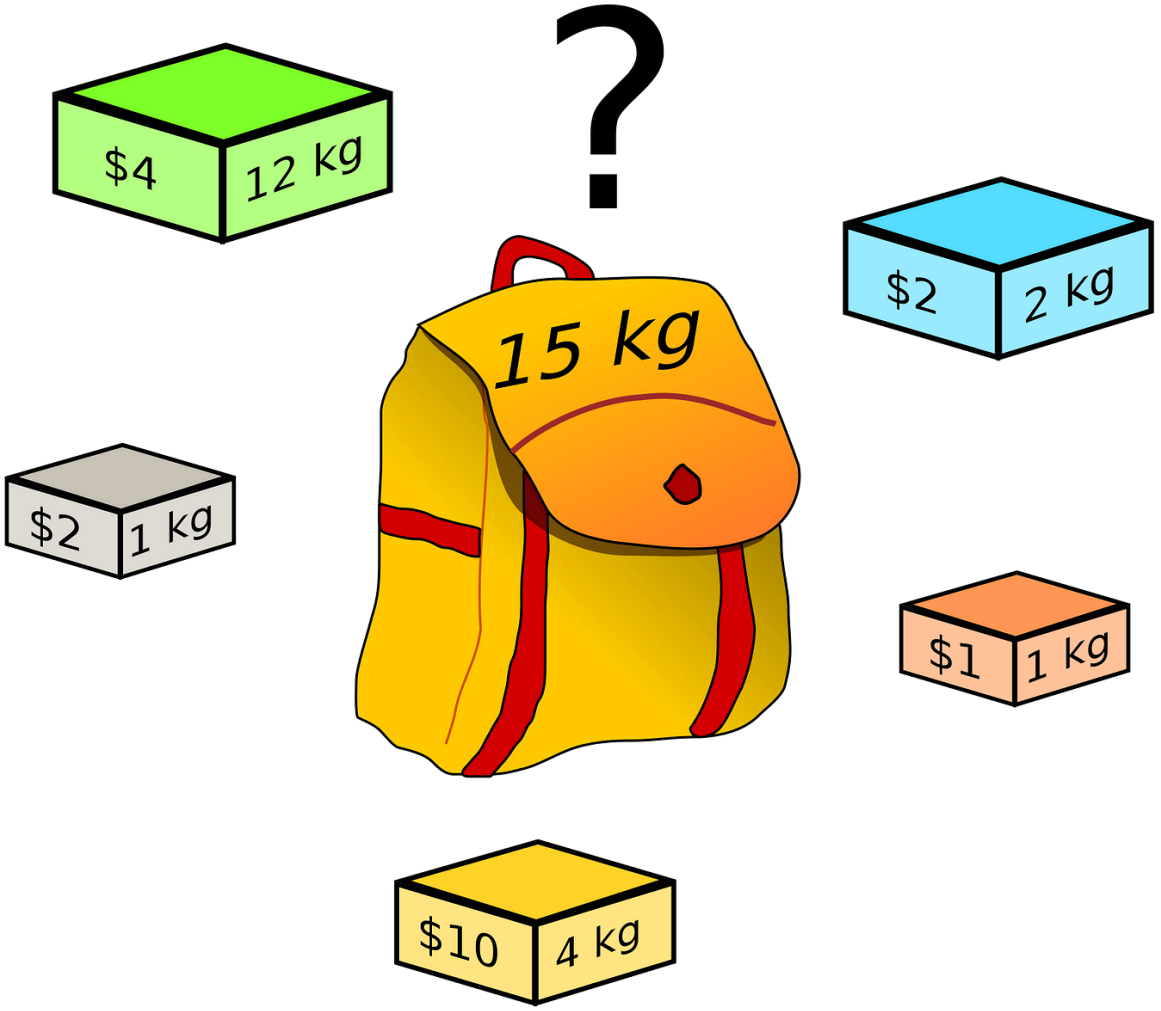
Práctica 3
Resolución del Problema de la Mochila Cuadrática mediante algoritmos BMB, ES e ILS



Santiago Herron Mulet

Metaheurística Grupo de Prácticas 3

ÍNDICE

1. Descripción del problema.

a. Planteamiento:

El problema de la mochila cuadrática (QKP) es un problema de optimización combinatoria NP-completo que busca maximizar el beneficio obtenido al seleccionar un conjunto de objetos para colocar en una mochila con capacidad limitada. Cada objeto tiene un peso, un valor y un beneficio adicional que se obtiene si se seleccionan dos objetos específicos. El objetivo es encontrar la combinación de objetos que maximice el beneficio total sin exceder la capacidad de la mochila.

b. Formulación matemática:

i. Función a maximizar f(x):

$$\text{maximize} \left\{ \sum_{i=1}^{n} p_i x_i + \sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} P_{ij} x_i x_j : x \in X, x \text{ binary} \right\}$$

ii. Sujeta a:

$$\text{subject to } X \equiv \left\{ x \in \{0,1\}^n : \sum_{i=1}^{n} w_i x_i \leq W; x_i \in \{0,1\} \text{ for } i = 1, \ldots, n \right\}$$

donde:
p_i: Valor del objeto i.
p_ij: Beneficio adicional obtenido al seleccionar los objetos i y j juntos.
w_i: Peso del objeto i.
W: Capacidad de la mochila.
x_i: Variable binaria que indica si el objeto i se selecciona (x_i = 1) o no (x_i = 0).

c. Propiedades y complejidad:

El QKP es un problema NP-completo, lo que significa que no existe un algoritmo conocido que pueda resolverlo de manera eficiente para todas las instancias. A medida que aumenta el tamaño del problema, la complejidad computacional para encontrar la solución óptima aumenta exponencialmente.

d. Algoritmos de resolución:

Existen diversos algoritmos para resolver el QKP, los cuales se pueden dividir en dos categorías principales:

i. Algoritmos exactos:

Estos algoritmos garantizan encontrar la solución óptima, pero pueden ser muy lentos para problemas de gran tamaño.

ii. Algoritmos heurísticos:

Estos algoritmos no garantizan encontrar la solución óptima, pero son mucho más rápidos que los algoritmos exactos.

