최적 구매 계획 도출



03 DRL Approach

❖ DRL 접근법의 주요 구성 요소

✓ 이분 그래프 표현 (Bipartite Graph Representation)

- 시장과 제품 간의 관계를 그래프로 표현
- 시장-제품 간의 공급량, 가격 등 관계 정보를 명확히 포착
- 문제 크기(시장/제품 수)에 독립적인 특징 차원 → 다양한 크기의 문제에 유연하게 적용 가능! (Size-agnostic)

✓ 정책 네트워크 (Policy Network) 설계

- 아키텍처: GNN(Graph Neural Networks) + MHA(Multi-Head Attention) 기반
- 역할: 시장-제품 간의 복잡한 관계 정보(예: 대체 가능성, 보완성)를 효과적으로 추출하여 최적의 경로 결정
- 효율성: 정적 정보(그래프 임베딩)와 동적 정보(부분 경로, 남은 수요) 처리를 분리하여 계산 효율성 극대화

✓ 메타 학습 전략 (Meta-Learning Strategy)

- 훈련 안정성/효율성: 초기 탐색 공간이 큰 대규모 TPP에서 훈련 붕괴(training collapse) 방지
- 일반화 능력(Generalization): 훈련 데이터에 없던 훨씬 큰 규모나 다른 분포의 인스턴스에 대한 'Zero-shot generalization' 가능
- 여러 인스턴스 분포에서 초기화된 정책을 학습 → 새로운 인스턴스에 빠르게 적응 가능

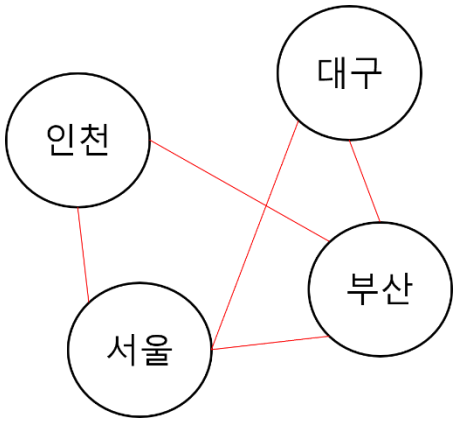
04 Graph vs Bipartite Graph

❖ 그래프(Graph)란?

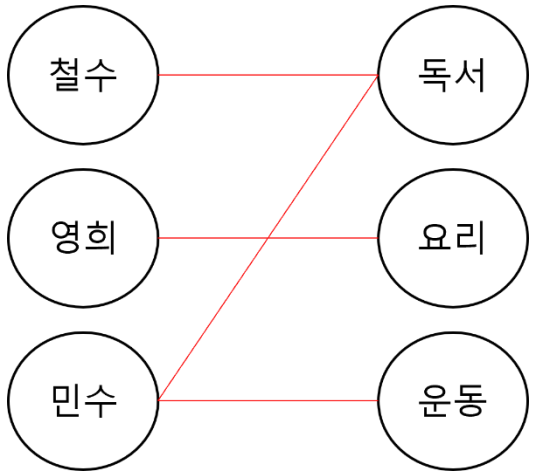
- ✓ 그래프는 점(Nodes/Vertices)들과 이 점들을 잇는 선(Edges/Links)들로 이루어진 구조
  - 점 (Node): 어떤 '개체'를 나타냄 (예: 도시, 사람, 시장, 제품)
  - 선 (Edge): 점들 사이의 '관계'를 나타냄 (예: 도시 간의 도로, 친구 관계, 시장에서 제품 구매 가능 여부)

❖ 이분 그래프(Bipartite Graph)란?

