Tarea 4: Algoritmos genéticos

Fernando Carrillo A01194204

28/09/20

Descripción del problema:

Utilizando 3 distintas variaciones de algoritmos genéticos se intentará generar soluciones óptimas al problema de la mochila. El problema de la mochila plantea la situación donde se tiene una mochila que puede soportar un peso máximo de W=165, y existen varios objetos con distintos pesos 'w' y profit 'p'. La meta es meter todos los objetos posibles a la mochila de tal manera que se maximice el valor y no se sobrepase el límite de peso W.

Descripción de la representación de los individuos:

Cada solución será representada como un arreglo de bits con un tamaño igual a la cantidad de objetos disponibles. Cada bit corresponde a un índice que indica si un objeto será o no incluido en la mochila.
Ej. [1, 0, 0, 1, 1, 0], indica que los objetos 1, 3, y 4 de la lista de objetos disponibles serán incluidos en la mochila.

Descripción de la función de evaluación:

La función de evaluación recibe la lista de objetos a incluir en la mochila, calcula el peso total de los objetos incluidos, y su profit. Si el peso total sobrepasa el límite W=165, se genera un castigo al profit. Este castigo reduce el valor total, indicando que la solución no es deseable.

Describir las diferencias entre los 3 algoritmos:

eaSimple: las nuevas generaciones están compuestas de cruces de los padres y en algunos casos se hacen mutaciones sobre estas. Todos los padres son reemplazados por los hijos ya que existe una proporción de 1:1.

eaMuPlusLambda: se produce una cantidad *Lambda* de hijos en cada generación, con una parte siendo creada por cruces entre padres, y otra por mutaciones. Después se seleccionan *Mu* individuos dentro de los padres e hijos disponibles.

eaMuCommaLambda: se produce una cantidad *Lambda* de hijos en cada generación, con una parte siendo creada por cruces entre padres, y otra por mutaciones. Después se seleccionan *Mu* individuos solamente dentro de los hijos disponibles.

Resultados obtenidos:

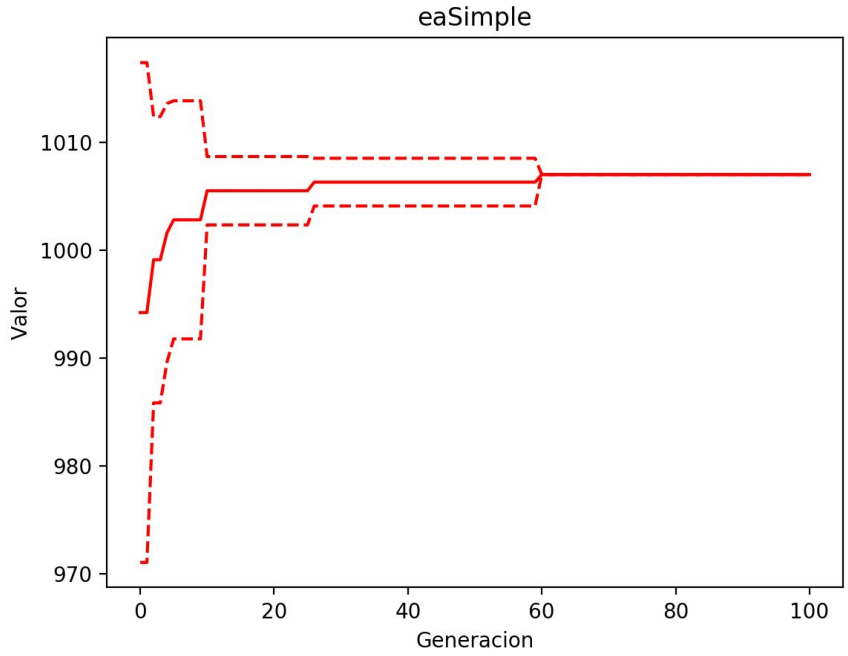
Los algoritmos fueron ejecutados con los siguiente datos:

id	w	p
1	23	92
2	31	57
3	29	49
4	44	68
5	53	60
6	38	43
7	63	67
8	85	847
9	89	87
10	82	72

Cada variación de los algoritmos genéticos será ejecutado 10 veces, con 100 generaciones en cada iteración.

