Búsqueda en un espacio de estados

1. Supongamos que, para resolver un problema de espacio de estados, dos personas deciden representar de la misma manera los estados y los operadores, y definen de la misma manera el estado inicial, el final y la descripción de las operaciones. Sin embargo, aplicando el procedimiento de búsqueda en profundidad obtienen soluciones distintas ¿Por qué puede ocurrir esto?
Porque DFS no asegura optimalidad, simplemente devuelve la primera solución válida que encuentre. Si en un problema existe más de un estado solución es perfectamente posible que devuelva diferentes soluciones.

Por el orden en el que hayan almacenado los operadores en la lista *operadores*. En el problema del paseo el algoritmo de búsqueda en profundidad no encuentra solución si el primer elemento es ir-a derecha pero si cuando el primer elemento es ir-a-izquierda.
Fuente: https://www.cs.us.es/cursos/iia-2004/examenes/examen1-sol.pdf

2. ¿En qué afecta al modelo general de resolución mediante búsqueda del agente deliberativo el no disponer de información completa sobre el estado del mundo o al no tener certeza acerca de los efectos de las acciones sobre el mundo?
Afectará en que tendrá que ser flexible para afrontar fallos, puesto que el agente podrá tomar decisiones que no se podrán llevar a cabo debido a la parte incompleta de su representación interna del mundo. Por lo tanto, **en vez de usar algoritmos exactos** para encontrar solución, **tendrá que usar métodos de búsqueda heurística**.

3. Describe en cinco líneas las diferentes estrategias de control para un sistema de búsqueda.
Estrategias irrevocables: En cada momento, sólo mantenemos en memoria un único nodo, que incluye la descripción completa del sistema en ese momento.

Estrategias tentativas: En memoria se guardan todos los estados o nodos generados hasta el momento, de forma que la búsqueda puede proseguir por cualquiera de ellos. Las estrategias tentativas pueden ser backtracking (se puede retroceder en la búsqueda de nodos) y búsqueda de grafos (se tiene una representación de todos los grafos).

4. Describir brevemente el parecido y diferencias entre la búsqueda en anchura y el descenso iterativo.
El descenso iterativo está basado en la DFS en lugar de la BFS, sin embargo, al añadir una bajada por niveles al BFS hace que actúe de forma similar al DFS. Aún así, hay diferencias muy palpables, el descenso iterativo explora los mismos nodos una y otra vez por cada iteración en la que aumenta la profundidad, y además, a pesar de que explora hasta un tope de profundidad en cada iteración, este realiza la exploración como un algoritmo DFS, expandiendo los “hijos” de cada nodo antes que sus “vecinos”.

5. Enumerar las condiciones que se requieren para que un problema se pueda descomponer. Enumerar las condiciones que se requieren para un problema se pueda resolver mediante descomposición. ¿Son las mismas?
Para que se pueda descomponer, el problema se debe poder trasladar a una base de datos, y los datos deben poder tratarse con operaciones. Para que se pueda resolver, además, estas operaciones deben ser resolubles totalmente por separado.

6. ¿Qué es la búsqueda no informada y en qué se diferencia de la búsqueda informada (o heurística)?
La diferencia entre ambos tipos de búsqueda es que según el conocimiento que tenga el agente sobre el espacio de estados podría acelerar la búsqueda usando ese conocimiento.
Cuando el sistema agente no posee ningún tipo de información del espacio de estados, se utilizan técnicas de búsquedas búsqueda no informadas.

7. Describir en menos de cinco líneas el algoritmo de escalada de reinicio aleatorio, y detallar las condiciones para que sea aplicable a un problema.
Hill climbing de reinicio aleatorio es meta-algoritmo construido sobre la base de hill climbing. Es también conocido como Shotgun hill climbing. Este realiza, iterativamente, el hill-climbing, cada vez con una condición inicial aleatoria Xo. La mejor Xm es guardada: si una nueva corrida del hill climbing produce una mejor Xm que el estado guardado, lo reemplaza.

