Sartenejas, 2 de noviembre de 2009
Universidad Simón Bolívar
Inteligencia Artificial II - CI-5438
María Gabriela Valdes 05-39020
Juan García 05-38207
Sep – Dic 2009

Informe Tarea 2

1. Descripción de la Implementación:

Primeramente cabe destacar que la implementación fue desarrollada enteramente por nosotros, sin uso de alguna librería El lenguaje utilizado fue Python, ya que el mismo es fácil de usar y nos encontramos muy familiarizados con el. Se implementaron una clase y una serie de funciones, algunas directamente relacionadas con el desarrollo del algoritmo genético y otras que funcionaban como herramientas o apoyos a otras funciones.

La clase principal es “Hypotheses”, que representa una hipótesis en nuestro programa y se encuentra compuesta por una serie de campos: Un vector de 0's y 1's que modela el la cadena completa de cromosomas de una hipótesis, un campo entero que representa el largo de un cromosoma individual y un campo real que contiene el fitness de una hipótesis. Los métodos pertenecientes a la clase “Hypotheses” son los operadores genéticos, es decir, “crossover”, “mutation”, “addAlternative” y “dropCondition”, estos últimos encapsulados en la función “AA_DC”. Se implementaron dos tipos de Crossover, el de “gabilCrossover”, con selección de dos puntos para la recombinación y el “onePointCrossover” de un punto.

Como métodos principales, pero que no pertenecen a la clase “Hypotheses” se encuentran “selection” y “computeFitness”. Para la selección se implementaron tres versiones, selección por ruleta, selección por probabilidad y una versión que entrega siempre los mejores de cada población “computeFitness” se usa para calcular el fitness de cada individuo con respecto a nuestro set de entrenamiento.

Se debe realizar una acotación importante con respecto a la función que calcula el fitness. Debido al lenguaje que utilizamos para implementar la tarea, Python, las corridas resultaron ser, en tiempo de ejecución, muy lentas. En donde la función que se llevaba mas tiempo de procesamiento era la de calculo de fitness. Por lo tanto se tradujo dicha función a un lenguaje de mas bajo nivel, en este caso C++, que permitiría que corriera mas rápido Con el uso de herramientas como Swig, se importo la librería de dicha función en nuestro programa en Python, y se realizo dicho calculo con la función de fitness importada de C++. Los resultados y la mejora en tiempo obtenida fue mas que relevante, la ejecución fue mucho mas rápida y permitió realizar todas las pruebas necesarias para obtener los resultados.

Ademas de las funciones antes mencionadas, se utilizan métodos especiales que funcionan como apoyo, tales como las funciones que traducen los datos de entrenamiento a cadenas binarias. Funciones que convierten números decimales a binarios y viceversa, Funciones de comparación para

ordenar vectores, funciones para calcular los fitness y los ratios de los individuos para realizar la selección de ruleta y la función para transformar los conceptos continuos en categorías para ser representadas mas fácilmente por cadenas de bits.

2. Descripción del algoritmo genético y parámetros base que se utilizaron:

El algoritmo genético que implementamos esta basado en el sistema GABIL. Este sistema permite al algoritmo genético aprender conceptos booleanos representados como un conjunto de reglas proposicionales.

Lo primero que se hace es leer los datos del conjunto de entrenamiento y traducirlos al mismo formato de representación de las hipótesis Es decir, listas de valores 0's y 1's. Luego se genera una población aleatoria del tamaño fijo que se indique en la llamada a la función principal. Los individuos de dicha población son de tamaño variable; se escoge aleatoriamente; entre 1 y 10, el numero de reglas que tendrá cada individuo antes de su creación Luego se calcula el fitness de cada individuo de la población

