Journal of the Korean Institute of Industrial Engineers
Vol. 52, No. 2, pp. 95-105, April 2026.
ISSN 1225-0988 | EISSN 2234-6457
https://doi.org/10.7232/JKIIE.2026.52.2.095

조선소 중일정 계획의 부하 평준화를 위한 2단계 최적화 알고리즘 개발

이 홍[1] · 우종훈[1,2†]

[1]서울대학교 조선해양공학과 , [2]서울대학교 해양시스템공학연구소

Two Stage Optimization Algorithm for Resource Leveling Problem

Hong Lee[1] · Jong Hun Woo[1,2†]

[1]Department of Naval Architecture and Ocean Engineering, Seoul National University
[2]Research Institute of Marine Systems Engineering, Seoul National University

Resource leveling in shipyard mid-term scheduling is critical for operational efficiency, yet existing approaches face significant limitations. Constraint programming methods suffer from computational scalability issues, while Meta Heuristic algorithms show restricted search space exploration. This study proposes a two-stage optimization framework integrating heuristic sequencing with reinforcement learning. In Stage 1, an Enhanced Kahn's Algorithm generates efficient activity sequences by incorporating resource requirements and durations. In Stage 2, Q-Learning dynamically determines optimal start times through episodic learning, progressively minimizing resource variance. Experimental evaluations using actual shipyard mid-term scheduling data demonstrate that the proposed algorithm achieves superior performance in large-scale problems compared to constraint programming and Meta Heuristic approaches. Results indicate that heuristic-based sequencing provides substantially improved computational efficiency, while reinforcement learning achieves rapid convergence and superior outcomes. This research establishes the practical applicability of integrating heuristics with reinforcement learning for complex industrial scheduling problems.

Keywords: Optimization, Scheduling, Reinforcement Learning, Heuristic, Resource Leveling Problem

1. 서 론

조선산업의 생산계획은 장기 계획, 중일정 계획, 단기 계획으로 구분되며, 그 중 중일정 계획은 수천 개의 액티비티로 구성되어 복잡한 네트워크 구조를 형성한다. 프로젝트 규모는 액티비티 수에 따라 소규모(100개 미만), 중규모(100~1,000개), 대규모(1,000개 이상)로 구분되며, 조선소 중일정 계획은 대규모 프로젝트에 해당한다(Kyriklidis and Dounias, 2023). 조선소 중일정 계획은 프로젝트 스케줄링의 자원 평준화 문제(Resource Leveling Problem)로 정의할 수 있다. 자원 평준화 문제는 프로젝트 기간 동안 자원 사용량의 변동성을 최소화하여 효율성을 극대화하는 최적화 문제이다(Herroelen 2005). 자원 사용량의 급격한 변동은 추가 비용과 관리 복잡도를 증가시키므로, 안정적인 자원 배분은 프로젝트 수행의 핵심 요인이다. 그러나 자원 평준화 문제는 NP-hard로 분류되며, 대규모 프로젝트에서는 액티비티 간 다층적 의존 관계와 해 공간의 지수적 증가로 인해 최적해 도출이 현실적으로 불가능하다.

자원 평준화 문제는 프로젝트 스케줄링 분야의 핵심 연구로서 다양한 방법론을 통해 연구되어 왔다. 기존 연구들은 주로 Meta Heuristic과 Exact Algorithm을 중심으로 이루어져 왔다. Meta Heuristic은 모든 액티비티의 시작 시간을 동시에 결정하는 특성을 가지므로, 선후행 제약 조건을 만족시키기 위해 각 액티비티의 가능한 시작 시간 범위를 선후행 관계를 항상 지킬 수 있는 범위로 초기에 제한하는 인코딩 방법을 주로

† 연락저자 : 우종훈 교수, 08826 서울특별시 관악구 관악로 1 서울대학교 34동 410호, Tel : 02-880-7330, E-mail : j.woo@snu.ac.kr
2025년 12월 16일 접수; 2025년 12월 31일 수정본 접수; 2026년 1월 2일 게재 확정.

활용해왔다. 이러한 해 공간 제한 방법은 제약 위반 해를 탐색 과정에서 배제하여 실행 가능한 해만을 생성하지만, 제한된 탐색 공간 내에서는 전역 최적해가 포함된 영역을 탐색하지 못할 가능성이 높아 해의 품질이 저하되고 local optimum에 빠지기 쉬운 한계를 가진다. 이러한 한계를 극복하기 위해 제한된 탐색 공간 내에서 local optimum을 회피하는 메커니즘을 개선하는 연구들이 진행되어 왔다. Zhang and Yang(2017, 2018)과 Lin *et al.*(2023)은 PSO 기반 알고리즘에 Markov 체인 확률, 가속 계수, Cauchy 변이 등을 적용하여 local optimum 회피 성능을 향상시켰으나, 9~233개의 소, 중규모 프로젝트에서만 우월성을 입증하였다. Li *et al.*(2018)은 이러한 인코딩 방식의 해 공간 제한 문제를 근본적으로 해결하기 위해 Random-Shift Key 표현법을 개발하여 모든 실행 가능한 해를 표현할 수 있는 방법을 제시하였으나, 50개 액티비티를 가지는 소규모 프로젝트에서만 우월성을 입증하였다. 이처럼 Meta Heuristic 기반 연구들은 해 공간 제한 인코딩으로 인한 해의 품질 저하 문제와 대규모 프로젝트로의 확장성 문제를 동시에 가지며, 수천 개의 액티비티를 가지는 대규모 프로젝트에서는 제한된 탐색 공간과 복잡도 증가로 인해 고품질의 해를 탐색하기 어렵다는 근본적 한계를 보인다. Exact Algorithm은 제약 구현이 용이하며 최적성을 보장하지만, computing time에 대한 한계가 있어 이를 극복하기 위한 탐색 효율성 연구가 주로 진행되어 왔다. Mutlu(2010), Afshar-Nadjafi *et al.*(2013), Coughlan *et al.*(2015)은 Branch and Bound 기반 알고리즘에 dominance rule, Dantzig-Wolfe 분해 등을 적용하여 탐색 효율성을 개선하였으나, 20~50개 액티비티를 가지는 소규모 프로젝트에서만 효과를 입증하였다. Ponz-Tienda *et al.*(2017)은 Parallel Branch and Bound 알고리즘을 제안하여 실제 건설 프로젝트에서도 우수한 성능을 보였으나, 문제의 크기가 증가함에 따라 computing time이 비선형적으로 증가하며 메모리 사용량이 급격하게 증가하여 하드웨어 제약으로 인한 한계를 가진다. 따라서 실제 조선소와 같은 대규모 프로젝트에서 합리적인 시간 내에 해를 구하기 어려운 scalability 문제를 갖는다.

