Daniel I. Núñez López A01654137
Carlos Mateos Pérez A01654085

Planeación de rutas para la exploración en Marte

Código de Python

Superficies.py > ...

```python
import numpy as np
mars_map = np.load('mars_map.npy')
nr, nc = mars_map.shape
from simpleai.search import SearchProblem, astar, depth_first, breadth_first, greedy
class ProblemaMarte(SearchProblem):

    def actions(self, state):
        x = state[0]
        y = state[1]
        step = state[4]
        height = state[5]

        lst = [(-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0), (1, 1)]
        actions = []

        for i in range(len(lst)):
            a, b = lst[i]
            if abs(mars_map[x+a,y+b] - height) <= step and mars_map[x+a][y+b] != -1:
                actions.append((x+a, y+b))

        return actions

    def result(self, state, action):
        newx = action[0]
        newy = action[1]
        return newx, newy, state[2], state[3], state[4], mars_map[newx, newy]

    def is_goal(self, state):
        currx = state[0]
        curry = state[1]
        goalx = state[2]
        goaly = state[3]

        return goalx == currx and goaly == curry

    def cost(self,state, action, state2):
        return 1

    def heuristic(self, state):
        currx = state[0]
        curry = state[1]
        goalx = state[2]
        goaly = state[3]
        return np.sqrt((currx-goalx)**2+(curry-goaly)**2)


x0 = nr - int(6400/10.0174) #y
y0 = int(2850/10.0174) #x
```

Daniel I. Núñez López A01654137
Carlos Mateos Pérez A01654085

```
49
50  xf = nr - int(6800/10.0174) #y
51  yf = int(3150/10.0174) #x
52
53  robot = ProblemaMarte(initial_state=(x0, y0, xf, yf, 0.25, mars_map[x0][y0]))
54
55  solution_astar = astar(robot, graph_search=True)
56  print()
57  print("A* (",len(solution_astar.path()),"):")
58  print()
59
60  solution_depth = depth_first(robot, graph_search=True)
61  print()
62  print("Cantidad de pasos para el algoritmo depth (",len(solution_depth.path()),"):")
63  print()
64
65  solution_greedy = greedy(robot, graph_search=True)
66  print()
67  print("Cantidad de pasos para el algoritmo greedy (",len(solution_greedy.path()),"):")
68  print()
69
70  solution_breadth = breadth_first(robot, graph_search=True)
71  print()
72  print("Cantidad de pasos para el algoritmo breadth (",len(solution_breadth.path()),"):")
73  print()
74
75  # <= 500 metros
76
77  x0 = nr - int(6845/10.0174) #y
78  y0 = int(4683/10.0174) #x
79
80  xf = nr - int(6445/10.0174) #y
81  yf = int(4672/10.0174) #x
82
83  robot = ProblemaMarte(initial_state=(x0, y0, xf, yf, 0.25, mars_map[x0][y0]))
84
85  solution_astar = astar(robot, graph_search=True)
86  print()
87  print("A* (",len(solution_astar.path()),"):")
88  print()
89
90  # 1000 metros < x < 5000 metros
91
92  x0 = nr - int(14613/10.0174) #y
93  y0 = int(2346/10.0174) #x
94
```

```
95   xf = nr - int(13487/10.0174) #y
96   yf = int(2431/10.0174) #x
97
98   robot = ProblemaMarte(initial_state=(x0, y0, xf, yf, 0.25, mars_map[x0][y0]))
99
100  solution_astar = astar(robot, graph_search=True)
101  print()
102  print("A* (",len(solution_astar.path()),"):")
103  print()
104
105  # >= 10000 metros
106
107  x0 = nr - int(14386/10.0174) #y
108  y0 = int(2384/10.0174) #x
109
110  xf = nr - int(1326/10.0174) #y
111  yf = int(2647/10.0174) #x
112
113  robot = ProblemaMarte(initial_state=(x0, y0, xf, yf, 0.25, mars_map[x0][y0]))
114
115  solution_astar = astar(robot, graph_search=True)
116  print()
117  print("A* (",len(solution_astar.path()),"):")
118  print()
```

Prueba de algoritmos de búsqueda

- ¿Qué algoritmos lograron encontrar una ruta válida?

