공학박사학위논문

자기지도 기반 심층강화학습을 이용한 납기 제약 하에서의 셋업 스케줄링

Setup Change Scheduling Under Due-date Constraints Using Deep Reinforcement Learning with Self-supervision

2021 년 8 월

서울대학교 대학원

산업공학과

팽 보 형

자기지도 기반 심층강화학습을 이용한 납기 제약 하에서의 셋업 스케줄링

Setup Change Scheduling Under Due-date Constraints Using Deep Reinforcement Learning with Self-supervision

지도교수 박 종 헌

이 논문을 공학박사 학위논문으로 제출함

2021 년 7 월

서울대학교 대학원

산업공학과

팽 보 형

팽보형의 공학박사 학위논문을 인준함

2021 년 7 월

위 원 장	조 성 준
부위원장	박 종 헌
위 원	이 경 식
위 원	김 관 호
위 원	허 재 석

초록

납기 제약 하에서 셋업 스케줄을 수립하는 것은 현실의 여러 제조 산업에서 쉽게 찾아 볼 수 있으며 학계의 많은 관심을 끌고 있는 중대한 문제이다. 그러나 납기와 셋업 제약이 동시에 존재함에 따라 문제의 복잡도가 증가하게 되며, 시시각각 새로운 생산 계획이 주어지고 초기 설비 상태가 변화되는 환경에서 고품질의 스케줄 수립은 더 어려워진다. 본 논문에서는 학습된 심층신경망이 상기한 변화가 발생한 스케줄링 문제도 재학습 없이 해결할 수 있도록, 자기지도 기반 심층강화학습 기법을 제안한다. 구체적으로, 상태와 행동 표현을 생산 계획과 설비 상태에 무관한 차원을 갖도록 설계한다. 동시에 주어진 상태로부터 효율적으로 신경망을 학습하기 위해 파라미터 공유 구조를 도입한다. 이에 더하여, 스케줄링 문제에 적합한 자기지도를 고안하여 설비와 잡의 수, 생산 계획의 분포가 상이한 평가 환경으로도 일반화 가능한 심층신경망을 학습한다. 제안 기법의 유효성을 검증하기 위해 현실의 병렬설비 및 잡샵 공정을 모사한 대규모 데이터셋에서 집약적인 실험을 수행하였다. 제안 기법을 메타휴리스틱 기법과 다른 강화학습 기반 기법, 규칙 기반 기법과 비교함으로써 납기 준수 성능과 연산 시간 관점에서 우수성을 입증하였다. 더불어 상태 표현, 파라미터 공유, 자기지도 각각으로 인한 효과를 조사한 결과, 개별적으로 성능 개선에 기여함을 밝혀냈다.

주요어: 강화학습을 이용한 제조 라인 스케줄링, 자기지도 기반 심층강화학습, 순서 의존적 셋업, 납기 제약 하의 병렬설비 스케줄링, 셋업 제약이 있는 잡샵 스케줄링

학번: 2013-21087

목차

표 목차

그림 목차

제 1 장 서론

1.1 연구 동기 및 배경

제조기업 간의 경쟁이 심화됨에 따라, 생산 스케줄링(production scheduling)은 현대의 대규모 제조 시스템에서 중요한 의사결정 문제로 대두되었다. 특히 기업 경쟁력과 함께 고객의 만족도를 높이기 위해서는, 현장의 셋업 제약을 고려하면서도 주어진 생산 주문량을 달성할 수 있는 스케줄을 수립해야 한다 [2]. 순서 의존적 셋업은 반도체 제조 라인의 전 공정 [3, 4]과 후 공정 [5], 제약 산업 [6], 식품 산업 [7]에서 흔히 발견되는 제약으로 앞선 제품과 다른 종류의 제품을 생산하기 위해서는 셋업 시간이 작업 시간만큼 소요된다 [8]. 일반적으로 셋업은 설비의 가동률을 떨어뜨리고 숙련된 작업자의 작업과 같은 높은 비용을 야기하지만, 주어진 생산 계획을 달성하기 위해서는 적시에 필요한 만큼의 셋업이 이루어져야 한다 [9].

더군다나 고객의 수요는 빈번하고 예측 불가능하게 변화하므로 [10], 제품의 생산 요구량과 납기에 연관된 가변성(variability)이 발생한다. 게다가 스케줄링 시점마다 설비의 남은 작업 시간과 셋업 상태가 달라지므로 초기 설비 상태에도 가변성이 존재할 수 있다. 점차 제조업계는 주문생산 방식, 다품종 소량생산 방식으로 이행하고 있어 위와 같은 가변성을 고려한 스케줄링이 더욱 중요해지고 있다. 따라서 본 논문에서 다루는 스케줄링 문제는 셋업 및 납기 제약이 동시에 존재함에 따라 난이도가 증가하였으며, 실제 현장의 가변성에 대응하여 스케줄 성능 하락을 최소화하는 기법을 요구한다.

본 논문에서는 연구 대상으로 두 개의 스케줄링 문제를 선정하여, 여러 생산 라인에 적용을 도모하였다. 첫째는 오랫동안 많은 연구자들의 관심을 끌어온 이종 병렬설비

스케줄링 문제(UPMSP, unrelated parallel machine scheduling problem)로, 각각의 잡을 작업 시간이 서로 다른 설비 중 하나에 배치하는 것이 목적이다 [11]. 이 때, 지연 시간 최소화 목적함수의 UPMSP는 NP-난해(NP-hard)한 문제로, 다항시간 안에 풀기 어려운 것으로 알려져 있다 [12]. 둘째는 잡의 재진입(re-entrance)이 존재하는 잡샵(job shop) 스케줄링 문제이다 [13]. 제안 기법을 두 가지 스케줄링 문제에서 풀이함으로써 기초적인 수준의 범용성을 달성하고자 하였다.

납기 제약 하에서 셋업 스케줄링을 수행하기 위하여 메타휴리스틱(metaheuristics) 접근법이 성공적으로 활용되어 왔다 [11, 14, 15]. 하지만 메타휴리스틱 접근법은 특정한 시간제한 안에 대규모 스케줄링 문제에서 고품질의 스케줄을 찾는 것을 보장하지 못한다. 대안으로, 연산 시간이 짧고 구현이 용이하다는 장점이 있는 규칙 기반 기법이 활발하게 채택되고 있다 [16]. 그러나 규칙 기반 기법은 근시안적으로 스케줄을 수립하는 측면이 있어, 스케줄 품질이 만족스럽지 않을 수 있다 [17].

이러한 규칙 기반 기법의 한계점을 극복하기 위해서, 수십 년 전부터 강화학습 접근법이 탐구되어 왔다 [18, 19]. 강화학습의 목표는 앞으로의 기대 보상 총합을 최대화하는 적응적인 정책(policy)을 학습하는 것으로, 강화학습 기반 스케줄링 기법은 시시각각 변화하는 제조 라인의 현 상황에 알맞은 유동적인 의사결정을 수립할 수 있다. 최근에는 심층강화학습(DRL, Deep Reinforcement Learning)의 놀라운 성공에 힘입어, 강화학습으로 모델링하기 적합한 여러 순차적 의사결정 문제에 도입되고 있다. DRL은 심층신경망(DNN, Deep Neural Network)을 정책 학습에 활용함으로써 현재 상황과 미래 보상값 간의 복잡한 연관관계를 학습할 수 있다. 제조 라인 스케줄링에서도 시뮬레이터를 외부 환경으로 구축하여 스마트 제조 시스템에 쉽게 융합할 수 있으며 [20], 목적함수에 대한 전역 최적화를 추구한다는 장점으로 활발하게 연구가 진행되고 있다.

그러나 대부분이 최대완료시간(makespan) 최소화를 목적함수로 다루었으며, 납기

관련 목적함수에 DRL을 적용하는 연구에 대한 관심은 상대적으로 적다. 특히 순서 의존적 셋업과 함께 생산 요구량과 납기의 가변성을 고려한 연구는 아직 이루어지고 있지 않다. 기존의 DRL 기법을 그대로 적용하기에는 크게 두가지 어려움이 존재한다. 첫 번째는 상태(state) 표현에 납기와 셋업을 모두 반영함에 따라 상태 공간의 크기가 커질 수 있고, 함수 근사와 DNN의 일반화가 어려워진다 [21]. 두 번째는 잡과 설비의 수가 늘어남에 따라 학습 복잡도가 빠르게 증가하기 때문에, 대규모 제조 시스템에서는 앞서 언급한 가변성이 발생할 때마다 재학습(re-training)을 수행하기 어렵다.

또 다른 관점으로, 상태와 행동(action) 표현에서의 추상화로 정보가 소실된다는 한계점을 들 수 있다. DNN을 도입하는 장점은 고차원의 넓은 상태 벡터로부터 복잡한 비선형적 연관관계를 학습한다는 것인데, 저차원으로 압축된 상태를 활용하게 되면 정확한 연관관계를 파악하기 어렵다. 반대로, 모든 잡과 설비에 대한 세세한 정보를 상태 및 행동에 표현하는 경우에는 가변성에 대응하는 것이 어려워진다. 만약 잡과 설비의 수가 변할 때마다 상태 및 행동 벡터의 차원이 달라진다면, 반드시 재학습이 요구된다. 또한 네트워크의 크기가 지나치게 커지는 경우, 다양한 상태 공간을 탐색하기가 어려워져 학습 효율이 떨어지는 문제가 있다.

1.2 연구 목적 및 공헌

본 논문에서는 납기를 준수하며 셋업 스케줄을 수립하는 자기지도 기반 DRL 기법을 제안하고자 한다. 무엇보다도 생산 요구량과 납기, 초기 설비 상태의 변화에도 강건한 DNN 파라미터를 학습하는 것이 연구의 주 목적이다. 목표를 달성하려면 먼저 납기 제약과 순서 의존적 셋업을 고려한 DRL 모델링을 통해 제약에 의한 성능 저하를 최소화해야 한다. 이에 더하여 생산 계획과 초기 설비 상태에 가변성이 존재하더라도 상태 및 행동의 차원이 유지되고, 현실적인 시간 안에 평가 스케줄링 문제를 풀 수 있는 기법을 고안해야 한다.

첫 번째 연구 목적은 병렬설비 스케줄링 문제를 대상으로 DRL을 이용한 셋업 스케줄링 기법을 제시하는 것이다. 보다 구체적으로, DNN에 적합한 고차원 벡터로 상태를 표현하면서도 납기와 셋업 제약을 반영하는 행동 및 상태 표현을 고안한다. 또한 파라미터 공유 네트워크 구조를 도입하여 빠른 학습 시간 안에 더 좋은 스케줄을 수립하고자 한다. 그리고 생산 요구량과 납기에 가변성이 있는 데이터셋에서 제안 기법의 유효성을 검증한다.

두 번째 연구 목적은 제안 기법이 다른 스케줄링 문제에도 도입될 수 있도록, 잡샵 스케줄링 문제에 적용하는 것이다. 새로운 공정이 추가되어도 대응할 수 있는 스케줄링 프레임워크를 소개하고, 일부 변경된 DRL 모델링 방식을 제시한다. 결과적으로 초기 설비 상태와 생산 계획이 함께 변화하는 데이터셋에서 제안 기법의 성능을 평가한다.

세 번째 연구 목적은 자기지도를 동반한 DRL을 통한 일반화 성능 개선이다. 대상 스케줄링 문제에 적합한 자기지도와 학습 기법을 제안하여, DNN이 정책 학습에 더불어 제조 시스템의 일반적인 특징을 파악할 수 있도록 한다. 최종적으로, 학습한 데이터셋의 스케줄링 문제와 편차가 큰 새로운 평가 데이터셋의 환경에서도 우수한 성능의 스케줄을 수립할 수 있음을 입증한다.

연구의 공헌을 요약하면 다음과 같다.

(a) 납기 제약 하에서의 셋업 스케줄링 문제를 풀기 위한 DRL 기법을 제안하여 생산 요구량과 납기, 초기 설비 상태가 학습 때와 다른 평가 스케줄링 문제에서도 재학습 없이 스케줄을 수립할 수 있다.

(b) 학습 효율을 높이고 일반화 성능을 확보하기 위한 파라미터 공유 구조와 자기지도 손실을 도입하므로써, 현실적인 시간 안에 학습한 DNN으로 우수한 품질의 스케줄을 수립할 수 있다.

(c) 제안 기법의 효과를 입증하기 위해 병렬설비 스케줄링과 잡샵 스케줄링 문제를 포함한 대규모 데이터셋에서 실험을 진행하였고, 기존의 메타휴리스틱 및 강화학습 기반 기법 대비 경쟁력 있는 결과를 확보하였다.

1.3 논문구성

본 논문은 7장으로 구성된다. 제 2장에서는 선행 연구와 함께 기법적 배경을 살펴본다. 제 3장은 대상 스케줄링 문제를 정의한다. 제 4장에서는 자기지도 기반 심층강화학습을 활용하여 병렬설비 스케줄링 문제를 해결하는 기법을 제안한다. 제 5장은 심층강화학습을 잡샵 스케줄링에 적용하는 기법을 제시한다. 뒤이어 제 6장에서는 제안 기법의 유효성을 검증하기 위해 대규모의 병렬설비 및 잡샵 스케줄링 문제에서 수행한 실험과 결과를 살펴본다. 마지막으로 제 7장에서는 결론과 향후 연구방향을 제시한다.

제 2 장 배경

2.1 순서 의존적 셋업이 있는 납기 제약 하에서의 스케줄링 문제

2.1.1 납기 제약 하에서의 스케줄링 문제

운용연구(operation research)에서 납기 제약 하에서의 스케줄링 문제는 반 세기 동안 광범위하게 연구되어 온 주요한 스케줄링 문제이다 [22]. 스케줄 이론에서는 earliness-tardiness 스케줄링 문제로도 칭하며, 잡이 주어진 납기보다 일찍 또는 늦게 완료될 때 패널티가 부과된다. 납기 관련 목적함수는 납기를 준수하지 못한 패널티를 최소화하는 것을 목표하고, 지연시간 총합(total tardiness), 지연 개수(number of tardy jobs), 산출량(throughput), 최대 지연시간(maximum lateness), 납기 편차(due-date deviation), 납기 발생 비용 등이 속한다.

기존 연구에서 다루어진 스케줄링 문제들은 제조 라인의 특성에 따라 몇몇 추가적인 특징을 가지는데, 이를 Graham 등이 제안한 표기법 [23]으로 요약할 수 있다. 구체적으로 $\alpha|\beta|\gamma$의 세 가지 특징으로 문제를 표현하며, α는 설비 특성, β는 문제의 제약, γ는 목적함수를 뜻한다. 예를 들어, 단순하지만 대표적인 NP-난해 문제로 알려진 단일설비에서의 가중 지연시간 총합 최소화문제는 $1||\sum w_j T_j$와 같이 표시된다. β에 해당하는 여러 제약 중에서 본 논문에서는 앞선 설비의 상태에 따라 작업 이전에 발생하는 비용으로 정의되는 순서 의존적 셋업에 집중하며, 흔히 s_{ij}로 표현한다 [8].

2.1.2 패밀리 셋업을 고려한 병렬설비 스케줄링

납기 관련 목적함수를 지닌 이종 병렬설비 스케줄링 문제 중에서도, 여러 연구들이 순서 의존적 패밀리 셋업을 주요한 제약으로 다루어 왔다 [24, 25, 26, 27]. 두 제약이 공존할 때 부각되는 문제의 복잡도 때문에, 동종 병렬설비 스케줄링 문제에 대해서도 꾸준히 연구가 진행되었다 [28, 29]. 최근에도, 현실적인 가정 하에 리소스 제약이 있는 제조 라인을 대상으로 한 연구를 찾아볼 수 있다 [30]

수립한 스케줄의 셋업 횟수를 감소하기 위해, 셋업 없이 하나의 설비에서 연속적으로 작업되는 잡들의 배치를 형성하는 배치 스케줄링 휴리스틱이 흔히 도입되었다 [31]. 가중 지연시간을 최소화하는 문제에서, 2단계 배치 휴리스틱이 납기가 동일하다는 가정과 [9], 모든 잡들이 동일하다는 가정 [32] 하에 제안되었다. 또한 작업 시간과 슬랙 시간, 그리고 패밀리 셋업 시간을 혼합한 BATCS (batch apparent tardiness cost with setup) 휴리스틱이 제시된 바 있다 [33]. 여기서 슬랙 시간이란 작업 시간과 납기까지의 남은 시간의 차를 의미한다. 지연시간 총합 목적함수에서는 배치 기반 repair 방법론을 통합함으로써 개선된 simulated annealing 휴리스틱이 개발되었다 [11].

한 편 지연시간 총합을 최소화하기 위한 패밀리 셋업이 있는 병렬설비 스케줄링 문제에서 메타휴리스틱 기법들이 주도적으로 활용되어 왔다. 메타휴리스틱 기법은 배치 형성 기술을 사용하여 셋업 횟수가 적은 스케줄만으로 해 공간을 제한한다는 특징을 갖는다. Iterated greedy (IG) 알고리듬 [14]은 반복적인 이웃 선택과 철저한 근접 탐색으로 초기 해를 개선시키는 방법론이며, 성공적으로 고객 우선순위 제약이 있는 스케줄링 문제를 풀어냈다. 보다 최근에는 batch swap 및 job insertion과 같은 여섯 가지 근접 탐색 연산을 고안하여 더욱 개선된 IG가 개발되었다 [15]. 해당 연구에서는 설비 수 6대에 잡 수 50개인 문제에서 경쟁력있는 결과를 보여주었다. 잡 시퀀스를 교환하는 교차 연산 등을 적용한 artifical bee colony (ABC) 알고리듬이 동일한 문제에서 제안되기도

하였다 [34].

타부 탐색(TS, tabu search) 알고리듬은 완료시간과 지연시간의 합계 [24, 25] 및 가중 지연시간 [26] 등의 여타 납기 관련 목적함수에서 우세한 기법 중 하나이다. 또한, 반복된 하이브리드 메타 휴리스틱(IHM)이 동일한 목적함수에서 활용된 바 있다 [27]. 마지막으로 언급할 만한 사항으로, 패밀리 셋업을 다룬 연구를 찾아보긴 어려우나 스케줄링 문제에 기계학습(machine learning)을 활용하는 기법이 꾸준히 연구되어 왔다 [35]. 병렬설비로 구성된 플로우샵에서 지지 벡터 머신(support vector machine) [36], 가우시안 프로세스(gaussian process) [37]가 활용되었고, 셋업이 있는 병렬설비 문제에서 인공신경망이 적용된 바 있다 [38].

요약하여, 표 2.1은 패밀리 셋업 제약을 다룬 기존 연구 목록을 Graham의 표기법을 응용하여 제시한다. R_m, P_m, S_m은 각각 이종 병렬설비, 동종 병렬설비, 단일설비를 의미한다. 제약으로 패밀리 셋업 s_{fg}을 반영한 연구와 달리 s_{ij}로 표기된 연구는 패밀리 셋업을 고려하지 않았다. 가령 타부 탐색을 도입하여 순서 의존적 셋업이 있는 스케줄링 문제를 푼 연구 [39]는 패밀리 셋업을 명시적으로 고려하지 않았기 때문에 본 연구에 적용되기 어렵다. 목적함수를 의미하는 γ에서 T_j, w_jT_j, c_j는 각각 지연시간 총합, 가중 지연시간, 완료시간을 의미한다.

2.1.3 셋업 제약이 있는 잡샵 스케줄링

하나의 잡이 복수 개의 공정을 거쳐야 작업 완료되는 스케줄링 문제는, 공정 시퀀스 순서대로 병렬설비에 잡을 할당하는 잡샵 스케줄링 문제로 풀 수 있다. 그 중에서도 셋업 제약이 있는 잡샵 스케줄링 문제는 많은 연구자들의 관심을 끌어온 전통적인 문제이다 [46]. 예를 들어, 반도체 제조 라인은 웨이퍼 준비 라인 [47]부터 웨이퍼를 식각하는 전 공정 팹(FAB)과 후 공정까지 복잡한 여러 개의 공정으로 이루어져 있으며 잡 패밀리 간의 셋업이 발생한다 [5].

Table 2.1: 패밀리 셋업 제약을 다룬 기존 연구 목록

대상 문제			수리계획, 휴리스틱	근접 탐색				진화 알고리듬	
α	β	γ		TS	IG	VNS	IHM	GA	ABC
R_m	s_f	T_j	[11]		[14, 15]				[34]
R_m	s_{fg}	w_jT_j	[32]	[26]			[27]		
R_m	s_{fg}	$w_j(T_j + c_j)$		[24, 25]					
R_m	s_{ij}	w_jT_j		[40]		[41]	[39]	[42]	[43]
P_m	s_{fg}	w_jT_j	[28, 29, 44, 30]	[45]		[6]			
S_m	s_{fg}	T_j	[16]		[15]				

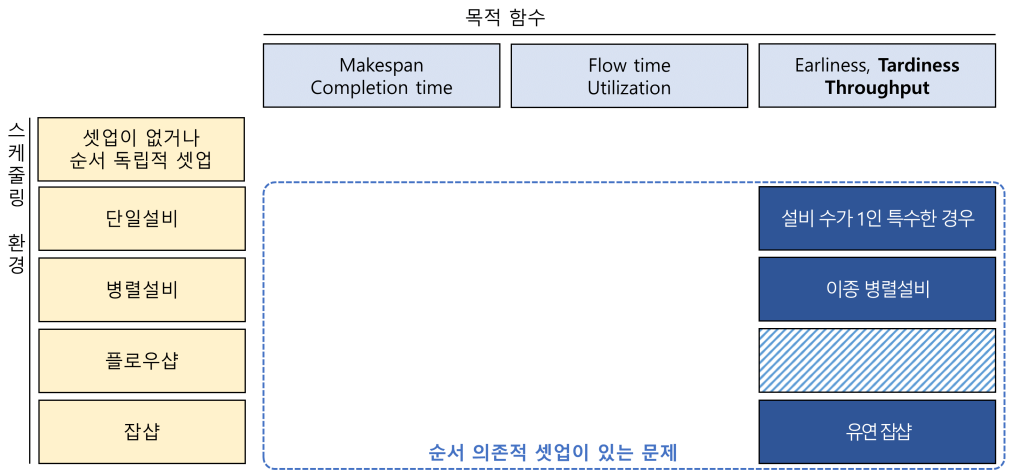


Figure 2.1: 연구 대상 스케줄링 문제의 범위

여러 연구들이 잡샵 스케줄링 문제를 풀기 위해 휴리스틱 기법을 제안하였다. 납기 관련 목적함수를 위해 잡의 생산 요구량을 결정하는 디스패치 규칙이 연구되었고 [48], 산출량을 최대화하기 위해 2단계 휴리스틱이 연구되었다 [49]. 셋업 제약을 고려하면서도 설비 가동률과 산출량을 최대화하기 위해 디스패치 규칙을 제안한 연구도 존재한다 [50]. 배치 단위로 셋업이 이루어지는 단일설비, 병렬설비, 잡샵에 대해 다양한 휴리스틱 규칙을 적용한 논문으로 [51]를 참조할 수 있다.

한편, 메타휴리스틱 접근법을 사용한 연구에서는 병렬적 GA와 추천 모듈을 통해 동시 셋업 작업자 수와 초기 셋업 상태 제약을 고려하였다 [52]. 또 다른 대안 접근방법으로, 교사학습(supervised learning) [13], 사례 기반 추론(case based reasoning) [17]과 같은 기계학습 기법이 반도체 패키징 라인 스케줄링 문제를 대상으로 연구되기도 하였다.

끝으로 본 논문에서 다루는 순서 의존적 셋업이 있는 납기 제약 하의 스케줄링 문제의 영역은 그림 2.1과 같이 요약된다.

2.2 강화학습 기반 스케줄링

2.2.1 이론적 배경

마르코프 프로세스(Markov process)는 시스템 역학(system dynamics)의 갑작스러운 변화가 있는 상황에서 제어 문제(control problem)를 해결하기 위해 널리 도입되어왔다 [53]. 일례로 퍼지 시스템(fuzzy system)인 마르코프 점프 비선형 시스템을 모델링할 수 있다는 이점으로, 마르코프 랜덤 프로세스가 활용될 수 있다 [54]. 반면 Markov decision process (MDP)는 시스템 역학이 알려져 있지 않은 의사결정 문제를 풀이하는 데에 유리하다 [1].

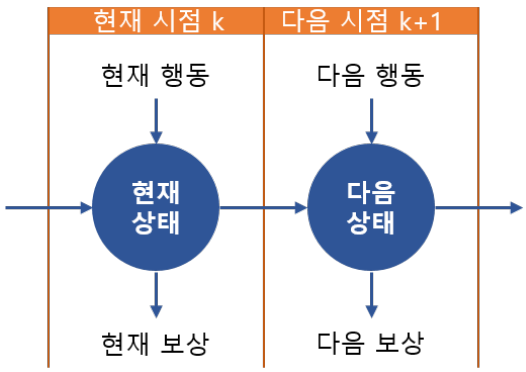


Figure 2.2: MDP에서의 상태 전이 도식[1]

MDP는 현재 시점(time-step) k에서 상태(state) s, 행동(action)인 a, 보상(reward) r과 다음 상태 s', 다음 상태로의 전이 확률 함수 p로 구성된다. MDP에서는 a가 실행됨에 따라 s에서 s'으로의 상태 전이가 이루어지며, 그 결과 보상 r이 관찰된다. 이때 s로부터 s'까지의 시간 간격을 기간(period)이라 정의한다. p를 수식으로 나타내면 다음과 같다.

$$p(s', r|s, a) = Pr\{s_{k+1} = s', r_k = r|s_k = s, a_k = a\} \tag{2.1}$$

마르코프 성질을 만족한다는 것은 수식 2.1에 드러나 있듯 r과 s'가 현재의 상태 및 행동인 s_k, a_k에만 의존적이고, 과거 상태에는 영향을 받지 않음을 뜻한다. 전이 확률 함수 p는 MDP 모델 또는 시스템 역학으로 불리며, 모델을 명시적인 수식으로 나타내는 접근법은 현실 제약을 반영하기 어렵고 난이도가 높다 [55].

명시적으로 모델을 알지 못해도 MDP 정책을 학습할 수 있다는 장점으로, MDP로 모델링되는 여러 의사결정 문제에 강화학습을 활용할 수 있다. 강화학습은 에이전트(agent)가 외부에 놓인 모든 것을 아우르는 환경(environment) 간의 상호작용을 통해 학습하는 것을 목표로 한다. 그 중에서도 큐 러닝(Q-learning)은 대표적인 model-free 강화학습 접근법으로 널리 채택되고 있다 [56].

2.2.2 강화학습을 이용한 제조 라인 스케줄링

스케줄링 문제를 해결하기 위해 제조 시스템을 MDP로 공식화(formulation)한 후, 강화학습이 정책을 학습하는 데에 도입되었다. 제조 시스템에 상응하는 외부 환경으로부터 관찰된 상태가 주어지면, 에이전트는 각 행동의 추정되는 가치인 Q 값을 예측함으로써 스케줄링 의사결정을 수립한다. 제조 라인 스케줄링에서 행동은 휴리스틱 기법 또는 후보 잡의 목록으로 정의된 스케줄링 의사결정을 뜻한다.

많은 기존 연구들이 우선순위 지수(priority index) 기반 휴리스틱 기법을 행동으로 정의하는 방식을 채택하였다 [19]. 이러한 정의 방식을 응용하여 earliest due date (EDD), shortest processing time (SPT)와 같은 여러 디스패치 규칙들을 혼합한 행동 정의가 잡샵 스케줄링 문제에 활용되기도 하였다 [57]. 학습된 강화학습 기반 규칙을 특정 기간 동안 유지되도록 함으로써, 역동적인 환경에도 성공적으로 대응할 수 있었다. 보다 최근에는, 납기와 생산 요구량의 가변성이 고려되기도 하였다 [10]. 또 다른 부류의 행동은 현 상황에 적합한 후보 잡의 목록으로 정의된다 [2]. 또는 상위 K개의 EDD 잡이거나 [58] 잡의 속성 [19], 잡의 제품 타입 [59, 60]이 될 수도 있다. 일부 연구들에서는

행동이 다른 기법에 의해 만들어진 스케줄을 다시 변경하는 정책을 의미하기도 한다. 예를 들어, 에이전트는 simulated annealing 기법 [18], VNS [61], cuckoo search [20]에 활용될 하이퍼파라미터를 선택하는 방법을 배운다.

병렬설비 스케줄링 문제를 대상으로 한 일련의 연구들에서 [62, 63], 비중 지연시간 최소화 목적의 UPMSP를 풀기 위하여 큐 러닝 기법이 도입되었다. 저자들은 선형 기저 함수(linear basis function)를 활용해 Q 값을 근사하였고, 상태 피쳐(features)에 모든 잡과 설비들의 상태(status)를 표현하였다. 여섯 가지 휴리스틱을 행동으로 설계하고, 강화학습 기법이 유동적으로 규칙을 선택함으로써 개별 휴리스틱을 능가할 수 있었다. 비록 앞선 연구[62]는 순서 의존적인 셋업을 다루었으나, 에이전트는 학습 시와 동일한 스케줄링 문제에서만 평가되었다. 반면, 후속 연구 [63]에서는 셋업을 고려하지는 않았으나 학습 문제와 다른 여러 문제에서 평가가 이루어졌다.

Yuan 등 [64, 65]은 각각 지연시간 총합과 지연 잡 수 최소화 목적함수로 준비 시간 제약 및 설비 고장(breakdown)을 고려하는 데에 초점을 맞추었다. Q 값 근사에는 탐색한 상태-행동 쌍의 Q 값을 표에 저장하는 기법이 활용되었다. 이러한 기법에서는 제한된 크기의 표에 상태를 표현하기 위해 연속적인 값을 갖는 속성을 몇 개 그룹으로 분리할 필요성이 있다. 예를 들어, 잡의 평균 lateness 값이 0을 기준으로 크거나 작은 두 종류로 구분되어 상태 표현되었다 [65]. 대안 기법으로, self-organizing map [57], k-means nearest neighbor [61, 66]와 같은 비교사 학습(unsupervised learning) 기법이 스케줄링 문제에 활용된 바 있다.

잡샵 스케줄링 문제 또한 그 복잡도와 산업에서의 유용성 덕분에 오래 전부터 강화학습을 도입한 연구가 시도되었다 [67]. Bouazza 등 [68]은 납기 제약 하의 유연잡샵(flexible job-shop) 스케줄링 문제에서 테이블 기반 큐 러닝 기법을 활용하였다. 셋업이 있는 잡샵 스케줄링 문제인 반도체 테스트 공정에서도 추가적인 리소스 제약까지

Table 2.2: 납기 준수를 목적으로 한 강화학습 연구

주요 제약	단일설비	병렬설비	플로우샵	잡샵
순서 의존적 셋업	Rummukainen 등 [59]	Zhang 등 [62]	Hong 등 [67]	Zhang 등 [69]
준비 시간	Wang 등 [72]	Zhang 등 [63], Yuan 등 [64]		Wang 등 [66]
설비 관련 제약		Yuan 등 [65], Terekhov 등 [71]	Shiue 등 [57]	Bouazza 등 [68]

고려하여 강화학습을 적용한 연구가 이루어졌다 [69]. 반도체 팹에서 설비 가동률을 최대화하기 위해 디스패치 규칙을 상황에 따라 적용시키는 강화학습 연구도 찾아볼 수 있다 [70]. 주목할 만한 연구로, 강화학습 기반 스케줄링 연구에 대기열 이론(queuing theory)을 접목하여 스케줄링 문제와 대기열 문제를 함께 다룬 바 있다 [71]. 표 2.2는 언급된 강화학습을 이용한 선행 연구들 중에서 납기 준수를 목적으로 한 연구를, 주요 제약과 설비 환경을 기준으로 정리하여 나타낸다.

