Development of an AGV Algorithm for Parking Lot

Implementation of an AGV Control Algorithm and Verification via Simulation

Advisor : 안병권 교수

20202840 한정호

20203023 우상욱

ABC LAB
숭실대학교 차세대연결기술연구실

숭실대학교
Soongsil University

INDEX

Definition of AGV

- **What is the AGV?**

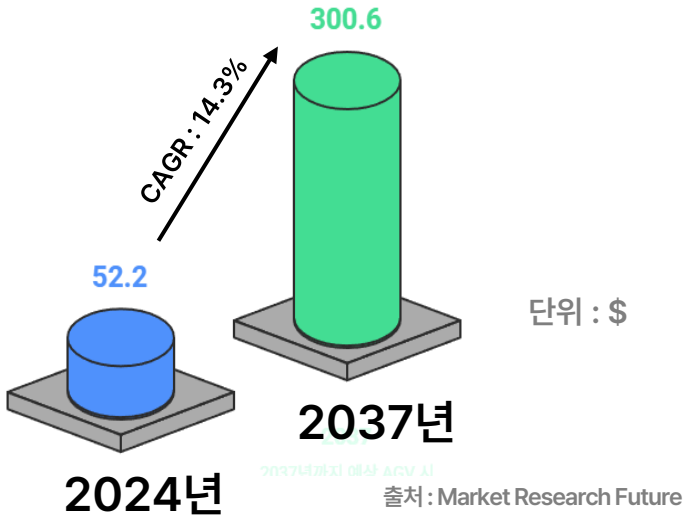
  ➢ Automated Guided Vehicle (무인 운반차)

  – 정해진 경로를 따라 자체 동력으로만 자재를 운반하는 목적을 가진 자동화 차량

출처 : Stiles Machinery

AGV Market Size

- **Present Market size and its projected future growth.**
  - ➢ 2024년 AGV시장규모, 약52억 2천만 달러로 평가
  - ➢ 2037년 약 300억 6천만 달러 예상
  - ➢ 연평균 성장률 약 14.3% 예측



Utilization of AGV Control System

- **Parking Robot**

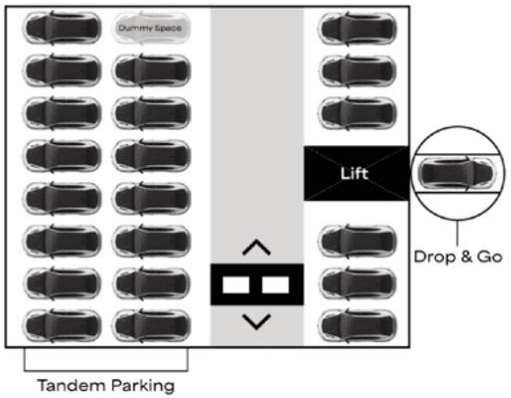
➢ 현대위아 vs HL만도, 주차로봇 대결 막 올랐다.

현대위아와 HL만도의 주차로봇 (사진=각사)

출처 - ZD코리아

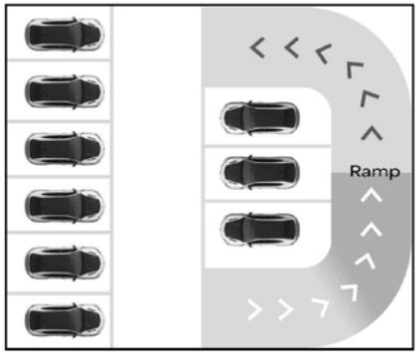
Necessity of AGV In Parking lot

- **주차장 공간 효율성 증가**
  - ➢ 차량 사이 간격 최소화로 주차 밀도 향상
  - ➢ 기존의 주차 도로를 주차공간으로 활용 가능



MPSystem

25 Spaces ↑



Conventional Parking

8 Spaces

출처 : 삼표

Necessity of AGV In Parking lot

- **사고 예방**
  - ➢ 인적 오류로 인한 충돌 사고 제거
  - ➢ 알고리즘 기반 안전 주행으로 차량 손상 방지
  - ➢ 2023년 주차장 사고 건수 1542건, 하루 평균 4.2건 꼴로 발생 (출처 : 경찰청)

2023년 주차장 사고 건수

1542건

하루평균 4.2건의 주차장 사고 발생

Necessity of AGV In Parking lot

- **환경 오염 감소**
  - ➢ 주차장 무인화에 따른 차량 주행 배기가스 배출 '0mg'
  - ➢ 주차장 주변 환경의 공기질 개선

<지하주차장의 평균 일산화탄소 농도>

$$K = \frac{0.06 \cdot X \cdot M \cdot L}{V \cdot Q}$$

단, K : 지하주차장의 평균 일산화탄소 농도 [m^3/m^3]

X : 자동차 출입대수 [대/h]

M : 주행시 일산화탄소 평균배출량 [$m^3/min \cdot 대$]

L : 평균주행거리 [m/대]

V : 평균주행속도 [$km/h \cdot 대$]

Q : 환기량 [m^3/h]

여기서, 0.06 은 환산계수이다.

Multi-Agent Path Finding [MAPF]

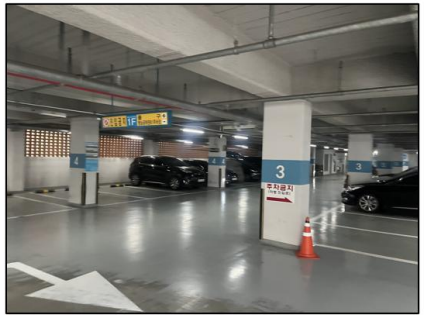
- **Definition of MAPF**
  - ➢ 여러 에이전트가 공유된 환경에서 충돌없이 이동하는 경로 집합을 찾는 문제
- **Key Components of MAPF**
  - ➢ Agents
  - ➢ Environment
  - ➢ Start / Goal
  - ➢ Time
- **Collision Types**
  - ➢ 정점 충돌 [Vertex Confilct]
    - 동일 시간에 동일 정점[위치] 차지

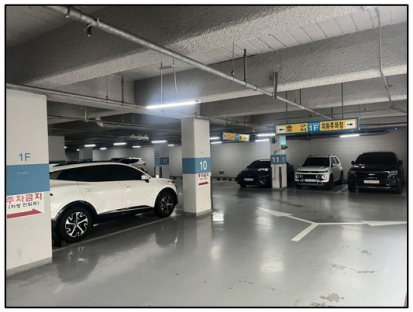
  - ➢ 간선 충돌 [Edge Conflict]
    - 동일 시간 사이에 동일 간선[경로]를 교차하여 통과

