



Bases de la Inteligencia Artificial

LEARN MORE ABOUT IT

01

Busqueda en grafos

Utilizando Algoritmos de búsqueda en grafos es posible encontrar soluciones a un problema planteado

02

Probabilidad

Utilizando modelos probabilísticos logra representar la incertidumbre en los datos esto debido a que los modelos probabilísticos permiten hacer predicciones y tomar decisiones basadas en la probabilidad de que un evento ocurra



ENFOQUES IA

03

Logica

Utilizando la lógica formal logra representar el conocimiento y las relaciones entre los datos. La idea es crear reglas lógicas que describan el problema y luego utilizar la inferencia lógica para obtener nuevas conclusiones.

01

Busqueda en grafos

Son algoritmos utilizados para explorar un grafo o red de nodos interconectados en busca de una solución o un objetivo específico.

Planificación

La planificación en la búsqueda en grafos se refiere al proceso de diseñar una estrategia de búsqueda eficiente para encontrar la solución de un problema modelado como un grafo



Permite encontrar una solución a un problema sin utilizar información específica sobre el problema o el espacio de búsqueda



Puede llegar a ser ineficiente en algunos casos, ya que puede explorar caminos innecesarios o pasar por el mismo estado varias veces.

01 Busqueda NO INFORMADA

Utiliza algoritmos de búsqueda ciega como :

- Búsqueda en profundidad (DFS)
- Búsqueda en anchura (BFS)
- Búsqueda de costo uniforme



Utilizada en problemas de búsqueda simples en los que no se dispone de información específica sobre el espacio de búsqueda.



Busqueda EN ANCHURA



El metodo permite recorrer todos los nodos de un grafo de manera sistemática, comenzando desde un nodo inicial y explorando todos los nodos adyacentes a ese nodo antes de pasar a los nodos adyacentes de los nodos recién explorados.

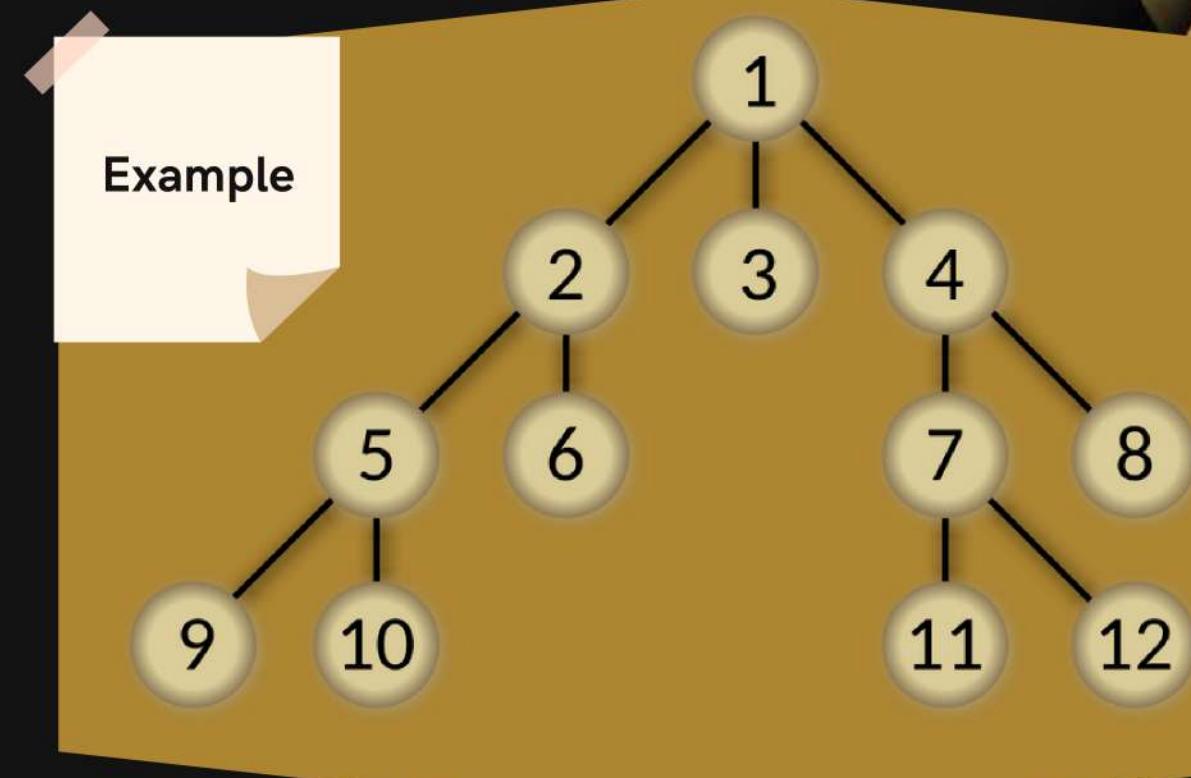
Utiliza una estructura de datos llamada cola para almacenar los nodos adyacentes a medida que se descubren

Complejidad



$$O(b^d)$$

b = número máximo de nodos adyacentes por nodo y
d = es la profundidad del nodo objetivo



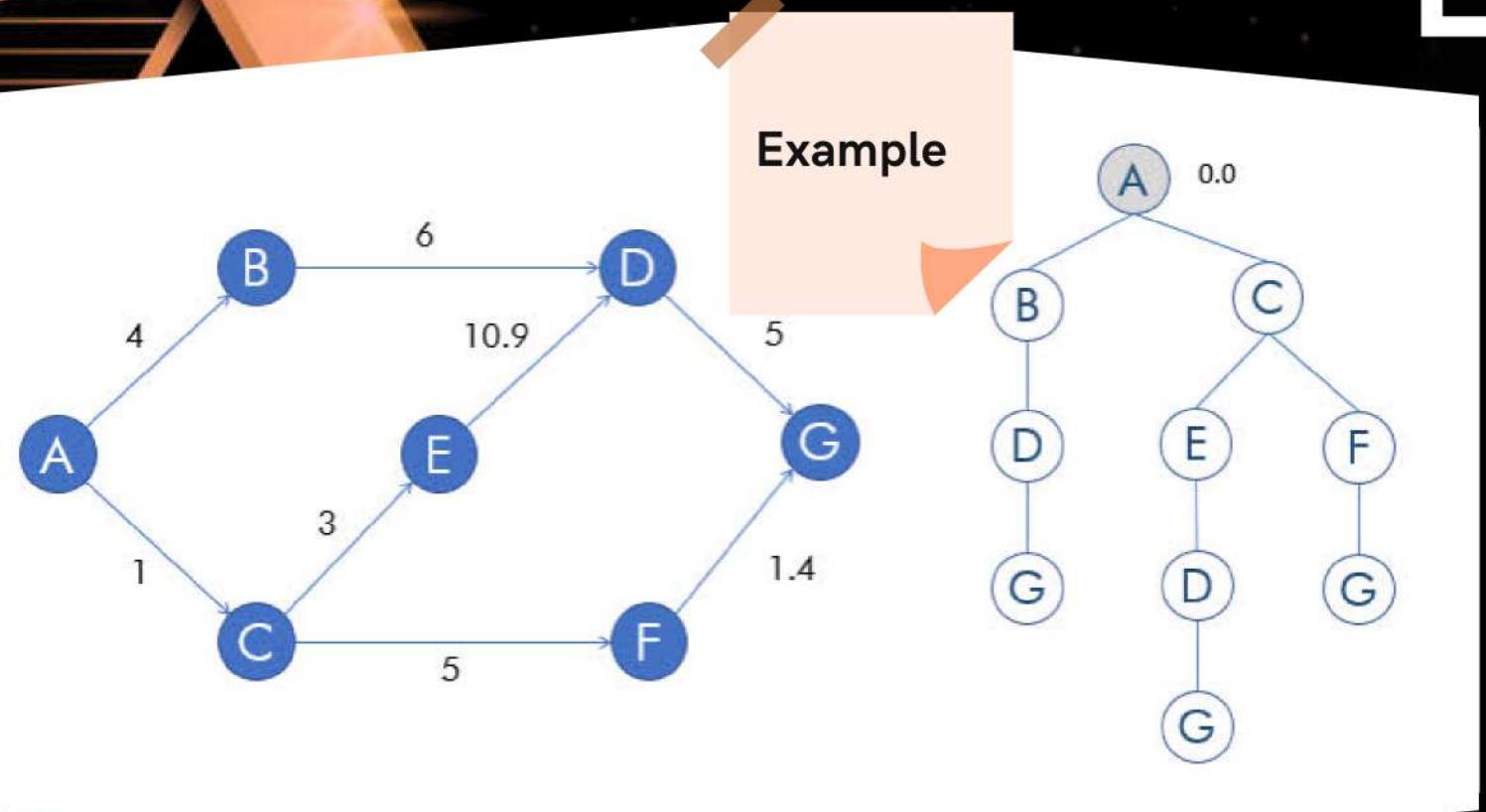
Este metodo suele ser ineficiente en grafos grandes o con muchos nodos adyacentes

EJERCICIO



github.com/ArturoMR14/Bases_AI

Busqueda EN ANCHURA DE COSTO UNIFORME



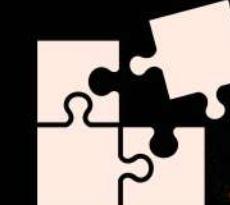
El metodo busca encontrar la ruta de menor costo desde un nodo inicial a un nodo objetivo en un grafo ponderado.



Este metodo suele ser muy lento y consumir mucha memoria en grafos grandes



Siempre encuentra la ruta de menor costo entre el nodo inicial y el objetivo



la busqueda se realiza utilizando una cola de prioridad, en la que se ordenan los nodos según su costo acumulado, de menor a mayor.

