Práctico 1

Gonzalo Torterolo
Facultad de ingeniería
UDELAR
Montevideo, Uruguay
Email: gonzalo.torterolo@fing.edu.uy

Gisel Cincunegui
Facultad de ingeniería
UDELAR
Montevideo, Uruguay
Email: gisel.cincunegui@fing.edu.uy

***Abstract*—En el presente informe el objetivo es probar las ventajas y limitaciones de las actuales librerias que implementan algoritmos genéticos mediante la resolución de 2 problemas típicos de computación. Además se realizan pequeos analisis de las soluciones con el fin de verter conocimientos teóricos adquiridos en el curso.**

***Index Terms*—Algoritmos Evolutivos, AE, Algoritmos Genéticos, AG, Malva, Mallba, Whatchmaker, Knapshak, Mochila**

I. Introducción

En la consigna presentada (ver letra en [1]) se deben resolver dos variantes del problema de la mochila (explicadas en la consigna también) utilizando ideas de los algoritmos genéticos.

Por otra parte, el análisis y modelado del problema así como las ventajas de utilizar este tipo de algorimos para el escenario queda relegado a una segunda instancia, pues ya estaban resueltos en las consignas. En todo momento primó la aplicación de los conceptos teóricos del curso antes que la obtención de resultados como rendimiento, modularidad, etc. No nos interesa, por lo tanto, seguir aquí un enfoque práctico para esta primer aproximación a los algoritmos evolutivos donde ambos problemas son típicos y ampliamente estudiados.

A. Aplicación de AG

Se analizan las diferentes soluciones a las dos variantes del problema de la mochila aquí presentadas, concluyendo con una implementación para ambos casos. Para comenzar a definir las soluciones a los problemas se debe determinar las siguientes carácterízticas de importancia para un AE típico.

1) Codificación.
   - Tratamiento de codificaciones no factibles.
2) Población inicial.
   - Evaluación de coste/beneficio de utilizar criterios inteligentes de inicialización.
   - Velocidad de convergencia a un óptimo.
3) Funcion de fitness.
   - Carácterízticas especiales requeridas (e.g.: no negatividad de la función para seleccíon proporcional)
   - Transformación para otros escenarios (e.g.: aplicación para maximización/minimización)
   - Ajustes para mejorar soluciones (escalado, parámetros de configuración).
4) Seleccción:
   - Elitismo o no.
5) Cruzamiento.
   - Definir operadores de cruzamiento y estudiar la deriva genética que estos puedan provocar.
6) Mutación.
   - Definir operadores de mutación.

B. Particularidades y observaciones de las implementación/Librerias

- Malva
  - Malva utiliza otro concepto para la mutación, la probabilidad se aplica por individuo y no por gen.
  - Parece bastante desprolija, aunque quizás mucho más performante y escalable que cualquier otra, pero como ya se dijo, estas carácterízticas no interesan en este momento.
- Whatchmaker
  - Permite elitismo y esta cantidad es extra a la de la población especificada.
  - Es bastante menos performante que Malva, pero consisten en una estructura mucho más ordenada y flexible, por la que la preferimos a la hora de prototipar nuestra implementación.

C. Problemas de los aperitivos

Para este primer problema interesa obtener el óptimo de un problema. La aplicación de AE no nos parecío muy adecuada, pues no se estaba buscando una solución aceptable, sino que solo la mejor solución posible es aceptable para los comensales. Por otra parte, el problema es multimodal y NP computacionalmente dificil, lo que lo hace atractivo a una solución de AG. En esta variante del problema de la mochila se buscan soluciones enteras pero se aceptan repeticiones, y no existen pesos, o puede entenderse que el peso es proporcional a la ganancia o costo (es el mismo). Se utilizó la libería Malva en este problema. Se definen las caracterizticas de la solución relativas al AG:

Notación

$$X_i, C_i$$

son la cantidad de aperitivos, costo de aperitivo para el tipo i respectivamente

$$N, O$$

la dimensión del problema y costo objetivo respectivamente

1) Codificación. Nos es importante utilizar una codificación conveniente a para facilitar el trabajo del cruzamiento y utilizar SPX o cruzamiento uniforme que ya se encuentran implementadas.
2) Evaluamos la posibilidad de utilizar 2 tipos de codificaciones. La primera de ella muy parecida a la pedida en el próximo problema, donde asignamos un gen binario (los alelos toman valores en {0,1}) para cada posible aperitivo en caso de que se elija o no. En el caso deberíamos asignar una cota a la cantidad del genoma, dada por **??** para cada tipo de

aperitivo. Esta codificación es bastante intuitiva pero tiene la desventaja de asignar varios genotipos a un mismo fenotipo. El cruzamiento SPX tiene un comportamiento menos discruptivo en esta representación. Podría ser útil en etapas avanzadas de la evolución.