Environment : 형남공학관 주차장

- **형남공학관 주차장**

  ➢ 형남공학관 주차장 지하2층 전경

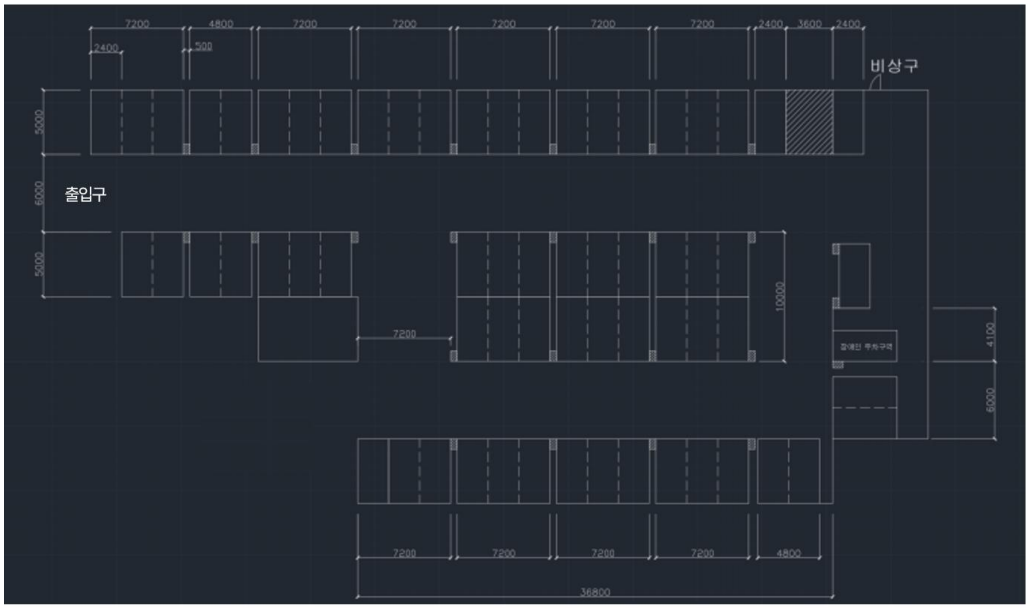




Environment : 형남공학관 주차장

- **형남공학관 주차장**

  ➢ 형남공학관 주차장 지하2층 도면



[0] Priority System : Score / Stuck Boost

- **Priority Score**
  - ➢ $\text{Priority}(A) = (Base \times 100) + StuckBoost - A.id$
  - ➢ 출차 [RETURNING_WITH_CAR] : 3
  - ➢ 충천 필요 [GOING_TO_CHARGE] : 2
  - ➢ 주차 / 차량 수거 [GOING_TO PARK / GOING TO COLLECT] : 1
  - ➢ 대기 / 충전 중 / 기지로 복귀 [IDLE / CHARGING / RETURNING_HOME] : 0

- **Stuck Boost Mechanism**
  - ➢ $StuckBoost = \begin{cases} 1000 & if\ A.stuck \geq DEADLOCK\_THRESHOLD \\ A.Stuck \times 10 & otherwise \end{cases}$

- **Tie-Breaking Mechanism**
  - ➢ AGENT ID = $A.id$ [ex. A > B > C]

[0] Priority System : pseudo-code

- **Pseudo-code[의사코드]**

➢

```
// basic priority based on agent.state
p = base(state)

//Stuck Boost
if (stuck_steps >= deadlock_threshold)
   p += stuck_boost_hard
else if (stuck_steps > 0)
   p += stuck_steps * stuck_boost_mult

// Tie-breaking mechanism
tie_break(A.id)

// plan and application
plan_step() → reservation → move → stuck update→ (metrics_notify_all)
```

[1] A *algorithm : Definition

- **What is the A*algorithm?**

➢ A* algorithm

- 그래프 기반 최단 경로 탐색에서 가장 널리 쓰이는 알고리즘
- Shortest Path = Path with Min[$f(n)$]

➢ $g(n)$

- 출발 노드부터 현재 노드까지 실제 소요 비용

➢ $h(n)$ [Manhattan Distance]

- $h(n) = |x_n - x_{goal}| + |y_n - y_{goal}|$
- 현재 노드부터 목표 노드까지 예상 비용
- 휴리스틱 함수 활용 추정치

➢ $f(n)$

- 총 예상 비용 [Total Estimated Cost]
- $f(n) = g(n) + h(n)$

[1] A *algorithm : pseudo-code / Diagram

- pseudo-code

```
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) then
      continue

    temp_obstacles =
    get_higher_priority_moves(agent, next_pos) +
    get_lower_priority_positions(agent, agents)
    temp_pathfinder = create_pathfinder(agent.pos,
agent.goal)
    path = compute_path(
          temp_pathfinder, map, temp_obstacles
      )
    desired_move = get_next_step(path)
    destroy_pathfinder(temp_pathfinder)
    next_pos[agent.id] = apply_rotation(agent, desired_move)

resolve_final_conflicts(next_pos, sorted_agents)
return next_pos
```

[2] D* lite algorithm : Definition

- **What is the D*lite algorithm?**

➢ D* lite algorithm

- 동적 환경에서 주로 쓰이는 알고리즘
- 목표에서 역방향으로 모든 지점까지의 최단 경로 비용 계산
- Shortest Path = All Consistent path

➢ $g(s)$

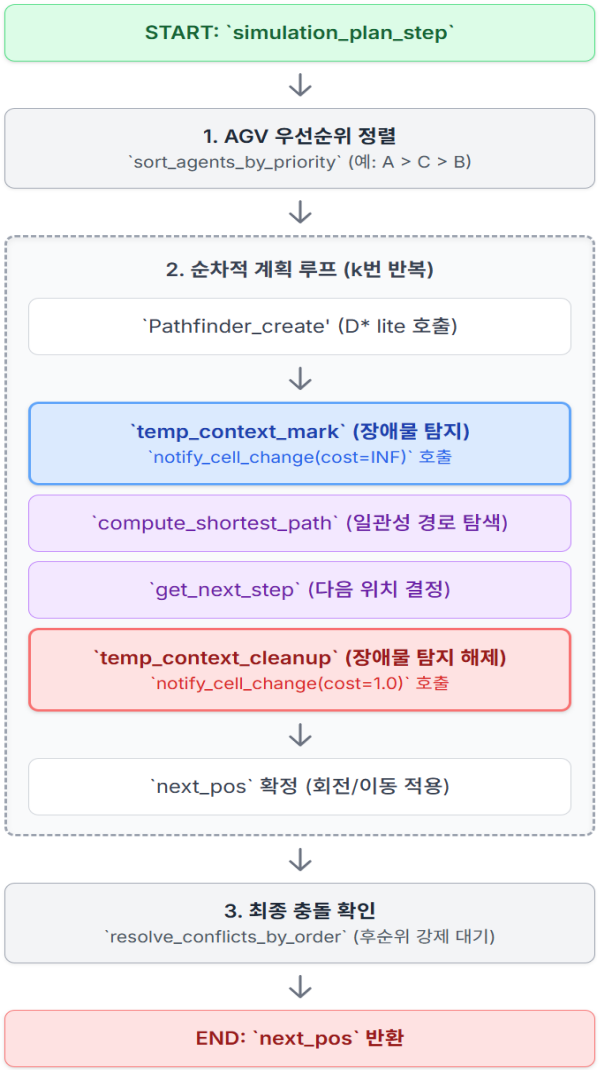
- 목표 노드부터 현재 노드까지 '**알려진**' 추정 소요 비용

➢ $rhs(s)$

- $g(s)$ 의 1- step 예측 값.
- $rhs(s) = \begin{cases} 0 & if\ s = s_{goal} \\ min_{s' \in Succ(s)}(c(s,s') + g(s')) & otherwise \end{cases}$

