

Searching based path planning

- Dijkstra

- Graph traversal and path search algorithm

- 시작점에서 목표점까지 최소의 cost 를 갖는 optimal 한 경로를 얻는데 사용됨

- Search space

- 다양한 search space 에서 문제 구성
 - Figure 1. → Dijkstra 사용, (x, y) space 에서 search (정해진 goal 의 x, y 좌표로 이동하는 경로 생성)
 - Figure 2. → A* 사용, (t, s, l) space 에서 search (정해진 시간 t 까지 종방향 s 및 횡방향 l 경로 생성)
 - Figure 3. → Hybrid A* 사용, (x, y, θ) space 에서 search (정해진 goal 의 $x, y, \text{heading}$ 에 도달하는 경로 생성) 가능

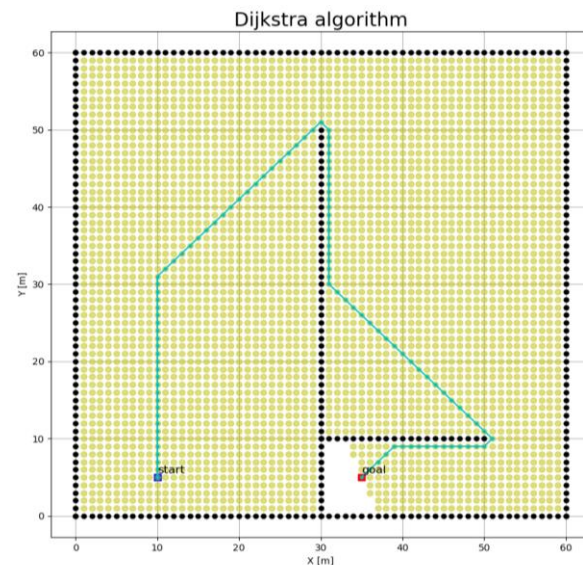


Figure 1. 2D configuration space (x, y)

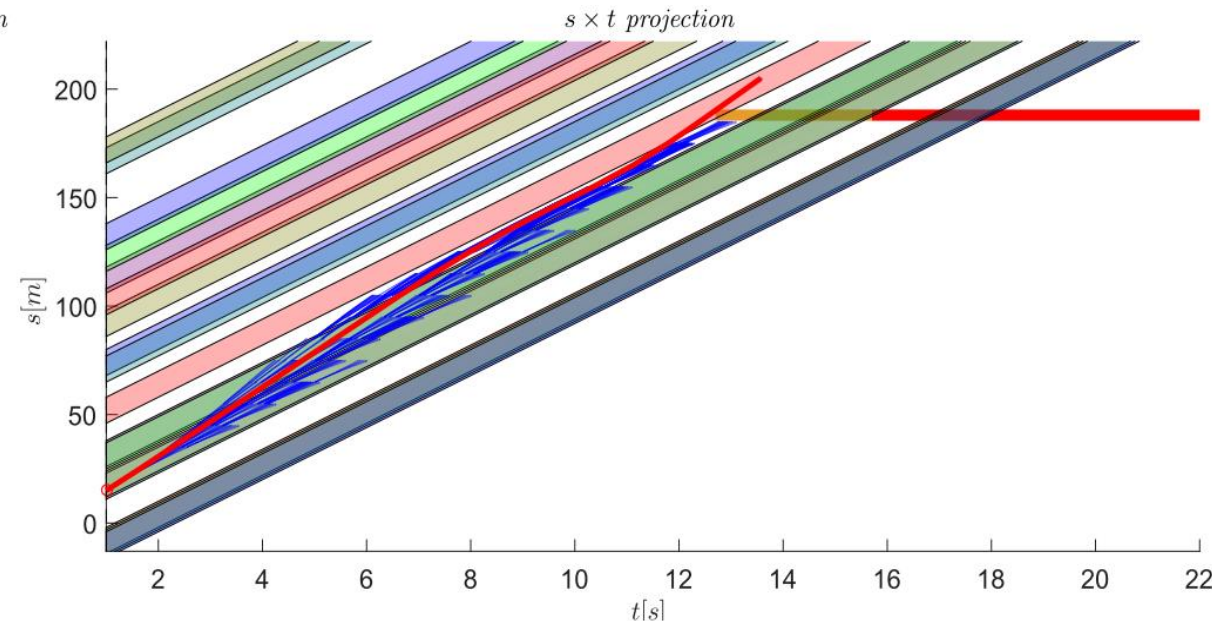
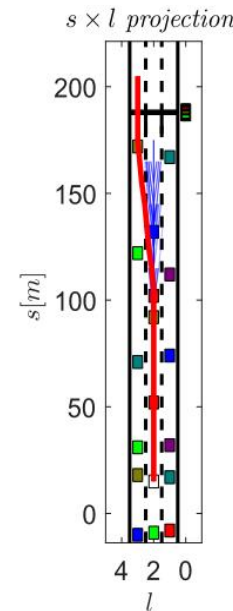


Figure 2. 3D configuration space (t, s, l)

