



DESARROLLO DE UNA HERRAMIENTA SOFTWARE PARA LA RESOLUCIÓN DE
PROBLEMAS DE OPTIMIZACIÓN CON ALGORITMOS GENÉTICOS

Proyecto de Fin de Carrera de modalidad oferta general

Realizado por: FRANCISCO JAVIER GARCÍA PAREDERO

Dirigido por: DICTINO CHAOS GARCÍA

Supervisado por: DICTINO CHAOS GARCÍA

Tribunal calificador:

Presidente: D./Da.

Secretario: D./Da.

Vocal: D./Da.

Fecha de lectura y defensa:

Calificación:

1. Resumen

2. Palabras clave

3. Overview

4. Keywords

Índice

1. Resumen.....	3
2. Palabras clave.....	4
3. Overview	5
4. Keywords	6
5. Índice de figuras	12
6. Introducción	15
7. Algoritmos genéticos	16
7.1. Representación de individuos.....	17
7.2. Algoritmo general.....	17
7.3. Operador de selección	18
7.3.1. Método de la ruleta.....	18
7.3.2. Selección por torneo.....	20
7.4. Operador de cruce.....	22
7.4.1. Cruce en un punto	22
7.4.2. Cruce en dos puntos	23
7.5. Operador de mutación	24
7.6. Limitaciones de los algoritmos genéticos	24
8. Planteamiento del problema.....	26
9. Diseño	28

9.1. Requisitos	28
9.1.1. Introducción de la función de coste	28
9.1.2. Codificación del vector de parámetros o cromosoma	29
9.1.3. Introducción del tamaño de la población	29
9.1.4. Establecimiento del número de generaciones	30
9.1.5. Establecimiento del número de eras.....	30
9.1.6. Selección de elitismo	30
9.1.7. Introducción de la probabilidad de cruce	30
9.1.8. Introducción de la probabilidad de mutación.....	31
9.1.9. Selección de un operador de selección	31
9.1.10. Validación de la configuración.....	31
9.1.11. Inicialización de la población.....	31
9.1.12. Evaluación de la función de coste	32
9.1.13. Operador de selección	32
9.1.14. Operador de cruce	32
9.1.15. Operador de mutación	32
9.1.16. Ejecución del algoritmo.....	32
9.1.17. Cancelación de la ejecución	33
9.1.18. Resultados parciales	33
9.1.19. Resultados finales.....	33

9.1.20.	Guardado de la ejecución	34
9.1.21.	Carga de la configuración guardada	35
9.2.	Casos de uso	35
9.3.	Componentes del sistema	48
9.4.	Diagrama de clases	50
9.4.1.	Modelo de datos	53
9.4.2.	Módulo de ejecución del algoritmo.....	55
9.4.3.	Módulo de presentación de resultados	56
9.4.4.	Módulo de interfaz gráfica de usuario	60
9.4.5.	Módulo de guardado de datos	61
9.5.	Flujo de ejecución	63
9.5.1.	Flujo de ejecución general	63
9.5.2.	Flujo de ejecución del algoritmo genético	64
9.5.3.	Cancelación de la ejecución	65
9.5.4.	Presentación de resultados parciales	66
9.6.	Interfaz gráfica de usuario	67
9.6.1.	Ventana principal de la aplicación	68
9.6.2.	Ventana de progreso del cálculo	69
9.6.3.	Ventana de resultados de una era	71
10.	Funcionamiento de la aplicación.....	72

10.1.	Requisitos previos.....	72
10.2.	Acceso a la aplicación	72
10.3.	Introducción de la función de coste	73
10.4.	Codificación del vector de parámetros	75
10.5.	Configuración de la ejecución	77
10.6.	Ejecución	78
10.7.	Cancelación de la ejecución.....	80
10.8.	Resultados de la ejecución.....	81
10.9.	Detalles de la evolución de una era	83
10.10.	Guardado de datos	84
10.11.	Carga de datos guardados	85
10.12.	Ayuda de la aplicación.....	86
11.	Ejemplos de ejecución	88
11.1.	Caso de prueba 1.....	88
11.2.	Caso de prueba 2.....	89
11.3.	Caso de prueba 3: Función de Ackley	90
11.4.	Caso de prueba 4: Función de Schaffer	91
11.5.	Caso de prueba 5.....	95
11.6.	Caso de prueba 6.....	96
12.	Planificación y presupuesto	98

13.	Conclusiones	99
14.	Referencias y bibliografía	100
15.	Siglas y acrónimos	101

5. Índice de figuras

Figura 1 Selección por torneo	21
Figura 2 Cruce en un punto.....	23
Figura 3 Cruce en dos puntos.....	24
Figura 4 Componentes del sistema	49
Figura 5 Diagrama de clases	51
Figura 6 Diagrama de clases	52
Figura 7 Modelo de datos.....	54
Figura 8 Módulo de ejecución	55
Figura 9 El patrón Observer.....	57
Figura 10 Módulo de presentación de resultados.....	58
Figura 11 Módulo de interfaz gráfica	61
Figura 12 Módulo de guardado de datos.....	62
Figura 13 Flujo de la ejecución general	64
Figura 14 Flujo de ejecución del algoritmo genético.....	65
Figura 15 Cancelación de la ejecución	66
Figura 16 Presentación de resultados parciales.....	67
Figura 17 Ventana principal de la aplicación.....	69
Figura 18 Ventana de progreso	70
Figura 19 Ventana de resultados de una era	71

Figura 20 Ventana principal de la aplicación.....	73
Figura 21 Área de la función de coste.....	75
Figura 22 Área de introducción de nuevos parámetros.....	75
Figura 23 Listado de parámetros declarados.....	76
Figura 24 Área de configuración de la aplicación	77
Figura 25 Ventana de progreso	79
Figura 26 Sección gráfica de los resultados	82
Figura 27 Mejores cromosomas de la ejecución	83
Figura 28 Evolución del cálculo de una era	84
Figura 29 Diálogo de guardado de datos.....	85
Figura 30 Diálogo de apertura de datos	86
Figura 31 Acceso a la ayuda de la aplicación	87
Figura 32 Caso de prueba 1.....	89
Figura 33 Caso de prueba 3. Función de Ackley	90
Figura 34 Caso de prueba 4. Función de Schaffer	92
Figura 35 Resultados con la configuración por defecto.....	93
Figura 36 Evolución de la mejor era	94
Figura 37 Resultados con 1000 generaciones	95
Figura 38 Caso de prueba 5.....	96
Figura 39 Caso de prueba 6.....	97

6. Introducción

Es un poco hablar de lo que estás haciendo y de que otras herramientas existen actualmente para hacer lo mismo. También deberías hablar aquí de las decisiones básicas a la hora de hacer la aplicación (ej: ¿Por qué has usado Java?) y los objetivos de la misma.

7. Algoritmos genéticos

Los algoritmos genéticos se apoyan en los principios de la evolución mediante selección natural, enunciados por Charles Darwin en su libro El Origen de las especies. Estos principios pueden resumirse así:

- Cada individuo tiende a transmitir sus rasgos a su progenie.
- Sin embargo, la naturaleza produce individuos con rasgos diferentes.
- Los individuos más adaptados, tienden a producir mayor progenie.
- Durante largos periodos de tiempo se puede acumular la variación produciendo nuevas especies completamente adaptadas a nichos particulares.

A mediados de la década de 1970, un investigador de la Universidad de Michigan llamado John Holland desarrolló las ideas que más tarde se convertirían en los algoritmos genéticos.

Los componentes básicos de un algoritmo genético son los siguientes:

- Una representación para los individuos.
- Una medida del grado de adaptación de un individuo al medio, la función de calidad.
- Un operador de selección, con probabilidad de selección de cada individuo proporcional a su calidad.
- Un operador emparejamiento o reproducción, que dará lugar a nuevos individuos en la siguiente generación.
- Un operador mutación, capaz de alterar las características de los nuevos individuos, incrementando la riqueza genética de la población.

En cada generación se crea un nuevo conjunto de individuos utilizando el material genético de los mejores individuos de la generación anterior.

7.1.Representación de individuos

Los individuos pueden representarse de diversas formas. Dependiendo de la naturaleza de esta representación, la implementación de los operadores básicos será diferente.

Habitualmente, la teoría clásica de algoritmos genéticos utiliza para la representación la idea de cromosoma. Un cromosoma es el conjunto de los parámetros que determinan la estructura de un individuo. Cada uno de estos parámetros recibe el nombre de gen.

Existen diversos modos de representar los genes y por tanto los cromosomas. Una posibilidad habitual es utilizar valores binarios. Se asigna un número de bits a cada gen.

Sin embargo, también pueden existir representaciones que asignen a cada cromosoma un valor entero, real o cualquier otro tipo de datos específicos al campo de aplicación del algoritmo.

7.2.Algoritmo general

El algoritmo trabaja sobre una población de individuos, representados por sus cromosomas. Cada uno de estos individuos tiene asociado un valor de su bondad determinado por la función de calidad o de coste.

El funcionamiento genérico de un algoritmo genético podría ser el siguiente:

Generar una población aleatoria de N_i individuos

Mientras que no se cumpla el criterio de terminación

 Crear una nueva población

 Seleccionar padres

Cruzar los padres con probabilidad p_c

Mutar los individuos resultantes con probabilidad p_m

Finalmente devolver como solución el individuo con mayor calidad de la población

Los criterios de terminación pueden ser variados:

- Los mejores individuos de la población generada representan soluciones suficientemente buenas.
- Se han alcanzado el número de generaciones especificado.
- La población ha convergido. Cuando la mayoría de los cromosomas de la población tienen la mayoría de sus genes iguales se dice que la población ha convergido. Cuando se da esta situación, el hecho de repetir sucesivas iteraciones no va a producir mejores resultados.

7.3. Operador de selección

Este operador sirve para extraer individuos de una población, de modo que la probabilidad de extracción de un individuo sea proporcional al valor tomado por la función de calidad. De este modo Existen diversos métodos.

7.3.1. Método de la ruleta

Se trata de un método de selección entre N elementos no equiprobables. A cada uno de los individuos se le asigna una probabilidad acumulada en función de su calidad. Se genera un número aleatorio dentro del rango $0,1$ y se selecciona el individuo correspondiente.

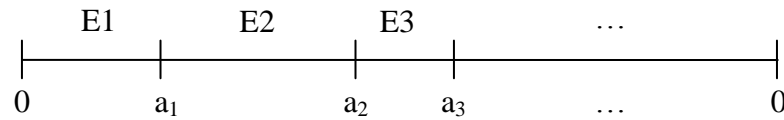
1. En primer lugar se calcula la calidad de cada individuo de la población.

2. Dividiendo cada calidad por la calidad total, obtenemos una serie de probabilidades (suman 1). Estas serán las probabilidades de selección de cada individuo. Se cumple que cuanto mayor es la calidad de un individuo, mayor es su probabilidad de selección.

$$p_i = \frac{c_i}{\text{sum}(c)}$$

3. A continuación se calculan las probabilidades acumuladas.

$$a_k = \sum_{i=1}^k p_i$$



4. Finalmente se genera un valor aleatorio entre 0 y 1 y se selecciona el individuo correspondiente.

Este método presenta un inconveniente pues únicamente funciona cuando la función de calidad no devuelve números negativos. Para solventar este problema, se puede normalizar el valor de calidad del individuo para lograr que siempre obtenga valores positivos. El método quedaría del siguiente modo.

1. En primer lugar se calcula la calidad de cada individuo de la población.
2. Se obtiene el menor valor de calidad del conjunto de la población.
3. En caso de que este valor de calidad sea inferior a 0 se obtiene el factor de normalización.

$$f = \min(c_i) + 1$$

4. El algoritmo continúa del mismo modo, pero a partir de ahora, se utiliza la calidad normalizada.

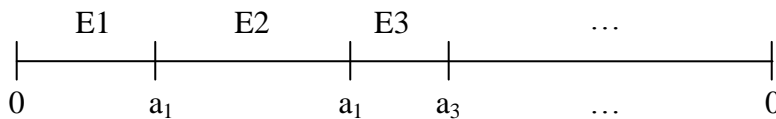
$$c'_i = c_i + f$$

5. Dividiendo cada calidad por la calidad total, obtenemos una serie de probabilidades (suman 1). Estas serán las probabilidades de selección de cada individuo. Se cumple que cuanto mayor es la calidad de un individuo, mayor es su probabilidad de selección.

$$p_i = \frac{c'_i}{\text{sum}(c)}$$

6. A continuación se calculan las probabilidades acumuladas.

$$a_k = \sum_{i=1}^k p_i$$



7. Finalmente se genera un valor aleatorio entre 0 y 1 y se selecciona el individuo correspondiente.

7.3.2. Selección por torneo

En la selección por torneo, se elige un número determinado de individuos de forma aleatoria de entre los individuos de la población. Los individuos seleccionados se comparan entre sí y se elige el que tiene mayor aptitud.

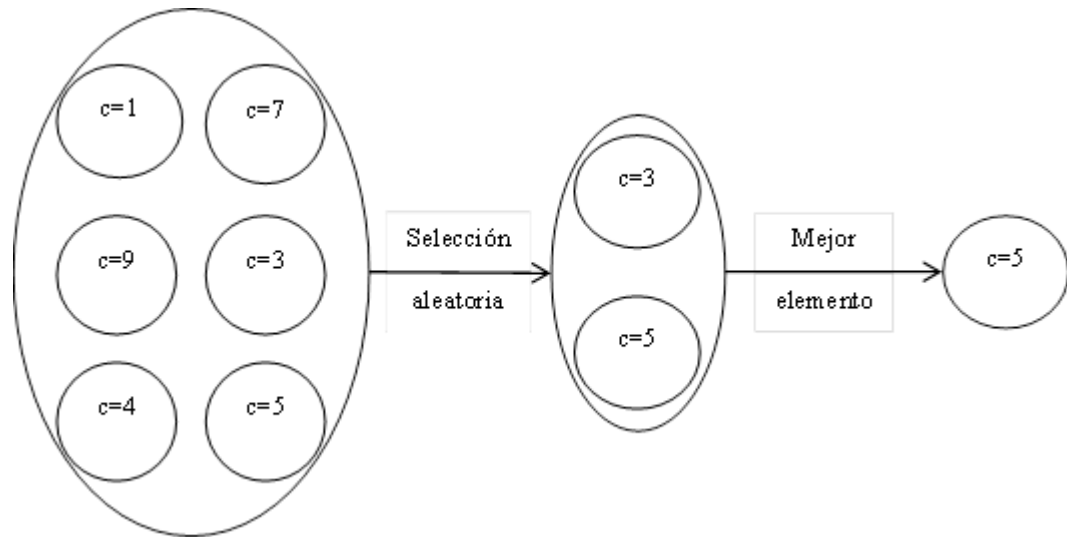


Figura 1 Selección por torneo

En función del número de individuos que participen en el torneo se modifica la presión de selección. Cuando el número de individuos es elevado, la presión de selección es alta, los peores individuos apenas tienen posibilidades de ser seleccionados y la búsqueda de soluciones se centra en el espacio próximo a las mejores soluciones actuales. Por el contrario cuando el número de individuos es reducido, los peores tienen más posibilidades de ser seleccionados y la búsqueda de soluciones puede explorar nuevas regiones del espacio de búsqueda.

Debe tenerse en cuenta que la selección por torneo únicamente comprueba qué elementos son mejores que los otros, pero no tiene en cuenta cuánto mejores son. Esto hace que la presión de selección se mantenga constante. Así, aunque hubiera un individuo extraordinariamente mejor que los demás, este no pasaría a dominar la siguiente generación, lo que comprometería la diversidad de la población. Por contra, se

amplifican pequeñas diferencias en la calidad entre dos elementos, descartando elementos que son mínimamente inferiores a sus contendientes en el torneo.

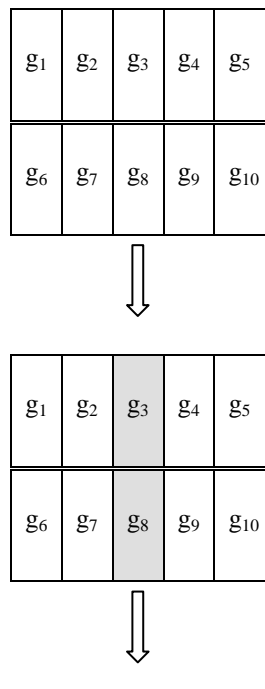
7.4. Operador de cruce

El operador de cruce intercambia información genética entre dos individuos de la población.

Existen diversos algoritmos de cruce que se exponen a continuación.

7.4.1. Cruce en un punto

En este algoritmo se forma un vector compuesto por los genes del individuo. A continuación se selecciona un punto de corte en el vector de forma aleatoria generando dos segmentos diferenciados en cada vector. Las colas resultantes de cada vector se intercambian entre sí resultando dos vectores descendientes que heredan información genética de cada uno de los padres.



g ₁	g ₂	g ₃	g ₄	g ₅
g ₆	g ₇	g ₈	g ₉	g ₁₀



g ₁	g ₂	g ₈	g ₉	g ₁₀
g ₆	g ₇	g ₃	g ₄	g ₅

Figura 2 Cruce en un punto

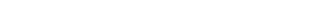
7.4.2. Cruce en dos puntos

Se trata de una evolución del algoritmo de cruce en un punto. En este caso se seleccionan dos puntos en el vector de genes. Se intercambian entre ambos vectores los intervalos delimitados por los dos puntos.

g ₁	g ₂	g ₃	g ₄	g ₅	g ₆
g ₇	g ₈	g ₉	g ₁₀	g ₁₁	g ₁₂



g ₁	g ₂	g ₃	g ₄	g ₅	g ₆
g ₇	g ₈	g ₉	g ₁₀	g ₁₁	g ₁₂



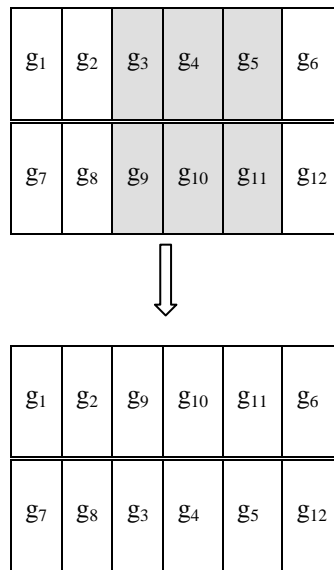


Figura 3 Cruce en dos puntos

7.5. Operador de mutación

Se trata de un operador que provoca que se modifique alguno de los genes del individuo con un valor aleatorio. Esto suele hacerse con una probabilidad preestablecida generalmente muy baja.

El efecto de este operador es el de mantener la diversidad genética de una población, escapar de los óptimos locales y desplazar a los individuos hacia zonas del espacio de búsqueda que no pueden alcanzarse mediante el resto de operadores.