2. Descripción de los ámbitos generales de los algoritmos de trayectorias.

a. A Nivel Teórico

i. Naturaleza heurística:

Los algoritmos de trayectorias, como BMB, ILS, ES e ILS-ES, son algoritmos heurísticos diseñados para encontrar soluciones aproximadas a problemas complejos como el problema de la mochila cuadrática (QKP). Estos algoritmos no garantizan encontrar la solución óptima, pero son efectivos para explorar y optimizar soluciones en espacios de búsqueda grandes y complejos.

ii. Exploración y explotación:

Los algoritmos de trayectorias equilibran la exploración y la explotación del espacio de búsqueda.

Exploración: Implica generar y evaluar nuevas soluciones en diferentes áreas del espacio de búsqueda para descubrir regiones prometedoras. La exploración ayuda a evitar que el algoritmo se quede atrapado en óptimos locales.
Explotación: Se centra en mejorar las soluciones existentes mediante movimientos hacia soluciones vecinas de mejor calidad. La búsqueda local es un claro ejemplo de explotación, donde el algoritmo busca mejorar iterativamente una solución actual.
Naturaleza estocástica y exploratoria:
La aleatoriedad es una característica fundamental en los algoritmos de trayectorias:

BMB: Utiliza múltiples puntos de partida aleatorios para realizar búsquedas locales desde diferentes áreas del espacio de búsqueda.
ILS: Combina búsqueda local con perturbaciones aleatorias para escapar de óptimos locales y explorar nuevas regiones del espacio de búsqueda.

ES: Introduce aleatoriedad al aceptar ocasionalmente soluciones peores, especialmente al inicio, para evitar el estancamiento en óptimos locales y fomentar la exploración.
ILS-ES: Combina las estrategias de ILS y ES, utilizando enfriamiento simulado en lugar de búsqueda local para la explotación intensiva con una componente estocástica significativa.

iii. Parámetros y ajustes:

El rendimiento de los algoritmos de trayectorias depende de varios parámetros clave:

Parámetros comunes: Número máximo de iteraciones, número máximo de evaluaciones, tamaño de vecindad, etc.
Parámetros específicos:
BMB: Número de reinicios, criterios de parada para cada búsqueda local.
ILS: Intensidad de perturbaciones, criterios de aceptación.
ES: Temperatura inicial, tasa de enfriamiento, probabilidad de aceptación de soluciones peores.
ILS-ES: Combina los parámetros de ILS y ES.
La correcta configuración de estos parámetros es crucial para el equilibrio entre exploración y explotación, y para mejorar el rendimiento del algoritmo en la solución del QKP. Ajustar los parámetros adecuadamente permite a los algoritmos de trayectorias adaptarse mejor a las características específicas del problema y encontrar soluciones de alta calidad de manera más eficiente.

b. Esquema de representación de soluciones:

En el problema de la mochila cuadrática, cada individuo de la población se representará como un objeto de la clase Mochila. Esta clase contendrá un atributo asignacion, que será un vector booleano. Cada elemento del vector representará si el objeto correspondiente está presente (true) o ausente (false) en la mochila. Por ejemplo, si el índice i es true, significa que el objeto i ha sido seleccionado para ser incluido en la mochila; de lo contrario, no ha sido seleccionado.

Además, cada mochila también tendrá los atributos peso y beneficio. El peso representará la suma de los pesos individuales de los objetos incluidos en la mochila, mientras que el beneficio será el beneficio total asociado a esa mochila, el cual se detallará en la siguiente sección.

c. Descripción de la función objetivo (fitness):

En nuestro problema de la mochila, el fitness vendrá representado por el beneficio de la mochila, es decir, la suma de los beneficios individuales de cada objeto más los beneficios por interdependencia de cada unos de los objetos incluidos entre sí. En mi caso, he incluído dicho cálculo del fitness en un método de la clase mochila. Como se observa en el siguiente pseudocódigo, he añadido también a esta función

el cálculo del peso de la mochila para no añadir complejidad añadida a su cálculo:

```
Función Fitness():
   beneficio = 0
   peso = 0
   Para cada índice i en el rango de 0 a tamaño de asignacion - 1:
      Si asignacion[i] es verdadero:
         beneficio += b_individual[i]
         peso += pesos[i]
         Para cada índice j en el rango de 0 a tamaño de asignacion - 1:
            beneficio += b_interdependencias[i][j] * asignacion[j]
   Devolver beneficio
```

d. Descripción de operadores comunes:

i. Operador de generador de vecinos para BL e ES.

Se usará el mismo generador de vecinos que en las anteriores prácticas en el que se consideran soluciones vecinas aquellas soluciones factibles en las que sólo varíe desde la solución inicial una permutación de un objeto incluído por otro que no lo estaba:

```
Función VecinosPermutacion(actual, W):
   vecinos = nuevo vector
   v = obtener asignación de actual

   peso_actual = obtener peso de actual
   Para cada índice i en el rango de 0 a tamaño de v - 1:
      Si v[i] es verdadero:
         Para cada índice j en el rango de 0 a tamaño de v - 1:
            Si no v[j] y i distinto de j:
               Si peso_actual - peso del objeto i + peso del objeto j <= W:
                  Agregar  (i, j) a vecinos
   Devolver vecinos
```

ii. Búsqueda Local.

