Journal of the Korean Institute of Industrial Engineers
Vol. 52, No. 4, pp. 291-302, August 2026.
ISSN 1225-0988 | EISSN 2234-6457
https://doi.org/10.7232/JKIIE.2026.52.4.291

에이전트 기반 제조실행시스템을 위한 분산형 심층강화학습 기반 동적 라우팅

백종호 · 신문수†

국립한밭대학교 산업경영공학과

Distributed Deep Reinforcement Learning-based Dynamic Routing for Agent-based Manufacturing Execution Systems

Jongho Baek · Moonsoo Shin

Department of Industrial and Management Engineering, Hanbat National University

In smart manufacturing, routing decisions for work-in-process (WIP) are critical factors affecting operational performance, including lead time, makespan, and congestion. Conventional approaches, such as shortest-path algorithms and rule-based heuristics, rely on predefined cost structures and thus have limitations in adapting to dynamic production environments. In this paper, we propose a distributed dynamic routing framework employing deep reinforcement learning (DRL) within an agent-based manufacturing execution system (MES). The routing problem is formulated as a multi-agent Markov decision process, where each WIP unit acts as an independent agent selecting its next movement based on locally observable states. A parameter-sharing deep Q-network (DQN) is used for scalable learning and decentralized execution, and the reward structure is optimized to minimize the average lead time. Simulation results demonstrate that the proposed DQN-based approach significantly outperforms Dijkstra-based routing and tabular Q-learning while maintaining low computational latency, demonstrating its applicability to real-time, large-scale manufacturing environments.

Keywords: Smart Manufacturing, Manufacturing Execution Systems, Deep Reinforcement Learning, Multi-agent Systems, Distributed Routing

1. 서 론

스마트팩토리는 공정설비와 자재취급설비(material handling equipment), 작업물 등이 네트워크로 연결되어 실시간으로 상호작용하는 지능형 시스템이다(Noh *et al.*, 2026). 이러한 환경에서 자재의 흐름을 효율적으로 제어하는 것은 생산 시스템 전반의 리드타임과 설비 가동률, 병목 발생 빈도, 재공(work-in-process; WIP) 수준 등에 직접적인 영향을 미친다. 특히 AGV(automated guided vehicle)나 AMR(automated mobile robot) 같은 AMHS(automated material handling system) 기반 이송 시스템에서는 자재의 이동 경로 결정 방식이 생산 효율성의 중요한 결정 요인이다(Monostori, 2014; Leitão *et al.*, 2016).

전통적인 제조실행시스템(manufacturing execution system; MES)은 중앙 서버에서 전체 설비 상태와 자재 정보를 통합 관리하며 이송 경로를 할당하는 중앙집중형 구조를 따른다. 이러한 구조는 시스템 구현이 명확하고 전역적 최적화(global optimization)가 가능하다는 강점이 있다. 하지만 시스템 규모가 커질수록 계산 부하가 급증하며, 중앙 서버의 결함이 전체 시스템에 영향을 미치는 단일 장애점(single point of failure) 위험을 안고 있다. 뿐만 아니라 설비 고장이나 우선순위 변경 등

이 논문은 국립한밭대학교 2025학년도 교내학술연구비의 지원을 받아 수행되었음.
† 연락저자 : 신문수 교수, 34158 대전광역시 유성구 동서대로 125(덕명동) 국립한밭대학교 산업경영공학과, Tel : 042-821-1758, Fax: 042-825-1671, E-mail: shinms@hanbat.ac.kr
2026년 3월 11일 접수; 2026년 4월 20일 게재 확정.

현장의 동적인 변화에 유연한 대응이 어렵다(Shin, 2013).

이러한 문제를 해결하기 위해 분산형 제어 구조가 대안으로 제시되고 있다(Van Brussel *et al.*, 1998; Ryu and Jung, 2003). 분산형 시스템에서는 각 설비와 작업물이 독립적인 에이전트로 동작하며, 내부 및 인접 정보에 기반하여 자율적인 의사결정을 수행한다. 이러한 구조는 확장성(scalability)과 견고성(robustness), 적응성(adaptability) 측면에서 장점이 있다(Shin, 2020). 특히 제조시스템의 사이버-물리 시스템(cyber-physical system; CPS)화가 진행됨에 따라, 에이전트 기반 분산형 제어는 스마트 제조 환경의 핵심 아키텍처로 자리 잡고 있다.

경로 결정 방식은 크게 정적(static) 접근과 동적(dynamic) 접근으로 구분된다. Dijkstra 알고리즘(Dijkstra, 1959)과 Bellman의 동적 계획법(Bellman, 1957), A* 알고리즘(Hart *et al.*, 1968)과 같은 정적 경로 탐색 기법은 사전에 정의된 비용 정보를 기반으로 최단 경로를 계산한다. 이러한 정적인 접근법은 계산 효율성과 구현 용이성 측면에서 장점이 있으나, 실시간으로 변화하는 혼잡도와 설비 고장 여부, 자재 간의 상호작용 등 다양한 환경의 변화 요인을 충분히 반영하기 어렵다. Koenig and Likhachev(2002)가 제안한 D* Lite와 같은 증분적 탐색 기법 또한 환경 변화에 따라 경로를 재계산할 수는 있으나, 기본적으로 단기적 최단 경로 재탐색에 초점을 두며 시스템 수준의 장기 성능 최적화를 보장하지는 않는다.

강화학습은 에이전트가 환경과의 상호작용을 통해 장기 누적 보상을 최대화하는 정책을 학습하는 방법론이며, 동적이고 불확실한 환경에서의 의사결정 문제에 적합하다(Sutton and Barto, 2018). 특히 심층 강화학습(deep reinforcement learning; DRL)은 신경망을 이용하여 고차원 상태 공간을 효과적으로 표현할 수 있어 복잡한 제조시스템 환경에 적용 가능성이 높다(Mnih *et al.*, 2015). 제조 및 물류 영역에서도 강화학습을 활용한 스케줄링 및 경로 최적화 연구가 점차 증가하고 있으며, 특히 멀티에이전트(multi-agent) 강화학습 기반 제조시스템 관련 연구가 많은 관심을 받고 있다(Oroojlooy and Hajinezhad, 2023; Zhang *et al.*, 2021).

기존의 강화학습 기반 경로 결정 연구는 주로 단일 로봇 혹은 소수 로봇의 이동 경로 최적화 문제에 초점을 두고 있다. 일반적으로 좌표 기반 내비게이션 또는 충돌 회피 중심으로 수행되어 왔다. 다수 로봇 환경을 대상으로 하는 연구에서는 중앙집중형 학습 구조를 채택하는 경우가 많아, 전형적인 분산형 물류 제어 문제와는 구조적으로 차이가 있다. 특히 개별 WIP 단위 에이전트가 스스로 경로를 선택하고, 그 결과가 시스템 전체 흐름에 영향을 미치는 MES 환경에서의 분산형 강화학습 라우팅 연구는 상대적으로 부족하다.

이에 본 연구는 심층 강화학습을 기반으로 하는 분산형 에이전트의 이송 라우팅 시스템을 제안한다. 특히 단위 WIP과 자재취급설비를 각각의 독립적인 에이전트로 정의하며, 각 WIP이 자율적으로 이송 경로를 선택한다. 평균 리드 타임 최소화를 목표로 하는 장기 보상 구조를 정의함으로써, 중앙 제어 없이도 자재 흐름의 자기조직화(self-organizing) 현상을 유도한다. 본 연구는 에이전트 기반 분산형 MES 프레임워크 내에서 강화학습 기반 동적 라우팅 메커니즘을 통합적으로 구현하였다는 점에서 학술적 의의를 가진다.

