uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática

Curso académico 2024-2025

Trabajo Fin de Grado

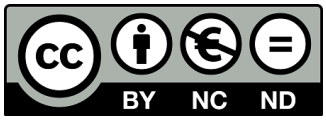
"Algoritmo de Optimización para la Recolección de Pedidos en el sector Retail: Equilibrando Eficiencia y Reglas Logísticas"

Pablo González Madroño

Tutor

Ricardo Aler Mur

Lugar y fecha de presentación prevista



RESUMEN

Palabras clave:

DEDICATORIA

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

2. ESTADO DEL ARTE

Empezar a hablar aqui sobre el estado del arte. Cosas de las que quiero hablar

2.1. Introducción general al problema

En todas las empresas de retail a nivel mundial, existe un proceso logístico fundamental que puede determinar si un pedido llega a tiempo o si una venta resulta rentable. Cada día se gestionan miles de pedidos compuestos por diversos artículos que deben enviarse tanto a clientes finales como a distintos puntos de venta.

La velocidad con la que se lleva a cabo este proceso logístico es un factor clave. De su eficiencia no solo depende directamente la satisfacción del cliente, sino que también puede impactar en la imagen de la empresa. Un retraso puede traducirse en pérdida de confianza, daño reputacional y consecuencias económicas importantes.

Vamos a entender mejor el proceso de la recogida de pedidos, también conocida como *order picking*. Este proceso consiste en la localización, recogida y preparación de cada uno de los artículos que componen un pedido realizado por un cliente. Esto ocurre tanto en almacenes como en tiendas con acceso al público, como por ejemplo los supermercados. Este último es el foco de este proyecto.

Existen diferentes técnicas para la recolección de pedidos:

- **Pick-to-order**: los pedidos se recogen de manera completa e independiente. Es decir, cada pedido está asociado a un empleado, quien realiza la recolección completa.
- **Batch picking**: se recolectan varios pedidos al mismo tiempo para optimizar los recorridos. Es decir, se agrupan los pedidos de tal manera que un mismo empleado recolecta varios pedidos simultáneamente. Sin embargo, esto requiere una posterior reorganización de los artículos para dividirlos en cada una de las órdenes.
- **Zone picking**: a cada empleado se le asigna una zona específica, y este recolecta los artículos correspondientes a su área. Posteriormente, se realiza la organización necesaria para preparar los pedidos individuales.

Cada una de estas técnicas de preparación de pedidos presenta ventajas e inconvenientes particulares. Según el contexto específico, unas resultan más convenientes que otras. En el caso concreto de este proyecto, centrado en la optimización del proceso de recogida de pedidos en un supermercado, la empresa emplea la técnica de **Pick-to-order**.

Aunque esta metodología puede ser menos eficiente en términos de velocidad de recolección en comparación con *Batch picking* o *Zone picking*, reduce significativamente

los errores en la fase de reagrupación de pedidos. Esto es especialmente relevante en el contexto de un supermercado, donde los pedidos suelen contener un gran número de artículos, muchas veces muy similares entre sí. En estas situaciones, la complejidad y el esfuerzo asociados a la clasificación y división posterior de productos hacen que utilizar las otras técnicas mencionadas no resulte rentable a nivel operativo.

Por otro lado, en otras secciones destinadas a centros comerciales o almacenes, donde la cantidad media de artículos por pedido es considerablemente menor, sí se recurre a la técnica de *Batch picking*. En estos casos, agrupar varios pedidos y recolectar varios artículos simultáneamente permite mejorar notablemente la eficiencia del proceso, sin que ello suponga un riesgo elevado de errores en la organización posterior.

La elección de la técnica de preparación de pedidos adecuada, como hemos visto, responde a una necesidad de equilibrio entre precisión y eficiencia operativa, adaptándose a las características concretas del entorno y del tipo de pedido. Sin embargo, esta decisión no está exenta de retos. La implementación práctica de estas estrategias plantea una serie de desafíos que van más allá de la pura elección metodológica, siendo el más destacado de ellos los altos costes logísticos.

Según Koster, Le-Duc y Roodbergen [1], la recolección de pedidos es la actividad más costosa y con mayor consumo de mano de obra en los almacenes, representando hasta un 55 % del coste operativo total de un almacén. Este estudio menciona cómo la complejidad de este proceso presenta una tendencia creciente debido a las nuevas exigencias del mercado. Con cada vez más frecuencia, se solicitan pedidos más pequeños, altamente personalizados, y con plazos de entrega más reducidos, lo cual es necesario para mantener la competitividad en un entorno comercial cada vez más exigente.

