강화학습의 기초 프로젝트
RL 기반 3교대 근무표 개발

팀원: A72068_이남준 (개인 팀)

Git 주소: https://github.com/dlskawns/Sogang-RLbasic-NurseRostering

프로젝트 진행 배경 및 주제

배경:

본 프로젝트는 간호사 근무표 생성 문제(NRP)를 해결하기 위해 다양한 강화학습(RL) 알고리즘을 적용하고, 그 성능을 비교 분석한 실험 보고서입니다. 강화학습 에이전트가 복잡한 제약조건을 스스로 학습하는지 검증합니다.

문제 정의:

기존의 병원 간호사의 3교대 근무표는 수간호사가 수기로 입력했어야 했습니다. 특히, 근로기준법과 보건복지부 야간전담간호사 가이드라인 강화에 따라 이를 놓칠 경우, 병원에 징계가 따를 수 있어 확실하게 해결될 수 있도록 해야 합니다.
아래와 같은 조건들을 만족하면서 처리할 수 있도록 해야 합니다.

- 병원에 따라 다르지만, 수십명의 간호사를 30일 x 3교대로 동시에 조율해야 함
- 야간(N) -> 조간(D) 금지, 연속 5일 근무 금지, Night 전담 간호사는 D, E는 불가능 등
- 매달 바뀌는 휴가 / 오프 요청의 반영
- 신규 간호사와 경력 간호사를 섞어 배치하는 팀 밸런스 유지

위와 같은 조건이 많아지면서 높은 비용의 수간호사 인력이 한 달 동안 근무표만 작성하기도 하는 등 **병원 측에서는 이러한 리소스를 줄이고자 하는 니즈**가 있으며, 이를 해결하기 위해, **수간호사들의 근무표 생성 방식(정책)을 강화학습을 통해 최적화하는 프로젝트**를 진행합니다.

주제:

강화학습(RL)을 이용한 다중 제약조건 간호사 근무표 최적화

목표:

1. NP-Hard 문제인 간호사 스케줄링을 강화학습 방식으로 해결하는 모델 설계.
2. Bandit, DQN, REINFORCE, PPO 4가지 알고리즘의 성능 비교.
3. 법적 제약(Hard Constraints) 준수 및 간호사 선호도(Soft Constraints) 반영 여부 확인.

데이터셋 - 간호사 정보

1. Nurses 데이터 (nurses.csv)

   스키마:

컬럼명	타입	설명
scenario_id	int	시나리오 ID (1~15)
nurse_id	int	간호사 ID (0~39)
experience_years	float	경력 연수 (0.5~15 랜덤)
is_night_only	int (0/1)	나이트 전담 여부
team_id	int / null	팀 ID (랜덤, 균등 배치)
min_off_per_month	int	최소 월 휴무일 (7~9 랜덤)

   예시:

scenario_id	nurse_id	experience_years	is_night_only	team_id	min_off_per_month
1	0	10.23	0	1	8
1	1	4.55	1	0	7
1	2	2.11	0	1	9

데이터셋 - 필요요건 정보

2. 필요요건 데이터 requirements.csv

스키마:

컬럼명	타입	설명
scenario_id	int	시나리오 ID
day	int	날짜 (1~31)
shift_type	str	“D”, “E”, “N”
min_staff	int	최소 필요 인원

예시:

scenario_id	day	shift_type	min_staff
1	1	D	14
1	1	E	13
1	1	N	9

데이터셋 - 업무 선호정보

3. 간호사의 선호 데이터 preferences.csv

스키마:

컬럼명	타입	설명
scenario_id	int	시나리오 ID
nurse_id	int	간호사 ID
day	int	날짜
request_type	str	“OFF”, “PREF_D”, “PREF_E”, “PREF_N”
weight	float	선호 강도 (기본 1.0)

예시:

scenario_id	nurse_id	day	request_type	weight
1	3	2	OFF	1.0
1	12	5	PREF_D	1.0
1	7	10	PREF_N	1.0

환경 셋팅 - State 설정 (Nurse / Global)

Preprocessing & State Representation

Raw 데이터(`nurses.csv`, `requirements.csv`, `preferences.csv`)를 신경망이 이해할 수 있는 2가지 Tensor로 변환합니다.
이후, 아래와 같이 간호사 개인의 State와 근무표 환경의 State로 나누어 모델 설계 시 집중해서 처리할 State를 분산합니다.

2.2.1 Nurse State Tensor

- Shape: $(N, D, 5)$
- 의미: 각 간호사 n, 날짜 d에 대해 5차원 특징 벡터를 가집니다.

Feature Index	내용	예시 (값)
0	Day 근무 여부 (D=1, else=0)	Day 배정 시 1, 아니면 0
1	Evening 근무 여부 (E=1, else=0)	Evening 배정 시 1
2	Night 근무 여부 (N=1, else=0)	Night 배정 시 1
3	Off 여부 (O=1, else=0)	휴무일이면 1
4	휴무 신청 여부 (preferences 기준)	해당 날짜에 OFF 요청이 있으면 1

> 예: 3번 간호사의 5일차 상태가 Evening 근무이고, OFF 요청이 없는 경우
> Nurse State[3, 4] = `[0, 1, 0, 0, 0]`

2.2.2 Global State Tensor

- Shape: $(D, 3)$
- 의미: 날짜 d마다 Shift별 인력 부족분을 나타냅니다.

Feature Index	내용	예시 (값)
0	Day 부족분 (= 필요인원 - 현재인원)	필요 14명, 현재 12명 → +2
1	Evening 부족분	필요 12명, 현재 10명 → +2
2	Night 부족분	필요 10명, 현재 13명 → -3 (여유)

> 예: 10일차에 Day 2명 부족, Evening 딱 맞음, Night 1명 초과인 경우
> Global State[9] = `[2, 0, -1]`

환경 셋팅 - State / Action / Reward

3.1 State (S_t)

- 한 타임스텝 t에서 에이전트는 다음을 관측합니다.
  - Nurse State: $(N, D, 5)$ — 간호사별 현재/과거 근무와 휴무 요청
  - Global State: $(D, 3)$ — 날짜별 D/E/N 인력 부족 정도
- 이 두 텐서는 네트워크 내부에서 각각 1D-Conv로 인코딩된 뒤, 결합되어 하나의 상태 표현 벡터로 사용됩니다.

3.2 Action (A_t)

근무표의 특정 지점을 수정하는 행동입니다. Action Space가 매우 크므로($N \times D \times 4$), 3개의 Head로 분리하여 추론합니다.

- Action Tuple: `(Nurse Index, Day Index, Target Shift)`
  - *Who?*: 어떤 간호사? — `0 ~ N-1`
  - *When?*: 몇 번째 날짜? — `0 ~ D-1`
  - *What?*: 어떤 근무로 바꿀까? — `0=O, 1=D, 2=E, 3=N`

> 예: `(5, 10, 3)` → 5번 간호사의 11일차 근무를 Night(3)로 변경

3.3 Reward (R_t)

변화량 기반의 즉각 보상(Dense Reward)을 사용합니다. $[Rt = *Score*{\text{new}} - Score_{\text{old}}]$

- Score Function 구성:
  - Hard Violation (법적 제약 위반): 위반 1건당 -10점
    - 예: 위반이 3건 → (-30)점 감점
  - Coverage Shortage (인원 부족): 부족 1명당 -5점
    - 예: Day 2명, Night 1명 부족 → (-15)점 감점
  - Soft Preference (선호도 위반): 요청과 다르게 배치 시 -1점
    - 예: OFF 요청 4일 무시 → (-4)점 감점

즉, 에이전트는 법적 제약을 먼저 만족시키면서(큰 벌점 회피), 그 다음으로 인력 부족을 줄이고, 개인 선호를 맞추는 방향으로 학습하게 됩니다.

알고리즘 특징 및 선정 이유

4.1 Baseline: Greedy Bandit (Heuristic)

- 특징: 신경망 없음. 통계적 규칙(Heuristic) 기반.
- 방식: ϵ-Greedy 전략. 인력이 부족한 날짜를 우선적으로 찾아 랜덤하게 수정 시도.
- 선정 이유: RL이 실제로 무작위 탐색보다 나은지 검증하기 위한 기준점(Baseline).

4.2 Value-based: DQN (Deep Q-Network)

- 특징: "행동의 가치"를 학습. ($Q(s, a)$)
- 방식: Experience Replay Buffer를 사용하여 과거 경험을 재학습. Off-policy.
- 선정 이유: 이산적인 행동 공간(Discrete Action Space)에서 가장 대표적인 알고리즘. 샘플 효율성이 높음.

4.3 Policy-based: REINFORCE

- 특징: "최적의 확률 분포"를 직접 학습. ($\pi_\theta(a|s)$)
- 방식: 에피소드 단위로 Monte-Carlo 업데이트. 보상이 높았던 궤적(Trajectory)의 확률을 높임.
- 선정 이유: DQN과 달리 가치 함수 근사 없이 정책을 학습하며, 구현이 직관적임.

4.4 Actor-Critic: PPO (Proximal Policy Optimization)

- 특징: Actor(정책)와 Critic(가치)을 모두 사용하며, 학습의 안정성(Stability)을 극대화.
- 방식: `Clipping` 을 통해 정책이 한 번에 너무 급격하게 변하는 것을 방지.
- 선정 이유: 현재 강화학습 분야의 SOTA(State-of-the-art) 알고리즘. 가장 높은 성능 기대.

실험 세팅

- 학습 / 테스트 시나리오
  - Train: `scenario_id = 1` (2024-01, 31일)
  - Test: `scenario_id ∈ {3, 5, 7, 8, 10, 12, 13, 15}`
    (모두 31일인 달로, 학습된 정책의 일반화 성능을 보기 위한 테스트셋)
- Episodes
  - Bandit: 500 에피소드
  - DQN / REINFORCE / PPO: 각 200 에피소드
- Max Steps per Episode
  - 1 에피소드당 최대 200번의 근무 수정(Action)을 허용한 뒤 종료
- Random Seed
  - 모든 알고리즘과 시나리오에 공통으로 `seed = 42` 사용
- 모델 저장 방식
  - DQN / REINFORCE / PPO는 매 에피소드 종료 시점에
    - Hard Violation 개수가 최소이고,
    - Hard Violation이 같다면 Final Score가 최대인 스냅샷을 "최적 모델"로 저장
  - Bandit은 파라미터를 학습하지 않으며, 탐험률 ε와 휴리스틱 규칙 자체가 모델 설정값 역할을 수행
- 평가 지표 (Evaluation Metrics)

i. Total Reward: 에피소드별 누적 보상 (학습 속도·안정성 확인용)
ii. Hard Violations: 법적 제약 위반 개수 (낮을수록 좋음)
iii. Final Score: 최종 근무표 품질 점수 (높을수록 좋음)

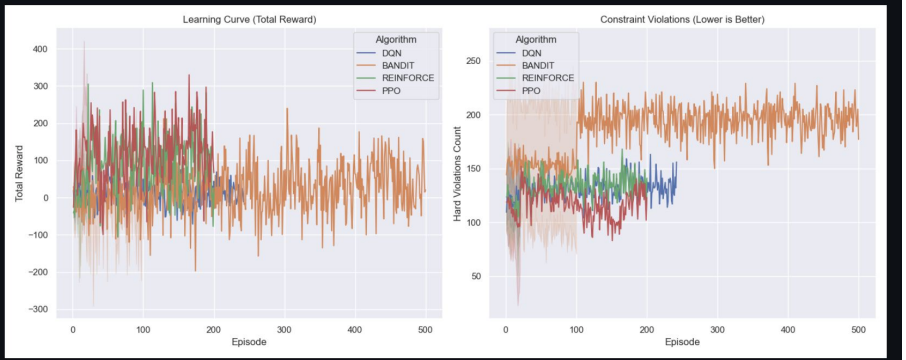
실험 진행 결과

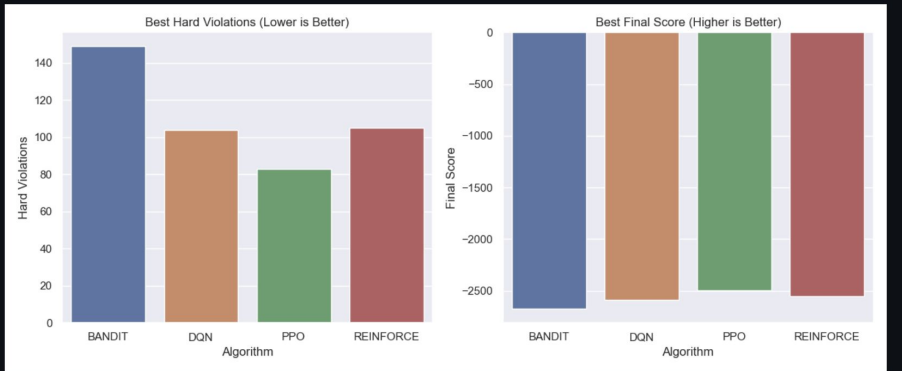
실험 결과는 `logs/` 디렉토리에 CSV로 저장되며, 아래와 같은 그래프로 시각화하여 분석합니다.

1. Learning Curve Comparison:
   - X축: Episode, Y축: Average Reward
   - 4개 알고리즘의 학습 속도와 최종 성능 비교.
2. Constraint Satisfaction:
   - X축: Episode, Y축: Hard Violation Count
   - 규칙을 얼마나 빠르게 깨우치는지 확인.
3. Robustness (Std Dev):
   - Seed 변경에 따른 성능 편차를 Shadow Area로 표시.

실험 진행 결과

6.1 학습 결과





6.2 알고리즘별 요약 비교 (Scenario 1 기준)

아래 표는 동일한 시나리오(Scenario 1)에서 학습/실행한 결과를 요약한 것입니다.
Hard Violation은 낮을수록 좋고, Final Score는 높을수록 좋습니다.

알고리즘	학습 설정	Best Train (Hard / Score)	Test 평균 Hard (시나리오 3,5,7,8,10,12,13,15)
Bandit	500 ep, ε=0.3	≈ 175 / -2688	(별도 평가 시 추가 가능)
DQN	200 ep	104 / -2588	≈ 130
REINFORCE	200 ep	105 / -2554	≈ 131
PPO	200 ep	83 / -2494	≈ 113

대략적으로 PPO > (REINFORCE ≈ DQN) > Bandit 순으로 Hard Violation을 줄이는 데 효과적이며,
Final Score 측면에서도 PPO가 가장 높은 품질의 근무표를 생성하는 경향을 보였습니다.

실험 진행 결과

토의 및 결론 (Discussion)

- PPO의 장점과 보상 스케일 이슈
  PPO는 Bandit / DQN / REINFORCE 대비 일관되게 Hard Violation을 가장 많이 줄였고,
  최종 점수도 가장 높게 나와 본 실험에서 가장 우수한 알고리즘으로 나타났다.
  다만 클리핑을 적용했음에도 Score 차이에 기반한 보상과 손실의 절대 스케일이 여전히 큰 편이라
  Advantage 정규화, Reward 스케일링, Value 표준화 등의 추가적인 안정화 기법을 적용했다면
  액션 선택이 더 뚜렷해지고 학습 곡선의 진동도 줄었을 가능성이 있다.

- 하이퍼파라미터 및 에피소드 수의 한계
  계산 자원 제약으로 인해 각 알고리즘에 대해 단일 시드(42)만 사용했고,
  DQN / PPO는 200 에피소드라는 비교적 짧은 학습만 수행하였다.
  학습률, γ, 네트워크 크기, PPO의 `eps_clip` · `k_epochs` ,
  DQN의 버퍼 크기·배치 크기, Bandit의 ε 등 하이퍼파라미터를 체계적으로 탐색하지 못한 점이 아쉽다.
  더 긴 에피소드와 다중 시드 실험을 통해 알고리즘 간 성능 차이를 통계적으로 검증하는 것이 향후 과제로 남는다.

- 환경·모델링 측면의 제약과 개선 여지
  초기 근무표를 랜덤으로 생성하고, 일부 법적 제약만을 점수 함수로 근사했기 때문에
  실제 병원 현장의 복잡한 규칙(세부 패턴 제약, 팀 배치 전략, 나이트 전담 간호사 로직 등)을 충분히 반영하지 못했다.
  또한 월 일수(31일)에 고정된 상태 인코딩을 사용하여 28일·30일과 같은 다양한 달에 바로 적용하기 어렵고,
  이를 해결하려면 패딩/마스킹 또는 "주(week) 단위 슬라이딩 윈도우"와 같은 보다 일반적인 구조 설계가 필요하다.

- 향후 연구 방향

  i. 보상 스케일·Advantage를 세밀하게 조정하여 PPO/DQN의 학습 안정성과 수렴 속도 향상
  ii. 쉬운 제약부터 점차 난이도를 높이는 Curriculum Learning 도입으로 Hard Violation을 0에 가깝게 줄이는 실험
  iii. CP-SAT로 얻은 해 또는 과거 실제 근무표를 초기 상태로 사용하는 "하이브리드 RL+최적화" 구조 검증
  iv. 테스트 시나리오를 더 다양화하고, 실제 병동 데이터와 비교하는 외부 타당성(External Validity) 평가 수행