본 논문의 이후 구성은 다음과 같다. 제2장에서는 강화학습 기반 라우팅 관련 선행연구를 고찰하며, 제3장에서는 에이전트 기반 제조실행 프레임워크와 본 연구에서 다루는 라우팅 문제를 소개한다. 제4장에서는 심층 강화학습을 위한 MDP(Markov decision process) 모델을 정의하고, 제5장에서는 시뮬레이션 기반 실험 결과를 제시한다. 제6장에서는 결론과 향후 연구 방향을 설명한다.

2. 관련 연구

2.1 경로 결정 문제

경로 결정 문제는 그래프 이론에 기반한 최단 경로 탐색 문제로부터 발전해왔다. Bellman(1957)은 단계적 최적화 원리를 기반으로 동적 계획법을 제시하였으며, Dijkstra(1959)는 가중 그래프에서의 효율적 최단 경로 계산 알고리즘을 제안하였다. Hart *et al.*(1968)의 A* 알고리즘은 휴리스틱 함수를 도입하여 탐색 효율을 향상시켰다. 이러한 기법들은 고정된 비용 구조를 전제로 하며, 계산 안정성과 효율성 측면에서 강점을 가진다.

그러나 제조 환경과 같이 설비 상태와 혼잡도가 실시간으로 변화하는 시스템에서는 고정 비용 기반 최단 경로가 전체 시스템 성능을 보장하지 못한다. Koenig and Likhachev(2002)는 환경 변화에 대응하기 위한 증분적 재탐색 알고리즘(D* Lite)을 제안하였으나, 이는 단일 에이전트의 경로 수정 문제에 초점을 두며 다수 작업물 간 상호작용을 고려한 장기 최적화로 확장되지는 않았다.

강화학습은 상태-행동-보상 구조를 기반으로 장기 누적 보상을 최대화하는 정책을 학습함으로써, 동적 환경에서의 순차적 의사결정 문제를 해결할 수 있다(Sutton and Barto, 2018). 특히 심층 강화학습은 신경망을 활용하여 복잡한 상태 공간을 근사할 수 있어 대규모 제조시스템에도 적용 가능하다(Mnih *et al.*, 2015). 이러한 접근은 단순 최단 경로 계산이 아닌, 시스템 수준의 성능 지표를 직접적으로 최적화할 수 있다는 점에서 기존 경로 탐색 기법과 구별된다.

2.2 강화학습 기반 라우팅

강화학습 기반 경로 결정 기법은 제조시스템 분야에서 모바일 로봇의 내비게이션 문제에 널리 활용되고 있다. Park *et al.*(2019)은 스마트팩토리 환경에서 DQN(deep Q-network) 기반의 최적 경로 탐색 기법을 제안하였다. 단일 로봇의 경로 탐색 문제를 다루었으며, 물리적 공간에 대한 격자 기반의 상태

표현과 목표 도달 여부 중심의 보상 구조를 적용하였다. 또한 Kong and Lee(2025)는 SAC(soft actor critic) 알고리즘을 바탕으로 동적 장애물이 존재하는 환경에서 로봇의 이동 경로 탐색 문제를 다루었다. 이러한 연구는 강화학습 기법의 경로 탐색 문제로의 적용 가능성과 효용성을 제시하였으나, 문제 범위가 개별 단위 로봇의 이동 최적화에 한정되는 한계가 있다.

다수 로봇의 이동 경로를 다루기 위해서는 개별 로봇의 의사결정에 대한 조율(coordination)이 필요하다. 이에 따라 다수 로봇의 경로 탐색을 위한 중앙집중형 강화학습 구조가 활용되고 있다. Jeon *et al.*(2023)은 중앙 제어기가 시스템 내의 모든 로봇의 상태를 종합하여 행동을 결정하는 DQN 기반 경로 탐색 구조를 제안하였다. 이는 충돌 회피와 전체 이동 거리 최소화를 목표로 하는데, 시스템 규모의 확장과 상태 차원의 증가에 따라 학습 복잡도가 급격히 증가하는 문제를 안고 있다. 무엇보다 분산형 시스템 제어 구조에 적합하지 않다. Kang *et al.*(2022)은 Q-learning 기반 AMHS 경로 결정 알고리즘을 제안하였다. 정체 유발 차량 대수를 의미하는 혼잡도를 상태 변수에 포함함으로써 다수 로봇 환경을 고려하는 동적 경로 결정을 수행한다. 하지만 Q-learning의 테이블 기반 학습 구조로 인해 상태 공간 확장에 제약이 있다.

종합하면, 기존의 강화학습 기반 경로 탐색 연구는 단일 로봇의 이동 최적화 문제에 집중하거나, 다수 로봇에 대한 중앙집중형 제어를 기반으로 하는 한계가 있다. 개별 에이전트가 분산적으로 의사결정을 수행하고, 개별 에이전트 간의 상호작용이 시스템 전반의 운용 성능에 직접적으로 연계되는 분산형 제어 구조에 대한 연구는 미흡한 실정이다. 따라서 분산형 에이전트 기반 제조 환경을 대상으로 심층 강화학습을 활용한 경로 탐색 메커니즘에 대한 연구가 필요하다.

3. 멀티에이전트 기반 라우팅 아키텍처

3.1 제조실행 프레임워크

본 연구는 기존에 제안된 에이전트 기반 제조실행 프레임워크(Park and Shin, 2024)를 기반으로 한다(<Figure 1> 참조). 해당 프레임워크는 실행 계층(execution layer)과 제어 계층(control layer)으로 구성되며, 두 계층은 양방향 정보 교환을 통해 실시간 의사결정을 처리한다. 이는 분산형 의사결정 구조를 지원하기 위한 참조 아키텍처로 활용되며, 본 연구에서는 이를 확장하여 WIPAgent의 라우팅 의사결정 문제를 심층강화학습 기반으로 정식화하고 해결하는 데 초점을 둔다.

실행 계층은 다수의 자율 에이전트로 구성된 에이전트 기반 MES가 핵심이다. 대표적으로 WIP Agent(WIPAgent), Processing Agent(PAgent), Material Handling Agent(MHAgent)가 존재하며, 각 에이전트는 담당 객체의 상태 정보를 관리하고 의사결정을 수행한다. 이를 통해 전통적인 중앙집중형 MES와 달리, 의사결정이 에이전트 단위에서 수행됨으로써 분산 협업형 의사결정 구조를 갖는다. 각 에이전트의 주된 기능과 역할은 다음과 같다.

- WIPAgent: 개별 WIP에 대한 상태 정보를 관리하며, WIP의 입장에서 공정설비와 자재취급설비를 선택하고, 이송 경로를 결정한다. 본 연구에서 다루는 라우팅 문제는 WIP의 입장에서 최선의 이송 경로를 결정하는 것이다. 따라서 제안된 프레임워크 내에는 여러 유형의 에이전트가 존재하지만, 경로 결정 문제를 WIPAgent의 관점에서 정식화한다.
- PAgent: 개별 공정설비의 상태 정보를 관리하며, 공정설비 입장에서 의사결정에 참여한다. 처리할 WIP을 선택하며, WIPAgent나 다른 PAgent에 담당 설비의 현황 정보를

