

(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0072117

(43) 공개일자 2025년05월23일

(51) 국제특허분류(Int. Cl.)

G01C 21/34 (2006.01)

(52) CPC특허분류

G01C 21/3446 (2025.05)

G01C 21/3453 (2013.01)

(21) 출원번호 10-2023-0158941

(22) 출원일자 2023년11월16일

심사청구일자 2023년11월16일

(71) 출원인

서강대학교산학협력단

서울특별시 마포구 백범로 35 (신수동, 서강대학교)

(72) 발명자

남창주

경기도 고양시 일산서구 강선로 169, 1502동 302호 (일산동, 후곡마을15단지아파트)

심준열

서울특별시 강서구 강서로10길 169, 501호 (화곡동, 동원홈타운)

(74) 대리인

특허법인엠에이피에스

전체 청구항 수 : 총 16 항

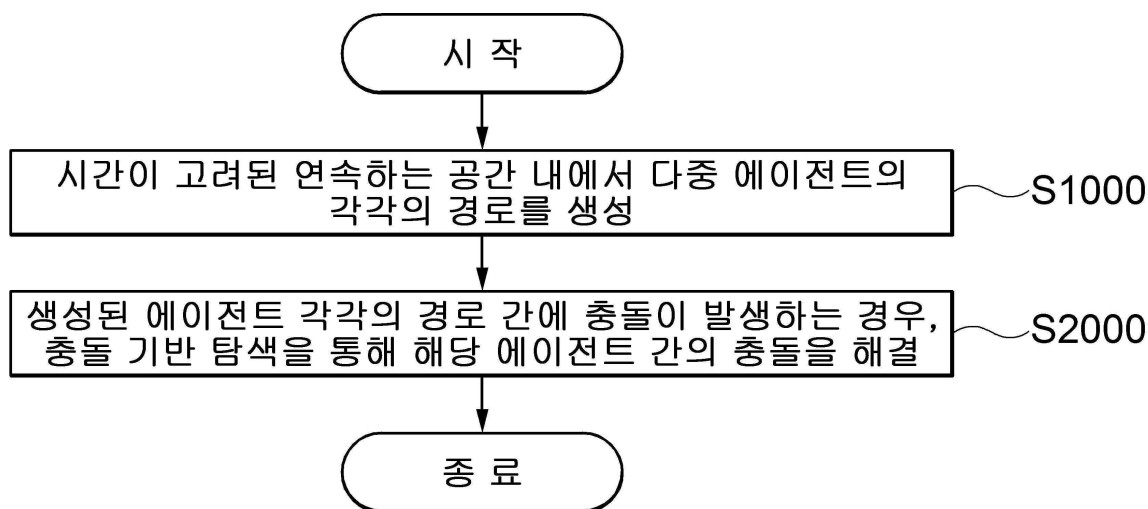
(54) 발명의 명칭 연속 공간에서의 다중 에이전트 경로 탐색을 위한 시공간 충돌 기반 검색 방법, 시스템 및 프로그램

(57) 요약

본 발명의 연속하는 공간에서의 다중 에이전트 경로 탐색을 위한 시공간 충돌 기반 경로 검색 방법은 시간이 고려된 연속하는 공간 내에서 다중 에이전트의 각각의 경로를 생성하는 제1 검색 단계 및 생성된 에이전트 각각의 경로 간에 충돌이 발생하는 경우, 충돌 기반 탐색을 통해 해당 에이전트 간의 충돌을 해결하는 제2 검색 단계를 포함할 수 있다.

본 발명에 따르면 높은 수준의 충돌 해결 과정에서 낮은 수준의 트리 표현에 직접적인 제약 조건을 부여하여 충돌을 해결하고, 이전에 생성된 트리를 다시 활용하여 다시 경로 계획을 수행함으로써 효율성을 향상시킬 수 있다.

대표도 - 도2



이 발명을 지원한 국가연구개발사업

과제고유번호	1711188873
과제번호	2022R1C1C1008476
부처명	과학기술정보통신부
과제관리(전문)기관명	한국연구재단
연구사업명	우수신진연구
연구과제명	다수 매니퓰레이터의 물체조작을 위한 협업 및 작업계획 기술
기 여 율	1/1
과제수행기관명	서강대학교
연구기간	2023.03.01 ~ 2024.02.29

명세서

청구범위

청구항 1

연속하는 공간에서의 다중 에이전트 경로 탐색을 위한 시공간 충돌 기반 경로 검색 방법에 있어서,
시간이 고려된 연속하는 공간 내에서 다중 에이전트의 각각의 경로를 생성하는 제1 검색 단계; 및
생성된 에이전트 각각의 경로 간에 충돌이 발생하는 경우, 충돌 기반 탐색을 통해 해당 에이전트 간의 충돌을
해결하는 제2 검색 단계;를 포함하는 경로 검색 방법.

청구항 2

제 1 항에 있어서,
상기 제1 검색 단계는
시간이 고려된 연속하는 공간 내 무작위 노드를 결정하는 단계;
상기 무작위 노드와 가장 가까운 노드를 선택하는 단계;
상기 가장 가까운 노드와 상기 무작위 노드 사이의 중간 노드를 추가하는 단계; 및
상기 가장 가까운 노드와 중간 노드를 연결하여 상기 경로를 생성하는 단계;를 더 포함하는 것을 특징으로 하는
경로 검색 방법.

청구항 3

제 2 항에 있어서,
상기 가장 가까운 노드와 상기 중간 노드 간의 시공간 거리가 기준값과 같은 경우, 상기 중간 노드와 상기 무작
위 노드는 동일한 시공간 위치인 것을 특징으로 하는 경로 검색 방법.

청구항 4

제 1 항에 있어서,
상기 제2 검색 단계는
충돌 감지된 CT 노드를 두 개의 자식 CT 노드로 분할하는 단계;
상기 분할된 자식 CT 노드 트리에 제약 조건을 적용하는 단계;
시공간 비용이 가장 낮은 CT 노드를 선택하는 단계;를 더 포함하는 것을 특징으로 하는 경로 검색 방법.

청구항 5

제 4 항에 있어서,
상기 시공간 비용은 시공간 거리에 비례하는 것을 특징으로 하는 경로 검색 방법.

청구항 6

제 4 항에 있어서,
상기 제약 조건을 적용하는 단계는
충돌이 감지된 CT 노드를 식별하여 무효화하는 단계;
상기 충돌이 감지된 CT 노드와 이웃하는 이웃 CT 노드를 가지치기하는 단계;
상기 충돌된 CT 노드의 자식 CT 노드를 가지치기하는 단계;

가지치기된 자식 CT 노드에서 트리를 재계획하는 단계;를 더 포함하는 것을 특징으로 하는 경로 검색 방법.

청구항 7

제 6 항에 있어서,

상기 재계획하는 단계는 상기 이웃 노드의 가지치기된 시공간 위치를 고려하여 가지치기된 자식 CT 노드에서 새로운 경로를 계획하는 것을 특징으로 하는 경로 검색 방법.

청구항 8

연속하는 공간에서의 다중 에이전트 경로 탐색을 위한 시공간 충돌 기반 경로 검색 방법을 수행하는 시스템에 있어서,

시간이 고려된 연속하는 공간 내에서 다중 에이전트의 각각의 경로를 생성하는 제1 검색 프로세스; 및

생성된 에이전트 각각의 경로 간에 충돌이 발생하는 경우, 충돌 기반 탐색을 통해 해당 에이전트 간의 충돌을 해결하는 제2 검색 프로세스;를 포함하는 시스템.

청구항 9

제 8 항에 있어서,

상기 제1 검색 프로세스는

시간이 고려된 연속하는 공간 내 무작위 노드를 결정하고, 상기 무작위 노드와 가장 가까운 노드를 선택하며, 상기 가장 가까운 노드와 상기 무작위 노드 사이의 중간 노드를 추가하고, 상기 가장 가까운 노드와 중간 노드를 연결하여 상기 경로를 생성하는 것을 특징으로 하는 시스템.

청구항 10

제 9 항에 있어서,

상기 가장 가까운 노드와 상기 중간 노드 간의 시공간 거리가 기준값과 같은 경우, 상기 중간 노드와 상기 무작위 노드는 동일한 시공간 위치인 것을 특징으로 하는 시스템.

청구항 11

제 8 항에 있어서,

상기 제2 검색 프로세스는

충돌 감지된 CT 노드를 두 개의 자식 CT 노드로 분할하고, 상기 분할된 자식 CT 노드 트리에 제약 조건을 적용하며, 시공간 비용이 가장 낮은 CT 노드를 선택하는 것을 특징으로 하는 시스템.

청구항 12

제 11 항에 있어서,

상기 시공간 비용은 시공간 거리에 비례하는 것을 특징으로 하는 시스템.

청구항 13

제 11 항에 있어서,

상기 제2 검색 프로세스는 상기 제약 조건을 적용하기 위해 충돌이 감지된 CT 노드를 식별하여 무효화하고, 상기 충돌이 감지된 CT 노드와 이웃하는 이웃 CT 노드를 가지치기하며, 상기 충돌된 CT 노드의 자식 CT 노드를 가지치기하고, 가지치기된 자식 CT 노드에서 트리를 재계획하는 것을 특징으로 하는 시스템.

청구항 14

제 13 항에 있어서,

상기 제2 검색 프로세스는 상기 이웃 노드의 가지치기된 시공간 위치를 고려하여 가지치기된 자식 CT 노드에서

새로운 경로를 계획하는 것을 특징으로 하는 시스템.

청구항 15

제 1 항 내지 제 7 항 중 어느 한 항에 따른 경로 검색 방법을 수행하는 프로그램이 저장된 컴퓨터 판독 가능한 기록 매체.

청구항 16

제 1 항 내지 제 7 항 중 어느 한 항에 따른 경로 검색 방법을 실행하기 위한 프로그램 코드를 포함하는 컴퓨터 판독 가능한 기록 매체에 저장된 프로그램.

발명의 설명

기술 분야

[0001] 본 발명은 연속하는 공간에서의 다중 에이전트 경로 탐색을 위한 시공간 충돌 기반 검색 방법, 시스템 및 프로그램에 관한 것이다.

배경 기술

[0002] 최근 자율 주행 자동차, 로봇, 군사, 게임 등 다양한 분야에서 Multi-Agent Pathfinding (MAPF, 다중 에이전트 경로 탐색)이 활발하게 활용되고 있다. MAPF는 여러 에이전트가 충돌 없이 움직이며 주어진 목표 비용 함수를 최적화하기 위한 경로를 찾는 것을 의미한다.

[0003] 종래의 MAPF 방법은 일반적으로 결합된(coupled), 비결합된(decoupled), 그리고 동적 결합된(dynamically-coupled) 방법으로 분류되었다.

[0004] 결합된 방법(예를 들어, Operator Decomposition(Standley 2010), Enhanced Partial Expansion A*(Goldenberg 등 2014))은 모든 에이전트의 고차원 상태 공간을 사용하지만 에이전트 수가 증가하면 확장 불가능하며, 완전성과 최적성을 보장하지만 실용적이지 않았다.

[0005] 비결합된 방법(예를 들어, Prioritized Planning Variants A*(Velagapudi, Sycara 및 Scerri 2010), Hierarchical Cooperative A*(Silver 2005))은 에이전트를 별도로 고려하여 검색 공간을 줄이지만 이론적 보장이 부족하였다.

[0006] 동적 결합된 방법(예를 들어, CBS(Sharon 등 2015), M*(Wagner 및 Choset 2015))은 에이전트를 필요할 때만 결합하여 전자 두 방법의 이점을 결합하며, 향상된 CBS(Enhanced CBS)와 명시적 추정 CBS(Explicit Estimation CBS)와 같은 다양한 변형이 존재한다.

[0007] 다만, 과거의 MAPF 방법들은 중간 규모의 이산 공간 문제에 효과적이었지만, 문제 규모가 커짐에 따라 한계가 있었다.

[0008] 또한, 종종 연속하는 환경을 이산화해야 하는 단점이 있었으며, 에이전트 수가 증가하면 복잡성이 기하급수적으로 증가하여 종래의 방법으로는 대응하기 어려웠다.

[0009] 결과적으로, 이러한 이유로 과거의 MAPF 알고리즘들은 완전성과 최적성을 포기하거나 특정 상황에서만 적용 가능한 알고리즘으로 한정되었다.

[0010] 그러므로, 연속 공간에서 고속으로 다중 에이전트의 경로를 탐색하는 방법의 필요성이 더욱 중요해지고 있는 상황이다.