본 연구는 Meta Heuristic의 제한된 탐색 공간으로 인한 해의 품질 저하 문제와 Exact Algorithm의 대규모 프로젝트에서의 computing time 및 메모리 한계를 동시에 극복하기 위해 휴리스틱과 강화학습을 결합한 2단계 접근법을 제안한다. Stage 1에서는 대규모 프로젝트의 복잡한 선후행 제약 조건을 완벽히 준수하면서도 연산 효율성을 극대화하기 위해 위상 정렬(Topological Sorting) 알고리즘인 Kahn's Algorithm(Kahn, 1962)을 활용하여 액티비티 결정 순서를 생성한다. Liu(2014)에 따르면, Kahn's Algorithm은 모든 정점과 간선을 선형 시간 내에 처리하는 선형의 시간 복잡도를 유지하면서도 유효한 액티비티 순서를 보장하므로, 수천 개의 액티비티가 연결된 대규모 네트워크를 처리하기에 적합하다. 특히, Zhang *et al.*(2025)과 Kalvin and Varol(1983)이 분석한 바와 같이, 간선을 가진 복잡한 네트워크 구조이나 프로젝트 액티비티 간의 의존 관계의 최적의 실행 순서를 도출하는 검증된 표준 방법론이다. 본 연구에서는 이러한 기술적 이점을 바탕으로 Kahn's Algorithm을 활용하여 대규모 프로젝트인 조선소 중일정 계획의 선후행 관계를 효율적으로 정렬하고, 이를 통해 2단계 강화학습을 위한 안정적인 의사결정 기반을 마련한다. Stage 2에서 활용한 Q-Learning은 각 의사결정 시점에서 상태(State)와 행동(Action) 쌍의 가치를 학습하며, 에피소드 기반의 반복 학습을 통해 점진적으로 최적 정책(Policy)에 근접한다. 이러한 강화학습 기반 접근법은 Jang *et al.*(2019)에서 기술한 바와 같이, 복잡한 제약 조건이 존재하는 조합 최적화 문제에서 시행착오를 통한 학습으로 고품질의 해를 탐색할 수 있다는 장점을 가진다. 특히, Zhou *et al.*(2024)과 He *et al.*(2025)이 분석한 것과 같이 Q-Learning은 환경에 대한 사전 정보가 부족하거나 동적으로 변화하는 복잡한 시나리오에서도 누적 보상을 최대화하는 최적 전략을 자율적으로 도출하는 데 효과적이다. 이는 정적 휴리스틱이 빠지기 쉬운 지역 최적해(Local optimum)를 회피하고 전역적 관점에서 고품질의 해를 탐색할 수 있게 하며, 결과적으로 자원 평준화와 같은 다단계 의사결정 문제에서 더욱 정교한 스케줄링 결과를 산출한다. 본 연구에서는 Stage 1에서 확보된 안정적인 액티비티 순서를 활용하여 학습의 탐색 효율을 높임으로써, 조선소 중일정과 같은 대규모 프로젝트에서 최적의 자원 평준화 결과를 도출하고자 한다. 결론적으로, 1단계에서는 개선된 Kahn's Algorithm 기반 휴리스틱을 통해 자원 사용량과 작업 기간을 고려한 효율적인 액티비티 순서를 생성하며, 2단계에서는 Q-Learning을 통해 각 액티비티의 최적 시작 시간을 동적으로 학습한다. 이와 같은 2단계 통합 접근은 넓은 탐색 공간을 효율적으로 반영하면서도 합리적인 연산 시간 내에 고품질의 해를 도출할 수 있다는 점에서 중요한 학술적 및 실무적 의의를 가진다.

2. 문제 정의

본 연구는 기존 연구에서 정의된 자원 평준화 문제(Resource Leveling Problem)를 기반으로 한다(Demeulemeester, Herroelen *et al.*, 2002). 프로젝트는 N 개의 액티비티 $A=\{1,2,...,N\}$로 구성되며, <Table 1>에서 보이는 바와 같이, 각 액티비티 $i \in A$는 earliest start time(es_i), latest start time(ls_i), duration(d_i), daily resource usage(r_i), 그리고 Set of predecessor activities(P_i) 같은 속성을 가진다. 본 문제의 목적은 제약 조건을 만족하면서 일별 자원 사용량의 분산 합을 최소화하는 최적의 액티비티 스케줄을 구하는 것이며, 이는 각 액티비티의 시작시간 s_i를 결정하는 NP-hard 조합 최적화 문제로 정의된다.

제약조건은 총 2가지로 구성된다. 첫째, 가능한 시작시간 범위 제약(Time constraint)는 각 액티비티 i의 시작시간 s_i가 가능한 시작시간 범위 내에 있어야 함을 의미한다.

Table 1. Example of Activity Data

Activity	es_i	ls_i	d_i	r_i	P_i
A_1	1	2	1	1	-
A_2	2	4	1	1	A_1
A_3	2	4	2	2	A_1
A_4	4	6	1	1	A_2, A_3

$$es_i \le s_i \le ls_i \quad \forall i \tag{1}$$

둘째, 선후행 제약(Precedence Constraint)은 선행 액티비티가 종료된 이후에만 후행 액티비티가 종료할 수 있도록 하는 Finish-to-Finish(FF) 관계를 나타낸다.

$$s_j + d_j \le s_i + d_i, \forall j \in P_i, \forall i \in A \tag{2}$$

목적함수(Objective function)는 일별 자원 사용량의 분산 합을 최소화하는 것이다.

$$Minimize\ \sigma^2 = \frac{1}{T} \times \sum_{t=1}^{T} (R(t) - R_{avg})^2 \tag{3}$$

여기서 $R(t)$는 t일의 총 자원사용량, R_{avg}은 전체 기간의 평균 자원 사용량, T는 프로젝트 전체 기간을 나타낸다.

	t (day)					
	1	2	3	4	5	6(T)
A_1						
A_2						
A_3						
A_4						
$R(t)$	1	0	3	2	0	1

Figure 1. Feasible Schedule of Table 1

<Figure 1>은 <Table 1> 기반 실행 가능한 일정(Feasible schedule) 중 하나를 보인 예시를 나타낸다. s_1은 1, s_2은 3, s_3은 3, s_4은 6에 시작하는 일정이며, 각 선후행 관계 제약과 가능한 시작 시간 범위 제약을 모두 만족한다. 전체 기간(T)는 6이므로 평균 자원 사용량(R_{avg}) 는 1.17으로 산출된다. 주어진 정보를 통하여 <Figure 1>의 최종 목적함수(σ^2)은 1.14로 계산된다. 이와 같은 단계를 통하여 최종 목적함수가 계산된다.

3. 모델링

자원 평준화 문제에서 각 액티비티의 해 공간은 선행 액티비티의 스케줄에 따라 동적으로 변화한다. 이러한 의존 관계로 인해 모든 액티비티의 시작 시간을 동시에 결정하는 방식은 방대한 해 공간을 탐색해야 하며, 특히 대규모 프로젝트에서는 계산 복잡도가 급격히 증가한다. Demeulemeester and Herroelen(2002)은 이러한 복잡도를 줄이기 위해 자원 평준화 문제를 두 개의 순차적 의사결정 문제로 분해하는 접근법을 제안하였다: (1) 어떤 순서로 액티비티의 시작 시간을 결정할 것인가(Activity Sequencing), (2) 각 액티비티를 어떤 시작 시간에 배정할 것인가(Activity Scheduling). 본 연구는 이러한 2단계 프레임워크를 기반으로 하되, 각 단계에 휴리스틱과 심층 강화학습을 결합한 새로운 최적화 방법론을 제안한다.

Stage 1에서는 개선된 Kahn's algorithm(Kahn, 1962) 기반 휴리스틱을 통해 액티비티 결정 순서를 생성한다. 이는 선행 제약조건을 만족하도록 선행 액티비티가 후행 액티비티보다 먼저 결정되는 순서를 보장하며, 원형 알고리즘과 달리 자원 사용량과 작업 기간을 추가로 고려하여 일별 자원 사용량 평준화에 유리한 순서를 도출한다. Stage 2에서는 tabular Q-Learning 기반 강화학습을 통해 각 액티비티의 최적 시작 시간을 순차적으로 결정한다. 강화학습의 환경이 제약 조건을 고려한 가능한 시작 시간 범위를 동적으로 계산하며, Q-table을 통해 각 상태-행동 쌍의 가치를 학습한다. 각 단계의 구체적인 알고리즘은 3.2, 3.3 섹션에서 상세히 설명한다(<Figure 2>).

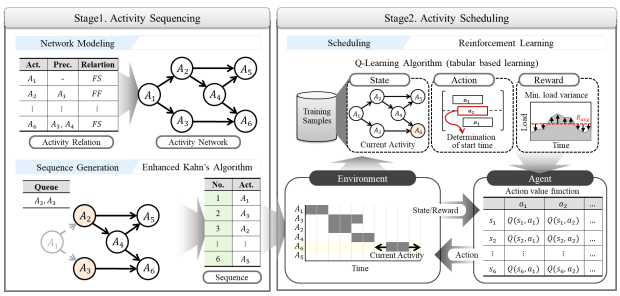

Figure 2. Framework of Proposed Algorithm

3.1 Stage 1: Enhanced Kahn's algorithm

액티비티 결정 순서 문제는 방향성 비순환 그래프(Directed Acyclic Graph, DAG)에서의 위상 정렬 문제(Topological Sorting Problem)로 정의할 수 있다. 앞서 설명한 바와 같이 각 액티비티의 해 공간은 선행 액티비티의 스케줄에 따라 동적으로 변화하므로, 선행 액티비티가 후행 액티비티보다 먼저 시작 시간을 결정하는 순서가 필요하다. 프로젝트 스케줄링에서 각 액티비티는 노드, 선후행 관계는 방향성을 가진 간선으로 표현되며, 이는 순환이 없는 DAG 구조로 형성된다. 위상 정렬(Topological Sorting)은 DAG의 모든 노드를 선후행 관계를 위반하지 않는 순서로 배열하는 문제로, 본 연구에서는 이를 통해 선행 액티비티가 우선적으로 결정되는 액티비티 스케줄링 순서를 도출한다. 본 연구에서는 Kahn's algorithm을 기반으로 하되, 선후행 관계만 고려하는 기존 알고리즘과 달리 자원 사용량이 높은 액티비티에게 높은 우선순위를 부여하여 자원 평

준화에 유리한 순서를 도출하도록 구현하였다. 기존 Kahn's algorithm은 진입 차수(in-degree)가 0인 노드부터 순차적으로 제거하며 순서를 결정하지만, 본 연구에서는 Harris(1990) 의 연구에서 제시한 액티비티의 자원 사용량(r_i)과 t일의 총 자원 사용량(R_t)과의 관계성을 통하여 자원 사용량이 큰 순서대로 우선순위를 부여한 것을 활용하여 진입 차수가 0인 노드들 중에서 자원 사용량(r_i)과 작업시간(d_i)의 곱이 큰 순서대로 우선순위를 부여한다.

