학사학위 청구논문

주차장 AGV 운용을 위한 제어 알고리즘 구현 및 시뮬레이션 기반 검증

(Implementation and Simulation-based Verification of Control Algorithms for Parking Lot AGV Operation)

2025년 12월 24일

숭실대학교 IT대학

전자정보공학부 전자공학전공

우 상 욱

학사학위 청구논문

주차장 AGV 운용을 위한 제어 알고리즘 구현 및 시뮬레이션 기반 검증

(Implementation and Simulation-based Verification of Control Algorithms for Parking Lot AGV Operation)

지도교수 : 안 병 권

이 논문을 학사학위 논문으로 제출함

2025년 12월 24일

숭실대학교 IT대학

전자정보공학부 전자공학전공

우 상 욱

우상욱의 학사학위 논문을 인준함

심사위원장 이 재 진 (인)

심 사 위 원 X X X (인)

2025년 12월 24일

숭실대학교 IT대학

감사의 글

논문이 완성되도록 도와주신 모든 분께 감사드립니다.

목 차

표 및 그림 목차

국문초록

주차장 AGV 운용을 위한 제어 알고리즘 구현 및 시뮬레이션 기반 검증

전자정보공학부 전자공학전공 우 상 욱
지도교수 안 병 권

AGV(Automated Guided Vehicle) 기반 자동화는 물류·의료·주차 등 다양한 산업에서 확산되고 있으며, AGV 관련 시장은 2024년 약 52억 2천만 달러에서 2037년 약 300억 6천만 달러까지 성장(연평균 14.3%)할 것으로 전망된다.
특히 주차장 분야에서는 차량 간 간격 최소화 및 주차 동선의 주차면 전환을 통해 공간 효율을 높일 수 있고, 인적 오류에 의한 사고를 감소시키며, 주차장 내 차량 주행을 줄여 배기가스 배출 저감 및 공기질 개선 효과를 기대할 수 있다. 그러나 다수 AGV가 제한된 주차 공간을 공유하는 환경에서는 충돌(conflict)과 교착(deadlock)이 빈번하게 발생하며, 이는 다중 에이전트 경로 탐색(MAPF) 문제로 모델링될 수 있다. MAPF에서는 동일 시간 동일 위치 점유(정점 충돌) 및 동일 시간 구간 동일 경로 교차(간선 충돌) 등 시간-공간 충돌이 핵심 제약으로 작용한다.

본 연구는 무인 주차장 환경에서의 다중 AGV 운용을 대상으로, 우선순위 기반 제어와 윈도우 기반 협력 경로 계획(WHCA*), 교착 탐지(WFG/SCC) 및 국소 협상(Partial CBS)을 결합한 AGV 제어 알고리즘을 구현하고 시뮬레이션으로 검증한다. WHCA*는 예약 테이블을 활용해 우선순위 순차 계획을 수행하되 교착 위험이 존재하므로, 본 연구에서는 WFG 및 SCC를 이용해 교착을 탐지하고, 교착 집단에 Partial CBS를 적용해 충돌을 국소적으로

해소하며 실패 시 우선순위 기반/풀오버(pull-over) 절차로 복구한다.
제안 방법은 실제 주차장(형남공학관 지하2층) 및 확장 맵(MAP2, MAP3; Parking=61/390/900, AGV 최대 8대)에서 평가하였다.
실험 결과, 저밀도 환경에서는 A* 및 D* Lite가 높은 처리량(throughput)을 보인 반면, 고밀도 환경에서는 WHCA*+WFG/SCC+CBS 조합이 가장 높은 처리량을 보였으며 동일 조건에서 계산량도 가장 효율적인 것으로 나타났다.

I. 서 론

AGV는 정해진 경로를 따라 자체 동력으로 이동하며 물류/운반 작업을 자동화하는 이동체로서, 다양한 산업 현장에서 생산성과 안전성을 동시에 향상시키는 핵심 수단으로 활용되고 있다. AGV 기반 시스템의 확산은 시장 성장 전망에서도 확인되며, 2024년 약 52억 2천만 달러 규모에서 2037년 약 300억 6천만 달러로 확대(연평균 14.3%)될 것으로 예측된다.

특히 주차장 자동화는 도심의 공간 제약, 안전 문제, 환경 요인 등 복합적 과제를 동시에 다루는 응용 분야로서 중요성이 커지고 있다. 무인 주차 시스템은 차량 간 간격을 최소화하여 주차 밀도를 향상시키고, 기존 주행로를 주차 공간으로 전환함으로써 공간 효율을 증대시킬 수 있다. 또한 인적 오류로 인한 충돌 사고를 줄일 수 있다. 경찰청 통계에 따르면 2023년 주차장 사고는 1,542건(하루 평균 4.2건) 수준으로 보고되어 안전성 측면의 개선 필요성이 존재한다. 나아가 차량 주행 자체를 배제한 운용 시스템을 통해 배기가스 발생을 근본적으로 방지하며, 주차 공간 내 대기오염 물질을 모두 없애 쾌적한 환경을 조성할 수 있다.

그러나 다수의 AGV가 동일한 주차 환경을 공유할 때, 경로 간섭은 단순 충돌을 넘어 시스템 정지로 이어질 수 있다. 이 문제는 다중 에이전트가 시간 축을 포함한 공유 환경에서 충돌 없이 이동하는 경로 집합을 찾는 MAPF 문제로 정의되며, 대표적인 충돌 유형으로 동일 시점 동일 위치 점유(정점 충돌) 및 동일 시간 구간 동일 간선 교차(간선 충돌)가 존재한다. 이러한 충돌이 누적되면 에이전트들이 서로의 이동을 기다리며 진행이 멈추는 교착 상태가 발생할 수 있고, 고밀도 환경에서 이는 처리량 저하 및 운영 안정성 악화의 주요 원인이 된다.