➢ $C(s, s')$

- S′ -> S 비용



[2] D* lite algorithm : Definition

- **What is the D*lite algorithm?**

➢ *Consistency*

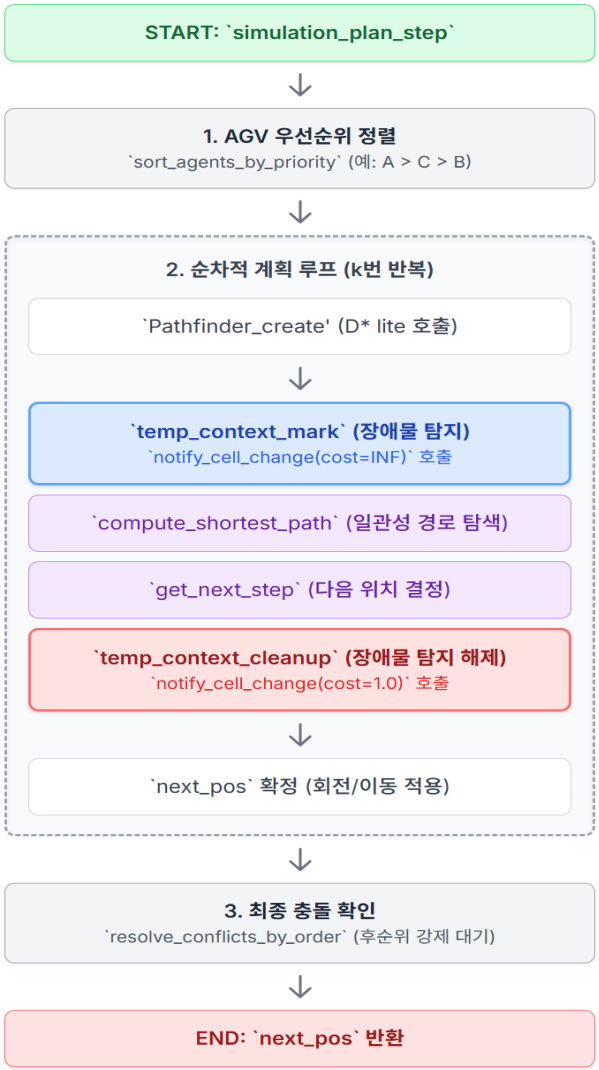
- $g(s) = rhs(s)$: Node 's' is consistent.
- $g(s) \neq rhs(s)$: Node 's' is inconsistent. -> Update Target

➢ *Priority Key*

- $Key(s) = [k_1(s); k_2(s)]$
- 우선순위 큐 사용하여 '비일관성' 노드 갱신 -> 일관성 's' 탐색
- 총 예상 경로 비용 : $k_1(s) = min(g(s), rhs(s)) + h(s_{start}, s) + k_m$
- 현재노드 ~ 목표노드 예상 비용 : $k_2(s) = min(g(s), rhs(s))$

➢ *ComputerShortestpath()*

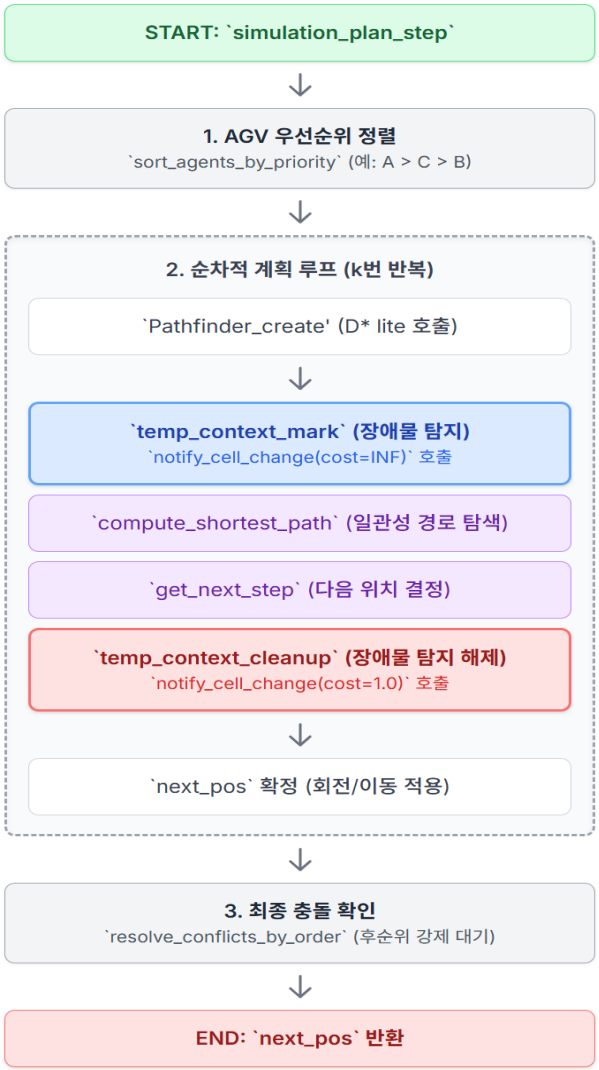
- while $\left(\min_{u \in U} key(u) < key(s_{start})\right)$ or $(rhs(s_{start}) \neq g(s_{start}))$



[2] D* lite algorithm : pseudo-code / Diagram

- pseudo-code

```
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) then
      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
```



[3] D* Lite + WHCA* + WFG/SCC + Partial CBS : WHCA*

- **What is the WHCA* algorithm?**

➢ WHCA* [Windowed Hierarchical Cooperative A*]

- 우선순위, 예약 테이블을 활용한 MAPF 알고리즘

➢ Pathfinder Algorithm

- D* lite : Provides the Ideal route

➢ W : Space-Time Reservation Table

- [time][x][y]의 3차원 예약 테이블 (0 < time < H) *H = Horizon

➢ H [Hierachical]

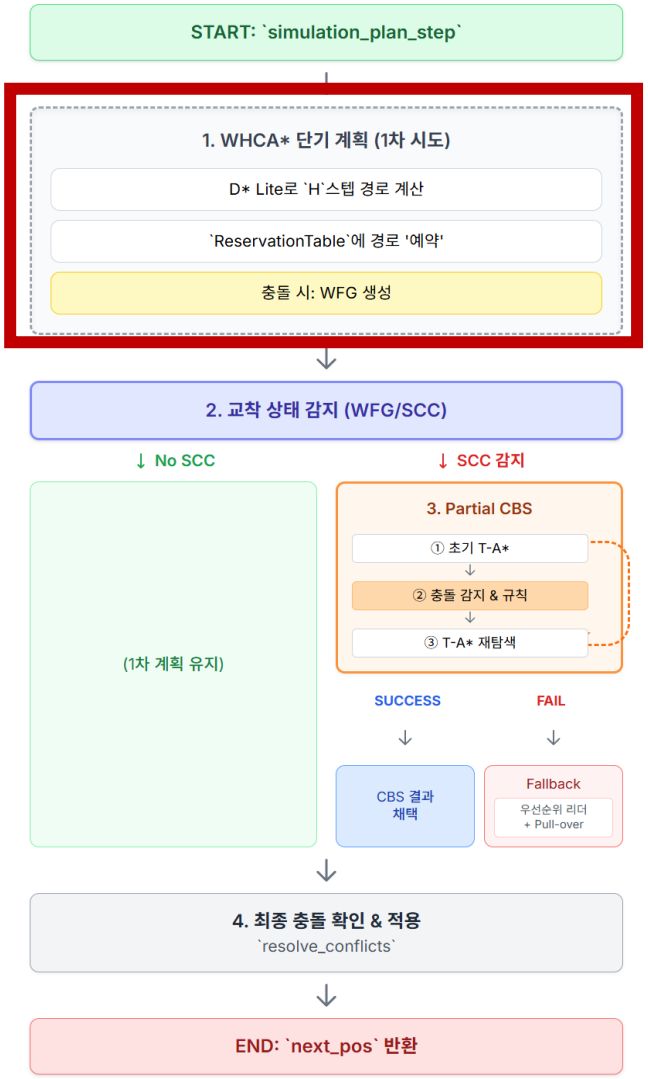
- 우선순위에 따른 순차적 경로 계획

➢ C [Cooperative]

- 타 에이전트 예약 노드 -> 장애물

➢ Limitations

- 우선순위 기반 알고리즘 -> 'Risk of Deadlock'



[3] D* Lite + WHCA* + WFG/SCC + Partial CBS : WFG/SCC

- **What is the WFG?**

➢ WFG [Wait-For Graph]