Enhanced Kahn's algorithm의 구체적 절차는 <Figure 2>와 같다. 초기 단계에서 모든 액티비티의 진입 차수를 계산하고, 진입 차수가 0인 액티비티들을 순서 결정 대상 집합(Queue)에 추가한다. Queue에서 액티비티를 선택할 때는 $r_i \times d_i$값이 가장 큰 액티비티를 우선적으로 선택하여 최종 액티비티 순서를 저장하는 Result Array에 추가한다. 순서가 결정된 액티비티를 제외하고 남은 액티비티들의 진입 차수를 다시 계산한다. 새롭게 진입 차수가 0이 된 액티비티들은 Queue에 삽입되며, 이 과정을 모든 액티비티가 Result Array에 추가될 때까지 반복한다. 최종적으로 Result Array는 선행 제약을 만족하면서 자원 사용량이 큰 액티비티가 우선적으로 배치된 순서를 나타낸다.

<Figure 3>은 Enhanced Kahn's algorithm의 실행 과정을 <Table 1>의 데이터를 기반으로 한 예시다. 초기 상태에서 in-degree가 0인 A_1이 Queue에 포함되어 첫 번째 순서로 결정된다.(Step 1) 순서가 결정된 A_1과 관련된 간선 및 노드를 제외하고 다시 in-degree를 계산한다. in-degree 재계산을 통해 순서 결정 대상인 액티비티는 A_2, A_3가 Queue에 추가되며, 두 액티비티 중 $r_i \times d_i$가 더 큰 A_3 가 우선 순서가 되며, A_2 가 후순서가 되어 대상 액티비티의 순서를 모두 결정한다. (Step 2) 동일한 방식으로 in-degree 재계산 후, A_4가 후순위로 배치되며 모든 액티비티의 순서가 결정되게 된다. (Step 3) 본 예시처럼 이는 단순히 선행 제약만을 고려하는 Kahn's algorithm과 달리 자원 사용량과 작업 기간을 함께 고려하여 자원 평준화에 유리한 순서를 생성한다.

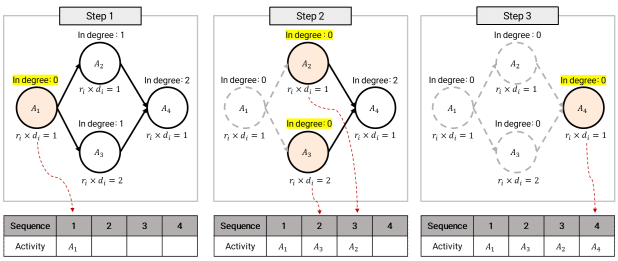

Figure 3. Example of Enhanced Kahn's algorithm

3.2 Stage 2: Q-Learning Algorithm

Stage 1에서 결정된 액티비티 순서에 따라 각 액티비티의 최적 시작 시간을 순차적으로 결정한다. 본 단계는 각 액티비티가 결정 순서에 따라 자신의 시작 시간을 선택하는 순차적 의사결정 문제로 정의되며, Q-Learning 기반 강화학습을 통해 일별 자원 사용량 분산을 최소화하는 최적 시작 시간을 학습한다. 전체 프로젝트 스케줄 생성을 하나의 에피소드로 정의하고, 다수의 에피소드를 반복하면서 각 액티비티의 시작 시간 선택에 대한 가치를 학습하여 점진적으로 최적 스케줄에 근접한다(<Figure 4>).

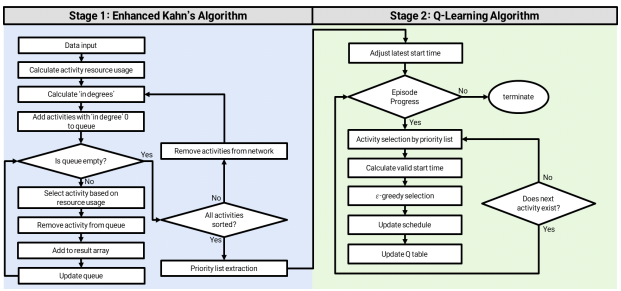

Figure 4. Flowchart of Proposed Algorithm

본 연구는 각 액티비티의 최적 시작 시간을 결정하기 위해 Q-Learning 기반 강화학습을 활용한다. Q-Learning은 model-free 강화학습 알고리즘으로, 환경의 dynamics를 명시적으로 모델링하지 않고도 최적의 행동 정책을 학습할 수 있다(Watkins and Dayan, 1992). 본 문제는 각 액티비티의 가능한 시작 시간이 유한하고 이산적이므로, Q-table을 활용한 tabular 방식의 Q-Learning을 적용한다. 에피소드 기반 학습을 통해 전체 프로젝트 스케줄을 반복적으로 생성하며, 각 에피소드마다 얻은 경험을 바탕으로 state-action 쌍의 가치(Q-value)를 점진적으로 개선한다. Environment는 현재까지 결정된 스케줄 상태를 유지하며, 각 액티비티가 선택 가능한 시작 시간 범위를 동적으로 계산하는 역할을 수행한다. 액티비티 의 가능한 시작 시간은 다음의 제약 조건을 만족해야 한다. 첫째, 시간 범위 제약(Time Constraint)으로, 시작 시간은 $es_i \le t_i \le ls_i \quad \forall i$로 정의된 시간 범위 내에 있어야 한다. 이는 각 액티비티 i 가 프로젝트 전체 일정 내에서 실행 가능한 기본적인 시간 범위를 나타낸다. 두 번째 제약조건은 $s_j + d_j \ge s_i + d_i, \forall j \in P_i, \forall i \in A$ 으로 선, 후행 제약 만족을 의미한다. 본 제약은 Stage 1의 순서에 따라 순차적으로 액티비티를 배치하는 과정에서 동적으로 계산되므로, 각 액티비티의 가능한 시작 시간 범위는 이전에 배치된 선행 액티비티들의 스케줄에 따라 달라진다. 특히 선행 액티비티의 일정은 Stage 1에 의해 이미 결정된 상태이므로, 제약 만족이 보장된다. 이러한 제약 조건을 모두 고려하여 각 액티비티의 실행 가능한 시작 시간 집합이 계산되며, 이는 Q-Learning의 action space를 구성한다. Environment는 agent가 선택한 시작 시간을 현재 스케줄에 반영하고, 해당 선택이 전체 스케줄의 자원 사용량 분산에 미치는 영향을 계산하여 reward로 제공한다.