본 연구는 무인 주차장 환경에서의 다중 AGV 운용을 위해, 우선순위 기반 제어와 협력적 경로 계획 및 교착 처리 절차를 결합한 제어 알고리즘을

구현하고 시뮬레이션으로 검증하는 것을 목표로 한다. 제안 방법은 (1) 우선순위 및 예약 테이블을 활용하는 WHCA* 기반 계획(경로 탐색기로 D* Lite 활용), (2) 대기 관계를 표현하는 WFG와 SCC를 이용한 교착 탐지, (3) 교착 집단에 대한 Partial CBS 적용 및 실패 시 우선순위 기반 및 풀오버 절차를 통한 경로계획으로 구성된다. 제안 알고리즘은 형남공학관 지하2층 주차장을 모델링한 MAP1과 대형마트 주차장 형태를 모델링한 MAP2, MAP3(주차면 61/390/900, AGV 최대 8대)에서 성능을 비교·분석하였다. 또한 결과를 통해 저밀도 환경에서는 A* 및 D* Lite가 유리하고, 고밀도 환경에서는 WHCA*+WFG/SCC+CBS 조합이 처리량 및 계산 효율 측면에서 우수함을 확인함으로써, “환경에 따른 최적 알고리즘 선택”의 필요성을 제시한다.

Ⅱ. AGV System 이론적 배경

Ⅱ-1. AGV 및 AGV Control System(ACS)

Ⅱ-1.1 AGV 정의

AGV(Automated Guided Vehicle)는 “정해진 경로를 따라 자체 동력으로만 자재를 운반하는 목적을 가진 자동화 차량”으로 정의된다. 본 연구에서는 무인 주차장 환경에서 AGV가 주차/출차 작업을 수행하는 상황을 고려하며, 다수 AGV 동시 운용 시 충돌 및 교착 문제의 해결이 핵심 과제로 나타난다.

Ⅱ-1.2 ACS 정의 및 역할

AGV Control System(ACS)은 “AGV를 제어(Control)하는 시스템”으로, Process Control Level에 해당하며 Corporate/Works Level과 Control/Field Level 사이의 중간 계층에 위치한다.

ACS의 역할은 크게 다음과 같이 정리된다.

1) 상위 시스템과의 인터페이스: 상위 시스템의 작업 지시를 수신하고 해석함

2) 하위 장비 제어: AGV의 실시간 경로 제어, 충돌 방지, 정지 최소화, 배차 스케줄링 등을 수행함

3) 상황 조율: 타 제어시스템과의 데이터 교환을 통해 운용 상황을 조율함

즉, 무인 주차장과 같이 공간이 제한된 환경에서 ACS는 단순 “최단 경로 산출”을 넘어, 다중 AGV 간 충돌 회피 및 교착 상태를 포함한 운용 안정성을 보장할 수 있는 제어 전략을 요구한다.

Ⅱ-2. 다중 에이전트 경로 탐색(MAPF) 및 충돌 모델

Ⅱ-2.1 MAPF 정의

MAPF(Multi-Agent Path Finding)는 “여러 에이전트가 공유된 환경에서 충돌 없이 이동하는 경로 집합을 찾는 문제”로 정의되며, 핵심 구성요소로 Agents, Environment, Start/Goal, 그리고 Time을 포함한다. 주차장 운용 문제는 시간에 따라 점유가 변하는 공간에서 다수 AGV가 이동하므로 MAPF로 자연스럽게 모델링된다.

Ⅱ-2.2 충돌 유형

MAPF에서 대표적인 충돌은 다음 두 가지로 구분된다.

1) 정점 충돌(Vertex Conflict): 동일 시간에 동일 정점(위치)을 점유

2) 간선 충돌(Edge Conflict): 동일 시간 사이에 동일 간선(경로)을 교차하여 통과

이러한 충돌은 즉각적인 물리 충돌 위험뿐 아니라, 충돌 회피 과정에서 대기(wait)가 연쇄적으로 발생하여 교착(deadlock)으로 이어질 수 있다는 점에서, 본 연구의 알고리즘 설계에서 핵심 제약으로 작용한다.

Ⅱ-3. 단일 에이전트 경로 계획 알고리즘

본 절에서는 다중 AGV 계획의 구성요소로 사용되는 단일 경로 탐색 알고리즘(A*, D* Lite)을 정리한다. 다중 에이전트 문제를 직접 최적화하기보다, 우선순위/예약/제약 기반으로 단일 탐색기를 반복 호출하는 구조를 채택하는 경우가 많으며, 본 프로젝트 또한 동일한 접근을 사용한다

Ⅱ-3.1 A* 알고리즘

A*는 그래프 기반 최단경로 탐색에서 널리 사용되며, f(n)=g(n)+h(n) 평가함수를 통해 f가 최소인 경로를 최단 경로로 선택한다.

g(n): 시작(출발) 노드부터 현재 노드까지의 실제(또는 최소) 소요 비용

h(n): 현재 노드부터 목표 노드까지의 최소 예상 비용(휴리스틱), 예: 맨해튼 거리

f(n): 총 예상 비용(휴리스틱 비용 함수)

Ⅱ-3.2 D* Lite 알고리즘

D* Lite는 동적 환경에서 주로 사용되는 경로 계획 기법으로, 목표에서 역방향으로 각 지점까지의 최단 경로 비용을 유지·갱신하는 방식으로 설명된다.

D* Lite의 핵심 상태값으로 다음을 정의한다.

g(s) : 목표 노드부터 현재 노드까지의 '알려진' 추정 소요 비용

rhs(s) : g(s)의 1-step 예측 값으로, 목표에서는 0이고 그 외에는 후속 노드 비용 최소로 정의됨

Consistency: g(s)=rhs(s)이면 consistent, 다르면 inconsistent로 분류

Priority Key: 우선순위 큐를 통해 비일관(inconsistent) 노드를 갱신하여 일관성을 확보하는 탐색을 수행

Ⅱ-4. 다중 AGV 협력 경로 계획 요소

Ⅱ-4.1 WHCA* 개요

WHCA*(Windowed Hierarchical Cooperative A*)는 우선순위와 예약 테이블을 활용한 MAPF 알고리즘으로 정리되며, 본 프로젝트에서는 경로 탐색기로 D* Lite를 사용한다고 명시되어 있다.