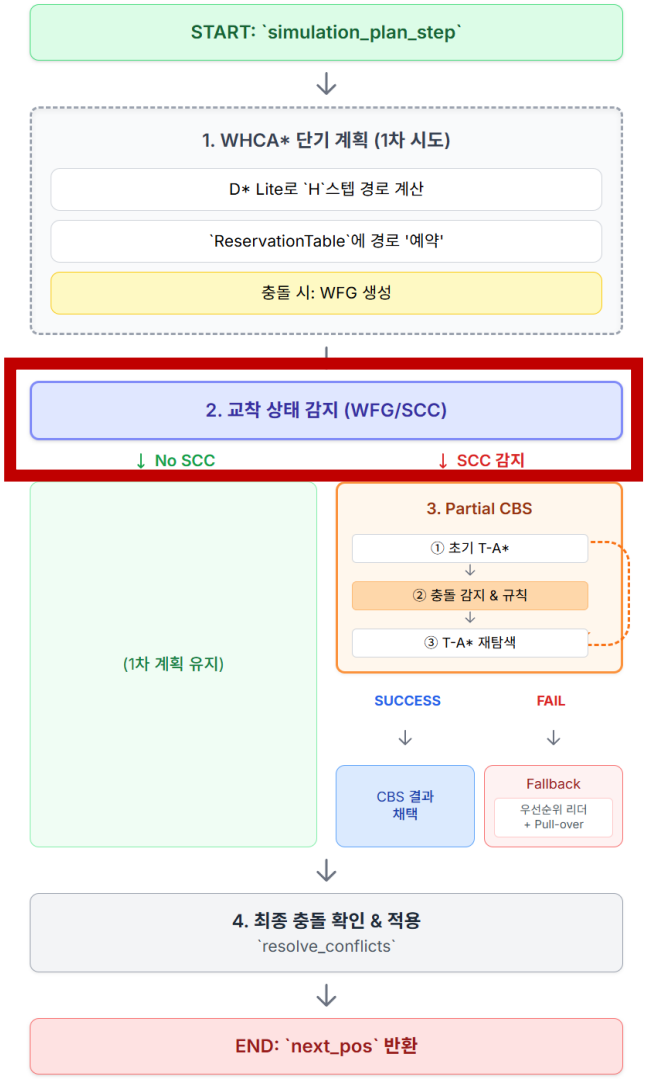
- 현재 시스템 대기 상태 표현하는 자료구조.

➢ Role

- 교착 상태 탐지위한 자료구조

➢ Components

- $G = (V, E)$
- V [Vertex / Node]: 정점의 집합 (운용 AGV: $\{A_1, A_2, \dots\}$)
- E [Edge] : 간선의 집합 (대기 관계: $\{(A_i, A_j) \mid A_i \text{ waits for } A_j\}$)



[3] D* Lite + WHCA* + WFG/SCC + Partial CBS : WFG/SCC

- **What is the SCC?**

➢ SCC [Strongly Connected Compnents]

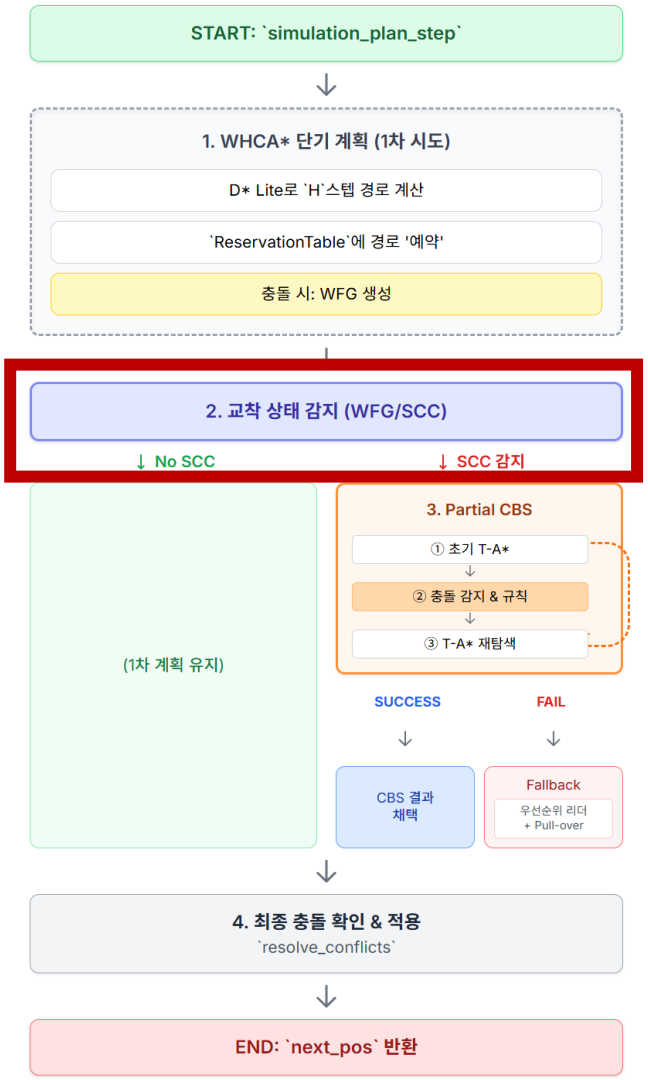
- 그래프 내에서 서로에게 도달 가능한 정점들의 집합
- 크기 2이상 SCC 존재 [ex. $A_i \leftrightarrow A_j$], -> 교착 상태 의미

➢ Role

- 자료구조에서 교착상태 감지

➢ DeadLock Detetction

- Deadlock $\Leftrightarrow \exists$ SCC $C \subseteq V$ s.t. $|C| \geq 2$



[3] D* Lite + WHCA* + WFG/SCC + Partial CBS : CBS

- **What is the CBS algorithm?**

➢ CBS [Conflict – Based Search]

- 지역적 충돌 해소 위한 MAPF 알고리즘
- 소수 에이전트 간의 충돌을 효율적으로 해결함.
- 시스템의 계산 효율성을 향상.

➢ 저수준 [Low – Level] : Time A* -> 개별 경로 탐색기

- [time][x][y] 구조에서 A* 탐색 수행
- State (상태): $s = (x, y, t)$
- 평가 함수 (f-value): $f(s) = g(s) + h(s)$
- $g(s)$: $g(x, y, t) = t$ ($(x_0, y_0, 0)$ -> (x, y, t) : 비용 t)
- $h(s)$: $h(x, y, t) = |x - x_{goal}| + |y - y_{goal}|$

[3] D* Lite + WHCA* + WFG/SCC + Partial CBS : CBS

- **What is the CBS algorithm?**

➢ 고수준 [High – Level] : 충돌 해결 위한 제약 조건 결정

- Cost(N) 가장 낮은 노드 N -> 우선순위 큐에서 선택

$$Cost(N) = \sum_{i \in Agents} PathCost(P_i)$$

- N: 충돌 트리의 노드 [제약 조건의 집합]
- P_i: 노드 N의 제약 조건 & 저수준 탐색 기반 에이전트 i의 경로
- PathCost(P_i): 경로 비용