Complejidad

$O(b^d)$

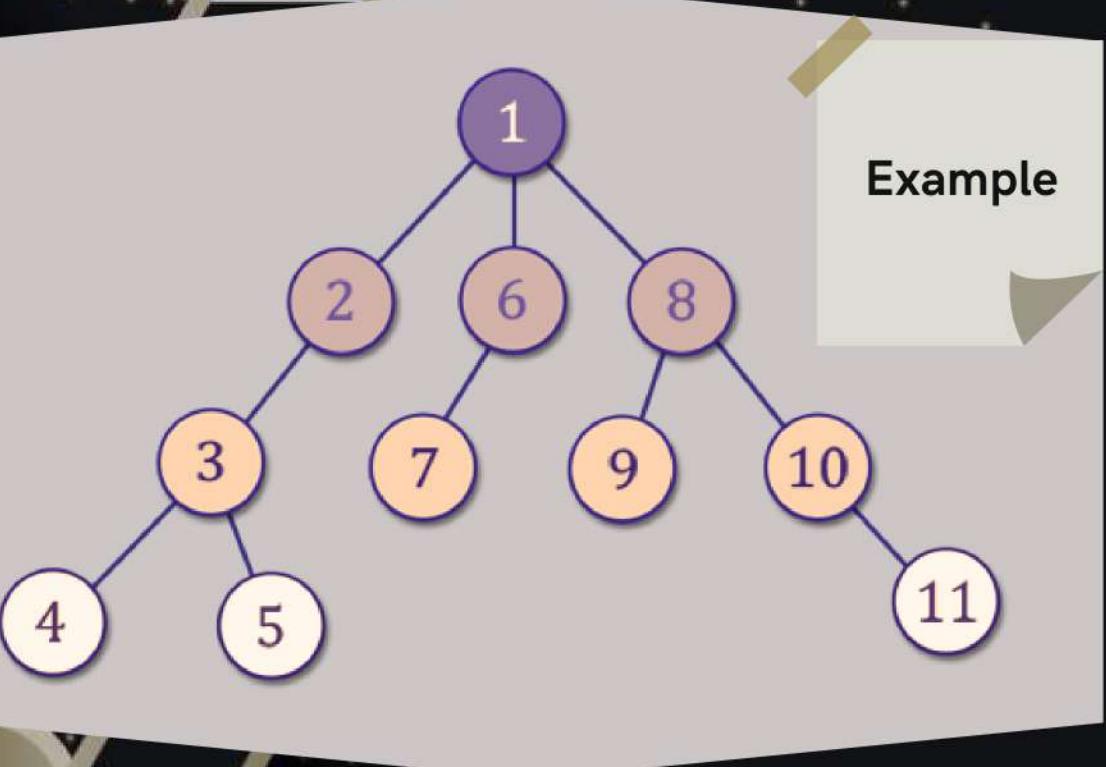
B = factor de ramificación del árbol de b búsqueda
D = profundidad de la solución más cercana.

EJERCICIO



github.com/ArturoMR14/Bases_AI

Busqueda POR PROFUNDIDAD

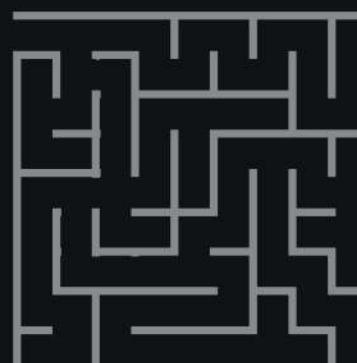


El metodo recorre del nodo inicial al nodo más profundo de la rama actual, una vez que se ha explorado completamente esa rama, retroceder al nodo anterior y continuar con la exploración de la siguiente rama más profunda.



Este modelo no garantiza encontrar la solución óptima ademas que puede caer en bucles infinitos si el grafo tiene ciclos.

la busqueda se realiza mediante una pila (stack) que almacena los nodos a explorar y cuando se expande un nodo, se agregan sus hijos a la pila en el orden inverso al que se desean explorar.



Es eficiente en términos de memoria y puede encontrar soluciones rápidamente en grafos muy profundos.

Complejidad

$O(bm)$

b = factor de ramificación

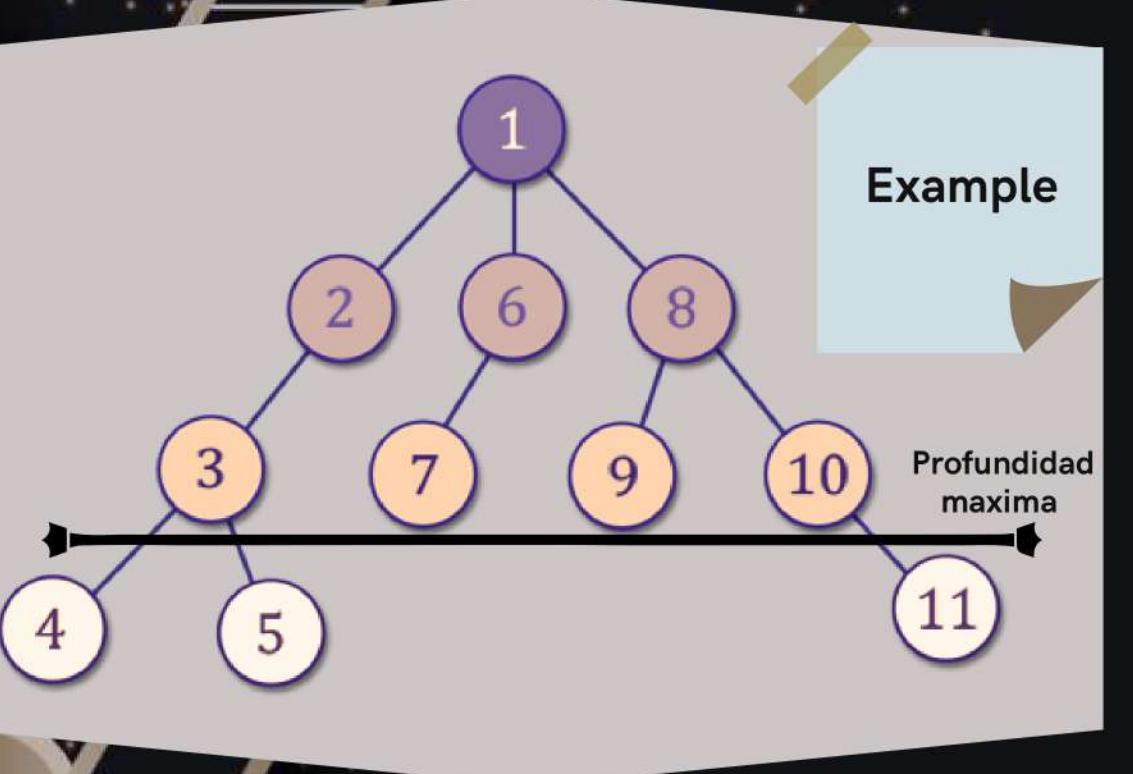
m = profundidad máxima del árbol.

EJERCICIO



github.com/ArturoMR14/Bases_AI

Busqueda POR PROFUNDIDAD LIMITADA



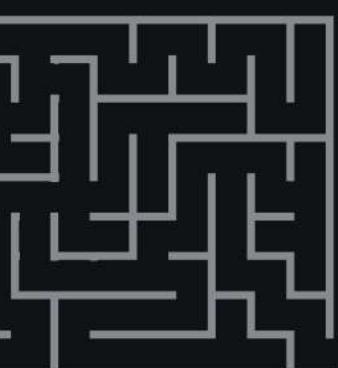
El metodo recorre del nodo inicial al nodo con la profundidad establecida de la rama actual, una vez que se ha explorado esa rama, retroceder al nodo anterior y continuar con la exploración de la siguiente rama más profunda.

El objetivo es limitar el tiempo de ejecución y la memoria utilizada.

Es útil cuando se sabe que la solución se encuentra en un nivel específico del árbol



Si el límite de profundidad establecido es muy pequeño, es posible que se pierda una solución óptima que se encuentre en niveles más profundos



Complejidad

O(bm)

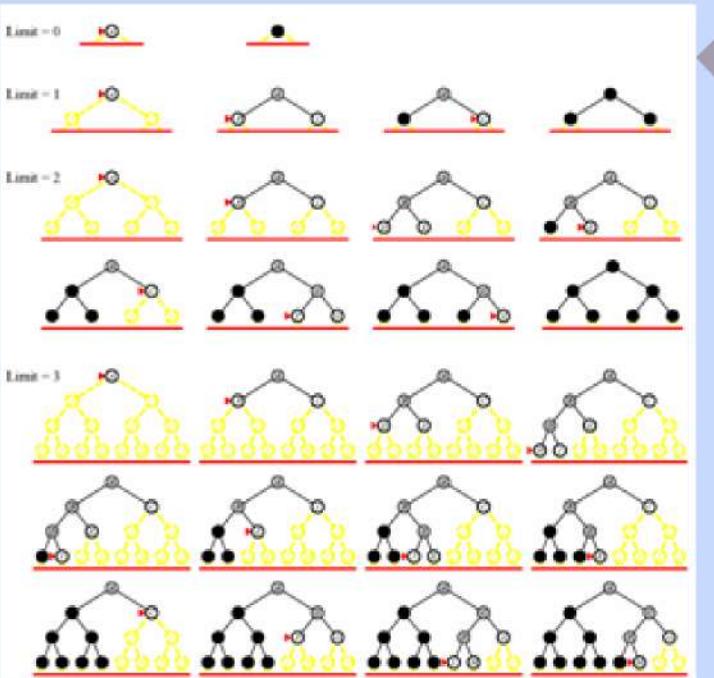
b = factor de ramificación
m = profundidad máxima del árbol.

EJERCICIO



github.com/ArturoMR14/Bases_AI

Busqueda POR PROFUNDIDAD ITERATIVA



Example

El metodo es una técnica de búsqueda incremental que repite la búsqueda en profundidad con diferentes límites de profundidad

Es util para problemas de búsqueda en los que la profundidad de la solución es desconocida y los recursos de memoria son limitados.

Complejidad



$$O(b^d)$$

b = factor de ramificación del arbol
d = profundidad de la solución



dificultad para manejar ciclos
ineficiencia con grafos muy grandes

EJERCICIO



github.com/ArturoMR14/Bases_AI

Busqueda BIDIRECCIONAL

El metodo busca una solución en un espacio de estados explorando desde dos direcciones: una dirección desde el estado inicial y otra desde el estado objetivo.

Puede reducir significativamente el número de nodos que deben ser explorados en la búsqueda, en comparación con la búsqueda unidireccional.

Es necesario definir un criterio de parada, como la detección de una solución o la exploración de un número máximo de nodos.



Requiere un espacio de estado bien definido
Requiere memoria adicional para almacenar el espacio de estado y los estados explorados desde ambos extremos

Complejidad

$$O(b^{d/2})$$

b = factor de ramificación del árbol
d = distancia entre el estado inicial y el estado objetivo

EJERCICIO



github.com/ArturoMR14/Bases_AI

Permite encontrar una solución a un problema utilizando información adicional para guiar la búsqueda hacia el objetivo deseado.

