DEGREE IN COMPUTER ENGINEERING AND VIDEOGAMES

INTELLIGENT SYSTEMS

FINAL ASSIGNMENT

Made by:

Javier Jiménez Tomás
Álvaro Tomás Lozano

4º Videogames and computers

18/01/2020

- (5 points) Describe the TABU, GENETIC and A* Search algorithm

    1. In which category do you classify it. Indicates advantages and disadvantages with the algorithms that have been studied in the classes

    2. Which advantages and disadvantages do you discover in this algorithm to solve the traveling salesman problem.

TABU ALGORITHM

Los orígenes de "Tabu" se remontan a finales de los 70. Tanto el nombre y la metodología fueron originados por Fred Glover en dos artículos en 1989. Es una de las estrategias que intentan usar la memoria del proceso de búsqueda para mejorar el rendimiento evitando que la búsqueda se concentre en el mismo área. En los origenes del método, el propósito era sólo evitar la repetición en la misma área de búsqueda recordando las últimas soluciones visitadas. Sin embargo, posteriormente se han hecho numerosas propuestas para hacer la memoria a medio o largo plazo rentable.

Tabu Search se caracteriza por varios aspectos, alguno de ellos son que tiene un mecanismo de generación de "vecinos" modificado que evita la exploración de áreas del espacio de búsqueda que ya se ha visitado: generación de tabú restringido a entornos. Otro de ellas es utilizar mecanismos para mejorar la capacidad del algoritmo para explorar o "explotar" el espacio buscado. Para poder llevar a cabo las tareas anteriormente mencionadas, hace uso de diferentes estructuras adaptables de memoria como la memoria a corto plazo (lista tabu), que permite guiar la búsqueda de inmediato, desde el comienzo del procedimiento, y por otro lado la memoria a largo plazo, que se encarga de almacenar la información que permite guiar la búsqueda, después de una primera etapa en la que una o más ejecuciones del algoritmo en la cual la memoria a corto plazo ha sido aplicada.

La información almacenada en esta memoria puede ser usada para intensificar la búsqueda regresando a las regiones buenas ya exploradas para estudiarlas más a fondo. Para esto, la aparición de esos atributos asociados con buenas soluciones encontradas son apreciados. Además con ella también podemos diversificar la búsqueda, es visitar nuevas áreas inexploradas del espacio de soluciones. Para hacer esto, las reglas de elección se modifican para incorporar atributos que no han sido usados con frecuencia.

1.A-CLASIFICACIÓN

Podemos clasificar el algoritmo Tabu dentro de métodos metaheurísticos pues su finalidad es obtener mejores resultados que los logrados por estos métodos tradicionales.Dentro de estos métodos metaheurísticos podemos clasificar un

algoritmo Tabu dentro de los métodos de búsqueda de los que ahora vamos a hablar.
Son métodos que presuponen que existe una solución y realizan procedimientos de búsqueda, la diferencia con los métodos analíticos es que no necesariamente encontrará la solución óptima. Uno de los riesgos al usar un algoritmo de búsqueda es alcanzar un óptimo local a partir del cual ya no es posible salir.

1.B-VENTAJAS

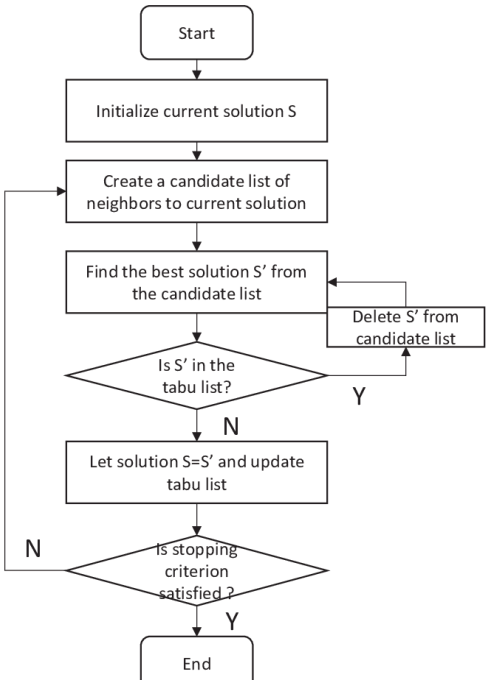
La cantidad de pesos que el procedimiento de entrenamiento necesita ajustar se reduce en "m + 1".