- ✓ 일반 그래프는 노드들이 자유롭게 연결됨
- ✓ 이분 그래프는 두 그룹으로 나뉜 노드 집합만 존재
  - 같은 그룹 안에서는 연결 불가
  - 오직 다른 그룹 노드끼리만 연결
- ✓ 예시:
  - 그룹 A : 사람 (철수, 영희, 민수)
  - 그룹 B : 취미 (독서, 요리, 운동)
  - 연결: 철수 ↔ 독서 / 영희 ↔ 요리 / 민수 ↔ 운동, 독서



<일반 그래프>

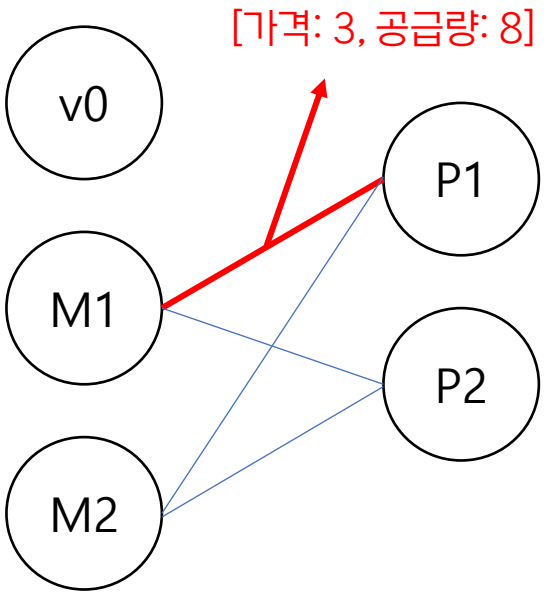


<이분 그래프>

04 Graph vs Bipartite Graph

❖ TPP를 위한 이분 그래프 구성

- ✓ 시장 : M1, M2 (총 2개 시장 + depot (출발/도착 지점 v0)
- ✓ 제품 : P1, P2 (총 2개 제품)
- ✓ 수요량: P1 = 10개, P2 = 5개
- ✓ 시장별 공급 정보:
  - M1: P1을 8개 공급 (가격 3), P2를 3개 공급 (가격 4)
  - M2: P1을 5개 공급 (가격 2), P2를 4개 공급 (가격 5)
- ✓ 노드 구성:
  - 그룹 A: 시장 노드 (M1, M2, v0)
    - 노드 특징: 각 시장의 위치 좌표, depot 여부
  - 그룹 B : 제품 노드 (P1, P2)
    - 노드 특징: 각 제품의 총 수요량
- ✓ 엣지 (시장-제품 관계):
  - 연결 조건: 시장에서 특정 제품을 구매할 수 있다면 연결
  - 엣지 특징 :
    - 해당 시장에서 해당 제품을 구매할 수 있는 가격 / 해당 시장에서 해당 제품의 최대 공급량
    - 예: M1 ↔ P1 : [가격: 3, 공급량: 8] / M1 ↔ P2 : [가격: 4, 공급량: 3]



04 Graph vs Bipartite Graph

❖ **TPP를 위한 이분 그래프 구성**

- ✓ 시장 노드 : M1, M2, v0 (depot)
- ✓ 제품 : P1, P2

	X	Y
v0	0	0
M1	10	20
M2	30	5

시장 노드 피처

	d
P1	10
P2	5

제품 노드 피처

	P1	P2
v0	0	0
M1	1	1
M2	1	0

이분 인접 행렬

	p	q
e_{11}	3	8
e_{12}	4	3
e_{21}	2	5

엣지 피처

- ✓ 제품 노드 임베딩 g0_k
- ✓ 시장 노드 임베딩 h0_m

05 Policy Network

❖ 정책 네트워크 구조

- ✓ Input Embedding Module : 정적 TPP 인스턴스(이분 그래프)를 임베딩
- ✓ Market Encoder : 시장 노드 간의 관계를 심층적으로 인코딩
- ✓ Decoder : 현재 상태와 인코딩된 정보를 바탕으로 다음 시장을 선택

