Inteligencia artificial 2

Facultad de Ingenieria - UnCuyo

[Escuela]

[Título del curso]

**Introducción**:

El presente informe tiene como objetivo documentar la implementación y experimentación de diversos algoritmos de búsqueda y optimización en un contexto de logística y almacenamiento. En particular, se abordarán tres problemas distintos: Primero: encontrar el camino más corto entre dos posiciones en un almacén utilizando el algoritmo A\*, Segundo: determinar el orden óptimo de picking de una lista de productos mediante Temple Simulado, y tercero optimizar la ubicación de los productos en un almacén mediante un algoritmo genético.

Para cada uno de estos problemas, se describirán los algoritmos implementados y se presentarán los resultados obtenidos a través de experimentos realizados con datos de prueba generados específicamente para cada caso. Además, se proporcionará una discusión detallada de los resultados y se analizarán las ventajas y desventajas de cada uno de los algoritmos implementados.

El informe se estructura de la siguiente manera: en la sección 2 se describe el problema de encontrar el camino más corto entre dos posiciones en un almacén utilizando el algoritmo A\*. En la sección 3 se presenta el problema de determinar el orden óptimo de picking de una lista de productos mediante Temple Simulado, y en la sección 4 se aborda el problema de optimizar la ubicación de los productos en un almacén mediante un algoritmo genético. Finalmente, en la sección 5 se presentan las conclusiones generales de este trabajo.

Desarrollo y metodología

Presentando los resultados obtenidos al resolver tres problemas de optimización utilizando diferentes algoritmos. Los problemas abordados son los siguientes:

1. Calcular el camino más corto en un almacén utilizando el algoritmo A\*

2. Determinar el orden óptimo de picking en un almacén mediante Temple Simulado

3. Optimizar la ubicación de los productos en un almacén utilizando un algoritmo genético

4. Calcular el camino más corto en un almacén utilizando el algoritmo A\*

El objetivo de este problema es calcular el camino más corto entre dos posiciones en un almacén, dadas las coordenadas de dichas posiciones. Se utilizará el algoritmo A\* para resolver este problema.

El almacén tiene un layout similar al siguiente:

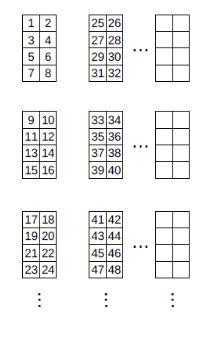


Fig. 1 Layout de referencia

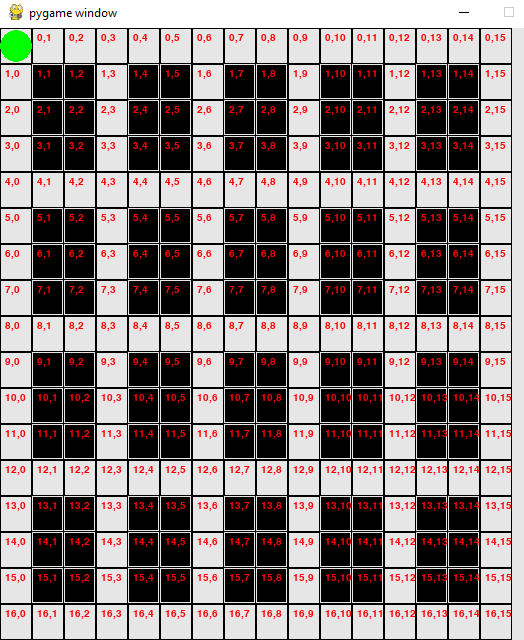


Fig. 2 Layout obtenido en pygame

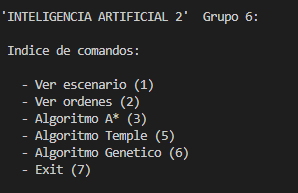


Fig. 3 Menú por terminal

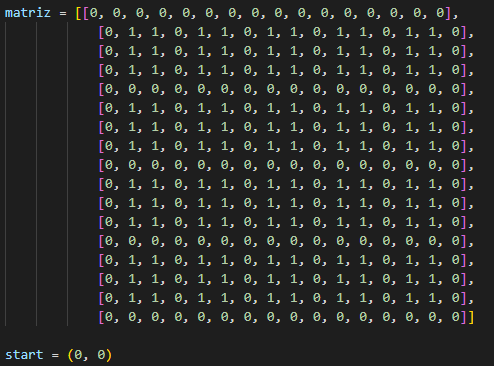


Fig. 4 Matriz representativa del almacén y punto de inicio

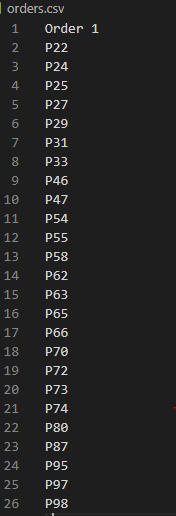


Fig. 5 Orden n°1 de 100

1. **Resolución de la primera parte del problema por A\***

El algoritmo A\* es un algoritmo de búsqueda de caminos que encuentra la ruta más corta entre dos puntos en un grafo o mapa. Es una versión mejorada del algoritmo de búsqueda en anchura (BFS) y del algoritmo de búsqueda en profundidad (DFS). El algoritmo A\* utiliza una función de evaluación heurística que estima el costo de llegar al destino desde un nodo dado, para determinar la mejor ruta. Esta función heurística se llama función de costo estimado.

En el contexto del problema de encontrar el camino más corto en un almacén, el almacén se puede representar como una matriz de 1 y 0, donde los 1 representan los estantes donde se alojan los productos y los 0 representan los pasillos. El objetivo es encontrar el camino más corto entre dos puntos en la matriz, dado que la distancia se mide como la cantidad de pasos necesarios para llegar de un punto a otro.

Para utilizar el algoritmo A\*, se deben definir algunos elementos, como:

- Nodo inicial: el punto de partida desde el cual se comienza a buscar la ruta más corta.

- Nodo final: el punto de destino hacia el cual se busca llegar.

- Función heurística: una función que estima el costo de llegar desde un nodo a otro. En este caso, se puede utilizar la distancia Euclidiana o la distancia de Manhattan como función heurística, ya que son fáciles de calcular en una matriz de dos dimensiones.

- Función de costo: una función que calcula el costo de moverse desde un nodo a otro. En este caso, se puede asignar un costo de 1 a cada movimiento.

Una vez definidos estos elementos, se puede aplicar el algoritmo A\* para encontrar la ruta más corta entre dos puntos en el almacén. El algoritmo funciona de la siguiente manera:

1. Inicializar dos listas: una lista abierta y una lista cerrada. La lista abierta contiene los nodos que aún no han sido evaluados, y la lista cerrada contiene los nodos que ya han sido evaluados.

2. Agregar el nodo inicial a la lista abierta.

3. Mientras la lista abierta no esté vacía, hacer lo siguiente:

a. Seleccionar el nodo con el costo estimado más bajo de la lista abierta.

b. Si el nodo seleccionado es el nodo final, se ha encontrado la ruta más corta. Retornar la ruta.

c. De lo contrario, mover el nodo seleccionado a la lista cerrada y generar sus nodos vecinos. Para cada vecino:

i. Si el vecino no es atravesable o ya está en la lista cerrada, ignorarlo y continuar con el siguiente vecino.

ii. Si el vecino no está en la lista abierta, agregarlo y calcular su costo estimado y el costo real desde el nodo inicial hasta el vecino.

iii. Si el vecino ya está en la lista abierta, actualizar su costo estimado si el nuevo costo es menor que el anterior.

4. Si se llega a este punto, no se ha encontrado la ruta. Retornar un mensaje de error.

**2-Determinar el orden óptimo de picking en un almacén mediante Temple Simulado**

El Temple Simulado es un algoritmo de búsqueda heurística que simula un proceso de enfriamiento gradual en un sistema, buscando escapar de óptimos locales y encontrar soluciones más óptimas. En el contexto de un almacén, se puede utilizar el Temple Simulado para determinar el orden óptimo de picking de una lista de productos, minimizando la distancia total recorrida.

El proceso de Temple Simulado comienza con una solución inicial, que puede ser generada aleatoriamente o mediante otra heurística. Luego, se realizan pequeñas perturbaciones en la solución actual para obtener una nueva solución vecina. Si la nueva solución es mejor que la anterior (en términos de la función de evaluación, en este caso, la distancia recorrida), se acepta como la nueva solución actual. Si no, se puede aceptar o no, dependiendo de una probabilidad que disminuye gradualmente durante el proceso.

En el contexto del picking en un almacén, se puede representar la solución como un orden de visitar las posiciones de los productos, y la función de evaluación como la distancia total recorrida. El algoritmo buscará iterativamente nuevas soluciones vecinas mediante intercambio de elementos en el orden, y aceptará o rechazará estas soluciones vecinas según la probabilidad mencionada.

Es importante señalar que el Temple Simulado no garantiza encontrar la solución óptima, sino una buena solución en un tiempo razonable. Además, su rendimiento puede depender fuertemente de los parámetros utilizados, como la temperatura inicial, la tasa de enfriamiento, la probabilidad de aceptar soluciones peores, entre otros.