발명의 내용

해결하려는 과제

[0011] 본 명세서는 확장 가능하고 효율적이며 확률적으로 완전함을 보장하는 이론적 보장을 가지기 위해 연속 공간에서의 다중 에이전트 경로 탐색을 위한 시공간 충돌 기반 검색 방법, 시스템 및 프로그램을 제공하는 것을 목적으로 한다.

[0012] 본 명세서는 상기 언급된 과제로 제한되지 않으며, 언급되지 않은 또 다른 과제들은 아래의 기재로부터 통상의

기술자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

- [0013] 상술한 목적을 달성하기 위한 본 발명의 일 실시 예에 따른 연속하는 공간에서의 다중 에이전트 경로 탐색을 위한 시공간 충돌 기반 경로 검색 방법은 시간이 고려된 연속하는 공간 내에서 다중 에이전트의 각각의 경로를 생성하는 제1 검색 단계 및 생성된 에이전트 각각의 경로 간에 충돌이 발생하는 경우, 충돌 기반 탐색을 통해 해당 에이전트 간의 충돌을 해결하는 제2 검색 단계를 포함할 수 있다.
- [0014] 또한, 상기 제1 검색 단계는 시간이 고려된 연속하는 공간 내 무작위 노드를 결정하는 단계, 상기 무작위 노드와 가장 가까운 노드를 선택하는 단계, 상기 가장 가까운 노드와 상기 무작위 노드 사이의 중간 노드를 추가하는 단계 및 상기 가장 가까운 노드와 중간 노드를 연결하여 상기 경로를 생성하는 단계를 더 포함할 수 있다.
- [0015] 또한, 상기 가장 가까운 노드와 상기 중간 노드 간의 시공간 거리가 기준값과 같은 경우, 상기 중간 노드와 상기 무작위 노드는 동일한 시공간 위치일 수 있다.
- [0016] 또한, 상기 제2 검색 단계는 충돌 감지된 CT(Constraint Tree) 노드를 두 개의 자식 CT 노드로 분할하는 단계, 상기 분할된 자식 CT 노드 트리에 제약 조건을 적용하는 단계, 시공간 비용이 가장 낮은 CT 노드를 선택하는 단계를 더 포함할 수 있다.
- [0017] 또한, 상기 시공간 비용은 시공간 거리에 비례할 수 있다.
- [0018] 또한, 상기 제약 조건을 적용하는 단계는 충돌이 감지된 CT 노드를 식별하여 무효화하는 단계, 상기 충돌이 감지된 CT 노드와 이웃하는 이웃 CT 노드를 가지치기하는 단계, 상기 충돌된 CT 노드의 자식 CT 노드를 가지치기하는 단계, 가지치기된 자식 CT 노드에서 트리를 재계획하는 단계를 더 포함할 수 있다.
- [0019] 또한, 상기 재계획하는 단계는 상기 이웃 노드의 가지치기된 시공간 위치를 고려하여 가지치기된 자식 CT 노드에서 새로운 경로를 계획할 수 있다.
- [0020] 한편, 연속하는 공간에서의 다중 에이전트 경로 탐색을 위한 시공간 충돌 기반 경로 검색 방법을 수행하는 시스템은 시간이 고려된 연속하는 공간 내에서 다중 에이전트의 각각의 경로를 생성하는 제1 검색 프로세스 및 생성된 에이전트 각각의 경로 간에 충돌이 발생하는 경우, 충돌 기반 탐색을 통해 해당 에이전트 간의 충돌을 해결하는 제2 검색 프로세스를 포함할 수 있다.
- [0021] 상기 제1 검색 프로세스는 시간이 고려된 연속하는 공간 내 무작위 노드를 결정하고, 상기 무작위 노드와 가장 가까운 노드를 선택하며, 상기 가장 가까운 노드와 상기 무작위 노드 사이의 중간 노드를 추가하고, 상기 가장 가까운 노드와 중간 노드를 연결하여 상기 경로를 생성할 수 있다.
- [0022] 또한, 상기 가장 가까운 노드와 상기 중간 노드 간의 시공간 거리가 기준값과 같은 경우, 상기 중간 노드와 상기 무작위 노드는 동일한 시공간 위치일 수 있다.
- [0023] 또한, 상기 제2 검색 프로세스는 충돌 감지된 CT 노드를 두 개의 자식 CT 노드로 분할하고, 상기 분할된 자식 CT 노드 트리에 제약 조건을 적용하며, 시공간 비용이 가장 낮은 CT 노드를 선택할 수 있다.
- [0024] 또한, 상기 시공간 비용은 시공간 거리에 비례할 수 있다.
- [0025] 또한, 상기 제2 검색 프로세스는 상기 제약 조건을 적용하기 위해 충돌이 감지된 CT 노드를 식별하여 무효화하고, 상기 충돌이 감지된 CT 노드와 이웃하는 이웃 CT 노드를 가지치기하며, 상기 충돌된 CT 노드의 자식 CT 노드를 가지치기하고, 가지치기된 자식 CT 노드에서 트리를 재계획할 수 있다.
- [0026] 또한, 상기 제2 검색 프로세스는 상기 이웃 노드의 가지치기된 시공간 위치를 고려하여 가지치기된 자식 CT 노드에서 새로운 경로를 계획할 수 있다.

발명의 효과

- [0027] 본 명세서에 따르면, 기존의 이산 공간 MAPF 접근 방식을 다양한 환경에 적용할 수 있도록 연속 공간으로 확장할 수 있다.
- [0028] 또한, 본 발명은 UST-RRT*알고리즘을 활용하여 에이전트가 기다리고 이전 위치로 이동할 수 있게 함으로써, 밀집된 제한된 공간에서 교착 상태를 피할 수 있는 효과가 있다.

- [0029] 또한, 본 발명은 낮은 수준에서 RRT 트리를 재사용하고 높은 수준에서 제약 조건을 직접 유지하지 않고 트리를 사용하여 충돌 검출을 간소화함으로써, 빠르게 경로 계획을 수행할 수 있다.
- [0030] 또한, 본 발명은 높은 수준의 충돌 해결 과정에서 낮은 수준의 트리 표현에 직접적인 제약 조건을 부여하여 충돌을 해결하고, 이전에 생성된 트리를 다시 활용하여 다시 경로 계획을 수행함으로써 효율성을 향상시킬 수 있다.
- [0031] 본 발명의 효과들은 이상에서 언급된 효과로 제한되지 않으며, 언급되지 않은 또 다른 효과들은 아래의 기재로부터 통상의 기술자에게 명확하게 이해될 수 있을 것이다.

도면의 간단한 설명

- [0032] 도 1은 본 발명의 일 실시예에 따른 연속 공간과 이산 공간을 나타낸 예시도이다.
- 도 2는 본 발명의 일 실시예에 따른 연속 공간에서의 다중 에이전트 경로 탐색을 위한 시공간 충돌 기반 검색 방법을 나타낸 흐름도이다.
- 도 3은 본 발명의 일 실시예에 따른 시공간을 나타낸 예시도이다.
- 도 4는 본 발명의 일 실시예에 따른 UST-RRT* 알고리즘을 나타낸 예시도이다.
- 도 5는 본 발명의 일 실시예에 따른 제1 검색 단계를 보다 구체적으로 나타낸 흐름도이다.
- 도 6은 본 발명의 일 실시예에 따른 중간 노드를 추가하는 방법을 나타낸 예시도이다.
- 도 7은 본 발명의 일 실시예에 따른 리와이어링 과정을 나타낸 예시도이다.
- 도 8은 본 발명의 일 실시예에 따른 제1 검색 단계의 동작 과정을 나타낸 예시도이다.
- 도 9는 본 발명의 일 실시예에 따른 ST-CBS(Spatio-Temporal CBS) 알고리즘을 나타낸 예시도이다.
- 도 10은 본 발명의 일 실시예에 따른 제2 검색 단계를 보다 구체적으로 나타낸 흐름도이다.
- 도 11은 본 발명의 일 실시예에 따른 에이전트들의 경로 간 충돌을 나타낸 예시도이다.
- 도 12은 본 발명의 일 실시예에 따른 제약 조건 적용 단계를 보다 구체적으로 나타낸 흐름도이다.
- 도 13은 본 발명의 일 실시예에 따른 가지치기 과정을 나타낸 예시도이다.
- 도 14는 본 발명의 일 실시예에 따른 제약 조건 적용 과정을 나타낸 예시도이다.
- 도 15는 본 발명의 일 실시예에 따른 두 개의 자식 CT 노드에 제약 조건을 적용한 것을 나타낸 예시도이다.
- 도 16은 본 발명의 일 실시예에 따른 제2 검색 단계의 동작 과정을 나타낸 예시도이다.
- 도 17은 본 발명의 일 실시예에 따른 시스템의 구성을 나타낸 블록도이다.
- 도 18은 본 발명의 일 실시예에 따른 실험 환경을 나타낸 예시도이다.
- 도 19 및 도 20은 본 발명의 일 실시예에 따른 시험 결과를 나타낸 그래프이다.

발명을 실시하기 위한 구체적인 내용

- [0033] 이하의 내용은 단지 본 발명의 원리를 예시한다. 그러므로 당업자는 비록 본 명세서에 명확히 설명되거나 도시되지 않았지만 본 발명의 원리를 구현하고 본 발명의 개념과 범위에 포함된 다양한 장치를 발명할 수 있는 것이다. 또한, 본 명세서에 열거된 모든 조건부 용어 및 실시 예들은 원칙적으로, 본 발명의 개념이 이해되도록 하기 위한 목적으로만 명백히 의도되고, 이와 같이 특별히 열거된 실시 예들 및 상태들에 제한적이지 않는 것으로 이해되어야 한다.
- [0034] 상술한 목적, 특징 및 장점은 첨부된 도면과 관련한 다음의 상세한 설명을 통하여 보다 분명해질 것이며, 그에 따라 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자가 본 발명의 기술적 사상을 용이하게 실시할 수 있을 것이다.
- [0035] 또한, 본 발명을 설명함에 있어서 본 발명과 관련된 공지 기술에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에 그 상세한 설명을 생략하기로 한다.

- [0036] 이하에는 첨부된 도면을 참조하여 본 발명의 다양한 실시 예에 대하여 상세하게 설명한다.
- [0037] 도 1은 본 발명의 일 실시예에 따른 연속 공간과 이산 공간을 나타낸 예시도이다.
- [0038] 도 1을 참조하면, 이산 공간(Discrete space, 12)은 상대적으로 작은 탐색 공간으로 구현이 용이하나 공간 내 환경의 과도한 단순하게 표현되며 다양한 형태의 에이전트 다루기가 어려우며, 에이전트의 동작에 제한이 있다는 문제가 있다.
- [0039] 이와 다르게 연속적인 공간(Continuous space, 11)은 공간 내 환경을 정확하게 표현할 수 있고 다양한 형태의 에이전트를 쉽게 다룰 수 있으며, 에이전트의 연속적인 동작이 좋지만, 구현이 어려우면, 극도로 큰 탐색 공간을 가진다는 단점이 있다.
- [0040] 이러한 연속적인 공간에서 다중 에이전트 경로 탐색(MAPF)을 하기 위한 종래의 ST-RRT*(Spatio-Temporal Rapidly-exploring Random Tree*), K-CBS(K-Corridor-Based Search), CBS-MP(Conflict-Based Search with Multi-Path extensions) 알고리즘들이 연구되었으나, ST-RRT*는 개별 경로 계획의 결합이 없어서 제한된 공간에서 MAPF 문제를 해결할 수 없는 단점이 있다.
- [0041] 또한, K-CBS는 에이전트 병합을 통해 문제를 해결할 수 있고 CBS-MP는 재샘플링을 통해 문제를 해결할 수 있지만 모두 로봇 수의 증가에 따른 확장성이 부족하다는 단점이 있다.
- [0042] 본 발명은 이전의 다중 에이전트 경로 탐색 방법의 한계를 극복하고 연속적인 공간에서 교착 상태를 회피하며 경로를 찾을 수 있는 확장성 높은 다중 에이전트 경로 탐색 알고리즘으로 고안되었다. 이와 관련하여 도 2를 참조하여 시스템(100)의 시공간 충돌 기반 에이전트 경로 검색 방법을 설명하기로 한다.
- [0043] 도 2는 본 발명의 일 실시예에 따른 연속 공간에서의 다중 에이전트 경로 탐색을 위한 시공간 충돌 기반 검색 방법을 나타낸 흐름도이다.
- [0044] 도 2를 참조하면, 연속하는 공간에서의 다중 에이전트 경로 탐색을 위한 시공간 충돌 기반 경로 검색하기 위해 시스템(100)은 시간이 고려된 연속하는 공간 내에서 다중 에이전트의 각각의 경로를 생성할 수 있다(S1000). 여기서, 시간이 고려된 연속하는 공간은 시공간 구성 공간(Spatio-Temporal configuration space)을 의미한다. 이와 관련하여 도 3을 추가로 참조하여 설명한다.
- [0045] 도 3은 본 발명의 일 실시예에 따른 시공간을 나타낸 예시도이다.
- [0046] 도 3을 참조하면, 시공간 내 제1 특정 노드(v)와 제2 특정 노드(v')는 각각(X, t)와 (X', t')에 위치할 수 있다.

