Algoritmo genético

Oscar Quiñonez

25 de noviembre de 2020

1. Objetivo

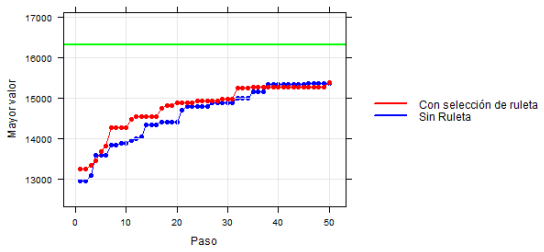
Se plantea un problema conocido como "el problema de la mochila" en el que se debe regular la cantidad de objetos que se colocan dentro de ella, debido a que tiene una capacidad limitada en cuestión de peso, en base a este problema, se busca realizar una simulación de un algoritmo genético el cual se varía la selección de los padres de la manera conocida como "ruleta" [1].

2. Metodología

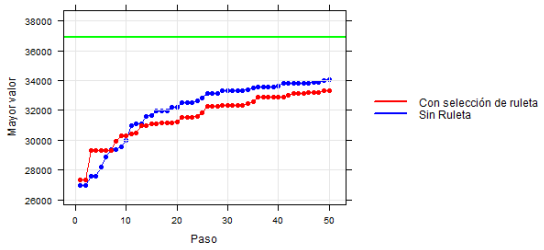
En esta simulación fue necesario el uso del programa Python 3.7 en el que se tomó como código base el proporcionado durante la clase [2] para posteriormente plantear 3 escenarios posibles: 1) el valor y el peso son independientes entre ellos, 2) el valor y el peso están relacionados entre ellos y 3) el peso es independiente pero el valor es inversamente proporcional a este. Al variar la selección de padres a tipo "ruleta", cada uno tenía una probabilidad de ser tomado en cuenta, y al tomar los 3 criterios anteriormente planteados, se puede determinar el valor del algoritmo genético y compararlo con el algoritmo exacto. Los valores de n usados en esta simulación fueron 100, 200, 300 y 400.

3. Resultados y Discusión

Al ejecutar el código con las variaciones mencionadas se obtuvieron primeramente las gráficas que nos indican la diferencia de los valores al usar o no usar la ruleta, estas gráficas se pueden apreciar en la figura 1 que se presenta a continuación. En el cuadro1 se muestra los valores obtenidos al variar la cantidad de objetos y se destacan los valores máximos y óptimos de cada uno, de manera mas representativa se pueden ver las gráficas de estos valores y como varían entre ellos en las figuras 2, 3, 4 y 5.



(a) 100 objetos

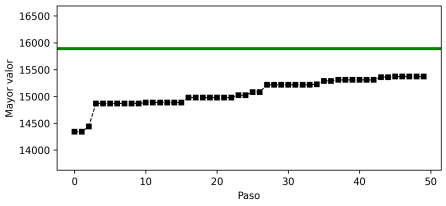


(b) 200 objetos

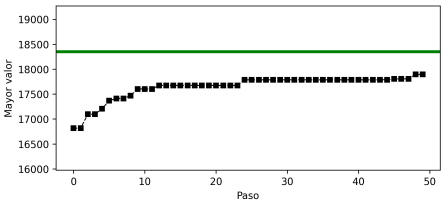
Figura 1: Comparación de la ruleta con 100 y 200 objetos

Cuadro 1: Tabla de resultados para las distintaas cantidades de n.

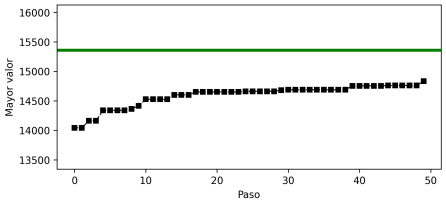
Objetos	Valor máximo	Valor óptimo	Instancia
100	7326.288381842508	75851.55212057414	1
100	68468.98351482986	55795.300566274185	2
100	51964.798682299166	43479.15872730447	3
200	15368.983690630148	13571.3126008619583	1
200	17894.877000934695	8721.51234211876	2
200	14832.970082181477	12135.028209092558	3
300	32763.83926134712	21985.93465812870	1
300	28754.157541097403	18963.883314069720	2
400	52301.06212532473	21369.348512678901	1
400	42034.773495068867	35991.030450982477	2
400	40782.74748446381	27004.364789015584	3