1.C-DESVENTAJAS

El modelo de regresión debe resolverse cada vez que cualquiera de los primeros "m (n + 1)" pesos se cambia para calcular el error cuadrático medio.

2-PROBLEMA DEL VIAJANTE

La búsqueda tabú puede utilizarse para hallar una solución satisfactoria para dicho problema. Primero, la búsqueda tabú comienza con una solución inicial, que puede ser generada con el algoritmo del vecino más cercano. Para establecer nuevas soluciones, el orden en que dos ciudades son visitadas es intercambiado. La distancia total recorrida entre todas las ciudades es utilizada para calificar cuánto mejor es una solución que otra. Para prevenir ciclos y para salir de los óptimos locales, una solución es agregada a la lista tabú si es que es aceptada en N*(x), el vecindario de soluciones. Se continúan creando nuevas soluciones hasta que se cumple algún criterio de parada, como por ejemplo un número arbitrario de iteraciones. Una vez que la búsqueda tabú se detiene, la mejor solución es aquella que cuya distancia total a recorrer entre las ciudades es la menor. Los inconvenientes de este método son principalmente aquellos asociados por la lista tabu, que cuanto mayor sea mayor mejora habrá, pero cuando la diferencia entre instancias sea muy alta, será menos apreciable, en cambio si no lo es, podremos ver las mejoras de una manera efectiva.


Start
Initialize current solution S
Create a candidate list of neighbors to current solution
Find the best solution S' from the candidate list
Delete S' from candidate list
Is S' in the tabu list?
Y
N
Let solution S=S' and update tabu list
Is stopping criterion satisfied ?
N
Y
End

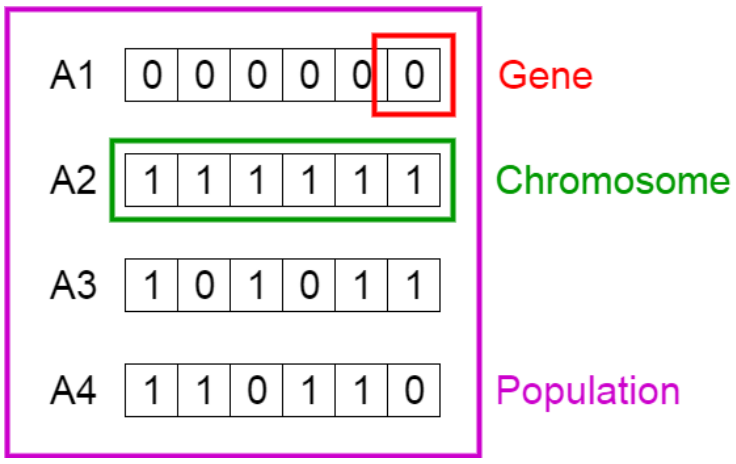
GENETIC ALGORITHM

Un algoritmo genético es una búsqueda heurística inspirada en la teoría de la evolución natural de Charles Darwin. Este algoritmo refleja el proceso de selección natural donde se seleccionan los individuos más aptos para la reproducción con el fin de producir descendencia de la próxima generación.

El proceso de selección natural comienza con la selección de los individuos más aptos de una población. Producen descendientes que heredan las características de los padres y se agregaran a la próxima generación. Si los padres tienen una mejor condición física, sus hijos serán mejores que los padres y tendrán una mejor oportunidad de sobrevivir. Este proceso continúa iterando y, al final, se encontrará una generación con los individuos más aptos. Esta noción se puede aplicar para un problema de búsqueda. Consideramos un conjunto de soluciones para un problema y seleccionamos el conjunto de las mejores entre ellas.Se consideran cinco fases en un algoritmo genético:

1. **Población inicial:**

El proceso comienza con un conjunto de individuos que se llama población. Cada individuo es una solución al problema que desea resolver.Un individuo se caracteriza por un conjunto de parámetros (variables) conocidos como Genes. Los genes se unen en una cadena para formar un cromosoma (solución).En un algoritmo genético, el conjunto de genes de un individuo se representa usando una cadena, en términos de un alfabeto. Por lo general, se utilizan valores binarios (cadena de 1s y 0s). Decimos que codificamos los genes en un cromosoma.



2. **Función "Fitness" o de condición física:**

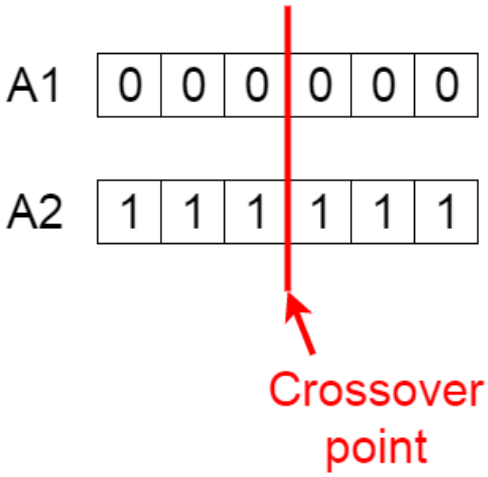
La función de condición física determina qué tan en forma está un individuo (la capacidad de un individuo para competir con otros individuos). Le da un puntaje de aptitud a cada individuo. La probabilidad de que un individuo sea seleccionado para la reproducción se basa en su puntaje de aptitud física.

3. **Selección:**

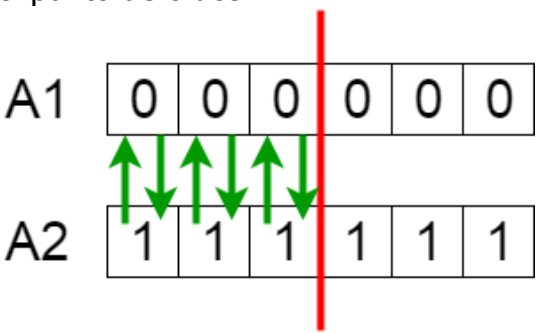
   La idea de la fase de selección es seleccionar a los individuos más aptos y dejarlos pasar sus genes a la próxima generación.Se seleccionan dos pares de individuos (padres) en función de sus puntajes de condición física. Las personas con buena condición física tienen más posibilidades de ser seleccionadas para la reproducción.

4. **Crossover:**

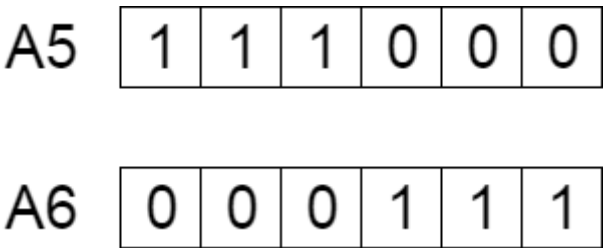
   El cruce o crossover es la fase más significativa en un algoritmo genético. Para cada pareja de padres que se van a unir, se elige un punto de cruce al azar dentro de los genes.Por ejemplo, considere que el punto de cruce es 3 como se muestra a continuación.



La descendencia se crea intercambiando los genes de los padres entre ellos hasta que se alcanza el punto de cruce.