Por otro lado, una ineficiencia en las rutas de los recolectores conlleva un desgaste tanto físico como mental. Esto no solo reduce la eficiencia del trabajador, sino que también incrementa la probabilidad de cometer errores durante la preparación de los pedidos.

Es por todo lo anterior que resulta vital optimizar las rutas de recolección. La optimización de rutas consiste en diseñar recorridos más eficientes, con el objetivo de minimizar la distancia total recorrida por el *picker*. En el estudio de Koster, Le-Duc y Roodbergen [1] el autor argumenta que, este desplazamiento representa hasta un 50 % del tiempo total de trabajo de estos empleados. Por tanto, la consecuencia natural de lograr este objetivo es la reducción del tiempo total de preparación de los pedidos, maximizando al mismo tiempo el nivel de servicio. Esto se traduce en menores tiempos de espera para el cliente y una reducción significativa en los errores cometidos durante el proceso logístico.

Ahora que ya hemos comprendido por qué es importante optimizar las rutas en la recogida de pedidos, vamos a abordar el problema desde una perspectiva técnica. Dado que nos enfrentamos a un desafío inherentemente complejo, resulta conveniente dividirlo en subproblemas más pequeños y manejables. Esto facilita no solo una comprensión más clara y precisa de cada parte involucrada, sino que también permite identificar de forma específica la información necesaria para resolver cada subproblema. Como resultado, se-

remos capaces de establecer con mayor precisión un camino hacia una solución global eficiente.

Tal como se ha discutido anteriormente, el objetivo principal radica en determinar la ruta más corta posible que permita recorrer todos los artículos que forman parte de un pedido específico. Para ilustrar este concepto, consideremos un ejemplo sencillo en el que un pedido está compuesto por tres artículos distintos: *leche*, *huevos* y *pollo*. Existen múltiples rutas posibles para recolectar estos artículos, comenzando desde un punto de origen (que podría ser el área inicial donde se sitúa el *picker*) y concluyendo en un punto de finalización (por ejemplo, la zona destinada al empaquetado).

En este escenario concreto, las combinaciones potenciales para realizar el recorrido serían:

- Punto de origen → Leche → Huevos → Pollo → Punto de finalización
- Punto de origen → Leche → Pollo → Huevos → Punto de finalización
- Punto de origen → Huevos → Leche → Pollo → Punto de finalización
- Punto de origen → Huevos → Pollo → Leche → Punto de finalización
- Punto de origen → Pollo → Leche → Huevos → Punto de finalización
- Punto de origen → Pollo → Huevos → Leche → Punto de finalización

Sin embargo, la pregunta clave radica en cómo determinar cuál de estas posibles rutas es la óptima. El criterio para elegir la mejor ruta es aquella que **minimice la distancia total recorrida**, ya que, generalmente, una distancia menor se traduce directamente en un menor tiempo empleado en la recolección. Por lo tanto, para evaluar cada una de las rutas y encontrar cuál es la que recorre una menor distancia, es necesario calcular la distancia entre cada uno de los artículos involucrados en el pedido, así como la distancia desde el punto d origen hacia cada artículo y desde cada artículo hacia el punto de finalización.

Para modelar adecuadamente este problema podemos utilizar la teoría de grafos. El problema se puede representar mediante un **grafo conectado** en el que cada nodo corresponderá a un artículo del pedido, incluyendo dos nodos adicionales para el punto inicial y el punto final del recorrido. Las aristas que conectan estos nodos representarán las distancias entre los distintos artículos.

Este enfoque nos permite utilizar algoritmos y técnicas de la teoría de grafos para resolver el problema. En particular, el problema que se nos presenta es una variante del conocido *Travelling Salesman Problem* (TSP), en el que un agente debe visitar un conjunto de nodos exactamente una vez y regresar al punto inicial, minimizando la distancia total recorrida. En este caso, se trata de una versión **asimétrica**, ya que puede haber puntos de entrada y salida distintos.

Por lo tanto, podemos dividir el problema en dos partes fundamentales:

1. Calcular la distancia entre todos los artículos del supermercado.
2. Usar algoritmos de búsqueda para encontrar la ruta que minimiza la distancia total recorrida.

A continuación, se detallará cada una de las partes del problema.

2.2. Cálculo de distancias entre productos

2.2.1. Introducción al cálculo de distancias

Antes de abordar el cálculo de distancias en el entorno real de un supermercado, es conveniente entender cómo se definirían estas distancias en un escenario idealizado, en el que suponemos que no existen obstáculos que interfieran con la navegación. Es decir, en el que se pudiera navegar libremente entre todos los artículos, permitiendo considerarse un desplazamiento directo entre cualquier par de puntos.