❖ Input Embedding Module

- ✓ TPP 인스턴스의 정적 정보, 즉 이분 그래프 표현을 받아 신경망이 처리할 수 있는 고차원 임베딩으로 변환하는 역할
- ✓ 시장/제품 특징과 그래프 연결 구조를 바탕으로, 각 시장 노드와 제품 노드에 대한 의미 있는 벡터를 생성
- ✓ 세부과정:

I. 초기 선형 임베딩: 각 시장 노드, 제품 노드, 엣지의 원시 특징(위치, 수요량, 가격, 공급량 등)을 고정 차원의 벡터로 변환

II. 두 단계 메시지 전달 (GNN):

- 1단계 (제품 노드): 각 제품 노드는 연결된 시장 노드들로부터 자신의 공급 관련 정보를 취합하여 임베딩 업데이트 ${g^0}_k$
- 2단계 (시장 노드): 각 시장 노드는 연결된 제품 노드들로부터 자신의 제품 공급 및 다른 시장과의 관계 정보를 취합하여 임베딩 업데이트 ${h^0}_m$

05 Policy Network

❖ 정책 네트워크 구조

- ✓ Input Embedding Module : 정적 TPP 인스턴스(이분 그래프)를 임베딩
- ✓ Market Encoder : 시장 노드 간의 관계를 심층적으로 인코딩
- ✓ Decoder : 현재 상태와 인코딩된 정보를 바탕으로 다음 시장을 선택

❖ Market Encoder

- ✓ 입력 임베딩 모듈에서 나온 시장 노드 임베딩 ${h^0}_m$ 를 더욱 심층적으로 처리
- ✓ 시장 노드들 간의 관계를 깊게 학습하여, 각 시장이 전체 시장 네트워크 내에서 어떤 '역할'을 하는지 파악
- ✓ 세부과정 (Transformer Architecture 기반) :
  - I. MHA (Multi-Head Attention) Layer : 각 시장 노드 m 은 다른 모든 시장 노드들 n 과의 관계를 학습
  - II. Node-wise MLP Feed-Forward Layer : 어텐션으로 취합된 정보를 각 노드별로 추가 처리
  - III. 잔차 연결(Residual Connection) 및 배치 정규화(Batch Normalization) 적용
- ✓ 결과: ${h^N}_m$ (최종 시장 노드 임베딩)
- ✓ 이들의 평균을 취한 전역 임베딩 h_m도 생성됨

05 Policy Network

❖ Decoder

- ✓ 디코더는 경로가 완성될 때까지 반복적으로 실행됨
- ✓ 현재까지의 경로, 남은 제품 수요, 그리고 TPP 인스턴스 전체의 정보(h_M)를 종합하여 다음에 방문할 최적의 시장을 선택
- ✓ 세부과정 :

I. 디코딩 컨텍스트 (h_d) 생성:

- 글로벌 임베딩(h_M) : 시장 인코더에서 얻은 TPP 인스턴스 전체의 고수준 정보
- 수요 컨텍스트($g^t{}_K$) : 현재 시점에서 남은 제품 수요량과 제품 노드 임베딩을 가중 합한 벡터
- 경로 컨텍스트($o^t{}_\pi$) : 지금까지 방문한 부분 경로의 정보를 담은 LSTM(Long Short-Term Memory) 벡터

II. 윈-투 매니 어텐션:

- 디코딩 컨텍스트 (h_d)를 쿼리로 사용, 시장 인코더에서 나온 최종 시장 노드 임베딩 $h^N{}_m$ 들을 키와 밸류으로 사용
- 마스킹: 제약 조건을 준수하지 않는 경우는 스코어를 $-\infty$로 설정

