**Preguntas del oponente**

**Pregunta 1:**

A pesar de que el algoritmo genético lograra un valor lo suficientemente pequeño de la función objetivo en menos iteraciones, ¿es aconsejable su uso a pesar de haber reducido el conjunto inicial a 41 casos, a diferencia de GRASP donde quedaron 163 casos?

**Respuesta 1:**

La respuesta a este tipo de preguntas siempre está sujeta al contexto. Es muy difícil generalizar sin saber el escenario. Ahora se está tratando el problema de multiplicación de polinomios. La característica más importante en estas circunstancias, es que el universo de los casos de prueba es muy pequeño (si tratamos de restringirlo de alguna forma a casos “independientes” entre sí). Por lo tanto, se puede concluir que sí es recomendable.

Utilizar un conjunto grande (de tamaño 161 en este caso), donde haya exceso de representación por cada tipo de caso, podría traer resultados no deseados como una de las observaciones que se hacen en la aplicación del algoritmo genético con las soluciones de los estudiantes en el examen de Anagramas: el aumento del porciento de efectividad de cada solución. Sucede que, al haber más casos, es probable que en ese conjunto final haya muchos que sean resueltos de manera satisfactoria por alguna solución, la cual, sin poder resolver todavía algunos casos, se ve beneficiada en el sentido del porciento. Habría que entonces hacer un mejor ajuste de los parámetros (lo esperado para cada nota) ante este tipo de situación.

**Pregunta 2:**

Si fuera a resolver el problema de la generación automática de casos, ¿qué consideraciones habría que tener? ¿Qué se debería garantizar para un correcto funcionamiento de la propuesta? Exponga brevemente cómo atacaría el problema.

**Respuesta 2:**

La mayor garantía que se debería tener en cuenta es lograr la representación del universo de casos. Para el correcto funcionamiento de cualquier algoritmo a aplicar, es necesario que reciba como entrada un conjunto representativo de casos de prueba de ese universo. Además, sería idóneo que cada tipo de caso esté representado con varios representantes, tratar de llegar a un equilibrio entre el exceso y el defecto.

Para resolver este problema diseñaría una clasificación para cada caso e intentaría generar una cantidad suficiente de cada tipo como solución final. Un tipo de clasificación podría ser los 4 grupos denominados “tipos” de casos de prueba, observación descrita en el Experimento 1 durante la ejecución de GRASP.

**Pregunta 3:**

Los experimentos realizados fueron sobre el problema "Suma de polinomios", donde comprobar si la solución es correcta para un caso se realiza en tiempo polinomial. ¿Es factible la utilización de esta solución en problemas donde comprobar la correctitud de una solución tenga costo exponencial?

**Respuesta 3:**

Aquí no entiendo muy bien a que se refiere. Siempre es necesario comprobar si una solución es correcta y ese proceso se hace una sola vez. Como tal el grueso de la tesis no se basa en la comprobación de si una solución es correcta, es más bien el empleo de metaheurísticas en la reducción de un conjunto. Si los datos de entrada estuviesen etiquetados, no habría necesidad de clasificarlos como parte de la propuesta.

Aun así, pienso que sí sería factible, solo que habría que tener en consideración otros factores como el tamaño de la entrada a clasificar. Ahora mismo se trató de ser abarcador en la búsqueda de un conjunto representativo para un mejor rendimiento.

**Pregunta 4:**

En los resultados obtenidos por GRASP se observa que el porcentaje deseado para la nota de 3 es de 30%, en la práctica no es un valor real deseado. ¿Qué sucede si se espera un valor en el rango [60,70]?

**Respuesta 4:**

Este ajuste fue experimental por las características del problema. Es preciso conocer cuáles fueron las soluciones usadas como nota de 3 y como nota de 4.