2.2.3 스케줄링 문제에서의 심층강화학습

전통적인 강화학습 기법과 달리, DRL은 DNN을 도입하여 고차원 상태 벡터로부터 Q 값을 근사화하는 것을 목표로 한다 [73]. 또한 학습 안정성을 높이기 위해 타겟 네트워크를, 상태-행동 쌍간의 상관관계를 줄이기 위해 경험 재현을 제안하였다는 차이점을 갖는다. 이를 통해 에이전트가 크고 연속적인 상태 공간에서 비선형 정책을 학습할 수 있다는 이점을 갖는다. DRL이 에너지 공급 조절 [74, 75], 경로 설정 [76], 위험 영역 식별 [77], 전력 시장가 결정 [78] 등 다양한 의사결정 문제에 광범위하게 연구됨에 따라 스케줄링 분야에서도 점차 두각을 나타내고 있다.

최근까지, 여러 연구가 DRL의 적용 범위를 컴퓨터 리소스 관리 [79, 80]와 분산형 시스템 [81, 82] 분야로 확장해 나가고 있다. 언급한 연구들은 모두 상태를 리소스와 다가오는 time-step을 두 축으로하는 이차원 행렬로 인코딩하였다. 상태를 표현한 행렬은 일차원 벡터로 평탄화(flatten) 함으로써, 완전연결(fully-connected) 레이어로 구성된 DNN 입력으로 활용되었다. 대부분 DNN의 입력으로 평탄한 벡터를 사용하였지만, 일부 상태 벡터는 이미지로 취급되어 컨볼루션 신경망(CNN, convolutional neural network) [83]에 삽입되었다.

이에 더하여, DRL 기법은 제조 시스템의 스케줄링 문제를 푸는 데에도 적용되었다. 혼합 플로우샵 스케줄링 문제에서, 에이전트는 각 설비들이 유휴 상태인지, 작업 중인지, 종료되었는지를 나타낸 상태에 근거하여 잡을 할당한다 [84]. 연속적인 흐름을 조절해야 하는 화학 공정에서도 복합적인 목적함수를 최적화하기 위하여 DRL이 적용되었다 [85]. 여러 스케줄링 문제 중에서도, 최대완료시간을 최소화하기 위한 잡샵 스케줄링 문제가 활발하게 연구되어왔다. Lin 등 [86]은 설비 상태와 작업 시간의 통계값들을 상태에 표현하여 각 설비 별로 디스패치 규칙을 추론하였다. 또 다른 연구는 상태에 모든 설비의 셋업 상태와 셋업 기록을 아울러서 셋업 시간을 고려하였다 [60]. 최근까지도 여러 가지 디스패칙 규칙을 행동으로 정의한 연구가 지속되고 있다 [87]. [88, 89]은 CNN을 도입하여 잡, 공정, 그리고 설비 간의 관계를 나타내는 이차원 행렬로 표현된 상태를 다루기도 하였다.

반면 납기 관련 목적함수를 가진 스케줄링 문제는 상대적으로 덜 연구되었다. 산출량을 최대화하는 문제에 적용된 DRL 연구에서는 모든 대기 중인 잡의 납기 정보를 상태 표현에 활용하였다 [90]. 단일설비 스케줄링 문제에서 지연시간 총합을 최소화하는 문제에서는 슬랙 시간이 활용된 바 있다 [91]. 납기 편차를 최소화하기 위한 반도체 생산 스케줄링 문제에서 셋업 제약이 수용되기도 하였다 [58]. 해당 연구에서는 모든

설비의 셋업 상태와 잡의 납기를 상태 표현에 활용했기 때문에, 학습된 네트워크는 잡의 수가 바뀐 새로운 문제를 풀기위해서는 재학습이 필요하다. Luo 등 [21]은 유연 잡샵 스케줄링 문제에서 새로운 잡이 삽입되는 상황에 초점을 맞추었고, 지연시간 총합 목적함수로 준비 시간 제약도 고려하였다. 일곱 종류의 피쳐를 추출하여 벡터로 구성한 상태를 활용하였고, 행동은 여러 휴리스틱 중 하나를 선택하는 방식으로 정의되었다. 그러나 셋업을 고려하지 않았다는 한계점을 갖는다.

끝으로 표 2.3은 본 연구와 관련이 깊은 DRL 기반 스케줄링 연구의 목록과 그 특징을 요약한다. 연구 대상 스케줄링 환경에 더하여 상태 표현 방식의 차이와 잡 및 설비 수 변화에의 의존성을 정리하였다.

Table 2.3: 심층강화학습 기반 스케줄링 연구 목록 및 특징

기존 연구	대상 문제	목적함수	셋업 유무	2차원 행렬 활용	납기 반영 방식	잡 수에 의존적	설비 수에 의존적
Washneck 등 (2018) [58]	유연잡샵	납기 편차	o	x	표준 편차 등	o	o
Thomas 등 (2018) [90]	잡샵	산출량	x	x	목표 산출량 대비 현재 산출량의 비율	o	o
Zheng 등 (2019) [91]	단일설비 (컴퓨터)	산출량	x	o	표준 편차 등	x	o
Cao 등 (2019) [82]	병렬설비 (컴퓨터)	대기 시간	x	o	-	x	o
Park 등 (2019) [60]	유연잡샵	최대완료 시간	o	x	-	x	x
Luo 등 (2020) [21]	유연잡샵	지연시간 총합	x	x	설계된 피쳐	x	x
Han 등 (2020) [89]	잡샵	최대완료 시간	x	o	-	o	o

2.3 자기지도 기반 심층강화학습

상태 표현 학습(state representation learning)은 피쳐 학습 기법의 한 종류로 강화학습과 같이 피쳐가 시간에 따라 변화하고 행동 또는 상호작용에 영향을 받는 특수한 경우에 해당한다 [92]. 십 수 년 전부터 로보틱스 분야의 연구는 적은 양의 데이터에서 빠르게 정책을 학습하기 위하여, 고차원의 이미지 입력 데이터를 저차원의 은닉 피쳐로 전환해주는 auto-encoder를 사전 학습하는 전략을 취하였다 [93]. DRL에서는 역동적인 환경에서도 수렴 속도를 앞당기기 위하여, 상태 벡터로부터 보상 값이나 다음 상태를 추론하는 모델을 학습하는 전략으로 지속적으로 발전해왔다. 모델 학습 과제는 표 2.4와 같이 크게 네 종류로 분류할 수 있다 [92, 94].

Table 2.4: 강화학습에서의 모델 학습 과제

명칭	모델 함수	설명
관찰 재구축	$o_t = f^{-1} \circ f(s_t)$	상태 s_t와 인코딩 함수 f, 디코딩 함수 f^{-1}을 거쳐 재구축한 관찰 o_t 간의 오차 최소화
정방향 모델	$s_{t+1} = f(s_t, a_t)$	환경에 대한 모델인 상태전이 함수 학습
역방향 모델	$a_t = f(s_t, s_{t+1})$	상태전이 함수의 결과로부터 행동을 추론하는 함수 학습
사전 지식 활용	$\Pi = f(s_{1:n})$	일련의 상태 집합으로부터 사전 지식 정보 Π를 추론

사전 지식의 예시로, 로봇 및 무인 자동차 제어에서는 물리적인 법칙에 기반을 두어 미래의 상황을 예측한다. 무인 자동차 연구 [95]에서는 더욱 안정적인 상태 표현을 위하여 일반적인 가치 함수(GVF, general value function) 추론과 함께 도로가 휘어있는 각도를 예측하도록 하였다. 자기지도(self-supervision)는 교사 학습(supervised learning)을 위한 표현 학습에도 흔히 등장하는 개념으로, 데이터 쌍 내부에서 획득할

수 있는 레이블을 출력 값으로 활용하여 추가적인 학습을 수행하는 기법을 포함한다. 따라서 광의로 해석하면 강화학습에서 벨만 방정식에 근거해 계산한 타겟 값을 활용하는 것도 자기지도의 일종으로 볼 수 있다. 그러나 본 논문에서는 협의의 의미로, 강화학습에서 일반화를 촉진시키기 위해 부여하는 추가적인 타겟 값으로 자기지도를 정의하고자 한다.

강화학습 기법은 기본적으로 외부 환경으로부터 받은 보상 값으로부터 학습한다. 이러한 외재적 보상(extrinsic reward)과 구분하기 위해 강화학습 모델 내적으로 얻을 수 있는 보상을 내재적 보상(intrinsic reward)이라고 칭한다. 내재적 보상에는 정방향 모델, 역방향 모두가 포함되며 미로게임에서의 탐색 능력을 향상시키기 위해 이러한 자기지도가 활용된 바 있다 [96]. 학습 시에 추가 정보를 활용한다는 점에서, 내재적 보상과 상태 표현 학습 기법 둘 다 자기지도에 해당된다고 볼 수 있다.

게임과 로봇 영역에서의 여러 연구들은 기존 손실 함수 외에 복수 개의 손실 함수를 동시에 업데이트하는 구조를 통하여, 학습하지 않은 새로운 환경에서 성능을 개선하였으며 더 빠른 수렴 속도를 보여준다. 협동 및 경쟁을 수행하는 환경에서 다른 에이전트의 정책을 학습하는 보조 과제가 도입되었다 [97]. 전체 에이전트의 행동 결과로부터 손실을 부여함으로써 성능을 향상시켰으나, 보조 손실의 비중을 유동적으로 변화시키는 기법은 효과를 보지 못하였으며 에이전트끼리 파라미터를 공유하는 방식보다는 성능이 뒤떨어졌다.

Jaderberg 등 [98]이 제안한 UNREAL 모델은 에이전트에 보조 과업을 할당하여 비지도 방식으로 실시간 학습을 견인한다. 에이전트는 다음 프레임까지 변이하는 픽셀 양, 상태 가치와 다음 보상값을 예측하는 세 가지 보조 과업을 수행한다. 멀티 플레이어 게임에 적용된 연구 [99]에 따르면, 화면 내 상대방의 존재 여부를 판가름하는 주요 피쳐를 추가적인 타겟 값으로 활용하는 경우에, 상태 값으로 활용할 때에 비해 더

좋은 성능을 도출하였다. 두 개의 나무토막을 쌓는 로봇 팔 작업에서는, 보상 희소성(sparsity) 문제를 완화하기 위한 중간 보상이 자기지도로 주어졌다 [100]. 유사한 기술이 여러 뒤따르는 연구들에서 채택되었으며 공통적으로 학습 안정성 및 학습 속도가 개선되었다고 보고하였다 [94, 101]. Bellemare 등 [102]은 기하학적인 특징 공간 분석을 통해 자기지도가 과적합(over-fitting) [103]을 줄이는 데에 기여한다고 주장하였다.

제 3 장 문제 정의

3.1 병렬설비 스케줄링 문제

3.1.1 지연시간 최소화를 위한 병렬설비 스케줄링 문제

제안 기법을 소개하기에 앞서, 순서 의존적 패밀리 셋업이 존재하는 병렬설비 스케줄링 문제를 정의한다. 먼저 N_J 개의 잡 J_j가 존재하며, 이 때 j는 잡의 인덱스를 가리킨다. 각 잡 J_j의 패밀리 F_j는 $\mathcal{F} = \{1, 2, ..., N_F\}$ 와 같이 정의되는 패밀리 집합의 한 원소이다. 각 잡은 i 번째 설비가 M_i로 표시되는 N_M 개의 설비 중 하나에서 작업될 수 있다. 잡 J_j가 M_i에서 작업될 때 소요되는 작업 시간은 잡의 패밀리 F_j와 설비 i에 의존적으로 결정된다. 특정 설비에서 한 번 작업이 종료되면 잡은 작업 완료되고 완료 시간이 결정된다. 한편, 패밀리 f에 속하는 잡의 개수를 $P(f)$로 정의함으로써 패밀리 f에 대한 생산 요구량을 표현한다. 이 때 다음 수식이 만족된다.

$$\sum_{f=1}^{N_F} P(f) = N_J \tag{3.1}$$

스케줄링의 시작 시점에, d_j로 표기되는 J_j의 납기가 주어진다. M_i의 현재 셋업 상태를 $G_i \in \mathcal{F}$ 라 정의하고, 여기서 셋업 상태가 g와 동일하다는 의미는 패밀리가 g인 잡이 셋업 변경 없이 해당 설비에서 작업될 수 있음을 뜻한다. 반면 $f \neq g$인 패밀리 f의 잡이 셋업 상태가 g인 설비 M_i에 할당된다면, $\sigma_{f,g}$로 표시되는 셋업 시간이 작업 시작 이전에 발생한다.

스케줄링의 목적은 각 잡들을 설비들 중 하나에 배치함으로써, 다음과 같이 정의되

는 지연시간 총합 TT를 최소화하는 것이다.

$$TT = \sum_{j=1}^{N_J} \max(0, c_j - d_j) \tag{3.2}$$

여기서 c_j는 J_j의 완료시간을 의미한다. 납기가 모두 0인 본 문제의 특별한 경우는 완료시간 총합 목적함수를 가진 병렬설비 스케줄링 문제와 동일해지고, 이는 강한 NP-난해(strongly NP-hard) 문제임이 증명되어 있다 [22]. 제 2장의 α, β, γ를 활용하여 3.1절에서 다루고 있는 UPMSP를 표현하면 다음과 같다.

$$R_m|s - batch, s_{f,g}|TT \tag{3.3}$$

마지막으로 본 연구에서 가정한 내용은 다음과 같다.

- 스케줄링 시작 시점부터 모든 잡이 준비되어 있으며 모든 설비가 셋업되었다.
- 작업 시간과 셋업 시간은 미리 결정되어 있다.
- 설비 고장은 고려하지 않는다.
- 셋업은 분리할 수 없다. 즉, 설비의 셋업 변경이 완료된 후 즉시 작업이 시작된다.
- 설비가 작업을 시작한 후 유휴 시간을 스케줄에 포함할 수 없다.
- 설비는 한 번에 하나의 잡만 처리할 수 있다.
- preemption이 허용되지 않는다.
- 각 잡의 이동 시간은 0이다.

3.1.2 혼합정수계획 모형

$$\text{Minimize} \qquad TT \tag{3.4}$$

Subject to

$$\begin{gathered} x_{jkm} \leq \sum_{i=0, i\neq j, i\neq k}^{N_j} x_{ijm}; \\ \forall j, k = 1, 2, ..., N_J,\ j \neq k,\ \forall m = 1, 2, ..., N_M \end{gathered} \tag{3.5}$$

$$\begin{gathered} c_j + V \cdot (1 - x_{ijm}) \geq c_i + \sigma_{F_i, F_j} + p_{i, F_j}; \\ \forall i = 0, 1, ..., N_J, \forall j = 1, 2, ..., N_J, i \neq j, \forall m = 1, 2, ..., N_M \end{gathered} \tag{3.6}$$

$$\sum_{j=1, j\neq i}^{N_J} x_{ijm} \leq 1,\ \ \forall i = 0, 1, ..., N_J, \forall m = 1, 2, ..., N_M \tag{3.7}$$

$$T_j \geq c_j - d_j,\ \ \forall i = 1, 2, ., , , N_J \tag{3.8}$$

$$c_j \geq 0, T_j \geq 0,\ \ \forall i = 1, 2, ., , , N_J \tag{3.9}$$

$$\begin{gathered} x_{ijm} \in \{0, 1\}; \\ \forall i = 0, 1, ..., N_J, \forall j = 1, 2, ..., N_J, i \neq j, \forall m = 1, 2, ..., N_M \end{gathered} \tag{3.10}$$

이상의 수식 3.4–3.10은 Kim 등 [30]에서 splitting 특성이 없도록 변형된 혼합정수계획법 모형을 표현한다. 참고로 i, j, k는 잡의 인덱스로, m은 설비의 인덱스로 사용되었다. 잡의 인덱스가 0이란 것은 dummy 잡을 뜻하며 스케줄링 시작 시점 이전에 설비에서 작업되는 잡이다. 결정변수 c_i, T_i, x_{ijm}은 각각 잡의 완료시간, 지연시간, 그리고 설비 m에서 잡 i가 잡 j에 선행하는 지의 여부를 의미한다.

목적 함수 3.4는 전체 잡의 지연시간 총합 최소화이다. 제약 3.5는 잡들이 알맞게

한 설비 내에서 이어지도록 보장한다. 즉, 어떤 잡이 특정 설비에서 작업되려면 선행 잡이 동일한 설비에 존재해야 한다. 제약 3.6은 각 설비에서의 잡 종료 시간이 이전 잡의 종료시간과 두 잡 간의 셋업 시간과 잡의 작업 시간의 합보다 크거나 같아야 함을 의미한다. 제약 3.7은 어떠한 잡에 이어서 작업되는 잡의 개수가 둘 이상일 수 없도록 한다. 제약 3.8과 3.9는 잡의 종료 시간과 지연을 계산한다. 제약 3.10은 결정 변수의 정의역을 제한한다.

3.1.3 예시 공정

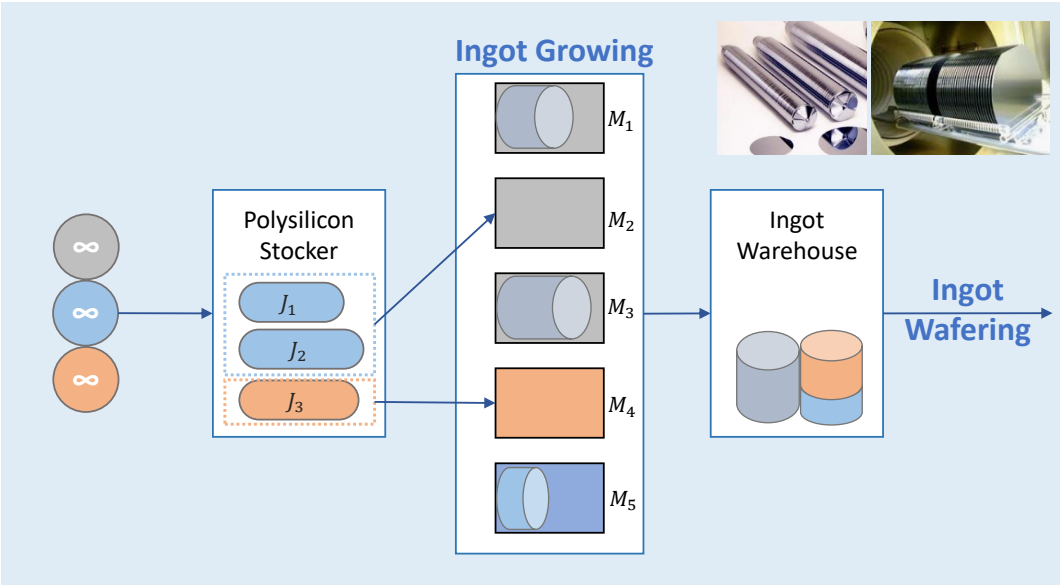


Figure 3.1: 병렬설비로 구성된 잉곳형성 공정

그림 3.1이 나타내고 있는 잉곳형성 공정은 웨이퍼 준비 라인 [47]의 투입 단계로, 지연시간이 최소화되어야 하며 원자재인 poly-silicon의 공급이 무한하다는 특징을 갖는다.

3.2 잡샵 스케줄링 문제

3.2.1 투입량 최대화를 위한 유연잡샵 스케줄링

본 장에서는 병렬설비로 이루어진 복수 개의 공정을 포함하는 잡샵 스케줄링 문제를 정의한다. 본 논문에서 주로 다루는 제약인 순서 의존적 셋업에 더하여, 재진입 특성을 고려한다.

먼저 $J = \{J_j | j = 1, ..., N_J\}$로 정의되는 일련의 잡이 주어지며, 잡 J_j는 F_j로 표기되는 잡 패밀리를 갖는다. 하나의 잡은 미리 정해진 순서를 따르는 공정 시퀀스(sequence)로 이루어져 있으며, J_j의 k 번째 공정을 $O_{j,k}$로 표기한다. 임의의 두 잡의 공정 순서는 잡 패밀리가 같다면 동일하다. 또한 공정 종류가 $O_{j,k}$인 모든 공정의 작업 시간은 $p_{j,k}$로 동일하다. 전체 설비의 집합 $\mathcal{M} = \{M_i | i = 1, ..., N_M\}$이 존재하며, 공정 $O_{j,k}$는 공정 $O_{j,k}$을 작업할 수 있는 설비 집합 $A(O_{j,k}) \in \mathcal{M}$ 중 하나에서 작업될 수 있다. 그러나 공정을 작업하기에 앞서, 설비의 셋업 교체(setup change) 시간이 요구된다. G_i를 M_i의 셋업 상태(setup status)로 정의하고, 최근에 작업한 공정을 $O_{j',k'}$라 한다면 $G_i = O_{j',k'}$이다. 이 때 필요한 셋업 시간은 $\sigma_{j,k,j',k'}$로 정의된다. 이때 $j = j'$이고 $k = k'$이라면 셋업 시간은 0이며 나머지 경우에는 그 이상의 양수 값을 갖는다.

셋업 패밀리 $f = 1, ..., N_F$에 대하여 투입요구량 $In(f)$가 주어지며, 다음 수식이 만족된다.

$$\sum_{f=1}^{N_F} In(f) = N_J \tag{3.11}$$

다음으로 각 잡 J_j에 대하여 투입납기(in-time) $\tilde{d}_j$가 주어질 때, 목적함수인 투입량 INT는 다음과 같이 계산된다.

$$INT = \sum_{j=1}^{N_J} I(t_j < \tilde{d}_j) \tag{3.12}$$

여기서 $I(\cdot)$은 입력문이 참일 때 1을 출력하고 그렇지 않으면 0을 출력하는 지시 함수를 의미하고, t_j는 J_j가 첫 번째 공정의 설비로 투입된 시점을 뜻한다. 또한 패밀리 f의 우선 비중을 w_f라 정의할 때, 가중투입량 $WINT$는 다음과 같이 계산된다.

$$WINT = \sum_{j=1}^{N_J} w_{F_j} \cdot I(t_j < \tilde{d}_j) \tag{3.13}$$

마지막으로 본 연구에서 가정한 내용은 다음과 같다.

- 작업 및 셋업 시간은 미리 알려져 있으며 변화하지 않는다.
- 매거진 스토커의 용량은 무한하다고 가정한다.
- 각 공정은 이전 공정이 종료된 후 수행될 수 있다.
- 각 공정은 시작된 이후 중단되지 않으며, 한 번에 하나의 설비에서만 작업이 수행된다.
- 잡 스토커와 설비 사이에는 일정한 이동 시간이 존재한다.

요약하여 수학 기호를 정리하면 표 3.1과 같다.

3.2.2 예시 공정

반도체 산업은 스마트폰, 웨어러블, 사물인터넷(IoT, Internet of Things) 기기 등에 탑재할 소형 메모리에 대한 수요를 충당하기 위해, 고수준의 집적도를 지닌 다중칩 제품(MCP, Multi-Chip Product) 생산에 주력하고 있다. MCP 조립 라인은 메모리 제품의 소형성(compactness)과 함께 대용량(high-capacity)을 달성할 수 있다는 장점이 있으나, 서로 얽혀있는 복잡한 조립 단계로 구성된다 [52]. 보다 구체적으로, 반도체 팹의 전 공정을 거친 후 후 공정의 패키징 라인에서는 back lap, wafer sawing, die attach

Table 3.1: 잡샵 스케줄링 문제 구성 요소

표기	설명
N_M	설비 대수
N_J	잡 개수
N_F	잡 패밀리 개수
N_O	공정 종류의 수
M_i	i번째 설비
G_i	M_i의 셋업 상태
J_j	j번째 잡
$O_{j,k}$	j번째 잡의 k번째 공정
$A(O_{j,k})$	공정 $O_{j,k}$을 작업 가능한 설비 집합
$p_{j,k}$	공정 $O_{j,k}$의 작업 시간
$\sigma_{j',k',j,k}$	$O_{j,k}$을 $O_{j',k'}$로 셋업된 설비에서 작업하기 위해 필요한 셋업 시간
F_j	j번째 잡의 잡 패밀리
$In(f)$	잡 패밀리 f의 투입 요구량
$\tilde{d}_j$	j번째 잡의 투입납기
t_j	j번째 잡의 투입 시점
INT	투입량
$WINT$	가중투입량

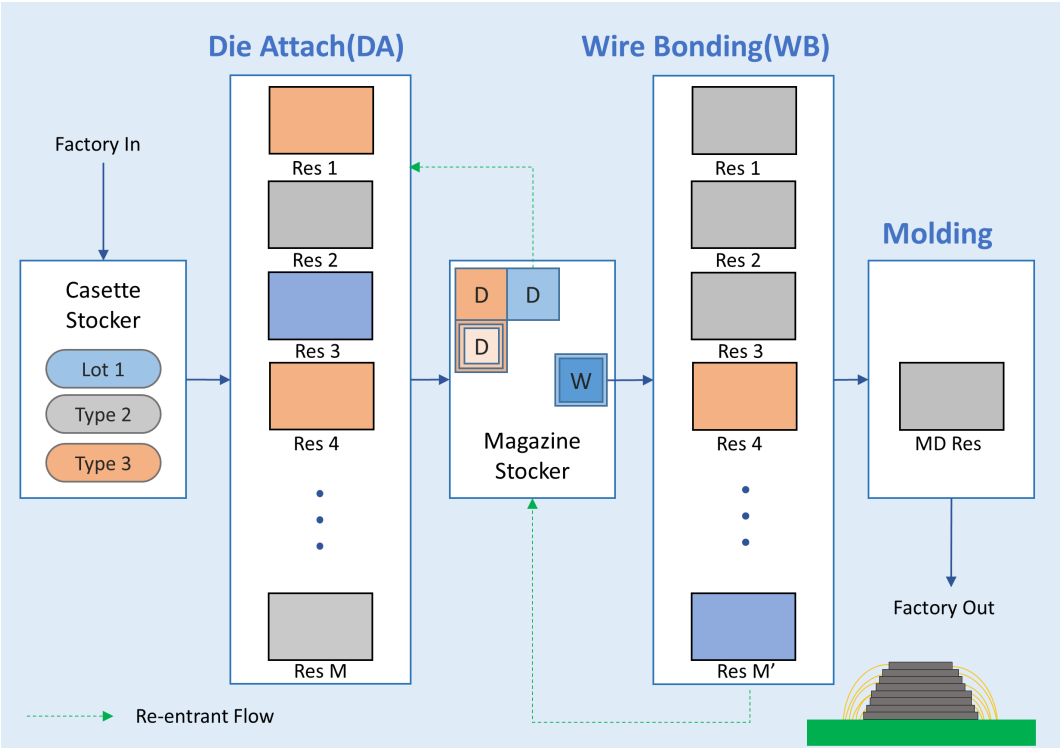


Figure 3.2: 패키징 라인의 DA 및 WB 공정 흐름도

(DA), wire bonding (WB), molding (MD)을 거쳐 제품을 생산하게 된다. 특히 DA 및 WB 단계(stage)에서는 웨이퍼가 로트(lot)로 그룹화 되어 처리되며, 여러 번 재진입이 이루어진다 [13].

그림 3.2는 예시 공정인 반도체 패키징 라인에서 DA 및 WB 단계에서의 로트 흐름을 보여준다. 각각의 로트는 WB 공정 진입 이전에 DA 공정에서 작업되어야 하며, 최종적으로 WB 공정을 작업한 후 MD 공정을 거쳐 factory out된다. 초록색으로 표현된 파선은 재진입 흐름을 나타내며, WB 공정을 완료한 후 다시 DA 공정을 시작하는 로트의 흐름을 보여준다. 각 단계의 중간에는 로트가 일시적으로 머무는 두 가지 유형의 스토커(stocker)가 존재한다. 먼저 카세트 스토커(casette stocker)는 새로 준비된 로트가 첫 번째 DA 공정으로 투입되기를 기다리는 위치를 제공한다. 반면 매거진 스토

커(magazine stocker)는 이미 투입된 로트가 기다리는 위치로, WB 공정을 완료한 후 다음 DA 공정으로 재진입을 기다리는 로트와 DA 공정을 완료한 후 WB 공정으로의 진입을 기다리는 로트가 속한다. 즉, 새로 도착한 로트와 재진입 로트가 함께 위치하여 복잡한 흐름이 발생한다.

이러한 재진입 특성으로 인해 초래되는 어려움으로 WIP(work-in-process) 수준 제어와 함께 산출량(throughput)의 감소를 들 수 있다. 이미 투입되어 한 번 이상 DA 또는 WB 공정을 작업한 로트는 매거진 스토커에서 대기하므로, 지속적으로 WIP이 증가하게 된다. 이는 결과적으로 패키징 라인의 병목 구간인 DA와 WB에서 작업 시간이 길어짐에 따라, 전체 산출량의 감소로 이어진다 [49]. 구체적으로 DA 공정 설비가 재진입 로트를 고려하지 않고 새로 준비된 로트의 투입을 우선시 한다면, WIP이 과도하게 올라간다. 반대로 재진입 로트에 높은 우선순위를 부여하면 WB 설비 가동률이 저하되어 산출량이 감소할 수 있다.

이러한 어려움을 극복하여 납기를 준수하기 위해, 현장에서는 산출량(out-target) 대신에 투입량(in-target)을 최대화하는 방식을 취하고 있다 [48]. 이 때 투입량은 각 제품 종류별로 투입납기(in-time) 안에 투입되어야하는 할당량을 뜻한다. 따라서 투입이 이루어지는 DA 공정에서 주어진 투입납기에 알맞게 로트를 배치하고, WB 공정에서는 재진입하는 로트의 수를 조절하는 스케줄링 의사결정이 요구된다. 해당 패키징 라인의 스케줄링 문제는 재진입이 있는 유연잡샵 스케줄링 문제로 간주할 수 있으며, 주어진 투입량을 최대화하며 로트를 투입하는 것이 목적이다.

제 4 장 자기지도 기반 심층강화학습을 이용한 병렬설비 스케줄링

4.1 MDP 모형

DRL을 도입하려면 고려 대상인 스케줄링 문제가 MDP로 공식화되어야 한다. 따라서 본 장에서는 행동, 상태, 보상, 상태 전이에 대하여 상세하게 정의할 것이다. 본 연구에서는 k 번째 단계(time-step)에서의 상태, 행동, 보상을 각각 s_k, a_k, r_k로 표시하며, 각 기간에서 머무는 시간이 T로 일정하다고 모델링하였다.

4.1.1 행동 정의

행동이 임의의 잡과 설비의 쌍들 각각에 대해 정의되면, 행동 공간의 크기는 ${N_F}^{N_M}$으로 급격히 늘어나게 된다 [82]. 본 연구에서는 잡과 설비의 수가 늘어나도 행동 공간의 크기를 유지하기 위하여, 잡의 패밀리와 설비의 셋업 상태의 튜플(tuple)로 행동을 정의하였다. 그 다음 임의의 기간에서 에이전트는 특정한 설비에서의 셋업 변경을 1회 결정할 수 있다고 가정하였다. 먼저 가능한 모든 행동의 집합 $\mathcal{A}$는 다음과 같이 정의된다.

$$\mathcal{A} = \{(f, g) | f = 1, ..., N_F, g = 1, ..., N_F\} \tag{4.1}$$

그리고 나서 s_k일 때의 가능한 행동 집합을 $\mathcal{A}_k \subset \mathcal{A}$으로 표시한다. 이 때 행동 $a_k = (f_k, g_k)$는 기간 k 동안에 패밀리 f_k의 어떤 잡이 셋업 상태가 g_k인 한 설비에 배치될 것이란 의미를 지닌다. 단, $a_k \in \mathcal{A}_k$ 이다. 다시 말하여 a_k는 오직 패밀리 f_k인 대기 잡이 존재하고 g_k로 셋업된 유휴 설비가 기간 k 동안에 존재할 때에만 가용하다고 간주된다.