1) W: Space-Time Reservation Table – [time][x][y] 형태의 3차원 예약 테이블(0 < time < H)

2) H (Hierarchical) – 우선순위에 따른 순차적 경로 계획

3) C (Cooperative) – 타 에이전트의 예약 노드를 장애물로 간주하여 협력적 계획

다만 WHCA*는 우선순위 기반인 만큼 "Risk of Deadlock"이라는 한계가 함께 제시된다. 따라서 실제 운용 수준에서는 교착 탐지/해소 모듈을 결합하는 방식이 요구된다.

Ⅱ-4.2 WFG 및 SCC 기반 교착 탐지

WFG(Wait-For Graph)는 "현재 시스템 대기 상태를 표현하는 자료구조"이며, 교착 상태 탐지를 위한 역할을 수행한다.

1) G=(V,E)로 정의되며,

2) V: 운용 AGV 집합(정점)

3) E : "Ai가 Aj를 기다린다"는 대기 관계의 간선 으로 구성된다.

SCC(Strongly Connected Components)는 "그래프 내에서 서로에게 도달 가능한 정점들의 집합"이며, 크기 2 이상 SCC가 존재하면 교착 상태로 판단한다는 조건이 제시된다. 즉, 교착 탐지는 "대기 관계 그래프에서 사이클(강연결 집합)의 존재 여부"로 귀결된다.

Ⅱ-4.3 CBS 기반 국소 충돌 해소

CBS(Conflict-Based Search)는 “지역적 충돌 해소를 위한 MAPF 알고리즘”으로, “소수 에이전트 간 충돌을 효율적으로 해결”하여 시스템 계산 효율성을 향상시키는 것으로 설명된다. CBS는 두 단계로 구성된다.

1) 저수준(Low-Level): [time][x][y] 공간-시간 상태에서 A* 탐색을 수행(상태) f(s)=g(s)+h(s))

2) 고수준(High-Level): 충돌 해결을 위한 제약 조건(Constraints)을 결정하고, 비용이 낮은 노드를 우선순위 큐로 선택하여 탐색하는 구조

본 프로젝트에서는 CBS를 교착으로 판정된 소그룹(SCC 그룹)에 부분 적용(Partial CBS)하는 방식으로 연결하여, 혼잡 환경에서의 교착 해소를 실용적으로 수행하는 파이프라인을 구성한다(교착 감지 → SCC 그룹 Partial CBS → 실패 시 우선순위 기반 해결 → Pull-over)

Ⅲ-2. 우선순위 기반 제어(Priority System)

Ⅲ-2.1 우선순위 점수 정의

우선순위는 상태 기반 Base 점수와 stuck 기반 boost를 결합하여 다음과 같이 정의한다.

1) Priority(A)=Base×100+StuckBoost−A.id

2) Base = Agent_State

  2-1) 출차(RETURNING_WITH_CAR)=3

  2-2) 충전필요(GOING_TO_CHARGE)=2

  2-3) 주차/수거(GOING_TO_PARK/GOING_TO_COLLECT)=1

  2-4) 대기/충전/기지복귀(IDLE/CHARGING/RETURNING_HOME)=0

StuckBoost는 정체 누적 정도에 따라 가중치를 부여하며, 임계치 이상에서는 큰 부스트를 부여하여 교착 상황에서 우선 해소 대상이 되도록 설계한다. 또한 tie-breaking은 AGENT ID를 기준으로 하여 계획 결과의 결정성을 확보한다.

Ⅲ-2.2 우선순위의 역할

본 연구에서 우선순위는 단순히 “먼저 이동할 에이전트 선택”에 그치지 않고, (1) 협소 환경에서의 병목 구간을 통과시키기 위한 통제 수단, (2) stuck 증가에 대한 보상(가중치)으로 교착 상황을 완화하기 위한 제어 변수로 작용한다. 특히 WHCA* 기반

협력 계획에서도 “Hierarchical(우선순위 기반 순차 계획)”이 핵심 구성으로 포함된다.

Ⅲ-3. 단일 탐색기 기반 다중 AGV 계획(A*, D* Lite)

본 절에서는 A*, D* Lite를 다중 AGV 상황에 적용하기 위한 계획 절차를 기술한다. “우선순위 순차 계획 + 임시 장애물(타 에이전트 점유/이동)을 반영하여 충돌을 회피”하는 구조로 설계하였다.

Ⅲ-3.1 A* 기반 계획 절차

A*는 f(n)=g(n)+h(n)로 정의되는 평가 함수를 통해 최단 경로를 탐색한다(맨해튼 거리 기반 휴리스틱 사용). 다중 AGV 적용 시, 의사코드는 다음 흐름을 제시한다.

1. 모든 에이전트의 현재 위치(next_pos) 취합

2. 우선순위 기준으로 에이전트 정렬

3. 각 에이전트에 대해, 상위 우선순위의 “다음 이동”과 하위 우선순위의 “현재 위치”를 임시 장애물로 구성

4. 임시 장애물을 반영하여 A*로 경로 계산 후, 다음 스텝을 선택

5. 최종 충돌 해소 단계(resolve_final_conflicts) 수행

이 방식은 구현이 단순하고 저밀도 환경에서 계산 오버헤드가 낮다는 장점이 있으나, 시간-공간 충돌을 완전하게 모델링하지 않으므로 혼잡 환경에서는 대기 연쇄 및 교착 위험이 커질 수 있다.

```
function plan_astar(agents, map):
    next_pos      = get_current_positions(all_agents)
    sorted_agents = sort_by_priority(agents)

    for agent in sorted_agents:
        if agent.is_waiting(rotation or action):
            continue

        temp_obstacles =
            get_higher_priority_moves(agent, next_pos)
            + get_lower_priority_positions(agent, agents)

        pf   = create_pathfinder(agent.pos, agent.goal)
        path = compute_path(pf, map, temp_obstacles)
```

```
        desired_move = get_next_step(path)
        destroy_pathfinder(pf)

        next_pos[agent.id] = apply_rotation(agent, desired_move)

    resolve_final_conflicts(next_pos, sorted_agents)
    return next_pos
```

<표1-1 : pseudo-code 1>

Ⅲ-3.2 D* Lite 기반 계획 절차

D* Lite는 동적 환경에서 경로를 복구(repair)하는 방식으로 설명되며, 다중 AGV 적용에서는 “임시 장애물 통지(notify)” 후 “경로 복구(repair_path)”를 수행하는 흐름이 제시된다.