Es importante destacar que lo que se persigue no es tanto obtener las distancias absolutas entre productos, sino garantizar que todas estén expresadas en una misma escala relativa. De esta forma, lo que importa no es si entre el producto A y B hay 50 unidades y entre B y C hay 10, o si las distancias fuesen 5 y 1 respectivamente, sino que las proporciones se mantengan y reflejen fielmente las relaciones espaciales.

Si no se respetan estas proporciones, el algoritmo de optimización podría generar *soluciones subóptimas*, ya que estaría operando sobre un sistema de distancias inconsistentes. En consecuencia, se vería inducido a seleccionar rutas que, aunque aparenten ser eficientes dentro del modelo, no lo son en la realidad. Esta distorsión compromete la calidad de los resultados, al basarse en una representación espacial que no refleja fielmente las verdaderas relaciones entre los productos.

Para poder calcular estas distancias de forma sistemática, es necesario primero representar espacialmente los productos. Una práctica común en áreas como la *logística*, la *robótica* o los *sistemas de recomendación* consiste en modelar el entorno mediante una cuadrícula regular. En este modelo, cada producto se asocia a una posición dentro de la cuadrícula, la cual puede interpretarse como un sistema de coordenadas bidimensionales. Así, cada artículo queda definido por un par ordenado $(x, y) \in \mathbb{N} \times \mathbb{N}$ que indica su ubicación en el espacio.

Este sistema permite representar todos los productos dentro de un mismo marco espacial, lo que facilita la comparación directa entre ellos. Además, posibilita el cálculo objetivo de las distancias, ya que de esta manera, se transforma el problema de relaciones entre productos en un problema geométrico, donde se pueden aplicar métricas de distancias bien definidas.

Esta configuración también cumple con la necesidad de respetar las proporciones

reales de todas las distancias entre productos, ya que todas están representadas en una misma escala relativa.

2.2.2. Distancias en entornos sin obstáculos

- Distancia euclídea, Manhattan, Chebyshev
- Cuándo se utilizan y sus limitaciones

Como hemos visto, la selección de una métrica de distancia apropiada no solo influye en la calidad de la solución obtenida, sino también en la eficiencia del algoritmo utilizado. En este tipo de problemas destacan tres métricas para calcular la distancia entre dos puntos cualesquiera: la distancia euclídea, la distancia Manhattan y la distancia Chebyshev. A continuación se describe cada una de ellas junto a sus principales limitaciones.

Distancia euclídea

La distancia euclidiana se remonta a las antiguas matemáticas griegas y recibe su nombre por el famoso Euclides. Esta métrica es la forma más intuitiva de medir la distancia recta entre dos puntos en el espacio. Se corresponde a la longitud del segmento recto que une dos puntos cualesquiera. Se obtiene aplicando *El teorema de Pitágoras*. Veamos cómo se calcula matemáticamente:

Dados dos puntos cualesquiera $\mathbf{p}$ y $\mathbf{q}$, que pertenecen al espacio euclídeo $\mathbb{R}^n$, la **distancia euclídea** entre ellos se define como:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

En el caso particular de dos dimensiones ($n = 2$), si los puntos son:

$$\mathbf{p} = (x_p, y_p), \quad \mathbf{q} = (x_q, y_q)$$

entonces la distancia euclídea se calcula como:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$$

TODO: AÑADIR EJEMPLO CALCULO CON NUMEROS Y VISUALIZACION PYTHON

Esta métrica es adecuada en entornos donde el agente se puede desplazar libremente en cualquier dirección. Es comúnmente utilizada en algoritmos de aprendizaje automático, como k-Nearest Neiighbors y clustering y navegacion automática

TODO: AÑADIR REFERENCIAS PARA ESTO

No obstante, la principal desventaja de esta métrica es su alto coste computacional debido al calculo de la raíz cuadrada. Lo que en sistemas con grandes volúmenes o en tiempo real puede ser limitante.