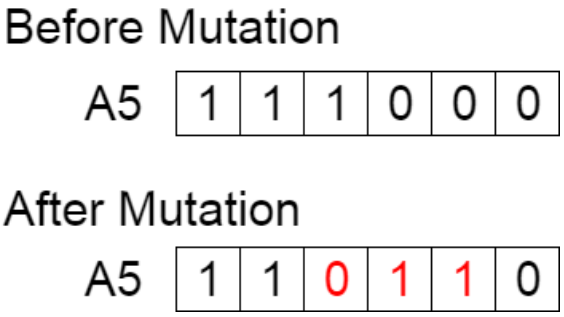


Los nuevos descendientes se agregan a la población.



5. **Mutación:**

En ciertos nuevos descendientes formados, algunos de sus genes pueden estar sujetos a una mutación con una probabilidad aleatoria baja. Esto implica que algunos de los bits en la cadena de bits se pueden voltear.



La mutación ocurre para mantener la diversidad dentro de la población y prevenir la convergencia prematura.

1.A-CLASIFICACIÓN

Podemos clasificar el algoritmo genético dentro de los métodos metaheurísticos igual que el algoritmo Tabu ya que su finalidad es obtener mejores resultados que los logrados por los métodos heurísticos tradicionales.Dentro de estos métodos metaheurísticos podemos clasificar un algoritmo genético dentro de los métodos evolutivos; son métodos que están creando un conjunto de soluciones a diferencia de los otros métodos que simplemente funcionan de una solución a otra en cada iteración.El procedimiento consiste en generar, seleccionar, combinar y reemplazar un conjunto de soluciones.

1.B-VENTAJAS

La exploración aleatoria puede encontrar soluciones que la búsqueda local no puede, y además es un método curioso que relaciona la genética con el área de la programación.

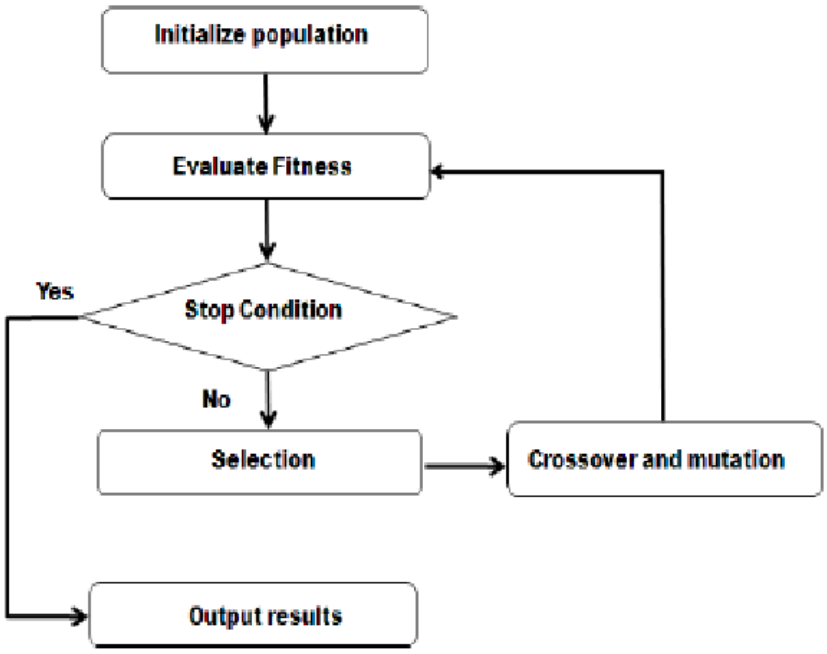
1.C-DESVENTAJAS

Debido a su gran cantidad de variables es complicado replicar el rendimiento de un problema a otro, además de su falta de buenos estudios empíricos en comparación con otros métodos más simples. Otra ventaja en comparación con el método tabú estudiado anteriormente sería que es más rápido, es decir, en una interacción entre padres avanzas mucho más que con el algoritmo de tabú.

2-PROBLEMA DEL VIAJANTE

El algoritmo genético genera soluciones óptimas que permiten tener un control de la población y de cada uno de los individuos a través de los operadores

genéticos y genera una mejor solución, lo que indica que es un algoritmo eficiente.