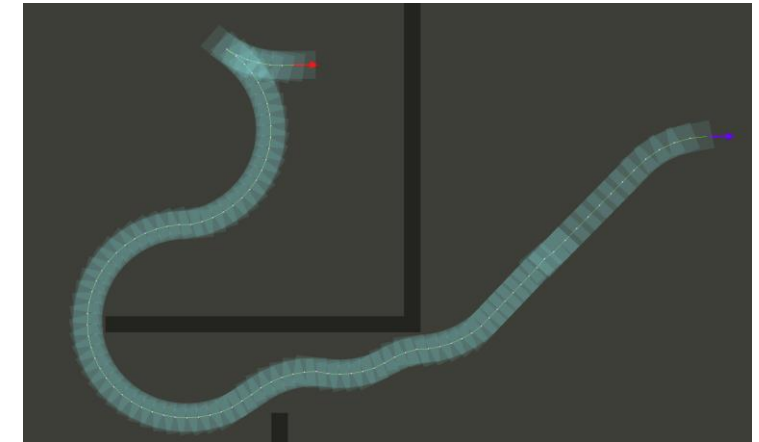


Figure 3. 3D configuration space (x, y, θ)

Refs : https://github.com/karlkurzer/path_planner

Searching based path planning

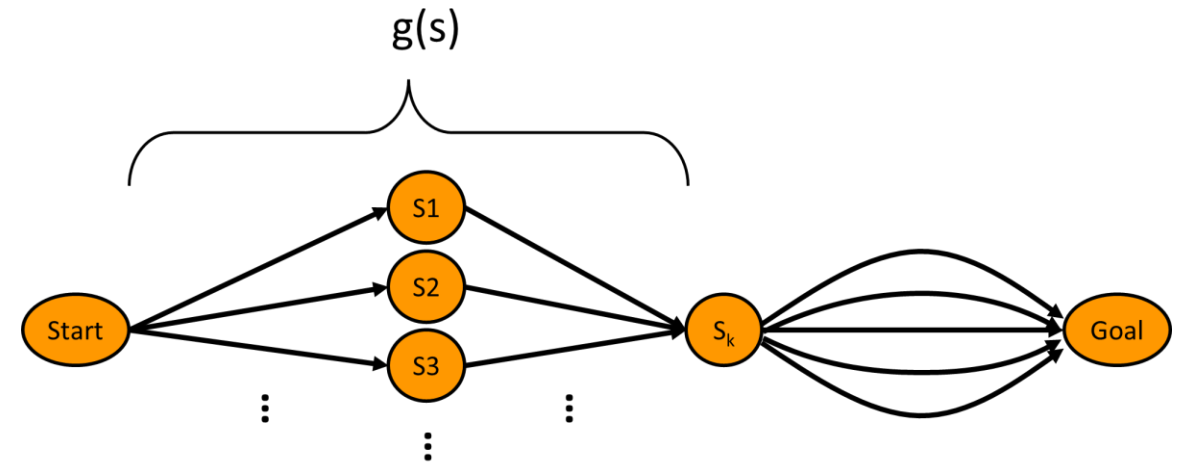
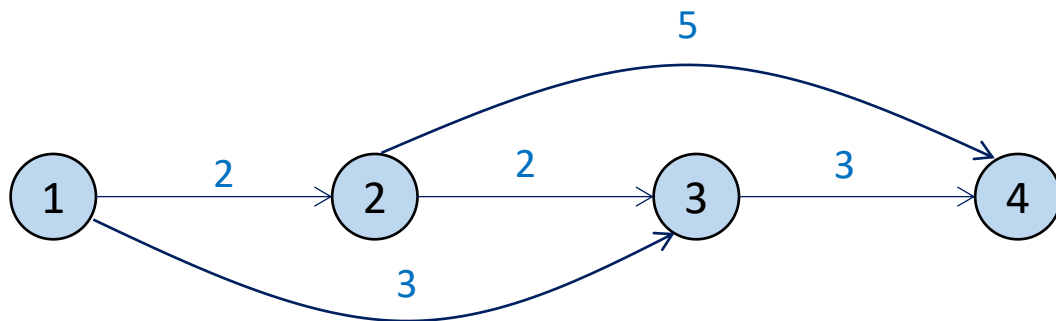
• Dijkstra

• Concept

- 최적 경로는 각각 최적 경로들로 이루어져 있음
- Cost, $f = g$ 에 따라 state expand.
- g : the cost of a shortest path from start state to current state
- 다소 비효율적

• Algorithm sequence

- ① 방문(고려)하지 않은 노드 중 최소 cost의 노드 방문
- ② 해당 노드를 통해 각 노드의 Cost update
- ③ Goal 도착(최소의 Cost를 가짐)이 아니라면 ①단계로(반복)




iteration	1	2	3	4
현재 노드	1			
미방문 노드 (cost)	[2, 3, 4] [2, 3, inf]			
방문 노드	-			

Searching based path planning

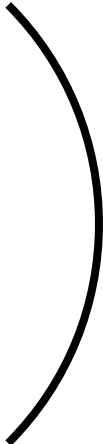
- Dijkstra

- Concept(2D)

- 2D grid map 에서 문제 구성(discrete searching space)
 - Start state 로부터의 shortest path cost($f=g$) 만으로 searching
 - 즉, start state를 중심으로 하는 원을 넓혀가며 goal 에 다다를 때 까지 탐색