El algoritmo se detiene cuando el individuo con mayor fitness en la población sobrepasa cierto umbral fijo. Este valor es pasado como parámetro en la llamada a la función principal. En cada iteración el algoritmo selecciona un porcentaje de la población actual, en este caso ((1 – crossover_rate) * population_length) individuos que pasaran a formar parte de la siguiente generación Luego se seleccionan (crossover_rate * population_length) individuos de la población actual que se cruzaran para dar como resultado (crossover_rate * population_length) nuevos individuos que junto con el primer conjunto seleccionado, constituirán la nueva población Antes de calcular los nuevos fitness de cada individuo se llama a la función de mutación que dependiendo del valor mutation_rate, selecciona un porcentaje de la población que sufrirá modificaciones en su estructura. Una vez finalizada la llamada a esta función se calcula el valor de fitness de cada individuo en le nueva población que se construyo.

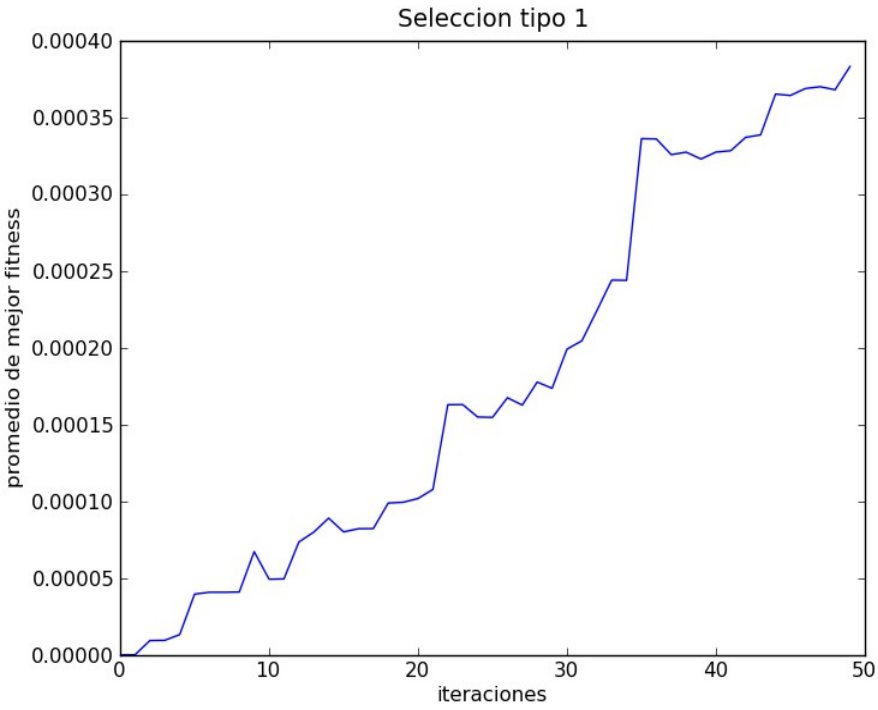
Los parámetros crossover_rate = 0.6 y mutation_rate = 0.001 fueron escogidos en base a los experimentos hechos por DeJong con poblaciones entre 100 y 1000 individuos.

