Instituto Tecnológico y de Estudios Superiores de Monterrey

Diseño de Agentes Inteligentes

Situación problema : Programming Mars 2020

Fernando Guevara Moreno | A00828723
Ian Timothy Henry Suárez | A01701578

Profesor:

Dr. Leonardo Garrido

Marzo 10, 2021

Parte 1: Ambiente Marciano

1 Sensores

La misión Mars 2020 tiene como objetivo enviar un robot inteligente al planeta rojo, el cual tiene como objetivo la exploración del planeta, así como también la recolección de muestras para su futura experimentación en la tierra. Este robot tiene como nombre Perseverance, y el 18 de febrero de 2021, aterrizó exitosamente, siendo una de las misiones más prometedoras y ambiciosas de la agencia espacial.

Ahora bien, para que un robot de esta índole pueda llevar a cabo tareas tan exigentes y arduas como la exploración, recolección, visualización y comunicación en un planeta distante, debe contar con la ayuda de distintos sensores que sean capaces de realizar estas actividades.

De acuerdo con el portal oficial de la NASA, el Perseverance cuenta con 7instrumentos que recabarán información atmosférica y geológica del planeta rojo, además de buscar signos de vida en el pasado así como las condiciones ambientales. Los instrumentos son:

- Mastcam - Z: Cámara panorámica y estereoscópica capaz de hacer zoom. Tiene como objetivo determinar la mineralogía de la superficie marciana.
- MEDA: (Mars Environmental Dynamic Analyzer): Sensores que proporcionan medidas de temperatura, viento, así como su velocidad y dirección, presión, humedad y polvo.
- MOXIE: (Mars Oxygen ISRU Experiment): Artefacto capaz de producir oxígeno a partir del dióxido de carbono marciano.
- PIXL: Espectrómetro capaz de proporcionar detalladamente un análisis de los elementos químicos sobre la superficie del planeta rojo.
- RIMAX: Taladro con el que se penetrara la superficie con el objetivo de obtener pequeñas muestras de la tierra y su composición química.
- SHERLOC: Este usa un láser ultravioleta que determina la mineralogía y puede detectar compuestos orgánicos.
- SuperCam: Principalmente proporciona imágenes de alta calidad, al mismo tiempo es capaz de detectar presencia orgánica en las rocas a una distancia relativamente amplia

Consideramos que uno de los principales y más importantes sensores del robot es MastCam - Z ya que este le permite ver el mundo a su alrededor, proporcionando una imagen limpia y amplia del lugar en el que se encuentra, de esta manera será capaz de buscar los mejores resultados así como la exploración eficiente de la superficie marciana.

2 Actuadores

Sabemos de antemano que un actuador es un dispositivo que permite al robot moverse, ya sea rotativa o linealmente. O bien es capaz de interactuar con el entorno. Primordialmente, el robot debe contar con un actuador que le permita realizar movimientos libremente, un ejemplo claro podría ser los neumáticos, o si es un robot humanoide como Atlas, un par de extremidades. Ahora bien, una vez que puede moverse, también requiere “sentir” el entorno por lo que un brazo robótico suele ser uno de los actuadores más utilizados en los robots exploradores.

3 PEAS

PEAS				
Tipo	Medida de desempeño	Entorno	Actuadores	Sensores
Robot Explorador	Obtener información del entorno, permanecer con energía, cubrir más superficie, obtener muestras del suelo, soportar las condiciones ambientales.	Superficie de Marte; arena, rocas, montes, brechas, cráteres, viento, tormentas de polvo.	Neumaticos, taladro	Mastcam - Z, MEDA, MOXIE, PIXL, RIMAX, SHERLOC, SuperCam

Existen varios elementos que determinan si el robot actúa de forma racional. Para que un agente inteligente actúe racionalmente, necesita llevar a cabo acciones para maximizar su medida de rendimiento. El robot encuentra patrones y analiza la superficie del planeta, sus acciones giran alrededor de mejorar su medida de desempeño.

4 Entorno

Un agente inteligente como el robot Perseverance actúa en función de su entorno. Para que esto sea posible, el agente debe conocerlo y poder percibirlo. El entorno de un agente inteligente cuenta con una serie de características que varían entre distintos entornos.

En el caso del robot Perseverance, el entorno consiste en la superficie del planeta Marte. El robot debe tener noción sobre lo que pueda llegar a encontrar o suceder en su entorno para poder tomar una decisión adecuada. A continuación se muestra la caracterización del entorno del robot Perseverance.

La primera característica consiste en ser completamente o parcialmente observable. En el caso del ambiente del robot Perseverance, es parcialmente observable. Los sensores del agente no tienen la capacidad de visualizar toda la superficie del planeta. El robot lo percibe mientras lo recorre.

