

Memory-Enhanced Neural Solvers for Routing Problems

26. 02.06

노정빈



❖ 경로 최적화 문제 (Routing Problems)

- ✓ 외판원 문제(TSP), 차량 경로 문제(CVRP) 등
- ✓ 물류/운송 분야의 핵심 CO 문제
- ✓ 노드 수가 증가할수록 해 공간이 기하급수적으로 커짐
- ✓ 현실적 시간 내 최적해 보장이 매우 어려움

❖ 신경망 기반 솔버 (Neural Solvers)

- ✓ 경로를 step-by-step으로 한 번에 한 노드씩 구성하는 정책 기반 방법
- ✓ 매 step에서 후보 노드에 대한 점수/확률(Logits)을 출력해 다음 선택 수행
- ✓ 대표 모델: Attention Model, POMO 등
- ✓ 한 번의 시도만으로 좋은 해를 찾기 어렵기 때문에, 주어진 예산(budget) 내에서 여러 번 반복하여 best 해를 선택

❖ 기존 연구의 한계

- ✓ 반복된 시도에서 얻은 경험을 다음 시도에 재사용하지 못함 → 주어진 예산 내 성능 개선이 비효율적
- ✓ 이를 해결하기 위한 Back-propagation 기반의 방식은 비싸고 느림

❖ MEMENTO

✓ 핵심 아이디어

- 문제를 푸는 도중 얻은 경험을 메모리에 기록하고, 다음 시도에서 이를 참고하여 행동 확률(Logits)을 보정하자

✓ 설계 목표

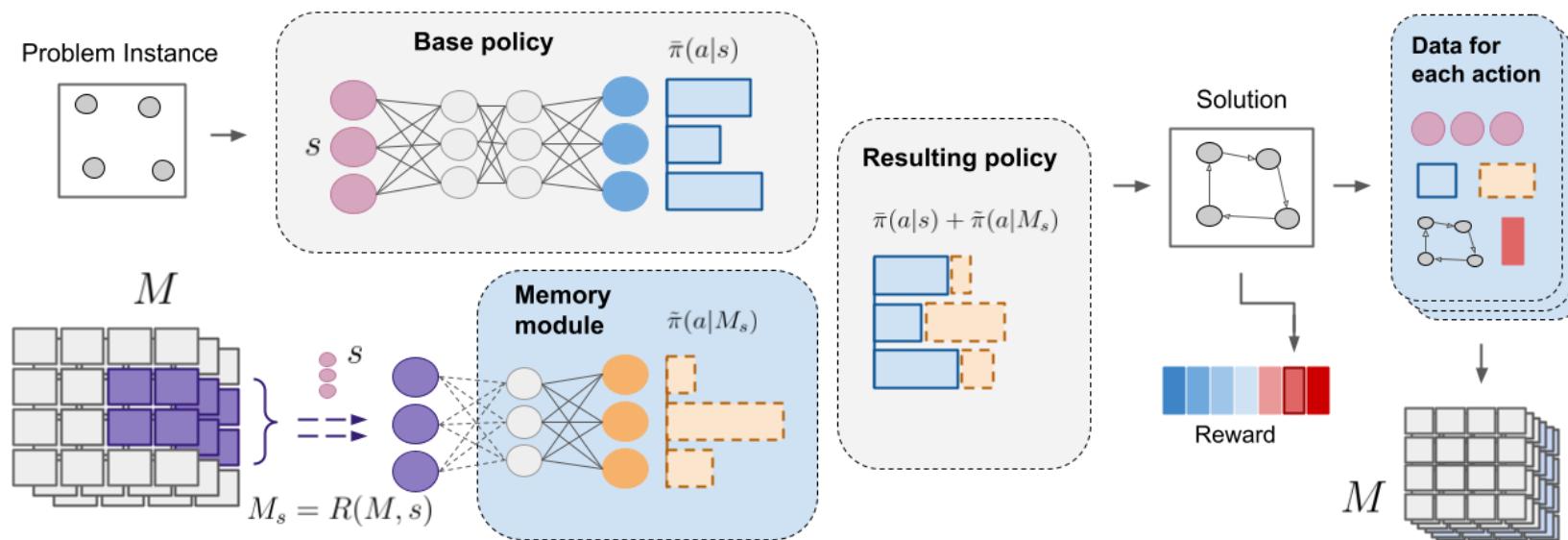
- 학습 가능: 학습을 통해 과거 경험 활용법 습득
- 경량: 추론 시간 오버헤드 최소화
- 아키텍처 독립: 특정 네트워크 구조에 의존하지 않는 plug-in 방식 → 다양한 뉴럴 솔버에 적용할 수 있음
- 사전 학습된 정책 활용: 기존 정책 위에 쉽게 추가 가능

✓ 특징

- 가중치 업데이트 없이 메모리 기반 보정으로 출력 분포만 적응
- 메모리를 누적하며 logits 보정을 통해 점진적으로 성능 향상
- 남은 예산을 사용해 탐색-활용 균형을 자동 조절

