# Cuestiones del primer examen de IA (UGR)

Francisco Navarro Morales

25 de abril de 2017

# 1. CUESTIONES

**1.1. El concepto de agente. Agentes Inteligentes vs Agentes Racionales. Arquitecturas de Agentes.**

**1.2. Diferencias y similitudes entre Agentes Reactivos y Deliberativos.**

**1.3. El concepto de Heurística. Importancia de la Heurística en I.A.**

**1.4. Características esenciales de los Métodos de Escalada.**

**1.5. Características esenciales del Algoritmo A***

**1.6. Describe los elementos característicos de un Algoritmo Genético. ¿Que problemas pueden resolverse mediante un Algoritmo Genético?**

# 2. RESPUESTAS

## 2.1. El concepto de agente. Agentes Inteligentes vs Agentes Racionales. Arquitecturas de Agentes.

Un agente es un sistema situado en algún ambiente, capaz de actuar de forma autónoma de acuerdo con el estado del ambiente (que puede percibir por medio de sensores) para cumplir sus objetivos de diseño. Además, dicho sistema puede contar con una serie de actuadores que le permiten producir cambios en el ambiente en función de las acciones que decida realizar. La racionalidad de un agente viene dada por la capacidad que tiene para dirigir sus acciones hacia la obtención del mayor beneficio posible, luego depende del criterio que se utilice para definir el desempeño del agente en su actividad y el grado de éxito obtenido. Es preciso señalar que la racionalidad de un agente en un momento determinado está condicionada a lo que ha percibido con anterioridad (ya que una mayor percepción del entorno propiciará mejores decisiones) y de la información con la que cuenta el agente sobre el medio (qué puede esperar del entorno en adelante, qué falta por descubrir, etc..) así como de las acciones que puede llevar a cabo en su estado actual (a mayor abanico de acciones a ejecutar, mayor posibilidad de actuar correctamente tendrá el agente). La arquitectura de un agente es cómo está diseñado su comportamiento. Podemos distinguir, desde agentes puramente reactivos, que generan una acción en cada momento según la información percibida a través de sus sensores (a diferencia de otros agentes reactivos, que tienen en cuenta también las percepciones anteriores y mantienen un modelo estimado del estado del medio) hasta agentes cuyo comportamiento está guiado por deseos, suposiciones e intenciones (BDI agents) codificadas en estructuras de datos, pasando por agentes guiados por metas, agentes estructurados por capas (de forma que si un módulo superior del esquema se cumple no da lugar a que los módulos inferiores se ejecuten), o agentes que aprenden.

## 2.2. Diferencias y similitudes entre Agentes Reactivos y Deliberativos.

Un agente reactivo decide en cada instante su próxima acción en función de la última información recibida del medio (y en algunos casos, de la información recibida anteriormente), de forma que en su diseño están contempladas multitud de situaciones que pueden darse para las que se establece una acción lo más adecuada posible. No así en el caso de los agentes delibe-

rativos, que son capaces de establecer objetivos y localizar secuencias de acciones (planes) mediante algoritmos de búsqueda (planificación) que pueden llevarles a conseguir dichos objetivos. Estos agentes serán mucho más complicados porque incluyen, además de la planificación, la monitorización de los planes creados (actualizarlos para adaptarlos a las circunstancias cambiantes del entorno) y la decisión de cuando realizar búsquedas para cumplir objetivos se hace también una decisión importante de diseño (ya que las búsquedas son muy lentas en comparación a las decisiones casi instantáneas propias de un agente reactivo), para emplear el menor tiempo posible. Por lo general, un agente deliberativo puede encapsular a un agente reactivo para poder reaccionar a los cambios producidos en el entorno durante la ejecución de sus planes, o para tener un comportamiento base mientras no se tenga ningún plan. Además, ambos tipos de agentes suelen tener un modelo del entorno (aunque en el caso del agente reactivo no es indispensable) y tienen una medida de del desempeño que define el éxito de sus decisiones. Por lo general los agentes deliberativos son mucho más lentos (en cuanto al tiempo necesario para elegir una acción) debido al tiempo empleado en planificar y al tiempo empleado en determinar cuando y qué planificar.

