Escuela Técnica Superior de Ingeniería Informática

Universidad Nacional de Educación a Distancia

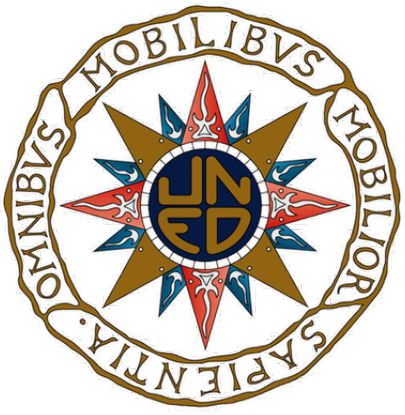
Departamento de Inteligencia Artificial

Máster Universitario en Inteligencia Artificial Avanzada:

Fundamentos, Métodos y Aplicaciones

Trabajo Fin de Máster:

Técnicas Robustas de Control de la Fase de Agrupamiento en Algoritmos Genéticos



Alumno: **Antonio de Dios San Feliciano**.

Ingeniero en Informática por la Universidad de Valladolid

Tutor: Dr. Severino Fernández Galán

Índice

Índice de figuras

Índice de tablas

Agradecimientos

A mis padres, quienes me infundieron la ética y el rigor que guían mi transitar por la vida.

Mi agradecimiento a cuantas personas han hecho posible la realización del presente trabajo con cita especial al Dr. Don Severino Fernández Galán, director del presente trabajo fin de máster, por su asesoramiento científico y estímulo para seguir creciendo intelectualmente.

Resumen

El presente trabajo trata la optimización de funciones de variable real. Para llevar a cabo dicha tarea se emplean métodos evolutivos, que consituyen una opción útil cuando los métodos analíticos no son aplicables a estos problemas de optimización.

Los algoritmos genéticos son métodos evolutivos de búsqueda estocástica, inspirados en la teoría de la evolución, que permiten optimizar funciones. Sin embargo, los algoritmos genéticos poseen el problema de la convergencia prematura, que consiste en que la población tiende a concentrarse sobre ciertos óptimos locales, produciendo que el proceso de búsqueda arroje como solución un resultado subóptimo. En la práctica, este problema es difícil de detectar pues muchas veces no se conoce a priori el valor del óptimo global.

Una de las causas de la convergencia prematura es una elevada presión selectiva, que restringe la exploración en favor de la explotación. La fase o técnica de Agrupamiento (*crowding* en inglés) se emplea en algoritmos genéticos para mantener la diversidad de la población y así evitar la convergencia prematura. A grandes rasgos, esta técnica consiste en emparejar cada descendiente con un individuo similar en la población actual (fase de emparejamiento) y decidir cuál de los dos individuos sobrevive (fase de reemplazo).

El presente trabajo se centra en controlar la fase de agrupamiento en algoritmos genéticos, de manera que se aplique en cada generación del algoritmo la presión selectiva adecuada para que el proceso de búsqueda sea lo más efectivo posible. Cuando hay demasiada presión selectiva el algoritmo puede acabar en óptimos locales y cuando hay poca presión selectiva se desperdicia mucho tiempo de computación explorando soluciones poco prometedoras. Por tanto, se trata de conseguir un equilibrio entre estos dos aspectos a lo largo de todo el proceso de búsqueda.

Capítulo 1: INTRODUCCIÓN

1.1. Explicación del Problema

Los algoritmos genéticos (AGs) [1], [2] son métodos de búsqueda estocástica, basados en la evolución natural, que se han aplicado con éxito a la optimización de problemas en numerosos campos.

Generalmente, el objetivo de un AG es encontrar la solución óptima (o valor próximo al óptimo) al problema planteado. Este caso puede ampliarse a identificar un conjunto de soluciones diversas con alto valor de adaptación (*fitness* en inglés). Durante este proceso pueden darse problemas de convergencia prematura hacia óptimos locales, lo cual es una dificultad que con frecuencia aparece cuando se aplican AGs a problemas complejos. Este inconveniente suele estar relacionado con la pérdida de diversidad en la población. Por otro lado, demasiada diversidad deteriora la eficiencia del AG. Un equilibrio entre la explotación de los mejores individuos y la exploración de regiones del espacio de búsqueda alternativas es importante en la investigación y aplicación práctica de AGs.

