OPTIMIZACIÓN USANDO METAHEURÍSTICAS

COHORTE 1: IMPLEMENTACIÓN DE ALGORITMOS DE ESTADO SIMPLE EN PROBLEMAS CONTINUOS, DISCRETOS Y BINARIOS

OBJETIVO:

Realizar el algoritmo Greedy Randomized Adaptive Search (GRASP) y el Simulated Annealing (SA), y los dos algoritmos base de referencia realizados en clase (Hill Climbing y Hill Climbing with Random Restarts) para resolver los tres (3) problemas vistos en clase, a saber:

1) Optimización de funciones mono y multi modales, separables y no separables (optimización continua sin restricciones) usando 50 dimensiones.
2) Problema de la mochila binaria - 0/1 knapsack (optimización binaria con restricciones) con los problemas provistos en clase.
3) Problema del vendedor viajero o TSP (optimización discreta sin restricciones) con los problemas provistos en clase.

Los operadores propuestos (diferentes a los vistos en clase) en GRASP y SA para la generación de la solución inicial (inicialización o construcción como se llama en GRASP) y Tweak para la búsqueda local, deben incluir aleatoriedad (operador probabilístico), no puede ser del todo determinístico.

Luego, comparar los resultados de los cuatro algoritmos (GRASP, SA, HC y HCRR) en una tabla que muestre el fitness promedio de 31 repeticiones de cada experimento por cada problema que se resuelve. Además, reportar el mejor y el peor valor de fitness, la desviación estándar del fitness y el tiempo promedio de ejecución.

Realizar una figura donde se comparen las curvas de convergencia promedio (de las 31 repeticiones) para cada algoritmo (GRASP, SA, HC y HCRR) en cada experimento.

PRESENTACIÓN DEL TALLER:

Se debe hacer una exposición en la que se debe utilizar el texto estrictamente necesario y mayoritariamente gráficas para una mayor comprensión de lo que se explique. Debe contener como mínimo lo siguiente:

1. Variaciones de los operadores (Inicialización y Tweak) en cada problema (1 diapositiva por operador en cada problema … máximo 6 diapositivas):
   - Esquema gráfico o seudocódigo del operador implementado con ejemplo sencillo.

2. Evaluación de resultados de los algoritmos:
   - Para esta sección tener en cuenta lo siguiente:
     - Afinar los parámetros del algoritmo teniendo en cuenta como mínimo tres valores por cada parámetro.
     - Para el número máximo de evaluaciones de la función objetivo (MAX_EFOs) se debe tomar 50.000, pero se debe incluir en todos los algoritmos, las instrucciones requeridas para que el algoritmo se detenga cuando se encuentre el valor óptimo reportado en la literatura (en los casos que se conozca dicho valor óptimo con una precisión de 1E-5 en el caso de números reales). Asegure en todo caso una comparación JUSTA. NOTA: Tener en cuenta esto para graficar apropiadamente las curvas de convergencia.
   - Reportar los resultados siguiendo la **Tabla 1** para cada problema con sus respectivas instancias o problemas; es claro que se deben reportar los cuatro algoritmos (GRASP, SA, HC y HCRR).
   - Realiza un análisis de los resultados obtenidos. Revise el test no paramétrico de Friedman y Wilcoxon (https://www.statology.org/friedman-test-python/) usando scipy.stats de python, el software KEEL Suite 3.0 (http://www.keel.es/) u otro según considere apropiado. Ver ejemplo abajo en Python.

3. Presentar sus conclusiones del trabajo.

Mochila Binaria	… HCRR					SA …				
	Media	**Desviación**	**Mejor**	**Peor**	**Tiempo**	**Media**	**Desviación**	**Mejor**	**Peor**	**Tiempo**
F1	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor
F3	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor
..	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor
Knapsack6	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor
Promedio	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor

Tabla 1 Comparación HC, HCRR, SA y GRAPS para el problema de la Mochila Binaria.

NOTA: El resultado promedio de cada experimento (valor que se coloca en cada fila de la tabla de resultados) es el promedio de las 31 repeticiones (ejecuciones) del mismo algoritmo con los mismos parámetros usando diferente semilla. Esto permite que el cálculo de la media, desviación estándar y tiempo muestre un valor esperado más real (teorema del límite central).

ENTREGA VÍA E-MAIL
Entregar vía e-mail a ccobos@unicauca.edu.co lo siguiente:
1. Código de los algoritmos implementados. NO ENVIAR EJECUTABLES.
2. Diapositivas en Power Point de la presentación.

FECHAS DE SUSTENTACIÓN
Octubre 6 2023 de 9:00 a.m. a 11:00 a.m.

DURACIÓN Y ORDEN DE SUSTENTACIÓN
La duración de cada sustentación será de máximo 15 minutos. El orden será basado en orden alfabético de lista. Si alguien tiene problemas con el día y la hora DEBE previamente solicitar que se adelante su sustentación.

TRABAJO EN GRUPO
Los grupos deben ser de 2 personas y máximo se aceptará que una (1) persona trabaje individualmente. El plazo máximo para informar por correo a ccobos@unicauca.edu.co la conformación de los grupos es el viernes 22 de octubre a las 11:00 a.m. Hora Colombia (GMT-5) y se tendrá en cuenta la fecha del correo para la definición de la persona que trabajará sola.

GRUPOS

Id	Nombres y Apellidos	Hora/Nota	Observaciones
1			
2			
3			
4			
5			
6			
7			

Pruebas no paramétricas de Friedman, Mann-Whitneyu y Wilcoxon

```
from scipy.stats import friedmanchisquare
from scipy.stats import mannwhitneyu
from scipy.stats import wilcoxon
from numpy import mean

alpha = 0.05

groups = []
groups.append([4, 6, 3, 4, 3, 2, 2, 7, 6, 5])
groups.append([5, 6, 8, 7, 7, 8, 4, 6, 4, 5])
groups.append([2, 4, 4, 3, 2, 2, 1, 4, 3, 2])

for i in range(0,3):
  print('Mean for group[' + str(i) + '] = ' + str(mean(groups[i])))
print('')

stat, p = friedmanchisquare(groups[0], groups[1], groups[2])
print('Friedman ChiSquare Statistics = %.3f, p = %.3f' % (stat, p))
if p > alpha:
  print('Todos tienen distribución similar (Se falla al rechazar H0)')
else:
  print('Al menos un grupo tienen una  distribución diferente (Se rechaza H0)')

print ('')
for i in range(0,2):
  for j in range(i+1,3):
    print ('group[' + str(i) + '] vs group[' + str(j)+ ']:', end = ' ')
    stat, p = mannwhitneyu(groups[i], groups[j])
    print('Mann-Whitneyu Statistics = %.3f, p = %.3f' % (stat, p), end = '. ')
    if p > alpha:
      print('Similar distribución (Se falla al rechazar H0)')
    else:
      print('Diferente distribución (Se rechaza H0)')

print ('')
for i in range(0,2):
  for j in range(i+1,3):
    print ('group[' + str(i) + '] vs group[' + str(j)+ ']:', end = ' ')
    stat, p = wilcoxon(groups[i], groups[j], mode='auto')
    print('Wilcoxon Statistics = %.3f, p = %.3f' % (stat, p), end = '. ')
    if p > alpha:
      print('Similar distribución (Se falla al rechazar H0)')
    else:
      print('Diferente distribución (Se rechaza H0)')
```