A* ALGORITHM

En inglés se pronuncia algoritmo “A star”, así que lo llamaremos algoritmo de “A estrella”. A estrella es un algoritmo informático que se utiliza ampliamente en la búsqueda de rutas y el recorrido de gráficos, que es el proceso de encontrar una ruta entre puntos múltiples, llamados "nodos". Disfruta de un uso generalizado de su rendimiento y precisión. Sin embargo, en los sistemas prácticos de enrutamiento de viaje, generalmente es superado por algoritmos que pueden preprocesar el gráfico para lograr un mejor rendimiento, aunque otro trabajo ha encontrado que A estrella es superior a otros enfoques.

El secreto de su éxito es que combina las piezas de información que el algoritmo Dijkstra utiliza (favoreciendo vértices que están cerca del punto de partida) e información que “Greedy Best-First-Search” utiliza (favorece los vértices que están cerca de la meta). En el estándar usado cuando se habla de A *, g (n) representa el costo exacto de la ruta desde el punto de partida para cualquier vértice n, y h (n) representa el costo heurístico estimado del vértice na la meta. En los diagramas anteriores, el amarillo (h) representa vértices lejos del objetivo y el trullo (g) representa vértices lejos del punto de partida. A * equilibra los dos a medida que se mueve desde el punto de partida para la meta. Cada vez a través del bucle principal, examina el vértice “n” que tiene el más bajo f (n) = g (n) + h (n).

1.A-CLASIFICACIÓN

Podríamos decir que pertenece al conjunto de métodos heurísticos y dentro de ellos puede calificarse como un algoritmo genérico de búsqueda.

1.B-VENTAJAS

Es un algoritmo que elegirá el nodo de mejor coste.

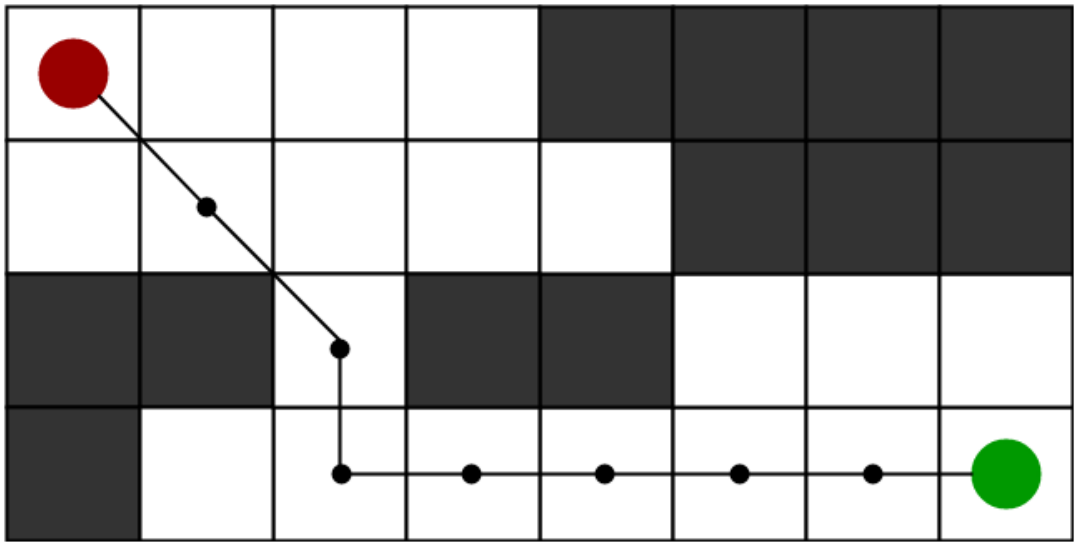
1.C-DESVENTAJAS

Por otro lado tarda quizás demasiado en términos de computo.

2-PROBLEMA DEL VIAJANTE:

Generalmente es superado por algoritmos que pueden preprocesar el gráfico para lograr un mejor rendimiento, debido a que el viaje desde este algoritmo comienza desde cualquier otro nodo, sin embargo, estos algoritmos buscan encontrar el camino más corto entre dos puntos,independientemente del número de nodos visitados. De hecho, dependen del hecho de que la ruta más corta de X a Y a través de algún nodo Z, la ruta de X a Z es irrelevante si tiene el mismo costo.

Aún así se podría generar un nuevo grafo representando los nodos visitados y con el algoritmo de A estrella ser capaz de encontrar una heurística que corra en una cantidad asequible de tiempo.



¿Qué es un problema de entrenamiento excesivo y cómo resolverlo en redes neuronales y especialmente en Keras?

Si entrenas durante demasiado tiempo un modelo, el modelo comenzará a sobreajustarse(overfitting) y a aprender patrones de los datos de entrenamiento que no se generalizan a los datos de la prueba. Necesitamos encontrar un equilibrio. Comprender cómo entrenar para un número apropiado de intentos (epochs) como explicaremos a continuación es una habilidad útil.

Para evitar el sobreajuste, la mejor solución es utilizar datos de entrenamiento más completos. El conjunto de datos debe cubrir el rango completo de entradas que se espera que maneje el modelo. Los datos adicionales solo pueden ser útiles si cubren casos nuevos e interesantes.

Un modelo entrenado en datos más completos naturalmente generaliza mejor. Cuando eso ya no sea posible, la siguiente mejor solución es utilizar técnicas como la regularización. Estas restricciones imponen la cantidad y el tipo de información que el modelo puede almacenar. Si una red solo puede permitirse memorizar una pequeña cantidad de patrones, el proceso de optimización la obligará a centrarse en los patrones más destacados, que tienen una mejor oportunidad de generalizar bien y evitaremos de esta manera tener que sobreajustar (overfit) nuestro modelo.

En conclusión, estas son las formas más comunes de prevenir el sobreajuste en redes neuronales y sobre todo en Keras:

- **Tener más datos de entrenamiento.**

- **Reducir la capacidad de la red.**

- **Añadir regularización de pesos. (weight regularization)**
  Una forma común de mitigar el sobreajuste es imponer restricciones a la complejidad de una red forzando sus pesos solo a tomar valores pequeños, lo que hace que la distribución de los valores de peso sea más "regular". Esto se llama "regularización de peso", y se realiza agregando a la función de pérdida de la red un costo asociado con tener grandes pesos. Este costo viene en dos "condimentos": la regularización de L1 y L2.

  En tf.keras, la regularización de peso se agrega al pasar instancias de regulador de peso a capas como argumentos clave.

- **Añadir "Dropout"**

El "Dropout" es una de las técnicas de regularización más efectivas y más utilizadas para redes neuronales, desarrollada por Hinton y sus estudiantes en la Universidad de Toronto.La explicación intuitiva para el abandono O "Dropout" es que debido a que los nodos individuales en la red no pueden confiar en la salida de los demás, cada nodo debe generar características que sean útiles por sí mismas.

- **Aumento de datos (Data-augmentation)**

- **Normalización de lotes (batch normalization)**

Da un ejemplo original de razonamiento basado en el encadenamiento hacia atrás (backward chaining)

El encadenamiento hacia atrás[backward chaining] (o el razonamiento hacia atrás [backward reasoning]) es un método de inferencia descrito coloquialmente como trabajando hacia atrás desde la meta (working backward from the goal).

En IA, el encadenamiento "backward chaining" se usa para encontrar las condiciones y reglas por las cuales un resultado lógico o conclusión fue obtenido. Una IA puede utilizar el encadenamiento hacia atrás para encontrar información relacionada con conclusiones o soluciones en ingeniería inversa o teoría de juegos/aplicaciones.

Una vez explicado en qué consiste el "backward chaining", podemos empezar con nuestro ejemplo:

Imaginemos que vamos a un restaurante y nos va a servir el camarero pronto "la especialidad de la casa", y nos venda los ojos para que descubramos con nuestro paladar un plato que todavía no sabemos qué es, una vez llega sólo sabemos dos cosas sobre la especialidad:

- Sabe dulce.
- Su textura es blanda.