수학식 1

[0048]
$$d_s(v, v') = |X - X'|$$

- [0049] 또한, 제1 특정 노드(v)와 제2 특정 노드(v') 간의 공간 거리($d_s(v, v')$)는 위의 수학식 1을 통해 산출될 수 있다.

수학식 2

[0051]
$$d_T(v, v') = |t - t'|$$

- [0052] 또한, 제1 특정 노드(v)와 제2 특정 노드(v') 간의 시간 거리($d_T(v, v')$)는 위의 수학식 2를 통해 산출될 수 있다.

수학식 3

[0054]
$$d_{ST}(v, v') = \lambda dS(v, v') + (1 - \lambda) dT(v, v')$$

[0055] 또한, 제1 특정 노드(v)와 제2 특정 노드(v') 간의 시공간 거리(또는 시간공간 거리, $d_{ST}(v, v')$)는 위의 수학식 3을 통해 산출될 수 있다. 이때, 시공간 거리는 매개변수($\ddot{\Theta}$)에 의해 결정될 수 있다.

[0056] 이하 본 발명에서는 2차원을 나타내는 X축과 Y축과 시간을 나타내는 Z축으로 구현된 연속하는 시공간을 기준으로 다중 에이전트의 각각의 경로를 생성 방법을 설명하기로 한다.

[0057] 또한, 제1 검색 단계(S1000)에서 시스템(100)은 에이전트가 공간 상태를 변경하지 않고 시간 상태만을 변경하여 진행할 수 있는 경로를 생성할 수 있다.

[0058] 또한, 제1 검색 단계(S1000)는 ST-CBS의 낮은 수준(low level) 탐색(또는 저수준 탐색)을 의미하며, 제1 검색 단계(S1000)는 도 4와 같은 UST-RRT 알고리즘이 적용될 수 있다.

[0059] 이와 관련하여 도 5를 추가로 참조하여 설명한다.

[0060] 도 5는 본 발명의 일 실시예에 따른 제1 검색 단계를 보다 구체적으로 나타낸 흐름도이다.

[0061] 도 5를 참조하면, 시스템(100)은 시간이 고려된 연속하는 공간 내 무작위 노드를 결정할 수 있다(S1100).

[0062] 구체적으로, 시스템(100)은 에이전트 마다 시공간 내 노드를 샘플링하여 시공간 내 위치한 무작위 노드를 결정할 수 있다. 이때, 시스템(100)은 도 4의 line 3과 같이 SAMPLING 함수를 통해 시공간 내 (X_{rnd} , t_{rnd})에 위치한 무작위 노드(V_{rnd})를 결정할 수 있으며, 결정된 무작위 노드(V_{rnd})는 모든 위치 상태 및 시간 상태에서 결정될 수 있다.

[0063] 만약, 에이전트 마다 목적지가 있는 경우, 해당 목적지의 시공간 위치가 무작위 노드로 결정될 수도 있다.

[0064] 다음, 시스템(100)은 무작위 노드와 가장 가까운 노드를 선택할 수 있다(S1200). 구체적으로, 시스템(100)은 무작위 노드와 인접한 경로들의 노드들 중 시공간 거리가 짧은 노드를 가장 가까운 노드로 선택할 수 있다. 이때, 시스템(100)은 도 4의 line 4와 같이 NEAREST 함수를 통해 가장 가까운 노드(V_{near})를 선택할 수 있다.

[0065] 그리고, 시스템(100)은 가장 가까운 노드와 무작위 노드 사이의 중간 노드를 추가할 수 있다(S1300). 이는 상대적으로 거리가 먼 무작위 노드를 연결하면 에이전트가 너무 빨리 이동해야 하는 긴 경로(즉, 엣지)가 생성되므로, 가장 가까운 노드와 무작위 노드 사이에 공간적 간격을 채우기 위한 중간 노드가 추가될 수 있다.

[0066] 구체적으로, 시스템(100)은 가장 가까운 노드와 무작위 노드 간 사이의 직선 상에 중간 노드를 추가할 수 있다. 이때, 중간 노드는 무작위 노드에 도달하기 전까지 여러 개가 추가될 수 있다. 이와 관련하여 도 6을 추가로 참조하여 설명한다.

[0067] 도 6은 본 발명의 일 실시예에 따른 중간 노드를 추가하는 방법을 나타낸 예시도이다.

[0068] 도 6을 참조하면, 시스템(100)은 가장 가까운 노드와 무작위 노드 간 사이의 직선 상에 중간 노드(V_{new})들을 추가하며, 도 5의 (a)와 같이 추가된 중간 노드 중 가장 끝에 위치한 중간 노드(V_s)와 가장 가까운 노드(V_{near}) 간의 시공간 거리($\|x_s - x_{near}\|$)가 기준값(d_{max}) 이하인 경우, 중간 노드(V_{new})로 추가할 수 있다.

[0069] 또한, 도 5의 (b)와 같이 추가된 중간 노드 중 가장 끝에 위치한 중간 노드(V_s)와 가장 가까운 노드(V_{near}) 간의 시공간 거리($\|x_s - x_{near}\|$)가 기준값(d_{max})과 같은 경우, 중간 노드(V_s)를 추가하지 않을 수 있다. 이때, 중간 노드(V_s)는 무작위 노드(V_{rnd})와 동등할 수 있다.

[0070] 즉, 가장 가까운 노드와 무작위 노드 간의 시공간 거리가 기준값 미만인 경우 중간 노드와 무작위 노드는 동일

한 시공간 위치일 수 있다.

- [0071] 이때, 시스템(100)은 도 3의 line 5와 같이 STEER 함수를 통해 가장 가까운 노드(V_{near})와 무작위 노드(V_{rnd}) 사이의 중간 노드(V_{new})를 선택할 수 있다.
- [0072] 다음, 시스템(100)은 가장 가까운 노드와 중간 노드를 연결하여 경로를 생성할 수 있다(S1400).
- [0073] 구체적으로, 시스템(100)은 기존의 트리 내 가장 가까운 노드를 선택하여 시공간 거리가 최소가 되도록 중간 노드를 연결할 수 있다.
- [0074] 다음, 시스템(100)은 리와이어링(REWIRING)을 통해 에이전트에 더 짧은 경로를 탐색할 수 있다. 이때, 시공간 거리가 최소가 되는지 판단하여 리와이어링할 수 있다.
- [0075] 이와 관련하여 도 7을 추가로 참조하면, 시스템(100)은 연결된 중간 노드(V_{new})를 리와이어링하여 트리를 최적화할 수 있다. 중간 노드(V_{new})의 자식이 될 수 있는 후보 노드들(v^+, v^-) 중 중간 노드(V_{new})에 연결되는 것이 현재 거리에 비해 시공간 거리가 줄어드는지를 평가하여 리와이어링을 결정할 수 있다.
- [0076] 이때, 시스템(100)은 도 4의 line 8-11와 같이 NEIGHBOR 함수, CONNECTPARENT 함수, REWIRE 함수를 통해 트리 내 가장 가까운 노드를 선택하여 중간 노드를 연결하고 리와이어링할 수 있다.
- [0077] 그리고, 시스템(100)은 중간 노드가 목표 지점과 동일한 시공간 위치 또는 매우 가까운 경우, 해당 트리를 에이전트의 경로로 생성할 수 있다(S1500).
- [0078] 이때, 시스템(100)은 도 4의 line 15-18와 같이 GENERATEPATH 함수를 통해 경로를 생성할 수 있다.
- [0079] 만약, 아닌 경우에는 시스템(100)은 상술한 (S1100) 내지 (S1400)단계를 재 수행할 수 있다.
- [0080] 이어서, 도 8을 참조하여 상술한 제1 검색 단계(S1000)의 전반적인 동작 과정을 설명한다.
- [0081] 도 8은 본 발명의 일 실시예에 따른 제1 검색 단계의 동작 과정을 나타낸 예시도이다.
- [0082] 도 8은 시공간 내 위치한 장애물(Obstacles, 82)을 피해 제1 에이전트(81)의 제1 검색 과정을 나타낸 것이다.
- [0083] 도 8을 참조하면, 제1 에이전트(81)의 경로 생성을 위해 시스템(100)은 시공간 내 무작위 노드(83)를 결정하고, 무작위 노드(83)과 가장 가까운 노드(84)를 선택할 수 있다.
- [0084] 그리고, 시스템(100)은 무작위 노드(83)와 가장 가까운 노드(84) 사이의 중간 노드(85)를 추가하고, 중간 노드(85)와 가장 가까운 노드(84)를 연결(86)하여 제1 에이전트(81)의 경로를 생성할 수 있다.
- [0086] 다시 도 2를 참조하면, 시스템(100)은 생성된 에이전트 각각의 경로 간에 충돌(Conflict)이 발생하는 경우, 충돌 기반 탐색을 통해 해당 에이전트 간의 충돌을 해결할 수 있다(S2000).
- [0087] 그리고, 제2 검색 단계(S2000)에서 에이전트 경로 간의 충돌이 해결되는 경우, 시스템(100)은 다시 제1 검색 단계(S1000)를 수행하여 다중 에이전트의 경로를 생성할 수 있다. 이때, 다중 에이전트 경로 간에 충돌이 추가로 발생하는 경우, 시스템(100)은 제2 검색 단계(S2000)를 재수행한 후, 제1 검색 단계(S1000)를 수행할 수 있다.
- [0088] 또한, 제2 검색 단계(S2000)는 ST-CBS의 높은 수준(high level) 탐색(또는 고수준 탐색)을 의미하며, 제2 검색 단계(S2000)는 도 9와 같은 ST-CBS 알고리즘이 적용될 수 있다.
- [0089] 이와 관련하여 도 10을 추가로 참조하여 설명한다.
- [0090] 도 10은 본 발명의 일 실시예에 따른 제2 검색 단계를 보다 구체적으로 나타낸 흐름도이다.
- [0091] 도 10을 참조하면, 시스템(100)은 제1 검색 단계(S1000)에서 생성된 에이전트들의 경로들이 서로 충돌하는 경우, 충돌 감지된 CT 노드를 두 개의 자식 CT 노드로 분할할 수 있다(S2100). 여기서, CT 노드는 생성된 에이전트의 경로 상의 충돌이 발생한 지점을 의미할 수 있다. 이와 관련하여 도 11을 추가로 참조하여 설명한다.
- [0092] 도 11은 본 발명의 일 실시예에 따른 에이전트들의 경로 간 충돌을 나타낸 예시도이다.
- [0093] 도 11을 참조하면, 생성된 에이전트 i의 경로(111)와 에이전트 j의 경로(112) 간에 충돌이 일어난 경우, 해당

충돌 위치의 에이전트 i의 경로(111) 상의 노드(113)와 에이전트 j의 경로(112) 상의 노드(114)는 각각 CT 노드일 수 있다.