III. Softmax를 통한 확률 분포 생성 $p_\theta(a_t|s_t)$

IV. 행동 선택:

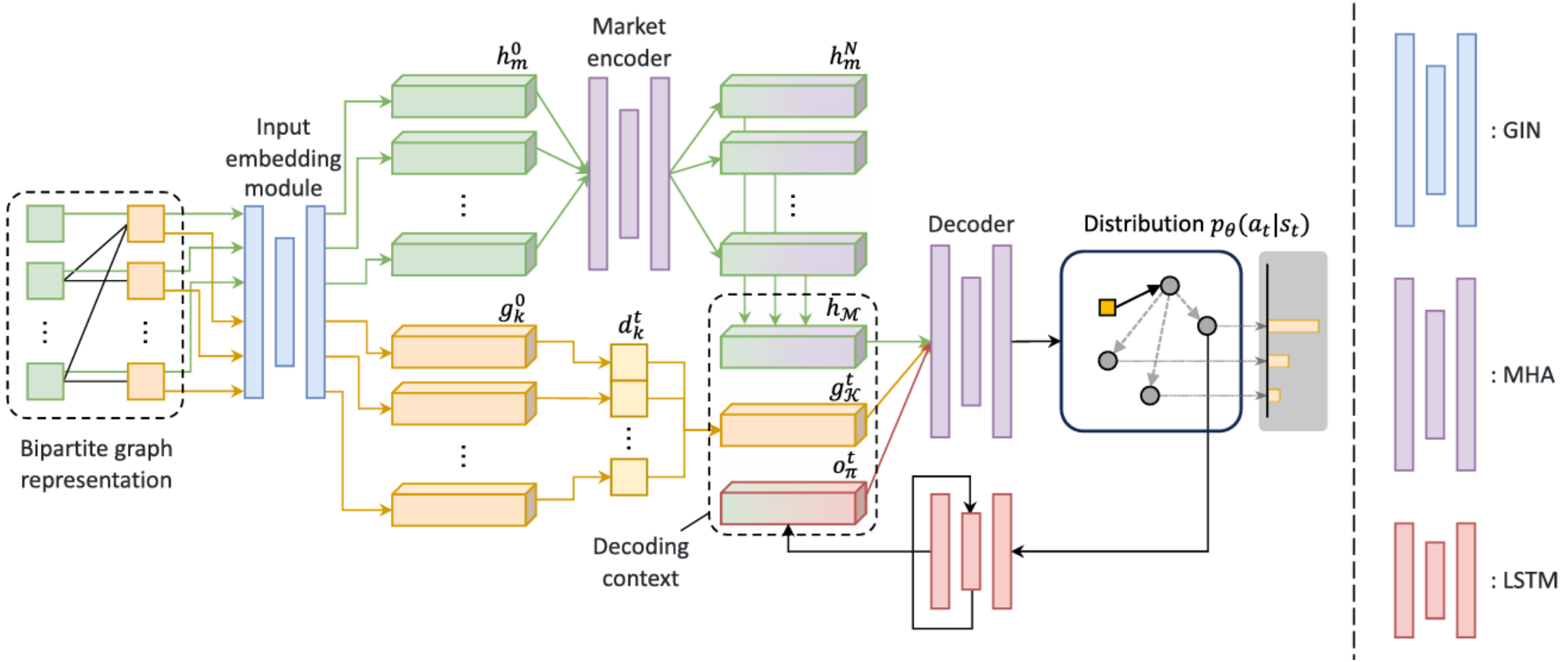
- 훈련 시: 이 확률 분포에서 다음 시장을 샘플링(sampling)하여 탐색(exploration)을 유도
- 추론 시: 가장 높은 확률을 가진 시장을 greedy 선택하여 최적의 경로를 찾음

V. 상태 업데이트:

- 선택된 시장이 부분 경로에 추가되고, 남은 수요량 d_k가 업데이트됨

05 Policy Network

❖ Architecture



06 Training Strategy

❖ DRL 학습 과정

Algorithm 1: REINFORCE Algorithm With Greedy Rollout Baseline.

