Hard Tasks First:
Multi-Task Reinforcement Learning Through Task Scheduling

25.01.23

김정현

I Introduction

❖ 강화학습의 발전 및 한계

- DRL은 복잡한 제어 문제 (ex. 게임 마스터링, 바둑 승리, 로봇 운동 제어 등)에서 뛰어난 성과를 보여 왔음
- 기존 방법의 한계:
  - ✓ 단일 태스크에 집중하기 때문에 복잡한 태스크에서 샘플 효율성이 낮음
  - ✓ 즉, 하나의 태스크만 학습하면서 다양한 태스크에서 사용할 수 있는 일반적인 지식을 배우지 못함

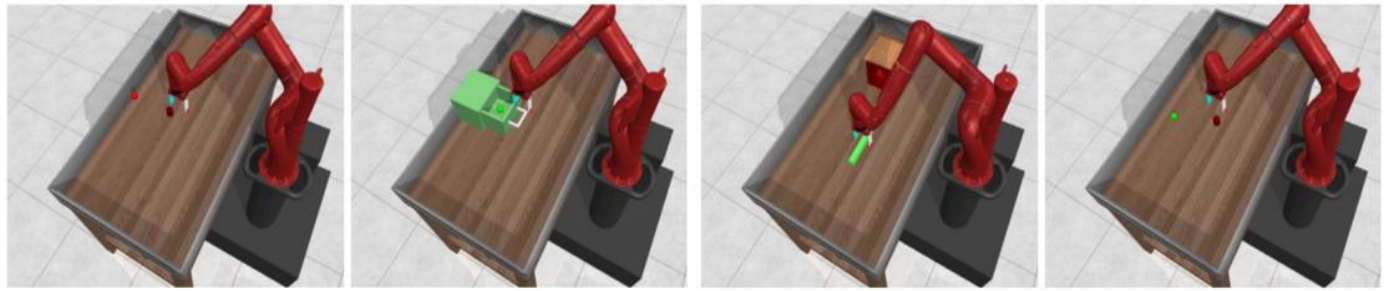
❖ Multi-Task Reinforcement Learning (MTRL)

- 하나의 정책 (Policy)으로 여러 개의 태스크 (Task)를 학습하는 RL 기법
- 파라미터 공유를 통해 샘플 효율성을 높이고, 더 일반화된 정책을 학습하는데 도움
- 쉬운 태스크가 너무 빨리 학습되며 어려운 태스크를 방해하는 단순성 편향 문제가 발생함
- 예시:
  - ✓ 단일 태스크 RL: 로봇 팔이 하나의 작업 (ex. 물건 집기)만 학습
  - ✓ 멀티 태스크 RL: 로봇 팔이 여러 개의 작업 (ex. 집기, 밀기, 회전시키기)을 학습