1. 우선순위 정렬 후 각 에이전트에 대해 pathfinder 유지/생성
2. 상·하위 우선순위 에이전트의 점유/이동을 임시 장애물로 구성
3. 장애물 통지 후 repair_path로 경로 갱신
4. 다음 스텝 선택 및 최종 충돌 해소 수행

D* Lite는 동일 환경에서의 반복 계획 시 경로 재활용이 가능하므로, 주차장처럼 “상태가 변하고 대기/회전이 빈번한” 상황에서 A* 대비 유리한 계산 특성을 기대할 수 있다.

```
function plan_dstar_lite(agents, map):
    next_pos      = get_current_positions(all_agents)
    sorted_agents = sort_by_priority(agents)

    for agent in sorted_agents:
        if agent.is_waiting(rotation or action):
            continue

        ensure_pathfinder_exists(agent)

        temp_obstacles =
            get_higher_priority_moves(agent, next_pos)
            + get_lower_priority_positions(agent, agents)

        notify_obstacles(agent.pathfinder, temp_obstacles)
        path = repair_path(agent.pathfinder, map)
```

```
        desired_move = get_next_step(path)
        cleanup_notify_obstacles(agent.pathfinder, temp_obstacles)

        next_pos[agent.id] = apply_rotation(agent, desired_move)

    resolve_final_conflicts(next_pos, sorted_agents)
    return next_pos
```

<표1-2 : pseudo-code 2>

Ⅲ-4. 협력 경로 계획 및 교착 처리: WHCA* + WFG/SCC + Partial CBS

본 절은 혼잡 환경에서의 운용 안정성을 확보하기 위한 핵심 설계(제안 파이프라인)를 기술한다. WHCA*는 예약 테이블 기반 협력 계획을 제공하지만, 우선순위 기반 특성상 교착 위험이 존재한다는 한계가 명시되어 있다. 따라서 본 연구는 WHCA*에 교착 탐지(WFG/SCC) 및 교착 해소(Partial CBS)를 결합한다.

Ⅲ-4.1 WHCA* 1차 계획(Reservation Table 기반)

WHCA*는 3차원 예약 테이블 W=[time][x][y]를 사용하여 제한된 호라이즌(0 < time < H) 내에서 시간-공간 점유를 예약한다. 또한 우선순위 기반 계층적 계획(Hierarchical)과 협력(Cooperative) 특성을 가지며, 타 에이전트 예약 노드를 장애물로 간주한다.

Ⅲ-4.2 교착 탐지: WFG 및 SCC

WFG는 대기 관계를 표현하는 그래프로, 정점은 AGV 집합, 간선은 “Ai waits for Aj” 관계로 정의된다. SCC는 그래프 내 상호 도달 가능한 정점 집합이며, 크기 2 이상 SCC의 존재를 교착으로 판정한다. 즉, 교착 탐지는 “대기 관계 그래프의 순환(강연결 구조)” 검출로 귀결된다.

Ⅲ-4.3 교착 해소: Partial CBS 및 Fallback

CBS는 지역적 충돌 해소를 위해 효율적인 MAPF 알고리즘이며, 소수 에이전트 간 충돌을 효율적으로 해결하여 계산 효율을 향상시키는 것으로 설명된다. 본 연구에서는 전체 에이전트에 대해 CBS를 수행하는 대신, SCC로 검출된 교착 집단에 Partial CBS

를 적용하여 국소적으로 충돌을 해소한다. 교착 해소 절차는 발표자료에 다음과 같이 정리되어 있다.

1. WFG/SCC로 교착 감지
2. SCC 그룹에 Partial CBS 적용
3. 실패 시 우선순위 기반 해결(리더 전략)
4. 우선순위 실패 시 pull-over(리더 제외 AGV 후진)

WHCA* 전체 의사코드는 “WHCA* 1차 계획 → WFG 구성 및 SCC 탐지 → Partial CBS 협상(성공 시 overwrite, 실패 시 leader moves) → 최종 충돌 해소”의 흐름으로 구성되어 있다.

```
function plan_WHCAstar(agents, map):

    // 1) WHCA* (First attempt)
    rt = create_reservation_table()
    wf_edges = []
    sorted_agents = sort_by_priority(agents)

    for agent in sorted_agents:
        path = dstar_find_path(agent.pf, h_steps, rt)

        if has_conflict(path) with other_agent:
            add_wait_edge(wf_edges, agent, other_agent)
            provisional_plan[agent.id] = agent.pos         // wait
        else:
            reserve_path(rt, path)
            provisional_plan[agent.id] = path[1]           // move 1 step

    // 2) Deadlock detection (WFG/SCC)
    sccmask = detect_scc(wf_edges)

    // 3) Deadlock resolution (Partial CBS)
    if sccmask > 0:
        deadlocked_group = get_agents_from_mask(sccmask)

        negotiated_plans, ok = run_partial_cbs(deadlocked_group, rt)

        if ok == true:
            overwrite_next_pos(provisional_plan, negotiated_plans)
        else:
```

```
            fallback_leader_moves(provisional_plan, deadlocked_group)
    else:
        // no deadlock: keep provisional_plan

    // 4) Final
    resolve_final_conflicts(provisional_plan, sorted_agents)
    return provisional_plan
```

