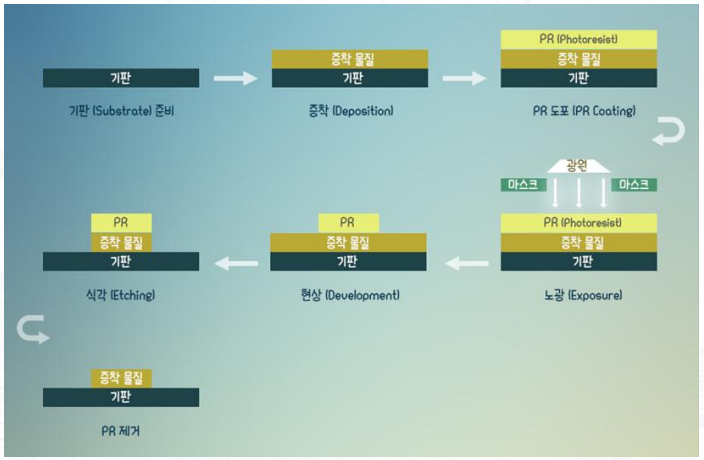
Problem Definition(1)

- AMOLED Manufacturing [1]
  - 반도체 공정과 유사한 복잡한 제조 공정
  - **Backplane** → Evaporation → Assembly
    - Backplane 공정에서, photolithography를 통해 여러 층의 패터닝 작업 수행
  - Photomask(Mask) 리소스 존재
    - Photolithography 단계에서 원하는 회로 패턴을 기판에 전사
  - Photolithography 단계는 AMOLED 제조에 있어, 주요 병목 공정
    - 높은 비용과 큰 설비 공간 필요
    - 디스플레이가 더 높은 해상도와 성능으로 발전하면서 더욱 복잡한 공정 요구
    - 전체 제조라인의 생산 용량을 결정
  - Photolithography 공정은 Unrelated Parallel Machine 문제로 모델링 가능
    - Resource constraints 존재
    - Eligibility restriction 존재
    - Sequence dependent setup 존재

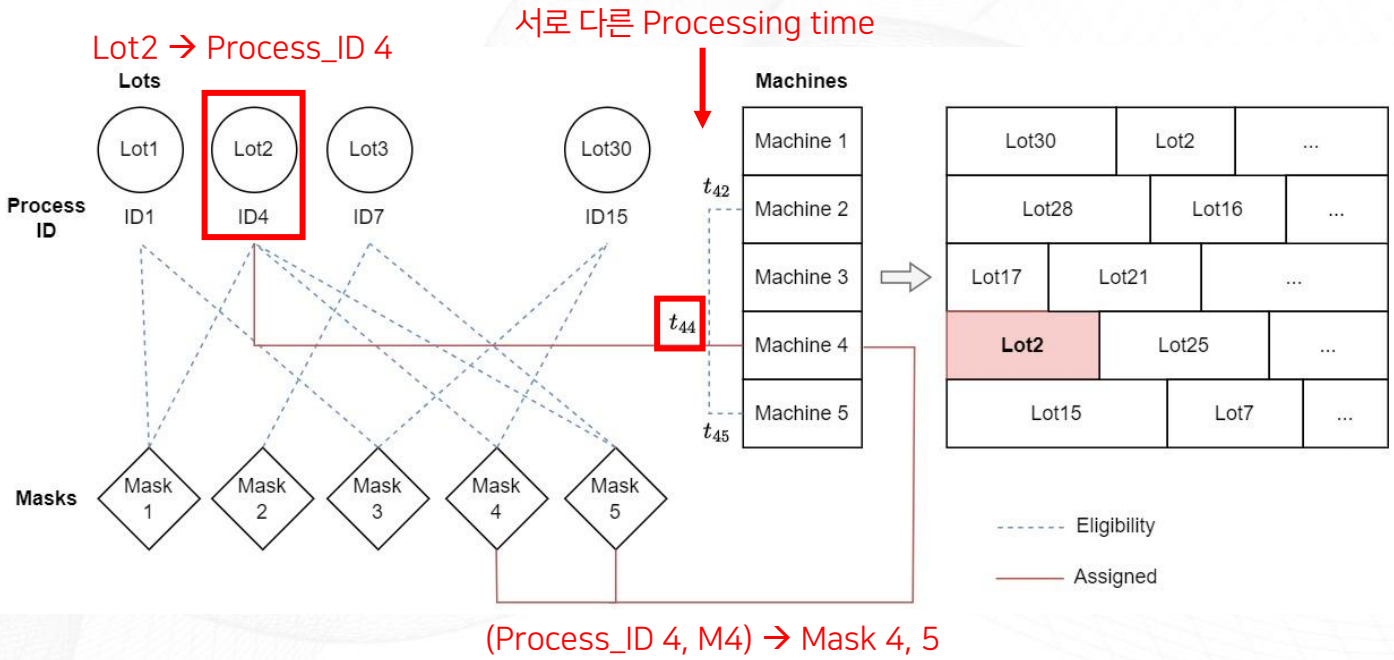


→ **Photolithography 공정의 최적화는 AMOLED 생산의 효율성, 품질, 비용 측면에서 매우 중요**

[1] Kim, Eungjin, et al. "Practical reinforcement learning for adaptive photolithography scheduler in mass production."

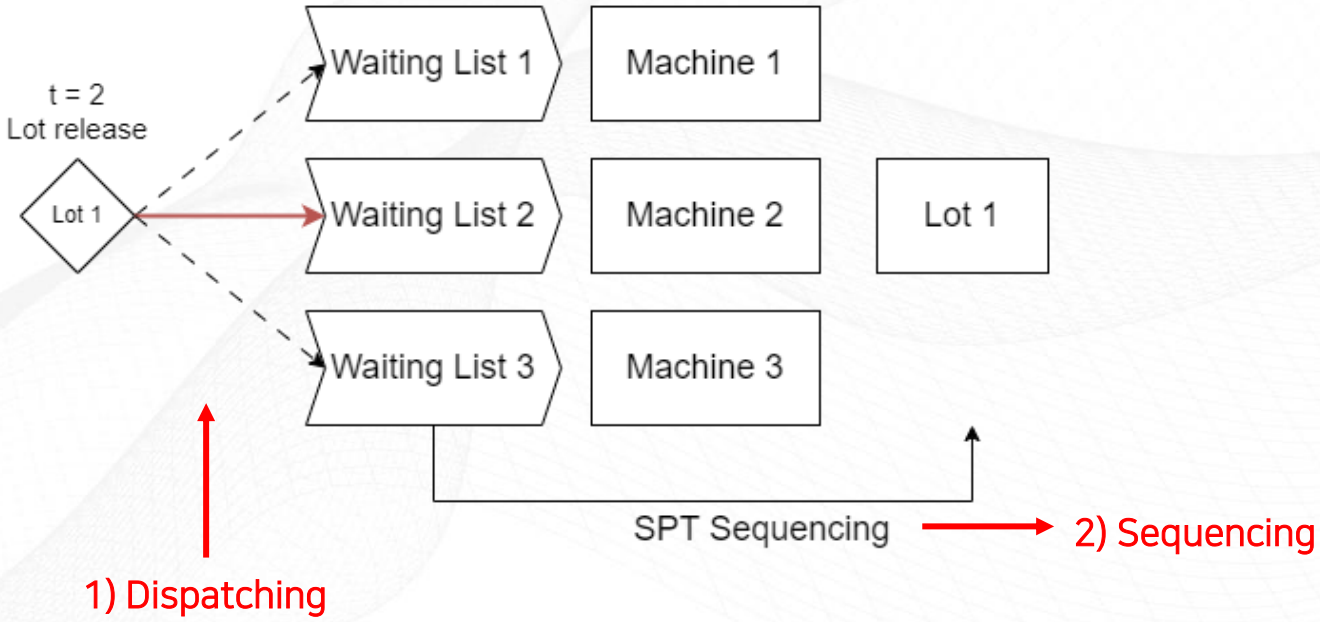
Problem Definition(2)