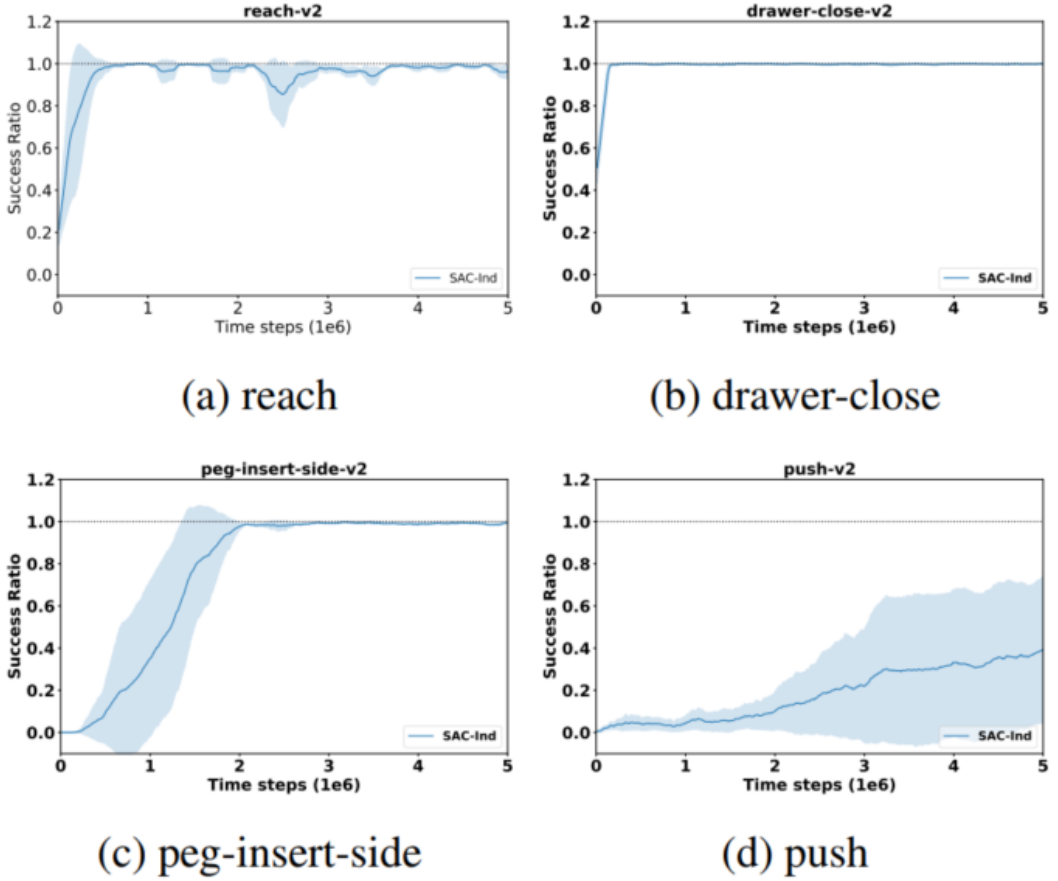
Ⅱ Motivation

❖ 태스크 간 복잡도의 차이 (Heterogeneity in Task Complexity)

- MTRL에서는 태스크마다 난이도가 다르고, 어려운 태스크는 훨씬 더 많은 샘플과 학습 시간이 필요함
- Meta-World 벤치마크 (Yu et al., 2019) : 로봇 제어 태스크 모음
  - ✓ reach, drawer-close, peg-insert-side, push
- 동일한 알고리즘 (SAC)을 사용하여 각 태스크를 개별적으로 학습한 결과:
  - ✓ 쉬운 태스크 (a, b): 빠르게 학습되고 높은 성능을 달성
  - ✓ 어려운 태스크 (c, d): 학습이 느리고 학습 성능의 변동성이 큼



(a) reach (b) drawer-close (c) peg-insert-side (d) push



(a) reach (b) drawer-close (c) peg-insert-side (d) push

Ⅱ Motivation

❖ 단순성 편향 (The Simplicity Bias)

- MTRL에서 태스크 간 복잡도 차이가 학습에 미치는 영향을 분석
- 태스크 난이도가 다를 경우, 쉬운 태스크가 지나치게 빠르게 학습되며, 어려운 태스크 학습이 방해받는 단순성 편향 발생
- 실험 설정:

  1. 두 가지 태스크 세트 구성:
     - ✓ 세트 1 (Γ_1): 쉬운 태스크 'reach' + 어려운 태스크 'push'
     - ✓ 세트 2 (Γ_2): 어려운 태스크 'peg-insert-side' + 어려운 태스크 'push'
  2. 학습 결과 비교:
     - ✓ 세트 1 (Γ_1): 'reach'가 빠르게 학습되면서 'push' 학습이 방해됨 → 'push' 태스크 학습 실패
     - ✓ 세트 2 (Γ_2): 'peg-insert-side'와 'push'가 비슷한 난이도를 가지므로 균형 잡힌 학습이 가능 → 'push' 태스크 부분적 학습 성공

- 어려운 태스크(push)는 쉬운 태스크와 함께 학습될 때보다, 다른 어려운 태스크(peg-insert-side)와 함께 학습될 때 더 잘 학습됨

