**Universidad Nacional de Costa Rica**

**Facultad de Ciencias Exactas y Naturales**

**Escuela de Informática**

**Paradigmas de programación**

**Profesor:**

M. Sc. Georges Alfaro

***Detalle de implementación del proyecto sobre algoritmos genéticos utilizando programación funcional***

**Estudiantes**

|  |  |  |
| --- | --- | --- |
| Joan Carballo Badilla | José Mejía Bravo | Jesús Murillo Miranda |
| 1-1590-0574 | 1-1654-0323 | 1-1632-0918 |

**Grupo:** 6 p.m.

7 de noviembre del 2017

**Descripción de funciones**

|  |  |
| --- | --- |
| Función | Resultado |
| distribuir | Método padre, llama a sus hijos para distribuir sus elementos de forma aleatoria. |
| d1 | Obtiene lo hijos de una lista principal para proceder a su distribución de manera aleatoria. |
| d2 | Obtiene los hijos de una lista principal para ordenarlos de manera aleatoria. (Se usa en d1). |
| sumaSubL | Esta función se encarga de generar la suma de cada sublista. |
| obtieneMayorEnLista | Esta función se encarga de devolver el primer elemento de una lista ordenada de acuerdo a la suma de su sublistas, devolviendo el elemento mayor de la lista. |
| ordenaPorSuma | Función encargada de ordenar la lista de acuerdo a las sumas de sus sublistas de una manera descendente. |
| uneIndividuoSuma | Esta función se encarga de adjuntar el resultado de la suma previamente conseguida por suma-l y adjuntarla al lado derecho de la lista |
| distribuyeCompleto | Valida que los todos los elementos que se crearon después de una mutación no presente ninguno vacío, si es el caso, vuelve a generar la lista de la generación. |
| individuoVacio? | Verifica que las sablistas que se generen de manera aleatoria en la primera generación, no creen elementos vacíos. |
| existeVacia? | Función para validar que no exista una sablista vacía. |
| generacionInicial | Subdivide los elementos de manera aleatoria y así crea la primera generación a ser evaluada. |
| dropSumas | Separa las sumas previamente generadas de cada sublista, para poder volver a generar nuevas generaciones. |
| parIndividuoDesviacion | Adjunta la desviación estándar que exista dentro de todo un hijo de una generación y lo adjunta al final de la misma. |
| ordenaDesviacion | Ordena la generación para que quede de menor a mayor según la desviación estándar previamente evaluada en cada hijo. |
| getOptimo | Devuelve el resultado final del proyecto si se obtiene en algún momento un hijo con una desviación estándar igual a 0. |
| dropDesviacion | Remueve la desviación estándar previamente generada de cada elemento de la generación. |
| dropTodasSumas | Elimina la suma de cada sublista de todos los elementos de la generación. |
| getEnPosicion | Obtiene un elemento específico de la generación |
| auxEnPosicion | Evalúa que los elementos a obtener no sean el mismo. |
| ejecutarMutacion | Muta dos sublistas obtenidas de manera aleatoria. |
| mutacion | Genera los índices de las sublistas a mutar. |
| intercambio | Validación para cambiar elementos de una sublista a otra- |
| auxIntercambio | Cambia elementos de una sublista a otra llamando a intercambio para verificar el procedimiento. |
| mutarRestantes | Agarra la lista original presente en mutarEnLista y procede a agregarle los hijos mutados para generar una nueva generación a evaluar. |
| mutarEnLista | Lista de hijos antes de mutar, mantiene el orden |
| obtenerSolucion | Función principal del proyecto, se encarga de llamar apropiadamente a cada sub método para generar todo el algoritmo genético. |
| generarProximaGen | Se encarga de obtener la siguiente generación a partir de ciertas mutaciones para volver a ser evaluado. |
| preparaProximaGen | Aplica los movimientos necesarios para que cada generación sea creada y evaluada |