Start



Obstacle



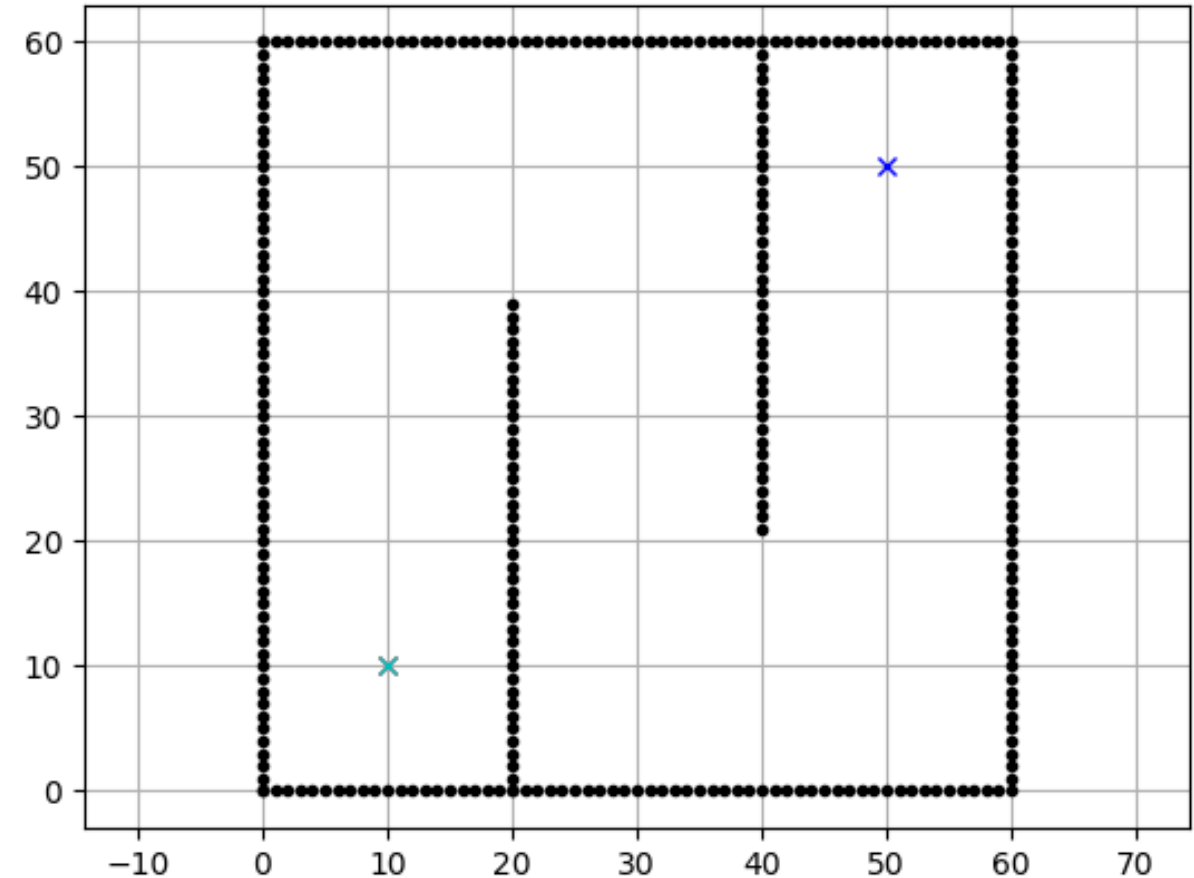
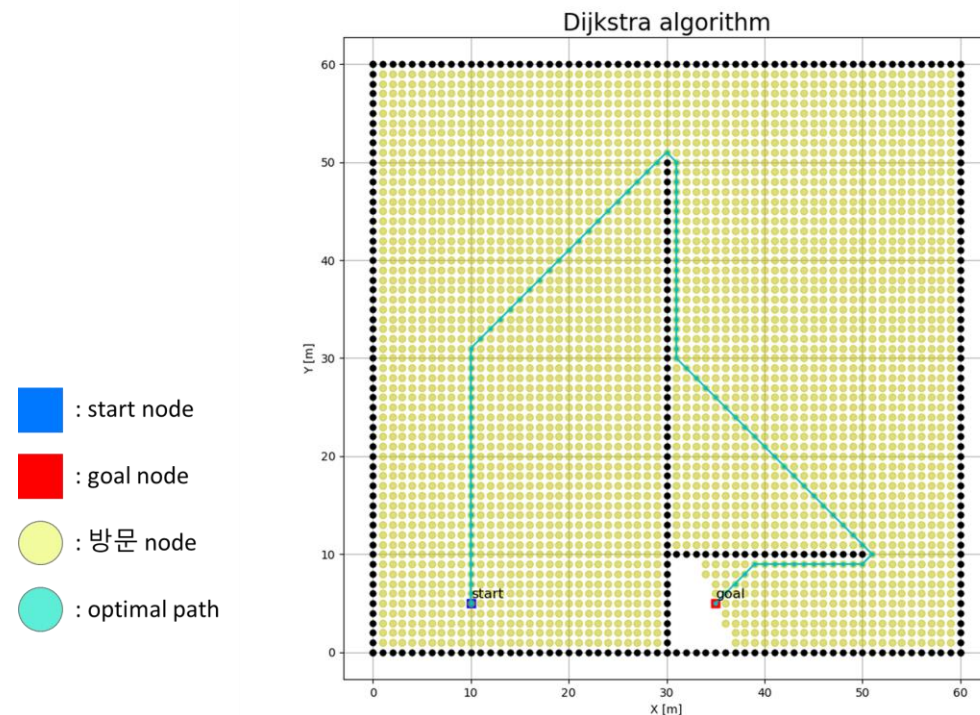
Goal

