

# 속도 프로파일과 회전 비용을 고려한 다중 에이전트 경로 탐색

## Multi-Agent Path Finding With Velocity Profile and Rotation Cost

신 승 민<sup>1</sup>, 어 규 호<sup>2\*</sup>  
 (Seungmin Shin<sup>1</sup> and Gyuho Eoh<sup>2,\*</sup>)

<sup>1</sup>Department of Electronics Engineering, Tech University of Korea

<sup>2</sup>Department of Mechatronics Engineering, Tech University of Korea

**Abstract:** This paper presents Continuous-time Conflict-Based Search with Velocity Profile and Rotation Cost (CCBS-VP-RC) for multi-agent path finding under kinematic constraints. Continuous-time Conflict-Based Search (CCBS) is a two-level algorithm that resolves agent conflicts through constraint generation in continuous time domains. However, CCBS does not consider real-world kinematic constraints such as velocity profiles, angular velocity limits, or a rotation cost matrix for differential drive robots. The rotation cost matrix precomputes the rotation time for all heading transitions, enabling efficient cost retrieval during path planning. The CCBS-VP-RC integrates velocity profiles with rotation costs to generate physically feasible trajectories that satisfy kinematic bounds. Simulations were conducted on grid-based environments with varying obstacle densities and agent populations. Compared to traditional Multi-Agent Path Finding (MAPF) algorithms, the CCBS-VP-RC has slightly longer makespan and flowtime due to kinematic constraints. Nevertheless, the proposed method achieves higher success rates in complex environments, demonstrating its effectiveness for practical multi-robot systems operating under real-world kinematic constraints.

**Keywords:** conflict-based search, differential drive agent, multi-agent path finding, rotation cost, velocity profile

### 1. 서론

최근 인공지능 및 로봇 기술의 발전으로 다중 에이전트 시스템이 산업 현장에 광범위하게 도입되면서 MAPF (Multi-agent path finding)의 중요성이 급격히 증가하고 있다. 특히 수백 대의 로봇이 동시 작업하는 물류 창고 자동화에서 MAPF는 충돌 방지와 작업 효율 극대화를 위해 필수적이며 그 외에도 자율주행 교통수단, 디지털 엔터테인먼트 등 다양한 분야에서 핵심 기술로 활용되고 있다[1-3].

MAPF 알고리즘의 주요 성능 지표는 최적성(optimality)과 강인함(robustness)이다. 최적성은 전체 에이전트의 최종 완료 시간(makespan) 또는 총 이동 거리(flowtime)를 최소화하는 것으로 시스템의 효율과 작업 완료 시간을 결정한다. 강인함은 다양한 장애물이나 환경 변화에 대한 대응 능력이다. MAPF에서는 일반적으로 최적의 해를 도출하는 것이 중요한 문제이지만 실제 응용에서는 에이전트의 물리적 제약을 만족하는 실행 가능한 해를 도출하는 것이 더욱 중요하다. MAPF 연구가 발전함에 따라 해의 최적성을 최대한 유지하면서 실제 환경에서의 실행 가능성을 확보하기 위한 연구가 활발히 진행되고 있다[4-6].

한편, CBS [7], ICTS (Increasing Cost Tree Search) [8]와 같은 기존의 전통적인 MAPF 알고리즘은 모든 행동의 지속 시간이 동일하고 시간이 이산적으로 구분된다는 단순화된 가정을 기반으로 한다. 이러한 가정은 실제 다중 에이전트 시스템 운용 시 발생하는 다양한 물리적 제약을 충분히 반영하지 못

하는 한계를 갖는다. 차동구동형 에이전트의 경우, 가속 및 감속 과정에서의 속도 프로파일과 회전 동작 시의 각속도 제약과 같은 운동학적 특성이 경로 계획에 중요한 영향을 미치지만 전통적인 MAPF에서는 해당 물리적 제약을 고려하지 않는다. 이를 보완하기 위해 차량형 에이전트를 고려한 CL-CBS (Car-Like Conflict-Based Search) 알고리즘이 제안되었으나 이 또한 에이전트의 속도 프로파일과 각속도 제약을 명시적으로 반영하지 못한다는 한계가 존재한다[9].

속도 프로파일과 회전 비용 고려의 필요성을 나타내는 예시는 그림 1과 같다. 그림 1은 동일한 경로(Start→Turn→Goal)를 이동할 때, CCBS의 이론적 시간 비용과 속도 프로파일(VP) 및 회전 비용(RC)을 반영한 실제 시간 비용이 어떻게 달라지는지를 비교한 것이다. 시간 변수  $t_a$ 와  $t_b$ 는 각각 Segment 1과 Segment 2의 이론적 CCBS 시간 비용을 의미한다.  $t_{a+i}$ 와  $t_{b+i}$ 는 속도 프로파일을 반영하여 Segment 1과 Segment 2의 이동 시간이 증가한 값을 나타내며  $t_j$ 는 Turn 단계에서 발생하는 회전 시간 비용을 의미한다. 에이전트는 Start 지점에서 시작 시간  $t_0$ 에서 출발하여 Segment 1 (직선 이동) → Turn (방향 전환) → Segment 2 (직선 이동)을 거쳐 Goal 지점에 도착한다고 가정한다. 그림 1(a)는 이론적 CCBS의 시간 비용을 나타낸다. CCBS에서는 각 Segment 1,2의 이동 시간 비용  $t_a, t_b$ 만 계산되며 Turn 단계에서는 회전 비용을 고려하지 않기 때문에 추가적인 시간 증가가 없다. 따라서 총 이동 시간은  $t_a + t_b$ 이다. 그림 1(b)는 속도 프로파일을

\* Corresponding Author

Manuscript received November 24, 2025; revised December 15, 2025; accepted December 15, 2025

신승민: 한국공학대대학교 전자공학부 학부생(smshinkr2@tukorea.ac.kr, ORCID<sup>®</sup> 0009-0002-9801-2976)

어규호: 한국공학대대학교 메카트로닉스공학부 교수(gyuho.eoh@tukorea.ac.kr, ORCID<sup>®</sup> 0000-0003-4931-4396)

※ 본 연구는 경기도의 경기도지역협력연구센터(GRRC) 사업의 일환으로 수행하였음 [GRRC한국공대2023-B03].

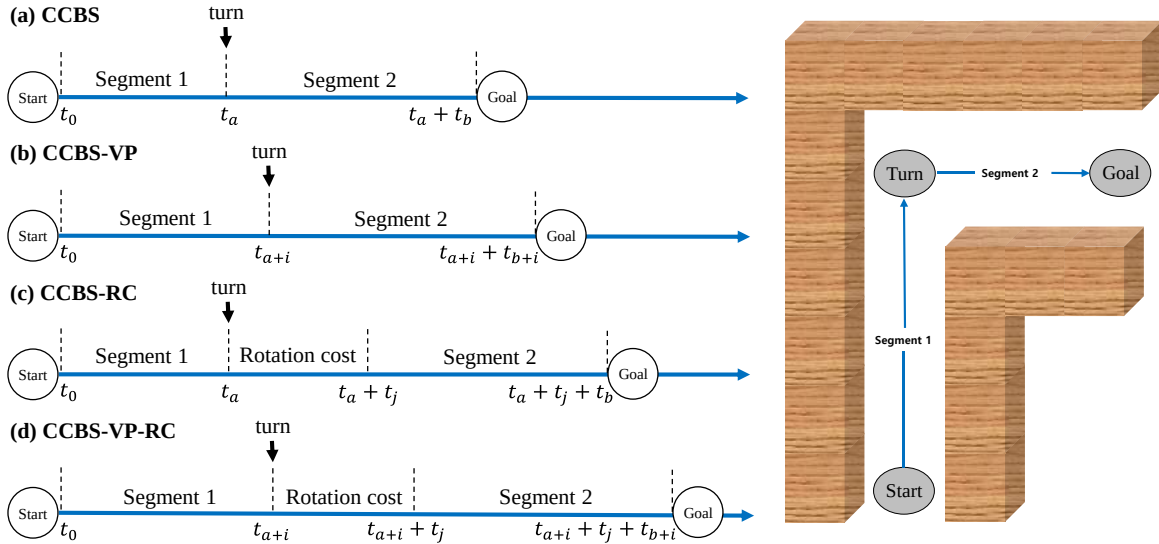


그림 1. 동일 경로 (Start→Turn→Goal)를 이동할 때 이론적 CCBS 시간 비용과 운동학적 제약 (속도 프로파일·회전 비용)을 반영한 시간 비용 비교. (a) 이론적 CCBS 시간 비용. (b) 속도 프로파일 (VP)을 반영한 통합 시간 비용. (c) 회전 비용 (RC)을 반영한 통합 시간 비용. (d) 속도 프로파일과 회전 비용을 모두 반영한 통합 시간 비용.

