Tema-3.pdf

Blublu__3

Inteligencia Artificial

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

TEMA 3

3.1 Búsqueda sin información

Estas búsquedas escogen el siguiente nodo a explorar, una vez generados los sucesores, de forma ciega siguiendo siempre la misma táctica. Vamos a ver 4 técnicas de búsqueda sin información básicas:

3.1.1 Búsqueda en anchura

Este algoritmo es un algoritmo de búsqueda basado en grafos. Sus principales características son las siguientes:

- **Completo:** lleva a cabo un proceso sistemático por niveles que encuentra la solución si existe y si el sistema de ramificación es finito en cada nodo.
- **Optimalidad**: la primera solución que encuentra es óptima si todos los operadores tienen el mismo coste y encuentra esa solución óptima porque va haciendo una exploración por niveles por lo tanto la primera solución que encuentre siempre va a ser desde el punto de vista del coste siempre va a ser la más barata porque si hubiera otra la hubiera encontrado antes.
- **Eficiencia**: no es muy eficiente menos cuando la meta esta cercana. Si la meta está lejana no lo será porque el uso de memoria será exponencial.

3.1.2 Búsqueda con costo uniforme

Es una búsqueda en anchura con la diferencia de que cada nodo guarda información sobre el coste del camino desde el estado inicial hasta ese nodo y que la lista de abiertos es una cola con prioridad.

3.1.3 Búsqueda en profundidad

Es igual que una búsqueda en anchura cambiando la política de selección de nodos en abiertos de FIFO a LIFO. Este algoritmo es un algoritmo de búsqueda basado en grafos.

3.1.4 Búsqueda en profundidad retroactiva

En esta búsqueda no se utilizan listas para almacenar nodos ni empleamos un bucle. Vamos a utilizar recursividad. Sus características son las siguientes:

- Completitud: no asegura encontrar la solución porque puede que en un problema en el que haya ciclos este algoritmo entre en un bucle infinito.
- Optimalidad: no asegura encontrar una solución óptima (no asegura ni encontrar solución).
- Eficiencia: bueno cuando las metas están alejadas del estado inicial, o hay problemas de memoria.

WUOLAH

3.1.5 Búsqueda por descenso iterativo.

Combina las virtudes de la búsqueda en anchura con la búsqueda en profundidad retroactiva. El adjetivo iterativo se debe a que aplica iterativamente búsqueda en profundidad retroactiva con un límite de profundidad que se aumenta en cada iteración. En cada iteración la búsqueda arranca siempre desde el nodo inicial y se exploran todos los nodos del nivel correspondiente a la iteración. Por tanto en cada iteración se vuelven a revisar todos los nodos de la iteración anterior.
Esto no es un proceso costoso porque no almacena los nodos visitados y porque en un árbol balanceado la mayor parte de los nodos estarán en los niveles inferiores.

3.1.6 Búsqueda bidireccional

Integra búsqueda desde el estado inicial con búsqueda desde el estado final con la esperanza de que al final se puedan encontrar. La búsqueda desde el estado inicial se denomina búsqueda progresiva en la que se aplican operadores hacia delante y los sucesores de un nodo son estados transformados por esos operadores. En el razonamiento regresivo los sucesores de un nodo son los estados previos desde los que se puede llegar a ese nodo. En este algoritmo se reduce en gran medida el espacio de búsqueda si se usan ambas técnicas por separado y en paralelo. Si tenemos un factor de ramificación b y una profundidad d la complejidad especial será O(b^d) que es mucho mayor que si sumamos las complejidades de búsqueda que se pueden obtener de los algoritmos por separado (O(b^d/2) + O(b^d/2).

3.2 Heurísticas.

La información heurística es un conocimiento parcial sobre un problema que me va a permitir resolverlo de manera más eficiente que la búsqueda sin información dentro de ese dominio.