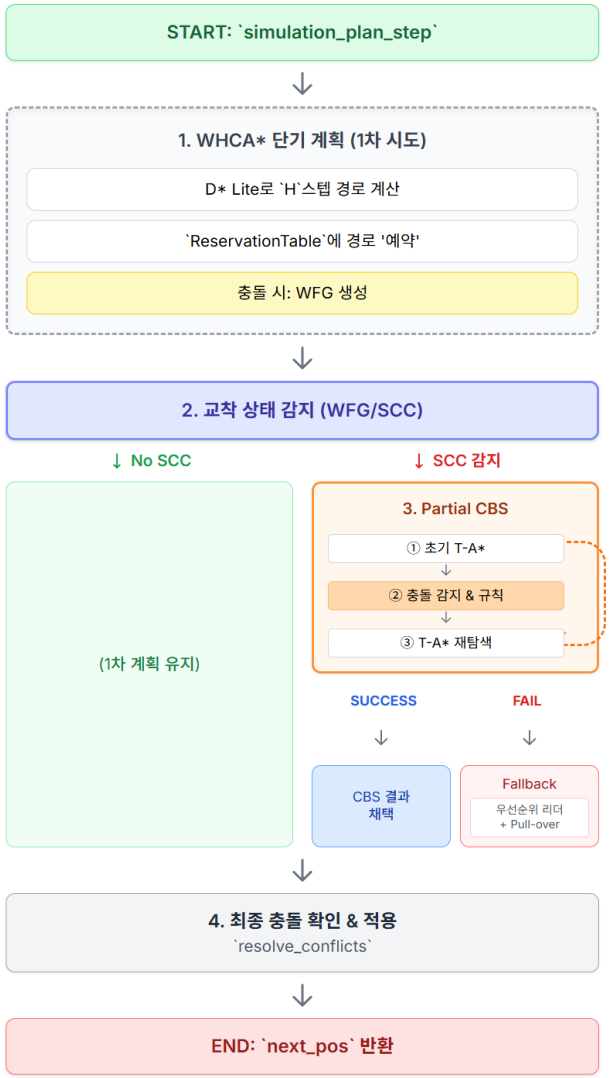
$$\alpha \cdot "\{moves\} + \beta \cdot "\{waits\} + \gamma \cdot h\_\{residual\}$$

➢ 제약 조건 (Constraint) 예시:

- High Level에서 $(Agent\ A, (5,5), t = 3)$ 제약이 내려오면,
- Low Level은 $(5,5,3)$ 상태를 고려하지 않는다.

[3] D* Lite + WHCA* + WFG/SCC + Partial CBS : DeadLock resolve

- **1. 교착 감지**
  - ➢ WFG & SCC 활용하여 교착 상태 식별
- **2. Partical CBS**
  - ➢ SCC 식별 그룹에 Partial CBS 적용
- **3. 우선순위 기반 해결**
  - ➢ CBS 실패할 시, 우선순위 기반 해결
  - ➢ A*, D* lite와 동일한 우선순위 기준
- **4. Pull – over**
  - ➢ Max Stuck value AGV -> Leader AGV
  - ➢ 우선순위 실패할 시, Leader 제외 AGV 한 칸씩 후진.



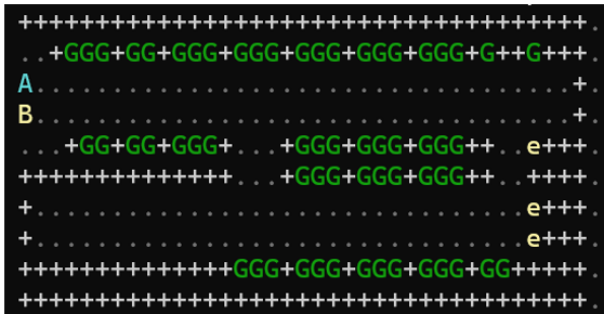
[3] D* Lite + WHCA* + WFG/SCC + Partial CBS : pseudo-code

```
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:
// (Plan H_steps using D* Lite)
  path = dstar_find_path(agent.pf, h_steps, rt)

// (Build WFG - Wait-For Graph)
  if has_conflict(path) with other_agent then
   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 forward

 // 2. wfg/scc (Deadlock detection)
 sccmask = detect_scc(wf_edges)
```

```
// 3. partial cbs (Deadlock resolution branch)
if sccmask > 0 then
  // (Deadlock occurred)
 deadlocked_group = get_agents_from_mask(sccmask)
  // (Attempt negotiation)
 negotiated_plans, ok
        = run_partial_cbs(deadlocked_group, rt)

  if ok == true then
   // (Negotiation success: Overwrite with CBS results)
    overwrite_next_pos(provisional_plan, negotiated_plans)
  else
// (Negotiation failure: Only leader moves)
    fallback_leader_moves(provisional_plan, deadlocked_group)
// No deadlock. Keep the first-attempt plan
else : (provisional_plan)

 // 4. Final
 resolve_final_conflicts(provisional_plan, sorted_agents)

 return provisional_plan
```