Distancia de Manhattan

La distancia de Manhattan es una métrica utilizada para determinar la distancia entre dos puntos en forma de cuadricula. a diferencia de la distancia euclidiana, que mide la distancia mas corta posible entre dos puntos, la distancia de Manhattan mide la suma de las diferencias absolutas entre las coordenadas de dos puntos. El nombre surge porque simula un taxi que circula por las calles cuadriculadas de Manhattan, donde no hay diagonales y debe recorrer las lineas de la cuadricula. Veamos como se calcula:

Dados dos puntos cualesquiera $\mathbf{p}$ y $\mathbf{q}$, que pertenecen al espacio n-dimensional $\mathbb{R}^n$, la **distancia de Manhattan** entre ellos se define como la suma de las diferencias absolutas de sus cordenadas cartesianas:

$$d_{\text{Manhattan}}(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} |p_i - q_i|$$

En el caso particular de dos dimensiones ($n = 2$), si los puntos son:

$$\mathbf{p} = (x_p, y_p), \quad \mathbf{q} = (x_q, y_q)$$

entonces la distancia de Manhattan se calcula como:

$$d_{\text{Manhattan}}(\mathbf{p}, \mathbf{q}) = |x_p - x_q| + |y_p - y_q|$$

Esta métrica representa la distancia total recorrida siguiendo caminos paralelos a los ejes del plano

TODO: AÑADIR EJEMPLO CALCULO CON NUMEROS Y VISUALIZACION PYTHON

Esta distancia resulta adecuada para aquellos problemas en los que el movimiento del agente está restringido a los movimientos principales (norte, sur, este y oeste). Pero, a diferencia de la distancia euclídea, es muy eficiente computacionalmente.

El mayor inconveniente de esta metrica es en entornos donde se permiten movimientos diagonales, esta métrica puede sobrestimar la distancia real, conduciendo a trayectorias subóptimas o poco naturales

Distancia Chebyshev

La distancia Chebyshev mide el numero mínimo de pasos necesarios para alcanzar un punto en entornos donde se permitan desplazamiento en todas las direcciones incluyendo las diagonales. Se define como el máximo de las diferencias absolutas entre las coordenadas. Se calcula de la siguiente manera:

Dados dos puntos cualesquiera **p** y **q**, que pertenecen al espacio euclídeo $\mathbb{R}^n$, la **distancia de Chebyshev** entre ellos se define como:

$$d_{\text{Chebyshev}}(\mathbf{p}, \mathbf{q}) = \max_{1 \leq i \leq n} |p_i - q_i|$$

En el caso particular de dos dimensiones ($n = 2$), si los puntos son:

$$\mathbf{p} = (x_p, y_p), \quad \mathbf{q} = (x_q, y_q)$$

entonces la distancia de Chebyshev se calcula como:

$$d_{\text{Chebyshev}}(\mathbf{p}, \mathbf{q}) = \max\left(|x_p - x_q|,\ |y_p - y_q|\right)$$

Una forma tradicional de visualizar esta métrica es pensar en el número mínimo de movimientos necesarios en un tablero de ajedrez para que una pieza como el rey se desplace de un punto a otro. Esta distancia es especialmente útil en entornos de cuadricula donde se permite el movimiento en las ocho direcciones posibles, como por ejemplo en videojuegos.

Su principal ventaja es su bajo coste computacional, ya que solo incluye operaciones de suma, resta y máximo. sin embargo, sus uso está limitado a los casos en los que se permita movimiento diagonal, ya que, en caso contrario, puede ofrecer estimaciones de coste poco precisas.

2.2.3. Distancias en entornos con obstáculos

- Cómo se modela un supermercado: cuadrículas, grafos y nodos
- Representación de obstáculos: estanterías, pasillos y paredes

Las métricas de distancia descritas, Euclídea, Manhattan y Chebyshev. Resultan adecuadas en entornos ideales en los que no existen obstáculos físicos que interfieran en el desplazamiento del agente o sistema. Bajo estas condiciones, dichas métricas permiten calcular de forma computacionalmente eficiente una estimación precisa de la distancia entre dos puntos dentro de una cuadrícula. No obstante, cuando se incorporan obstáculos al entorno, por ejemplo estanterías, muros o zonas inaccesibles dentro de un supermercado, estas métricas dejan de reflejar la distancia real que debe recorrer un empleado. Las

distancias calculadas mediante los métodos anteriores no representan trayectorias factibles, sino que proporcionan una cota inferior del coste real. Esto se debe a que el agente ya no puede desplazarse en línea recta ni por los caminos más cortos posibles definidos por las métricas puras, sino que debe bordear los obstáculos, incrementando necesariamente la longitud del trayecto.

TODO: Introducir imagen que se vea esto

Esta discrepancia se vuelve especialmente problemática en el contexto de algoritmos de optimización que requieren información precisa sobre los costes de desplazamiento. Tal y como se expuso anteriormente, una estimación errónea o demasiado optimista de las distancias puede inducir al sistema a seleccionar rutas que aparentan ser eficientes según el modelo, pero que en la práctica resultan inviables o subóptimas. Por tanto, para que los algoritmos optimizadores generen soluciones verdaderamente óptimas en entornos realistas, es imprescindible emplear representaciones de distancia que tengan en cuenta la topología del espacio, incluyendo la presencia de obstáculos.

