

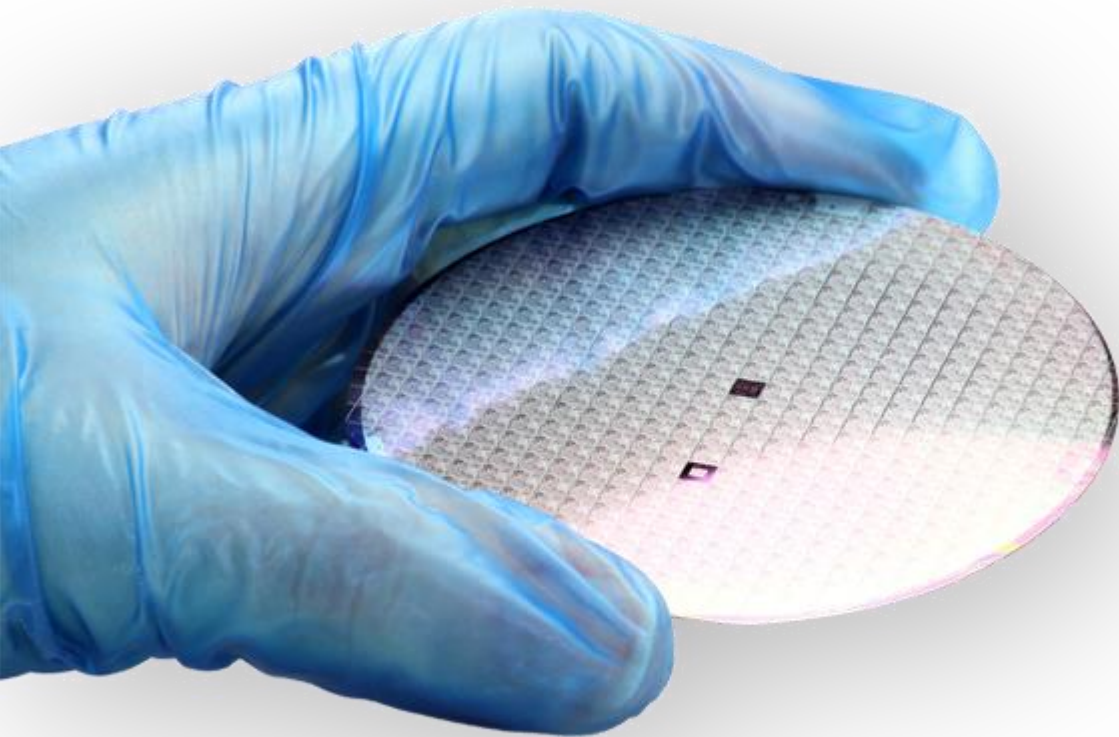


강화학습 논문 리뷰 스터디 8기

Optimization of global production scheduling with deep reinforcement learning

김용회(Kim Yong Hae)

논문에 앞서 알아 두면 좋은 것들



45억



20

조



공정

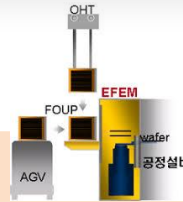
DIFF

PHOTO

ETCH

THIN

물류/반송



생산설비



자동화
시스템

Application	MES	OOO	OO	OO	ooo	...	OOO
OS/Middle Ware	OS	DBMS	Web Server	WAS	Messaging	...	EAI
Infra	Server			Network			



Max **3000** s Step



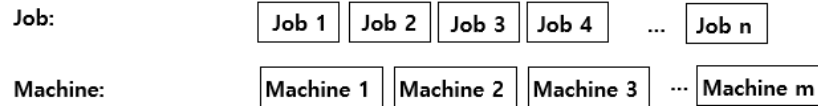
수억

~

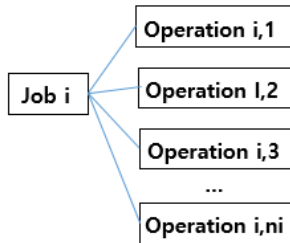
수천억



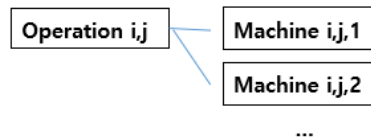
Job shop scheduling?



Job의 구성:
각 job마다 상이한
종류/개수의
operation의 집합으로 구성



일반적으로 하나의
operation에서는 수행할 수 있는
machine 후보가 여러 개 있을 수 있음.