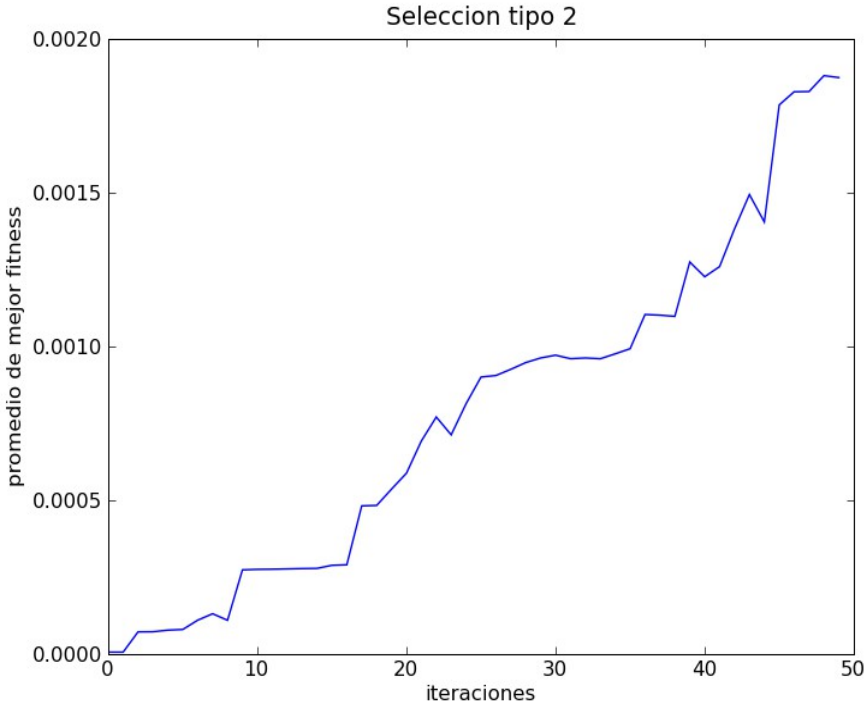
3. Descripción y análisis de los experimentos realizados:

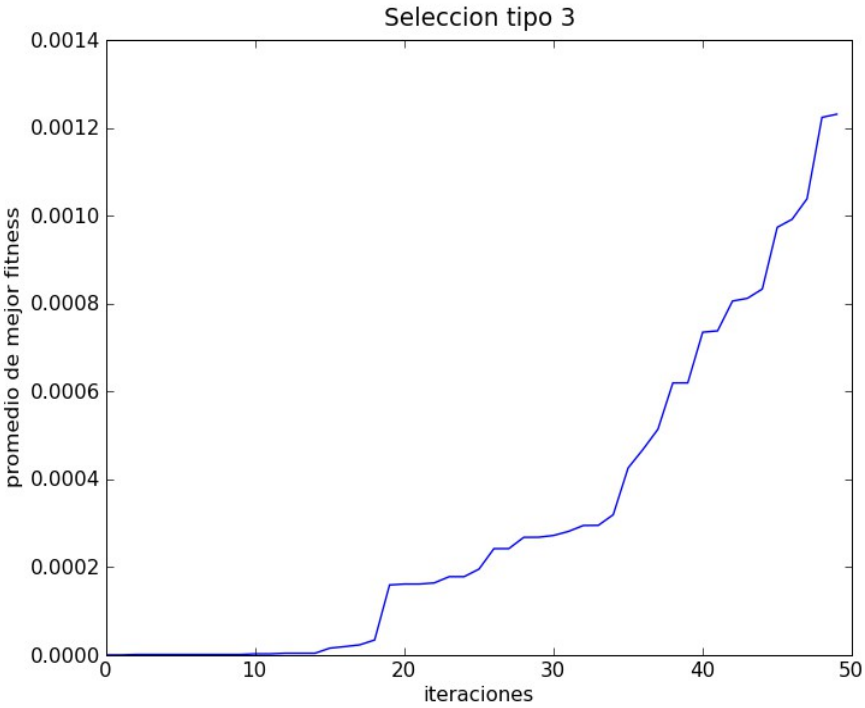
Todos los experimentos realizados fueron de 10 corridas del mismo algoritmo, con una población de 20 individuos generados aleatoriamente, con un numero de reglas inicial entre 1 y 5. Cada corrida terminaba trascurridas 50 iteraciones del algoritmo principal. Los resultados finales de cada configuración se sacaron en base al promedio de de los resultados de cada una de las 10 corridas.

Primero se realizaron 3 experimentos para escoger el mejor tipo de selección de entre los tres que se implementaron: rueda de ruleta, probabilidad o escogiendo los mejores de la población Los parámetros de crossover_rate y mutation_rate que se mantuvieron constantes e iguales para todos los grupos de corridas y sus valores fueron 0.6 y 0.001 respectivamente.

La selección tipo 1 se refiere a la selección “rueda de ruleta”, en la selección tipo 2, la escogencia se hace en base a una probabilidad. En este tipo de selección, la probabilidad de que cierto individuo sea seleccionado es proporcional a su fitness e inversamente proporcional al fitness del resto de las hipótesis en la población La selección tipo 3 escoge a los individuos con mejor fitness en la población


Seleccion tipo 1
promedio de mejor fitness
iteraciones
0.00040
0.00035
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
0
10
20
30
40
50

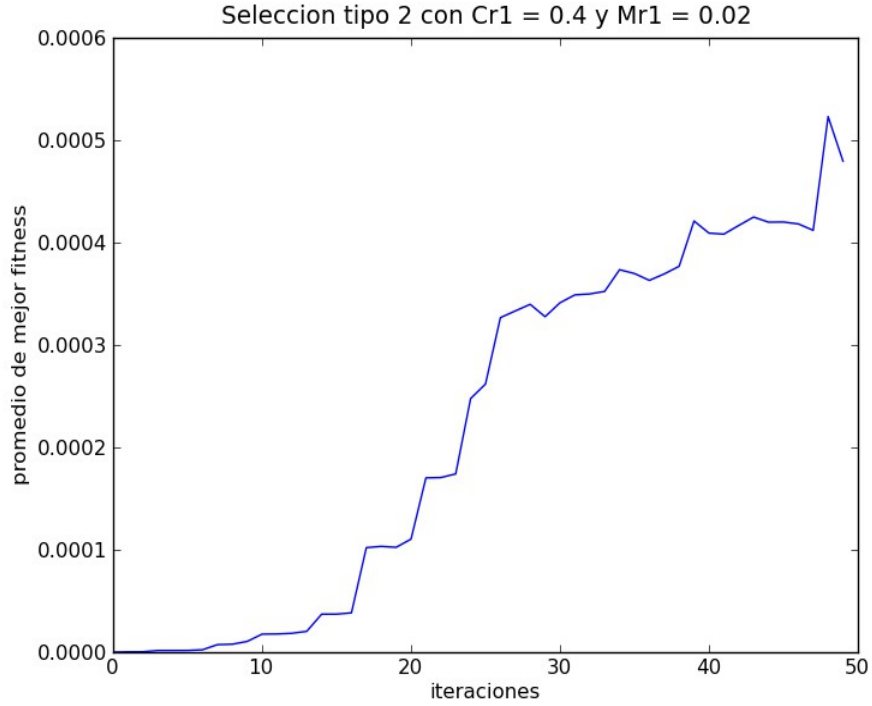

Seleccion tipo 2
promedio de mejor fitness
iteraciones
0.0020
0.0015
0.0010
0.0005
0.0000
0
10
20
30
40
50