El *Agrupamiento* fue introducido por De Jong [3] como una técnica para preservar la diversidad de la población y evitar la convergencia prematura. Es aplicado en la fase de selección de supervivientes en AGs para decidir qué individuos entre los descendientes y la población actual deben pasar a la siguiente generación. Esta técnica se realiza en dos pasos: emparejamiento y reemplazo. En la etapa de emparejamiento, cada hijo o descendiente de la población actual es emparejado con un individuo similar de la propia población actual en función de una métrica que mide la semejanza genotípica o fenotípica entre individuos. En la fase de reemplazo se toma la decisión para cada pareja de qué individuo continua en la población.

Existen dos tipos típicos de reemplazo en función de cómo se realiza el mismo:

- El Agrupamiento determinista [4],[5] selecciona el individuo de mayor *fitness* de cada pareja para que sobreviva. El diseño de la regla de reemplazo tiene un gran impacto en el rendimiento del Agrupamiento. En este sentido, el Agrupamiento determinista realiza una estrategia explotativa, que puede dar lugar a convergencia prematura.

- El Agrupamiento probabilístico [6],[7] selecciona el individuo que sobrevive, basándose en una fórmula probabilística que tiene en cuenta el *fitness*. Este Agrupamiento promueve la exploración de individuos con menor *fitness*. El grado de exploración es definido a través de una fórmula que no cambia durante la ejecución del AG.

Sin embargo, recientemente se ha desarrollado una técnica denominada *Agrupamiento Generalizado* que permite aplicar más exploración o más explotación de forma sencilla a lo largo del proceso de búsqueda [8].

El Agrupamiento Generalizado amplía el modelo de Agrupamiento probabilístico; sin embargo, el grado de exploración puede ser controlado a través de un parámetro denominado factor de escalado. De esta manera, aplicando un determinado valor a dicho factor se obtiene una presión selectiva fija durante toda la ejecución de un AG. El Agrupamiento Generalizado se transforma en Agrupamiento determinístico cuando el factor de escalado es igual a cero.

El Agrupamiento Generalizado sólo había sido aplicado hasta ahora manteniendo fijo el factor de escalado [8]. Debido al hecho de que este parámetro determina la presión selectiva, el presente trabajo investiga cómo se debería controlar el valor del mismo para mejorar la efectividad en la búsqueda de la solución óptima o lo más cercana posible. Por ejemplo, llevando a cabo más exploración en las primeras generaciones del AG y más explotación al final de las generaciones se consiguen generalmente mejores resultados.

En este trabajo se exploran diferentes métodos de control del factor de escalado: control determinista, control adaptativo en función de la diversidad de la población y control autoadaptativo.

1.2. Motivación y Objetivos

Este trabajo estudia el problema de la convergencia prematura en problemas de optimización de funciones reales de varias variables. Concretamente, se han seleccionado seis funciones introducidas por Ballester y Carter en [9], ya que se observaron en ellas comportamientos heterogéneos dependiendo del valor del factor de escalado usado en el Agrupamiento Generalizado. Por el contrario, en funciones como las de Michalewicz, Ackley, Schwefel, etc. siempre se obtuvo un comportamiento más homogéneo que favorecía los valores de escalado más bajos. Por tanto, de cara a hacer un estudio completo del control del factor de escalado, las seis funciones de Ballester y Carter ofrecían un conjunto de problemas de experimentación más adecuado y fiable.

El objetivo inicial de este trabajo era validar la teoría del Agrupamiento Generalizado con funciones de variable real para comprender y observar cómo se realiza el mantenimiento de la diversidad de una población en un AG. Posteriormente se dio paso al estudio de qué valores del factor de escalado favorecen la búsqueda de soluciones óptimas.