개별 operation은 하나의 machine에서 수행

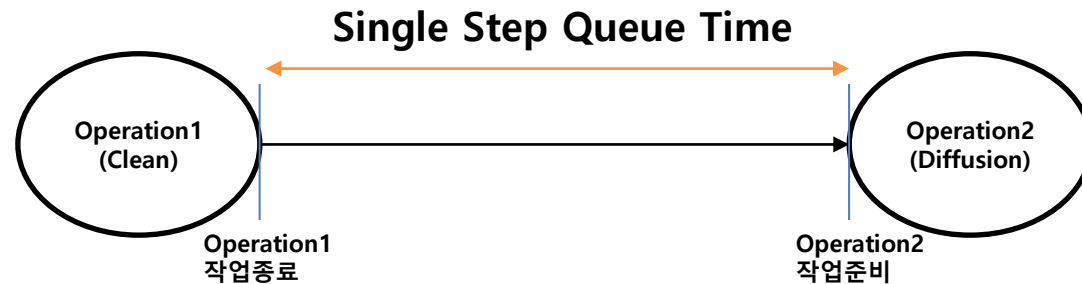


기계에서 operation 수행하는 시간: $p_{i,j,k}$ (setup time을 원하면 포함 가능)

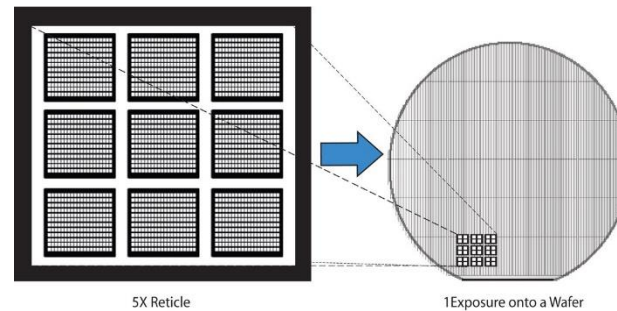


various constraints

e.g., Queue Time



e.g., Reticle Management



순번	제목	구분	분류
1	병렬설비를 위한 주기적 일정계획	국내/논문	2019
2	Genetic Algorithm을 이용한 Job-Shop 공정의 생산 스케줄링 최적화 분석	국내/논문	2019
3	유전 알고리즘을 이용한 두 단계 이중병렬기계 시스템의 일정계획	국내/논문	2021
4	대기시간을 허용하지 않는 두 단계 조립시스템에서 총 작업완료시간 최소화를 위한 일정계획 연구	국내/논문	2019
5	시간 제약과 순서 의존 준비시간이 있는 흐름공정의 일정계획 휴리스틱 알고리즘 연구	국내/논문	2020
6	긴급주문과 순서의존 준비시간이 있는 흐름공정의 일정계획 휴리스틱 알고리즘 연구	국내/논문	2020
7	유전알고리즘을 활용한 중첩 대기시간 제약 플로우샵 스케줄링	국내/논문	2021
8	정규화 학습을 이용한 심층 강화학습 기반 반도체 패키징 라인 스케줄링 기법의 강건성 향상	국내/논문	2019
9	심층강화학습 기반 반도체 패키징 라인의 생산일정계획	국내/논문	2022
10	Deep reinforcement learning based scheduling within production plan in semiconductor fabrication	해외/논문	2022
11	Scalable Scheduling of Semiconductor Packaging Facilities Using Deep Reinforcement Learning	해외/논문	2021
12	Agent-based approach integrating deep reinforcement learning and hybrid genetic algorithm for dynamic scheduling for Industry 3.5 smart production	해외/논문	2021
13	A Practical Deep Reinforcement Learning Approach to Semiconductor Equipment Scheduling	해외/논문	2021
14	Simulation and deep reinforcement learning for adaptive dispatching in semiconductor manufacturing systems	해외/논문	2021
15	A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities	해외/논문	2019

디지털 트윈

위키백과, 우리 모두의 백과사전.

디지털 트윈(digital twin)은 미국 제너럴 일렉트릭(GE)이 주창한 개념으로, 컴퓨터에 현실 속 사물의 쌍둥이를 만들고, 현실에서 발생할 수 있는 상황을 컴퓨터로 시뮬레이션함으로써 결과를 미리 예측하는 기술이다. 디지털 트윈은 제조업뿐 아니라 다양한 산업·사회 문제를 해결할 수 있는 기술로 주목 받는다.^{[1][2]} 그리고 기본적으로는 다양한 물리적 시스템의 구조, 맥락, 작동을 나타내는 데이터와 정보의 조합으로, 과거와 현재의 운용 상태를 이해하고 미래를 예측할 수 있는 인터페이스라고 할 수 있다. 물리적 세계를 최적화하기 위해 사용될 수 있는 강력한 디지털 객체로서, 운용 성능과 사업 프로세스를 대폭 개선할 수 있다.^[3]



활용 영역 [편집]

- 공장 최적화^[4]
- 가상 제조^[5]
- 센서를 활용한 자동화^[6]