Fig. 1. Comparison of the theoretical CCBS time cost and the time costs incorporating kinematic constraints (velocity profile and rotation cost) for the same path (Start→Turn→Goal). (a) Theoretical CCBS time cost. (b) Integrated time cost considering velocity profile (VP). (c) Integrated time cost considering rotation cost (RC). (d) Integrated time cost considering both VP and RC.

고려한 경우이다. 실제 에이전트는 즉시 최대 속도에 도달할 수 없기 때문에 가속을 포함한 Segment 1의 이동 시간은  $t_{a+i}$ 로 증가하고 Segment 2의 이동 시간도  $t_{b+i}$ 로 증가한다. 이에 따라 전체 이동 시간은  $t_{a+i} + t_{b+i}$ 가 된다. 속도 프로파일만 반영하며 회전 시간 비용은 고려되지 않는다. 그림 1(c)는 회전 시간 비용을 반영한 경우이다. 방향 전환 시 실제 회전 동작에서는 각속도에 따른 회전 시간 비용이 발생한다. 이론적 CCBS에서 회전 비용만 고려하여 두 직선 구간의 이동 시간은 기존  $t_a$ 와  $t_b$ 이며 Turn 단계에서 회전 시간  $t_j$ 가 추가된다. 그림 1(d)는 속도 프로파일과 회전 시간 비용이 모두 반영된 경우이다. Segment 1, 2는 속도 프로파일을 고려한  $t_{a+i}, t_{b+i}$ 가 되며 Turn 단계에서 회전 시간 비용  $t_j$ 가 발생한다. 전체 이동 시간은  $t_{a+i} + t_j + t_{b+i}$ 로 네 가지 상황 중 가장 길다. 그림 1에서 살펴본 바와 같이 실제 에이전트의 운동학적 제약을 반영했을 때 CCBS 이론적 시간 비용과 실제 실행 시간 사이의 차이가 크게 벌어질 수 있으며 이러한 시간 지연 때문에 CCBS의 결과를 그대로 사용 시 충돌이 발생할 수 있다.

따라서 본 논문에서는 에이전트의 속도 프로파일과 각속도 제약을 명시적으로 고려한 CCBS-VP-RC (Continuous-time CBS with Velocity Profile and Rotation Cost) 알고리즘을 제안한다. 제안하는 알고리즘은 기존 CCBS를 기반으로 하되, 차동구동형 에이전트의 운동학적 제약을 추가 반영하여 실행 가능한 경로를 계획한다. 본 연구의 기여는 다음과 같다.

- 1) 차동구동형 에이전트의 속도 프로파일과 각속도 제약을 고려한 CCBS-VP-RC 알고리즘을 제안한다.
- 2) 에이전트의 가속-등속-감속 이동 메커니즘 반영한 속도 프로파일을 적용하여 경로의 실행 가능성을 높인다.

3) 각속도 제약에 따른 회전 시간을 사전 계산한 회전 비용 행렬을 제안하여 경로 계획 과정에서 계산 효율성을 향상시킨다.

4) 기존의 CL-CBS 차량형 알고리즘과의 비교 실험을 통해 제안하는 알고리즘의 우수성을 보여주었다.

본 연구는 다음과 같이 구성되어 있다. II장에서는 전통적인 MAPF와 운동학적 제약을 고려한 MAPF에 대한 관련 연구를 소개한다. III장에서는 MAPF 문제 정의 및 제안하는 방법의 가정을 설명한다. IV장에서는 제안하는 알고리즘의 구조와 핵심 구성 요소를 설명하고, V장에서는 시뮬레이션을 통해 제안하는 방법의 성능을 검증하고 실험 결과를 분석한다. VI장에서는 실험 결과에 대한 논의 및 고찰을 다루며, VII장은 본 연구의 결론 및 향후 연구 방향을 제시한다.

## II. 관련 연구

### 1. 다중 에이전트 경로 탐색(MAPF)

기존의 MAPF 연구들은 주로 탐색 효율 향상과 최적성 보장을 위한 알고리즘적 구조 개선에 초점을 맞추어 발전해왔다. 초기 MAPF의 대표적 접근법으로는 A\*기반의 Cooperative A\*와 Windowed Hierarchical Cooperative A\* (WHCA\*) 등이 제안되었다[10,11]. Cooperative A\*에서는 각 에이전트가 전역 지도를 공유하면서 다른 에이전트의 예약된 경로 정보를 참조하여 충돌을 회피하는 방식으로 탐색을 수행한다. WHCA\*는 이러한 협력 탐색을 일정한 시간 창(window) 단위로 제한하여 부분 경로만 계획하고 주기적으로 재탐색을 수행함으로써 계산 효율을 향상시켰다. 또한 고정된 에이전트 우선순위를 동적으로 변경하여 충돌 누적 문제를 완화하고 협력 탐색과 추상 탐색을 병행하여 부분 최적성과 전역 일관성을 동

시에 확보하였다. 그러나 이러한 휴리스틱 기반 기법들은 탐색 범위가 제한적이므로 전역 최적 경로를 보장하기 어렵고 에이전트 밀도가 높은 환경에서는 충돌이 누적되는 한계를 갖는다.

이후 연구들은 전역 최적성을 보장할 수 있는 완전 탐색 기반 접근으로 발전하였다. 대표적으로 Conflict-Based Search (CBS)는 다중 에이전트 경로 탐색을 고수준과 저수준 탐색으로 분리하여 해결하는 이중 구조를 갖는다. 고수준 탐색은 에이전트 간 충돌을 검출하고, 충돌이 발생한 시공간 영역에 대해 제약을 추가하여 탐색 트리인 제약 트리(Constraint Tree)를 분기한다. 각 노드는 제약 조건의 집합과 그에 대응하는 경로 집합을 포함한다. 저수준 탐색에서는 각 에이전트에 대해 A\* 알고리즘을 사용하여 주어진 제약을 만족하는 단일 경로를 계산한다. 이러한 계층적 구조는 충돌 발생 시 최소한의 제약만을 추가함으로써 불필요한 상태 확장을 줄이고 완전성과 최적성을 동시에 보장한다. 한편, CBS는 이산 시간과 균일한 동작 지속 시간이라는 단순화된 가정을 전제로 한다. 그러나 실제 로봇은 연속 시간 영역에서 동작하며 가속 및 감속 과정을 거쳐야 하므로 동작 지속 시간이 균일하지 않다. 따라서 이러한 가정은 실제 에이전트 시스템에 적용하는 데 한계로 작용한다.

이를 보완하기 위해 CBS 알고리즘을 확장한 CCBS (Continuous-time CBS)가 제안되었다[12,13]. CCBS는 기존 CBS의 효율적인 계층적 구조를 유지하면서도 시간을 연속적으로 처리하여 동작의 실제 지속 시간을 반영하고 충돌 지점을 정밀하게 판정한다. 그러나 CCBS 역시 에이전트의 속도 프로파일과 가속도 제약 등 운동학적 특성을 반영하지 않기 때문에 실제 에이전트 시스템의 물리적 제약들을 충분히 고려하지 못하는 한계가 있다.