Para ello, se hace necesaria la introducción de algoritmos de búsqueda de caminos. Estos algoritmos permiten encontrar trayectorias viables entre dos puntos de un espacio discretizado, minimizando el coste total del recorrido y considerando explícitamente las restricciones impuestas por la configuración del entorno. A diferencia de las métricas geométricas estáticas, los algoritmos de búsqueda se apoyan en una exploración activa del entorno, identificando rutas factibles y óptimas según el criterio establecido (por ejemplo, distancia mínima, tiempo mínimo o riesgo mínimo).

2.3. Algoritmos de búsqueda

Para poder aplicar algoritmos de búsqueda de caminos en un entorno discretizado, es necesario realizar una modelización formal de problema. Tal como se describe en Russel y Norvig [2], este tipo entorno se representa habitualmente como un grafo dirigido, en el cual los nodos corresponden a los estados posibles del sistema. Las aristas representan las acciones que permiten la transición de un estado a otro.

El conjunto completo de estados alcanzables conforma lo que se denomina espacio de estados. El espacio de estados constituye el dominio sobre el que se opera el algoritmo de búsqueda.

Según la formulación clásica expuesta por Russel y Norvig [2], antes de aplicar un algoritmo de búsqueda es necesario definir los siguientes componentes fundamentales:

- **Estado inicial**: La posición de partida, desde la que se inicia la búsqueda del camino.
- **Estado objetivo**: Representa el destino que se desea alcanzar

- **Espacio de estados**: Conjunto de todos los estados posibles del sistema, definidos por la configuración de entorno y las reglas de movimiento.
- **Acciones**: Conjunto de operaciones o movimientos permitidos que permiten transitar de un estado a otro.
- **Función de coste**: Función que asigna un valor cuantitativo al uso de cada acción.

En el apartado dedicado al diseño de la solución, se detallará como se han parametrizado estos componentes para el caso específico del cálculo de distancia entre productos en un supermercado.

Los algoritmos de búsqueda, se clasifican en dos tipos: Algoritmos de búsqueda no informada y Algoritmos de búsqueda informada. Veamos cada uno de ellos, pero antes, definamos cuatro parámetros utilizados comúnmente para comparar algoritmos de búsqueda:

- **Completitud**: La completitud se refiere a la capacidad del algoritmo para encontrar una solución si esta existe en un tiempo finito.
- **Optimalidad**: La optimalidad indica si el algoritmo es capaz de encontrar la mejor solución posible. Es decir, la que minimize/maximice el coste total.
- **Complejidad temporal**: La complejidad temporal mide cuantos nodos del espacio de búsqueda debe generar y procesar un algoritmo en el peor de los casos. Esta métrica mide la eficiencia en tiempo de ejecución y suele expresarse en función del factor de ramificación b y de la profundidad de la solución d.
- **Complejidad espacial**: La complejidad espacial se refiere a la cantidad de memoria necesaria para almacenar todos los nodos generados durante la búsqueda. Es un criterio crítico en problemas donde los recursos de memoria son limitados.

2.3.1. Algoritmos de búsqueda no informada

Los algoritmos de búsqueda no informada, constituyen la base teórica fundamental de enfoques más avanzados dentro del ámbito de la inteligencia artificial. Su principal característica radica en la ausencia de conocimiento específico sobre el estado objetivo o sobre la estructura del espacio de búsqueda. En consecuencia, las decisiones sobre que camino explorar se toman sin disponer de estimaciones que orienten la búsqueda hacia la solución, lo que implica una exploración sistemática del espacio de estados..

No obstante, estos algoritmos ofrecen un marco de referencia objetivo que permite comparar el rendimiento de otros algoritmos más complejos. De hecho, en escenarios donde la información heurística es inexistente, poco fiable o computacionalmente exigente de conseguir, los algoritmos de búsqueda no informada pueden representar una

alternativa eficaz. Además, son algoritmos fácil de implementar y fácilmente adaptables a cualquier contexto, por lo que resuelven problemas de manera fehaciente en contextos donde enfoques más especializados son ineficientes.

Hay dos algoritmos de búsqueda no informada básicos:

Breadth-First Search (BFS)

La búsqueda en anchura explora el espacio de estados de manera sistemática y nivel por nivel, comenzando por el nodo raíz y expandiendo todos los nodos presentes a una determinada profundidad antes de pasar a la siguiente. Este algoritmo garantiza la completitud, ya que encontrará una solución si existe, y es óptimo siempre que el coste de todas las acciones sea uniforme. No obstante, presenta una alta complejidad espacial, ya que debe almacenar en memoria todos los nodos generados a cada nivel, y, en problemas con un espacio de estados grande puede llegar a ser inviable de aplicar por falta de recursos.