원리 [편집]

생명체의 환경 변화 반응 기제를 떠올리면 이해가 쉬울 수 있다. 실제로 인간의 몸은 부단히 변화하는 환경에서도 체온과 혈류(血流) 등 모든 조건을 평형 상태로 유지해야 한다. 그러기 위해 인체 표면과 내부에선 무수한 센서 세포가 끊임없이 활동 중이다. 이 세포들은 큰 작든 이상(異常)이 감지되면 다른 세포에 신호를 보내 해당 이상으로 입은 피해를 복구하고 추가 피해도 막는다. 익히 알려진 면역 기능의 작동 과정이다.^[7] 생산 공정, 혹은 이미 생산된 제품의 사용 과정에도 이와 비슷한 원리를 적용할 수 있다. 가상 복제와 비슷한 방식으로 디지털 트윈을 만들고 공정 관리자의 태블릿에 이를 제어할 수 있는 프로그램을 심은 후 생산과 소비의 전 과정에 센서를 설치해 거기서 발생하는 신호가 태블릿 속 디지털 트윈에 실시간으로 반영될 수 있도록 하는 것이다. 이렇게 되면 특정 제품(혹은 공정)의 디지털 트윈 프로그램 공유자는 언제 어디서나 제품 관련 문제 발생 여부를 실시간으로 알 수 있게 된다. 그와 거의 동시에 이들의 집단지성을 기반으로 최적의 솔루션이 도출, 현장에 곧장 전달되고 가장 적절한 조치가 취해진다. 모든 제품이 이런 방식으로 제작, 관리되면 생산 공정 오류로 인한 비용 손실을 줄일 수 있을 뿐 아니라 소비자 요구에도 한층 더 완벽에 가깝게 부응할 수 있다.^[8]

※ 출처 : https://ko.wikipedia.org/wiki/%EB%94%94%EC%A7%80%ED%84%B8_%ED%8A%B8%EC%9C%88

논문 리뷰

- 반도체 제조와 같은 복잡한 Job Shops 스케줄링에 심층 강화 학습을 적용
- 반도체 업종은 전통적으로 메모리칩과 같은 소규모 제품 포트폴리오를 가지고 있었으나 사물인터넷 등으로 인해 소량 생산 센서와 같은 광범위한 주문형 칩을 필요로 함
- 소형화(미세공정을 의미), 수율 개선, 웨이퍼 크기 증가(200mm -> 300mm)와 같은 전통적인 생산 효율 개선 방법은 최대로 활용되고 있음
- 반도체 생산 시설의 부분들과 같은 작은 규모의 생산 스케줄을 수학적 최적화를 통해 해결될 수 있음, NP-hard와 같은 큰 동적 환경 모델의 복잡성과 실행시간은 JSP(Job-Shop Scheduling Problem) 응용프로그램을 제한함
- 결과적으로 최적화는 지역적으로 사용되고 workcenter 들에서 분리됨
- 복잡한 Job shop에서 이러한 생산 스케줄링의 지역 최적화는 생산을 위한 비최적화 글로벌 솔루션으로 이어질 수 있음



- 본 논문에서 DQN 에이전트를 생산 스케줄링에 사용하며, 최적화를 위해 유연한 사용자 정의 목표를 가진 RL 환경에서 훈련함
- 각 DQN 에이전트는 다른 에이전트의 작업을 모니터링 하고 글로벌 상을 최적화 하면서 한 workcenter 에서 규칙을 최적화하며, 규칙은 시뮬레이션에서 직접 테스트 되고 개선됨
- 레거시 시스템의 데이터로 훈련할 수도 있고 시뮬레이션 환경에서 완전 새로운 훈련을 하는 것도 가능
- 장점
 - 유연성 : 에이전트는 몇시간 안에 retrained 될 수 있음
 - 글로벌 투명성 : 신경망은 사람의 경험에 기반한 계층적 구성과 같은 제약에 구속되지 않고 목표에 적합한 균형을 모델하는 방법을 가짐
 - 글로벌 최적화 : DQN 에이전트 시스템은 로컬이 아닌 글로벌하게 자동으로 최적화되며, 생산 목표를 수동으로 세분화할 필요 없음
 - 자동화 : 디스패칭 룰이 전문가에 의해 구현될 필요 없음
 - 연속성 : DQN 에이전트는 기존 디스패칭 시스템에서 Pre-trained 될 수 있음.
- 단점
 - 훈련에 필요한 계산 비용이 많이 듭혀들고, 신경망은 블랙박스 모델이기 때문에 DQN 에이전트가 알려지지 않은 상황에서 어떻게 행동할지 예측 하기 어려움