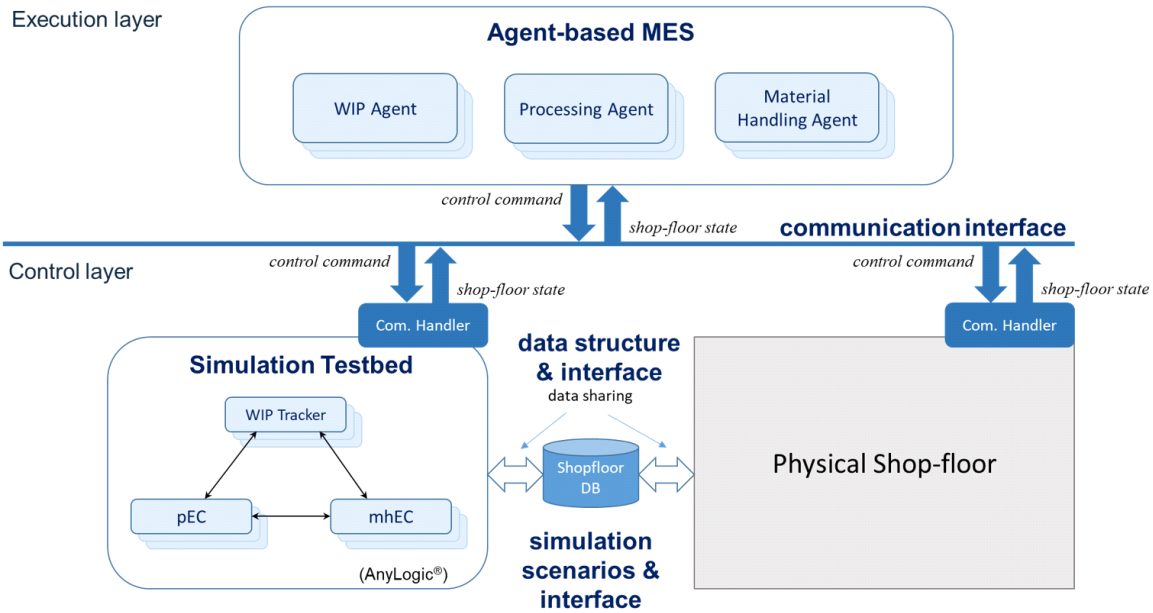

Figure 1. Agent-based Manufacturing Execution Framework(Park and Shin, 2024)

제공함으로써 원하는 WIP을 할당받기 위한 경쟁에 참여한다.

- MHAgent: 리프트나 AMR(autonomous mobile robot) 등의 자재취급설비의 상태 정보를 관리하며, 해당 설비 입장에서 의사결정에 참여한다. 본 연구에서는 WIP의 이송 경로 결정 과정에 각각의 MHAgent가 경로를 구성하는 개별적인 자재취급 노드 하나를 담당하여 참여한다.

제어 계층은 실제 제조 현장과 시뮬레이션 테스트베드로 구성된다. 시뮬레이션 테스트베드는 실제 공정과 동일한 데이터 구조 및 통신 인터페이스를 사용하여 디지털 트윈 환경을 구현하며, 현장 모니터링 및 각종 what-if 시뮬레이션 플랫폼으로서 역할을 수행한다. 특히 본 연구에서 제안하는 강화학습 기반 라우팅 로직의 학습 및 검증을 수행한다. 본 연구에서는 실행 계층 내 다수의 WIPAgent가 동시에 존재하는 멀티에이전트 환경을 고려한다. 각 WIPAgent는 자신의 이동 경로를 결정하는 독립적인 의사결정 주체로 동작한다.

3.2 라우팅 문제 정의

본 연구에서의 라우팅 문제는 다수의 노드로 구성된 유향 그래프 상에서 현재 위치로부터 다음 이동 노드를 결정하는 문제로 정의된다. 물류 네트워크는 다음과 같은 유향 그래프로 표현된다.

$$G=(V,E) \tag{1}$$

여기서 V는 컨베이어 분기점이나 워크스테이션, 층간 연결 지점 등으로 구성된 노드 집합이며, $E\subseteq V\times V$는 노드와 노드를 연결하는 유향 엣지 집합을 의미한다. 이때 시점 t에서 WIPAgent의 현재 위치는 $s_t\in V$로 정의된다.

현재 노드 s_t에서 선택이 가능한 엣지 집합은 다음과 같다.

$$E(s_t)=\{(s_t,v)\in E \mid v\in V\} \tag{2}$$

라우팅 의사결정 문제는 현재 노드에서 인접 엣지 중 하나를 선택하는 문제이다. 즉, 시점 t에서의 의사결정은 다음과 같이 나타낼 수 있으며,

$$a_t=(s_t,v_t)\in E(s_t) \tag{3}$$

이에 따라 다음 상태는 $s_{t+1}=v_t$로 전이된다.

라우팅 의사결정의 목적은 단일 이동의 최적화가 아니라, 목적지 $g\in V$에 도달할 때까지의 전체 이동 효율을 극대화하는 것이다. 이를 누적 보상 또는 누적 비용 관점에서 다음과 같이 표현할 수 있다.

$$\max_{\pi}\ \mathbf{E}\left[\sum_{t=0}^{T}\gamma^t r(s_t,a_t)\right] \tag{4}$$

여기서 π는 경로 선택 정책이며, $r(s_t,a_t)$는 선택된 경로에 의해 유발된 이동 시간이나 혼잡도 등을 반영한 보상 함수이다.

3.3 멀티에이전트 의사결정 및 실행 구조

본 연구에서 다루는 라우팅 의사결정은 제조시스템 내에서 동시에 존재하는 여러 WIP이 각자의 다음 이송 목적지를 결정하는 과정으로 전개된다. 시점 t에서 N개의 WIPAgent가 존재한다고 할 때, 각 에이전트 $i\in\{1,\cdots,N\}$는 자신의 관측 정보 o_i^t를 기반으로 행동 a_i^t를 선택한다.

$$a_i^t=\pi_\theta(o_i^t) \tag{5}$$

여기서, π_θ는 공유된 정책 함수를 의미하며, θ는 모든 에이전트가 공유하는 파라미터이다. 본 연구에서는 다수의 에이전트가 존재하지만, 모든 에이전트는 동일한 정책 파라미터 θ를 공유하는 파라미터 공유 구조를 따른다.

$$\theta_1=\theta_2=\ldots=\theta_N=\theta \tag{6}$$

따라서 학습은 하나의 공유 정책에 대해 수행되며, 실행은 각 에이전트가 독립적으로 수행한다. 파라미터 공유 구조는 다음과 같은 장점을 갖는다(Ma *et al.*, 2026; Ni *et al.*, 2023): 1) 표본 효율성 향상(sample efficiency), 2) 학습 안정성 증가, 3) 에이전트 수 증가 시 확장성 확보, 4) 독립 학습 시 발생하는 non-stationarity 완화.

또한 전체 시스템의 전역 상태를 S^t라 할 때, 각 에이전트는 전역 상태 전체를 관측하지 않는다. 대신 다음과 같이 부분적인 관측 정보만을 사용한다.

$$o_i^t\subset S^t \tag{7}$$

이는 부분 관측 가능성에 기반하는 분산 실행을 의미하며, 실제 제조 현장의 제약 조건을 반영한다. 예를 들어, 각 WIPAgent의 관측 정보는 현재 위치와 목적지, 인접 스테이션의 대기 WIP 수량과 잔여 작업 시간 등으로 구성된다. 각 WIPAgent는 직접 관리하는 지역 정보를 기본으로 하며, 주변 스테이션의 상태 정보는 해당 PAgent와의 제한적인 정보 교환을 통해 취득한다. 즉, 각 에이전트는 시스템의 전체 상태를 파악하지 않고도 지역 정보와 최소한의 공유 정보를 바탕으로 독립적인 의사결정을 수행한다.