- Unrelated Parallel Machine Scheduling with Mask Constraints
  - 각 로트마다 Process_ID가 정해져 있고, Process_ID 별 사용 가능한 Machine set와 Mask set가 정해져 있음
  - Process_ID가 사용 가능한 머신 별 처리 시간이 모두 다름
  - Process_ID와 Machine 쌍 별로 사용 가능한 Mask set가 모두 다름
  - Mask는 같은 시간에서, 한 개의 기계에서만 사용 가능
  - 로트 마다 Ready(release) time이 존재함 → 실시간으로 Lot가 도착한다고 가정
  - Machine 별 대기열 존재 → 현실적 상황 고려
  - Minimizing Makespan



Problem Definition(3)

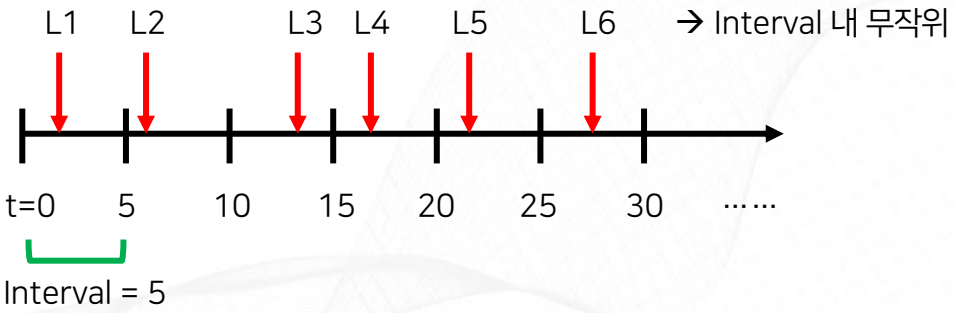
- 2-Step Scheduling
  - **Step 1**: Ready 된 로트를 Machine의 Waiting List에 할당
    - 작업 부하와, 기계 별 처리 시간을 고려한 Dispatching
  - **Step 2**: Waiting List에 있는 로트를 작업에 할당
    - 대기열에 있는 로트들의 실제 처리 순서를 결정
    - 처리 가능한 로트가 대기열에 있다면, 즉시 할당
    - SPT 사용 → Makespan 최소화 목적



Data Preparation

- Data Generation Method
  - 문제의 특성을 반영하는 데이터 생성 함수 작성
  - Lot의 Ready time은 시간 간격을 두고, 랜덤하게 지정
  - (Process_ID 개수) = (Lot 개수) ÷ 2
  - (Mask 개수) = (Machine 개수)
  - Process_ID 별로 사용 가능한 Machine, Mask 모두 랜덤
  - Processing time 20~50 사이로 랜덤

[예시: Interval이 5일때 Ready time]



[예시: Machine 3개, Lot 10개]

	Lot_ID	Process_ID	Ready_time
0	L01	P04	2
1	L02	P03	7
2	L03	P02	10
3	L04	P05	17
4	L05	P01	25
5	L06	P03	27
6	L07	P04	34
7	L08	P05	37
8	L09	P01	41
9	L10	P02	46

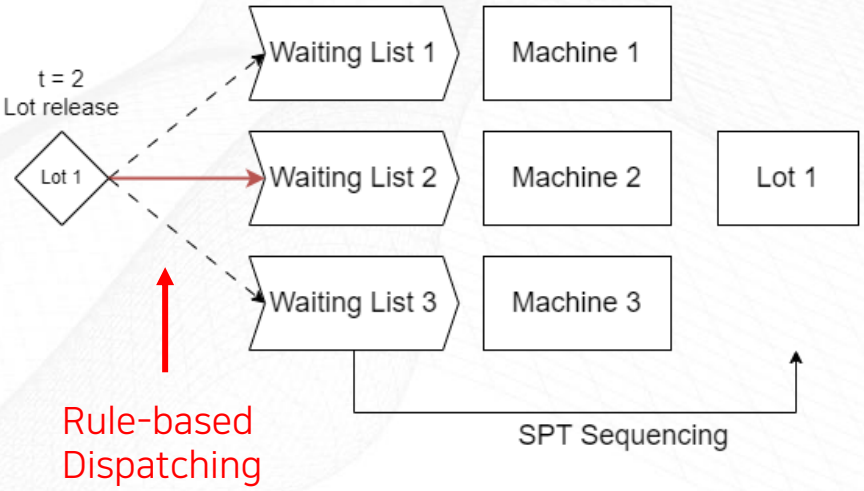
Process_ID 별 사용 가능한 Machine과 Mask 존재

	Process_ID	Machine	Mask_set	Processing_time
0	P01	M1	MSK3/MSK1	45
1	P01	M2	MSK2/MSK3	34
2	P02	M3	MSK1/MSK2	26
3	P02	M1	MSK3/MSK1/MSK2	22
4	P03	M2	MSK3/MSK1	33
5	P03	M3	MSK2/MSK3/MSK1	31
6	P04	M1	MSK2/MSK3	25
7	P04	M2	MSK1/MSK2/MSK3	39
8	P05	M3	MSK1/MSK2/MSK3	26
9	P05	M1	MSK1/MSK2/MSK3	49

