
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 2023, 1.0)

## 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 un problema de *optimización de uso de una red de bicicletas.* El comportamiento de los algoritmos de OCH implementados deberá compararse con un *Algoritmo Greedy*.

### Diseño, justificación y desarrollo de un Algoritmo de optimización de una planta solar.

Este algoritmo debe maximizar los ingresos diarios para una instalación de planta solar que tiene una capacidad total de 1000 m2 con un rendimiento del 20% de la energía recibida. La planta posee una batería que puede almacenar energía antes de venderse de 300 kWh de capacidad.

Recibirás el día anterior una previsión meteorológica con la radiación por hora en un vector R (24 enteros con w por m2) y el precio previsto de la energía cada hora en P (24 valores reales en euros por cada kWh)

El algoritmo debe decidir que decidir por cada hora cuánta energía se venderá / comprará, la energía que se produce cada hora (o compra) y no se vende se almacena en la batería (que tiene un límite de 300kwh y comienza vacía todos los días). Toda la energía que no almacena ni vende se desperdicia.

Precio por día por kWh: a continuación, se muestran los precios reales para el día de hoy en la península, genera una lista de valores a mano deduciéndolos de la grafica en una variable o en un fichero con los valores en céntimos de euro por horas (24 valores) . Asume estos valores como precios de venta de los kWh producidos en planta.

Problema 1(datos reales) pc y pv (precio por hora en cts por kW) y r (radiación por hora en W * m^2)

```
pc = [26, 26, 25, 24,23,24,25,27,30,29,34,32,31,31,25,24,25,26,34,36,39,40,38,29]

pv = [24, 23, 22, 23,22,22,20,20,20,19,19,20,19,20,22,23,22,23,26,28,34,35,34,24]

r = [0, 0, 0, 0, 0, 0, 0, 0, 100, 313, 500, 661, 786, 419, 865, 230, 239, 715, 634, 468, 285, 96,
0, 0]
```

Problema 2 (aleatorizado)

```
Pc= [7,7,50,25,11,26,48,45,10,14,42,14,42,22,40,34,21,31,29,34,11,37,8,50]

Pv= [1,3,21,1,10,7,44,35,4,1,23,12,30,7,30,4,9,10,6,9,8,27,7,10]

r = [274,345,605,810,252,56,964,98,77,816,68,261,841,897,75,489,833,96,117,956,970,255,74,926]
```

Una posible solución del experto podría ser:

00- Vender 30 kwh
01- Comprar 100 kwh
… ..
23- Vender 0 Kwh (también es posible no hacer nada)

El resultado esperado es la cantidad en euros conseguida en el día, calculada como el precio por kWh vendidos en esa Hora. Dado que lo que podamos vender dependerá de la carga de la batería en dicho momento no se puede saber de antemano el resultado, por lo que el resultado será calculado mediante una simulación usando los precios y radiación. La simulación interpretará las instrucciones en la solución actual a evaluar calculando cada hora la carga de la batería y la cantidad en euros, que puede ser negativa.

**Base**

El siguiente código es total o parcialmente reutilizable. Codifica con la suficiente generalización para su uso en los algoritmos de esta y siguientes prácticas con la denominación, parámetros y retorno que estimes oportuno. Ten en cuenta el uso de semillas preestablecidas para el generador de números aleatorios.
Las soluciones deben ser siempre válidas (dar un valor en euros en la simulación sin errores)

a) Función de evaluación.
b) Generación de la solución inicial.
c) Operador de movimiento . El operador de movimiento debe elegir dos posiciones aleatorias del vector de capacidades y mover un número de espacios de aparcamiento de una a otra.

**Algoritmo de comparación: *Greedy***

Para efectuar la comparativa de resultados entre los distintos algoritmos de búsqueda, se debe implementar como algoritmo básico, un *Greedy*, siguiendo la heurística de guardar desde el principio hasta que se llene y luego vender en el pico de precio del dia todo. A partir de ese momento vender todo.

**Búsqueda Aleatoria**

El Algoritmo de Búsqueda Aleatoria (BA) consistirá en generar aleatoriamente una solución en cada iteración debiéndose ejecutar 100 iteraciones con cada semilla devolviendo la mejor de las iteraciones.

**Búsquedas Locales *el primer mejor*** y el **mejor vecino**

Se implementará siguiendo el esquema de ***el primer mejor*** y el **mejor vecino**, según el Tema 1 de teoría.

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 se llame al cálculo de la función de coste. Este valor para la Búsqueda Local será de 3000 llamadas a la función de coste.

Para el número de vecinos del primer mejor se pasará a la siguiente solución cuando se encuentre un vecino mejor mediante una operación de movimiento o se haya alcanzado el número máximo de intentos sin mejora.

Haz un estudio de granularidad del operador de movimiento (al menos tres valores) respecto a la calidad de la solución encontrada. Para ello deberá mostrar varios gráficos de mejor solución (total euros generados) en cada iteración para cada velocidad.

**Búsqueda Local VND**

Se implementará 5 niveles de velocidad variable según lo indicado en la teoría.

## 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*, *Tk*

$$= T0/(1 + k)$$

- *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

Justificar con graficas los valores elegidos de vecinos y condición de parada. Para ello elegir un numero máximo de evaluaciones objetivo y dependiendo del LT determinar la condición de parada.