El objetivo es descubrir si la comida es un postre, basándonos en una base de reglas (inventadas) que son estas cuatro:

1. Si X es dulce y X es de textura blanda - **Entonces es tarta.**
2. Si X es salado y X es de textura más líquida - **Entonces es una sopa.**
3. Si X es una tarta - **Entonces X es un postre.**
4. Si X es una sopa - **Entonces X es un primero.**

Con “backward chaining”, un motor de inferencia puede determinar si la especialidad es un postre en cuatro pasos. Para comenzar, la consulta está redactada como una afirmación de objetivo que debe probarse: "La especialidad es un postre".

1. La especialidad es sustituida por X en la regla #3 para ver si su consecuente coincide con el objetivo, entonces la regla #3 se convierte en:

Si la especialidad es una tarta - Entonces la especialidad es un postre.

Como el consecuente coincide con el objetivo ("La especialidad es un postre"), el motor de reglas ahora necesita ver si se puede probar el antecedente ("La especialidad es una tarta"). El antecedente se convierte en el nuevo objetivo:

La especialidad es una tarta.

2. De nuevo sustituimos la especialidad por X, y la regla #1 se queda así:

Si la especialidad es dulce y la especialidad es de textura blanda - Entonces la especialidad es una tarta.

Como el consecuente coincide con el objetivo actual ("La especialidad es una tarta"), el motor de inferencia ahora necesita ver si se puede probar el antecedente ("La especialidad es dulce y tiene una textura blanda”). El antecedente se convierte en el nuevo objetivo:

La especialidad es dulce y la especialidad es de textura blanda.

3. Dado que este objetivo es una conjunción de dos declaraciones, el motor de inferencia lo divide en dos submetas, las cuales deben probarse:

La especialidad es dulce
La especialidad es de textura blanda

4. Para probar estos dos objetivos secundarios, el motor de inferencia ve que ambos objetivos secundarios se dieron como hechos iniciales. Por lo tanto, la conjunción es verdadera:

La especialidad es dulce y la especialidad es de textura blanda.

Por lo tanto, el antecedente de la regla #1 es verdadero y el consecuente debe ser verdadero:

La especialidad es una tarta.

Por lo tanto, el antecedente de la regla #3 es verdadero y el consecuente debe ser verdadero:

La especialidad es un postre.

Por lo tanto, esta derivación permite que el motor de inferencia demuestre que la tarta es un postre. Las reglas # 2 y # 4 no fueron utilizadas. Hay que tener en cuenta que los objetivos siempre coinciden con las versiones afirmadas de los consecuentes de las implicaciones (y no con las versiones negadas como en el modus tollens) e incluso entonces, sus antecedentes se consideran como los nuevos objetivos (y no las conclusiones como para afirmar el consecuente), que finalmente debe coincidir con hechos conocidos (generalmente definidos como consecuentes cuyos antecedentes son siempre verdaderos); por lo tanto, la regla de inferencia utilizada es modus ponens.

Debido a que la lista de objetivos determina qué reglas se seleccionan y usan, este método se llama dirigido por objetivos, en contraste con la inferencia de encadenamiento hacia adelante basada en datos. El enfoque de encadenamiento hacia atrás a menudo es empleado por sistemas expertos (Expert Systems).

Queremos resolver el problema del viajante, ¿cómo lo resolverías? ¿Qué algoritmo usarías y por qué? ¿Cómo debe codificarse el problema para resolverlo con el algoritmo de "búsqueda tabú"?

