

**Implementación**

**Algoritmo Genético**

*Álvaro Cuevas*

*Eugenio Álvarez*

*Pablo Dubikin*

*Inteligencia Artificial II – Grupo 5* 08/05/2017

**ÍNDICE**

[I. Diagrama de flujo del programa 3](#_Toc481845233)

[II. Manual de uso 5](#_Toc481845234)

[III. Resolución de ejercicios 6](#_Toc481845235)

[i. Funcionamiento con distintas fases de prueba 6](#_Toc481845236)

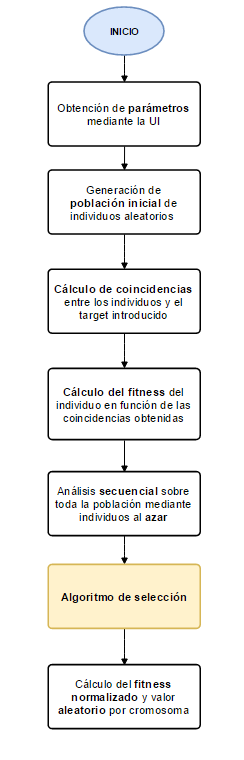
[ii. Combinación LTar - NPOB 6](#_Toc481845237)

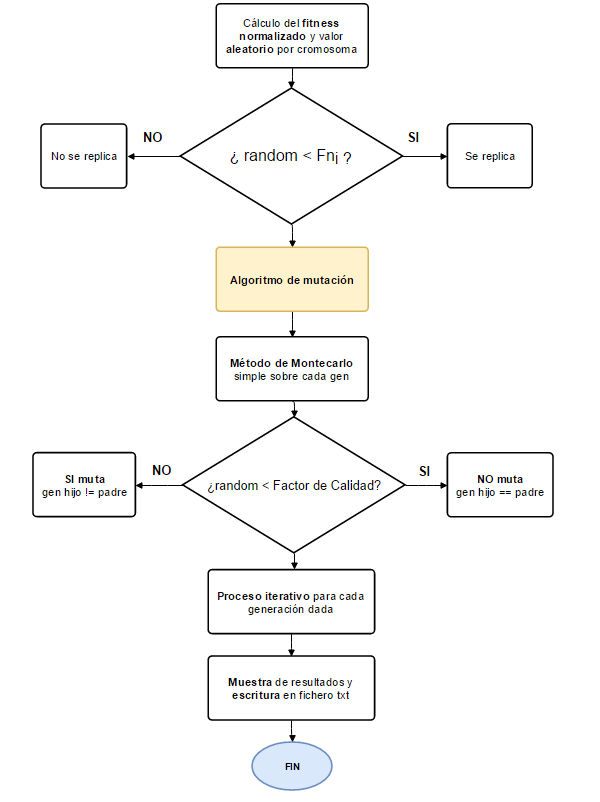
[iii. Estudio de la frase 6](#_Toc481845238)

[IV. Discusión 6](#_Toc481845239)

[V. Bibliografía 8](#_Toc481845240)

# Diagrama de flujo del programa





# Manual de uso

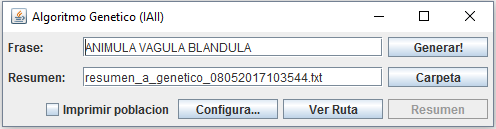
Mediante el seguimiento de los siguientes pasos consecutivos que conforman el manual de uso, se pretende que cualquier usuario logré ejecutar la aplicación final contenida en un archivo ejecutable de extensión *.jar* para recrear por sí mismo un algoritmo genético que logre alcanzar la frase insertada por el usuario a partir de un conjunto de individuos aleatorios mediante selección y mutación de los mismos.

El usuario podrá determinar en la UI del programa tanto la escritura de la frase objetivo como todo parámetro que tendrá en cuenta el algoritmo genético previa ejecución del procesamiento entre los cuales cabe enumerar los siguientes:

* Número de individuos de la población
* Número máximo de generaciones para el proceso
* Número límite de generaciones para muestra de resumen por pantalla
* Factor de calidad o probabilidad de que un gen no mute
* Carpeta destino del fichero resultado *\*opcional*
* Impresión del resultado por pantalla *\*opcional*

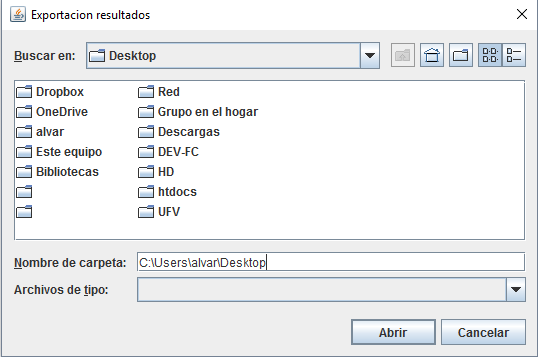
Todos los pasos detallados a continuación son consecutivos por lo que deben llevarse a cabo en el estricto orden que se detalla:

1. Hacer doble clic sobre el archivo ejecutable de extensión *.jar* para iniciar el programa por pantalla.

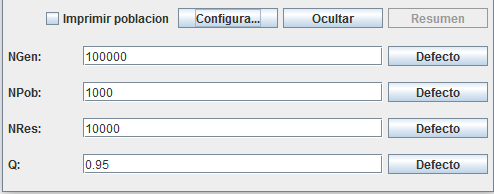


*Figura I: UI del programa*

1. Insertar a mano los respectivos parámetros, la carpeta destino para los resultados y la frase a elección libre del usuario para ser alcanzada por el algoritmo genético.