02 Busqueda INFORMADA

Utiliza algoritmos de búsqueda como :

- Búsqueda A*
- Búsqueda voraz.
- Búsqueda de ascension de colinas

Utilizada en problemas grandes o complejos, ya que se enfoca en explorar los nodos que parecen ser más prometedores en términos de llevar a la solución.

Heuristica

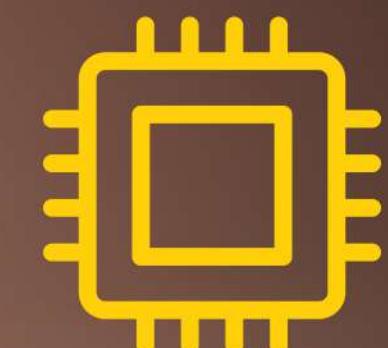
La heurística proporciona una "información" adicional sobre el grafo que se puede utilizar para guiar la exploración hacia la solución de manera más eficiente

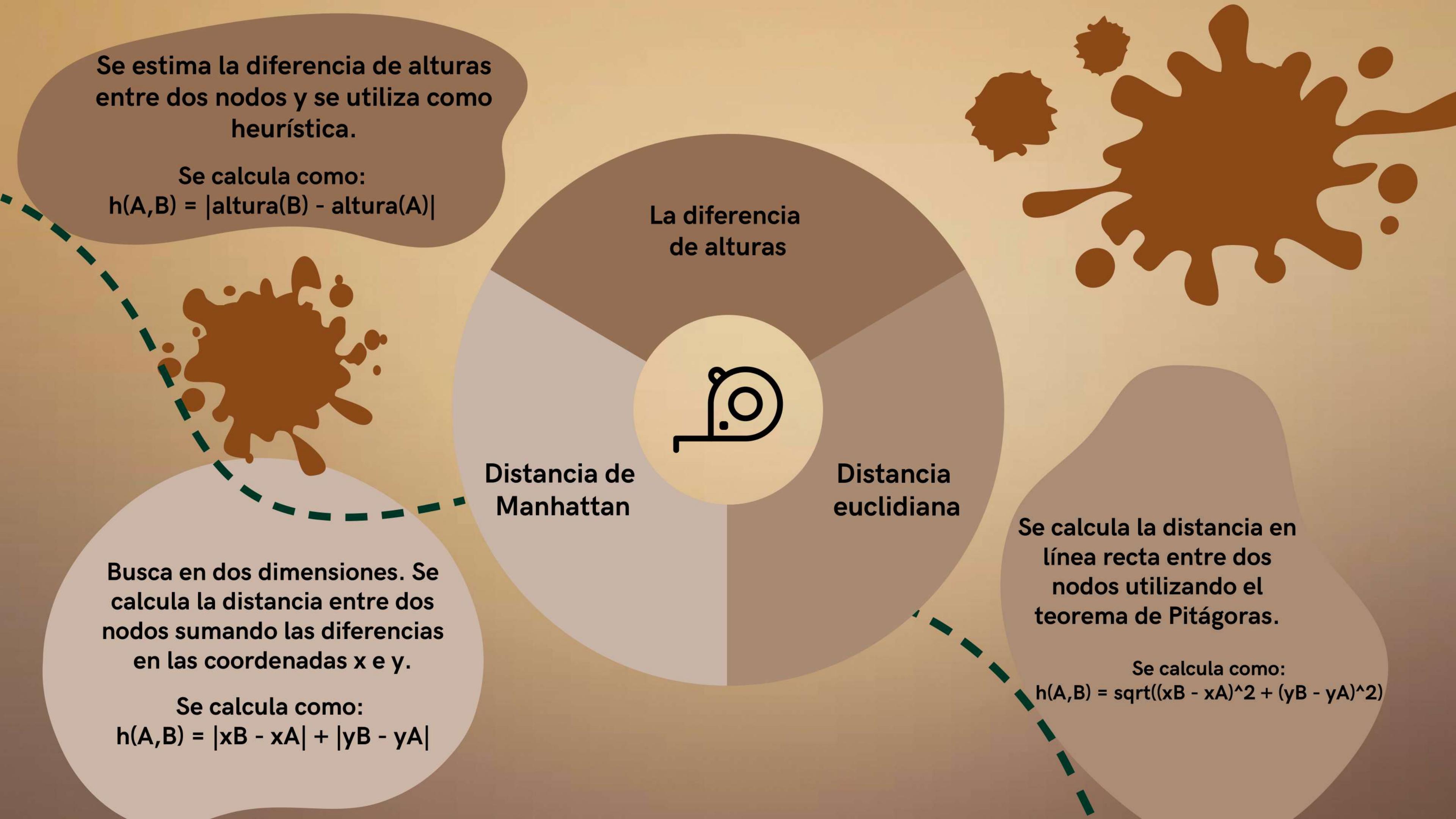


La calidad de la heurística es crucial para el éxito de la búsqueda informada

Algunas Heuristicas serian

Distancia euclíadiana
Distancia de Manhattan.





Se estima la diferencia de alturas entre dos nodos y se utiliza como heurística.

Se calcula como:

$$h(A,B) = |\text{altura}(B) - \text{altura}(A)|$$



La diferencia de alturas



Distancia de Manhattan



Distancia euclíadiana

Busca en dos dimensiones. Se calcula la distancia entre dos nodos sumando las diferencias en las coordenadas x e y.

Se calcula como:

$$h(A,B) = |x_B - x_A| + |y_B - y_A|$$



Se calcula la distancia en línea recta entre dos nodos utilizando el teorema de Pitágoras.

Se calcula como:

$$h(A,B) = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

En algunos problemas de búsqueda pueden tener reglas específicas que pueden utilizarse como heurística.

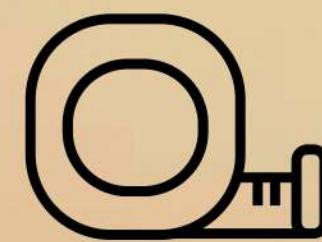
Por ejemplo, en el juego de ajedrez, se puede utilizar el número de piezas restantes de un jugador como heurística



Estima el número mínimo de aristas necesarias para expandir un nodo hasta alcanzar un estado objetivo

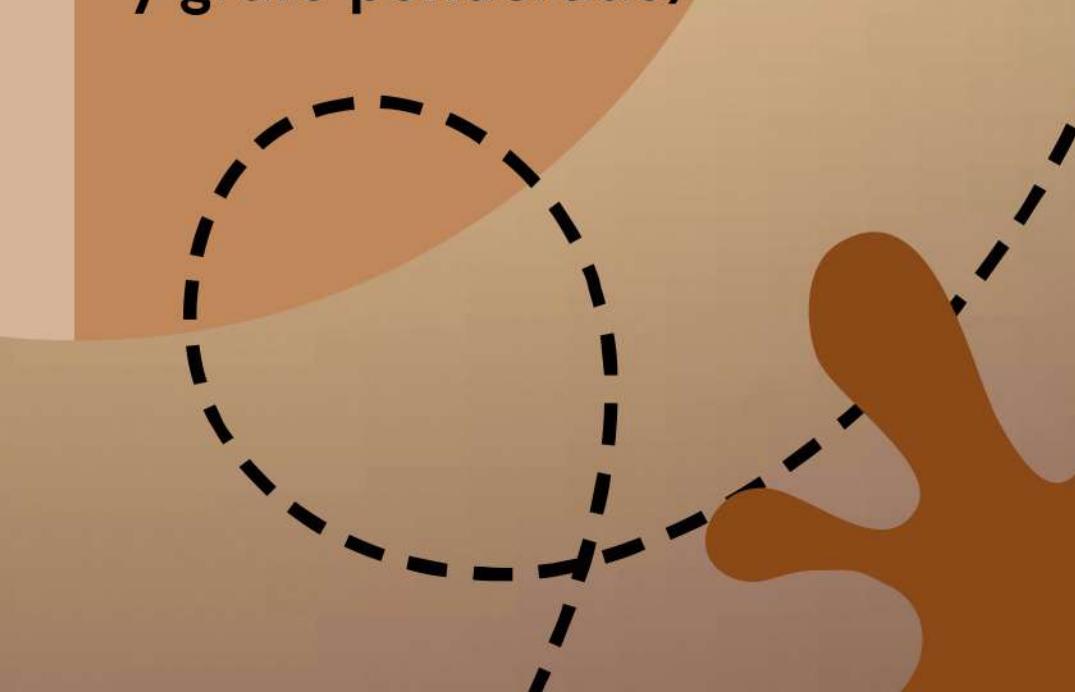
Se calcula como:
 $h(A)$ = número mínimo de aristas necesarias para expandir el nodo A hasta alcanzar un estado objetivo

Mínima expansión de un nodo



Distancia más corta

(grafo no ponderado y grafo ponderado)



Esta heurística simplemente estima la distancia más corta entre dos nodos contando el número de aristas en el camino más corto

NO PONDERADO

Se calcula como:

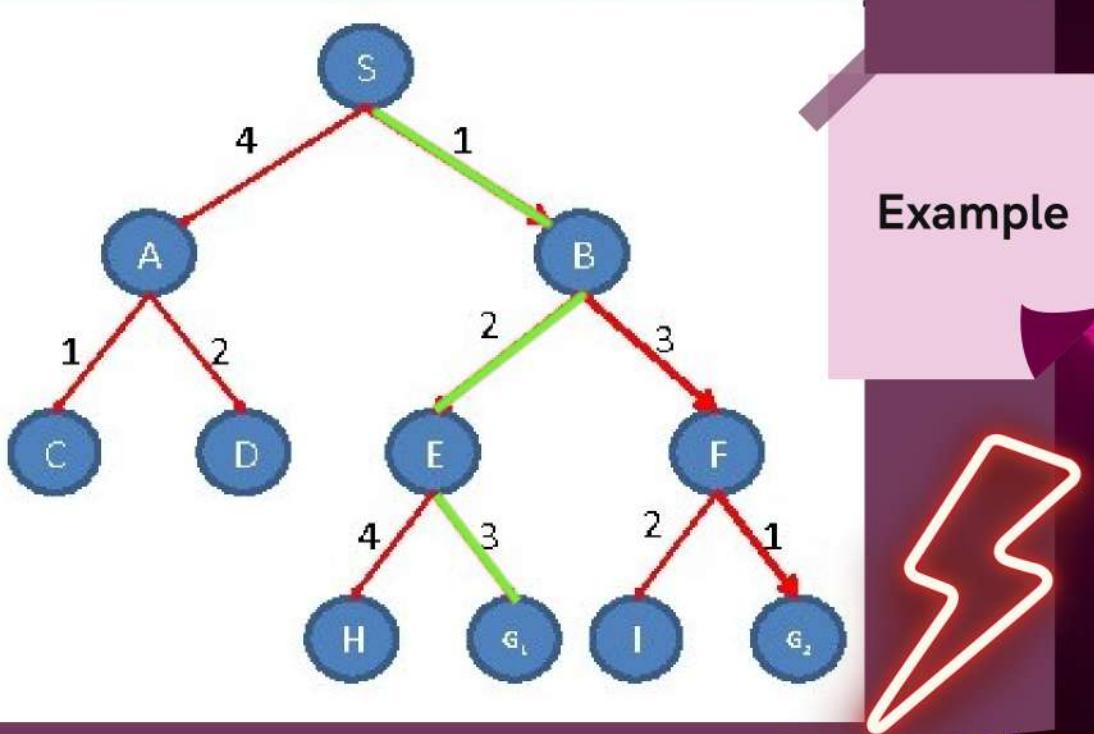
$h(A,B)$ = número de aristas en el camino más corto entre A y B