- [0094] 다음, 시스템(100)은 분할된 자식 CT 노드 트리에 제약 조건을 적용할 수 있다(S2200). 여기서, 제약 조건은 에이전트들 간의 움직임을 제한하거나 수정하는 규칙 또는 제약사항을 의미한다. 이하, 제약 조건 적용 단계(S2200)와 관련하여 도 12를 참조하여 구체적으로 설명한다.
- [0095] 도 12는 본 발명의 일 실시예에 따른 제약 조건 적용 단계를 보다 구체적으로 나타낸 흐름도이다.
- [0096] 도 12를 참조하면, 시스템(100)은 충돌이 감지된 CT 노드를 식별하여 무효화할 수 있다(S2210). 여기서, 무효화는 해당 CT 노드를 무시하거나 제거하여 알고리즘이 해당 충돌 지점을 무시하고 다른 경로를 계속 찾도록 하는 것을 의미한다.
- [0097] 즉, 충돌이 감지된 경우 해당 충돌 지점을 무효화하면, 해당 지점에서 발생한 충돌을 무시하고 새로운 경로를 찾음으로써, 에이전트의 경로 간의 충돌을 피하고 안전한 경로를 계획할 수 있다.
- [0098] 다음, 시스템(100)은 충돌이 감지된 CT 노드와 이웃하는 이웃 CT 노드를 가지치기(prune)할 수 있다 (S2220). 여기서, 가지치기는 트리(또는 그래프)에서 불필요한 가지(분기)를 제거하거나 간소화하는 과정으로, 계산 비용을 줄이거나 최적의 해를 찾는 과정 중 하나이다.
- [0099] 이때, 시스템(100)은 이웃 CT 노드와 이웃 CT 노드의 자식 CT 노드를 가지치기할 수 있으며, 이는 충돌이 감지된 CT 노드와 인접한 경우 추가 충돌 가능성이 높기 때문이다.
- [0100] 예를 들어, 도 13과 같이 시스템(100)은 충돌이 감지된 CT 노드(V_c)로부터 소정 영역(S) 내에 위치한 이웃 CT 노드와 이웃 CT 노드의 자식 노드들(도 12의 보라색 부분)을 가지치기(prune)할 수 있다.
- [0101] 그리고, 시스템(100)은 충돌된 CT 노드의 자식 CT 노드를 가지치기할 수 있다(S2230).
- [0102] 또한, 시스템(100)은 가지치기된 지점을 기억하여 트리 재계획에 이용할 수 있다.
- [0103] 이때, 시스템(100)은 도 9의 line 12-14와 같이 PRUNENEIGHBOR 함수, PRUNECHILDREN 함수를 통해 이웃 CT 노드와 자식 CT 노드를 가지치기하며, MARKCONFLICTVERTEX 함수를 통해 제약점(constraint vertex)을 마크할 수 있다.
- [0104] 다음, 시스템(100)은 가지치기된 자식 CT 노드에서 트리를 재계획할 수 있다(S2230). 구체적으로, 시스템(100)은 이웃 노드의 가지치기된 시공간 위치나 를 고려하여 가지치기된 자식 CT 노드에서 새로운 경로를 계획할 수 있다.
- [0105] 즉, 시스템(100)은 가지치기된 이웃 노드의 위치 및 시간을 기억하고, 이를 피해 새로운 경로를 계획할 수 있다.
- [0106] 이어서, 도 14를 참조하여 상술한 제약 조건 적용 단계(S2200)의 전반적인 동작 과정을 설명한다.
- [0107] 도 14는 본 발명의 일 실시예에 따른 제약 조건 적용 과정을 나타낸 예시도이다.
- [0108] 도 14를 참조하면, 시스템(100)은 충돌이 감지된 CT 노드를 식별하여 무효화(1301)하고, 충돌이 감지된 CT 노드와 이웃하는 이웃 CT 노드를 가지치기(1302)할 수 있다.
- [0109] 그 후, 시스템(100)은 충돌된 CT 노드의 자식 CT 노드를 가지치기(1303)하고, 가지치기된 자식 CT 노드에서 트리를 재계획(1304)할 수 있다. 이러한 제약 조건을 적용하는 과정은 자식 CT 노드 마다 각각 수행될 수 있다.
- [0110] 이와 관련하여 도 15를 추가로 참조하여 설명한다.
- [0111] 도 15는 본 발명의 일 실시예에 따른 두 개의 자식 CT 노드에 제약 조건을 적용한 것을 나타낸 예시도이다.
- [0112] 도 15를 참조하면, 충돌한 에이전트 j와 에이전트 i가 충돌하는 경우, 시스템(100)은 초기 솔루션(1501) 내 초기 에이전트 j의 트리(1502)와 초기 에이전트 i의 트리(1503)에 제약 조건을 각각 적용할 수 있다.
- [0113] 그러면, 시스템(100)은 재계획된 솔루션(1504) 내 에이전트 j의 트리(1505)와 재계획된 솔루션(1506) 내 에이전트 i의 트리(1507)를 생성할 수 있다.
- [0114] 다시, 도 10을 참조하면, 시스템(100)은 시공간 비용이 가장 낮은 CT 노드를 선택할 수 있다(S2300). 여기서,

시공간 비용은 시공간 거리에 비례할 수 있다.

- [0115] 예를 들어, 시공간 거리가 최소가 되는 CT 노드의 시공간 비용이 가장 낮을 수 있다.
- [0116] 이어서, 도 16을 참조하여 상술한 제2 검색 단계(S2000)의 전반적인 동작 과정을 설명한다.
- [0117] 도 16은 본 발명의 일 실시예에 따른 제2 검색 단계의 동작 과정을 나타낸 예시도이다.
- [0118] 도 16을 참조하면, 시스템(100)은 에이전트 경로 간에 충돌이 없는 해결책을 찾을 때까지 제2 검색 단계를 1회 내지 4회를 반복할 수 있다.
- [0119] 구체적으로, 시스템(100)은 트리(1501)의 CT 노드에서 충돌 감지하고, 트리(1601)의 CT 노드를 두 개의 하위 CT 노드로 분할하여 각각의 트리(1602, 1603)로 형성할 수 있다.
- [0120] 그리고, 시스템(100)은 하위 CT 노드들의 트리(1602, 1603)에 제약 조건을 부과하여 시공간 비용(Cost)을 33, 32로 각각 산출하고, 하위 CT 노드들의 트리(1602, 1603) 중 시공간 비용(Cost)이 최소인 CT 노드의 트리(1603)를 선택할 수 있다. 이때, 트리(1603)에는 에이전트 경로 간에 충돌이 여전히 존재할 수 있다.
- [0121] 그러면, 시스템(100)은 다시 트리(1603)의 CT 노드를 두 개의 하위 CT 노드로 분할하여 두 개의 트리(1606, 1607)로 형성한 후, 분할된 하위 CT 노드들의 트리(1606, 1607)에 제약 조건을 부과하여 시공간 비용(Cost)을 산출하여 시공간 비용이 최소인 CT 노드의 트리(1602)를 선택할 수 있다. 이때, 트리(1602)에는 에이전트 경로 간에 충돌이 여전히 존재할 수 있다.
- [0122] 또한, 다시 시스템(100)은 트리(1602)의 CT 노드를 두 개의 하위 CT 노드로 분할하여 두 개의 트리(1604, 1605)로 형성한 후, 분할된 하위 CT 노드들의 트리(1604, 1605)에 제약 조건을 부과하여 시공간 비용(Cost)을 산출할 수 있다.
- [0123] 그리고, 시스템(100)은 생성된 트리들 중 시공간 비용이 최소인 CT 노드의 트리(1606)를 선택하고, 트리(1606)에 에이전트 경로 간에 충돌이 없는 경우에 제2 검색 단계(S2000)를 중단하고, 다시 제1 검색 단계(S1000)를 수행할 수 있다.
- [0124] 즉, 상술한 본 발명은 시공간 내에서 다중 에이전트들의 제1 검색 단계(S1000)를 수행하고, 제1 검색 단계(S1000)에서 충돌이 발생하는 경우에 제2 검색 단계(S2000)를 통해 다중 에이전트 간의 충돌을 해결할 수 있다.
- [0125] 그리고, 다시 제1 검색 단계(S1000)를 수행하여 시공간 내에서 다중 에이전트 간의 충돌 없는 경로를 생성할 수 있다. 이때, 시스템(100)은 충돌 여부에 따라 제1 검색 단계(S1000) 및 제2 검색 단계(S2000)를 여러 번 반복할 수 있다.
- [0126] 한편, 시스템(100)은 제1 검색 단계(S1000)에서 충돌이 일어나지 않는 경우(예를 들어, 에이전트가 1개인 경우)에는 상술한 제2 검색 단계(S2000)를 수행하지 않을 수 있다.
- [0127] 또한, 본 발명은 시간을 고려한 연속하는 2차원 공간에서의 다중 에이전트 경로 탐색 방법에 대해 설명하였으나, 이에 한정되지 않고 시간을 고려한 연속하는 3차원 공간에서도 적용 가능하다.
- [0128] 한편, 상술한 본 발명의 방법은 두 레벨의 알고리즘인 Spatio-Temporal Conflict-based Search(ST-CBS)으로, 낮은 수준(제1 검색 단계)에서는 상호 에이전트 간 충돌을 고려하지 않고 각 에이전트를 위한 공간-시간 상태 공간에서 경로를 생성하기 위해 Unidirectional Spatio-Temporal RRT (UST-RRT)를 수행할 수 있다.
- [0129] 그리고, 높은 수준(제2 검색 단계)에서는 에이전트 경로에서 발견된 충돌을 해결하기 위해 제약 트리(Constraints Tree)에서 최적 우선 탐색을 수행할 수 있다.
- [0130] 또한, 본 발명은 높은 수준의 충돌 해결 과정에서 낮은 수준의 트리 표현에 직접적인 제약 조건을 부여하여 충돌을 해결하고, 이전에 생성된 트리를 다시 활용하여 다시 경로 계획을 수행함으로써 효율성을 향상시킬 수 있다.
- [0131] 이를 통해 에이전트 간의 충돌을 방지하고 에이전트들의 경로를 효과적으로 재 계획하여 교착 상태를 최소화할 수 있다.
- [0133] 이어서, 도 17을 참조하여 연속 공간에서의 다중 에이전트 경로 탐색을 위한 시공간 충돌 기반 검색을 수행하는 시스템(100)의 구성에 대해 설명한다.

- [0134] 도 17은 본 발명의 일 실시예에 따른 시스템의 구성을 나타낸 블록도이다.
- [0135] 도 17을 참조하면, 시스템(100)은 제1 검색 프로세스(110) 및 제2 검색 프로세스(120)를 일부 또는 전부로 구성될 수 있다.
- [0136] 제1 검색 프로세스(110)는 시간이 고려된 연속하는 공간 내에서 다중 에이전트의 각각의 경로를 생성할 수 있다.
- [0137] 구체적으로, 제1 검색 프로세스(110)는 시간이 고려된 연속하는 공간 내 무작위 노드를 결정하고, 무작위 노드와 가장 가까운 노드를 선택하며, 가장 가까운 노드와 무작위 노드 사이의 중간 노드를 추가하고, 가장 가까운 노드와 중간 노드를 연결하여 경로를 생성할 수 있다.
- [0138] 즉, 제1 검색 프로세스(110)는 상술한 제1 검색 단계(S1000)를 수행할 수 있다.
- [0139] 제2 검색 프로세스(120)는 생성된 에이전트 각각의 경로 간에 충돌이 발생하는 경우, 충돌 기반 탐색을 통해 해당 에이전트 간의 충돌을 해결할 수 있다.
- [0140] 구체적으로, 제2 검색 프로세스(120)는 충돌 감지된 CT 노드를 두 개의 자식 CT 노드로 분할하고, 분할된 자식 CT 노드 트리에 제약 조건을 적용하며, 시공간 비용이 가장 낮은 CT 노드를 선택할 수 있다.
- [0141] 또한, 제2 검색 프로세스(120)는 제약 조건을 적용하기 위해 충돌이 감지된 CT 노드를 식별하여 무효화하고, 충돌이 감지된 CT 노드와 이웃하는 이웃 CT 노드를 가지치기하며, 충돌된 CT 노드의 자식 CT 노드를 가지치기하고, 가지치기된 자식 CT 노드에서 트리를 재계획할 수 있다.
- [0142] 즉, 제2 검색 프로세스(120)는 상술한 제2 검색 단계(S2000)를 수행할 수 있다.
- [0143] 한편, 상술한 본 발명은 경로 검색 방법을 수행하는 프로그램이 저장된 컴퓨터 판독 가능한 기록 매체로 구현될 수 있으며, 경로 검색 방법을 실행하기 위한 프로그램 코드를 포함하는 컴퓨터 판독 가능한 기록 매체에 저장된 프로그램으로 구현될 수도 있다.
- [0144] 이어서, 도 18 내지 도 20을 참조하여 상술한 연속 공간에서의 다중 에이전트 경로 탐색을 위한 시공간 충돌 기반 검색 방법에 대한 실험에 대해 설명한다.

[0146] 실험

- [0147] 상술한 본 발명의 알고리즘(ST-CBS)은 어려운 환경(예를 들어, 좁은 환경, 제한된 환경, 혼잡한 환경)에서도 교착 상태가 없는 경로를 생성하는 데 이점이 있으며, 경쟁 상대군과 비교하여 상대적으로 큰 문제 인스턴스를 다룰 수 있으면서 확률적 완결성을 보장할 수 있다.
- [0148] 이러한 본 발명의 장점은 아래의 실험 결과를 통해 증명될 수 있다. 또한, 본 발명의 알고리즘을 사용하여 로봇에 직접 동적 시뮬레이션(Dynamic Simulation)을 통해 솔루션을 적용할 수 있음을 보여준다.