본 문제는 Markov Decision Process(MDP)로 모델링되며, 이는 state(S), action(A), reward(R), state transition probability

(P) 로 구성된 $< S, A, R, P >$으로 정의된다. State는 현재 결정 순서에 있는 액티비티 i으로 정의된다. 여기서 i는 액티비티 인덱스를 나타낸다. Stage 1에서 결정된 순서에 따라 액티비티가 순차적으로 state로 주어지며, 각 state는 해당 액티비티의 시작 시간을 결정하는 의사결정 시점을 나타낸다. 액티비티의 작업 기간과 자원 사용량을 state에 포함함으로써, agent는 각 액티비티가 전체 스케줄에 미치는 영향을 고려하여 더 효과적으로 시작 시간을 선택할 수 있다. Action은 액티비티 i의 시작 시간으로 정의된다. Q-table은 모든 액티비티의 가능한 시작 시간 범위를 포괄하도록 구성되며, 구체적으로 전체 액티비티 중 가장 빠른 es_i 값부터 가장 늦은 ls_i 값까지의 시간 범위를 기준으로 형성된다. 실제 의사결정 시에는 Environment가 현재 스케줄 상태와 제약 조건을 고려하여 선택 가능한 시작시간만 마스킹(masking)하며, 선택된 action에 대해서만 Q-value가 업데이트된다. 따라서 각 액티비티의 실제 action space는 state와 현재 스케줄 상태에 따라 동적으로 결정된다. Reward는 해당 액티비티가 전체 스케줄의 자원 사용량 분산에 미치는 기여도(contribution)로 정의된다. 각 state-action 쌍에 대한 보상은 다음과 같다.

$$R(s,a) = \frac{\sigma_{k-1}^2 - \sigma_k^2}{\sigma_{k-1}^2} \tag{4}$$

식 (4)의 k는 현재 결정 순서의 단계(step)을 의미하며, σ_k^2는 k 번째 액티비티까지 배치된 스케줄의 자원 사용량 분산이며, σ_{k-1}^2는 $k-1$번째 액티비티까지 배치된 스케줄의 분산이다. 양수의 reward는 해당 시작 시간 선택이 전체 분산을 감소시켰음을 의미하며, 음수의 reward는 분산을 증가시켰음을 의미한다. 이때, 시작 시간이 결정되지 않은 미배치 액티비티의 부하량은 모두 0으로 처리되어 분산 계산에서 제외되도록 설계하여 에피소드 중간에서도 계산이 가능하도록 구성된다. 이러한 즉각적 보상 체계는 에이전트의 행동 시점과 보상 시점을 일치시키기 위해 설계되었다. 에이전트가 특정 액티비티를 배치하는 시점은 이전 단계까지 형성된 누적 부하 상황에 자신의 행동을 반영하는 과정이며, 즉각적 보상은 해당 행동이 현재 단계에서의 부하 상황에 미친 기여도를 보상 신호로 직접 전달한다. State transition은 우선순위 순서에 따라 다음 액티비티로 이동하는 결정론적(deterministic) 전이로 정의된다. 액티비티 i의 시작 시간이 결정되면, state는 확률 1로 우선순위 순서상 다음 액티비티 $i+1$과 그 속성 정보$(i+1, d_{i+1}, r_{i+1})$으로 전이된다. 상태의 전이는 고정된 순서를 따르나, 본 연구의 보상 구조는 각 단계(Step)에서의 의사결정이 후행 단계들과 유기적인 상관관계를 갖도록 설계되었다. 즉, 특정 단계에서의 시작 시간 선택(Action)은 단순히 해당 시점의 보상에 그치지 않고, 후행 액티비티들이 배치될 수 있는 시작 시간 선택지(Action space)를 결정함으로써 후속 단계들이 얻게 될 보상에 연쇄적인 영향을 미친다. 에이전트는 반복적인 에피소드 수행을 통해 이러한 단계별 결정 간의 상호의존성을 경험적으로 학습하며, 현재의 선택이 미래의 보상 기댓값(Q-value)에 어떠한 변화를 주는지를 파악하여 전체 분산을 최소화하는 방향으로 정책을 최적화한다. 모든 액티비티의 시작 시간이 결정되면 에피소드가 종료된다. Q-Learning 알고리즘은 각 에피소드마다 Stage 1에서 결정된 순서에 따라 모든 액티비티의 시작 시간을 순차적으로 결정하며, 각 결정에 대한 Q-value를 업데이트하여 학습한다. 알고리즘의 구체적 절차는 다음과 같다.

먼저, 각 에피소드의 시작 시 빈 스케줄로 초기화하고, 우선순위 순서의 첫 번째 액티비티부터 순차적으로 진행한다. 각 액티비티에 대해 Environment는 현재 step 기준 결정된 액티비티의 시작 시간 상태를 고려하여 현재 step 의 결정 대상 액티비티의 가능한 시작 시간 집합을 계산한다. agent는 $\epsilon-greedy$ 정책을 따라 시작 시간을 선택한다. $\epsilon-greedy$ 정책는 확률 ϵ 정책으로 가능한 시작 시간 중 무작위로 선택(exploration)하고, 확률 $1-\epsilon$ 으로 현재 Q-value가 가장 높은 시작 시간을 선택(exploitation)한다. 구체적으로, 각 가능한 action에 대한 선택 확률은 다음과 같이 정의된다.

$$\Pi(a|s) = \begin{cases} \frac{\epsilon}{|A_s|} + 1 - \epsilon \text{ if } a = \arg max_{a'} Q(s,a') \\ \frac{\epsilon}{|A_s|} \qquad otherwise \end{cases} \tag{5}$$

식 (5)에서 $|A_s|$는 state s 에서 가능한 action의 개수이다. 학습 초기에는 높은 ϵ을 사용하여 충분한 탐색을 수행하고, 에피소드가 진행됨에 따라 ϵ값을 지수적으로 감소시켜 점진적으로 greedy한 action을 강화하도록 한다.

$$\epsilon_t = \max(\epsilon_{\min}, \epsilon_0 \cdot e^{-\lambda m}) \tag{6}$$

식 (6)에서 ϵ_0는 초기 ϵ값, $\epsilon_{\min}$은 최소 ϵ값, m는 현재 에피소드 번호, λ는 감소율이다. 선택된 시작 시간은 현재 스케줄에 반영되고, Environment는 해당 액티비티의 분산 기여도를 계산하여 reward를 반환한다. 이후 state는 다음 우선순위 액티비티로 전이되며, 모든 액티비티의 시작 시간이 결정될 때까지 이 과정을 반복한다. 에피소드가 종료되면, 완성된 스케줄의 전체 자원 사용량 분산을 계산하고 최적해를 갱신한다. 이후 우선순위 순서에 따라 각 액티비티에 대해 Q-value를 업데이트한다. Q-value는 다음의 Q-Learning 업데이트 공식을 통해 갱신된다(Clifton and Laber 2020).

$$Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \alpha(r_i + \gamma \max_{a'} Q(s_{i+1}, a') - Q(s_i, a_i)) \tag{7}$$

식 (7)에서 s_i는 state 인 액티비티, a_i는 선택된 시작 시간(action), r_i 는 해당 액티비티의 분산 기여도(reward), α는 학습률(learning rate), γ는 할인계수(discount factor), s_{i+1}는 다음 순

서의 액티비티를 나타낸다. 마지막 액티비티의 경우 다음 state가 존재하지 않으므로(terminal state), $\max_{a'}Q(s_{i+1},a')$ 항은 0으로 처리되어 즉각적인 reward만 반영된다. 이러한 업데이트 과정을 통해 각 액티비티의 시작 시간 선택에 대한 장기적 가치를 학습하며, Q-table은 점진적으로 최적 정책으로 수렴한다. 전체 학습 과정은 설정된 에피소드 횟수만큼 반복되며, 각 에피소드에서 생성된 스케줄 중 최소 분산을 가지는 스케줄을 최종 해로 선택한다. 학습이 진행됨에 따라 Q-value가 안정화되고, 생성되는 스케줄의 품질이 향상되어 자원 평준화 목적을 효과적으로 달성하게 된다.

4. 실험

본 연구에서는 제안 알고리즘의 성능을 종합적으로 검증하기 위해 실제 조선소 블록 도장 일정 데이터를 기반으로 한 실험을 수행하였다. 다양한 프로젝트 규모(199개~5,828개 액티비티)의 15개 test instance를 대상으로 Constraint Programming과 Meta Heuristic 등 서로 다른 접근 방식의 기존 알고리즘들과 성능을 비교하였다. 모든 알고리즘은 동일한 시간 제한(3,600sec) 내에서 실행되었으며, 평가 지표를 활용하여 각 알고리즘이 제약 조건을 만족하면서 얼마나 효과적으로 자원 평준화를 달성하는지를 측정하였다.

4.1 성능 평가 데이터

본 연구에서는 실제 조선소 중일정 계획을 기반으로 한 데이터를 활용하여 제안 알고리즘의 실용적 적용 가능성을 검증하였다. 조선소 중일정 계획은 수천 개의 액티비티와 복잡한 선행 관계로 구성된 대규모 프로젝트 스케줄링 문제로, 다양한 제약 조건과 자원 제약이 동시에 고려되어야 한다. 특히 블록 도장 작업은 조선소 생산 공정의 핵심 단계로, 선행 작업인 블록 조립 공정과 후행 작업인 탑재 공정 간의 긴밀한 연계가 요구되며, 제한된 도장 설비와 인력 자원의 효율적 배분이 중요하다. 이러한 실제 산업 현장의 특성을 반영한 데이터를 활용함으로써, 제안 알고리즘이 이론적 환경뿐만 아니라 실제 대규모 프로젝트 환경에서도 효과적으로 적용될 수 있음을 검증하고자 하였다.