Nota de 4. Solo falla en un tipo de casos: en el que uno de los dos sea el polinomio 0 y el otro un polinomio de grado mayor que 1.

public static int[] MultPol4(int[] a, int[] b)

{

int[] result = new int[a.Length + b.Length - 1];

for (int i = 0; i < a.Length; i++)

for (int j = 0; j < b.Length; j++)

result[i + j] += a[i] \* b[j];

return result;

}

Nota de 3. Aquí el error es minúsculo en términos de similitud entre códigos, pero ese = en defecto del += causa que se obvie lo anteriormente acumulado de ese mismo grado. Por lo cual, esta solución falla en casi todos los casos, excepto en aquellos polinomios en que sean excluyentes los grados entre sí.

public static int[] MultPol3(int[] a, int[] b)

{

int[] result = new int[a.Length + b.Length - 1];

for (int i = 0; i < a.Length; i++)

for (int j = 0; j < b.Length; j++)

result[i + j] = a[i] \* b[j];

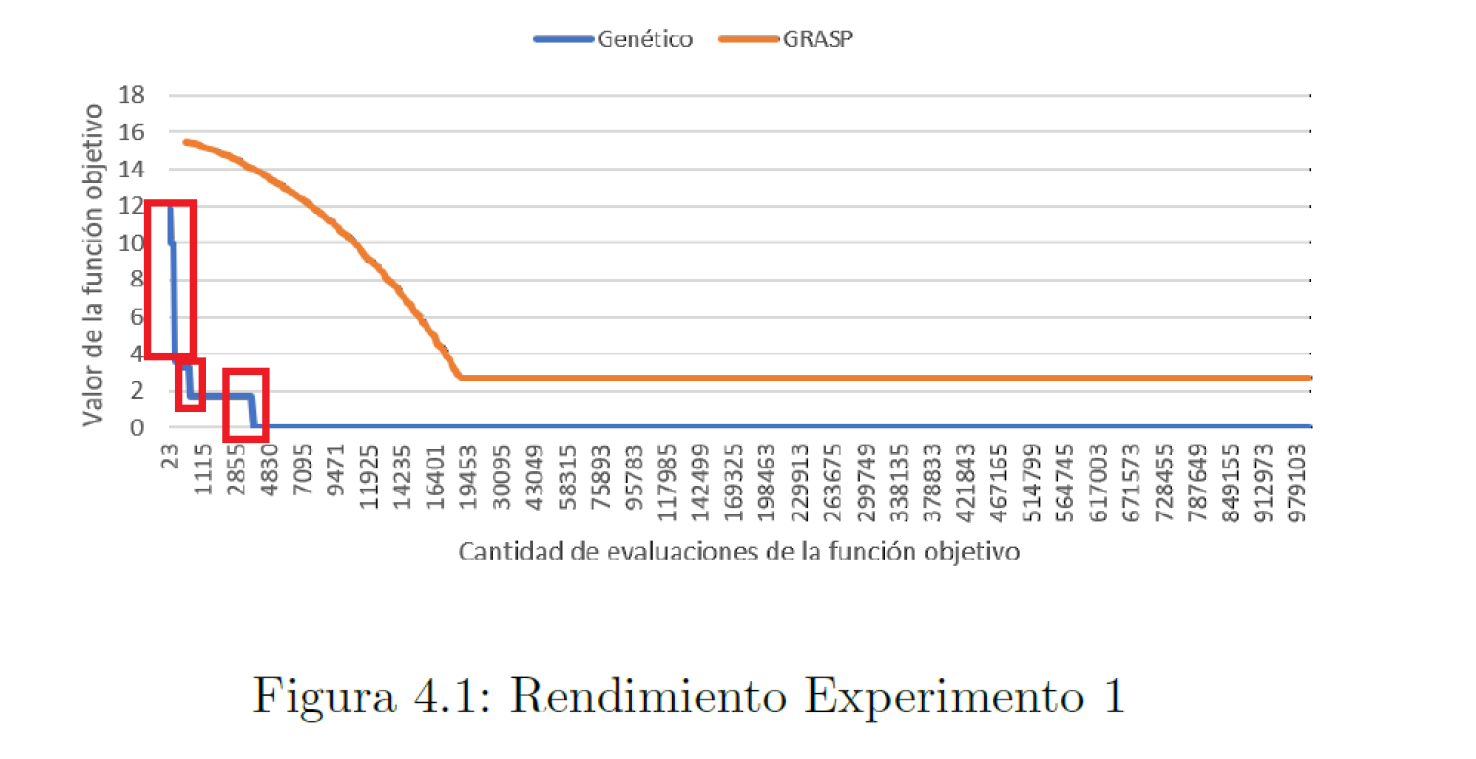
return result;