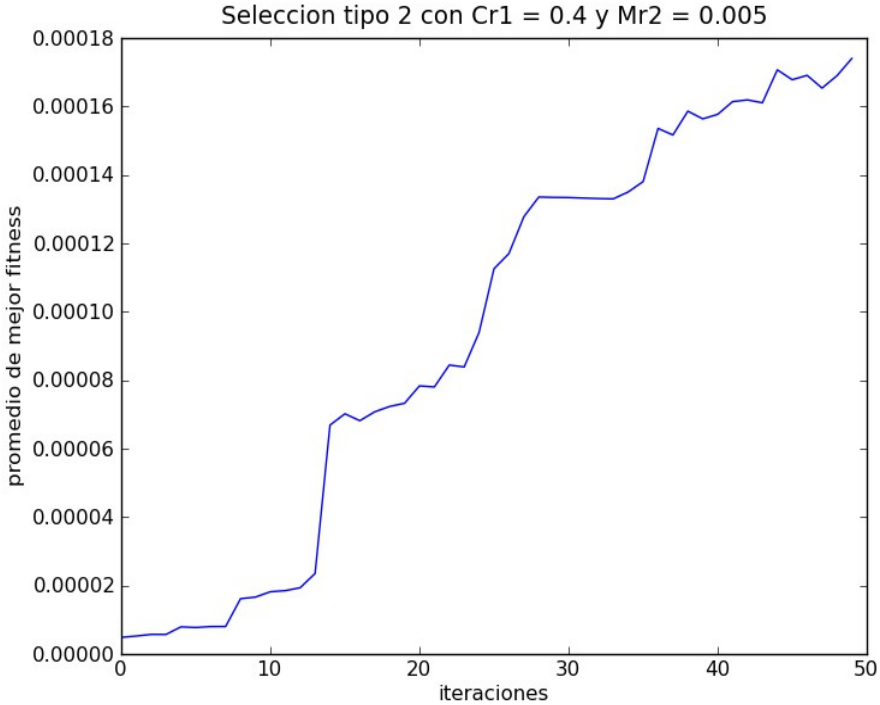


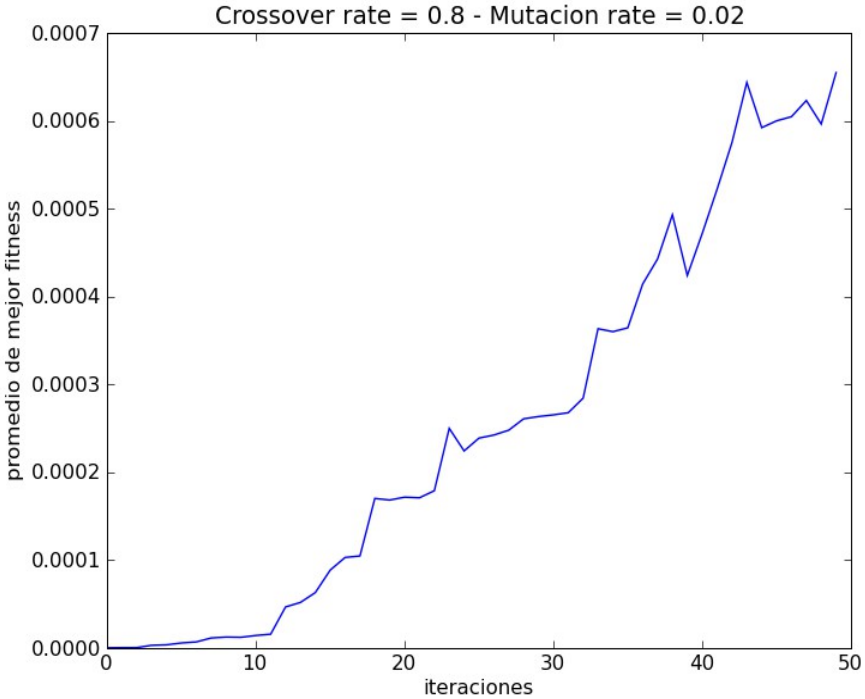
Según los resultados de los experimentos con cada tipo de selección, la selección por probabilidad fue la que arrojo mejores resultados. Dicha selección es la que se utilizara en los experimentos siguientes.