Table 4.1: 상태를 구성하는 6개 행렬과 1개 벡터

기호	명칭	차원
$\mathbf{S}_w$	대기 잡 상태	$N_F \times 2H_w$
$\mathbf{S}_p$	작업 중인 잡 상태	$N_F \times H_p$
$\mathbf{S}_c$	작업 중인 잡의 특징	$N_F \times 4H_c$
$\mathbf{S}_s$	설비 셋업 시간	$N_F \times N_F$
$\mathbf{S}_u$	가동률 히스토리	$N_F \times 3$
$\mathbf{S}_a$	행동 히스토리	$N_F \times 2$
$\vec{S}_f$	패밀리 독립적인 히스토리	3

4.1.2 상태 표현

강화학습 기반 기법을 스케줄링 문제에 적용할 때, 현재 설비 및 잡의 상태 그리고 그들 사이의 관련성이 상태 표현에 활용되는 주요한 정보로 기능한다. 만약 표현한 상태의 차원이 잡과 설비의 수가 변화함에 따라 달라진다면, DNN이 재학습되어야 할 것이다. 상기한 변화가 일어나는 상황에서도 재학습 없이 스케줄링 문제를 풀기 위하여, 본 연구에서는 생산 요구량과 납기 및 설비 수에 독립적인 차원을 갖는 패밀리 기반 상태 표현(FBS, Family-Based State)을 제안한다.

제안한 상태는 표 4.1에 정리된 기호, 명칭, 차원을 갖는 여섯 개의 이차원 행렬과 하나의 벡터로 이루어져 있다. s_k에 대한 이해를 돕기 위해, 패밀리 수가 3인 스케줄링 문제에서의 예시를 그림 4.1에 표현 하였다. 빨강, 초록, 파랑은 각각 패밀리 1, 2, 3을 나타내며 행렬 내부의 숫자는 s_k의 상태값과 상응한다. 이제부터 s_k의 각 구성성분들을 정의할 것이다. 이 때 표기의 간결성을 위해 각각의 구성성분의 기호에 인덱스 k를 생략하였다.

- $\mathbf{S}_w$: 대기 중인 잡의 납기를 반영하기 위해, f 번째 행의 값은 납기가 하나의 기간에

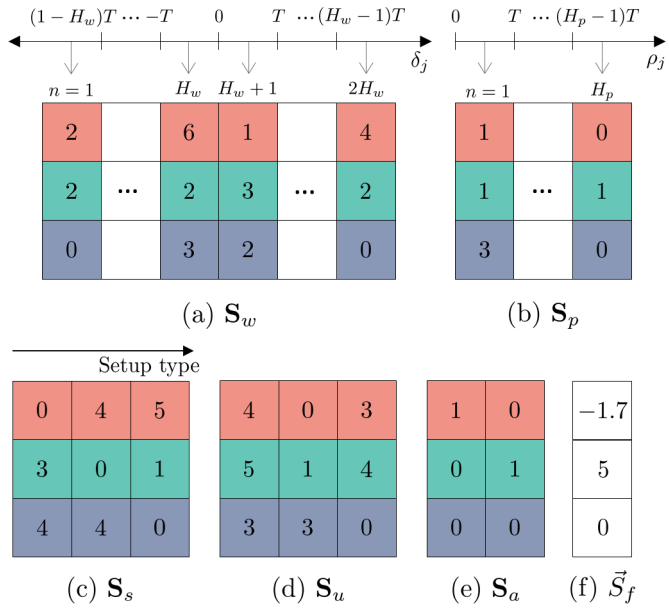


Figure 4.1: 이차원 행렬로 표현된 s_k의 예시

속하는 패밀리 f에 대한 대기 잡 개수를 의미한다. 각 대기 잡 J_j를 $\delta_j = d_j - kT$로 표시되는 납기까지의 남은 시간에 따라 개별적으로 셈하기 위하여, 다음과 같이 행렬 $\mathbf{S}_w(f,n)$을 정의한다.

$$\mathbf{S}_w(f,n) = \begin{cases} |\{J_j \in \mathcal{W}_k(f) \,|\, \delta_j \le (1-H_w)T\}| & n = 1 \\ |\{J_j \in \mathcal{W}_k(f) \,|\, \delta_j > (H_w-1)T\}| & n = 2H_w \\ |\{J_j \in \mathcal{W}_k(f) \,|\, \left\lceil \frac{\delta_j}{T} \right\rceil = n - H_w\}| & \text{otherwise} \end{cases} \tag{4.2}$$

이 때 $\mathcal{W}_k(f)$, $\lceil \cdot \rceil$, H_w는 각각 시점 s_k에서 패밀리가 f인 대기 잡의 집합, 올림 함수(ceiling function), $\max_j d_j/T$ 보다 작은 값을 갖는 양의 정수를 의미한다. 수식 4.2에 표시되어 있듯이, $\mathbf{S}_w(f,1)$와 $\mathbf{S}_w(f,2H_w)$는 각각 δ_j 값이 $(1-H_w)T$ 보다 작고 $(H_w-1)T$ 보다 큰 패밀리 f인 대기 잡의 개수를 의미한다. 이는 모든

대기 잡의 납기를 파악하면서도 $\mathbf{S}_w$의 열 개수를 제한하기 위한 표현 방식이다. 실용적인 관점에서, 에이전트는 현재 시점과 거리가 먼 납기를 가진 잡을 더 이상 구분할 필요가 없다. 반면 납기일이 비교적 근접한 다른 잡들은 해당하는 기간 $\left\lceil \frac{\delta_j}{T} \right\rceil$에 따라 별도로 집계하는 편이 효율적이다. $\mathbf{S}_w$를 통해, 에이전트는 얼마나 많은 잡이 앞으로 배치되어야 하며 얼마나 그들의 납기가 임박했는 지에 대해 파악할 수 있다. 그림 4.1(a)의 예시에서, $\mathbf{S}_w(1, H_w)$는 패밀리가 1이고 $\left\lceil \frac{\delta_j}{T} \right\rceil = 0$인 대기 잡이 여섯 개 있음을 나타낸다.

- $\mathbf{S}_p$: 이 행렬은 남은 작업 시간이 하나의 기간에 포함되는 패밀리 별 작업 중인 잡의 수를 포함한다. 여기서 작업 중인 잡이란 현 시점에서 어떠한 설비에서 작업이 진행 중인 잡을 의미한다. 작업 중이거나 작업 완료된 잡의 지연시간은 이미 결정된 까닭에, 작업 중인 잡의 납기는 지연시간 최소화와 관련이 없다. 반면 남은 작업 시간은 작업 중인 잡이 완료된 이후에 같은 설비에 배치될 대기 잡의 완료시간에 영향을 미친다. 따라서 s_k 시점에서 패밀리 f인 작업 중인 잡의 집합 $\mathcal{P}_k(f)$에 속하는 각 잡 J_j이 남은 작업 시간에 따라 분류되도록 설계하였다. $\mathbf{S}_p(f, n)$의 정의는 다음과 같다.

$$\mathbf{S}_p(f,n) = \begin{cases} |\{J_j \in \mathcal{P}_k(f) \,|\, (n-1)T < \rho_j \leq nT\}| & n < H_p \\ |\{J_j \in \mathcal{P}_k(f) \,|\, (n-1)T < \rho_j\}| & n = H_p \end{cases} \tag{4.3}$$

여기서 H_p는 $\max_j \rho_j/T$ 보다 작은 양의 정수를 뜻한다. 수식 4.2와 유사한 방식으로 행렬 $\mathbf{S}_p$의 행 차원이 제한되었음에 주목하라. 그림 4.1(b)를 예로 들면, $\mathbf{S}_p(2,1)$의 값이 1이므로 다음의 두 가지 조건을 충족하는 잡이 한 개 존재함을 알 수 있다. 즉, 잡이 $\mathcal{P}_k(2)$에 속하면서도, 잡의 남은 작업 시간이 T보다 짧은 잡이 한 개 존재한다.

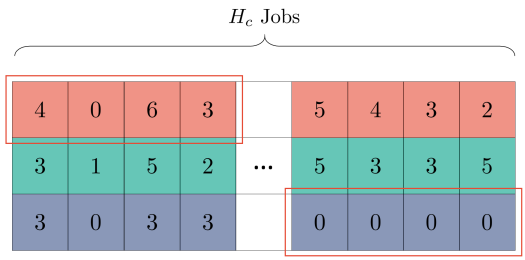


Figure 4.2: 이차원 행렬로 표현된 $\mathbf{S}_c$의 예시

- $\mathbf{S}_s$: s_k 이후 발생할 지연시간을 예측할 수 있도록 셋업 시간을 파악하기 위하여 패밀리 g에서 f로 셋업 변경하는 데에 소요되는 셋업 시간을 $\mathbf{S}_s(f,g)$에 나타내었다. 구체적으로, $\mathbf{S}_s(f,g)$은 다음과 같이 정의된다.

$$\mathbf{S}_s(f,g) = \begin{cases} \sigma_{f,g} & \text{if } (f,g) \in \mathcal{A}_k \\ \sigma_{\max} & \text{otherwise.} \end{cases} \tag{4.4}$$

  $\sigma_{\max}$는 최대 셋업 시간을 의미한다. 만약 $(f,g) \notin \mathcal{A}_k$이라면, $\mathbf{S}_s(f,g)$ 값은 최악의 경우를 고려하기 위하여 $\sigma_{\max}$으로 설정된다.

- $\mathbf{S}_c$: 이 행렬은 패밀리 별로 $\mathbf{S}_p$를 표현하는 과정에서 손실된 정보를 보완하기 위해 설계되었다. 특징 정보로는 작업 시간, 셋업 시간, 납기, 남은 작업 시간을 표현하였는데, 네 가지 특징을 선정한 이유는 상태 차원을 작게 제한하면서도 동일한 패밀리의 서로 다른 두 잡을 구별하기 위함이다. 또한 행렬의 차원을 N_J와 독립적으로 유지하기 위하여, 각 패밀리 별로 납기가 가장 짧은 H_c 개 잡에 대한 정보만을 포함한다. 이 때, H_c는 N_M보다 작거나 같은 양의 정수로 설정된다. $\mathcal{P}_k(f)$에 속한 잡들 중 l번째로 납기가 작은 잡을 J_j로 표시하고 M_i에서 작업중이라 가정할 때, $\mathbf{S}_c(f,n)$은 다음과 같이 정의된다. 단, $\forall l = 1,...,H_c, \forall n = 1,...,4H_c$

이다.

$$\mathbf{S}_c(f,n) = \begin{cases} p_{i,f} & n = 4l-3 \\ \sigma_{f,g'} & n = 4l-2 \\ d_j & n = 4l-1 \\ c_j - kT & n = 4l \end{cases} \tag{4.5}$$

여기서 g'은 J_j를 작업하기 이전에 M_i의 셋업 종류를 의미한다. 그림 4.2를 예로 들면, 패밀리 1의 첫 번째 잡의 특징 값 $\mathbf{S}_c(1, l=1)$은 $(4,0,6,3)$이다. 한편 패밀리 3의 H_c 번째 잡의 특징 $\mathbf{S}_c(3,H_c)$는 모두 0인데, 이는 $|\mathcal{P}(3)|$이 H_c보다 작아 해당하는 잡이 존재하지 않기 때문이다.

- $\mathbf{S}_u, \mathbf{S}_a, \vec{S}_f$: DRL에 기반을 둔 스케줄링 문제 풀이에 효과적이라 알려져 있는 s_k까지의 잡과 설비, 에이전트의 히스토리 정보를 통합하기 위해 고안되었다. $\mathbf{S}_u(f,1)$와 $\mathbf{S}_u(f,2)$는 각각 패밀리 f인 잡들을 작업하기 위해 s_k 시점까지 소요된 작업 시간과 셋업 시간의 비율을 나타낸다. 그리고 $\mathbf{S}_u(f,3)$는 패밀리가 f인 작업 완료된 잡의 개수를 의미한다. 다음으로, $\mathbf{S}_a$는 최근 행동인 a_{k-1}을 표현한다. $\mathbf{S}_a(\cdot,1)$와 $\mathbf{S}_a(\cdot,2)$은 각각 잡의 패밀리와 설비의 셋업 종류를 의미하며, One-hot 부호화 [104]를 통해 N_F-차원의 벡터로 표현되었다. 마지막으로, $\vec{S}_f$는 특정한 패밀리로 그룹화할 수 없는 세 개의 히스토리 특징으로 구성된다. $\vec{S}_f(1)$, $\vec{S}_f(2)$, $\vec{S}_f(3)$은 각각 r_{k-1}, k와 s_k가 마지막 상태일 때에는 1이고 나머지 경우엔 0인 이진 변수를 뜻한다. 그림 4.1(e)와 (f)를 보면, $k=5$이고 $r_4 = -1.7$이며 $a_4 = (1,2)$이고 s_5는 마지막 상태가 아니라는 네 가지 정보를 알아낼 수 있다.

요약하자면, 제안 상태의 차원 총합은 $N_F \times (2H_w + H_p + 4H_c + N_F + 3 + 2) + 3$으로 차원성이 생산 요구량 및 납기와 설비 대수에 독립적이다.

DNN의 입력값 크기로 인한 왜곡을 최소화하기 위하여 [105], 실제 학습 시에는 s_k의 각 값들을 0에서 1사이로 정규화하였다. $\mathbf{S}_w$와 $\mathbf{S}_p$의 각 원소들을 비롯하여 잡의 개수를 셈한 값들은 모두 N_J로 나누어서 1 이하의 값이 되도록 보장하였다. $\mathbf{S}_u$의 첫 번째, 두 번째 열의 값들은 시점 kT까지 발생한 설비의 셋업 및 작업 기록을 의미하므로 kTN_M으로 나누어주었다. $\mathbf{S}_s$의 셋업 시간과 $\mathbf{S}_c$의 셋업 시간, 작업 시간, 남은 작업 시간은 모두 $\max(\sigma_{f,g} + p_{i,j})$로 나누었다. 납기를 나타내는 값들은 kT로 나누었다.

4.1.3 보상 정의

본 연구의 보상은 Zhang 등 [62]에서 정의된 수식을 기반으로 하며, 각 기간에서 소요되는 시간이 T로 일정하다는 가정에 맞게 다음의 클립 함수를 활용하여 수정되었다.

$$\lambda_k(t) = \max(kT, \min((k+1)T, t)) \tag{4.6}$$

그리고 r_k를 다음과 같이 정의한다.

$$r_k = \sum_{j=1}^{N_J} - \max(0, \lambda_k(c_j) - \lambda_k(d_j)) \tag{4.7}$$

수식 4.7은 본 연구의 보상 값이 개별 잡들의 지연시간을 수식 4.6으로 클립한 값들의 음수 총합과 일치함을 의미한다. 각각의 c_j와 d_j가 기간 k의 시작 시점 kT와 종료 시점 $(k+1)T$로 클립됨에 따라 오직 기간 k 동안에 발생한 지연만이 r_k 에 포함된다. 보상 계산 시에, s_{k+1} 시점까지 대기 중인 잡으로 남아있는 J_j의 c_j 값은 무한대라 가정하였고, $(k+1)T$로 클립된다. 최종적으로 모든 기간에 대한 보상 총합은 TT와 같으며, 이는 [62]에 증명되어 있다. 따라서 Q 값을 정확하게 추론하는 것이 TT 최소화와 상응한다고 기대할 수 있다.

Algorithm 1 행동에 의한 상태 전이

Input: $a_k = (f_k, g_k)$, $\mathcal{W}_k = \bigcup_{f'=1}^{N_F} \mathcal{W}_k(f')$
Output: State s_{k+1}, reward r_k, $\mathcal{W}_{k+1}$
1: $\mathcal{M} \leftarrow \{M_i | G_i = g_k\}$, $i = 1, ..., N_M$
2: $M_* \leftarrow \arg\min_{M_i \in \mathcal{M}} p_{i,f_k}$
3: **while** $\mathcal{W}_k \neq \emptyset$ and $\min_{i'} E_{i'} < (k+1)T$ **do**
4: $M_i \leftarrow \arg\min_{i'} E_{i'}$
5: $g \leftarrow G_i$
6: **if** $M_* = M_i$ **then**
7: $f \leftarrow f_k$
8: **else**
9: $f \leftarrow \arg\min_{f'} \sigma_{f',g}$, where $|\mathcal{W}_k(f')| > 0$
10: **end if**
11: $J_j \leftarrow \arg\min_{J_{j'} \in \mathcal{W}_k(f)} d_{j'}$
12: Assign J_j on M_i
13: $E_i \leftarrow E_i + p_{i,f} + \sigma_{f,g}$
14: $\mathcal{W}_k \leftarrow \mathcal{W}_k \setminus \{J_j\}$
15: **end while**
16: Obtain transited state s_{k+1}
17: Calculate r_k from Eq. (4.7)

4.1.4 상태 전이

a_k를 실행한 후 s_k에서 s_{k+1}로 상태 전이가 이루어진다. 이러한 상태 전이 절차를 알고리듬 1에 설명하였다. 여기서 E_i는 설비 M_i에서 현재 작업 중인 잡이 완료될 시점으로 정의된다.

먼저 첫 번째 줄에서 셋업 상태가 g_k인 설비로 구성된 집합 $\mathcal{M}$을 얻는다. 두 번째 줄에서는 $\mathcal{M}$ 중에서 패밀리 f_k인 잡에 대한 작업 시간이 가장 짧은 설비 M_*를 선택한다. 세 번째에서 15번째 줄 까지는 하나 이상의 대기 작업이 있고 모든 설비 중 최소 E_i는 현재 기간의 종료 시간 k를 초과하지 않는다는 두 가지 조건을 충족할 때까지

반복된다. 네 번째 줄에서는 E_i가 모든 머신 중 가장 작은 머신 M_i가 선택된다. 그런 다음, g라고 하는 설비의 셋업 상태를 구한다(다섯 번째 줄). 6번째에서 10번째 줄까지는 F로 표시된 M_i에서 수행할 다음 작업의 패밀리를 결정한다. M_*가 M_i와 같으면 f는 a_k에 따라 f_k로 설정된다(7번째 줄). 그렇지 않으면 M_i에서 발생하는 설정 시간을 최소화하기 위해 f를 선택한다(9번째 줄). 패밀리가 f인 대기 중인 작업 중 가장 빠른 납기를 가진 작업 J_j이 선택된다(11번째 줄). 12번째 줄에서 J_j가 M_i에 할당된 후, E_i와 $\mathcal{W}_k$가 업데이트된다(13번째 및 14번째 줄). 마지막으로 s_{k+1}, $\mathcal{A}_{k+1}$ 및 r_k를 구한다(16번째 및 17번째 줄).

4.1.5 예시

그림 4.3은 k 에서 $k+1$ 기간 동안 수립된 스케줄을 보여준다. 여기서 빨강, 초록, 파랑은 각각 패밀리 1, 2, 3을 나타낸다. s_k에서 M_1과 M_2는 각각 J_1과 J_2를 수행하고 있으며, J_3에서 J_7까지의 잡이 할당되기를 기다리고 있다. a_k는 $(2,1)$이므로 셋업 상태

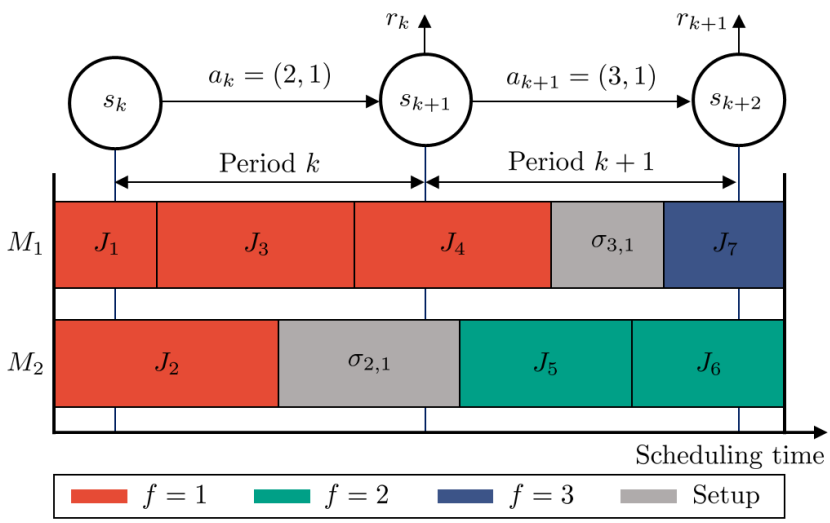


Figure 4.3: 상태 전이로부터 획득한 스케줄의 예시

가 1인 설비에 패밀리 2인 잡이 할당되어야 한다. 알고리듬 1의 두 번째와 11번째 줄을 따름으로써 J_5는 M_2에 할당되며, 잡을 작업하기 전에 $\sigma_{2,1}$가 발생한다. 한편, 패밀리 1에 속하는 J_3과 J_4는 G_1이 1인 M_1에 연속적으로 할당된다. 기간 종료 시 k, s_{k+1} 및 r_k는 상태 전환의 결과로 구해진다. $a_{k+1} = (1, 3)$를 실행한 후, M_2가 동일한 패밀리의 J_6을 계속 작업하는 동안 패밀리 3의 잡 J_7이 M_1에 할당된다. 요약하면 J_3, J_4, J_5는 a_k에 의해 할당되고 a_{k+1}에 따라 J_6 및 J_7이 할당된다.

4.2 신경망 학습

4.2.1 심층신경망 구조

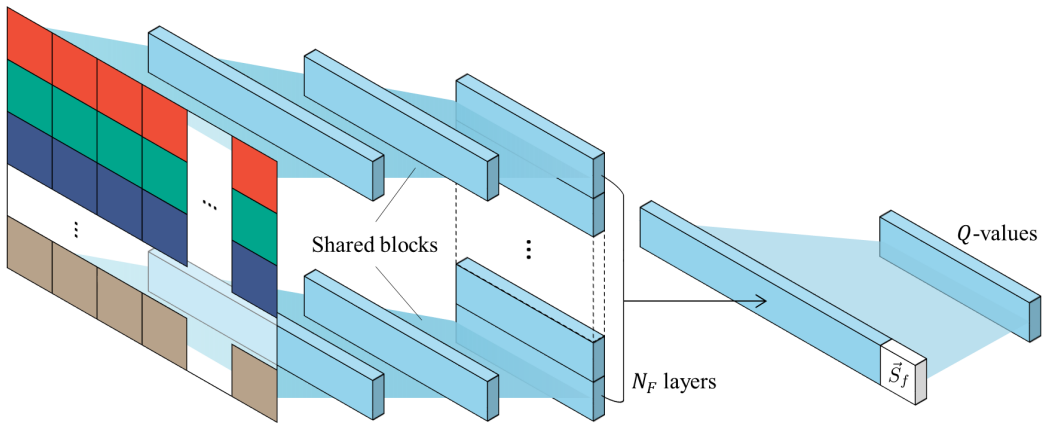


Figure 4.4: 제안하는 파라미터 공유 기반 DNN 구조도

주어진 상태에 대한 Q-값을 추정하기 위해 [73]에서 제안된 심층 Q-네트워크(DQN, Deep Q Network)를 사용했다. DQN은 상태 s를 입력으로 사용하고 $Q_\theta(s, a)$라고 하는 가능한 모든 작업에 대해 Q-값을 출력한다. 여기서 θ는 신경망의 파라미터이고 $a \in \mathcal{A}$이다. 그림 4.4는 파라미터 공유에 기반을 둔 완전연결 신경망의 구조(architecture)를 보여준다. 빨강, 초록, 파랑과 갈색은 각기 다른 패밀리를 나타내며, 하늘색은 은닉층과 출력층을 가리킨다. 파라미터 공유 구조는 단일 로트 크기 조정 [59] 및 컴퓨팅 플랫폼 리소스 스케줄링 [106] 문제를 해결하기 위해 채택된 바 있다.

DQN의 입력은 여섯 개의 행렬 $\mathbf{S}_w$, $\mathbf{S}_p$, $\mathbf{S}_c$, $\mathbf{S}_s$, $\mathbf{S}_u$ 및 $\mathbf{S}_a$를 연결하여 합친 행렬과 같다. 입력의 각 행 벡터는 여러 개의 은닉층으로 구성된 블록(block)에 연결된다. 네트워크 파라미터 크기를 줄이고 학습 효율을 높이기 위해 각 블록에 대한 파라미터가 동일하도록 공유된다. 마지막 은닉층은 N_F 개 블록의 마지막 층의 노드 값과 상태 벡터 $\vec{S}_f$를 연결하여 구성된다. 마지막으로, 출력층의 노드 수는 가능한 모든 행동의 개수인

$N_F{}^2$와 같다. 음수 Q-값을 나타내기 위한 출력층을 제외하고는 ReLU 함수 [107]를 활성화 함수로 채택하였다.

4.2.2 손실 함수

DQN은 예측값 $Q_\theta(s,a)$과 target Q-값 [73]에 의해 계산되는 TD 오차(temporal difference error)로부터 학습된다. 주어진 (s_u, a_u, r_u, s_{u+1})로부터의 TD 오차는 다음과 같이 공식화된다.

$$\eta_u = r_u + \gamma \max_{a' \in \mathcal{A}_{u+1}} Q_{\hat{\theta}}(s_{u+1}, a') - Q_\theta(s_u, a_u) \tag{4.8}$$

여기서 γ와 $\hat{\theta}$는 각각 할인율(discount factor)과 학습 중인 DQN과 동일한 네트워크 구조를 가진 target DQN의 파라미터를 나타낸다.

트랜지션(transition)은 상태, 행동, 보상 및 다음 상태의 quadruple로 정의한다. 이 때 DQN의 손실 $L_Q(\theta)$는 추출(sample)된 트랜지션으로부터 TD 오차를 계산하여 얻어지는 값이며 다음과 같이 정의된다.

$$L_Q(\theta) = \frac{1}{N_{TR}} \sum_{u=1}^{N_{TR}} h(\eta_u) \tag{4.9}$$

여기서 N_{TR}는 추출된 트랜지션의 개수이다. 또한 h는 다음과 같이 정의된 손실 함수이다.

$$h(\eta) = \begin{cases} \frac{1}{2}\eta^2 & \text{if } \eta \leq \frac{1}{2}, \\ \frac{1}{2}(|\eta| - (\frac{1}{2})^2) & \text{otherwise.} \end{cases} \tag{4.10}$$

DQN 학습의 안정성을 더욱 높이기 위해, 식 4.10과 같이 Huber 손실 [108]을 평균 제곱 오차 대신에 채택했다.

Algorithm 2 DQN 학습 절차

Input: Scheduling problem
Output: Q-network

1: **Initialization** : Set network Q_θ with random weight θ, target network $Q_{\hat{\theta}}$ with $\hat{\theta} = \theta$, and replay buffer B to size N_B.
2: **for** $e = 1, 2, \ldots, N_E$ **do**
3: $k \leftarrow 0$
4: Initialize F_j, d_j of N_J jobs, and status of N_M machines.
5: Observe s_k, and $\mathcal{W}_k$
6: **while** $\mathcal{W}_k \neq \emptyset$ **do**
7: With probability ε select a_k randomly from $\mathcal{A}_k$
8: otherwise $a_k \leftarrow \arg\max_{a \in \mathcal{A}_k} Q(s_k, a)$
9: Get s_{k+1}, $\mathcal{A}_{k+1}$, r_k, $\mathcal{W}_{k+1}$ from **Algorithm 1**
10: Store transition (s_k, a_k, r_k, s_{k+1}) in B
11: Sample N_{TR} transitions $(s_u, a_u, r_u, s_{u+1}) \in B$
12: Calculate loss L from (4.8)-(4.10)
13: Perform a gradient descent step on L w.r.t. θ
14: $k \leftarrow k + 1$
15: **end while**
16: Synchronize $\hat{\theta}$ to θ at every N_U episodes
17: **end for**
18: **return** Q-network

4.2.3 DQN 학습 절차

알고리듬 2는 제안된 DQN의 학습 절차를 설명한다. DQN 학습을 위한 스케줄링 문제가 주어질 때, 대기 중인 잡이 없는 한 스케줄링 프로세스는 계속 반복된다 (세 번째에서 16번째 줄). 이 때 하나의 스케줄링 프로세스의 시작부터 끝까지를 에피소드(episode)라 명명하며, e는 현재 수행 중인 에피소드의 인덱스를 나타낸다. 스케줄링 프로세스는 e가 N_E로 표시된 학습 에피소드 수에 도달할 때까지 계속된다.

에피소드가 시작될 때 k는 0으로 설정되고 N_J 작업과 N_M 설비는 초기화된다(세 번째와 네 번째 줄). 다섯 번째 줄에서 에이전트는 처음에 $\mathcal{W}_k$와 함께 s_k를 관찰한다.

각 시점 k에 대해 에이전트는 7번째 및 8번째 줄에 제시된 ε-greedy 정책 [56]으로 부터 a_k를 선택한다. 여기서 $\varepsilon \in [0, 1]$은 무작위로 행동을 선택할 수 있는 확률이다. 상태 전이가 s_k에서 s_{k+1}로 일어난 후 (9번째 줄), 상태, 행동, 보상 및 다음 상태로 구성된 한 개의 트랜지션이 재현 버퍼(replay buffer)에 저장된다 (10번째 줄). 이 때, 재현 버퍼와 그 크기는 각각 B와 N_B로 표시된다. 또한 만약 B에 저장된 트랜지션 수가 N_B를 초과하면 가장 오래된 트랜지션이 제거된다. N_{TR}을 추출하는 트랜지션의 개수로 정의하면, 11번째 줄은 N_{TR} 개의 트랜지션이 DQN을 학습시키기 위해 추출된다는 것을 나타낸다. 앞서 4.2.2 절에서 설명한 대로 $L(\theta)$로 표시된 TD 손실을 계산함으로써 네트워크 파라미터 θ가 업데이트된다(12번째 및 13번째 줄). 17번째 줄은 타겟 네트워크 파라미터 $\hat{\theta}$가 θ [73]로 주기적으로 교체된다는 것을 보여준다. 마지막으로, N_E번째 에피소드에 도달하며 학습이 완료된 DQN을 획득한다(18번째 줄).

4.2.4 DQN 평가 절차

DQN 학습을 마친 후, 평가 절차(test procedure)를 수행하여, 학습 문제로부터 생산 요구량과 납기가 변화한 평가 스케줄링 문제를 풀이한다. 평가 절차 중에는 알고리듬 2의 8번째 줄의 무작위 탐색 과정은 더 이상 실행되지 않는다. DQN을 학습하는 데 필요한 10-13 및 16번째 줄을 제외한 나머지 절차는 알고리듬 2와 동일하다.

그림 4.5는 학습 및 평가 절차를 포함한 제안 기법의 전체 흐름도를 나타낸다. DQN 학습 시에는 스케줄 횟수가 N_E에 도달할 때까지 절차가 반복된다. 학습된 DQN과 평가 문제가 주어졌을 때 제안 기법은 알고리듬 1과 2를 반복하여, 스케줄링 문제 수만큼의 스케줄을 얻을 수 있다. 평가 시엔 하나의 스케줄링 문제를 거의 즉각적인 시간 안에 여러 번 풀이할 수 있다. 따라서 예측할 수 없는 가변성에 대비하기 위해 복수 개의 DQN을 저장한 후, 이 중에서 가장 성능이 좋은 스케줄을 적용하는 방식을 취할 수 있다.