PONDERADO

Se calcula como:

$h(A,B)$ = suma de los costos de las aristas en el camino más corto entre A y B

Complejidad

$$O(b^d)$$


Búsqueda VORAZ PRIMERO EL MEJOR Best-First Search

b = factor de ramificación del árbol de búsqueda
 m = profundidad máxima del árbol de búsqueda

El método utiliza una función de evaluación heurística para seleccionar el siguiente nodo a expandir en la búsqueda. Esta función de evaluación asigna un valor heurístico a cada nodo, lo que indica cuánto se acerca a la solución óptima

Metodo eficiente

Solo expande el nodo más prometedor en cada iteración. Esto hace que sea mucho más rápido que la búsqueda en anchura o la búsqueda en profundidad, especialmente en problemas grandes.

Utiliza una función de evaluación heurística para seleccionar el mejor nodo a expandir en cada momento

EJERCICIO



github.com/ArturoMR14/Bases_AI

A* combina la búsqueda en profundidad con una heurística que estima el costo de moverse desde el nodo actual hasta el nodo objetivo.

A* utiliza una función de evaluación que combina el costo real del camino desde el nodo inicial hasta el nodo actual $g(n)$, con una heurística $h(n)$ que estima el costo de moverse desde el nodo actual hasta el nodo objetivo. La función de evaluación se define como $f(n) = g(n) + h(n)$.

A* es eficiente en la búsqueda de soluciones óptimas, siempre y cuando la heurística sea admisible y consistente.

Complejidad
 $O(b^d)$

b = factor de ramificación del grafo (es decir, el promedio de nodos sucesores por cada nodo)
 d = profundidad del nodo objetivo más cercano



Búsqueda A*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| 7 | 6 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | | 19 | 20 | 21 | 22 |
| 6 | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | | 18 | 19 | 20 | 21 |
| 5 | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | 17 | 18 | 19 | 20 |
| 4 | 3 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | 16 | 17 | 18 | 19 |
| 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 15 | 16 | 17 | 18 |
| 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | | 14 | 15 | 16 | 17 |
| 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 13 | 14 | 15 | 16 |
| 4 | 3 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | 12 | 13 | 14 | 15 |
| 5 | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 6 | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

Example

Requiere mucho tiempo y memoria para explorar todos los nodos posibles en problemas muy grandes o complejos.

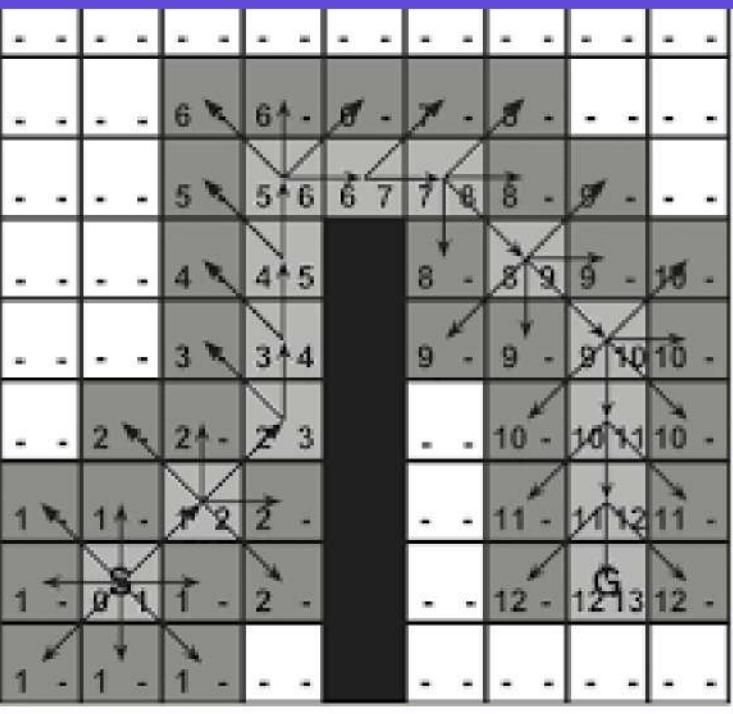


EJERCICIO



github.com/ArturoMR14/Bases_AI

AO* logra reparar soluciones subóptimas. Si se encuentra una solución subóptima, AO* puede continuar buscando soluciones mejores sin tener que reiniciar la búsqueda desde cero.



Example

Búsqueda AO*

Tiene la capacidad de proporcionar soluciones incrementales y su eficiencia en la búsqueda de soluciones en problemas complejos

AO* es un algoritmo anytime, lo que significa que puede proporcionar soluciones en cualquier momento y mejorarlas de forma incremental a medida que se le asigna más tiempo o recursos

Al igual que el método de búsqueda A* La función de evaluación se define como $f(n) = g(n) + h(n)$.

EJERCICIO



github.com/ArturoMR14/Bases_AI

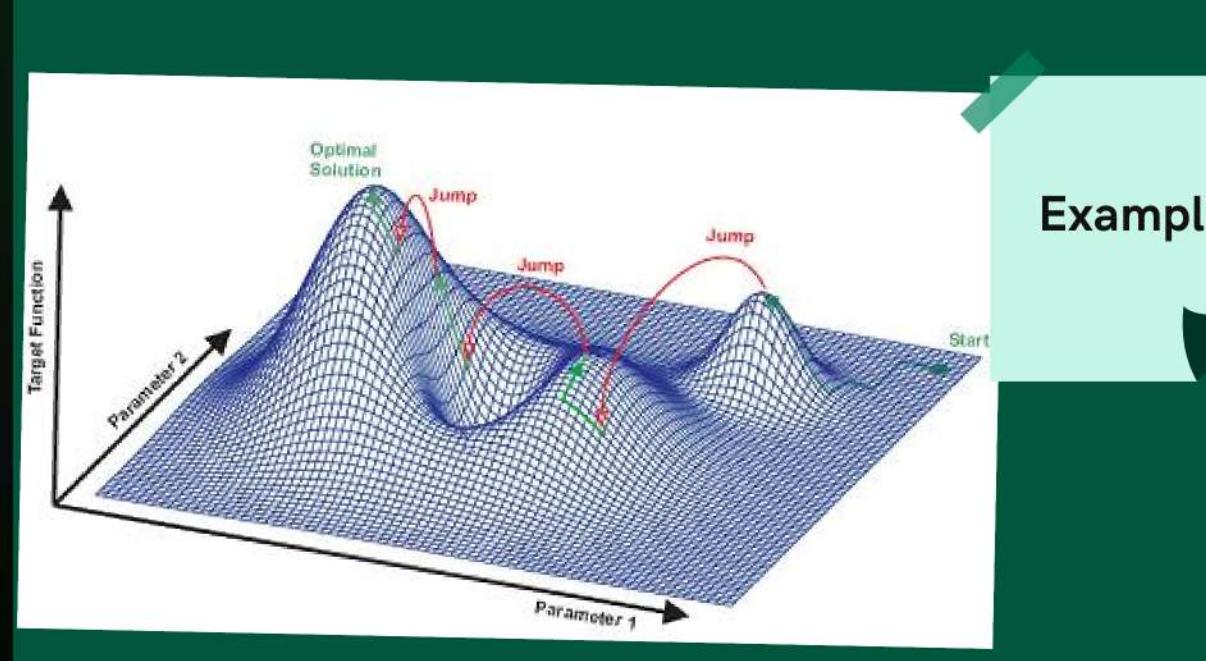
Es un método sencillo y eficaz para encontrar soluciones aproximadas a problemas de optimización.

El algoritmo genera una lista de estados vecinos al estado actual y selecciona el estado con el valor de la función de evaluación más alto (o más bajo, según sea el caso)



Complejidad $O(b^*d)$

b = tamaño del espacio de búsqueda
 d = número de iteraciones necesarias para alcanzar una solución



Example

EJERCICIO

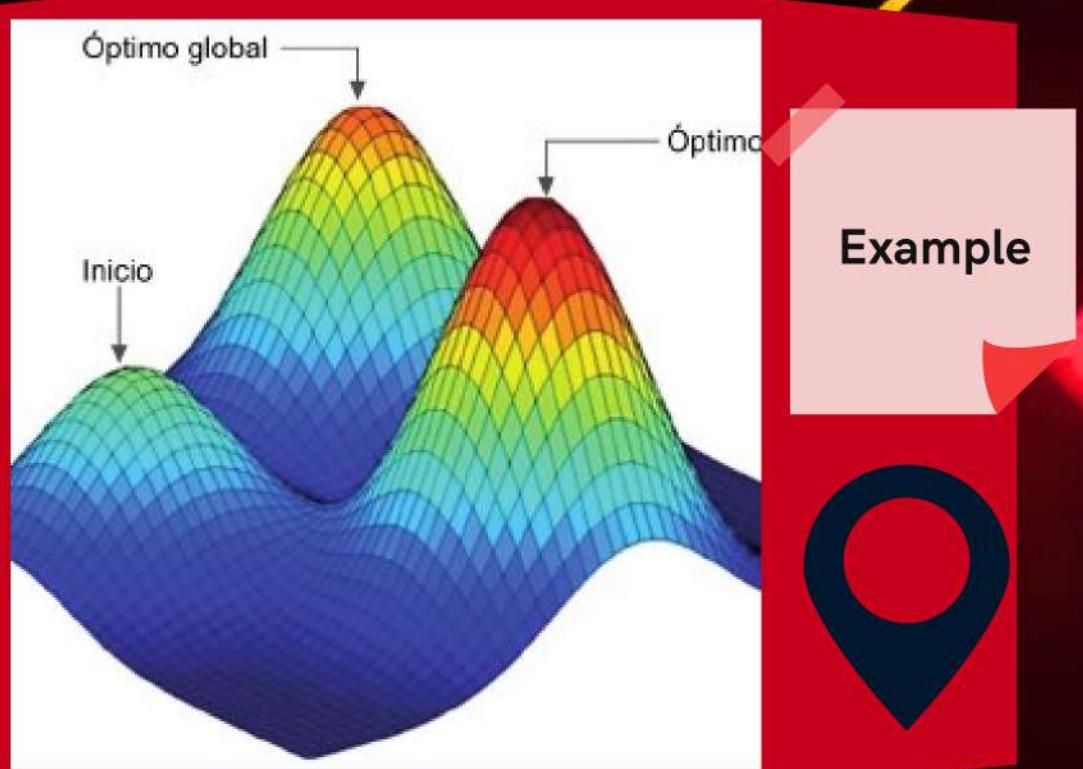
Utiliza una función de evaluación que mide la calidad de una solución. El objetivo es maximizar o minimizar el valor de la función de evaluación, según sea el caso.

