Tarea 1. Algoritmos Genéticos

Análisis de Algoritmos

1 Introducción

Muchos problemas de optimización que nos interesa resolver tienen la característica que no se les conoce un algoritmo exacto determinístico de complejidad P para resolverlos, pero por estar en NP, sí podemos, si nos dan una solución candidata y verificar la calidad de esta rápidamente.

Este tipo de problemas, sugiere una buena estratégia de solución es generar respuestas candidatas al azar para verificarlas. Un modelo de solución de problemas que se adapta bien a este tipo de problemas es el modelo de algoritmos genéticos.

En esta tarea exploraremos un poco a los algoritmos gen´eticos, los cuales son una estratégia de solución de problemas inspirada en la teoría de evolución que se presta bien para resolver este tipo de problemas, en particular, los problemas NP, NP-completos y NP-duros.

En esta tarea ud. debe programar 2 algoritmos genticos para resolver dos problemas distintos, en el proceso de resolverlos debe parametrizar sus algoritmos tal que pueda hacer pruebas con distintos valores para cada uno de los algoritmos, a continuacin una descripcin de cada uno de los problemas.

La lectura del disco de cada archivo problema es dependiente del problema y se explica en el apartado para los distintos tipos de problema a implementar.

2 Estructura del algoritmo genético

En el diseño de este algoritmo genético ud debe desacoplar al máximo la estructura del algoritmo genético de la estructura del problema. Para ello debe construir la clase del algoritmo genético,, y una clase abstracta para el problema, de la cual debe derivar para construir cualquier problema que quiera resolver con el algoritmo genético.

```
abstract class Problema
        readProblema(file : String) // Lee un problema de un archivo
        geneSize() : Int // tamaño del gen
        fitness(x : gene) : Real // cálculo del fitness
        name() : String // hilera que indica el nombre del problema
end

type Seleccion = Azar | RuedaFortuna | Torneo Int

type gene = vector{0,1}

class Algoritmo(
        Politica : Seleccion,
        NumCruces : Int, // cruce 1, 2 o mas puntos
        Mutacion : Real, // porcentaje de mutación
        TamPoblacion : Int, // Tamaño de la población
        P : class Problema)

        resetPoblacion(Tamaño : Int)
        readPoblacion(file : String)
        writePoblacion(file : String)
        generacion()
        getBest() : gene
end
```

2.1 Clase Problema

La clase problema se encarga de leer de disco de un archivo de texto con la especificación de un problema particular, se debe heredar de la clase problema para hacer un problema particular. Cada archivo problema debe indicar que tipo de problema es, En su programa ud. debe heredar de la clase problema para crear cada tipo de problema. El tipo de problema estara indicado por un número que corresponde a la numeración asignada al tipo de problema que se encuentra en el libro *Computers and Intractability*.

2.2 Clase Algoritmo

La clase Algoritmo es la clase que implementa el algoritmo genético y requiere como parámetros para su constructor a la política, el número de puntos de cruce, el nivel de mutación y el problema. Los demás métodos se explican a continuación.

- resetPoblacion toma un tamaño de población y genera una población aleatoria, este método consulta al objeto problema para saber el tamaño del gen.
- readPoblación : lee una población de un archivo, el archivo de población tiene el siguiente formato
  1. nombre del problema en la primera linea
  2. cantidad de genes
  3. lista de genes, 1 por linea escritos como una secuencia de 1's y 0's.

3 GT1 Vertex Cover

Ud debe resolver el problema de recubrimiento de vertices (Vertex Cover) tal y como se especifica en la página 190 de computers and intractabiity.

Utilizando algoritmos genéticos. La información para de la especificación del problema se encontrará en un archivo de texto con el siguiente formato:

3.1 Archivo de datos

El archivo de datos vendrá con los datos en en siguiente orden, un dato en cada linea:

1. la hilera "GT1" en la primera linea indicando el tipo de problema
2. Un número N indicando la cantidad de vertices (los vertices se asumen numerados de 1 hasta N)
3. Un número M indicando la cantidad de aristas (arcos)
4. Un número K indicando el tamaño del recubrimiento deseado
5. Para cada arco, un par de numeros indicando los vertices que dicho arco conecta

Para este problema ud debe 1. escribir el código de la clase que hereda de la clase problema. y 2. Escribir un programa que recibe 3 parámetros, N, M y K y genera un archivo problema aleatorio con un grafo de N nodos y M aristas.

4 SP5 Recubrimiento Mínimo

Este problema se encuentra en la página 222 del libro *Computers and Intractability*

4.1 Archivo de datos

Para este segundo problema el archivo de datos va a contener la siguiente información

1. la hilera "SP5" en la primera linea del archivo
2. número N indicando el tamaño del del conjunto S (se asume que los elementos de S son los enteros $1 \ldots N$
3. número M indicando la cantidad de subcojuntos de S que se encuentran en C
4. en cada linea siguiente una lista de enteros de tamaño variable que se refiere a cada uno de los elementos de C

5 Sus programas

Ud debe desarrollar un programa principal que resuelva los problemas anteriores, además de un generador de problemas para cada tipo de problema.

El programa que resuelve el problema debe ejecutarse desde la linea de comandos y tener los siguientes parámetros.

1. nombre del archivo del problema.
2. nombre del archivo de datos con la población inicial.
3. cantidad de generaciones
4. tamaño de la población, si este valor se especifica en 0 entonces debe ignorarse y tomar la población que se encuentra especificada en el archivo de datos, si no entonces debe empezar con una nueva y generarla aleatoriamente.
5. nivel de mutación (en porcentaje).

6. tipo de selección de padres a utiizar (0 = simple al azar, 1 = rueda de la fortuna, 2 torneo), si utiliza cruce simple al azar su algoritmo debe ser de punto fijo

7. Archivo de datos para el output, este archivo debe estar en el mismo formato que el archivo de datos de entrada.

Por ejemplo la siguiente llamada:

```
> genetico mincover.txt datos.txt 100 1000 3 2 output.txt
```

El formato del archivo de salida de su programa debe ser idéntico al de entrada, tal que lo pueda reutilizar para mas iteraciones. El programa debe terminar presentando un reporte a pantalla desplegando la mejor solución y su costo asociado.

5.1 Entregables

5.1.1 Programas

Debe entregar sus programas en código fuente mas un pequeño documento explicando sus resultados mas cuadros comparativos para distintos niveles de mutación, cruce, y tamaño de la población, los análisis de resultados se explican a continuación.

5.1.2 Análisis de resultados

Debe hacer gráficos comparativos que evaluen la influencia de los siguientes parámetros en la convergencia de su algoritmo genético con respecto a la calidad de la respuesta final (fitness, o bien la capacidad del algoritmo genético de resolver el problema propuesto).

Debe contemplar por lo menos 3 valores distintos en cada parámetro a analizar.

- tipo de selección de padres (Simple al azar, ruleta, torneo)
- tamaño de la población (50, 500, 5000)
- mutación (1%, 3%, 5%)
- cruce (1, 2, 3 puntos)

6 Fecha de Entrega: Miércoles 15 de Octubre

La tarea se puede trabajar en parejas o individualmente.