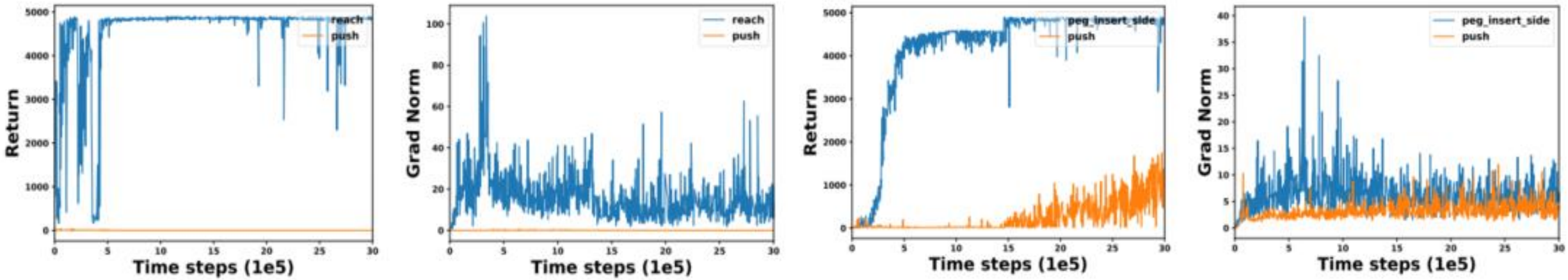
Ⅱ Motivation

❖ 단순성 편향 (Simplicity Bias)

1. 두 가지 작업 세트:

   ✓ 세트 1 (Γ_1): 쉬운 작업 'reach' + 어려운 작업 'push'

   ✓ 세트 2 (Γ_2): 중간 난이도 작업 'peg-insert-side' + 어려운 작업 'push'



(a) Return (b) Gradient norm

세트 1 (Γ_1)

(a) Return (b) Gradient norm

세트 2 (Γ_2)

❖ Gradient Magnitude Analysis of the Simplicity Bias

- 정책 그래디언트 분석을 통해 단순성 편향이 MTRL 학습에서 발생하는 이유 분석
- 학습 목표에 대한 정책 그래디언트:

$$\nabla_\theta J_{MT}^{conv} = \frac{1}{|\Gamma_j|} \sum_{\mathcal{T}_i \in \Gamma_j} \nabla_\theta J(\pi_\theta, \mathcal{T}_i), \quad j = 1, 2.$$

- 미니배치 샘플링 방식으로 리플레이 버퍼에서 데이터를 가져와 그래디언트를 계산함 (Lillicrap et al., 2016)
- 세트 1 (Γ_1) 실험 결과:
  - ✓ 'reach' 태스크의 그래디언트 크기는 초기 학습 단계에서 빠르게 증가
  - ✓ 'push' 태스크의 그래디언트 크기는 거의 증가하지 않음
  - ✓ 쉬운 태스크의 높은 보상(return)이 초기 정책 업데이트를 지배하여 어려운 태스크 학습을 방해
- 세트 2 (Γ_2) 실험 결과:
  - ✓ 'push'와 'peg-insert-side' 태스크의 그래디언트 크기가 유사하게 증가함
  - ✓ 두 태스크 모두 난이도가 높아 초기 학습에서 비슷한 수준의 보상을 얻으며 균형잡힌 학습이 가능

❖ Scheduled Multi-Task Training (SMT)

- 태스크 난이도를 평가하고 이를 기반으로 학습 순서를 동적으로 조정하는 알고리즘을 제안
- 초기 학습 단계에서 더 어려운 태스크를 우선적으로 학습하여 쉬운 태스크의 영향을 줄임
- 핵심 기법:
  - ✓ Complexity-Based Scheduling: 어려운 태스크를 먼저 해결하여, 쉬운 태스크가 학습을 지배하는 현상을 방지함
  - ✓ Reset Mechanism: 정책 네트워크가 쉬운 태스크에 과적합되는 것을 방지하기 위해, 일정 주기마다 네트워크 재초기화
  - ✓ Budget-Based Training: 학습 예산을 조정하여 불필요한 학습 리소스 낭비를 방지하고 효율적인 학습 진행