Técnicamente todos los algoritmos lograron encontrar una ruta válida, en el caso de los algoritmos no informados (breadth y depth first), el tiempo en el que encuentran la ruta es muy extenso, a diferencia de los algoritmos informados, que son mucho más rápidos.

- ¿Es necesario utilizar búsquedas informadas para este caso?

No es necesario, pero son las más eficientes, ya que no se extienden por todas las posibilidades para llegar a una ruta, sino que utilizan los costos para determinar qué ruta es la mejor entre cada nodo.

- ¿Qué función heurística resultó adecuada para este problema?

Daniel I. Núñez López A01654137
Carlos Mateos Pérez A01654085

Tanto greedy como astar fueron buenas funciones adecuadas para resolver el problema, sin embargo, como se trata de una misión de exploración y el tiempo no es necesariamente un factor a considerar por la duración de las misiones, greedy es el algoritmo más rápido que encontró una ruta, puede no ser la óptima, pero si lo que se busca es rapidez entre una gran cantidad de datos, este gana.

Rendimiento de los algoritmos de búsqueda para rutas cortas y largas

- ¿En qué casos el algoritmo es capaz de resolver el problema en un tiempo aceptable?

En las rutas cortas los algoritmos son buenos, sin embargo, cuando las rutas se alargan, se tarda mucho más en correr el programa y en encontrar una ruta, ya que son muchos los datos.

- En los casos que el algoritmo no encuentra un resultado, ¿qué acciones se podrían realizar para ayudar al algoritmo a resolver el problema?

Intentar mejorar la heurística o recortar la cantidad de datos que se tiene en el mapa, igualmente, el no usar algoritmos de búsqueda no informada, es de utilidad. Otra opción podría ser mejorar el robot para que tenga menos dificultades al moverse en la superficie debido a las diferencias de alturas.

Conclusión personal Daniel:

Durante esta entrega tuvimos que aprender a ver el problema desde distintas perspectivas y fuera de la caja para poder resolverlo, primero empezamos por descargar la imagen de la superficie, la cual no es nada ligera, esto por la cantidad de información que se tiene en ella, luego, para poderla usar la tuvimos que convertir en un formato que pudiera ser de utilidad para nuestro objetivo, en el que creamos una matriz que contuviera las coordenadas y su altura asociada, esto para poder utilizar las coordenadas y una vez teniéndose, poder acceder a las alturas y comprobar las diferencias de las alturas, y si ellas son menores al requerimiento, poderse mover en esa dirección. Para lo anterior, utilizamos un método en el que se revisan los 8 píxeles que están alrededor del robot por medio de coordenadas y luego acceder a la altura.