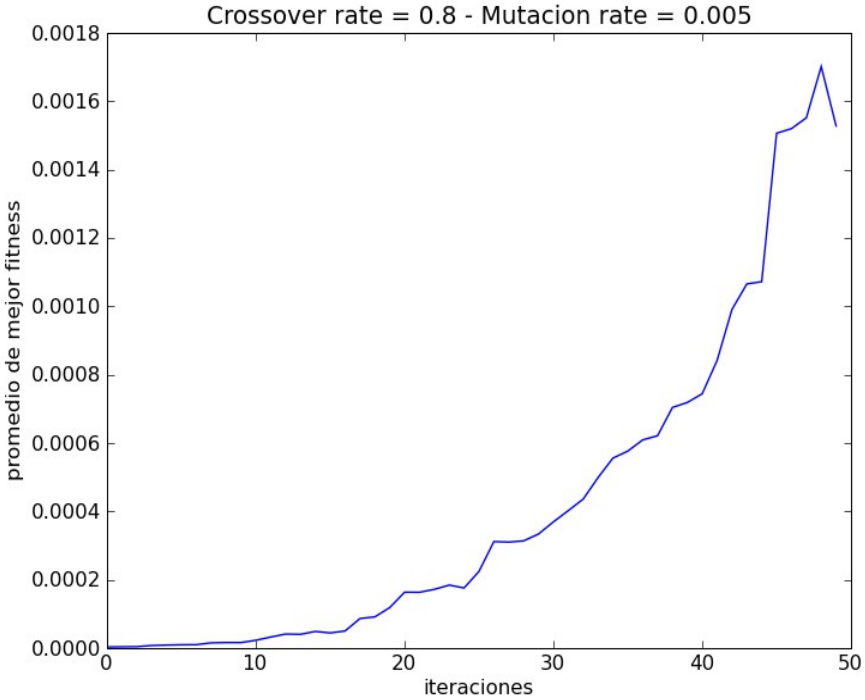
Luego se escogieron 2 posibles valores para el crossover_rate y para el mutation_rate. Los 4 experimentos siguientes se hicieron combinando todos estos posibles valores y utilizando la selección por probabilidad. Los valores de crossover_rate que escogimos fueron 0.8 y 0.4, y los de mutation_rate 0.02 y 0.005 respectivamente.

En las siguientes gráficas se muestran los resultados de las 4 posibles combinaciones de los valores antes mencionados.


Seleccion tipo 2 con Cr1 = 0.4 y Mr1 = 0.02
promedio de mejor fitness
iteraciones
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0
10
20
30
40
50

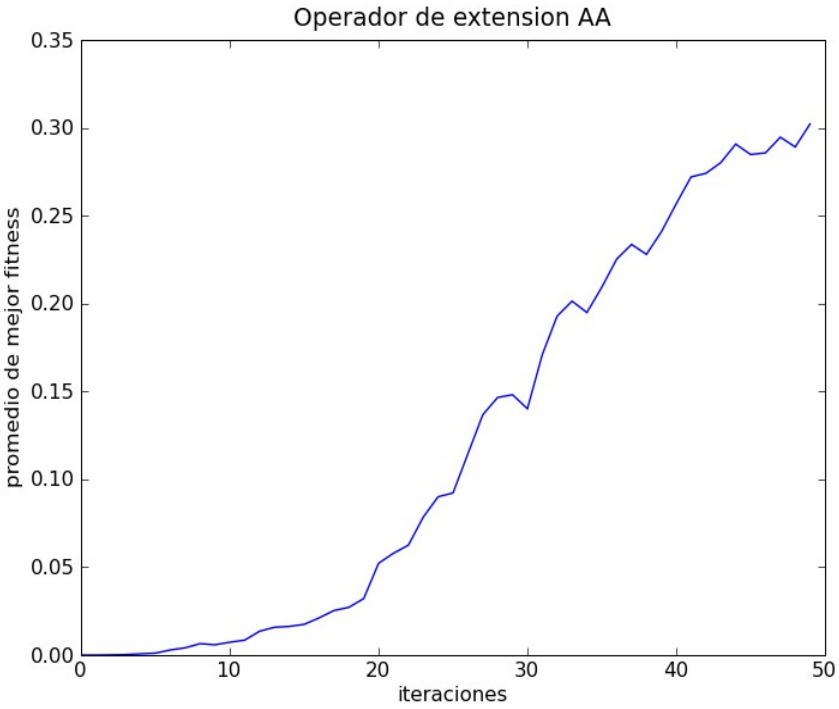

Seleccion tipo 2 con Cr1 = 0.4 y Mr2 = 0.005
promedio de mejor fitness
iteraciones
0.00000
0.00002
0.00004
0.00006
0.00008
0.00010
0.00012
0.00014
0.00016
0.00018
0
10
20
30
40
50