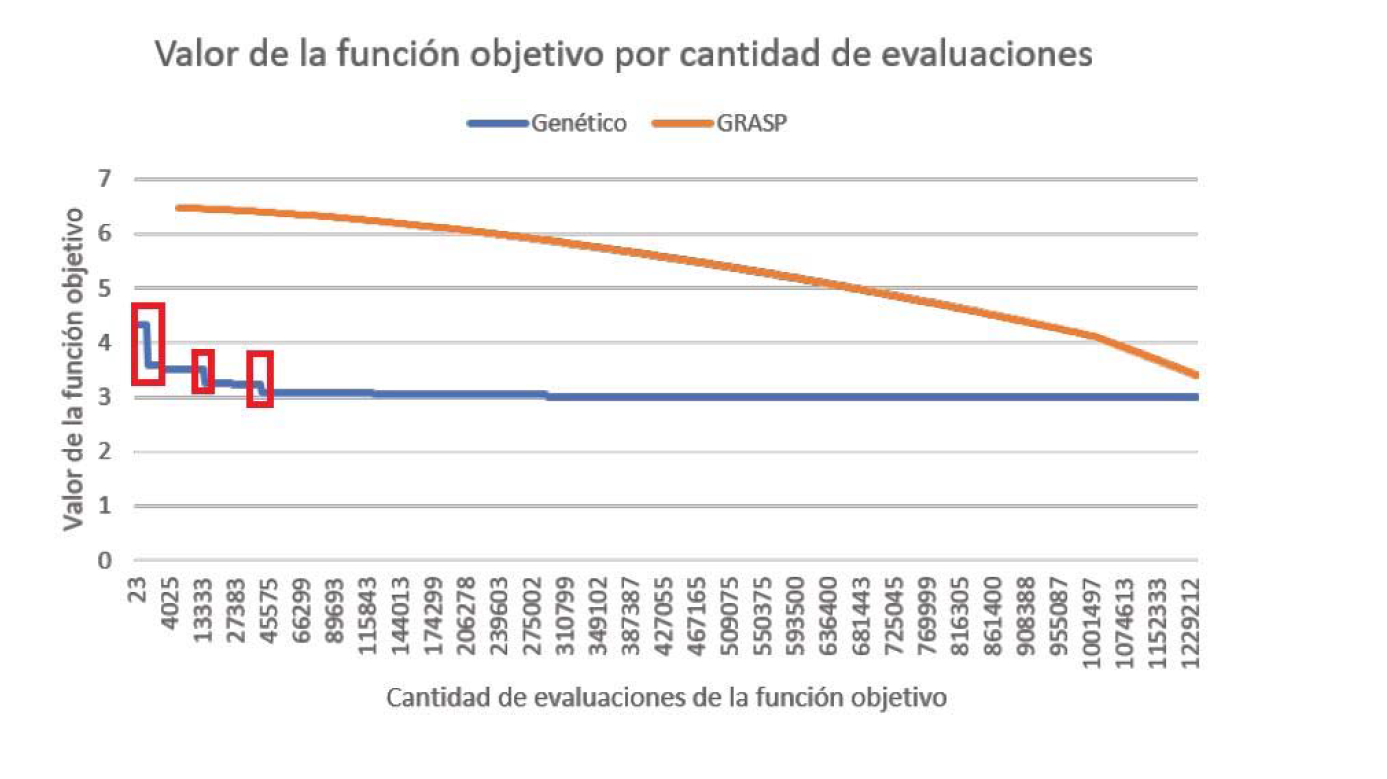
}

Entonces, dado que los casos que acierta el 3 son muy pocos si se recibe una muestra representativa de entrada, es necesario definir un bajo porciento de acierto para que los algoritmos tengan un bajo margen de error. Lo que pasaría si se define un rango entre [60, 70] para la nota de 3, está en dependencia de la entrada. Si es lo suficientemente grande se espera que los algoritmos logren reducir de manera aceptable la función objetivo, de lo contrario lanzarán márgenes de error muy grandes como sucedió en las experimentaciones iniciales donde se definían porcientos similares.