❖ 정책 성능을 평가하는 확률 모델

- Levine (2018)의 연구를 기반으로, 정책 π_θ가 얼마나 최적의 정책과 가까운지를 평가하는 모델을 사용함
- 정책이 생성한 경로 τ가 최적의 경로에 가까운지를 축정하는 확률 모델 정의
- 이를 위해, 이진 확률 변수 O를 도입
  - ✓ $O = 1$: 경로 τ가 최적의 경로임, $O = 0$: 경로 τ가 최적이 아님
- $$p(\mathcal{O} = 1 \mid \tau) = \exp\left(\sum_{t=1}^{H} \gamma^{t-1}(r_t - R_{\max})\right),$$

$$\begin{aligned} & -\mathrm{KL}\left(q_\theta(\tau) \,\|\, p(\tau \mid \mathcal{O} = 1)\right) \\ & = \mathbb{E}_{\tau \sim q_\theta}\left[\log p(\tau \mid \mathcal{O} = 1) - \log q_\theta(\tau)\right] \\ & = \mathbb{E}_{\tau \sim q_\theta}\left[\sum_{t=1}^{H}\left(\gamma^{t-1} r_t - \log \pi_\theta(a_t|s_t)\right)\right] - C, \end{aligned}$$

❖ 정책과 최적 경로 분포 사이의 KL Divergence 최적화

- 강화학습에서는 최적 정책이 최적 경로 분포와 유사해지도록 학습해야 함
- 현재 정책이 생성한 경로 분포와 최적 경로 분포 간의 차이를 최소화하기 위해 KL Divergence 사용
- 정책이 높은 보상을 받는 경로를 샘플링하면서도, 탐색을 유지하도록 학습하는 원리 제공

❖ Complexity-Based Scheduling

- 기본 원칙: 1) 태스크 난이도가 큰 차이가 나는 경우, 동시에 학습하지 않음 2) 어려운 태스크에 더 많은 학습 시간을 할당
- Training Pool (P_t) : 현재 가장 어려운 K개의 태스크로 구성된 태스크 집합 → P_t에 있는 작업은 먼저 학습
- Main Pool (P_m) : P_t에 포함되지 않은 나머지 태스크 집합

  → 학습이 진행됨에 따라 태스크 난이도가 점차 낮아지면, 해당 태스크는 P_m으로 이동하고, 새로운 어려운 태스크가 P_t에 추가됨
- 태스크 난이도 평가: 각 태스크의 난이도를 측정
- 일정 시간(T_{eval}) 간격마다 각 태스크에서 n_{eval}개의 경로를 생성하여 평균 성능을 평가
- 해당 태스크에서 얻은 평균 보상과 엔트로피 값을 기반으로 난이도를 정량화함
- 난이도를 지속적으로 평가하여, 항상 가장 어려운 태스크 K개를 학습하도록 구성됨

❖ Complexity-Based Scheduling

- 난이도 평가 매트릭: 최적 경로와 현재 정책이 생성한 경로 사이의 차이를 측정함 (Levine, 2018)
- 하나의 경로(Trajectory, τ)는 강화학습 환경에서 정책에 따라 수행된 일련의 상태-행동 시퀀스를 의미
- 최적 경로의 확률 정의 $p(\mathcal{O} = 1 \mid \tau) = \exp\left(\sum_{t=1}^{H} \gamma^{t-1}(r_t - R_{\max})\right)$
- KL Divergence 기반의 평가 방법

$$= \mathbb{E}_{\tau \sim q_\theta}\left[\sum_{t=1}^{H}\left(\gamma^{t-1} r_t - \log \pi_\theta(a_t|s_t)\right)\right] - C,$$