[0150] 알고리즘 테스트

- [0151] 본 알고리즘 테스트에서는 ST-CBS의 성능을 다양한 측정 항목으로 보여줄 수 있다. 여기서, 주요 측정 항목에는 에이전트 경로의 시간 비용 합(SOC), 모든 목표 지점에 도달하는 소요 시간(makespan) 등이 포함될 수 있다.
- [0152] 이러한 측정 항목은 충돌이 더 많이 예상되는 밀집된 환경에서 이동 거리의 합보다 더 중요할 수 있다.
- [0153] 본 실험에서 성공률은 5분 이내에 해결된 인스턴스의 비율로 정의되며, 성공한 경우에는 평균 계산 시간을 계산한다.
- [0154] 또한, 모든 측정 항목의 통계는 50번의 무작위 반복을 통해 얻어졌으며, 그렇지 않은 경우는 따로 언급하지 않는다.
- [0155] 큰 인스턴스는 충분한 성공 실행을 얻는 데 어려움을 겪을 수 있으며, 이러한 경우에는 계산 시간은 데이터 포인트가 적은 것으로 계산될 수 있다.
- [0156] 본 알고리즘 테스트의 시험 환경에는 공간적으로 넓고 개방된 환경으로 장애물이 적고 자유롭게 움직일 수 있는

열린(Open) 공간(또는 열린 환경), 장애물이 많고 복잡한 환경으로, 다양한 장애물로 인해 이동이 어려울 수 있는 혼잡한(Cluttered) 공간(또는 혼잡한 환경), 공간이 협소하고 좁은 환경으로, 이동이 제한되고 특정 경로로의 이동이 쉽지 않을 수 있는 좁은(Narrow) 공간(또는 좁은 환경) 및 이동이 매우 제한되고 국한된 환경으로, 보다 협소하고 갇혀 있는 제한된(Confined) 공간(또는 제한된 환경)이 포함될 수 있다.

- [0157] 또한, 열린 공간은 33X33의 장애물이 없는 지역으로, 모든 에이전트의 반경이 0.5일 수 있다.
- [0158] 그리고, 나머지 공간은 도 18과 같이 좁은 공간(1801), 제한된 공간(1802) 혼잡한 공간(1803)일 수 있다.
- [0159] 여기서, 혼잡한 공간(1803)은 열린 공간과 같은 크기이지만 장애물로 가득 차 있을 수 있다.
- [0160] 또한, 좁은 공간(1801) 및 제한된 공간(1802)은 좁은 복도가 있는 작은 환경으로, 한 번에 한 명의 에이전트만 통과할 수 있다.
- [0161] 또한, 좁은 공간(1801)의 경우 시간적 차원을 고려하지 않을 경우 에이전트가 교착 상태에 빠지도록 고정시킨 상태이다.
- [0162] 한편, 열린 공간 및 혼잡한 공간(1803)에서는 최대 30까지의 에이전트 수를 늘렸는데, 기존 방법으로 처리하기 어려운 경우가 많았다.
- [0163] 본 테스트에서 파라미터 설정은 $\epsilon = 0.5$, $\gamma = 2$, $d_{max} = 10$, $maxIter = 1000$ 일 수 있다. 와 d_{max} 의 최적 값은 환경에 따라 다를 수 있지만, 시험한 설정은 모든 네 가지 다른 환경에서 일관된 성능을 보여줄 수 있다.

[0165] 비교 결과

- [0166] 본 실험에서는 상술한 본 발명의 알고리즘(ST-CBS)과 두가지 최신 기술인 Continuous-space MAPF 알고리즘인 K-CBS와 우선순위 기반 계획을 사용하는 ST-RRT를 서로 비교한다.
- [0167] 한편, CBS-MP는 에이전트 수와 함께 확장되지 않는 것으로 보이며 SSSP는 솔루션을 병렬화하기 위한 추가 후처리가 필요하다.
- [0168] 이하, 도 19 및 도 20을 참조하여 실험에 통한 본 알고리즘과 타 알고리즘 간 비교 결과에 대해 설명한다.
- [0169] 도 19 및 도 20은 본 발명의 일 실시예에 따른 시험 결과를 나타낸 그래프이다.
- [0170] 도 19의 (a)는 열린 환경에 대해 측정한 메트릭(metric)을 나타낸 것이고, 도 19의 (b)는 혼잡한 환경에 대해 측정한 메트릭을 나타낸 것이다.
- [0171] 도 20은 좁은 환경에 대해 측정한 메트릭을 나타낸 것이다.
- [0172] 우선 도 19의 (a)를 참조하면, 본 발명의 알고리즘(ST-CBS)은 모든 테스트 인스턴스를 성공적으로 해결하였으나 K-CBS의 성공률은 $k > 15$ 및 ST-RRT의 성공률은 $k > 10$ 에서 상당히 낮으며, 이는 네 번째 그래프에서 나타난다.
- [0173] 본 발명의 알고리즘은 모든 에이전트 수에 대해 시간 비용 함(SOC) 및 모든 목표 지점에 도달하는 소요 시간(makespan)에서 가장 낮은 값으로 나타난다.
- [0174] 또한, 일반적으로 본 발명의 알고리즘보다 ST-RRT이 더 빠르게 계산되지만 K-CBS는 비교할 수 없을 만큼 느리게 계산되었다.
- [0175] 하지만, 에이전트(k)가 10개를 초과하는 모든 메트릭에서 본 발명의 결과가 그러나 우리는 $k = 10$ 을 초과하는 모든 메트릭을 비교하는 데 주의해야 합니다. 비교된 방법의 결과는 매우 적은 샘플에서 계산되었기 때문입니다.
- [0176] 예를 들어, 에이전트(k)가 30 인 경우 ST-RRT는 50회 시도 중 1회만 성공했으나, 본 발명의 알고리즘은 100%의 성공률을 보였다. 이때, 본 발명의 방법은 더 많은 에이전트($k = 30$)와 함께 계산이 느려졌다.
- [0177] 한편, 충돌은 $k = 5$ 에서 20까지는 드물게 발생하거나 중간 정도 발생하였고, 이는 열린 환경에 장애물이 없기 때문이다.
- [0178] 따라서, 본 발명의 알고리즘은 충돌 해결보다는 주로 낮은 수준에서 시간을 소비하였다.
- [0179] 특히, $k = 25$ 에서는 대부분의 시간이 충돌 해결에 소요되므로 하위 레벨에서 사용되는 시간이 우세하였고, $k =$

30에서는 에이전트 밀도가 증가함에 따라 평균 경로 길이가 감소하므로 충돌 수가 감소(평균 434에서 242로)할 수 있다.

[0180] 따라서, 본 발명의 알고리즘은 $k = 25$ 보다 $k = 30$ 에서 충돌 해결에 소요되는 시간이 감소하여 총 계산 시간이 더 빨라졌다.

[0181] 다음, 도 19의 (b)를 참조하면, 열린 환경과 유사하게, 비교된 방법의 성공률은 $k > 10$ 일 때 상당히 낮아졌다.

[0182] 또한, ST-RRT는 체계적인 검색의 부재로 인해 우선 순위 기반 계획이 연쇄 충돌을 발생시키기 때문에 $k = 5$ 에서조차 100% 성공을 달성하지 못했으나, 때로는 운이 좋아서 많은 에이전트 수에서 몇번의 성공을 달성하였다.

[0183] 반면에 K-CBS는 $k > 15$ 에서 대해 전혀 성공하지 못했다.

[0184] 그러나, 본 발명의 알고리즘은 시간 비용 합 및 모든 목표 지점에 도달하는 소요 시간이 상당히 낮게 나왔다.

[0185] 다른 비교 대상인 알고리즘의 낮은 성공률로 인해 신뢰할 수 있는 통계 데이터는 $k = 5$ 일 때 얻을 수 있었으며, 이 경우 모두 85% 이상의 성공을 달성하였다.

[0186] 또한, $k = 5$ 인 경우, 본 발명의 알고리즘 시간 비용 합은 57.9이며 K-CBS와 ST-RRT의 시간 비용 합은 각각 473.5와 403.4로 나타났다. 이때, 본 알고리즘의 시간 비용 합이 최대 87.7%까지 감소하여 상당한 향상을 보였다.

[0187] 마찬가지로, 모든 목표 지점에 도달하는 소요 시간은 본 발명의 알고리즘, K-CBS 및 ST-RRT에서 각각 18.5, 141.3 및 190.9으로 나타났다. 이때, 본 알고리즘의 모든 목표 지점에 도달하는 소요 시간은 최대 90.3%로 감소하였다.

[0188] 이어서, 도 20을 참조하면, 본 발명의 알고리즘과 K-CBS는 모든 인스턴스를 성공적으로 실행하지만, ST-RRT는 5개의 인스턴스만 완료할 수 있었다.

[0189] 본 발명의 알고리즘은 모든 메트릭을 기준으로 모두 다른 알고리즘을 능가하였다.

[0190] 구체적으로, K-CBS 및 ST-RRT의 계산 시간은 본 발명의 알고리즘 보다 각각 10배 및 172배 더 느리게 나타났다.

[0191] 한편, 제한된 환경에서는 본 발명의 알고리즘만이 인스턴스를 해결할 수 있었으며, 다른 알고리즘(K-CBS 및 ST-RRT)은 모두 완료하지 못하였다.

[0192] 본 발명의 알고리즘은 교착상태(deadlock)가 발생하기 쉬운 제한된 환경과 혼잡한 환경에서 강점을 보였다.

[0193] 이는 모든 환경에서 본 발명의 알고리즘의 짧은 계산 시간은 본 발명의 알고리즘이 확장 가능하다는 것을 보여준다.

[0194] 한편, 본 실험에서 설정한 $\ddot{e} = 0.5$ 는 경로를 따라 이동하는 데 소요되는 시간(travel time)과 거리를 균형 있게 고려한 값이다. $\ddot{e}$ 을 0, 0.25, 0.5, 0.75, 1로 테스트한 결과, 이 중에서 $\ddot{e} = 0.5$ 에서 거의 최상의 결과를 보였다.

[0196] 동적 시뮬레이션(Dynamic Simulation)

[0197] 상술한 본 발명의 방법을 실제 세계 문제에 적용 가능한지 확인하기 위해 고사심(Isaac Sim, NVIDIA 2023)을 사용한 로봇 시뮬레이션 환경을 구축하였고 높은 정밀도의 물리 엔진이 함께 사용되었다.

[0198] 또한, 제한된 환경에서 충돌이 없는 해결책 10개를 선택했으며, 시뮬레이션에서는 모든 해결책이 충돌 없이 성공적으로 실행되었음을 확인하였다.

[0199] 또한, 알고리즘에서 나온 경로는 로봇의 동역학 제약을 고려하지 않기 때문에, 실제 로봇으로 실행할 경우 약간 다른 결과가 나타났다.

[0200] 그러나 성공률이 100%인 것은 본 발명의 방법이 실제 세계 문제에 사용될 수 있는 타당성을 보여주는 중요한 결과이다.

[0202]

결론

[0203]

본 발명은 연속 공간에서의 MAPF 문제에 대해 다루었으며, 본 발명은 확장 가능하고 효율적인 이론적으로 확률적 완전성을 보장하는 이중 알고리즘을 제안했다.

[0204]

상술한 실험을 통해 본 발명의 알고리즘의 강점(교착상태 방지, 확장성)에 대한 주장이 실험적으로 지지되었다.

[0205]

또한, 본 알고리즘의 솔루션을 로봇을 이용한 동적 시뮬레이션으로 실행했으며, 그 결과 본 발명의 알고리즘이 혼잡한 환경에서의 다중 로봇 내비게이션과 같은 실용적 문제를 해결하는 능력이 있다는 것을 보여주었다.

[0206]

나아가, 여기에 설명되는 다양한 실시예는 예를 들어, 소프트웨어, 하드웨어 또는 이들의 조합된 것을 이용하여 컴퓨터 또는 이와 유사한 장치로 읽을 수 있는 기록매체 내에서 구현될 수 있다.

[0207]

하드웨어적인 구현에 의하면, 여기에 설명되는 실시예는 ASICs (application specific integrated circuits), DSPs (digital signal processors), DSPDs (digital signal processing devices), PLDs (programmable logic devices), FPGAs (field programmable gate arrays, 프로세서(processors), 제어기(controllers), 마이크로 컨트롤러(micro-controllers), 마이크로 프로세서(microprocessors), 기타 기능 수행을 위한 전기적인 유닛 중 적어도 하나를 이용하여 구현될 수 있다. 일부의 경우에 본 명세서에서 설명되는 실시예들이 제어 모듈 자체로 구현될 수 있다.

