EIF400 – Paradigmas de Programación
Proyecto de programación #1

Prof. M.Sc. Georges E. Alfaro S.

DESCRIPCIÓN DEL PROBLEMA

Se utilizarán algoritmos genéticos para agrupar *n* números en *k* grupos disjuntos, donde se busca minimizar la diferencia de la suma de los elementos de los conjuntos. Es decir, se busca que la distribución de la suma de los números sea lo más uniforme posible. Los conjuntos pueden ser de cualquier tamaño, pero existe la restricción que ninguno puede ser vacío. El problema es similar al problema de la mochila (*Knapsack Problem*).

Deberá escribir un conjunto de funciones en Scheme para resolver el problema planteado.

ALGORITMOS GENÉTICOS

En los años 1970, de la mano de John Henry Holland, surgió una de las líneas más prometedoras de la inteligencia artificial, la de los algoritmos genéticos, (AG). Son llamados así porque se inspiran en la evolución biológica y su base genético-molecular.

Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados.

Los algoritmos genéticos se enmarcan dentro de los algoritmos evolutivos, que incluyen también las estrategias evolutivas, la programación evolutiva y la programación genética.[1]

Los algoritmos genéticos (AG) funcionan entre el conjunto de soluciones de un problema llamado fenotipo, y el conjunto de individuos de una población natural, codificando la información de cada solución en una cadena, generalmente binaria, llamada cromosoma. Los símbolos que forman la cadena son llamados genes. Cuando la representación de los cromosomas se hace con cadenas de dígitos binarios se le conoce como genotipo. Los cromosomas evolucionan a través de iteraciones, llamadas generaciones. En cada generación, los cromosomas son evaluados usando alguna medida de aptitud. Las siguientes generaciones (nuevos cromosomas), son generadas aplicando los operadores genéticos repetidamente, siendo estos los operadores de selección, cruzamiento, mutación y reemplazo.

[1] https://es.wikipedia.org/wiki/Algoritmo_gen%C3%A9tico

FUNCIONALIDAD DEL PROGRAMA

La función principal del programa deberá recibir como parámetro: el número máximo de generaciones a evaluar, el tamaño de la población, un valor lógico indicando si se desea utilizar elitismo o no[2], el número de grupos (*k*) y una lista con los valores a utilizar (la longitud de la lista determina el valor de *n*).

Por ejemplo:

```
(resolver 5000 30 #t 4 '(30 60 90 25 20 15 16 120 200 43 18 30 30)
```

indica que se van a agrupar 13 números en 4 grupos, evaluando 5000 generaciones de 30 individuos utilizando elitismo.

En cada generación se mostrará el mejor individuo de la población. La solución obtenida es el mejor individuo cuando el algoritmo finalice, ya sea porque se han evaluado todas las generaciones solicitadas o porque se haya cumplido el criterio de optimalidad.

La solución es una lista de sublistas, donde cada una incluye el grupo formado y la suma obtenida.

Por ejemplo, un individuo podría estar descrito por:

```
'(((30 60 90) 180) ((25 20 15 16 120) 196) ((200) 200) ((43 18 30 30) 121))
```

Finalmente, cuando se cumplan las condiciones de finalización indicadas en la invocación, se mostrará el resultado con la conformación de cada uno de los grupos, junto con la diferencia que tiene cada grupo con los demás.

Por ejemplo:

```
(resolver 5000 30 #t 4 '(1 2 3 5 6 9 32 1 2 5 12 31 15))
```

podría producir un individuo como:

```
'(((32) 32) ((31) 31) ((12 15 1 1) 29) ((2 3 5 6 9 2 5) 32))
```

Tenga en cuenta que esta es únicamente la representación externa del individuo. Es decir, la manera en que se muestra al usuario. La representación interna utilizada por el algoritmo puede ser completamente diferente. Observe por ejemplo que esta representación obliga a almacenar todos los elementos cada vez, lo cual podría ser muy ineficiente desde el punto de vista de espacio de almacenamiento.

El valor de las diferencias (positivas) entre la suma de cada grupo y los demás es:

```
((0 1 2 1) (1 0 2 1) (3 2 0 3) (0 1 3 0))
```

Se deberá definir una función que calcule la diferencia total (por ejemplo, evaluando el promedio de la distancia entre cada vector), la cual deber ser minimizada, para utilizarla como criterio de finalización.