El entorno de un agente inteligente puede ser estocástico o determinista. El entorno del robot Perseverance es estocástico. El siguiente estado del entorno no depende de su estado actual o de las acciones del agente.

La siguiente característica del entorno de un agente inteligente es ser episódica o secuencial. En el caso del entorno del robot explorador, el entorno es secuencial. El robot no repite la misma acción. El siguiente movimiento del robot depende de sus acciones previas.

Existen entornos de agentes inteligentes dinámicos, semi dinámicos y estáticos. La superficie de Marte como entorno del agente es dinámica. El entorno puede cambiar mientras el agente procesa la información y determina su siguiente acción.

El entorno de un agente puede ser discreto o continuo. En el caso del entorno del robot Perseverance, el entorno es continuo. La superficie de Marte tiene un estado infinito de estados,

va cambiando constantemente. Tiene que ser analizada continuamente.

Un agente inteligente puede ser individual o multiagente. Si un agente es multiagente, puede ser competitivo o cooperativo. En el caso del robot Perseverance, es un agente multiagente y cooperativo. El robot cuenta con un segundo agente, un helicóptero pequeño llamado Ingenuity. Este helicóptero complementa la visibilidad de Perseverance y es autónomo. Colaboran para poder llevar a cabo su misión con éxito.

Por último, el entorno de un agente puede ser conocido o desconocido. El entorno del robot Perseverance es desconocido. El robot tiene que ir guardando información sobre el entorno mientras lo recorre. Va creando un mapa de su entorno. Los estados del entorno son desconocidos. Inicialmente el agente no cuenta con información previa de su entorno.

Parte 2: Programming Mars

5 Introducción

Mars 2020 es una de las misiones más ambiciosas y prometedoras que se han llevado a cabo en el planeta rojo, desde su primera misión Viking I, en 1976. La exploración e investigación de Marte siempre ha sido uno de los objetivos más interesantes y a su vez demandantes de la ingeniería espacial, ya que se deben tomar en cuenta una infinidad de variables, parámetros y métricas para que una misión de esta índole cumpla exitosamente con su meta. La misión Mars 2020 contribuirá en gran medida al desarrollo tecnológico y establecerá bases para futuras exploraciones. Se buscarán rastros en la superficie que demuestren que alguna vez existieron organismos biológicos. Esta misión toma como base muchos aspectos que ya se habían utilizado con anterioridad y al mismo tiempo, implementó nuevos elementos, estrategias, y modelos para estudiar exitosamente el suelo y superficie del planeta rojo.

La exploración de Marte es un desafío científico y tecnológico, el cual requiere un trabajo multidisciplinario e ingenieril. Sobre todo aquellas disciplinas computacionales que resultan sumamente importantes al momento de realizar misiones espaciales, ya que los algoritmos necesarios para el funcionamiento de un robot son sumamente complejos.

6 Descripción del problema

La inteligencia artificial juega un papel importante en esta misión ya que le permite a Perseverance, un robot autónomo que registra todo lo que lo rodea, tomar decisiones y así alcanzar el objetivo predeterminado, al mismo tiempo que maximiza su rendimiento. La primera etapa del proyecto consistió en caracterizar el entorno del robot explorador de Marte; la superficie del planeta. La siguiente fase del proyecto consistió en la implementación de de algoritmos de búsqueda para que el agente sea capaz de determinar la ruta más apropiada o la trayectoria óptima entre dos puntos en la superficie de Marte.

Uno de los principales desafíos en la exploración de Marte es la distancia entre este y la Tierra. La señal tarda mucho tiempo en ir y regresar de un planeta a otro, por lo que no es factible controlar un robot de forma remota desde la Tierra. Es necesario que el robot sea inteligente para poder desplazarse de manera autónoma por la superficie. El agente podrá percibir su entorno, procesar y analizar la información, y tomar decisiones. Para poder implementar un agente inteligente, es importante conocer y clasificar el entorno donde llevará a cabo sus

tareas. De esta forma podremos determinar qué algoritmos son apropiados para nuestro agente, y estará preparado para resolver los problemas que se le presenten. El entorno del agente inteligente es la superficie de Marte. El suelo de marte es arenoso, tiene piedras, brechas, cráteres y montes. Sus condiciones climáticas son variables, puede haber viento o tormentas de arena. Es importante tomar esto en consideración.

El objetivo de este proyecto consiste en implementar varios algoritmos de búsqueda que le permitan al robot explorador desplazarse por la superficie de Marte evitando obstáculos o desniveles. Para lograr esto debemos obtener información sobre la elevación del suelo a partir de una imagen satelital con la altura representada por un código de colores RGB.

Primeramente, debemos importar la imagen obtenida de la página HiRISE, la cual nos proporciona una biblioteca con imágenes del planeta marciano, las cuales fueron valorizadas de acuerdo a las respectivas alturas de la superficie con un método llamado "falso color". La imagen se muestra en la Fig. 1.