❖ Reset Mechanism

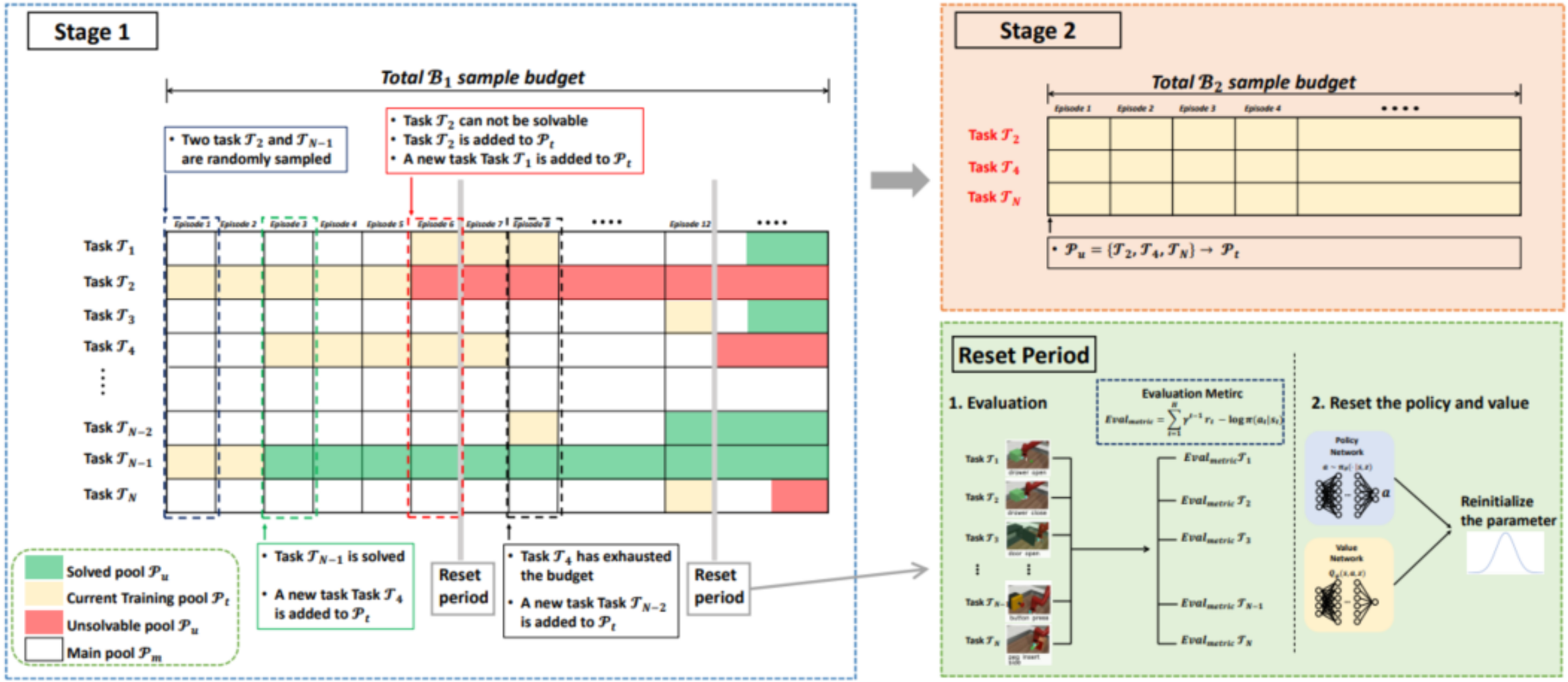
- 학습 초기에는 태스크 난이도를 알 수 없으므로 쉬운 태스크가 선택될 가능성이 높음
- 초기 학습 단계에서 에이전트가 쉬운 태스크에 과적합되는 문제를 해결하고, 어려운 태스크 학습 기회를 확대하는 데 기여함
- 정책 파라미터 (θ)와 Q-value 함수 파라미터 (ψ)를 주기적으로 초기화 (Nikishin et al., 2022)
- 리플레이 버퍼는 초기화하지 않고 유지하여, 이전 학습 데이터를 지속적으로 활용할 수 있음