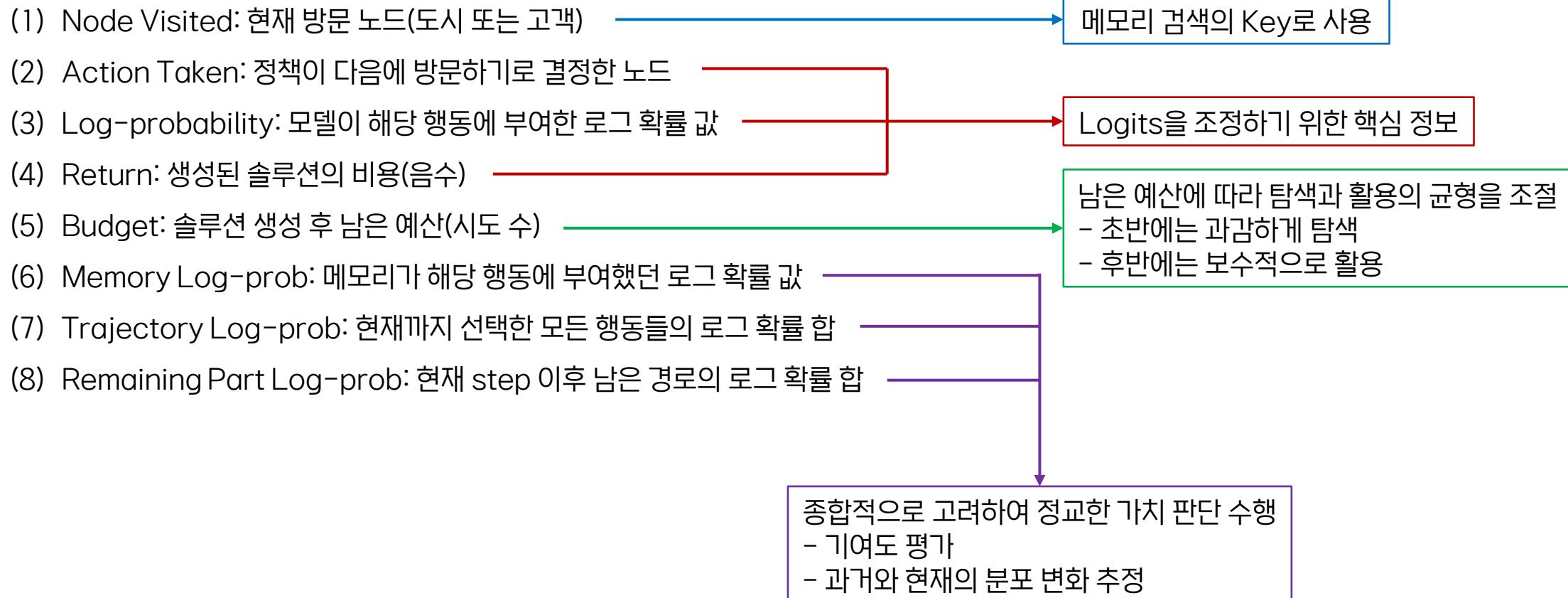
❖ MEMENTO 개요

- ① 현재 상태 기반으로 관련 데이터 검색
- ② 검색된 데이터와 상태로부터 특징 생성
- ③ 특징으로부터 보정 로짓(correction logits)을 계산하여 기존 로짓과 합산
- ④ 최종 로짓으로 솔루션 생성
- ⑤ 실행 과정에서 얻은 전이(Transition) 데이터들을 메모리에 저장



- | | | | | | |
|---------------------------------------------|-------------------------------------------|---------------------------------|----------------------------------------|---------------------------------|-----------------------------------------|
| 1. Retrieve memories based on current state | 2. Create features using state & memories | 3. Compute logits from features | 4. Rollout policy to create a solution | 5. Create data from transitions | 6. Add data to memory and go back to 1. |
|---------------------------------------------|-------------------------------------------|---------------------------------|----------------------------------------|---------------------------------|-----------------------------------------|

❖ 메모리 저장 정보



❖ 메모리 검색 방식

✓ Dilemma

- 가장 좋은 정보를 얻으려면 현재의 전체 경로 상황과 가장 유사한 경험을 찾아야 함 (ex. K-nearest neighbor(KNN) 알고리즘)
- 이 방식은 매번 계산해야 하므로 비용이 매우 비싸고 느림

✓ MEMENTO의 접근 방식

- 현재 같은 노드에 있다는 사실 자체가 가장 강력한 유사성 지표
- 복잡한 계산 없이 노드 번호만을 이용해서 데이터 검색
- 검색된 데이터 외에, 현재의 잔여 예산도 정책 업데이트를 위한 특징으로 추가

✓ 장점

- KNN 검색보다 훨씬 빠름
- 동일 노드에서의 과거 결정은 현재 상황과 높은 관련성을 가짐

❖ Logits update 과정

- ① 특징 및 액션 분리: 검색된 데이터에서 액션과 관련된 특징 f_{a_i} 분리((2)~(8))
- ② MLP 처리: 각 특징을 정규화 후, 특징 벡터를 H_{θ_M} 에 입력하여 **스칼라 가중치** 출력
 - H_{θ_M} : 작은 신경망(MLP). MEMENTO의 보조 두뇌 → “이 상황에서 어떤 경험을 믿고 Logits을 얼마나 바꿀지”를 스스로 학습

③ 보정 로짓(l_M) 계산:

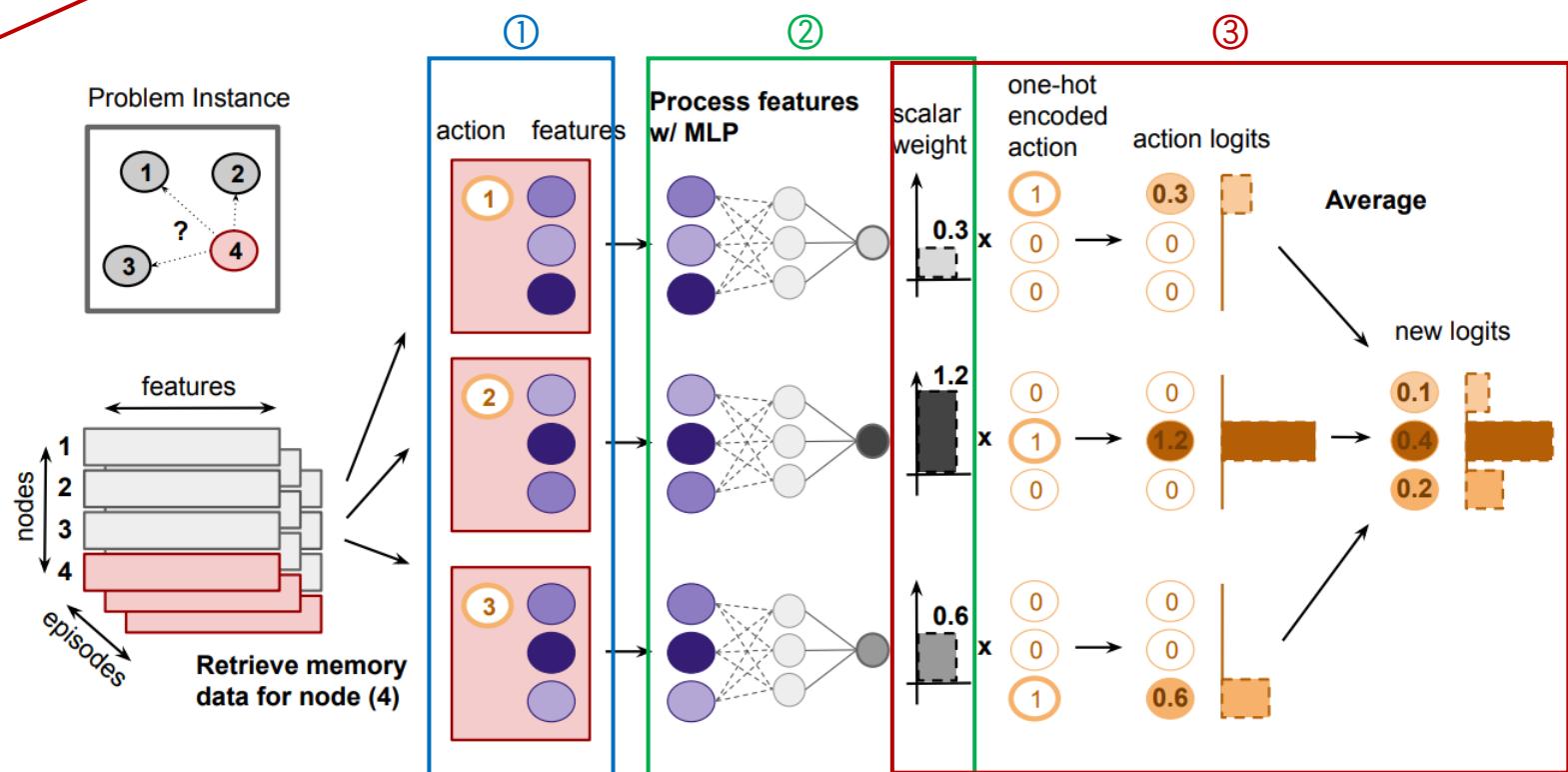
$$l_M = \sum_i a_i H_{\theta_M}(f_{a_i})$$