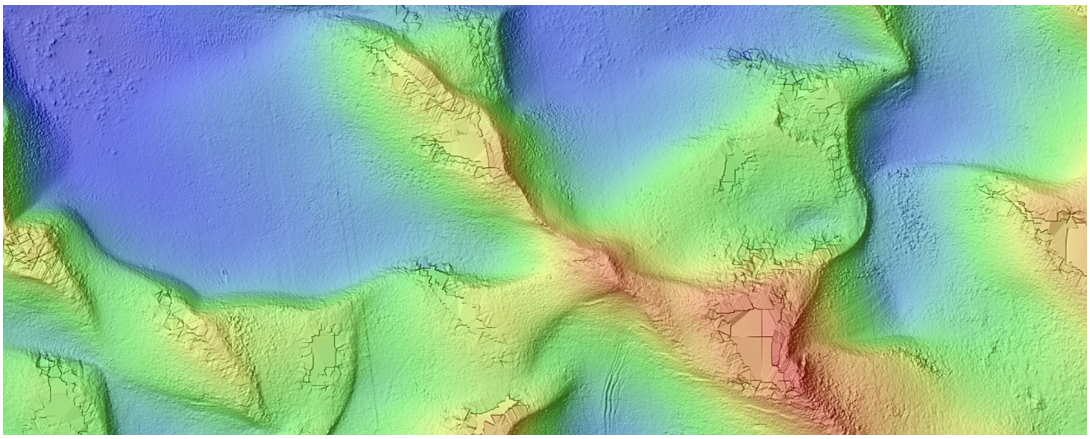

Figure 1: Gullies Monitoring

Una vez obtenida la imagen, y a través de la función ***.size*** podremos conocer las dimensiones de nuestra imagen. Siendo esta de tamaño de (1680, 666). Sin embargo, nos percatamos que es una cantidad muy grande de datos, los cuales complicaron y relentizaran el procesamiento de estos. Es por esto que decidimos recortar la imagen, ajustándose a la nueva medida de 400 x 400 píxeles como se muestra en la Fig. 2.

Con las siguientes líneas de código, obtenemos una porción de la imagen que nos permitirá localizar mejor nuestro camino, así como los métodos que serán utilizados para la detección de "paredes". Lo que hacemos es disminuir la calidad de la imagen para que tengamos únicamente una matriz de 40 x 40 pixeles.

```
# Cargamos la imagen
from PIL import Image
import numpy as np

image = Image.open('planeta_nuevo.jpg', 'r')
image2 = image.resize((40,40)) # Reducimos la imagen para poder visualizar los pixeles
width, height = image2.size # Obtenemos las dimensiones
indexed2 = np.array(image2) # Creamos un arreglo con las coodenadas y los valores RGB
#indexed = np.array(image)
```

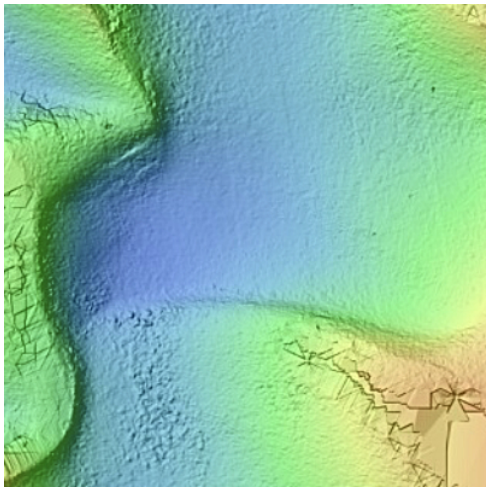

Figure 2: Gullies Monitoring Recortada

```

indexed2.shape
```

Los valores RGB son una escala de colores que nos permiten visualizar toda la paleta de colores existentes mediante una combinación de números que van desde 0 hasta 255. Por lo mismo que cada pixel cuenta con 3 valores diferentes, debemos buscar una manera de reducir estas variables a una, por lo que usamos la fórmula de la escala de grises.

$$Y = R \cdot 0.3 + G \cdot 0.59 + B \cdot 0.11$$

La siguiente parte de nuestro código nos permitirá hacer esta transacción:

```
# Transformamos el formato RGB a una escala de grises para tener un solo valor para cada
    color
gray = lambda rgb : np.dot(rgb[... , :3] , [0.299 , 0.587, 0.114])
gray = gray(indexed2);
```

Ahora ***gray*** es un arreglo que tiene un solo valor por pixel, el cual nos permitirá visualizar el mapa en escala de grises Fig. 3. Una vez hecho esto, procedemos a mapear estos valores a las alturas. Los valores de altura en nuestro mapa van desde 780 metros hasta 1093 metros de elevación.