## 2. 물리적 제약을 고려한 MAPF

실제 다중 에이전트 시스템에서는 대부분 차동구동형(differential drive) 또는 차량형 구조를 갖기 때문에 비홀로노믹(non-holonomic) 제약이 필수적으로 고려되어야 한다[14,15]. 이러한 에이전트들은 즉각적인 방향 전환이나 제자리 회전이 불가능하며 선속도와 각속도에 의해 운동이 제약된다. 이에 따라 이중 에이전트 및 운동학적 제약을 다루는 연구[16,17], 그룹 기반 계획을 다루는 연구[18], 그리고 모델 예측 제어를 결합한 연구[19] 등이 진행되어 왔다. 특히 연속 시간 영역에서의 충돌 회피를 다루는 연구들이 최근 활발히 이루어지고 있다[20].

초기에는 개별 에이전트의 운동 특성을 고려한 kinodynamic path planning 연구들이 제안되었고 다중 에이전트 시스템으로 확장하는 시도가 이어졌다[21-23]. 이러한 접근법은 가속도, 각속도, 회전반경 등의 상태 제약을 포함시켜 실제 실행 가능한 경로를 탐색하지만 연속 상태 공간의 차원이 증가할수록 탐색 복잡도가 기하급수적으로 커져 계산 효율 확보가 어렵다. Velocity Obstacle (VO) 기반 접근은 에이전트 간 상대 속도를 이용해 충돌 가능 영역을 직접 계산하는 방식으로 이러한 계산 부담을 크게 줄였다[24]. VO 기반 접근 동적 충돌 회피 측면에서는 우수하지만 전역 최적성을 보장하지 못하고 복잡한 환경에서는 경로 품질이 불안정해지는 한계가 있었다.

이후 전역 최적 탐색 구조를 유지하면서도 에이전트의 물리적 제약을 통합하는 방향의 CBS 기반 알고리즘들이 제시되었다. Car-Like CBS (CL-CBS)는 CBS의 제약 트리 구조를 기반으로 차량형 에이전트의 회전 반경과 각속도 제약을 제약 조건으로 포함하여 실제 실행 가능한 경로를 탐색하였다. Kinematic CBS는 각 에이전트의 속도 및 가속도 제약을 상태 공간에 명시적으로 포함시켜 물리적으로 일관된 경로를 생성하였고[25,26], db-CBS는 경로상의 불연속을 일정 경계 내에서 허용함으로써 운동학적 제약을 정밀하게 처리하였다[27]. 이들 연구는 CBS의 제약 트리 구조 위에서 점진적으로 물리적 제약을 확장해왔으나 CL-CBS는 가감속 과정의 속도 변화를 명시적으로 다루지 않았으며, Kinematic CBS와 db-CBS 역시 각속도 제약이나 속도 프로파일 전체를 계산하지 않고 불연속 지점을 근사적으로 보정하기 때문에 실제 차동구동형 에이전트의 운동 특성과는 차이가 존재한다.

본 연구에서는 연속 시간 기반의 CCBS 프레임워크에 속도 프로파일과 각속도 제약에 따른 회전 비용을 명시적으로 반영한 CCBS-VP-RC 알고리즘을 제안한다. 제안 알고리즘은 차동구동형 에이전트의 운동학적 특성을 현실적으로 반영함으로써 기존 MAPF 알고리즘들의 물리적 한계를 보완하고 다양한 장애물 환경에서의 경로 탐색 성공률을 향상시킨다.

## III. 문제 정의

다중 에이전트 경로 탐색(MAPF)은 그래프 환경에서 여러 에이전트가 각자의 시작 위치에서 목표 위치까지 서로 충돌하지 않고 이동할 수 있는 경로 집합을 찾는 문제로 정의된다[28,31]. 일반적인 목적 함수에 대해 MAPF의 최적해를 구하는 것은 에이전트 수  $k$ 에 따라 상태 공간이 기하급수적으로 증가하는 NP-hard 문제로 알려져 있다[29].

MAPF 문제는 무향 그래프  $G = (V, E)$ 로 표현되며 여기서  $V$ 는 가능한 위치들의 집합,  $E$ 는 두 위치 간 이동 가능한 경로들의 집합이다. 에이전트 집합은  $A = \{a_1, a_2, \dots, a_k\}$ 로 정의되며 각 에이전트  $a_i$ 는 시작 정점  $s_i \in V$ 와 목표 정점  $g_i \in V$ 를 갖는다. 에이전트는 주어진 시간 단계마다 이동(move) 또는 대기(wait) 동작을 수행할 수 있으며 동일한 정점을 동시에 점유하거나 서로 교차하는 간선을 같은 시점에 이동하는 경우 충돌로 간주된다. 에이전트  $a_i$ 의 경로  $\pi_i$ 는 일련의 동작 시퀀스로 표현되며 전체 경로 집합  $\Pi = \{\pi_1, \pi_2, \dots, \pi_k\}$ 이 충돌이 발생하지 않을 때 이를 MAPF의 해(solution)라고 한다.

기존의 전통적인 MAPF는 모든 동작의 지속 시간이 동일하고 시간이 이산적으로 구분된다는 단순화된 가정을 전제로 한다. 이러한 가정은 이론적 분석이나 최적성 검증에는 유리하지만 실제 에이전트 시스템의 물리적 제약을 충분히 반영하기 어렵다는 근본적인 한계를 갖는다. 특히 실제 다중 에이전트 시스템은 보통 차동구동형 구조를 가지며 이들은 비홀로노믹 특성으로 인해 즉각적인 방향 전환이 불가능하며 선속도와 각속도의 제약을 받는다. 또한 가속 및 감속 과정에서 속도의 시간적 변화가 발생한다. 이로 인해 실제 에이전트의 주행 경로는 전통적인 MAPF에서 가정하는 이산 이동 모델과 운동학적으로 불일치하게 된다.

본 연구에서는 실제 에이전트의 물리적 제약을 반영한 MAPF 문제 설정을 가정한다. 에이전트는 차동구동형 구조를 가지며 비홀로노믹 제약으로 인해 선속도와 각속도의 결합에 따라 방향 전환이 제한된다. 이동 환경은 격자 기반의 이산 공간으로 구성하고 에이전트는 각 정점에서 인접한 8개 정점으로 이동할 수 있다. 모든 에이전트는 동일한 최대 속도 및 등가속도 조건을 가진다. 회전 시 최대 각속도 ( $\omega_{max}$ )는  $\frac{\pi}{2} rad/s$  이고, 해당 최대 각속도 범위에서 각가속도가 전체 회전 시간에 미치는 영향은 거의 미미하므로 각가속도는 무시한다. 이러한 가정은 실제 물류 창고 환경에서 운용되는 차동구동형 에이전트들의 적절한 각속도 제약과 실제 환경에서 운용되는 로봇의 운동학적 특성을 적절히 반영하고 있다.

#### IV. 속도 프로파일과 각속도 제약을 고려한 CCBS

본 논문에서는 기존 CCBS의 2단계 탐색 구조를 유지하면서도 실제 차동구동형 에이전트의 운동학적 특성을 반영할 수 있도록 개선한 CCBS-VP-RC 알고리즘을 제안한다. 제안된 방법은 에이전트 이동 시의 속도 프로파일과 각속도 제약에 따라 발생하는 회전 비용을 반영하여 실제 다중 에이전트 시스템에서의 경로 실행 가능성을 향상시키는 것을 목표로 한다.