$$M_i = \lceil \frac{N}{C_i} \rceil \tag{1}$$

Otra codificación elimina el problema de que varios genotipos se correspondan con un mismo fenotipo. La misma consite en una representación de enteros, donde cada gen representa la cantidad de aperitivos de un tipo dado. Aplicada junto a un cruzamiento SPX esta codificación tiene un comportamiento más discruptivo que la anterior, se mueve por bloques.
Se puede incluso, utilizar ambas representaciones en distintas etapas de la evoluciones, de hecho propondremos una función de fitness que es compatible para ambas codificaciones.
En las codificaciones anteriores se le puede agregar semántica en el ordenamiento del cromosoma. Se discute la posibilidad de ordenar los genes dentro del cromosoma en función del costo. Para la implementación se elige la segunda opción, sin agregar información en el ordenamiento.
Todas las soluciones representables serán consideraciones factibles.

3) Selección Usaremos selección proporcional tanto para la selección de cruzamiento como selección generacional, ya implementada en malva utilizando el algoritmos de la ruleta. La función de fitness toma valores positivos, por lo que es adecuada para este tipo de selección.
4) Fitness Tomaremos como función de fitness la distancia al valor objetivo. Podemos configurar la librería Malva para resolver un problema de minimización en vez de maximización, por lo que no hace falta transformar la función de fitness.

$$v = |\sum_{i=0}^{N}(C_i X_i) - O|$$

es la distancia del valor objetivo.
5) Cruzamiento Se ha enfocado la elección de las anteriores caracterízticas de AG para utilizar SPX, pero otros algoritmos de cruzamientos donde se apliquen operadores numéricos podría surgir, como por ejemplo, se propone el promedio entre valores de cada padre, o diferentes promedios ponderados. Sin embargo, para la implementación se opta por SPX.
6) Mutación Se utilizó una mutación uniforme sobre los valores posibles de alelo de cada gen, expresada por la formula:

$$X_i = (X_i + rand(0, M_i))\%M_i$$

Población inicial La población inicial será muy importante para determinar una buena solución, pues a partir de ella se tiene casi todo el material genético nuevo. En las ejecuciones donde se ha probado la probabilidad de mutación baja hace que no se genere nuevo material genético más del aportado por la población inicial. Sin embargo, una cantidad suficiente de individuos en cada generación logra cubrir el espacio de busqueda bastante bien. Para la generación de la población inicial se utilizó un criterio aleatorio, pero se asegura que existen al menos un aperitivo de cada tipo al asignar el máximo valor factible a en algún gen de cada individuo (el gen numero de individuo % cantidad de genes del cromosoma).

D. Problemas de la mochila

- Descripcion del problema: El problema es un problema de optimización combinatoria perteneciente a la familia de problemas N-P difíciles. El problema se define de la siguiente manera: Dados un conjunto de n objetos, cada uno con una ganancia asociada gi y un peso asociado pi, el objetivo del problema es encontrar el subconjunto de objetos que maximiza la ganancia total, manteniendo el peso total por debajo de la capacidad máxima de la mochila (W). En ecuaciones ver 2 y 3:

$$G_{tot} = \sum_{i=0}^{n} g_i x_i \tag{2}$$

$$P_{tot} = \sum_{i=0}^{n} p_i x_i < W \tag{3}$$

- Descripción de la solución: Este problema constiste en obtener una solución lo mas próxima posible a la óptima del problema, implementando para ello técnicas de algoritmos evolutivos. A continuación se detallará las desiciones para la implementación del algoritmo evolutivo del problema de la mochila:

1) Codificacion:
   - Arreglo binario con el largo igual a la cantidad de elementos totales que puedo incorporar en la mochila, donde cada posicion del arreglo correspone a el objeto i de la solución. Dicho arreglo será ordenado por peso, manteniendo la referencia del ordenamiento del arreglo original, ya que se busca con esto que la operacion de cruzamiento operada por el algoritmo evolutivo intente probar distintos valores de ganancia "manteniendo" relativamente el peso de la solucion mas factible. El odenamiento de el arreglo se hará manteniendo dos estructuras de indices para la codificacion (arreglo original -¿ arreglo ordenado , fenotipo -¿ genotipo) y la decodificacion (arreglo ordenado -¿ arreglo original, genotipo -¿ fenotipo). La función de fitness se evaluará con el fenotipo, las operaciones se realizaran con el genotipo, y el resultado retornará el fenotipo de la mejor solucion encontrada.
   - Factibilidad de la solucion:
     Se tomará como solucion factible aquella que no sobrepase el limite de peso de la mochila.
2) Población inicial:
   - Para la población inicial se optó por la generacion aleatoria de los individuos, pues podemos ver que en varias ejecuciones se alacanza rápidamente un óptimo del problemas.
3) Funcion de fitness:
   La ecuacion final de la misma ha mutado de acuerdo a problemas que se nos han planteado a lo largo de la implementacion. Al principio se optó por lo siguiente:
   - Funcion con penalizacion:
     La idea de esta función es mantener los individuos no factibles a lo largo de las generaciones para evitar la deriva genética, ya que los mimos pueden brindar informacion relevante para llegar a la optimizacion de la solucion final. Si el peso total del individo no sobrepasa al maxímo permitido para la mochila(candidato factible), la función de fitness retornará la ganancia total del individo evaluado. Si el peso sobrepasa el máximo estipulado (candidato no factible), la ganancia total se penalizará restando la

ganancia proporcional a la diferencia de peso, quedando a formúla como la siguiente:

$$fitness(x) = \begin{cases} G_{tot}, & \text{for } P_{tot} \leq k \quad (4a) \\ G_{tot} - (\frac{2G_{tot}(k-p)}{k}), & \text{for } P_{tot} > k \quad (4b) \end{cases}$$

- Funcion sin penalizacion:
  Como la solución anterior presentó problemas con respecto a la solucion retornada por el algoritmo, debido a que se devolvian soluciones no factibles, se optó por solo maximizar la ganancia, e implementar otros mecanimos para cuando nos topamos con casos de no factibilidad que se detallará más adelante.

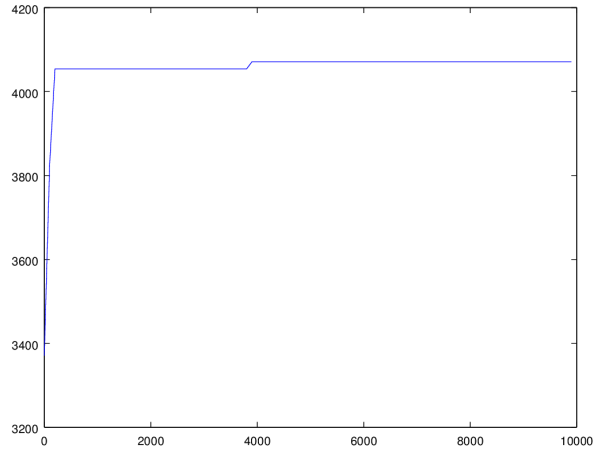
4) Selección

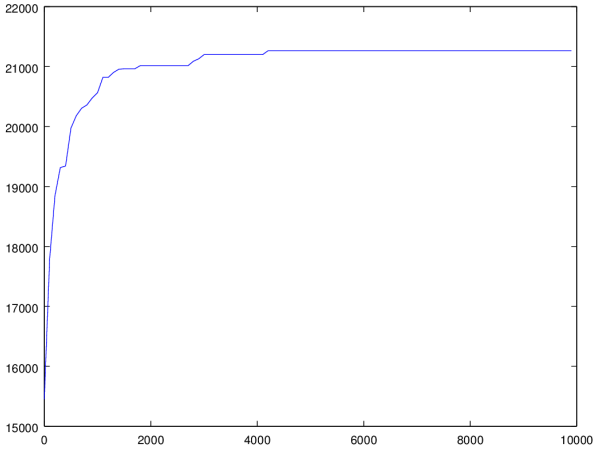
- Para la selección se optó utilizar el mecanismo de la selección proporcional implementada con el algoritmo de la ruleta, para poder mantener individuos menos factibles en la poblacion generacional y así evitar la deriva genetica.