Uno de los puntos importantes de este proyecto de investigación es el desarollo de un AG que implemente los mecanismos de Agrupamiento Generalizado, para observar la respuesta que da dicho tratamiento ante diferentes funciones de variable real. Junto con el desarrollo del algoritmo se han codificado tres controles (determinístico, adaptativo y autoadaptativo) del anteriormente denominado factor de escalado, cuyas especificaciones serán explicadas en los diferentes capítulos de esta memoria. Se trata por tanto de estudiar estos tres enfoques y la mejoras que ofrecen respecto del método tradicional de establecer el valor del factor de escalado de manera manual antes de cada ejecución del AG y mantenerlo fijo a lo largo de todo el proceso de búsqueda.

Alcanzado este objetivo, se dispone de un marco de trabajo basado en el Agrupamiento Generalizado, que minimiza la parte de configuración de un AG, mejorando el rendimiento en la búsqueda de soluciones óptimas y aplicable a un amplio abanico de funciones matemáticas que se deseen optimizar, independientemente de su configuración multimodal.

Por último, otra parte fundamental del trabajo es el análisis de los datos obtenidos a partir de los resultados de las seis funciones estudiadas para la extracción de conclusiones.

1.3. Contribuciones

Los experimentos con AGs desarrollados en este trabajo se han aplicado a la optimización de funciones de variable real y muestran cómo los tres controles del factor de escalado implementados obtienen mejores resultados que el Agrupamiento Generalizado, en el que el factor de escalado permanece fijo durante toda la ejecución del AG.

Estos tres métodos de control son una contribución novedosa, que se encuentra desarrollada en el artículo **"Adaptive Generalized Crowding for Genetic Algorithms"** [10], en estado de revisión para su publicación, en la revista *IEEE Transactions on Evolutionary Computation* de la *IEEE Computation Intelligence Society*.

Los controles adaptativo y autoadaptativo propuestos resultan ser robustos para un amplio rango de problemas de optimización. Además, en el capítulo 5 de Evaluación se muestra cómo el control determinista, aunque menos robusto que los dos anteriores, puede llegar a ser una buena elección.

Finalmente, cabe destacar que los controles implementados constituyen un marco de trabajo aplicable a todo tipo de funciones a optimizar, independientemente de si a éstas les es más favorable una estrategia de explotación o exploración.

1.4. Organización de la Memoria

El resto de la memoria se estructura de la siguiente manera. El segundo capítulo contiene la revisión de la bibliografía y constituye una presentación del concepto de la Computación Evolutiva, centrada en la técnica del Agrupamiento Generalizado de los AGs. El tercer capítulo describe las nuevas técnicas robustas de control de la fase de Agrupamiento de un AG, que son originales del presente trabajo. El cuarto capítulo contiene la especificación del sistema informático, que implementa los mecanismos de Agrupamiento y los diferentes controles propuestos del factor de escalado. El quinto capítulo detalla el diseño y la ejecución de los experimentos llevados a cabo sobre las funciones de Ballester y Carter. Por último, se detallan las conclusiones y posibles trabajos futuros de la presente investigación en el sexto capítulo de esta memoria. El documento finaliza con la presentación de la bibliografía empleada para el desarrollo de toda la acción

investigadora de este trabajo.

Capítulo 2: COMPUTACIÓN EVOLUTIVA

2.1. Definición y Objetivos

La Computación Evolutiva es una rama de la Inteligencia Artificial, centrada en la resolución de problemas de optimización, modelado y simulación, basada en los principios de la evolución biológica. De forma general tiene como base las ideas de la evolución de Charles Darwin y los estudios de genética de Gregor Mendel.

Como aspectos importantes a tener en cuenta en el diseño de un algoritmo evolutivo se pueden citar los siguientes:

- Representación
- Función de evaluación
- Población
- Mecanismo de selección de padres
- Operadores de variación
- Mecanismo de selección de supervivientes

La representación consiste en la determinación de la estructura del cromosoma de los individuos. Se basa en definir una relación entre el fenotipo y su codificación en un formato entendible por el algoritmo, denominado genotipo.

La función de evaluación, también conocida como función de adaptación, consiste en la definición de los requisitos a los que los individuos de la población se adaptan. Esta función es también denominada función *fitness* en Computación Evolutiva.