[0208]

소프트웨어적인 구현에 의하면, 본 명세서에서 설명되는 절차 및 기능과 같은 실시예들은 별도의 소프트웨어 모듈들로 구현될 수 있다. 상기 소프트웨어 모듈들 각각은 본 명세서에서 설명되는 하나 이상의 기능 및 작동을 수행할 수 있다. 적절한 프로그램 언어로 적혀진 소프트웨어 어플리케이션으로 소프트웨어 코드가 구현될 수 있다. 상기 소프트웨어 코드는 메모리 모듈에 저장되고, 제어모듈에 의해 실행될 수 있다.

[0209]

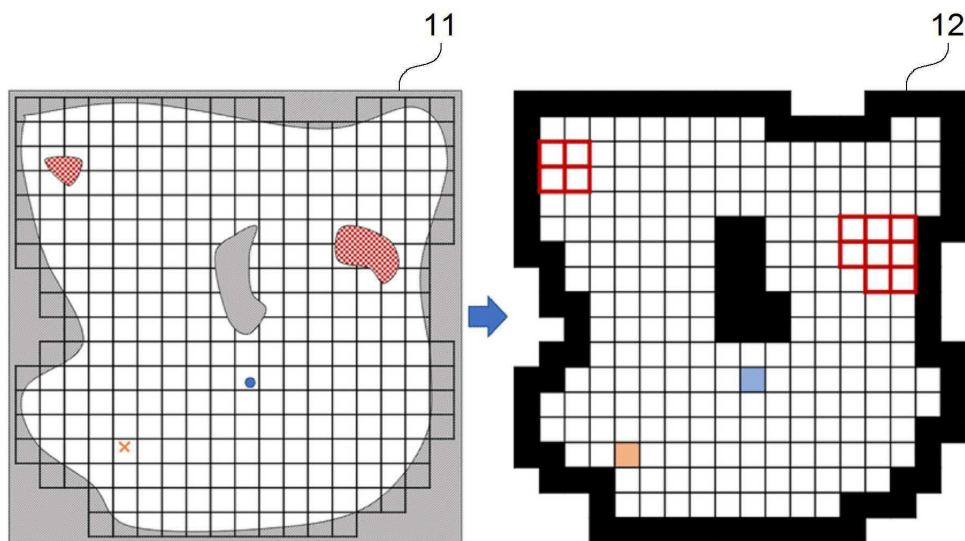
이상의 설명은 본 발명의 기술 사상을 예시적으로 설명한 것에 불과한 것으로서, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자라면 본 발명의 본질적인 특성에서 벗어나지 않는 범위 내에서 다양한 수정, 변경 및 치환이 가능할 것이다.

[0210]

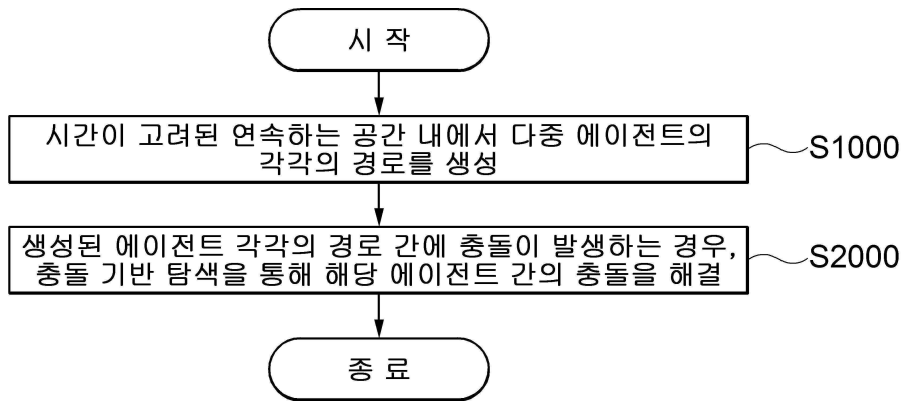
따라서, 본 발명에 개시된 실시 예 및 첨부된 도면들은 본 발명의 기술 사상을 한정하기 위한 것이 아니라 설명하기 위한 것이고, 이러한 실시 예 및 첨부된 도면에 의하여 본 발명의 기술 사상의 범위가 한정되는 것은 아니다. 본 발명의 보호 범위는 아래의 청구 범위에 의하여 해석되어야 하며, 그와 동등한 범위 내에 있는 모든 기술 사상은 본 발명의 권리 범위에 포함되는 것으로 해석되어야 할 것이다.

도면

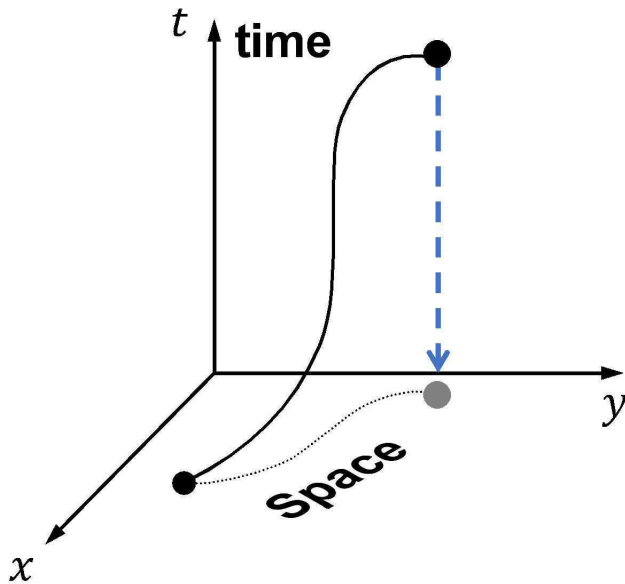
도면1



도면2



도면3



도면4

Algorithm 1: UST-RRT*

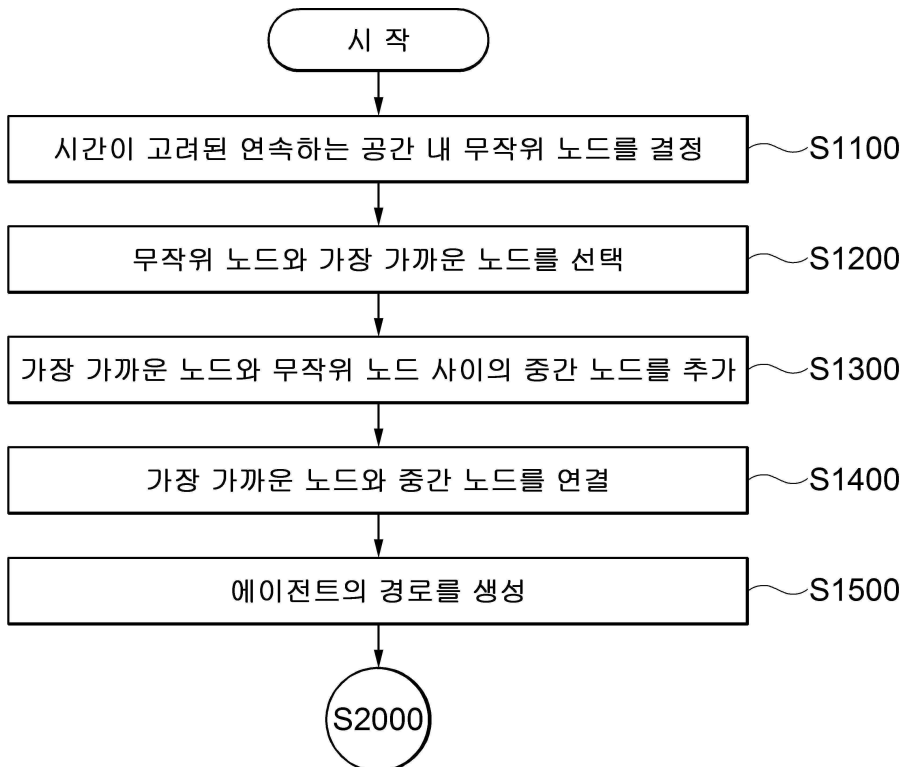
Input: $\mathcal{W}_{ST}^f, \mathcal{O}, G(\mathcal{V}, \mathcal{E}), v_s, v_g, \epsilon, d_{\max}, \maxIter$
Output: $\mathcal{P}(v_s, v_g), G$

```

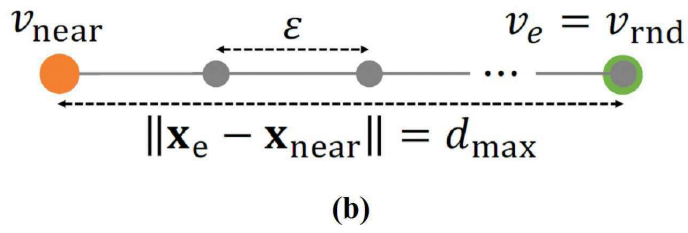
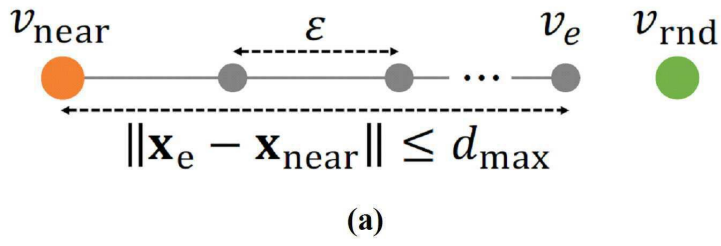
1:  $t_u \leftarrow 1, \mathcal{V} \leftarrow \{v_s\}, \mathcal{E} \leftarrow \emptyset$ 
2: for  $i = 1$  to  $\maxIter$  do
3:    $v_{\text{rnd}} \leftarrow \text{SAMPLING}(\mathcal{W}_{ST}^f, t_u)$ 
4:    $v_{\text{near}} \leftarrow \text{NEAREST}(G, v_{\text{rnd}})$ 
5:    $\mathcal{V}_{\text{new}} \leftarrow \text{STEER}(v_{\text{near}}, v_{\text{rnd}}, d_{\max}, \epsilon)$ 
6:   for  $v_{\text{new}}$  to  $V_{\text{new}}$  do
7:      $\mathcal{V} \leftarrow \mathcal{V} \cup v_{\text{new}}$ 
8:      $\mathcal{V}^- \leftarrow \text{NEIGHBOR}(G, v_{\text{new}}, \epsilon, -1)$ 
9:      $\text{CONNECTPARENT}(v_{\text{new}}, \mathcal{V}^-, \mathcal{O})$ 
10:     $\mathcal{V}^+ \leftarrow \text{NEIGHBOR}(G, v_{\text{new}}, \epsilon, +1)$ 
11:     $\text{REWIRE}(v_{\text{new}}, \mathcal{V}^+, \mathcal{E}, \mathcal{O})$ 
12:    if  $t_{\text{new}} > t_{\max}$  then
13:       $t_u = t_{\text{new}} + 1$ 
14:    end if
15:    if  $v_{\text{new}} = v_g$  then
16:       $\mathcal{P}(v_s, v_g) \leftarrow \text{GENERATEPATH}(G, v_s, v_g)$ 
17:      return  $\mathcal{P}(v_s, v_g), G$ 
18:    end if
19:  end for
20: end for
21: return  $\mathcal{P}(v_s, v_g) = \emptyset, G$ 