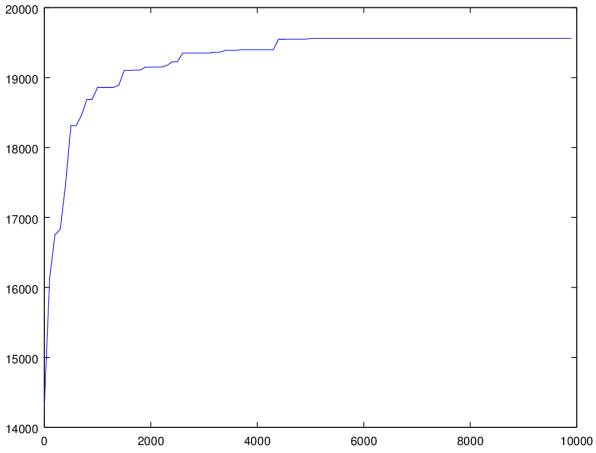
5) Cruzamiento

- SPX. Idema anterior.
- Cruzamiento con probabilidad 0.75
- Para la selección se optó utilizar el mecanismo de la selección proporcional implementada con el algoritmo de la ruleta, para poder mantener individuos menos factibles en la poblacion generacional y así evitar la deriva genetica.
- Para evitar el problema de obtener individuos no factibles, lo que se realizó fue que luego de aplicar la operacion de cruzamiento se evalúa la factibilidad de los mismos, si estos no son factibles se realizará una mutación a el objeto que tenga menos peso, con valor 1 en la codificacion. (otra opcion seria mutar algun objeto aleatoriamente o a aquel que posea menos ganacia), y aplicar esto hasta conseguir la factibilidad del candidato. Asi, logramos no tener candidatos no factibles, y por lo tanto no tener soluciones no factibles.

6) Mutación. Mutación de bit aleatorio con probabilidad 0.01.
7) Condición de parada: 10000 generaciones.

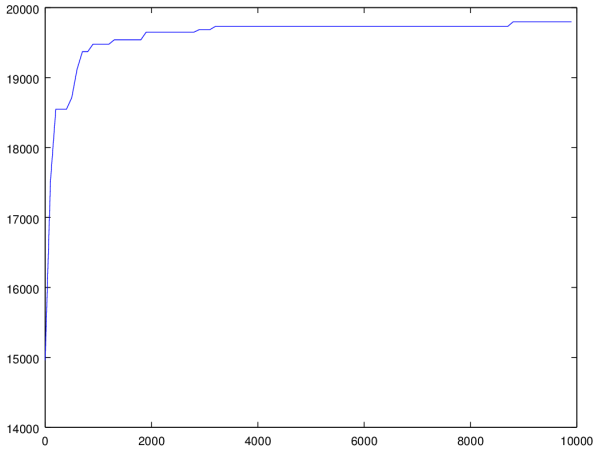


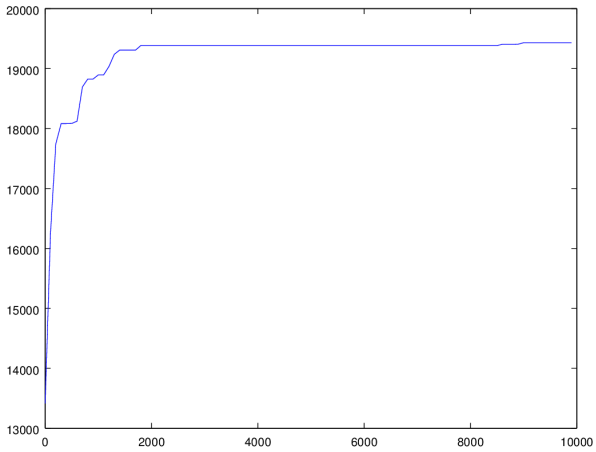




II. Ejecuciones

Se muestran algunas gráficas de cantidad de generaciones VS fitness sobre las ejecuciones del algoritmo de la mochila para distintas semillas.





III. Conclusión

Al final de este laboratorio hemos podido decicir aquella librería que se adapta mejor a nuestras necesidades y que nos gustaría usar para resolver el proyecto final. Además, en el proceso de realización de este primer práctico, logramos interiorizarnos en aspectos prácticos y la aplicación de algoritmos genéticos y la síntesis de informes con Latex y otras herramientas.

References

[1] U. de Málaga. (2008) Referencia mallba/malva. [Online]. Available: http://neo.lcc.uma.es/mallba/easy-mallba/index.html