Una población es un conjunto de individuos que son mantenidos como posibles soluciones al problema. Una característica fundamental de toda población es la diversidad de la misma. La diversidad mide el número de soluciones diferentes presentes.

La labor de la selección de padres consiste en escoger los individuos basados en su calidad y, en particular, permitir que los mejores individuos se conviertan en padres de la siguiente generación.

Los operadores de variación son la recombinación y la mutación. Por una parte, el operador de recombinación o cruce consiste en la generación de la descendencia a partir de un conjunto de padres. Por otro lado, la mutación se basa en la alteración generalmente mínima de un genotipo en función de una determinada tasa de mutación o probabilidad de mutación.

La selección de supervivientes consiste en la definición del mecanismo que decide qué individuos sobreviven y pasan a la siguiente generación.

Los algoritmos genéticos son métodos evolutivos basados en la evolución de un conjunto de individuos, denominados población, tratando de mejorar los individuos con el paso de las generaciones de tal forma que las soluciones, que codifican dichos elementos, mejoren.

2.2. Fases Típicas de un Algoritmo Genético

El esquema general de un AG se presenta en la tabla 1. En primer lugar, la inicialización consiste en la creación de la población inicial, normalmente asignando valores aleatorios a cada individuo.

Durante la fase de evaluación se asigna a cada indivudio un indicador de aptitud (o capacidad para resolver el problema propuesto). Este cálculo se lleva a cabo mediante la aplicación de una función de desempeño, denominada función de adaptatción o función de *fitness*.

A partir de este punto se define un bucle cuya condición de terminación suele ser alcanzar una solución lo suficientemente buena o alcanzar un determinado contador que indica el número de generaciones en las que evolucionará la población inicial.

La población de la generación siguiente se obtiene a través de la ejecución del operador de recombinación entre parejas de padres de la población. De este emparejamiento se obtienen

Inicio
INICIALIZAR población con candidatos de soluciones aleatorias
EVALUAR candidatos
REPETIR HASTA (Condición de fin)
SELECCIONAR padres
RECOMBINAR parejas de padres
MUTAR la descendencia
EVALUAR los nuevos candidatos
SELECCIONAR individuos para la siguiente generación
Fin repetir hasta
Fin

Tabla 1: Esquema general de un AG

los descendientes de la población, que pasan a una fase de mutación. En la fase de mutación, conforme a una probabilidad dada, los genes de los individuos son alterados. Posteriormente, de estos nuevos descendientes se calcula su función de adaptación, que determina su capacidad para resolver el problema.

Finalmente, se seleccionan los individuos que pasan a la siguiente generación y el bucle continúa.

2.3. Fase de Agrupamiento

El Agrupamiento es una técnica que persigue preservar la diversidad en la población mediante el mantenimiento de diferentes nichos, siendo un nicho una zona del espacio de búsqueda que rodea a un óptimo local. En dicha técnica se crean torneos locales para realizar los reemplazos en la población. En estos torneos una función distancia juega un papel importante. La función distancia mide la semejanza entre el genotipo de dos individuos, generalmente un padre y uno de sus hijos. El Agrupamiento dispone parejas entre un padre y uno de sus hijos minimizando la distancia de sus genotipos, de tal forma que cada padre se encuentra agrupado con su hijo

más similar.

Al realizar el emparejamiento entre el padre y su hijo más similar, se pretende mantener la diversidad en la población, puesto que a la siguiente generación pasa un individuo que o bien ya existía en la población por ser el padre, o es su hijo más similar y por tanto continúa existiendo al menos un individuo en ese mismo nicho en el que estaba el individuo original. De esta forma se tiene tendencia a no abandonar los nichos, aunque se tengan candidatos con mejor *fitness*, que pueden pertenecer a óptimos locales y no al global.

Existen distintos tipos de Agrupamiento, en función de quién gana el torneo entre el padre y el hijo más similar. Si siempre gana el individuo mejor adaptado, es decir, con mejor valor de la función *fitness*, se habla de un Agrupamiento determinista. Sin embargo, cuando se introduce una probabilidad de que sea el peor individuo el que gane el torneo y por tanto pase a la siguiente generación, se habla de Agrupamiento probabilístico.