En mi caso, la búsqueda local utilizada en las anteriores prácticas era suficientemente generalista para usarla directamente cambiando únicamente algunos parámetros.

```
Funcion busquedaLocal(actual, n, W, max_instancias, n_instancias, maximo_local,
Vecinos, max_vecinos_evaluados)
   mejor_asignacion = asignacion(actual)
   mejor_fitness = fitness(actual)
   capacidad = W
   maximo_local = Falso
   vecinos_evaluados = 0
```

```
   Si max_vecinos_evaluados == -1 Entonces
      max_vecinos_evaluados = n * (n - 1) / 2
   Fin Si

   Mientras n_instancias < max_instancias Y no maximo_local Y vecinos_evaluados
< max_vecinos_evaluados Hacer
      vecinos = Vecinos(actual, capacidad)
      aleatorizar(vecinos)

      mejora_encontrada = Falso
      Mientras n_instancias < max_instancias Y no vacio(vecinos) Y no
mejora_encontrada Y vecinos_evaluados < max_vecinos_evaluados Hacer
         vecinos_evaluados += 1
         (i1, i0) = ultimoElemento(vecinos)
         eliminarUltimo(vecinos)
         asignacion_nueva = copia(asignacion(actual))
         intercambiar(asignacion_nueva, i0, i1)
         nueva = copia(actual)
         actualizarFitness(nueva, asignacion_nueva, i0, i1)
         n_instancias += 1

         Si fitness(nueva) > fitness(actual) Entonces
            actual = nueva
            mejora_encontrada = Verdadero
         Fin Si
      Fin Mientras

      Si no mejora_encontrada Entonces
         maximo_local = Verdadero
      Fin Si
   Fin Mientras

   retornar asignacion(actual)
Fin Funcion
```

iii. Creación de solución aleatoria.

La solución inicial se generará de manera totalmente aleatoria. Se iniciará con una solución sin ningún objeto incluido y se eligirán objetos de manera aleatoria hasta que no quepa ninguno más.

```
Función solucionInicial(n, capacidad):
   s = nuevo vector de tamaño n, inicializado a falso
   o = una copia del vector de objetos
```

```
    Mientras o no esté vacío:
        Mezclar aleatoriamente los objetos en o
        Si capacidad >= peso del último objeto en o y el último objeto no está en
s:
            Reducir capacidad en el peso del último objeto en o
            Establecer el último elemento en s a verdadero
        Eliminar el último objeto en o

    Devolver s
```

3. BMB:

a. Estructura General:

i. Generación de solución aleatoria.
ii. Aplicación de búsqueda local sobre la solución aleatoria.
iii. Comprobar si la solución actual es mejor que la mejor solución encontrada y en tal caso actualizar la mejor solución encontrada
iv. Repetir un número prefijado de veces o hasta que se cumpla la condición de parada.

b. Algoritmo

```
Funcion BMB(max_iteraciones, max_evaluaciones_BL, n, W)
    soluciones = max_iteraciones soluciones aleatorias
    Para i desde 0 hasta max_iteraciones - 1 Hacer
        soluciones[i] = busquedaLocal(soluciones[i])
        Si fitness(soluciones[i]) > fitness(mejor_solucion) Entonces
            mejor_solucion = copia(actual_solucciones[i])
        Fin Si
    Fin Para
    retornar asignacion(mejor_solucion)
Fin Funcion
```

4. ES:

a. Estructura General:

i. Definir los parámetros de control (tasa de enfriamiento, temperatura final, maximos-vecinos, maximos-exitos…).
ii. Bucle principal (hasta que la temperatura sea menor que la temperatura final):
    1. Para cada iteración a una temperatura dada:
        a. Generar una solución vecina de la solución actual.
        b. Calcular el valor de la función objetivo para la solución vecina.
        c. Calcular la diferencia en la función objetivo entre la solución vecina y la solución actual.
        d. Si la solución vecina es mejor que la solución actual, aceptar la solución vecina.
        e. Si la solución vecina es peor que la solución actual, aceptar la solución vecina con una probabilidad que depende de la diferencia en la función objetivo y la temperatura actual.
    2. Actualizar temperatura si se cumple la condición de enfriamiento.
iii. Devolver la mejor solución.

b. Definición de parámetros de control:

i. Temperatura inicial (T0):
La temperatura inicial se calculará en función de la siguiente fórmula:

$$T_0 = \frac{\mu \cdot C(S_0)}{-\ln(\varphi)}$$

donde T0 es la temperatura inicial, C(S0) es el coste de la solución inicial (fitness) y [0,1] es la probabilidad de aceptar una solución un por 1 peor que la inicial. En las ejecuciones se considerará =0,3 y =0,1.
ii. Temperatura final (Tf):
Se inicializa en 10^-3 pero para asegurarnos que es menor que T0, mientras sea mayor que T0 se dividirá entre 10.
iii. Máximos vecinos = 5*n (número de objetos).
iv. Máximos éxitos = 0.1*max_vecinos.
v. Máx evaluaciones = 90000.
vi. Beta:

$$\beta = \frac{T_0 - T_f}{M \cdot T_0 \cdot T_f}$$

c. Esquema de enfriamiento.

Para el enfriamiento se usará el enfriamiento Cauchy modificado:

$$T_{k+1} = \frac{T_k}{1 + \beta \cdot T_k}$$

d. Algoritmo:

```
Funcion ES(actual, n, W, max_instancias, n_instancias, phi, mu, Vecinos)
    max_vecinos = 5 * n
    max_exitos = 0.1 * max_vecinos
    M = max_instancias / max_vecinos
    T0 = (mu * fitness(actual)) / (-log(phi))
    Tf = 0.001

    Mientras Tf >= T0 Hacer
        Tf = Tf / 10
    Fin Mientras

    beta = (T0 - Tf) / (M * T0 * Tf)
    T = T0
    exitos = 0
    num_vecinos = 0
    vecinos = Vecinos(actual, W)

    Mientras T > Tf Y n_instancias < max_instancias Y exitos < max_exitos Y
num_vecinos < max_vecinos Hacer
        exitos = 0
        num_vecinos = 0
        vecinos = Vecinos(actual, W)
        aleatorizar(vecinos)
```

```
      Mientras no vacio(vecinos) Y n_instancias < max_instancias Y
num_vecinos < max_vecinos Y exitos < max_exitos Hacer
         num_vecinos += 1
         (i1, i0) = obtenerUltimoElemento(vecinos)
         eliminarUltimoElemento(vecinos)
         actualizarFitness(nueva_solucion, nueva_asignacion, i0, i1)
         n_instancias += 1

         delta = fitness(actual) - fitness(nueva_solucion)
         Si delta < 0 O probabilidadAleatoria() <= exp(-delta / T) Entonces
            actual = nueva_solucion
            exitos += 1
            Si fitness(nueva_solucion) > fitness(mejor_solucion) Entonces
               mejor_solucion = copia(nueva_solucion)
            Fin Si
            romper
         Fin Si
      Fin Mientras

      Si exitos == 0 Entonces
         romper
      Fin Si

      Si no vacio(vecinos) Entonces
         T = enfriamientoCauchy(T, beta)
      Fin Si
   Fin Mientras

   retornar asignacion(mejor_solucion)
Fin Funcion
```

5. ILS.

a. Estructura general:

i. Inicializar la mejor solución a una aleatoria.
ii. Aplicar búsqueda local a la mejor solución con un máximo de evaluaciones prefijado.
iii. Iniciar un bucle con un número de iteraciones prefijado.
    1. Crear una nueva solución a partir de la mejor solución con una mutación significativa.

