PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ALGORITMOS AVANZADOS

5ta. práctica (tipo B)
(Primer Semestre 2024)

Duración: 2h 00 min.

- **No puede utilizar apuntes, solo hojas sueltas en blanco.**
- En cada función el alumno deberá incluir, a modo de comentario, la forma de solución que utiliza para resolver el problema. De no incluirse dicho comentario, el alumno perderá el derecho a reclamo en esa pregunta.
- No puede emplear plantillas o funciones no vistas en los cursos de programación de la especialidad.
- Los programas deben ser desarrollados en el lenguaje C++. Si la implementación es diferente a la estrategia indicada o no la incluye, la pregunta no será corregida.
- Un programa que no muestre resultados coherentes y/o útiles será corregido sobre el 50% del puntaje asignado a dicha pregunta.
- Debe utilizar comentarios para explicar la lógica seguida en el programa elaborado. El orden será parte de la evaluación.
- Se utilizarán herramientas para la detección de plagios, por tal motivo si se encuentran soluciones similares, se anulará la evaluación a todos los implicados y se procederá con las medidas disciplinarias dispuestas por la FCI.
- **Solo está permitido acceder a la plataforma de PAIDEIA, cualquier tipo de navegación, búsqueda o uso de herramientas de comunicación se considera plagio por tal motivo se anulará la evaluación y se procederá con las medidas disciplinarias dispuestas por la FCI.**
- Para esta evaluación solo se permite el uso de las librerías **iostream, iomanip, climits cmath, fstream, vector, algorithm, string, cstring, numeric, ctime y cstdlib**
- Su trabajo deberá ser subido a PAIDEIA.
- **Es obligatorio usar como compilador NetBeans.**
- Los archivos deben llevar como nombre su código de la siguiente forma codigo_LAB5_P# (donde # representa el número de la pregunta a resolver)

Pregunta 1 (20 puntos)

Adaptar el Algoritmo Genético del problema de la mochila (archivo Knapsack_genetico.cpp en PAIDEIA) para el siguiente problema:

Se tiene un conjunto de N activos a comprar (terrenos, inmuebles, acciones, etc.), los cuales tienen un precio actual, un precio futuro optimista dentro de un año y un precio futuro pesimista dentro de un año. Lo que se quiere es que el algoritmo escoja un portafolio de activos que maximicen la suma total de posibles ganancias, substrayendo la suma total de posibles pérdidas, es decir el fitness que se quiere maximizar seria: |suma de precios optimistas del portafolio – precio actual del portafolio| - |suma de precios pesimistas del portafolio - precio actual del portafolio). Se dispone de una cantidad de dinero máxima para invertir. Las soluciones no deberían pasarse de ese monto.

Para almacenar la información de activos se sugiere la siguiente estructura:

```
struct Activo {
    double precio_actual;
    double precio_futuro_optimista;
    double precio_futuro_pesimista;

    Activo(double pa, double pfo, double pfp) : precio_actual(pa), precio_futuro_optimista(pfo), precio_futuro_pesimista(pfp) {}
};
```

La representación cromosómica del portafolio puede ser un vector binario del tamaño del pool de activos, donde el valor de cada elemento indica si el activo correspondiente forma parte del portafolio (valor 1) o no (valor 0).

Usted deberá:

- Implementar la función calculate_fitness( ) de acuerdo a los requerimientos del problema
- Modificar función evaluate_population() para recibir los argumentos adecuados para evaluar una población de individuos
- Hacer los cambios pertinentes a la función genetic_algorithm() para pasar los argumentos adecuados a todas las funciones anteriormente modificadas. Hacer que genetic_algorithm() retorne el fitness del mejor individuo encontrado.
- Crear una nueva función init_population1() que inicialice individuos que no se pasen de la restricción del dinero máximo que se puede invertir.

Una vez hechas las modificaciones, experimentar el algoritmo con el siguiente pool de 20 activos:

```
vector<Activo> assetPool = {
  {100, 150, 80}, {200, 250, 180}, {150, 220, 130}, {120, 180, 100}, {180, 230, 160},
  {110, 170, 90}, {130, 200, 110}, {160, 210, 140}, {140, 190, 120}, {210, 270, 190},
  {170, 240, 150}, {190, 260, 170}, {220, 300, 200}, {200, 280, 180}, {180, 250, 160},
  {195, 260, 175}, {205, 275, 185}, {215, 285, 195}, {225, 295, 205}, {235, 305, 215}
};
```

Usando una máxima inversión de S/1,000.00, con una población inicial de 10 individuos, con 100 generaciones, tamaño de torneo = 3 o una ruleta con n de 50% de progenitores y finalmente debe usar un cruzamiento uniforme. Con la información mencionada debe realizar los siguientes experimentos:

1) Para cada una de las siguientes tazas de mutación: **0**, **0.4** y **0.8**, ejecutar 10 veces el algoritmo, en cada vez inicializando la población con función init_population( ). Registrar el promedio del mejor fitness encontrado para cada taza de mutación.
2) Repetir los experimentos anteriores cambiando la función de inicialización de población por la función init_population1( ) . Registrar el promedio del mejor fitness encontrado para cada taza de mutación.
3) Comentar los resultados en cuanto al efecto de la taza de mutación y la función de inicialización de individuos. Justificar por qué una función de inicialización de individuos genera mejores resultados que la otra.

Los resultados y comentarios deben ser registrados en una hoja de calculo que debe guardarse en la carpeta del proyecto desarrollado.

Al finalizar el laboratorio, comprima la carpeta de su proyecto empleando el programa Zip que viene por defecto en el Windows, **no se aceptarán los trabajos compactados con otros programas como RAR, WinRAR, 7zip o similares**. Luego súbalo a la tarea programa en Paideia para este laboratorio.

Profesores del curso:

Edwin Villanueva
Rony Cueva

San Miguel, 22 de junio del 2024