El esquema de un AG con Agrupamiento se muestra en la tabla 2.

Inicio
INICIALIZAR población con candidatos de soluciones aleatorias
EVALUAR candidatos
REPETIR HASTA (Condición de fin)
Agrupamiento
Fin repetir hasta
Fin

Tabla 2: Esquema general de un AG con Agrupamiento

El Agrupamiento lleva a cabo las siguientes fases:

- Selección de padres uniforme y aleatoria (Nótese que no se aplica selección de padres basada en *fitness*, de cara a preservar la diversidad en la población antes de las fases posteriores del AG.).
- Recombinación y mutación de los descendientes.

- Cálculo de distancias.
- Agrupamiento para el torneo en función de las distancias calculadas.
- Torneo.

El esquema original de Agrupamiendo desarrollado por De Jong [3] selecciona aleatoriamente, para cada descendiente o hijo, γ individuos de la población actual. El hijo reemplaza al individuo de entre los γ seleccionados que es más similar a él. El parámetro γ es conocido como factor de Agrupamiento y normalmente $\gamma = 2$.

El esquema de De Jong fue modificado por Mahfoud [4] para preservar la diversidad de la población de manera eficiente empleando las siguientes restricciones:

- Los individuos de la población actual son emparejados aleatoriamente, de tal forma que todos los individuos de la población se convierten en padres.
- Con probabilidad P_C (probabilidad de cruce), los padres de cada pareja (p_1, p_2) son recombinados. Los dos hijos (c_1, c_2) son mutados con probabilidad P_M (probabilidad de mutación).
- Cada hijo compite con uno de sus dos padres para ser incluido en la población de la siguiente generación. Sea d(i_1,i_2) la distancia entre dos individuos i_1 e i_2:

  Si $d(p_1, c_1) + d(p_2, c_2) < d(p_1, c_2) + d(p_2, c_1)$

  $p_1 \leftarrow$ ganador de la competición entre p_1 and c_1

  $p_2 \leftarrow$ ganador de la competición entre p_2 and c_2

  else

  $p_1 \leftarrow$ ganador de la competición entre p_1 and c_2

  $p_2 \leftarrow$ ganador de la competición entre p_2 and c_1

Bajo este esquema, cada descendiente (o hijo) compite por sobrevivir con su padre más similar. La competición entre p y c está definida por una regla de reemplazo. Existen dos enfoques de dicha regla, el reemplazo determinista en el que el ganador del torneo es el individuo con

mejor *fitness* y el reemplazo probabilístico. Sea P_c la probabilidad de que el hijo c reemplace al padre p en la población. Esta probabilidad puede expresarse de la siguiente manera para un remplazo determinista.

$$P_c = \begin{cases} 1 & \text{if } f(c) > f(p) \\ 0{,}5 & \text{if } f(c) = f(p) \\ 0 & \text{if } f(c) < f(p) \end{cases},$$

donde se asume la maximización de la función de *fitness* f: $\{0,1\}^n \longrightarrow \Re$.

Por otro lado, el Agrupamiento probabilístico introducido por Mengshoel y Goldberg [6] usa una regla no determinista para establecer el ganador de la competición entre el padre p y el hijo c. La probabilidad de que c reemplace a p en la población es la siguiente:

$$P_c = \frac{f(c)}{f(c) + f(p)}.$$

De manera similar a los esquemas deterministas y probabilísticos surge el Agrupamiento Generalizado introducido por Galán y Mengshoel [8], consistente en una fase de emparejamiento y una de reemplazo. La novedad es el uso de un factor de escalado ϕ en la fase de reemplazo. Dicho factor de escalado permite usar un amplio abanico de reglas de reemplazo sin más que ajustar ϕ.