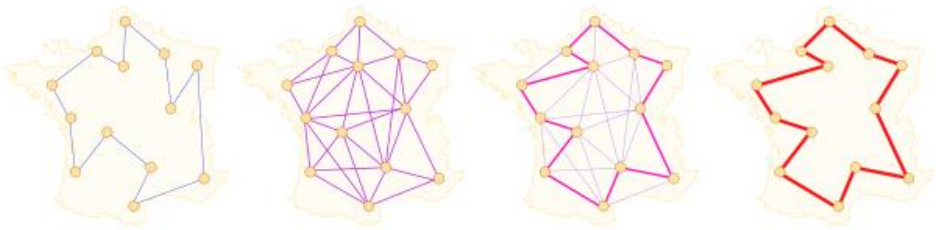
OPTIMIZACIÓN POR COLONIA DE HORMIGAS (ACS)

Queremos una solución lo más ideal posible, por lo que para empezar desechar opciones como el algoritmo del vecino más próximo sería una opción, puesto que es un método que a pesar de generar rápidamente un camino corto, no es generalmente el ideal. Nosotros hemos optado por elegir dentro del campo de los algoritmos heurísticos y aproximados, la mejora aleatoria de la optimización por Colonia de Hormigas (ACS).

El investigador de Inteligencia artificial, Marco Dorigo describió en 1997 un método que genera heurísticamente “buenas soluciones” para el TSP usando una simulación de una colonia de hormigas llamada ACS (Ant Colony System). El cual modela el comportamiento observado en las hormigas reales de encontrar caminos cortos entre las fuentes de comida y su nido, emergió como un comportamiento la preferencia de cada hormiga de seguir el sendero de feromonas depositado por las otras hormigas.

ACS envía un gran número de hormigas (agentes virtuales) para explorar las posibles rutas en el mapa. Cada hormiga elige probabilísticamente la próxima ciudad a visitar basada en una heurística, combinando la distancia a la ciudad y

la cantidad de feromonas depositadas en la arista hacia la ciudad. La hormiga exploradora, deposita feromonas en cada arista que ella cruce, hasta que complete todo el camino. En este punto la hormiga que completó el camino más corto deposita feromonas virtuales a lo largo de toda la ruta recorrida (actualización del camino global). La cantidad de feromonas depositadas es inversamente proporcional a la longitud del camino: el camino más corto, tiene más cantidad de feromonas.



1 2 3 4

Elegimos este método/algoritmo puesto que sus métodos son de grafos y se basan en un componente biológico tal como el de las hormigas, para poder hacer un diseño muy efectivo para encontrar caminos óptimos.

BÚSQUEDA TABÚ DEL PROBLEMA DEL VIAJANTE

La búsqueda tabú puede utilizarse para hallar una solución satisfactoria para dicho. Primero, la búsqueda tabú comienza con una solución inicial, que puede ser generada con el algoritmo del vecino más cercano. Para establecer nuevas soluciones, la disposición en que dos ciudades son visitadas es intercambiado. La distancia general recorrida entre todas las ciudades es utilizada para calificar cuánto superior es una solución que otra. Para prevenir ciclos y para salir de los óptimos locales, una solución es agregada a la lista tabú si es que es aceptada en N*(x), el vecindario de soluciones. Se continúan creando nuevas soluciones hasta que se cumple algún criterio de parada, como por ejemplo un dígito arbitrario de iteraciones. Una vez que la búsqueda tabú se detiene, la mejor solución es aquella que cuya distancia general a recorrer entre las ciudades es la menor.

BIBLIOGRAFÍA

- **El E-Book: "Metaheuristic Procedures for Training Neural Networks".**
- https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3
- https://medium.com/@becmjo/genetic-algorithms-and-the-travelling-salesman-problem-d10d1daf96a1
- https://github.com/inanoxTM/A-STAR-search-algorithm-implemented-in-ROS
- https://www.algorithmhalloffame.org/algorithms/a-star-search/
- http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html
- https://www.tensorflow.org/tutorials/keras/overfit_and_underfit
- https://whatis.techtarget.com/definition/backward-chaining
- https://en.wikipedia.org/wiki/Backward_chaining
- https://es.wikipedia.org/wiki/Algoritmo_de_la_colonia_de_hormigas