

Faculty of computers and Artificial intelligence

Cairo University

Faculty of computers and artificial intelligence – Cairo university

**AI313** - Autonomous Multiagent Systems

April 2023

Term project ( **Autonomous car parking** simulator ) - **Phase 2** Document

Section **3AI-S4 ( 12:45 Wednesday )**

TA. **Eng. Toqa.**

**1. Team members :**

|  |  |  |
| --- | --- | --- |
| Name | ID | Section |
| Bemwa Malak ( Team Leader ) | 20200116 | S4 |
| Kerolos George | 20200399 | S4 |
| Kevin Marco | 20200401 | S4 |
| Maher Mohsen | 20200415 | S4 |
| David George | 20201059 | S4 |

**2. Table of Contents :**

* **1. Problem definition** ( statement formalization )
  + 1.1 Selected problem Summary
  + 1.2 Formalization
  + 1.3 I/O and Processing
* **2. A\* search Algorithm** ( description , assumptions )
  + 2.1. A\* Description
  + 2.2. g(n) , past knowledge function
  + 2.3. h(n) , heuristic function
  + 2.4. Does A star is the best for this task / project ?
* **3. Apply A\* on demo version** ( state diagram / tree )

**3.1. Problem definition** ( statement formalization )

**3.1.1 Selected problem Summary :**

* we defined our problem “autonomous Car parking” .
* we have our agent “Car”.
* environment “Garage/Parking”.
* our target is to find a free parking slot in “minimum number of steps” and without hitting any obstacle.
* the smart garage provides agent with “the positions of free slots” at the initial state.
* guaranteed to have no trapped ( unreachable ) free parking slots.
  + 1. **Formalization :**
    - We define our set of actions [ ‘N’ , ‘S’ , ‘E’ , ‘W’ , ‘Park’ ].
    - We define our environment as 2D grid.
    - Each cell represents specific type [ ‘Road’ , ‘Obstacle’ , ‘Parking Slot’ ] , encoded as [ 0 , -1 , 1 ]
    - Obstacles generally defined as slots cannot be either visited or parked in.
    1. **I/O and Processing :**
* Input :
  + grid dimensions ( n x m )
  + grid state / cell values ( N.B : usually randomized )
  + initial state / position
  + all possible goal state(s) / position(s) ( N.B : review 3.1.1 )
* Processing :
  + Apply Algorithm to reach goal ( N.B : next section )
* Output :
  + Parking state [ ‘Successful’ , ‘Unsuccessful’ ]
  + Number of steps

**3.2. A\* search Algorithm** ( description , assumptions )

**3.2.1. A\* Description :**

* The A\* search algorithm builds on the principles of Dijkstra’s shortest path algorithm to provide a faster solution when faced with the problem of finding the shortest path between two nodes.
* It achieves this by introducing a heuristic element to help decide the next node to consider as it moves along the path.
* We define instance of A\* algorithm , we have multiple possible targets (Goal states ), once we reach any of them the goal is “Satisfied”.
* F(n) = g(n) + h(n)

**3.2.2. g(n) , past knowledge function :**

* Defined as the cost of the path from the starting node / initial state until current node / position.
* g(child) = g(parent) + 1

**3.2.3. h(n) , heuristic function :**

* Defined as the estimate cost from current state / node to goal state.
* Cost in our problem relates to path distance.
* Since cost is distance , we cannot move in diagonal steps, so we choose our heuristic as “Manhattan distance”.
* Manhattan distance defined as m( p1 , p2 ) = abs( p1.x – p2.x ) + abs( p1.y – p2.y ) , where p1 , p2 are coordinate grid cells.
* N.B avoid overestimate heuristic values.

**3.3. Does A star is the best for this task / project ?**

* For the **standard version of A\* star** algorithm we define only single goal state , and have heuristic for this goal only , but in our problem, we have multiple possible candidate goals ( all free slots exist in garage/parking ), so the standard A\* isn’t a good choice for our problem.
* In case we define variation of this standard A\* , which is **A\* with multiple candidate goals ,** we can solve our problem and reach nearest possible goal in minimum number of steps but this puts complexity factor into algorithm core which is the **linear time function z(x)** that goes for each successor and search for minimum heuristic value for this successor with all possible candidate goals / solutions so if we assume we have n possible goal we add o(n) complexity to each expansion step in the algorithm , so this solution is good but as the candidates increases the performance also decreases.
* Once our parking slots ( candidate goals ) increases for large cases / large garages A\* will be not the best solutions , we may think about other algorithms as NNS ( Nearest neighbor search algorithm ) , modified flood fill and more.

**3.4. Apply A\* on demo version ( state diagram / tree )**

A screenshot of a game

Description automatically generatedA screenshot of a game

Description automatically generatedA\*( , ) =

A screenshot of a game

Description automatically generated with medium confidence

A picture containing diagram, screenshot, plan, design

Description automatically generated