En el Agrupamiento Generalizado el ganador de la competición entre un padre p y un hijo c se establece utilizando la siguiente fórmula.

$$P_c = \begin{cases} \frac{f(c)}{f(c)+\phi\times f(p)} & \text{if } f(c) > f(p) \\ 0{,}5 & \text{if } f(c) = f(p) \\ \frac{\phi\times f(c)}{\phi\times f(c)+f(p)} & \text{if } f(c) < f(p) \end{cases}, \tag{1}$$

donde P_c es la probabilidad de que c reemplace a p en la población, f es la función de *fitness* a maximizar y $\phi \in \Re^+ \cup \{0\}$ denota el factor de escalado.

La idea clave del Agrupamiento Generalizado es, previamente a aplicar el agrupamiento probabilista, realizar el escalado del valor de adaptación, f, del individuo menos adaptado de entre p y c.

En la ecuación 1, si $f(p) < f(c)$ entonces $f(p)$ es trasnforamdo en $\phi \times f(p)$; en caso contrario, si $f(c) < f(p)$ entonces $f(c)$ es transformado en $\phi \times f(c)$. Cuando $\phi = 0$ en la ecuación 1, el Agrupamiento Generalizado se convierte en Agrupamiento determinista.

Cuando $0 < \phi < 1$, es posible que el individuo con menos *fitness* entre p y c gane el reemplazo. Cuando $\phi = 1$, el Agrupamiento Generalizado se convierte en Agrupamiento probabilístico.

Finalmente, cuando $\phi > 1$ la probabilidad de que el peor individuo gane la fase de reemplazo es mayor que en el Agrupamiento probabilista. Esto puede ser beneficioso en problemas altamente multimodales. Un problema multimodal es aquel en el que existen gran cantidad de óptimos locales.

El factor de escalado ϕ permite aplicar un amplio rango de presiones selectivas. Cuanto más grande es ϕ, resulta más probable que sean exploradas partes del espacio de búsqueda que no son óptimas localmente.

Capítulo 3: NUEVAS TÉCNICAS ROBUSTAS DE CONTROL DE LA FASE DE AGRUPAMIENTO

3.1. Introducción

Cualquier parámetro de un AG puede ser configurado de manera manual antes de la ejecución del AG o controlado automáticamente durante la propia ejecución. El control automático se lleva a cabo de manera transparente al usuario del AG. Además, el valor del parámetro se puede adaptar al estado del proceso de búsqueda.

Escoger un valor apropiado de ϕ en el Agrupamiento Generalizado es crucial para el buen rendimiento del mismo. Sin embargo, dicha elección es una tarea difícil y dependiente del problema, es decir, para cada problema existe un valor óptimo de ϕ.

El presente capítulo describe tres nuevos enfoques para controlar el parámetro ϕ: control determinista, control adaptativo basado en diversidad y control autoadaptativo.

3.2. Control Determinista

El control determinista del factor de escalado ϕ consiste en cambiar el valor de ϕ por medio de una regla determinista. Esta regla es predeterminada por el usuario y permanece fija durante toda la ejecución del AG. Además, este control no utiliza retroalimentación por parte de la población del AG.

Generalmente es conveniente reducir progresivamente el valor del factor de escalado, disminuyendo así el grado de exploración. Algunos ejemplos de técnicas de control determinista de ϕ

son los siguientes:

- Decrecimiento exponencial del factor de escalado

  Según este esquema, el factor de escalado en la generación t, $\phi(t)$, es calculado multiplicando $\phi(t-1)$ por una constante de decrecimiento $k \in [0,1]$. De esta forma,

  $$\phi(t) = \phi(1) \times k^{t-1},$$

  donde $\phi(1)$ es el valor del factor de escalado definido por el usuario para $t = 1$, la primera generación del AG. Cuando $k \approx 1$ este esquema es similar al Agrupamiento Generalizado manteniendo ϕ fijo. Adicionalmente, cuando $k \approx 0$ este esquema resulta equivalente al Agrupamiento determinista. En general, un mayor valor de k proporciona mayor exploración.

- Decrecimiento lineal del factor de escalado

  Según este esquema, el factor de escalado en la generación t, $\phi(t)$, es calculado restando a $\phi(t-1)$ una constante de decrecimiento $k \in \Re^+ \cup \{0\}$. Entonces,

  $$\phi(t) = \phi(1) - \{k \times (t-1)\},$$

  donde $\phi(1)$ es el valor del factor de escalado definido por el usuario para la primera generación. Obviamente si, la fórmula obtiene valores negativos, entonces $\phi(t)$ se iguala a cero. En este esquema, a mayor valor de k, menor exploración es aplicada.