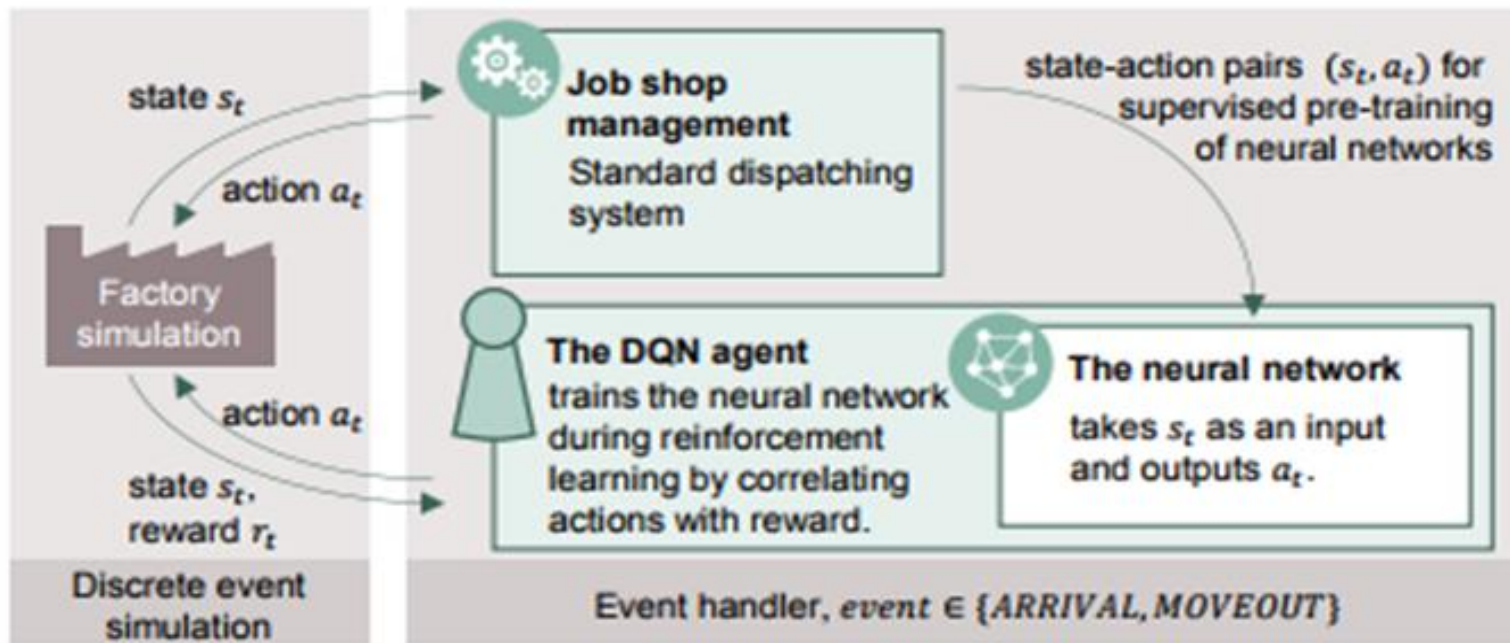
- ML 적용을 위해 복잡하고 동적인 것이 고려된 생산 환경을 선택
- workcenter는 유사장비가 그룹화 되어 있고 각 프로세스는 여러 장비에서 처리될 수 있으며, 대부분은 병렬로 가동되는 동일 설비를 통해 진행됨
- 제약 조건 및 작업 할당 결정

- **Technological Constraints:** Sequence-dependent setup times, different types of processes (e.g. single jobs vs. batch processing), time coupling, varying process times.
- **Logistic Constraints:** Re-entrant flows of the jobs, prescribed due dates of the jobs, different lot sizes, varying availability of tools (e.g. machine breakdowns).
- **Production Quantity:** In a mass production emergent phenomena become visible as a result of interactions jobs (e.g. *Work In Progress (WIP)* waves).

Different products p in the production portfolio take different routes r_p , which consist of a number of N ordered *Single Process Steps* $r_p = (SPS_{p,1}, \dots, SPS_{p,N})$. Each *SPS* has to be handled on a specific resource in a resource pool of M resources for a certain duration. The dedication matrix d determines the possible allocation of jobs to machines:

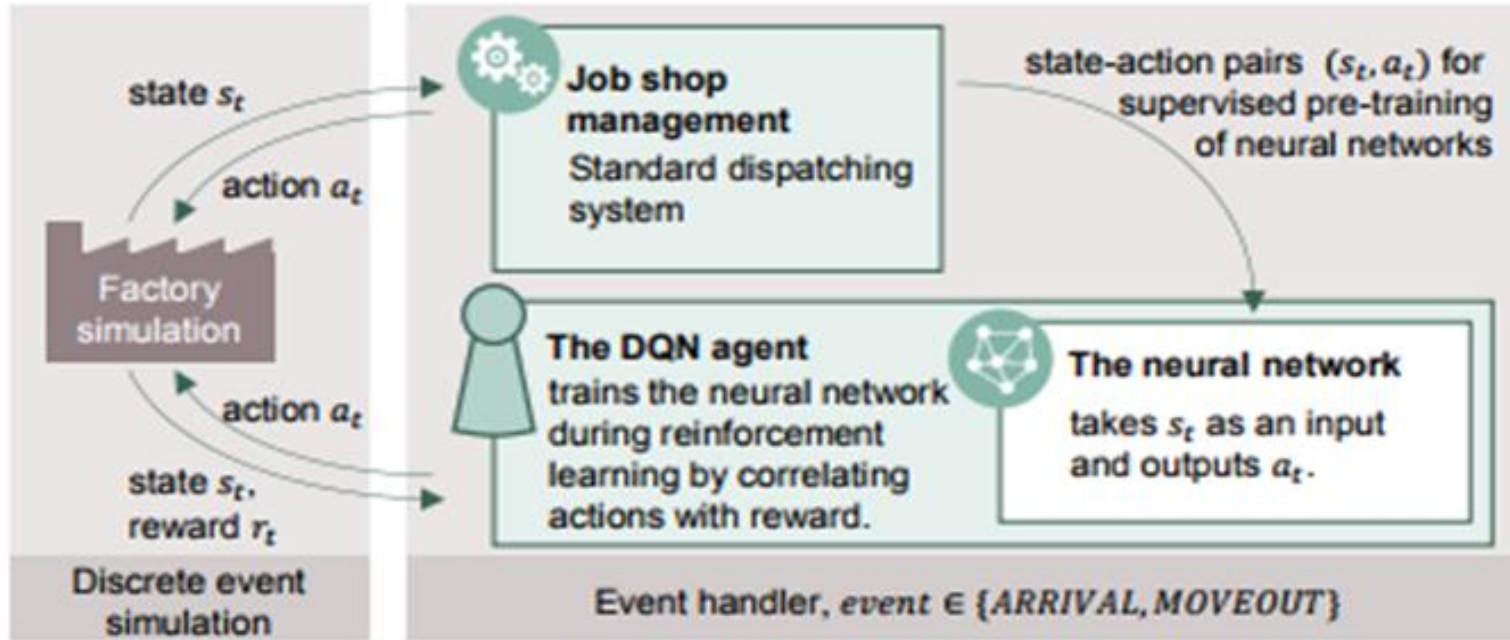
$$d_{(p,n),m} = \begin{cases} 1 & \text{if machine } m \text{ can process } SPS_{p,n} \\ 0 & \text{if machine } m \text{ can not process } SPS_{p,n}. \end{cases} \quad (1)$$

- 스케줄링 : 대기 Lot을 사용 가능한 자원에 할당하는 정적 계획 프로세스
- 디스패칭 : 특정 설비에서 다음 작업에 대한 실시간 결정을 말함



The state space $S = S_{\text{machines}} \times S_{\text{jobs}}$ is a combination of machine states $\mathbf{s}_{\text{machine}} = \langle s_1, \dots, s_{M_w} \rangle \in S_{\text{machines}}$ for M_w machines at a workcenter w and the state of surrounding jobs $\mathbf{s}_{\text{jobs}} = \langle s_1, \dots, s_j \rangle \in S_{\text{jobs}}$ for j jobs. The machine space is defined by machine capabilities, availability (breakdowns) and the setup. Machine capabilities are described by the dedication matrix d (see Section 1.1). Availability av is in most cases binary, $av \in \{0, 1\}^{M_w}$. In this example factory, all machines at one workcenter are identical and breakdowns are not explicitly considered. Therefore only the setup needs to be encoded for the workcenter specific agent: The machine state s_x reduces to a one-hot vector $s_x \in \{0, 1\}^{ST}$ for ST Setup Types for each machine.

The second part of the state space are the properties of the jobs s_j . Firstly, it comprises the product type, which is encoded in a one-hot vector $\{0, 1\}^p$. The locations, which correspond to the workcenters, are also encoded as one-hot vector $\{0, 1\}^{\text{locations}}$. Then the processing percentage of the product is included, which is a relative measure for the number of processing steps already completed. Last, the deviation of a set due date for an operation is given.

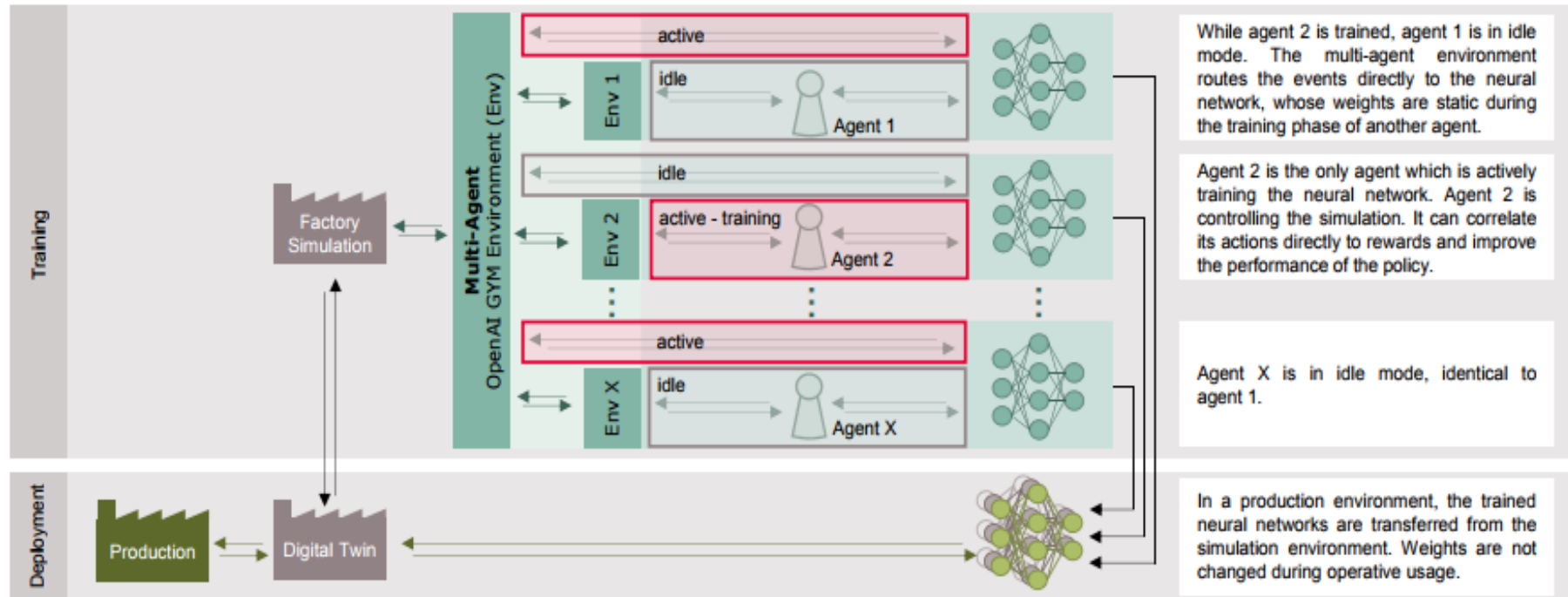


The DQN agents are based on Q -learning, which is used to optimize the action-selection policy $\pi_t(a|s)$ in such a way that it maximizes the reward. The policy $\pi_t(a|s)$ is the probability distribution that $a_t = a$ if $s_t = s$ [22]. The Q -function $Q : S \times A \rightarrow \mathbb{R}$ gives the reward over successive steps weighted by discount factor γ . The optimal action-value function $Q^*(s, a)$ is approximated in the course of the Q -learning algorithm: [3]








$$Q^*(s, a) = \max_{\pi} \mathbb{E} \left[\sum_t r_t \cdot \gamma^t \mid s_t = s, a_t = a, \pi \right]. \quad (2)$$

The training of the DQN agents is separated into two phases:

- **Phase A:** While the one DQN agent is trained, the other workcenters are controlled by heuristics. As DQN agents are model-free, they start without any knowledge about the system (if no prior supervised learning was done). A linear annealed ϵ -greedy policy is used to diversify the samples of the agent. Each DQN agent is trained once.
- **Phase B:** All workcenters are controlled by DQN agents which are learning separately. The ϵ -greedy policy is set to a fixed value and the learning rate can be reduced. The DQN agents are trained in cycles, each time for a relatively short number of steps.