7.6. Limitaciones de los algoritmos genéticos

En numerosas ocasiones los algoritmos genéticos tienden a converger hacia valores óptimos locales en lugar de soluciones óptimas globales del problema. Tienden a buscar mejoras a corto plazo desdénando mejoras a largo plazo. Este problema puede solventarse modificando la función de evaluación, aumentando la probabilidad de mutación o mediante el uso de operadores de selección que mantengan la diversidad genética en la población. Otra posible solución al problema es reiniciar el cálculo una

vez se realice la evolución durante un número determinado de generaciones. Se denomina era a cada uno de los ciclo de generaciones.

Un posible problema se da cuando el mejor miembro de una población falle en producir descendientes para la siguiente generación. Esto se puede resolver mediante una técnica denominada elitismo. Esta técnica consiste en copiar el mejor elemento de una generación en la generación siguiente.

8. Planteamiento del problema

El objetivo de este proyecto consiste en el desarrollo de una aplicación para resolver problemas de optimización mediante el uso de algoritmos genéticos.

Dada una función de evaluación o de coste, con un recorrido dentro del conjunto de los reales:

$$J: \mathbb{R}^n \rightarrow \mathbb{R}$$

La función de evaluación recibe una serie de parámetros reales acotados entre un valor máximo y un valor mínimo.

$$x_i \in [x_i^{\min}, x_i^{\max}]$$

El problema a resolver consiste en encontrar el conjunto de parámetros óptimo que maximice el valor de la función de evaluación empleando para ello un algoritmo genético.

El algoritmo que se va a emplear es el siguiente:

Durante un número determinado de eras

Generar una población aleatoria de N_i individuos

Durante un número determinado de generaciones

Crear una nueva población

Seleccionar padres

Cruzar los padres con probabilidad p_c

Mutar los individuos resultantes con probabilidad p_m

Opcionalmente aplicar un operador de elitismo

Finalmente devolver como solución el individuo con mayor calidad de la población

Se debe proporcionar una interfaz de usuario agradable. Además debe proporcionarse una visualización de los resultados obtenidos clara y deben presentarse detalles de la evolución del programa durante su ejecución.

Adicionalmente, deben proporcionarse mecanismos para almacenar los detalles de la ejecución del programa.

9. Diseño

Con los requisitos, análisis de casos de uso, etc. También debes de especificar los componentes que tiene, poner un diagrama de clases e indicar que hace cada una de ellas. Indicar cómo se evalúan las expresiones, cómo se organiza la interfaz gráfica... No incluyas código (esto irá en el CD) a no ser que fuese imprescindible. Se trata de documentar las clases indicando que hacen y cómo está organizada la aplicación no de describir exhaustivamente cada línea del programa...

9.1.Requisitos

En el análisis del problema se han detectado los siguientes requisitos funcionales.

9.1.1. Introducción de la función de coste

Se trata de la función que se va a tratar de optimizar mediante el uso de la herramienta. Esta función tiene como parámetros una serie de números reales y tiene como resultado un valor positivo en el conjunto de los números reales:

$$J: \mathbb{R}^n \rightarrow \mathbb{R}$$

Los posibles elementos que podrán incluirse en la función son

- Números reales.
- Parámetros, expresados por su nombre.
- Los siguientes operadores: +, -, /, *, $\sqrt{}$, sin, cos, tan, asin, atan, abs, log, \log_{10} , \log_2
- Los símbolos de paréntesis “(” y “)”.
- El número π , expresado mediante el símbolo “pi”.

- El número e , expresado mediante el símbolo “ e ”.

9.1.2. Codificación del vector de parámetros o cromosoma

El cromosoma o vector de parámetros es el conjunto de parámetros que se aplican a la función de coste. Está formado por una serie de elementos o genes, que consisten en los parámetros de la función de evaluación.

$$\theta = [x_1, \dots, x_n]$$

Cada uno de los genes es un número real acotado entre un valor mínimo y un valor máximo.

$$x_i \in [x_i^{\min}, x_i^{\max}]$$

Por este motivo, los valores necesarios para definir un gen son los siguientes:

- Un nombre.
- Un valor real mínimo, para expresar la cota inferior.
- Un valor real máximo, para expresar la cota superior.
- Un valor entero, para expresar la precisión, el número de decimales con los que trabajará.

9.1.3. Introducción del tamaño de la población

El tamaño de la población es el número de elementos (cromosomas) con los que trabajará el algoritmo. Se trata de un valor numérico entero comprendido entre 1 y 1000. Inicialmente tendrá un valor por defecto de 100 elementos.

9.1.4. Establecimiento del número de generaciones

El número de generaciones determinará el número de iteraciones que se realizarán durante la ejecución del algoritmo. Se trata de un valor numérico entero comprendido entre 1 y 1000. Este valor deberá poderse introducir por pantalla. Inicialmente tendrá un valor por defecto de 100 generaciones.

9.1.5. Establecimiento del número de eras

El número de eras determinará el número de ejecuciones del algoritmo que se realizarán. Se realizan varias ejecuciones distintas del algoritmo para evitar problemas en la ejecución como la aparición de superelementos que dominen por completo la evolución. Se trata de un valor numérico entero comprendido entre 1 y 100. Este valor debe poder ser introducido por pantalla. Inicialmente tendrá un valor por defecto de 10 eras.

9.1.6. Selección de elitismo

Se debe poder seleccionar si el algoritmo operará con elitismo o no. El selector estará activado por defecto.

9.1.7. Introducción de la probabilidad de cruce

La probabilidad de cruce determinará la probabilidad de que los cromosomas sean seleccionados para aplicar el operador de cruce. Se trata de un valor numérico real entre 0 y 1 que debe poderse introducir por pantalla. Inicialmente tendrá un valor por defecto de 0.2.

9.1.8. Introducción de la probabilidad de mutación

La probabilidad de mutación determina la probabilidad de que a un cromosoma se le aplique el operador de mutación. Se trata de un valor numérico real entre 0 y 1 que debe poderse introducir por pantalla. Inicialmente tendrá un valor por defecto de 0.015.

9.1.9. Selección de un operador de selección

El usuario debe poder elegir entre distintos operadores de selección. En concreto debe poder elegir entre los siguientes operadores:

- Método de la ruleta
- Selección por torneo

9.1.10. Validación de la configuración

Antes de comenzar la ejecución del algoritmo debe validarse que la configuración seleccionada es válida. En particular deben validarse los siguientes aspectos:

- Que la función de coste introducida es sintácticamente correcta.
- Que la configuración del cromosoma es correcta y congruente con la función de coste. Esto significa que los genes introducidos coinciden en número y nombre con los parámetros de la función introducida.

9.1.11. Inicialización de la población

Al inicio de la ejecución del algoritmo genético se debe generar de forma aleatoria una población de individuos que cumplan con las restricciones especificadas para el cromosoma y cada uno de sus genes.

9.1.12. Evaluación de la función de coste

Dado un cromosoma con valores debe poderse evaluar la función de coste para los genes que contiene.

9.1.13. Operador de selección

Se ejecutará el operador de selección que se haya configurado en la herramienta.

9.1.14. Operador de cruce

El operador de cruce que se utilizará es el operador de cruce en un punto con la probabilidad de cruce introducida anteriormente.

9.1.15. Operador de mutación

El operador de mutación que se aplicará es el de mutación uniforme con la probabilidad configurada inicialmente.

9.1.16. Ejecución del algoritmo

Debe ejecutarse un algoritmo genético según la configuración introducida. El número de eras determinará el número de ejecuciones distintas del algoritmo que se realizarán. El número de generaciones determinará el número de iteraciones que realizará el algoritmo. El tamaño de la población determinará el número de elementos con los que trabajará y las probabilidades de mutación y de cruce determinarán la probabilidad con que se aplicarán los operadores de mutación y de cruce.

9.1.17. Cancelación de la ejecución

La ejecución del algoritmo debe poder cancelarse en cualquier momento. Al cancelar la ejecución se mostrarán los resultados finales con los datos calculados hasta el momento de la cancelación.

9.1.18. Resultados parciales

Durante la ejecución del algoritmo deben mostrarse los datos que se vayan calculando, así como información del progreso de la ejecución y del tiempo transcurrido. En concreto deben mostrarse los siguientes datos:

- Tiempo de ejecución del programa.
- Número de Era actual.
- Número de generación actual.
- Mejor cromosoma obtenido hasta el momento.
- Valor medio de la función de coste durante la generación anterior.
- Desviación estándar de la función de coste en la generación anterior.
- Mejora porcentual obtenido en el mejor valor de la función de coste en la generación anterior.
- Valor medio de los mejores valores de la función de coste obtenidos a lo largo de las distintas eras.

9.1.19. Resultados finales

Al finalizar la ejecución del algoritmo deben mostrarse los datos de los cálculos realizados. Los datos que se deben mostrar son:

- El mejor cromosoma obtenido en cada era.
- El mejor valor de la función de coste obtenido en cada era.

Además deben mostrarse presentaciones gráficas de los datos obtenidos. En concreto, deben mostrarse de forma gráfica los siguientes datos:

- La evolución del mejor valor de la función de coste obtenido a lo largo de las generaciones en una era.
- La evolución del valor medio de la función de coste para los individuos de la población a lo largo de las generaciones en una era.
- La evolución de la desviación típica de la función de coste para los individuos de la población a lo largo de las generaciones en una era.

9.1.20. Guardado de la ejecución

La herramienta debe permitir almacenar en un fichero los datos correspondientes a la ejecución del algoritmo. Los datos que deben almacenarse son:

- La función de coste.
- La codificación del cromosoma.
- El tamaño de la población.
- El número de eras a ejecutar.
- El número de generaciones.
- La probabilidad de mutación.
- La probabilidad de cruce.
- La existencia de elitismo.

- El operador de selección a emplear.
- Los resultados de la ejecución.

9.1.21. Carga de la configuración guardada

La herramienta debe poder cargar una configuración a partir del fichero guardado. Los datos que debe poder cargar son los especificados en el punto 9.1.20

9.2. Casos de uso

El único actor que se ha identificado es el usuario final de la herramienta. Del análisis de los requisitos se han identificado los siguientes casos de uso:

CU1	Introducción de la función de coste
Descripción: Permite al usuario introducir la función de coste que se pretende optimizar mediante el uso de la herramienta.	
Actores: El usuario final.	
Precondiciones: Ninguna.	
Flujo Normal:	

El usuario introduce la función que desea optimizar.
Flujo Alternativo: No aplica.
Post condiciones: No aplica.

CU 1 Introducción de la función de coste

CU2	Introducción del número de eras
Descripción: Permite al usuario introducir un valor numérico entre uno y cien representando el número de eras que se desea que se ejecuten.	
Actores: El usuario final.	
Precondiciones: Ninguna.	
Flujo Normal: El usuario introduce en el campo que contiene las generaciones un número entero entre uno y cien.	

Flujo Alternativo:

Si el usuario introduce un valor no válido, bien por no ser numérico o bien por estar fuera del rango aceptable, el valor del campo vuelve a su valor anterior.

Post condiciones:

No aplica.

CU 2 Introducción del número de eras

CU3	Introducción del número de generaciones
Descripción: Permite al usuario introducir un valor numérico entre uno y mil representando el número de generaciones que se desea que se ejecuten.	
Actores: El usuario final.	
Precondiciones: Ninguna.	
Flujo Normal: El usuario introduce un número entero entre uno y mil.	

<p>Flujo Alternativo:</p> <p>Si el usuario introduce un valor no válido, bien por no ser numérico o bien por estar fuera del rango aceptable, el valor del campo vuelve a su valor anterior.</p>
<p>Post condiciones:</p> <p>No aplica</p>

CU 3Introducción del número de generaciones

CU4	Introducción de la codificación de los cromosomas
<p>Descripción:</p> <p>Permite al usuario introducir la codificación de los cromosomas que va a emplear el algoritmo.</p>	
<p>Actores:</p> <p>El usuario final.</p>	
<p>Precondiciones:</p> <p>Ninguna.</p>	
<p>Flujo Normal:</p> <ol style="list-style-type: none"> 1. Por cada gen que componga el cromosoma el usuario introduce el nombre, valor mínimo, valor máximo y la precisión numérica que se va a emplear. 	

<ol style="list-style-type: none"> 2. El usuario pulsa el botón de añadir. 3. La herramienta valida que el gen tenga un nombre, que no exista ningún gen con el mismo nombre y que el valor máximo no sea inferior al valor mínimo. 4. La herramienta añade el gen a la lista de genes del cromosoma.
<p>Flujo Alternativo:</p> <ol style="list-style-type: none"> 4. El sistema valida que el gen tenga nombre, que el nombre no esté repetido y que el valor máximo no sea inferior al valor mínimo. Si los datos del gen no son válidos, la herramienta muestra un mensaje indicando el error.
<p>Post condiciones:</p> <ol style="list-style-type: none"> 1. Se creará un listado con los genes configurados para el cromosoma.

CU 4 Introducción de la codificación de los cromosomas

CU5	Introducción del tamaño de la población
<p>Descripción:</p> <p>Permite al usuario introducir el número de cromosomas que contiene la población durante la ejecución del algoritmo genético.</p>	
<p>Actores:</p> <p>El usuario final.</p>	
<p>Precondiciones:</p>	

Ninguna.
Flujo Normal: El usuario introduce en el campo tamaño de la población un valor numérico entre uno y mil.
Flujo Alternativo: Si el usuario introduce un valor no válido, bien por no ser numérico o bien por estar fuera del rango aceptable, el valor del campo vuelve a su valor anterior.
Post condiciones: No aplica.

CU 5 Introducción del tamaño de la población

CU6	Introducción de la probabilidad de cruce
Descripción: Permite al usuario introducir el valor que se utilizará como probabilidad para ejecutar el operador de cruce en el algoritmo genético.	
Actores: El usuario final.	
Precondiciones:	

Ninguna.
<p>Flujo Normal:</p> <p>El usuario introduce en el campo probabilidad de cruce un valor numérico entre cero y uno.</p>
<p>Flujo Alternativo:</p> <p>Si el usuario introduce un valor no válido, bien por no ser numérico o bien por estar fuera del rango aceptable, el valor del campo vuelve a su valor anterior.</p>
<p>Post condiciones:</p> <p>No aplica.</p>

CU 6 Introducción de la probabilidad de cruce

CU7	Introducción de la probabilidad de mutación
<p>Descripción:</p> <p>Permite al usuario introducir el valor que se utilizará como probabilidad para ejecutar el operador de mutación en el algoritmo genético.</p>	
<p>Actores:</p> <p>Si el usuario introduce un valor no válido, bien por no ser numérico o bien por estar fuera del rango aceptable, el valor del campo vuelve a su valor anterior.</p>	

<p>Precondiciones:</p> <p>Ninguna.</p>
<p>Flujo Normal:</p> <p>El usuario introduce en el campo probabilidad de cruce un valor numérico un valor numérico entre cero y uno.</p>
<p>Flujo Alternativo:</p> <p>Si el usuario introduce un valor no válido, bien por no ser numérico o bien por estar fuera del rango aceptable, el valor del campo vuelve a su valor anterior.</p>
<p>Post condiciones:</p> <p>No aplica.</p>

CU 7 Introducción de la probabilidad de mutación

CU8	Guardar datos
<p>Descripción:</p> <p>Permite al usuario guardar en un fichero los datos de configuración y de ejecución del algoritmo.</p>	
<p>Actores:</p> <p>El usuario final.</p>	

Precondiciones:

Se han introducido valores de configuración en la herramienta.

Flujo Normal:

1. El usuario pulsa en el botón de guardar configuración.
2. El usuario selecciona un fichero en el que guardar su configuración.
3. La herramienta almacenará en el fichero los datos de configuración existentes:
 - a. Número de eras.
 - b. Número de generaciones.
 - c. Tamaño de la población
 - d. Probabilidad de mutación.
 - e. Probabilidad de cruce.
 - f. Función de coste.
 - g. Codificación de cromosomas.
 - h. Algoritmo de selección.
 - i. Utilización o no de elitismo.
4. La herramienta almacenará en el fichero los datos correspondientes a la ejecución.

Flujo Alternativo:

3. En caso de producirse un error al escribir el fichero, la herramienta muestra un

mensaje de error.

Post condiciones:

Se creará un fichero con los datos de configuración y de ejecución del algoritmo.

CU 8 Guardar datos

CU9

Cargar datos

Descripción:

Permite al usuario cargar los datos de configuración y de ejecución almacenados en un fichero.

Actores:

El usuario final.

Precondiciones:

Ninguna.

Flujo Normal:

1. El usuario pulsa en el botón de cargar configuración.
2. El usuario selecciona el fichero que contiene la configuración.
3. En la pantalla principal de la herramienta aparecerán introducidos los parámetros de configuración que contiene el fichero. Estos parámetros son:

- a. Número de eras.
- b. Número de generaciones.
- c. Tamaño de la población
- d. Probabilidad de mutación.
- e. Probabilidad de cruce.
- f. Función de coste.
- g. Codificación de cromosomas.
- h. Algoritmo de selección.
- i. Utilización o no de elitismo.

- 4. En caso de existir datos con resultados de ejecución en el fichero se mostrarán mediante la herramienta.

Flujo Alternativo:

- 3. En caso de que el formato del fichero no sea legible por la herramienta, se muestra un mensaje de error

Post condiciones:

Los parámetros de configuración contenidos en el fichero se mostrarán en los campos correspondientes en la herramienta.

La herramienta mostrará los resultados de ejecución contenidos en el fichero si estos existen.

CU10	Resolución de problemas de optimización
Descripción: Hace que la herramienta resuelva el problema de optimización configurado.	
Actores: El usuario final	
Precondiciones: La herramienta debe estar correctamente configurada	
Flujo Normal: <ol style="list-style-type: none"> 1. El usuario pulsa en el botón de ejecutar. 2. El sistema comprueba que existe una función de coste correcta. 3. El sistema valida que se han configurado los cromosomas de forma correcta y congruente con la función de coste 4. El sistema comienza la ejecución de un algoritmo genético con las propiedades configuradas. 5. El sistema muestra resultados parciales durante su ejecución. 6. Una vez finalizada la ejecución, el sistema muestra los resultados finales del cálculo. 	
Flujo Alternativo:	

<ol style="list-style-type: none"> 1. Si la función de coste no es correcta, el sistema muestra un mensaje de error 2. Si la configuración de los cromosomas no es coherente con la función de coste o no es correcta, el sistema muestra un mensaje de error. 4. Si durante la ejecución del algoritmo se produjese algún error, se detiene la ejecución, se muestra un mensaje de error y se muestran los resultados obtenidos hasta el momento.
<p>Post condiciones:</p> <p>Al finalizar la ejecución, el sistema mostrará los resultados finales del cálculo.</p>

CU 10 Resolución de problemas de optimización

CU11	Cancelación de la ejecución
<p>Descripción:</p> <p>El usuario puede cancelar en todo momento la ejecución.</p>	
<p>Actores:</p> <p>El usuario final.</p>	
<p>Precondiciones:</p> <p>El sistema debe estar calculando la solución a un problema de optimización.</p>	
<p>Flujo Normal:</p>	

- | |
|---|
| <ol style="list-style-type: none">1. El usuario pulsa en el botón de cancelar.2. La ejecución se detiene antes de finalizar.3. La aplicación mostrará los resultados calculados hasta el momento. |
| Flujo Alternativo:

No aplica. |
| Post condiciones:

La ejecución se detendrá sin haber finalizado. |

CU 11Cancelación de la ejecución

9.3.Componentes del sistema

Los componentes que conforman el sistema, así como sus dependencias con componentes de terceros, se presentan en el siguiente diagrama.

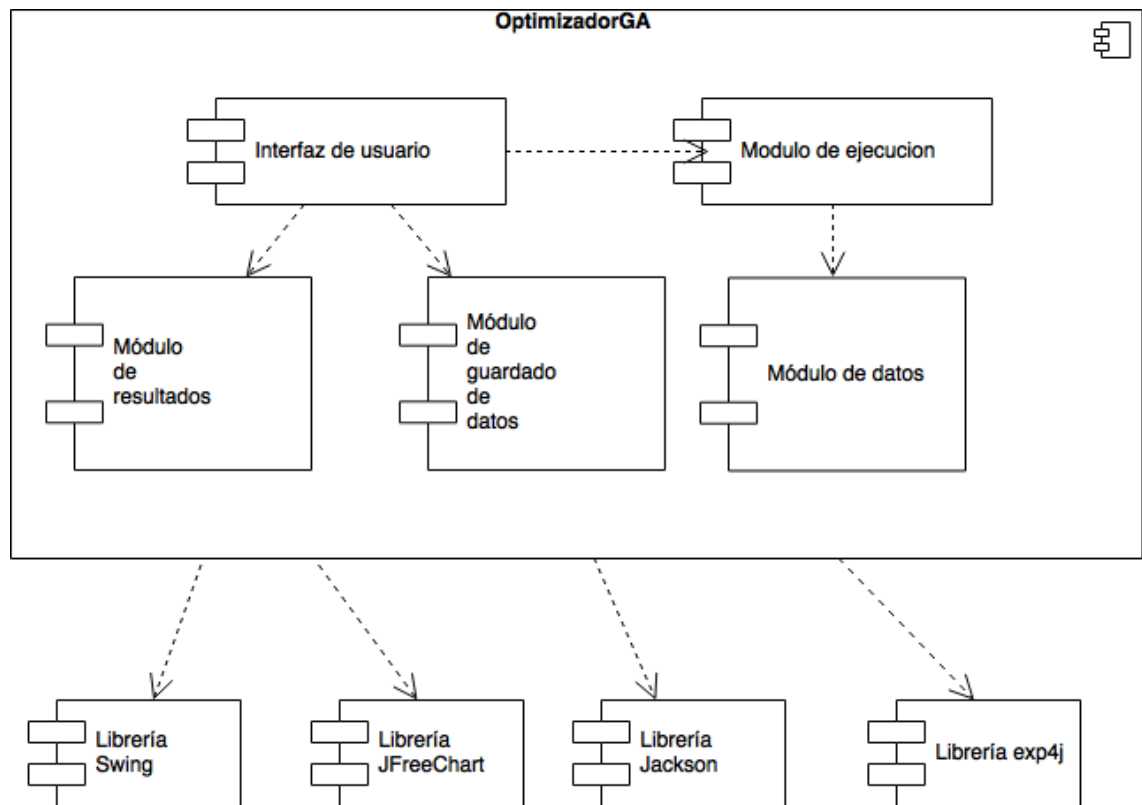


Figura 4 Componentes del sistema

Como puede observarse, el sistema está formado por un único componente. Este componente está compuesto por varios módulos:

- **Interfaz de usuario:** es el módulo encargado de la interacción con el usuario.
- **Módulo de ejecución:** este módulo se encarga de gestionar la ejecución del algoritmo.
- **Módulo de datos:** este módulo se encarga de gestionar el modelo de datos con los que trabaja la aplicación.
- **Módulo de resultados:** este módulo está encargado de mostrar los resultados al usuario.
- **Módulo de guardado de datos:** este módulo se encarga de almacenar los resultados y la configuración de la ejecución.

Además, la herramienta realiza uso de determinadas librerías externas:

- **Swing:** librería estándar de Java empleada para la construcción de interfaces gráficos de usuario.
- **JFreeChart:** librería destinada a la visualización de gráficas. Se utiliza para mostrar los resultados de la ejecución de forma gráfica. Dispone de licencia LGPL, que permite su uso de forma libre. <http://www.jfree.org/jfreechart/>
- **Jackson:** librería destinada al manejo de datos en formato JSON. Se utiliza para el almacenamiento de resultados y configuración en ficheros. Dispone de licencia Apache 2.0, que permite su uso de forma libre. <http://wiki.fasterxml.com/JacksonHome>
- **Exp4j:** librería destinada a la evaluación de expresiones matemáticas. Emplea el algoritmo shunting yard desarrollado por Edsger Dijkstra. Se utiliza para la evaluación de la función de coste. Dispone de licencia Apache 2.0, que permite su uso de forma libre. <http://www.objecthunter.net/exp4j>

9.4.Diagrama de clases

A continuación se presenta el diagrama con las clases fundamentales del sistema. Este diseño se verá de forma más detallada en secciones siguientes.

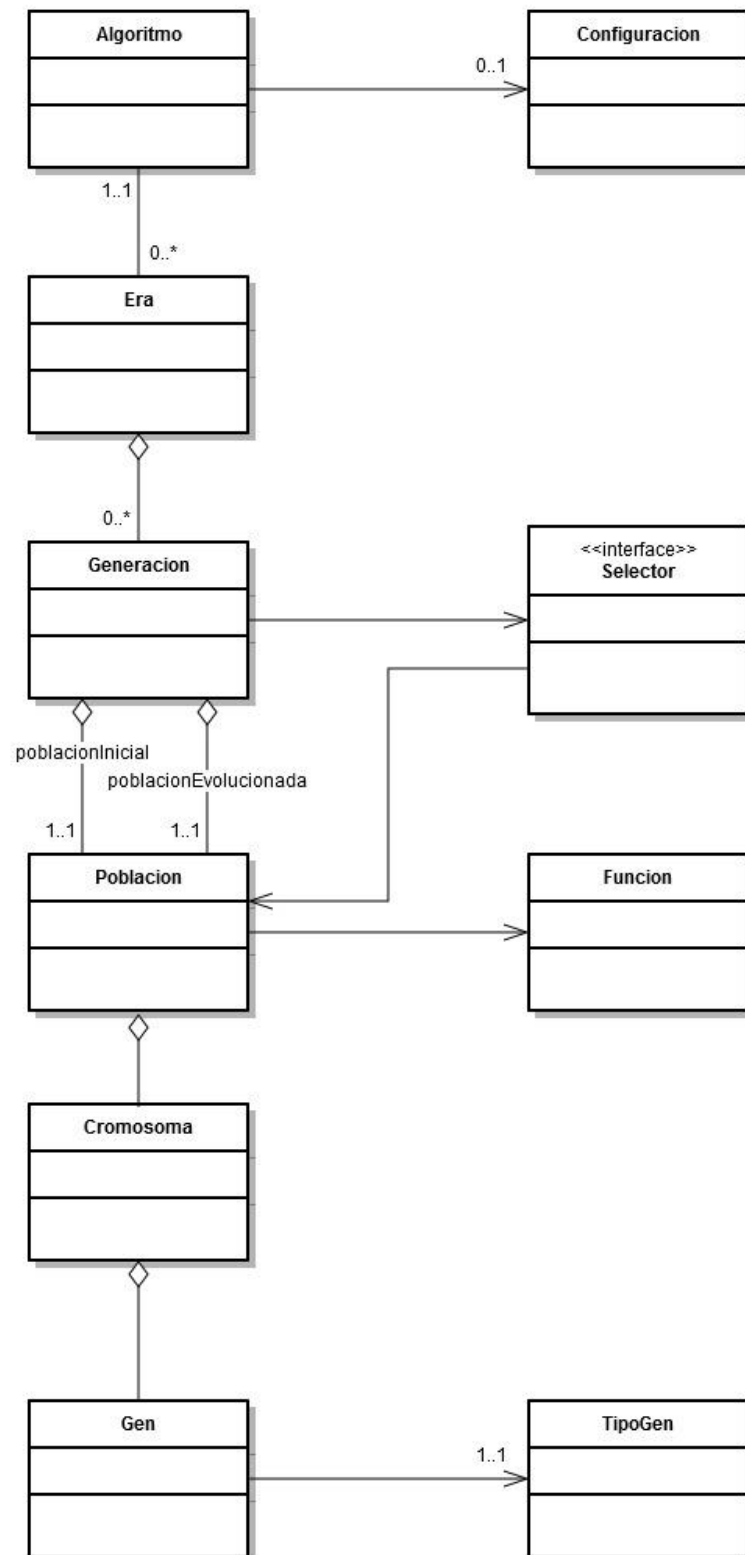


Figura 5 Diagrama de clases

Figura 6 Diagrama de clases

Se han identificado las siguientes clases:

- **Configuracion:** Se trata de una clase diseñada para almacenar los datos de configuración del algoritmo. Contiene número de eras y número de generaciones a ejecutar, tamaño de la población, probabilidad de cruce y de mutación y algoritmo de selección. Al ser necesaria únicamente una instancia de esta clase, se ha diseñado implementando el patrón Singleton.
- **Algoritmo:** Esta clase está diseñada para gestionar el flujo de ejecución del algoritmo genético, se encarga de controlar la ejecución de las sucesivas eras.
- **Era:** esta clase está diseñada para gestionar el flujo de ejecución de una era, se encarga de controlar la ejecución de las generaciones que forman parte de cada era.
- **Generacion:** Esta clase está diseñada para gestionar los pasos evolutivos que se dan en una generación. En su creación recibe una población inicial y aplica un paso en la evolución para crear una población evolucionada.
- **Poblacion:** Esta clase contiene los cromosomas que forman parte de cada población.
- **Cromosoma:** Esta clase representa un individuo de la población. Tiene un valor asociado a la función de evaluación.
- **Gen:** Esta clase representa cada uno de los genes que conforman un individuo. Dispone de dos atributos, un número real representando el valor del propio gen y su codificación, representada por la clase TipoGen.
- **TipoGen:** Esta clase representa la codificación de los genes, dispone de varios atributos: un nombre, un valor máximo, un valor mínimo y un valor de precisión.

- **Funcion:** Representa la función de evaluación o coste asociada al algoritmo. Realiza uso de la clase Expression procedente de la librería Exp4j.
- **Selector:** Se trata de una interfaz empleado para representar el algoritmo de selección empleado. Declara un método, seleccionar, que dada una población inicial debe devolver una población seleccionada para su evolución.

9.4.1. Modelo de datos

En el modelo de datos se incluyen aquellas clases cuyo objetivo fundamental es la de contener los datos de la aplicación. Se trata de clases con muy poco o ningún comportamiento asociado por lo que sus métodos son o bien constructores, o bien métodos de factoría estática o bien métodos de acceso a sus atributos. Estos métodos se han incluido dentro del paquete `com.uned.optimizadorga.elementos`

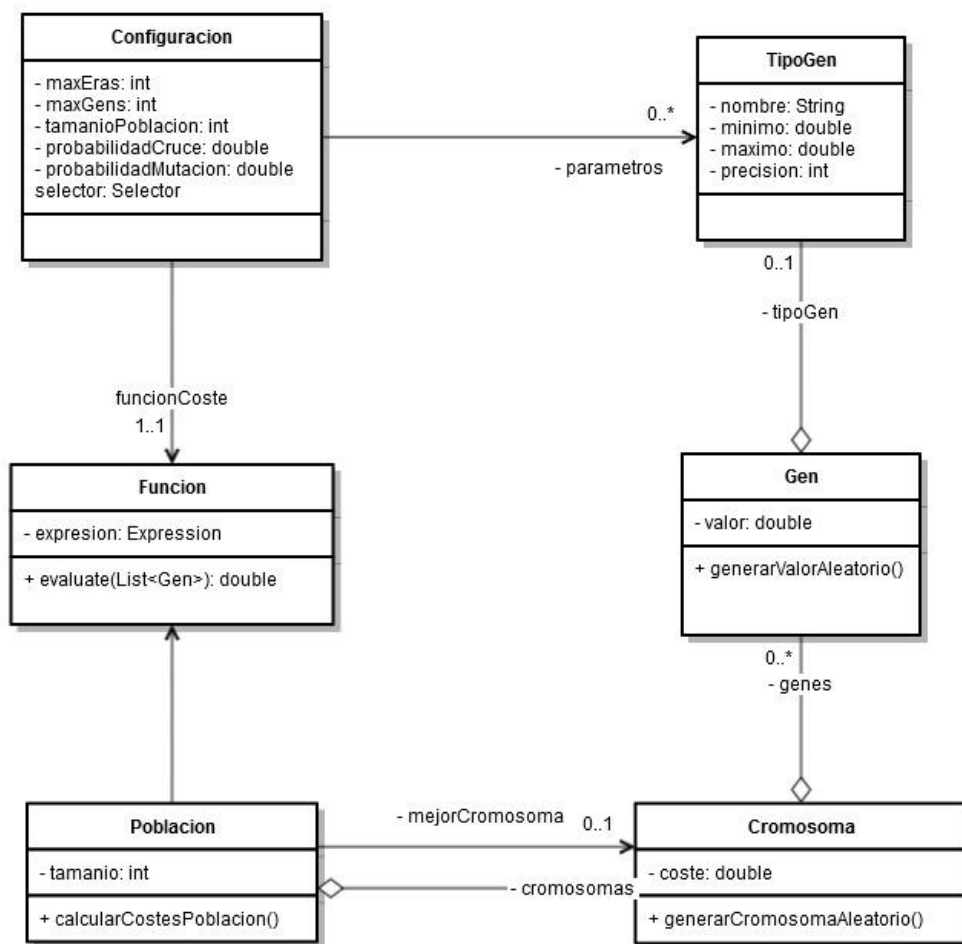


Figura 7 Modelo de datos

El paquete elementos contiene las clases que almacenan los datos de la aplicación. Las clases implicadas en esta área son:

- **TipoGen**
- **Gen**
- **Cromosoma**
- **Población**
- **Función**

- **Configuración**

9.4.2. Módulo de ejecución del algoritmo

El módulo de ejecución del algoritmo alberga las clases dedicadas a controlar la ejecución del algoritmo genético. Estas se encuentran situadas dentro del paquete com.uned.optimizadorga.algoritmo y sus correspondientes subpaquetes.

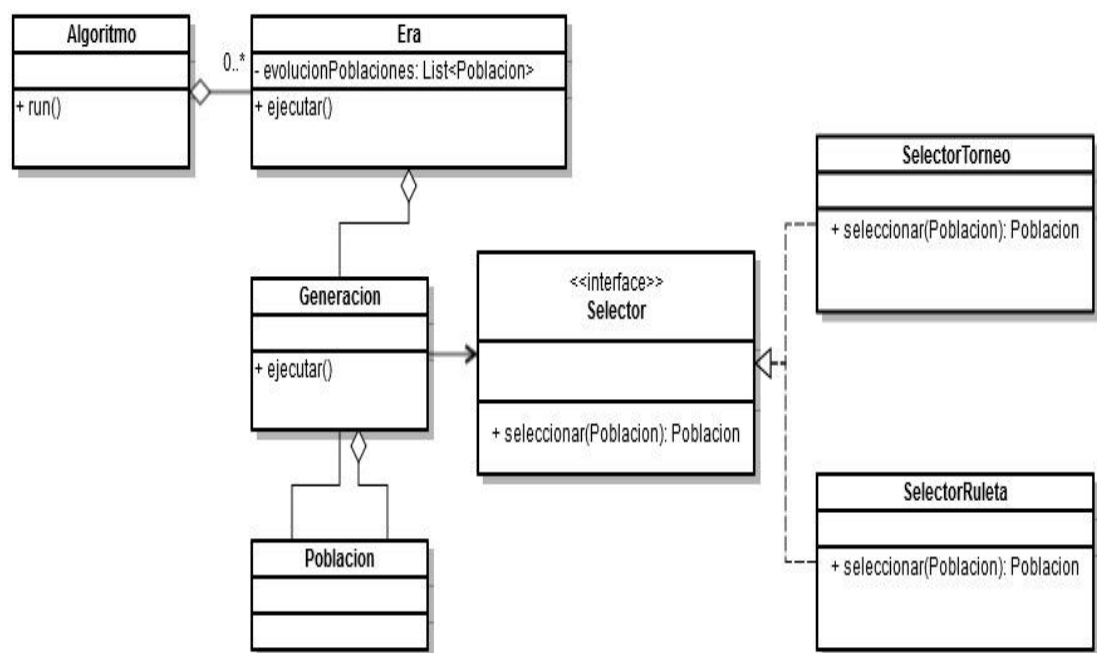


Figura 8 Módulo de ejecución

Las clases identificadas en el módulo que se encarga de la ejecución del algoritmo son:

- **Algoritmo:** Esta clase gestiona la implementación del algoritmo a alto nivel. Implementa la interfaz `Runnable` de java lo que le permite emplear un hilo de ejecución distinto de la clase que lo invoque. Mediante el método `run` crea las eras necesarias e invoca su ejecución.
- **Era:** esta clase gestiona la ejecución de una era en concreto. Dispone de un método `ejecutar`. Este método crea una población inicial con sus genes

inicializados con valores aleatorios. Posteriormente la evolución durante las generaciones especificadas.

- **Generación:** Esta clase contiene la implementación de cada generación. El estado que contiene consiste en una población inicial y la correspondiente población resultado de la evolución. Al igual que en la clase `Era`, dispone de un método público `ejecutar` que realiza la evolución correspondiente. Este método aplica los operadores de selección, cruce, mutación y elitismo que se hayan configurado.
- **Poblacion:** Esta clase es tanto la entrada como la salida de la evolución.
- **Selector:** Dado que se ha optado por implementar diversos operadores de selección, se ha decidido extraer el comportamiento necesario en la interfaz `Selector`. Esta obliga a implementar el método `seleccionar` a las clases que lo implementen. Se hace uso de polimorfismo para seleccionar la implementación adecuada en tiempo de ejecución.
- **SelectorRuleta:** Se trata de una implementación de la interfaz `Selector` que especifica el método de la ruleta.
- **SelectorTorneo:** Se trata de una implementación de la interfaz `Selector` que especifica una selección por torneo, el número de contendientes que emplea en el torneo es por defecto de dos.

9.4.3. Módulo de presentación de resultados

En principio la presentación de resultados debería ser extremadamente sencilla, pues bastaría con tomar los resultados de la ejecución y aplicar las transformaciones necesarias para mostrar las métricas de interés. Sin embargo, la necesidad de obtener datos sobre la evolución temporal del programa implica una serie de complicaciones que justifican la necesidad de este módulo. En él intervienen clases del paquete `com.uned.optimizadorga.algoritmo`.

Para mostrar datos sobre la evolución del programa se ha optado por la utilización del patrón Observer tanto sobre las eras como sobre el propio algoritmo. Para ello se han utilizado los interfaces AlgoritmoSubject, AlgoritmoObserver, EraSubject y EraObserver.

El patrón Observer define la relación entre un objeto y varios objetos dependientes de modo que cuando el estado del objeto cambie, todos los objetos dependientes sean notificados y actualizados. En este caso, se desea que cuando se produzca un cambio en el estado del cálculo (bien por la finalización del cálculo de una era como por la finalización del cálculo de una generación), se informe a la interfaz de usuario. La estructura básica de este patrón sería la siguiente:

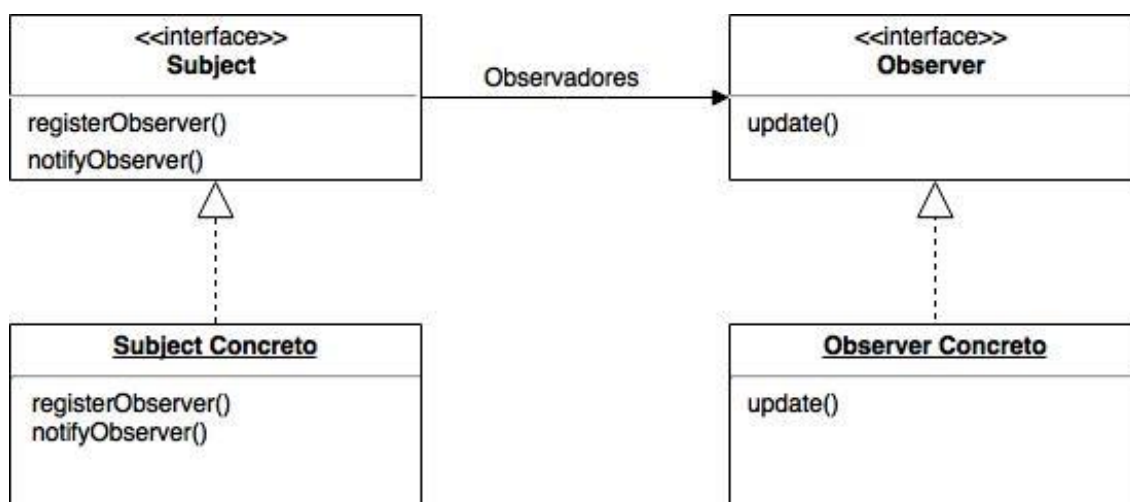


Figura 9 El patrón Observer

Cuando una clase desea mantenerse actualizada de los cambios de estado en una clase, interpreta el papel del Observer, y la clase cuyos cambios son observados se denomina Subject. La clase Subject proporciona el método `registerObserver` para que otras clases puedan incorporarse como observadores de su estado. Además dispone de un método `notifyObserver` que es el que se encargará de notificar cambios en su estado. Este método lo que hace es invocar al método `update` del observador informándole del nuevo estado.

Se explicará su funcionamiento concreto cuando se comenten las clases correspondientes.

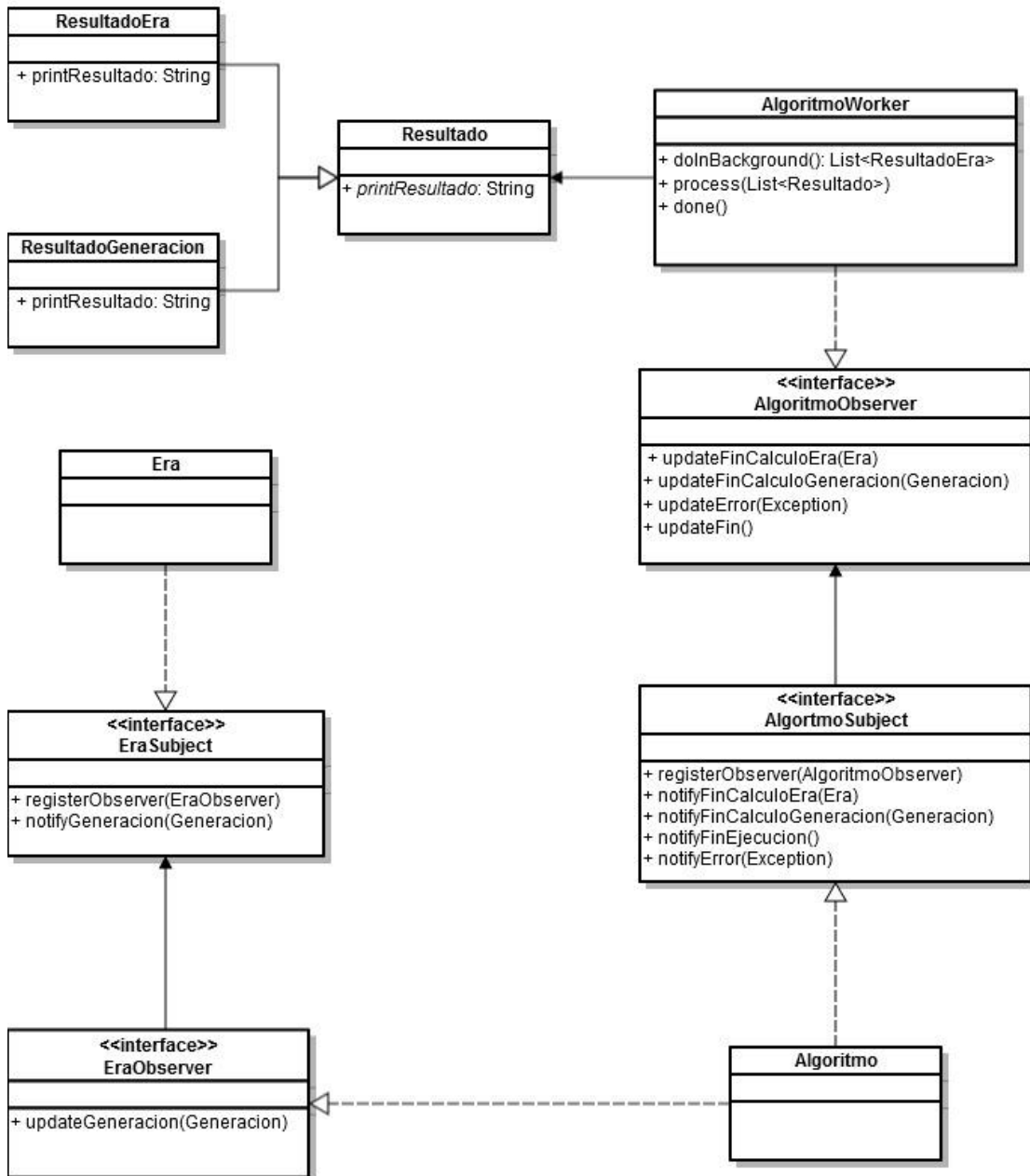


Figura 10 Módulo de presentación de resultados

Las clases que componen este módulo son:

- **Resultado:** Clase abstracta destinada a hacer de superclase para definir los resultados, tanto sean los resultados parciales correspondientes al cálculo de una generación como los correspondientes al cálculo de una era. Dispone de un método denominado `printResultado` que debe ser definido en las clases hijas y obliga a definir el modo en que se mostrará el resultado.
- **ResultadoEra:** Implementación de la clase `Resultado` correspondiente a los resultados procedentes del procesamiento de una era.
- **ResultadoGeneracion:** Implementación de la clase `Resultado` correspondiente a los resultados procedentes de la evolución de una generación.
- **EraSubject:** Interfaz empleada para la implementación del `Subject` en el patrón `Observer`. En este caso, el aspecto que se desea observar es la evolución del cálculo dentro de una era, que consiste en la finalización del cálculo de una generación. Lo que hace que la clase `Era` deba implementar esta interfaz.
- **EraObserver:** Interfaz empleada para la implementación de los observadores en el patrón `Observer`. En este caso, la clase interesada en mantenerse actualizada de los cambios en una era es la clase que ha lanzado su ejecución, la clase `Algoritmo`.
- **AlgoritmoSubject:** Interfaz empleada para la implementación del `Subject` en el patrón `Observer`. En este caso, los aspectos a observar serán los correspondientes a la evolución del algoritmo: la finalización del cálculo de una generación, la finalización del cálculo de una era, la finalización de la ejecución o que se haya producido un error. Se proporcionan métodos para notificar cada uno de estos eventos. La clase que implementa esta interfaz es el `Algoritmo`.
- **AlgoritmoObserver:** Interfaz empleada para la implementación de los observadores en el patrón `Observer`. En este caso, la clase interesada en

mantenerse actualizada de los cambios en una era es la clase que ha lanzado su ejecución, la clase `AlgoritmoWorker`.

- **Era:** Esta clase debe implementar la interfaz `EraSubject` para informar de los resultados parciales obtenidos en el cálculo de una generación.
- **Algoritmo:** Esta clase debe implementar la interfaz `EraObserver` para recoger los resultados parciales obtenidos en el cálculo de una generación. Además debe implementar la interfaz `AlgoritmoSubject` para informar de los resultados parciales obtenidos en el cálculo de la era, de errores en la ejecución o del fin del cálculo.
- **AlgoritmoWorker:** Esta clase se utiliza para iniciar el cálculo en un hilo de ejecución diferente del correspondiente a la interfaz de usuario. Esta clase extiende la clase de utilidad `SwingWorker`, proporcionada por la librería estándar `Swing`, que proporciona métodos de utilidad para actualizar a la interfaz de usuario con datos correspondientes a la evolución de la ejecución, y a los datos finales resultado del cálculo. Para poder informar a la interfaz de estos datos, debe establecerse como un observador sobre el algoritmo implementando la interfaz `AlgoritmoObserver`.

9.4.4. Módulo de interfaz gráfica de usuario

La interfaz gráfica está compuesta por una serie de clases que utilizan los componentes definidos en la librería `Swing` de Java. Estas clases se encuentran situadas en el paquete `com.uned.optimizadorga.gui`

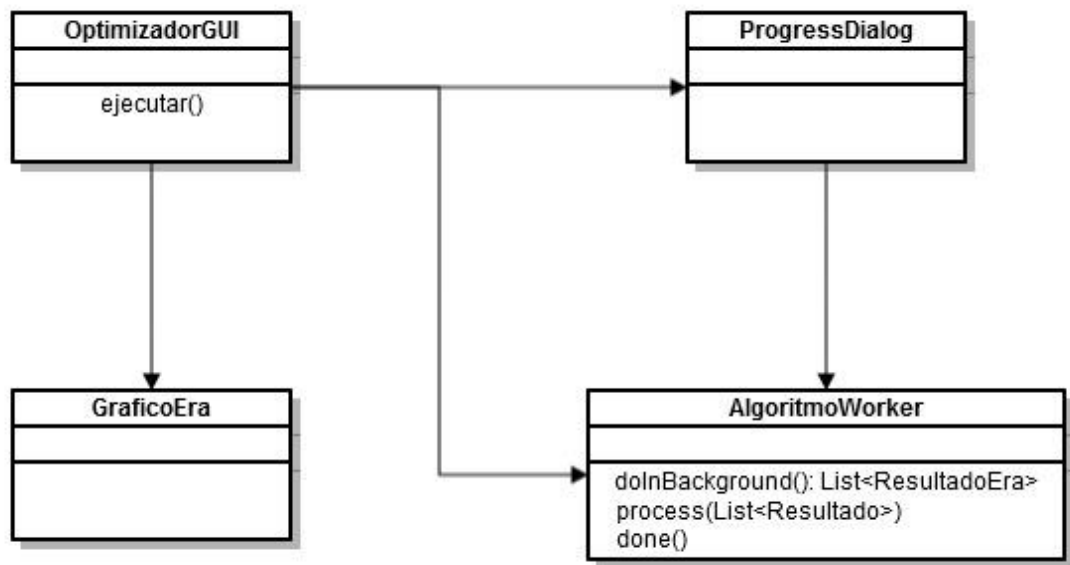


Figura 11 Módulo de interfaz gráfica

- **OptimizadorGUI:** Se trata de la ventana principal de la aplicación. Constituye el punto de entrada a la aplicación y se contiene tanto los elementos necesarios para configurar la ejecución como los resultados finales del cálculo.
- **ProgressDialog:** Se trata de una ventana que informa sobre el progreso del cálculo. Se abre al inicio de la ejecución y muestra los resultados parciales que se van calculando.
- **GraficoEra:** Se trata de una clase empleada para mostrar los datos detallados de los resultados obtenidos en el cálculo de una era.
- **AlgoritmoWorker:** Esta clase es la que actúa de nexo entre el algoritmo en ejecución y la interfaz gráfica.

9.4.5. Módulo de guardado de datos

El módulo de guardado de datos es el más sencillo de todos. El objetivo de este es almacenar los datos de configuración del sistema, y en caso de existir, los resultados de la ejecución. Además proporciona mecanismos para poder recuperar estos datos de un

fichero externo. Se trata de operaciones que afectan a la interfaz de usuario y por este motivo se encuentra situado en clases del paquete `com.uned.optimizadorga.gui`

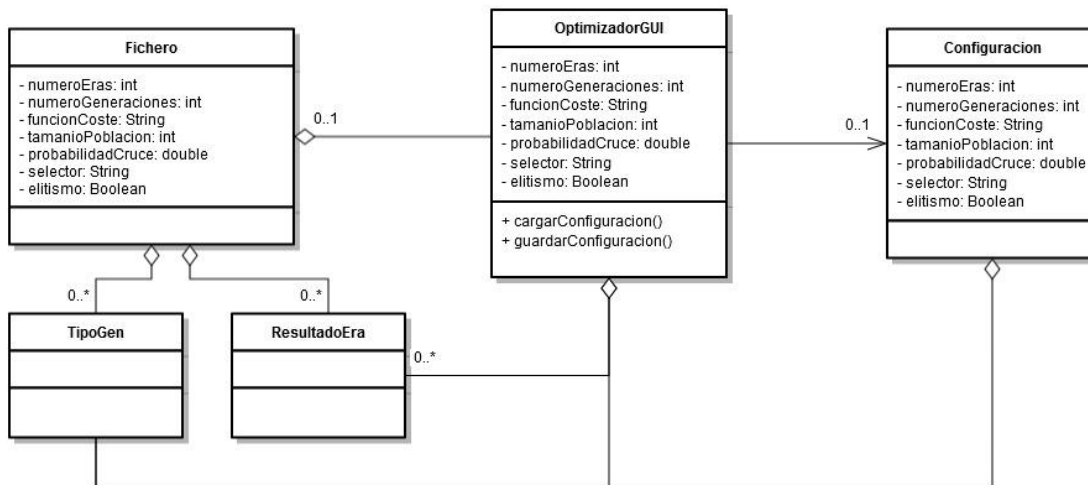


Figura 12 Módulo de guardado de datos

- **OptimizadorGUI:** La clase que gestiona la interfaz de usuario. En esta clase se incluyen los métodos destinados a guardar y cargar datos de un fichero. Además puede contener los datos de configuración si no se ha realizado una ejecución del algoritmo en el momento del guardado.
- **TipoGen:** La clase que contiene los datos de codificación de los cromosomas. Puede pertenecer a la clase OptimizadorGUI o a la clase Configuración en función de si se ha ejecutado el algoritmo.
- **ResultadoEra:** Los resultados de una ejecución del algoritmo.
- **Configuracion:** Esta clase contiene los datos de configuración una vez que el algoritmo se ha ejecutado.
- **Fichero:** En esta clase se almacenan todos los datos, tanto de configuración como resultado de una ejecución. Haciendo uso de la librería Jackson, se almacenará en un fichero en formato JSON.

9.5. Flujo de ejecución

En esta sección se pretende mostrar los flujos que sigue la ejecución de los distintos procesos que forman la herramienta.

9.5.1. Flujo de ejecución general

La ejecución general comienza desde la interfaz de usuario, una vez que el usuario ha establecido los datos requeridos por el algoritmo y pulsa en el botón ejecutar.

Desde la interfaz de usuario se crean los objetos necesarios para la ejecución y se pasan a la clase AlgoritmoWorker.

El objeto AlgoritmoWorker inicia el hilo que ejecutará el algoritmo. Durante su ejecución, el algoritmo envía actualizaciones de forma asíncrona sobre su evolución al objeto AlgoritmoWorker, que a su vez se lo reenvía a un objeto ProgressDialog que muestra datos sobre la evolución del algoritmo.

Una vez que finalice la ejecución, el algoritmo se lo notifica al AlgoritmoWorker, que a su vez envía los resultados a la interfaz de usuario para que los procese y los muestre por pantalla.

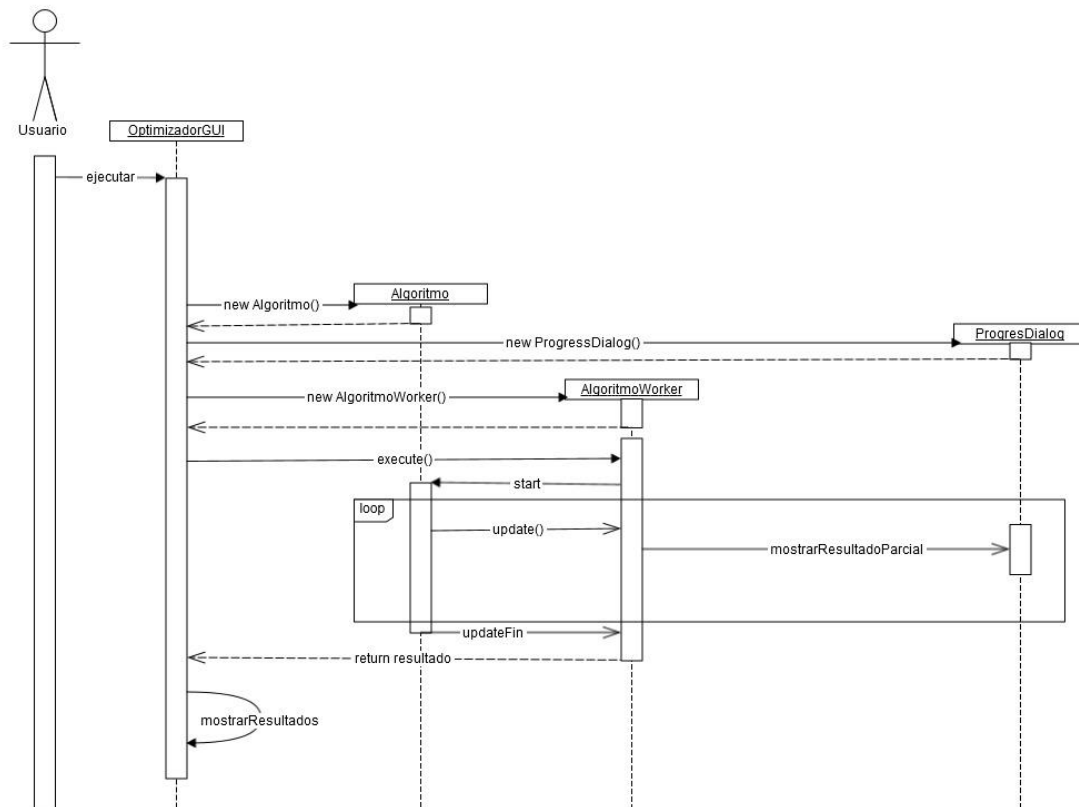


Figura 13 Flujo de la ejecución general

9.5.2. Flujo de ejecución del algoritmo genético

El siguiente diagrama de secuencia muestra la evolución de la ejecución del algoritmo genético. Como puede verse, se ejecutan en bucle todas las eras, asimismo, dentro de cada era se ejecutan en bucle todas las generaciones.