##### 1. CCBS 알고리즘 개요

CCBS는 기존 CBS의 2단계 계층적 탐색 구조를 가지며, 이 구조를 유지하면서 세 가지 측면에서 확장하였다. 첫째, 충돌 검사를 위해 에이전트의 기하학적 형상을 고려한 충돌 검사 기법을 사용한다. 둘째, 충돌 해결 시 위치-시간 쌍 대신 행동-시간 쌍에 대한 제약을 추가한다. 셋째, 저수준 탐색에서 CCBS 제약을 처리하는 SIPP (Safe Interval Path Planning)를 사용한다[30].

고수준 탐색에서는 제약 트리를 구성하여 탐색을 수행한다. 제약 트리의 각 노드  $N$ 은  $N = (\Pi, C)$  로 정의된다. 여기서  $\Pi$ 는 각 에이전트의 경로 집합,  $C$ 는 제약 집합이다. 경로 탐색 과정은 다음과 같다.

- 1) 초기 노드를 생성하고  $A^*$  기반의 저수준 탐색을 수행하여 각 에이전트별로 최단 경로를 구한다.
- 2) 모든 경로 조합에서 충돌이 존재하는지 검사한다.
- 3) 충돌이 발견되면 해당 충돌을 유발한 두 에이전트  $i, j$ 에 대해 각각 새로운 제약을 추가한 두 개의 자식 노드를 생성한다.
- 4) 새로 생성된 노드들에 대해 다시 저수준 탐색을 수행하여 충돌 없는 경로를 찾는다.
- 5) 충돌이 없는 노드에 도달하면 그 노드의 경로 집합이 MAPF의 해가 된다.

이 과정은 우선순위 큐(priority queue)를 이용한 형태로 수행되며, 저수준 탐색이 최적 경로를 반환한다면 탐색 종료 시점에서 충돌이 없는 최소 비용 해를 보장한다.

저수준 탐색에서는 주어진 제약 조건을 만족하면서 단일 에이전트의 최적 경로를 계산한다. CCBS에서는 연속시간 SIPP를 기반으로 하며 SIPP의 탐색 상태는 정점  $v$ 와 안전 구간  $[t, t']$ 의 쌍  $(v, [t, t'])$ 로 표현된다. 연속 시간 상에서 두

에이전트의 이동 세그먼트가 동일한 시간 구간을 공유하면서 서로의 반지름 합 이하로 접근하면 충돌로 판단한다. 충돌이 발생하면 CCBS는 해당 동작에 대해 충돌 시간 구간을 금지하는 행동-시간 쌍 제약을 생성하여 제약 트리를 분기한다. 주어진 제약으로부터 안전 구간을 계산하며 제약을 위반하지 않는 경로를 탐색한다.

결과적으로, CCBS는 연속 시간 기반의 충돌 검사를 통해 기존 CBS의 이산 시간 한계를 개선하고 2단계 탐색 구조를 유지하면서 다중 에이전트 간의 충돌을 효과적으로 해결한다. 그러나 CCBS도 에이전트의 가속과 감속 과정이나 회전 동작 시 생기는 시간 지연을 반영하지 않아 실제 시스템에서는 경로의 실행 가능성이 제한될 수 있다. 본 연구에서는 CCBS의 기본 구조를 유지하면서도 에이전트의 속도 프로파일과 각속도 제약을 명시적으로 고려하여 실행 가능한 경로를 생성하는 CCBS-VP-RC 알고리즘을 제안한다.

##### 2. 속도 프로파일을 고려한 비용 계산

실제 에이전트의 이동은 일정한 속도가 아닌 가속구간, 등속구간, 감속구간의 연속적인 운동 과정으로 구성된다. 기존 CCBS 및 CL-CBS 알고리즘은 이러한 물리적 특성을 고려하지 않기 때문에 실제 에이전트의 속도 변화에 따른 시간 지연을 정확히 반영하지 못한다. 이를 보완하기 위해 본 연구에서는 저수준 탐색 단계에서 이동 구간의 길이에 따라 가속구간, 등속구간, 감속구간을 구분하는 속도 프로파일을 적용하였다.

그림 2와 같이 이동 거리  $c$ 가 충분히 길 경우, 에이전트는 최대 속도  $v_{max}$ 에 도달한 후 일정 구간 동안 등속으로 이동하므로 속도-시간 그래프는 사다리꼴 형태(trapezoidal case)를 갖는다. 이동 거리가 짧아 최대 속도에 도달하지 못하는 경우에는 가속과 감속 구간으로만 이루어진 삼각형 형태(triangular case)의 속도-시간 그래프를 형성한다. 이러한 두 가지 이동 형태에 따른 총 이동 시간  $t_{total}$ 은 각 구간의 시간적 면적을 모두 합산한 값으로 다음과 같이 정의된다.

$$t_{total} = \begin{cases} 2\sqrt{\frac{c}{a}}, & c \leq 2s_a \\ \frac{v_{max}}{a} + \frac{c-2s_a}{v_{max}}, & c > 2s_a \end{cases} \quad (1)$$

이 때,  $a$ 는 가속도이고  $s_a = \frac{v_{max}^2}{2a}$ 는 최대속도에 도달하기 위한 거리이다.  $v_{max}$ 는 에이전트가 이동 중 도달할 수 있는 최대 속도를 의미한다. 속도-시간 그래프에서 이동거리  $c$ 는 각 구간의 면적에 해당하므로 식 (1)은 가속구간-등속구간-감속구간 단계에서의 시간적 면적 합을 통해 총 이동 시간을

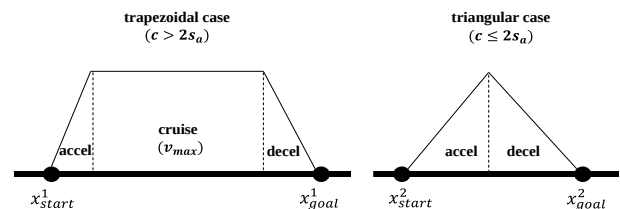


그림 2. 속도 프로파일을 고려한 시간 비용.

Fig. 2. Velocity profile temporal cost.

**Algorithm 1:** Velocity Profile Temporal Cost

---

**Input:**  $s_{\text{start}}, s_{\text{goal}}$ : start and goal states of an agent  
 $c$ : Euclidean distance between  $s_{\text{start}}$  and  $s_{\text{goal}}$   
 $a^+, a^-$ : acceleration and deceleration ( $a^+ = a^-$ )  
 $s_a^+$ : distance needed to reach  $v_{\text{max}}$  with  $a^+$   
**Output:**  $g(s_{\text{goal}})$ : updated cumulative cost of  $s_{\text{goal}}$

```

1 if  $c > 2s_a^+$  then
2   trapezoidal case  $\leftarrow$  area of trapezoid:
3    $t_{\text{total}} \leftarrow t_{a^+} + t_{v_{\text{max}}} + t_{a^-}$ 
4 else
5   triangular case  $\leftarrow$  area of triangle:
6    $t_{\text{total}} \leftarrow t_{a^+} + t_{a^-}$ 
7  $g(s_{\text{goal}}) \leftarrow g(s_{\text{start}}) + t_{\text{total}}$ 
8 return  $g(s_{\text{goal}})$ 

```

---

산출하는 관계식을 나타낸다. 이와 같이 정의된  $t_{\text{total}}$ 은 저 수준 탐색 단계에서 각 이동의 시간 기반 이동 비용으로 사용되며 이를 누적함으로써 에이전트의 총 이동 시간을 계산한다.