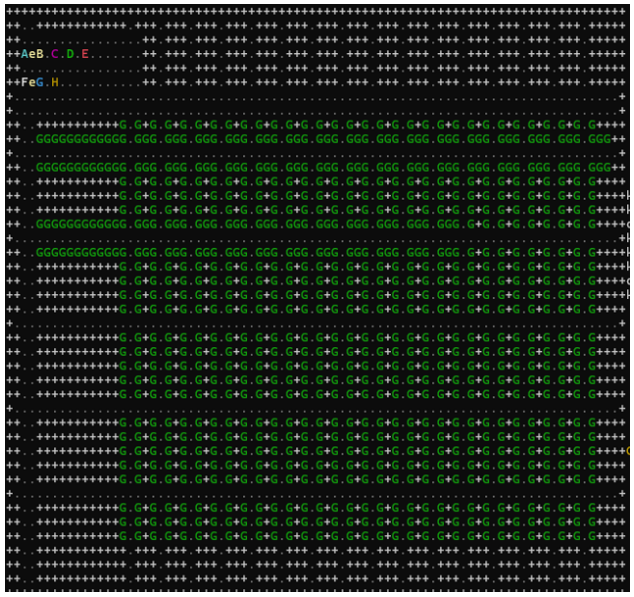
Simulation Map

- **형남공학관 주차장**
  - AGV : 1~4
  - Parking : 61

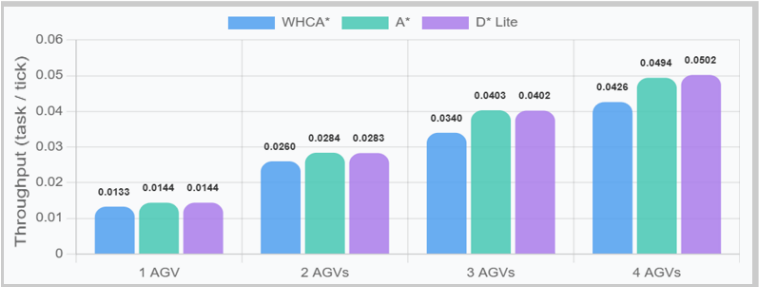
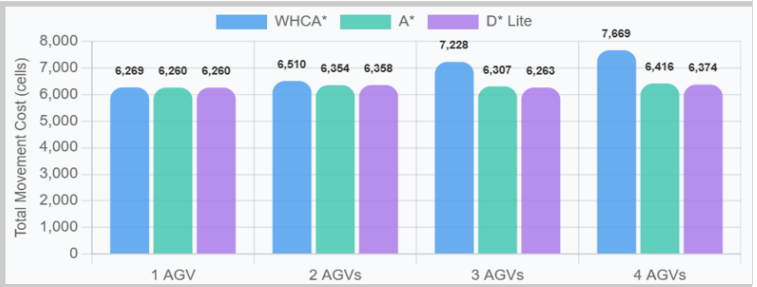
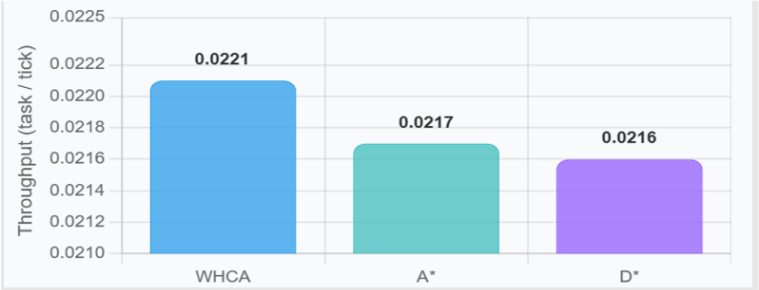
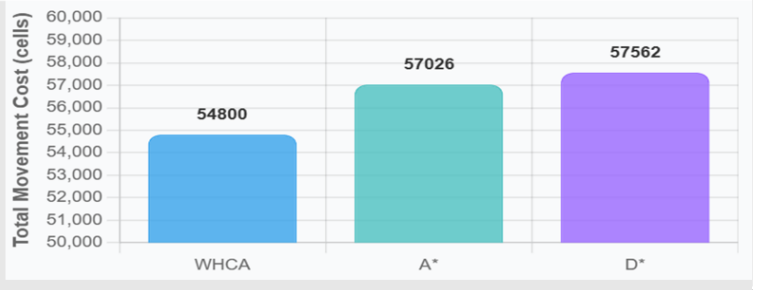
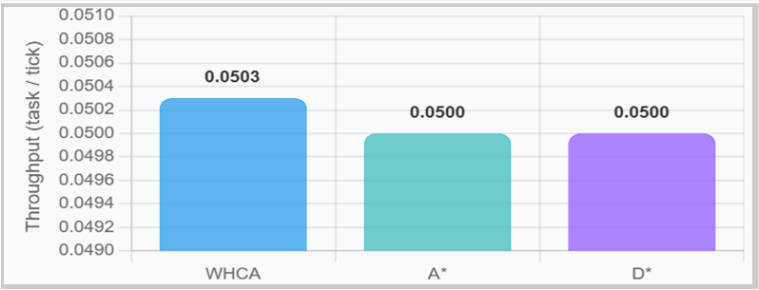
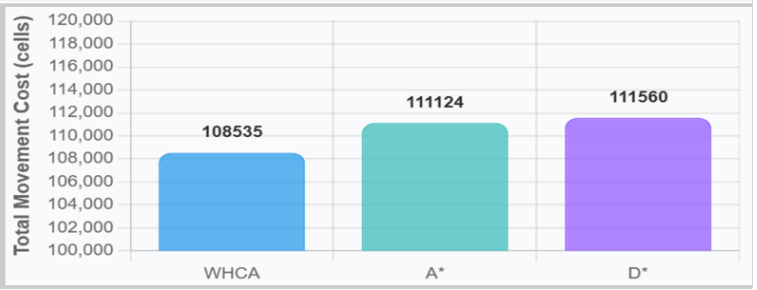


- **MAP2**
  - AGV : 4
  - Parking : 390

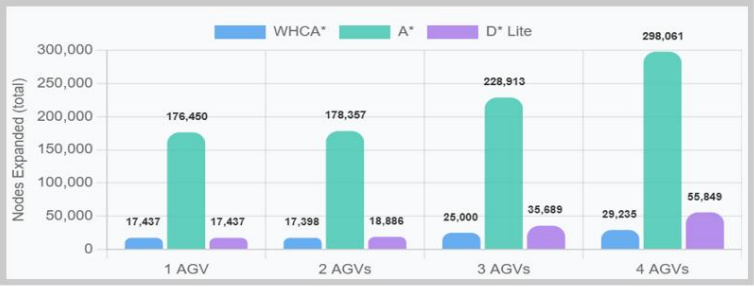
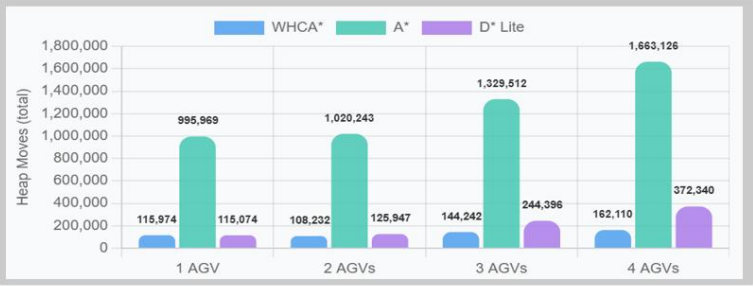
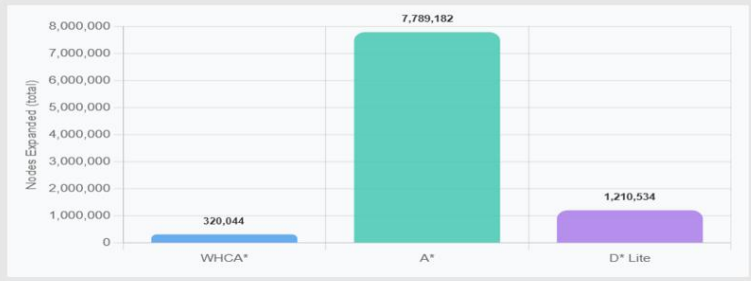
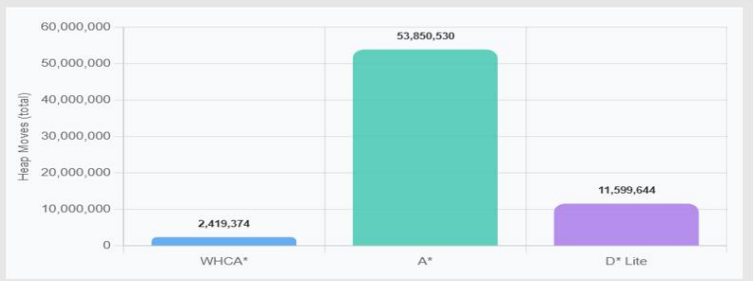
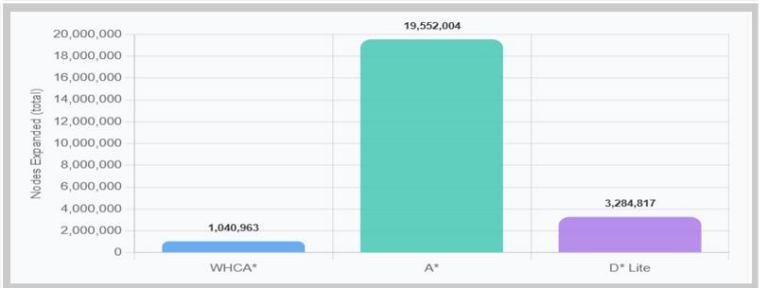
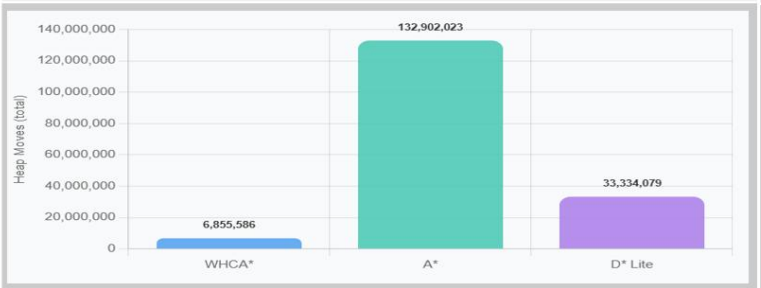
- **MAP3**
  - AGV : 8
  - Parking : 900