Searching based path planning

- Dijkstra

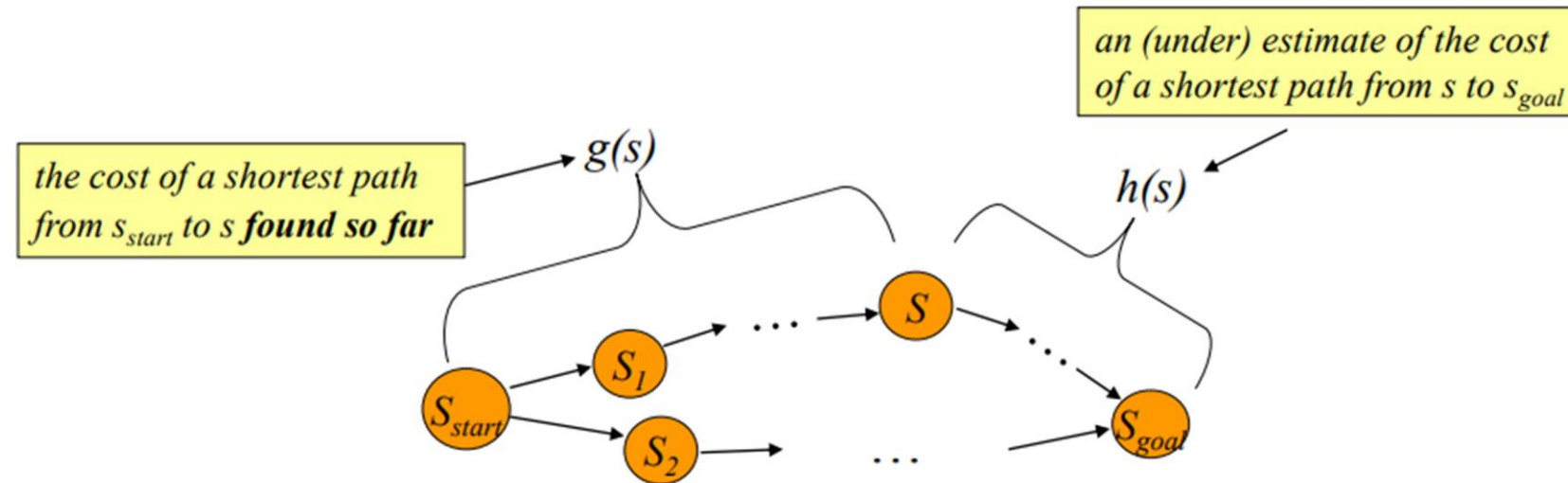
- Dijkstra result

- Dijkstra 알고리즘을 통한 optimal 경로 생성
 - cost (직선 : 1, 대각선 : $\sqrt{2}$)
 - Result → cost : 122.1, 방문 node : 3371



Searching based path planning

- A star algorithm
 - Concept
 - Cost, $f = g + h$ 에 따라 state expand.
 - g : the cost of a shortest path from start state to current state
 - h : heuristic cost, Euclidean distance from current state to goal state

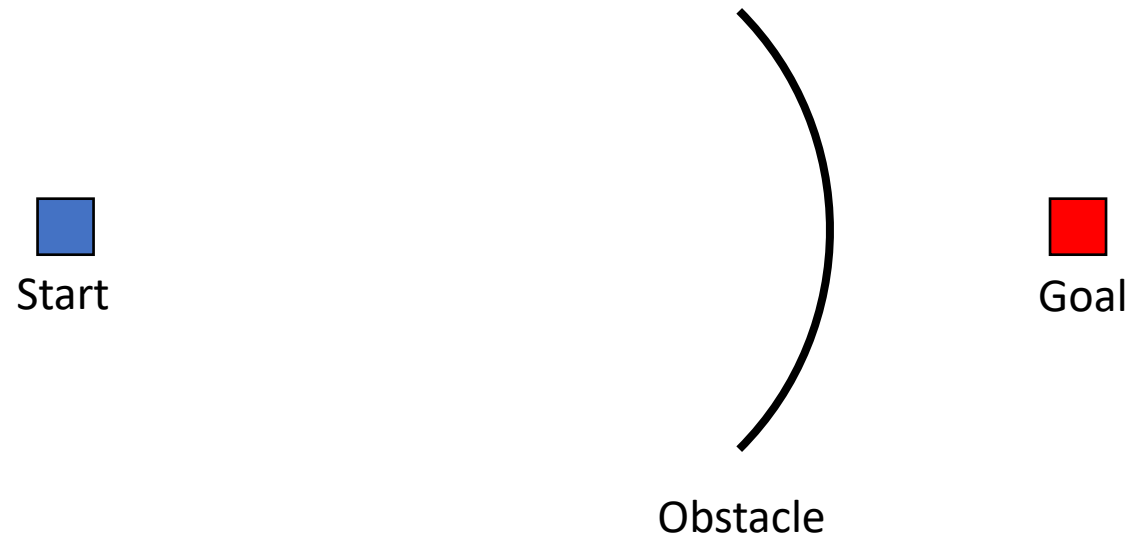


Searching based path planning

- A star algorithm

- Concept

- start state로부터 expansion 시작
 - $f = g + h$ 의 cost 로 searching
 - 따라서 start state 에서 goal state 로 향하는 (heuristic 으로 인해) 방향으로 expansion 하며, goal state 에 다다르는 경로가 나올때까지 searching 진행