A menudo converge a una solución subóptima debido a que el algoritmo puede quedar atrapado en un máximo o mínimo local



github.com/ArturoMR14/Bases_AI

Es un algoritmo de búsqueda local que utiliza una técnica de memoria a largo plazo para evitar quedar atrapado en óptimos locales y así poder explorar un espacio de búsqueda más amplio en busca de soluciones óptimas.



Example



Búsqueda TABU



Es aplicado con éxito en una amplia variedad de problemas de optimización, incluyendo la planificación de la producción, la programación de horarios, la optimización de rutas, el diseño de redes y la asignación de recursos.

Es muy útil para problemas de optimización combinatoria, donde se busca encontrar la mejor combinación de elementos de un conjunto finito de posibilidades

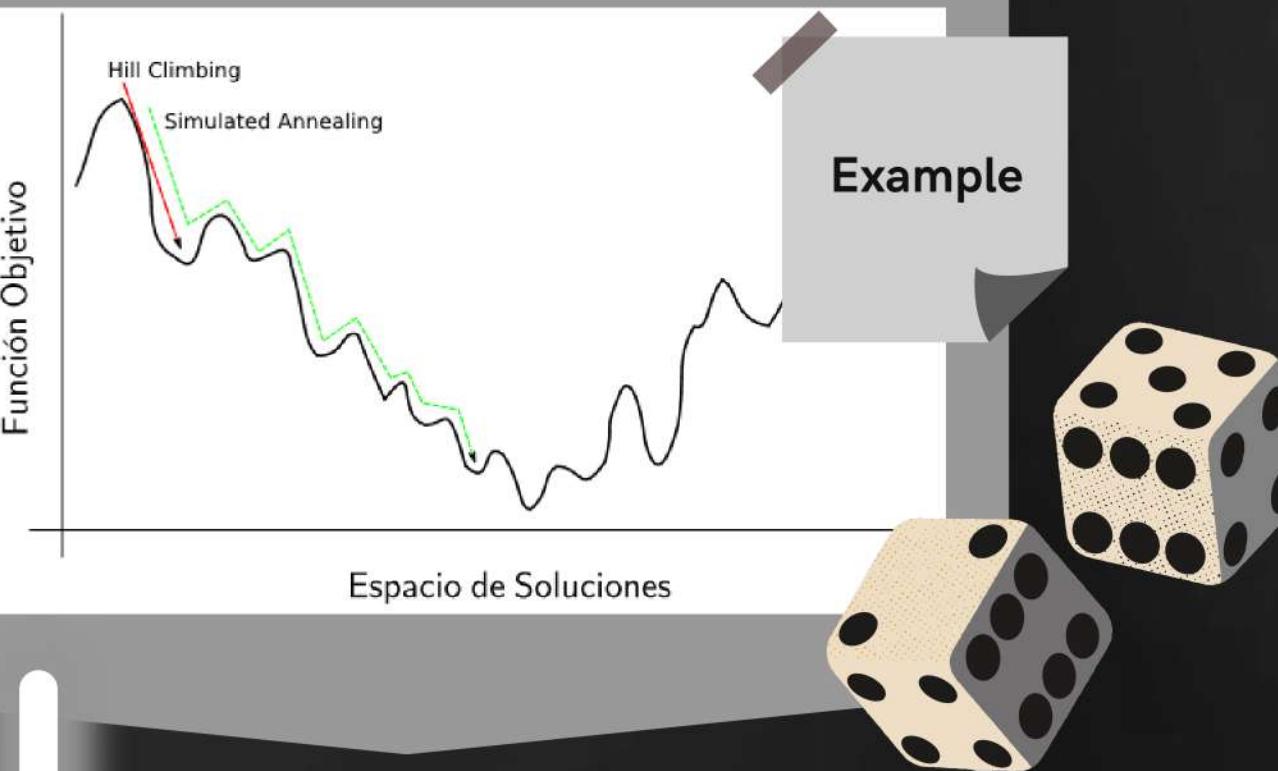
EJERCICIO



github.com/ArturoMR14/Bases_AI

Es un algoritmo de optimización global que se inspira en el proceso físico de enfriamiento de un objeto para encontrar soluciones óptimas a un problema.

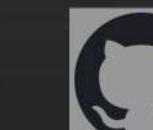
Búsqueda TEMPLE SIMULADO



El método especialmente útil en problemas de optimización que tienen múltiples óptimos locales y que requieren la exploración de un espacio de búsqueda amplio

Es un mecanismo que se utiliza para escapar de los óptimos locales y explorar un espacio de búsqueda más amplio

EJERCICIO



github.com/ArturoMR14/Bases_AI

Búsqueda HAZ LOCAL



Es una técnica de búsqueda local que permite la exploración de un espacio de búsqueda amplio a través de la generación de múltiples soluciones iniciales y la selección de las mejores soluciones en cada iteración

Evalúa utilizando una función objetivo, que mide la calidad de la solución en términos del criterio de optimización específico del problema.

Se genera un conjunto de soluciones iniciales (también conocido como haz) al azar o mediante una heurística

El nuevo haz se compone de las mejores soluciones de los haces generados en la iteración anterior, junto con un conjunto de soluciones vecinas



El método no suele ser adecuado para problemas con restricciones, ya que no garantiza la factibilidad de las soluciones.

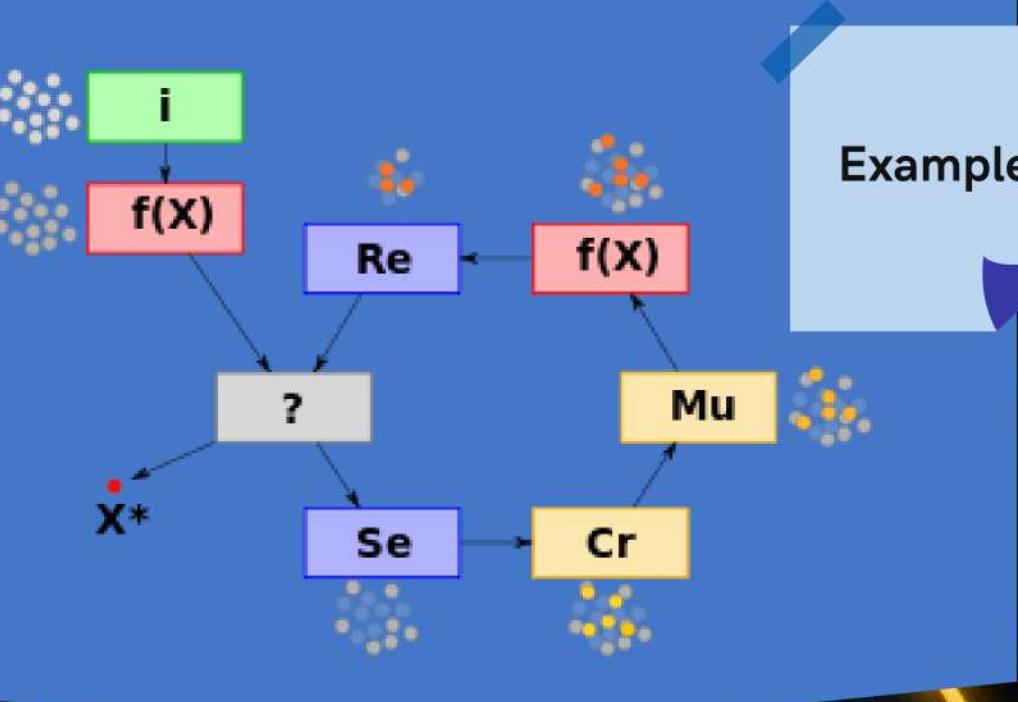
EJERCICIO



github.com/ArturoMR14/Bases_AI

Es una técnica de optimización inspirada en la evolución biológica. Se basa en el concepto de selección natural y la reproducción de individuos para producir soluciones cada vez mejores.

ALGORITMOS GENETICOS



El proceso de selección, reproducción y mutación se repite hasta que se alcanza un criterio de terminación, como un número máximo de generaciones o una mejora suficiente en la aptitud de la población.

En cada generación, se evalúa la aptitud de cada solución de la población en función de una función de aptitud que mide el rendimiento de la solución en el problema

Pueden ser adaptados para resolver una amplia variedad de problemas de optimización, como la selección de características, la clasificación, el enrutamiento y la planificación

EJERCICIO



También conocido como búsqueda incremental, se utiliza para buscar soluciones en un espacio de búsqueda que cambia dinámicamente. En este método, la solución se construye paso a paso, en lugar de generarse de una sola vez.



Se lleva a cabo en iteraciones, en las que se evalúa la solución actual en función de algún criterio de evaluación y se realizan ajustes incrementales para mejorarla.

Búsqueda ONLINE



Depende en gran medida de la elección de los ajustes incrementales. Es importante elegir los ajustes que mejoren la solución en lugar de empeorarla o dejarla sin cambios.

