EXAMEN III: PLANEACIÓN DE MOVIMIENTO EN ROBÓTICA MULTI-AGENTE

Presentado por:

Juan Diego Gomez Chavarro

Presentado por:

German Andrés Holguín Londoño

Universidad Tecnológica de Pereira

14 de junio del 2024

INTRODUCCIÓN

La planificación de movimiento en robótica multi-agente es un área de estudio que se centra en el diseño y control de varios robots que interactúan en un entorno común. El objetivo principal en la planificación de movimiento de sistemas multi-agente es asegurar que cada robot llegue a su destino (tareas) sin chocar con obstáculos en el entorno. Esto requiere resolver problemas complejos, como la generación de trayectorias óptimas y la adaptación a cambios en el entorno.

El entorno puede ser representado mediante un grafo planar $G(E,V)$, donde cada vértice representa una ubicación en el espacio (ver Figura 1). Esta representación gráfica permite utilizar algoritmos como *BFS*, *DFS* o Dijkstra, entre otros, para encontrar la trayectoria más corta entre posiciones.

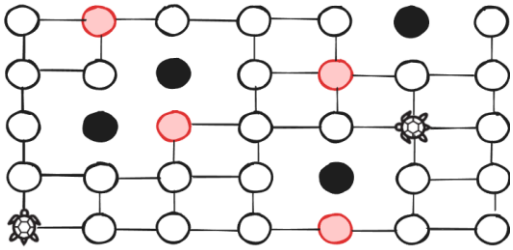

Figura 1: representación grafica del entorno.

Por otro lado, un método común para abordar el problema de planificación es la programación lineal, que a partir de un conjunto de restricciones lineales y variables de decisión se busca minimizar una función objetivo que comúnmente es la distancia recorrida por los agentes.

METODOLOGÍA

A. Generación de espacio de trabajo.

Como se mencionó anteriormente, el espacio de trabajo puede ser representado mediante un grafo planar $G(V,E)$. En cada posición del grafo puede haber un agente (tortuga), un obstáculo (círculo negro) o una tarea (punto rojo), los cuales se ubican aleatoriamente con la condición de que las tortugas y las tareas no se superpongan con los obstáculos (ver Figura 1). Al colocar un obstáculo en un vértice, se eliminan las aristas conectadas a dicho vértice para evitar el acceso de los agentes. De este modo, el agente deberá encontrar una ruta que no pase por ese vértice.

B. Algoritmo de Dijkstra

Para encontrar el camino entre un agente y una tarea, se utilizará el algoritmo de Dijkstra. Este método, empleado en la teoría de grafos, sirve para hallar las rutas más cortas desde un nodo de origen a todos los demás nodos en un grafo dirigido y ponderado. Funciona seleccionando el nodo con la menor distancia acumulada desde el origen y actualizando las distancias a sus vecinos si se encuentra una ruta más corta. Este proceso se repite hasta que todos los nodos hayan sido procesados.

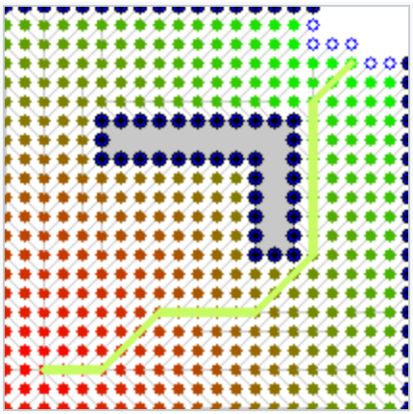

Figura 2: Camino basado en Dijkstra

Dado que en este caso las aristas tienen el mismo peso, el algoritmo de Dijkstra se comporta como un algoritmo BFS (búsqueda en amplitud), lo cual tiene sentido, ya que Dijkstra elige los vértices más cercanos al vértice inicial.

A. Minimización de distancia

Para minimizar la distancia recorrida por las tortugas para realizar todas las tareas, se implementará, por su similitud al problema actual, un algoritmo basado en programación lineal basado en las restricciones del Problema de Enrutamiento de Vehículos con Múltiples Depósitos (*MDVRP*) que son planteadas por *Andrew (Doi:* 10.1109/TASE.2005.853472*)*. A continuación, se presenta la descripción del MDVRP adaptada a nuestro problema.