Input: Initial policy network parameter θ, TPP instance distribution $\mathcal{P}$,
number of epochs E, batch size B, steps per epoch T, learning rate ϵ, significance α.
Output: The learned policy network parameter θ.
Initialize baseline network parameter $\theta^{\mathrm{BL}} \leftarrow \theta$;
for $e = 1, \ldots, E$ **do**
  for $t = 1, \ldots, T$ **do**
    Generate B instances randomly from $\mathcal{P}$;
    Sample route $\pi_i \sim p_\theta(\pi_i|U_i),\ i \in \{1, \ldots, B\}$;
    Greedy rollout $b(U_i)$ from $p_{\theta^{\mathrm{BL}}},\ i \in \{1, \ldots, B\}$; // compute baseline
    $\nabla\mathcal{L} \leftarrow \sum_{i=1}^{B} \left(L\left(\pi_i|U_i\right) - b(U_i)\right) \nabla_\theta \log p_\theta\left(\pi_i|U_i\right)$; // get gradient
    $\theta \leftarrow \mathrm{Adam}\left(\theta, \nabla\mathcal{L}\right)$; // update policy network
  end
  if $\mathrm{OneSidedPairedTTest}\left(p_\theta, p_{\theta^{BL}}\right) < \alpha$ **then**
    $\theta^{\mathrm{BL}} \leftarrow \theta$; // update baseline network
  end
end

07 Meta-Learning Strategy

❖ 메타 학습 필요성

- ✓ 작은 규모의 TPP 인스턴스 → 기존 DRL 학습 과정으로도 잘 작동
- ✓ 대규모 인스턴스 → 무작위 초기화된 정책망이 합리적 해 탐색 실패, 학습의 불안정, training collapse 발생
- ✓ 특정 분포에서만 훈련된 정책망은 다른 분포/규모로 전이 시 성능 저하
- ✓ 따라서 다양한 TPP 인스턴스 분포 (시장 수/제품 수)를 경험하며 학습하는 전략 필요