- a_i : 과거 행동 a_i 의 one-hot 인코딩

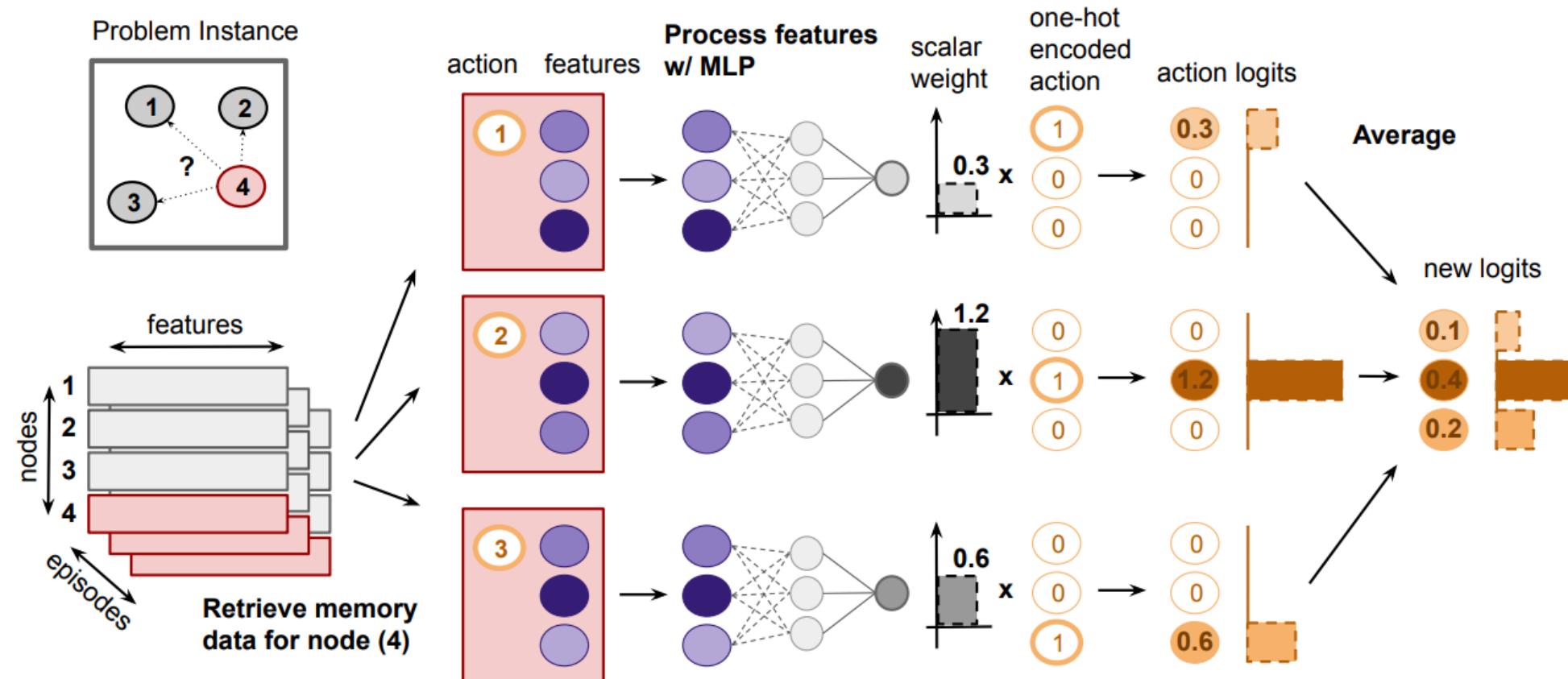
④ 최종 로짓 계산:

$$l_{final} = l_{base} + l_M$$

- l_{base} : 기본 정책의 logits
- l_M : 메모리를 통해 나온 보정값



❖ Logits update 과정



❖ MEMENTO 학습 절차

- ✓ 목표: 다중 시도(multi-shot) 환경에서 최적의 솔루션을 찾도록 MEMENTO 훈련
- ✓ 기반 모델: 사전 학습된 단일 에이전트 모델(ex. POMO)을 기본 정책으로 사용
- ✓ 학습 과정

Algorithm 1 MEMENTO Training

```

1: Input: problem distribution  $\mathcal{D}$ , problem size  $N$ , memory  $M$ , batch size  $B$ , budget  $K$ , number of
   training steps  $T$ , policy  $\pi_\theta$  with pre-trained parameters  $\theta$ .
2: initialize memory network parameters  $\phi$ 
3: combine pre-trained policy parameters and memory network parameters  $\tilde{\theta} = (\theta, \phi)$ 
4: for step 1 to  $T$  do
5:    $\rho_i \leftarrow \text{Sample}(\mathcal{D}) \forall i \in 1, \dots, B$ 
6:   for attempt 1 to  $K$  do
7:     for node 1 to  $N$  do
8:        $m_j \leftarrow \text{Retrieve}(M) \forall j \in 1, \dots, B$  {Retrieve data from the memory}
9:        $\tau_i^j \leftarrow \text{Rollout}(\rho_i, \pi_{\tilde{\theta}}(\cdot | m_j)) \forall i, j \in 1, \dots, B$ 
10:       $m_j \leftarrow f(m_j, \tau_i^j)$  {Update the memory with transition data}
11:       $R_i^* \leftarrow \max(R_i^*, \mathcal{R}(\tau_i^j)) \forall i \in 1, \dots, B$  {Update best solution found so far}
12:       $\nabla L(\tilde{\theta}) \leftarrow \frac{1}{B} \sum_{i \leq B} \text{REINFORCE}(\text{ReLU}(\tau_i^j - R_i^*))$  {Estimate gradient}
13:       $\nabla L(\tilde{\theta}) \leftarrow \frac{1}{K} \sum_{i=1}^K \nabla L(\tilde{\theta})$  {Accumulate gradients}
14:       $\tilde{\theta} \leftarrow \tilde{\theta} - \alpha \nabla L(\tilde{\theta})$  {Update parameters}

```

- | weight | Advantage |
|--------------------------|-------------------------------------|
| $\log(1 + \epsilon + i)$ | $\text{ReLU}(R(\tau_i) - R_{best})$ |
- 손실 함수 $\mathcal{L} = - \sum_{i=1}^B \log(1 + \epsilon + i) \cdot \text{ReLU}(R(\tau_i) - R_{best}) \cdot \sum_t \log \pi_M(a_t | s_t, M_t)$
 - $\text{ReLU}(R(\tau_i) - R_{best}) = \max(R(\tau_i) - R_{best}, 0)$
 - $\log(1 + \epsilon + i)$: 예산이 소진될수록(시도 횟수 i 가 늘어날수록) 기록 갱신은 점점 더 어려워짐