8. ¿Qué característica esencial aporta el algoritmo de enfriamiento simulado frente al resto de métodos de escalada? ¿Qué le aporta? ¿Cuál es la principal dificultad de aplicar dicho algoritmo en un caso concreto?
La principal aportación es que permite visitar soluciones peores a la actual, evitando por tanto el estancamiento en óptimos locales. Se basa en principios de Termodinámica y es un método de búsqueda local.

La principal dificultad es que **requiere muchas pruebas de ensayo y error** hasta ajustar los parámetros óptimos (temperatura inicial, número de vecinos a generar en cada estado…).

9. ¿Cómo combinación de qué dos estrategias de búsqueda puede verse el algoritmo A*?
El algoritmo es una combinación entre búsquedas del tipo primero en anchura (BFS) con primero en profundidad (DFS).

10. El algoritmo A* utiliza una lista de ABIERTOS y una lista de CERRADOS. Describe el propósito de cada una de esas listas.
La lista de nodos abiertos contiene los nodos que no han sido explorados todavía pero están listos para ser explorados, a diferencia de la lista de cerrados, que han sido explorados y expandidos.

11. La búsqueda A* utiliza una heurística combinada para seleccionar el mejor camino a seguir a través del espacio de estados hacia el objetivo. Define las dos funciones utilizadas (h(n) y g(n)).
g(n): Coste total acumulado para llegar desde el inicio hasta el nodo actual.
h(n): Distancia ideal al nodo objetivo.

12. El algoritmo A* no termina mientras no se seleccione un nodo objetivo para su expansión. Sin embargo, podríamos encontrar un camino al objetivo mucho antes de que se seleccione un nodo objetivo para su expansión. ¿Por qué no se termina en el momento en que se encuentre un nodo objetivo? Ilustra la respuesta con un ejemplo.
Porque podría darse la situación de encontrar un camino con menos coste si todavía no se ha expandido el nodo objetivo.

13. ¿Cuál es el objetivo de una función heurística aplicada a la búsqueda en el espacio de estados?
Utilizar la información que disponemos sobre el problema para minimizar la búsqueda hasta el estado objetivo.

14. ¿Cuál es la definición de heurística admisible?
Una heurística es admisible si nunca sobreestima el costo de alcanzar el objetivo, o sea, que en el punto actual la estimación del costo de alcanzar el objetivo nunca es mayor que el menor costo posible.

15. ¿Qué condiciones garantizan que el algoritmo A* obtenga la solución óptima?
Que la heurística utilizada sea admisible, es decir, el coste estimado debe de ser menor o igual que el menor coste posible, pero en ningún caso mayor y que el coste por arco sea siempre positivo. Además, cada expansión de un nodo no debe abrir infinitos nodos.

Que la heuristica sea perfecta, es decir que al evaluar el costo hasta llegar al objetivo nos devuelva el costo real. Es decir: h(n) = h'(n)
Otra posibilidad para que nos devuelva el resultado optimo es que solo haya un camino posible para llegar al objetivo

16. ¿Es la búsqueda primero en anchura un caso especial de la búsqueda de coste uniforme? Razona la respuesta.
Sí. Búsqueda Primero en Anchura es un caso especial de BCU cuando los costos de las aristas son positivos e idénticos.
Búsqueda primero en anchura visita primero el nodo con la longitud del camino más corto (número de nodos) desde el nodo raíz, en cambio, Búsqueda de coste uniforme (UCS) primero visita el nodo con la ruta más corta en costo (suma de los pesos de las aristas) desde el nodo raíz.

17. ¿Son la búsqueda primero en anchura, búsqueda primero en profundidad, y la búsqueda de coste uniforme casos especiales de la búsqueda primero el mejor? Razona la respuesta.

Sí, ya que BPA (y análogamente el BPP) es un caso especial de BCU en el que los costos de las aristas son positivos e idénticos. A su vez, BCU es una variante del algoritmo Búsqueda primero el mejor.

18. ¿Es la búsqueda de coste uniforme un caso especial de la búsqueda A*? Razona la respuesta.
La Búsqueda de Costo Uniforme es un caso particular del algoritmo de búsqueda A* si la heurística de este último es una función constante. Si A* se utiliza con una heurística monótona, entonces se puede convertir en una Búsqueda de Costo Uniforme restando de cada costo de arista a la disminución en el valor heurístico a lo largo de esa arista.