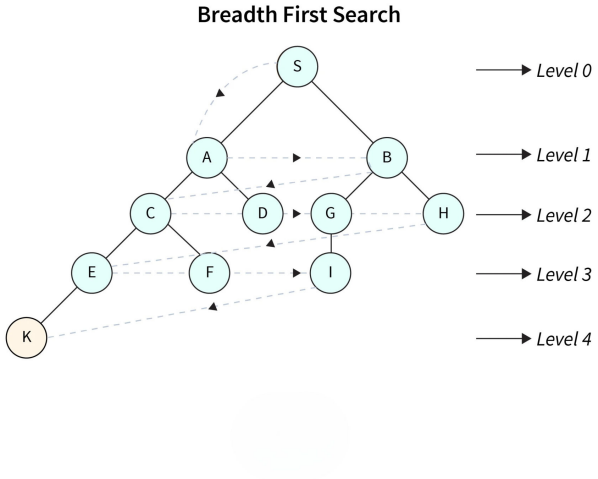


Fig. 2.1. BFS aplicado a un arbol de búsqueda

Depth-First Search (DFS)

El principio operativo de la búsqueda en profundidad, consiste en expandir el camino más profundo posible en el espacio de búsqueda antes de retroceder (backtracking) para considerar rutas alternativas. Es decir, a partir del nodo inicial se expande recursivamente el siguiente nodo hijo disponible hasta alcanzar un estado objetivo o hasta que no sea posible continuar, moomento en el que se retrocede hacia el nodo anterior para explorar otros caminos no visitados.

Una de las ventajas principales de esta estrategia es su eficiencia en términos de uso

de memoria, ya que no necesita mantener todos los nodos en memoria simultaneamente, en su lugar, únicamente requiere almacenar el camino en exploración. No obstante, este algoritmo no es completo si el espacio de búsqueda es infinito o contiene ciclos, ya que puede quedarse atrapado explorando un camíno infinito. Además DFS no garantiza la optimalidad, ya que puede encontrar una solución subóptima antes que otras de menor coste.

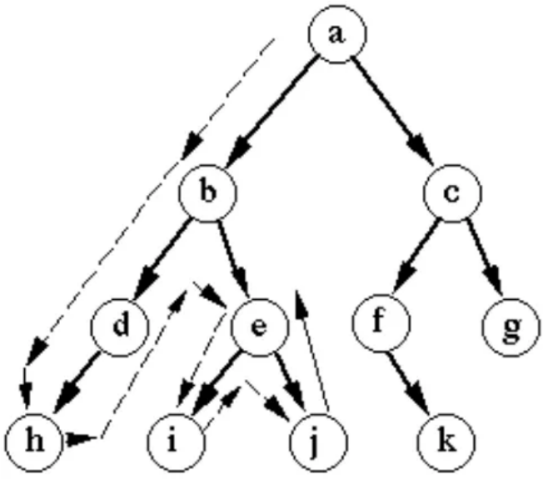


Fig. 2.2. DFS aplicado a un arbol de búsqueda

2.3.2. Algoritmos de búsqueda informada

Los algoritmos de búsqueda informada incorporan conocimiento adicional sobre el problema o sobre el espacio de búsqueda mas allá de la definición del problema en si. Su principal característica es el uso de funciones heurísticas que estiman el costo o la distancia hacia el objetivo, permitiendo guiar la búsqueda de manera más eficiente que en los enfoques no informados.

Gracias a esta orientación, los algoritmos informados exploran primero los caminos que parecen más prometedores con la esperanza de encontrar la solución antes, reduciendo así el número de nodos explorados y mejorando la eficiencia computacional y temporal. Esto les convierte en herramientas especialmente útiles en escenarios donde el espacio de búsqueda es grande o las soluciones óptimas son complicadas de encontrar mediante exploración exhaustiva.

Sin embargo, el rendimiento de estos algoritmos depende en gran medida de la calidad de la heurística utilizada. Además, en muchas ocasiones, no es trivial el diseño de una función heurística que estime de manera cercana a la realidad el coste de llegar al objetivo. Una estimación mal diseñada puede llevar a resultados subóptimos o incluso a un desempeño inferior al de algoritmos no informados. Aun así, en contextos donde se dispone de conocimiento fiable o aproximaciones razonables del objetivo, los algoritmos de búsqueda informada son ampliamente utilizados para encontrar soluciones óptimas de

manera mas eficaz.

