

Planeación de rutas para la exploración en Marte

Marzo de 2024

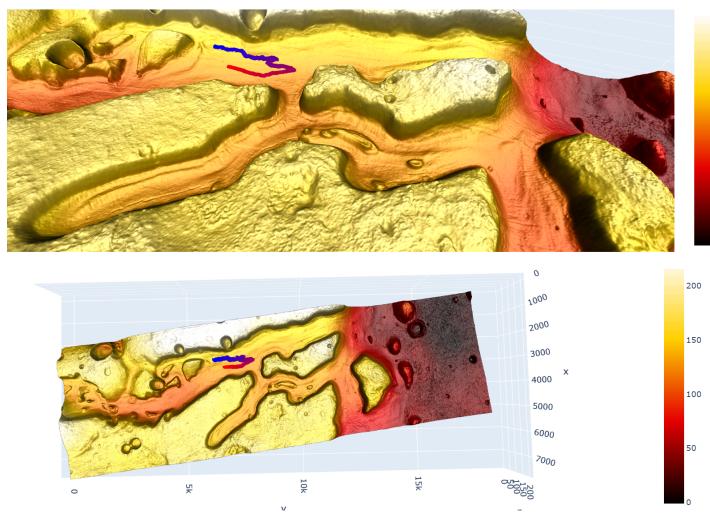
Gabriel Reynoso Escamilla | A01643561

Santiago Mora Cruz | A01369517

Guillermo Villegas Morales | A01637169

Prueba de algoritmos

Algoritmo A-Star:



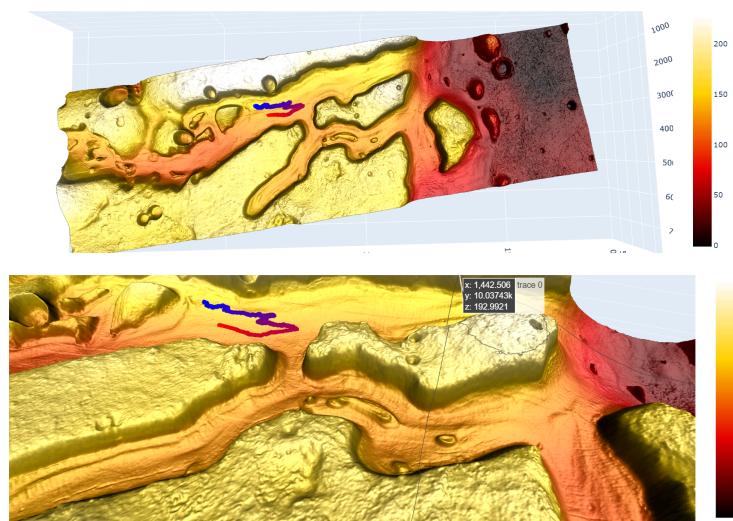
Ruta de rover al usar algoritmo A

```
(base) C:\Users\santi\OneDrive\Escritorio\python_scripts\TEC\4\agentes_inteligentes>python .\marcianito.py
Node <(1136, 314)>
Total distance 3518.898160498266
```

Distancia total de la ruta: 3518.898 m.

Encontró la solución en el nodo (1138, 314).

Búsqueda codiciosa de primero el mejor

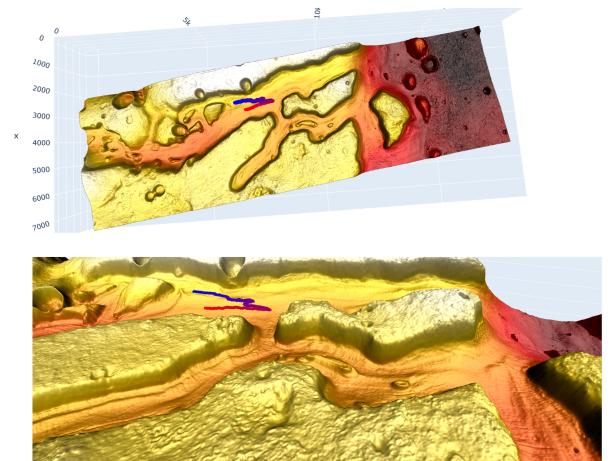


```
(base) C:\Users\santi\OneDrive\Escritorio\python_scripts\TEC\4\agentes_inteligentes>python .\marcianito.py  
Node <(1136, 314)>  
Total distance 4179.498584939848
```

Distancia total de la ruta: 4179.499 m.

Encontró la solución en el nodo (1136, 314).