❖ 메타 학습 절차

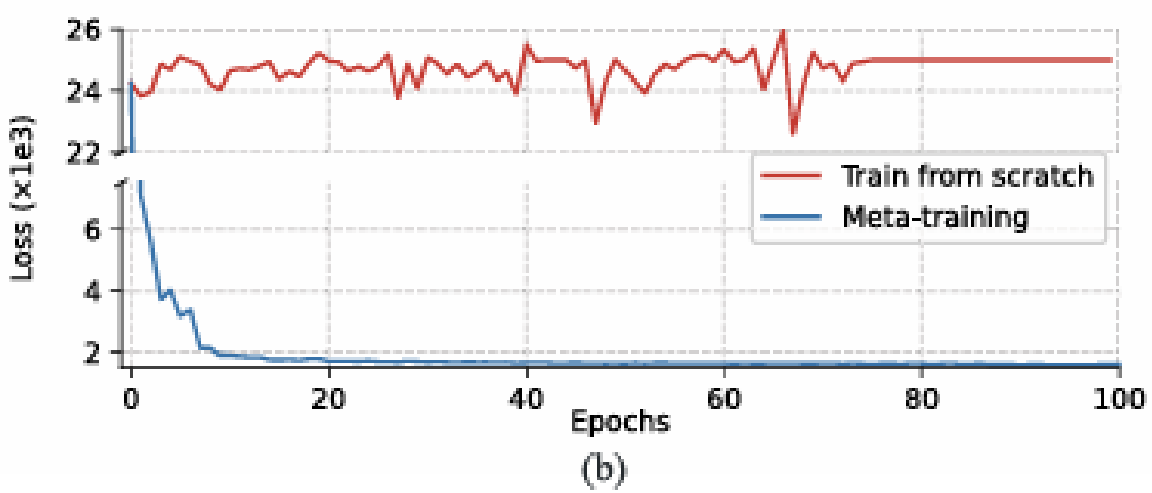
I. 메타 정책 네트워크 θ 초기화:
- θ를 무작위로 초기화

II. 문제 유형 P_{in} 선택 :
- TPP 문제 집합 D_{TPP}에서 무작위 선택

III. 임시 정책 네트워크 θ_{in} 생성:
- θ를 복사하여 θ_{in} 생성

IV. 특정 P_{in} 문제 유형에 맞춰 훈련:
- θ_{in}를 학습 알고리즘(REINFORCE with Baseline)을 사용해서 N번 업데이트

V. 메타 정책 네트워크 θ 업데이트:
- θ를 θ_{in} 방향으로 업데이트



08 Result

❖ **실험 결과**

- ✓ U-TPP (Unrestricted Traveling Purchaser Problem): 무제한 공급 여행 구매자 문제
- ✓ R-TPP (Restricted Traveling Purchaser Problem): 제한된 공급 여행 구매자 문제

TABLE II
RESULTS ON SYNTHETIC U-TPP INSTANCES

Instance		GSH + TRH		CAH + TRH		RL - E2E		RL + TRH	
$\|M\|$	$\|K\|$	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
50	50	2221	0.006	1910	0.017	1897	0.017	**1857**	0.024
50	100	2750	0.008	2552	0.033	2542	0.025	**2446**	0.033
100	50	2050	0.011	1571	0.033	1563	0.020	**1524**	0.027
100	100	2542	0.016	2185	0.072	2111	0.027	**2044**	0.036

TABLE III
RESULTS ON SYNTHETIC R-TPP INSTANCES

Instance			GSH + TRH		CAH + TRH		RL - E2E		RL + TRH	
$\|M\|$	$\|K\|$	λ	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
50	50	0.99	2152	0.016	2257	0.073	2032	0.020	**1954**	0.030
50	100	0.99	2671	0.032	2863	0.161	2567	0.026	**2466**	0.042
100	50	0.99	2062	0.058	2174	0.243	1753	0.029	**1711**	0.043
100	100	0.99	2578	0.142	2853	0.704	2302	0.033	**2235**	0.053
50	50	0.95	2845	0.044	2914	0.124	2645	0.025	**2594**	0.036
50	100	0.95	3569	0.095	3709	0.243	3457	0.034	**3368**	0.059
100	50	0.95	3384	0.300	3440	0.598	3027	0.040	**2980**	0.073
100	100	0.95	4281	0.696	4405	1.499	3993	0.053	**3920**	0.111
50	50	0.9	3910	0.099	3965	0.201	3704	0.033	**3644**	0.052
50	100	0.9	5080	0.195	5137	0.350	4927	0.043	**4855**	0.080
100	50	0.9	5293	0.789	5310	1.218	4956	0.060	**4900**	0.124
100	100	0.9	6963	1.666	7014	2.637	6672	0.073	**6660**	0.186

08 Result

❖ 실험 결과

✓ Opt 값은 TPPLIB 벤치마크의 웹사이트에서 제공된 각 인스턴스의 최적해 또는 현재까지 알려진 가장 좋은 솔루션 값

TABLE IV
RESULTS ON TPPLIB BENCHMARK INSTANCES

Instance	Opt.		GSH + TRH			CAH + TRH			RL - E2E			RL + TRH		
	Obj.	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
EEuclideo.50.50	1482	4	1779	17.20%	0.007	1643	9.97%	0.015	1524	2.63%	0.014	**1497**	**1.06%**	0.019
EEuclideo.50.100	2417	6	2652	9.74%	0.008	2726	13.82%	0.032	2588	7.01%	0.025	**2446**	**1.03%**	0.037
EEuclideo.100.50	1655	72	1979	24.89%	0.009	1796	9.45%	0.031	1701	2.96%	0.019	**1688**	**2.30%**	0.027
EEuclideo.100.100	2085	183	2388	15.47%	0.013	2251	8.65%	0.075	2220	7.09%	0.025	**2146**	**2.99%**	0.064
CapEuclideo.50.50.99	1862	6	2189	20.86%	0.017	2243	23.72%	0.089	2047	9.09%	0.020	**1929**	**3.51%**	0.027
CapEuclideo.50.100.99	2313	7	2578	12.30%	0.031	2710	18.23%	0.160	2483	7.18%	0.025	**2394**	**3.33%**	0.033
CapEuclideo.100.50.99	1504	58	1951	29.97%	0.061	1988	35.73%	0.179	1561	3.91%	0.025	**1531**	**1.84%**	0.034
CapEuclideo.100.100.99	1865	134	2283	21.76%	0.118	2406	27.95%	0.686	1955	4.86%	0.028	**1914**	**2.79%**	0.039
CapEuclideo.50.50.95	2444	10	2904	21.16%	0.036	2751	15.61%	0.104	2643	7.40%	0.028	**2581**	**5.09%**	0.040
CapEuclideo.50.100.95	3187	23	3441	7.97%	0.072	3672	15.75%	0.234	3421	7.35%	0.036	**3299**	**3.56%**	0.054
CapEuclideo.100.50.95	2860	466	3144	9.85%	0.199	3221	12.73%	0.629	3026	5.93%	0.039	**2962**	**3.66%**	0.063
CapEuclideo.100.100.95	3555	1178	3991	12.31%	0.521	4096	15.24%	1.249	3769	5.94%	0.049	**3664**	**3.00%**	0.094
CapEuclideo.50.50.9	3571	28	3927	10.05%	0.061	3873	8.49%	0.145	3744	4.73%	0.036	**3673**	**2.81%**	0.053
CapEuclideo.50.100.9	4668	30	4961	6.33%	0.128	5046	8.07%	0.289	4876	4.47%	0.043	**4834**	**3.57%**	0.067
CapEuclideo.100.50.9	4674	243	4981	6.64%	0.439	5106	9.26%	0.999	4891	4.66%	0.053	**4825**	**3.25%**	0.097
CapEuclideo.100.100.9	6442	537	6961	8.11%	1.668	6850	6.43%	2.307	6637	3.01%	0.070	**6534**	**1.42%**	0.156
Average	2912	187	3257	14.66%	0.212	3274	14.94%	0.451	3068	5.51%	0.033	**2995**	**2.83%**	0.057

08 Result

❖ **실험 결과**

✓ 대규모 인스턴스에 대한 일반화 능력 (Zero-shot Generalization)

- 학습 인스턴스: M, K가 최대 (100, 100)인 인스턴스에서 메타 학습 전략을 사용하여 정책 네트워크를 훈련
- 평가 인스턴스: 더 큰 인스턴스 ((150, 150), (200, 200), (300, 300))에 대해 미세 조정 없이 적용하여 성능 평가

TABLE V
ZERO-SHOT GENERALIZATION ON LARGER-SIZED INSTANCES

Instance			GSH + TRH		CAH + TRH		RL - E2E		RL + TRH	
$\|M\|$	$\|K\|$	λ	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
150	150	/	2672	0.027	2460	0.190	2514	0.028	**2380**	0.075
200	200	/	2885	0.034	2454	0.259	2485	0.040	**2262**	0.142
300	300	/	3445	0.041	**3074**	0.373	3206	0.045	3088	0.198
150	150	0.99	2918	0.265	3056	1.202	2576	0.039	**2519**	0.063
150	150	0.95	**5642**	1.318	5884	1.862	6133	0.074	6014	0.150
150	150	0.9	**10008**	3.027	10199	4.908	10707	0.095	10155	0.205

09 Conclusion

❖ **결론 및 향후 연구**

- ✓ 경로 (Routing)와 구매 (Purchasing) 결정이 강하게 연동된 복잡한 NP-hard 문제인 Traveling Purchaser Problem (TPP)
- ✓ 기존 해법들은 계산 비용 높거나 일반화에 한계
- ✓ 제안: "따로 풀고, 전체를 학습(Solve Separately, Learn Globally)"하는 DRL (Deep Reinforcement Learning) 기반 프레임워크 제시