Una desventaja de estos métodos es que el usuario necesita asignar un valor k antes de la ejecución del AG. Puesto que k determina el grado de exploración aplicado, aparece el mismo problema que en el caso del factor de escalado fijo. Es imposible saber el grado apropiado de exploración para un problema de optimización, lo que fuerza al usuario a dedicar un importante esfuerzo para configurar apropiadamente el valor de k. Por este motivo, en el presente trabajo haremos especial énfasis en los métodos de control adaptativo y autoadaptativo, que no requieren la configuración de parámetros adicionales y, por tanto, resultan más cómodos para el usuario.

3.3. Control Adaptativo

En el control adaptativo, basado en el concepto de diversidad, el valor del factor de escalado es actualizado a partir de información proporcionada por la población. Dicha información determina la magnitud del cambio de ϕ. El mecanismo de actualización usado es aplicado entre dos ciclos consecutivos del AG. En concreto, ϕ puede ser configurado en cada generación proporcionalmente a la diversidad de la población de la siguiente manera:

$$\phi(t) = \phi(1) \times \frac{div(t)}{div(1)}, \tag{2}$$

donde $\phi(1)$ es el factor de escalado definido por el usuario para la primera generación y $div(t)$ es la diversidad de la población en la generación t. De esta forma, ϕ se reduce a la misma velocidad que la diversidad de la población disminuye. Consecuentemente, el grado de exploración se reduce conforme el AG progresa.

En este trabajo se usa la medida de la diversidad basada en el concepto de la entropía de la población. La entropía es utilizada como una medida que determina directamente el valor del factor de escalado en cada generación. Concretamente, se define $div(t) = H(t)$ en la ecuación 2, donde $H(t)$ es la entropía media de las individuos en la población en la generación t. Si el genotipo está formado por N genes, el valor de la entropía media se define como sigue:

$$H(t) = \frac{1}{N} \sum_{i=1}^{N} H_i(t),$$

donde $H_i(t)$ es la entropía del gen i. $H_i(t)$ puede ser calculado de la siguiente forma:

$$H_i(t) = -\sum_{j=1}^{v_i} P_{ij} \times \log_{v_i} P_{ij}, \tag{3}$$

donde v_i es el número de alelos o posibles valores del gen i, y P_{ij} es la tasa de individuos de la población cuyo gen i adopta el j-ésimo posible valor del gen. Independientemente de la magnitud de v_i en la ecuación 3, $H_i(t)$ siempre pertenece al intervalo $[0, 1]$, lo cual también ocurre con $H(t)$.

3.4. Control Autoadaptativo

El control autoadaptativo ha sido ampliamente estudiado en algoritmos evolutivos. Por ejemplo, en estrategias evolutivas [13], los tamaños de pasos de mutación son autoadaptados a lo largo del proceso de búsqueda. Otro ejemplo es la autoadaptación del parámetro denominado índice de emparejamiento [14], que tiene lugar en algoritmos genéticos donde el emparejamiento previo a la fase de recombinación no es necesariamente aleatorio y se rige por las preferencias de los individuos. En cualquier caso, en este trabajo nos ocupamos de la autoadaptación del factor de escalado utilizado en AGs que aplican agrupamiento generalizado.

Llevamos a cabo el control autoadaptativo del factor de escalado codificando este parámetro dentro del cromosoma y aplicando las operaciones de recombinación y mutación a los valores de ϕ. De esta forma, los individuos con un valor efectivo de ϕ tienden a sobrevivir.

Si un individuo j es representado como $\langle x_{j,1}, \ldots, x_{j,N}\rangle$, se amplia su representación para dar cabida al factor de escalado, quedando $\langle x_{j,1}, \ldots, x_{j,N}, x_{j,N+1}\rangle$, donde $x_{j,N+1} = \phi_j$ es el factor de escalado del individuo j. En otras palabras, ