Ayudantía Tarea 1

Resolución de problemas mediante búsquedas

Agustín Ghent
agustinghent@alumnos.uai.cl

Qué haremos hoy

- Repasar el concepto de **búsqueda heurística.**
- Introducir **Traveling Salesman Problem (TSP)**
- Profundizar en el método **Simulated Annealing (SA)**
- **Implementación del método SA a un caso práctico (TAREA 1)**

OJO aquí

- Introduciremos (algo tarde) el problema que se representa en la TAREA 1
- **Lo ideal aquí es que entiendan y *repliquen.***

Problemas de búsqueda

Muchos problemas interesantes en ingeniería se resuelven usando búsquedas

Un problema de búsqueda se compone de:

- **Un espacio de búsquedas:**
  - Set de objetos entre los cuales nosotros buscamos la solución

    Ejemplo: Rutas que conectan las ciudades de V región
- **Una condición objetivo:**
  - Características del objeto el cual queremos encontrar en el espacio de búsqueda

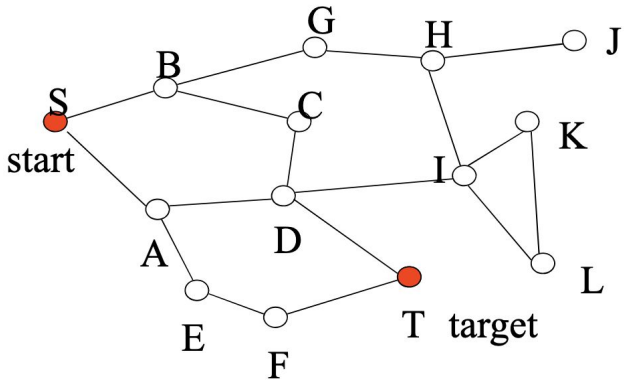
    Ejemplo: Ruta entre Quilpué y San Antonio

Representación de un problema de búsqueda

Los problemas de búsqueda a menudo pueden ser representados utilizando grafos

- **Ejemplo típico: Búsqueda de ruta**

  Objetivo: Encontrar una ruta (secuencia de movimientos) en el grafo de S a T



Representación de un problema de búsqueda

Para representar un problema de búsqueda necesitamos

- **Set de estados**
  - Set de objetos los cuales buscamos la solución (e.j. ciudad de inicio)
- **Estado inicial:**
  - Estado donde comenzamos a buscar (e.j. ciudad de inicio)
- **Estado objetivo**
  - Estado que buscamos llegar (e.j. ciudad destino)
- **Función sucesor:**
  - Función que permiten pasar de un estado a otro (e.j. conexiones válidas entre ciudades)

Tipos de búsqueda

Normalmente se utilizan dos tipos de búsqueda, donde la diferencia es algo sutil:

- **Búsqueda ciega:**
  - Hace referencia a las estrategias en las que se evalúa el estado siguiente sin conocer a priori si es mejor o peor que su predecesor
- **Búsqueda heurística: (*):**
  - Definición Heurística:
    - **"Un proceso que puede resolver un problema dado, pero que no ofrece ninguna garantía de que lo hará, se llama una heurística para ese problema"**
  - **Los métodos de búsqueda heurística disponen de alguna información sobre la proximidad de cada estado a uno objetivo.**

Búsqueda heurística:

- **No garantizan que se encuentre una solución: (aunque las haya):**
- **Si se descubre una solución, no se puede asegurar de que tenga las mejores propiedades.**
- **En algunas ocasiones (no se puede determinar ex ante), encontrarán una solución (razonablemente buena)**

Ejemplo: Simulated annealing (SA)

Ejemplo: Simulated annealing (SA)

Definición:

- **SA es un algoritmo de búsqueda heurística para problemas de optimización global. Su origen se basa en una solución para el proceso de enfriamiento del metal.**

Objetivo:

- **El el objetivo es encontrar una "buena" aproximación al valor óptimo de una función en un espacio de búsqueda.**

- **El óptimo global corresponde a la solución del problema de interés para el que no existe un mejor valor. En caso de un problema de minimización, el óptimo será cual la función tenga el más pequeño posible de todos los de su espacio de búsqueda.**

Resúmen Algoritmo SA

Sea *f(s)* el coste de la solución s y sea *G(s)* su entorno.
Seleccionar una solución inicial S*;*
Seleccionar una temperatura inicial $T_i > 0$;
Seleccionar una función de reducción de la temperatura α;
Seleccionar un número de iteraciones *N;*
Seleccionar un *criterio de parada;*
Repetir
    Repetir
        Seleccionar aleatoriamente una solución $S' \in G(s)$;
        Sea $\Delta f = f(S') - f(S)$;
        SI $\Delta f < 0$ Entonces $S_{OPTIMO} = S'$;
        Si no
            Generar aleatoriamente $t \in L(0,1)$;
            SI $t < e^{(\Delta f / T)}$ Entonces $S_{OPTIMO} = S'$;
        Fin Si no
    Hasta que iteraciones = *N*
    $T_{i+1} = T_i * \alpha$;
Hasta que criterio de parada = Cumpla