2. Aplicar búsqueda local a esta nueva solución.
3. Si la nueva solución tiene mejor fitness que la mejor solución, actualizar mejor solución a la nueva.

iv. Devolver mejor solución.

b. Operador de mutación para ILS:

Selección de Segmento:
Selecciona aleatoriamente dos índices dentro de la solución y define un segmento de longitud t entre estos índices.

Reorganización del Segmento:
Extrae el segmento seleccionado, lo mezcla aleatoriamente, y lo vuelve a insertar en la solución en su posición original.

Ajuste de Peso:
Si el peso total de la nueva solución supera la capacidad W, ajusta la solución eliminando elementos aleatorios dentro del segmento hasta que el peso sea válido.

```
Funcion mutacionILS(mejor, n, W, t)
    actual = copia(mejor)
    rand_indice1 = aleatorio(0, n-1)
    rand_indice2 = (rand_indice1 + t) % n

    asignacion = obtenerAsignacion(actual)
    segmento = extraerSegmento(asignacion, rand_indice1, rand_indice2, t)
    mezclar(segmento)
    insertarSegmento(asignacion, segmento, rand_indice1, rand_indice2)

    Mientras peso(actual) > W Hacer
        i = seleccionarIndiceAleatorio(rand_indice1, rand_indice2, n)
        Si asignacion[i] == Verdadero Entonces
            asignacion[i] = Falso
            actualizarAsignacion(actual, i)
        Fin Si
    Fin Mientras

    calcularFitness(actual)
    retornar actual
Fin Funcion
```

c. Algoritmo:

```
Funcion ILS(max_iteraciones, max_evaluaciones_BL, n, W)
   mejor = generarSolucionAleatoria(n, W)
   mejor = busquedaLocal(mejor, n, W, max_evaluaciones_BL)

   Para iteracion desde 1 hasta max_iteraciones Hacer
      actual = mutacion(mejor, n, W)
      actual = busquedaLocal(actual, n, W, max_evaluaciones_BL)

      Si fitness(actual) > fitness(mejor) Entonces
         mejor = actual
      Fin Si
   Fin Para

   retornar asignacion(mejor)
Fin Funcion
```

Para implementar el algoritmo ILS-ES, es decir, el ILS con explotación de soluciones mediante Enfriamiento SImulado, lo único que hizo falta cambiar fue las dos líneas donde aparece busquedaLocal por ES.

6. Procedimiento para desarrollar la práctica.

a. Organización del proyecto.

Para comenzar el desarrollo de la práctica, lo primero fue tomar las decisiones a nivel de estructuras. Con ese fin, se creó la clase Mochila ya explicada.
A continuación se decidió la estructura de ficheros. Usando un fichero para el main y otro para las funciones de los diferentes algoritmos.
Por último, con el fin de facilitar el uso del software, se desarrolló un Makefile para el compilado automático de las fuentes y diversos bash scripts para generación de múltiples resultados para posteriormente analizarlos.

b. Manual de usuario:

i. Compilación:

Con la orden make.
Se genera un ejecutable llamado QKP.

ii. Ejecución:

Los argumentos del programa son.

1. ./$ejecutable
2. $Algoritmo. Opciones:
    - BL para búsqueda local.
    - BMB para Búsqueda Multiarranque Básica.
    - ES para Enfriamiento Simulado
    - ILS para Iterative Local Search.
    - ILS-ES para Iterative Local Search con Enfriamiento Simulado.
3. $fichero de entrada de datos
4. (opcional) $semilla (Para greedy no es necesaria, para el resto, si no se indica, no se fijará semilla).

iii. Salida:

Se mostrará por pantalla lo siguiente:

1. W: capacidad de la mochila.
2. Peso de la mochila final.
3. Beneficio final.
4. Asignación (unos y ceros)
5. Última línea: $beneficio $milisegundos

iv. Scripts:

Importante: si se desea utilizar alguno de los scripts se deben comentar todas las salidas de texto por pantalla menos la última línea que saca el beneficio y el tiempo y volver a compilar con make.

Cada uno de esos scripts ejecuta el programa con el algoritmo correspondiente para cada uno de los archivos datos de entrada. Para aquellos con tamaño y densidad comunes, hace la media del beneficio y del tiempo y guarda los resultados en ficheros en formato csv.
Se desarrollaron:

1. ScriptBL.sh
2. ScriptBMB.sh
3. ScriptES.sh
4. ScriptILS.sh
5. ScriptILS-ES.sh

7. Experimentación:

a. Recogida de datos.