Con los parámetros adecuados, es factible obtener la respuesta óptima en un número de generaciones relativamente pequeño.

Observe que, si existe una distribución perfecta, es decir, donde las diferencias entre la suma de cada grupo es 0, el algoritmo debería terminar inmediatamente. De la misma manera, si existe tal solución, se espera que el algoritmo converja a dicha solución. Por supuesto, esto no excluye la posibilidad de que existan varias soluciones óptimas.

[2] Es decir, si se desea conservar o no los mejores individuos de una generación en la siguiente.

OBJETIVOS DEL PROYECTO

El proyecto tiene como objetivo estudiar algunas de las técnicas fundamentales de programación declarativa, especialmente el uso de recursividad y la composición como estructura fundamental de organización para un programa. También se busca estudiar formas alternativas de representación de datos por medio de listas.

Se pretende también que el estudiante conozca y aplique algunos conceptos de programación probabilística, tales como el diseño e implementación de algoritmos genéticos.

REFERENCIAS

Adjunto a este enunciado encontrará un documento sobre algoritmos genéticos. Además, se pueden consultar las siguientes referencias:

http://www.it.uc3m.es/jvillena/irc/practicas/06-07/05.pdf

http://sabia.tic.udc.es/mgestal/cv/aaggtutorial/tutorialalgoritmosgeneticos.pdf

https://www.tutorialspoint.com//genetic_algorithms/index.htm

https://karczmarczuk.users.greyc.fr/TEACH/IAD/GenDoc/carrGenet.pdf

Mitchel, Melanie. An Introduction to Genetic Algorithms, MIT Press, 1999, ISBN 0-262-13316-4 (HB)

ENTREGA Y EVALUACIÓN

El proyecto debe entregarse **por medio del aula virtual**. La fecha límite de entrega es antes del día **viernes 27 de octubre de 2017**. No se aceptará ningún proyecto después de esa fecha, ni se admitirá la entrega del proyecto por correo electrónico. Este proyecto debe realizarse en grupos de **dos personas como máximo**.

Incluya comentarios en el código de los programas y describa adecuadamente cada una de las funciones utilizadas. Todas las funciones o predicados deberán ser descritas en un documento aparte, en formato PDF, donde se explique la definición de la función y como calcula el resultado.

En caso de que las aplicaciones no funcionen adecuadamente, efectúe un análisis de los resultados obtenidos, indicando las razones por las cuales el o los programas no trabajan correctamente, y cuáles son las posibles correcciones que se podrían hacer.

El proyecto se evaluará de acuerdo con la siguiente rúbrica:

Rubro	Valor			
Definición de los individuos	**5%**	-1% Si la valoración de los individuos utiliza un algoritmo con un costo mayor a O(n)	-5% Si no representa correctamente el problema	
Definición de la función objetivo	**5%**	-2% Si la función objetivo tiene un costo mayor a $O(n^2)$	-5% Si la función objetivo está mal definida o no concuerda con el problema	-2% Si no documenta adecuadamente la función objetivo
Definición y uso de atributos de ejecución (cruces, mutaciones, elitismo)	**15%**	-2% Si no se utilizan 1 o 2 atributos de ejecución	-5% Si n se utilizan más de 2 atributos de ejecución	
Funcionamiento general	**60%**	-5% Si no se muestran el mejor individuo o la élite de cada generación	-30% Si el algoritmo no minimiza las diferencias	-60% Si el algoritmo no converge o no encuentra una solución aceptable
Estructura del código	**15%**	-5% Si utiliza ciclos en lugar de recursividad	-5% Si utiliza variables de manera innecesaria en lugar de parámetros	-5% Si no utiliza parámetros funcionales cuando es adecuado

OBSERVACIONES GENERALES:

- Los trabajos no se copiarán de ninguna llave USB u otro dispositivo en el momento, sino que se deben entregar en el formato solicitado.
- Cualquier trabajo práctico que no sea de elaboración original de los estudiantes (plagio) se calificará con nota 0 (cero) y se procederá como lo indiquen los reglamentos vigentes de la universidad.