A diagram illustrating path planning. On the left is a blue square labeled 'Start'. On the right is a red square labeled 'Goal'. Between them is a black curved line representing an 'Obstacle'. The labels 'Start', 'Obstacle', and 'Goal' are positioned below their respective symbols.

Start

Obstacle


Goal


Searching based path planning


- Weighted A star algorithm

- Concept

- $f = g + \varepsilon h$ 에 따라 state expand (h : heuristic)
 - g : the cost of a shortest path from start state to current state
 - h (heuristic cost) : Euclidean distance from current state to goal state
 - ε (weight) : bias towards states that are closer to goal state ($\varepsilon > 1$)
 - ε -suboptimal : $\text{cost}(\text{solution}) \leq \varepsilon \text{ cost}(\text{optimal solution})$
 - heuristic 의 bias ε 가 1 이상이므로 searching 시 goal state 로 향하는 방향으로 진행
 - cost 측면에서 solution 의 quality 가 낮아지지만 searching space 가 작아져 speed 향상


Start

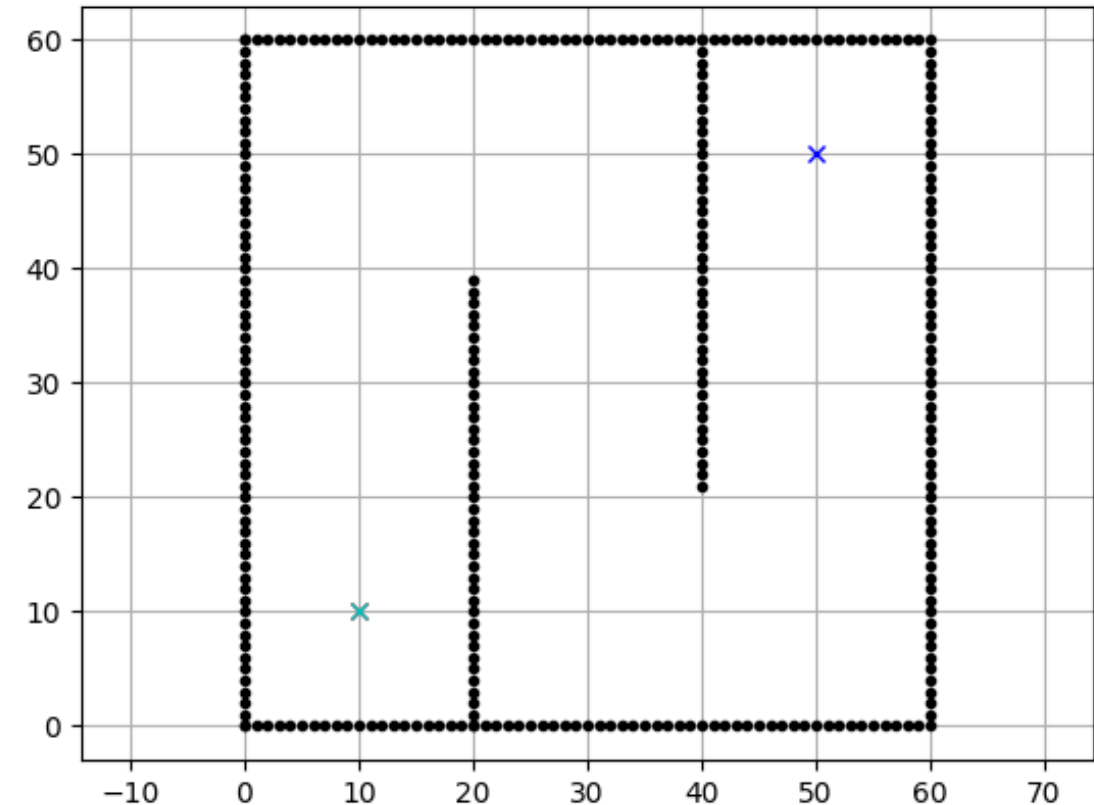
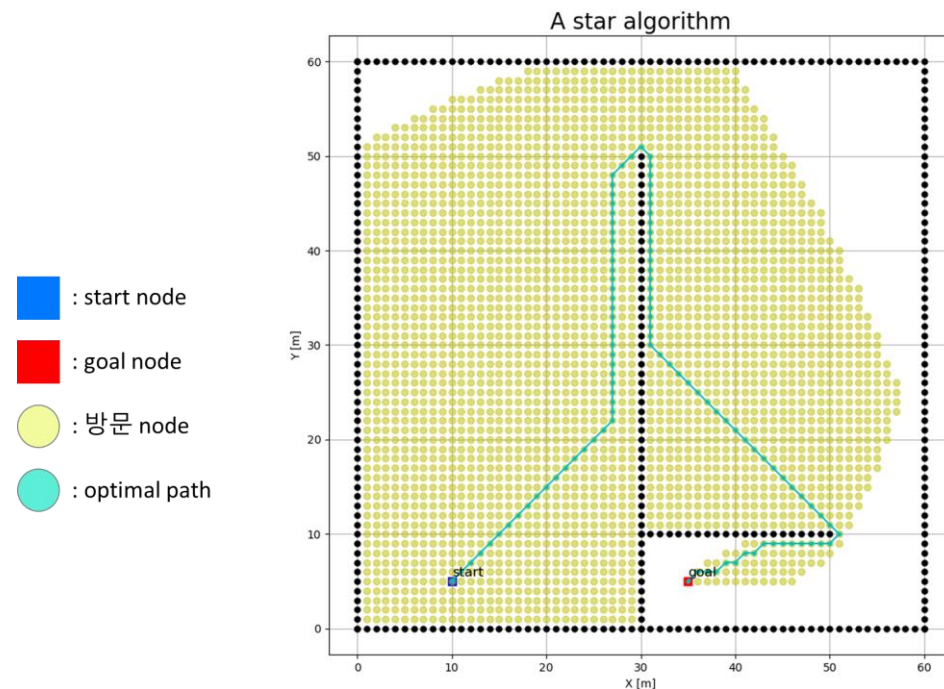

Goal


Obstacle

Searching based path planning

- A*/Weighted A* result

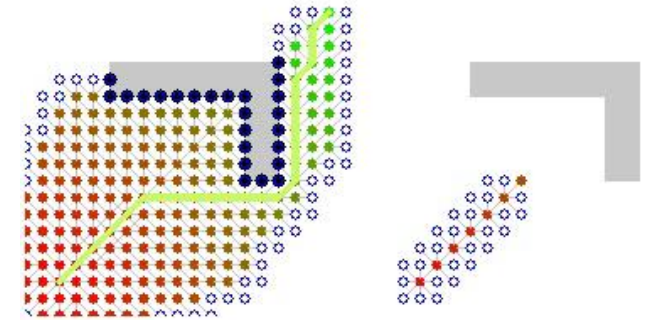
- A* 알고리즘을 통한 경로 생성
- Result → cost : 122.1, 방문 node : 2725
- Dijkstra 알고리즘(cost : 122.1, 방문 node : 3371) 과 cost 동일, 방문 node 는 적음 → optimal, speed up



Searching based path planning

- A*/Weighted A* result

- Weighted A* 알고리즘을 통한 경로 생성
- Figure 1. 은 weight (ϵ) 1.2 로 설정한 결과 \rightarrow cost : 142.6, 방문 node : 2331
- Figure 2. 는 weight (ϵ) 5 로 설정한 결과 \rightarrow cost : 141.4, 방문 node : 1602
- 기존 A* (cost : 122.1, 방문 node : 2725) 보다 cost 는 높지만 방문 node 는 줄어듬 \rightarrow trade off optimality for speed



A-star (left) vs. Weighted A-star with $\epsilon=4$ (right)

