Modelos Bioinspirados y
Heurísticas de Búsqueda
4 Curso Grado Ingeniería
en Informática
Área de Ciencias de la Computación e
Inteligencia Artificial
Departamento de
Tecnologías de la Información

# PRÁCTICA 1 (Versión 2021, 2.0)

# Algoritmos basados en Entornos y Trayectorias

## Objetivos

El objetivo de esta práctica es estudiar el funcionamiento de los *Algoritmos de Búsqueda Aleatoria, Local, Enfriamiento Simulado y Búsqueda Tabú*. Para ello, se requerirá que el alumno implemente estos algoritmos, para resolver el *Problema del Viajante de Comercio.* El comportamiento de los algoritmos de OCH implementados deberá compararse con un *Algoritmo Greedy*.

## Enunciado de la práctica

El Problema del Viajante de Comercio (TSP) es uno de los problemas de optimización combinatoria más conocidos. En su formulación más general, dadas una serie de ciudades, el objetivo consiste en encontrar el circuito de menor coste que parta de una ciudad concreta, pase por todas las demás una sola vez y retorne a la ciudad de origen.

En nuestro caso trabajaremos con 6 instancias del problema obtenidas de la biblioteca TSPLIB, todas ellas correspondientes al TSP simétrico (misma distancia o coste independientemente de si es en un sentido u otro). Serán las siguientes:

- St70: Tamaño 70 ciudades. Coste de la solución óptima: 675
- Ch130: Tamaño 130 ciudades. Coste de la solución óptima: 6.110
- A280: Tamaño 280 ciudades. Coste de la solución óptima: 2.579

Nivel Medio

- Pa654: Tamaño 654 ciudades. Coste de la solución óptima: 34.643
- Vm1084: Tamaño 1084 ciudades. Coste de la solución óptima: 239.297

Nivel Avanzado

- Vm1748: Tamaño 1748 ciudades.

Todos estos ficheros presentan el mismo formato, una lista con dos valores para cada ciudad que representan sus coordenadas en el plano. Para componer la matriz de costes se deberá calcular la *distancia euclídea* entre cada par de ciudades (*i, j*). Los costos obtenidos han de ser números enteros, es decir, no se considerarán decimales. Así, la matriz de distancias se calcularía, por ejemplo en lenguaje C, de la siguiente forma:

```
xd = x[i]  - x[j];
yd = y[i]  - y[j];
dij = rint( sqrt( xd*xd + yd*yd ) );
```

donde `rint` es la función de redondeo y `sqrt` es la raíz cuadrada.

A continuación, mostramos los valores de los parámetros a considerar en la experimentación de cada uno de estos algoritmos (parámetros, soluciones iniciales, condiciones de parada, etc).

**Algoritmo de comparación: *Greedy***

Para efectuar la comparativa de resultados entre los distintos algoritmos de búsqueda, se debe implementar como algoritmo básico, un *Greedy*, es decir, siguiendo la heurística del vecino más cercano: parte de una ciudad concreta, en nuestro caso la número 1, la elimina del conjunto de nodos, y escoge como su seguidora en el circuito aquella que se encuentre situada a menor distancia. Este proceso se repite iterativamente, empleando como ciudades alcanzables sólo el conjunto de las restantes, hasta que dicho conjunto este vacío, para finalmente contabilizar el retorno a la ciudad de origen (en esta práctica, con todos los algoritmos consideraremos el retorno al origen, lo cual hace la solución independiente de la ciudad elegida inicialmente).

**Búsqueda Aleatoria**

El Algoritmo de Búsqueda Aleatoria (BA) consistirá simplemente en generar aleatoriamente una solución en cada iteración, usando listas de nodos (o ciudades) candidatos no visitados (ya que no se pueden repetir).

La búsqueda aleatoria completa debe ejecutarse 10 veces, cada vez con una semilla distinta (por tanto, se deben anotar las 10 semillas que se utilizarán sistemáticamente), para el generador aleatorio, y para cada conjunto de datos con $1600 \cdot n$ llamadas a la función de evaluación.

**Búsquedas Locales**

Se implementará siguiendo los esquemas de ***el mejor***, y de ***el primer mejor*** vecino, según se explicó en el Tema 1 de teoría, usando el operador 2-opt.