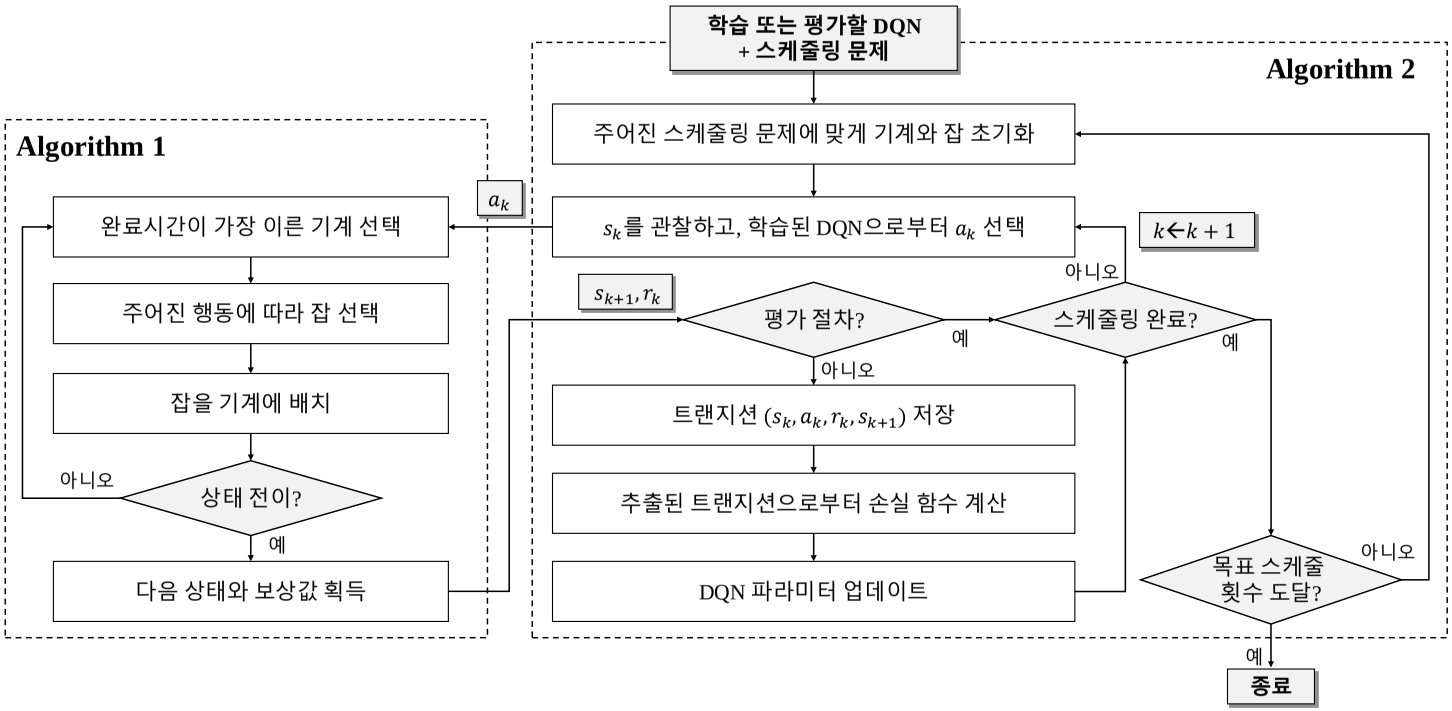


Figure 4.5: 알고리듬 1, 2를 포함한 제안 기법의 흐름도

4.3 스케줄링 문제에서의 자기지도

본 장에서는 먼저 기존의 자기지도 기반 DRL 연구에서 공통적으로 활용되는 내재적 보상을 소개한다. 이에 더하여, 스케줄링 문제에 특화된 자기지도를 제안한다. 이 후 [98]의 보조 손실(auxiliary loss)을 참고하여 자기지도 손실 함수를 제시한다.

4.3.1 내재적 보상 설계

내재적 보상에는 강화학습의 환경 모델에 대한 추론, 이미지로 표현된 상태를 대상으로 한 선행 연구들에서 다음 상태값 [101]이나 현재 프레임과의 차이점에 관련한 정보[98]가 공통적으로 활용되었다. 첫째로 다음 상태값을 자기지도로 활용하였다. 제안한 FBS를 입력으로 다음 상태에 대한 추론을 수행하기 위한 모형으로는 신경망 기반 auto-encoder를 채택하였다.

두 번째 내재적 보상은 다음 보상값에 대한 예측으로, 모델 기반 강화학습에서 보상 추론 모델을 학습하는 것과 동일하게 설정하였다. 기존 연구 [98, 94]는 보상값을 그대로 활용하는 대신에 클래스로 구분한 후 분류 과제로 자기지도를 수행하였다. 그러나 본 연구에서는 직접적으로 보상값을 정확하게 추론하는 것을 목표로 하였다. 이는 할인율이 0으로 설정한 강화학습을 동시에 수행하는 걸로도 해석가능하다.

세 번째로 여러 연구들에서 활용된 GVF [102, 95]에서 영감을 얻어 현재 상태의 가치값을 자기지도로 설계하였다. DQN 학습 시에 입력값 s_k에 대하여, s_{k+1}, r_k, $V(s_k)$이 타겟값으로 활용함으로써, 내재적 보상이 자기지도로 역할 한다. s_{k+1}, r_k는 트랜지션에 포함되어 별도의 계산 없이 자기지도로 활용할 수 있다. 반면, $V(s_k)$는 에피소드가 종료될 때마다 저장된 트랜지션을 통해 다음 수식과 같이 계산한다.

$$V(s_k) = \begin{cases} r_k & s_{k+1} \text{ is terminal.} \\ r_k + \gamma V(s_{k+1}) & \text{otherwise.} \end{cases} \tag{4.11}$$

4.3.2 셋업 스케줄링을 위한 선호도 점수 설계

일반적으로, 자기지도는 도메인 지식에 의해 설계된 예측 과업으로 표현 학습 과정을 촉진하고 정규화하는 것을 목표로 한다 [102]. 납기 제약이 있는 스케줄링 문제에서는 앞으로 남은 생산 요구량에 대한 파악이 중요하다. 따라서 규칙 기반 휴리스틱 연구들 [109, 51] 은 적절한 셋업 계획 및 잡 할당을 목적으로 우선순위 지수를 제안하였다.

보조 손실로 활용하기 위해, 패밀리 별 셋업 변경이 필요한 정도와 연관된 선호도 점수를 설계한다. 구체적으로 $\Pi(f,k)$로 표시된 s_k에서 패밀리 f의 선호 벡터는 다음과 같이 정의되는 세 가지 선호 점수 Π_1, Π_2, Π_3로 구성된다.

$$\Pi_1(f,k) = \frac{-\sum_{i=1}^{N_M} \sigma_{f,G_i}/N_M}{\bar{s}} \tag{4.12}$$

$$\Pi_2(f,k) = \frac{-\max(\min_{J_j \in \mathcal{W}_k, F_j = f} d_j - \sum_{i=1}^{N_M} p_{i,f}/N_M - kT, 0)}{\bar{p}} \tag{4.13}$$

$$\Pi_3(f,k) = \frac{\sum_{j \in \mathcal{W}_k} I(F_j = f)}{\sum_{i=1}^{N_M} I(G_i) = f)} \tag{4.14}$$

여기서 $\bar{\sigma}$와 $\bar{p}$는 각각 평균 셋업 및 작업 시간이다. Π_1와 Π_2는 각각 BATCS [33]에서 제안된 셋업 시간과 슬랙 시간 항을 가리킨다. 단, BATCS에서 배치 단위로 셋업 시간과 작업 시간을 대입하는 대신에, 제안 기법에서는 패밀리 단위로 전체 설비에 대한 셋업 시간과 작업 시간의 기대값을 산출하도록 변경하였다. Π_3은 설비용량 지수(machine

capacity index)를 나타내며, $F_j = f$인 대기 중 잡의 수를 셋업 상태가 f인 설비 수로 나누어 계산한다.

수식 4.12와 4.13을 설계한 이유는 본 논문에서 다루는 주요 제약인 셋업과 납기를 반영하기 위함이다. 특히 패밀리 셋업이 있는 문제에서 효과가 입증되었으며, 상태 관찰과 함께 쉽게 계산할 수 있다는 장점을 가져 채택하였다. 수식 4.14는 각 패밀리의 우선순위로 해석가능하다. 설비용량 지수가 높으면 해당 패밀리로의 셋업 변경이 필요한 정도가 높다는 의미고, 지수가 낮으면 해당 패밀리로 더 이상 셋업할 필요가 없음을 의미한다.

4.4 자기지도 기반 DQN 학습

4.4.1 자기지도 손실 함수

소개한 타겟값 $\Pi(f,k)$, r_k, s_{k+1}, $V(s_k)$ 각각에 대해 다음 수식과 같이 네 가지 자기지도 손실값이 계산된다.

$$L(\theta) = \begin{cases} \frac{1}{N_{TR}} \sum_{u=1}^{N_{TR}} \sum_{f=1}^{N_F} |\Pi(f,u) - \Pi_\theta(f,u)| \\ \frac{1}{N_{TR}} \sum_{u=1}^{N_{TR}} |r_u - r_\theta(u)| \\ \frac{1}{N_{TR}} \sum_{u=1}^{N_{TR}} |s_{u+1} - s_\theta(u)| \\ \frac{1}{N_{TR}} \sum_{u=1}^{N_{TR}} \frac{\nu}{2} (V(s_k) - V_\theta(u))^2 \end{cases} \tag{4.15}$$

여기서 Π_θ, r_θ, s_θ, V_θ 각각은 네트워크가 θ로 예측한 선호 벡터, 보상, 다음 상태, 상태 가치 값을 가리킨다. 자기지도 손실을 $L_{SS}(\theta)$로 표기하며, 수식 4.15에 나열된 손실의 합으로 정의된다. [98]의 보조 손실 함수에 채택된 대로 상태 가치값에 대해서는 L2-손실이 활용된 반면, 나머지 세 가지 자기지도에 대해서는 L1-손실이 활용되었다. 따라서 $V(s_k)$에 대한 손실에만 별도로 ν의 비중을 곱하였고, 나머지 자기지도 성분은 동일한 비중으로 합산하였다.

최종적으로, 자기지도를 동반한 학습에 활용되는 손실 함수 $L(\theta)$는 두 손실 함수 $L_Q(\theta)$와 $L_{SS}(\theta)$의 가중 합으로 정의된다.

$$L(\theta) = w \cdot L_Q(\theta) + (1-w) \cdot L_{SS}(\theta) \ \ (0 < w \leq 1) \tag{4.16}$$

여기서 w는 L_{SS}에 대한 L_Q의 가중치를 나타낸다. $w = 0$일 경우 작업이 랜덤하게 실행되기 때문에 w가 0보다 크다는 점에 유의하라.

4.4.2 학습 절차

상태 가치함수 $V(s_k)$에 대한 타겟값은 한 에피소드가 종료된 후에 역산하는 방식으로만 알아낼 수가 있다. 따라서 재현 버퍼를 거치지 않고 최근의 경험에서 학습 데이터를 추출하도록 하였다. 이는 재현 버퍼에서 off-policy 방식으로 N_{TR} 개의 트랜지션을 무작위 추출하는 것에 반해, 최근 에피소드에서 생성된 가변적인 개수의 트랜지션 만을 활용하는 on-policy 방식으로 이해할 수 있다.

Algorithm 3 자기지도 기반 심층강화학습의 학습 절차

Input: Scheduling problem
Output: Q-network

Initialization : Set network Q_θ with random weight θ, target network $Q_{\hat{\theta}}$ with $\hat{\theta} = \theta$, and replay buffer B to size N_B.
for $e = 1, 2, \ldots, N_E$ **do**
    $k \leftarrow 0$
    Initialize F_j, d_j of N_J jobs, and status of N_M machines.
    Observe s_k, and $\mathcal{W}_k$
    while $\mathcal{W}_k \neq \emptyset$ **do**
        With probability ε select a_k randomly from $\mathcal{A}_k$
        otherwise $a_k \leftarrow \arg\max_{a \in \mathcal{A}_k} Q(s_k, a)$
        Get s_{k+1}, $\mathcal{A}_{k+1}$, $r_k, \mathcal{W}_{k+1}$ from **Algorithm 1**
        Store transition (s_k, a_k, r_k, s_{k+1}) in B
        Sample N_{TR} transitions $(s_u, a_u, r_u, s_{u+1}) \in B$
        if s_{k+1} is terminal **then**
            Sample transitions of the current episode e
            Get self-supervision $V(s_u)$ from (4.11)
        end if
        Calculate self-supervision $\Pi(\cdot, u)$ from (4.12)-(4.14)
        Calculate loss L_Q from (4.8)-(4.10)
        Calculate loss L from (4.15)-(4.16)
        Perform a gradient descent step on L w.r.t. θ
        $k \leftarrow k + 1$
    end while
    Synchronize $\hat{\theta}$ to θ at every N_U episodes
end for
return Q-network

알고리듬 3은 자기지도가 포함된 손실 함수로 DQN을 학습하는 절차를 보여주고 있다. 우선 첫 번째 줄에서 11번째 줄, 19번째 줄에서 마지막 줄까지는 4.2.3 절의 알고리듬 2와 동일한 절차이다. 그러나 17번째 줄로 간단하였던 손실 $L(\theta)$을 계산하는 과정이 세분화되어 제시되어 있다(12–18번째 줄). 구체적으로 12번째 줄부터 15번째 줄까지는 s_{k+1}이 최종 상태인 경우에 한하여 상태 가치에 관련된 자기지도를 구한다. 그 다음에는 패밀리 별 셋업 선호도 Π를 계산하고(16번째 줄), 벨만 손실을 계산한다(17번째 줄). 이어서 18번째 줄에서 자기지도를 포함한 손실 함수를 획득한다. 마지막으로 그림 4.6은 네 가지 종류의 자기지도를 함께 표현한 DQN 구조를 나타낸다.

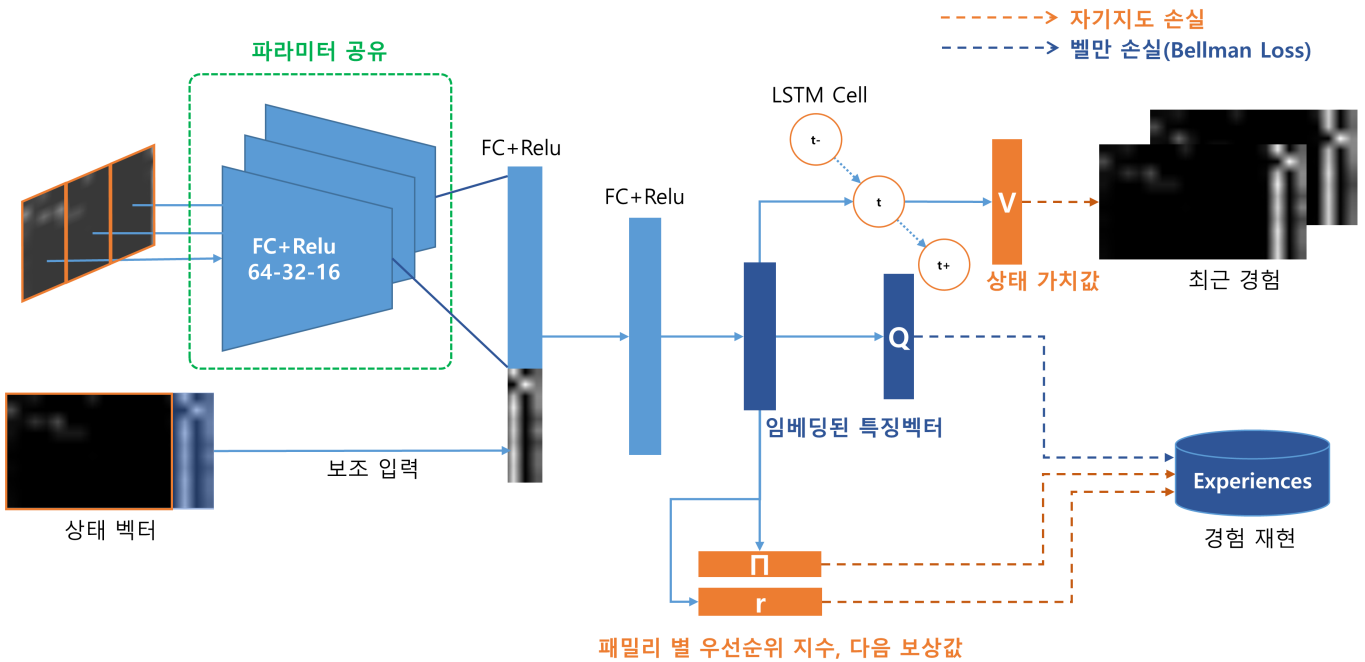


Figure 4.6: 자기지도 기반 DQN 구조

제 5 장 자기지도 기반 심층강화학습을 이용한 잡샵 스케줄링

5.1 스케줄링 프레임워크

제 4장의 제안 기법을 잡샵 스케줄링 문제에 적용하기 위해서, 공정의 수가 여러 개일 때에도 DQN을 학습하는 프레임워크를 고안한다. 먼저 스케줄러로써 DQN이 의사결정하는 범위를 제한하기 위한 병목 공정을 정의하고, 나머지 스케줄링 의사결정을 수행하는 디스패치 규칙을 소개한다. 다음으로 학습 데이터를 생성하고 제조 시스템을 모사하기 위한 시뮬레이터를 설명한다. 마지막으로 DQN 기반 스케줄러의 학습 절차를 제시한다.

5.1.1 병목 공정 정의

Yepes 등 [110]에 따르면 대부분의 multi-stage 제조 시스템 내에 병목 공정이 존재하며, 해당 공정을 병렬설비 스케줄링 문제로 풀이할 수 있다. 위 사실에 착안하여, 병목 공정에 대해서만 DQN을 이용하여 스케줄링 함으로써 제안 기법을 적용할 수 있다.

그림 5.1과 5.2는 반도체 패키징 라인에서 각각 DA와 WB가 병목 공정인 예시 스케줄을 보여주고 있다. 현장에서는 DA 공정과 WB 공정 설비 수의 비율에 따라 병목 공정은 달라질 수 있다. 대상 문제의 목적함수가 투입 목표량 최대화임을 감안할 때, 투입이 이루어지는 DA 공정이 스케줄의 성능을 좌지우지하는 핵심 공정으로 볼 수 있다. 만약 그림 5.2와 같이 WB 공정이 병목인 경우에는, DA 공정의 스케줄링 의사결정과 상관없이 투입량이 결정될 수 있다. 따라서 본 장에서는 DA 공정이 병목이라는 합리적인 가정 하에서, DA 공정에서의 셋업 스케줄을 수립하는 제안 기법을 소개한다.

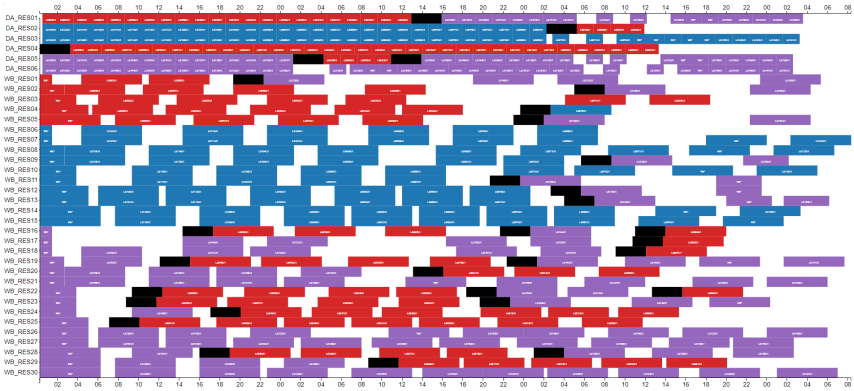


Figure 5.1: DA 공정이 병목인 스케줄 예시

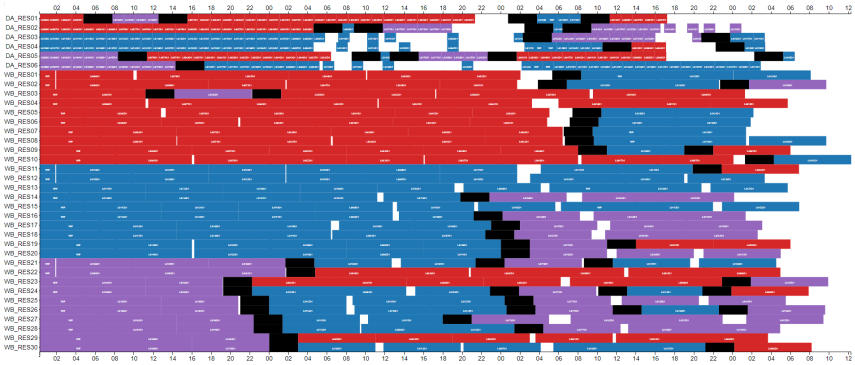


Figure 5.2: WB 공정이 병목인 스케줄 예시

5.1.2 디스패치 규칙

각 공정 단계에서 디스패치 규칙의 역할은, 스케줄러가 결정한 셋업 계획에 상응하는 잡을 설비에 할당하는 것이다. 이를 위하여 특정 패밀리의 여러 잡들 중에서 하나를 선택하기 위한 휴리스틱 규칙이 필요로 한다. 먼저 투입 공정인 DA 공정 휴리스틱에서는 DA 설비에 할당할 수 있는 잡들을 후보로 선정한다. 후보 잡은 카세트 스토커 또는 매거진 스토커에 위치하거나, WB 설비에서 작업 중일 수도 있다. 만약 WB 설비에서 작업 중인 후보 잡이 선택되는 경우, 매거진 스토커에 도달하고 DA 설비로 이동하기까지의 시간 동안 의도적인 유휴(intentional delay)가 발생할 수 있다. 본 장에서

다루는 스케줄링 문제의 목적함수는 투입량의 최대화이므로, 남은 차수가 가장 큰 잡을 선택하도록 하는 MOR (most operation remaining) 휴리스틱을 적용하였다.

비병목공정인 WB 공정의 목표는 막힘없이 DA 공정으로의 재진입을 유도하는 것으로, TAT(total around time)를 최소화할 수 있어야 한다. 따라서 설비가 빌 때마다 가용한 잡들 중에서 셋업 시간과 대기시간을 최소화하는 잡을 선택하도록 구현하였다. 이 때 가용한 잡에는 매거진 스토커에 있는 잡 뿐만 아니라, DA 설비에서 작업 중인 잡까지를 포함함으로써 불필요한 셋업을 방지하고자 하였다. 즉, 미래의 특정 시점 안에 WB 스토커에 진입하게 될 잡 또한 목록에 포함시켰다. 먼저 SST 휴리스틱을 통해 셋업 시간을 최소화하는 잡의 목록을 추려낸 후, 셋업 시간이 동일한 잡 사이에서는 LOR (least operation remaining) 휴리스틱을 적용하였다.

제안한 스케줄링 프레임워크에서, 디스패치 규칙으로 활용하는 휴리스틱의 성능이 최종 스케줄 성능에 중요한 역할을 한다. 만약 동일한 DQN이 DA 공정의 스케줄러로 주어진다면, 디스패치 규칙과 무관하게 동일한 셋업 계획이 생성된다. 그러나 잡을 할당하는 데에 있어서 일반적인 MOR 휴리스틱을 사용하는지, DA 공정에 맞게 개량한 휴리스틱 규칙을 활용하는지가 투입량에 직접적인 영향을 미칠 수 있다. 예를 들어, 일반적인 MOR로는 매거진 스토커에 위치한 잡을 선택하는 상황에서 제안한 DA 휴리스틱 규칙은 카세트 스토커의 잡을 투입할 수 있다. 또는, 제안 규칙은 MOR과 달리 의도적인 유휴를 발생시키면서 아직 매거진 스토커에 위치하지 않은 잡을 할당할 수 있어 성능이 개선될 수 있다.

5.1.3 이산 사건 시뮬레이터

사건(event)이란 시뮬레이터에 변화를 일으키는 순간적인 발생으로 정의된다 [111]. 예를 들어, 대기 중인 잡의 작업 시작과 같은 사건은 잡 또는 설비 상태의 변화를 수반한다. 뿐만 아니라 완료된 잡의 개수를 집계하는 등 시뮬레이터의 내적 변화를 일으키는

사건도 포함된다.

본 연구에서는 사건을 순차적으로 처리하기 위하여 이산 사건 시뮬레이터(discrete event simulator)를 구현하였다. 새로운 에피소드가 시작할때마다 시뮬레이터가 초기화되고, 시뮬레이션 시점 0부터 시간 순서대로 사건을 처리해나간다. 구체적으로는 에이전트에서 시뮬레이터를 호출하는 구조 [112]를 차용하여 step 함수를 구현하였고, step 함수 내에서 시뮬레이션을 구동한다. 이를 통해 시뮬레이션 속도를 빠르게 하고, 타 방법론과의 비교를 수월하도록 하였다.

5.1.4 스케줄러 학습

그림 5.3은 제 5장에서 스케줄러를 학습하는 과정을 도식화하고 있다. DRL 에이전트는 제조 시스템으로부터 상태 벡터를 입력받아 행동 벡터를 출력하는 역할을 담당한다. 환경의 역할을 하는 제조 시스템은 디스패처와 시뮬레이터로 구성되며, 에이전트가 결정한 행동 벡터에 따른 결과인 보상과 다음 상태를 반환한다.

에이전트는 시뮬레이터에서 제공받은 현 시점의 상태 벡터를 DQN에 입력하여, 행동 벡터를 출력한다. 그 다음으로 행동 벡터가 셋업 계획의 형태로 디스패처에게 전달된다. 디스패처는 시뮬레이터와 연속적으로 상호작용하며 DA 및 WB 공정 휴리스틱에 기반을 둔 의사결정을 반환한다. 매 의사결정마다 시뮬레이터는 상태 전이가 발생하는지의 여부를 평가한 후, 상태 전이가 발생할 때까지 디스패처의 의사결정 결과에 따라 시뮬레이션을 진행한다. 상태 전이가 발생하면 시뮬레이터는 상태 및 보상 생성자를 통해 보상과 다음 상태를 계산하고, 결과적으로 (상태, 행동, 보상, 다음 상태)의 쌍이 재현 버퍼에 저장된다. 마지막으로 학습 알고리듬은 정해진 주기마다 DQN의 그라디언트를 업데이트하고, 타겟 네트워크를 동기화한다. 위에 서술한 일련의 과정이 반복되며 DQN 학습이 이루어진다. 이는 알고리듬 1로 간소화된 상태 전이가 디스패처와 시뮬레이터의 상호작용으로 보다 복잡해졌다는 차이를 제외하고는, 4.2.3 절의 학습

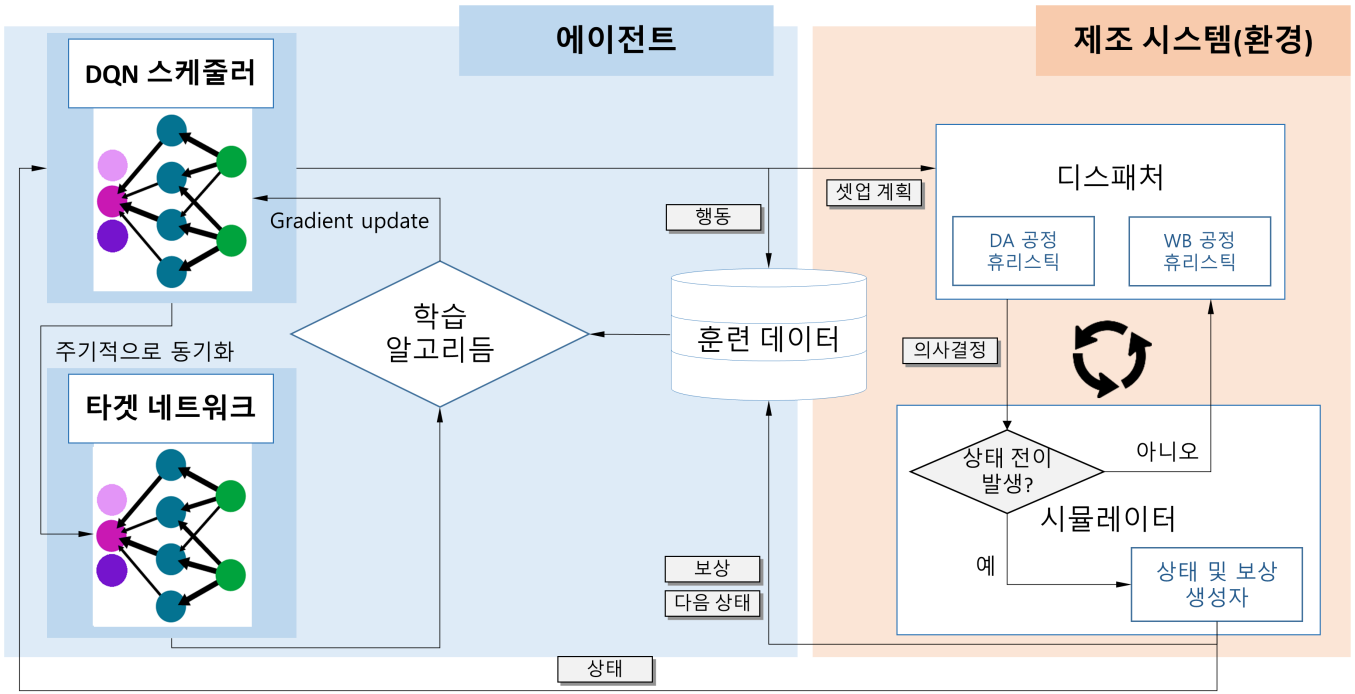


Figure 5.3: 심층강화학습을 이용한 학습 프레임워크 도식

절차와 동일하다.

5.2 투입 정책과 자기지도

잡샵 스케줄링 문제를 위한 자기지도를 제안하기 앞서, DA 공정에 특화된 투입 정책을 소개한다. 투입 정책은 현재 시점의 설비 셋업 상태와 남은 투입량에 기반을 두어, 셋업 변경을 수행할 설비의 셋업 상태 f^R과 셋업 변경이 필요한 잡의 패밀리 f^J를 찾는 규칙으로 정의할 수 있다. 먼저 N_{DA}와 N_{WB}는 각각 DA 공정을 수행할 수 있는 설비의 수와 WB 공정을 수행할 수 있는 설비의 수를 의미한다. 즉, $N_M = N_{DA} + N_{WB}$ 이다. 투입 정책은 다음과 같이 정의되는 패밀리 별 우선순위 지수 $\Pi(f)$를 계산함으로써 동작한다.

$$\Pi(f) = \frac{In(f) - INT_f}{\sum_{i=1}^{N_{DA}} I(G_i = f)} \tag{5.1}$$

여기서 INT_f는 현재시점에서 패밀리 f인 잡이 충족시킨 투입량을 지시하며, 수식 3.12에서 잡 인덱스 j를 $F_j = f$로 한정시켜 계산한 값이다. 따라서 수식 5.1의 분자는 투입 요구량에서 현재까지의 투입량을 뺀 값이고, 분모는 셋업 상태가 f인 설비의 수를 의미한다. 그리고 f^R과 f^J는 다음과 같이 얻어진다.

$$f^R, f^J = \arg\min_f \Pi(f), \arg\max_f \Pi(f) \tag{5.2}$$

따라서 $\Pi(f)$가 가장 낮은 패밀리에서 가장 높은 패밀리로 투입 대상을 변경하는 정책으로 이해할 수 있다.

본 논문의 잡샵 스케줄링 문제에서 자기지도로 앞서 정의한 $\Pi(f)$를 활용하였다. 이러한 자기지도는 4.4.1 절의 설비용량지수 Π_3이 3.2 절의 투입량 목적함수에 맞게 변경된 결과로, 대상 문제에 적합한 사전 지식으로 볼 수 있다.

5.3 MDP 모형 수정

앞 장의 스케줄링 프레임워크에 맞게 DRL 기반 기법을 적용하기 위해서는, 잡샵 스케줄링 문제에 알맞은 상태, 행동, 보상에 대한 정의가 필요하다. 행동 정의 방식에 따라 DQN이 출력한 행동 벡터가 셋업 계획의 형태로 디스패처에게 전달된다. 그리고 상태 및 보상 정의에 맞게 시뮬레이터가 벡터를 생성한다. 4.1 절과 동일하게 s_k로부터 s_{k+1}까지의 시간 간격을 기간 k라 정의하고, 각 기간에서 머무는 시간이 T로 일정하다고 모델링하였다.

5.3.1 행동 정의

투입 정책을 활용하지 않아도 a_k를 (f^R, f^J)의 쌍으로 정의될 수 있다. 이는 4.1.1 절과 마찬가지로 기간 k 동안에 패밀리 f^J의 어떤 잡이 셋업 상태가 f^R인 한 설비에 배치될 것이란 의미를 지닌다. 여기에 더하여, 투입 정책에 기반을 둔 두 가지 방식의 행동 정의를 제안한다. 첫 번째로 binary 행동 정의 방식에서 a_k는 기간 k에서 셋업을 수행할 지, 하지 않을 지만을 결정한다. 그리고 셋업 계획은 투입 정책에 따라 f^R과 f^J를 얻어냄으로써 디스패처에게 전달한다. 두 번째로 family 행동 정의 방식에서 a_k는 f^J만을 결정하고, f^R은 투입 정책에 의해 결정된다. 투입 정책 기반 행동 정의는 행동의 차원을 줄여 네트워크의 크기를 줄이는 장점이 있으며, 특히 binary 행동 정의는 패밀리 수가 변화해도 재학습이 필요 없다.