본 연구에서는 <Table 2>와 같이 조선소 블록 도장 일정 계획을 기반으로 총 15개의 test instance를 구성하였다. 테스트 케이스는 프로젝트 규모를 기준으로 소규모(Test set 1), 중규모(Test set 2), 대규모(Test set 3)로 분류하여 각 규모별로 5개씩 구분하였다. 이러한 분류는 프로젝트 규모에 따른 알고리즘의 성능 변화를 체계적으로 분석하기 위함이다. 각 test instance의 precedence groups는 액티비티 간의 선행 관계를 나타내며, 프로젝트 규모가 증가함에 따라 선행 관계의 복잡도도 함께 증가하는 특성을 보인다. 최소 규모인 S1은 199개의 액티비티와 132개의 선행 관계 그룹을 가진다. 최대 규모인 E3는 5,828개의 액티비티와 2,974개의 선행 관계 그룹으로 구성되어 대규모 프로젝트 스케줄링 문제의 특성을 충실히 반영한다. Test set 3의 경우 모두 4,000개 이상의 액티비티를 포함하여 기존 연구들이 다루지 못한 대규모 문제 영역에서의 알고리즘 성능을 검증할 수 있도록 구성하였다.

Table 2. Experiment Dataset

Test set	Test instance	Number of activities	Number of precedence group
Test set 1	S1	199	132
	S2	419	192
	S3	618	324
	S4	1,809	1,107
	S5	2,008	1,239
Test set 2	M1	2,228	1,299
	M2	2,427	1,431
	M3	3,401	1,543
	M4	3,600	1,675
	M5	3,820	1,735
Test set 3	L1	4,019	1,867
	L2	5,210	2,650
	L3	5,409	2,782
	L4	5,629	2,842
	L5	5,828	2,974

4.2 성능 비교 알고리즘

본 연구에서는 제안 알고리즘의 성능을 검증하기 위해 Resource Leveling Problem의 기존 연구 방법론들을 본 연구의 문제 정의 및 제약 조건에 맞추어 재현(reproduce)하여 비교 알고리즘으로 활용하였다. 비교 알고리즘은 Constraint Programming과 Meta Heuristic 접근법으로 구성된다.

Constraint Programming은 제약 조건을 명시적으로 정의하고 제약 전파(constraint propagation)와 백트래킹(backtracking)을 통해 실행 가능한 해 공간을 체계적으로 탐색하는 방법론으로 Exact Algorithm 중 하나이다. 본 연구에서는 Google OR-Tools의 CP-SAT Solver와 IBM ILOG CP Optimizer를 활용하였다. 두 solver는 제약 프로그래밍 기반의 최적화 엔진으로, 이론적 최적성을 보장하며 다양한 제약 조건을 명시적으로 모델링할 수 있다는 장점을 가진다. 본 연구에서는 시간 범위 제약과 Finish-to-Finish 선행 제약을 반영하여 일별 자원 사용량 분산을 최소화하는 목적 함수로 모델링하였다.

Meta Heuristic은 문제의 구조적 특성을 활용한 탐색 전략을 통해 대규모 해 공간에서 효율적으로 우수한 해를 찾는 확률론적 최적화 방법론이다. 본 연구에서는 ADPSO(Adaptive

Dissipative Particle Swarm Optimization)와 SA(Simulated Annealing)를 활용하였다. ADPSO는 Lin *et al.*(2023)에서 제안한 local escape 메커니즘이 적용된 알고리즘으로, Cauchy 변이와 Mean distance를 활용하여 local optimum을 효과적으로 회피할 수 있다. SA는 최대 부하인 일자에 속하는 액티비티의 일정을 변경하는 이웃해 생성 방법을 적용하였으며, 온도 감소 전략을 통해 전역 탐색과 지역 탐색의 균형을 조절한다.

기존 연구들은 제안된 목적 함수 및 제약 조건에서 차이점을 가지므로, 본 연구에서는 모든 알고리즘을 본 연구의 문제 정의에 따라 재구현하여 공정한 비교를 수행하였다. 이를 통해 본 연구에서 제시하는 문제 환경에서 각 알고리즘의 성능을 분석하고, 제안 알고리즘(Proposed Algorithm)과의 비교를 통해 2단계 최적화 프레임워크의 효과성을 검증하고자 한다. 탐색 및 학습시간을 모두 3,600초를 기준으로 성능을 검증한다.

4.3 평가 지표

본 연구에서는 테스트 세트별 규모와 자원 사용량의 절대적 차이가 커서 단순 수치 비교만으로는 알고리즘의 보편적인 성능 평가에 한계가 있었기에, 데이터 규모에 따른 수치 왜곡을 방지하고 객관적인 성능을 검증하고자 Baseline 알고리즘 대비 개선율(Performance Improvement Ratio, PIR)을 평가지표로 활용하였다. 본 연구의 Baseline 알고리즘은 Stage 1에서 확정된 액티비티 순서를 기반으로 하되, 각 액티비티의 시작 시간을 선후행 관계가 준수되는 허용 범위 내에서 무작위로 결정하여 항상 실행 가능한 일정을 도출하도록 설계된 알고리즘이다. 이러한 무작위 기반 시작 시간 결정에서 발생할 수 있는 결과의 편차를 최소화하고 지표의 신뢰성을 확보하기 위해, Baseline 알고리즘을 10회 반복 수행하여 산출된 평균 분산값을 최종 기준점으로 설정하였다.

$$PIR = \frac{\sigma^2_{\text{baseline}} - \sigma^2_{algorithm}}{\sigma^2_{\text{baseline}}} \times 100(\%) \quad (8)$$

여기서 $\sigma^2_{baseline}$은 10회 반복 수행된 Baseline 알고리즘의 평균 자원 사용량의 분산이며, $\sigma^2_{algorithm}$은 각 비교 알고리즘으로 얻은 자원 사용량의 분산이다. PIR은 식 (8)과 같이 정의되며, 이 값이 높을수록 Baseline 알고리즘의 결과 대비 목적함수인 자원 사용량의 분산이 크게 감소하여 더욱 균등한 자원 배분이 달성되었음을 의미한다.

4.4 실험 결과

<Figure 5>~<Figure 7>은 각 test set 별 알고리즘의 PIR(%)을 나타낸다. 실험 결과, 제안 알고리즘은 프로젝트 규모가 증가할수록 기존 알고리즘 대비 우수한 성능을 보이며, 특히 대규모 프로젝트 문제에서 일관되게 최고 성능을 달성하였다.

<Figure 5>는 과도기적인 특성을 보인다. 작은 규모의 프로젝트에서는 Constraint Programming 방법론이 상대적으로 우수한 성능을 나타냈다. S1에서는 CP-SAT가 42.46%의 PIR로 가장 높은 성능을 보였으며, CPO(42.04%), SA(41.62%)가 그 뒤를 이었고 제안 알고리즘은 40.37%로 상대적으로 낮은 성능