<Figure 2>는 WIPAgent와 MHAgent 사이의 정보 공유 과정을 도식화한 것이다. 먼저 WIPAgent는 제조 현장 전반의 구조적인 정보를 관리하는 Fab Manager(FabMgr)로부터 현재 위치를 기준으로 접근 가능한 주변 노드에 관련한 접속 정보를 구

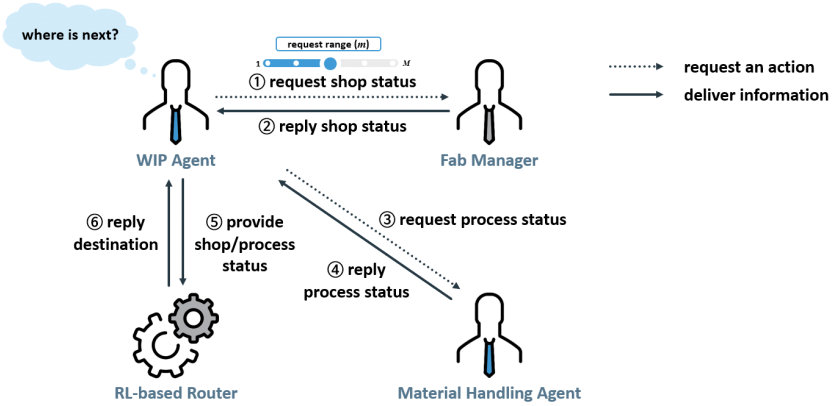

Figure 2. Agent-based Routing Decision Flow

한다. WIPAgent는 이를 바탕으로 각 노드에 해당하는 MHAgent로부터 실제 제조 현장의 상태 정보를 취득한다. 이때 WIPAgent가 접근 가능한 관측 정보의 범위는 인접 노드($m=1$) 만으로 제한되는 경우에서부터 현장 전역($m=M$)이 가능한 상황까지 다양하게 정의될 수 있다.

요약하면, 본 연구에서 제안하는 의사결정 체계는 공유 파라미터 기반의 학습과 국소적 관측 기반 독립 의사결정의 특성을 갖는 분산 협업형 구조로 이루어진다. 다음 장에서는 DQN 기반 학습 모델을 제시한다.

4. 심층 강화학습 기반 라우팅 모델

본 장에서는 에이전트 기반 제조실행 환경에서의 라우팅 문제를 멀티에이전트 MDP 관점에서 정의하고, 파라미터 공유 기반 DQN 학습 구조를 설명한다.

4.1 MDP 정의

본 연구의 라우팅 환경은 동시에 다수의 WIPAgent가 존재하는 멀티에이전트 환경으로 정의된다. 이를 다음과 같은 마르코프 게임 형태로 표현할 수 있다.

$$\text{MDP} = \left(S, \{A_i\}_{i=1}^{N}, P, R\right) \tag{8}$$

여기서, S는 전역 상태 공간을 의미하며, A_i는 i번째 에이전트의 행동 공간, 그리고 P와 R은 각각 상태 전이 확률과 보상 함수를 의미한다. 이때 각 WIPAgent i는 전역 상태 전체를 관측하지 않고, 자신의 로컬 관측 정보만을 이용하여 행동을 선택한다.

(1) 상태 공간 정의

전역 상태 S^t는 시스템 내 모든 WIP의 위치, 스테이션 상태, 대기 WIP 수, 잔여 작업 시간 등을 포함한다. 그러나 각 에이전트는 전역 상태 전체를 관측하지 않고, 다음과 같은 로컬 관측 정보만을 활용한다.

$$o_i^t = \left\{s_{1,i}^t,\ s_{2,i}^t,\ s_{3,i}^t,\ b_1^t,\ \ldots,\ b_m^t,\ c_1^t,\ \ldots,\ c_m^t\right\} \tag{9}$$

각 요소는 다음과 같이 정의된다.

- $s_{1,i}^t$: i번째 WIP의 현재 위치
- $s_{2,i}^t$: i번째 WIP의 이송 목적지
- $s_{3,i}^t$: i번째 WIP의 이송 목적지가 한 번의 단위 이송만으로 접근 가능한지 여부
- b_j^t: j번째 스테이션의 대기 WIP의 수
- c_j^t: j번째 스테이션의 잔여 작업 시간(정규화 값)

여기서 m은 WIPAgent i가 관측 가능한 스테이션 개수이다. 이와 같이 구성된 상태 벡터는 각 에이전트의 의사결정에 필요한 최소한의 정보만을 포함하며, 중앙집중식 전역 조정 메커니즘이 없어도 행동 선택이 가능하다. DQN은 신경망을 통해 상태-행동 가치함수를 근사하므로 연속 상태 공간을 직접 입력으로 사용할 수 있다. 이와 같은 연속 상태 표현은 스테이션의 혼잡 수준과 작업 진행 상황을 정밀하게 반영할 수 있으며, 고차원 상태 공간에서도 일반화된 정책 학습이 가능하다.

(2) Action Space

각 WIPAgent는 주어진 위치에서 이동 가능한 방향 중 하나를 선택한다. 행동 집합은 다음과 같이 정의된다:

$$A_i(s_t) = \{v \mid (s_t, v) \in E(s_t)\} \tag{10}$$

즉, WIPAgent i가 선택할 수 있는 행동은 시점 t에서의 상태에 따라 달라진다. 행동 선택은 공유 정책 π_θ에 의해 결정된다 (식 (5) 참조).

(3) Reward Design

본 연구의 목적은 평균 리드 타임(lead time) 및 총 작업 완료

시간(makespan)의 최소화다. 이를 위해 보상 함수는 에피소드 기반 과제(episodic task)에서의 안정적인 학습을 유도하기 위한 종단 보상(terminal reward)과 경로 선택 과정에서의 점진적 개선을 반영하는 밀집 패널티(dense penalty)의 결합 형태로 정의된다(식 (11) 참조). 특히 목적지 도달 시 제공되는 +1의 보상은 희소 보상(sparse reward) 환경에서의 학습 안정성을 확보하기 위한 것이며, 이동 시간 기반 패널티는 각 의사결정 단계에서의 상대적 비효율성을 반영한다. 이동 시간 패널티는 식 (12)와 같이 정규화함으로써 종단 보상과의 스케일 균형을 유지한다.

$$r_i^t = \begin{cases} 1 & \text{if reaches destination} \\ -f_{norm} & \text{otherwise} \end{cases} \tag{11}$$

$$f_{norm} = \frac{f_{ij}}{f_{ij}^{\max}} \tag{12}$$

여기서, f_{ij}는 직전 의사결정 이후 발생한 스테이션 i와 j 사이의 이동 시간을 의미하며, $f_{ij}^{\max}$는 관측된 최대 이동 시간을 의미한다. 이러한 보상 구조는 다음 특성을 갖는다: 1) 불필요한 경유 최소화, 2) 목적지 도달 장려, 3) 이동 효율성 반영. 각 에이전트는 개별적으로 보상을 수신하며, 전체 시스템의 평균 성능은 모든 에이전트의 누적 보상에 의해 간접적으로 반영된다.

4.2 학습 메커니즘

학습 단계에서는 모든 WIPAgent로부터 수집된 경험을 단일 리플레이 버퍼에 저장한다. 각 상태 전이는 다음과 같은 형태를 갖는다:

$$o_i^t \xrightarrow{a_i^t} \left(r_i^t, o_i^{t+1}\right) \tag{13}$$

DQN의 목표 Q-value는 다음과 같이 정의된다.

$$y = r_i^t + \gamma \max_{a'} Q\left(o_i^{t+1}, a'; \theta^-\right) \tag{14}$$

손실 함수는 다음과 같다.

$$L(\theta) = \mathbb{E}\left[\left(y - Q\left(o_i^t, a_i^t; \theta\right)\right)^2\right] \tag{15}$$

정책 파라미터는 다음과 같이 업데이트된다.

$$\theta \leftarrow \theta - \alpha \nabla_\theta L(\theta) \tag{16}$$

파라미터 공유 구조를 통해 모든 WIPAgent의 경험이 단일 신경망 학습에 활용되므로, 표본 효율성과 학습 안정성이 향상된다.

5. 시뮬레이션 실험

5.1 실험 환경 구성

본 연구에서는 국립한밭대학교의 스마트팩토리 테스트베드(HBSF)를 대상으로 에이전트 기반 라우팅 시스템을 구현하고, 제안된 심층 강화학습 기반 라우팅 모델의 성능을 검증한다. HBSF는 <Figure 3>에 나타난 바와 같이 비동기식 복층 컨베이어 시스템이며, 입·출고 스테이션과 층별 7개의 컨베이어, 그리고 6개의 리프트로 구성된다. 각 컨베이어는 독립적으로 동작하며, 리프트를 통해 WIP의 상·하층 간 이동이 가능하다. 특히 본 연구에서는 상층에 5개의 가상 워크스테이션을 추가함으로써 다양한 경로 결정 상황이 발생하는 라우팅 문제를 구성하였다. 각 WIP은 품목별 공정 순서에 따라 공정별 워크스테이션을 경유한다.