## 2.3. El concepto de Heurística. Importancia de la Heurística en I.A.

Los métodos de búsqueda empleados en el diseño de agentes deliberativos pueden necesitar tiempos de ejecución demasiado grandes como para que nos interese realizar dichas búsquedas. Es por ello que se requieren técnicas que orienten la búsqueda de una solución aportando un criterio para distinguir lo bueno o malo que sería seleccionar una acción en cada momento, de entre todas las posibles. Llamamos pues Heurística a, dado un problema de búsqueda para el que se han establecido una serie de estados posibles, un estado inicial y un estado objetivo, y la correspondencia entre el paso de un estado a otro y las distintas acciones posibles que puede tomar el agente (es decir, tenemos un grafo de búsqueda o podemos construirlo), el método que nos permite determinar de forma aproximada lo cerca que queda un estado de la solución final, con la intención de dirigir la búsqueda hacia estados que prometan dar con la solución rápidamente, si existe. Una buena heurística es indispensable para realizar buenas búsquedas en el grafo de estados de un problema, pues encapsula la información que tenemos sobre el mundo en que habita el agente que diseñamos y puede permitirnos realizar las búsquedas en tiempos mucho menores que si no se utilizaran. Además, es importante saber que una mala heurística no solo dará lugar a

un algoritmo más lento que una buena heurística, sino que además podría dar lugar a soluciones que no son óptimas. Si el objetivo de la búsqueda es encontrar caminos lo más cortos posibles, una heurística que diera a los estamos estimaciones superiores de las reales al valor de la distancia hasta el objetivo probablemente no daría lugar a soluciones óptimas. Así pues, decimos que una heurística es admisible si no sobrestima la distancia entre los nodos que participan en la búsqueda y el nodo objetivo. Un ejemplo de heurística sería el del algoritmo A estrella, que utiliza como criterio la suma del costo real o aproximado para llegar desde el nodo inicial a un nodo intermedio y el costo aproximado para llegar del nodo intermedio al nodo objetivo. Si, por ejemplo, la búsqueda se da en un mapa cuadriculado en el que se permiten caminos en diagonal, una buena heurística sería aquella que considera que el costo desde el nodo intermedio hasta el objetivo no es mayor que la distancia mínima (línea recta en diagonal) entre ambos puntos; y una mala heurística sería una que estableciera dicha distancia como la distancia que emplearía el agente en ir al nodo objetivo haciendo un camino horizontal y uno vertical (que serían los catetos empleados para hallar la diagonal que representa la línea recta entre los nodos) puesto que este camino siempre será mayor que el que se haría en línea recta diagonal y se le daría preferencia a los nodos que están en el eje horizontal o vertical de la solución, cuando los que no lo están podrían ser mejores si toman la diagonal entre los nodos.

## 2.4. Características esenciales de los Métodos de Escalada.

Los métodos de escalada son un tipo de algoritmos de búsqueda local con conocimiento que se basan en la búsqueda en profundidad, ya que parten de un nodo inicial y seleccionan de entre sus hijos al mejor de ellos para continuar la búsqueda con él. Así, dada una función objetivo que determine el éxito del agente, estos métodos serán capaces de encontrar máximos (o mínimos) locales en dicha función (puesto que la búsqueda dirige al agente a través del hijo más prometedor del nodo seleccionado en cada momento, y cuando encuentra un nodo mejor que todos sus hijos no puede avanzar), que pueden o no ser los extremos globales. Por tanto, a no ser que la función objetivo sea monótona (creciente o decreciente), una búsqueda en escalada no tiene por qué encontrar la solución buscada (luego no es completo ni admisible). La ventaja de este algoritmo es su velocidad en la búsqueda de extremos locales; si la función es monótona encuentra la mejor solución muy rápidamente, y si no, encuentra una solución buena dentro del vecindario del problema. Su principal inconveniente es en el caso de funciones con muchos extremos locales en las que la diferencia entre