Algorithm 1은 이러한 시간 비용 계산 과정을 나타낸다. 총 이동 시간  $t_{\text{total}}$ 과 시간 비용을 결정짓는 프로파일의 형태는 이동 거리  $c$ , 에이전트의 가속도  $a^+$ 와 최대속도  $v_{\text{max}}$ 에 의해 결정된다. 이동 거리  $c$ 가 임계 거리  $2s_a^+$  보다 클 경우, 에이전트는 가속 과정을 거쳐 최대 속도에 도달한 후 일정 시간 동안 등속 구간을 유지하고 다시 감속 과정을 거치는 사다리꼴 프로파일(trapezoidal case)을 따른다(1-3줄). 반면  $c \leq 2s_a^+$  경우, 가속 단계를 진행하다가 최대 속도에 도달하기 전에 감속 단계로 전환되는 삼각형 프로파일(triangular case)이 형성된다(5-6줄). 이동 거리  $c$ 와 에이전트의 속도 파라미터를 입력으로 받아 두 형태의 속도 프로파일을 결정하고 이를 g-value 누적 과정에 반영하여 총 이동 시간을 계산한다(7-8줄). 두 형태의 프로파일은  $c = 2s_a^+$ 에서 연속적으로 연결되어 실제 에이전트의 속도 변화 연속적으로 연결되어 실제 에이전트의 속도 변화를 부드럽게 근사한다.

### 3. 각속도를 고려한 회전 비용 행렬

실제 에이전트의 이동은 단순히 직선 이동만으로 구성되지 않는다. 차동구동형 에이전트는 두 바퀴의 회전 속도 차이를 통해 방향을 전환하며 이러한 회전 동작에는 각속도에 따른 시간이 소요된다. 본 연구에서는 그리드 기반 이산 경로 환경의 특성을 고려하여 방향 전환 시 제자리에서 회전 완료 후 직선 이동을 이어서 수행하는 방식을 적용하였다. 이 때, 차동구동형 에이전트의 제자리 회전 시간은 회전해야 하는 각도와 각속도에 의해 결정된다.

#### 3.1 방향 벡터의 이산화

연속적인 방향 벡터를 처리하기 위해 이를  $N$ 개의 이산 방향으로 양자화하는 방법을 구현하였다. 본 연구에서는 그리드 환경의 8방향 연결성을 가정하여  $N = 8$ 로 설정하였으며 이는 360도를 45도 간격으로 균등 분할한 동서남북 4개 방향과 4개 대각선 방향을 포함한다.

에이전트의 이동 방향 벡터  $v = (v_i, v_j)$ 가 주어졌을 때, 먼저 4사분면 역탄젠트 함수를 이용하여 방향 각도를 계산한다. 이는 벡터의 부호를 고려하므로 계산 결과는  $(-\pi, \pi]$  범위의 라디안 값을 반환한다. 계산된 각도  $\theta$ 는 다음과 같이  $[0, 2\pi)$  범위로 정규화하였다.

$$\theta' = \begin{cases} \theta, & \text{for } \theta \geq 0 \\ \theta + 2\pi, & \text{for } \theta < 0 \end{cases} \quad (2)$$

정규화된 각도  $\theta'$ 를 이산 방향 인덱스로 변환하기 위해 각도를  $\frac{2\pi}{N}$  간격으로 나누고 반올림하여 가장 가까운 방향 인덱스  $d_{\text{index}}$ 를 결정한다.

$$d_{\text{index}} = \left\lfloor \frac{\theta' \cdot N}{2\pi} + 0.5 \right\rfloor \bmod N \quad (3)$$

여기서 mod는 modulo 연산으로 각도가  $2\pi$ 를 초과하거나 음수가 되더라도 해당 각도가 속하는 방향 인덱스를 0부터  $N - 1$  범위 내에서 계산하도록 한다. 방향 벡터의 크기가 임계값  $\varepsilon = 10^{-8}$  이하인 경우는 정지 상태로 간주하여 -1을 반환한다. 이러한 양자화 과정을 통해 연속적인 방향 정보를 8개의 이산 상태로 단순화하였으며 회전 비용 계산 시 일관된 인덱스 참조가 가능케 하였다. 계산된  $d_{\text{index}}$ 는 0부터 7까지에 대해 각각  $0^\circ, 45^\circ, 90^\circ, 135^\circ, 180^\circ, 225^\circ, 270^\circ, 315^\circ$ 에 대응된다.

#### 3.2 회전 비용 매트릭스

매 탐색마다 회전 비용을 계산하면 연산 부하가 커지므로 8방향 회전 비용에서 발생 가능한 모든 회전 비용을 사전에 계산하여 매트릭스 형태로 저장하는 방법을 제안하였다. 방향 인덱스  $v_i$ 에서  $v_j$ 로 회전할 때 에이전트는 시계 방향 또는 반시계 방향 중 더 짧은 경로를 선택한다. 두 방향 인덱스 간의 차이를 고려하여 최단 회전 각도  $\Delta\theta_{ij}$ 를 결정하며 에이전트의 각속도  $\omega$ 가 주어졌을 때 방향 인덱스  $v_i$ 에서  $v_j$ 로의 회전 시간  $t_{\text{rot}}$ 은 다음과 같이 정의된다.

$$t_{\text{rot}}(v_i, v_j) = \frac{\Delta\theta_{ij}}{\omega} \quad (4)$$

회전 비용 매트릭스  $\mathcal{R} \in \mathbb{R}^{8 \times 8}$ 은 모든 방향 쌍에 대한 회전 시간을 사전에 계산하여 저장한다.

$$\mathcal{R}[i][j] = t_{\text{rot}}(i, j), \quad \forall i, j \in \{0, 1, \dots, 7\} \quad (5)$$

여기서  $i$ 와  $j$ 는 각각 출발 방향과 도착 방향을 나타낸다. 본 연구에서 설정한 각속도  $\omega = \frac{\pi}{2} \text{ rad/s}$ 를 기준으로 각 회전 각도에 대한 비용은  $45^\circ$  회전 시 0.5초,  $90^\circ$  회전 시 1.0초,  $135^\circ$  회전 시 1.5초,  $180^\circ$  회전 시 2.0초로 계산되며 동일 방향을 유지하는 경우 회전이 발생하지 않으므로 회전 비용이 발생하지 않는다.

표 1. 8방향 회전 비용 매트릭스(단위: step).

Table 1. 8-Direction Rotation Cost Matrix (Unit: steps).

| $t_{\text{rot}}(i, j)$ | 0( $\rightarrow$ ) | 1( $\nearrow$ ) | 2( $\uparrow$ ) | 3( $\nwarrow$ ) | 4( $\leftarrow$ ) | 5( $\swarrow$ ) | 6( $\downarrow$ ) | 7( $\searrow$ ) |
|------------------------|--------------------|-----------------|-----------------|-----------------|-------------------|-----------------|-------------------|-----------------|
| 0                      | 0.0                | 0.5             | 1.0             | 1.5             | 2.0               | 1.5             | 1.0               | 0.5             |
| 1                      | 0.5                | 0.0             | 0.5             | 1.0             | 1.5               | 2.0             | 1.5               | 1.0             |
| 2                      | 1.0                | 0.5             | 0.0             | 0.5             | 1.0               | 1.5             | 2.0               | 1.5             |
| 3                      | 1.5                | 1.0             | 0.5             | 0.0             | 0.5               | 1.0             | 1.5               | 2.0             |
| 4                      | 2.0                | 1.5             | 1.0             | 0.5             | 0.0               | 0.5             | 1.0               | 1.5             |
| 5                      | 1.5                | 2.0             | 1.5             | 1.0             | 0.5               | 0.0             | 0.5               | 1.0             |
| 6                      | 1.0                | 1.5             | 2.0             | 1.5             | 1.0               | 0.5             | 0.0               | 0.5             |
| 7                      | 0.5                | 1.0             | 1.5             | 2.0             | 1.5               | 1.0             | 0.5               | 0.0             |

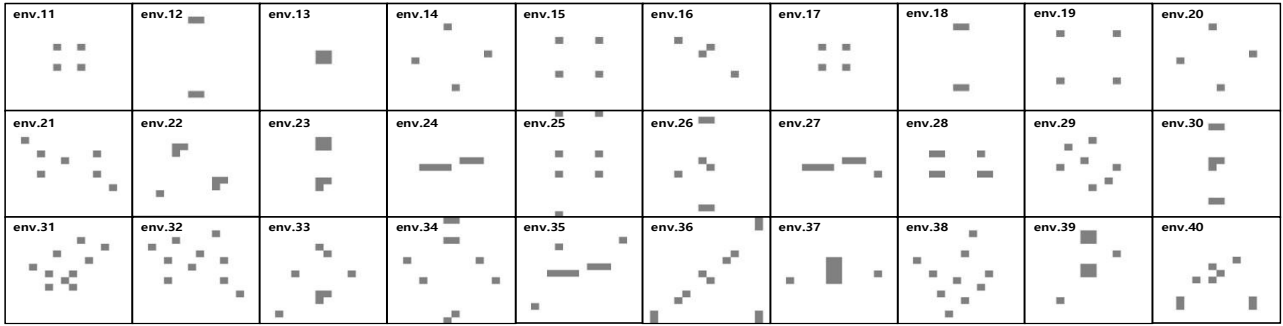


그림 3. 저밀도, 중간, 고밀도  $16 \times 16$  환경의 장애물 구성.