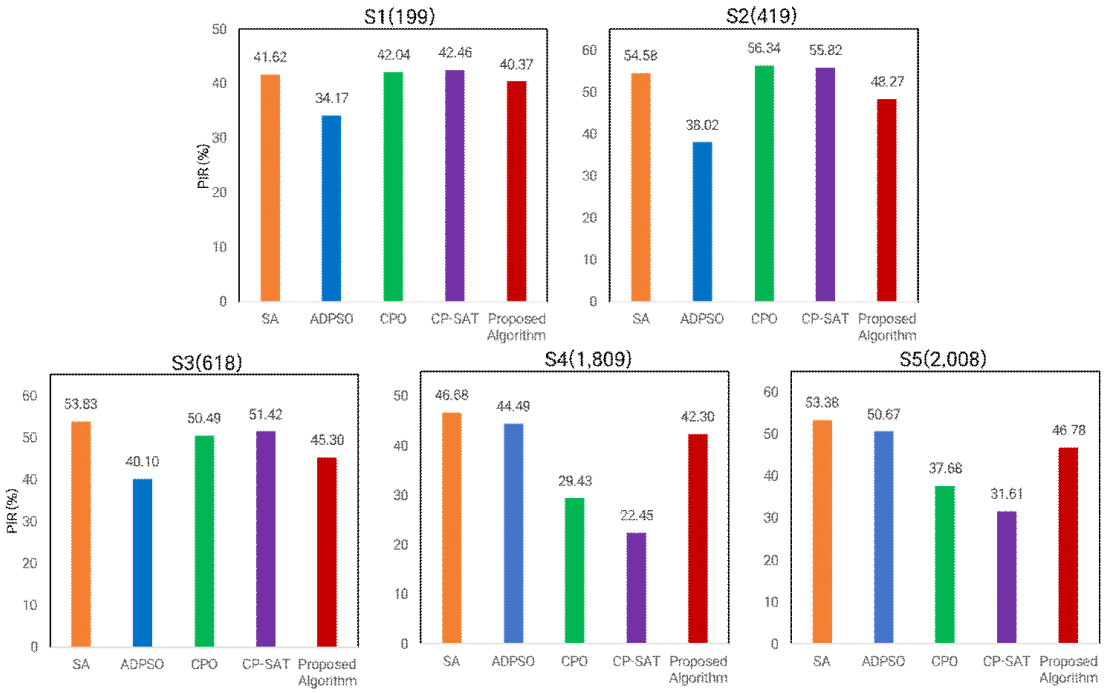

Figure 5. Test set 1

을 기록하였다. S2에서는 CPO(56.34%)와 CP-SAT(55.82%)가 높은 PIR을 달성하였으나 제안 알고리즘은 48.27%에 그쳤으며, S3에서도 SA(53.83%), CP-SAT(51.42%), CPO(50.49%)가 제안 알고리즘(45.30%)보다 우수한 성능을 보였다. 이는 소규모 문제에서 Constraint Programming이 제한된 해 공간을 주어진 시간 내에 효과적으로 탐색할 수 있음을 보인다. S4, S5에서부터는 Meta Heuristic 역시 적절한 성능을 발휘함을 시사한다. S4에서는 SA(46.68%)와 ADPSO(44.49%)가 높은 성능을 기록하였으며 제안 알고리즘은 42.30%로 3위를 차지하였으며, S5에서도 SA(53.38%)와 ADPSO(50.67%)가 제안 알고리즘

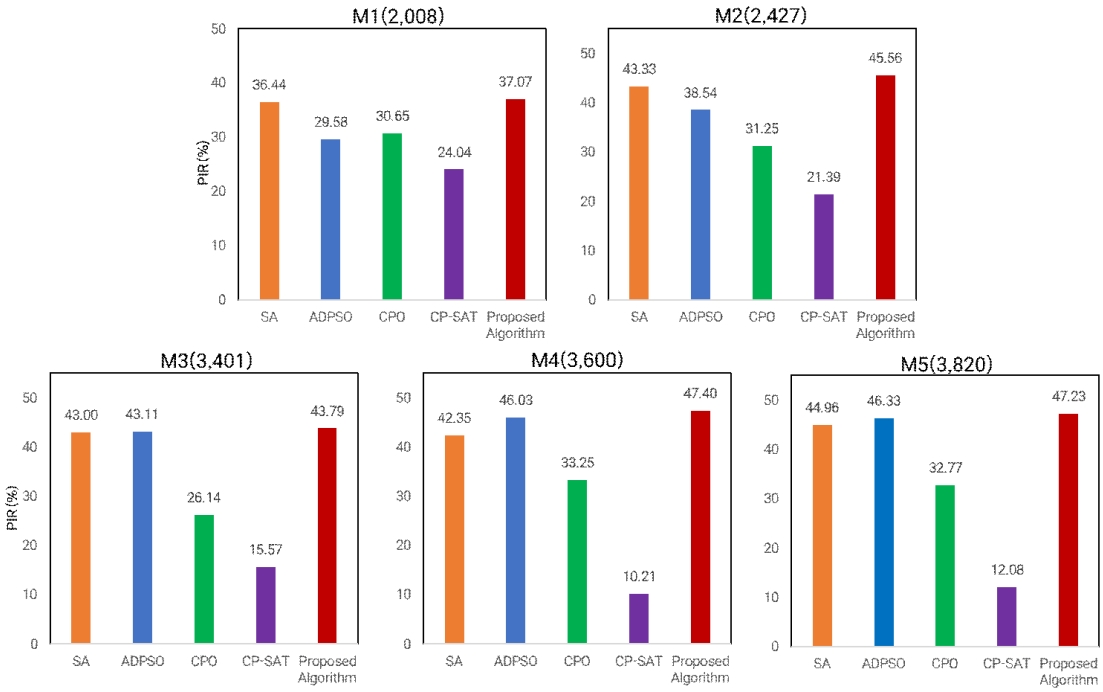

Figure 6. Test set 1

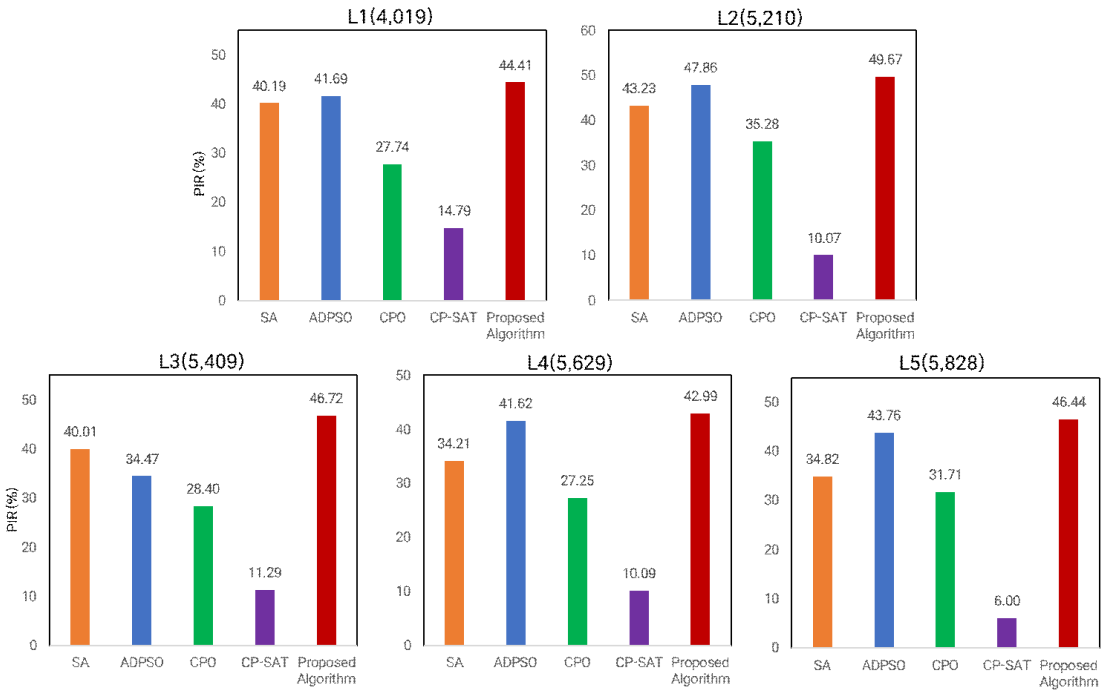

Figure 7. Test set 2

(46.78%)보다 우수한 성능을 보였다. 이는 Meta Heuristic 방법론이 Constraint Programming 보다 robust 한 성능을 보임을 확인할 수 있다. 그러나 Test set 2인 M1에서 제안 알고리즘(37.07%)이 처음으로 최고 성능을 달성하며, 이후 모든 대규모 문제에서 지속적인 우위를 보이기 시작하였다. <Figure 6>에서 제안 알고리즘이 모든 instance에서 최고 성능을 달성하였다. M2에서 제안 알고리즘은 45.56%의 PIR을 기록하며 SA(43.33%), ADPSO(38.54%)를 앞섰고, M3에서는 43.79%로 SA(43.00%) 및 ADPSO(43.11%)와 근소한 차이를 보였다. M4에서는 제안 알고리즘이 47.40%로 ADPSO(46.03%) 대비 1.37%p 높은 성능을 나타냈다. <Figure 7>은 최대 규모 즉 가장 높은 복잡도를 가진 문제로, 제안 알고리즘의 우수한 성능이 입증되었다. L3에서 46.72%, L4에서 42.99%, L5에서 46.44%의 PIR을 달성하며 전 구간에서 최고 성능을 유지하였다.