❖ Budget-Based Training

- 복잡도 기반 스케줄링의 문제점: 학습이 불가능한 태스크가 너무 많은 자원을 소비하는 경우가 발생할 수 있음
- 충분한 샘플을 제공하면 학습될 수 있는 태스크가 샘플 부족으로 학습되지 못할 가능성이 있음
- 이를 방지하기 위해 각 태스크에 학습 예산(Budget)을 동적으로 할당하고, 필요에 따라 학습을 조기 종료하는 기법을 도입
- 각 태스크에 학습 예산 b_i 를 설정하고 그 타임스텝 동안 학습을 진행한 후, 성능을 평가하여 학습 지속여부를 결정함
  - ✓ 학습이 진행되면서, 최근 n_{train} 개의 경로의 평균 보상을 평가
  - ✓ 평가 기준: m = 태스크가 해결 불가능하다고 간주되는 하한 임계값, M = 태스크가 해결되었다고 간주되는 상한 임계값
- 태스크 재분류:
  - ✓ 해결된 태스크 (P_s) : 평균 보상이 M을 초과하면, 해당 태스크는 해결된 것으로 간주하고 더 이상 학습하지 않음
  - ✓ 해결 불가능한 태스크 (P_u): 평균 보상이 m 미만이면, 해당 태스크는 해결 불가능한 것으로 간주하고 학습 풀에서 제외
- 전체 학습 예산을 두 개의 단계로 나눠서 사용

❖ Budget-Based Training

- Step 1: 강화학습 환경 및 모델 초기화
  - ✓ 학습할 MTRL 환경 C 정의 (로봇팔의 다양한 태스크)
  - ✓ 정책 네트워크, Q- value 함수, 태스크 임베딩 네트워크 초기화
  - ✓ 각 태스크에 대한 개별적인 경험 리플레이 버퍼 D 생성
  - ✓ Stage 1(B_1)과 Stage 2(B_2)에 사용할 총 학습 예산 B_{total}을 설정
- Step 2: 초기 태스크 풀 구성
  - ✓ 랜덤으로 K개의 태스크를 선택하여 Training Pool (P_t) 생성
  - ✓ 각 태스크에 학습 예산 할당 → $b_i = \kappa B$
- Step 3: 학습 실행 및 성능 평가
  - ✓ 학습을 진행하며, 결과 데이터를 경험 리플레이 버퍼에 저장
  - ✓ 태스크 성능 평가 → 최근 n개의 에피소드에서 평균 보상 계산
- Step 4: 태스크 상태 업데이트
  - ✓ 평균 보상이 M을 초과 → P_s로 이동 (해결됨)
  - ✓ 예산을 모두 소진했음에도 평균 보상이 m 미만→ P_u로 이동 (해결 불가)
  - ✓ 학습이 진행중이지만 해결되지 않은 태스크 → P_m으로 이동
- Step 5: 네트워크 초기화
  - ✓ 일정 주기마다 파라미터 초기화
- Step 6: 새로운 태스크 추가
  - ✓ P_t에서 태스크가 부족하면 난이도가 높은 태스크부터 추가
- Step 7: Stage 1에서 해결되지 않은 P_u의 태스크를 다시 학습



IV Experiments

❖ Environment

- Meta-World는 로봇 조작 태스크 50개로 구성된 강화학습 벤치마크 (Yu et al., 2019)
- MuJoCo(Multi-Joint dynamics with Contact) 환경에서 동작하는 Sawyer 로봇팔을 사용
- MT10 및 MT50 두 가지 실험 모드 사용

❖ Baseline

Table 1. Comparisons of per-task and average success ratios (%) of the Meta-World MT10 benchmark. For the task name corresponding to each task ID, see Appendix A.