Fig. 3. Obstacle configurations of Sparse, Medium, and Dense  $16 \times 16$  environments.

모든 방향 쌍에 대한 회전 비용 매트릭스는 표 1과 같다. 8 방향 회전 비용 매트릭스는 대칭 행렬의 형태를 가지며  $\mathcal{R}[i][j] = \mathcal{R}[j][i]$  관계가 성립한다. 이는 특정 방향 쌍 간의 회전 비용이 회전방향에 관계없이 동일함을 나타낸다.  $i = j$ 인 경우에는 회전이 필요하지 않아 회전 비용이 0으로 계산된다. 이렇게 계산된 회전 비용 매트릭스는 탐색 과정에서 회전 동작이 필요한 경우 즉시 참조된다. 이를 통해 매 탐색 단계마다 반복적으로 소요되는 회전 비용 연산 시간을 줄일 수 있다.

## V. 시뮬레이션

### 1. 시뮬레이션 환경

제안한 CCBS-VP-RC 알고리즘의 성능을 검증하기 위해 기존의 CCBS [12]와 CL-CBS [9] 알고리즘을 대상으로 비교 시뮬레이션을 수행하였다. 본 시뮬레이션의 목적은 제안 알고리즘이 실제 차동구동형 에이전트의 운동학적 제약을 반영하여 복잡한 환경에서도 안정적인 경로를 생성할 수 있는지를 평가하는 데 있었다. 실험은 모든 알고리즘에 동일한 초기 조건과 탐색 파라미터를 적용하여 수행하였으며 비교 지표로는 makespan, flowtime, success rates를 사용하였다. 이를 통해 단순한 경로 최적성뿐 아니라 실제 환경에서의 실행 가능성을 함께 검증하였다. 각 실험 결과는 다양한 환경 복잡도와 에이전트 수 변화에 따른 알고리즘의 성능 변화를 중심으로 비교하였다.

시뮬레이션은  $16 \times 16$  격자 환경 기반에서 장애물 밀도에 따라 Empty, Sparse, Medium, Dense의 네 가지 환경으로 구분하였다(그림 3). Empty 환경(env.1~env.10)은 장애물이 없는 빈 환경이기 때문에 그림 3에서는 생략하였다. 각 환경은 장애물 개수를 각각 0, 4, 7, 10개로 설정하여 복잡도를 점진적으로

증가시켰으며 장애물의 위치는 각 환경마다 무작위로 배치하였다. 각 환경에서 에이전트의 출발 위치와 목표 위치는 환경의 가장자리에 중 서로 반대편에 위치하도록 무작위로 배치되되, 장애물과 겹치지 않도록 설정하였다. 이러한 배치 방식은 에이전트 간 경로가 서로 교차하도록 유도하여 충돌 회피 상황을 효과적으로 생성하였다. 동일한 환경 내에서 에이전트 수가 증가할 때는 이전 에이전트의 출발 및 목표 위치를 유지한 채 새로운 에이전트를 추가하는 누적 방식을 사용하였다. 단, 환경이 변경될 때는 장애물 배치가 달라지므로 에이전트의 출발 및 목표 위치도 새로 가장자리에 무작위로 생성하였다. 이를 통해 각 환경의 특성에 맞는 다양한 충돌 회피 시나리오를 구성하였다.

### 2. 시뮬레이션 결과

그림 4은  $16 \times 16$  환경에서 장애물 밀도와 에이전트 수를 달리하여 세 알고리즘의 경로 계획 성능을 비교한 결과이며 경로 탐색에 실패한 경우는 그래프에 값을 표기하지 않았다. 표 2는 네 가지 환경에서 에이전트와 알고리즘에 따른 경로 계획 성공률을 나타낸다.

CCBS는 모든 환경에서 가장 짧은 makespan, flowtime을 기록하였지만 이동 시간 계산 시 가속·감속 및 회전 비용을 고려하지 않기 때문에 수치상 우수한 성능이 실제 주행에서 동일하게 재현된다고 보기 어렵다. CL-CBS는 차량형 이동 제약을 포함하여 회전 반경이나 진행 방향 제약과 같은 기본적인 비홀로노믹 특성은 반영하지만 실제 주행에서 중요한 가속·감속 과정, 각속도 제약에 따른 회전 시간을 고려하지 않는다. 제안된 방법은 가속·감속과 회전 시간을 포함해 이동 시간을 계산하므로 makespan과 flowtime이 상대적으로 길어지지만 환경 변화에 따른 성능 편차가 작고 성공률이 안정적으로 유지되었다.

표 2.  $16 \times 16$  환경에서의 성공률.

Table 2. Success rates for  $16 \times 16$  environments.

| Method | empty 16x16      |                  |                  | Sparse 16x16     |                  |                  | Medium 16x16     |                  |                  | Dense 16x16      |                  |             |
|--------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|-------------|
|        | Number of Agents |                  |                  |                  |                  |                  |                  |                  |                  |                  |                  |             |
|        | 2                | 4                | 6                | 2                | 4                | 6                | 2                | 4                | 6                | 2                | 4                | 6           |
| CCBS   | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 9/10 (90 %) |
| CL-CBS | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 8/10<br>(80 %)   | 8/10<br>(80 %)   | 10/10<br>(100 %) | 6/10<br>(60 %)   | 5/10<br>(50 %)   | 10/10<br>(100 %) | 9/10<br>(90 %)   | 5/10 (50 %) |
| Ours   | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 10/10<br>(100 %) | 8/10<br>(80 %)   | 10/10<br>(100 %) | 10/10<br>(100 %) | 8/10 (80 %) |

Empty 환경에서는 장애물이 없어 알고리즘 간 간섭이 최소화된 조건이므로 2대 에이전트 기준에서는 세 알고리즘의 makespan과 flowtime 차이가 가장 작게 나타났다. 그러나 에이전트 수가 점점 증가하면 동일한 빈 공간에서도 경로 교차와 충돌 회피가 필요한 상황이 늘어나고 이에 따라 방향 전환이 추가적으로 발생하였다. 방향 전환이 증가하면 실제 주행에서는 회전 비용과 가속·감속 구간이 누적되기 때문에 이러한 요소를 반영하는 제안 방법은 makespan과 flowtime이 상대적으로 높게 나오는 경향을 보였다.

Sparse 환경(env.11~env.20)에서는 장애물이 추가되면서 환경의 복잡도가 증가하였다. 특히 4대 에이전트 조건부터는 장애물과 에이전트의 경로 복잡도 증가하며 CL-CBS에서 제한 시간 내 경로 탐색 실패가 나타났다. env.16의 경우 4개의 장애물이 환경 중앙에 사각형 형태로 밀집 배치되어 에이전트들이 이 장애물 군집을 우회하거나 간격 사이를 통과해야 하는 상황이 발생하였다. 이러한 환경에서는 경로 선택에 따라 회전 동작의 빈도와 가속·감속 구간의 길이가 크게 달라진다. 제안된 방법은 정확한 회전 비용과 속도 프로파일을 반영하여 회전 횟수를 최소화하면서 충돌 없는 경로를 생성하였다. 그 결과 에이전트 4대 조건에서 기존 차량형 알고리즘인 CL-CBS는 경로 탐색에 실패하였으나 제안된 방법은

경로 탐색을 완료하여 실행 가능한 해를 도출하였다.