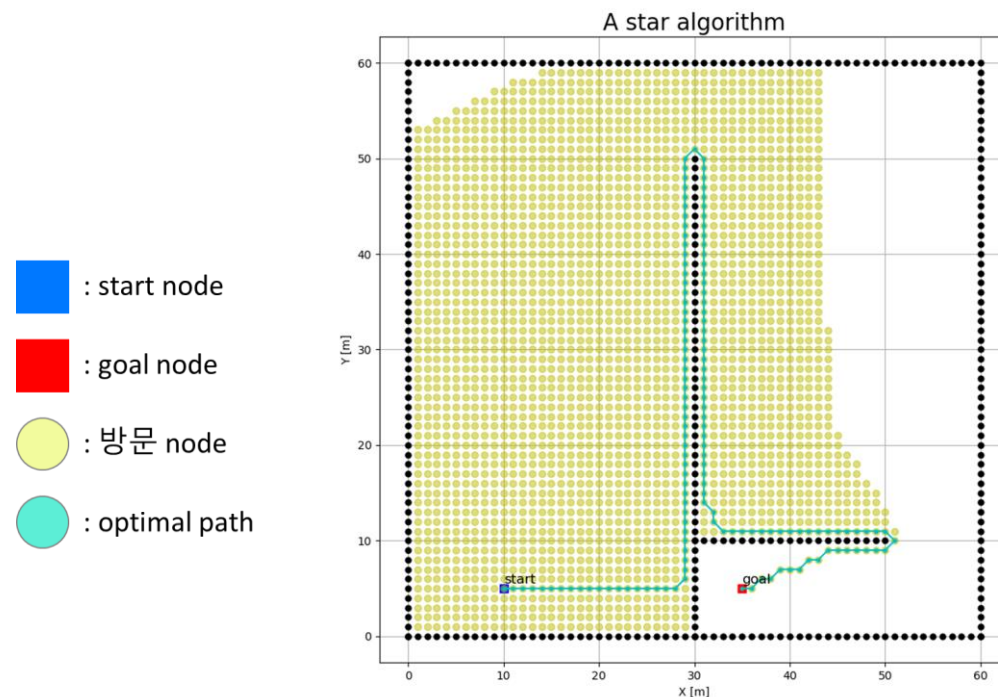


Figure 1. $\epsilon = 1.2$

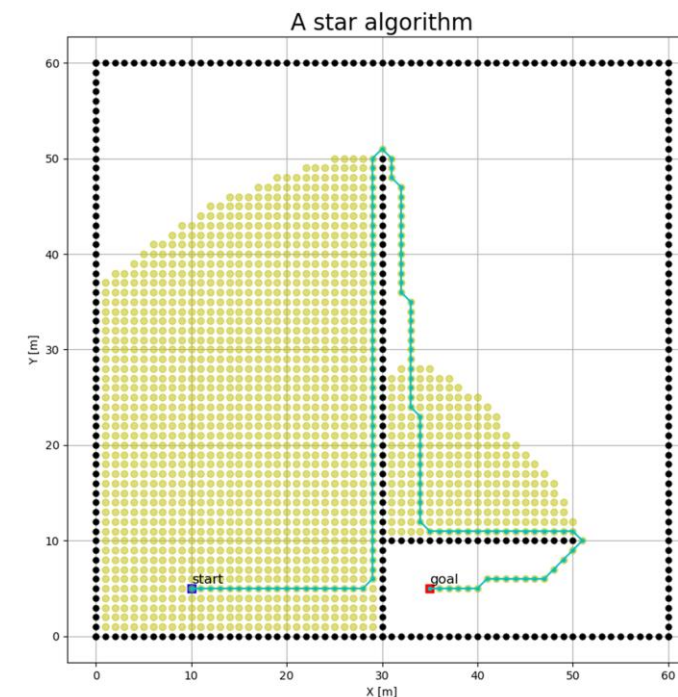


Figure 2. $\epsilon = 5$

Searching based path planning

- Hybrid A star algorithm

- Concept

- A* 알고리즘의 단점 : discrete search node 의 결과로 Figure 1 과 같이 차량이 주행하기 어려운 경로형태
 - Control action in A* : 주위 state 로 직선 형태의 expansion
 - Hybrid A* 를 통해 continuous 한 state 로 expansion 시킴
 - Control action in hybrid A* : discretized steering 을 입력으로 주어 차량이 주행가능한 경로를 생성

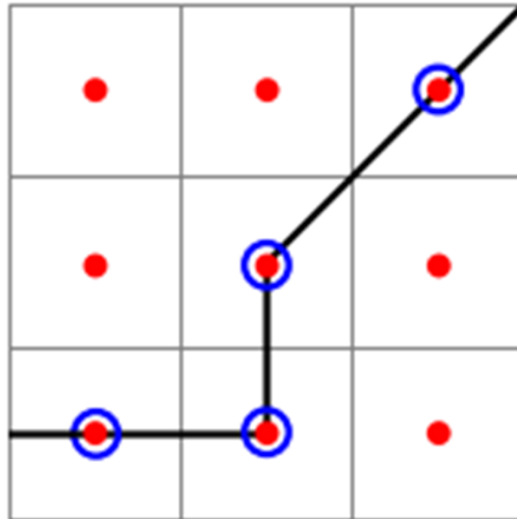


Figure 1. A*

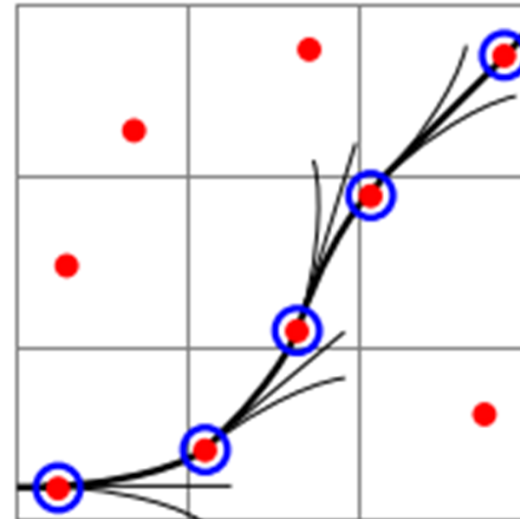
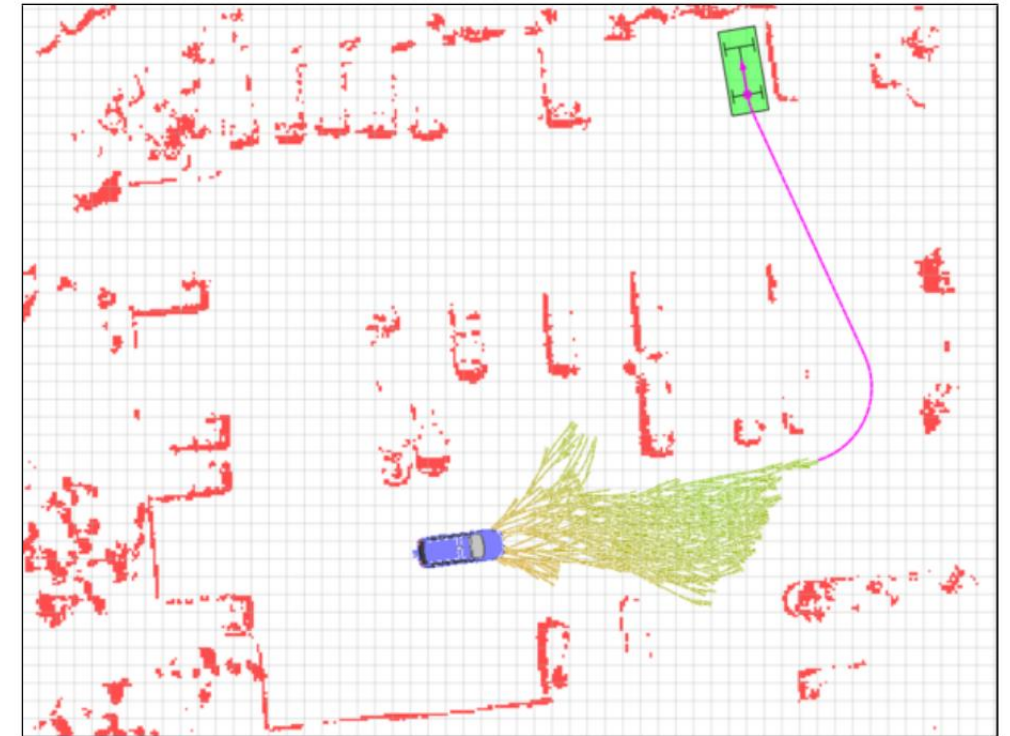