La clase generación recibe una población inicial y durante su ciclo de vida aplica los operadores definidos para obtener una población evolucionada. Una vez finaliza cada generación, es la era a la que pertenece la que se encarga de notificar la finalización para poder computar resultados parciales.

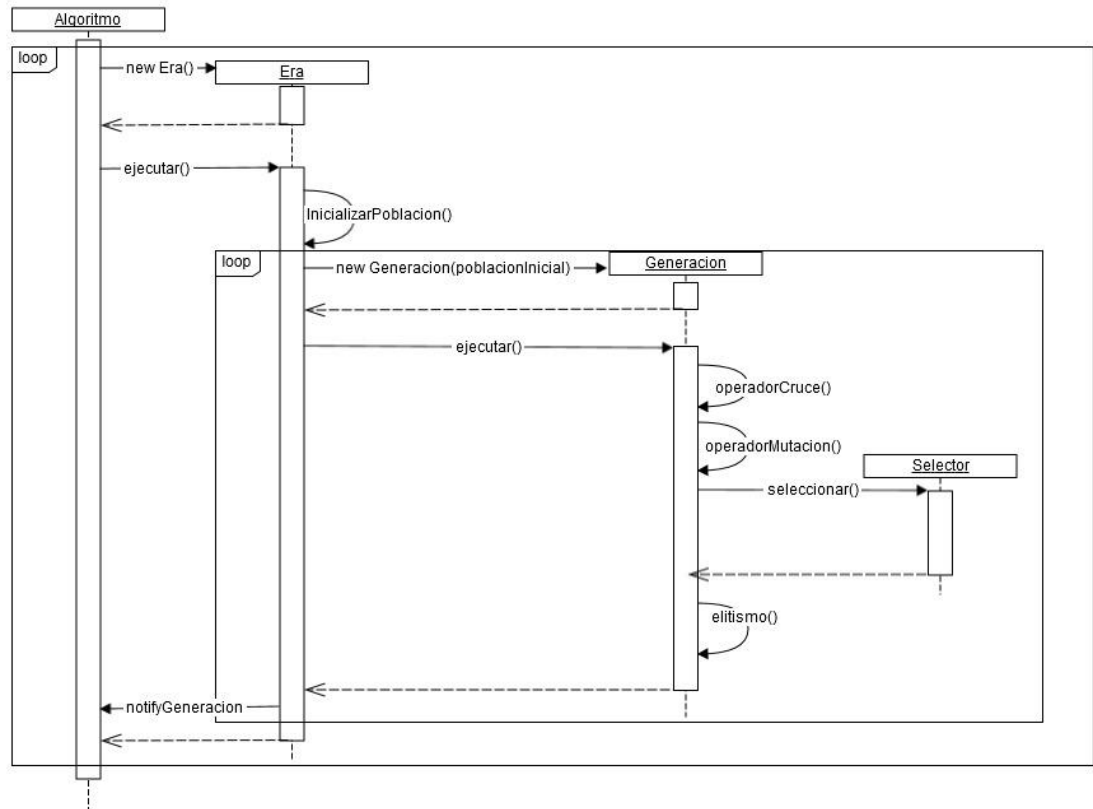


Figura 14 Flujo de ejecución del algoritmo genético

9.5.3. Cancelación de la ejecución

El siguiente diagrama de secuencia muestra el evento de la cancelación de una ejecución. El proceso comienza con el usuario pulsando el botón cancelar del diálogo que muestra el progreso. El controlador de eventos de este botón invoca el método cancel heredado de la clase `SwingWorker` que cambia el estado de la clase `AlgoritmoWorker`. La clase `AlgoritmoWorker` se encuentra en un bucle comprobando su estado. En el momento en que detecta este cambio, envía una interrupción a la clase `Algoritmo` para cancelar su ejecución.

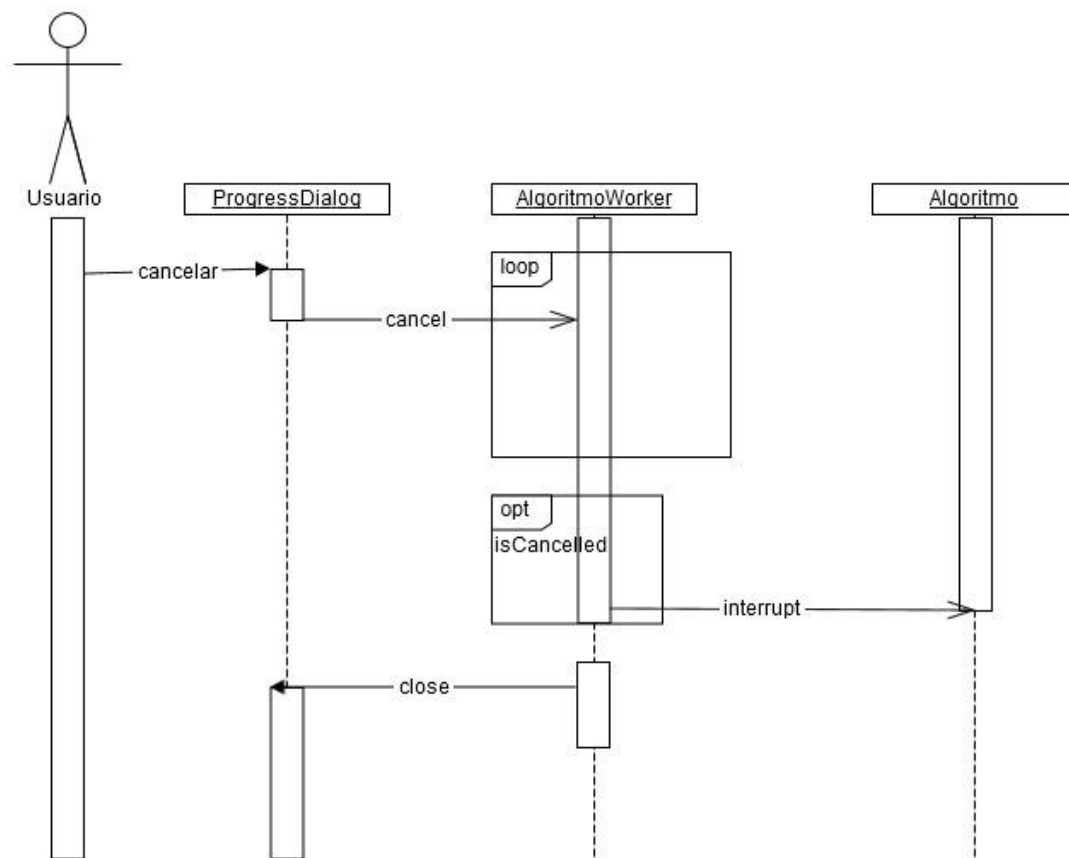


Figura 15 Cancelación de la ejecución

9.5.4. Presentación de resultados parciales

El siguiente diagrama de flujo muestra el flujo de la presentación de los resultados parciales. Se ha implementado el patrón Observer tanto entre la clase era y la clase algoritmo para la notificación de los resultados parciales provenientes de una generación como entre la clase algoritmo y la clase AlgoritmoWorker para la notificación de los resultados parciales provenientes de la generación y de la ejecución de una era.

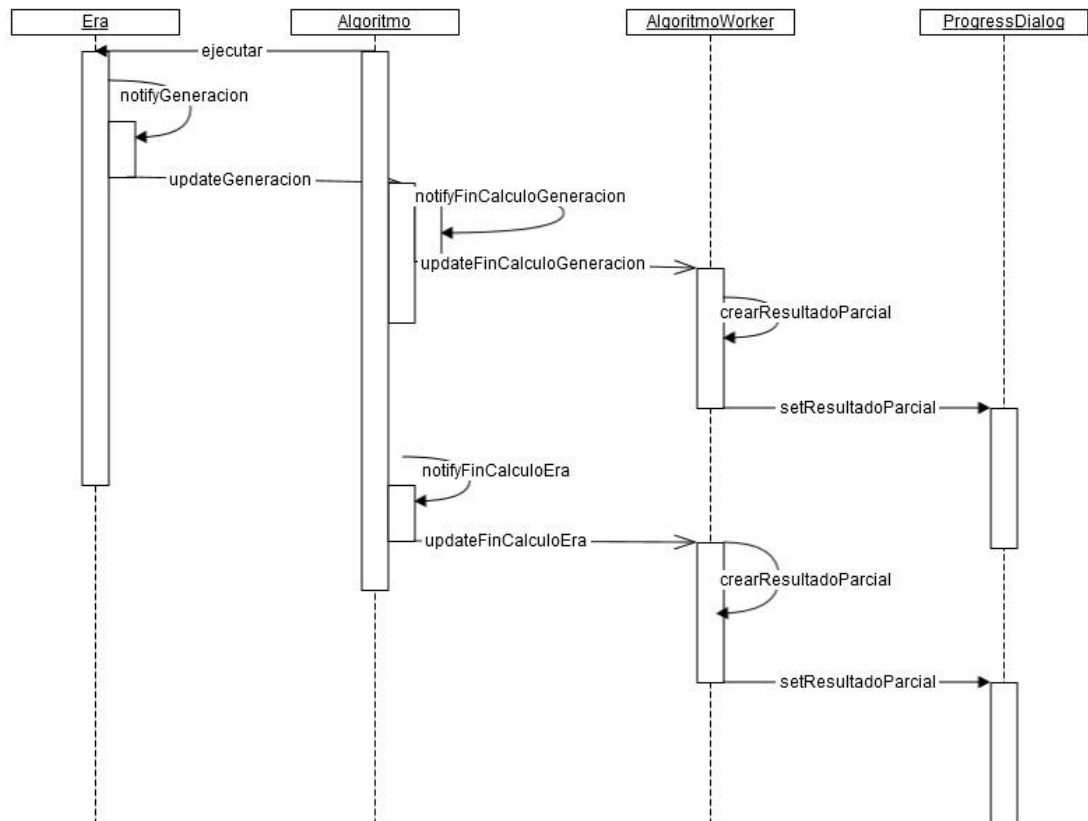


Figura 16 Presentación de resultados parciales

9.6. Interfaz gráfica de usuario

Para implementar la interfaz gráfica de usuario, se ha empleado la librería Swing, incluida en el estándar de Java. Esta librería incluye los componentes gráficos y utilidades de soporte necesarias para gestionar la interacción con el usuario.

El objetivo que se ha buscado en el diseño de la interfaz es la simplicidad, por este motivo se han intentado reducir al mínimo el número de ventanas existentes. La herramienta se ha diseñado compuesta por una ventana principal y dos ventanas secundarias:

- Una ventana principal de la aplicación.

- Una ventana modal para detallar el progreso del cálculo.
- Una ventana modal para detallar los resultados de una era.

9.6.1. Ventana principal de la aplicación

La ventana principal de la aplicación está destinada a configurar la ejecución, iniciar la ejecución del algoritmo y mostrar los resultados cuando estén disponibles. Esta ventana se ha compuesto por varios paneles, cada uno destinado a una funcionalidad distinta.

1. Panel de configuración del algoritmo.
2. Panel de configuración de la función de coste.
3. Panel de parámetros.
4. Panel de edición de parámetros.
5. Panel de presentación de resultados.

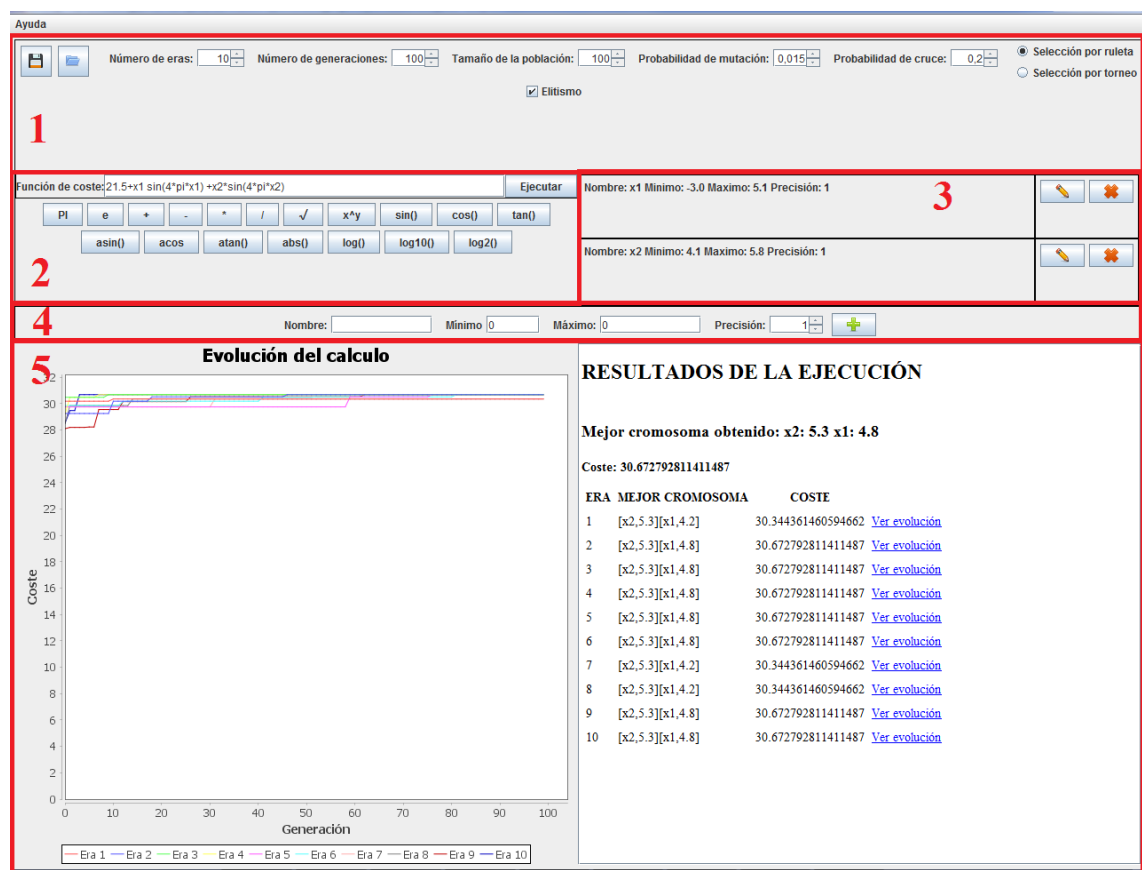


Figura 17 Ventana principal de la aplicación

9.6.2. Ventana de progreso del cálculo

La ventana de progreso se muestra mientras el cálculo se está realizando. Se trata de una ventana modal, que impide el acceso a otras secciones de la herramienta mientras el algoritmo se está ejecutando. Proporciona datos parciales sobre el cálculo y sobre progreso de la ejecución. Además proporciona la posibilidad de cancelar la ejecución del algoritmo. Esta ventana está compuesta por cuatro secciones:

1. Una sección que proporciona información sobre el progreso y tiempo transcurrido durante la ejecución.
2. Una sección que muestra un botón para cancelar la ejecución.
3. Una sección que muestra información sobre la última era calculada.

- Una sección que muestra información sobre la última generación calculada.



Figura 18 Ventana de progreso

9.6.3. Ventana de resultados de una era

En esta ventana se muestra, en detalle y de forma gráfica, la evolución del cálculo realizado para una era, la evolución del valor medio del coste para la población y la evolución de la desviación típica del coste en la población.

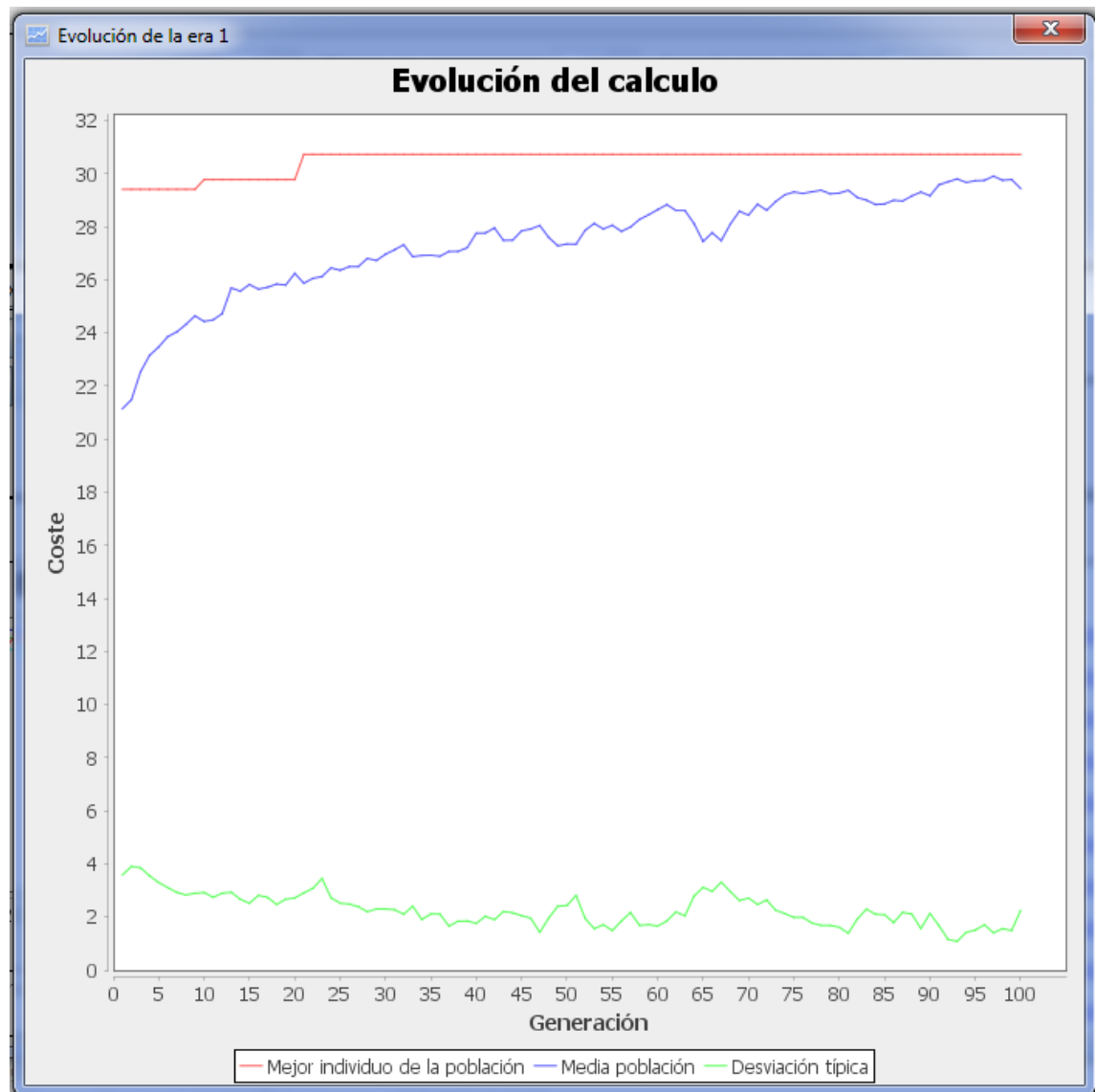


Figura 19 Ventana de resultados de una era

10. Funcionamiento de la aplicación

10.1. Requisitos previos

Dado que la aplicación está construida en el lenguaje Java, es requisito para su ejecución disponer de una máquina virtual (JRE) adecuada. La versión necesaria debe ser igual o superior a la 1.7.