Medium 환경(env.21~env.30)에서는 장애물이 7개로 늘어나 좁은 통로와 복잡한 교차부가 형성되었다. CL-CBS는 환경이 더 복잡해짐에 따라 탐색 시간이 크게 늘어 제한 시간 내 경로를 찾지 못하는 경우가 증가하였다. Env.29의 경우 7개의 장애물이 대각선 방향을 따라 비대칭적으로 분산 배치되어 환경 내 복잡도가 증가하였다. CL-CBS는 이러한 복잡한 교차 상황에서 각속도 제약과 속도 프로파일을 고려하지 않아 에이전트 4대와 6대 조건에서 경로 탐색에 모두 실패하였다. 반면, 제안된 방법은 에이전트의 운동학적 제약을 명시적으로 반영하여 동일 조건에서 모두 경로 탐색에 성공하였으며 복잡한 Medium 환경 전체에서 평균 93% 이상의 성공률을 달성하였다.

Dense 환경(env.31~env.40)에서는 장애물이 10개까지 증가하여 회전·정지·제가속이 빈번하게 발생하였다. 4대 이상의 에이전트 조건부터는 CL-CBS 알고리즘이 차량형 제약을 반영하면서 복잡한 경로 탐색을 성공하지 못하는 한계를 보였다. Env.35의 경우 중앙부에 수평으로 긴 장애물이 배치되어 환경을 상하로 분할하는 형태를 이루었으며 나머지 장애물들이 주변에 분산 배치되어 복잡도가 더욱 증가하였다. 이러한 구조에서는 에이전트들이 중앙 장애물을 우회하면서



그림 4. 16x16 환경에서의 MAPF 알고리즘 성능 비교(빈, 저밀도, 중간, 고밀도).

Fig. 4. Performance comparison of MAPF solvers on 16x16 environments (empty, sparse, medium, dense).

동시에 분산된 장애물들을 피해야 하므로 연속적인 회전이 빈번하게 발생하였다. CL-CBS는 이러한 고밀도 환경에서 경로 탐색에 실패하였으나 제안된 방법은 가속·감속 구간과 각속도 제약을 명시적으로 반영하여 경로 생성에 성공함으로써 고밀도 환경에서도 실행 가능한 해를 도출할 수 있음을 입증하였다.

종합적으로 제안한 CCBS-VP-RC는 환경 복잡도 증가와 에이전트 수 확대에 따라 회전 및 속도 변화가 빈번하게 발생하는 상황에서도 운동학적 제약을 반영하여 높은 성공률을 유지하였다. CL-CBS는 차량형 제약을 반영함으로써 CCBS보다 현실적인 경로를 생성하지만 속도 프로파일과 각속도에 기반한 회전 비용을 고려하지 않아 복잡한 환경에서 탐색 실패가 빈번하게 발생하였다. 반면, 제안된 방법은 CCBS 구조를 기반으로 속도 프로파일 및 각속도 제약을 명시적으로 반영하여 실제 이동 시간을 더 정밀하게 계산하였다. 그 결과 CL-CBS 대비 makespan과 flowtime은 전반적으로 증가하였으나 다양한 환경에서 더 높은 성공률을 확보하였다. 특히 일부 복잡한 환경에서는 제안된 방법이 CL-CBS보다 makespan과 flowtime 측면에서도 우수한 성능을 보여 운동학적 제약을 고려한 경로 계획이 성공률뿐만 아니라 경로 효율성 향상에도 기여할 수 있음을 확인하였다.

전체 환경을 기준으로 제안한 CCBS-VP-RC는 기존 알고리즘들에 비해 최적성은 약 14.6% 손실되었으나 실제 에이전트의 운동학적 특성을 효과적으로 반영함으로써 다양한 환경에서 약 14.9% 높은 성공률을 달성하였다. 이는 제안한 알고리즘이 실제 다중 에이전트 시스템에서 실행 가능성 높은 경로를 제공할 수 있음을 보여준다.

## VI. 논의 및 고찰

본 연구에서는 제안한 CCBS-VP-RC 알고리즘의 성능을 기존의 최적 알고리즘 CCBS와 차량형 에이전트를 고려한 CL-CBS와 비교하였다. 시뮬레이션 결과, 제안 알고리즘의 makespan과 flowtime은 CCBS보다 다소 증가하였으나 이는 실제로 로봇의 물리적 제약을 반영한 결과로 해석된다. 기존 CCBS는 모든 이동을 등속으로 가정하며 회전 동작에 따른 시간 지연이나 각속도 제한을 고려하지 않는다. 따라서 CCBS가 산출한 경로는 이론적으로는 최적일 수 있으나 실제 환경에서는 가속·감속 과정 및 회전 구간에서의 시간 불일치로 인해 충돌이 발생할 가능성이 높다.

CCBS-VP-RC 알고리즘은 이동 과정의 가속구간·등속구간·감속구간을 포함하는 속도 프로파일과 회전 동작의 각속도 제약을 명시적으로 반영하였다. 이를 통해 이론적 최적성에서 일부 손실을 감수하는 대신 실제 환경에서의 실행 가능성을 확보하였다. 또한 회전 동작을 8방향으로 이산화하고 각속도에 따른 회전 비용을 사전 계산하여 행렬 형태로 구현함으로써 탐색 복잡도를 완화하였다.

CL-CBS와의 비교 결과 CCBS-VP-RC는 속도 변화와 회전 비용을 모두 포함하면서도 CL-CBS 대비 makespan과 flowtime에서의 성능 저하 폭이 작거나 일부 환경에서는 더 효율적인 경로를 생성하기도 하였다. 이러한 결과는 제안 알고리즘이 운동학적 제약을 반영하면서도 최적성과 실행 가능성 사이의 균형을 효과적으로 달성했음을 의미한다.

결과적으로 CCBS-VP-RC는 CCBS보다 수치적 최적성은 다소 낮지만 에이전트의 운동학적 제약을 반영함으로써 실제 환경에서 적용 시 발생할 수 있는 시간 불일치와 충돌 가능성을 줄였다. 또한 차량형 CL-CBS가 고려하지 못한 속도 프로파일과 각속도 제약을 포함하여 실제 차동구동형 로봇의 운동 특성을 반영한 경로를 계획할 수 있었다. 이는 이론적 최적성에 집중한 기존 접근법들과 달리, 실제 적용성을 중심으로 한 현실지향적 MAPF 확장 구조를 제시한다는 점에서 의미가 있다.

## VII. 결론

본 논문에서는 차동구동형 에이전트의 실제 운동학적 제약을 반영하기 위해 속도 프로파일과 회전 비용을 통합한 CCBS-VP-RC 알고리즘을 제안하였다. 제안된 알고리즘은 기존 CCBS의 효율적인 이중 탐색 구조를 유지하면서 이동 구간의 가속·등속·감속 속도 프로파일과 회전 시 각속도 제약을 명시적으로 반영하여 실제 환경에서도 실행 가능한 경로를 생성한다. 시뮬레이션 결과, 제안된 CCBS-VP-RC는 여러 장애물과 에이전트 환경에서 평균 약 97%의 성공률을 달성하였다. 속도 프로파일과 각속도 제약을 반영함에 따라 makespan과 flowtime이 상대적으로 증가하였으나 이는 경로의 실제 실행 가능성을 고려한 결과로 볼 수 있다. 제안된 알고리즘은 복잡한 장애물 환경과 다수의 에이전트 구성에서도 기존 차량형 알고리즘 대비 높은 성공률을 보여주었다.