Daniel I. Núñez López A01654137
Carlos Mateos Pérez A01654085

En este caso, utilizamos algoritmos tanto de búsqueda informada como no informada, en lo personal creo que los algoritmos de búsqueda son los mejores porque no revisan todas las opciones posibles exhaustivamente, al contrario, se basan en un mecanismo de heurística que les permite ir mejorando sin tener que revisar cada opción una por una. Al tratarse de un problema de recorrer superficies, y al contar con una gran cantidad de datos con los cuales se puede tomar una decisión, creo que es más fácil si utilizamos astar, porque además de querer recorrer la ruta en el menor tiempo posible, se busca optimizar los materiales de las ruedas del rover, lo cual es un problema que enfrentan los rovers en Marte, ya que el material debe ser lo suficientemente poroso para tener un buen agarre en la superficie, pero al ser un clima árido, el material tiende a deteriorarse con el tiempo, por lo que este es crucial para lograr completar la mayor cantidad de objetivos posibles. Dentro de las cosas más retadoras que tuvimos que hacer en esta entrega, fue idear el código en el que obtenemos las coordenadas, igualmente, la heurística y definir actions, resultados, y goal. Para los algoritmos de búsqueda no informados es fácil hacer el cambio entre uno y otro, ya que requieren los mismos "atributos" esto a diferencia de los algoritmos informados, que requieren adicionalmente la heurística para funcionar, hablando de esto, fue un problema más, ya que depende de la computadora que tengas para resolver el problema en menos tiempo, en el caso de los no informados, ya que se revisan todas las posibles opciones a recorrer antes de seleccionar una, lo cual toma demasiados recursos, esta es una razón más por la que los algoritmos como astar y greedy son mejores. Sobre la actividad de rutas cortas y rutas largas, se puede ver más la diferencia entre los dos tipos de algoritmos, ya que son muchos más datos los que se tienen que recorrer; creo que si se pudiera delimitar el área de búsqueda, sería más rápido aún llegar al objetivo, ya que serían menos los datos a recorrer.

Conclusión personal Carlos:

Durante la realización del proyecto, se encontraron múltiples dificultades que llegaron a frenar las resolución del proyecto, dicha explicación se dividirá en dos fases, en la primera se explicarán las dificultades a la hora de plantear el problema, en la segunda fase, se explicarán las dificultades a la hora de realizar el código y correrlo. Para empezar, a la hora de leer el problema, me resultó complicado entender las simulaciones y las gamas de colores, además de los obstáculos que se presentan en la superficie, ya que al ser de pequeña escala y distintas texturas

resultaba complicado analizarlo, esto se resolvió investigando problemas similares y tratando de ponerlo en contexto con el problema planteado. Este problema resulta muy interesante debido a que se debe comprender que se quiere que el robot camine sobre una plataforma con obstáculos y que probablemente se encuentre en puntos máximos locales, y se deberá de encontrar la manera en que no se atore y siga caminando hasta explorar la superficie correspondiente. Al momento de realizar el análisis para resolver el problema, me pareció adecuado y necesario realizar un análisis del ambiente del agente, el cuál quedó de la siguiente manera, no observable, ya que debido a su restricción sólo se puede alrededor de dónde se encuentre, es estocástico, ya que al ser parcialmente observable, el cambio constante del ambiente, el cambio constante de las acciones del agente, además de otros factores, indican que será cuestión de suerte que el agente pueda encontrar datos precisos y no se quede atorado al momento de encontrar mínimos o máximos locales, además de depender de no caerse y que el robot pueda sufrir daños, dinámico, el ambiente va cambiando mientras el agente piensa que hacer o qué acción realizar, no se sabe con que se puede encontrar en la superficie, al estar en el espacio, el ambiente va cambiando continuamente, puede ocurrir algo en la superficie o que encuentre algún obstáculo en el camino que impida que pueda desplazarse libremente por la superficie, continuo, las acciones que se realizan no pueden ser enumeradas, se va analizando sobre una superficie que contiene máximos, mínimos y algún obstáculo que se pueda encontrar, tiene que caminar, buscar, analizar, al realizar este análisis se facilitó resolver el problema en papel para entrar en contexto con el agente, pensar como él y generar ideas de cómo se podría comportar. Se concluyó en el planteamiento inicial, que el agente debería ser capaz de desplazarse sin importar la altura en la que se encuentra, no atorarse en mínimos o máximos locales y ser capaz de saltar o rodear obstáculos para llegar al objetivo deseado. Al tener el problema planteado, se procedió a la segunda fase, la de realizar el código, dónde se encontraron algunos problemas que nos pusieron a pensar, el hecho de que haya algunas pendientes grandes hacían pensar al agente que eran máximos o mínimos y se tuvo que investigar algunas otras técnicas que ayudarán al agente a seguir con su recorrido, evitando estos problemas, además de los obstáculos para llegar a su objetivo.