1- Hay *M* posiciones de las tortugas (denotadas como un conjunto *D*) y cada posición $k(k \epsilon D)$, pertenece a exactamente una tortuga v_k
2- Cada tortuga, v_k tiene una ruta, R_k, realizando las diferentes tareas, iniciando desde su correspondiente posición y retornando a la misma posición inicial.
3- Todas las *N* Tareas (denotadas como un conjunto *C*) deben ser realizadas y cada una de ellas deben ser visitadas por solamente una tortuga.

Basado en la distancia *Manhattan* entre los puntos (Tareas y Tortugas), $d_{ij}(i; j \in C \cup D)$, calculada por sus ubicaciones, esta versión *MDVRP* adaptada para este problema, tiene como objetivo determinar a qué tortuga se le asigna cada cliente y el orden de realización de las tareas para cada tortuga. Dado que se busca minimizar la distancia recorrida por las tortugas la función objetivo puede ser representa por la siguiente expresión.

$$min \sum_{k \epsilon F} \sum_{J \epsilon C \cup D} d_{ij} * \sum_{I \epsilon C \cup D} x_{ijk}$$

Donde d_{ij} es la distancia entre la tarea i y la tarea j. x_{ijk} es una variable binaria que es uno si la tortuga k va de la tarea i a la tarea j, y es cero de no ser así.

A continuación, se presentan las restricciones necesarias para plantear el algoritmo de programación lineal.

1. Todas las tareas deben ser visitadas exactamente una vez.

$$\sum_{k \in D} \sum_{i \in C \cup D} x_{ijk} = 1 \ \forall j \in C$$

2. Una tortuga que llega a una tarea debe salir de ella *(restricción de conservación de flujo).*

$$\sum_{j \in C \cup D} x_{ijk} = \sum_{j \in C \cup D} x_{jik} \qquad \forall k \in M, \qquad i \in C \cup D,$$

3. La siguiente restricción es la conocida restricción de eliminación de sub-tours, la cual impone que para cada vehículo v_k, al menos una ruta salga de cada conjunto de clientes S visitado por v_k

$$\sum_{i \in S} \sum_{j \in S} x_{ijk} \leq |S| - 1 \quad \forall S \subseteq C, \qquad |S| \geq 2, \qquad k \in D$$

4. Requerimiento binario sobre las variables de decisión.

$$x_{ijk} \epsilon \{0, 1\} \qquad \forall i, \qquad k \epsilon D, \qquad j \epsilon C$$

RESULTADOS

Para verificar el funcionamiento de la metodología mostrada anteriormente, se ha creado un espacio de trabajo de 10 filas y 20 columnas, con diferente número de agentes, tareas y obstáculos.

Generación espacio de trabajo 10 x 20:

Ver video: https://www.youtube.com/watch?v=VD6p1mwxcmY

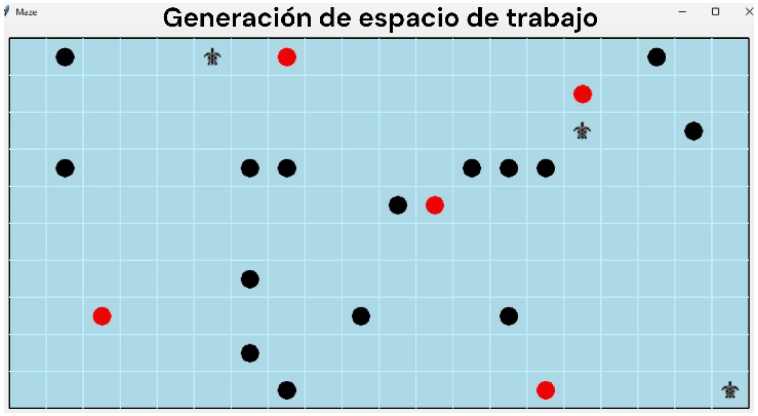

Figura 3: Generación de espacio de trabajo

Solución utilizando 1 agente 15 tareas:

Ver video: https://youtu.be/VsbMuaHRBfQ

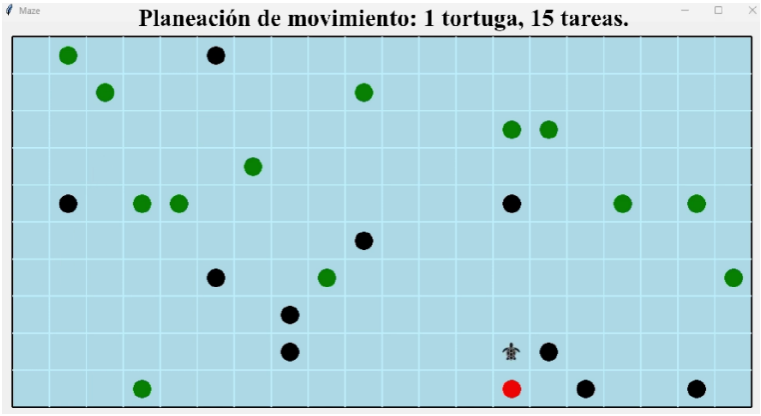

Figura 4: Planeación de movimiento: 1 tortuga, 15 tareas.