Con el siguiente código, logramos el cambio de escala de grises a escala de metros:

```
# Creamos una funci n para mapear las alturas en la escala de grises
def mapAltura(datos, maxR, minR):
    alturas = np.zeros([datos.shape[0],datos.shape[1]])
    m = (maxR-minR)/(np.max(datos)-np.min(datos))
    b = minR - (m * np.min(datos))
    for row in range(datos.shape[0]):
        for col in range(datos.shape[1]):
            alturas[row, col] = m * datos[row][col] + b
    return alturas


```

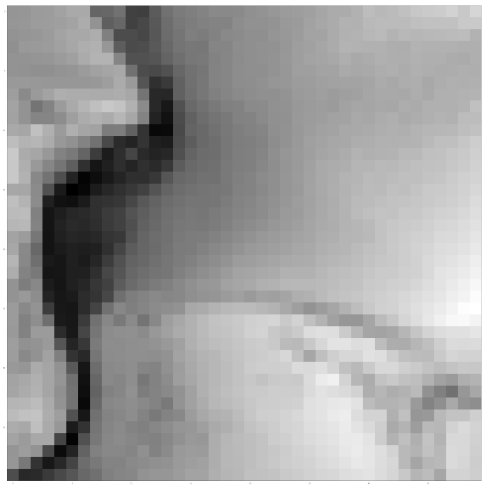

Figure 3: Escala de grises

```
# In[9]:


# Comprobamos la funci n
maxM = 1093
minM = 780

alt = mapAltura(gray, maxM, minM)
print(np.max(alt))
print(np.min(alt))
```

Una vez teniendo las alturas, realizamos un heatmap para poder visualizar de mejor manera las alturas por cada espacio en el array. Esta función crea un plot, con un rango de alturas, y que debe ser similar a la imagen original. Fig. 4.

```
# Obtenemos el mapa de calor con los valores de las alturas en escala de grises
sns.heatmap(alt)
```

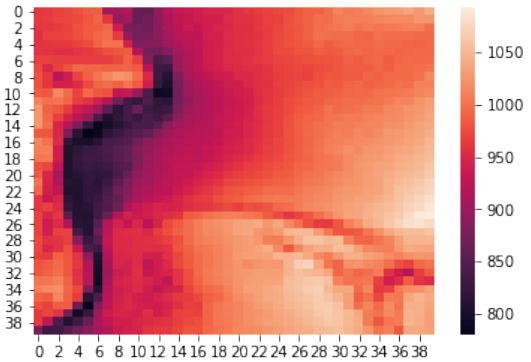

Figure 4: Heatmap de alturas.

6.1 Algoritmos de busqueda

Existen distintos algoritmos de búsqueda, las cuales le permiten a los agentes llegar a una meta predefinida cumpliendo con ciertos parámetros de acuerdo con el entorno donde se utiliza. Para la realización de nuestro proyecto, utilizamos las siguientes búsquedas: Breadth First Search, Depth First Search, Uniform Cost Search con y sin barreras, Greedy Search con dos funciones heurísticas y finalmente A* Search.

Para cada algoritmo tuvimos que establecer un tipo de entorno distinto, ya que algunos algoritmos tienen algunos requisitos. Para los algoritmos de búsqueda como Breadth First Search o Depth First Search tuvimos que implementar barreras en las regiones por donde el robot no puede circular ya sea por obstáculos o por desniveles pronunciados. Hubo otros algoritmos de búsqueda, como Uniform Cost Search o A* Search, para los que tuvimos que asignar un costo de paso en función del desnivel entre un nodo y su sucesor.

Para poder implementar los algoritmos de búsqueda creamos una clase para el grid base. Este código fue proporcionado por el profesor en una tarea previa. El grid nos permitirá crear un espacio para establecer los límites y trazar las rutas del agente.

Como se mencionó previamente, varios de los algoritmos requieren de la presencia de barreras para que su desempeño pueda ser evaluado. Las barreras deben ser colocadas donde haya un desnivel por donde el robot no pueda cruzar. Por conveniencia, establecimos una diferencia de 10 metros para nuestro código. Para localizar las barreras comparamos el valor de altura de cada nodo con sus nodos vecinos para comprobar si se cumple o no con la diferencia predeterminada.

```
# Creamos las paredes en los nodos donde haya una diferencia de altura de mas de 10 metros con alguno de sus vecinos
walls = []

for y in range(height-1):
    for x in range(width-1):
        dif1 = abs(alt[y][x] - alt[y-1][x])
        dif2 = abs(alt[y][x] - alt[y][x+1])
        dif3 = abs(alt[y][x] - alt[y+1][x])
        dif4 = abs(alt[y][x] - alt[y][x-1])
        dif = 10                                                    # Diferencia en metros
        if (dif1 > dif) or (dif2 > dif) or (dif3 > dif) or (dif4 > dif):
            walls.append((x+1,height-y-1))

len(walls)
```

Es importante mencionar que para que se establezca una pared en un nodo en particular, es necesario que la diferencia entre su valor y el valor de por lo menos un nodo vecino no cumpla con las condiciones preestablecidas. Las paredes consisten en un arreglo con coordenadas. El grid final se muestra en la Fig. 5.