Breadth first



```
(base) C:\Users\santi\OneDrive\Escritorio\python_scripts\TEC\4\agentes_inteligentes>python .\marcianito.py  
Node <(1136, 314)>  
Total distance 3430.0487171934683
```

Distancia total de la ruta: 3430.049 m.

Encontró la solución en el nodo (1136, 314).

Depth first

```
(base) C:\Users\santi\OneDrive\Escritorio\python_scripts\TEC\4\agentes_inteligentes>python .\marcianito.py
```

Este algoritmo no logró encontrar la solución en un tiempo razonable.

1. ¿Qué algoritmos lograron encontrar una ruta válida?

Utilizamos los algoritmos de A-star, primero el mejor, anchura primero y profundidad primero.

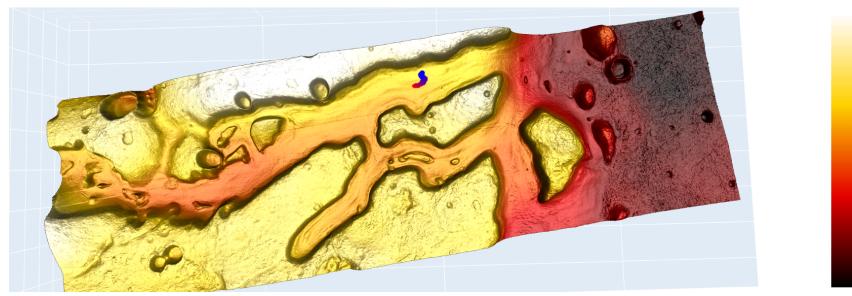
2. ¿Es necesario utilizar búsquedas informadas para este caso?

Si, ya que por la dimensión del árbol es necesario que el agente conozca, al menos, si se está acercando o alejando del objetivo. En este caso, como el objetivo estaba relativamente cerca del estado inicial, la búsqueda por anchura primero fue capaz de encontrar una solución en poca distancia, pero si estos puntos estuvieran más lejos entre sí, es más probable que las búsquedas informadas lo logren más pronto.

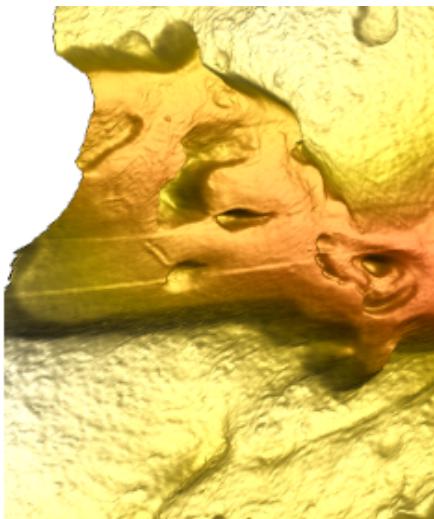
3. ¿Qué función heurística resultó adecuada para este problema?

Utilizamos el cálculo de la distancia de Manhattan como función heurística, pero otra distancia como Hamming también sería adecuada.

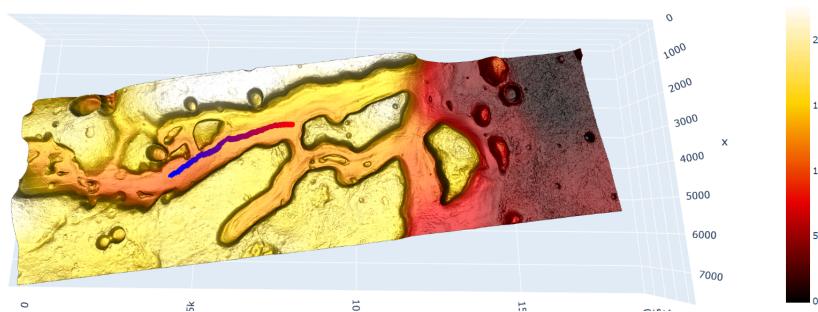
Rendimiento de los algoritmos de búsqueda para rutas cortas y largas



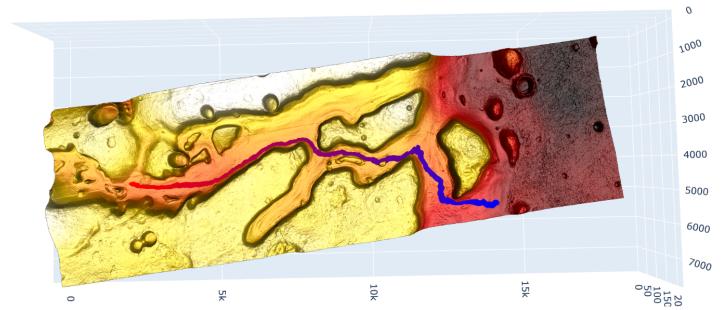
De (1800, 9550) a (2100, 9300) ~ 360 m
Distancia total de la ruta: 955.293 m



De (4500, 500) a (4600, 1050) ~ 559 m.
Fue incapaz de encontrar una ruta.