<표1-3 : pseudo-code 3>

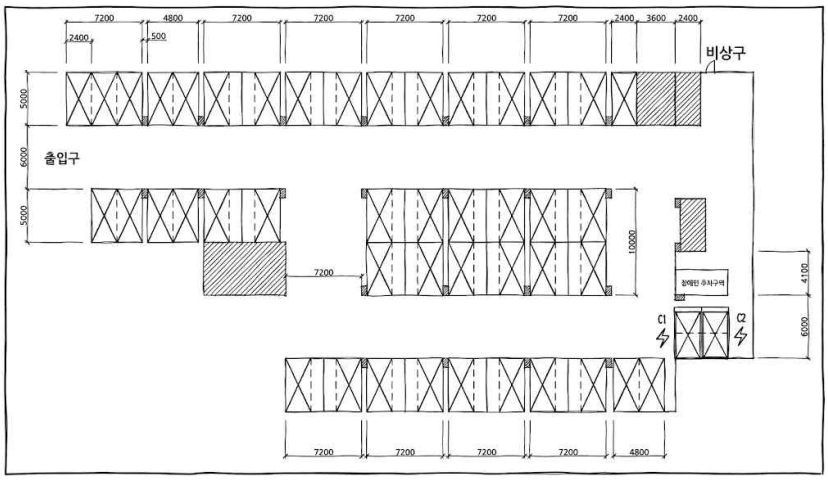
Ⅲ-5. 알고리즘 선택 및 비교 전략

본 연구는 환경 밀도 변화에 따른 알고리즘 특성을 비교하기 위해, A*, D* Lite, WHCA*+WFG/SCC+CBS의 세 가지 접근을 동일 시뮬레이션 환경에서 평가한다(발표자료 결론에서 저밀도/고밀도별 유리 알고리즘을 구분). 이를 통해 “저밀도에서는 단순 탐색기가 계산 오버헤드가 낮아 유리할 수 있으나, 고밀도에서는 협력 계획과 교착 처리 모듈이 처리량 및 계산 효율을 개선한다”는 가설을 실험적으로 검증하는 구조를 갖는다.

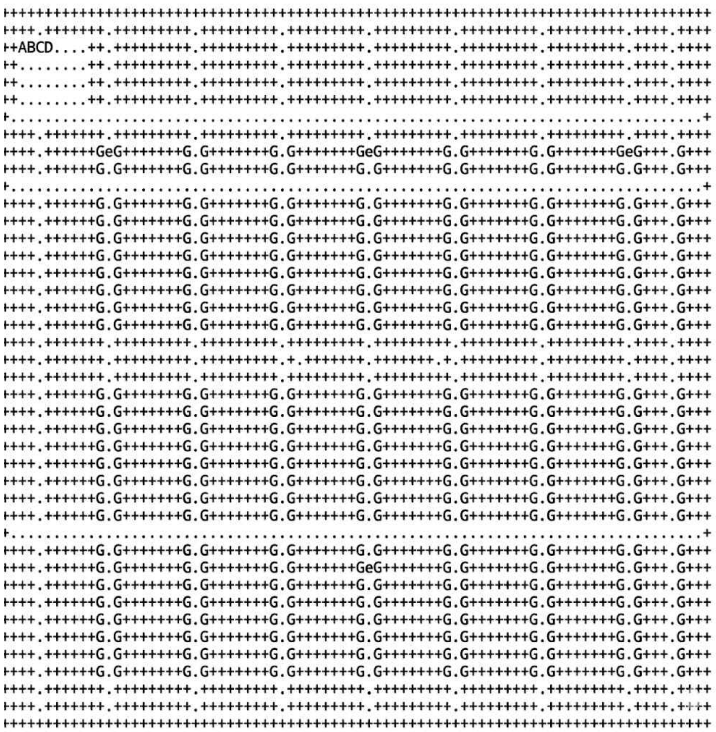
Ⅳ. 모의실험 결과 및 토의

Ⅳ-1 실험 환경 및 시나리오 구성

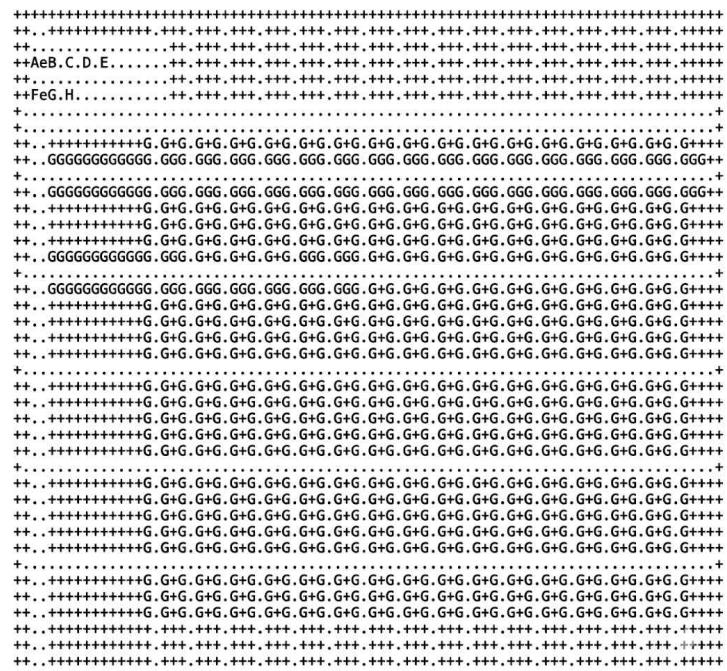
본 연구에서는 제안한 AGV 제어/경로계획 알고리즘의 성능을 검증하기 위해, 주차장 환경을 그리드 맵으로 모델링한 시뮬레이션을 구성하였다.