Se partirá de una solución inicial aleatoria. Los algoritmos de búsqueda local tienen su propia condición de parada, pero adicionalmente, en prevención de tiempos excesivos en algún caso, se añadirá una condición de parada alternativa (OR) basada en el número de evaluaciones que esté realizando la búsqueda, es decir, el número de veces que se llame al cálculo de la función de coste. Este valor para la Búsqueda Local será de $1600 \cdot n$ llamadas a la función de coste.

La búsqueda con cada algoritmo se debe ejecutar 10 veces con semillas distintas, como en el caso de la Aleatoria.

Se propone como alternativa al primer mejor una implementación alternativa, quien quiera experimentar puede probar estaversión aleatorizado, probando con varios valores en el numero de vecinos que consideramos suficiente en cada exploración de un nuevo mínimo.

**Procedimiento Búsqueda Local del Mejor Vecino Aleatoreizado**
**Inicio**
GENERA($S_{act}$);
**Repetir**
I = 0 ;
**Repetir**
I++
S' ← GENERA_VECINO($S_{act}$);
**Hasta** (Objetivo(S') **mejor que** Objetivo(Mejor Vecino)) **O** (i>NumVecinosLimite)
Si Objetivo(S') **mejor que** Objetivo($S_{act}$) entonces
$S_{act}$ ← S';
**Hasta** (Objetivo(S') **peor o igual que** Objetivo($S_{act}$));
DEVOLVER($S_{act}$);
**Fin**

## Enfriamiento Simulado

Se ha de implementar el algoritmo de Enfriamiento Simulado (ES) con las siguientes componentes:

- *Esquema de enfriamiento*: Se implementará el esquema de *Cauchy*,

$$T_k = T_0/(1 + k)$$

- *Condición de enfriamiento* $L(T)$: Se enfriará la temperatura, y finalizará a la iteración actual, cuando se haya generado un número máximo de vecinos (independientemente de si han sido o no aceptados).
- *Condición de parada*: El algoritmo finalizará cuando se alcance un número máximo de iteraciones (enfriamientos).

Se calculará la temperatura inicial en función de la siguiente fórmula:

$$T_0 = \frac{\mu}{-\log(\phi)} \cdot C(S_i)$$

donde $T_0$ es la temperatura inicial, $C(S_i)$ es el costo de la solución inicial y $\Phi[0,1]$ es la probabilidad de aceptar una solución un $\mu$ por 1 peor que la inicial. En las ejecuciones se considerar $\Phi=\mu= 0,3$.

El número de soluciones generadas en cada temperatura será $L(T) = 20$ y el número de enfriamientos (iteraciones) será $80 \cdot n$.

Se debe repetir también 10 veces con distintas semillas, partiendo de una solución inicial aleatoria.

## Búsqueda Tabú

Se implementará la versión de BT utilizando una lista de movimientos tabú y tres estrategias de reinicialización.

Sus principales características son:

- *Estrategia de selección de vecino*: Consistirá en examinar 40 vecinos para coger el mejor de acuerdo a los criterios tabú.

- *Selección de estrategias de reinicialización*: La probabilidad de escoger la *reinicialización construyendo una solución inicial aleatoria* es 0,25, la de usar la *memoria a largo plazo al generar una nueva solución greedy* es 0,5, y la de utilizar la *reinicialización desde la mejor solución obtenida* es 0,25.

  Además de saltar a una solución concreta según las reinicializaciones comentadas, también se alterará un parámetro del algoritmo de búsqueda para provocar un cambio de comportamiento del algoritmo más efectivo. Este consistirá en variar el tamaño de la lista tabú, incrementándola o reduciéndola en un 50% según una decisión aleatoria uniforme.

El número máximo de iteraciones en total, incluyendo las reinicializaciones, será de 40 · n. Se realizarán 4 reinicializaciones, es decir, una cada 8 · *n* iteraciones. El tamaño inicial de cada lista tabú será *n*=2, estos valores cambiarán después de las reinicializaciones según se ha comentado.

Dado el carácter probabilístico del algoritmo, deberá ejecutarse 10 veces (con semillas distintas para el generador aleatorio, para cada caso del problema (*dataset*).

## Metodología de Comparación

El alumno tendrá que contabilizar el número de evaluaciones (llamadas realizadas a la función de coste) producidas por los distintos algoritmos, que será empleado como una medida adicional de comparación de su calidad.