Machine 별 Processing time → UPMSP

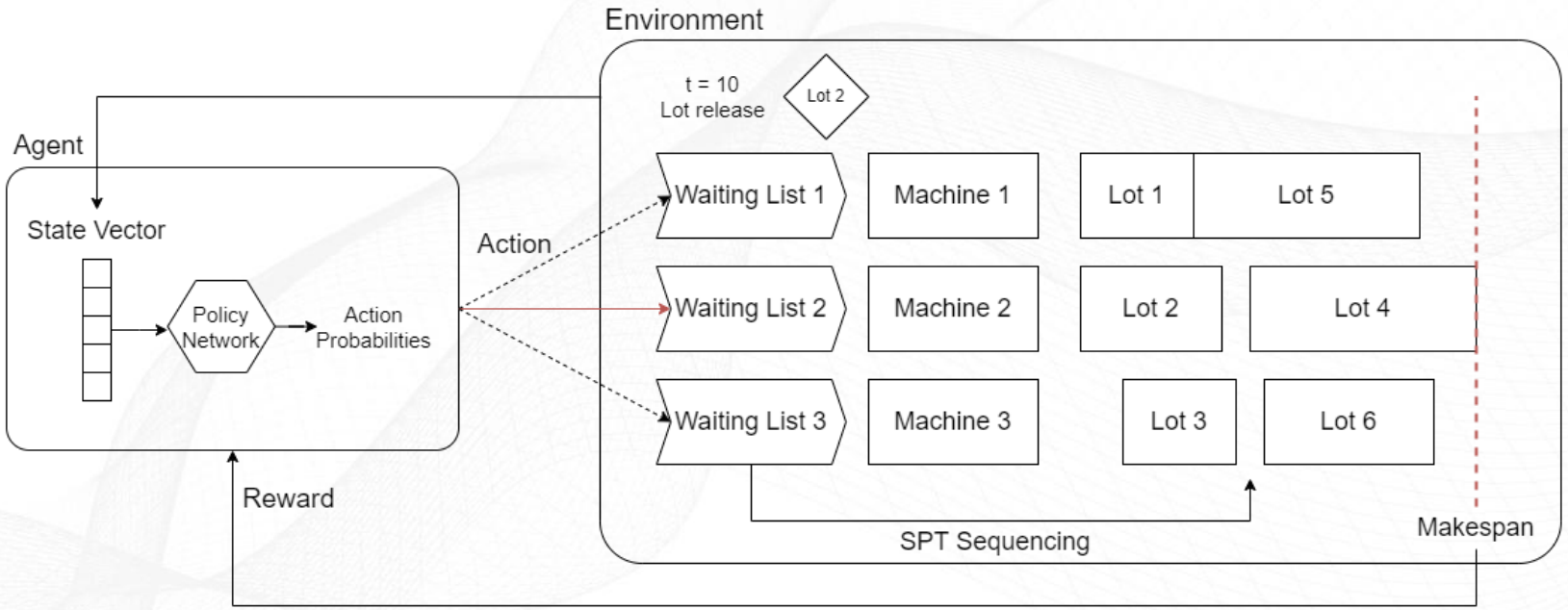
Rule-based Scheduling

- Dispatching Rule
  - Ready time이 된 로트만 처리
  - 하나의 로트가 여러 기계에서 처리 가능한 경우:
    - 누적 Idle time이 가장 큰 기계 우선 선택
    - 동일한 Idle time을 가진 기계가 여러 개면, waiting list 길이가 가장 짧은 기계 선택
    - 그래도 동일한 조건의 기계가 여러 개면, 가장 앞선 번호의 기계 선택
- Sequencing Rule
  - 각 머신의 waiting list에서 처리 가능한 로트 중, processing time이 가장 짧은 것을 선택 (SPT)
  - 하나의 로트가 여러 마스크를 사용할 수 있는 경우:
    - 가장 적게 사용된 마스크 선택



RL based Scheduling

- Dispatching
  - State가 주어졌을 때, Agent는 액션으로 기계를 선택, 로트를 할당 → 후속 로트 정보 부재, Dynamic 한 scheduling 구현 시도
- Sequencing
  - 각 머신의 waiting list에서 처리 가능한 로트 중, processing time이 가장 짧은 것을 선택 (SPT)
- Reward
  - 한 에피소드가 모두 끝난 후, 최종 Makespan의 음수를 reward로 반환



RL based Scheduling – Environment(1)

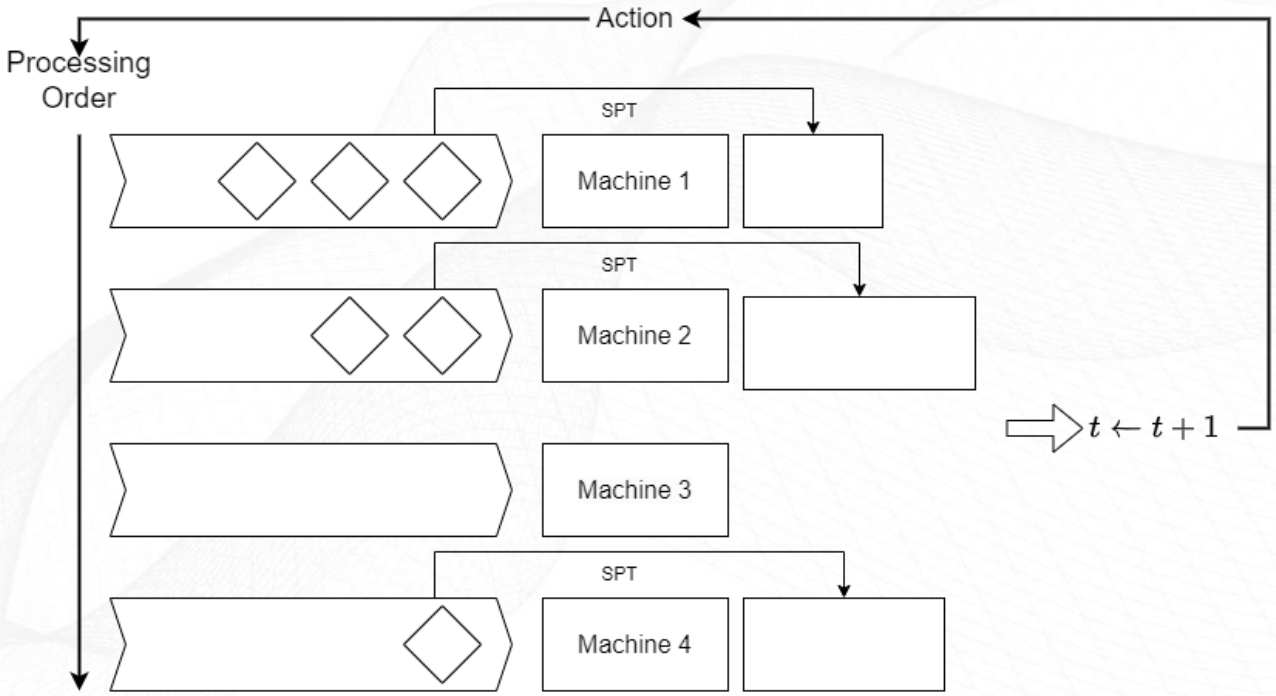
- State Description
  - 어떤 한 시점에서, 환경의 상태를 1D Vector로 표현

State Components	Normalization	Explanation
각 기계 별 대기열의 로트 수	전체 로트 수	각 기계의 작업 부하 상태를 파악
각 마스크의 사용 가능 여부 (0 또는 1)	-	현재 사용 가능한 Mask가 무엇인지 파악
각 마스크의 총 사용 횟수	전체 로트 수	Mask 활용의 균형을 맞출 수 있도록 유도
다음 ready 로트의 각 머신에서의 처리 시간	Current makespan	다음 작업의 처리 시간을 고려하여 효율적인 기계 선택
각 기계의 총 처리 시간	Current makespan	각 기계의 작업 부하 상태를 파악
각 기계의 완료된 작업 수	전체 로트 수	각 기계의 생산성 파악
각 기계의 유휴 시간	Current makespan	비효율적인 스케줄 생성 방지
각 기계 대기열의 총 처리 시간	Current makespan	앞으로 예상되는 작업 부하 파악
각 기계의 idle 상태 (Idle이면 1, 아니면 0)	-	즉시 새로운 작업을 할당할 수 있는 머신 파악

→ [State1, State2, ⋯]로 이어 붙여서 1D vector로, 각 시점에서의 State를 표현

RL based Scheduling – Environment(2)