<그림1 : MAP1>



<그림2 : MAP2>



<그림3 : MAP3>

실험 맵은 (1) MAP1(저밀도, 형남공학관 주차장), (2) MAP2(격자구조, 중밀도, 대형마트), (3) MAP3(격자구조, 고밀도, 대형마트)로 구성되며, 각각의 맵에서 AGV 대수 및 주차(점유) 규모를 달리하여 혼잡도 변화에 따른 성능 변화를 확인하였다. 맵 별 파라미터는 다음과 같이 설정하였다: MAP1은 Parking 61 기준으로 AGV 1~4대를 변화시키며 측정하였고, MAP2는 AGV 4대/ Parking 390, MAP3는 AGV 8대/ Parking 900 조건으로 실험하였다.

비교 대상 알고리즘은 (1)단일 에이전트 기반 경로 탐색인 A*, (2)재계획 기반의 D* Lite, 그리고 (3)다중 AGV 환경을 고려한 WHCA* + WFG/SCC + (Partial) CBS 결합 알고리즘(이하 “제안 알고리즘”)이다.

IV-2. 성능 평가 지표(메트릭) 정의

본 실험의 평가는 “서비스 관점(얼마나 많이 처리했는가)”과 “연산 관점(얼마나 비싸게 계산했는가)”를 동시에 확인할 수 있도록 다음 지표들을 사용한다.

1. 처리량(Throughput)

처리량은 전체 물리 시간(시뮬레이션 time step) 대비 완료한 작업(Task)의 비율로 정의한다. 구현 상에서는 tasks_completed_total / recorded_steps로 계산되며, 출력 포맷 또한 “task / total physical time(step)”으로 정의되어 있다.

2. 총 이동 비용(Total Movement Cost)

AGV가 실제로 이동한 총 누적 비용을 셀(cell) 단위로 누적한 값으로 정의하며, 시뮬레이터 성능 요약 출력에서 “Total Movement Cost (cells)”로 집계한다.

3. 탐색/연산 메트릭(계산 복잡도 관점)

경로계획 과정의 연산 부담을 보기 위해 다음을 기록한다.;

Nodes Expanded (total): 확장된 노드 수

Heap Moves (total): 우선순위 큐(힙) 조작량

Generated Nodes (total): 생성된 후보 노드

Valid Expansions (total): 유효한 완화(relaxation) 횟수

Valid Expansion Ratio: valid / generated 비율

알고리즘의 계획 비용을 정량적으로 비교하기 위한 목적이다.

IV-3. 시뮬레이션 결과

IV-3.1 혼잡도(저밀도 vs 고밀도)에 따른 처리량 비교

실험 결과, 저밀도 환경에서는 A* 및 D* Lite가 상대적으로 높은 처리량을 보였고, 고밀도 환경에서는 제안 알고리즘(WHCA* + WFG/SCC + CBS)이 가장 높은 처리량을 보였다.

이 결과는 다중 에이전트 협조 기법의 특성과 일치한다. 즉, 저밀도에서는 충돌/교착 자체가 드물기 때문에, 예약 테이블 관리·교착 감지·협상(CBS) 등의 오버헤드가 상대적으로 “불필요한 비용”으로 작용한다. 반면 고밀도에서는 교착 및 충돌 회피가 성능의 병목이 되므로, 제안 알고리즘의 “교착 감지 + 국소 협상 기반 재계획” 구조가 대기/정지를 줄여 총 완료 작업 수를 증가시키는 방향으로 작동한다.

	AGV 대수	제안 알고리즘	A*	D* Lite
MAP1	1	0.0133	0.0144	0.0144
	2	0.0260	0.0284	0.0283
	3	0.0340	0.0403	0.0402
	4	0.0426	0.0494	0.0502
MAP2	4	0.0221	0.0217	0.0216
MAP3	8	0.0503	0.0500	0.0500

[표2-1 : Throughput]

IV-3.2 이동 비용(Total Movement Cost) 비교

이동 비용은 처리량과 함께 “운영 효율”을 나타낸다. 처리량이 높더라도 이동 비용이 과도하게 증가하면, 실제 시스템에서는 에너지 소모 및 마모 증가, 평균 처리 시간 증가로 이어질 수 있다. 따라서 본 연구에서는 처리량 우위 여부와 함께 이동 비용의 증감을 함께 비교하였다

일반적으로 고밀도 조건에서는 충돌 회피 및 교착 해소 과정에서 대기(wait) 또는 우회(detour)가 증가할 수 있는데, 제안 알고리즘은 교착 그룹에 한정하여 CBS 기반 협상을 수행함으로써 “무의미한 대기”를 줄이고, 결과적으로 이동 비용의 증가를 억제하는 방향으로 설계되었다.

	AGV 대수	제안 알고리즘	A*	D* Lite
MAP1	1	6269	6260	6260
	2	6510	6354	6358
	3	7228	6307	6263
	4	7669	6416	6374

MAP2	4	54800	57026	57562
MAP3	8	108535	111124	111560

[표2-2 : Movement Cost]

IV-3.3 연산량(탐색/힙 조작) 비교

연산 메트릭 관점에서는, 동일 조건에서 “어떤 알고리즘이 가장 적은 계산을 요구하는가”에 대한 결론이 제시되었으며, 실험 요약에서는 제안 알고리즘, D* Lite, A* 순으로 적은 계산량을 요구한다.

이는 구현 관점에서 다음과 같이 해석할 수 있다.

1) A*: 매 계획 시점에서 목표까지의 경로를 새로 탐색하며(재사용 약함), 혼잡도가 증가할수록 확장 노드 수 및 힙 연산이 크게 증가할 수 있다.

2) D* Lite: 환경 변화/재계획에 강점이 있어, 완전 재탐색 대비 연산을 절약할 수 있지만, 전역적인 충돌 및 교착 해소로 인해 연산이 크게 증가할 수 있다.

3) 제안 알고리즘(WHCA*+WFG/SCC+CBS): 기본은 WHCA*의 윈도우 기반 계획으로 문제 크기를 제한하고, 교착이 “발생한 부분”에만 CBS를 적용하기 때문에, 전역 최적화 대비 계산을 억제하면서도 고밀도에서의 충돌/교착을 효과적으로 처리한다.

	AGV 대수	제안 알고리즘	A*	D* Lite
MAP1	1	17437	176450	17437
	2	17398	178357	18886
	3	25000	228913	35689
	4	29235	298061	55849
MAP2	4	320044	7789182	1210534
MAP3	8	1040963	19552004	3284817

[표2-3 : Node Extension]

	AGV 대수	제안 알고리즘	A*	D* Lite
MAP1	1	115974	995969	115074
	2	108232	1020243	125947
	3	144242	1229512	244396
	4	162110	1663126	372340
MAP2	4	2419374	53850530	11599644
MAP3	8	6855586	132902023	33334079

[표2-4 : Heap Moves]

	AGV 대수	제안 알고리즘	A*	D* Lite
MAP1	1	36484	430873	39469
	2	34386	430874	51173
	3	43072	510499	84884

	4	46795	624886	126104
MAP2	4	549232	14884414	2265680
MAP3	8	1927573	38189125	6906435

[표2-5 : Generate Node]

	AGV 대수	제안 알고리즘	A*	D* Lite
MAP1	1	17437	176450	17437
	2	15293	178357	17749
	3	17712	228913	29676
	4	17675	298061	42230
MAP2	4	286296	7789182	741036
MAP3	8	1013709	19552004	2006326

[표2-6 : Valid Expansion]

	AGV 대수	제안 알고리즘	A*	D* Lite
MAP1	1	0.4779	0.4097	0.4418
	2	0.4458	0.4139	0.3468
	3	0.4112	0.4484	0.3496
	4	0.3777	0.4770	0.3349
MAP2	4	0.5213	0.5233	0.3271
MAP3	8	0.5259	0.5120	0.2905

[표2-7 : Valid Expansion Ratio]

IV-4. 토의(Discussion)

1. 상황 적응형 알고리즘 선택의 필요성

본 실험의 핵심 결론은 “단일 알고리즘이 모든 조건에서 항상 최적이 아니다”는 점이다. 저밀도 환경에서는 단순 경로계획(A*, D* Lite)이 오버헤드가 적어 유리하며, 고밀도 환경에서는 다중 AGV 협조 및 교착 해소 전략이 포함된 제안 알고리즘이 유리하다.

2. 교착(Deadlock) 처리의 효과

고밀도 환경에서 처리량이 크게 갈리는 주요 원인은 “충돌 회피”보다 “교착 상태에서의 진행 불가”에 가깝다. 제안 알고리즘은 WFG & SCC로 교착을 감지하고, 교착 그룹에 Partial CBS를 적용해 국소 협상을 수행한다는 점에서, 단순히 우선순위만으로 밀어내는 방식 대비 안정적으로 교착을 해소할 수 있다.

3. 실시간성 관점의 장점

실제 주차장 AGV 시스템은 “최단거리”보다 “멈추지 않고 계속 처리”가 중요한 경우가 많다. 제안 알고리즘은 전역 CBS처럼 계산이 폭증하는 구조가 아니라, 교착 그

룹에 한정된 협상을 수행하도록 설계되므로, 고밀도 환경에서 처리량 향상과 계산량 억제라는 두 목표를 동시에 달성할 가능성이 있다.

Ⅴ. 결 론

Ⅴ-1. 연구 요약

본 논문은 무인 주차장 환경에서 다수 AGV(Automated Guided Vehicle)가 동시에 운용될 때 발생하는 충돌(conflict) 및 교착(deadlock) 문제를 완화하기 위해, 다중 에이전트 경로 탐색(MAPF) 관점에서 AGV 제어 알고리즘을 구현하고 시뮬레이션을 통해 검증하였다. MAPF는 시간(Time)을 포함한 공유 환경에서 여러 에이전트가 충돌 없이 이동하는 경로 집합을 찾는 문제로 정의되며, 정점 충돌과 간선 충돌이 대표적인 제약이다. 또한 주차장과 같은 협소·혼잡 환경에서는 충돌 회피 과정의 대기(wait)가 연쇄적으로 발생하며 교착으로 이어질 수 있으므로, 단일 최단경로 탐색만으로는 안정적 운용이 어렵다.

이를 해결하기 위해 본 연구는 우선순위 기반 제어를 기반으로 (1) A*, (2) D* Lite, (3) WHCA + WFG/SCC + Partial CBS* 결합 알고리즘(이하 제안 알고리즘)을 구현하였다. WHCA*는 예약 테이블을 이용하는 협력적 MAPF 알고리즘이지만 우선순위 기반 특성상 교착 위험이 존재한다는 한계가 제시된다. 따라서 본 연구는 WFG(Wait-For Graph)와 SCC(Strongly Connected Components)를 통해 교착을 탐지하고, 교착 그룹에 대해서만 Partial CBS를 적용하여 국소적으로 충돌을 해소하며, 실패 시 우선순위 기반 fallback 및 pull-over로 복구하는 파이프라인을 구성하였다.

Ⅴ-2. 실험 결과 및 핵심 결론

실험은 형남공학관 지하2층 주차장 (MAP1)과 MAP1보다 높은 밀도에서의 운용을 점검하기 위한 대형마트 구조의 확장 맵(MAP2, MAP3)에서 수행되었으며, 맵 밀도 및 AGV 수를 변화시키며 처리량과 연산량 메트릭을 비교하였다(예: Parking 61/390/900, AGV 최대 8). 결과는 다음의 핵심 결론으로 정리된다.

1. 혼잡도에 따라 최적 알고리즘이 달라진다.

저밀도 환경에서는 A* 및 D* Lite가 높은 처리량을 보인 반면, 고밀도 환경에서는 제안 알고리즘(WHCA* + WFG/SCC + CBS)이 가장 높은 처리량을 달성하였다.

2. 고밀도 환경에서는 교착 처리 모듈이 성능을 좌우한다.