프로젝트 규모에 따른 성능 변화를 종합적으로 분석하면, 제안 알고리즘은 소규모 문제에서는 Constraint Programming 방법론 및 Meta Heuristic 알고리즘에 다소 밀렸으나, 대규모 문제부터는 일관되게 최고 성능을 달성하였다. Constraint Programming 방법론은 소규모 문제에서 우수한 성능을 보였으나 규모 증가에 따라 급격한 성능 저하를 보여 확장성의 한계를 드러냈으며, Meta Heuristic 알고리즘 중 ADPSO는 대규모 문제에서도 30% 이상의 PIR을 유지하며 선전하였으나 제안 알고리즘에는 미치지 못하였다. 이러한 결과는 제안 알고리즘의 2단계 최적화 프레임워크가 대규모 프로젝트 스케줄링 문제에서 우수한 확장성과 안정성을 가지며, 특히 실제 조선소와 같은 대규모 프로젝트 환경에서 실용적 적용 가능성이 높음을 입증한다.

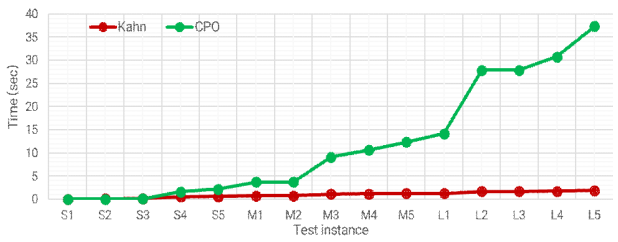

Figure 8. Test set 3

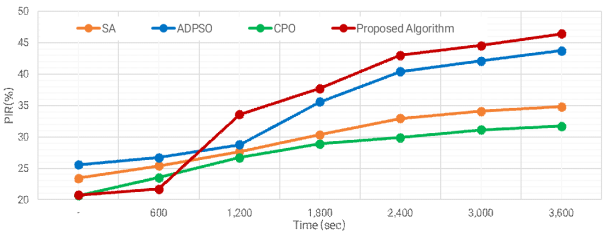

Figure 9. Computing time analysis of stage 1

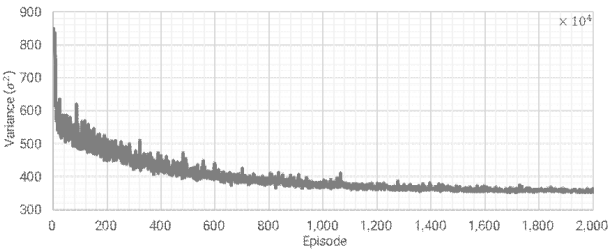

Figure 10. Learning Curve of Q-Learning

4.5 알고리즘 분석

본 세션은 제안 알고리즘의 효율성을 다각도로 분석하기 위해 세 가지 측면에서 성능을 평가한다. 첫째, Stage 1의 Enhanced Kahn's Algorithm과 Constraint Programming의 액티비티 순서 결정 소요 시간을 비교하여 휴리스틱 기반 접근법의 효율성을 검증한다. 둘째, 학습 시간에 따른 알고리즘의 수렴 패턴을 분석하여 제안 알고리즘의 탐색 효율성과 안정성을 평가한다. 마지막으로, 에피소드 진행에 따른 목적함수 값의 변화를 학습 곡선(Learning curve)을 통해 분석하여 알고리즘의 유효성을 검증한다. 이는 매 에피소드마다 경험이 누적됨에 따라 Q-value 업데이트가 이루어지고, 결과적으로 스케줄의 분산이 점진적으로 감소하는 학습 과정을 확인하기 위함이다.

첫 번째 분석으로, Stage 1의 Enhanced Kahn's Algorithm의 효율성을 검증하기 위해 Constraint Programming과의 순서 결정 소요 시간을 비교하였다. 순서 결정 단계에서는 명확한 목적 함수가 정의되지 않으므로, Constraint Programming은 실행 가능한 해를 탐색하는 방식으로 모델링하였다. 결정 변수는 각 액티비티의 순서를 나타내는 정수 변수로 설정하였으며, 제약 조건은 다음과 같이 구성하였다: (1) 모든 정수 변수는 중복될 수 없으며, (2) 선행 관계가 존재하는 경우 선행 액티비티의 정수 변수가 후행 액티비티의 정수 변수보다 작아야 한다. IBM ILOG CP Optimizer(CPO)는 이러한 제약 조건을 만족하는 첫 번째 실행 가능한 해를 발견하면 탐색을 종료하도록 설정하였다. <Figure 8>은 프로젝트 규모에 따른 두 방법론의 순서 결정 소요 시간을 나타낸다. Enhanced Kahn's Algorithm은 S1에서 0.01초, M5에서 1.88초로 프로젝트 규모 증가에 비례하여 선형적인 시간 증가를 보였다. 반면 CPO는 S1에서 0.01초로 시작하여 초기에는 유사한 성능을 보였으나, S4(1.64초)부터 급격히 증가하기 시작하여 M5에서는 37.33초에 도달하였다. 이는 약 19.9배의 시간 차이로, 대규모 문제에서 CPO의 탐색 공간이 비선형적으로 증가함을 보여준다. 이러한 차이는 두 방법론의 근본적인 접근 방식에서 기인한다. Enhanced Kahn's Algorithm은 위상 정렬 기반의 greedy 방식으로 액티비티 수와 선행 관계 수에 선형적으로 비례한다. 반면 CPO는 제약 전파와 백트래킹을 통해 가능한 순서 조합을 탐색하므로, 액티비티 수가 증가할수록 탐색 공간이 기하급수적으로 증가한다. 선행 제약 조건을 통해 탐색 공간이 축소되더라도 대규모 문제에서는 여전히 상당한 계산 시간이 요구됨을 확인할 수 있다. Stage 1에서 휴리스틱 기반 접근법을 채택함으로써, 전체 2단계 프레임워크는 순서 결정 단계에서 최대 2초 이내

의 효율적인 성능을 달성하며, 이는 Stage 2의 Q-Learning에 충분한 학습 시간을 할애할 수 있게 한다. Stage 1에서 휴리스틱 기반 접근법을 채택한 것이 전체 알고리즘의 효율성과 확장성 측면에서 타당함을 입증한다.

두 번째로, 학습 시간에 따른 알고리즘의 수렴 특성을 분석하기 위해 M5 test instance를 대상으로 각 알고리즘의 PIR 변화를 측정하였다(<Figure 9>). 모든 알고리즘은 동일한 시간 간격(600 sec)에서 성능을 기록하였으며, 이를 통해 탐색 효율성과 수렴 속도를 비교하였다. <Figure 9> 는 학습 시간에 따른 각 알고리즘의 PIR 변화를 나타낸다. 초기 단계에서는 ADPSO(25.55%)와 SA(23.41%)가 제안 알고리즘(20.74%)보다 우수한 성능을 보였으나, 1,200초 이후부터 제안 알고리즘(33.57%)이 역전하기 시작하였다. 최종적으로 3,600초에서 제안 알고리즘은 46.44%의 PIR을 달성하여 ADPSO(43.76%), SA(34.82%), CPO(31.71%) 대비 가장 높은 성능을 기록하였다. 제안 알고리즘은 초기 성능은 낮았으나 학습이 진행됨에 따라 급격한 개선을 보였으며, 최종적으로 25.70%p의 향상을 달성하였다. 반면 Meta Heuristic 알고리즘들은 초기에는 우수한 성능을 보였으나 후반부에는 개선 속도가 둔화되었고, CPO는 전 구간에서 가장 느린 수렴을 나타냈다. 이는 제안 알고리즘이 학습 기반 접근법을 통해 충분한 시간이 주어질 경우 효과적으로 최적해에 근접할 수 있음을 보여준다.

마지막 분석으로, 학습의 유효성을 확인하기 위한 에피소드 진행에 따른 목적함수 값을 변화를 보인 학습곡선을 측정하였다. 분석 대상으로는 S4 test instance 를 선정하였으며, 2,000 에피소드 동안의 분산 수치 변화를 추적하였다. <Figure 10>은 에피소드 진행에 따른 S4 test instance의 목적함수 변화를 나타낸다. 초기 단계의 빠른 하락 이후에도 수렴 지점에 도달하기 전까지 분산 값이 지속적으로 감소하며 스케줄의 품질이 점진적으로 개선됨을 확인하였다. 특히 에피소드가 반복됨에 따라 에이전트가 더 나은 해를 탐색하고 학습을 축적함에 따라 개선 속도는 점차 완만해졌으나, 약 1,200 에피소드 이후부터는 분산 값이 360×10^4선에서 안정화되며 수렴되는 양상을 보였다. 이러한 결과는 Q-value 업데이트가 전 학습 과정에 걸쳐 유효하게 이루어지며 실질적인 스케줄의 품질 향상을 이끌어냄을 보인다.

5. 결 론

본 연구는 조선소 중일정 계획과 같은 대규모 프로젝트에서 일별 자원 사용량 분산을 최소화하는 Resource Leveling Problem을 해결하기 위해 휴리스틱과 강화학습을 결합한 2단계 최적화 프레임워크를 제안하였다. 제안 알고리즘은 Stage 1에서 Enhanced Kahn's Algorithm을 통해 자원 사용량과 작업 기간을 고려한 효율적인 액티비티 순서를 생성하고, Stage 2에서 Q-Learning을 통해 각 액티비티의 최적 시작 시간을 동적으로 학습한다. 이러한 2단계 접근법은 Meta Heuristic의 제한된 탐색 공간 문제와 Exact Algorithm의 계산 시간 한계를 동시에 극복할 수 있다는 장점을 가진다.

실제 조선소 블록 도장 일정 데이터를 기반으로 한 실험 결과, 제안 알고리즘은 프로젝트 규모가 증가할수록 기존 알고리즘 대비 우수한 성능을 보였다. 중규모 문제에서는 Constraint Programming 방법론이 제한된 해 공간을 효과적으로 탐색하여 경쟁력 있는 성능을 보였으나, 대규모 문제로 갈수록 제안 알고리즘의 우위가 명확해졌다. 특히 수천 개 이상의 액티비티를 포함하는 대규모 문제에서 Constraint Programming은 급격한 성능 저하를 보인 반면, 제안 알고리즘은 일관되게 높은 성능을 유지하였다. Meta Heuristic 알고리즘은 전 구간에서 비교적 안정적인 성능을 보였으나, 제안 알고리즘에는 미치지 못하였다.

본 연구의 주요 기여는 다음과 같다. 첫째, 휴리스틱과 강화학습을 결합한 2단계 접근법을 통해 대규모 Resource Leveling Problem을 효과적으로 해결할 수 있는 새로운 방법론을 제시하였다. 둘째, 실제 조선소 데이터를 활용한 실험을 통해 산업 현장에서의 실용적 적용 가능성을 입증하였다. 셋째, 기존 Constraint Programming 및 Meta Heuristic 방법론과의 종합적인 비교를 통해 대규모 문제에서 제안 알고리즘의 우수성을 확인하였다.

본 연구는 대규모 프로젝트 스케줄링 문제에서 휴리스틱과 강화학습의 효과적인 결합 가능성을 제시하였으며, 특히 조선소와 같은 복잡한 생산 환경에서 자원 평준화를 달성하는 데 기여할 수 있을 것으로 기대된다. 향후 연구에서는 여러 방향의 확장이 가능하다. 첫째, 본 연구에서 활용한 Q-Learning은 각 프로젝트 인스턴스에 대해 개별적으로 학습이 필요하므로, 심층 강화학습(Deep Reinforcement Learning)을 적용하여 다양한 프로젝트 간 일반화 성능을 확보하는 연구가 필요하다. Graph Neural Network 또는 Attention 메커니즘과 같은 심층 신경망을 활용하면 학습된 정책이 새로운 프로젝트 인스턴스에 대해서도 즉각적으로 적용 가능하며, 이는 실제 산업 현장에서의 활용성을 크게 향상시킬 수 있다. 둘째, 다중 자원 제약 및 불확실성을 고려한 확장 연구가 필요하다. 실제 프로젝트 환경에서는 인력, 장비, 예산 등 다양한 자원이 동시에 제약되며, 작업 기간 및 자원 가용성의 불확실성이 존재한다. 이러한 실무적 복잡성을 반영한 알고리즘 개발은 산업 현장에서의 실용성을 더욱 높일 수 있을 것이다.

참고문헌

Afshar-Nadjafi, B., Khalaj, Z., and Mehdizadeh, E. (2013), A branch and bound approach to solve the preemptive resource leveling problem, *International Journal of Manufacturing Engineering,* **2013**(1), 930920.

Clifton, J. and Laber, E. (2020), Q-learning: Theory and applications, *Annual Review of Statistics and Its Application*, **7**, 279-301.

Coughlan, E. T., Lübbecke, M. E., and Schulz, J. (2015), A branch-price-

and-cut algorithm for multi-mode resource leveling, *European Journal of Operational Research*, **245**(1), 70-80.
Demeulemeester, E. L. and Herroelen, W. S. (2002), *Project Scheduling: A Research Hand Book*, Springer Science & Business Media.
Harris, R. B. (1990), Packing Method for Resource Leveling (PACK), *Journal of Construction Engineering and Management*, **116**(2), 331-350.
He, Q., Zhang, L., Fang, H., Wang, X., Ma, L., Yu, K., and Zhang, J. (2025), Multistage Competitive Opinion Maximization With Q-Learning-Based Method in Social Networks, *IEEE Transactions on Neural Networks and Learning Systems*, **36**(4), 7158-7171.
Herroelen, W. (2005), Project scheduling—Theory and practice, *Production and Operations Management*, **14**(4), 413-432.
Jang, B., Kim, M., Harerimana, G., and Kim, J. W. (2019), Q-learning algorithms: A comprehensive classification and applications, *IEEE Access*, **7**, 133653-133667.
Kahn, A. B. (1962), Topological sorting of large networks, *Communications of the ACM*, **5**(11), 558-562.
Kalvin, A. D. and Varol, Y. L. (1983), On the generation of all topological sortings, *Journal of Algorithms*, **4**(2), 150-162.
Kleinrock, L. and Kamoun, F. (1980), Optimal clustering structures for hierarchical topological design of large computer networks, *Networks*, **10**(3), 221-248.
Kyriklidis, C. and Dounias, G. (2023), A ship-construction dataset for resource leveling optimization in large project management problems, *DatainBrief*, **49**, 109340.
Li, H., Xiong, L., Liu, Y., and Li, H. (2018), An effective genetic algorithm for the resource levelling problem with generalised precedence relations, *International Journal of Production Research*, **56**(5), 2054-2075.
Lin, J. C. W., Lv, Q., Yu, D., Srivastava, G., and Chen, C. H. (2023), Adaptive particle swarm optimization model for resource leveling, *Evolving Systems*, **14**(4), 593-604.
Liu, R. (2014), A Low Complexity Topological Sorting Algorithm for Directed Acyclic Graph, *International Journal of Machine Learning and Computing*, **4**(2), 194-197.
Mutlu, M. Ç. (2010), *A branch and bound algorithm for resource leveling problem*, Middle East Technical University (Turkey).
Ponz-Tienda, J. L., Salcedo-Bernal, A., Pellicer, E., and Benlloch-Marco, J. (2017), Improved adaptive harmony search algorithm for the resource leveling problem with minimal lags, *Automation in Construction*, **77**, 82-92.
Watkins, C. J. C. H. and Dayan, P. (1992), Q-learning, *Machine Learning*, **8**(3), 279-292.
Zhang, H. and Yang, Z. (2017), Large-Scale Network Plan Optimization Using Improved Particle Swarm Optimization Algorithm, *Mathematical Problems in Engineering*, **2017**, 3271969.
Zhang, H. and Yang, Z. (2018), Accelerated Particle Swarm Optimization to Solve Large-Scale Network Plan Optimization of Resource-Leveling with a Fixed Duration, *Mathematical Problems in Engineering*, **2018**, 9235346.
Zhang, L., Chen, J., and Liu, W. (2025), Research on Enterprise Workflow Engine Optimization based on Knowledge Graph Algorithm, *Procedia Computer Science*, **262**, 244-251.
Zhao, Y., Zheng, Z., Zhang, X., and Liu, Y. (2017), Q learning algorithm based UAV path learning and obstacle avoidance approach, *Proceedings of the 36th Chinese Control Conference (CCC)*, 6397-6401.
Zhou, Q., Lian, Y., Wu, J., Zhu, M., Wang, H., and Cao, J. (2024), An optimized Q-Learning algorithm for mobile robot local path planning, *Knowledge-Based Systems*, **286**, 111400.

저자소개

이 홍: 부산대학교 조선해양공학과에서 2024년 학사학위를 취득하고 서울대학교 조선해양공학과에서 석사과정에 재학 중이다. 연구분야는 강화학습, 최적화, 생산공학, 시뮬레이션이다.

우종훈: 서울대학교 조선해양공학과 교수로 재직 중이다. 연구분야는 생산관리, 시뮬레이션, 최적화, 심층강화학습이다.