10.2. Acceso a la aplicación

Para ejecutar la aplicación debe ejecutarse el fichero en formato jar en el cual se encuentra empaquetada. Este fichero se denomina optimizadorGA.jar.

El modo de ejecución puede ser bien desde el gestor gráfico de ficheros del sistema operativo haciendo doble click en el fichero o bien mediante línea de comandos. El comando adecuado es: `java -jar optimizadorGA.jar`.

Se mostrará la ventana principal de la aplicación.

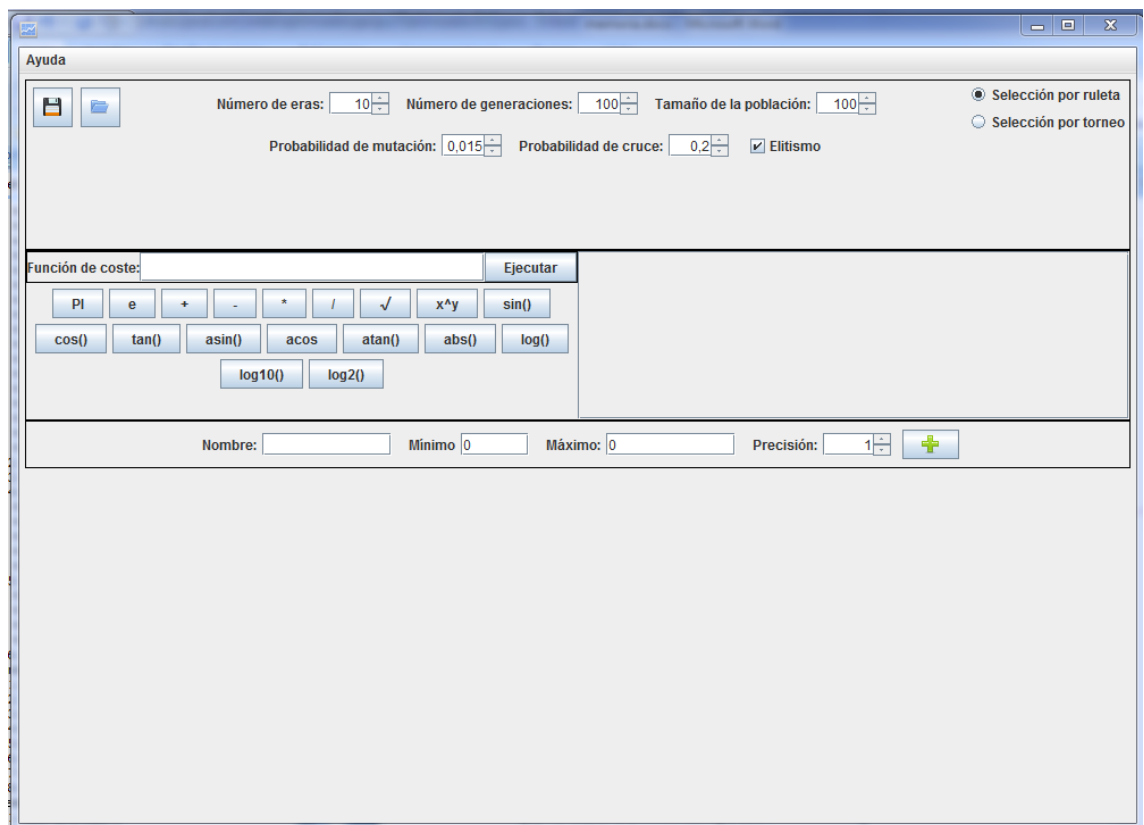


Figura 20 Ventana principal de la aplicación

10.3. *Introducción de la función de coste*

Debe introducirse la función de evaluación en el área reservada para ello. En esta misma área existen una serie de botones destinados a ayudar a introducir las funciones, operadores y constantes predefinidas por la aplicación.

Las funciones y operadores matemáticas predefinidas son:

Función/Operador	Símbolo
Suma	+

Resta	-
Multiplicación	*
División	/
Raíz cuadrada	$\sqrt{}$
Potencia	\wedge
Coseno	cos()
Seno	sin()
Tangente	tan()
Arcocoseno	acos()
Arcotangente	atan()
Arcoseno	asin()
Valor absoluto	abs()
Logaritmo neperiano	log()
Logaritmo decimal	log10()

Logaritmo en base 2	log2()
---------------------	--------

Tabla 1 Funciones y operadores predefinidos

Las constantes matemáticas predefinidas en la aplicación son las siguientes:

Constante	Símbolo
Número Pi	PI
Número de Euler	E

Tabla 2 Constantes matemáticas predefinidas


Figura 21 Área de la función de coste

10.4. Codificación del vector de parámetros

Para el correcto funcionamiento de la ejecución, debe especificarse una codificación para el vector de parámetros. La codificación de cada parámetro debe introducirse en el área de introducción de nuevos parámetros.

Figura 22 Área de introducción de nuevos parámetros



Al pulsar el botón , se valida la corrección del nuevo parámetro. En caso de existir alguna incorrección se muestra un mensaje informativo por pantalla. Si la codificación introducida para el parámetro es correcta, se añade el nuevo parámetro a la lista de parámetros declarados.






Nombre: x Minimo: -5.0 Maximo: 6.0 Precisión: 1	 
Nombre: y Minimo: 0.0 Maximo: 10.0 Precisión: 1	 


Figura 23 Listado de parámetros declarados

Las validaciones que se realizan sobre un nuevo parámetro son las siguientes:

- El nombre no es vacío.
- El nombre no comienza por un dígito.
- El nombre no coincide con el de otro parámetro ya existente.
- El nombre no contiene el mismo símbolo que algún operador, función o constante predefinida.
- El límite superior no es menor que el límite inferior.

Una vez que se ha declarado un parámetro, este puede eliminarse, para ello debe

pulsarse el botón  situado en la misma fila que el parámetro correspondiente.

Asimismo, puede modificarse el parámetro, para ello, debe pulsarse el botón  situado en la misma fila que el parámetro correspondiente. Al pulsarse el botón, se cargan los datos del parámetro a modificar en el área de introducción de parámetros.

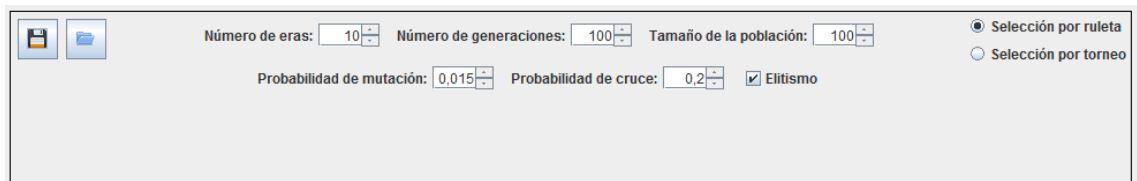
10.5. Configuración de la ejecución

La aplicación inicialmente contiene una configuración por defecto. Esta configuración es la siguiente:

Número de eras	10
Número de generaciones	100
Tamaño de la población	100
Probabilidad de mutación	0.015
Probabilidad de cruce	0.2
Elitismo	Activado
Tipo de selección	Selección por ruleta

Tabla 3 Configuración por defecto

La configuración puede modificarse en el área de configuración de la aplicación.




The screenshot shows a configuration window with the following settings:

- Number of eras: 10
- Number of generations: 100
- Population size: 100
- Mutation probability: 0.015
- Crossover probability: 0.2
- Elitism: ☒ Elitismo
- Selection type: ☒ Selección por ruleta, ☐ Selección por torneo

Figura 24 Área de configuración de la aplicación

10.6. Ejecución

Una vez configurada la aplicación, puede lanzarse su ejecución. Para ello, se pulsa en el

botón  situado en el área correspondiente a la función de coste.

Al pulsar el botón de ejecución, se valida que la configuración de la función de coste es correcta. Los aspectos que se validan de la función de coste son los siguientes:

- Demasiados puntos decimales en un valor numérico.
- Concuerdan Los símbolos de paréntesis que se abren con los que se cierran.
- Los operadores no se aplican sobre un número o un parámetro.
- El uso de parámetros no declarados.

Una vez realizada la validación, comienza la ejecución del algoritmo. Se muestra una nueva ventana que proporciona información sobre el progreso del algoritmo.

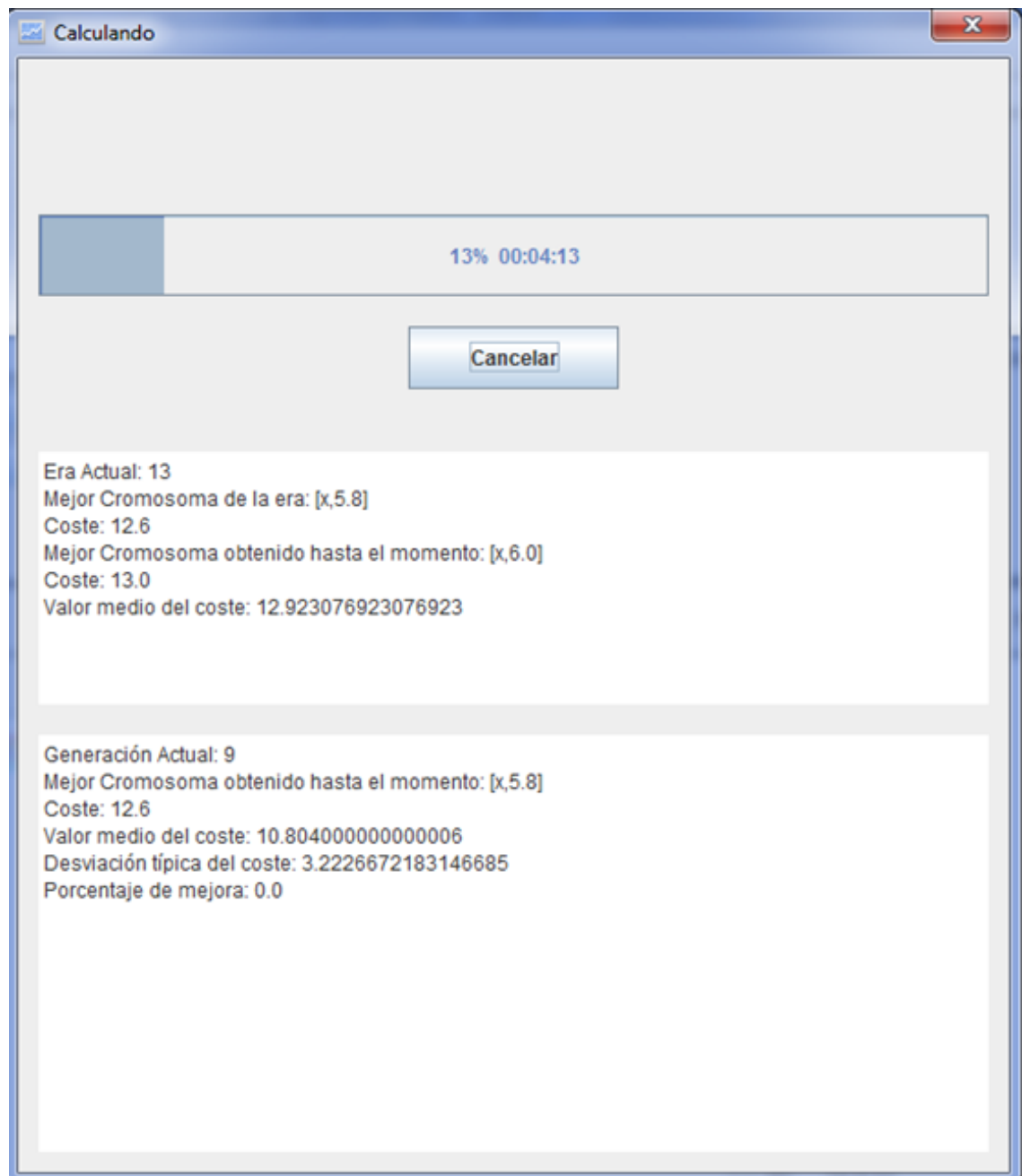


Figura 25 Ventana de progreso

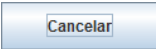
La información que se muestra durante la ejecución es la siguiente:

- Porcentaje completado de la ejecución.
- Tiempo de ejecución transcurrido.

- Número de era que se está calculando.
- Mejor cromosoma de la era actual.
- Coste del mejor cromosoma obtenido en la era actual.
- Mejor cromosoma obtenido en todas las eras.
- Coste del mejor cromosoma obtenido durante toda la ejecución.
- Valor medio del coste en la era actual.
- Número de generación que se está calculando.
- Mejor cromosoma obtenido en la generación.
- Coste del mejor cromosoma obtenido en la generación.
- Valor medio del coste de los cromosomas de la población evolucionada.
- Desviación típica del coste de los cromosomas de la población evolucionada.
- Porcentaje de mejora respecto del mejor cromosoma de la generación anterior.

Esta ventana se cierra automáticamente en el momento en que finalice la ejecución.

10.7. Cancelación de la ejecución

La ejecución del algoritmo puede cancelarse en todo momento. Para ello debe pulsarse el botón  presente en la ventana de progreso. Al cancelar la ejecución, se cierra la ventana de progreso y se detiene la ejecución.

10.8. Resultados de la ejecución

Al finalizar la ejecución del algoritmo, bien porque este llegue a su fin o bien porque se cancele su ejecución, se muestran en la ventana principal los resultados obtenidos durante el cálculo. Estos resultados se muestran en la sección destinada para ello.

Esta sección se compone de dos subsecciones. Una parte muestra de forma gráfica la evolución del mejor cromosoma obtenido en cada una de las eras a lo largo de todas las generaciones. La otra sección muestra el mejor cromosoma obtenido durante toda la ejecución así como el mejor cromosoma obtenido en cada una de las eras y un enlace para mostrar detalles sobre la evolución de cada era en particular.

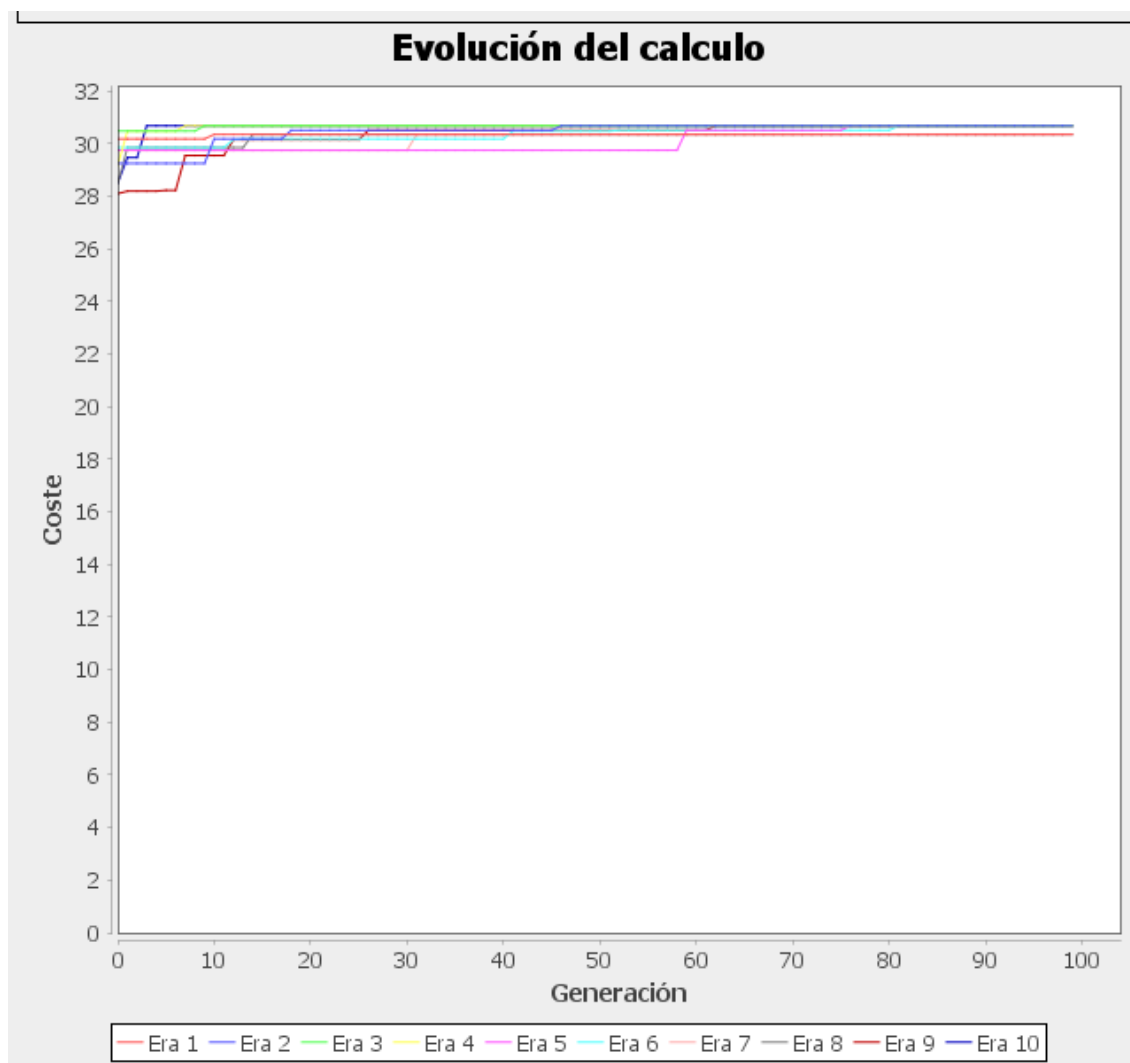


Figura 26 Sección gráfica de los resultados

RESULTADOS DE LA EJECUCIÓN

Mejor cromosoma obtenido: x2: 5.3 x1: 4.8

Coste: 30.672792811411487

ERA	MEJOR CROMOSOMA	COSTE
1	[x2,5.3][x1,4.2]	30.344361460594662 Ver evolución
2	[x2,5.3][x1,4.8]	30.672792811411487 Ver evolución
3	[x2,5.3][x1,4.8]	30.672792811411487 Ver evolución
4	[x2,5.3][x1,4.8]	30.672792811411487 Ver evolución
5	[x2,5.3][x1,4.8]	30.672792811411487 Ver evolución
6	[x2,5.3][x1,4.8]	30.672792811411487 Ver evolución
7	[x2,5.3][x1,4.2]	30.344361460594662 Ver evolución
8	[x2,5.3][x1,4.2]	30.344361460594662 Ver evolución
9	[x2,5.3][x1,4.8]	30.672792811411487 Ver evolución
10	[x2,5.3][x1,4.8]	30.672792811411487 Ver evolución

Figura 27 Mejores cromosomas de la ejecución

10.9. Detalles de la evolución de una era

Al pulsar en el enlace “Ver evolución”, situado junto al mejor cromosoma de cada era, se abre una ventana que muestra detalles sobre la evolución del cálculo de la era. En concreto, muestra de forma gráfica, a lo largo de las generaciones, el mejor cromosoma obtenido, la media de la población y la desviación típica.