❖ MEMENTO 추론 절차

- ✓ 상태: MEMENTO의 학습이 완료된 후, 기본 뉴럴 솔버와 메모리 처리 모듈의 모든 파라미터는 고정
- ✓ 과정
 - 주어진 문제 인스턴스에 대해 앞서 설명한 MEMENTO의 사이클 반복
 - 메모리를 수집된 시도들로 채우고, 검색된 데이터를 처리하여 액션 로짓 업데이트
 - 이 과정은 주어진 추론 예산이 모두 소진될 때까지 반복
- ✓ 장점
 - 역전파(back-propagation) 없이 빠른 추론 속도 유지
 - 어떤 뉴럴 솔버와도 결합 가능

❖ 실험 설정

- ✓ 평가 문제: Traveling Salesman Problem (TSP) & Capacitated Vehicle Routing Problem (CVRP)
- ✓ 데이터셋
 - 훈련 분포 내: N=100 (10,000 인스턴스)
 - 훈련 분포 외: N=125, 150, 200 (각 1,000 인스턴스)
 - 대규모 인스턴스: N=500
- ✓ 기반 모델 (Base Policy): POMO (Kwon et al., 2020) – 대부분의 RL 솔버의 기반
- ✓ 비교 대상 (Baselines)
 - POMO (greedy/sampling): 기본 정책의 성능
 - EAS (Hottung et al., 2022): SOTA 정책 그레이디언트 기반 온라인 적응 방법
 - SGBS (Choo et al., 2022): 트리 탐색 기반 방법
 - COMPASS (Chalumeau et al., 2023b): SOTA 다양성 기반 솔버 (Zero-shot 결합 시 비교)
 - 산업 솔버 (Industrial Solvers): LKH3, Concorde, HGS (최적성 갭 비교 기준)

❖ MEMENTO vs. Policy Gradient Fine-tuning (EAS)

- MEMENTO는 동일 budget에서 EAS 대비 일관되게 더 낮은 tour length(GAP 개선)

(a) TSP

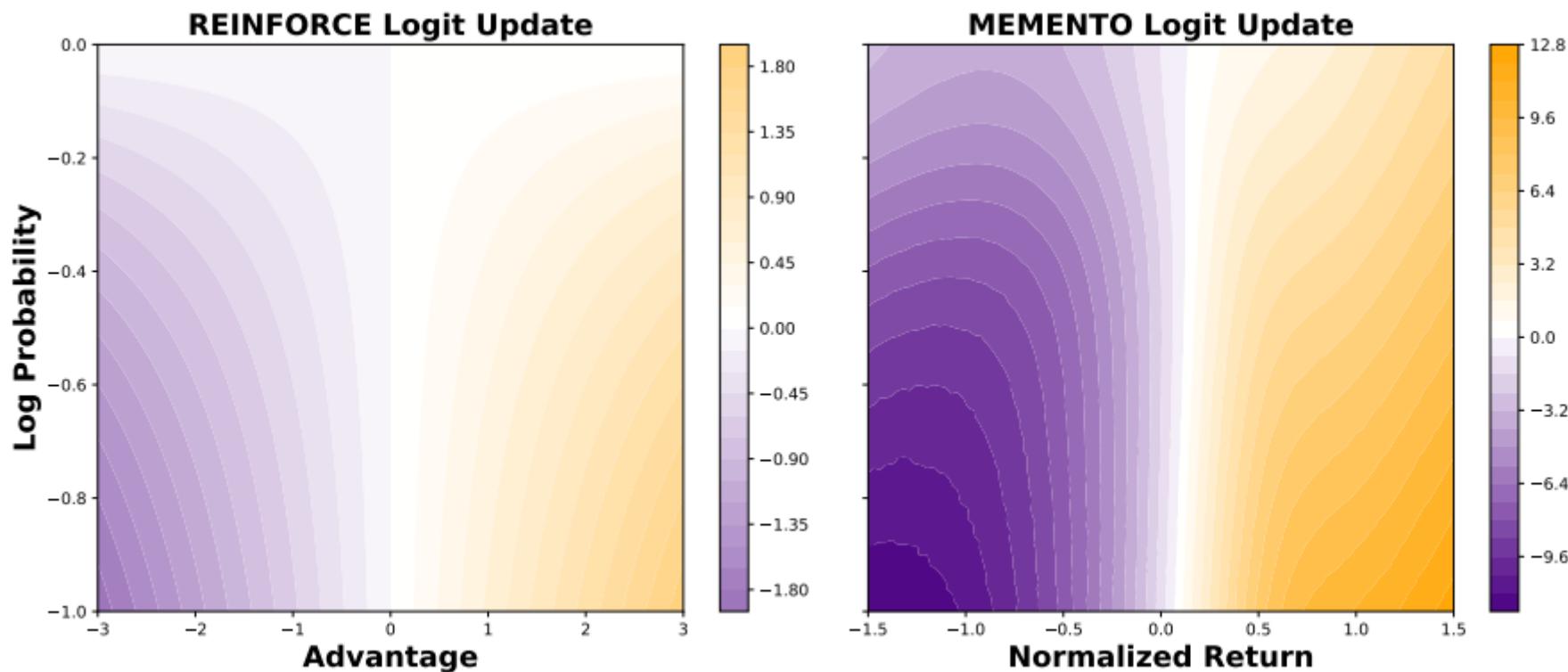
Method	Training distr.		Generalization					
	$n = 100$		$n = 125$		$n = 150$		$n = 200$	
	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap
LKH3	7.765	0.000%	8.583	0.000%	9.346	0.000%	10.687	0.000%
POMO (greedy)	7.796	0.404%	8.635	0.607%	9.440	1.001%	10.933	2.300%
POMO (sampling)	7.779	0.185%	8.609	0.299%	9.401	0.585%	10.956	2.513%
SGBS *	7.769*	0.058%	-	-	9.367*	0.220%	10.753*	0.619%
EAS-Emb	7.778	0.161%	8.604	0.238%	9.380	0.363%	10.759	0.672%
MEMENTO	7.768	0.045%	8.592	0.109%	9.365	0.202%	10.758	0.664%