Funciones heurísticas

La manera mas habitual de añadir información adicional externo a la definición del problema es a través de funciones heurísticas. Una función heurística, denotada $h(n)$, son funciones matemáticas que se caracterizan por ser arbitrarias, no negativas y especificas para cada problema. Además cualquier función heurística debe cumplir que si n es un nodo objetivo, entonces $h(n) = 0$. Russel y Norvig [2]

Mas allá de estas propiedades fundamentales, existen dos criterios relevantes a la hora de evaluar la calidad de una heurística, su admisibilidad y si es informada o no.

- **Admisibilidad**: Una heurística se considera admisibles si se cumple que nunca sobreestima el coste real de alcanzar el objetivo desde cualquier nodo n. Es decir se debe cumplir:

  $$\forall n \in E, \quad h(n) \leq h^*(n)$$

  donde:

  - E: es el espacio de estados.
  - $h(n)$: es el valor estimado del costo desde el nodo n hasta el objetivo.
  - $h^*(n)$: es el costo real del camino óptimo desde n hasta el nodo objetivo.

- **Heurística informada:** Una heurística informada es aquella que se aproxima de manera precisa al coste real. En términos comparativos, una heurística h_1 se dice que es más informada que otra h_2 si, para todos los nodos n, se cumple que:

  $$h_1(n) \geq h_2(n)$$

  sin perder la propiedad de admisibilidad. Una heurística más informada se traduce en una búsqueda más eficiente, ya que se reduce la exploración de caminos menos prometedores.

  Un ejemplo de función heurística admisible es la distancia de Manhattan, en problemas de planificación de rutas sobre cuadrículas donde el movimiento está restringido a direcciones ortogonales. En contextos donde se permite el movimiento en cualquier dirección, una heurística más informada podría ser la distancia Euclídea, que, sin dejar de ser admisible si no sobreestima el coste real, proporciona una mejor aproximación a la distancia efectiva al objetivo

Greedy best-first search

El algoritmo *Greedy Best-First Search* es una estrategia de búsqueda informada que orienta la exploración del espacio de estados utilizando una función de evaluación exclusivamente formada por la función heurística $h(n)$. Es decir, no tiene en cuenta el coste acumulado desde el nodo inicial. Su objetivo es expandir, en cada paso se selecciona el nodo con el menor valor de $h(n)$, en un intento de alcanzar la solución de manera rápida siguiendo el camino más prometedor.

Formalmente, la función de evaluación utilizada por el algoritmo es:

$$f(n) = h(n)$$

Este algoritmo se caracteriza por su eficiencia en términos de tiempo, ya que en muchos casos puede encontrar soluciones en menos pasos que otros algoritmos. Sin embargo, esta eficiencia es alcanzada a expensas de la optimalidad y la completitud. Al no considerar el coste acumulado desde el nodo inicial$g(n)$, Greedy Best-First Search puede elegir caminos que aparentan ser cercanos al objetivo pero que, en realidad, implican un coste total elevado o incluso conducen a estados sin sucesores, y al desestimar todas las demás rutas puede no encontrar solución, por lo que no es un algoritmo completo ni óptimo.

Entre sus ventajas destaca su bajo requerimiento de memoria, así como su capacidad para explorar con rapidez grandes espacios de búsqueda cuando se dispone de una heurística informada. No obstante, su desempeño está fuertemente condicionado por la calidad de dicha heurística: una estimación imprecisa puede desviar al algoritmo hacia regiones no óptimas del espacio de estados, o incluso hacer que no encuentre ninguna solución si esta se encuentra tras nodos con una evaluación elevada por la función heurística.

Algoritmo A*

El algoritmo A*, propuesto por Hart, Nilsson y Raphael [3] en 1968, introduce una estrategia de búsqueda informada que utiliza una función de evaluación heurística para decidir qué nodo expandir en cada paso del proceso de búsqueda. Esta función se define como:

$$f(n) = g(n) + h(n)$$

donde:

- $g(n)$ representa el costo real acumulado desde el nodo inicial hasta el nodo n,
- $h(n)$ es una estimación heurística del costo desde el nodo n hasta el objetivo.

El algoritmo mantiene dos listas: nodos *abiertos* (candidatos a expansión) y *cerrados* (ya explorados), y selecciona siempre el nodo abierto con menor valor de $f(n)$ para su expansión.

Una de las contribuciones más relevantes del algoritmo A* es que garantiza encontrar una solución óptima si la heurística utilizada es **admisible** Hart, Nilsson y Raphael [3].