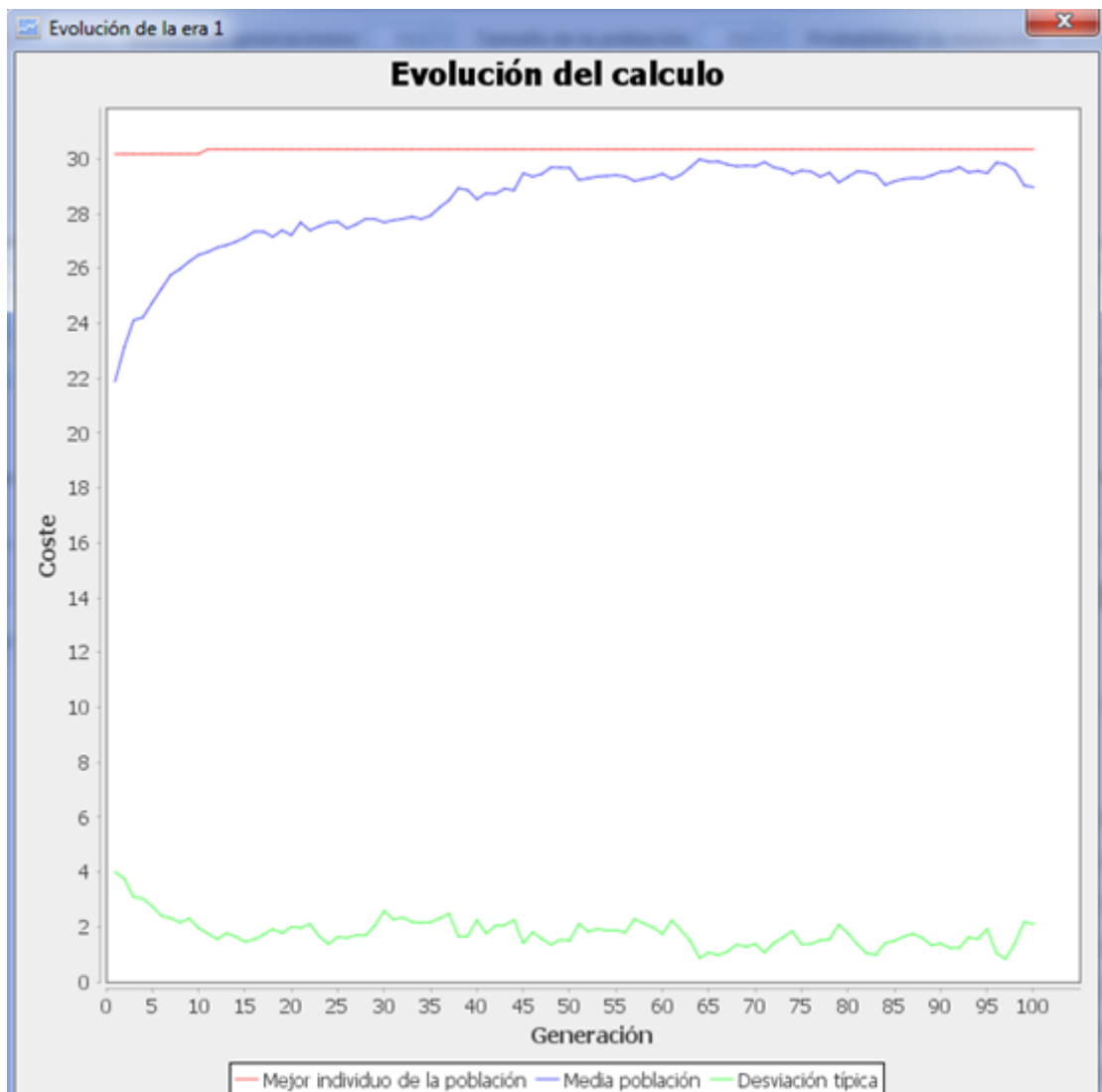



Figura 28 Evolución del cálculo de una era

10.10. Guardado de datos

Pueden almacenarse los datos de la aplicación pulsándose el icono . Al pulsar en el botón se abre un diálogo para seleccionar el archivo en el que se desea guardar.

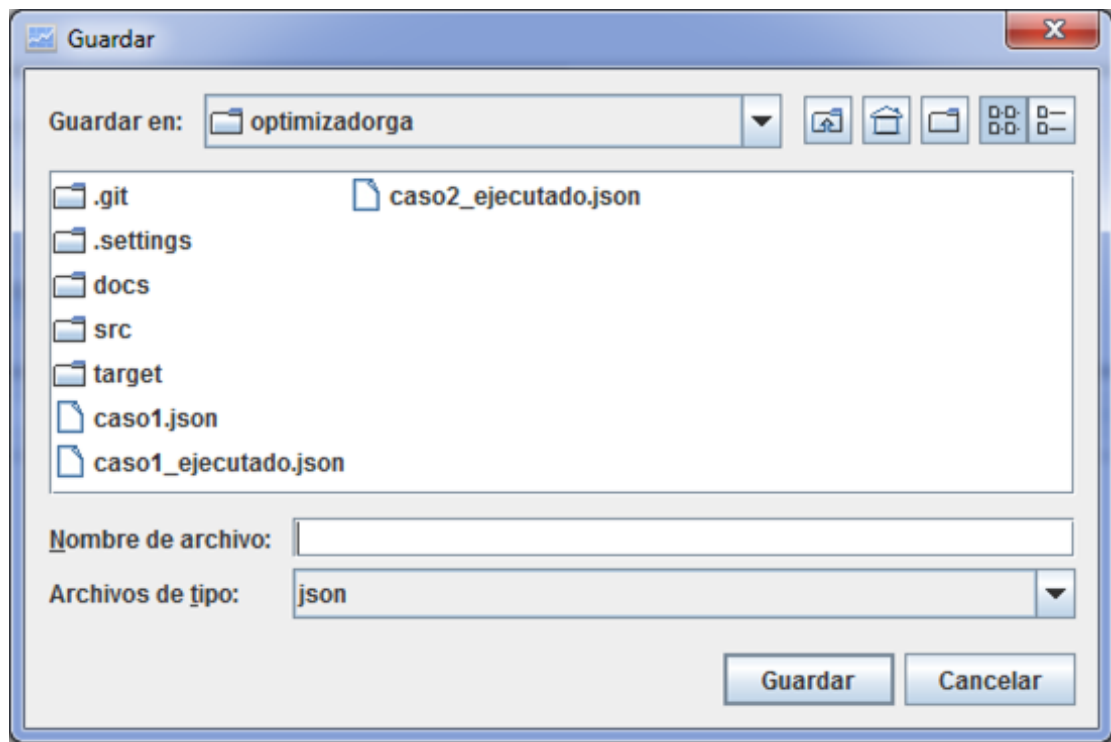


Figura 29 Diálogo de guardado de datos

Al introducir un nombre para el archivo y pulsar en el botón Guardar, se almacenan, en un fichero en formato json, los datos de la aplicación. Si se ha realizado una ejecución, también se almacenarán los resultados correspondientes a los cálculos realizados.

10.11. Carga de datos guardados

Para recuperar los datos almacenados en un fichero de datos, debe pulsarse el icono



. Al pulsarse el botón, se abre un diálogo para seleccionar el archivo.

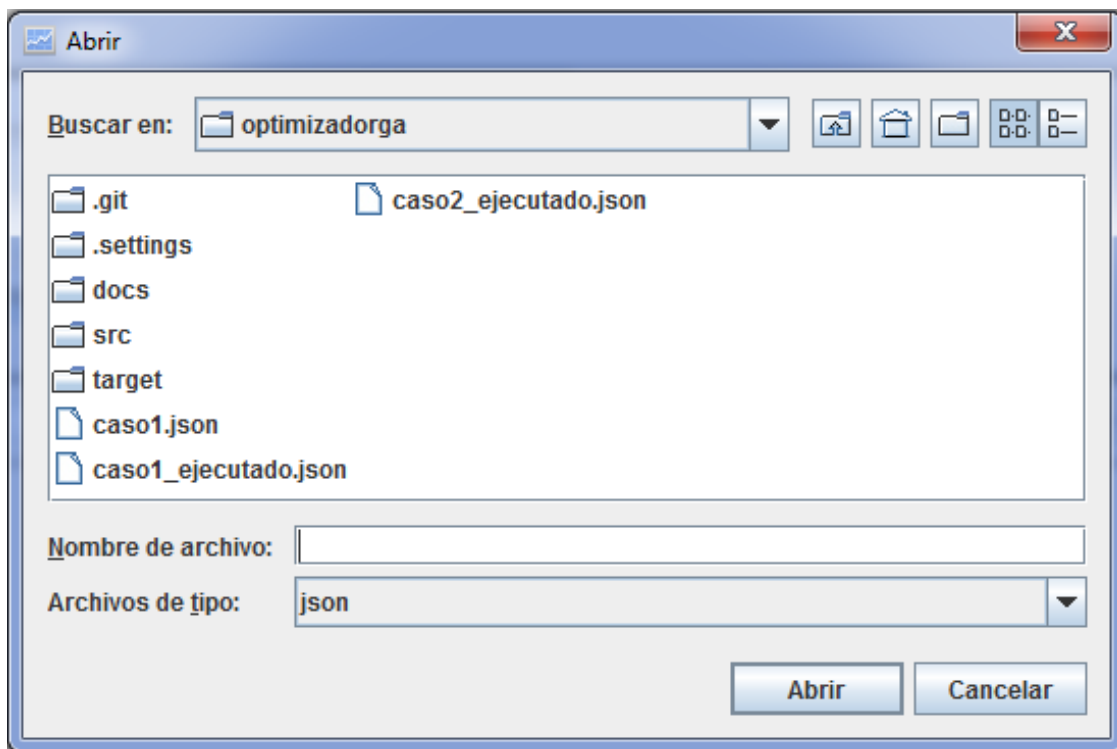


Figura 30 Diálogo de apertura de datos

Al seleccionar el archivo y pulsar en el botón Abrir, se cargan los datos correspondientes en la aplicación.

10.12. Ayuda de la aplicación

El manual de usuario es accesible en todo momento desde la propia aplicación. Para acceder, debe pulsarse en la barra de herramientas situada en la parte superior y acceder al apartado ayuda/ayuda. Al pulsarse esta opción se abre el manual de usuario.

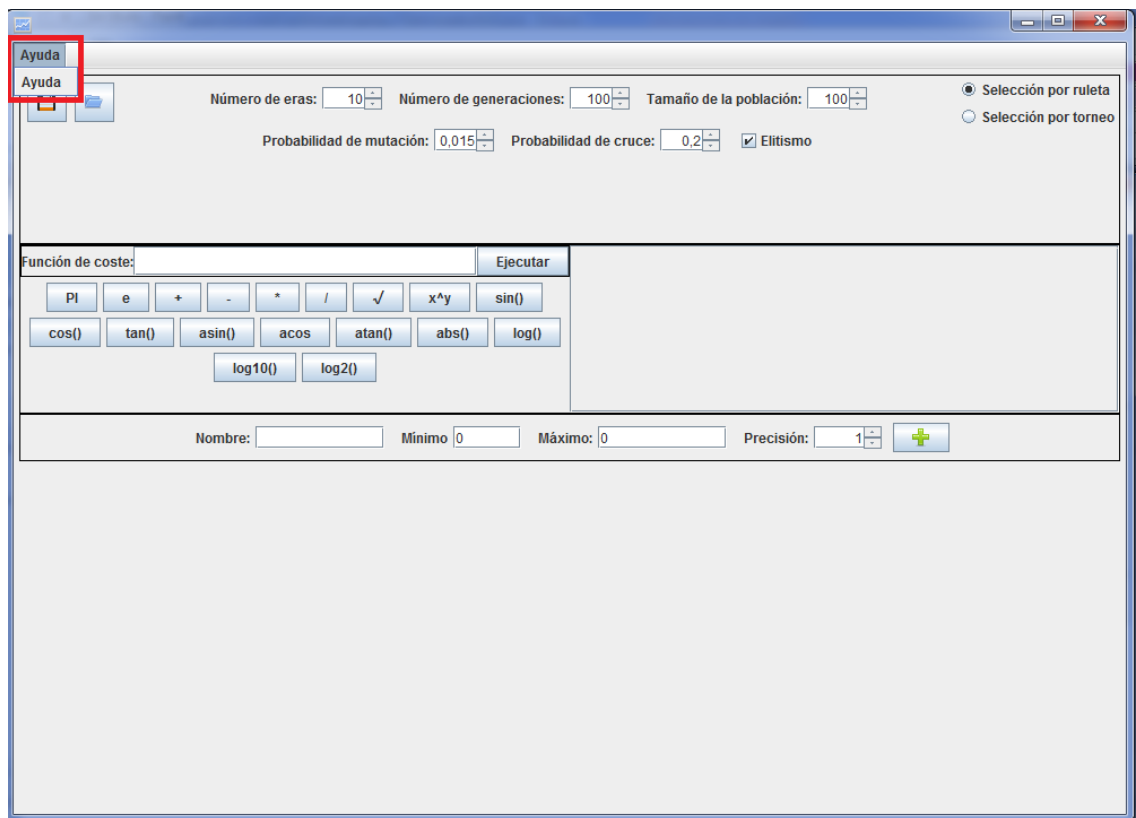


Figura 31 Acceso a la ayuda de la aplicación

11. Ejemplos de ejecución

Para probar la ejecución del algoritmo se han utilizado varias diversas funciones proporcionadas en las directrices del trabajo. A continuación se muestran los resultados de estas pruebas.

Las pruebas han sido realizadas utilizando la configuración por defecto. Esto es:

Número de eras	10
Número de generaciones	100
Tamaño de la población	100
Probabilidad de mutación	0.015
Probabilidad de cruce	0.2
Elitismo	Activado
Tipo de selección	Selección por ruleta

Tabla 4 Configuración por defecto

11.1. Caso de prueba 1

Función de evaluación:

$$f(x_1, x_2) = 21.5 + x_1 \cdot \text{sen}(4 \cdot \pi \cdot x_1) + x_2 \cdot \text{sen}(4 \cdot \pi \cdot x_2)$$

$$S = \{x_1 \in [-3.0, 12.1], x_2 \in [4.1, 5.8]\}$$

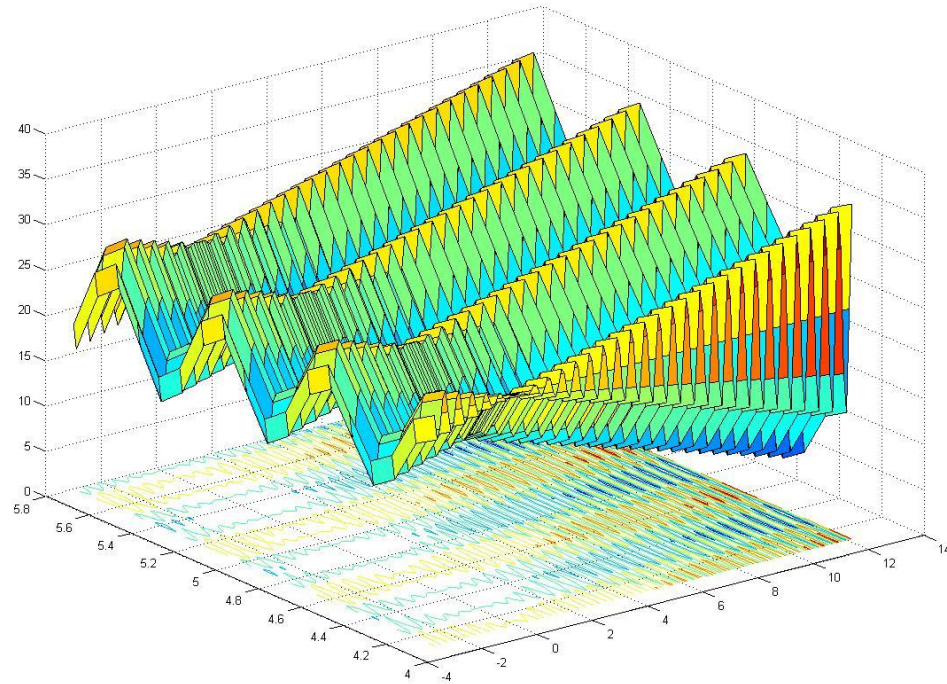


Figura 32 Caso de prueba 1

Como se puede observar, la función tiene varios máximos con valores próximos a 38,77

El mejor cromosoma obtenido por la herramienta con la configuración por defecto es el formado por $x_1=12,1$ y $x_2=5,6$ con un coste de 38,33.

11.2. Caso de prueba 2

Función de evaluación:

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = 100 - (x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2)$$

$$S = \begin{cases} x_1 \in [-3.0, 5.1], x_2 \in [2.1, 7.8], x_3 \in [-10.1, 20.3], \\ x_4 \in [-3.3, 4.2], x_5 \in [-15.3, 70.1], x_6 \in [-0.25, 0.35] \end{cases}$$

Como se puede observar, la función toma su valor máximo cuando todos los parámetros x_i tienen valor 0. El coste de este cromosoma es de 100.