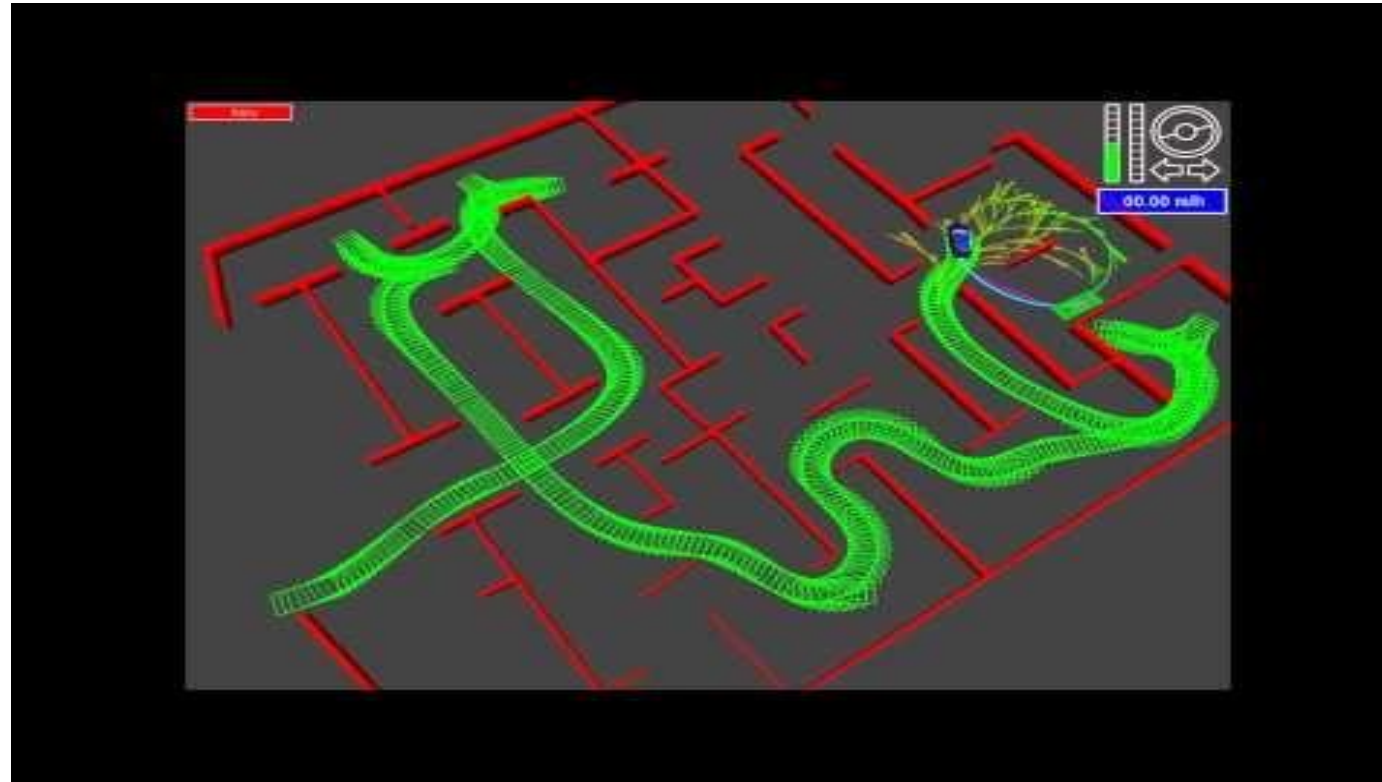
Figure 2. Hybrid A*



Searching based path planning

- Hybrid A star algorithm
 - Example

<https://youtu.be/qXZt-B7iUyw?si=62HCPXLkauvwAZ1o>



Thank You