un extremo y otro sean considerables, pues podría llevarnos a soluciones mucho peores que la óptima. Hay diversas variaciones, como por ejemplo la opción de ejecutar dicha búsqueda varias veces partiendo de nodos iniciales generados aleatoriamente (si no nos importa el estado inicial) y quedarse con el mejor resultado obtenido de entre todas las búsquedas; la distinción entre la escalada simple (que escoge el primer hijo mejor que el nodo seleccionado en cada momento) y la de máxima pendiente (que selecciona el mejor de entre todos los hijos en cada momento) o la técnica del enfriamiento simulado que permite escapar de óptimos locales permitiendo con cierta aleatoriedad seleccionar nodos un poco peores que su padre en algunas circunstancias por si los hijos de tal nodo fueran mucho mejores que los del primero a pesar de que seleccionar al padre no acercara a la solución desde el primer nodo seleccionado.

## 2.5. Características esenciales del Algoritmo A*

El algoritmo A* parte de la búsqueda primero el mejor, que es una combinación de las búsquedas en anchura y en profundidad. Dicho tipo de búsqueda se basa en mantener un conjunto de nodos denominado frontera en el que se introducen los hijos del nodo inicial y, posteriormente, los hijos del mejor nodo de la frontera en cada momento, descartando cada vez el nodo elegido. Para determinar cual es el mejor nodo se utiliza una heurística. Lo interesante de estos algoritmos es que no descartan ningún camino y van ampliando la frontera de búsqueda en una dirección que prometa ser buena según la heurística. El algoritmo A* es pues, un caso concreto de la búsqueda primero el mejor en el que la función heurística, f(n), de un nodo es la suma de la distancia desde el nodo inicial a dicho nodo, g(n), y la estimada entre dicho nodo y el objetivo, h(n). Así, estamos empleando una combinación del algoritmo de búsqueda uniforme o algoritmo de Dijkstra (que siempre encuentra el camino más corto pero que desperdicia tiempo buscando al probar caminos no prometedores), equivalente a suponer que h(n) = 0 para todos los nodos; y el algoritmo Greedy de búsqueda primero el mejor, que sería lo equivalente a suponer que g(n) = 0 en todos los nodos y tener en cuenta solo lo prometedor que es cada nodo (esta búsqueda sería muy rápida pero tiene el inconveniente de no dar garantía de aportar soluciones óptimas). Al combinar ambos algoritmos en el llamado A estrella, y siempre que la heurística empleada para determinar g(n) sea admisible (no debe sobrestimar la distancia de un nodo intermedio al nodo objetivo), puesto que de no serlo el algoritmo no sería A* sino simplemente A y las soluciones obtenidas no tendrían por qué ser óptimas. En cualquier caso, el algoritmo A* es completo, porque garantiza obtener una solución y óptimo

con una buena heurística. Es un algoritmo que requiere mucha memoria para ejecutarse (puesto que mantiene en todo momento la posibilidad de explorar caminos que anteriormente no eran prometedores)

## 2.6. Describe los elementos característicos de un Algoritmo Genético. ¿Que problemas pueden resolverse mediante un Algoritmo Genético?

Un algoritmo genético es un tipo de búsqueda meta-heurística (se utiliza cuando no existe una heurística adecuada para realizar la búsqueda) así como una técnica de optimización basada en principios presentes en la evolución natural (se trata, por tanto, de un algoritmo bioinspirado) que consiste en determinar los distintos estados o "soluciones"del problema y tratarlos como cromosomas por medio de unas operaciones que nos permitan seleccionar las mejores soluciones, combinar soluciones (generar descendencia de dos soluciones, con características propias de sus "padres") y mutar la descendencia para generar una variación añadida entre la población (al igual que ocurre en la naturaleza). La selección de las mejores soluciones se utilizará pues para determinar qué soluciones dan lugar a nuevas soluciones (descendencia) y, dado que esto provocará que las mejores soluciones den lugar a más descendencia, la población de soluciones será cada vez más adecuada, de forma que llegará un punto en que se obtenga una solución adecuada para nuestro problema.