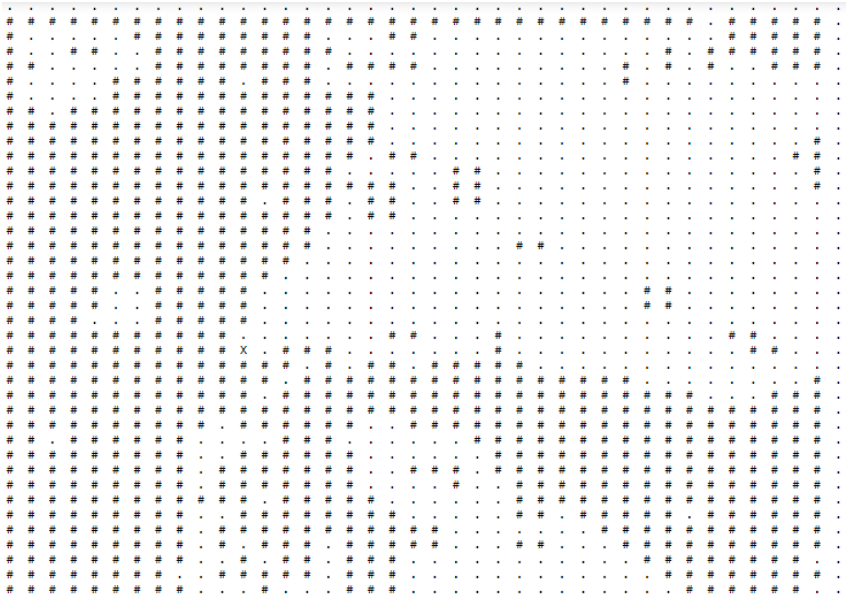

Figure 5: Grid del mapa.

6.2 Resultados de los diferentes algoritmos utilizados

Procederemos a revisar cada uno de los resultados obtenidos por los distintos algoritmos de búsqueda de este proyecto. Fue interesante ver cómo variaba el camino obtenido entre los métodos. Esto nos permite determinar qué algoritmo es mejor para llegar cumplir con nuestros objetivos.

6.2.1 Breadth First Search

Para esta búsqueda no informada, se desarrollan todos los nodos que nacen del nodo raíz hasta que se llega a la meta. Se hace uso de una estructura de datos tipo FIFO. Este algoritmo nos regresa siempre el camino más corto hasta la meta. Sin embargo, este algoritmo consume mucha memoria. Es por esto que el algoritmo no es eficiente cuando trabajamos con una gran cantidad de información. El resultado después de correr este algoritmo se muestra en la Fig. 6. El algoritmo llevó al agente desde la salida hasta la meta en ***48*** pasos.

Figure 6: Camino realizado por Breadth First Search.

6.2.2 Depth First Search

Para esta búsqueda se recorre siempre el siguiente nodo hijo. Si no encuentra el objetivo en esa rama, pasa a la siguiente. Utiliza una estructura de datos tipo LIFO. Este algoritmo tiene la ventaja de no utilizar tanta memoria como el algoritmo Breadth Firs Search. Sin embargo, no regresa una ruta óptima, sólo la primera que encuentra. La ruta que se encontró con este algoritmo se muestra en la Fig. 7. El algoritmo llevó al agente hasta la meta en ***120*** pasos.

Figure 7: Camino realizado por Depth First Search.

6.2.3 Uniform Cost Search

Para implementar este algoritmo se debe establecer un costo de paso para poder determinar la trayectoria con el menor costo. Lo que nos interesa a nosotros es que el robot se desplace por las superficies más planas posibles. Establecimos el costo de paso en función de la diferencia que existe entre un nodo y su sucesor. Este algoritmo de búsqueda utiliza una estructura de datos en la que el primero que sale es el costo acumulado menor. La trayectoria que nos regresó el algoritmo se muestra en la Fig. 8. Se llegó a la meta en un total de ***56*** pasos.

Figure 8: Camino realizado por Uniform Cost Search.

Este algoritmo también lo implementamos sin barreras, ya que el costo de paso es suficiente para determinar su trayectoria. El robot escoge su ruta teniendo como preferencia las superficies planas. La ruta se muestra en la Fig. 9. Completó la trayectoria en ***46*** pasos.

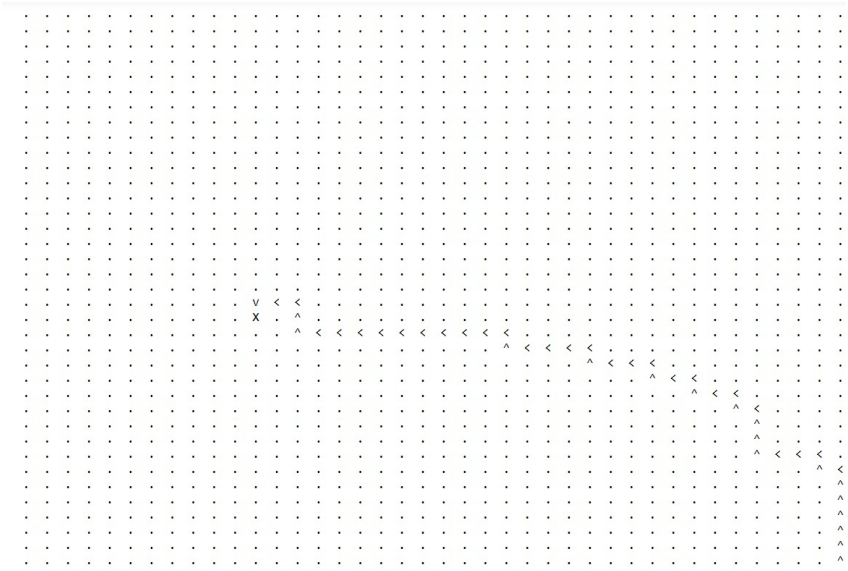

Figure 9: Camino realizado por Uniform Cost Search sin barreras.

6.2.4 Greedy Search

Greedy o su traducción en español "avaro", es un algoritmo que siempre busca el costo menor en su camino, por lo que siempre buscará los pasos que sea lo menor posible para poder llegar a su destino. Para determinar el costo menor, previamente se deben establecer estimaciones heurísticas, donde se mide el problema de forma simplificada. Este algoritmo hace uso de la misma estructura de datos que Uniform Cost Search. Implementamos el mismo algoritmo con dos heurísticas diferentes. La primera consiste en la "Distancia Manhattan"

$$|\, x1 - x2| + |\, y1 - y2|$$

El camino trazado por la primer heurística se muestra en la Fig. 10. Nos damos cuenta que se tiene un total de ***48*** pasos.

La segunda heurística consiste en la "Distancia euclidiana"

$$\sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

El camino trazado por la segunda heurística se muestra en la Fig. 11. El algorítmo encontra un camino de ***48*** pasos.

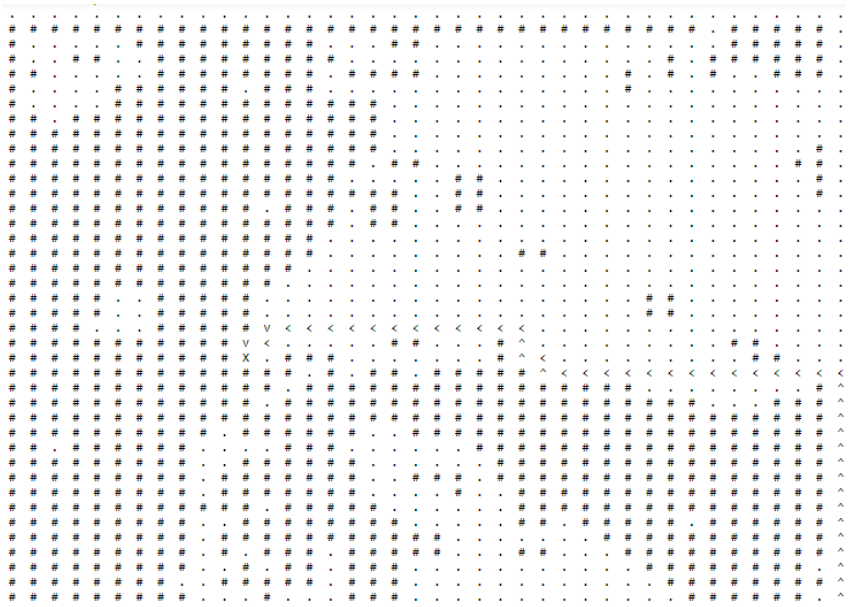

Figure 10: Camino realizado por Greedy Search con la primera heurística.