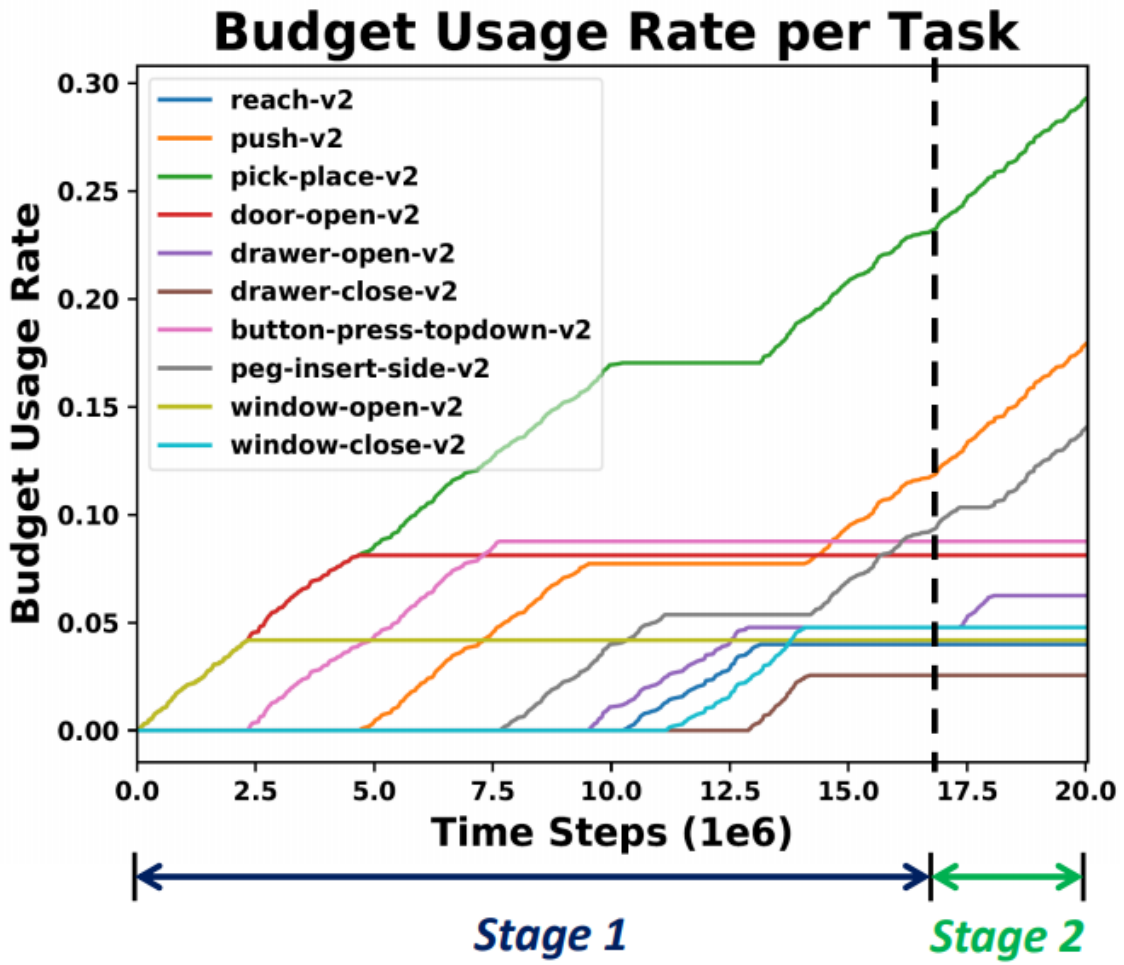
	Task ID										
Algorithm	0	1	2	3	4	5	6	7	8	9	Average
SAC-MT	98±2.4	0±0.0	0±0.0	**100±0.0**	**100±0.0**	**100±0.0**	96±2.0	0±0.0	**100±0.0**	**100±0.0**	69.4 ± 0.8
SAC-MT-MH	**100±0.0**	28±21.8	0±0.0	**100±0.0**	98±2.5	**100±0.0**	**100±0.0**	46±21.8	**100±0.0**	**100±0.0**	77.2 ± 11.9
Soft Modular	**100±0.0**	32±14.9	0±0.0	**100±0.0**	**100±0.0**	**100±0.0**	**100±0.0**	12±14.9	**100±0.0**	**100±0.0**	74.4 ± 10.5
PCGrad	94±3.7	0±0.0	0±0.0	**100±0.0**	**100±0.0**	**100±0.0**	**100±0.0**	54± 39.9	**100±0.0**	**100±0.0**	74.8 ± 13.7
PaCo	**100±0.0**	44±25.2	0±0.0	**100±0.0**	**100±0.0**	**100±0.0**	**100±0.0**	80 ± 40	**100±0.0**	**100±0.0**	82.4 ± 14.2
SMT (Ours)	96±3.7	62±17.9	**34±13.8**	**100±0.0**	**100±0.0**	**100±0.0**	**100±0.0**	**76±31.9**	**100±0.0**	**100±0.0**	**86.8 ± 8.6**

❖ Baseline

- 어려운 태스크에 더 많은 학습 샘플을 사용하고, 쉬운 태스크에는 적은 샘플을 사용함

Table 2. Comparison of average bottom-k success ratios of the Meta-World benchmark MT50 benchmark.