Para la recogida de datos experimentales se usaron los scripts mencionados usando la semilla **33**. Posteriormente, se organizaron en tablas.

b. Resultados:

i. BMB:

Resultados BMB			
Tamaño	Densidad	Fitness Medio	Tiempo Medio
100	25	5,71E+04	4,90E+01
100	50	1,11E+05	5,80E+01
100	75	1,59E+05	5,10E+01
100	100	2,24E+05	3,50E+01
200	25	2,08E+05	1,91E+02
200	50	4,54E+05	2,77E+02
200	75	3,75E+05	2,14E+02
200	100	7,33E+05	1,69E+02
300	25	3,45E+05	6,95E+02
300	50	9,28E+05	7,59E+02

ii. ES:

Resultados ES			
Tamaño	Densidad	Fitness Medio	Tiempo Medio
100	25	4,98E+04	5,00E+00
100	50	9,83E+04	1,00E+00
100	75	1,42E+05	1,00E+00
100	100	2,04E+05	0,00E+00
200	25	1,94E+05	8,00E+00
200	50	4,10E+05	1,00E+01
200	75	3,38E+05	8,00E+00
200	100	6,56E+05	1,00E+01
300	25	3,02E+05	1,30E+01

300	50	8,31E+05	1,80E+01

iii. ILS:

Resultados ILS			
Tamaño	Densidad	Fitness Medio	Tiempo Medio
100	25	5,17E+04	3,20E+01
100	50	9,98E+04	4,50E+01
100	75	1,43E+05	3,20E+01
100	100	2,10E+05	2,50E+01
200	25	1,99E+05	6,80E+01
200	50	4,09E+05	1,24E+02
200	75	3,28E+05	9,70E+01
200	100	7,00E+05	7,10E+01
300	25	3,18E+05	1,89E+02
300	50	8,73E+05	2,17E+02

iv. ILS-ES

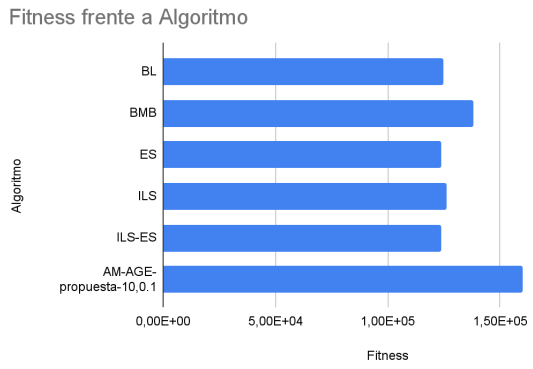
Resultados ILS-ES			
Tamaño	Densidad	Fitness Medio	Tiempo Medio
100	25	4,99E+04	6,00E+00
100	50	9,58E+04	1,40E+01
100	75	1,41E+05	7,00E+00
100	100	2,10E+05	5,00E+00
200	25	1,85E+05	8,00E+00
200	50	3,86E+05	1,40E+01
200	75	3,09E+05	1,10E+01
200	100	6,86E+05	9,00E+00
300	25	2,84E+05	1,40E+01
300	50	8,25E+05	2,10E+01

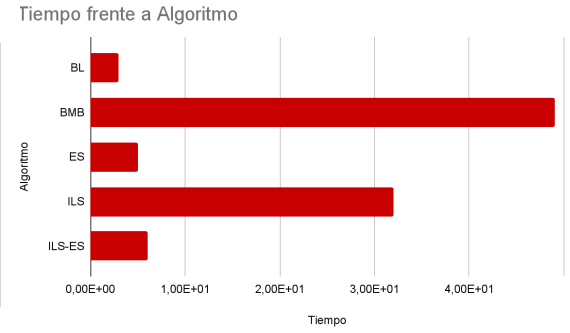
c. Resumen de Resultados:

(no se han incluido los gráficos de tiempo del meméticos pues estos son significativamente más lentos y no aportarían a la comparación de los otros algoritmos.)

i. Tamaño 100:

Resumen algoritmos Tamaño 100		
Algoritmo	Fitness	Tiempo
BL	1,25E+05	3,00E+00
BMB	1,38E+05	4,90E+01
ES	1,24E+05	5,00E+00
ILS	1,26E+05	3,20E+01
ILS-ES	1,24E+05	6,00E+00
AM-AGE-propuesta-10,0.1mej	1,60E+05	6,60E+02

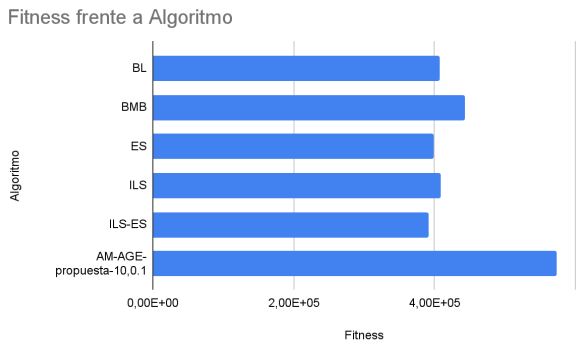


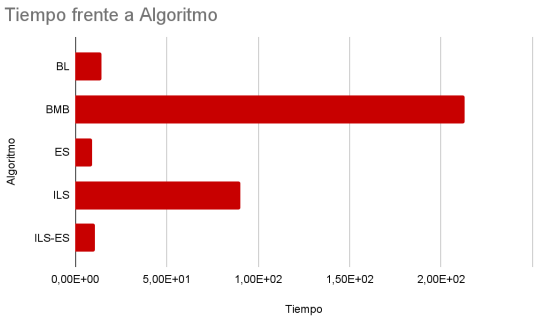


ii. Tamaño 200:

Resumen algoritmos Tamaño 200		
Algoritmo	Fitness	Tiempo
BL	4,07E+05	1,45E+01
BMB	4,43E+05	2,13E+02
ES	3,99E+05	9,00E+00
ILS	4,09E+05	9,00E+01

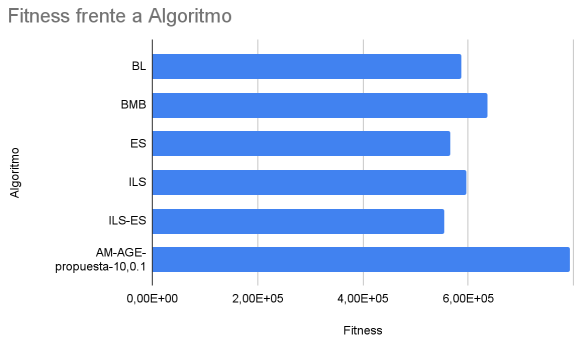
ILS-ES	3,91E+05	1,05E+01
AM-AGE-propuesta-10,0.1mej	5,74E+05	1,42E+03