La solución se construye paso a paso y se realizan ajustes incrementales a medida que se reciben nuevas informaciones

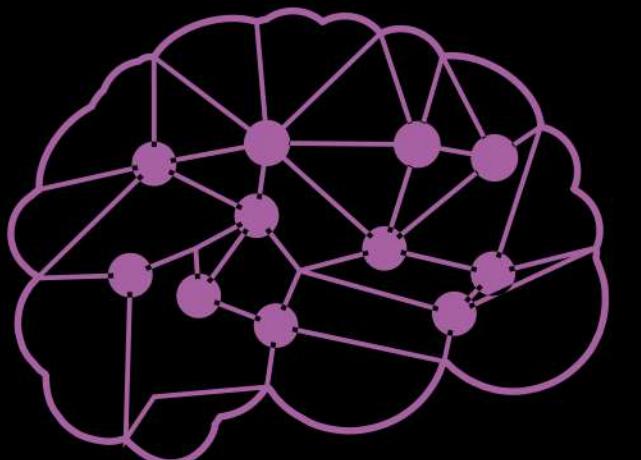
EJERCICIO



github.com/ArturoMR14/Bases_AI

03 Satisfaccion de restricciones

Es una técnica utilizada en inteligencia artificial para resolver problemas en los que existen varias restricciones que deben ser cumplidas al mismo tiempo

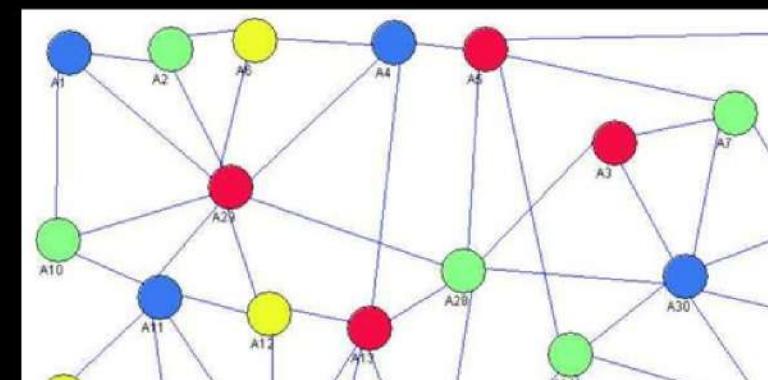


Se basa en la creación de un modelo matemático que representa el problema en términos de variables y restricciones. Cada variable representa una parte del problema que puede tomar diferentes valores y cada restricción representa una condición que debe ser cumplida por una combinación particular de valores de las variables.



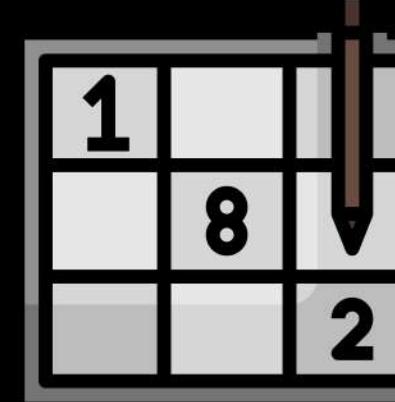
Problema del mapa de colores

Consiste en colorear un mapa con diferentes colores de manera que dos regiones adyacentes no tengan el mismo color



Problema de sudoku

Consiste en completar una cuadrícula de 9x9 casillas con números del 1 al 9, de manera que cada número aparezca solo una vez en cada fila, columna y región de 3x3 casillas



Satisfaccion de restricciones

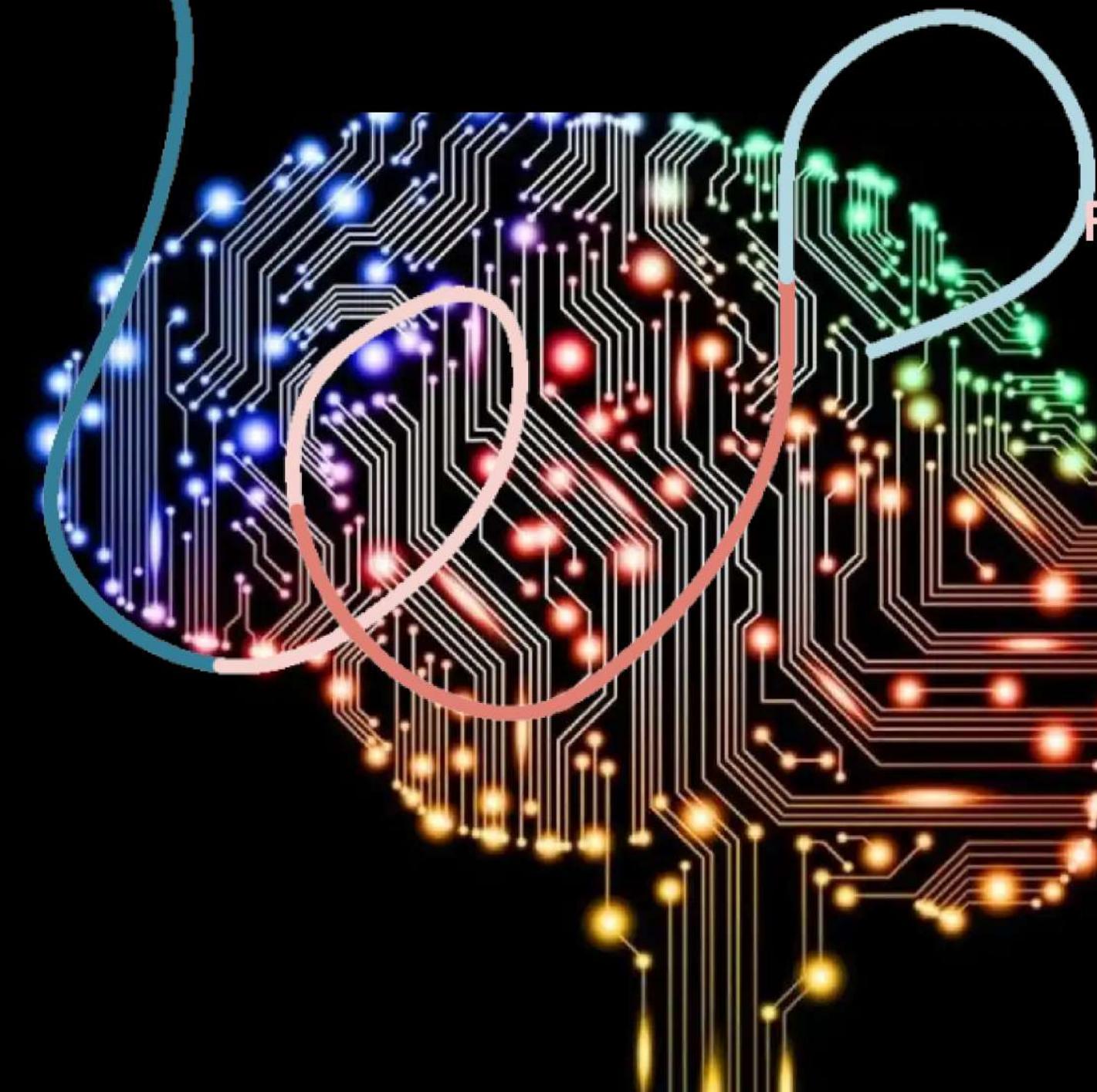
Problema de planificación de rutas

Consiste en planificar una ruta óptima para un conjunto de vehículos que deben visitar diferentes lugares, teniendo en cuenta las restricciones de tiempo, capacidad y distancia



Problema de asignación de recursos

Consiste en asignar recursos limitados a diferentes tareas o proyectos, teniendo en cuenta las restricciones de disponibilidad y de uso de los recursos



Busqueda VUELTA ATRAS



El metodo hace la exploración de todas las posibles combinaciones de valores que pueden ser asignados a las variables del problema, y luego verificar si se satisfacen todas las restricciones

Si una combinación de valores no cumple con las restricciones, se "vuelve atrás" (backtrack) y se prueba otra combinación



El metodo resulta muy costoso computacionalmente, ya que el número de combinaciones de valores a probar puede ser muy grande

| | | | | | |
|---|---|---|---|---|---|
| 5 | 3 | | 7 | | |
| 6 | | 1 | 9 | 5 | |
| | 9 | 8 | | | 6 |
| 8 | | | 6 | | 3 |
| 4 | | 8 | 3 | | 1 |
| 7 | | | 2 | | 6 |
| | 6 | | | 2 | 8 |
| | | 4 | 1 | 9 | |
| | | | 8 | | 7 |
| | | | | | 9 |

Example

EJERCICIO



github.com/ArturoMR14/Bases_AI

COMPROBACION HACIA ADELANTE

Realiza un seguimiento de los valores que se han asignado a las variables y las restricciones que se han impuesto en cada paso de la búsqueda. De esta manera, se puede evitar explorar opciones que ya se sabe que no conducen a una solución válida.

| | | |
|---|--|---|
| | | 2 |
| 6 | | 9 |
| 4 | | 8 |

| | | |
|---|--|---|
| | | 2 |
| 6 | | 9 |
| 4 | | 8 |

El proceso comienza con la selección de una variable que se debe asignar un valor, y luego se prueba un valor para esa variable.

- Se comprueba si el valor seleccionado cumple con todas las restricciones del problema.
- Se comprueba si aún hay valores posibles para las variables restantes que cumplan con las restricciones.

Si el valor seleccionado cumple con las condiciones, procede a la siguiente variable y se repite el proceso. Si no hay valores posibles para las variables restantes, se realiza un "backtrack" o vuelta atrás y se prueba con otro valor para la variable anterior.