혼잡도가 높아질수록 단순 충돌 회피보다 “교착으로 인한 진행 불가”가 처리량 저하의 주요 원인이 되며, WFG/SCC 기반 교착 탐지와 Partial CBS 기반 국소 협상이 정지/대기를 줄이는 방식으로 작동한다.

3. 연산 효율 측면에서도 제안 알고리즘이 유리하다.

동일 조건에서의 계산량 비교에서 제안 알고리즘이 가장 적은 계산을 요구하며, 그 다음이 D* Lite, 마지막이 A로 정리된다. 이는 WHCA의 윈도우 기반 계획으로 문제 규모를 제한하고, 교착이 발생한 부분에만 CBS를 적용하는 “부분 최적화” 구조가 전역 최적화 대비 계산 폭증을 억제하기 때문으로 해석할 수 있다.

V-3. 한계점

본 연구는 시뮬레이션 기반으로 알고리즘을 검증하였으며, 실제 로봇 운용 환경의 제약이 완전히 반영되지는 않았다. 예를 들어 실제 AGV에서는 센서 오차, 통신 지연, 동역학 제약(가감속), 안전거리 기반 정지/재개 정책 등이 성능에 영향을 줄 수 있다. 또한 WHCA*의 호라이즌 길이, stuck 임계치, CBS 적용 상한값 등 파라미터에 따라 성능이 달라질 수 있으며, 이는 환경(맵 구조, 요청 분포, AGV 수)에 따라 최적 값이 달라질 가능성을 의미한다.

V-4. 향후 연구

향후 연구는 다음 방향으로 확장될 수 있다.

1. 현실 제약 반영 및 실제 플랫폼 검증

프로젝트 계획에서도 S/W 시뮬레이션 외에 TurtleBot3 및 ROS 기반 실제 구현이 제시되므로, 실제 플랫폼에서 센서/통신/동역학 제약을 포함한 검증이 필요하다.

2. 파라미터 자동 최적화 및 적응형 제어

혼잡도 변화에 따라 고정 호라이즌/고정 임계치가 항상 최적이 아닐 수 있으므로, 작업 요청 밀도나 교착 빈도에 따라 WHCA* 호라이즌, CBS 적용 조건, stuck 부스트 등을 동적으로 조절하는 적응형 정책을 설계할 수 있다.

3. 요청 분포 및 대규모 시나리오 확장

현재는 주차/출차 요청 생성 조건을 기준으로 평가하였으나, 피크타임(버스트), 균일,

특정 구역 집중 등 현실적인 요청 분포로 시나리오를 확장하고, 더 큰 규모(AGV 수 증가, 맵 복잡도 증가)에서의 성능/안정성 평가가 필요하다.

참고문헌

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100 – 107, 1968.

[2] A. Stentz, “Optimal and efficient path planning for partially-known environments,” in Proc. IEEE International Conference on Robotics and Automation (ICRA), 1994.

[3] S. Koenig and M. Likhachev, “Fast replanning for navigation in unknown terrain,” IEEE Transactions on Robotics, vol. 21, no. 3, pp. 354 – 363, 2005.

[4] D. Silver, “Cooperative pathfinding,” in Proc. AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), 2005.

[5] Z. Bnaya and A. Felner, “Conflict-Oriented Windowed Hierarchical Cooperative A*,” in Proc. IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 3743 – 3748.

[6] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, “Conflict-based search for optimal multi-agent pathfinding,” Artificial Intelligence, vols. 219 – 220, pp. 40 – 66, 2015.

[7] Z. Ren, S. Rathinam, and H. Choset, “Multi-objective Conflict-based Search for multi-agent pathfinding,” in Proc. IEEE International Conference on Robotics and Automation (ICRA), 2021.

[8] J. Kottiger, S. Almogor, and M. Lanjian, “Conflict-based search for multi-robot motion planning with kinodynamic constraints,” in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022.
[9] G. S. Ho and C. V. Ramamoorthy, “Protocols for deadlock detection in distributed database systems,” IEEE Transactions on Software Engineering, vol. SE-8, no. 6, pp. 554 - 557, 1982.

[10] M. Sidi, A. Segall, and D. Hsu, “Local distributed deadlock detection by cycle detection and clustering,” IEEE Transactions on Software Engineering, vol. SE-13, no. 1, pp. 77 - 86, 1987.

[11] R. E. Tarjan, “Depth-first search and linear graph algorithms,” in Proc. IEEE 12th Annual Symposium on Switching and Automata Theory, 1971.

[12] A. Stentz, “The Focused D* algorithm for real-time replanning,” in Proc. International Joint Conference on Artificial Intelligence (IJCAI), 1995.

[13] A. Bhosekar, A. Işık, T. Ekşioğlu, S. Gilstrap, and R. Allen, “Simulation-optimization of automated material handling systems in a healthcare facility,” IISE Transactions on Healthcare Systems Engineering, vol. 11, no. 4, pp. 316 - 337, 2021.

[14] S. Koenig, M. Likhachev, and D. Furcy, “Lifelong Planning A*,” Preprint submitted to Elsevier Science on May 24, 2005.

[15] S. Ning, Z. Zhong, and X. Zheng, “Design of virtual intelligent parking lot system based on signal request mechanism,” in Proc. 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Dalian, China, 2021.

[16] S. Shi, Y. Pan, Y. Sun, X. Xie, R. Chen, W. Shen, and S. Shen, “Collaborative

planning of parking spaces and AGVs path for smart indoor parking system," in Proc. 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design (CSCWD), Dalian, China, 2018.

[17] V. Bobanac and B. Bogdan, "Routing and scheduling in multi-AGV systems based on dynamic banker algorithm," in Proc. 16th Mediterranean Conference on Control and Automation, 2008, pp. 1168 - 1173.

[18] 김종기, 박병운, 최진우, 「중소형 공동주택 지하주차장 환기개방성 향상 방안연구: 환기방식 실태조사를 중심으로」, 대한주택공사 주택연구소, 1997.