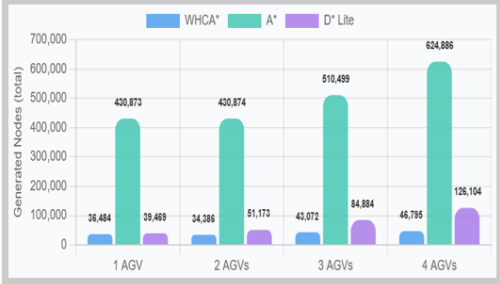
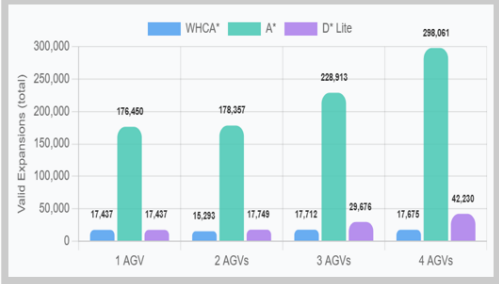
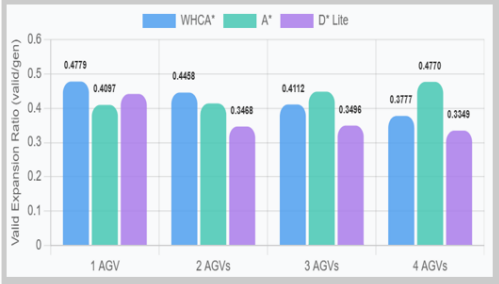
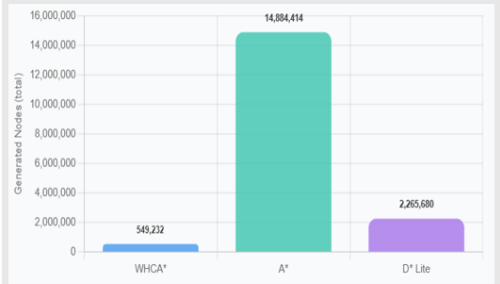
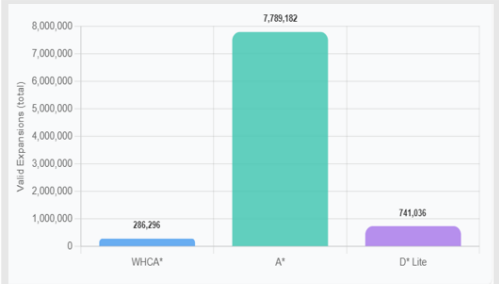
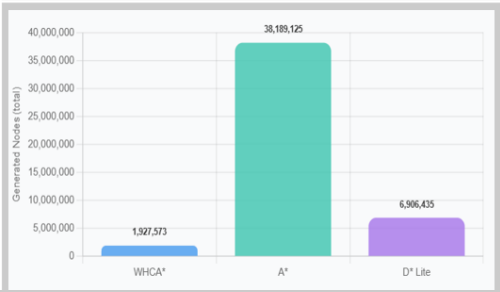
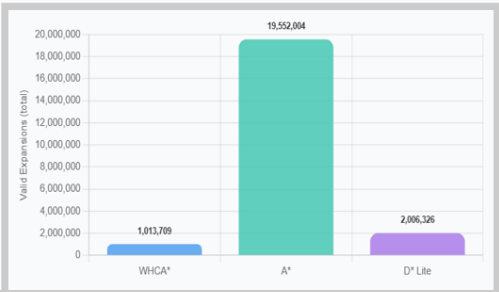
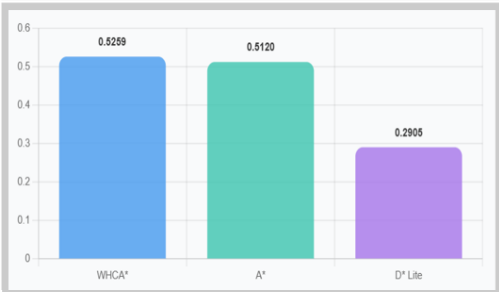
Simulation Result

MAP	Throughput	Total Movement Cost
형남공학관 주차장 Parking : 61		
AGV : 4 Parking : 390		
AGV : 8 Parking : 900		

Simulation Result

MAP	Nodes Expanded	Heap Moves
형남공학관 주차장 Parking : 61		
AGV : 4 Parking : 390		
AGV : 8 Parking : 900		

Simulation Result

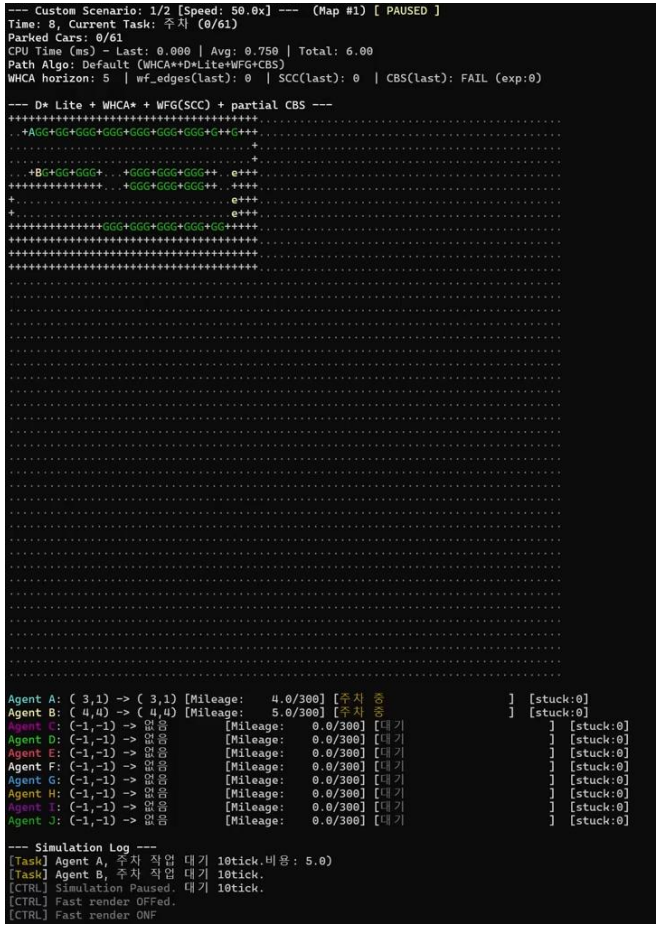
MAP	Generated Noes	Valid Expansions	Valid Expansion Ratio
형남공학관 주차장 Parking : 61			
AGV : 4 Parking : 390			
AGV : 8 Parking : 900			

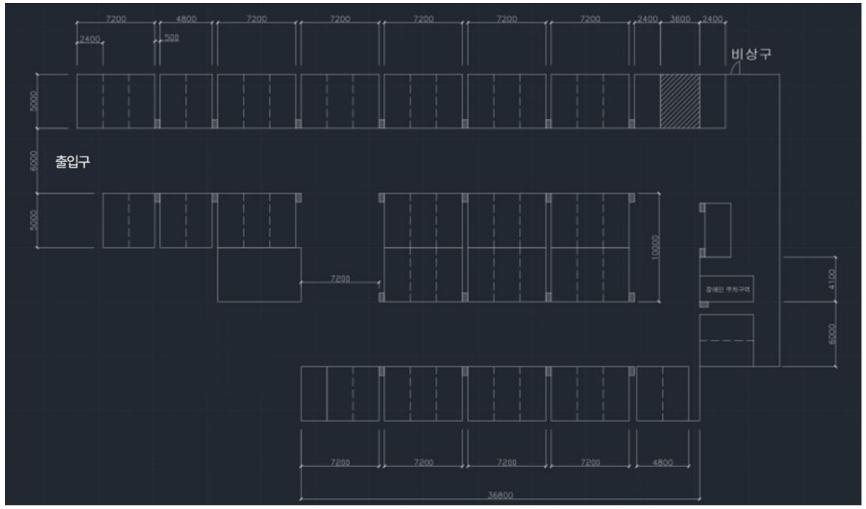
Conclusion

- **Which algorithm has the highest throughput in a low-density environment?**
  - ➢ A* algorithm And D* lite Algorithm
- **Which algorithm has the highest throughput in a High-density environment?**
  - ➢ WHCA* + WFG/SCC + CBS Algorithm
- **Which algorithm requires the least computation under the same conditions?**
  - ➢ 1st : WHCA* + WFG/SCC + CBS Algorithm
  - ➢ 2nd : D* ltie Algorithm
  - ➢ 3rd : A* Algorithm