Creo que es más correcto:
Sí, ya que para transformar A* en BCU lo único que debemos asumir el costo h es constante para cualquier nodo y que el coste total dependa únicamente de g.

Búsqueda con adversario y juegos

19. Explica qué se interpreta por juegos de Información Perfecta. Da un par de ejemplos de juegos con información perfecta, y otro de un juego que no lo sea.
Significan entornos deterministas, totalmente observables.

20. ¿Qué se entiende por Búsqueda con Adversario, y en qué difiere de la búsqueda tradicional?
21. La mayor parte de los programas para juegos no almacenan los resultados de las búsquedas de un movimiento para el siguiente movimiento, ya que normalmente empiezan de cero cada vez que es el turno del ordenador. ¿Por qué?
22. ¿Por qué los procedimientos de búsqueda en juegos parten de la posición inicial en vez de empezar por la posición objetivo y realizar la búsqueda hacia atrás?
23. El procedimiento minimax ¿es un procedimiento primero en profundidad o en anchura?
24. ¿Por qué podemos decir que la poda alfa-beta es un algoritmo mejor que el procedimiento minimax para la resolución de juegos?

Búsqueda en un espacio de estados

25. ¿Requiere percibir un agente deliberativo mientras construye el plan? ¿y cuando lo ejecuta?
Para construir el plan es necesario el estado inicial, por lo tanto el agente debe de percibir el ambiente inicialmente, además, es muy recomendable que perciba el ambiente en todo momento puesto que el entorno está expuesto a modificaciones continuas que deben ser actualizadas por el agente en su representación interna del mundo. Por ello, la información recolectada por los sensores ha de tener prioridad a la que el agente tenía almacenada.

26. ¿Qué diferencia hay entre una arquitectura de control retroactiva y una búsqueda en profundidad sobre grafos?

En la **retroactiva** sólo se guarda un hijo de cada estado, con lo que el grafo explícito es realmente una lista, mientras que en la **búsqueda en profundidad sobre grafos** se guardan todos los estados o nodos generados hasta el momento.

27. ¿Qué condiciones debe verificar un problema para que se pueda descomponer? ¿y para que sea resuelto de forma descomponible?
Un problema es descomponible si se puede descomponer en un conjunto de subproblemas independientes más sencillos.
Debe tener una base de datos inicial, unos operadores delimitados y un objetivo.

28. ¿Qué es un grafo Y/O y para que se utiliza sobre problemas descomponibles?

Los Grafos Y/O es una composición de grafos destinados a la resolución de problemas que se pueden descomponer. Son combinación de grafos Y y grafos O que indican el orden de consecución de tareas a realizar para alcanzar el objetivo

- Los grafos Y marcan los objetivos a completar para resolver un objetivo previo. Para resolverlos: Resolver todos los hijos y juntar soluciones.
- Los grafos O marcan dos objetivos de los cuales hay que completar uno para la un objetivo determinado. Para resolverlos: resolver un hijo y ver si tiene solución, en caso contrario devolvemos siguiente hijo, proceso iterativo.

29. ¿Utilizarías un método de escalada para problemas que requieran obtener una secuencia de acciones? Razona la respuesta.

(Igual valdría un grafo Y/O, pero no estoy muy seguro)

Sí, siempre y cuando existan soluciones parciales al dominio del problema. Es decir, si imaginamos que para llegar a un objetivo necesitamos realizar antes una serie de objetivos menores, véase abrir una puerta para cambiar salir al pasillo y finalmente llegar al baño, sí, en caso contrario, no.

30. ¿Cómo implementarías una estrategia retroactiva para problemas en los que dispones de una función heurística?
Como disponemos de una función heurística asumo que la búsqueda es con información así que usaría el algoritmo A* implementando una lista para los nodos ABIERTOS (nodos sin explorar y sin evaluar sus hijos) y al mismo tiempo una lista para CERRADOS que contendrán nodos que aún no se han explorado.

Si no tengo información usaría una búsqueda greedy, primero el mejor, donde la selección del nodo se seleccionando el hijo con menor coste de arco (valor heuristico).