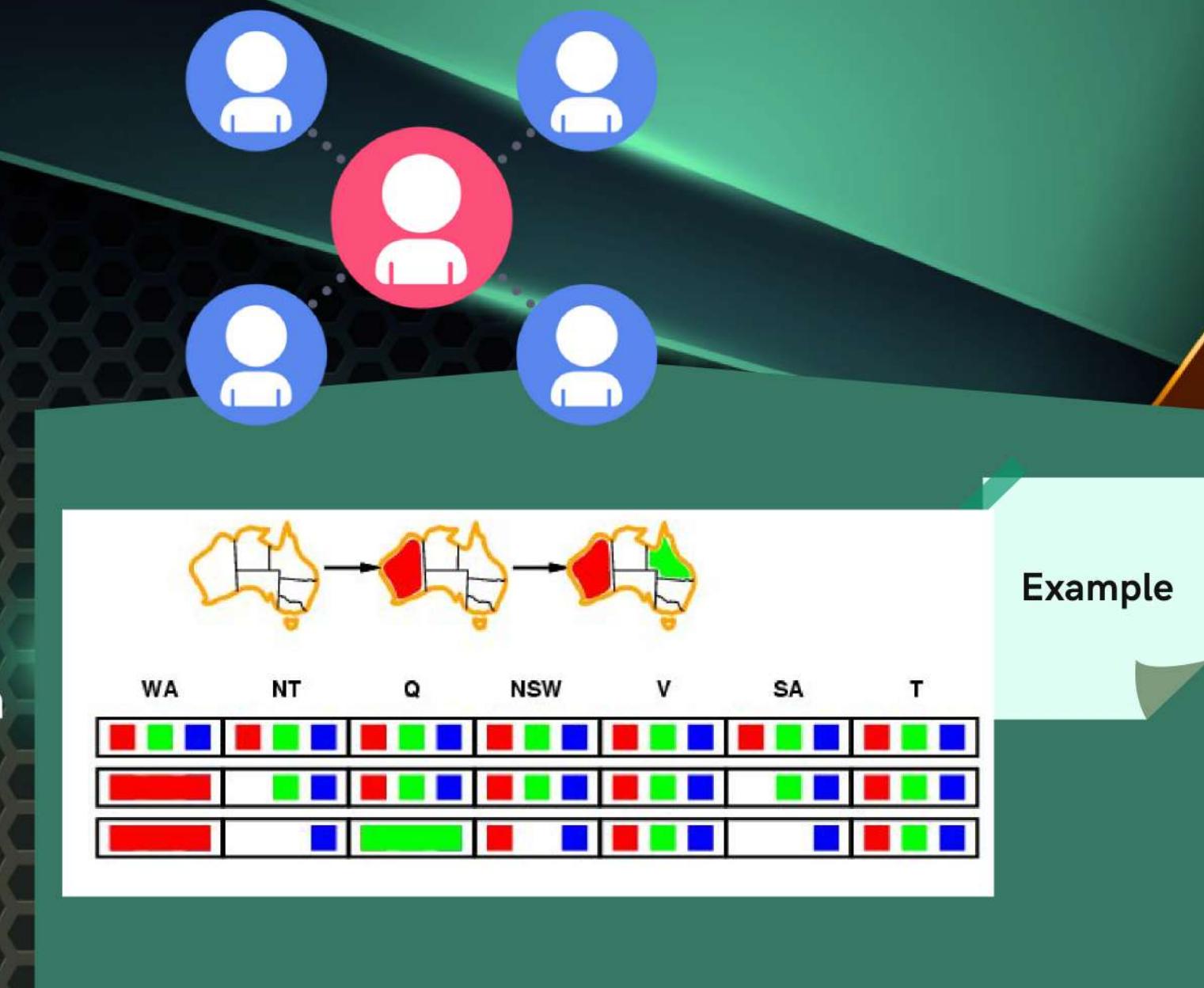
EJERCICIO



github.com/ArturoMR14/Bases_AI

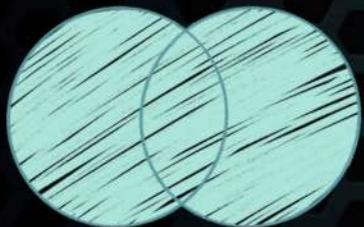
PROPAGACION DE RESTRICCIONES

Este método se basa en la propagación de información sobre las restricciones del problema para reducir el espacio de búsqueda y encontrar soluciones más rápido.



Example

Ayuda a reducir significativamente el espacio de búsqueda de soluciones y, por lo tanto, acelerar la búsqueda de una solución válida



Puede combinarse con otros métodos de búsqueda, como la búsqueda de vuelta atrás o la comprobación hacia delante



EJERCICIO



github.com/ArturoMR14/Bases_AI

SALTO ATRAS POR CONFLICTOS

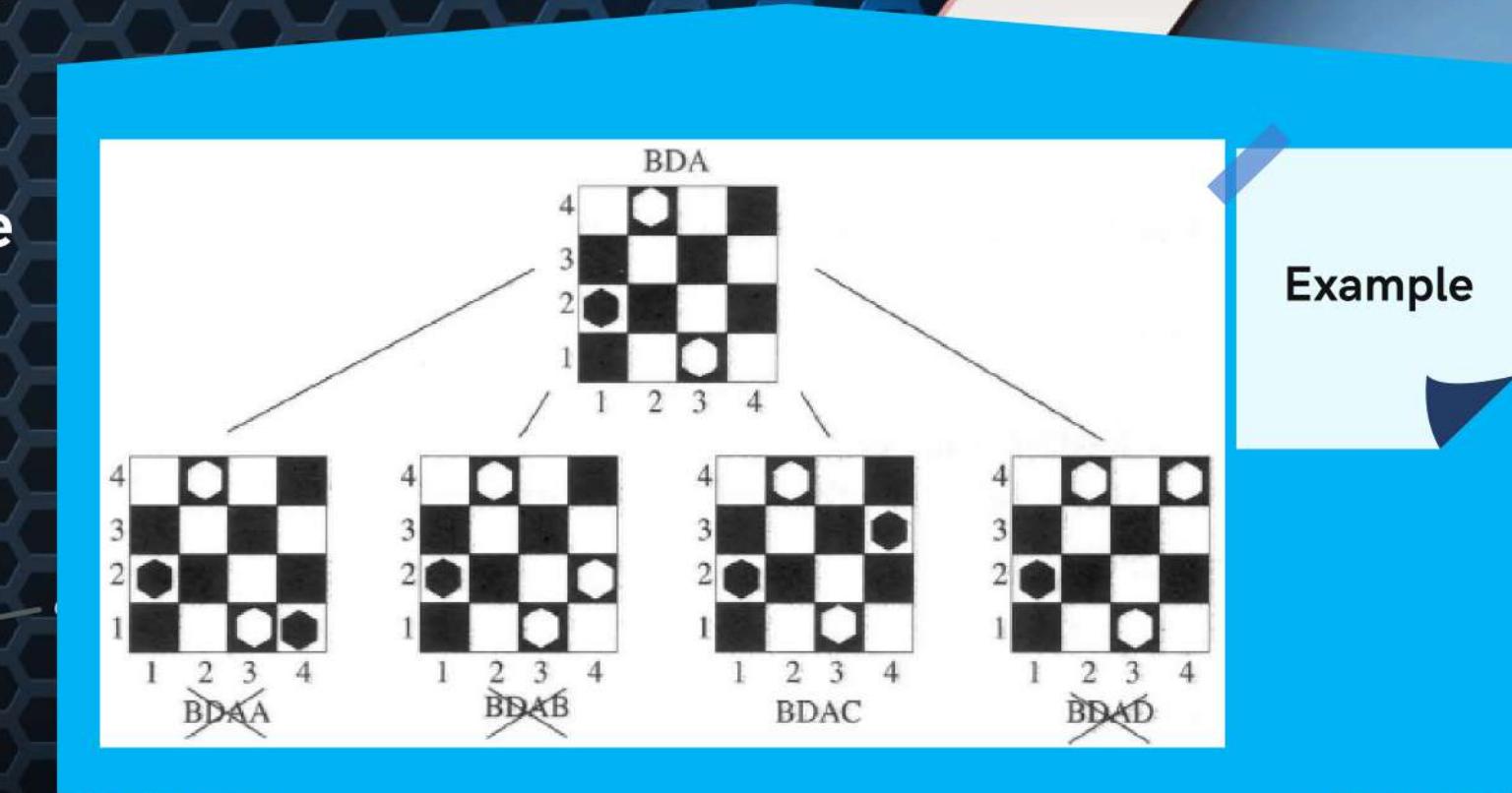
conflict-directed backjumping



El metodo realiza una búsqueda de vuelta atrás (backtracking) en el espacio de soluciones, pero en lugar de retroceder a la última decisión tomada, se retrocede a una decisión anterior que causó el problema.

Esto se hace mediante el seguimiento de una estructura de árbol de búsqueda, donde cada nodo representa una asignación de variable

Reduce significativamente el espacio de búsqueda de soluciones al retroceder a una decisión anterior que es más prometedora en términos de encontrar una solución válida.



EJERCICIO



github.com/ArturoMR14/Bases_AI

BUSQUEDA LOCAL MINIMOS-CONFLICTOS

Minimum-Conflict Heuristic

Se parte de una solución inicial y se van cambiando los valores de las variables que más restricciones violan, hasta que se llega a una solución donde no se viola ninguna restricción

No requiere de la exploración de un gran espacio de búsqueda de soluciones. Además, es utilizado para encontrar soluciones factibles cuando el problema tiene un gran número de restricciones y las soluciones óptimas son muy difíciles de encontrar.



No siempre garantiza la obtención de una solución óptima global

EJERCICIO



github.com/ArturoMR14/Bases_AI



La teoría de juegos es un campo de las matemáticas y la economía que estudia cómo las personas toman decisiones en situaciones estratégicas, a estas situaciones les llamamos juegos



Puede llegar a ser ineficiente en algunos casos, ya que puede explorar caminos innecesarios o pasar por el mismo estado varias veces.

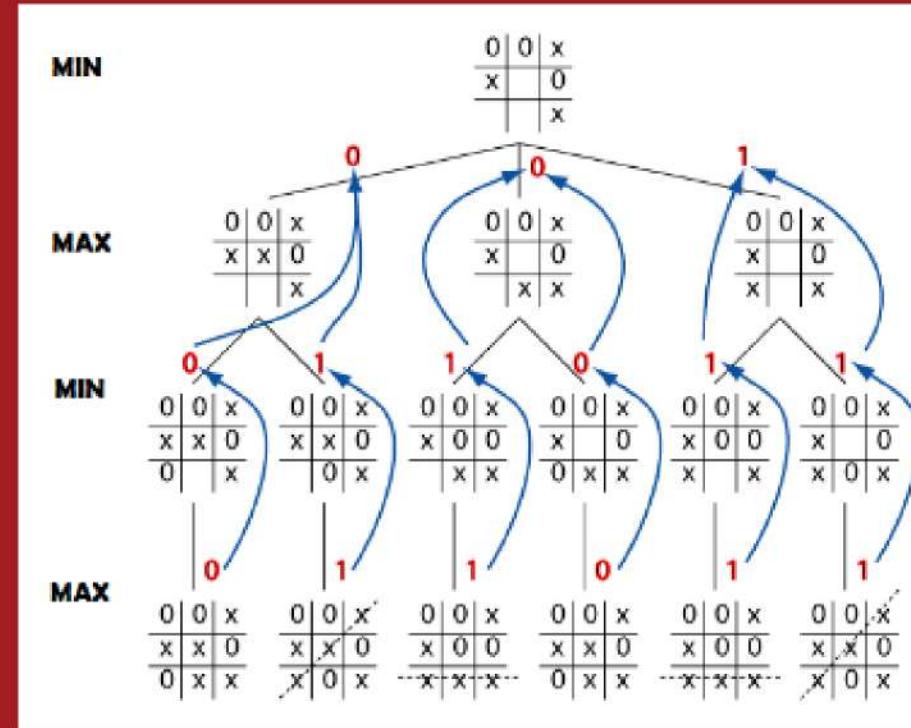
03 JUEGOS

START

Se analizan los posibles resultados de cada jugador en un juego y cómo se relacionan entre sí. Esto se hace a través de la identificación de los jugadores, las opciones que tienen y las posibles consecuencias de sus decisiones

</>

Los juegos se representan de varias maneras, la más común es a través de la matriz de pagos. Esta matriz muestra los resultados para cada jugador en función de las decisiones que tomen los otros jugadores.