```

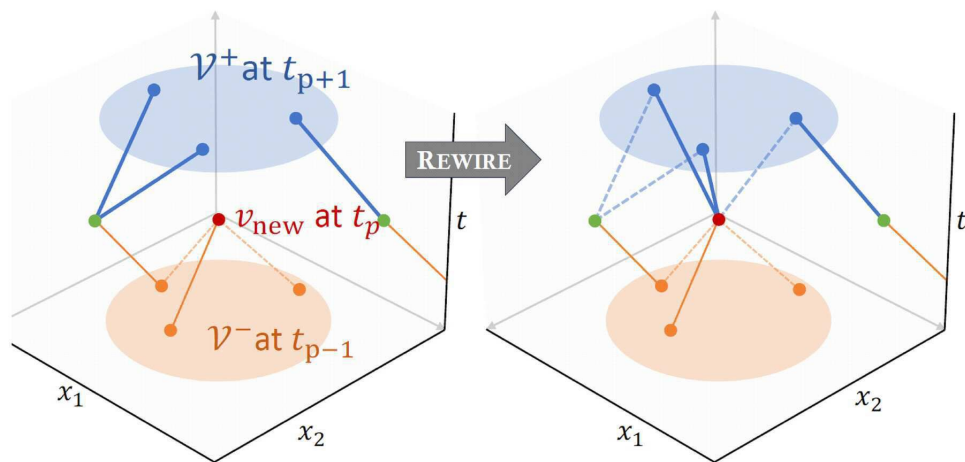
도면5



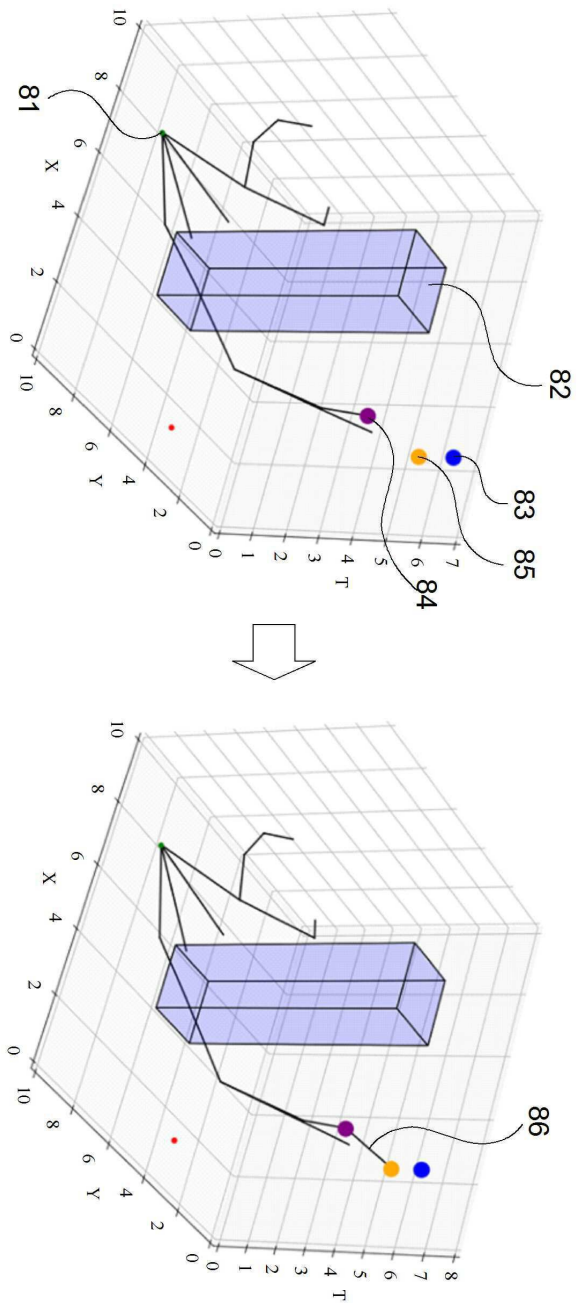
도면6



도면7



도면8



도면9

Algorithm 2: ST-CBS

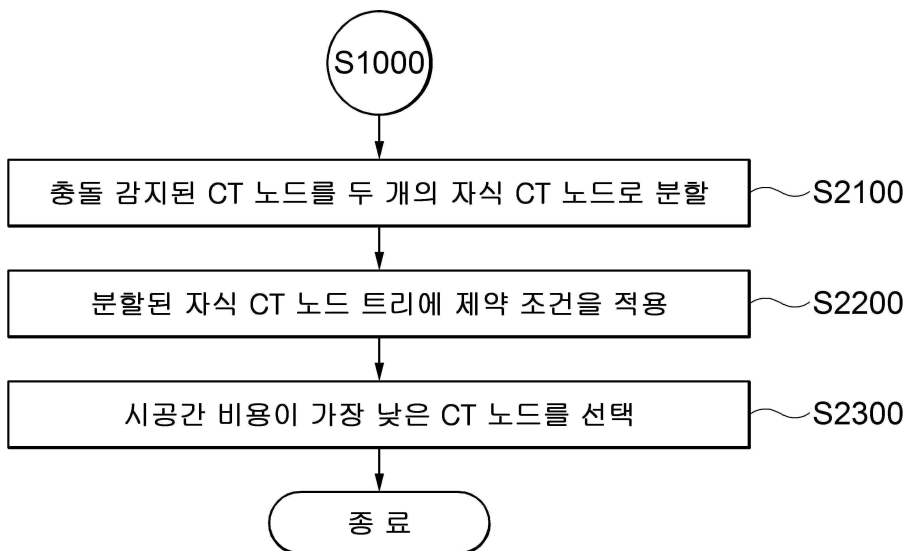
Input: All inputs of UST-RRT* for all agents
Output: Collision-free paths $\mathcal{P}$

```

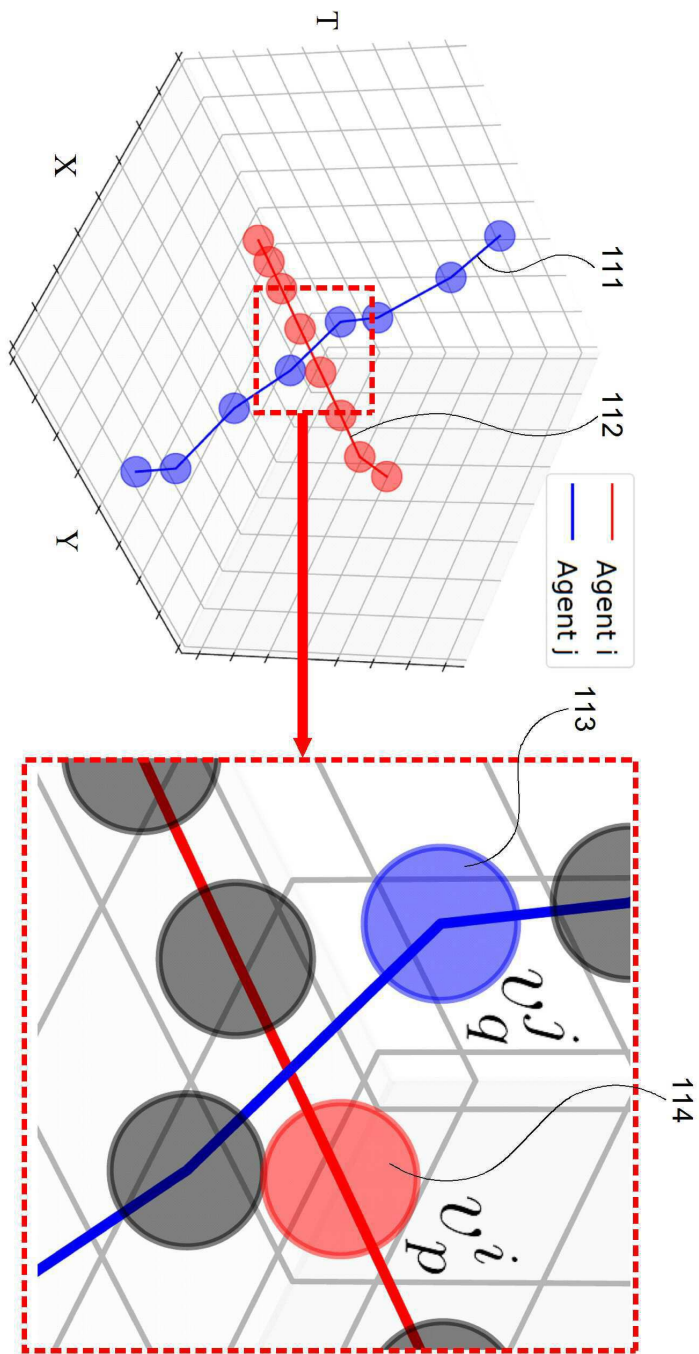
1:  $R.\mathcal{P}, R.G \leftarrow \text{UST-RRT}^*(\cdot)$  // omit input due to space limit
2:  $R.\kappa_{\text{ST}} \leftarrow h(R)$ 
3:  $\text{INSERT}(F, R)$ 
4: while  $F \neq \emptyset$  do
5:    $N \leftarrow \text{GETBESTNODE}(F)$ 
6:    $\mathcal{C} \leftarrow \text{GETFIRSTCONFLICT}(N.\mathcal{P})$ 
7:   if  $\mathcal{C} = \emptyset$  then
8:     return  $N.\mathcal{P}$ 
9:   end if
  \* Ln 10–20: superscript  $i, j$  on  $v, G$  are omitted for brevity \*
10:  for all  $v \in \mathcal{C} = \{v_c^i, v_c^j\}$  do
11:     $N' \leftarrow \text{COPY}(N)$ 
12:     $\text{PRUNENEIGHBOR}(v, N'.G)$ 
13:     $\text{PRUNECHILDREN}(v, N'.G)$ 
14:     $\text{MARKCONFLICTVERTEX}(v, N'.G)$ 
15:     $\mathcal{P}'(v_s, v_g) \leftarrow \text{GENERATEPATH}(N'.G, v_s, v_g)$ 
16:    if  $\mathcal{P}'(v_s, v_g) = \emptyset$  then
17:      continue
18:     $N'.\mathcal{P}(v_s, v_g) \leftarrow \mathcal{P}'(v_s, v_g), N'.\kappa_{\text{ST}} \leftarrow h(N')$ 
19:     $\text{INSERT}(F, N')$ 
20:  end for
21: end while
22: return  $\mathcal{P} = \emptyset$ 