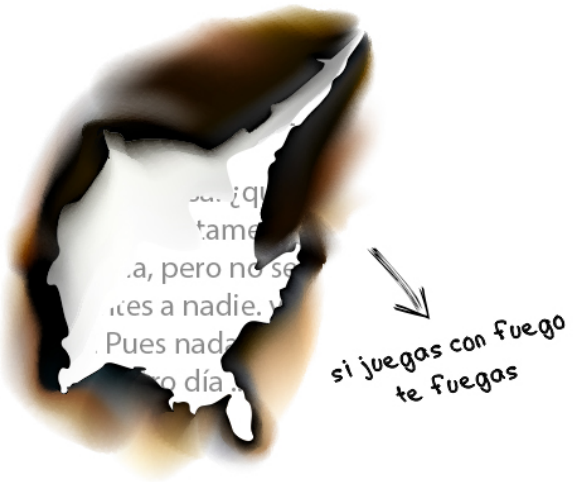
Las heurísticas son criterios, métodos o principios para decidir cuál de entre varias acciones promete ser la mejor para alcanzar una determinada meta.

Una heurística (computacionalmente) es una función (función heurística) que asigna a un estado un valor numérico que voy a utilizar para evaluar cómo de prometedor es un estado. En realidad, la voy a utilizar para comparar estados.

En IA, entendemos por heurística un método para resolver problemas que en general no garantiza la solución óptima, pero que en media produce resultados satisfactorios en la resolución de un problema. Además las heurísticas se van a utilizar para encapsular el conocimiento específico que se tiene sobre un problema, y sirve de guía para que un algoritmo de búsqueda pueda encontrar una solución válida aceptable. Una heurística puede devolver bajo determinadas circunstancias una solución óptima pero es importante saber que la búsqueda heurística no está asociada con búsqueda óptima.

La definición de heurísticas no es un proceso inmediato, un aspecto importante es determinar a qué fuente de conocimiento hay que acudir. En general la información de la que debemos disponer es información específica del problema que vamos a resolver, por ejemplo, en el ajedrez podemos valorar un tablero con información estándar como: la ventaja de piezas, la experiencia de un humano o de aprendizaje automático.

NO QUEMES TUS APUNTES



GANA 0,25 €

por cada PDF tuyo subido de calidad

* válido hasta el 3 de junio de 2022 o hasta llegar al tope de documentos para esta promoción

WUOLAH

3.3 Métodos de escalada

En estos algoritmos el camino NO es importante, la solución está en el propio nodo. Estos algoritmos son algoritmos de búsqueda local y son estrategias irrevocables. Este proceso de búsqueda local lo que va haciendo es seleccionar la solución mejor dentro de un vecindario de una solución inicial e ir saltando por el espacio de estados hasta encontrar un óptimo que puede ser local o global.

3.3.1 Algoritmo de escalada simple.

Este algoritmo parte de un estado actuall E, determina un operador A cualquiera, lo selecciona de alguna forma, y genera el estado sucesor de E, es decir, el estado resultante de aplicar el operador A(E) y compara la función heurística del estado actual con su vecino. Si el nuevo vecino es mejor que el estado actual, sustituye el estado actual por el nuevo vecino si este estado nuevo es el objetivo el algoritmo acaba y si no continúa ejecutándose.

El problema principal de este algoritmo es que si no encuentra un vecino mejor el algoritmo se queda atascado en un valor y decimos que el algoritmo se ha quedado atascado en un mínimo o máximo local dependiendo si estamos maximizando o minimizando.

3.3.2 Algoritmo de escalada por la máxima pendiente.

Este algoritmo es una mejora del algoritmo de escalada simple. Este algoritmo lo que hace es contemplar todos los sucesores (vecinos) que se puedan generar a partir de un estado E y escoge el mejor de ellos calculando la funcion heurística de todos los vecinos, selecciona el máximo entre ellos y si el vecino que tiene la función heurística mayor mejora al estado actual entonces se sustituye al estado actual por el vecino que tiene el mayor valor de la función heurística.

Las características de estas dos versiones simples de los algoritmos de escalada son las siguientes:

- Completitud: pueden que no encuentren la solución (no son completos).
- Optimalidad: al no ser completos no nos pueden garantizar encontrar la solución óptima.
- Eficiencia: son algoritmos muy útiles cuando disponemos de pocos recursos en memoria y son algoritmos muy rápidos y también útiles cuando la función es monótona (de)creciente.

Los algoritmos de búsqueda local son también útiles para resolver problemas de optimización, en los que el objetivo es encontrar el mejor estado de acuerdo a una función objetivo.

3.3.3 Variaciones estocásticas

El objetivo de estas variaciones es tratar de evitar que el algoritmo se quede atrapado en un máximo o mínimo local o en una zona de meseta. Lo que pretenden estas técnicas es introducir variabilidad en el proceso de búsqueda. Reciben el nombre de estocásticas porque utilizan técnicas basadas en probabilidad para ayudar a crear esa variabilidad en la selección del mejor nodo.

3.3.3.1 Algoritmo de escalada estocástico.

Produce una variación aleatoria basada en la probabilidad, en este método la función heurística f se transforma en una distribución de probabilidad, de manera que si tenemos un estado con 3 sucesores uno con una valoración heurística de 10 (n1) otro de 8 (n2) y otro de 4 (n3), la función de probabilidad para cada uno de los sucesores consiste en establecer la función de distribución de cada uno de ellos a partir de su función heurística y la suma de las funciones heurísticas de cada sucesor y, a continuación, se realiza un sorteo en el que la oportunidad de escoger de uno de los tres sucesores dependerá de la probabilidad que tenga asociada.

$p(n1) = f(n1)/\Sigma f(n)$.

3.3.3.2 Algoritmo de escalada de primera opción

El algoritmo de escalada de primera opción genera sucesores aleatoriamente hasta que genera uno que es mejor que el estado actual. Esto es una buena estrategia cuando un estado tiene muchos, en general, miles de sucesores.

3.3.3.3 Algoritmo de escalada con reinicio aleatorio

Se trata de generar una solución aleatoriamente cuando este se atasca o esta cerca de atascarse, por la que continua el proceso. Este algoritmo es útil solo cuando las características del problema me permite generar con facilidad soluciones aleatorias.

El éxito de este algoritmo dependerá mucho de la forma que tenga el espacio de estados, es decir, si hay pocos máximos locales y mesetas el reinicio aleatorio encontrará una buena solución rápidamente.

3.3.4 Enfriamiento simulado

Trata de unir características de la ascensión de colinas con las variaciones estocásticas (aleatorias). La idea clave de este algoritmo es que me va permitir visitar soluciones peores que la actual para evitar quedarse atrapado en óptimos locales.

Este algoritmo trata de hacer una analogía entre el proceso el modelo físico de proceso de enfriamiento minimizando una función en la búsqueda dentro de un espacio de estados, de manera que asocia estados del espacio de estados con los estados por los que pasa el sistema físico de partículas que equivalen a las soluciones factibles del algoritmo.

La energía del estado actual del sistema es el valor de la función objetivo de la solución actual. Y un cambio de estado en el sistema va a equivaler a la exploración de un vecino del sistema actual, por tanto, los diferentes cambios de estado lo que van a suponer es el viaje por las distintas soluciones vecinas. Finalmente el estado final estable va a ser la solución final de nuestro algoritmo.

Este algoritmo comienza con la creación de una solución inicial. Una vez generada, se procede a un proceso de búsqueda típico por ascensión de colinas con la peculiaridad de que se va a seleccionar el sucesor dependiendo de un cálculo de probabilidad en función de la temperatura y del incremento de energía.