5.3.2 상태 표현

4.1.2 절의 상태 표현 기법에 기반을 두어 패키징 라인의 DA 공정에 맞는 상태 벡터를 설계하였다. 주요한 차이점을 정리하면 다음과 같다.

- 이차원 행렬과 파라미터 공유 구조를 그대로 사용하지 않고, 일차원으로 이어붙힌

상태 벡터 FBS-1D를 활용하였다.

- $\mathbf{S}_w$: 납기 d_j 대신에 본 문제에서 주어진 투입납기 $\tilde{d}_j$를 기준으로 생성한다. $\mathbf{S}_w$와 동일한 방식으로 $2H_w \cdot N_F$ 차원 벡터로 정의된다.
- $\mathbf{S}_p$, $\mathbf{S}_s$: 앞 장과 동일한 방식으로 각각 $H_p \cdot N_F$, $N_F \cdot N_F$ 차원 벡터로 정의된다.
- $\mathbf{S}_c$, $\mathbf{S}_u$는 활용하지 않는다.
- $\mathbf{S}_a$, $\vec{S}_{inv}$: 동일한 방식으로 $2N_F + 3$ 차원의 벡터를 생성한다.

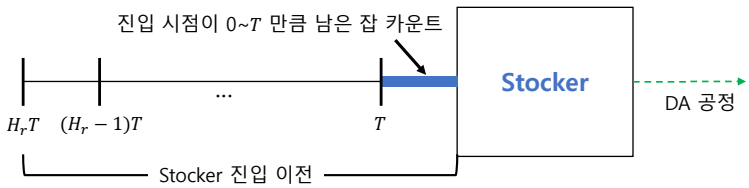


Figure 5.4: 준비 중인 잡 상태 $\mathbf{S}_r$

- $\mathbf{S}_r$: 추가적으로 패키징 라인에서 재진입 되는 잡을 표현하기 위하여, 준비 중인 잡 상태 벡터 $\mathbf{S}_r$을 정의한다. 그림 5.4는 준비 중인 잡의 개수를 셈하는 방식을 묘사하고 있다. 예시 그림에서 T는 기간 간격이자 관찰 간격을 의미한다. 실제로는 관찰 간격 T_r에 맞게, 스토커로의 진입시점이 $[0, T_r]$에서 $[(H_r-1)T, H_rT_r]$ 구간 사이에 속한 잡들을 잡 패밀리 별로 셈하여 H_r 차원의 벡터에 표현한다. 이러한 벡터가 총 N_F개 있으므로, 결과적으로 $\mathbf{S}_r$의 차원은 $H_r \cdot N_F$가 된다.

상태의 차원 총합은 $N_F \times (2H_w + H_p + H_r + N_F + 2) + 3$으로 차원성이 생산 요구량 및 납기와 설비 대수에 독립적이다.

5.3.3 보상 정의

r_k는 기간 k에서 투입된 잡의 개수를 셈하여 부여된다. 단, 이미 투입 요구량을 모두 달성한 후에 투입이 이루어진 경우에는 개수를 셈하지 않는다. INT가 아닌 $WINT$를 목적함수로 하는 경우에는 투입된 잡의 개수가 아닌 가중치 w_f를 곱한 가중투입량 값을 보상으로 부여한다.

제 6 장 실험 및 결과

6.1 병렬설비 스케줄링 문제

먼저 첫 번째 연구 목적으로 생산 요구량과 납기에 가변성이 있는 병렬설비 스케줄링 문제에서 제안 기법의 유효성을 검증하기 위해, 제 4장에서 자기지도를 포함하지 않은 제안 기법으로 실험을 수행하였다.

6.1.1 데이터셋

현실에서 찾아볼 수 있는 병렬설비 스케줄링 문제를 대상으로 성능을 검증하기 위하여, 본 연구에서는 한국의 잉곳형성(ingot growing) 공정을 모사한 8가지 데이터셋을 준비하였다. 표 6.1은 각 데이터셋의 스케줄링 문제에 대해 N_M, N_J, N_F 및 τ로 표시된 납기 tightness를 제시한다. τ를 제외하고, 데이터셋 1과 3은 각각 데이터셋 2와 4와 동일하다. 또한 데이터셋 5-8의 N_M과 N_J는 각각 데이터셋 1-4의 N_M보다 2.5배 크다.

각 데이터셋은 330개의 서로 다른 스케줄링 문제로 구성된다. 이 중 300개의 문제는 제안된 DQN을 학습시키기 위해 활용되며, 나머지 문제들은 학습된 DQN을 검증하는 성능 평가 과정에서 활용된다. 그림 6.1은 각 패밀리에 대한 생산 요구량의 분포를 보여주며, 패밀리 중에서 $P(f)$가 가장 큰 값과 작은 값 사이의 격차가 크다는 특징이 있다. 전체 데이터셋에서, 변동 범위가 가장 적은 패밀리도 최소 37% 이상 생산 요구량에 변동을 주었다. 스케줄링 문제마다 N_J 개 작업의 납기 d_j는 $L(0.5-\tau)$와 $L(1.5-\tau)$ 사이에 균일하게 분포되도록 설정되었다. 여기서 L은 여러 연구 [32, 11, 15]에서 채택된 최대완료시간의 예상 값을 의미한다. 실제 시나리오를 시뮬레이션하기 위해, 스케줄링 시작 시점에서 모든 설비에 한 개씩의 잡이 할당되었고 남은 처리 시간은 균일 분포를

Table 6.1: 데이터셋 특징

번호	N_M	N_J	N_F	τ
1	20	420	10	0.4
2	20	420	10	0.5
3	20	420	7	0.4
4	20	420	7	0.5
5	50	1050	10	0.4
6	50	1050	10	0.5
7	50	1050	7	0.4
8	50	1050	7	0.5

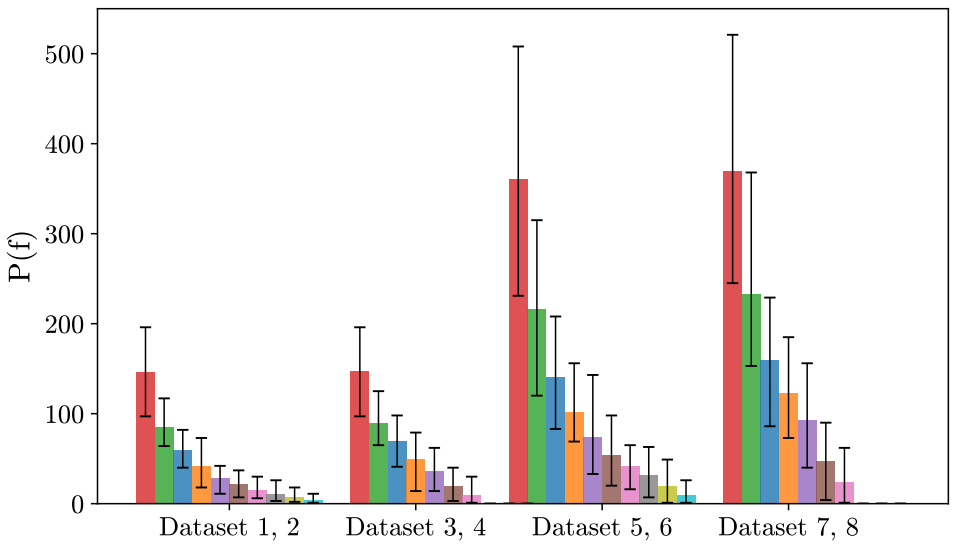


Figure 6.1: 8가지 데이터셋에서의 생산 요구량의 변동 범위

따르도록 생성하였다.

6.1.2 실험 세팅

실험에는 126GB 메모리를 가진 Xeon E5 2.2 GHz PC가 사용되었다. DRL 기반 기법을 도입할 때, 적절한 하이퍼파라미터 값을 선택하는 것이 DNN 성능에 중요한 역할을 한다. 하지만 하이퍼파라미터의 검색 공간이 넓어서 최적 값을 결정하기가 어려운 까닭에, 무작위 탐색(random search) [113]을 통해 최고 성능을 달성한 값을 찾아내었다.

그라디언트 하강법 단계(gradient descent step)를 수행하는 데에는 학습률이 2.5×10^{-3}로 설정된 RMSProp 옵티마이저(optimizer) [114]를 채택했다. ε-greedy 정책에서 ε의 초기 값은 0.2로 설정된다. 이 후 e가 $0.9 \times N_E$에 도달할 때까지 선형적으로 감소하여 값이 0이 된다. 또한 N_B, N_U, N_{TR} 및 γ는 각각 10^5, 50, 64 및 1로 설정된다. 제안된 DQN은 생산 요구량, 설비 수 및 납기가 학습 문제와 다른 새로운 스케줄링 문제를 풀 수 있지만, 보통 실제 제조 시스템에서 발생할 수 있는 극단적인 변화에 대처하기 위해 24시간마다 재교육한다. 이를 달성하기 위하여, 데이터셋 1–4에서는 $N_E = 10^5$, 데이터셋 5–8에서는 $N_E = 1.5 \times 10^4$으로 각각 설정하였다. 각 공유 블록은 첫 번째, 두 번째, 세 번째 계층의 노드 수가 각각 64, 32, 16인 3개의 은닉층으로 구성된다. 각 데이터셋에 대해, e가 각각 $0.91 \times N_E$, $0.92 \times N_E$, ..., $0.99 \times N_E$ 및 N_E와 같을 때 학습된 DQN이 저장된 후, 성능 비교에 사용되었다.

그림 6.2(a)–(c)는 각각 H_w, H_p, H_c를 변경하였을 때의 데이터셋 1의 평균 지연시간(mean tardiness)을 보여준다. 여기서 평균 지연시간은 TT를 N_J로 나누어 계산한 값이다. H_w와 H_p 값은 각각 수식 4.2와 4.3에 설명된 범위 안에서 선택하였다. 또한 4.1.2절에 설명된 정의를 기반으로 H_c의 경우 1–20의 범위에서 무작위 탐색을 수행하였다.

그림 6.2에서 TT는 각각 H_w, H_p, H_c가 6, 5, 4에 도달할 때까지 점차 감소하는 경향이 있다. 구체적으로, $H_w = 0$, $H_p = 0$, $H_c = 1$의 결과와 비교하여 TT는 각각 38%,

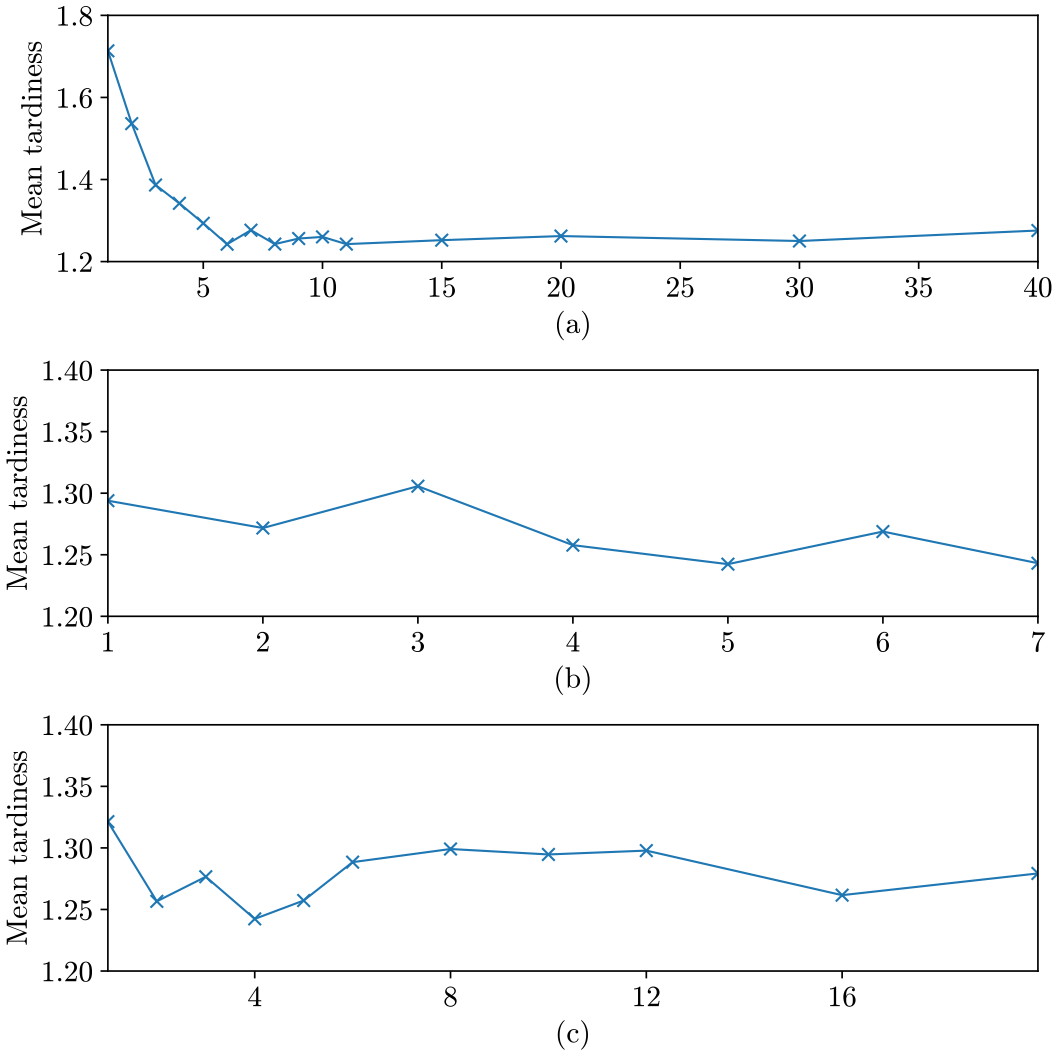


Figure 6.2: H_w, H_p, H_c의 변화에 따른 데이터셋 1에서의 평균 지연시간 결과

10%, 10% 감소하였다. 위의 관찰에 따르면, 제안된 상태 설계는 지연시간을 최소화하는 데 효과적인 것으로 보인다. 반면에, 최대한으로 감소한 이후로는 더 이상의 개선이 관찰되지 않았다. 이는 세부 정보가 제공되었음에도 불구하고 상태 차원의 증가에 따른 학습 복잡성 증가에 기인하는 걸로 보인다. 특히, 납기가 현재 시점과 먼 대기 잡은 에이전트 학습을 위해 구분할 필요가 없음으로 파악된다. 그림 6.2(b), (c)에서와 같이,

H_p 및 H_c를 변화시킴으로써 달성한 성능 향상은 다양한 H_w의 변화에 비해 상대적으로 유의성이 적었다. 대부분의 기간에서 대기 중인 잡이 작업 중인 잡보다 많으므로 $\mathbf{S}_w$에 $\mathbf{S}_p$ 및 $\mathbf{S}_c$보다 많은 관측치가 포함되어 이러한 차이가 발생하였다고 추정된다. 결과적으로, 이후 모든 실험은 H_w, H_p, H_c의 최적 값인 각각 6, 5, 4로 수행하였다.

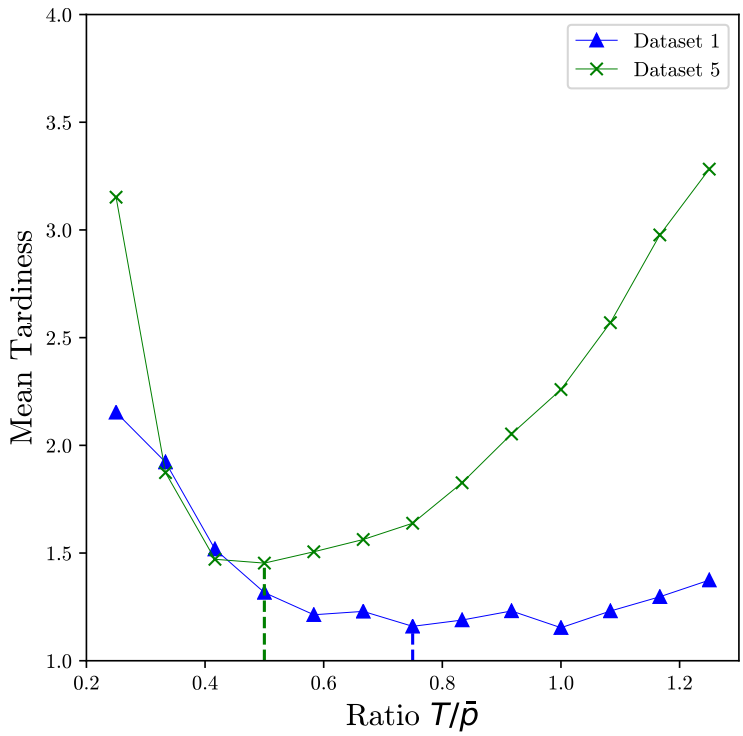


Figure 6.3: 기간 간격에 따른 데이터셋 1, 5에서의 스케줄링 성능 비교

다음으로는 상태 전이가 이루어지는 시간 간격인 하이퍼파라미터 T를 결정하기 위해 grid search 실험을 수행하였다. 제안 기법은 매 기간마다 한 번 셋업 의사결정을 수행할 수 있으므로, 설비의 수가 늘어날수록 이웃한 셋업 간의 시간 간격이 더 짧아질 필요가 있다. 따라서 데이터셋 1과 5를 대표로 선정하여, 스케줄링 문제 규모별로 적절한 T를 찾았다.

그림 6.3은 데이터셋 1과 5에서 평균 지연시간 결과(시간)를 나타낸다. 가로축은

평균 작업 시간 $\bar{p}$에 대비한 T의 비율을 나타낸다. 두 데이터셋에서 모두 T가 가장 짧은 $\frac{1}{4}\bar{p}$일 때에 TT가 가장 길었다. 이는 의사결정의 횟수가 늘어나는 만큼 해 공간의 크기가 넓어져 좋은 스케줄을 탐색하기가 어렵기 때문으로 추정된다. 반면 T가 길어짐에 따라 성능은 점점 개선되어 데이터셋 1에서는 비율이 $\frac{1}{2}$일 때, 데이터셋 5에서는 비율이 $\frac{3}{4}$일 때 TT는 최적 값을 기록하였다. 이후 T가 $\frac{5}{4}\bar{p}$로 계속 증가함에 따라 성능 하락이 관찰되었다. 이는 가능한 셋업의 수가 지나치게 제한됨에 따라 해 공간에 좋은 스케줄이 사라지기 때문으로 보인다. 따라서 데이터셋 1–4에서는 $T = \frac{3}{4}\bar{p}$으로, 데이터셋 5–8에서는 $T = \frac{1}{2}\bar{p}$로 설정하여 이후 실험을 진행하였다.

6.1.3 지연시간 총합 성능 비교

제안 기법의 유효성을 검증하기 위해 두 가지 RL 기반 기법인 two-phase DQN 기법(TPDQN) [58]과 선형 기저 함수를 활용한 QL 기법(LBF-Q) [62]과 함께 메타휴리스틱 IG 기법 [15]과 성능을 비교하였다. LBF-Q 및 TPDQN의 경우 6.1.2 절에 설명된 대로 10개의 모델을 저장한 후 그들 중 가장 좋은 성능을 비교하였다. 나머지 하이퍼파라미터는 각각 [58] 및 [62]의 하이퍼파라미터와 동일하다. IG를 시뮬레이션할 때에, 초기해와 매개 변수는 [15]와 동일하게 설정하였다. 즉 잡을 납기 순으로 정렬한 후, 설비가 빌 때마다 잡의 지연시간 증가량이 최소화되는 잡을 할당하여 설비 별 잡의 시퀀스로 초기 해를 생성하였다. 이후 한 번의 회차(iteration)에서 근접 탐색을 수행하는 횟수 k_{max}와 이웃한 해의 가짓수와 변경할 잡의 비율 매개 변수 α는 각각 500, 6, 0.15로 동일하게 활용하였다. 그러나 종료 조건은 스케줄링이 현장의 제조 시스템에서 보통 매 시간 단위로 수행되는 관계로 [17], 구동 1시간 후에 종료되도록 설정하였다.

추가로 제안 기법을 네 가지 규칙 기반 기법, 즉 BATCS [33], shortest setup with earliest due date (SSTEDD), least slack remaining (LSR) [16], COVERT [109]과 비교했다. LSR과 COVERT는 납기 관련 목적함수를 최적화하는 데에 널리 채택되는

반면 BATCS는 패밀리 셋업이 있는 스케줄링 문제를 풀 때 효과적이다. SSTEDD는 먼저 설정 시간이 가장 짧은 작업을 선택하고 그들 중 납기가 가장 빠른 작업을 결정하는 기법으로 고안되었다.

표 6.2는 각 방법론들의 평균 지연시간 결과(시간)를 제시하고 있다. 데이터셋 별로 가장 퍼포먼스가 좋은 방법론은 굵은 글씨로 표기하였다. 규칙 기반 기법 중에서는 SSTEDD가 모든 데이터셋에서 타 기법들을 능가하였다. 한편 LSR과 COVERT는 다른 규칙 기반 기법들과 대비하여 최상의 경우 3.2배, 최악의 경우 11.3배 더 긴 TT를 산출했다. 따라서 고려된 스케줄링 문제에서 지연시간 최소화를 위해서는 순서 의존적 패밀리 셋업을 다루는 게 특히 중요하다고 해석할 수 있다. 반면 LBF-Q 및 TPDQN은 모든 데이터셋에 대해 SSTEDD보다도 TT가 긴 것이 관찰되었다. 이는 두 가지 RL 기법의 상태 및 행동 표현 과정에서 패밀리 셋업을 수용하지 못했기 때문일 수 있다. IG의 성능은 모든 데이터셋에서 규칙 기반 및 다른 RL 기반 기법의 성능보다 우수했지만, IG가 달성한 TT는 제안 기법의 성능보다 13% – 55% 더 길었다. 이러한 결과를 바탕으로 제안 기법은 학습 때와 생산 요구량 및 납기가 변화해도 스케줄링 문제를 풀이하는 데 효과적인 것으로 보인다.

제안 기법의 우수성은 IG를 비롯한 메타휴리스틱 기법과 마찬가지로 배치형성을 유도함으로써 패밀리 셋업을 반영한 덕분으로 보인다. 주기적인 셋업으로 행동을 설계하고, 그에 상응하는 상태 전이 과정을 통해 학습 과정에서 탐색되는 스케줄을 셋업 횟수가 적도록 제한하였다. 또한 기간 간격을 일정하게 함으로써 시간적으로 이웃한 두 상태의 연관성을 파악하기 용이했을 수 있다. 추가적인 장점으로, [52]의 동시 셋업 작업자 수와 같은 현실적인 제약을 지키는 데에도 유리할 것으로 예상된다.

표 6.3은 제안 기법, TPDQN, LBF-Q, IG 및 SSTEDD에 의해 소요된 평균 연산 시간을 제시한다. 규칙 기반 기법 중에는 평균 연산 시간이 가장 짧은 SSTEDD의 결과

Table 6.2: 제안된 기법과 다른 방법론들의 평균 지연시간 결과

번호	BATCS	SSTEDD	LSR	COVERT	IG	LBF-Q	TPDQN	Ours
1	9.716	8.026	78.116	28.105	1.744	9.079	26.849	**1.243**
2	10.660	9.013	80.516	29.952	2.867	11.339	29.265	**2.490**
3	9.339	7.236	77.589	24.218	1.585	10.191	27.021	**1.196**
4	10.300	8.242	79.989	25.967	2.719	11.340	27.916	**2.404**
5	8.625	7.942	78.925	29.056	2.440	8.763	29.533	**1.796**
6	9.635	8.887	81.325	31.181	3.520	9.527	32.557	**2.934**
7	8.151	7.040	78.006	23.701	2.440	7.322	30.752	**1.572**
8	9.178	8.027	80.406	25.445	3.669	10.486	24.402	**2.724**

Table 6.3: SSTEDD, IG, LBF-Q, TPDQN 및 제안 기법의 연산 시간 결과(초)

번호	SSTEDD	IG	LBF-Q	TPDQN	Ours
1	0.366	3600	12.428	37.845	4.331
2	0.370	3600	12.930	40.360	4.450
3	0.369	3600	12.270	30.870	3.260
4	0.354	3600	13.010	32.160	3.420
5	2.238	3600	65.417	238.633	17.603
6	2.347	3600	73.210	248.780	15.640
7	2.189	3600	70.000	198.620	14.570
8	2.069	3600	73.020	194.020	13.780

만 제시하였다. 모든 데이터셋에서 제안 기법보다 LBF-Q 및 TPDQN의 연산 시간이 더 오래 소요되었다. 이는 스케줄 수립을 위한 Q-값 계산 횟수가 제안 기법에서는 총 기간의 수이고, LBF-Q 및 TPDQN에서는 총 잡의 수 N_J이기 때문이다. 또한 데이터셋 1–4에 대한 데이터셋 5–8의 평균 연산 시간 비율은 LBF-Q의 경우 5.56, TPDQN의 경우 6.23, 제안 기법의 경우 3.98이었다. 위의 관찰은 제안 기법의 DQN 파라미터의 수가 LBF-Q 및 TPDQN와는 다르게 N_M과 N_J에 독립적이기 때문이라고 할 수 있다.

가장 빠른 규칙 기반 기법과 비교하여 제안 기법의 연산 시간은 데이터셋 1–4의 경우 8.8배에서 12배, 데이터셋 5–8의 경우 6.7배에서 7.9배 증가했다. 그럼에도 불구하고, 제안 기법은 모든 데이터셋에 대해 20초 미만의 시간 만에 스케줄을 수립한 결과를 보였다. 이는 제안 기법이 병렬설비가 있는 실제 제조 시스템에서 연산 시간 측면에서의 적용 가능성을 시사한다.

작업 시간과 셋업 시간이 모두 확률적(stochastic)으로 결정되는 상황에서 제안 기법의 성능을 조사하기 위해 추가적인 성능 비교 실험을 수행했다. 구체적으로, $p_{i,j}$와 $\sigma_{f,g}$ 모두 사전에 알려져 있지 않았으며 각각 $[0.8p_{i,j}, 1.2p_{i,j}]$와 $[0.8\sigma_{f,g}, 1.2\sigma_{f,g}]$ 사이에 균일하게 분포되도록 설정되었다. 평가 스케줄링 문제가 동일하더라도 실험할 때마다 결과가 다르기 때문에, 서로 다른 무작위 시드(random seed)를 생성하여 30번 평가하였다.

Table 6.4: 작업 시간 및 셋업 시간이 확률적일 때 SSTEDD, IG, LBF-Q, 제안기법의 평균 지연시간 결과

번호	SSTEDD	IG	LBF-Q	Ours
1	8.013 ± 0.130	1.829 ± 0.118	9.014 ± 0.514	1.235 ± 0.082
5	7.929 ± 0.086	2.514 ± 0.074	8.816 ± 0.276	1.764 ± 0.078

표 6.4는 SSTEDD, IG, LBF-Q, 제안 기법의 평균 지연시간 평균과 표준 편차를 보여주고 있다. 각각 강화학습 기반 기법과 규칙 기반 기 법중에서는 가장 성능이 좋았던 LBF-Q와 SSTEDD만을 대표로 비교하였고, 작은 규모와 큰 규모를 각기 대표하는 데이터셋 1과 5에서 실험을 수행하였다. 전체 비교군 중에서 제안 기법이 TT의 평균과 표준편차 모두 가장 낮았다. 게다가 작업 시간과 셋업 시간이 고정된 표 6.2의 결과와 비교해서도 하락폭이 가장 낮았다. 이러한 결과로 미루어 보건대, 제안 기법은 작업 시간과 셋업 시간의 stochasticity에도 강건한 것으로 보인다. 이는 일정한 기간 간격으로 셋업만 의사결정하는 제안기법의 행동 정의 방식의 이점으로 보이며, LBF-Q의 표준편차가 크단 사실이 이를 뒷받침한다.

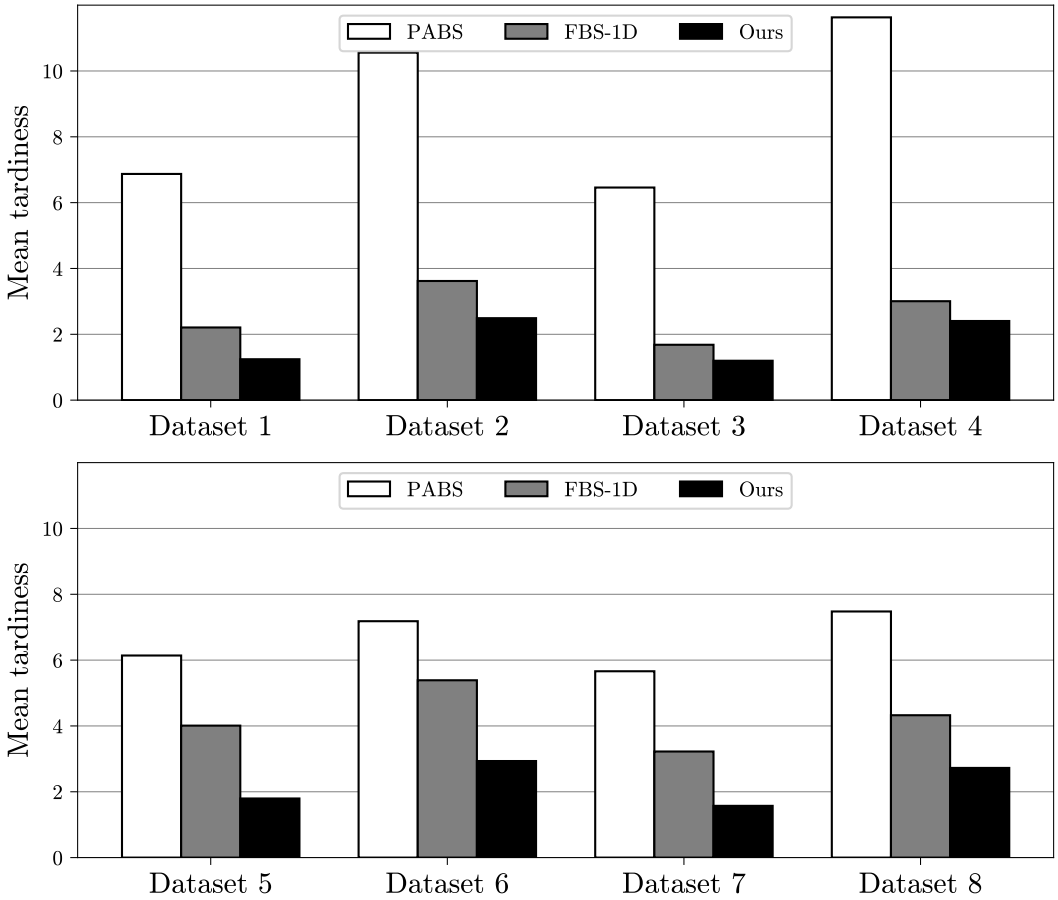


Figure 6.4: 모든 데이터셋에 대한 PABS, FBS-1D 및 제안 기법의 평균 지연시간 결과

6.1.4 상태 표현 방식에 따른 성능 비교

본 논문에서는 제안한 상태 표현과 파라미터 공유 구조의 효과를 추가로 분석하였다. 상태 표현 및 파라미터 공유로 달성된 성능 개선만을 추려내기 위해 제안 기법을 수정한 두 개의 베이스라인 기법과 비교했다. 먼저 [62]에서 제안된 제품 속성 기반 상태 표현(PABS, production attributes based state)을 채택했다. 다음으로, 4.1.2 절에 제안된 패밀리 기반 상태 표현의 $\mathbf{S}_w$, $\mathbf{S}_p$, $\mathbf{S}_c$, $\mathbf{S}_s$, $\mathbf{S}_u$, $\mathbf{S}_a$ 및 $\vec{S}_f$를 평탄화한 후 결합하여 형성한 1차원 벡터(FBS-1D)를 활용했다. PABS 및 FBS-1D의 경우 노드 수가 6.1.2 절

에서 언급했듯 3개의 은닉층으로 구성된 완전연결 신경망을 활용했다. 이 때 파라미터 공유를 배제하기 위해 상태 벡터를 단일 블록이 수신한다는 점을 제외하고 나머지 세부 사항은 제안 기법과 동일하다. 결과적으로 PABS와 FBS-1D는 상태 표현을 제외하고 동일하도록 설계되었다.