31. ¿Qué es una función heurística? ¿Para qué se usa?
Las funciones heurísticas son la forma más común de transmitir el conocimiento adicional del problema al algoritmo de búsqueda. Se utilizan para intentar minimizar el número de

nodos expandidos para llegar a la solución cuando se utiliza un conocimiento extra sobre el problema además de su planteamiento.

32. ¿Qué métodos heurísticos tienen garantía de conseguir la solución óptima? ¿Bajo qué condiciones?

Los métodos heurísticos con información por ejemplo, el A*. Siempre y cuando se use una heurística admisible, que nunca sobreestime el coste de alcanzar el objetivo. Las heurísticas admisibles son por naturaleza optimistas, porque piensan que el coste de resolver el problema es menor que el que es en realidad.

33. ¿Por qué los métodos de escalada utilizan criterios aleatorios en algunas decisiones?

Cómo se selecciona aleatoriamente un punto inicial, elimina el riesgo de que se alcancen óptimos locales pero nunca el óptimo global. Además, no es mucho más costoso que realizar un hill climbing simple puesto que tan solo se multiplica el coste por un factor constante (el número de veces que quieres hacer el reinicio aleatorio).

34. ¿Qué interpretación tiene la función f en el algoritmo A*?

f(n) = g(n) + h(n), donde g(n) es el coste desde el nodo inicial hasta n, y h(n) es el coste heurístico desde n hasta el nodo objetivo.

35. ¿Hay siempre garantía de que el algoritmo A* encuentra solución si ésta existe?

Si. En el peor de los casos y con la peor de las heurísticas el algoritmo A* se reduce a un algoritmo de búsqueda voraz primero el mejor.

36. ¿Encuentra A* siempre la solución óptima? Razona la respuesta

Si la función heurística es admisible, nunca sobrestima el coste mínimo real de llegada a la meta, entonces el propio A* es admisible (u óptimo) si no usamos un set cerrado.
Si se utiliza un set cerrado entonces h debe ser además monotónica (o consistente) para que A* sea óptimo.
Una heurística h(n) es consistente si, para cada nodo n y cada sucesor n` de n generado por cualquier acción a, el coste estimado de alcanzar el objetivo desde n no es mayor que el coste de alcanzar n` más el coste estimado de alcanzar el objetivo desde n:

$$h(n) = c(n,a,n`) + h(n)$$

37. ¿En qué consiste la búsqueda dirigida?

La búsqueda dirigida es un tipo de búsqueda del tipo A* en la que se especifica un factor de ramificación K, de modo que se limita el número de nodos vecinos que se expande en cada nivel seleccionando los n mejores.

38. ¿Qué interés tiene la versión paramétrica del algoritmo A* con función f(n)=g(n)+wh(n)?

Modificar el valor heurístico de h(n), de forma que se puede aumentar la velocidad al crear plan, a costa de encontrar soluciones un poco peores a las óptimas. Ésto lo explicó el profesor en prácticas. Normalmente w > 1.

Si se acerca a 1 provoca más ramificación y por ende mayor exploración, si se acerca a 0 provoca una búsqueda voraz.

39. ¿En qué contexto utilizarías la arquitectura de percepción/planificación/actuación?
En el de un agente deliberativo que tiene que resolver el desplazamiento en un entorno para alcanzar unos objetivos.

40. Describe la búsqueda jerárquica y da un ejemplo de su uso.
Un agente solo puede seguir el plan generado por el algoritmo previo durante un corto periodo de tiempo. Cuando alcanza el final del nivel más bajo del plan, necesitará planear la siguiente sección con mayor detalle.
Cuando un plan acaba, el algoritmo es llamado de nuevo y la siguiente sección es devuelta.