본 연구에서는 에이전트 기반 시뮬레이션 도구인 AnyLogic™ (The AnyLogic Company, 2026)을 활용하여 시뮬레이션 테스트베드를 구축하였다. <Figure 4>는 개별 WIP의 행동 모형을

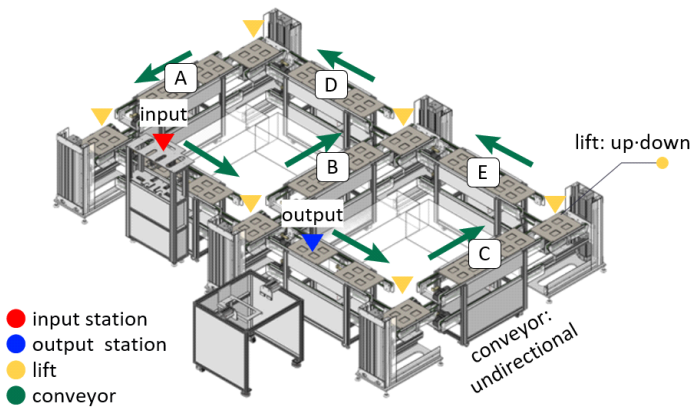

Figure 3. Hanbat Smart Factory Testbed(HBSF)

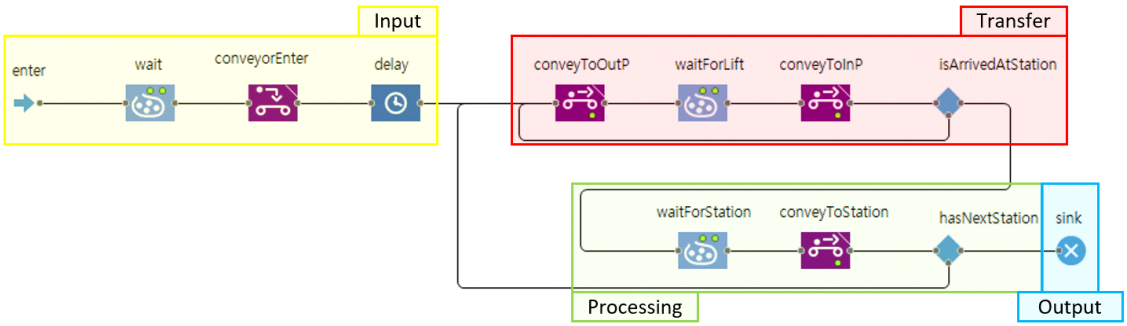

Figure 4. Workflow-based Behavioral Model of an Individual WIPAgent

Table 1. Summary of Key Building Blocks in the Simulation Model

Block Name	Block Type	Description
enter	Enter	WIP enters the system.
wait	Wait	WIP waits for being loaded onto entryway conveyor.
conveyorEnter	ConveyorEnter	WIP is loaded onto the entryway conveyor.
delay	Delay	WIP is processed at the entry station.
conveyToOutP	Convey	WIP moves to the exit position of the conveyor.
waitForLift	Wait	WIP waits for a lift.
conveyToInP	Convey	WIP moves to the entry position of the next conveyor.
isArrivedAtStation	SelectOutput	If there is the target station on the conveyor where the WIP arrives, go to the *waitForStation* block; otherwise, go to the *conveyToOutP* block.
waitForStation	Wait	WIP waits for the target station to become available.
conveyToStation	Convey	WIP moves to the target station.
hasNextStation	SelectOutput	If there is no more operation for the WIP, go to the *sink* block; otherwise, go to the *conveyToOutP* block.
sink	Sink	WIP is finalized.

작업 흐름의 형태로 모델링한 것이다. 전반적인 작업 흐름은 작업물의 투입부터 공정설비로의 이송과 공정 수행의 반복, 그리고 출고의 일련의 과정으로 구성된다. <Table 1>은 해당 시뮬레이션 모델의 주요 구성 블록과 그 기능을 요약하여 제시하고 있다. 시뮬레이션 환경 사양은 Intel i5-12400 CPU 및 16GB RAM으로 구성된다.

5.2 비교 대상 알고리즘

(1) Dijkstra 알고리즘

Dijkstra 알고리즘은 링크별 가중치를 기반으로 단일 최단경로를 계산하는 결정론적 방법이다. 각 노드까지의 이동 거리는 다음과 같은 경로 비용 갱신 식을 따른다.

$$d(j) \leftarrow \min(d(j), d(i) + w(i,j)), \ \forall (i,j) \in E \quad (17)$$

여기서 $d(j)$는 출발 노드 s에서 목적지 노드 j까지의 최단 이송 시간 추정값이며, $w(i,j)$는 링크 (i,j)에 대한 가중치이다. 본 연구에서는 편의를 위해 모든 링크의 가중치를 다음과 같이 정의함으로써 컨베이어 간 이동 횟수가 최소가 되는 경로를 최단 경로로 정의한다.

$$w(i,j) = N_{cnv}(i,j), \ \forall (i,j) \in E \quad (18)$$

이때 $N_{cnv(i,j)}$는 i에서 j로 이동하는 동안 거치는 컨베이어 구간의 수이다. 이 방법은 계산 효율이 높고 구현이 단순하다는 장점이 있으나, 실시간 혼잡 변화나 스테이션 상태 변동을 동적으로 반영하지 못한다는 한계를 가진다.

(2) 개선된 Dijkstra 알고리즘

개선된 Dijkstra 알고리즘은 각 링크의 가중치에 이동 시간과 혼잡도를 함께 결합하여 산정한다. 특히 동적으로 변화하는 스테이션의 상태를 반영하기 위해 링크 가중치를 다음과 같이 시간 t의 함수로 정의한다.

$$w(i,j,t) = 30\,N_{cnv(i,j)} + 5\,N_{lift(i,j)} + w_b(i,t) + w_c(i,t) \quad (19)$$

여기서 $N_{lift}(i,j)$는 리프트 사용 횟수이며, $w_b(i,t)$와

$w_c(i,t)$는 일종의 혼잡도 지수이다. 이는 각각 시점 t에 컨베이어 i 위에 위치한 스테이션에 대기 중인 WIP의 수($b_i(t)$)와 잔여 작업 시간($c_i(t)$)으로 인해 발생하는 페널티를 의미하며, 다음과 같이 정의한다.

$$w_b(i,t) = \begin{cases} 10, & b_i(t) = 0 \\ 20, & b_i(t) = 1 \\ 40, & b_i(t) \geq 2 \end{cases} \quad (20)$$

$$w_c(i,t) = \begin{cases} \mathrm{round}(c_i(t)), & c_{i(t)} \leq 20 \\ 50, & c_{i(t)} > 20 \end{cases} \quad (21)$$

따라서 개선된 Dijkstra 알고리즘의 경로 비용 갱신 식은 다음과 같다.

$$d(j) \leftarrow \min(d(j), d(i) + w(i,j,t)), \ \forall (i,j) \in E \quad (22)$$

이를 통해 기본 Dijkstra 대비 환경 정보를 부분적으로 고려할 수 있으나, 가중치 산정 방식은 사전에 정의되어 있으며 학습을 통해 자동으로 조정되지 않는다. 따라서 환경 변화에 대한 적응성은 제한적이다. 본 연구에서 적용된 가중치 계수 값은 테스트베드 환경에서 관측된 상대적인 이송 지연 특성을 반영하도록 경험적으로 설정하였다.

