3 tecnicas importantes para solucionar el problema:

Search: De por medio existe un mecanismo de búsqueda, es decir espacios de solución que puede existir, exploraciones en el campo, extraigo una que se adapte al problema actual, eso es buscar.

Uso del conocimiento (experiencia previa, use of knowledge): Aprendemos como usar la experiencia previa, que depende de tu creatividad para plasmarla en código, función, etc, osea se cuantifica la experiencia previa en una función o código.

Abstraction: Dividir problemas en sub-problemas, por ejemplo, tengo un nodo raíz que no significa que este sea el primero, estos conocen el problema que ellos tienen y a lo mucho solo del nodo adyacente, pero no más allá de estos adyacentes, los demás tengo que de alguna manera intuirlos a través de la experiencia que ya tengo.

Más experiencia mejora la búsqueda, mejora la base de conocimiento.

El secreto está en cómo represento el conocimiento, lo que aprendo lo plasmo en una función, formula, código, etc.

El primer principio de inteligencia artificial, es saber representar datos inteligentes.

Machine learnning: Tengo un conjunto de datos y quiero hacer una representación de datos, quiero hacer que la red aprenda.

Deep learning: Siempre se tiene que calibrar el conocimiento, porque el resultado que se espera es desconocidos, probablemente se están explorando árboles que son desconocidos, y no hay modelo matemático que lo resuelva, que vaya desenvolviéndose conforme se desenvuelve el problema.

Deep learnning: Aprendizaje profundo.

Generic computing

Hight Climbing:

Lo importante de la inteligencia artificial son los algoritmos que te permiten cambiar de un estado a otro., lo que le hace parecer que tiene un comportamiento inteligente.

Data driven:

Tecnicas de búsquedas:

Búsqueda algorítmica: Significa que ya existe una solución.

Busqueda exaustiva: búsqueda nodo a nodo, paso a paso, encontraras la solución con esos algoritmos si existe.

Cualquier decisión miro si ya está probada, si es asi miro en otro estado, otro nodo, el árbol es un árbol de soluciones.

Heuristic Serach: Busqueda más informal, es una búsqueda realizado por que alguien más me lo ha dicho, porque alguien más que conoce ya me lo ha dicho, es con información previa, esta solución es la mejor solución pero que tienes a mano, no es la mejor del total.

Hill Climbing: Es una versión mejorada de generate and test, genera una solución con un criterio añadido en base a conocimiento previo, pero acá se va a producir una realimentación que será añadida en el siguiente paso de búsqueda. La búsqueda llega hasta que empieza a empeorar la solución. (Guardo y repito nodos guardados en el pasado).

En hill climbing: el goal es el nodo hasta donde mi rendimiento ha sido mejor, pero el siguiente decae el rendimiento, entonces me detengo.

En hill climbing puedes llegar a un local extremo, donde al parecer si continua iterando la decisión es peor, pero eso solo es al inicio de la siguiente iteración, pero después de algunas iteraciones continua esta solución mejorando su desempeño para ello, se recomiendo por el camino reocrrido, slatar unas posiciones más adelantes, con el fin de obiar ese camino de bajo rendimiento

Simulated Annealing: Es una mejora de hill climbing, consiste en varios pasos mas adelante, e ir descendiendo, las desiciones futuras permiten desiciones peores, porque va retrocediendo, y te permite salir de locales máximos.

Sus carateristicas son:

Movimientos peores se aceptan.

El pasado puede ser mejor que el presente.

Best-first: Toma lo mejor de los algoritmos depth first y breadth first (primero lo ancho, primero lo profundo).

Funcion heurística:

F(n) = g(n) + h(n)

g(n) costo que pague para que llegue a n

h(n) costo que voy a pagar para llegar desde n al objetivo.

Greedy search:

Voy a ignorar mi propia experiencia, y solo voy a tener en cuenta la experiencia de terceros.