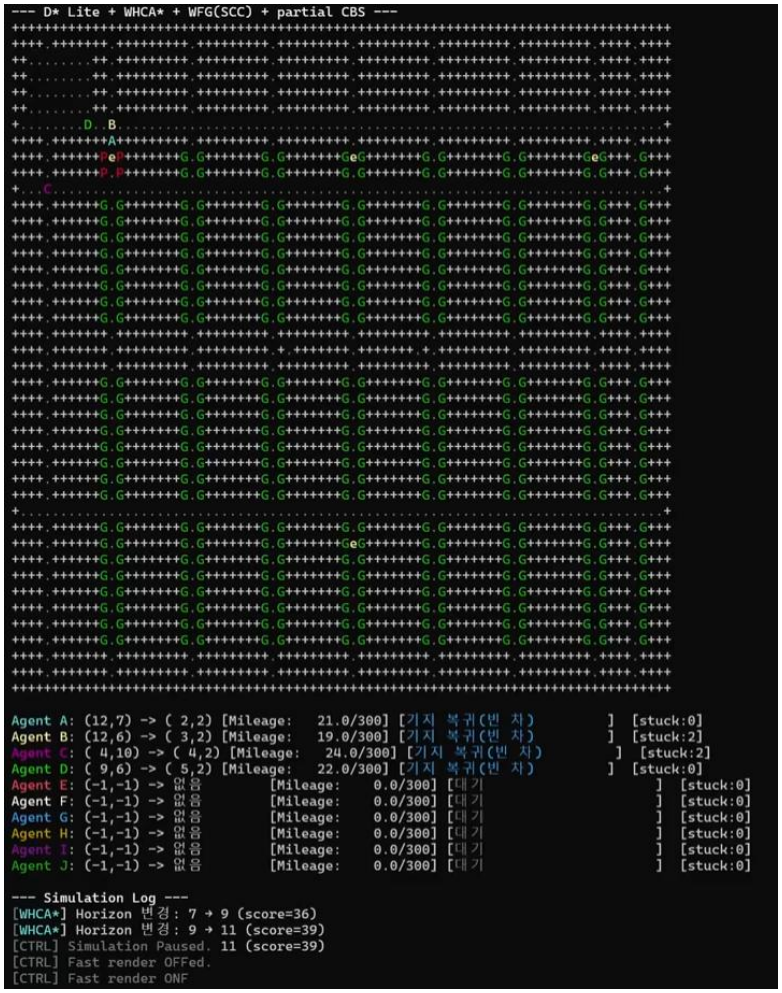
- **"The optimal algorithm varies depending on the context."**

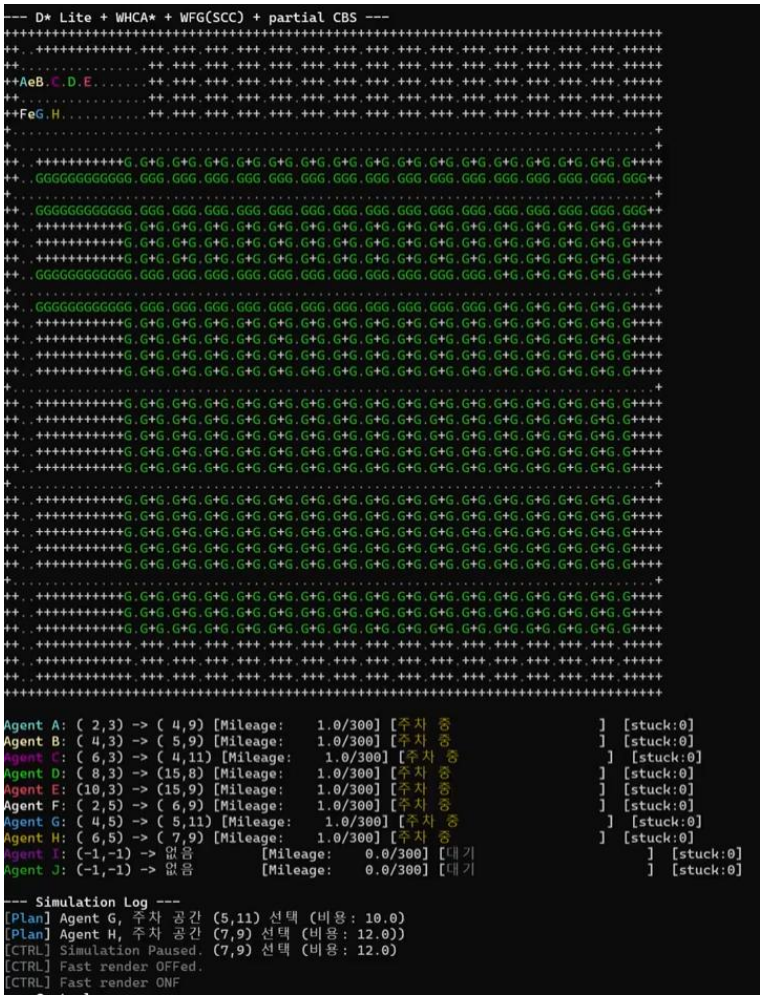
시뮬레이션 영상 [형남 공학관 주차장 + (WHCA*+WFG/SCC +CBS)]





시뮬레이션 영상 [Map2 / Map3]





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 Int. Conf. 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 Conf. on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), 2005.
- [5] Z. Bnaya and A. Felner, "Conflict-Oriented Windowed Hierarchical Cooperative A*," in Proc. IEEE Int. Conf. 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, vol. 219–220, pp. 40–66, 2015.
- [7] Z. Ren, S. Rathinam, and H. Choset, "Multi-objective Conflict-based Search for multi-agent path finding," in Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), 2021.
- [8] J. Kottinger, S. Almagor, and M. Lahijanian, "Conflict-based search for multi-robot motion planning with kinodynamic constraints," in Proc. IEEE/RSJ Int. Conf. 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 Symp. on Switching and Automata Theory, 1971.
- [12] A. Stentz, "The Focussed D* algorithm for real-time replanning," in Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI), 1995.
- [13] Bhosekar, A., Işık, T., Ekşioğlu, S., Gilstrap, K., & Allen, R. (2021). Simulation-optimization of automated material handling systems in a healthcare facility. IISE Transactions on Healthcare Systems Engineering, 11(4), 316–337.
- [14] Koenig, S., Likhachev, M., & Furcy, D. (2005). Lifelong Planning A.* Preprint submitted to Elsevier Science on May 24, 2005.
- [15] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100–107.
- [16] Ning, S., Zhong, J., & Zheng, X. (2021). Design of virtual intelligent parking lot system based on signal request mechanism. Proceedings of the 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Dalian, China.
- [17] Shi, Y., Pan, Y., Sun, X., Xie, R., Chen, W., & Shen, S. (2018). Collaborative planning of parking spaces and AGVs path for smart indoor parking system. Proceedings of the 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design (CSCWD), Dalian, China.
- [18] Bobanac, V., & Bogdan, S. (2008). Routing and scheduling in multi-AGV systems based on dynamic banker algorithm. In Proceedings of the 16th Mediterranean Conference on Control and Automation (pp. 1168–1173). IEEE
- [19] 김중기, 박병윤, 최진우. (1997). 『중소형 공동주택 지하주차장 환기기법 향상 방안연구 - 환기방식 실태조사를 중심으로 -』. 대한주택공사 주택연구소.