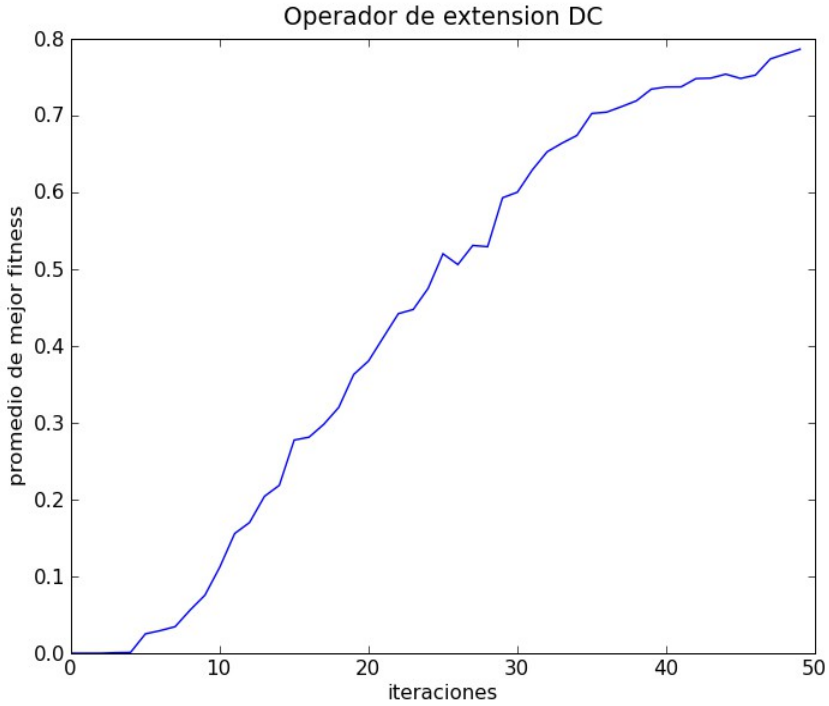

Crossover rate = 0.8 - Mutacion rate = 0.02
promedio de mejor fitness
iteraciones
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0
10
20
30
40
50

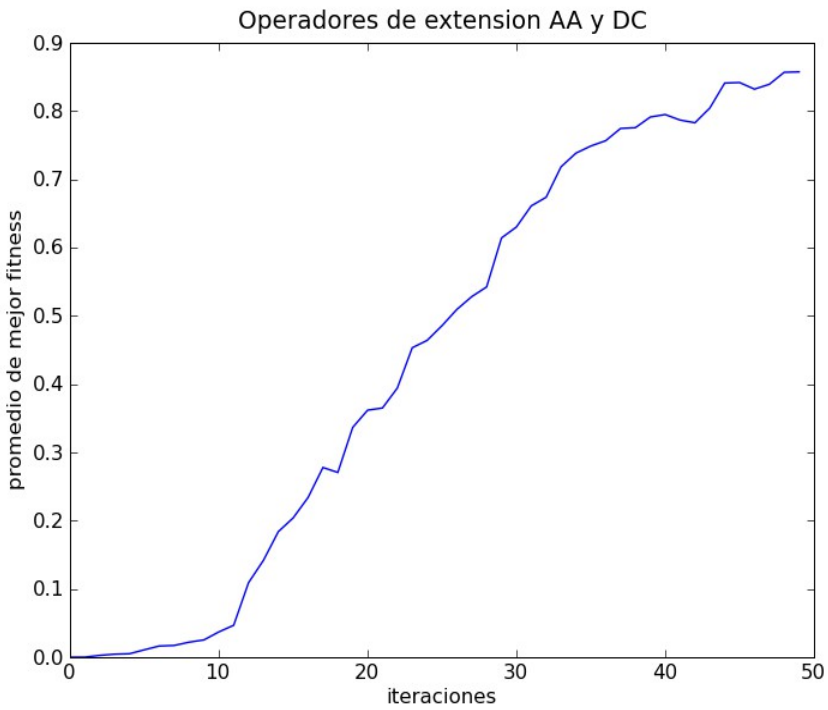

Crossover rate = 0.8 - Mutacion rate = 0.005
promedio de mejor fitness
iteraciones
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
0.0014
0.0016
0.0018
0
10
20
30
40
50