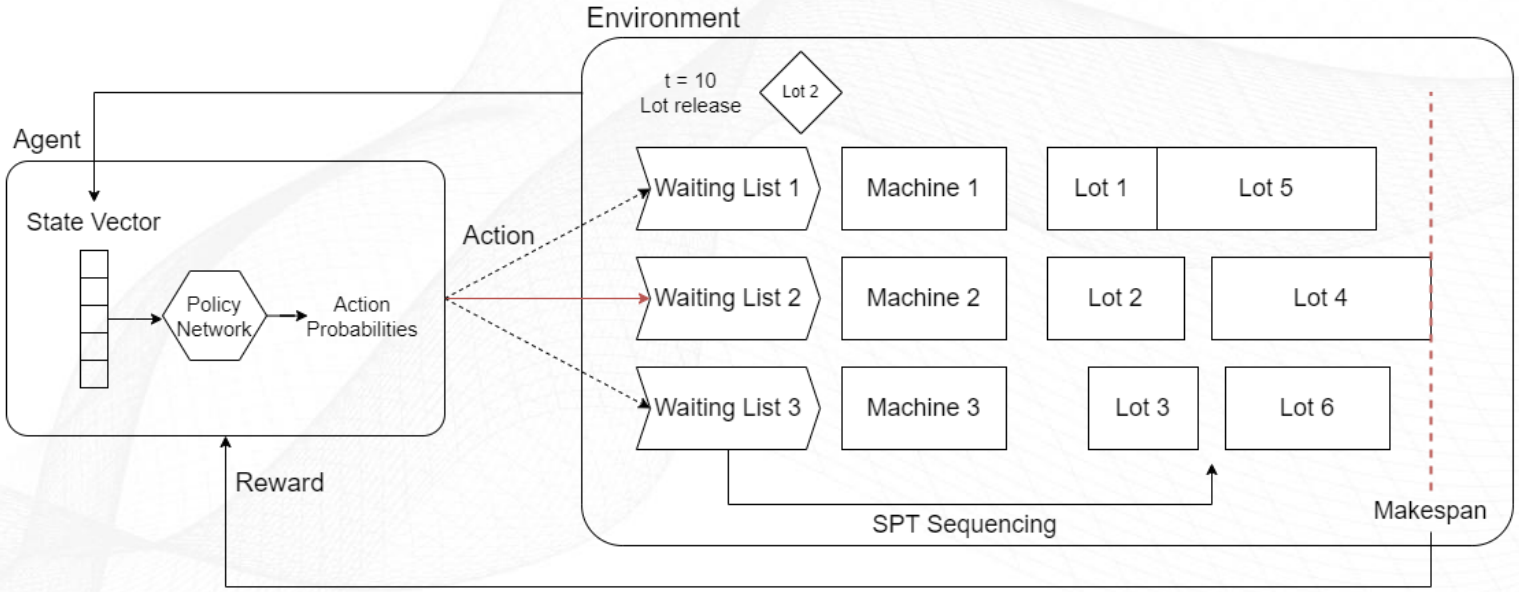
- Environment Step
  - MatNet의 FFSP 환경 구현과 유사하게 구현
  - 주어진 순서대로 각 기계를 순회하며 작업을 처리
  - 모든 기계를 순회하면, 전역 시간 t 증가
    - Ready 된 로트가 있다면, Action
    - Ready 된 로트가 없다면, 다음 ready 시간까지 시뮬레이션 진행
  - 모든 로트가 처리 완료되면 에피소드 종료



RL based Scheduling – Agent

- Policy-based Agent
  - Multi-Layer Perceptron(MLP) 구조의 정책 네트워크
  - 어떤 로트가 Ready 되었을 때 State 벡터를 입력으로 받으면, 액션으로 기계 중 하나를 선택
  - 후속 로트 정보 부재, Lot가 Ready 되면 Dispatching 역할 만 수행

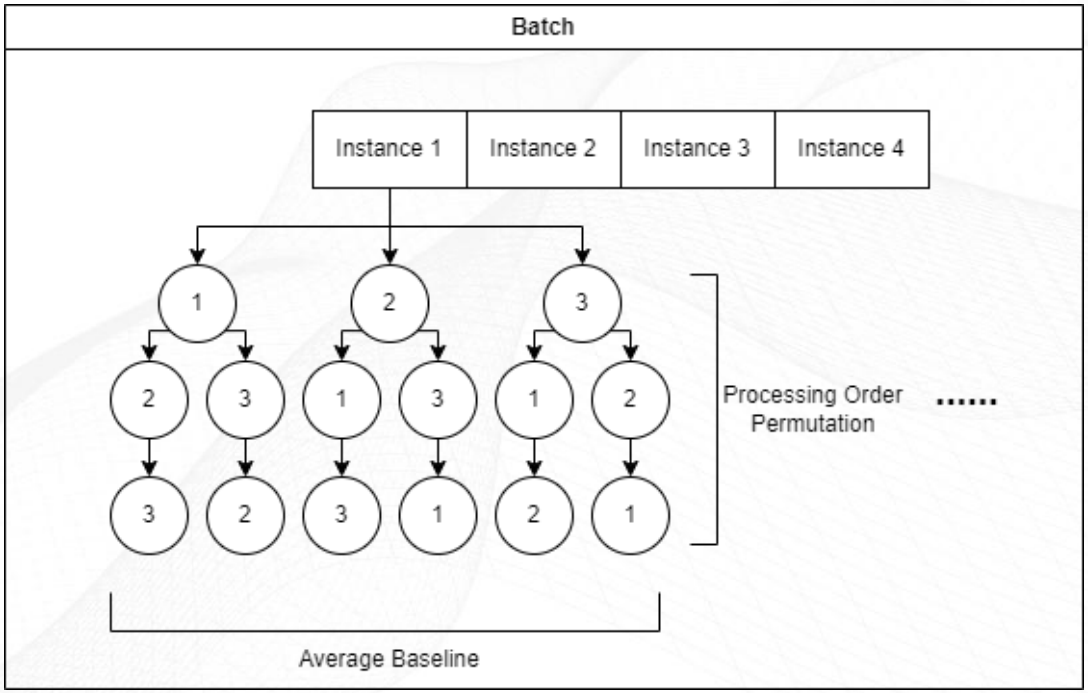
- REINFORCE Algorithm
  - 에피소드 종료 후, makespan에 음수를 취해 보상으로 반환
  - MatNet 코드에서 FFSP 문제를 풀 때 POMO baseline을 사용한 것을 참고



RL based Scheduling – Train

- Training
  - MatNet 코드 구현에서 착안하여, 한 기계 처리 순서를 POMO의 한 Trajectory로 간주 → n개의 기계: $n!$개 생성
  - 분산 감소를 위한 baseline으로 모든 trajectory의 평균을 사용
  - 한 Batch를 여러 인스턴스로 구성, 배치의 평균 loss로 업데이트 $\nabla_\theta J(\theta) \leftarrow \frac{1}{BN}\sum_{i=1}^{N}(R(\boldsymbol{\tau}_i^j) - b_i)\nabla_\theta \log p_\theta(\boldsymbol{\tau}_i^j)$ [2]

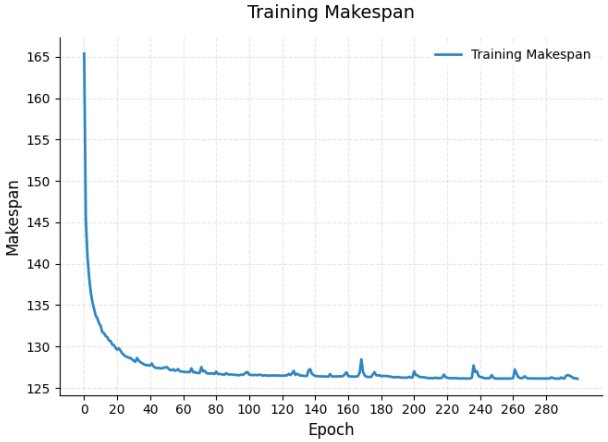
[예시: 머신이 3대 일때]