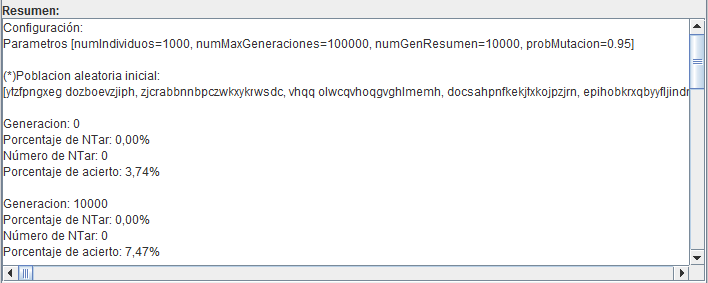
*Figura II: Asignación de ruta a elección*



*Figura III: Parámetros del algoritmo*

1. Insertados los valores deseados, clicar en el botón *Generar!* para iniciar el procesamiento del algoritmo y obtener el consiguiente resultado en un fichero .txt externo en la ruta previamente indicada.
2. Dirigirse a la ruta por defecto en la cual se ha extraído el fichero .txt resultante para poder visualizar los datos del proceso así como el número de generaciones llevadas a cabo hasta alcanzar la frase target introducida.

**\*Previa ejecución del algoritmo, el usuario podrá decidir si se muestra un resumen por la propia UI mediante el botón *Imprimir Población* o por el contrario, únicamente se generará el fichero final de extensión .txt**



*Figura IV: Resumen por pantalla*

# Resolución de ejercicios

## Funcionamiento con distintas fases de prueba

Con los parámetros por defecto de:

npob=1000

ngen=500000

nres=10000

q=0.95

Con 10 pruebas se saca media que:

* 2 caracteres tardan de 145.942 segundos en terminar la ejecución.
* 3 caracteres tardan de 164.818 segundos en terminar la ejecución.
* 4 caracteres tardan de 171.366 segundos en terminar la ejecución.
* 5 caracteres tardan de 176.116 segundos en terminar la ejecución.
* 6 caracteres tardan de 178.494 segundos en terminar la ejecución.
* 7 caracteres tardan de 180.216 segundos en terminar la ejecución.
* Con esto sacamos en conclusión que cuanto mayor es la cadena de caracteres que buscamos más tarda en ejecutarse.

Cambiando el npob con la misma palabra de 7 caracteres “palabra” encontramos:

Npob=2000 tarda en ejecutarse 360.278 segundos con:

Generacion: 490000

Porcentaje de NTar: 56,70%

NÃºmero de NTar: 1134

Porcentaje de acierto: 92,10%

Npob=1500 tarda en ejecutarse 268.903 segundos con:

268903

Generacion: 490000

Porcentaje de NTar: 61,47%

NÃºmero de NTar: 922

Porcentaje de acierto: 93,16%

## Combinación LTar - NPOB

Cuanto mayor sea LTar se necesita mas NPOB.

## Estudio de la frase

Para la frase target *ANIMULA VAGULA BLANDULA* se ha procedido a llevar a cabo un análisis de los resultados con el Factor de Calidad Q comprendido entre **0.8 y 1.0** viendo que ocurre con el **%NTar** entendido este como el número de targets incluidos en la población.