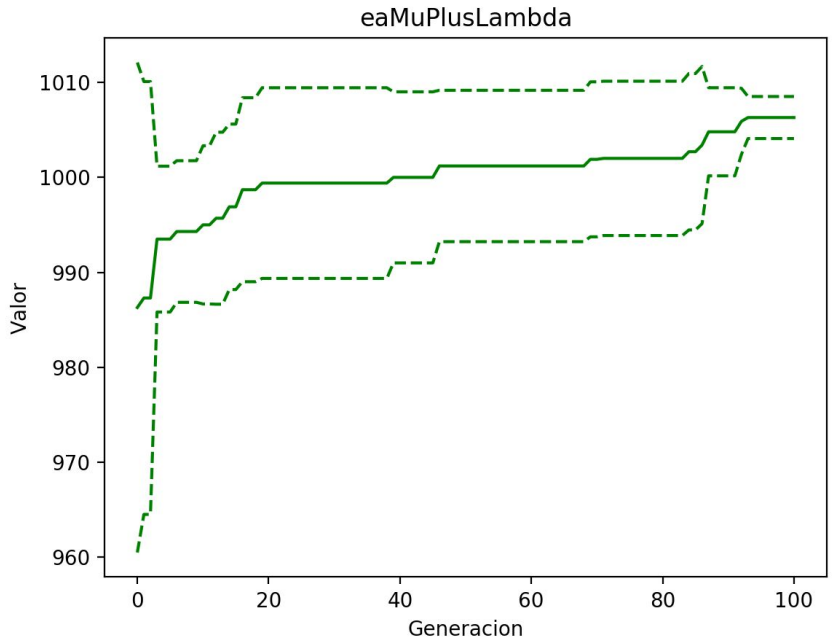
eaSimple

Attributo	Valor
cxpb	1.0
mutpb	0.5
ngen	100



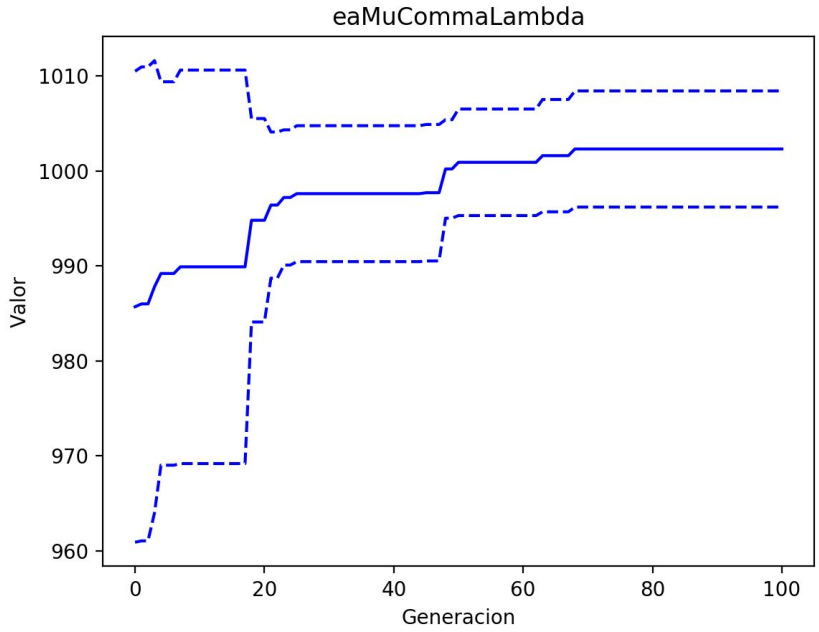
eaMuPlusLambda

Attributo	Valor
mu	10
lambda	20
cxpb	0.7
mutpb	0.3
ngen	100



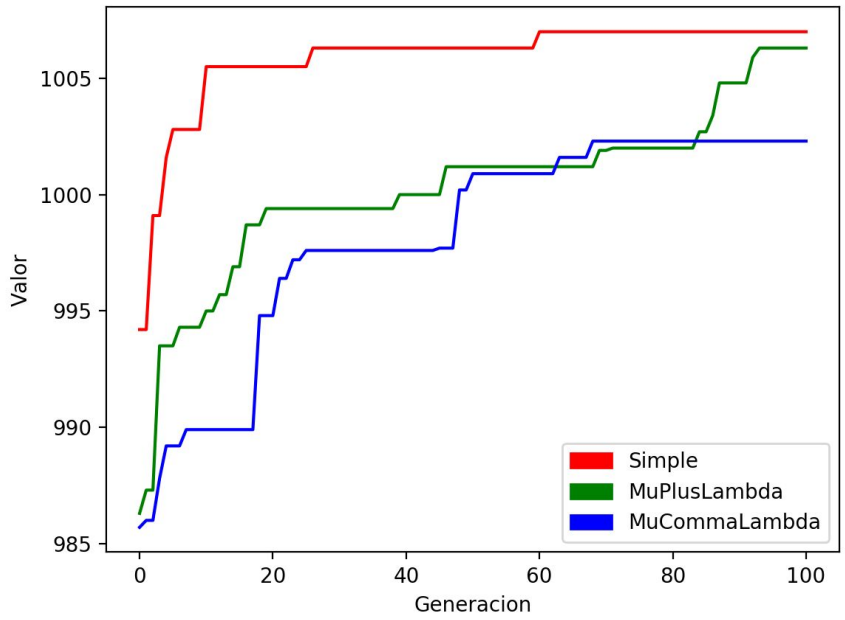
eaMuCommaLambda

Attributo	Valor
mu	10
lambda	20
cxpb	0.7
mutpb	0.3
ngen	100



Como se puede observar en la gráficas con la curva del mejor encontrado, el algoritmo *eaSimple* es el primero en converger, seguido por el *eaMuPlusLambda*, y por último el *eaMuCommaLambda*.
Es posible que esto se deba a que en el algoritmo *eaSimple* se estén actualizando las generaciones utilizando cruces siempre entre los mismos padres base con algunas mutaciones, mientras que en el *eaMuCommaLambda*, solamente se tomen en consideración los hijos y mutaciones nuevas, disminuyendo la posibilidad de incluir individuos padres que puedan tener una mejor solución y siempre teniendo variaciones diferentes a los padres anteriores. Esto también podría explicar la diferencia de desviación estándar entre *eaSimple* y *eaMuCommaLambda*.
En este problema el algoritmo más eficaz fue el *eaSimple* pero puede que en un problema con un espacio de búsqueda mayor, este algoritmo sea muy rápido en encontrar converger y no llegar a la solución óptima.

Gráfica con la curva de mejor encontrado con los 3 algoritmos:



En todas las corridas los tres algoritmos llegaron a la misma solución óptima:

- *Best for Simple: [[1, 0, 0, 1, 0, 0, 0, 1, 0, 0]]*
  Weight: 152 Profit: 1007
- *Best for MuPlusLambda: [[1, 0, 0, 1, 0, 0, 0, 1, 0, 0]]*
  Weight: 152 Profit: 1007