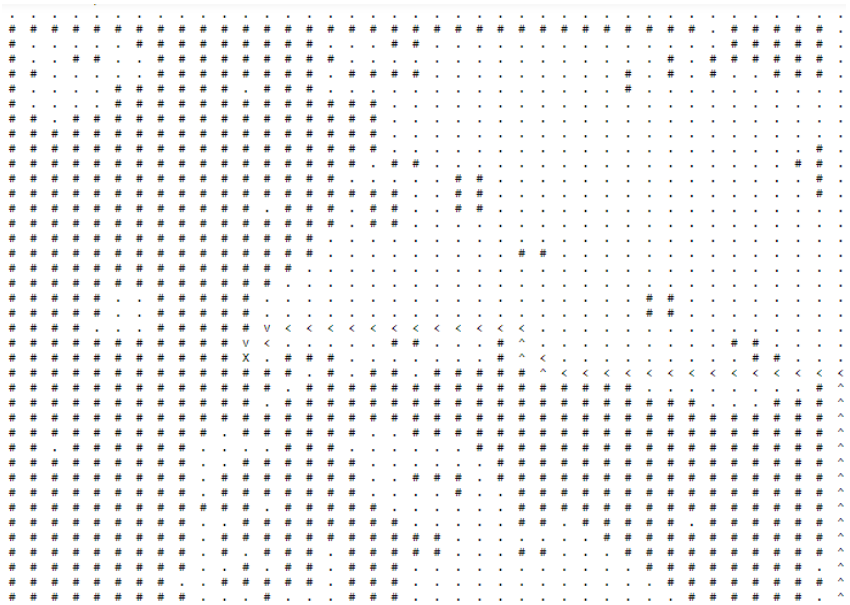

Figure 11: Camino realizado por Greedy Search con la segunda heurística.

6.2.5 A*

Por último, esta busqueda nos permite encontrar el costo mas bajo y la mejor estimación heurística. Podría decirse que es una combinación entre el algoritmo Uniform Cost Search y Greedy Search. Utiliza el mismo tipo de estructura de datos. El camino obtenido por el algoritmo A* se muestra en la Fig. 12. El total de pasos para este modelo fue de ***56***;

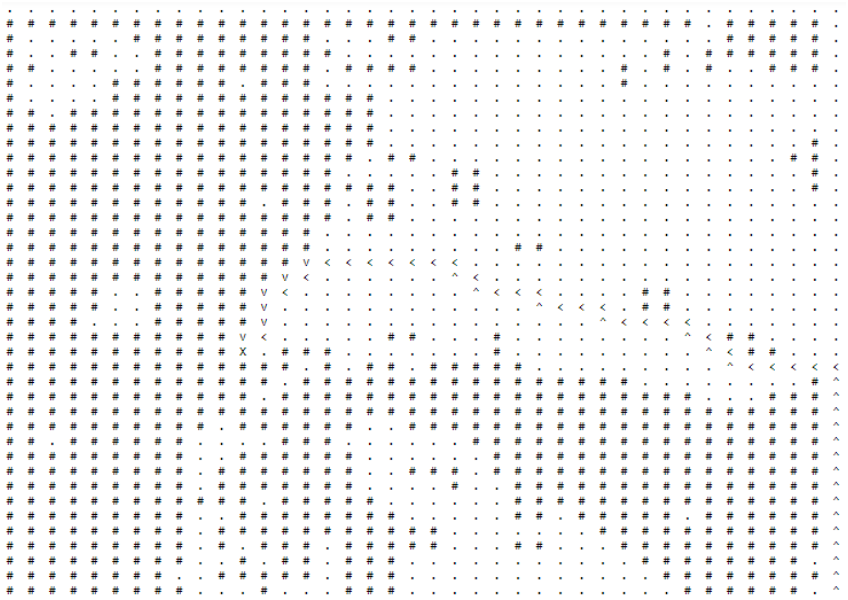

Figure 12: Camino realizado por A* Search.

7 Comparación de estos resultados.

Después de implementar los diferentes algoritmos de búsqueda y analizar sus resultados, podemos hacer una comparación entre su desempeño y seleccionar el más apropiado para nuestro agente inteligente. Podemos observar que, en general, los algoritmos de búsqueda siguieron una ruta muy similar. Con excepción del algoritmo Depth First Search. Este algoritmo no busca optimizar la ruta, por lo que fue el algoritmo que llegó a la meta en la mayor cantidad de pasos, 120. El algoritmo que tuvo el mejor desempeño en cuestión de cantidad de pasos fue el algoritmo Uniform Cost sin barreras, lo hizo en 46. El algoritmo estableció la ruta con el menor costo posible. Sin embargo, la ruta pudo haber cruzado por un desnivel considerable, si eso implicaba un costo acumulado menor. El algoritmo que tuvo más información para trazar la ruta fue A*. Este algoritmo conocía las barreras, es decir los lugares por donde podría caer o por donde no podría pasar, tomaba en cuenta el costo por subir o bajar en el terreno y contaba con las heurísticas. Este algoritmo obtuvo una ruta de 56 pasos. Otro algoritmo que obtuvo la misma cantidad de pasos es Uniform Cost. Este algoritmo conocía los lugares por lo que de plano no se puede pasar y tenía preferencia sobre superficies más planas. Los algoritmos que respetaron siempre las barreras y tuvieron la menor cantidad de pasos son Breadth First Search y Greedy search. Estos dos algoritmos alcanzaron la meta en 48 pasos. Breadth first search encontró la ruta óptima considerando que todos los nodos tienen el mismo costo. La desventaja es que consume demasiada memoria. El algoritmo Greedy Search, en sus dos heurísticas obtuvo el mismo número de pasos para llegar a la meta.