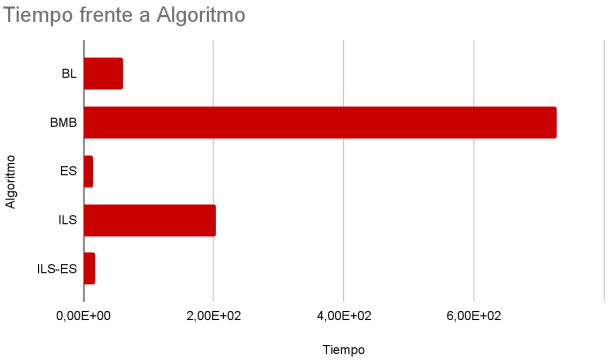




iii. Tamaño 300:

Resumen algoritmos Tamaño 300		
Algoritmo	Fitness	Tiempo
BL	5,88E+05	6,00E+01
BMB	6,36E+05	7,27E+02
ES	5,66E+05	1,55E+01
ILS	5,96E+05	2,03E+02
ILS-ES	5,54E+05	1,75E+01
AM-AGE-propuesta-10,0.1mej	7,93E+05	2,06E+03



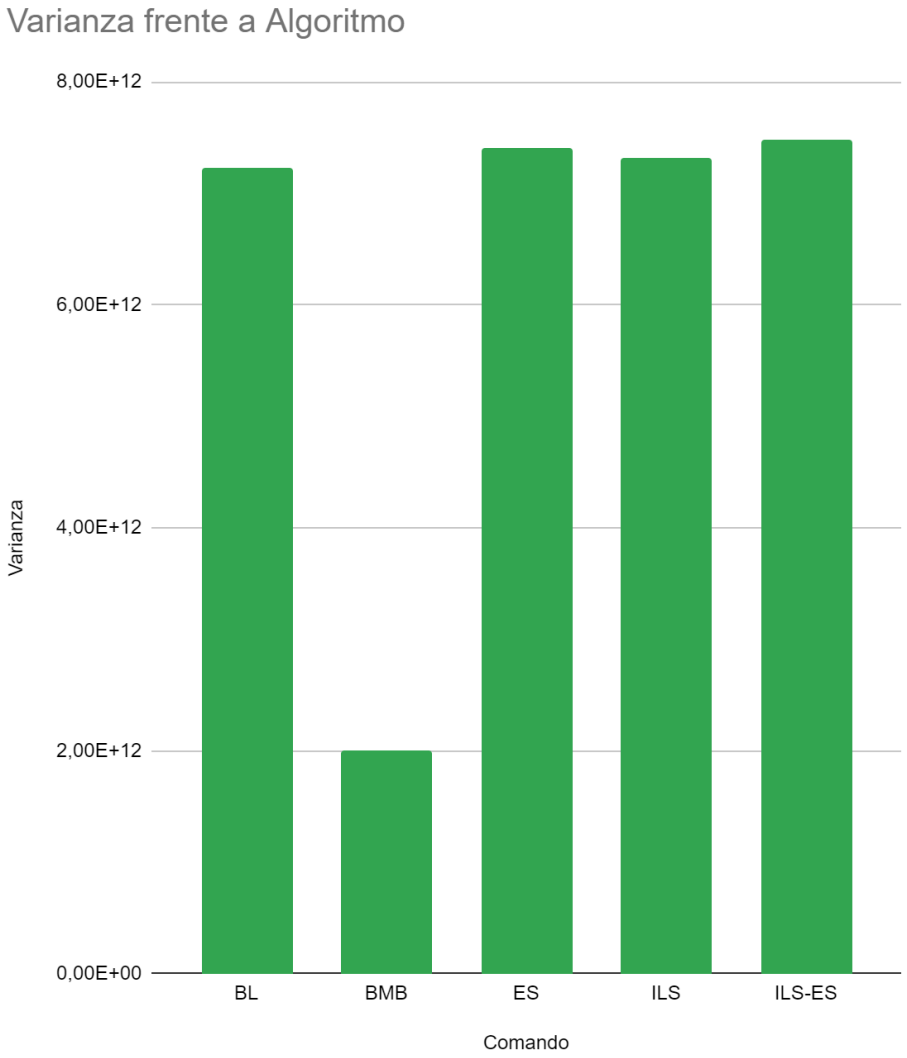


d. Estudio de la varianza de las soluciones para una misma ejecución.

A sabiendas del importante factor estocástico de los algoritmos se decidió hacer una comparativa de varianza entre ellos para el mismo fichero de entrada con el objetivo de ver hasta que punto pueden ser diferentes los resultados dependiendo del azar para cada algoritmo. Para ello se desarrolló un script que se puede encontrar en el software como EstudioVarianza.sh. Este script ejecuta 10000 veces cada algoritmo con el mismo fichero de entrada y sin fijación de semilla por lo que esta será aleatoria, almacena los resultados y calcula la varianza para cada algoritmo. Saca los resultados en un archivo csv resultados_varianza.csv.

Fichero de entrada:
bin/data/jeu_100_25_1.txt

Resultados:



Estos resultados son realmente interesantes pues se puede observar que:

- Claramente, el algoritmo BMB es mucho más consistente en cuanto al fitness que el resto de algoritmos.
- El ES varía algo más que el BL. Esto se puede observar tanto en los algoritmos por sí solos como en la comparación entre ILS y ILS-ES.
- Se observa que, para BL, ES ILS y la aleatoriedad es realmente crucial para obtener un buen resultado ya que sus valores de varianza son realmente altos.