그림 6.4는 데이터셋 1-8에서 PABS, FBS-1D 및 제안 기법의 평균 지연시간 결과(시간)를 강조한다. 모든 데이터셋에 대해 FBS-1D는 TT 측면에서 PABS를 일관되게 능가했다. 구체적으로, FBS-1D가 달성한 TT는 데이터셋 1–4에서는 PABS보다 70%, 데이터셋 5–8의 경우 34% 낮았으며, 이는 제안된 상태 표현의 우수성을 보여준다. 또한, FBS-1D와 비교하여, 제안 기법의 성능은 데이터셋 1–4의 경우 각각 30%, 데이터 세트 5–8의 경우 47% 더 우수했다. 위 관찰에 따르면, 파라미터 공유는 잡과 설비 수가 큰 스케줄링 문제를 해결하는 데 더 효율적이었다. 파라미터 공유로 인한 학습 효율의 증가 효과는 상태와 행동 차원이 같더라도 학습해야 할 연관관계가 더 복잡한 스케줄링 문제에서 더욱 뚜렷한 것으로 보인다.

6.2 잡샵 스케줄링 문제

두 번째 연구 목적인 잡샵 스케줄링 문제 적용을 위하여, 제 5장의 제안 기법을 생산 요구량, 납기, 초기 설비 상태가 변화하는 데이터셋에서 검증하였다.

6.2.1 데이터셋

표 6.5는 반도체 패키징 라인을 모사한 6가지 데이터셋의 특징을 요약하고 있다. 각 데이터셋은 설비 수와 잡 타입의 구성이 다르다는 차이점이 있다. 데이터셋 1–2는 설비 수 6개, 잡의 수는 각각 155, 156개로 작은 규모인 반면, 데이터셋 3–4는 설비 수가 30대로 큰 규모이다. 데이터셋 3은 데이터셋 2에서 설비와 잡을 모두 다섯 배씩 늘려서 생성하였다. 데이터셋 3과 4는 패밀리 수가 6과 9로 다르다. 데이터셋 1–4에서 DA 공정의 설비 수와 WB 공정의 설비 수는 동일하며, DA 공정이 병목공정으로 설정되었다. 데이터셋 5는 DA 공정의 설비 수가 90대, WB 공정이 450대인 현실 규모의 패키징 라인을 모사하고 있으며 학습이 아닌 평가에만 활용하였다.

Table 6.5: 패키징 라인 데이터셋의 특징

번호	N_{DA}	N_{WB}	N_F	N_J	N_O
1	6	6	3	155	560
2	6	6	6	156	578
3	30	30	6	780	2890
4	30	30	9	780	2815
5	90	450	9	2340	8445

표 6.6은 9개 잡 패밀리의 공정 흐름과 작업 시간을 나타낸다. 데이터셋 1에서는 $F_j = 1, 4, 7$, 데이터셋 2 및 3에서는 $F_j = 1, 2, 4, 6, 7, 9$ 을 활용하였다. 작업 시간은 분 단위로 주어지며, 셋업시간은 3시간이다.

Table 6.6: 잡 패밀리 별 특징

F_j	$O_{j,k}$	$A(O_{j,k})$	$p_{j,k}$
1	$(O_{j,1}, O_{j,2})$	(DA,WB)	(6000,4800)
2	$(O_{j,1}, O_{j,2})$	(DA,WB)	(6000,4800)
3	$(O_{j,1}, O_{j,2}, O_{j,3})$	(DA,DA,WB)	(6000,6000,4800)
4	$(O_{j,1}, O_{j,2}, O_{j,3}, O_{j,4})$	(DA,WB,DA,WB)	(6000,4800,6000,4800)
5	$(O_{j,1}, O_{j,2}, O_{j,3}, O_{j,4})$	(DA,WB,DA,WB)	(6000,4800,6000,4800)
6	$(O_{j,1}, O_{j,2}, O_{j,3}, O_{j,4}$,$O_{j,5}, O_{j,6})$	(DA,WB,DA,WB ,DA,WB)	(6000,4800,6000,4800 ,6000,4800)
7	$(O_{j,1}, O_{j,2}, O_{j,3}, O_{j,4}$,$O_{j,5}, O_{j,6})$	(DA,WB,DA,WB ,DA,WB)	(6000,4800,6000,4800 ,6000,4800)
8	$(O_{j,1}, O_{j,2}, O_{j,3}, O_{j,4}$,$O_{j,5}, O_{j,6})$	(DA,DA,WB,DA ,DA,WB)	(3000,3000,4800,3000 ,3000,4800)
9	$(O_{j,1}, O_{j,2}, O_{j,3}, O_{j,4}$,$O_{j,5}, O_{j,6}, O_{j,7}, O_{j,8})$	(DA,WB,DA,WB ,DA,WB,DA,WB)	(6000,4800,6000,4800 ,6000,4800,6000,4800)

스케줄링 시작 시점에서 향후 3일간의 생산 계획이 주어지고, 투입납기는 1일 단위로 주어진다. 따라서 각 스케줄링 문제의 J_j는 24시, 48시, 72시 세 종류의 투입 납기 값을 가질 수 있다. 데이터셋 별로 투입 요구량 $In(f)$의 값은 표 6.7에 제시되어 있다. 모든 데이터셋은 300개의 학습 문제와 30개의 평가 문제로 구성되어있으며, 투입 요구량에 최대 10% 만큼의 변동을 주었다. 현실 상황을 모사하기 위해, 모든 설비 및 각 스토커에 중간 차수의 잡이 WIP으로 할당되어 있고 남은 작업 시간이 결정되어 있다고 가정한다.

6.2.2 실험 세팅

DQN 학습을 위한 하이퍼파라미터는 표 6.8에 정리되어 있다.

Table 6.7: 데이터셋 별 평균 투입요구량

번호	$In(1)$	$In(2)$	$In(3)$	$In(4)$	$In(5)$	$In(6)$	$In(7)$	$In(8)$	$In(9)$
	30			17		13			
1	20			17		12			
	20			11		15			
	15	15		17		4	7		1
2	10	10		17		6	5		2
	10	10		11		10	3		3
	75	75		85		20	35		5
3	50	50		85		30	25		10
	50	50		55		50	15		15
	75	75	3	48	15	10	35	5	5
4	50	50	6	48	35	15	25	5	10
	50	50	6	29	50	25	15	5	15
	225	225	9	144	45	30	105	15	15
5	150	150	18	144	105	45	75	15	30
	150	150	18	87	150	75	45	15	45

Table 6.8: 하이퍼파라미터 세팅

분류	항목	설정값
네트워크	학습률	0.0025
	미니배치 크기	64
	Tau	1
	네트워크 구조	64 – 32 – 16
	손실 함수	Huber Loss
	옵티마이저	RMSProp
학습 데이터	γ	1
	재현 버퍼 크기	1,000,000
	웜업 크기	5000
에이전트	네트워크 갱신 주기	매 의사결정마다
	타겟 네트워크 갱신 주기	10 에피소드
	탐색 방법	ε-greedy 0.2 에서 0 선형 감소

학습률(learning rate)은 그라디언트 값에 곱해지는 신경망의 파라미터 업데이트 가중치이다. 미니배치 크기(mini-batch size)는 신경망을 1회 업데이트할 때 사용하는 트렌지션의 수이다. Tau 값은 온라인 네트워크의 파라미터를 타겟 네트워크에 업데이트할 때 반영하는 비중을 결정한다. 네트워크 구조는 은닉층의 수와 각 층별 노드의 수를 가리킨다. 손실 함수는 신경망 학습 시 사용되는 오차 함수이며, 옵티마이저는 그라디언트 값을 구하는 방식이다. 재현 버퍼 크기는 재현 버퍼의 최대 크기이다. 웜업(warm-up) 크기는 미니배치 추출을 통한 학습을 수행하기 전, 무작위 행동을 통해 일정량의 트랜지션을 저장하는 크기를 의미한다. 평가 절차에 활용되는 온라인 네트워크는 매 의사결정마다 업데이트하는 반면, 타겟 네트워크는 10 에피소드마다 업데이트 한다.

패키징 라인 시뮬레이터에 특화된 세팅으로, 설비가 가용해지기 30분 전에 잡을 할당하는 사건이 발생하도록 하였다. 또한 DA 및 WB 디스패치 규칙에서 가용한 잡을 스토커에 도착하기까지의 시간이 100분 이내인 잡으로 한정하였다. $\mathbf{S}_r$에 관련된 하이퍼파라미터인 T_r은 30분이며, 스토커에 도착하기까지 시간이 120분 이내인 잡만을 관찰하였다. 목적함수 및 보상 값으로 데이터셋 1, 2에서는 INT를 데이터셋 3, 4에서는 $WINT$를 채택하였다.

6.2.3 투입량 성능 비교

비교군 기법으로 네 종류의 규칙 기반 기법과 유연잡샵 문제인 반도체 팹 라인에 활용된 TPDQN [58]을 채택하였다. 보상값으로 납기 편차 대신에 투입량을 부여하였다는 것을 제외하고는 [58]과 동일하게 구현하였다. 규칙 기반 기법에는 셋업이 있는 문제에서 흔히 활용되는 SST와 함께 재진입이 있는 잡샵 문제에 널리 도입된 MOR (maximum operation remaining) 및 LOR (least operation remaining) [115] 기법이 포함된다. 추가적으로 SST로 패밀리를 결정한 후 DA 공정의 디스패치 규칙으로 잡을 할당하는 SSTDA 기법을 비교하였다.

Table 6.9: 데이터셋 1–4에서의 다른 기법들 대비 제안 기법의 투입량 증가율

번호	TPDQN	SST	SSTDA	MOR	LOR
1	105.7%	36.6%	30.1%	315.5%	104.3%
2	137.8%	72.1%	46.0%	419.6%	116.1%
3	73.4%	1.2%	9.4%	104.0%	187.0%
4	129.5%	15.1%	10.4%	90.0%	152.7%

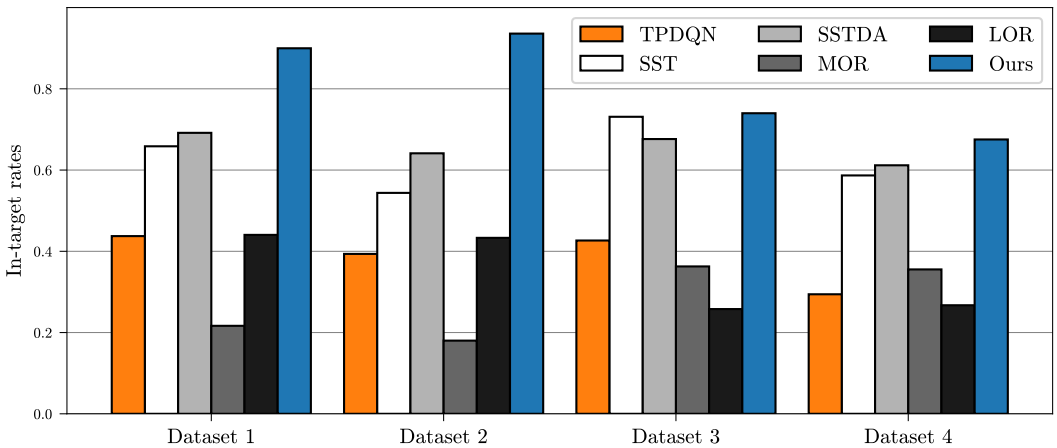


Figure 6.5: 데이터셋 1–4에서 TPDQN, SST, SSTDA, MOR, LOR, 제안 기법의 투입율 결과

표 6.9는 비교 기법들 대비 제안 기법의 투입량 증가율을 나타내고 있다. 모든 학습 데이터셋에서 투입량 증가율이 0% 이상으로 성능을 개선하였음을 알 수 있다. 특히 데이터셋 1, 2에서 큰 폭으로 성능이 개선되었다. 반면, 데이터셋 3에서는 SST에 비해 1.2%만이 개선되어, 좋은 셋업 정책을 학습하지 못하였다. 규칙 기반 기법 중에서는 셋업을 최소화하는 SST가 데이터셋 3에서, 셋업을 최소화하며 투입량 최대화를 위한 디스패치 규칙도 수행하는 SSTDA가 데이터셋 1, 2, 4에서 투입량 측면에서 가장 좋은

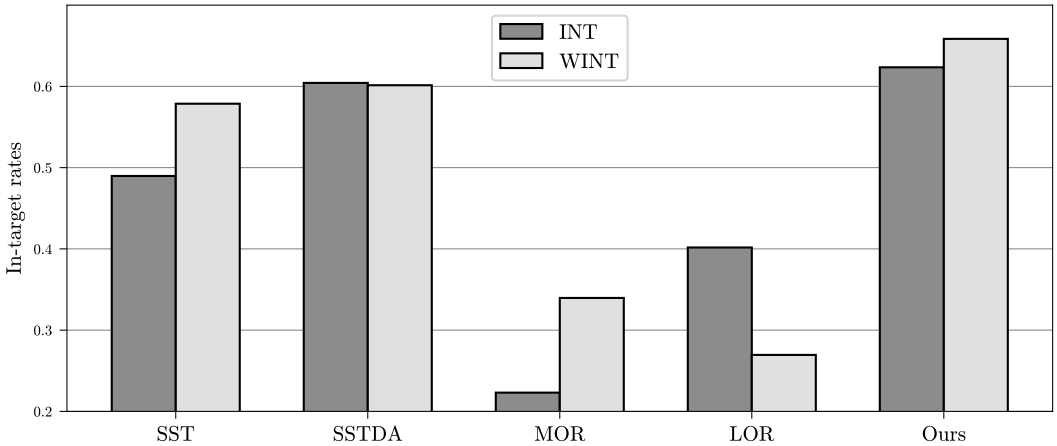


Figure 6.6: 현실 규모 데이터셋에서 SST, SSTDA, MOR, LOR, 제안 기법의 투입율 결과

성능을 보였다. TPDQN의 투입량은 모든 데이터셋에서 셋업을 수행하지 않는 SST보다 낮았다. 이는 TPDQN의 상태 및 행동 표현 방식으로는 투입 납기를 충분히 반영하지 못하였기 때문으로 추측할 수 있다.

그림 6.5는 DRL 기반 기법과 4가지 규칙 기반 기법, 제안 기법의 투입율 결과를 묘사한 결과는 그림 6.5와 같다. 여기서 투입율(in-target rate)은 투입량 또는 가중투입량을 최대값으로 나누어 계산하였다. INT 측면에서는 공정 시퀀스의 길이가 짧은 잡들을 우선 투입하는 LOR이 성능이 더 좋고, $WINT$ 측면에서는 w_f가 높은 공정 시퀀스 길이가 긴 잡들을 투입하는 MOR이 성능이 더 좋음을 확인할 수 있다.

다음으로 그림 6.6은 현실 규모 데이터셋 5에 평가를 수행했을 때의 투입량 결과를 나타낸다. 제안 기법에 의해 산출된 INT와 $WINT$이 모든 기법들 중 가장 높았고, DA 공정에 특화한 규칙 기반 기법 SSTDA가 뒤따랐다. 위 결과에 따르면 생산 요구량과 납기가 새로 주어진 현실 규모 문제에서도 제안 기법이 투입량 측면에서 우수한 성능을 유지하는 것으로 보인다. 이는 보상값으로 $WINT$를 부여함으로써 LOR처럼 맹목적으로 공정 시퀀스가 짧은 잡들을 우선시하는 대신에 잡 패밀리 간의 균형을 맞춘

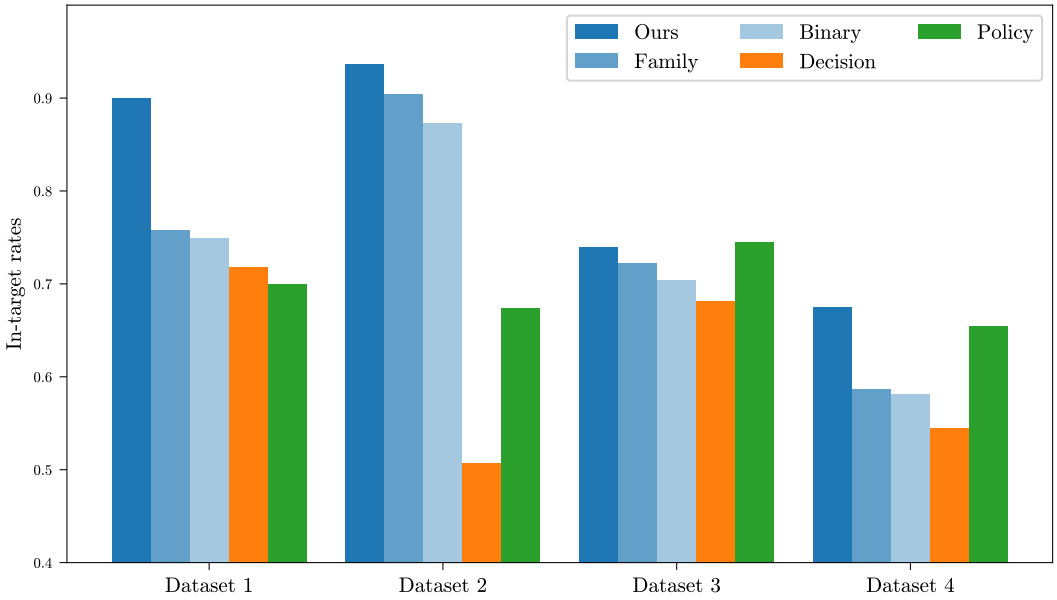


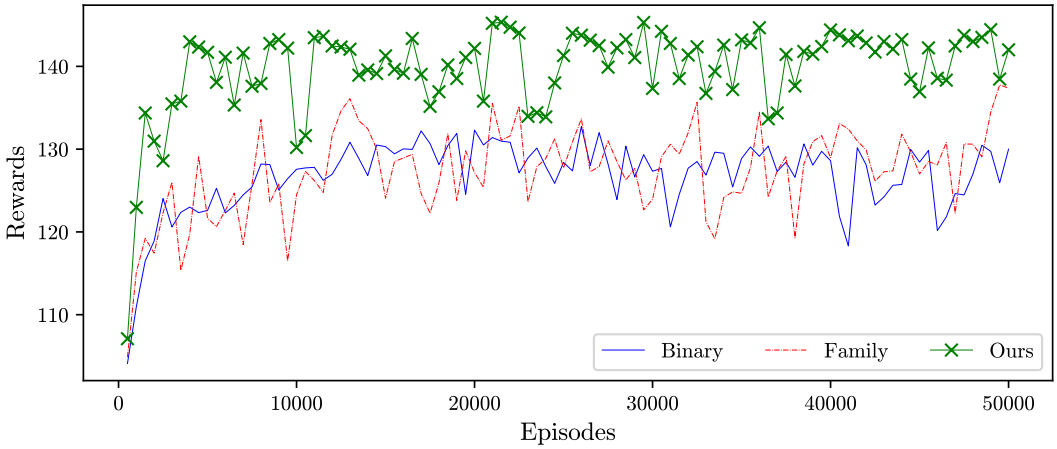
Figure 6.7: 모든 데이터셋에서 행동 정의 방식에 따른 투입율 결과

덕분으로 추정된다.

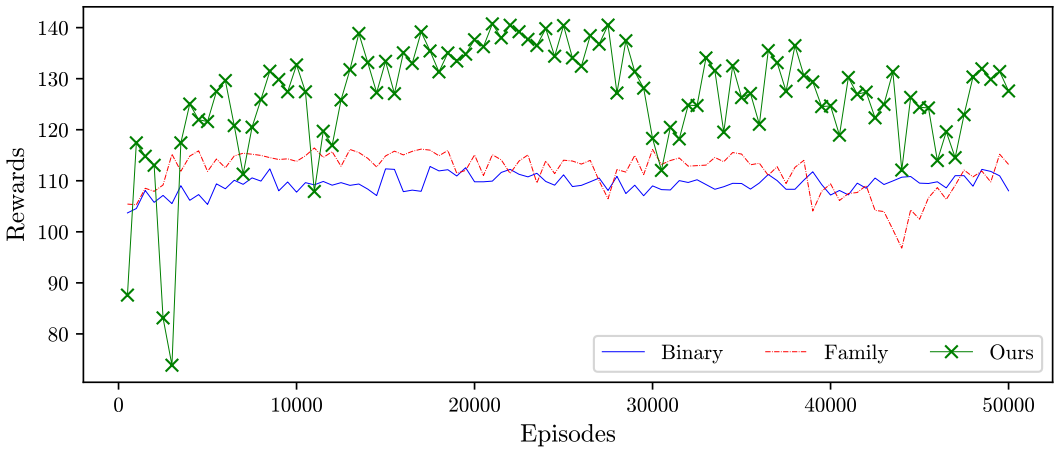
6.2.4 행동 정의 방식에 따른 성능 비교

행동 정의 방식에 따른 투입량 성능에의 영향을 분석하기 위해 추가 실험을 수행하였다. 제안 기법은 5.1.2 절의 투입 정책을 활용하는 방식에 따라 Ours, Family, Binary의 세 종류로 구현될 수 있다. 그리고 투입 정책의 유효성을 확인하기 위해 DQN을 사용하지 않는 투입 정책 기법(Policy)을 추가로 비교하였다. 이에 더하여 행동 정의만 TPDQN과 동일한 방식으로 구현한 제안 기법(Decision)을 구현하였다. 즉, T 시간 간격으로 매 기간마다 패밀리를 선택하지 않고 잡을 배치할 때마다 패밀리를 선택하는 행동 정의를 따랐다.

그림 6.7은 데이터셋 1–4에서 네 가지 행동 정의 방식에 따른 투입율 결과를 보여주고 있다. 모든 데이터셋에서 제안 기법(Ours)의 투입율이 가장 높았으며 Decision 행동



(a) 데이터셋 1

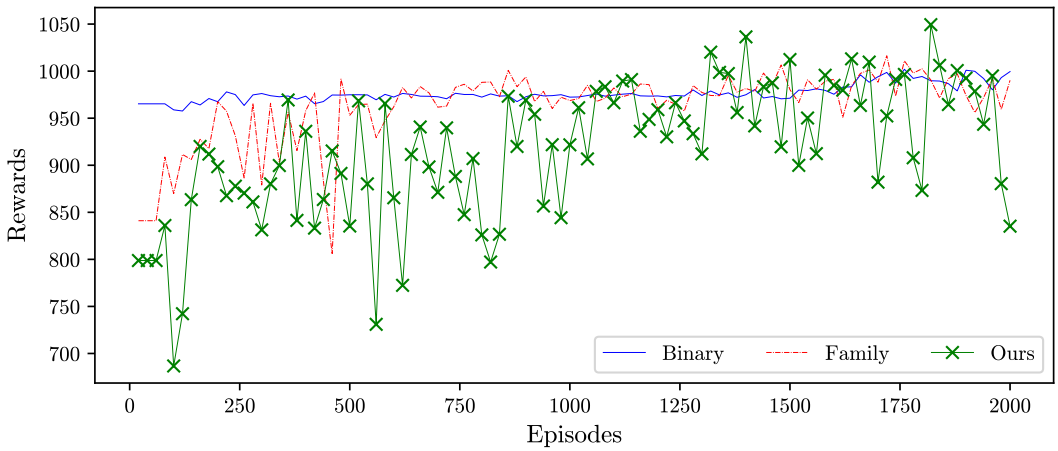


(b) 데이터셋 2

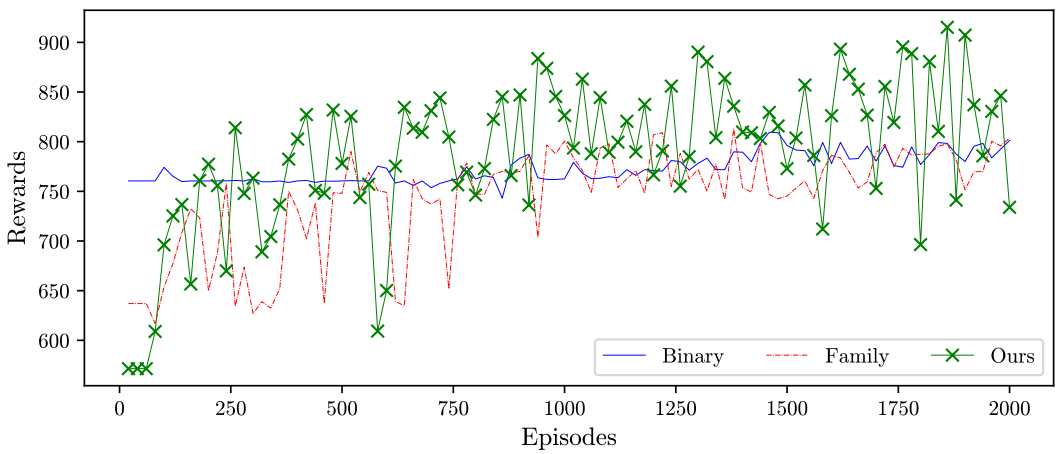
Figure 6.8: 에피소드 증가에 따른 INT 보상값 학습 곡선

정의의 투입율이 가장 낮았다. 이는 잡을 배치할 때마다 패밀리를 선택하는 decision 행동 정의 방식에서 지나치게 해 공간의 크기가 커짐에 따라, 좋은 정책을 학습할 수 없기 때문으로 추측된다. 따라서 제안 기법에서처럼 매 기간 k마다 의사결정하는 방식이 본 연구에서 다루는 스케줄링 문제에 적합하다고 볼 수 있다. 반면 투입 정책이 다른 규칙 기반 기법보다는 더 높은 투입율을 이끌어 낼 수 있음에도, 투입 정책을 활용하지 않는 것이 더 좋은 정책을 학습할 수 있었다. 이는 투입율이 높은 스케줄들로 해 공간의 크

기를 줄이는 이점이 투입율이 낮은 스케줄을 포함하여 자유도를 넓히는 이점보다 크지 않았음을 암시한다.



(a) 데이터셋 3



(b) 데이터셋 4

Figure 6.9: 에피소드 증가에 따른 $WINT$ 보상값 학습 곡선

보다 상세히 학습 양상을 관찰하기위해 그림 6.8은 데이터셋 1, 2에서의 학습 곡선을 드러내고 있다. 처음에는 세 가지 행동 정의 모두 좋은 정책을 찾지 못하였을 뿐 아니라, 그림 6.8(b)에서처럼 제안 기법의 투입율이 가장 낮았다. 그러나 학습이 반복될수록 더 좋은 정책으로 보상값이 우상향하며 그 폭은 제안 기법, Family 행동 정의, Binary 행동

정의 순서로 확인되었다.

그림 6.9는 데이터셋 3, 4에서의 학습 곡선을 보여준다. 학습 초반에 binary 행동 정의가 눈에 띄게 보상값이 높은 것으로 관찰되었다. 그리고 데이터셋 1, 2와 비슷한 양상으로 binary, family 행동 정의에서는 에피소드가 증가해도 $WINT$ 보상값의 상승 폭이 적었다. 그러나 마지막까지도 제안 기법이 확실하게 좋은 정책을 탐색하지는 못한 것으로 보인다. 위 관찰에 따르면 설비와 잡 패밀리의 수가 큰 스케줄링 문제에서는 투입 정책을 활용한 행동 정의가 빠르게 좋은 스케줄을 탐색하는 데에 유리한 것으로 보인다.

6.3 자기지도로 인한 효과

제안 기법의 자기지도로 인한 일반화 성능 개선 효과를 상세히 조사하기 위하여, 제3장의 병렬설비 및 잡샵 스케줄링 문제에 대해 추가적인 기법 내 비교 실험을 수행하였다.

6.3.1 데이터셋

표 6.10은 6.1절의 데이터셋을 확장한 11개 데이터셋을 제시한다. 데이터셋 S1, S2는 설비 수가 20대인 데이터셋이며, L1–L8은 설비 수가 50대인 상대적으로 규모가 큰 데이터셋이다. RI는 한국의 웨이퍼 준비 시설의 실제 규모와 생산 요구량 및 납기를 모사한 평가 데이터셋이다.

Table 6.10: 데이터셋 특징

명칭	N_M	N_J	N_F	τ	η
S1	20	420	10	0.4	2
S2	20	420	7	0.4	2
L1	50	1050	10	0.4	2
L2	50	1050	7	0.4	2
L3	50	1050	10	0.5	2
L4	50	1050	10	0.4	1
L5	50	1050	10	0.4	3
L6	50	1050	10	0.4	2
L7	50	1050	10	0.4	2
L8	50	1050	10	0.4	2
RI	57	1470	10	-	0.5

데이터셋 S2, L2는 각각 데이터셋 S1, L1에서 N_F을 7로 변화시켜 생성하였다. L3

은 L1을 기준으로 $\tau = 0.5$로 조절하였고, L4와 L5는 셋업 severity $\eta = \bar{\sigma}/\bar{p}$을 각각 1과 3으로 조절하였다. 데이터셋 L6–L8은 자기지도로 기인한 일반화 성능 측면의 이점을 확인하기 위해 추가적으로 가변성을 증가시킨 데이터셋이다. 먼저, L6은 $P(f)$의 값을 무작위로 교환하여 학습 데이터에서 개수가 많았던 패밀리가 개수가 적어질 수 있도록 변화시켰다. 다음 L7과 L8는 초기 셋업 상태의 가변성에 변화를 주었다. 구체적으로 L7의 스케줄링 문제는 다음의 3가지 극단적인 초기 셋업상태 중 하나로 설정되었다: 모든 설비가 하나의 패밀리, 두 패밀리가 2:8 비율로 분포, L1의 고정된 초기 셋업상태. L8는 L1의 고정된 초기 셋업 상태에서 최대 20%만큼의 설비의 초기 셋업 상태가 10개 패밀리 중 하나로 무작위 설정된 스케줄링 문제로 구성하였다.