(3) Q-learning

Q-learning은 상태-행동 가치를 테이블 형태로 저장하는 강화학습 기반 접근이다. 따라서 상태 공간의 이산화가 필요한데, 본 연구에서는 다음과 같은 상태 정보를 적용한다.

$$o_i^t = \left\{ s_{1,i}^t, s_{2,i}^t, s_{3,i}^t, d_1^t, ..., d_m^t \right\} \quad (23)$$

여기서 d_i^t는 스테이션별 상태 변수(b_j^t와 c_j^t)를 다음과 같이 범주화한 것이다.

$$d_j^t = \begin{cases} 0 & \text{if } c_j^t \geq k \text{ and } b_j^t < l \\ 1 & \text{if } c_j^t \geq k \text{ and } b_j^t \geq l \\ 2 & \text{if } c_j^t < k \text{ and } b_j^t < l \\ 3 & \text{if } c_j^t < k \text{ and } b_j^t \geq l \end{cases} \quad (24)$$

본 연구에서는 k=20, l=1로 설정하였다. 각 WIP의 현재 위치 변수($s_{1,i}^t$)와 목적지 관련 변수($s_{2,i}^t$, $s_{3,i}^t$)는 제안 모델과 동일하게 유지하며, 행동 공간과 보상 체계 또한 제안 모델과 동일하게 유지한다. 모든 에이전트는 동일한 Q-table을 공유하며, ε-greedy 정책을 사용하여 행동을 선택한다. 공유 Q-table은 다음과 같이 학습한다.

$$Q(o_i^t, a_i^t) \leftarrow Q(o_i^t, a_i^t) + \alpha \left[r_i^t + \gamma \max_a Q(o_i^{t+1}, a) - Q(o_i^t, a_i^t) \right] \quad (25)$$

이러한 Q-learning은 환경 상호작용을 통해 정책을 개선할 수 있다는 점에서 결정론적 방법과 구별되나, 이산화된 상태 표현을 사용하므로 상태 차원이 증가하는 경우 확장성이 제한된다. 본 연구에서는 이러한 테이블 기반 접근을 기준선으로 설정하여, 연속 상태 표현과 함수 근사 기반 학습을 적용한 DQN 모델의 효과를 비교 분석한다.

5.3 DQN 학습 수행 및 분석

(1) 학습 시나리오

본 연구에 적용된 학습 시나리오는 2종의 제품에 대한 4개의 주문 목록을 기반으로 구성된다. 각 제품별 작업 순서와 각 공정별 처리시간은 모두 다르다. 학습 과정에서 각각의 에피소드마다 4개의 주문 목록을 무작위 셔플 방식으로 모두 순회함으로써 일반화 성능을 제고한다. 주문 목록의 적용 순서가 동일할 경우 적용 순서에 편향된 학습이 이루어질 수 있다. 학습 초기에는 모든 WIPAgent가 무작위 경로를 선택하도록 설정하며, 불량이나 설비 고장 등 예외 상황은 고려하지 않는다.

<Table 2>는 본 연구에서 사용한 강화학습 알고리즘의 주요 하이퍼파라미터 설정값을 나타낸다. 제안된 DQN 모델의 경우, 학습률(learning rate)은 0.0005, 감가율(discount factor)은 0.99, 배치 크기(batch size)는 128로 설정하였으며, [128, 128] 구조의 심층 신경망을 구성하였다. 또한, 학습의 안정성을 위해 리플레이 버퍼(replay buffer) 크기는 50,000으로 지정하고, 타깃 네트워크(target network)는 매 300 스텝마다 갱신하도록 설계하였다. 탐색률(exploration rate)은 초기값 0.9에서 시작하여 학습 진행에 따라 점진적으로 감소시켰으며, 최대 에피소드 수는 200회로 제한하였다. 비교군인 Q-learning 기반 모델은 학습률 0.1, 감가율 1.0, 탐색률 0.9를 적용하였다. 두 모델 간 학습률과 감가율의 차이는 각 알고리즘의 구조적 특성에 따른 안정적 학습을 보장하기 위해 개별적으로 조정된 결과이다. DQN의 경우 신경망 기반 함수 근사를 사용하므로 작은 학습률이 요구되는 반면, Q-learning은 테이블 기반 업데이트 구조로 상대적으로 큰 학습률이 적용 가능하다. 본 연구에서는 각 알고리즘에 대해 안정적인 성능을 기준으로 하이퍼파라미터를 사전 튜닝하였으며, 이는 단순 동일 조건 비교가 아닌 각 알고리즘의 최적 성능 기준 비교를 수행하기 위함이다.

Table 2. Hyper-parameters for DQN and Q-learning Algorithms

Category	Hyper-parameters	DQN(Proposed)	Q-learning
Model specific	Learning rate (α)	0.0005	0.1
	Discount factor (γ)	0.99	1.0
	Network architecture	[128,128]	N/A
Stability	Replay buffer size	50,000	N/A
	Target update freq.	300 steps	N/A
	Batch size	128	N/A
Common	Exploration rate (ϵ)	0.9	0.9
	Max episodes	200	200

(2) 학습 결과

<Figure 5>는 제시된 하이퍼파라미터 설정 하에서 DQN 모델의 학습 곡선을 보이고 있다. 에피소드가 진행됨에 따라 평균 리드 타임과 평균 총 작업 완료 시간 모두 점진적으로 감소하며 일정 수준에 수렴하는 경향을 확인할 수 있다. 학습 초기에는 무작위 경로 선택의 비중이 높아 비효율적인 경로를 선택했으나, 에피소드가 진행될수록 모델이 보상 구조를 학습함으로써 이송 효율이 개선된 것으로 판단할 수 있다. 대략 100회 이후에는 평균 리드 타임과 총 작업 완료 시간이 모두 안정적으로 유지되고 있다.

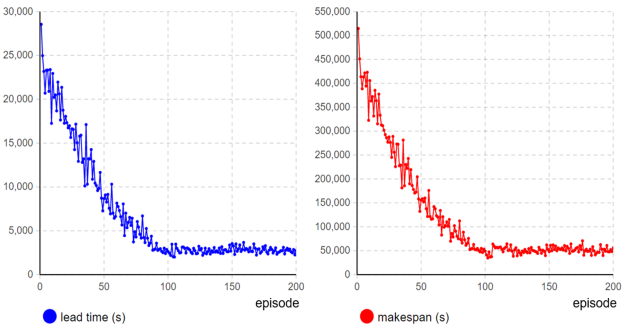

Figure 5. Training Result

5.4 실험 결과 및 분석

본 절에서는 제안하는 DQN 기반 라우팅 모델의 성능을 기존 방법들과 비교 분석한다. 비교 대상은 Dijkstra 알고리즘과 개선된 Dijkstra 알고리즘, 그리고 Q-learning이며, 모든 방법은 동일한 실험 환경과 주문 시나리오에서 평가되었다.

(1) 성능 지표 기반 비교

<Table 3>은 서로 다른 3건의 주문 목록에 대해 알고리즘별로 각각 100회의 시뮬레이션을 수행한 결과다. 각 알고리즘의 성능 평가는 독립적으로 생성된 난수 시드(random seed)를 기반으로 수행되었으며, 독립 표본 t-검정을 통해 통계적 유의성을 검증하였다. 특히 집단 간 분산이 동일하다는 가정을 배제하기 위해 웰치의 t-검정(Welch's t-test)을 적용하였다.

Dijkstra 알고리즘은 평균 리드 타임과 총 작업 완료 시간 모두에서 가장 열위의 성능을 보였다. 이는 링크 가중치를 고정값으로 설정하는 결정론적 접근이 실시간 혼잡 변화를 충분히 반영하지 못하기 때문이다. Q-learning은 Dijkstra 대비 큰 폭의 성능 개선을 보였다. 이는 환경과의 상호작용을 통해 정책을 개선할 수 있는 강화학습 접근의 장점을 보여준다. 그러나 상태 공간을 이산화하여 처리하기 때문에 복잡한 혼잡 패턴을 정밀하게 반영하는 데에는 한계가 존재한다.

제안된 DQN 모델은 Q-learning 대비 추가적인 성능 향상을 보였다. 특히 주문 목록 1과 2에 대해서는 리드 타임과 총 작업 완료 시간 모두에서 유의미한 개선이 확인되었다(<Table 4> 참조). 효과 크기($|d| \approx 0.42 \sim 0.58$)는 Q-learning 대비 DQN의 실무적 성능 향상이 중간 정도 수준임을 나타낸다. 주문 목록 3의 경우에도 통계적인 유의성은 확인되지 않았으나 평균치의 개선은 확인할 수 있다. 이는 연속 상태 표현과 신경망 기반 함수 근사를 통해 혼잡 및 작업 상태의 미세한 변화를 반영할 수 있기 때문으로 해석된다. 동일한 멀티에이전트 실행 구조 하에서 상태 표현 방식과 가치 함수 표현 방식의 차이가 성능 차이로 이어짐을 확인할 수 있다. 주문 목록 3에서 통계적 유의성이 나타나지 않은 것은 해당 시나리오에서 혼잡도 변동성이 감소했기 때문인 것으로 풀이된다.

Table 3. Simulation Result (# of runs = 100)

Order list	Routing rule	Lead time (Sec) (average)	Makespan (Sec) (average)
order list 1	Dijkstra	13,439	88,535
	Q-learning	4,715	31,742
	DQN	3,441	25,378
	improved Dijkstra	3,312	22,477
order list 2	Dijkstra	11,198	75,068
	Q-learning	3,879	26,922
	DQN	2,666	20,305
	improved Dijkstra	2,662	18,862
order list 3	Dijkstra	13,461	88,528
	Q-learning	3,523	24,576
	DQN	3,317	24,271
	improved Dijkstra	3,481	23,441
average	Dijkstra	12,699	84,044
	Q-learning	4,039	27,746
	DQN	3,141	23,318
	improved Dijkstra	3,152	21,593

Table 4. Independent Sample *t*-test between DQN vs. Q-learning

Metric	Order list	*t*-statistic	*p*-value	Cohen's *d*	Sig.
Lead Time	order list 1	-3.6317	0.0004	-0.5136	***
	order list 2	-4.1264	0.0001	-0.5836	***
	order list 3	-0.9953	0.3208	-0.1408	n.s.
Makespan	order list 1	-3.0081	0.0032	-0.4254	**
	order list 2	-3.5414	0.0005	-0.5008	***
	order list 3	-0.2396	0.8109	-0.0339	n.s.

Note: *p < 0.05, **p < 0.01, ***p < 0.001; n.s. = not significant. The results are based on 100 independent simulation runs per group.

Table 5. Independent Sample *t*-test between DQN vs. improved Dijkstra

Metric	Order list	*t*-statistic	*p*-value	Cohen's *d*	Sig.
Lead Time	order list 1	0.8382	0.4029	0.1185	n.s.
	order list 2	0.2184	0.8273	-0.0309	n.s.
	order list 3	-0.9723	0.3322	-0.1375	n.s.
Makespan	order list 1	3.0152	0.0029	0.4264	**
	order list 2	3.2639	0.0013	-0.4616	**
	order list 3	0.7963	0.4270	0.1126	n.s.

Note: *p < 0.05, **p < 0.01, ***p < 0.001; n.s. = not significant. The results are based on 100 independent simulation runs per group.

개선된 Dijkstra 알고리즘은 총 작업 완료 시간 측면에서 가장 우수한 결과를 보였다. 특히 주문 목록 1과 2에 대해서는 통계적인 유의성도 확인되었다(<Table 5> 참조). 리드 타임의 경우에는 통계적인 유의성은 확인되지 않았지만 주문 목록 1과 2에 대해서는 평균값이 개선되었다. 이는 사전에 설계된 가중치 조합이 해당 실험 환경에 최적화되어 있기 때문이다. 그러나 이러한 방법은 환경 변화에 따라 가중치 재설정이 필요하며, 확장된 상태 차원에 대한 일반화 능력은 제한적이다.

또한 개선된 Dijkstra 알고리즘은 최단 경로 산출 과정에서 동적인 현장 상황을 반영하기 위해 네트워크 전체에 대한 재계산을 필요로 한다. 반면 충분한 학습이 완료된 DQN 모델은 현재의 상태 벡터를 입력받아 순방향 추론(forward inference)을 수행함으로써 낮은 연산 비용으로 즉각적인 의사결정이 가능하다. 실제로 제안된 DQN 모델의 경로 탐색 의사결정에 소요된 평균 연산 시간은 약 267ns로 측정되었으며, 개선된 Dijkstra 알고리즘은 약 7,170ns의 연산 시간이 소요되었다(<Figure 6> 참조). 이러한 차이는 DQN이 매 의사결정 단계에서 전체 네트워크를 탐색하지 않고, 학습된 정책을 통해 즉시 행동을 선택하기 때문에 발생한다. 하지만, DQN 모델은 학습 과정에서 상당한 계산 비용이 요구되는 반면, Dijkstra 기반 방법은 별도의 학습 과정이 필요하지 않다.

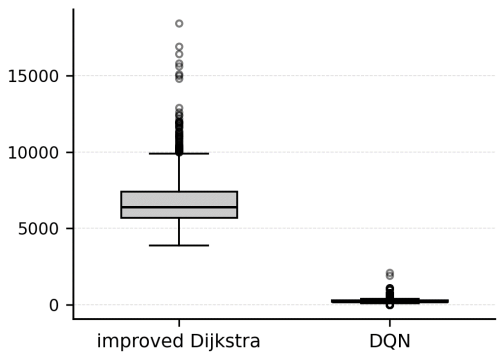

Figure 6. Execution Time(unit: ns; # of runs = 100)

(2) 종합 분석 및 토의

제안된 DQN 기반 라우팅 모델은 평균 리드 타임과 총 작업 완료 시간 측면에서 Q-learning 대비 통계적으로 유의미한 성능 개선을 보였으며(<Table 3>~<Table 4> 참조), 동시에 의사결정에 소요되는 평균 연산 시간이 개선된 Dijkstra 알고리즘 대비 현저히 낮게 나타났다(<Figure 6> 참조). 이러한 결과는 제안된 방법이 성능과 계산 효율성 측면에서 균형 잡힌 특성을 갖는다는 점을 시사한다. DQN 모델이 Q-learning 대비 성능 개선을 보인 것은 연속 상태 표현과 신경망 기반 함수 근사를 통해 보다 세밀한 시스템 상태 반영이 가능하기 때문으로 해석된다. 개선된 Dijkstra 알고리즘은 상대적으로 우수한 성능을 보이고 있지만, 이는 사전 설계된 가중치 조합에 의존한 결과이며, 주문 패턴이나 시스템 상태가 변화하면 가중치 재설계가 필요하다. 또한 시스템의 동적인 환경 변화를 반영하기 위해 네트워크 전반에 대한 반복적인 재계산이 필요하다. 반면에 DQN 모델은 연속 상태 표현과 학습 기반 정책을 통해 다양한 환경 변화에 대응할 수 있는 구조를 가지며, 실험 결과는 이러한 접근이 동적 환경에서의 적응 가능성을 시사한다. 다만, 일반화 성능에 대한 정량적 검증은 향후 다양한 시나리오를 통해 추가적으로 수행될 필요가 있다. 특히 본 연구에서 도

입한 파라미터 공유 기반 분산 실행 구조는 에이전트 수가 증가하더라도 중앙집중형 경로 재계산에 비해 안정적인 대응이 가능하다. 이는 스마트 제조 환경에서 자율적이고 적응적인 분산 의사결정 구조의 가능성을 제시한다는 점에서 학술적·실무적 의의를 가진다.