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*(*Si*) es el costo de la solución inicial y Φ[0,1] es la probabilidad de aceptar una solución un μ por 1 peor que la inicial. En las ejecuciones se considerar Φ,μ, entre 0.1 y 0.3 mediante una pequeña experimentación que determine el que el número de soluciones iniciales no aceptadas sea de un 20% para T0, tomando C(s) como el coste de la solución Greedy.

## 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.
- Codificación Matriz Frecuencia largo plazo: codificación de valores en cada hora. Cuantas veces toma un valor una hora.

- Lista Tabú: se admitirá cualquier propuesta con suficiente capacidad de restricción de los movimientos. Una opción válida para la codificación tabú puede ser tener en cuenta sólo que los movimientos realizados no sean los mismos.
- 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.
- La solución greedy debe generar soluciones con mayor probabilidad para los valores que menos veces se han producido en cada hora.
- Para cada hora se deberá calcular las inversas de los valores acumulados y luego normalizar para tener valores entre 0-1 correspondiente a su probabilidad. Se tirará un dado y se elige el primer valor que supera dicho valor añadiéndose a la solución greedy

Ejemplo:

Si tenemos una hora Ej con tres posibles valores valores(v1, v2 ,v3) con los siguientes valores de frecuencia acumulada :

matrizFrecuencias(…….
2, 3, 5
……..).

1. Calcular inversas (…1/2, 1/3 , 1/5…)
2. Calcular suma total 1,03333…
3. Normalizar dividiendo por la suma de las inversas : 0.48, 0.32, 0.19

*Atención: usar una resolución alta de decimales o tener en cuenta la posibilidad de que el algoritmo pueda salir sin encontrar el último valor.

Una vez hecho esto todas las posiciones de la matriz *matrizProbabilidades* tienen su probabilidad directa y se pasa a construir un greedy probabilístico a las proporciones de esta matriz.

```
soluciónGreedy = {}
Para cada hora en cromosoma
    Número = random(0,1)
    suma=0
    Para i = valor en valores
        Suma+= matrizProbabilidades (estación, i)
        If número < suma
            soluciónGreedy.add(valor)
          break()

return soluciónGreedy
```

Estimar el número máximo de iteraciones en total. Se realizarán 4 reinicializaciones, es decir, una cada Numtotal-iteraciones/4. El tamaño inicial de cada lista tabú será $n$=4, estos valores cambiarán después de las reinicializaciones según se ha comentado.

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, empezando la lista desde vacío en cada reinicialización.

**Metodología de Comparación**

Cada algoritmo debe ejecutarse 5 veces, cada vez con una semilla distinta (por tanto, se deben anotar las 5 semillas que se utilizarán sistemáticamente), para el generador aleatorio por cada problema para los parámetros finales definidos.

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 construirá una tabla global de resultados con la estructura mostrada en la tabla. El dato entre paréntesis debajo del nombre de la instancia en la tabla 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 dec evaluaciones realizadas por el algoritmo en las cinco ejecuciones (salvo en el caso del algoritmo *Greedy* que sólo se ejecuta una vez).

Problema X

| **Algoritmo** | **Ev.Medias** | **Ev. Mejor** | **EV Desv.** | **Mejor €** | **Media €** | **Desv.€** |
|---|---|---|---|---|---|---|
| Greedy | | | | | | |
| Aleatoria | | | | | | |
| BL Mejor | | | | | | |
| BL Primer | | | | | | |
| VND | | | | | | |
| Enf. Simulado | | | | | | |
| Tabú | | | | | | |

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).

Las prácticas se realizan individualmente, es natural compartir la idea general y se recomienda comparar los resultados en el foro de la asignatura, pero la codificación y el análisis debe ser un esfuerzo individual.

**Fecha y Método de Entrega;**

Habrá dos entregas.

**6 de marzo: Entrega intermedia**: Descripción de representación, movimiento y evaluación. Ejemplo con la BL Mejor y Enfriamiento Simulado

Se defenderá en las siguientes clases, es obligatoria, debe ejecutar y devolver un resultado, pero puede haber errores leves. El objetivo es que el nucleo de la práctica sea correcto para su entrega final. Si no se revisa antes de la entrega final de la práctica se dará por suspensa. En casos justificados que impidan asistir a uno de los dos grupos (lunes o miércoles) durante 10 minutos para esta defensa se organizará una tutoría zoom que debe solicitarse por correo electrónico.

**20 de marzo**: Debe entregar 1 fichero comprimido con el nombre Practica1_NombreApellidos.zip (o rar) que contenga:

- Documento 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 (5-6):**
  - Corrección del algoritmo
  - Utilización de la terminología apropiada (la utilizada en el material de clase)
  - Experimentación realizada
  - Justificación de los parametros
  **Nivel intermedio (7-8):**
  - Análisis e interpretación sobre la capacidad de exploración/explotación de cada algoritmo relacionado con los resultados obtenidos.
  - Gráficas y ejemplos concretos comparados.
  **Nivel Alto (9-10)**
  - 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.
  - Función de simulación que muestre la evolución de la batería y el saldo del resultado en una gráfica.
  - Experimentación extendidas de todos los parámetros de los algoritmos y uso de la teoría para justificar el comportamiento.
  - Uso de Biliografia científica, ejemplos similares de uso de estas técnicas.

Permanezca atento a posibles nuevas versiones mejoradas de este documento.