Table 6.11: 데이터셋 RI의 생산 요구량 및 납기 분포

납기	$P(1)$	$P(2)$	$P(3)$	$P(4)$	$P(5)$	$P(6)$	$P(7)$	$P(8)$	$P(9)$	$P(10)$
1주차	5	3	1	5	2	0	10	18	27	596
2주차	13	0	0	0	0	0	0	6	0	81
3주차	85	0	0	0	0	2	0	16	0	84
4주차	213	2	0	0	0	5	0	5	0	291
총계	316	5	1	5	2	7	10	45	27	1052

데이터셋 RI의 생산 요구량 및 납기 분포는 표 6.11에 나타나 있다. 납기는 현실의 1주 단위로 주어져 있으며, 생산 요구량을 토대로 1주가 평균 작업 시간 단위로 $8\bar{p}$에 해당한다고 추정하였다. 현장의 보안 관계상 작업 시간과 셋업 시간의 정확한 값을 알 수 없었기 때문에, 작업 시간과 셋업 시간은 미리 알려져 있지 않으며 $\pm 20\%$의 범위에서 확률적으로 결정된다고 가정하였다. 또한, 대략적으로 파악된 대로 $\eta = 0.5$로 설정하였다. RI는 초기 셋업상태가 L9의 방식으로 무작위 생성된 30개의 평가 스케줄링 문제로 구성되어 있다. 소개한 11개 데이터셋에 더하여 6.2 절의 데이터셋 1–4를

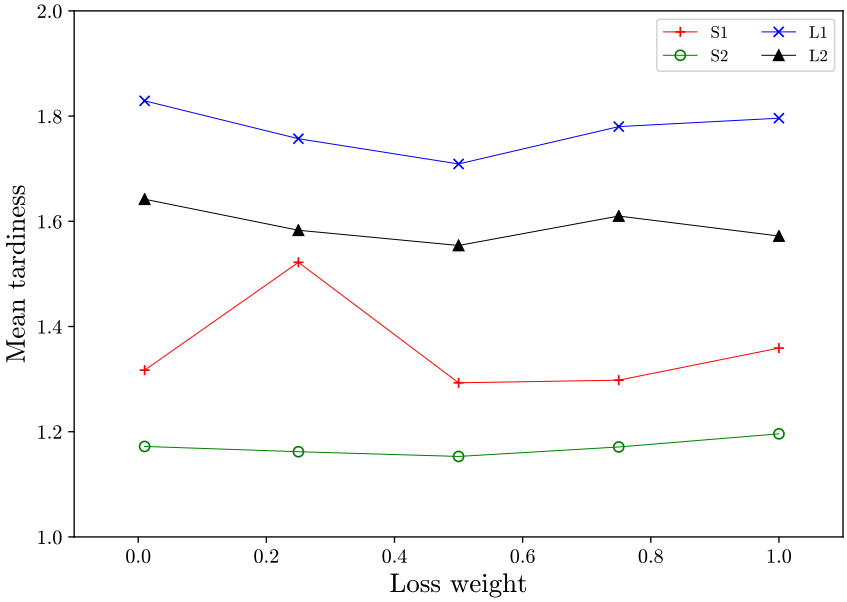


Figure 6.10: L_Q에 대한 L_{SS}의 손실 가중치 w의 변화에 따른제안 기법의 평균 지연시간 결과

데이터셋 P1–P4로 동일하게 실험에 활용하였다.

6.3.2 실험 세팅

자기지도 손실 L_{SS}의 가중치를 결정하기 위해 데이터셋 S1, S2, L1, L2에서 w 값으로 0.01, 0.25, 0.5, 0.75, 1.0의 다섯 개 값으로 변경시키며 실험을 수행하였다. 그 결과 그림 6.10에서와 같이 모든 데이터셋에서 $w = 0.5$로 자기지도 손실을 채택하는 제안 기법이 다른 방식보다 성능이 우수했다. 구체적으로, $w = 0.5$인 제안 기법의 $w = 1$일 때에 비한 개선율은 가장 좋은 경우 L1, S1에서 4.8%, 가장 나쁜 데이터셋 L2에서는 1.1%였다. 비록 개선율이 6.1.4 절에서 상태 표현 방식과 파라미터 공유 구조에 의한 개선율보다는 상대적으로 유의성이 낮게 나왔지만, 자기지도 손실이 패밀리의 수가 더 큰 스케줄링 문제에서 일반화 성능을 향상시키는 것으로 보인다. 이는 상태와 행동의 차원이 커질 때 자기지도로 인한 학습 속도의 향상이 두드러지기 때문으로 생각된다.

이후 실험부터 $w = 0.5$로 고정시켜 실험하였으며, 자기지도 손실 계산을 위해 필요한 ν는 0.01로 설정하였다. 나머지 하이퍼파리미터 및 세팅은 모두 6.1.2 절과 동일하다. 단 데이터셋의 규모가 커서 학습 시간이 오래 걸리는 경우에 한하여 N_E는 임의로 조절하였다.

6.3.3 파라미터 공유 여부에 따른 자기지도의 효과

자기지도로 인한 성능 개선율을 조사하기 위해, 파라미터 공유 여부에 따른 스케줄 성능을 비교하였다. 6.1.4 절에서 병렬설비 스케줄링 문제에 활용된 FBS-1D와 FBS 두 가지 방식에 따른 학습 곡선을 관찰하였다. 학습 곡선은 각 데이터셋의 30개 평가 문제를 학습 과정에서 주기적으로 풀이하여 획득하였다.

FBS-1D

그 다음으로 파라미터 공유 기법을 활용하지 않은 FBS-1D에서, L_{SS}를 구성하는 세 가지 성분인 r_k, $V(s_k)$, $\Pi(f, k)$ 각각의 포함 여부에 따른 자기지도의 효과를 조사하였다.

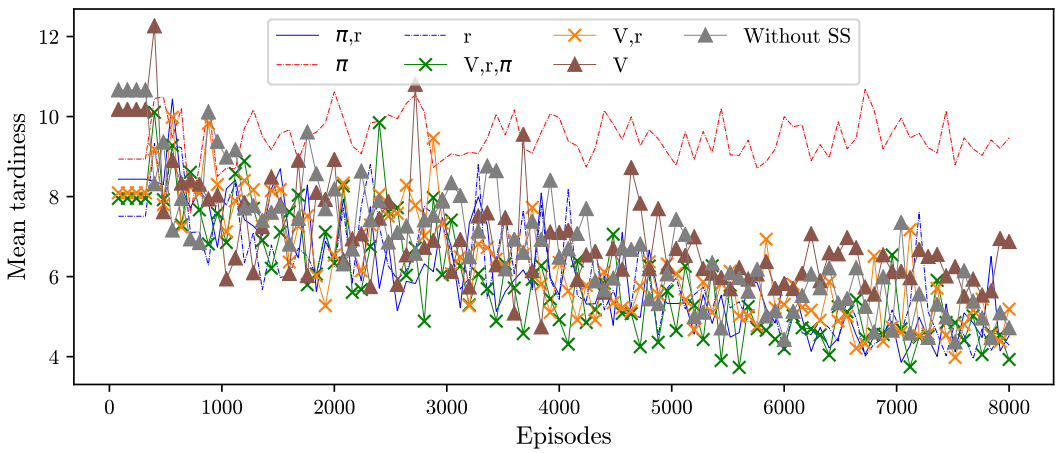


Figure 6.11: 데이터셋 L1에서 자기지도 포함 여부에 따른 평균 지연시간 결과

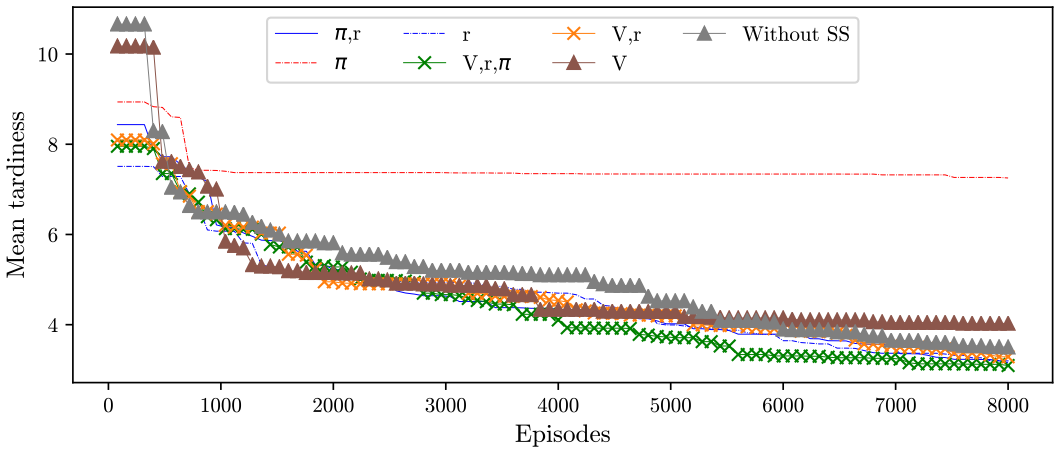


Figure 6.12: 데이터셋 L1에서 자기지도 포함 여부에 따른 평균 지연시간의 최저값 결과

그림 6.11은 자기지도 포함 여부에 따른 평균 지연시간의 학습 곡선을 보여주고 있다. 제안 기법의 평가 문제에서의 결과는 녹색 곱셈 기호로 표시되어 있으며, 7가지 방식 중 TT가 가장 낮았다. 그러나 Π 만을 자기지도에 포함하였을 때에 TT의 개선이 이루어지지 않고 마지막 에피소드까지 이어지는 것을 확인할 수 있다. 이외에는 자기지도를 포함하지 않은 학습곡선과의 뚜렷한 차이가 발생하지 않은 것으로 보인다.

보다 뚜렷하게 학습곡선간의 차이를 관찰하기 위하여, 그림 6.12는 동일한 실험에서 평균 지연시간의 최저값 결과를 보여주고 있다. 즉, 에피소드 0부터 e까지 생성한 스케줄 중에 가장 지연시간이 낮은값만을 표시하였다. 에피소드 3000 부근에서 8000까지 꾸준하게 모든 자기지도를 포함한 제안 기법이 가장 낮은 TT를 기록한 것을 관찰할 수 있다.

FBS

뒤 이어 제안 상태 표현 방식인 FBS에서의 자기지도의 효과와 비교하기 위해 동일한 학습곡선을 관찰하였다. 그림 6.13은 데이터셋 L1에서 평균 지연시간의 최저값

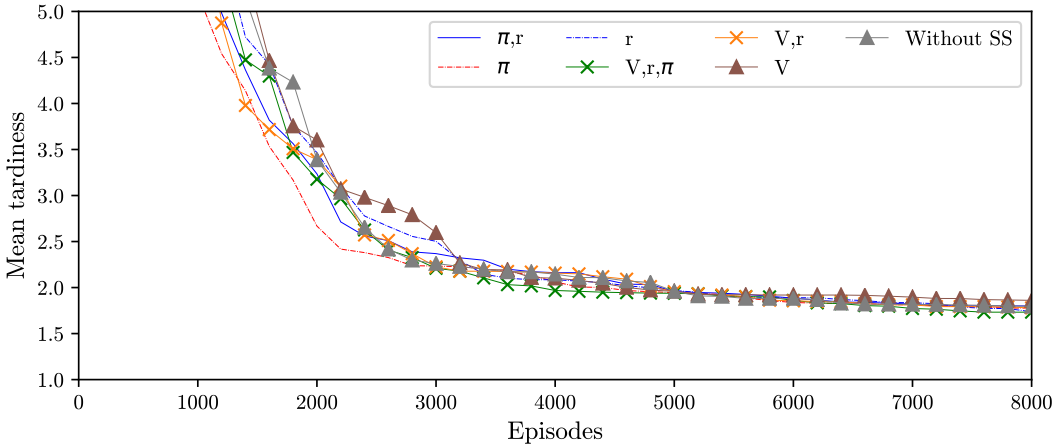


Figure 6.13: 데이터셋 L1에서 자기지도 포함 여부에 따른 평균 지연시간의 최저값 결과

곡선을 그리고 있다. FBS에서는 FBS-1D처럼 자기지도를 포함했을 때의 TT가 특정 에피소드 이후부터 일관되게 낮은 경향이 관찰되지 않았다. 다만 에피소드 4000 부근에서 일시적으로 자기지도로 인한 효과가 관찰되었고, 에피소드 3000 이전에 II 만을 자기지도에 포함하였을 때에 TT가 가장 낮은 상반되는 결과를 보였다. 위 관찰을 통해, FBS에서보다 FBS-1D에서 자기지도의 효과가 더 좋았고 자기지도의 설계 방식은 스케줄 성능에서의 유의미한 차이를 만들어내지 않았다.

이는 파라미터 공유 구조를 채택함으로 생긴 성능 개선 폭 이상으로 자기지도로 인한 효과를 얻기는 어렵기 때문으로 추정된다. 예를 들어 에피소드 8000에서 FBS-1D의 최저 TT는 자기지도가 없을 때 3.50, 포함했을 때 3.09로 11.7%만큼 하락하였다. 반면, FBS에서는 자기지도를 포함하지 않아도 1.63로 53.4%가 감소된 것으로 관찰되었고, 자기지도를 포함하였을 때는 1.61로 그 개선율이 낮았다.

다음으로 생산 요구량의 가변성이 커진 데이터셋 L7에서 학습곡선을 그린 결과가 그림 6.14에 제시되어 있다. 학습이 충분히 진행된 에피소드 4000부터는 7개 곡선간의 차이가 관찰되지 않았다. 그러나 학습 초반에는 자기지도를 포함하지 않았을 때보다 포

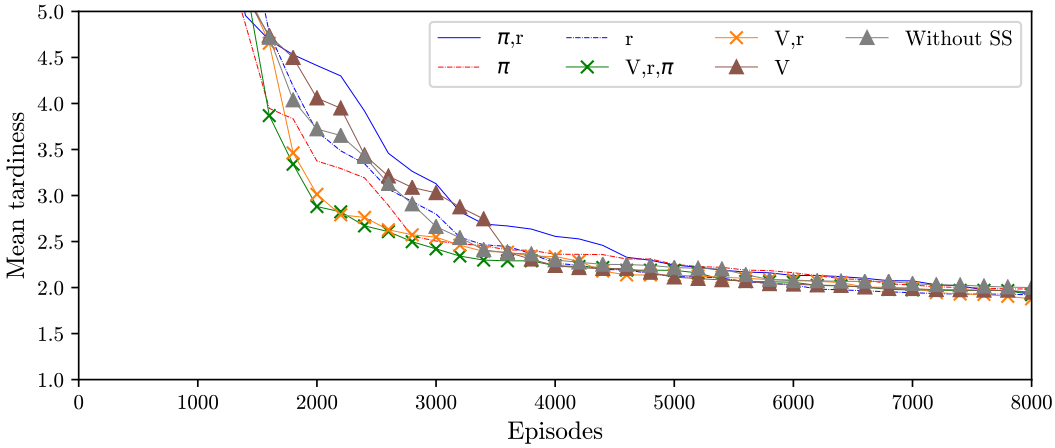
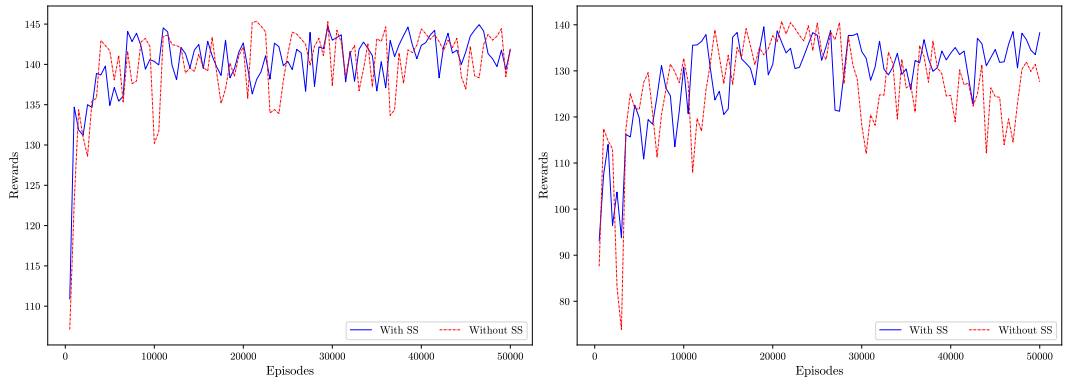


Figure 6.14: 데이터셋 L9에서 자기지도 포함 여부에 따른 평균 지연시간의 최저값 결과

함하였을 때의 TT가 눈에 띄게 낮음을 확인할 수 있다. 에피소드 2000에서 자기지도를 포함하지 않았을 때는 3.722, 포함했을 때는 2.879으로 22.6%의 개선율을 보였다. 이러한 결과는 [94]에서 자기지도 기반 DRL을 이용함으로써 DQN의 수렴 속도를 앞당긴 것과 같은 맥락이다. 따라서 가변성이 큰 스케줄링 문제 학습 시간을 단축시켜야 하는 상황에서는 자기지도의 효과가 더욱 두드러질 것으로 보인다.

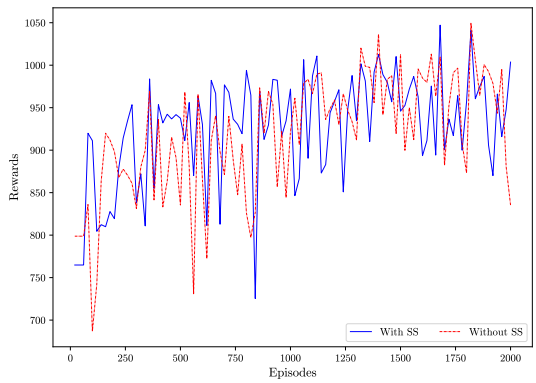
잡샵 스케줄링 문제의 FBS-1D

그림 6.15는 패키징 라인의 P1–P4 데이터셋에서의 자기지도 포함 여부에 따른 학습곡선을 표현하고 있다. 에피소드 증가에 따른 보상값의 변화 양상은 자기지도를 포함했을때(With SS)와 포함하지 않았을 때(Without SS) 뚜렷한 차이는 관찰되지 않았다. 그러나 데이터셋 P2의 경우에는 에피소드 40000 이후의 학습 후반부에서 보다 높은 보상값 결과가 반복적으로 관찰되었다. 또한 데이터셋 P3, P4에서 자기지도를 포함한 경우 에피소드 250 이전에도 보상값이 높은 결과가 기록되었다. 위 관찰에 따르면 패키징 라인에서도 자기지도로 인한 성능 개선이 가능한 것으로 보인다.

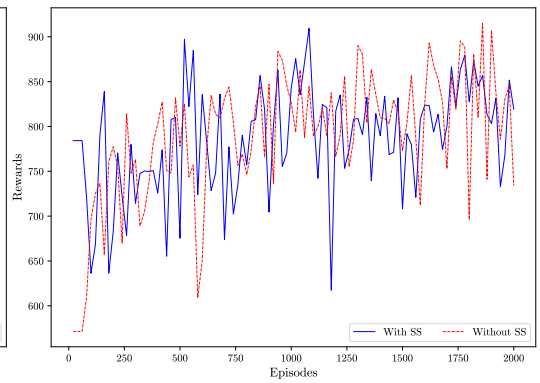


(a) 데이터셋 P1

(b) 데이터셋 P2



(c) 데이터셋 P3



(d) 데이터셋 P4

Figure 6.15: 패키징 라인 데이터셋에서 자기지도 포함 여부에 따른 보상값 학습 곡선

6.3.4 학습 시와 다른 데이터셋에서의 성능 평가

앞 장까지의 실험에서는 학습 때와 같은 데이터셋에 속한 30개 평가 스케줄링 문제를 활용하였다. 마지막으로 제안 기법의 데이터셋 변화에 대한 강건성을 검증하기 위하여, 학습 시와 다른 평가 데이터셋에서 성능을 비교하였다. 제안 기법은 생산 요구량 및 납기 뿐 아니라, 설비의 수와 초기 셋업 상태가 변화하여도 재학습없이 평가 스케줄링 문제를 풀 수 있다는 이점이 있어 모든 11개 데이터셋에 대한 교차 평가가 가능하다. 그 중에서도 먼저 조절 변인 η와 τ에 대한 제안 기법의 강건성을 조사하고자, 데이터셋 L1, L3, L4, L5에 대한 상호 평가를 시행하였다.

Table 6.12: 평가 데이터셋에서의 평균 지연시간 결과

학습 \ 평가	L1	L3	L4	L5
L1	1.7959	2.922	1.371	2.867
L3	1.987	2.934	1.486	2.633
L4	1.885	2.944	1.204	2.61
L5	1.876	2.936	1.418	2.708

표 6.12는 각 데이터셋에서 학습된 4개 스케줄러의 모든 평가 데이터셋에서의 평균 지연시간 결과를 나타낸다. 회색 음영으로 처리한 셀은 학습 데이터셋과 평가 데이터셋이 동일한, 기존의 실험 결과를 의미한다. 첫 번째 행에 제시된 결과에 따르면, 데이터셋 L1에서 자기지도를 동반하여 학습한 DQN을 데이터셋 L3–L5에 평가한 성능이 각각의 데이터셋에서 학습한 DQN으로 평가한 성능에서 크게 하락하지 않았다. 그 중에서도 데이터셋 L3의 경우 같은 데이터셋에서 학습한 스케줄러와의 성능차이가 0.57%로 근소하였다. 데이터셋 L3 및 L4로 학습한 스케줄러의 경우 데이터셋 L5에서 평가했을 때의 TT가 같은 데이터셋에서 학습한 스케줄러보다도 더 낮았다. 위 결과로 미루어 보건대, 제안 기법은 η, τ의 변화에 강건하다고 판단할 수 있다.

이후 학습 시와 다른 평가 데이터셋에서의 성능에 자기지도 기반 DRL이 미치는 영향을 파악하기 위해, 가변성을 키운 데이터셋 L6–L8에서 성능 비교를 수행하였다. 표 6.13은 4가지 종류의 평가 데이터셋에서 자기지도 포함 여부에 따른 평균 지연시간 결과를 보여준다. 예를 들어 L6에서 학습된 두 개 스케줄러는 동일 데이터셋인 L6과, L6에서 $\eta = 1$ 또는 3으로 변화한 데이터셋, $\tau = 0.5$인 데이터셋에서 평가되었다. 데이터셋 L7에서 동일 데이터셋으로 평가했을 때를 제외하고 모든 평가 데이터셋에서

자기지도를 포함했을 때의 TT가 더 낮았다. 데이터셋 L7의 경우 동일 데이터셋에서는 자기지도를 포함했을 때의 TT가 더 길지만, η 및 τ가 변화한 경우에는 7.7%–14.5% 만큼의 개선이 관찰되었다. 위 관찰에 따르면, 제안한 자기지도 기반 DRL 기법이 자기지도를 포함하지 않았을 경우보다 평가 데이터셋의 η 및 τ 변화에 더 강건한 성능을 보인다고 생각할 수 있다.

Table 6.13: 자기지도 포함 여부에 따른 평가 환경에서의 성능 차이

학습 \ 평가	동일 데이터셋	$\eta = 1$	$\eta = 3$	$\tau = 0.5$
L6(Without SS)	1.757	1.255	2.716	2.991
L6(With SS)	1.692	1.179	2.673	2.884
L7(Without SS)	3.591	1.907	3.851	3.384
L7(With SS)	3.740	1.630	3.436	3.125
L8(Without SS)	1.359	1.133	2.077	2.633
L8(With SS)	1.325	1.110	2.076	2.563

마지막으로 데이터셋 RI에서 자기지도 포함 여부에 따른 평가 결과를 비교하기 위하여, 데이터셋 L1에서 자기지도를 포함하여 학습한 DQN (With SS)과 자기지도 없이 학습한 DQN (Without SS)으로 성능 평가를 수행하였다. 또한 비교 기법으로 6.1.3 절의 IG와 제안 기법을 데이터셋 RI에서 재학습한 DQN (Re-training)을 활용하였다. 데이터셋 RI의 평가 스케줄링 문제는 모두 작업 시간 및 셋업 시간이 확률적으로 결정되며, 초기 셋업 상태가 서로 다르다는 사실에 주목하라.

그림 6.16은 데이터셋 RI의 30개 평가 스케줄링 문제에 대한 네 가지 기법의 평균 지연시간 결과 분포를 도포하여 보여준다. 자기지도를 포함하지 않았을 때의 TT 중간

값은 2.225 였으나 자기지도를 포함했을 때는 중간값 1.99로 유의미한 개선을 보였다. IG가 산출한 TT는 중간값 1.915로 제안 기법을 L1에서 학습한 후 RI에 평가한 TT 보다 더 낮았으나, 제안 기법을 RI에서 재학습한 결과인 1.77보다는 길었다. 얻은 실험 결과에 따르면, 기계 수, 잡 수, 생산 요구량 및 납기의 분포가 모두 다른 평가 데이터셋에서도 제안 기법은 재학습을 하지 않고도 사용 가능한 수준으로 보인다. 이러한 성과는 제안한 패밀리 별 우선순위 자기지도와 상태 및 보상 자기지도가 실제 현장과 유사한 환경에서도 보다 정확하게 미래의 TT값을 예측하는 데에 도움을 주는 덕분으로 해석된다.

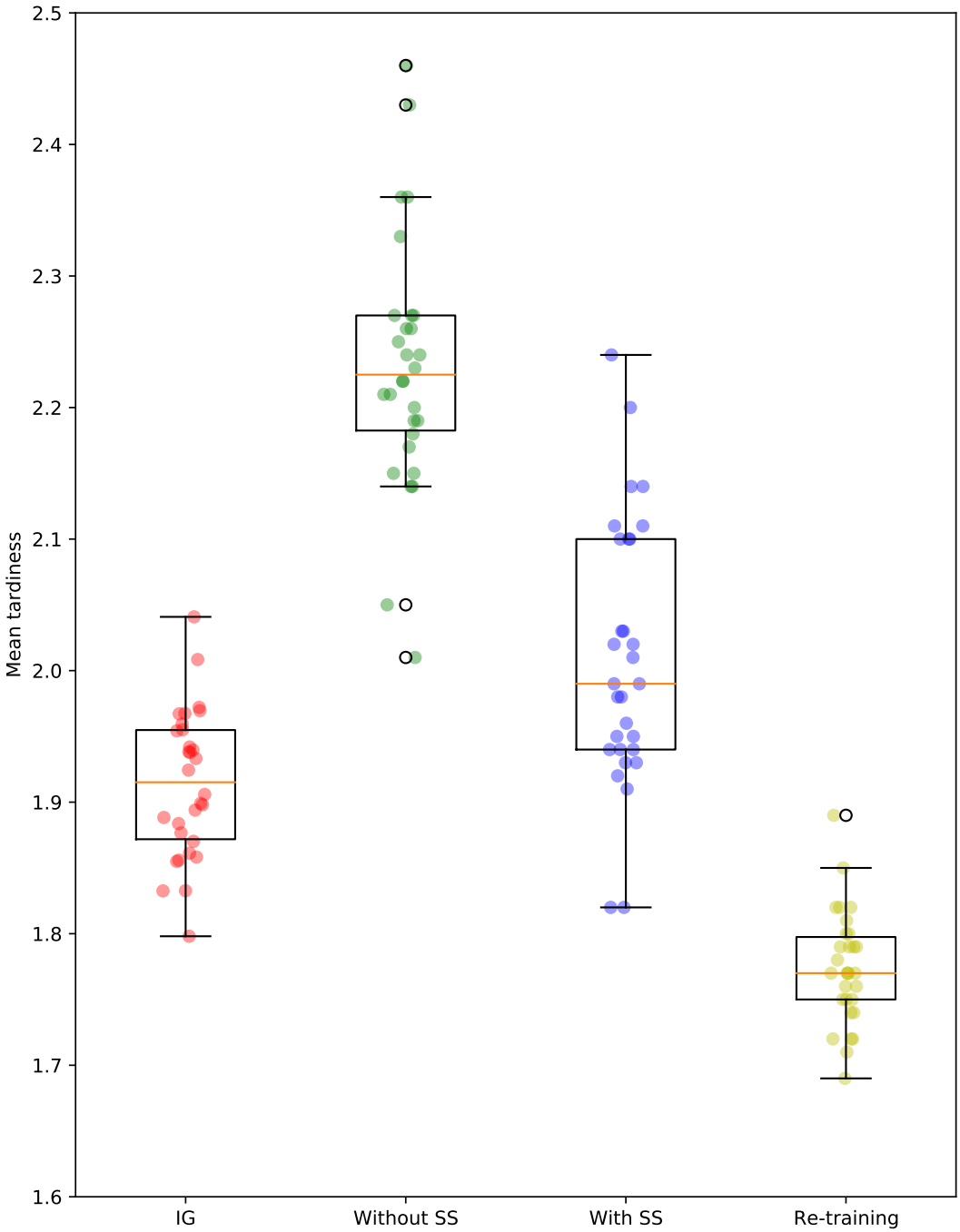


Figure 6.16: 데이터셋 RI에서의 IG 및 제안기법의 평균 지연시간 결과 분포

제 7 장 결론 및 향후 연구 방향

7.1 결론

본 논문에서는 납기 제약 하에서의 셋업 스케줄링 문제를 해결하기 위해 자기지도 기반 DRL 기법을 제안하였다. 구체적으로 납기와 셋업 제약을 함께 반영하면서, 생산 계획과 초기 설비 상태가 변화해도 차원이 유지되는 새로운 상태 및 행동 표현을 고안했다. 또한, 제안한 DQN 스케줄러를 효율적으로 학습하고 네트워크 크기를 줄일 수 있는 파라미터 공유 구조를 도입하였다. 끝으로 학습 속도와 일반화 성능을 향상시키기 위해 스케줄링 문제에 특화한 자기지도 손실을 부여하는 DRL 기법을 제안하였다.

제안 기법의 성능을 검증하고자 병렬설비 스케줄링 문제와 잡샵 스케줄링 문제로 대규모 데이터셋을 생성하여 실험을 수행하였다. 6.1 절에서는 지연시간 총합을 최소화하기 위한 병렬설비 스케줄링 문제에서 제안 기법을 최신 메타휴리스틱 기법과 두 개의 강화학습 기반 기법, 네 가지 규칙 기반 기법과 비교하였다. 실험 결과에 따르면 잡, 설비, 패밀리의 수가 다른 모든 데이터셋에서 지연시간 성능과 연산 시간 두 가지 측면에서 기존 기법을 능가하였다. 이에 더하여 제안 기법 내 비교를 통해 상태 표현과 파라미터 공유 구조가 유효하다는 것을 검증하였다.

다음으로 제안 기법을 투입량 최대화 목적의 잡샵 스케줄링 문제에 적용하기 위하여 스케줄링 프레임워크와 함께 일부 변형된 MDP 모형을 제시하였다. 이후 6.2 절에서 제안 기법의 유효성을 재검증하고자 DRL 기법과 다섯 가지 규칙 기반 기법과 성능 비교를 수행하였다. 또한 기존 방식의 행동 정의와 투입 정책에 기반을 둔 행동 정의와의 비교를 통해, 제안한 행동 표현의 우수성을 입증하였다. 결과적으로 학습된 DQN은 두

가지 공정 모두에서 생산 요구량과 납기, 초기 설비 상태가 새로 주어진 평가 스케줄링 문제를 신속하게 해결할 수 있었다.

마지막으로 자기지도로 인한 효과를 조사하기 위하여, 현실과 더욱 유사한 데이터셋에서 실험을 수행하였다. 제안 기법의 스케줄 성능 및 학습 곡선을 자기지도를 포함하지 않았을 때와 비교함으로써 자기지도로 인한 개선 효과를 관찰하였다. 파라미터 공유에 비하여 자기지도의 역할은 주목할 만하지 않았으나, 가변성이 큰 스케줄링 문제에서 학습 시간을 단축시키는 효과가 있는 것으로 추정된다. 또한 학습 시와 다른 데이터셋에서 평가 성능을 검증함으로써, 데이터 분포가 다른 스케줄링 문제에서도 재학습 없이 제안 기법을 이용할 수 있음을 확인하였다.

제안한 자기지도 기반 DRL 기법은 현장에 생산 계획과 초기 설비 상태의 가변성이 존재하더라도, 납기를 준수하는 고품질의 셋업 스케줄을 수립할 수 있다. 납기는 고객 만족과 생산비용 감소를 위한 중요한 지표로, 셋업 제약이 있는 여러 제조 라인에서 제안 기법이 제조기업의 생산성 향상에 직접적으로 기여할 수 있으리라 기대된다. 나아가 제안 기법을 응용하면 스마트 제조 시스템이 갖춰진 다양한 공정에 실제 적용될 수 있으며, 스케줄링 분야에서의 자기지도 기반 DRL 연구를 촉진시킬 수 있을 것이다.

7.2 향후 연구 방향

DRL을 이용한 스케줄링 기법은 가변성에 대응하여 빠르게 스케줄을 획득할 수 있으나, 재학습 없이도 성능을 보장하기 위해서는 가정한 제약과 가변성에 관한 면밀한 고려가 필요하다. 본 논문에서는 납기와 셋업 제약에 초점을 맞추었으나, 준비 시간이나 설비 고장 등이 존재하는 스케줄링 문제에서는 좋은 성능을 보장하기 어렵다. 따라서 향후에는 가용한 설비 변화나 잡의 준비 시간에 대한 정보도 상태 표현에 반영하기 위한 연구를 진행할 계획이다. 한편 현실의 웨이퍼 형성 공정, 반도체 패키징 라인을 모사하여 생산 요구량과 납기에 대한 분포를 가정하였으나, 가정한 가변성의 폭이 충분한가에 대해서는 논란의 여지가 있다. 가령 패키징 라인에서 생산 요구량은 10%만큼 변화할 수 있다고 하였으나, 추후 연구에서는 이러한 가변성의 폭을 더 늘려서 실험을 수행할 계획이다.