41. Describe la búsqueda orientada por subobjetivos y da un ejemplo de su uso.

42. ¿En qué modelo real se basa el algoritmo de enfriamiento simulado? ¿Qué propiedades de ese modelo son útiles para los métodos de escalada?
En metalurgia, el temple es el proceso utilizado para templar o endurecer metales y cristales calentándolos a una temperatura alta y luego gradualmente enfriarlos, así permite al material fundirse en un estado cristalino de energía baja. Para entender el temple simulado, cambiemos nuestro punto de vista de la ascensión de colinas al gradiente descendente (es decir, minimizando el coste) e imaginemos la tarea de colocar una pelota de ping-pong en la grieta más profunda en una superficie desigual. Si dejamos solamente rodar a la pelota, se parará en un mínimo local. Si sacudimos la superficie, podemos echar la pelota del mínimo local. El truco es sacudir con bastante fuerza para echar la pelota de mínimos locales, pero no lo bastante fuerte para desalojarlo del mínimo global. La solución del temple simulado debe comenzar sacudiendo con fuerza (es decir, a una temperatura alta) y luego gradualmente reducir la intensidad de la sacudida (es decir, a más baja temperatura).

43. ¿Cómo se mide la adecuación con el entorno en un algoritmo genético?
Con el valor de función objetivo (fitness). La función de fitness se define sobre la representación genética y mide la calidad de la solución representada.

44. ¿Qué caracteriza a los algoritmo genéticos como métodos de escalada?
La principal característica es que no necesitan partir de un nodo/estado inicial, ya que hay toda una población.
Su objetivo es encontrar una solución cuyo valor de función objetivo sea óptimo.

45. Describe en detalle bajo qué condiciones el algoritmo de enfriamiento simulado decide cambiar a un estado peor.
A mayor temperatura, mayor probabilidad de aceptación de soluciones peores. Así, el algoritmo acepta soluciones mucho peores que la actual al principio de la ejecución (exploración) pero no al final (explotación).

Por tanto, disminuye las posibilidades de saltar a una solución peor cuanto más cerca esté de un óptimo global.

46. Describe cómo se realiza una mutación en un cromosoma en un algoritmo genético.
Consiste en la modificación de un bit en un algoritmo genético. Esta modificación se efectúa con una probabilidad preestablecida, llamada **probabilidad de mutación**.

En una población, se asigna un porcentaje para las mutaciones. En la siguiente generación, este porcentaje de población tendrá pequeños cambios aleatorios. Rollo X-Men. <- xddddddddddddddddddddddddddddddddddddddddddd

47. Describe cómo se realiza un cruce entre dos nodos en un algoritmo genético.
Se selecciona un punto en el vector del primer parental. Todos los datos más allá de este punto en el vector del organismo se intercambiarán entre los dos organismos parentales. Los organismos resultantes son los hijos.

48. Describe cómo se realiza la selección en un algoritmo genético.
Después de saber la aptitud de cada cromosoma se procede a elegir los cromosomas que serán cruzados en la siguiente generación. Los cromosomas con mejor aptitud tienen mayor probabilidad de ser seleccionados.

49. En un algoritmo de búsqueda "primero el mejor" ¿que nodos se incluirán en la lista CERRADOS?

(Creo que el todos nodos expandidos excepto el que se ha seleccionado como el mejor).

Búsqueda con adversario y juegos

50. ¿Por qué la estructura básica para representar un juego es un árbol y no un grafo?
51. ¿Qué relación hay entre el árbol del juego y un árbol Y/O?
52. ¿Por qué en juegos se llama a la función heurística función de evaluación estática?
53. ¿Cuáles son los cuatro elementos que definen un algoritmo sobre juegos para obtener la mejor jugada inmediata?
54. ¿Por qué en el algoritmo Minimax es conveniente utilizar estrategias de búsqueda retroactivas para la búsqueda parcial?
55. ¿Por qué si aplicamos directamente la función de evaluación estática a los sucesores de un nodo sobre los que tenemos que decidir cuál es la mejor jugada el modelo no funcionará bien?
56. ¿Cuáles son las dos heurísticas en la que se basa la regla minimax?
57. ¿Qué es el factor de ramificación y cómo afecta a la complejidad de un juego?