(b) CVRP

Method	Training distr.		Generalization					
	$n = 100$		$n = 125$		$n = 150$		$n = 200$	
	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap
LKH3	15.65	0.000%	17.50	0.000%	19.22	0.000%	22.00	0.000%
POMO (greedy)	15.874	1.430%	17.818	1.818%	19.750	2.757%	23.318	5.992%
POMO (sampling)	15.713	0.399%	17.612	0.642%	19.488	1.393%	23.378	6.264%
SGBS *	15.659*	0.08%	-	-	19.426*	1.08%	22.567*	2.59%
EAS-Emb	15.663	0.081%	17.536	0.146%	19.321	0.528%	22.541	2.460%
MEMENTO	15.657	0.066%	17.521	0.095%	19.317	0.478%	22.492	2.205%

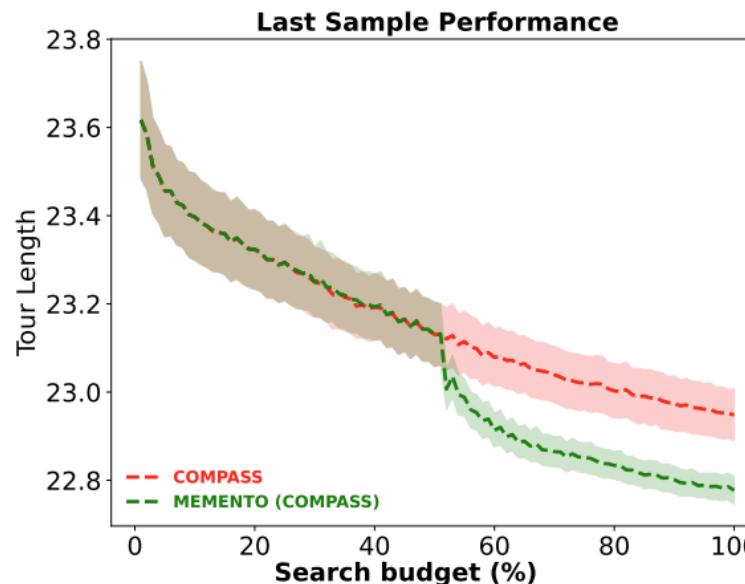
❖ REINFORCE vs. MEMENTO Logit Update Heatmap

- 낮은 확률이었지만 결과가 좋았던 행동을 강하게 강화
- 샘플링이 놓치는 좋은 선택을 점진적으로 끌어올림



❖ Zero-shot 결합

- ✓ 목표: MEMENTO가 사전 학습된 상태에서, 추가 재훈련 없이 COMPASS와 같은 다양성 기반 SOTA 솔버와 결합될 수 있는지 검증
- ✓ COMPASS: 다양한 사전 학습 정책 모음에서 탐색하여 적응, 하지만 온라인 데이터 활용은 미흡
- ✓ 결합 방법: COMPASS 모델에 MEMENTO의 메모리 모듈을 zero-shot으로 적용 (별도 훈련 없이)



Method	$n = 100$		$n = 125$		$n = 150$		$n = 200$	
	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap
COMPASS	15.644	-0.019%	17.511	0.064%	19.313	0.485%	22.462	2.098%
MEMENTO (COMPASS)	15.634	-0.082%	17.497	-0.041%	19.290	0.336%	22.405	1.808%

❖ 대규모 인스턴스 확장 (N=500, 훈련 시간=4일)

- ✓ CVRP 고예산에서 EAS가 MEMENTO를 능가

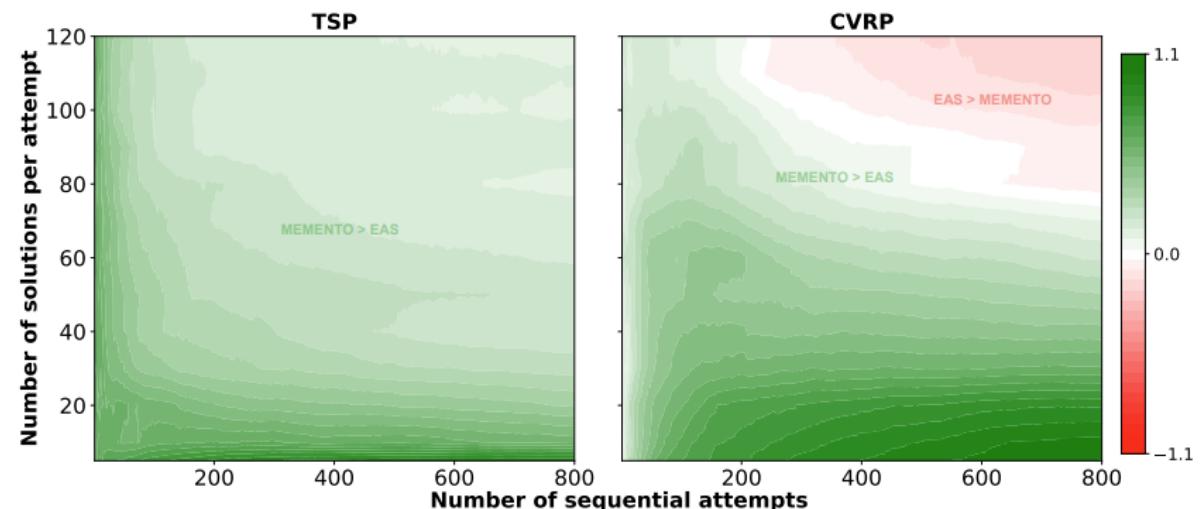
- 예산이 매우 커지면 메모리에 저장되는 데이터의 양이 엄청나게 많아지기 때문