8. Análisis de resultados:

a. BL vs ES.

i. Resultados de la Comparación

- Fitness: El fitness obtenido por ambos algoritmos es muy similar en general. Sin embargo, a medida que aumenta el tamaño del problema (número de objetos en la mochila), se observa una diferencia creciente en el fitness a favor de BL.
- Tiempo de Ejecución: Ambos algoritmos presentan tiempos de ejecución bajos. Sin embargo, ES es notablemente más rápido que BL, y esta diferencia aumenta con el tamaño del problema.
- Variabilidad de los Resultados: Ambos algoritmos muestran una alta variabilidad en los resultados, aunque ES presenta una variabilidad ligeramente mayor.

ii. Análisis de las Observaciones

Al comparar la Búsqueda Local (BL) con el Enfriamiento Simulado (ES) en el contexto del problema de la mochila cuadrática, emergen diferencias importantes que explican el desempeño de cada algoritmo. La Búsqueda Local se caracteriza por una intensa explotación del espacio de búsqueda, refinando sistemáticamente una solución hasta encontrar un óptimo local. Este enfoque permite que BL obtenga un mejor fitness, especialmente cuando aumenta el número de objetos en la mochila. La capacidad de BL para explorar exhaustivamente el entorno inmediato de una solución le confiere una ventaja en la calidad final de la solución obtenida. Sin embargo, BL no tiene mecanismos para escapar de óptimos locales, lo que puede limitar su capacidad para encontrar soluciones globalmente óptimas en problemas más complejos.

Por otro lado, el Enfriamiento Simulado (ES) introduce un componente estocástico crítico: la aceptación probabilística de soluciones peores. Este mecanismo permite a ES escapar de óptimos locales, explorando más ampliamente el espacio de búsqueda. Al permitir empeorar la solución temporalmente, ES puede superar barreras en el paisaje de la función objetivo que podrían atrapar a BL en óptimos locales. Esta característica es fundamental para explicar por qué ES, aunque más rápido debido a su enfoque menos exhaustivo, puede ser menos preciso en la explotación del espacio de búsqueda, resultando en un fitness ligeramente inferior en comparación con BL, especialmente en problemas de mayor tamaño.

La variabilidad de los resultados también es una diferencia notable entre los dos algoritmos. La naturaleza más determinista y sistemática de BL tiende a reducir la variabilidad de los resultados en diferentes ejecuciones, aunque no la elimina por completo debido a la dependencia de la solución inicial y la estructura del espacio de búsqueda. En contraste, ES presenta una mayor variabilidad debido a su componente estocástico. La aceptación probabilística de soluciones peores introduce variaciones significativas en los resultados obtenidos en diferentes ejecuciones, aumentando la diversidad de las soluciones encontradas.

iii. Conclusión

En conclusión, para el problema de la mochila cuadrática, la Búsqueda Local (BL) tiende a proporcionar soluciones de mayor calidad en términos de fitness, especialmente a medida que el tamaño del problema aumenta, aunque a costa de un mayor tiempo de ejecución.

Enfriamiento Simulado (ES) es más rápido y puede ser preferible en situaciones donde el tiempo de cálculo es crítico. Su capacidad para escapar de óptimos locales mediante la aceptación de soluciones peores le permite explorar el espacio de búsqueda de manera más amplia, aunque con menor precisión y mayor variabilidad en los resultados. La elección entre BL y ES dependerá de las prioridades específicas del problema: calidad de la solución versus tiempo de ejecución y la necesidad de evitar óptimos locales.

b. BMB vs ILS:

i. Resultados de la Comparación

- Fitness: BMB muestra un rendimiento superior en términos de fitness en comparación con ILS, sin importar el tamaño del problema.
- Tiempo de Ejecución: BMB presenta tiempos de ejecución considerablemente más altos que ILS.
- Variabilidad de los Resultados: BMB es consistentemente más estable en los resultados obtenidos, mientras que ILS muestra una mayor variabilidad.

ii. Análisis de las Observaciones

Al comparar el Búsqueda Multiarranque Básica (BMB) con la Búsqueda Local Iterativa (ILS) en el contexto del problema de la mochila cuadrática, se destacan varias diferencias clave que explican las observaciones experimentales. BMB y ILS son métodos metaheurísticos que combinan técnicas de exploración y explotación para encontrar soluciones aproximadas a problemas complejos, pero sus enfoques y características intrínsecas varían significativamente.

BMB se basa en la idea de generar múltiples soluciones iniciales aleatorias y aplicar Búsqueda Local (BL) a cada una de ellas. Esta estrategia permite a BMB explorar el espacio de búsqueda de manera más exhaustiva y diversificada, lo que resulta en un mejor fitness general. Al iniciar la búsqueda desde múltiples puntos, BMB tiene mayores probabilidades de encontrar óptimos globales o soluciones cercanas a ellos, a diferencia de ILS, que se enfoca en la exploración intensiva de una única solución inicial mejorada iterativamente. Este enfoque de ILS limita su capacidad para escapar de óptimos locales, especialmente en problemas de mayor tamaño y complejidad, donde la exploración exhaustiva de BMB le otorga una ventaja significativa.

Sin embargo, esta capacidad de BMB para explorar múltiples soluciones tiene un costo: el tiempo de ejecución. Evaluar varias soluciones iniciales y aplicarles Búsqueda Local repetidamente implica un mayor consumo de tiempo y recursos computacionales. Por el contrario, ILS, al mejorar iterativamente una sola solución inicial mediante pequeñas perturbaciones y subsecuentes aplicaciones de Búsqueda Local, es más eficiente en términos de tiempo. Esta eficiencia hace que ILS sea preferible en situaciones donde el tiempo de cálculo es una restricción crítica.

En términos de variabilidad, BMB muestra una consistencia notable en los resultados obtenidos. La diversidad de soluciones iniciales y la aplicación sistemática de Búsqueda Local contribuyen a una mayor estabilidad y menor dispersión en los valores de fitness alcanzados. ILS, al depender en gran medida de la calidad de la solución inicial y de la efectividad de las perturbaciones aplicadas, exhibe una mayor variabilidad en los resultados. La naturaleza estocástica de las perturbaciones y la exploración limitada pueden llevar a diferentes trayectorias de búsqueda y, por ende, a variaciones significativas en el fitness final.

iii. Conclusión

En resumen, para el problema de la mochila cuadrática, BMB sobresale en términos de calidad de las soluciones, proporcionando consistentemente mejores valores de fitness gracias a su enfoque de exploración múltiple y exhaustiva. Sin embargo, este rendimiento superior viene acompañado de un mayor tiempo de ejecución. ILS, aunque menos eficaz en términos de fitness, es significativamente más rápido y presenta una mayor variabilidad en los resultados debido a su enfoque de exploración intensiva de una única solución. La elección entre BMB e ILS dependerá de las prioridades específicas del problema, como la calidad de la solución versus la eficiencia en el tiempo de cálculo.