또 다른 한계로는 현장의 변화로 인해 상태 및 행동의 차원이 달라지는 경우에는 재학습이 필수적이라는 점이다. 제안 기법의 경우에는 패밀리 수가 바뀌는 경우 재학습 절차 없이 스케줄을 수립할 수 없다. 따라서 패밀리 수와 무관한 차원을 가진 상태 및 행동을 개발할 필요성이 있다. 이외에도 설비에 모델 종류가 존재하거나 잡의 속성 종류가 추가되는 경우에도 재학습이 필요하므로, 보다 범용적인 기법을 고안해야 한다. 추가적으로, 본 논문에서는 고려하지 않은 가변성을 반영할 수 있도록 제안 기법을 개선하거나, recurrent DNN이나 CNN 등의 다른 딥 러닝 모델을 활용하여 네트워크 구조를 개선할 수 있다.

본 논문의 자기지도 기반 DRL 기법은 추가적으로 스케줄링 문제에 적합한 자기지도 손실을 부여하였으나 뚜렷한 성능 개선을 이끌어내지는 못하였다. 로봇 제어, 게임 분야에서만큼 두드러지는 효과를 보기 위해서는 다양하고 복잡한 형태로 주어지는 더욱 차원이 큰 상태 표현을 활용하는 상황이 유리하다. 따라서 본 논문과는 다른 접근으로

실제 현장에서 제공되는 이미지, 오디오 데이터나 사물인터넷의 감지기가 제공하는 데이터를 함께 상태 표현에 활용하는 방안을 고려할 수 있다. 또한 더욱 규모가 크고 제약이 많은 스케줄링 문제에서 제안한 자기지도가 가져다주는 일반화 성능 측면에서의 이점이 증가할 것으로 예상된다. 추후에는 대상 공정을 확대하고 가용한 데이터를 늘려나가며 자기지도 기반 DRL을 이용한 스케줄링 기법을 지속적으로 발전시킬 계획이다.

참고 문헌

[1] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[2] S. Qu, J. Wang, and G. Shivani, "Learning adaptive dispatching rules for a manufacturing process system by using reinforcement learning approach," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2016, pp. 1–8.

[3] W. Pearn, S. Chung, and M. Yang, "The wafer probing scheduling problem (wpsp)," *Journal of the Operational Research Society*, vol. 53, no. 8, pp. 864–874, 2002.

[4] T. Yang, Y.-F. Wen, Z.-R. Hsieh, and J. Zhang, "A lean production system design for semiconductor crystal-ingot pulling manufacturing using hybrid taguchi method and simulation optimization," *Assembly Automation*, vol. 40, no. 3, pp. 433–445, 2020.

[5] L. Mönch, J. W. Fowler, and S. J. Mason, "Semiconductor manufacturing process description," in *Production Planning and Control for Semiconductor Wafer Fabrication Facilities*. Springer, 2013, pp. 11–28.

[6] L. Shen, L. Mönch, and U. Buscher, "An iterative approach for the serial batching problem with parallel machines and job families," *Annals of Operations*

Research, vol. 206, no. 1, pp. 425–448, 2013.

[7] C. A. Sáenz-Alanís, V. Jobish, M. A. Salazar-Aguilar, and V. Boyer, "A parallel machine batch scheduling problem in a brewing company," *The International Journal of Advanced Manufacturing Technology*, vol. 87, no. 1, pp. 65–75, 2016.

[8] A. Allahverdi, "The third comprehensive survey on scheduling problems with setup times/costs," *European Journal of Operational Research*, vol. 246, no. 2, pp. 345–378, 2015.

[9] R. H. Suriyaarachchi and A. Wirth, "Earliness/tardiness scheduling with a common due date and family setups," *Computers & Industrial Engineering*, vol. 47, no. 2-3, pp. 275–288, 2004.

[10] Y.-R. Shiue, K.-C. Lee, and C.-T. Su, "A reinforcement learning approach to dynamic scheduling in a product-mix flexibility environment," *IEEE Access*, vol. 8, pp. 106 542–106 553, 2020.

[11] J.-F. Chen, "Scheduling on unrelated parallel machines with sequence-and machine-dependent setup times and due-date constraints," *The International Journal of Advanced Manufacturing Technology*, vol. 44, no. 11-12, pp. 1204–1212, 2009.

[12] J. Du and J. Y.-T. Leung, "Minimizing total tardiness on one machine is np-hard," *Mathematics of operations research*, vol. 15, no. 3, pp. 483–495, 1990.

[13] J. Huh, I. Park, S. Lim, B. Paeng, J. Park, and K. Kim, "Learning to dispatch operations with intentional delay for re-entrant multiple-chip product assembly lines," *Sustainability*, vol. 10, no. 11, p. 4123, 2018.

[14] S.-W. Lin, C.-C. Lu, and K.-C. Ying, "Minimization of total tardiness on unrelated parallel machines with sequence-and machine-dependent setup times under due date constraints," *The International Journal of Advanced Manufacturing Technology*, vol. 53, no. 1-4, pp. 353–361, 2011.

[15] J. C. Pinheiro, J. E. C. Arroyo, and L. B. Fialho, "Scheduling unrelated parallel machines with family setups and resource constraints to minimize total tardiness," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 2020, pp. 1409–1417.

[16] Y. H. Lee, K. Bhaskaran, and M. Pinedo, "A heuristic to minimize the total weighted tardiness with sequence-dependent setups," *IIE transactions*, vol. 29, no. 1, pp. 45–52, 1997.

[17] J. Lim, M.-J. Chae, Y. Yang, I.-B. Park, J. Lee, and J. Park, "Fast scheduling of semiconductor manufacturing facilities using case-based reasoning," *IEEE Transactions on Semiconductor Manufacturing*, vol. 29, no. 1, pp. 22–32, 2015.

[18] W. Zhang and T. G. Dietterich, "A reinforcement learning approach to job-shop scheduling," in *IJCAI*, vol. 95. Citeseer, 1995, pp. 1114–1120.

[19] T. Gabel and M. Riedmiller, "Adaptive reactive job-shop scheduling with reinforcement learning agents," *International Journal of Information Technology and Intelligent Computing*, vol. 24, no. 4, pp. 14–18, 2008.

[20] Z. Cao, C. Lin, M. Zhou, and R. Huang, "Scheduling semiconductor testing facility by using cuckoo search algorithm with reinforcement learning and surrogate modeling," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 825–837, 2018.

[21] S. Luo, "Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning," *Applied Soft Computing*, vol. 91, p. 106208, 2020.

[22] A. Kramer and A. Subramanian, "A unified heuristic and an annotated bibliography for a large class of earliness–tardiness scheduling problems," *Journal of Scheduling*, vol. 22, no. 1, pp. 21–57, 2019.

[23] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," in *Annals of discrete mathematics*. Elsevier, 1979, vol. 5, pp. 287–326.

[24] M. A. Bozorgirad and R. Logendran, "Sequence-dependent group scheduling problem on unrelated-parallel machines," *Expert Systems with Applications*, vol. 39, no. 10, pp. 9021–9030, 2012.

[25] O. Shahvari and R. Logendran, "An enhanced tabu search algorithm to minimize a bi-criteria objective in batching and scheduling problems on unrelated-parallel machines with desired lower bounds on batch sizes," *Computers & Operations Research*, vol. 77, pp. 154–176, 2017.

[26] A. Ekici, M. Elyasi, O. Ö. Özener, and M. B. Sarıkaya, "An application of unrelated parallel machine scheduling with sequence-dependent setups at vestel electronics," *Computers & Operations Research*, vol. 111, pp. 130–140, 2019.

[27] J. R. Zeidi and S. MohammadHosseini, "Scheduling unrelated parallel machines with sequence-dependent setup times," *The International Journal of Advanced Manufacturing Technology*, vol. 81, no. 9, pp. 1487–1496, 2015.

[28] T. Park, T. Lee, and C. O. Kim, "Due-date scheduling on parallel machines with job splitting and sequence-dependent major/minor setup times," *The International Journal of Advanced Manufacturing Technology*, vol. 59, no. 1, pp. 325–333, 2012.

[29] J.-M. Yu, R. Huang, and D.-H. Lee, "Iterative algorithms for batching and scheduling to minimise the total job tardiness in two-stage hybrid flow shops," *International Journal of Production Research*, vol. 55, no. 11, pp. 3266–3282, 2017.

[30] J.-G. Kim, S. Song, and B. Jeong, "Minimising total tardiness for the identical parallel machine scheduling problem with splitting jobs and sequence-dependent setup times," *International Journal of Production Research*, vol. 58, no. 6, pp. 1628–1643, 2020.

[31] C. N. Potts and M. Y. Kovalyov, "Scheduling with batching: A review," *European journal of operational research*, vol. 120, no. 2, pp. 228–249, 2000.

[32] D.-W. Kim, D.-G. Na, and F. F. Chen, "Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective," *Robotics and Computer-Integrated Manufacturing*, vol. 19, no. 1-2, pp. 173–181, 2003.

[33] S. J. Mason, J. W. Fowler, and W. Matthew Carlyle, "A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops," *Journal of Scheduling*, vol. 5, no. 3, pp. 247–262, 2002.

[34] K.-C. Ying and S.-W. Lin, "Unrelated parallel machine scheduling with sequence-and machine-dependent setup times and due date constraints," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 5, pp. 3279–3297, 2012.

[35] D. E. Akyol and G. M. Bayhan, "Multi-machine earliness and tardiness scheduling problem: an interconnected neural network approach," *The International Journal of Advanced Manufacturing Technology*, vol. 37, no. 5-6, pp. 576–588, 2008.

[36] Q. Liao, "Study of svm-based intelligent dispatcher for parallel machines scheduling with sequence-dependent setup times," in *2018 6th International Conference on Mechanical, Automotive and Materials Engineering (CMAME)*. IEEE, 2018, pp. 46–50.

[37] J. Heger, J. Branke, T. Hildebrandt, and B. Scholz-Reiter, "Dynamic adjustment of dispatching rule parameters in flow shops with sequence-dependent set-up times," *International Journal of Production Research*, vol. 54, no. 22, pp. 6812–6824, 2016.

[38] W. Yoo, J. Seo, D. Lee, D. Kim, and K. Kim, "Scheduling generation model on parallel machines with due date and setup cost based on deep learning," *Journal of Society for e-Business Studies*, vol. 24, no. 3, 2020.

[39] C.-L. Chen, "Iterated hybrid metaheuristic algorithms for unrelated parallel machines problem with unequal ready times and sequence-dependent setup times," *The International Journal of Advanced Manufacturing Technology*, vol. 60, no. 5-8, pp. 693–705, 2012.

[40] J.-H. Lee, J.-M. Yu, and D.-H. Lee, "A tabu search algorithm for unrelated parallel machine scheduling with sequence-and machine-dependent setups: minimizing total tardiness," *The International Journal of Advanced Manufacturing Technology*, vol. 69, no. 9-12, pp. 2081–2089, 2013.

[41] A. Bilyk, L. Mönch, and C. Almeder, "Scheduling jobs with ready times and precedence constraints on parallel batch machines using metaheuristics," *Computers & Industrial Engineering*, vol. 78, pp. 175–185, 2014.

[42] Y.-J. Choi, "A genetic algorithm for minimizing total tardiness with non-identical parallel machines," *Journal of the Society of Korea Industrial and Systems Engineering*, vol. 38, no. 1, pp. 65–73, 2015.

[43] E. Caniyilmaz, B. Benli, and M. S. Ilkay, "An artificial bee colony algorithm approach for unrelated parallel machine scheduling with processing set restrictions, job sequence-dependent setup times, and due date," *The International Journal of Advanced Manufacturing Technology*, vol. 77, no. 9-12, pp. 2105–2115, 2015.

[44] J. Lamothe, F. Marmier, M. Dupuy, P. Gaborit, and L. Dupont, "Scheduling rules to minimize total tardiness in a parallel machine problem with setup and calendar constraints," *Computers & Operations Research*, vol. 39, no. 6, pp. 1236–1244, 2012.

[45] Ü. Bilge, F. Kıraç, M. Kurtulan, and P. Pekgün, "A tabu search algorithm for parallel machine total tardiness problem," *Computers & Operations Research*, vol. 31, no. 3, pp. 397–414, 2004.

[46] P. Sharma and A. Jain, "A review on job shop scheduling with setup times," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 230, no. 3, pp. 517–533, 2016.

[47] J. N. Gupta, R. Ruiz, J. Fowler, and S. Mason, "Operational planning and control of semiconductor wafer production," *Production Planning & Control*, vol. 17, no. 7, pp. 639–647, 2006.

[48] Y. H. Chung, S. Lee, and S. C. Park, "Dispatching for order-driven fabs with backward pegging," *Cogent Engineering*, vol. 3, no. 1, p. 1226460, 2016.

[49] Y.-H. Jeong, K.-H. Cho, Y.-I. Choung, and S.-C. Park, "Scheduling methodology for mcp (multi-chip package) with layer sequence constraint in semiconductor package," *Journal of the Korea Society for Simulation*, vol. 26, no. 1, pp. 69–75, 2017.

[50] Y. M. Joung, T. He, S. W. Yoon, R. Vancheeswaran, C. Abela, and H. R. Andres, "Multi-pass lot scheduling algorithm for maximizing throughput at

semiconductor final test facilities," *Procedia Manufacturing*, vol. 11, pp. 1992–2000, 2017.

[51] C. L. Monma and C. N. Potts, "On the complexity of scheduling with batch setup times," *Operations research*, vol. 37, no. 5, pp. 798–804, 1989.

[52] B.-s. Chung, J. Lim, I.-B. Park, J. Park, M. Seo, and J. Seo, "Setup change scheduling for semiconductor packaging facilities using a genetic algorithm with an operator recommender," *IEEE Transactions on Semiconductor Manufacturing*, vol. 27, no. 3, pp. 377–387, 2014.

[53] O. L. Costa and M. D. Fragoso, "Discrete-time lq-optimal control problems for infinite markov jump parameter systems," *IEEE Transactions on Automatic Control*, vol. 40, no. 12, pp. 2076–2088, 1995.

[54] H. Shen, M. Dai, Y. Luo, J. Cao, and M. Chadli, "Fault-tolerant fuzzy control for semi-markov jump nonlinear systems subject to incomplete smk and actuator failures," *IEEE Transactions on Fuzzy Systems*, 2020.

[55] D. P. Bertsekas, "Dynamic programming and optimal control 3rd edition, volume ii," *Belmont, MA: Athena Scientific*, 2011.

[56] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[57] Y.-R. Shiue, K.-C. Lee, and C.-T. Su, "Real-time scheduling for a smart factory using a reinforcement learning approach," *Computers & Industrial Engineering*, vol. 125, pp. 604–614, 2018.

[58] B. Waschneck, A. Reichstaller, L. Belzner, T. Altenmüller, T. Bauernhansl, A. Knapp, and A. Kyek, "Deep reinforcement learning for semiconductor production scheduling," in *2018 29th annual SEMI advanced semiconductor manufacturing conference (ASMC)*. IEEE, 2018, pp. 301–306.

[59] H. Rummukainen and J. K. Nurminen, "Practical reinforcement learning-experiences in lot scheduling application," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 1415–1420, 2019.

[60] I.-B. Park, J. Huh, J. Kim, and J. Park, "A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1420–1431, 2019.

[61] J. Shahrabi, M. A. Adibi, and M. Mahootchi, "A reinforcement learning approach to parameter estimation in dynamic job shop scheduling," *Computers & Industrial Engineering*, vol. 110, pp. 75–82, 2017.

[62] Z. Zhang, L. Zheng, and M. X. Weng, "Dynamic parallel machine scheduling with mean weighted tardiness objective by q-learning," *The International Journal of Advanced Manufacturing Technology*, vol. 34, no. 9-10, pp. 968–980, 2007.

[63] Z. Zhang, L. Zheng, N. Li, W. Wang, S. Zhong, and K. Hu, "Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning," *Computers & operations research*, vol. 39, no. 7, pp. 1315–1324, 2012.

[64] B. Yuan, L. Wang, and Z. Jiang, "Dynamic parallel machine scheduling using the learning agent," in *2013 IEEE international conference on industrial engineering and engineering management*. IEEE, 2013, pp. 1565–1569.

[65] B. Yuan, Z. Jiang, and L. Wang, "Dynamic parallel machine scheduling with random breakdowns using the learning agent," *International Journal of Services Operations and Informatics*, vol. 8, no. 2, pp. 94–103, 2016.

[66] Y.-F. Wang, "Adaptive job shop scheduling strategy based on weighted q-learning algorithm," *Journal of Intelligent Manufacturing*, vol. 31, no. 2, pp. 417–432, 2020.

[67] J. Hong and V. V. Prabhu, "Distributed reinforcement learning control for batch sequencing and sizing in just-in-time manufacturing systems," *Applied Intelligence*, vol. 20, no. 1, pp. 71–87, 2004.

[68] W. Bouazza, Y. Sallez, and B. Beldjilali, "A distributed approach solving partially flexible job-shop scheduling problem with a q-learning effect," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 15 890–15 895, 2017.

[69] Z. Zhang, L. Zheng, F. Hou, and N. Li, "Semiconductor final test scheduling with sarsa (λ, k) algorithm," *European Journal of Operational Research*, vol. 215, no. 2, pp. 446–458, 2011.

[70] N. Stricker, A. Kuhnle, R. Sturm, and S. Friess, "Reinforcement learning for adaptive order dispatching in the semiconductor industry," *CIRP Annals*, vol. 67, no. 1, pp. 511–514, 2018.

[71] D. Terekhov, T. T. Tran, D. G. Down, and J. C. Beck, "Integrating queueing theory and scheduling for dynamic scheduling problems," *Journal of Artificial Intelligence Research*, vol. 50, pp. 535–572, 2014.

[72] Y.-C. Wang and J. M. Usher, "Application of reinforcement learning for agent-based production scheduling," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 1, pp. 73–82, 2005.

[73] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[74] L. Yin, S. Li, and H. Liu, "Lazy reinforcement learning for real-time generation control of parallel cyber–physical–social energy systems," *Engineering Applications of Artificial Intelligence*, vol. 88, p. 103380, 2020.

[75] Y. Cheng, J. Peng, X. Gu, F. Jiang, H. Li, W. Liu, and Z. Huang, "Optimal energy management of energy internet: A distributed actor-critic reinforcement learning method," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 521–526.

[76] M. Nazari, A. Oroojlooy, L. V. Snyder, and M. Takáč, "Reinforcement learning for solving the vehicle routing problem," *arXiv preprint arXiv:1802.04240*, 2018.

[77] M. M. Hasan, K. Lwin, M. Imani, A. Shabut, L. F. Bittencourt, and M. A. Hossain, "Dynamic multi-objective optimisation using deep reinforcement learn-

ing: benchmark, algorithm and an application to identify vulnerable zones based on water quality," *Engineering Applications of Artificial Intelligence*, vol. 86, pp. 107–135, 2019.

[78] Y. Ye, D. Qiu, J. Li, and G. Strbac, "Multi-period and multi-spatial equilibrium analysis in imperfect electricity markets: A novel multi-agent deep reinforcement learning approach," *IEEE Access*, vol. 7, pp. 130 515–130 529, 2019.

[79] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM workshop on hot topics in networks*, 2016, pp. 50–56.

[80] Y. Bao, Y. Peng, and C. Wu, "Deep learning-based job placement in distributed machine learning clusters," in *IEEE INFOCOM 2019-IEEE conference on computer communications*. IEEE, 2019, pp. 505–513.

[81] A. Mirhoseini, H. Pham, Q. V. Le, B. Steiner, R. Larsen, Y. Zhou, N. Kumar, M. Norouzi, S. Bengio, and J. Dean, "Device placement optimization with reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2430–2439.

[82] Z. Cao, H. Zhang, Y. Cao, and B. Liu, "A deep reinforcement learning approach to multi-component job scheduling in edge computing," in *2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*. IEEE, 2019, pp. 19–24.

[83] W. Guo, W. Tian, Y. Ye, L. Xu, and K. Wu, "Cloud resource scheduling with deep reinforcement learning and imitation learning," *IEEE Internet of Things Journal*, 2020.

[84] D. Shi, W. Fan, Y. Xiao, T. Lin, and C. Xing, "Intelligent scheduling of discrete automated production line via deep reinforcement learning," *International Journal of Production Research*, vol. 58, no. 11, pp. 3362–3380, 2020.

[85] Z. He, K.-P. Tran, S. Thomassey, X. Zeng, J. Xu, and C. Yi, "A deep reinforcement learning based multi-criteria decision support system for optimizing textile chemical process," *Computers in Industry*, vol. 125, p. 103373, 2021.

[86] C.-C. Lin, D.-J. Deng, Y.-L. Chih, and H.-T. Chiu, "Smart manufacturing scheduling with edge computing using multiclass deep q network," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4276–4284, 2019.

[87] C. Zhang, W. Song, Z. Cao, J. Zhang, P. S. Tan, and X. Chi, "Learning to dispatch for job shop scheduling via deep reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[88] C.-L. Liu, C.-C. Chang, and C.-J. Tseng, "Actor-critic deep reinforcement learning for solving job shop scheduling problems," *IEEE Access*, vol. 8, pp. 71 752–71 762, 2020.

[89] B.-A. Han and J.-J. Yang, "Research on adaptive job shop scheduling problems based on dueling double dqn," *IEEE Access*, vol. 8, pp. 186 474–186 495, 2020.

[90] T. E. Thomas, J. Koo, S. Chaterji, and S. Bagchi, "Minerva: A reinforcement learning-based technique for optimal scheduling and bottleneck detection in distributed factory operations," in *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*. IEEE, 2018, pp. 129–136.

[91] S. Zheng, C. Gupta, and S. Serita, "Manufacturing dispatching using reinforcement and transfer learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2019, pp. 655–671.

[92] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat, "State representation learning for control: An overview," *Neural Networks*, vol. 108, pp. 379–392, 2018.

[93] S. Lange and M. Riedmiller, "Deep auto-encoder neural networks in reinforcement learning," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2010, pp. 1–8.

[94] E. Shelhamer, P. Mahmoudieh, M. Argus, and T. Darrell, "Loss is its own reward: Self-supervision for reinforcement learning," *arXiv preprint arXiv:1612.07307*, 2016.

[95] D. Graves, N. M. Nguyen, K. Hassanzadeh, and J. Jin, "Learning predictive representations in autonomous driving to improve deep reinforcement learning," *arXiv preprint arXiv:2006.15110*, 2020.

[96] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2778–2787.

[97] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Agent modeling as auxiliary task for deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 15, 2019, pp. 31–37.

[98] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," *arXiv preprint arXiv:1611.05397*, 2016.

[99] G. Lample and D. S. Chaplot, "Playing fps games with deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[100] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Wiele, V. Mnih, N. Heess, and J. T. Springenberg, "Learning by playing solving sparse reward tasks from scratch," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4344–4353.

[101] S. Racanière, T. Weber, D. P. Reichert, L. Buesing, A. Guez, D. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li *et al.*, "Imagination-augmented agents for deep reinforcement learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 5694–5705.

[102] M. G. Bellemare, W. Dabney, R. Dadashi, A. A. Taiga, P. S. Castro, N. L. Roux, D. Schuurmans, T. Lattimore, and C. Lyle, "A geometric perspective on optimal representations for reinforcement learning," *arXiv preprint arXiv:1901.11530*, 2019.

[103] C. Zhang, O. Vinyals, R. Munos, and S. Bengio, "A study on overfitting in deep reinforcement learning," *arXiv preprint arXiv:1804.06893*, 2018.

[104] D. Harris and S. Harris, *Digital design and computer architecture*. Morgan Kaufmann, 2010.

[105] J. Han, M. Kamber, and J. Pei, "Data mining concepts and techniques third edition," *The Morgan Kaufmann Series in Data Management Systems*, vol. 5, no. 4, pp. 83–124, 2011.

[106] D. Zhang, D. Dai, Y. He, and F. S. Bao, "Rlscheduler: Learn to schedule hpc batch jobs using deep reinforcement learning," *arXiv e-prints*, 2019.

[107] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[108] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[109] A. P. Vepsalainen and T. E. Morton, "Priority rules for job shops with weighted tardiness costs," *Management science*, vol. 33, no. 8, pp. 1035–1047, 1987.

[110] J. C. Yepes-Borrero, F. Perea, R. Ruiz, and F. Villa, "Bi-objective parallel machine scheduling with additional resources during setups," *European Journal of Operational Research*, vol. 292, no. 2, pp. 443–455, 2021.

[111] J. Banks, *Discrete event system simulation*. Pearson Education India, 2005.

[112] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[113] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[114] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[115] R. Haupt, "A survey of priority rule-based scheduling," *Operations-Research-Spektrum*, vol. 11, no. 1, pp. 3–16, 1989.

Abstract

Setup Change Scheduling Under Due-date Constraints Using Deep Reinforcement Learning with Self-supervision

Bohyung Paeng
Department of Industrial Engineering
The Graduate School
Seoul National University

Setup change scheduling under due-date constraints has attracted much attention from academia and industry due to its practical applications. In a real-world manufacturing system, however, solving the scheduling problem becomes challenging since it is required to address urgent and frequent changes in demand and due-dates of products, and initial machine status. In this thesis, we propose a scheduling framework based on deep reinforcement learning (RL) with self-supervision in which trained neural networks (NNs) are able to solve unseen scheduling problems without re-training even when such changes occur. Specifically, we propose state and action representations whose dimensions are independent of production requirements and due-dates of jobs while accommodating family setups. At the same time, an NN architecture with parameter sharing was utilized to improve the training efficiency.

Finally, we devise an additional self-supervised loss specific to the scheduling problem for training the NN scheduler robust to the variations in the numbers of machines and jobs, and distribution of production plans.

We carried out extensive experiments in large-scale datasets that simulate the real-world wafer preparation facility and semiconductor packaging line. Experiment results demonstrate that the proposed method outperforms the recent metaheuristics, rule-based, and other RL-based methods in terms of the schedule quality and computation time for obtaining a schedule. Besides, we investigated individual contributions of the state representation, parameter sharing, and self-supervision on the performance improvements.

Keywords: Manufacturing line scheduling based on reinforcement learning, Deep reinforcement learning with self-supervision, Sequence-dependent setups, Parallel machine scheduling with due-date related objectives, Semiconductor packaging line scheduling

Student Number: 2013-21087

감사의 글

처음 부푼 꿈을 안고 관악산 기숙사에서 학문을 시작하던 스물두 살의 미숙한 제가 어느덧 박사학위를 받고 넓은 세상으로 나아가게 되었습니다. 지금에 돌이켜 보니 지난 8년은 연구자로서 성장하고 평생의 자산이 되어 줄 덕목을 체득함과 아울러 귀감이 되는 경험으로 가득 찬 소중한 순간들이었습니다. 끝없는 터널같이 느껴졌던 길을 지나 이제 서른의 새로운 시작을 내딛기에 앞서, 그동안 어둠 속에서 흔들릴 때마다 등불이 되어 주신 많은 분들께 지면을 빌어 감사를 전합니다.

먼저 마지막까지 멈추지 않고 나아가도록 올바른 길을 안내해주신 박종헌 지도교수님께 감사드립니다. 연구자로서 첫 걸음을 내딛기 까지 교수님 덕분에 여러 자질을 쌓을 수 있었습니다. 학문을 대하는 자세, 명확하고 과학적인 의사소통 방식에 대한 가르침 잊지 않고 꾸준히 성장하는 연구자가 되겠습니다. 더불어 바쁘신 중에도 흔쾌히 논문의 심사를 맡아주시고 아낌없는 조언을 해주신 조성준 교수님 감사드립니다. 또한, 이전부터 제 연구 분야에 관심을 가져주시고 진심 어린 독려를 건네주신 이경식 교수님 감사합니다. 학위를 취득한 이후 정진해야 할 방향과 학술활동 등의 조언을 해주신 김관호 교수님, 함께 재학 중일 때부터 심사 종료까지 연구하고 협업하는 자세에 대해 알려주신 허재석 교수님 감사합니다. 학업적, 행정적으로 꾸준히 도움을 주신 산업공학과 교수님들과 교직원 분들께도 감사드립니다.

더하여 제가 수학할 수 있는 탄탄한 교육 체계와 연구 기반을 마련해준 서울대학교와 관련 정부 부처, 사회에서 필요로 하는 훌륭한 연구주제를 위탁했던 국내 유수의 기업들에게도 감사합니다. 특히 비상한 안목으로 흥미롭고 유망한 연구 과제를 제공한 프로페셔널 컨설팅 그룹의 곽경재 이사님, 윤영민 이사님, 이호열 박사님, 배순용 선배님, 최지훈 선배님 감사합니다. 저를 둘러싼 우수한 환경 덕분에 얻은 학문적 성취임을 잊지 않고 배운 지식을 인류의 번영을 위해 나눌 수 있는 연구자로 거듭나겠습니다.

연구실을 거쳐 가며 저와 함께 생활했던 선후배들에게도 감사를 전합니다. 1년차 때에 듬직하게 팀을 이끌어주시고 졸업 후 진로에 도움을 주신 범석이형, 2년차에 스마트폰 데이터 분석 과제에서 의미 있는 역할을 부여해준 최예림 선배님, 대학원 및 사회생활에 대해 알려주신 성호형님, 딥러닝 기법을 도입하고 연구실의 서버를 줄곧 관리해주셨던 승욱이형, 3년차에 제조업 팀을 이끌며 제가 흥미 있는 주제를 시작하도록 도우신 용석이형, 4년차부터 강화학습 스케줄링 과제장을 맡고 한 걸음 앞서 걸으며 연구를 도운 인범이형, 7년차 AI 기반 솔루션 과제를 함께하며 잘 따라와 준 동현이, 함께 입학했을 때부터 쭉 제 연구 내용에 대해 성심것 논의해주신 희웅이형 감사합니다. 연구 주제는 달랐으나 세미나 시간을 통해 활발히 교류했던 여러 재학생 분들에게도 감사합니다. 센서 데이터를 깊이 연구하며 최신 기법을 소개해 준 문정이형과 유민이, 앞으로 금융팀의 미래를 이끌어나갈 도균이형, 석현이형, 음악팀이지만 제 연구에도 관심을 가져주었던 종권이, 완이, 강화학습에 조예가 있던 성민이형과 진현이, 제가 방장을 맡을 때에 입학해서 연구실의 새 전력이 되었던 성은, 창진, 같은 학기에 졸업하면서 논문 작성과 저널 투고에 대한 피드백도 아끼지 않던 종혁이형, 교윤이 감사합니다. 또한 일일이 언급하지 못한 졸업생 및 재학생 일동께 감사드리며 앞으로도 좋은 인연으로 정보경영연구실의 이름이 빛날 수 있도록 함께 성장하는 동반자가 되길 희망합니다.

끝으로 긴 시간동안 옆을 지켜주며 감정적인 지원을 보낸 여러 친지, 가족 분들께 감사합니다. 할아버지, 할머니, 외할아버지, 외할머니, 네 분 모두 건강히 제 모습을 지켜봐주셔서 감사드리며, 앞으로도 오래오래 만수무강하시길 기원합니다. 저를 대신하여 가족의 기둥이 되어 준 외삼촌과 여동생을 비롯해 물심양면으로 지원을 아끼지 않으신 팽기석 박사님과 조미향 여사님께도 감사드립니다. 저를 세상에 태어나게 해주고 학업을 무사히 마칠 때까지 길러주신 은혜를 잊지 않고 행복하게 살아가는 모습 보여드리며 보답하는 아들이 되겠습니다. 지면의 부족으로 일일이 언급하지 못한 인연들, 이십대의 소중한 순간을 함께 공유한 친구 지인 선후배들 모두 저의 스승이고 등불이었습니다. 감사합니다.