<TSP>

Method	Obj.	Gap	Time
Concorde	16.55	0.000%	38M
LKH-3	16.55	0.003%	46M
DIMES	17.80	7.55%	2H
Difusco	17.23	4.08%	11M
DeepACO	18.74	13.23%	-
MOCO	16.84	1.72%	-
Pointerformer	17.14	3.56%	1M
Low Budget			
POMO (sampling)	16.999	2.736%	4M
EAS	16.878	2.006%	10M
COMPASS	16.811	1.603%	9M
MEMENTO (POMO)	16.838	1.766%	9M
MEMENTO (COMPASS)	16.808	1.586%	10M
High Budget			
POMO (sampling)	16.986	2.659%	9M
EAS	16.844	1.804%	24M
COMPASS	16.800	1.539%	22M
MEMENTO (POMO)	16.823	1.673%	23M
MEMENTO (COMPASS)	16.795	1.507%	29M

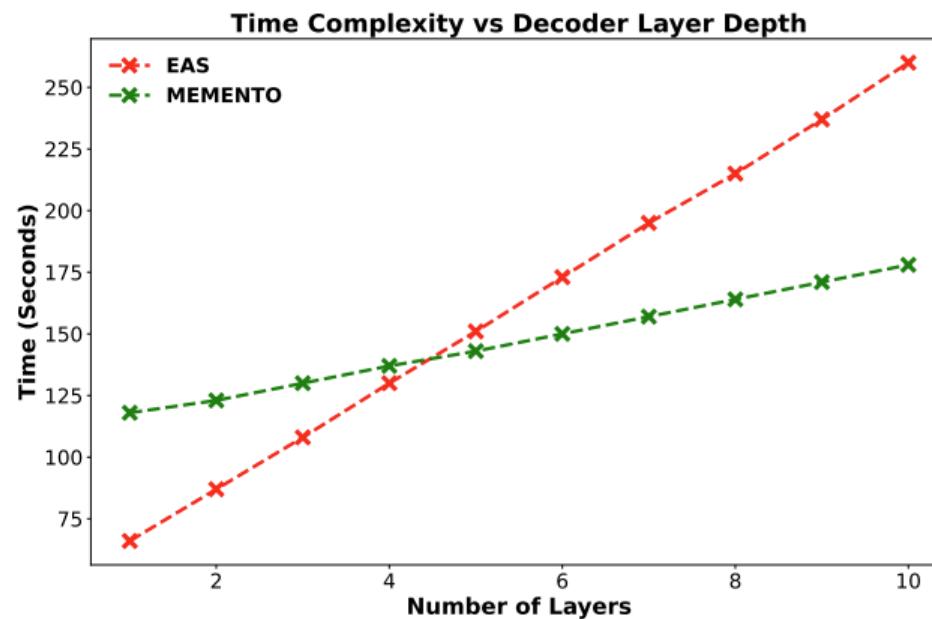
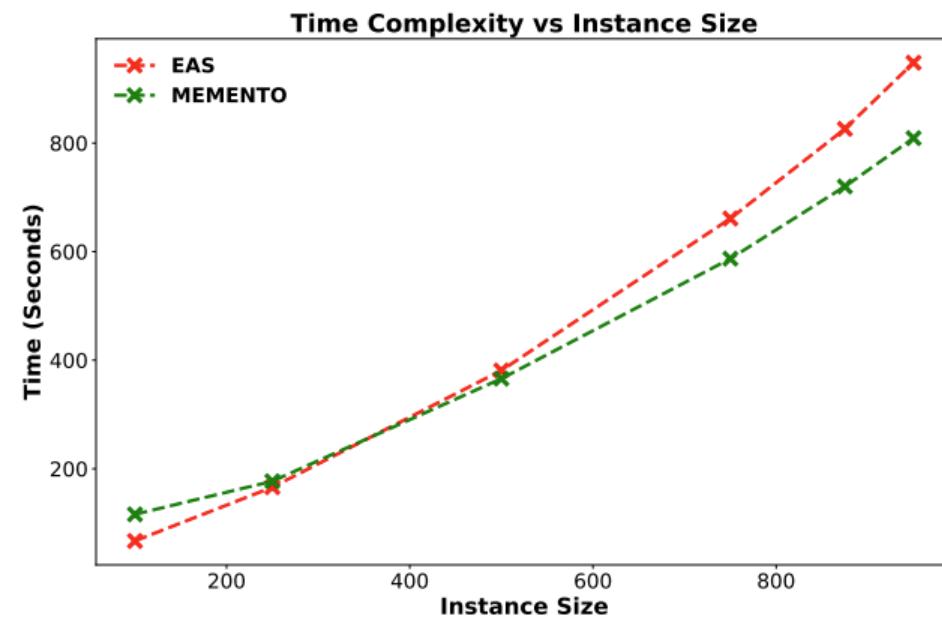
<CVRP>