Además, si la heurística es también **consistente** (satisface la desigualdad triangular: $h(n) \leq c(n, n') + h(n')$), entonces A* no necesita reabrir nodos y expande el número mínimo posible de nodos entre todos los algoritmos admisibles que usan la misma información heurística Hart, Nilsson y Raphael [3].

La calidad de la heurística utilizada afecta directamente al rendimiento del algoritmo. Una heurística informada (más cercana al costo real) permite reducir significativamente la cantidad de nodos explorados. En casos extremos:

- Si $h(n) = 0$ (sin información), A* se comporta como una búsqueda de costo uniforme.
- Si $h(n) = h^*(n)$ (conocimiento perfecto), A* se convierte en una búsqueda directa óptima sin expansión innecesaria.

El algoritmo A* puede adaptarse a distintas aplicaciones modificando la heurística según la naturaleza del problema. Por ejemplo, en un grafo geográfico, una heurística típica sería la distancia euclídea o Manhattan entre un nodo y el objetivo Hart, Nilsson y Raphael [3].

A continuación y a modo de resumen, se presenta una tabla comparativa de todos los algoritmos descritos, que se presentan como candidatos para el problema descrito de encontrar la distancia exacta entre cada par de productos.

Algoritmo	**Completitud**	**Optimalidad**	**Complejidad Temporal**	**Complejidad Espacial**	**Informado**
Búsqueda en anchura (BFS)	Sí*	Sí*	$O(b^d)$	$O(b^d)$	No
Búsqueda en profundidad (DFS)	No**	No	$O(b^m)$	$O(m)$	No
Greedy Best-First Search	No	No	$O(b^m)$	$O(b^m)$	Sí
A*	Sí***	Sí****	$O(b^{1+C^*/\epsilon})$	$O(b^{1+C^*/\epsilon})$	Sí

TABLA 2.1. COMPARATIVA DE ALGORITMOS DE BÚSQUEDA EN FUNCIÓN DE CRITERIOS FUNDAMENTALES.

* Solo si el coste de las acciones es uniforme.

** No garantiza completitud en espacios infinitos o con ciclos.

*** Si se utiliza una heurística admisible (nunca sobrestima el coste real al objetivo).

**** Requiere que la heurística sea admisible y consistente (cumple la desigualdad triangular).

En la sección de diseño de solución se detallará cual fue el algoritmo escogido y las razones para ello.

2.4. Ordenación óptima de la ruta: el Problema del Viajante de Comercio (TSP)

2.4.1. Historia y origen del TSP

- Contexto histórico: logística, matemáticas aplicadas, computación
- Importancia teórica y práctica

2.4.2. Formulación del problema

- Qué es el TSP y cómo se relaciona con la recogida de pedidos
- Representación como circuito cerrado: visitar todos los productos una sola vez

2.4.3. Métodos exactos para resolver el TSP

- Fuerza bruta (backtracking)
- Programación dinámica (Held-Karp), Branch and Bound

2.4.4. Métodos heurísticos y metaheurísticos

- Búsqueda local: nearest neighbor
- Algoritmos genéticos, recocido simulado (Simulated Annealing)
- Cuándo usar cada uno: equilibrio entre velocidad y calidad de solución

3. DISEÑO

4. RESULTADOS

BIBLIOGRAFÍA

[1] R. de Koster, T. Le-Duc y K. J. Roodbergen, "Design and control of warehouse order picking: A literature review," *European Journal of Operational Research*, vol. 182, n.º 2, pp. 481-501, 2007.

[2] S. Russel y P. Norvig, *Artificial intelligence: A Modern approach*, 4.ª ed. Prentice Hall, 2021.

[3] P. E. Hart, N. J. Nilsson y B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, n.º 2, pp. 100-107, 1968. DOI: 10.1109/TSSC.1968.300136.

[4] S. K. Hui, P. S. Fader y E. T. Bradlow, "Research Note—The Traveling Salesman Goes Shopping: The Systematic Deviations of Grocery Paths from TSP Optimality," *Marketing Science*, vol. 28, n.º 3, pp. 566-572, 2009.

[5] V. Chugani, *Comprender la distancia euclidiana: De la teoría a la práctica*, Datacamp.com, oct. de 2024. [En línea]. Disponible en: https://www.datacamp.com/es/tutorial/euclidean-distance.

[6] S. Ou, Z. H. Ismail y N. Sariff, "Hybrid Genetic Algorithms for Order Assignment and Batching in Picking System: A Systematic Literature Review," *IEEE Access*, vol. 12, pp. 23 029-23 042, 2024. DOI: 10.1109/ACCESS.2024.3357689.