El objetivo del robot consiste en explorar y analizar el suelo de la superficie de Marte. Para esto no se necesita necesariamente utilizar la ruta más corta. Lo que sí es importante es que detecte por dónde puede y por dónde no puede desplazarse. Así podrá seguir funcionando. Es por esto que aunque el algoritmo Uniform Cost sin barreras haya obtenido la ruta con menos movimientos, no significa que sea la mejor alternativa. El algoritmo Breadth First search tampoco sería muy eficiente para el robot, ya que debería emplear más recursos en

implementarse. El algoritmo Depth First Search haría que el robot tomará caminos un poco redundantes y poco útiles, por lo que no es la mejor alternativa. Consideramos que lo mejor sería implementar un modelo de tipo A* con barreras, ya que cuenta con la mayor cantidad de información sobre el entorno. Sabe en dónde se encuentran los puntos por donde hay desniveles que puedan dañarlo, asigna costos de paso dependiendo de las prioridades que se establezcan, y establece funciones heurísticas.

8 Conclusiones.

En el transcurso de estas cinco semanas, aprendimos nuevos temas de inteligencia artificial, ejemplos, dispositivos y algoritmos, sobre todo aquellos que tengan como objetivo la búsqueda de un objetivo, dada o no información previa sobre el entorno. Clase con clase nos dimos cuenta que esta rama de la computación es de suma importancia para la creación, desarrollo e implementación de algoritmos en robots o agentes inteligentes que deben cumplir con los objetivos propuestos por los desarrolladores. A lo largo de este mes aprendimos de robots que perciben y actúan sobre el entorno que los rodea, gracias a distintos parámetros que han sido descritos en las clases como el PEAS, los CSP o los algoritmos de búsqueda.

Con este proyecto final, logramos llevar a cabo un compendio de todo lo realizado en clase, desde las bases de un robot y su funcionamiento con el entorno, hasta los algoritmos de búsqueda para encontrar el mejor camino de acuerdo a los requisitos que se pedían.

9 Roles y actividades de cada miembro del equipo.

La clase de Agentes Inteligentes permite la sana y activa colaboración entre compañeros de carrera, ya que el trabajar en parejas daba lugar a una contribución más dinámica y a la vez enérgica. Mientras un integrante del equipo realizaba alguna tarea, el otro podría investigar o dar ideas para la exitosa elaboración de los trabajos.

En el caso de nuestro equipo de trabajo, las actividades, trabajos y tareas se realizaban con el objetivo de que ambos aprendiéramos y entendiéramos lo que estábamos haciendo. Si alguno tenía dudas, el otro estaba ahí para ayudar, y si los dos no comprendemos algo, recurrimos a las asesorías con el profesor.

Consideramos que se trabajó eficazmente y ambos aprendimos mucho en esta materia, está claro que por el corto periodo de tiempo de la clase no fuimos capaces de ver todos los temas con tanta profundidad. Sin embargo, se logró comprender lo suficiente para poder llevar a cabo el proyecto final con éxito.

References

[1] Rusell Stuart, Norvig, P. (2015). Artificial Intelligence: A Modern Approach, Third Edition. Pearson.

[2] Amos, J. (2020, December 23). Nasa's Mars Rover and the 'SEVEN minutes of terror'. Retrieved February 17, 2021, from https://www.bbc.com/news/science-environment-55413966

[3] First Flight on Another Planet! [Video file]. (2019, August 10). Retrieved February 17, 2021, from https://www.youtube.com/watch?v=GhsZUZmJvaMt

[4] Mars 2020 - NASA Jet Propulsion Laboratory. (n.d.). Retrieved February 17, 2021, from https://www.jpl.nasa.gov/missions/mars-2020-perseverance-rover

[5] Mission overview. (n.d.). Retrieved February 17, 2021, from https://mars.nasa.gov/mars2020/mission/overview/

[6] Benítez Guerrero, E. I., Dr. (n.d.). Introducción a IA. Retrieved March 02, 2021, from https://www.uv.mx/personal/edbenitez/files/2010/09/CursoIA10-I-2.pdf

[7] JeffoG92 Seguir. (n.d.). Agentes de búsqueda online y ambientes desconocidos. Retrieved March 02, 2021, from https://es.slideshare.net/JeffoG92/agentes-de-bsqueda-online-y-ambientes-desconoc idos