Como se puede ver en las gráficas anteriores, la configuración de crossover_rate = 0.8 y mutation_rate = 0.005 fue la que arrojo mejores resultados de fitness.

Los siguientes 3 experimentos se realizaron con la configuración de crossover_rate y mutation_rate antes mencionados y conservando el mismo tipo de selección El primer experimento consistió en incluir únicamente el operador de extensión *AddAlternative* al algoritmo principal, el segundo únicamente al operador *DropCondition* y el tercero ambos. Los resultados se presentan a continuación







La aplicación de estos operadores de extensión mejoro notablemente los resultados de fitness en cada iteración. Particularmente la aplicación de ambos operadores.

4. Respuestas y Conclusiones:

a.) ¿Cual es la mejor configuración de su algoritmo genético para clasificar los datos estudiados?

Después de realizar las distintas pruebas necesarias, el objetivo era determinar la mejor configuración siguiendo una serie de pasos. En primer lugar decidimos que tipo de algoritmo de selección resultaría mas efectivo, lo que se realizo probando tres tipos, selección por ruleta, por probabilidad y seleccionando los mejores, resultando ser el mas efectivo la selección por probabilidad. Luego se probo con las diferentes tasas de crossover y mutación, probando entre 0.4 y 0.8 para el crossover y entre 0,02 y 0.005 para la mutación, resultando la mejor configuración usar como tasa de crossover 0.8 y la tasa de mutación en 0.005. Por ultimo se probo la configuración anterior con los operadores de extensión de Gabil, probando primero con AddAlternative, luego con DropCondition y por ultimo con los dos, resultando lo mas efectivo usar los dos operadores simultáneamente.

Por lo tanto, la mejor configuración es usar selección por probabilidad, con tasa de crossover 0.8, tasa de mutación 0.005 y aplicando los dos operadores de extensión de Gabil de AddAlternative y DropCondition.

b.) ¿Son útiles los operadores de extensión?

Según los resultados obtenidos y las pruebas realizadas, se demuestra que son de mucha utilidad. Los resultados obtenidos previo a usar los operadores reflejaban un aumento en el fitness muy

precario, a veces sin pasar del 1% al final de una corrida. Sin embargo al usar los operadores de extensión el máximo fitness al evolucionar la población se mantenía entre el 80 % y 90 %. Gracias a estos datos y a las gráficas obtenidas, concluimos que es de mucha importancia la utilización de estos operadores para obtener individuos con un buen fitness, que puedan ayudar a clasificar el conjunto de prueba posteriormente

c.) ¿Cual es el mejor conjunto de reglas hallado por el algoritmo genético? Considere el numero de ejemplos clasificados correcta e incorrectamente.

El mejor conjunto de reglas es el siguiente, el cual tiene un fitness de 0.999017477036.

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1,
1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]
V
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1]
V
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0]
V
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0]
V
[1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,
1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1]
V
[1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,
1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0]
V
[1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0]
V
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
V
[1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
V
[1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]
\/
[1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0]
\/
[1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1,
1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]
\/
[1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
\/
[1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
\/
[1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1]
\/
[1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
\/
[1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
\/
[1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]
\/
[1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0]
\/
[1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1]
\/
[1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]
\/
[1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1]

En donde, para el conjunto de prueba logro clasificar con fitness igual a 1.0, 16280 y solo dejo un individuo por clasificar. Lo que nos da un porcentaje de 99.99 %.