Method	Obj.	Gap	Time
LKH-3	37.229	0.00%	6H
Low Budget			
POMO (sampling)	37.501	0.731%	9M
EAS	37.425	0.525%	16M
COMPASS	37.336	0.287%	10M
MEMENTO (POMO)	37.367	0.369%	16M
MEMENTO (COMPASS)	37.309	0.215%	12M
High Budget			
POMO (sampling)	37.456	0.608%	20M
EAS	37.185	-0.120%	36M
COMPASS	37.279	0.133%	25M
MEMENTO (POMO)	37.306	0.206%	65M
MEMENTO (COMPASS)	37.251	0.059%	36M



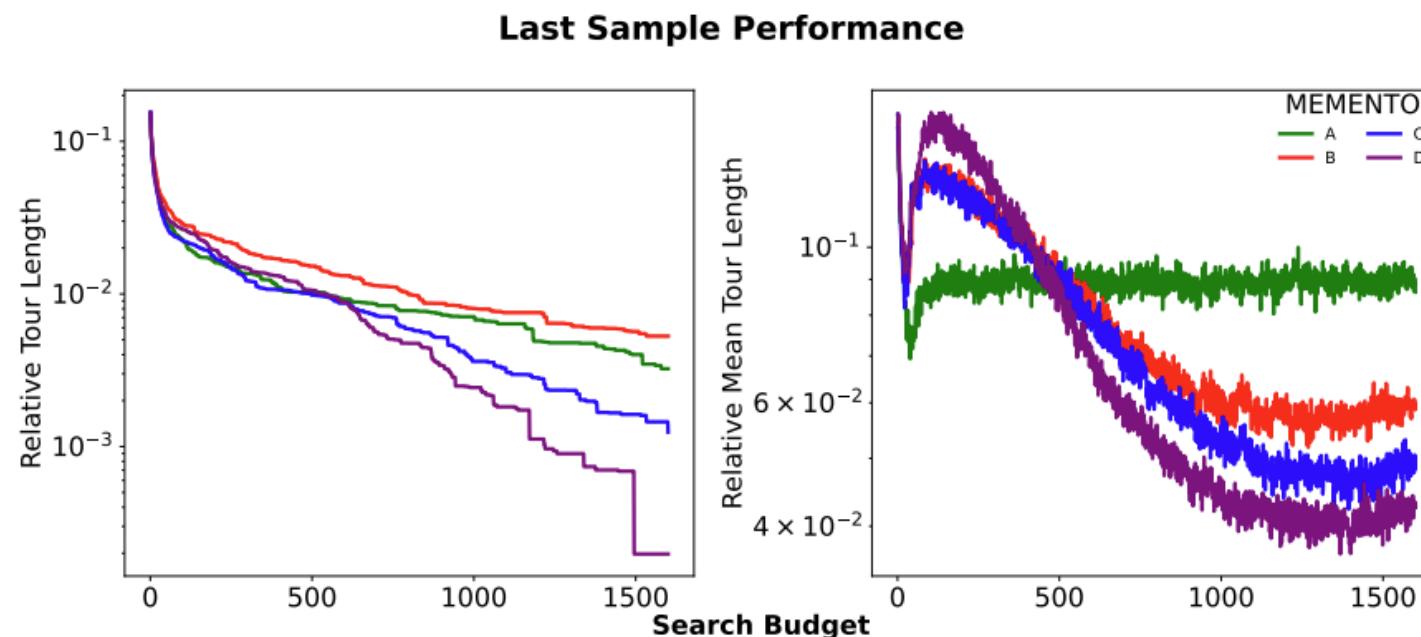
❖ 시간 복잡도 분석

- ✓ 인스턴스가 커질수록 빨라짐
- ✓ 디코더 크기가 커질수록 빨라짐



❖ Ablation Study

- A: return + logp
- B: A + remaining budget (잔여 예산)
- C: B + budget when action was taken (액션 시점 예산)
- D: C + memory logp + trajectory logp + end trajectory logp (메모리 로짓, 궤적 로짓, 잔여 궤적 로짓)



❖ 결론

- ✓ MEMENTO의 핵심: 가중치 업데이트 없이 메모리로 로짓을 보정해 성능을 끌어올림
- ✓ 기여
 - 학습된 온라인 적응: 확률적 샘플링, 트리 탐색, 정책 그레디언트 미세 조정을 능가하는 성능
 - SOTA 달성: TSP 및 CVRP 벤치마크 12개 중 11개 태스크에서 SOTA 달성
 - Zero-shot 결합: COMPASS와 같은 미지 솔버와 추가 재훈련 없이 성공적으로 결합, 시너지 효과 입증
 - 확장성 및 데이터 효율성: 대규모 인스턴스($N=500$) 및 낮은 예산 상황에서도 강건한 성능, 우수한 시간 복잡도 스케일링
 - 해석 가능한 업데이트 규칙: REINFORCE 스타일을 넘어선 정교한 탐색/활용 규칙 학습
- ✓ 한계 (Limitations)
 - 추가 계산/메모리 오버헤드: 기본 메모리 없는 정책 대비 오버헤드 존재
 - MLP 기반이 아닌 직접 수학적으로 도출하는 것이 향후 연구 방향이 될 수 있음