El mejor cromosoma obtenido por la herramienta con la configuración por defecto es el formado por los valores $\{-0.21, 2.1, 0.5, 0.0, -0.2, -0.06\}$ con un coste de 95,25.

11.3. Caso de prueba 3: Función de Ackley

Función de evaluación:

$$f(x_1, x_2) = -c_1 \cdot e^{\left(c_2 \sqrt{\frac{1}{2}(x_1^2 + x_2^2)}\right)} - e^{\frac{1}{2}(\cos(c_3 x_1) + \cos(c_3 x_2))} + c_1 + e$$

$$c_1 = 20 \quad c_2 = 0.2 \quad c_3 = 2\pi \quad e = 2.71828$$

$$S = \{-5 \leq x_1 \leq 5; -5 \leq x_2 \leq 5\}$$

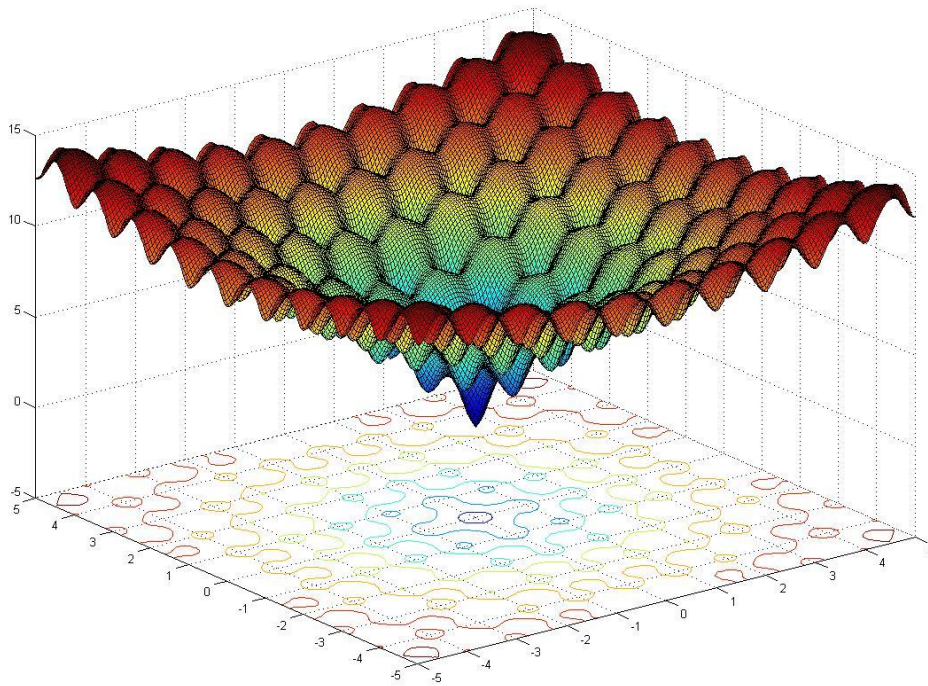


Figura 33 Caso de prueba 3. Función de Ackley

La función tiene cuatro máximos en los valores $\{(-4.6, 4.6), (4.6, -4.6), (-4.6, -4.6), (4.6, 4.6)\}$ con costes de 14.30.

El mejor cromosoma obtenido por la herramienta con la configuración por defecto es el formado por los valores $x_1 = 4.6$ y $x_2 = -4.6$ con un coste de 14.30.

La función de Ackley suele usarse para probar algoritmos de optimización en su variante de minimización. Dispone de varios mínimos locales por lo que conlleva dificultades para los algoritmos evolutivos. Dispone de un mínimo global en (0,0) con un coste de 0.

Al probar la herramienta para minimizar la función de Ackley (optimizar el valor inverso de la función) se obtiene el mejor cromosoma en (0.0, 0.0) con un coste de 0

11.4. Caso de prueba 4: Función de Schaffer

Función de evaluación:

$$f(x_1, x_2) = 100 - [(x_1^2 + x_2^2)^{0.25} \cdot [\text{sen}^2(50 \cdot (x_1^2 + x_2^2)^{0.1}) + 1.0]]$$

$$S = \{x_1 \in [-100,100], x_2 \in [-100,100]\}$$

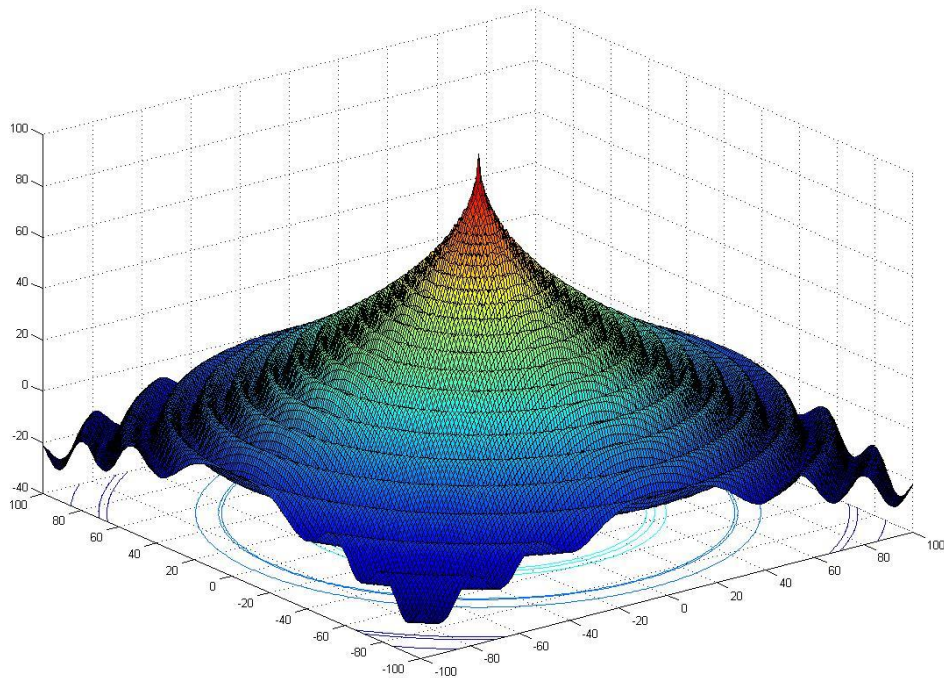


Figura 34 Caso de prueba 4. Función de Schaffer

Como puede observarse, esta función dispone de numerosos máximos locales y un máximo global claro alrededor del punto (0,0) con un valor de 100.

El mejor cromosoma obtenido por la herramienta con la configuración por defecto es el formado por los valores $x_1 = -0.1$ y $x_2 = -0.4$ con un coste asociado de 93.1

Se observa que el mejor valor de las distintas eras es relativamente bajo, con cromosomas cuyos costes oscilan entre 84 y 93.

ERA	MEJOR CROMOSOMA	COSTE
1	[x1,-0.2][x2,1.3]	88.17640770634893 Ver evolución
2	[x1,2.4][x2,0.8]	83.7174784340698 Ver evolución
3	[x1,1.0][x2,0.1]	89.92916719732565 Ver evolución
4	[x1,0.1][x2,0.6]	91.48261393130707 Ver evolución
5	[x1,-0.1][x2,-0.4]	93.10124238116529 Ver evolución
6	[x1,0.7][x2,1.9]	84.72666789190603 Ver evolución
7	[x1,-0.6][x2,-1.4]	86.8038525436497 Ver evolución
8	[x1,0.9][x2,1.7]	85.8789760871191 Ver evolución
9	[x1,2.2][x2,1.0]	84.45204911606933 Ver evolución
10	[x1,-0.2][x2,0.8]	90.28902339772215 Ver evolución

Figura 35 Resultados con la configuración por defecto

Al modificar la configuración y aumentar el número de generaciones a 1000, el mejor cromosoma obtenido es el formado por los valores $x1 = 0$ y $x2 = 0$ con un coste asociado de 100, encontrando el máximo global a partir de la generación número 500.

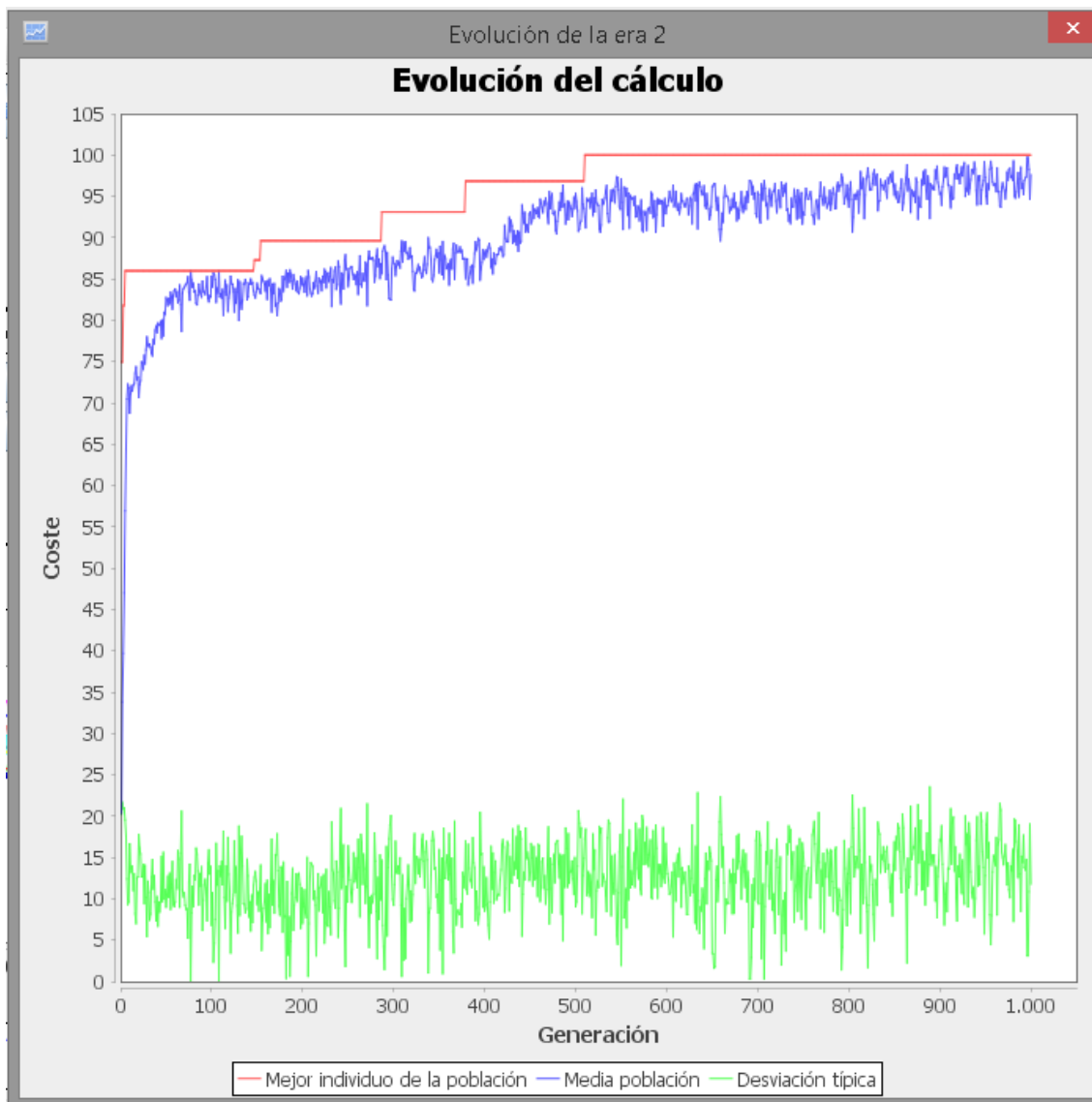


Figura 36 Evolución de la mejor era

Asimismo se observa que los resultados de todas las eras mejoran en general, obteniéndose cromosomas con costes entre 92 y 100.

ERA	MEJOR CROMOSOMA	COSTE
1	[x1,-0.2][x2,-0.1]	95.10156297372293 Ver evolución
2	[x1,-0.0][x2,-0.0]	100.0 Ver evolución
3	[x1,-0.2][x2,0.2]	94.24226738809473 Ver evolución
4	[x1,-0.5][x2,-0.1]	92.80330476463598 Ver evolución
5	[x1,-0.0][x2,0.1]	96.83224920799687 Ver evolución
6	[x1,-0.0][x2,-0.0]	100.0 Ver evolución
7	[x1,0.2][x2,-0.0]	95.08610529939716 Ver evolución
8	[x1,-0.2][x2,0.1]	95.10156297372293 Ver evolución
9	[x1,-0.1][x2,0.1]	96.0664103170124 Ver evolución
10	[x1,-0.1][x2,0.1]	96.0664103170124 Ver evolución

Figura 37 Resultados con 1000 generaciones

Al modificar la configuración, empleando selección por ruleta y aumentando el número de generaciones a 1000 se obtiene el máximo global a partir de la generación número 500.

11.5. Caso de prueba 5

Función de evaluación:

$$f(x_1, x_2) = 500 - [a \cdot (x_2 - b \cdot x_1 + c \cdot x_1 - d) + e \cdot (1 - f) \cdot \cos(x_1) + e]$$

$$a = 1 \quad b = \frac{5.1}{4 \cdot \pi^2} \quad c = \frac{5}{\pi} \quad d = 6 \quad e = 10 \quad f = \frac{1}{8 \cdot \pi}$$

$$S = \{-5 \leq x_1 \leq 10 \quad 0 \leq x_2 \leq 15\}$$

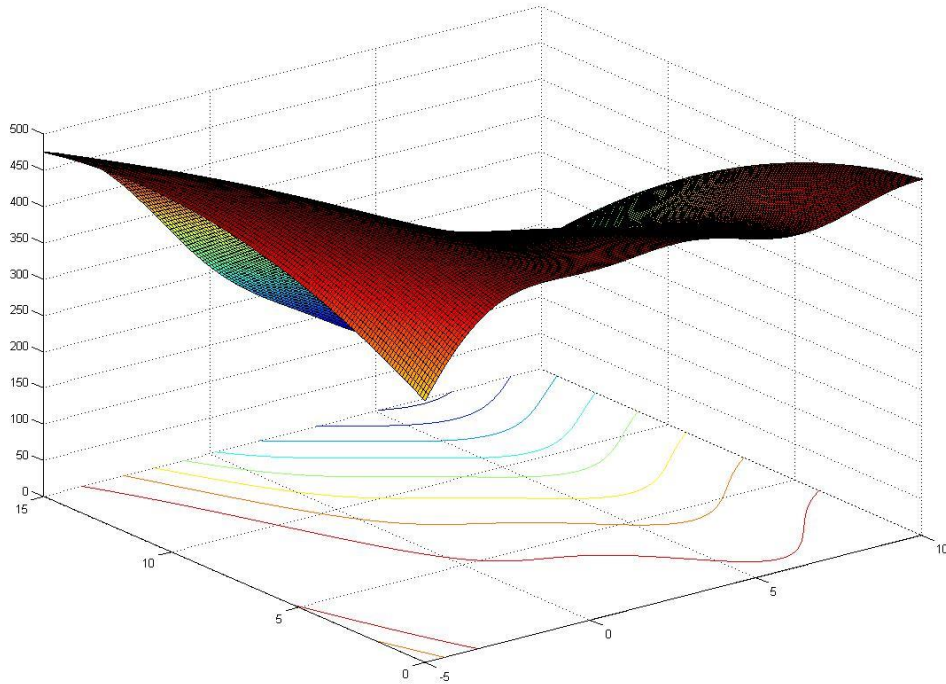


Figura 38 Caso de prueba 5

Existen numerosos valores que hacen máximo de la función con un valor cercano a 500.

El mejor cromosoma obtenido por la herramienta con la configuración por defecto es el formado por los valores $x_1 = 9.4$ y $x_2 = 2.5$ con un coste asociado de 499.88649.

11.6. Caso de prueba 6

Función de evaluación:

$$f(x, y) = 0.5 - \frac{\sin^2 \sqrt{x^2 + y^2} - 0.5}{(1 + 0.001 \cdot (x^2 + y^2))^2}$$

$$S = \{x, y \in [-100, 100]\}$$

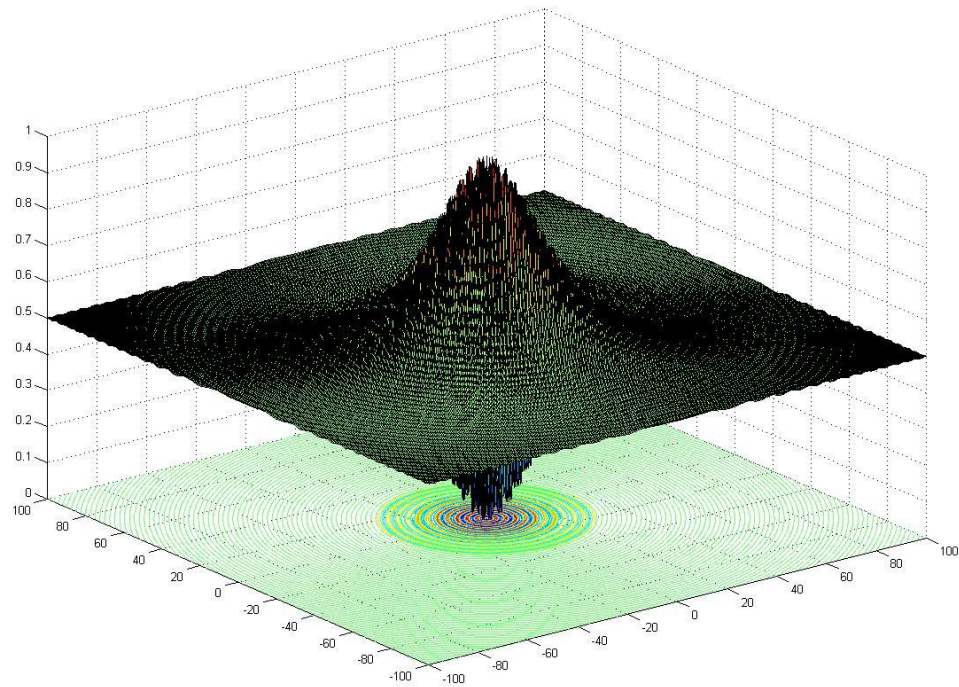


Figura 39 Caso de prueba 6

Puede observarse que la función dispone de un máximo en (0,0) con un coste de 1.

El mejor cromosoma obtenido por la herramienta con la configuración por defecto es el formado por los valores $x = 0.1$ e $y = -3.1$ con un coste asociado de 0.9889.

12. Planificación y presupuesto

Describe cuanto has tardado, cómo has dividido el trabajo y cuanto costaría tu aplicación (un diagrama de gant con las partes del proyecto ayuda mucho en esta fase)

13. Conclusiones

14. Referencias y bibliografía

15. Siglas y acrónimos