본 연구 결과는 이론적 최적성에 초점을 맞춘 기존 접근법과 달리 물리적 제약을 반영하여 실제 환경에서 실행 가능성이 높은 MAPF 해를 도출할 수 있음을 보여준다. 따라서 제안된 방법은 물류 창고 자동화, 산업용 자율 로봇 시스템, 실시간 다중 로봇 협업 환경 등 실제 응용 분야에서 효과적으로 활용될 수 있을 것으로 기대된다. 향후에는 방향 연결성을 세분화하여 연속적인 회전과 자유 이동에 근접한 경로 탐색을 구현함으로써 실행 가능성이 높고 부드러운 경로를 계획할 수 있도록 개선할 예정이다.

## REFERENCES

- [1] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. K. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding in large-scale warehouses," *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, pp. 11272–11281, May 2021.
- [2] S. Varambally, J. Li, and S. Koenig, "Which MAPF model works best for automated warehousing?," *Proc. of the international symposium on combinatorial search*, vol. 15, no. 1, pp. 190–198, Jul. 2022.
- [3] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Persistent and robust execution of MAPF schedules in warehouses," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1125–1131, Apr. 2019.
- [4] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," *Proc. of the international symposium on combinatorial search*, vol. 5, no. 1, pp. 19–27, Sep. 2021.
- [5] J. Li, W. Ruml, and S. Koenig, "EECBS: A bounded-suboptimal search for multi-agent path finding," *Proc. of the AAAI*

- Conference on Artificial Intelligence*, vol. 35, no. 14, pp. 12353–12362, May. 2021.
- [6] L. Cohen, T. Uras, T. K. S. Kumar, and S. Koenig, “Optimal and bounded-suboptimal multi-agent motion planning,” *Proc. of the international symposium on combinatorial search*, vol. 10, no. 1, pp. 44–51, Sep. 2021.
- [7] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, “Conflict-based search for optimal multi-agent pathfinding,” *Artificial Intelligence*, vol. 219, pp. 40–66, Jan. 2015.
- [8] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, “The increasing cost tree search for optimal multi-agent pathfinding,” *Artificial Intelligence*, vol. 195, pp. 470–495, Feb. 2013.
- [9] L. Wen, Y. Liu, and H. Li, “CL-MAPF: Multi-agent path finding for car-like robots with kinematic and spatiotemporal constraints,” *Robotics and Autonomous Systems*, vol. 150, p. 103997, Jan. 2022.
- [10] D. Silver, “Cooperative pathfinding,” *Proc. of the aai conference on artificial intelligence and interactive digital entertainment*, vol. 1, no. 1, pp. 117–122, Sep. 2005.
- [11] Z. Bnaya and A. Felner, “Conflict-oriented windowed hierarchical cooperative A\*,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3743–3748, May. 2014.
- [12] A. Andreychuk, K. Yakovlev, E. Boyarski, and R. Stern, “Improving continuous-time conflict-based search,” *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, pp. 11220–11227, May. 2021.
- [13] A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, and R. Stern, “Multi-agent pathfinding with continuous time,” *Artificial Intelligence*, vol. 305, p. 103662, Mar. 2022.
- [14] K. Fathian, S. Safaoui, T. H. Summers, and N. R. Gans, “Robust distributed planar formation control for higher order holonomic and nonholonomic agents,” *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 185–205, Feb. 2021.
- [15] K. Yakovlev, A. Andreychuk, and V. Vorobyev, “Prioritized multiagent path finding for differential-drive robots,” *European Conference on Mobile Robots (ECMR)*, pp. 1–6, Sep. 2019.
- [16] Y. Bai, S. Kotpalliwar, C. Kanellakis, and D. Nikovski, “Multi-agent path planning based on conflict-based search (CBS) variations for heterogeneous robots,” *Journal of Intelligent & Robotic Systems*, vol. 111, no. 1, p. 26, Jan. 2025.
- [17] A. Andreychuk, “Multi-agent path finding with kinematic constraints via conflict-based search,” *Russian Conference on Artificial Intelligence*, pp. 29–45, Sep. 2020.
- [18] C. Park, S. Lee, H. Yang, D. Shin, S. Kang, and Y. Kim, “Conflict-based search with partitioned groups of agents for real-world scenarios,” *20th International Conference on Ubiquitous Robots (UR)*, pp. 986–992, Jul. 2023.
- [19] A. Tajbakhsh, L. T. Biegler, and A. M. Johnson, “Conflict-based model predictive control for scalable multi-robot motion planning,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14562–14568, May. 2024.
- [20] W. Lin, W. Song, Q. Zhu, and S. Zhu, “Multi-agent path finding with heterogeneous geometric and kinematic constraints in continuous space,” *IEEE Robotics and Automation Letters*, vol. 10, no. 1, pp. 492–499, Jan. 2025.
- [21] Z. Ren, S. Rathinam, and H. Choset, “Search Algorithms for Multi-Agent Teamwise Cooperative Path,” *Proc. of the international symposium on combinatorial search*, vol. 16, no. 1, pp. 181–182, Jul. 2023.
- [22] L. Zhang, T. Zhang, Y. An, T. Niu, S. Wang, and J. Wang, “4WIDS-MAPF: Multiagent pathfinding for four-wheel independent drive/steering robots,” *IEEE Internet of Things Journal*, vol. 12, no. 18, pp. 38795–38803, Sep. 2025.
- [23] E. Erdem, D. Kisa, U. Oztok, and P. Schüller, “A general formal framework for pathfinding problems with multiple agents,” *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 27, no. 1, pp. 290–296, Jun. 2013.
- [24] F. Vesentini, R. Muradore, and P. Fiorini, “A Survey on Velocity Obstacle Paradigm,” *Robotics and Autonomous Systems*, vol. 174, p. 104645, 2024.
- [25] W. Hönig, T. Keselman, T. S. Kumar, H. Ma, T. Ayanian, and S. Koenig, “Multi-agent path finding with kinematic constraints,” *Proc. of the International Conference on Automated Planning and Scheduling*, vol. 26, pp. 477–485, Jun. 2016.
- [26] J. Kottinger, S. Almagor, and M. Lahijanian, “Conflict-based search for multi-robot motion planning with kinodynamic constraints,” *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13494–13499, Oct. 2022.
- [27] A. Moldagalieva, J. Ortiz-Haro, M. Toussaint, and W. Hönig, “db-CBS: Discontinuity-bounded conflict-based search for multi-robot kinodynamic motion planning,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14569–14575, May. 2024.
- [28] R. Stern, “Multi-agent pathfinding: Definitions, variants, and benchmarks,” *Proc. of the international symposium on combinatorial search*, vol. 10, no. 1, pp. 151–158, Sep. 2021.
- [29] J. Yu and S. M. LaValle, “Structure and intractability of optimal multi-robot path planning on graphs,” *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 27, no. 1, pp. 1443–1449, Jun. 2013.
- [30] M. Phillips and M. Likhachev, “SIPP: Safe interval path planning for dynamic environments,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5628–5635, May. 2011.
- [31] J. Lee and H. S. Na, “Hungarian-based Heuristic for Permutation-invariant Multi-robot Path Planning,” *Journal of Institute of Control, Robotics and Systems*, pp. 286–294, Apr. 2025.



신 승 민

2020년~현재 한국공학대학교 전자공학부 학사과정. 관심분야는 다중 에이전트 시스템 및 경로 계획.



어 규 호

2009년 서울대학교 전기공학부 졸업. 2011년 동 대학원 석사. 2016년 동 대학원 박사. 2016년~2020년 LG전자 로봇선행연구소 책임연구원. 2021년 충북대학교 산업인공지능연구센터 초빙교수. 2022년~현재 한국공학대학교 메카트로닉스공

학부 교수. 관심분야는 다개체 로봇 제어 및 협업, Fusion SLAM, 강화학습.