La mejor solución encontrada será la solución dada por el algoritmo

Algoritmo SA

El funcionamiento del SA se puede describir de la siguiente manera:

1. **Comienza con un cierto estado S.**
2. **A través de un proceso único crea un estado vecino S′ al estado inicial.**
3. **Si la energía o la evaluación del estado S′ son menores que el estado S cambia el estado S por S′.**
4. **Si la evaluación de S′ es mayor que la de S puede estar empeorando, por lo que elige S′ en vez de S con una cierta probabilidad que depende de las diferencias en las evaluaciones y la temperatura del sistema T.**

La probabilidad de aceptar de aceptar un estado peor se calcula como:

$$P(\Delta f, T) = e^{(\Delta f/T)}$$

- Δf: diferencia de las evaluaciones de la función para cada estado.
- T: temperatura del sistema

Algoritmo SA

El funcionamiento del SA se puede describir de la siguiente manera:

7. Inicialmente, con valores grandes para T, frecuentemente se aceptan soluciones con un mayor valor de función objetivo.

8. A medida que el valor de T disminuye, tal tipo de soluciones raramente se aceptan, y cuando T se acerca a cero, solo se aceptan aquellas soluciones que mejoran la anterior.

Varios estudios teóricos demuestran que si T decrece con la suficiente lentitud, el proceso converge a la solución óptima.

La función para reducción de temperatura más utilizada es: $T_{k+1}=T_k \alpha$

α es una constante entre intervalo [0,8-0,99].

Resúmen Algoritmo SA

El funcionamiento del SA se puede describir de la siguiente manera:

7. Inicialmente, con valores grandes para T, frecuentemente se aceptan soluciones con un mayor valor de función objetivo.

8. A medida que el valor de T disminuye, tal tipo de soluciones raramente se aceptan, y cuando T se acerca a cero, solo se aceptan aquellas soluciones que mejoran la anterior.

Varios estudios teóricos demuestran que si T decrece con la suficiente lentitud, el proceso converge a la solución óptima.

La función para reducción de temperatura más utilizada es: $T_{k+1}=T_k \alpha$

α es una constante entre intervalo [0,8-0,99].

Resúmen Algoritmo SA

Sea *f(s)* el coste de la solución s y sea *G(s)* su entorno.
Seleccionar una solución inicial S*;*
Seleccionar una temperatura inicial $T_i > 0$;
Seleccionar una función de reducción de la temperatura α;
Seleccionar un número de iteraciones *N;*
Seleccionar un *criterio de parada;*
Repetir
    Repetir
        Seleccionar aleatoriamente una solución $S' \in G(s)$;
        Sea $\Delta f = f(S') - f(S)$;
        SI $\Delta f < 0$ Entonces $S_{OPTIMO} = S'$;
        Si no
            Generar aleatoriamente $t \in L(0,1)$;
            SI $t < e^{(\Delta f / T)}$ Entonces $S_{OPTIMO} = S'$;
        Fin Si no
    Hasta que iteraciones = *N*
    $T_{i+1} = T_i * \alpha$;
Hasta que criterio de parada = Cumpla

La mejor solución encontrada será la solución dada por el algoritmo

Travel Salesman Problem (TSP)

https://algorithms.discrete.ma.tum.de/graph-games/tsp-game

Travel Salesman Problem (TSP)

TSP es un clásico problema de optimización donde se plantea la situación:

Un vendedor debe visitar un set de ciudades exactamente una vez y retornar al punto inicial mientras minimiza el total de la distancia recorrida.

- **El objetivo es determinar la ruta más corta que visita cada ciudad a la vez y que regresa a la ciudad de origen.**
- **Dado que es un problema NP-Complejo, las soluciones exactas poseen una alta complejidad computacional que aumenta a medida N es grande.**

Resolvamos TSP usando Python

Steps for Implementing Simulated Annealing

1. Initialize the current route and calculate its distance.
2. Set the initial temperature and cooling rate.
3. Iterate for a specified number of iterations:
    - Generate neighboring solutions.
    - Select a neighbor and calculate its distance.
    - Accept the neighbor based on a probability that decreases with temperature.
    - Update the best solution found so far.
4. Return the best route and its distance.

https://www.geeksforgeeks.org/hill-climbing-and-simulated-annealing-for-the-traveling-salesman-problem/