- MathWorks MAT-LAB에서 구현
- OpenAI Gym
- RL Framework : keras-rl
- network : densely connected layer(512, 128, 18)
- activation function : ReLU
- Optimizer / learning rate : Adam / 10^{-4} (Reward phase A), 10^{-5} (Reward phase B)

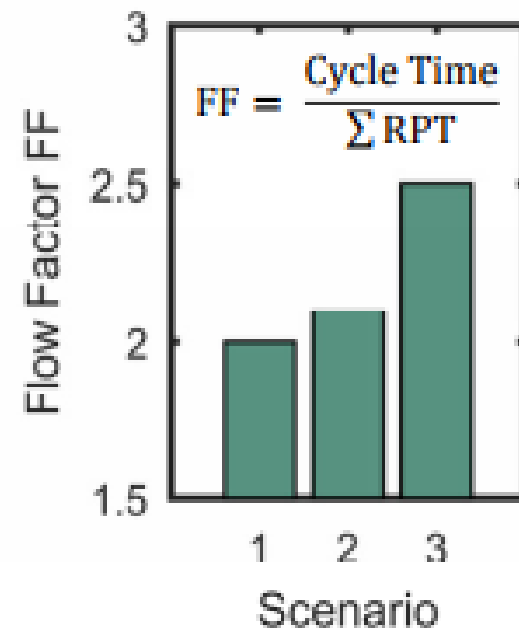
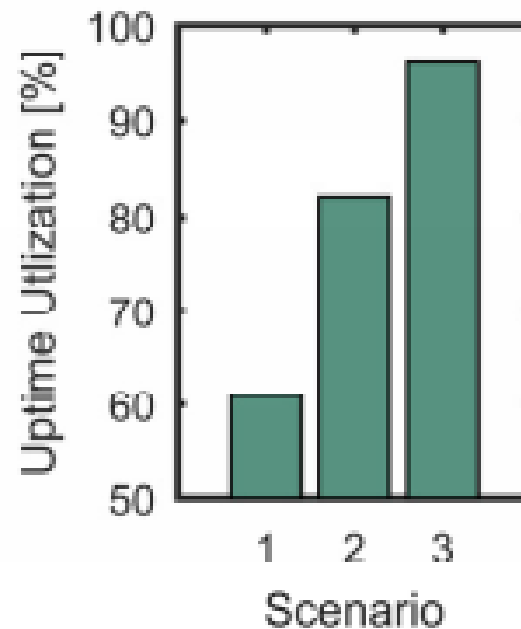
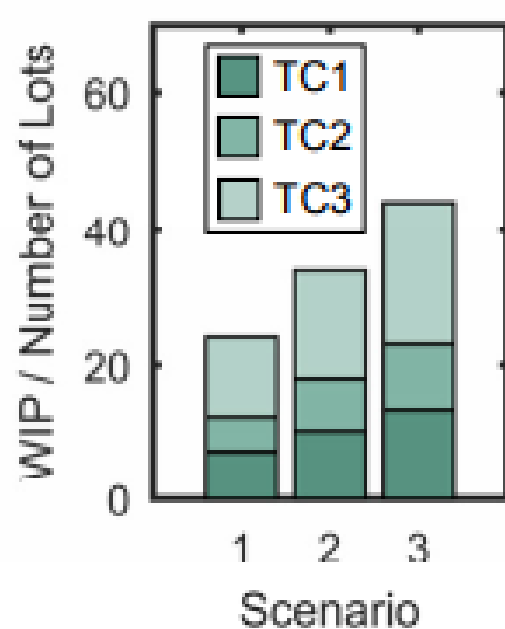
Factory Environment	Lithography	Implant	Buffer/ Other	Furnace
	 			  
Constraints	reticle management		batch transfer to the next step	time coupling
Dispatching	ODD, FIFO	setup optimization, ODD, FIFO		batching, ODD, FIFO
TC 1	18.8	8.0	25.0	48.0
TC 2	19.1	12.0	25.0	64.0
TC 3	16.9	7.7	25.0	52.1

Raw Process Times (RPTs) in arbitrary time units
RPTs have Coefficients of Variance (CoVs) of 50%.
Process flows are re-entrant:

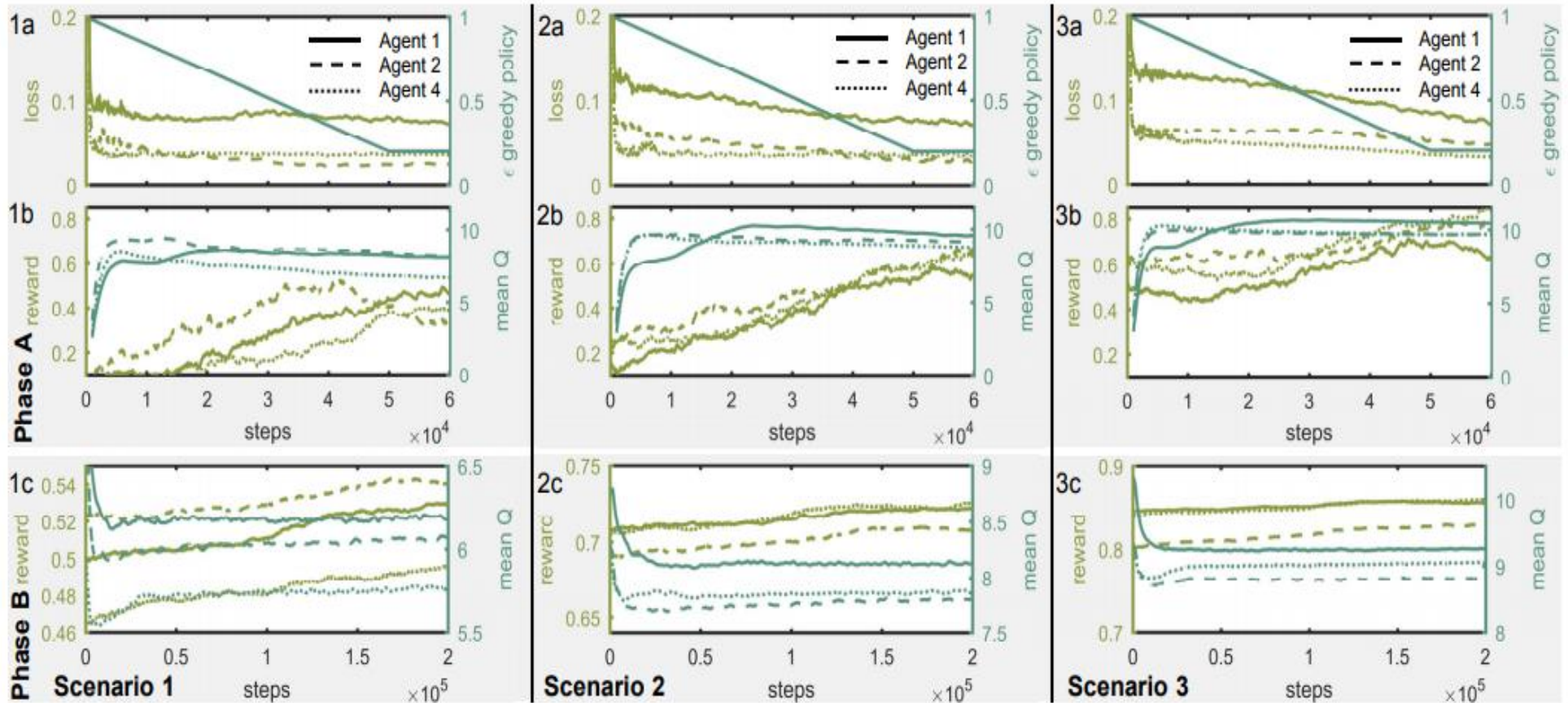
4x 5x 3x

- Lithography : 포토공정 사진인쇄와 비슷하게 빛을 이용하여 복잡한 회로 패턴을 인쇄
- Implant : 이온 주입 공정, 전기적 특성을 갖도록 하는 공정
- Furnace : 열처리 장비

[arbitrary time units]	TC 1	TC 2	TC 3
TC 1	0.0	1.1	1.9
TC 2	4.1	0.0	3.2
TC 3	1.3	3.0	0.0



실험 결과



- **Reward phase A:** The number of lots in process at a workcenter divided by the total capacity (UU at workcenter) plus negative penalties. For actions which are not possible to execute, e.g., when a reticle is already in use, a penalty of -1 is given. If 80% of the total WIP are in queue at one workcenter, the dispatching heuristics are activated to avoid a crash. A penalty of -2 is given in this case.
- **Reward phase B:** The UU of the whole line (all machines in the factory) plus the negative penalties.

배포 모드에서 디스패칭 휴리스틱 및 DQN 에이전트 최적화 비교 (제시된 보상값은 25,000 Steps에 대한 평균 보상)

[Reward in test mode]	Scenario 1	Scenario 2	Scenario 3
Benchmark dispatching	0.61	0.83	0.96
Dispatching with 10% random at workcenter 2	0.58	0.79	0.91
Dispatching with 30% random at workcenter 2	0.55	0.73	0.85
DQN agent optimization	0.62	0.83	0.94

- DQN 에이전트 알고리즘은 Benchmark(휴리스틱)과 비슷한 성능을 보임

- DQN 에이전트 기반의 RL을 생산 스케줄에 성공적으로 적용하는 방법 제시
- 제안 방안은 사람의 개입이나 전문가 지식 없이도 Benchmark와 자동으로 비슷한 스케줄 방법을 개발
- 휴리스틱을 이길 수는 없었지만 Training 2일만에 비슷한 수준에 도달
- Workcenter 2 에서 30% 랜덤 액션도입과 같은 구현에서 비 최적화 또는 오류 감지
- 시스템은 솔루션과 글로벌 최적화 목표간 직접 연결로 인해 높은 투명성 제공
- 몇시간내 훈련과 배포가 가능



감사합니다.