Solución utilizando 3 agentes 10 tareas:

Ver video: https://youtu.be/wmJkhXcyXUE

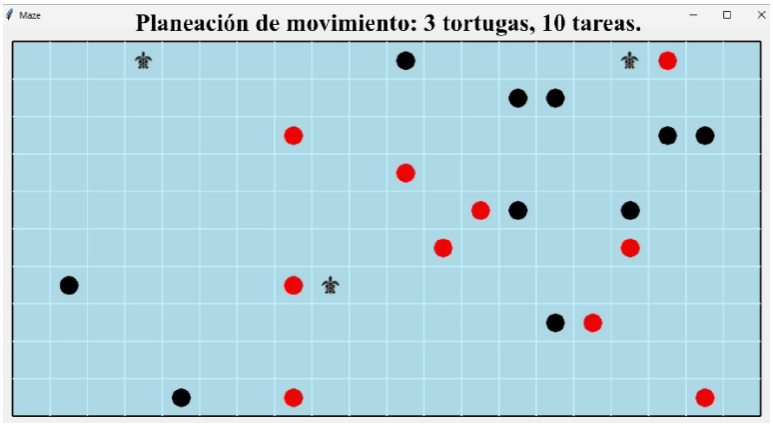

Figura 5: Planeación de movimiento: 3 tortugas, 10 tareas.

Solución utilizando 15 agentes 10 tareas:

Ver video: https://youtu.be/fEwtkBiK1Zo

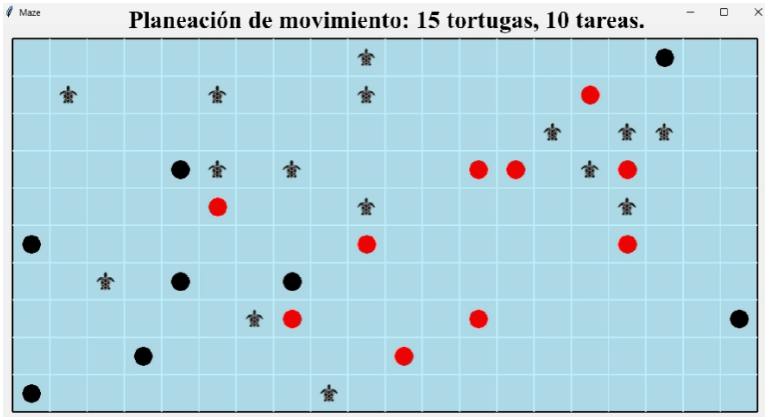

Figura 5: Planeación de movimiento: 15 tortugas, 10 tareas.

CONCLUSIONES

En este estudio se desarrolló un conjunto de algoritmos para abordar el problema de la planificación de movimiento en robótica multi-agente. Entre los algoritmos implementados se incluye el algoritmo de Dijkstra para determinar caminos entre dos puntos y un algoritmo de programación lineal aplicado al problema de enrutamiento de vehículos con múltiples depósitos, cuyo objetivo es minimizar la distancia recorrida por los agentes al realizar sus tareas.

En el mundo real, el problema de múltiples agentes tiene numerosas aplicaciones prácticas. Por ejemplo, puede ser aplicado en la distribución y logística de empresas de entrega de productos, en el transporte de pasajeros para diseñar rutas más eficientes, o en servicios de emergencia para coordinar la respuesta desde múltiples estaciones.

Durante la implementación de los algoritmos se observó que los agentes no siempre obtienen las rutas más óptimas. Esto se debe a que el modelo de programación lineal considera la distancia que toma al agente regresar a su posición inicial. Otro problema identificado surge cuando hay pocas tareas en comparación con el número de agentes, ya que el modelo obliga a los agentes a salir de su posición inicial, lo que resulta en movimientos innecesarios hacia la posición del agente más cercano.