(a) Instancia 1

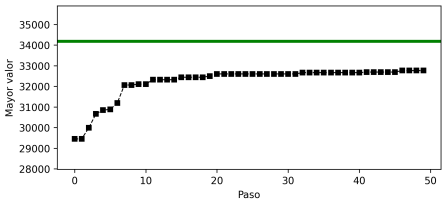


(b) Instancia 2

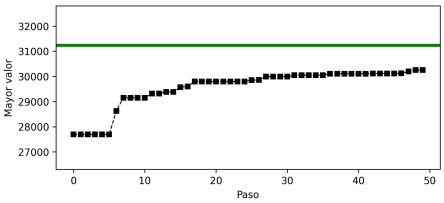


(c) Instancia 3

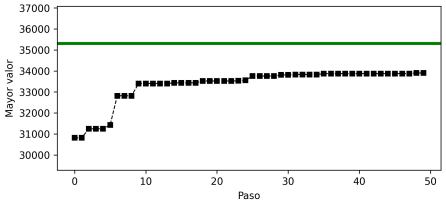
Figura 2: 3 instancias para n=100



(a) Instancia 1

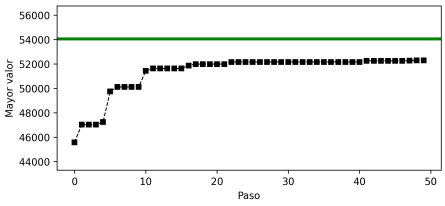


(b) Instancia 2

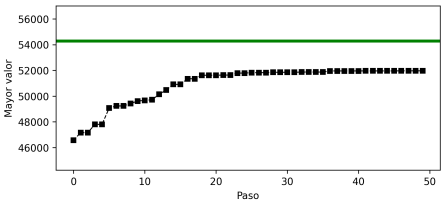


(c) Instancia 3

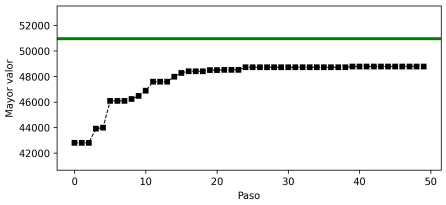
Figura 3: 3 instancias para n=200



(a) Instancia 1

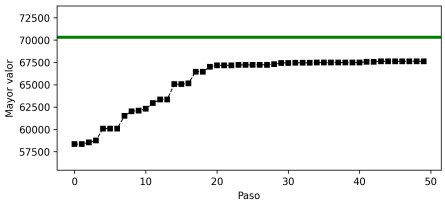


(b) Instancia 2

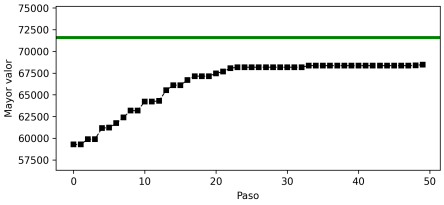


(c) Instancia 3

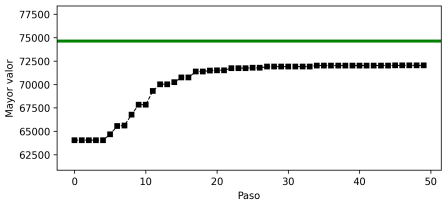
Figura 4: 3 instancias para n=300



(a) Instancia 1



(b) Instancia 2



(c) Instancia 3

Figura 5: 3 instancias para n=400

4. Conclusión

Como se puede ver en las gráficas, existe una línea verde la cual nos refleja el valor máximo de las 3 condiciones iniciales con respecto a las cantidades de n utilizadas en cada una de ellas [3]. Los valores óptimos serían los que tocan la línea verde en los valores de n indicados al pie de las gráficas, sin embargo, en esta simulación no se alcanzó este punto posiblemente por la cantidad de objetos o por la capacidad del CPU para procesar los datos, aunque se puede decir que las 3 instancias muestran resultados muy similares.

Referencias

[1] E. Schaffer. Algoritmo genético, 2020. URL `https://elisa.dyndns-web.com/teaching/comp/par/p10.html`.

[2] E. Schaffer. Algoritmo genético, 2020. URL `https://github.com/satuelisa/Simulation/blob/master/GeneticAlgorithm/basicGA.py`.

[3] O. Quiñonez. tareadiez, November 2020. URL `https://github.com/OscarNANO/OscarNANO/tree/master/tareadiez`.