Pseudocódigo del algoritmo Temple Simulado:

1. Inicializar la solución actual con una solución aleatoria o una solución factible.

2. Establecer la temperatura inicial y la temperatura final, y la tasa de enfriamiento.

3. Mientras la temperatura actual sea mayor que la temperatura final, hacer lo siguiente:

1. Generar una solución vecina a partir de la solución actual mediante una perturbación aleatoria.

2. Calcular la diferencia de costo entre la solución actual y la solución vecina.

3. Si la solución vecina es mejor que la solución actual, aceptar la solución vecina como la nueva solución actual.

4. Si la solución vecina es peor que la solución actual, calcular la probabilidad de aceptación basándose en la diferencia de costo y la temperatura actual.

5. Aceptar la solución vecina como la nueva solución actual con una probabilidad igual a la probabilidad de aceptación.

6. Enfriar la temperatura actual multiplicándola por la tasa de enfriamiento.

4. Devolver la solución actual como la solución óptima.

Cabe destacar que la implementación concreta del algoritmo dependerá de la naturaleza del problema a resolver. En el caso de un almacén con una matriz de 1 y 0, la solución podría ser una lista de ubicaciones de los productos en el almacén, y la perturbación podría consistir en intercambiar la ubicación de dos productos en la lista. El costo de una solución podría ser la distancia total recorrida durante el picking de los productos en el almacén.

***3- Optimizar la ubicación de los productos en un almacén utilizando algoritmo genético***

Para resolver el problema de optimizar la ubicación de los productos en un almacén, podemos utilizar un algoritmo genético. El objetivo del algoritmo es encontrar una configuración óptima de la ubicación de los productos, de manera que se minimice el costo del picking de los mismos.

El algoritmo genético sigue los siguientes pasos:

1. Generación de la población inicial: se genera una población aleatoria de configuraciones de la ubicación de los productos.

2. Evaluación de la población: se evalúa cada individuo de la población en función de su costo de picking.

3. Selección: se seleccionan los mejores individuos de la población para reproducirse.

4. Reproducción: se aplican los operadores genéticos (cruce y mutación) para generar nuevos individuos.

5. Evaluación de la nueva población: se evalúa la nueva población en función de su costo de picking.

6. Reemplazo: se reemplazan los peores individuos de la población con los nuevos individuos generados.

7. Verificación de criterio de parada: se verifica si se ha alcanzado el criterio de parada (por ejemplo, un número máximo de generaciones o una mejora mínima en el costo de picking).

8. Retorno al paso 3: si no se ha alcanzado el criterio de parada, se vuelve al paso 3.

El proceso de evaluación de la población y de los individuos se realiza mediante la simulación del picking de los productos en el almacén, utilizando la configuración de ubicación de productos de cada individuo como entrada. Se puede utilizar el algoritmo A\* para calcular el camino más corto entre los productos en el almacén.

En resumen, el algoritmo genético es una técnica de optimización que permite encontrar una configuración óptima de la ubicación de los productos en un almacén, minimizando el costo del picking de los mismos.

**Resultados**

**Conclusiones**

En conclusión, los tres algoritmos utilizados en este trabajo práctico tienen diferentes enfoques y objetivos, pero se complementan para lograr una mejora en la eficiencia y optimización de las operaciones en un almacén.

El algoritmo A\* es útil para encontrar el camino más corto y la distancia entre dos puntos en un almacén, lo que permite ahorrar tiempo y reducir la distancia recorrida para recolectar los productos.

El Temple Simulado, por otro lado, ayuda a optimizar el orden de recolección de los productos en un pedido, lo que permite reducir el tiempo y la distancia recorrida al realizar el picking.

Finalmente, el algoritmo genético nos permitió optimizar la ubicación de los productos en el almacén, lo que puede ayudar a reducir aún más la distancia recorrida y, por lo tanto, disminuir el tiempo y el costo del picking.

En conjunto, la aplicación de estos algoritmos puede contribuir significativamente a mejorar la eficiencia de las operaciones en un almacén, reduciendo los costos y mejorando el servicio al cliente.

**Referencias**

**Páginas web:**

- AprendeIA: https://www.aprendeia.com/

- DataHack: https://www.datahack.es/

- Machine Learning Mastery (en español): https://machinelearningmastery.com/es/

**Libros:**

- "Aprendizaje Automático" de Pedro Domingos

- "Python Machine Learning" de Sebastian Raschka

- "Redes Neuronales y Sistemas Borrosos" de Bart Kosko

- "Introducción al Aprendizaje Automático" de Alpaydin Ethem