La temperatura se calcula a partir de una función que se le pasa al algoritmo como parámetro que lo que hace es asignar un valor a la temperatura en función del tiempo (el número de iteraciones que lleva el algoritmo), esta temperatura va disminuyendo conforme más iteraciones llevemos hasta que llegue a 0 que ahí es cuando finalizará el algoritmo. Por tanto, la temperatura inicial y la final del proceso no la sabemos y se tienen que dar como entrada al algoritmo.

El siguiente paso es la generación de sucesores y la selección del candidato y finalmente llegamos a la parte fundamental del algoritmo que consiste en el cálculo del incremento de energía.

El incremento de energía se calcula restando al valor heurístico del sucesor escogido el valor heurístico del nodo actual. Si el incremento de energía es positivo, es decir si el sucesor es mejor, seguimos con el algoritmo. Si es al contrario vamos a seguir con un sucesor peor siempre y cuando hagamos un experimento aleatorio con una probabilidad calculada a partir del valor de temperatura y del incremento de energía que hemos calculado en las etapas anteriores.

$$P = e^{*}(IncrementoEnergía/Temperatura)$$

Una de las principales ventajas de este algoritmo es que permite salir de óptimos locales, es un algoritmo eficiente y es fácil de implementar. Sin embargo, el principal inconveniente que tiene es el cálculo de todos los parámetros que rodean al algoritmo por ejemplo determinar la temperatura inicial, la función para actualizar la temperatura, el número de vecinos a generar en cada iteración etc.. todos estos parámetros se tienen que calcular mediante ensayo y error.

Pese a todo el algoritmo puede proporcionar soluciones mucho mejores que si utilizamos métodos de escalada no probabilísticas.

3.5 Algoritmos Genéticos

Es una técnica de búsqueda local que está inspirada en el proceso de la evolución natural. Estos algoritmos siguen un proceso que trata de encontrar una configuración óptima, en una situación inicial, simulando el proceso de evolución mediante reproducción sexual.

Una diferencia fundamental entre los algoritmos genéticos y los procesos de búsqueda en un espacio de estados es que en un algoritmo genético no partimos de un único estado inicial si no de varios estados iniciales , es decir, partimos de una población de estados que toman el nombre de individuos y que se van transformando aleatoriamente. Por tanto tampoco hay operadores que transformen estados, en su lugar, tenemos los llamados operadores genéticos que operan sobre la población en su conjunto y la hacen evolucionar. El proceso de evolución de la población se detiene cuando algún individuo cumple los criterios de la función objetivo o hasta que no se puede mejorar la población

El proceso genérico de este algoritmo es el siguiente:

1. Primero se genera y evalúa una población inicial
2. Seleccionar los individuos que se deben de reproducir.
3. Se realizan cruces de padres para generar nuevos hijos
4. Se realizan mutaciones tantos de hijos como de padres.
5. Se evalúan todos los individuos de nuevo y se vuelve al paso 2.

3.5 Búsqueda del primero mejor

Son algoritmos basados en exploración de grafos pero que utilizan información heurística. Estos algoritmos se usan en problemas en los que vamos a necesitar información heurística (como de cerca o lejos estamos de la solución) y en los que nos interesan saber el camino desde el nodo inicial a la solución.

Esta búsqueda trata de seleccionar el siguiente nodo para expandir basado en una función de evaluación que se construye como una estimación del coste de llegar a la solución de manera que el nodo con la menor evaluación es el que primero se expande. Esta función de evaluación no es inmediato obtenerla y va a depender del problema que intentemos resolver.

Estos algoritmos incluyen como componente a la función de evaluación una función heurística (h(n)), es importante tener en cuenta que esta función es una estimación porque, en general, no vamos a poder saber exactamente el coste que hay desde el nodo actual hasta el nodo objetivo.

3.5.1 Búsqueda primera mejor greedy

Este algoritmo solo tiene en cuenta en la función de evaluación la función heurística del problema y va expandiendo los nodos con menor valor heurístico. Esta forma de seleccionar nodos no encuentra el camino óptimo porque no considera información del camino desde el nodo inicial hasta el nodo final (g(n)).

