EIF400 – Paradigmas de Programación
Proyecto de programación #1

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

PLANTEAMIENTO 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 (DrRacket) 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 en los algoritmos evolutivos, que incluyen también las estrategias evolutivas, la programación evolutiva y la programación genética.

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.

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, 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 500 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 500 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.

En este otro ejemplo:

```
(resolver 500 6 #t 4 '(1 2 3 5 6 9 32 1 2 5 12 31 15))
```

La función podría producir un individuo como:

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

El número de sublistas en el individuo es el número de grupos (4 en este caso). 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 que esta representación obligaría a almacenar todos los elementos cada vez, separando las listas y los subtotales, 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 puede ser:

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

Si los grupos deben sumar 31 (en el caso ideal), la sublista '(28 30 32 38) tendría las diferencias '(3 1 1 7). 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. Una solución óptima tendría un vector de diferencias: '(0 0 0 0).

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.

El esbozo general del programa es:

- se genera un conjunto aleatorio de individuos (según el tamaño de la población)
- se ordena la población de acuerdo con el valor la función objetivo
- mientras no se haya encontrado una solución ni se hayan generado la totalidad de las generaciones especificadas:
    - se separan las mejores soluciones obtenidas (si se emplea elitismo)
    - se completa la nueva generación por medio de cruces entre individuos y mutaciones (la proporción de cruces y mutaciones debe ser un parámetro de la función). Los individuos no factibles son eliminados.
    - se vuelve a evaluar la función objetivo

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 e inteligencia artificial, tales como el diseño e implementación de algoritmos genéticos.

CONSIDERACIONES DE IMPLEMENTACIÓN

Todas las funciones se escribirán en un único archivo fuente.

Puede utilizar funciones de la biblioteca estándar de DrRacket para resolver el problema, pero solamente deberá usar listas para representar los datos (no se permite el uso de arreglos u otras estructuras). En general, tampoco se permite el empleo de variables, sino que deberá depender solamente de los parámetros de cada función. Si usa alguna variable, es necesario incluir comentarios en el código que justifiquen su necesidad.

ENTREGA Y EVALUACIÓN

El proyecto debe entregarse **por medio del aula virtual, en el espacio asignado para ello**. La entrega es al finalizar la semana 13 (**sábado 6 de noviembre de 2021**). No se aceptará ningún proyecto después de esa fecha, ni se admitirá la entrega del proyecto por correo electrónico. El proyecto se puede realizar en grupos de **tres personas, como máximo**. Deberán enviar un correo al profesor indicado la lista de participantes del grupo antes del viernes de la semana 11 (22 de octubre de 2021).

Incluya comentarios en el código de los programas y describa detalladamente cada una de las clases y métodos utilizados. Todas las funciones deberán ser descritas en el código fuente, donde se explique la definición de cada función y como calcula el resultado

Incluya un comentario al inicio de cada archivo fuente, indicando información básica, como se muestra abajo:

```
;
; ================================================
;
; (c) 2021
; version 1.0.0 2021-10-24
;
; ------------------------------------------------
; EIF400 Paradigmas de Programación
; 2do ciclo 2021, grupo 01
; Proyecto 1
;
; 12345678 González Abarca, Adriana
; 87654321 Montero Rodríguez, Carlos
;
; ================================================
;
```

En caso de que la aplicación no funcione adecuadamente, efectúe un análisis de los resultados obtenidos, indicando las razones por las cuales el programa no trabaja correctamente, y cuáles son las posibles correcciones que se podrían hacer. Durante la revisión del proyecto, es muy importante poder defender adecuadamente la solución propuesta.

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 no 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 proyectos deben entregarse con toda la documentación, diagramas, código fuente y cualquier otro material solicitado, utilizado el mecanismo indicado por el profesor (el aula virtual, en este caso).
- Se debe indicar en cada documento el nombre completo y cédula de cada participante del grupo, indicando el nombre del curso, ciclo lectivo y descripción del trabajo que se entrega. Esto incluye comentarios en cada archivo fuente entregado.
- Si los materiales de entrega no están completos, se penalizará hasta un 15% de la nota correspondiente. Asimismo, 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.
- Los trabajos que se reciban después de la fecha señalada para su entrega, **en caso de ser aceptados, serán penalizados con un 30% de la nota por cada día de atraso.**

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

Carr, J. (16 de mayo de 2014). *An Introduction to Genetic Algorithms.* Obtenido de Whitman College: https://www.whitman.edu/Documents/Academics/Mathematics/2014/carrjk.pdf

Mallawaarachchi, V. (7 de julio de 2017). *Introduction to Genetic Algorithms*. Obtenido de Towards Data Science: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

Mitchel, M. (1999). *An Introduction to Genetic Algorithms.* Cambridge, Massachusetts, USA: MIT Press.

Muthee, A. (26 de mayo de 2021). *The Basics of Genetic Algorithms in Machine Learning*. Obtenido de Section: https://www.section.io/engineering-education/the-basics-of-genetic-algorithms-in-ml/