Example

Es un algoritmo utilizado para determinar la mejor estrategia en un juego de dos jugadores de suma cero

MINIMAX



El objetivo de un jugador es maximizar su ganancia y el objetivo del otro jugador es minimizar la ganancia del primer jugador

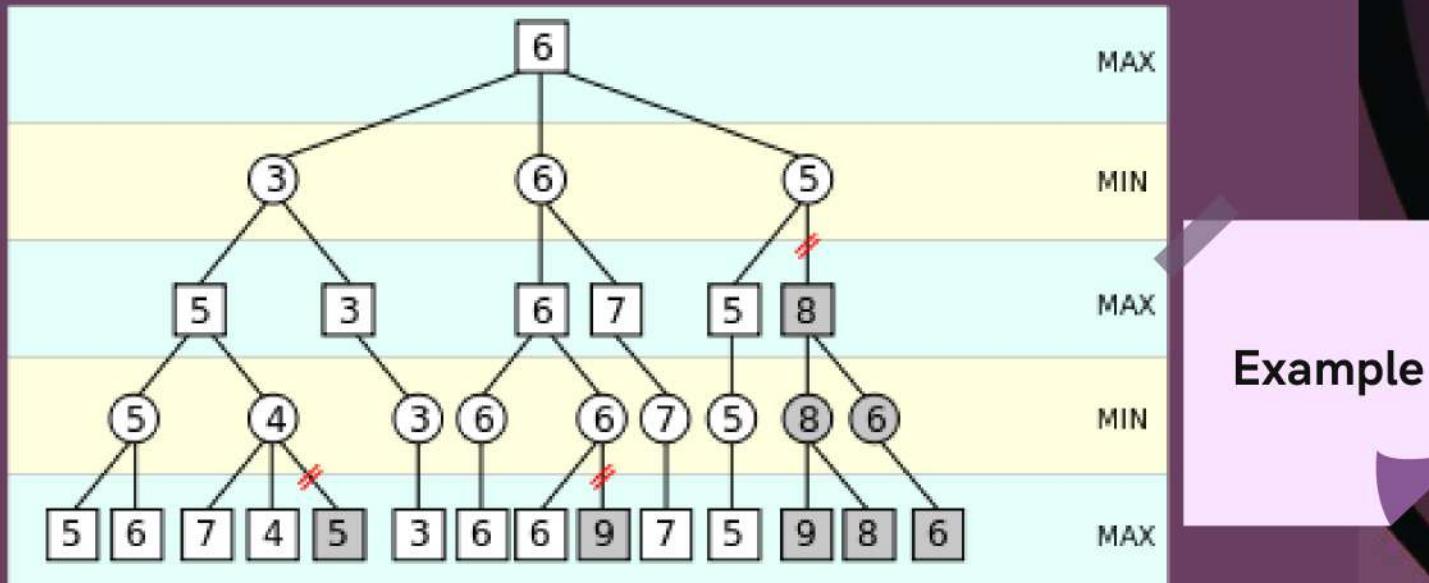
Cada jugador asume que el otro jugador siempre tomará la mejor decisión para sí mismo, y utiliza esta información para elegir su propia estrategia

proporciona una forma sistemática de analizar situaciones en las que los resultados de las decisiones están influenciados por las decisiones de otros agentes

EJERCICIO

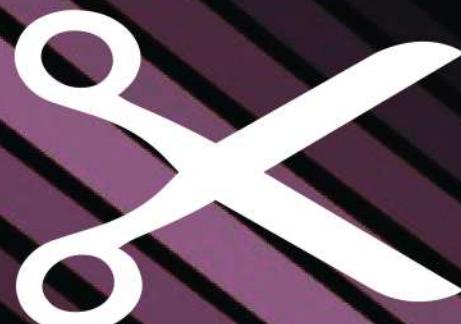


github.com/ArturoMR14/Bases_AI



En cada nodo del árbol de búsqueda, se evalúan las opciones disponibles y se aplica el algoritmo de poda alfa-beta para decidir si es necesario continuar explorando los nodos descendientes

PODA ALFA-BETA

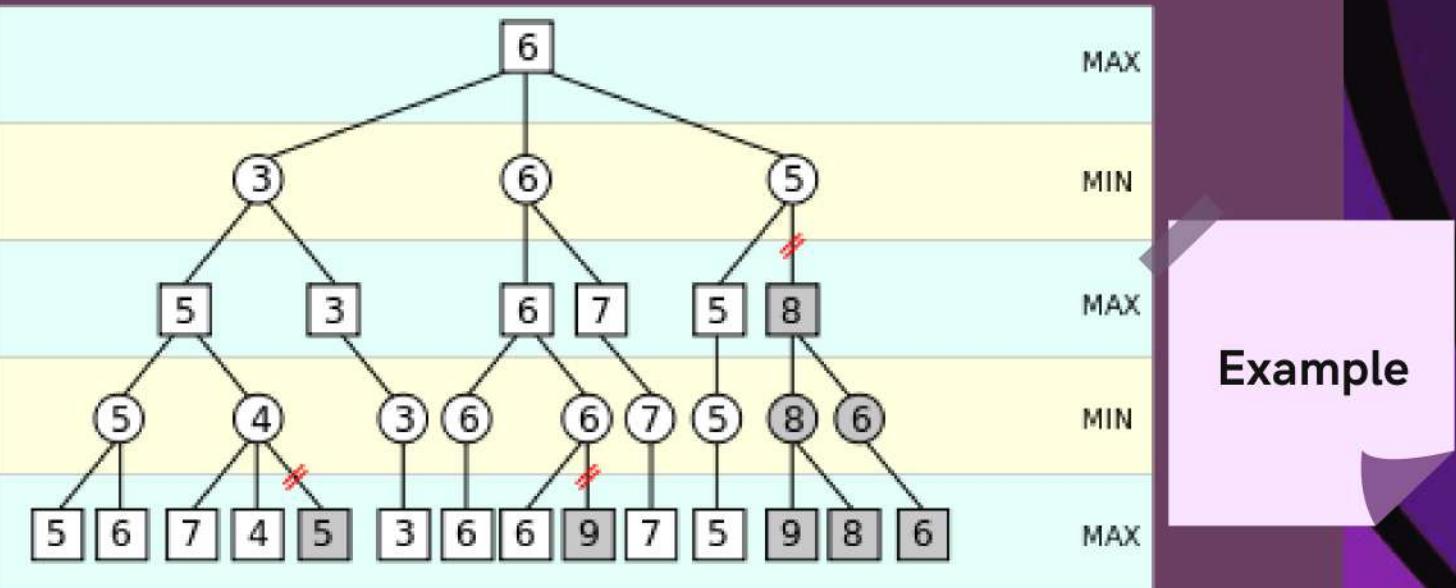


Este método es particularmente útil en juegos complejos, como el ajedrez, donde la cantidad de posibles movimientos puede ser muy grande.

Es un algoritmo eficiente que permite encontrar la solución óptima en un espacio de búsqueda de manera más rápida que el método minimax tradicional.

Alfa representa el mejor valor (la máxima ganancia) que el jugador maximizador ha encontrado hasta el momento, mientras que **beta** representa el peor valor (la mínima ganancia) que el jugador minimizador ha encontrado hasta el momento.

EJERCICIO



Example

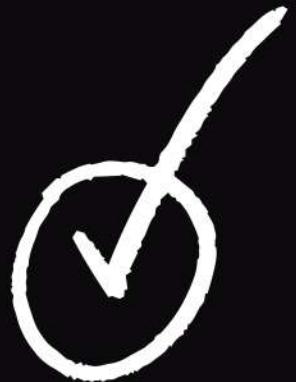
La función de evaluación asigna un valor numérico a un estado del juego, lo que permite al algoritmo de búsqueda evaluar la calidad de cada posible movimiento.



FUNCION DE EVALUACION

La función de evaluación toma en cuenta varios factores que pueden afectar la calidad de un estado del juego, como la posición de las piezas, la cantidad de piezas, la cantidad de movimientos posibles, etc.

Asigna una puntuación numérica a una posición de juego en función de la calidad de la posición para un jugador en particular



EJERCICIO



github.com/ArturoMR14/Bases_AI



EFECTO HORIZONTE

El efecto horizonte es el problema que surge cuando un algoritmo de búsqueda en juegos no puede considerar todos los posibles movimientos y estados futuros más allá de un cierto horizonte o profundidad en el árbol de búsqueda. Esto se debe a las limitaciones computacionales y de tiempo, ya que el número de posibles estados y movimientos en un juego puede crecer exponencialmente.

El corte de búsqueda efecto horizonte es una técnica que permite al algoritmo de búsqueda detener la evaluación de posibles movimientos después de una cierta profundidad en el árbol de búsqueda

El efecto horizonte puede ser problemático, ya que el algoritmo puede tomar decisiones subóptimas debido a la falta de información completa sobre el estado del juego

EJERCICIO



github.com/ArturoMR14/Bases_AI



MINIMAX ESPERADO

En cada nivel del árbol de búsqueda, se consideran todas las acciones posibles y se calcula la probabilidad de que ocurran y los posibles resultados asociados. Luego, se calcula la evaluación esperada para cada movimiento, utilizando la suma ponderada de los resultados posibles y sus probabilidades.

El algoritmo Minimax Esperado es una variante del algoritmo Minimax que se utiliza en juegos con incertidumbre, donde las acciones y sus resultados pueden tener cierta probabilidad asociada. En lugar de considerar solo los mejores y peores resultados posibles, el Minimax Esperado considera la esperanza matemática de los resultados de los movimientos posibles

Se utilizan técnicas de teoría de decisiones y probabilidad para calcular la evaluación de los estados del juego, es decir busca el movimiento que maximiza la ganancia esperada o minimiza la pérdida esperada

EJERCICIO



github.com/ArturoMR14/Bases_AI