3.5.2 Búsqueda primera mejor A*

Este algoritmo tiene en cuenta tanto la función heurística como el coste del camino en la función de evaluación.

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

En un nodo hay que guardar información sobre el estado, el mejor padre, sus hijos y el valor de f(n). En este algoritmo hay que tener en cuenta los repetidos en la cola de abiertos y los repetidos en la cola de cerrados, actualizando los nodos según el coste del camino como ya sabemos por las prácticas.

Las características de este algoritmo son:

- Completitud: son algoritmos completos siempre que el número de sucesores sea finito.
- Optimalidad: en el algoritmo greedy no la encuentra, el algoritmo A* sí la encuentra siempre y cuando se den las siguientes condiciones:
    1. El número de sucesores es finito para cada nodo.
    2. El coste de los arcos del grafo es positivo y mayor que una cantidad dada.
    3. La función h(n) es admisible, es decir, h(n) <= h*(n) siendo h*(n) el valor real de ir desde un nodo al objetivo.

3.6 Heurísticas sobre el proceso de búsqueda

En este apartado vamos a conocer qué otras modificaciones podemos aplicara los procesos de búsqueda en las que hay restricciones de tiempo y espacio. Estas modificaciones también son consideradas heurísticas pero estas modificaciones no son funciones que te devuelven un valor numérico si no que son métodos que se aplican en el proceso de búsqueda cuando tenemos estas limitaciones.

3.6.1 Búsqueda orientada a subobjetivos.

Se trata de identificar en el espacio de estados situaciones que se conocen que forman parte del camino al objetivo y que se van a utilizar como hitos en el proceso de búsqueda para encontrar el objetivo global.

3.6.2 Búsqueda con horizonte.

Trata de intentar encontrar primero una solución o limitar la búsqueda de una solución con una profundidad máxima que se denomina horizonte y plantear todo el proceso de búsqueda dentro de este horizonte. A veces es necesario cambiar el criterio de búsqueda del objetivo.

3.6.3 Búsqueda jerárquica

Partimos de que tenemos operadores a distintos niveles de actuación. Primero llevamos a cabo un proceso de búsqueda para encontrar un plan a un nivel alto y esas soluciones de nivel alto las utilizamos como subojetivos para encontrar pequeños trozos de problemas que se han descompuesto y encontrar la verdadera solución al nivel base o primitivo.

3.7 Grafos Y/O

Si a la hora de resolver problemas de búsqueda afrontamos el problema como una búsqueda en grafos normal podemos encontrarnos caminos que no llevan a ningún sitio o a caminos redundantes.

Una forma de evitar esto es darnos cuenta de que:

1. El estado inicial se puede dividir en componentes
2. Los estados se pueden descomponer de manera independiente.
3. Los operadores se pueden aplicar por separado a cada componente del estado.
4. Debemos tener en cuenta que tenemos que poder detectar si hemos resuelto cada subproblema por separado y, por tanto, debemos ser capaces de expresar que hemos resuelto el problema global a partir de la condición de terminación de cada subproblema componente.

Este proceso de resolución de problemas se representa con una estructura denominada grafo Y/O. Los nodos que se corresponden con estados compuestos, tienen sucesores etiquetados con sus componentes. A estos sucesores se les llama nodos Y porque, para resolver el problema asociado a su padre todos los componentes deben ser resueltos. Los conjuntos de nodos Y se indican con un círculo enlazando los diferentes arcos de entrada.

Los operadores se aplican a los subcomponentes del problema y los nodos resultantes de aplicar un operador se llaman nodos O. Porque para resolver un subproblema basta con procesar uno de los posibles operadores para encontrar una solución.

Y los nodos que se corresponden con estados componentes de otro que cumple la condición de terminación son los nodos terminales y, finalmente, es importante tener en cuenta que una solución en este tipo de grafos no es un camino sino es un subgrafo que representa un conjunto de aplicaciones de operadores para llegar a una solución.