6. 결 론

본 연구는 에이전트 기반 제조실행 환경에서의 라우팅 문제를 유향 그래프 기반 의사결정 문제로 정식화하고, 파라미터 공유 구조를 적용한 DQN 기반 멀티에이전트 라우팅 모델을 제안하였다. 제안된 모델은 각각의 에이전트가 독립적으로 판단하고 행동하는 분산형 의사결정 및 실행 구조를 보인다. 즉, 전지적 관점의 전역 상태 정보를 가정하지 않으며, 개별 에이전트가 직접 관측할 수 있는 제한적인 정보와 주변 에이전트와의 소통을 통해 수집한 정보의 조합에 기반하여 분산적으로 의사결정을 수행한다.

제안된 모델은 시뮬레이션 실험을 통해 Dijkstra 및 개선된 Dijkstra 알고리즘, 그리고 Q-learning 모델과 성능을 비교하였다. 각각 100회의 독립 반복 실험을 수행하였으며, 통계적 검정 결과 제안된 DQN 모델이 Q-learning 모델 대비 평균 리드타임과 총 작업 완료 시간 모두에서 유의한 수준으로 성능을 개선하였다. 이는 DQN 모델이 연속적인 상태 표현과 신경망 기반의 정책 함수 근사를 통해 이산적인 상태 표현에 기반하는 Q-learning 모델에 비해 정교한 상태 표현과 정책 일반화에 유리하기 때문이다. 개선된 Dijkstra 알고리즘은 평균 리드 타임과 총 작업 완료 시간 측면 모두에서 상대적으로 우수한 성능을 보였으나, 사전 가중치 설정에 의존적인 특성으로 인해 환경 변화에 대한 적응성에 한계가 있다. 또한 동적인 환경 변화를 반영하기 위해서는 각각의 의사결정 시점마다 네트워크 전반에 대한 재계산이 필요하다. 반면, 제안된 DQN 모델은 충분한 학습을 전제로 변화에 대한 적응성과 실시간성을 확보하고 있으며, 파라미터 공유 기반 분산형 의사결정을 통해 안정적인 확장성 또한 갖추고 있다.

본 연구는 제한된 규모의 주문 시나리오를 대상으로 수행되었다는 한계를 갖는다. 이는 제안된 방법론의 구조적 타당성 검증에 필요한 수준으로서 실제 제조 환경의 복잡성을 충분히 반영하지 못한다. 따라서, 다양한 주문 패턴 및 대규모 WIP 환경에서의 일반화 성능 검증이 제한된다. 향후 연구에서는 보다 다양한 시나리오와 확률적 환경을 고려한 확장적 검증이 필요하다. 또한 본 연구에서는 혼잡도를 상태 변수(대기 WIP 수, 잔여 작업 시간)에 간접적으로 반영하는 방식으로 고려하였으나, 향후 연구에서는 혼잡 수준을 보상 함수에 직접 반영하는 명시적 혼잡 패널티 도입을 검토할 필요가 있다. 이를 통해 보다 효과적인 경로 분산 및 시스템 전반의 흐름 최적화를 유도할 수 있을 것으로 기대된다.

참고문헌

Bellman, R. (1957), *Dynamic Programming*, Princeton University Press, Princeton, NJ.

Dijkstra, E. W. (1959), A Note on Two Problems in Connexion with Graphs, *Numerische Mathematik*, **1**, 269-271.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968), A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Transactions on Systems Science and Cybernetics*, **4**(2), 100-107.

Jeon, J., Cho, H., Lee, S., and Lee, H. (2023), Centralized DQN-based Optimal Path Control of Multiple Robots, *The Journal of Korean Institute of Communications and Information Sciences*, **48**(4), 449-456.

Kang, B., Kang, B. M., and Hong, S. (2022), A Dynamic OHT Routing Algorithm in Automated Material Handling Systems, *Journal of Society of Korea Industrial and Systems Engineering*, **45**(3), 40-48.

Koenig, S. and Likhachev, M. (2002), D* Lite, *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, 476-483.

Kong, S. J. and Lee, W. C. (2025), Deep Reinforcement Learning for Indoor Autonomous Navigation of Mobile Robot, *The Journal of the Institute of Korean Electrical and Electronics Engineers*, **29**(2), 254-261.

Leitão, P., Karnouskos, S., Ribeiro, L., Lee, J., Strasser, T., and Colombo, A. W. (2016), Smart Agents in Industrial Cyber-Physical Systems, *Proceedings of the IEEE*, **104**(5), 1086-1101.

Ma, L., Liu, Y., Liu, Y., Ma, C., and Wang, S. (2026), Coordinated Multi-Intersection Traffic Signal Control Using a Policy-Regulated Deep Q-Network, *Sustainability*, **18**(3), 1510.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015), Human-level Control Through Deep Reinforcement Learning, *Nature*, **518**, 529-533.

Monostori, L. (2014), Cyber-Physical Production Systems: Roots, Expectations and R&D Challenges, *Procedia CIRP*, **17**, 9-13.

Ni, W., Li, C., Wang, P., and Li, Z. (2023), Traffic signal optimization at T-shaped intersections based on deep Q networks, *Proceedings of the International Conference on Neural Information Processing*, 288-299.

Noh, S., Kim, D., Ryu, K., Baek, S., Shin, M., Lee, J., and Chang, T. (2026), *Smart Manufacturing*, Gyomoonsa, Paju-si, Korea.

Oroojlooy, A. and Hajinezhad, D. (2023), A Review of Cooperative Multi-agent Deep Reinforcement Learning, *Applied Intelligence*, **53**(11), 13677-13722.

Park, J. M. and Shin, M. (2024), Manufacturing Execution Framework Based on Agent-to-Agent Collaboration, *Journal of Society of Korea Industrial and Systems Engineering*, **47**(4), 120-131.

Park, K. S., Park, J. M., Yun, W. K., and Yoo, S. (2019), DQN Reinforcement Learning: The Robot's Optimum Path Navigation in Dynamic Environments for Smart Factory, *The Journal of Korean Institute of Communications and Information Sciences*, **44**(12), 2269-2279.

Shin, M. (2013), Employment Contract-based Management Model of Production Resources on Relation-driven Fractal Organization, *Journal of the Korean Institute of Industrial Engineers*, **39**(4), 278-289.

Shin, M. (2020), Employment Contract-based Self-organizing Mechanism

for Production Resources, *Journal of the Korean Institute of Industrial Engineers*, **46**(3), 282-295.
Sutton, R. S. and Barto, A. G. (2018), *Reinforcement Learning: An Introduction* (2nd Edition), MIT Press, Cambridge, MA.
The AnyLogic Company (2026), AnyLogic Simulation Software, https://www.anylogic.com (Accessed February 28, 2026).
Ryu, K. and Jung, M. (2003), Agent-based Fractal Architecture and Modelling for Developing Distributed Manufacturing Systems, *International Journal of Production Research*, **41**(17), 4233-4255.
Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., and Peeters, P. (1998), Reference Architecture for Holonic Manufacturing Systems : PROSA, *Computers in Industry*, **37**(3), 255-274.
Zhang, K., Yang, Z., and Başar, T. (2021), Multi-agent Reinforcement Learning: A Selective Overview of Theories and Algorithms, *Handbook of Reinforcement Learning and Control*, 321-384.

저자소개

백종호: 국립한밭대학교 산업경영공학과에서 2025년 학사학위를 취득하고, 동대학원에서 석사과정에 재학 중이다. 연구분야는 에이전트 기반 시뮬레이션 모델링, 스마트팩토리 운영 최적화이다.

신문수: 포항공과대학교 산업경영공학과에서 2000년 학사, 2002년 석사, 그리고 2008년에 박사학위를 취득하였다. 삼성전자 디스플레이 사업부 책임연구원을 역임하고, 2011년부터 국립한밭대학교 산업경영공학과 교수로 재직하고 있다. 주요 연구 분야는 스마트팩토리 운영 최적화, 분산생산자원의 동적 조직화, M2M 커뮤니케이션이다.