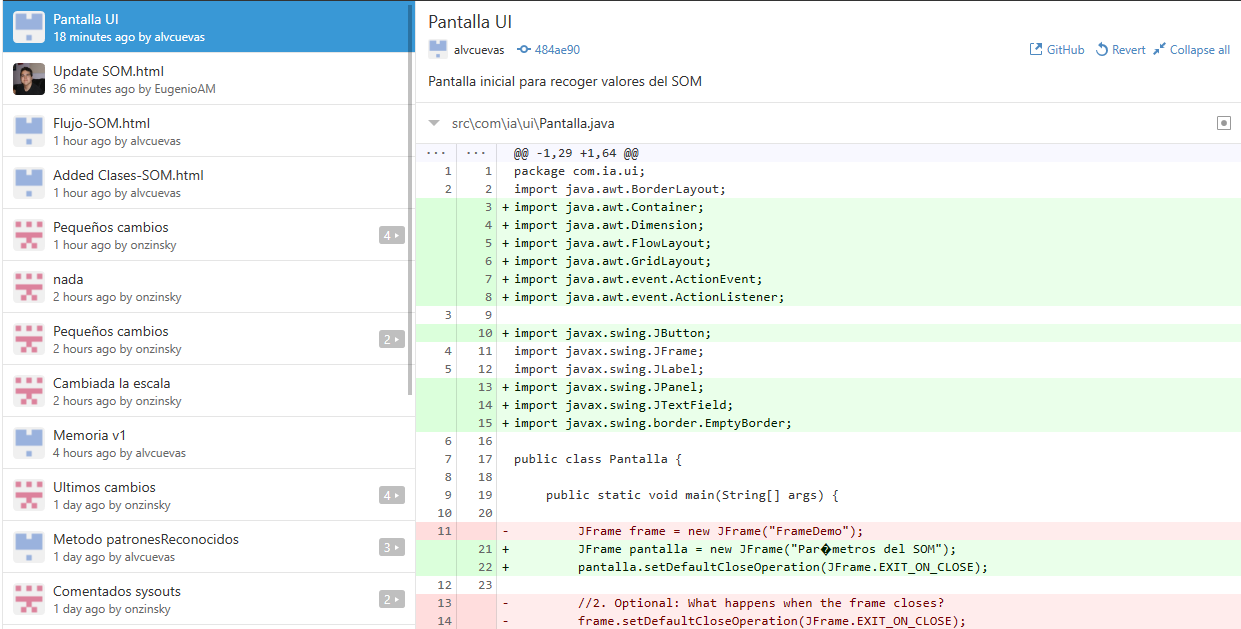
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | Población inicial | Max. Generaciones | %NTar | %Acierto | Tiempo |
| *0.95* | 2000 | 500000 | 18,05% | 92,56% | 397.324s |
| *0.9* | 2000 | 500000 | 0% | 71,12% | 365.133s |
| *0.85* | 2000 | 500000 | 0% | 53,14% | 357.871s |

# Discusión

Para la implementación de dicha práctica se optó por la elección de Java en cuanto a lenguaje de programación para su desarrollo debido a la comodidad que ofrece al tratarse de programación orientada a objetos y dado que, en este caso en concreto, se hace uso de varios objetos a los que continuamente se hace referencia.

Previo inicio del algoritmo enunciado en la práctica, desarrollamos por error la elaboración del algoritmo genético que alcanzara la frase target introducida mediante el algoritmo de selección elitista, a través del cual pudimos comprobar la extrema rapidez que puede adquirir un algoritmo genético bien implementado. Con el algoritmo de selección elitista como base, y puesto que nos habíamos equivocado en la manera de seleccionar los individuos, alteramos el algoritmo para adaptarlo a los requisitos enunciados en la práctica estableciendo la elección secuencial de todos y cada uno de los individuos de la población y no únicamente el mejor como habíamos decidido antes.

Del mismo modo, para un correcto seguimiento de los cambios y garantizar un control de versiones, se ha usado la plataforma *GitHub* para tener constancia de todo el desarrollo llevado a cabo por cada integrante del grupo además de tener la posibilidad de revertir cambios si fuera necesario ante posibles errores ocasionados durante la implementación. El uso de un repositorio git para almacenar el proyecto ofrece como ventaja su disponibilidad total para acceso de cualquier miembro del equipo en cualquier momento que quiera tocar algo del código o simplemente testear el funcionamiento. Mediante el uso de *commits,* los distintos miembros del grupo podrán enviar todos los cambios que hayan provocado dejando constancia de ello al resto de miembros, lo que provoca una buena organización en el desarrollo y una fácil solución cuando distintas tareas se distribuyen entre los miembros participantes.



*Figura V: Ejemplo de control de versiones en GitHub*

Tras la realización de esta práctica hemos podido ser conscientes del gran uso que se le puede dar a los algoritmos genéticos en diversos ámbitos reales como el reconocimiento de patrones, predicciones, optimización de problemas aplicados en robótica… siempre y cuando desde un primer momento se tenga muy clara la procedencia de la población inicial, la codificación de los cromosomas y la normalización de dichos datos para obtener la velocidad adecuada durante el procesamiento del algoritmo. Es muy fácil equivocarse si no se tiene clara la procedencia de los datos y los pasos consecutivos obligados que debe llevar a cabo todo algoritmo genético para lograr seleccionar, mutar y calcular los resultados como es debido.

Al igual que las redes neuronales, los algoritmos genéticos están a la orden del día para la resolución de numerosos problemas que aun a día de hoy se tratan de optimizar para sacar el máximo potencial, lo que supone que, esta disciplina y en concreto la aplicación de la inteligencia artificial en general, puede ser una buena temática en la que enfocar el proyecto de fin de grado de muchas carreras relacionadas con la informática.

# Bibliografía

[1] Moodle UFV, (2017). *Algoritmos Geneticos*. [online] Available at: http://moodleufv.ufv.es/moodle/mod/resource/view.php?id=153416 [Accessed 8 May 2017].

[2] Draw.io. (2017). *Flowchart Maker & Online Diagram Software*. [online] Available at: http://www.draw.io [Accessed 8 May 2017].

[4] Stackoverflow.com. (2017). *Stack Overflow*. [online] Available at: http://stackoverflow.com/ [Accessed 8 May 2017].

[5] Github, (2017). *Repositorio AG*. [online] Available at: https://github.com/alvcuevas/AG [Accessed 8 May 2017].