58. ¿Cómo se mide la complejidad de un algoritmo que obtiene la mejor jugada para un juego?
59. ¿Por qué el algoritmo Minimax no es el algoritmo estándar para resolver juegos?
60. A grandes rasgos, ¿cuál es el análisis de complejidad de la poda alfa beta?
61. ¿Sería muy complejo adaptar el modelo de resolución de juegos al caso de juegos que no sean bipersonales?
62. ¿Cómo se puede resolver un juego en el que hay incertidumbre?
63. ¿Qué es el efecto horizonte? ¿Tiene solución?
64. ¿Podría utilizarse el algoritmo STATUS para resolver el juego del ajedrez?¿Por qué?
65. ¿Qué tipo de juegos podría resolver el algoritmo STATUS?
66. Describa la diferencia entre resolver un juego y encontrar la mejor jugada inmediata
67. Describe las ideas básicas de la heurística propuesta en el juego de las damas de Samuel
68. Describe las ideas básicas de la heurística propuesta por Turing para el juego del ajedrez

Búsqueda en un espacio de estados

69. Relacione la idea de trabajar con soluciones parciales frente a soluciones completas con el uso de métodos de escalada.

70. ¿Puede utilizarse una estrategia irrevocable para problemas que impliquen encontrar un camino? Razonar la respuesta.
Como en una estrategia irrevocable por definición solo se guarda un único nodo estado en cada momento, no se podría recuperar los diferentes estados que han llevado hasta el estado solución.

*En realidad creo que se guarda toda la información del problema que se ha desarrollado, pero no se guarda la información antes de llegar al estado actual, por lo que sería imposible volver porque no sabemos por donde. Es casi lo mismo que lo que está escrito arriba.

(Puede que también valga): Sí, siempre y cuando la búsqueda de un camino por estrategia irrevocable esté implementada en un agente que pueda anular el camino por decisiones reactivas.

71. ¿Qué aporta el uso de aleatoriedad en los métodos de escalada?
Evita mínimos locales. Gracias a la aleatoriedad se puede conseguir encontrar el mínimo global realizando diferentes iteraciones de la búsqueda.

72. ¿En qué se diferencia el método de escalada estocástico del método de escalada de primera opción?¿Qué relación tienen estos dos métodos con los otros métodos de escalada no estocásticos?

73. ¿Qué es un programa de enfriamiento? ¿En qué algoritmos se usa? ¿Qué influencia tiene en la solución del problema?

74. Los algoritmos de búsqueda por el mejor nodo se pueden aplicar sobre representaciones de un problema en forma de grafo o de árbol. Analice las ventajas e inconvenientes de ambas sobre el algoritmos A*.

75. Si ejecutando el algoritmo A* paramos devolviendo una solución en el momento en el que un nodo objetivo entra en ABIERTOS, ¿qué puede ocurrir? propón un ejemplo que lo ilustre.
Que no se devuelva el camino resultado óptimo. (Falta ejemplo)

PREGUNTAS PARCIAL:
Tipo A

1. ¿Que es el test de turing y que espera conseguir?
2. ¿Qué es un sistema de producción?¿Cómo se usa para implementar agentes reactivos?
3. ¿Qué es un programa de enfriamiento?¿En qué algoritmos se usa?¿Qué influencia tiene en la solución del problema?
4. Describe cómo se debe codificar un estado si usamos un algoritmo genético
5. ¿Son la búsqueda primero en anchura, búsqueda primero en profundidad, y la búsqueda de coste uniforme casos especiales de la búsqueda con el algoritmo A*? Justifica y detalla la respuesta.

Tipo B

1. ¿Qué propiedades debe verificar un agente para ser llamado inteligente?
2. ¿Qué problema intenta resolver el campo de potencial artificial?¿Cómo lo hace?
3. ¿Qué característica esencial aporta el algoritmo de enfriamiento simulado frente al resto de métodos de escalada?¿Qué le aporta?¿Cuál es la principal dificultad de aplicar dicho algoritmo en un caso concreto?
4. ¿Qué caracteriza a los algoritmos genéticos como métodos de escalada? Detalla las características comunes y las que los hacen diferentes del resto.
5. Los algoritmos de búsqueda por el mejor nodo se pueden aplicar sobre representaciones de un problema en forma de grafo o árbol. Analice las ventajas e inconvenientes de ambas sobre el algoritmo A*.