A partir de la experimentación efectuada, se obtendrán varias tablas en las que se incluirán los resultados obtenidos por cada algoritmo en cada instancia, así como la media y desviación típica de estos, con la forma de la Tabla 1.1.

| | St70 | | Ch130 | | A280 | | Pa654 | | Vm1084 | | (Vm1748) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Coste | #Ev. | Coste | #Ev. | Coste | #Ev. | Coste | #Ev. | Coste | #Ev. | Coste | #Ev. |
| Ejecución1 | | | | | | | | | | | | |
| Ejecución2 | | | | | | | | | | | | |
| Ejecución3 | | | | | | | | | | | | |
| Ejecución4 | | | | | | | | | | | | |
| Ejecución5 | | | | | | | | | | | | |
| Ejecución6 | | | | | | | | | | | | |
| Ejecución7 | | | | | | | | | | | | |
| Ejecución8 | | | | | | | | | | | | |
| Ejecución9 | | | | | | | | | | | | |
| Ejecución10 | | | | | | | | | | | | |
| Media | | | | | | | | | | | | |
| Des.Tip. (s) | | | | | | | | | | | | |

Tabla 1.1 Tabla para 5/6 casos o problemas

Finalmente, se construirá una tabla global de resultados con la estructura mostrada en la Tabla 1.2. El dato entre paréntesis debajo del nombre de la instancia en la Tabla 1.2 muestra el coste de la solución óptima (o la mejor encontrada que se conoce a priori) de la instancia, valor que debe ser considerado en el análisis de resultados. La columna etiquetada con *Mej* indica el valor de la mejor solución encontrada, la columna σ refiere a la desviación típica y la columna etiquetada con *Ev* indica el número medio de

evaluaciones realizadas por el algoritmo en las cinco ejecuciones (salvo en el caso del algoritmo *Greedy* que sólo se ejecuta una vez).

Si fuera necesario, las Tablas 1.1 y 1.2 pueden subdividirse en tablas que contengan grupos de datasets (subdividir por columnas) preferentemente (no por grupos de problemas: preferimos siempre ver siempre los distintos algoritmos aplicados al mismo problema).

A partir de los datos mostrados en estas tablas, el alumno realizará un **análisis de los resultados obtenidos**, **que influirá de forma decisiva en la calificación de la práctica**. En dicho análisis, se deben comparar, para cada instancia, las distintas variantes de los algoritmos en términos de: número de evaluaciones, mejor resultado individual obtenido y mejor resultado medio (robustez del algoritmo). El análisis tendrá dos apartados: *resultados observados* (describir lo que se ve en las tablas), y *análisis de los resultados* propiamente (analizar porqué salen esos resultados).

| | St70 (675) | | | | Ch130 (6110) | | | | A280 (2579) | | | | Pa654 (34.643) | | | | Vm1084 (239.297) | | | | (Vm1748) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Modelo** | Med | Mej | s | Ev | Med | Mej | s | Ev | Med | Mej | s | Ev | Med | Mej | s | Ev | Med | Mej | s | Ev | Med | Mej | s | Ev |
| Greedy | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Aleatoria | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| BA Mejor | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| BA Primer Mejor | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| ES | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| BT | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

Tabla 1.2 Resultados globales de los algoritmos

Las prácticas se realizarán individualmente.

**Fecha y Método de Entrega;**

El día 7 del Abril durante la sesión de prácticas. Debe entregar 1 fichero comprimido ZIP, que contenga:

- Documento DOC (MS Word) ó PDF con las tablas de resultados y análisis.
- Ficheros de código fuente completo ejecutable utilizado.
- Scripts, si los ha utilizado.
- Se realizará una defensa de la práctica explicando cada una de las decisiones tomadas en el código. La evaluación tendrá en cuenta:
  **Nivel Básico (5):**
  - Corrección del algoritmo
  - Utilización de la terminología apropiada (la utilizada en el material de clase)
  **Nivel intermedio(5-7):**
  - Análisis e interpretación sobre la capacidad de exploración/explotación de cada algoritmo relacionado con los resultados obtenidos.
  - Gráficas y ejemplos concretos comparados.
  - Datasets asociados
  **Nivel Avanzado(7-10):**
  - Mejoras en el rendimiento de la aplicación
  - Resultados obtenidos razonablemente cercanos al optimo
  - Análisis de la estadística de los resultados. Algoritmos más estables. Desviación típica.
  - Datos avanzados ejecutados

Permanezca atento a posibles nuevas versiones mejoradas de este documento.