Preguntas del tribunal

Pregunta 1:

Cómo se muestra en los recuadros rojos en las imágenes mostradas a continuación de este párrafo, para el algoritmo genético existen momentos de una mejora rápida y pronunciada en la evaluación de la función objetivo. Muestre el contexto de la ejecución del algoritmo en esos momentos y haga un análisis de que rasgos del algoritmo genético utilizado están detrás de esas rápidas mejoras. 



El contexto de la ejecución del algoritmo en esos momentos es imposible de mostrar ahora, solo se conserva el conjunto de casos iniciales pasado de entrada y los resultados ya expuestos, pero el contexto es producto de la aleatoriedad presente en la metaheurística en un momento dado, una nueva corrida no lanzará el mismo escenario.

El porqué de ese ese comportamiento en las curvas está asociado a algunos rasgos del algoritmo genético, pero no se puede dejar de mencionar a las características del contexto.

**Algoritmo genético.** Tiene presente una gran exploración, superior a la explotación, la cual se evidencia en los resultados que arroja al lograr una buena reducción de la función objetivo.

**Contexto.** Como ambos son problemas sencillos, el espacio de búsqueda es pequeño. Tal es el caso que el algoritmo genético lo pone en evidencia al lograr en las “primeras” iteraciones bruscas reducciones de la función objetivo y una rápida convergencia al óptimo.

**GRASP** no logra esta curva por momentos pronunciada pues solo explota lo que recibe como entrada. Además, da pasos de tamaño 1 hacia una solución vecina, por lo que se demora más en converger.

**Notas añadidas**

**Nota 1:**

En el título del trabajo aparece el término aprendizaje, pero no se evidencia una etapa de entrenamiento o que aproveche las experiencias pasadas. ¿Acaso se entiende por aprendizaje a las decisiones que toman las heurísticas al recorrer el espacio o se refiere a que se utilizaron datos reales como base para generar los casos de prueba?

**Respuesta 1:**

El término aprendizaje hace referencia al proceso que tiene que ver con la generación de casos de prueba que es dependiente del problema. Un ejemplo de ello fue el ajuste de 30% esperado por la nota 3 para un mejor funcionamiento de las metaheurísticas. También alude al uso de casos reales, como los usados en Anagramas, para una mejor generación. Se prueba con un conjunto inicial de casos generados, el cual se va modificando hasta estar bien representado el universo de los mismos.

**Nota 2:**

¿Por qué la necesidad de explorar el espacio de esta forma? ¿Se puede resolver el problema de manera determinista mediante reducciones del conjunto inicial intentando ajustar el porcentaje deseado? ¿Sucedería igual si se tiene más de una implementación correcta por calificación?

**Respuesta 2:**

GRASP resuelve de manera determinista el problema mediante reducciones del conjunto inicial intentando ajustar el porcentaje deseado. Determinista en cuanto a valor final de la función objetivo y “tipos” de casos de prueba resultantes en cada conjunto solución. Esto fue apreciado en varias corridas sobre el mismo conjunto de datos.

La necesidad de explorar el espacio de manera diferente surge para lograr mejores resultados, es ahí donde entra el algoritmo genético con un alto índice de exploración.

Al tener más de una implementación por nota, se espera (con la debida contemplación en el código) obtener mejores resultados, pues se tendría más información de cada caso de prueba. Solo se cuenta con si acertó o no. En lugar de un vector binario de longitud 2, se tendría un valor continuo en el intervalo [0, 1] para cada implementación. Esto permite una mejor diferenciación entre casos y se podrían lograr mejores conjuntos finales, incluso en menos iteraciones.

---------------------

**Trabajando en:**

Con respecto al informe, señalar que debería haberse consultado bibliografía más reciente. Solo existe una con menos de 5 años de antigüedad. También añadir referencias bibliográficas donde se mencionen algoritmos, heurísticas, evaluadores y no se explique cómo funcionen.

El estado del arte abarca los problemas de Optimización Combinatoria y heurísticas, añadir algo más sobre la evaluación automática de algoritmos que es el problema que se intenta resolver a través de la selección automática de casos.