- *Best for MuCommaLambda: [[1, 0, 0, 1, 0, 0, 0, 1, 0, 0]]*
  Weight: 152 Profit: 1007

Conclusiones y retos encontrados:

Inicialmente se tenía planeado crear un algoritmo con optimización multivariable, donde se buscaría maximizar el profit y minimizar el peso. Sin embargo esta decisión de diseño resultó contra intuitiva ya que al tener que para minimizar el peso, el algoritmo podría simplemente eliminar cosas de la mochila, quitando objetos de valor y deteniendo a la función de maximización de alcanzar el máximo profit.

Por esta razón solamente se buscó maximizar el profit, y para descartar soluciones inválidas (peso > 165), se asignó un castigo a los individuos que sobrepasen este límite. El castigo fue reducir el profit total, por la diferencia entre el peso del individuo y el límite de peso multiplicado por 4.

Aún con este castigo, el algoritmo decidía que la mejor solución sobrepasaba el peso máximo:

- *Best for Simple: [[1, 1, 1, 0, 0, 0, 0, 1, 0, 0]]*
  Weight: 168 Profit: 1033
- *Best for MuPlusLambda: [[1, 1, 1, 0, 0, 0, 0, 1, 0, 0]]*

Weight: 168 Profit: 1033

- Best for MuCommaLambda: [[1, 1, 1, 0, 0, 0, 0, 1, 0, 0]]

Weight: 168 Profit: 1033

Se cambió el castigo a uno más drástico, donde la diferencia se multiplicaba por 13. Este castigo fue suficientemente fuerte para descartar soluciones inválidas, por lo que se produjeron las siguientes soluciones válidas:

- Best for Simple: [[1, 0, 0, 1, 0, 0, 0, 1, 0, 0]]

Weight: 152 Profit: 1007

- Best for MuPlusLambda: [[1, 0, 0, 1, 0, 0, 0, 1, 0, 0]]

Weight: 152 Profit: 1007

- Best for MuCommaLambda: [[1, 0, 0, 1, 0, 0, 0, 1, 0, 0]]

Weight: 152 Profit: 1007