De (4480, 4400) a (2900, 8000) ~ 3931 m.
Distancia total de la ruta: 4634.982 m.



De (5409, 13974) a (5036, 8002404) ~ 11576 m.

Distancia total de la ruta: 16883.373 m.

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

En general, todos los casos que logran resolver el problema lo hacen en un tiempo considerable, aunque el caso de más de 10k metros de distancia se tarda razonablemente más. A excepción del caso dos de las distancias menores a 500 metros, que no pudo llegar a la solución

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

El algoritmo no es capaz de resolver el problema cuando el camino requiere que escale una altura mayor a 0.25 metros. Se podría experimentar con estos parámetros para que el robot sea capaz de encontrar una ruta.

Conclusiones:

Gabriel Reynoso Escamilla:

A pesar de la complejidad que parecería tener un problema de este tamaño la implementación resultó ser sencilla ya que teníamos ejemplos previos de problemas similares que podrían ser usados como base para este, la dificultad se encontró en realmente comprender el problema, los datos de la superficie y su escalamiento, con esto considerado fue relativamente sencilla la implementación del problema y los algoritmos de búsqueda. Al momento de comenzar la implementación surgen cuestiones que no se habían planteado como si el robot se mueve en diagonal o no. Por último al evaluar los algoritmos surgen errores que cometimos, errores con los operadores booleanos en las acciones, los bordes del mapa, y también al crear nuevos estados iniciales y objetivos nos dimos cuenta que dentro del algoritmo que funciona hay problemas que salen de las capacidades del agente.

Santiago Mora Cruz:

De los problemas que se han hecho este ha sido el más retador por distintos factores y elementos que el programa necesita para funcionar, como es la representación del mapa de marte en una matriz de valores. No obstante, el marco teórico obtenido en clase, el trabajo previo en algoritmos de búsqueda similares y la resolución de problemas parecidos a este

hicieron que se tuviera una mejor experiencia al trabajar en el problema, y que se pudiera resolver las dificultades que se presentaron durante el desarrollo del proyecto. Entre estas dificultades se tuvo la implementación del movimiento del robot en diagonal, el cuál no se había implementado en el programa del laberinto e impedía que el algoritmo de búsqueda pudiera encontrar una solución al problema. Esto se resolvió analizando el código existente de acciones y combinándolo para hacer el movimiento en diagonal. Entre las otras dificultades ocurrieron errores al implementar las restricciones de cada movimiento en las acciones, como las condiciones del borde de la matriz y no solo del mapa existente dentro de la matriz; y la elección de puntos iniciales y terminales para experimentar con el algoritmo. En general, fue una buena experiencia de aprendizaje en la que se superaron los obstáculos que se presentaron y se logró solucionar el problema planteado con todos los conocimientos previos.

Guillermo Villegas Morales:

Nos facilitó mucho el trabajo tener previamente la solución a un problema similar, fue cuestión de cambiar las acciones posibles y uno que otro detalle. No tuvimos complicaciones significativas, sin embargo, nos encontramos con obstáculos que tenían más que ver con los detalles y el planteamiento del problema. Un ejemplo de esto fue que solo consideramos que el robot se pudiera mover arriba, abajo, izquierda y derecha, cuando en realidad también se podía desplazar en diagonal. Debido a este detalle, el programa no encontraba la solución. Otra situación que tuvimos fue encontrarnos que el agente se salía del mapa durante su búsqueda, esto porque sólo consideramos como restricción que el agente no podía cruzar a una celda si su valor era de -1 ya que este valor representaba áreas sin escanear. Analizando el mapa caímos en cuenta que todo el mapa no estaba delimitado por un margen completo de -1, si no que existían valores válidos en las orillas del mapa. De ahí en adelante los errores que encontrábamos tanto en el código como en las soluciones que arrojaba fueron solucionados de forma rápida y sencilla. Con este proyecto aprendimos a plantear problemas de búsqueda tomando en cuenta las limitaciones, condiciones y posibilidades de una situación particular, así mismo entendimos que debemos analizar el problema con más detenimiento antes de comenzar a resolverlo.