	Average Bottom-k Success Ratio (%)				
Algorithm	$k = 10$	$k = 20$	$k = 30$	$k = 40$	$k = 50$
SAC-MT	**0.0 ± 0.0**	3.7 ± 5.6	21.0 ± 13.8	40.7 ± 10.4	52.6 ± 8.3
SAC-MT-MH	**0.0 ± 0.0**	4.5 ± 6.1	26.1 ± 14.6	44.0 ± 11.0	55.2 ± 8.8
Soft Modular	**0.0 ± 0.0**	1.8 ± 3.7	23.7 ± 12.3	42.6 ± 9.5	54.1 ± 7.6
PCGrad	**0.0 ± 0.0**	**0.0 ± 0.0**	21.0 ± 12.9	39.9 ± 10.7	51.9 ± 8.5
PaCo	**0.0 ± 0.0**	4.6 ± 8.2	26.1 ± 15.0	44.6 ± 11.2	55.6 ± 9.1
SMT (Ours)	**0.0 ± 0.0**	**8.0 ± 8.9**	**26.8 ± 13.1**	**45.0 ± 9.9**	**56.0 ± 8.0**



❖ Ablation Studies

Table 3. Ablation studies on the hyperparameters

	Hyperparameters								
	κ		M		m		B_1		Default
Task Set	0.7	0.9	2000	3000	500	2000	1.5×10^7	2.0×10^7	
$\mathcal{C}_{\text{easy}}$	**99.6 ± 0.1**	95.6± 3.7	62.5± 28.3	83.2± 11.9	92.7± 5.8	99.5± 0.2	94.3±4.2	99.5±0.2	99.3±0.3
$\mathcal{C}_{\text{difficult}}$	50.3±34.3	**69.8±19.6**	54.1±27.9	61.1±18.4	68.2±22.3	63.3±25.8	67.6±23.4	66.2±10.2	68.0±16.1
$\mathcal{C}$	79.8±15.5	85.3±9.8	59.1±23.6	74.4±12.7	82.9±9.4	85.1±9.9	83.6±8.9	82.9±10.0	**86.8 ±8.6**

Table 4. Ablation study on the reset mechanism

	Algorithm	
Task set	SMT w/o reset	SMT
$\mathcal{C}_{\text{easy}}$	88.7±7.1	99.3±0.3
$\mathcal{C}_{\text{difficult}}$	43.8±31.8	68.0±16.1
$\mathcal{C}$	70.7 ±17.4	86.8 ±8.6

Table 5. Ablation study on the scheduling method

	Scheduling Method		
Task Set	Easy Tasks First	Random	Hard Tasks First(Ours)
$\mathcal{C}_{\text{easy}}$	88.7±7.1	89.8±4.3	99.3±0.3
$\mathcal{C}_{\text{difficult}}$	23.8±36.4	26.2±44.5	68.0±16.1
$\mathcal{C}$	62.7±17.4	62.6±21.3	86.8±8.6

V Conclusion

❖ Contributions

- 어려운 태스크를 먼저 학습하는 “동적 태스크 우선순위 조정“ 기법을 도입
- 태스크 난이도 측정 지표를 활용하여 학습 자원을 최적화
- 리셋 메커니즘을 적용하여 단순성 편향을 방지
- 예산 기반 학습을 통해 학습 리소스를 효율적으로 활용

❖ Future Work

- SMT와 다른 RL 알고리즘의 결합 연구
- 태스크 난이도 평가 지표 개선
- SMT의 실제 문제 적용 연구
- 다양한 리셋 메커니즘 및 태스크 우선순위 전략 연구