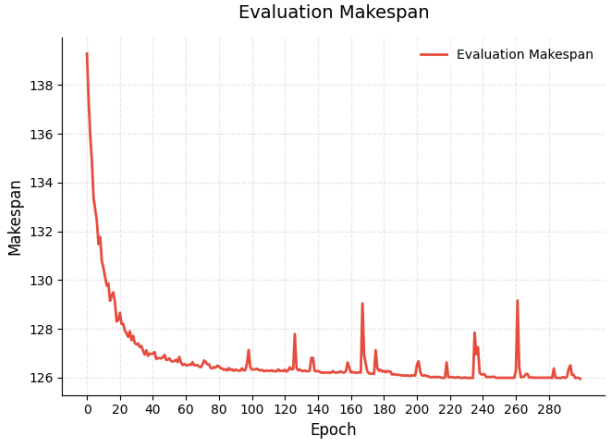


[2] Kwon, Yeong-Dae, et al. "Pomo: Policy optimization with multiple optima for reinforcement learning."

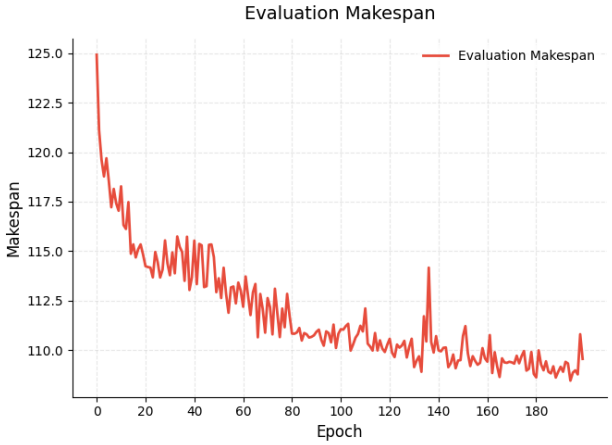
RL based Scheduling – Train Result(1)

[Machine 3개, Lot 10개]



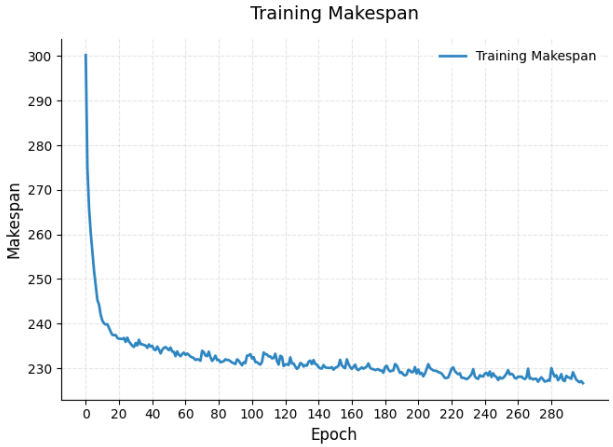


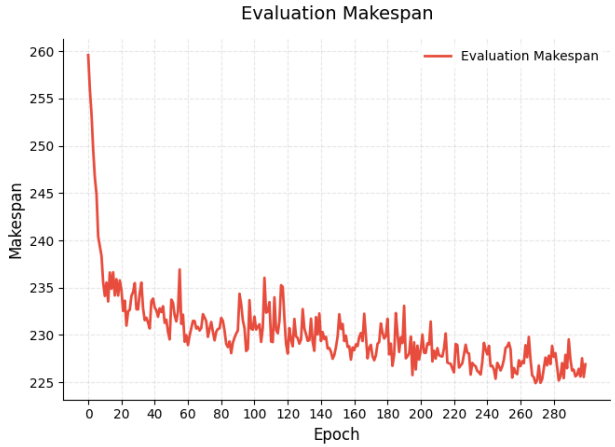
[Machine 4개, Lot 10개]

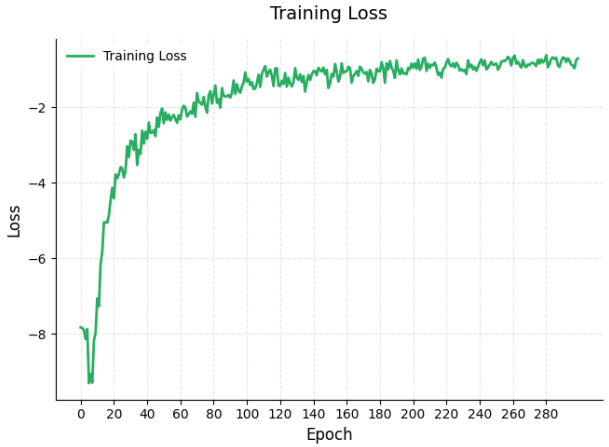


RL based Scheduling – Train Result(2)

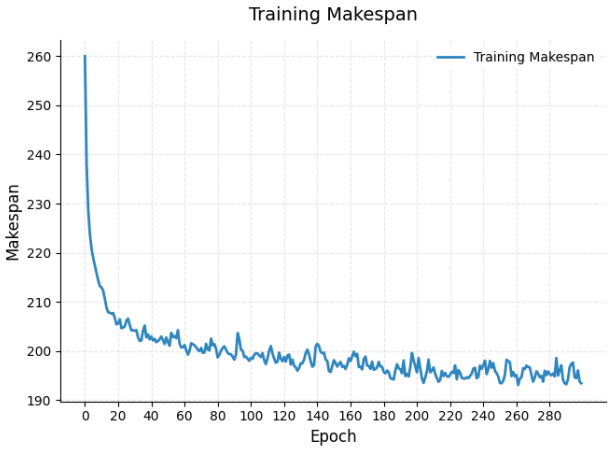
[Machine 3개, Lot 20개]

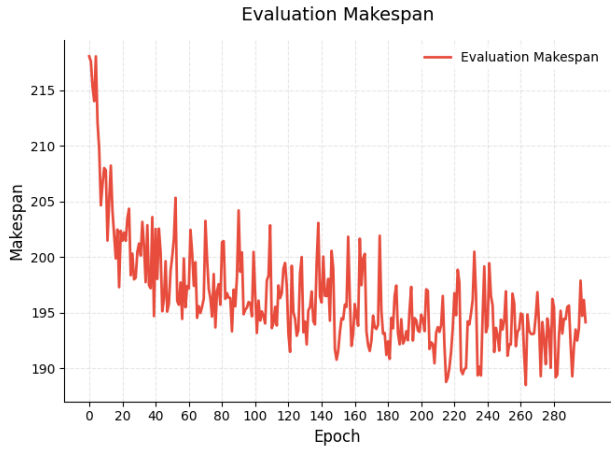


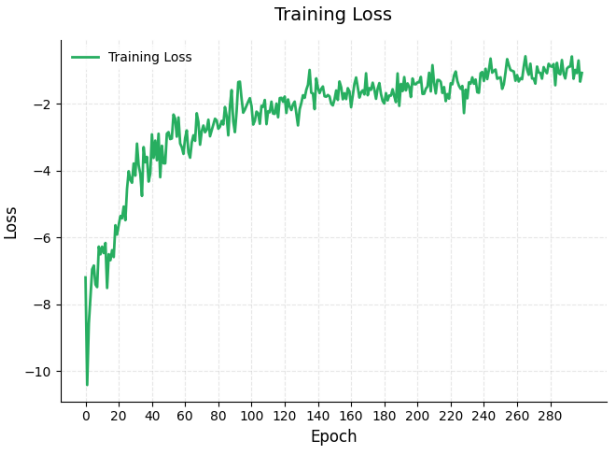




[Machine 4개, Lot 20개]







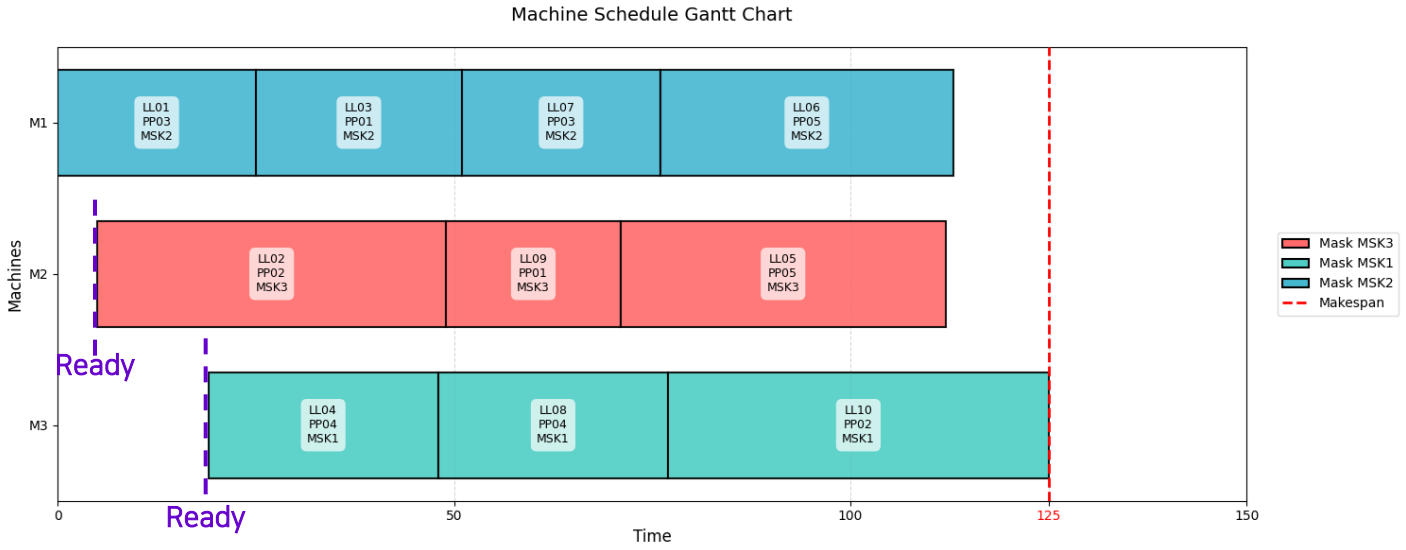
Model Comparison – RL vs Rule-based(1)

- RL vs Rule-based Scheduling
  - 각각의 케이스에 대하여 랜덤하게 생성된 100개의 인스턴스로 평가 결과 산출
  - 간결한 Rule-based 모델 보다는 RL이 더 짧은 Makespan을 보여줌
    - Rule-based 모델은 로트가 Ready가 되면 누적 Idle Time이 가장 작은 머신에 무조건적으로 할당 → 유연성 저하

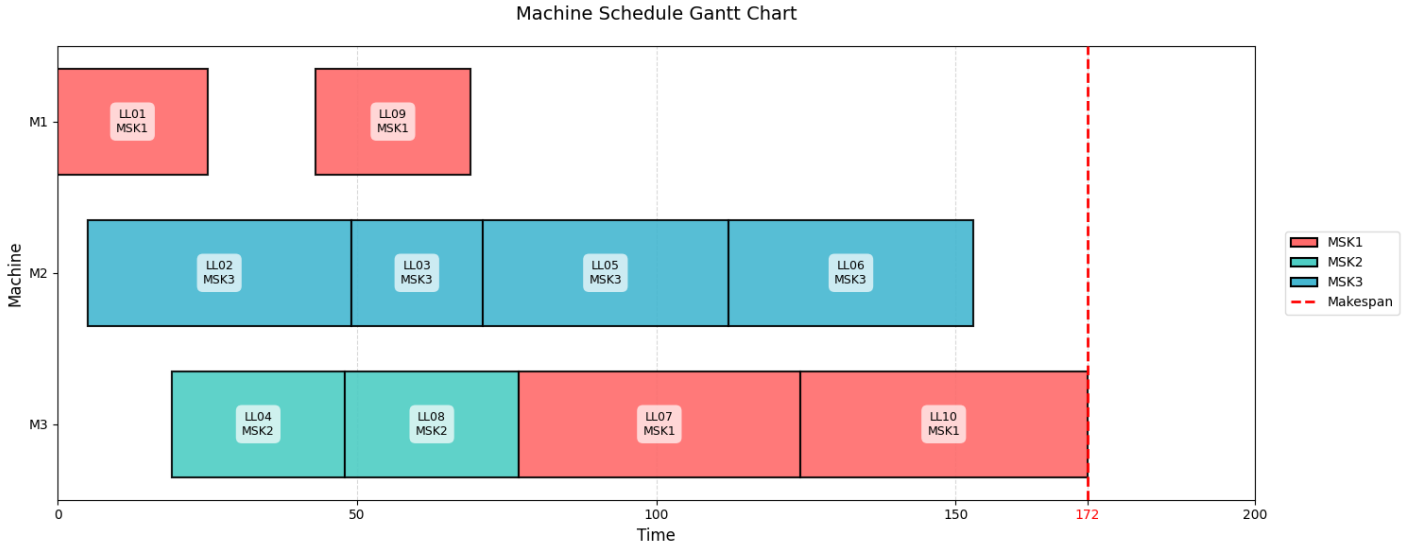
# of Lots	# of Machine	RL	Rule-based
10	3	129.85	213.03
10	4	118.05	175.71
20	3	234.31	356.15
20	4	198.31	300.83

Model Comparison – RL vs Rule-based(2)

[RL]

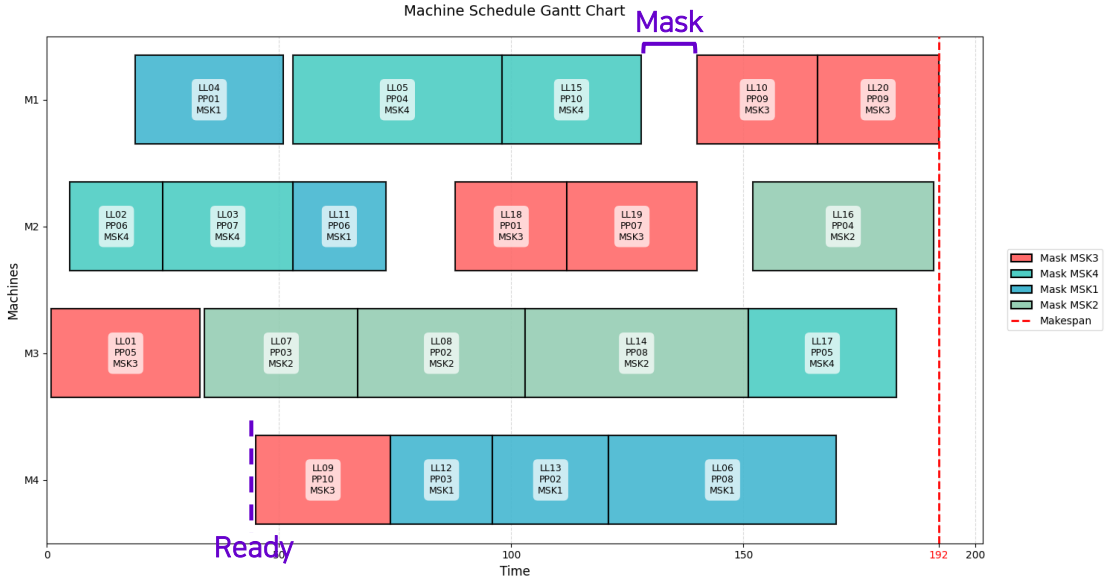


[Rule-based]

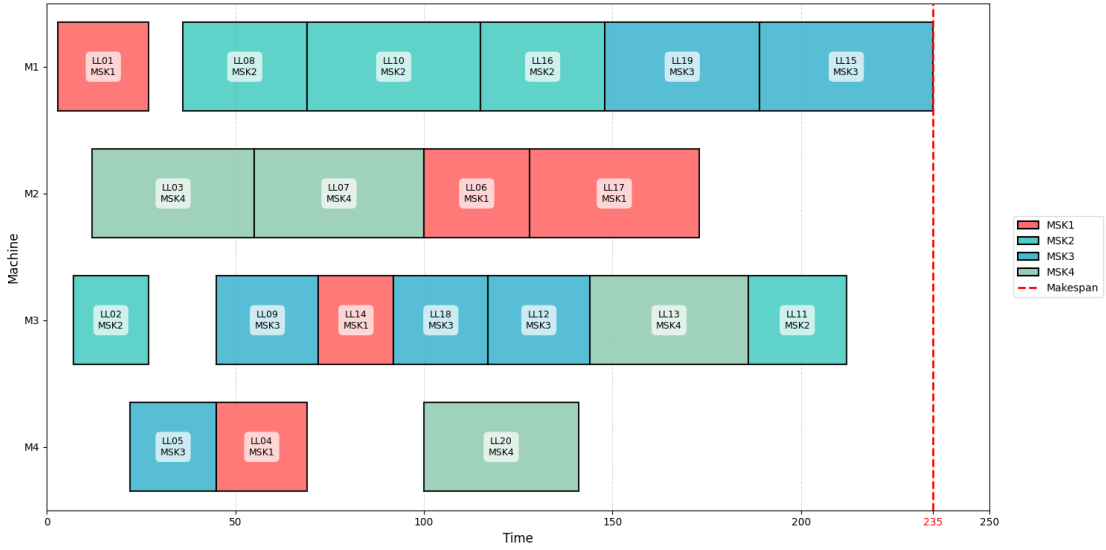


Model Comparison – RL vs Rule-based(3)

[RL]



[Rule-based]



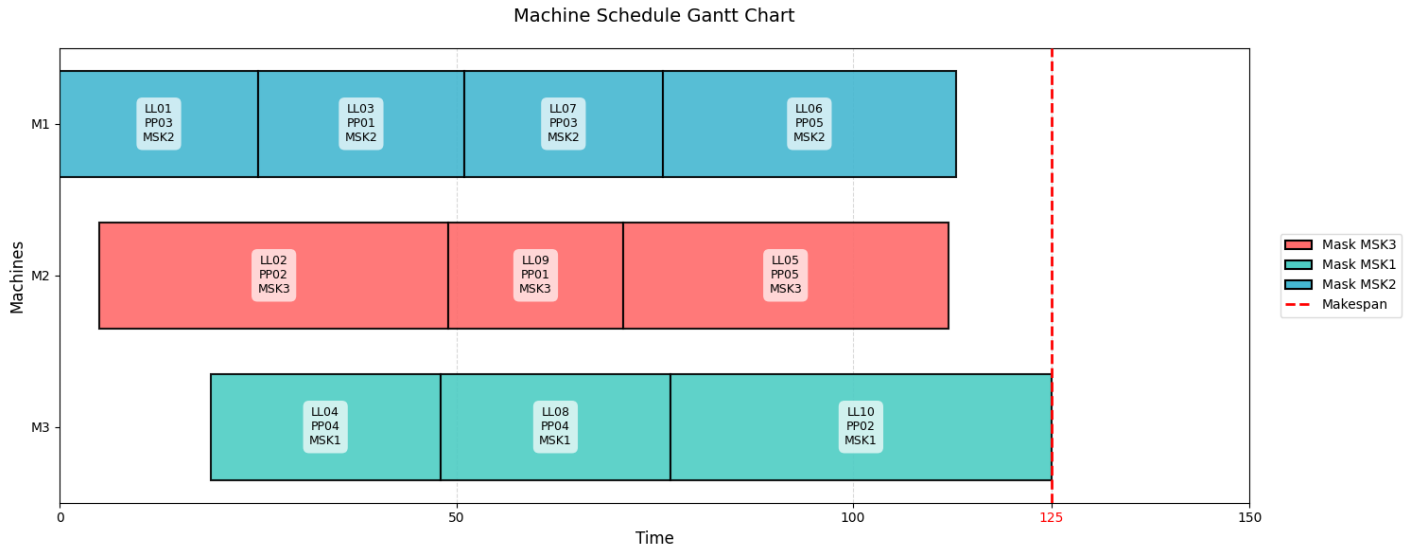
Model Comparison – RL vs CP(1)

- RL vs CP
  - 각각의 케이스에 대하여 랜덤하게 생성된 10개의 인스턴스로 평가 결과 산출
  - CP는 최적해를 찾는 반면에, RL은 최적해와 차이가 있는 것을 보임
  - RL은 실행 속도가 매우 빠르지만, CP는 로트 수가 늘어날 수록 기하급수적으로 시간이 증가
    - 10개 로트에 대해서는 10초 이내로 최적해를 찾지만, 20개 로트에 대해서는 최대 5~6분까지 소요
  - 후속 로트 정보에 따른 차이 존재
    - CP는 모든 로트의 정보를 알고, 그 정보를 기반으로 최적해를 찾음
    - RL은 후속 로트에 대한 정보가 부재, 로트가 Ready 되면 Dispatching 하는 방식

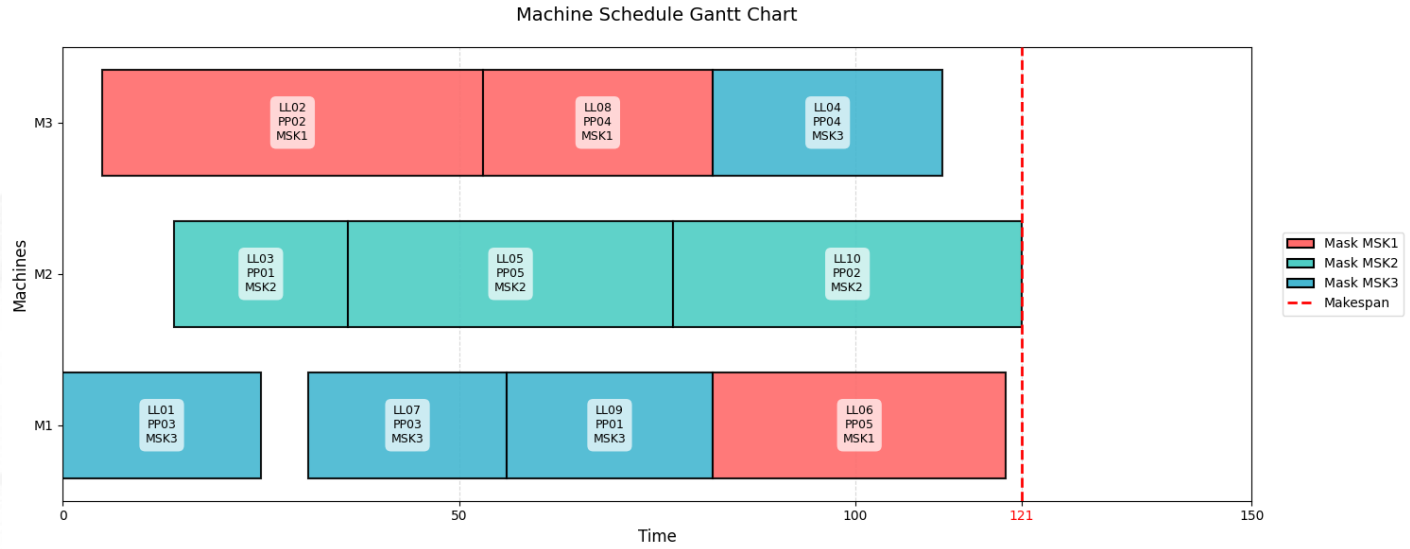
# of Lots	# of Machine	RL	CP
10	3	129.7	117.1
10	4	120.4	99.7
20	3	233.5	219.1
20	4	198.4	168.7

Model Comparison – RL vs CP(2)

[RL]

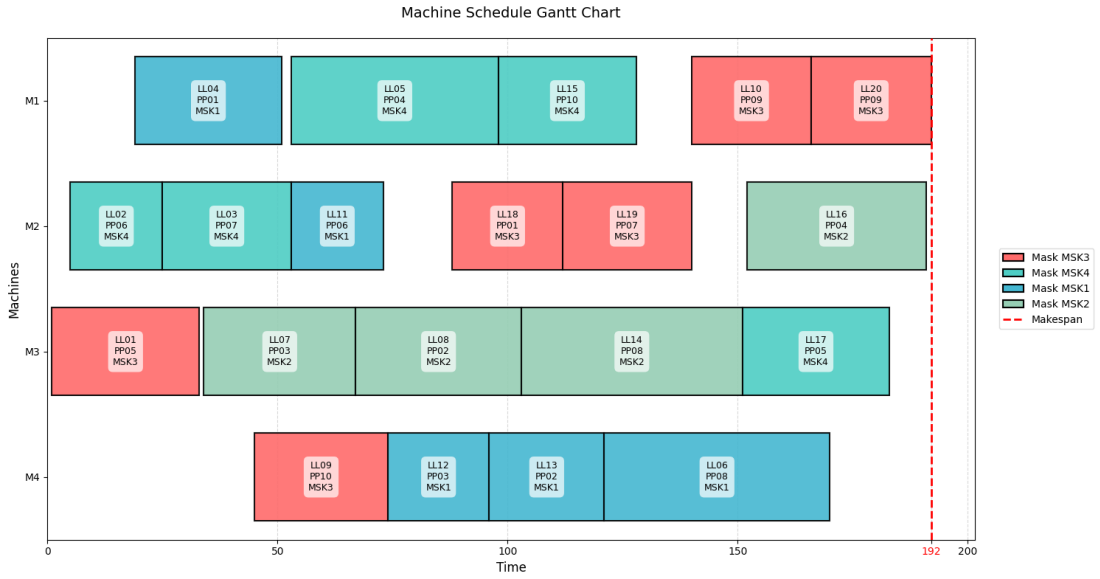


[CP]



Model Comparison – RL vs CP(3)

[RL]



[CP]

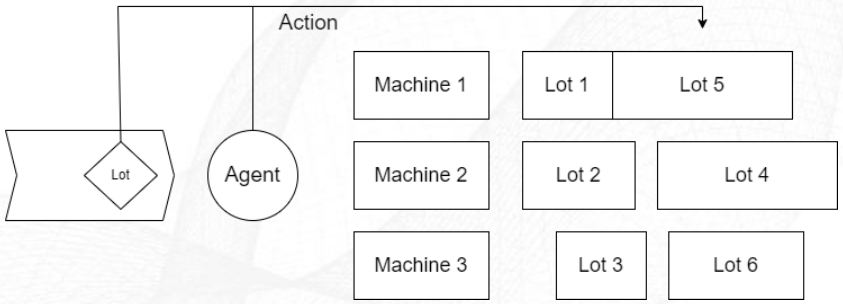
Limitations & Next Steps (1)

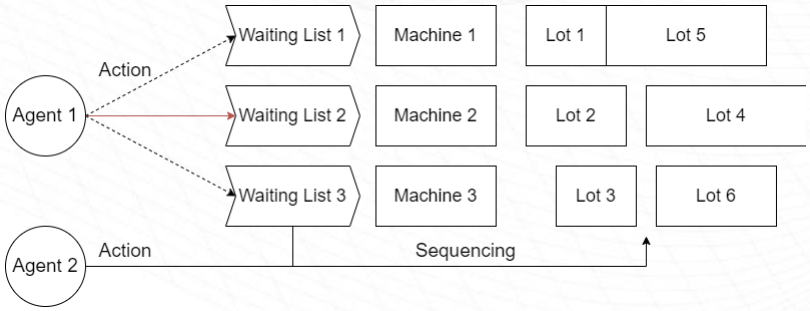
- Advanced State Description
  - 현재 State는 Mask에 대한 정보가 부족하다는 한계 존재
  - Agent가 기계들의 Workload Balance를 고려하게끔 학습되었다고 추측 → Mask 가용에 대한 세부적인 정보 필요
  - Ready 된 로트의 정보와 현재 환경 간의 관계를 효과적으로 표현할 수 있는 추가적인 State 표현 방법 도입 (CNN, GNN 등)

  → 현재 상황에서 중요한 정보를 추출하여 전달, Agent가 더 최적의 결정을 할 것이라고 기대

- Entire Scheduling Architecture
  - 현재는 2-step Scheduling → 직접 로트를 스케줄링하는 방식으로 전환
  - SPT 규칙으로 할당되는 것이 아닌, 에이전트가 직접 할당하여 2-step 보다 유연한 스케줄링을 할 것으로 기대

  - 또는, 현재 2-step Scheduling 구조에서 또 다른 Agent 도입 → Dispatching Agent / Sequencing Agent
  - Sequencing이 SPT 규칙으로 할당 되는 것이 아니라, Agent가 담당하여, SPT 보다 최적의 결정을 할 것으로 기대





Limitations & Next Steps (2)

- Objective
  - 현재는 Makespan 최소화가 목적
  - Due date를 도입하여, Tardiness와 같은 새로운 목적식으로 모델링 가능

- Additional Constraints
  - 현재는 Mask 이외의 제약은 고려하지 않음
  - 실제 Photolithography 공정에서는 잦은 Setup과 Transfer을 지양
  - Mask의 Setup time, Transfer time 등을 고려하여 모델링

- Environment Architecture
  - 현재 환경은 전역 시간 t를 1씩 증가시키며 진행 → 매우 비효율적
  - Event 기반 시뮬레이터를 구축하여, 의미 있는 시간만 체크
  - 효율적인 시간 진행으로 빠르고 효율적으로 학습이 가능할 것으로 기대