```

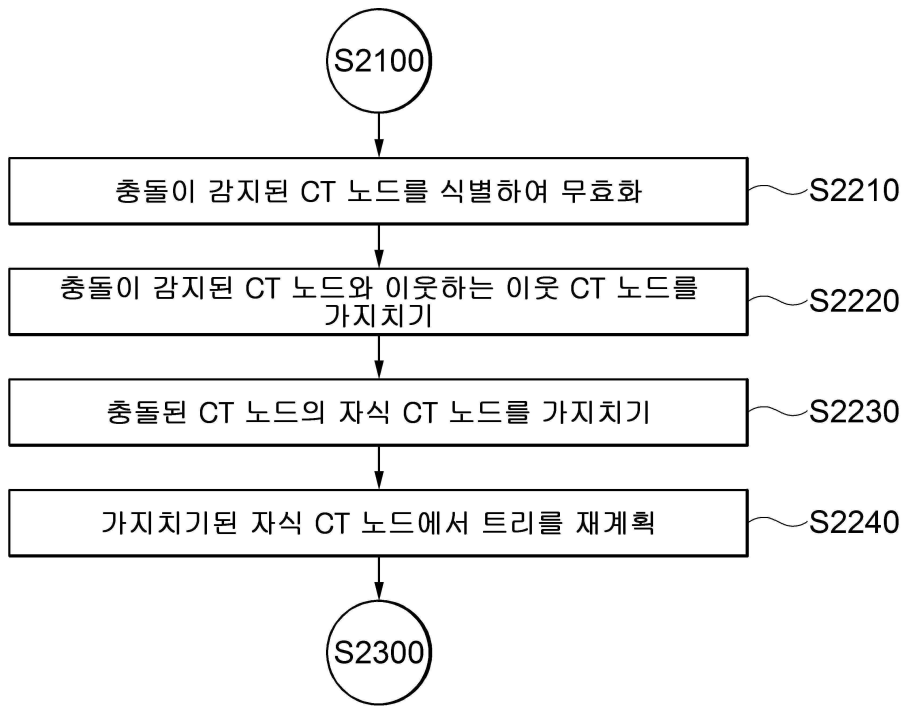
도면10



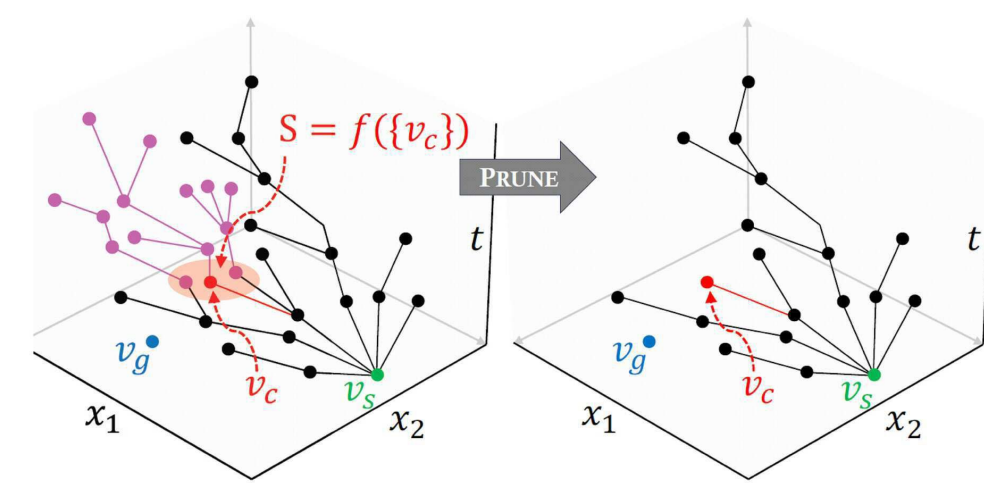
도면11



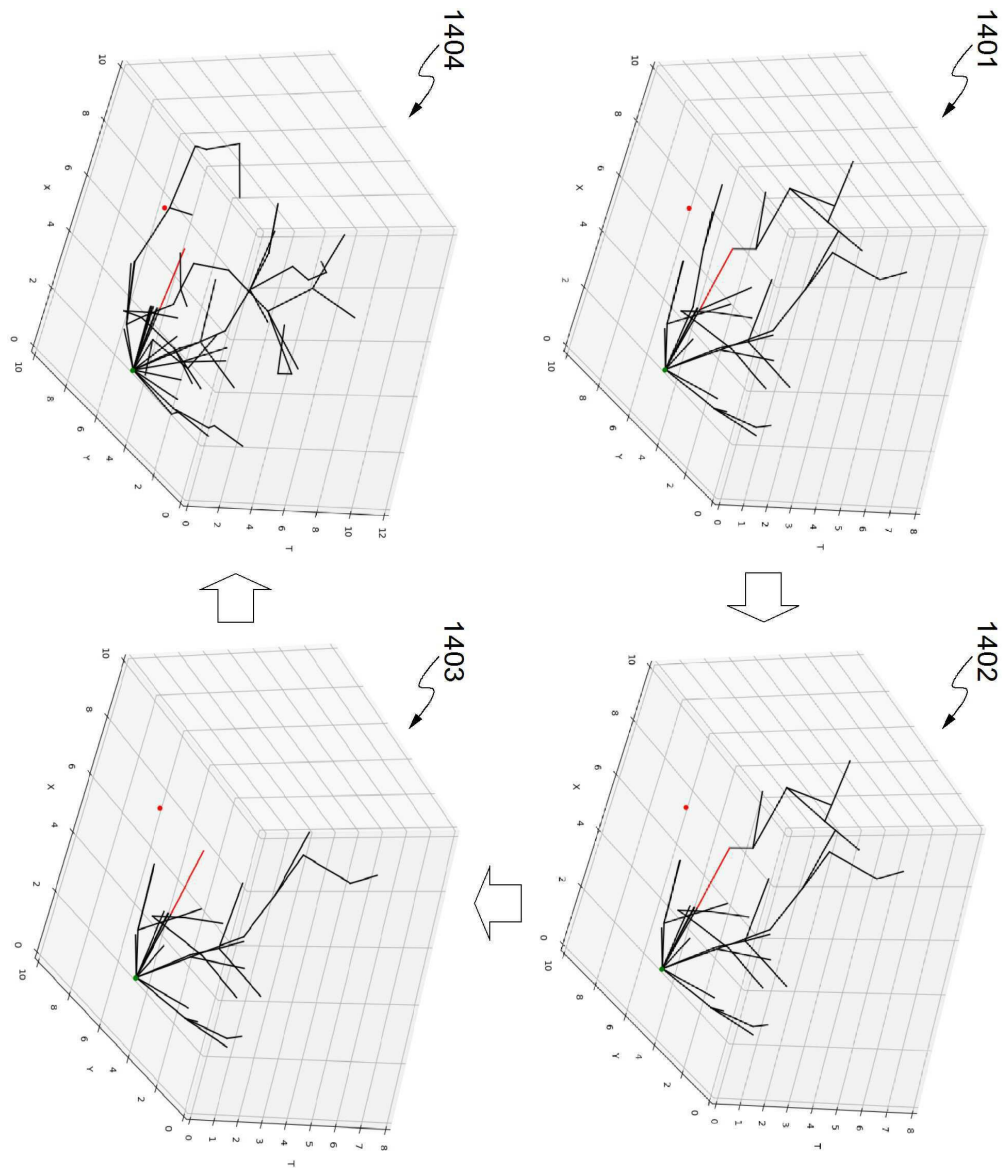
도면12



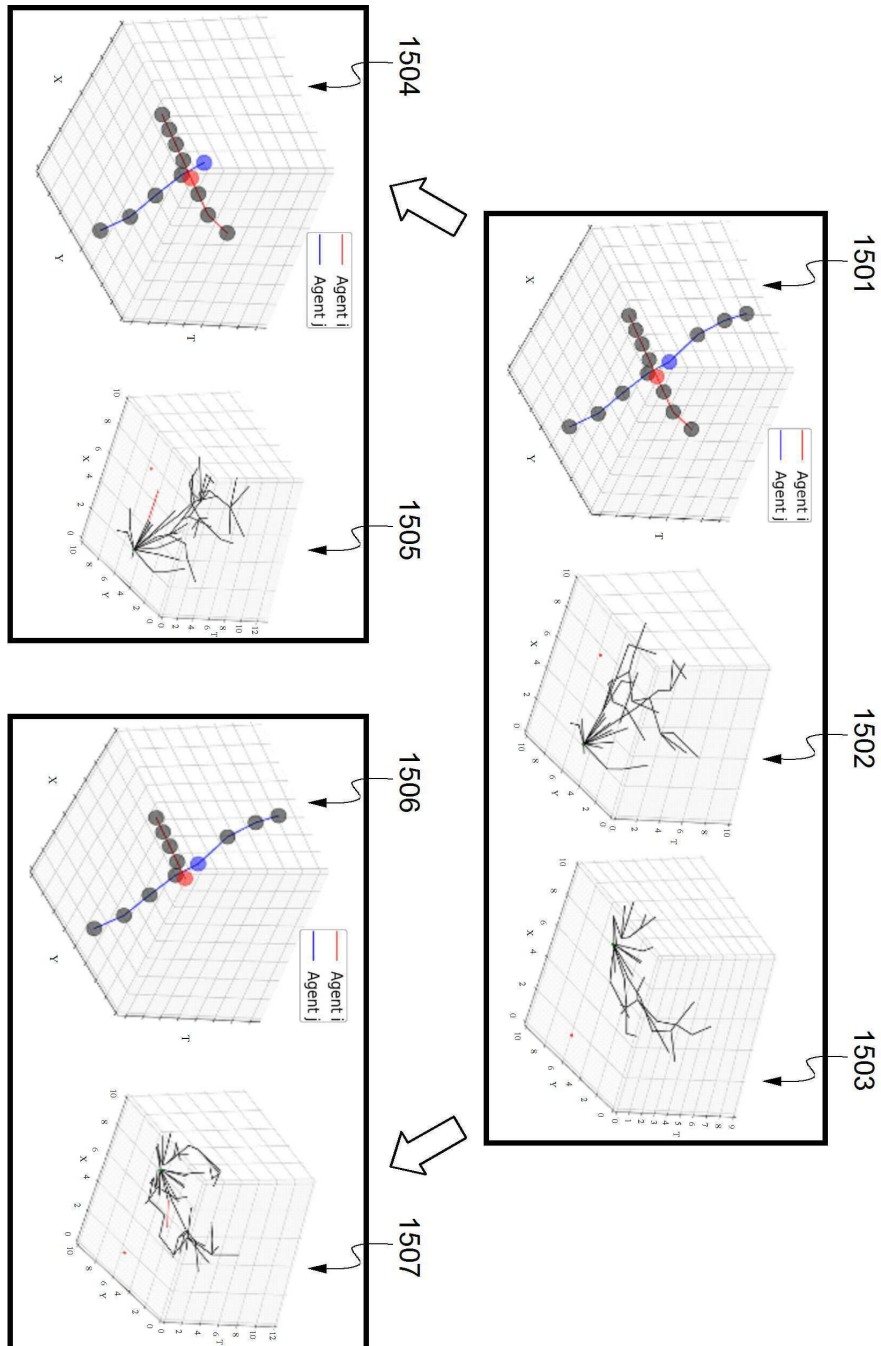
도면13



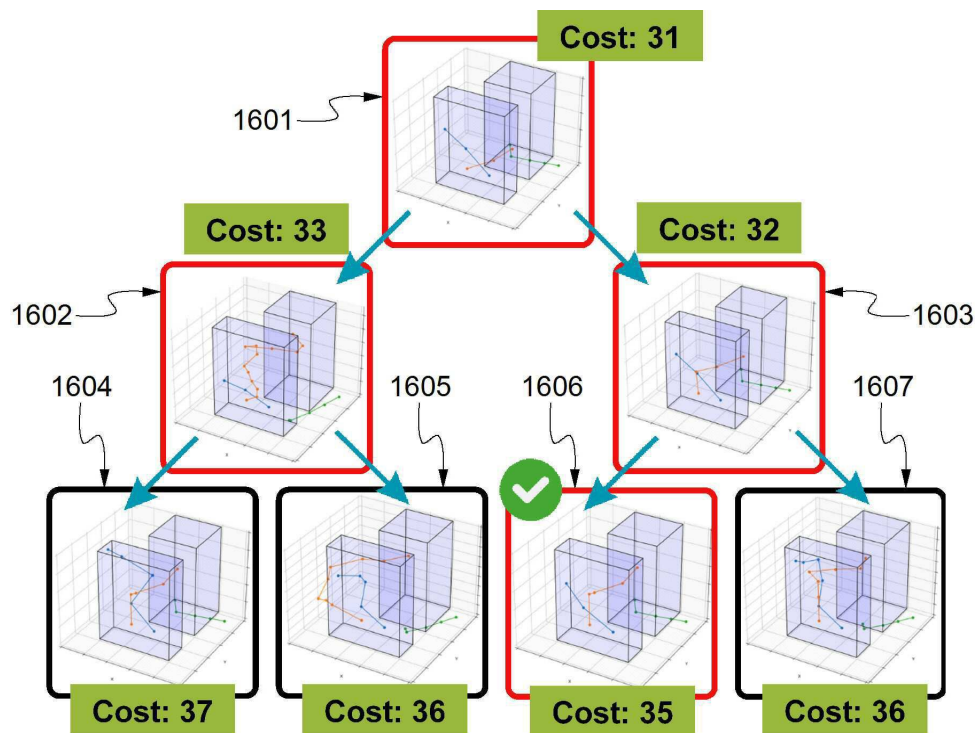
도면14



도면15

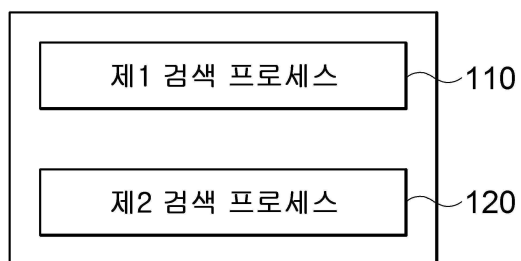


도면16

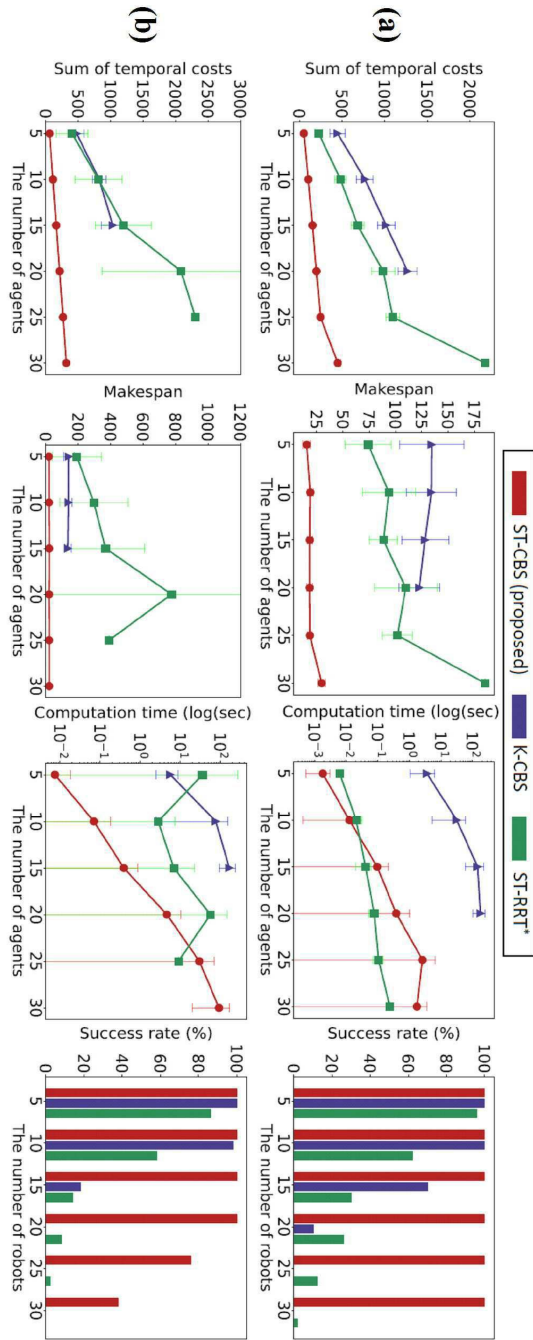


도면17

100



도면18



도면19