c. BL vs ILS y ES vs ILS-ES:

i. Resultados de la Comparativa: BL vs ILS / ES vs ILS-ES

- Fitness: ILS presenta una mejora marginal en fitness en comparación con BL, mientras que ILS-ES es similar o ligeramente inferior a ES.
- Tiempo de Ejecución: ILS es considerablemente más lento que BL, mientras que ILS-ES es ligeramente más lento que ES.
- Variabilidad de los Resultados: ILS exhibe una variabilidad ligeramente mayor en comparación con BL, y ILS-ES muestra una variabilidad ligeramente mayor que ES.

ii. Análisis de las Observaciones: BL vs ILS / ES vs ILS-ES

Al comparar BL con ILS y ES con ILS-ES, se destacan diferencias notables en términos de fitness, tiempo de ejecución y variabilidad de los resultados.

En relación al fitness, se observa que ILS logra una mejora marginal en comparación con BL. Esto sugiere que la estrategia iterativa de ILS permite una exploración más exhaustiva del espacio de búsqueda, lo que conduce a soluciones de mayor calidad en términos de fitness. Sin embargo, la diferencia en fitness entre ILS y BL puede no ser significativa en todos los casos.

Por otro lado, al comparar ES con ILS-ES, se observa que ILS-ES muestra resultados similares o ligeramente inferiores en términos de fitness en comparación con ES. Aunque ILS-ES incorpora el proceso de ILS con Enfriamiento Simulado, la adición de esta estrategia

no garantiza una mejora significativa en la calidad de las soluciones en comparación con ES independiente.

En cuanto al tiempo de ejecución, se evidencia que ILS es considerablemente más lento que BL. Esto se debe a las múltiples iteraciones involucradas en el proceso iterativo de ILS, que aumentan el tiempo de ejecución. Por otro lado, ILS-ES es ligeramente más lento que ES, ya que la inclusión del proceso de Enfriamiento Simulado en cada iteración agrega una sobrecarga adicional al tiempo de ejecución.

En relación a la variabilidad de los resultados, se observa que ILS exhibe una variabilidad ligeramente mayor en comparación con BL. Esto puede atribuirse a las diferentes trayectorias de búsqueda seguidas por ILS durante sus múltiples iteraciones. Por otro lado, ILS-ES muestra una variabilidad ligeramente mayor que ES, lo que sugiere que la combinación de estrategias de búsqueda puede conducir a resultados menos consistentes en comparación con ES independiente.

iii. Conclusión

En resumen, la comparativa entre BL vs ILS y ES vs ILS-ES resalta las diferencias en términos de fitness, tiempo de ejecución y variabilidad de resultados. Si bien ILS puede proporcionar una mejora marginal en fitness en comparación con BL, esta mejora puede no ser significativa en todos los casos. Además, la incorporación del proceso de Enfriamiento Simulado en ILS-ES no garantiza una mejora significativa en la calidad de las soluciones en comparación con ES independiente. La elección entre BL, ILS y ES, ILS-ES dependerá de las prioridades del problema en términos de calidad de la solución y eficiencia en el tiempo.

d. Conclusiones Finales

Tras analizar detalladamente los resultados y comparaciones entre los diferentes algoritmos metaheurísticos aplicados al problema de la mochila cuadrática, se pueden extraer varias conclusiones importantes.

i. Rendimiento de los Algoritmos:

El algoritmo de Búsqueda Multiarranque Básica (BMB) destaca como el mejor algoritmo estudiado en esta práctica. Su capacidad para explorar múltiples soluciones iniciales y aplicar Búsqueda Local a cada una de ellas le permite obtener consistentemente mejores valores de fitness en comparación con otros algoritmos como Búsqueda Local (BL), Iterative Local Search (ILS) y Enfriamiento Simulado (ES).

Si bien ILS y ES presentan ciertas ventajas en términos de tiempo de ejecución, su desempeño en términos de fitness y estabilidad de los resultados es inferior al de BMB. La capacidad de BMB para explorar exhaustivamente el espacio de búsqueda a través de múltiples arranques le otorga una ventaja significativa en la calidad de las soluciones obtenidas.

ii. Diferencias entre los Algoritmos:

La principal diferencia entre los algoritmos radica en su enfoque de exploración y explotación del espacio de búsqueda. Mientras que BL y BMB se centran en la explotación intensiva de soluciones locales, ILS y ES incorporan elementos de exploración más amplia mediante la exploración iterativa o la aceptación probabilística de soluciones peores, respectivamente.
Esta diferencia fundamental en los enfoques de búsqueda se refleja en los resultados obtenidos. Los algoritmos que priorizan la exploración exhaustiva, como BMB y ES, tienden a obtener soluciones de mayor calidad en términos de fitness, aunque a costa de un mayor tiempo de ejecución y variabilidad en los resultados.

iii. Elección del Algoritmo:

La elección del algoritmo adecuado depende de las prioridades y restricciones específicas del problema. En situaciones donde se valora principalmente la calidad de la solución, especialmente en problemas de tamaño y complejidad elevados, BMB es la mejor opción debido a su capacidad para encontrar soluciones óptimas o cercanas a óptimas.
Por otro lado, si el tiempo de ejecución es un factor crítico y se puede tolerar una ligera disminución en la calidad de la solución, ILS y ES pueden ser alternativas viables debido a su eficiencia en términos de tiempo.
En resumen, la elección del algoritmo metaheurístico adecuado debe basarse en un análisis exhaustivo de las características del problema, teniendo en cuenta tanto la calidad de la solución requerida como las restricciones de tiempo de ejecución. En el contexto del problema de la mochila cuadrática, Búsqueda Multiarranque Básica (BMB) destaca como el algoritmo más efectivo y versátil, ofreciendo un equilibrio óptimo entre calidad de la solución y eficiencia en el tiempo de cálculo.

Trabajo realizado por Santiago Herron Mulet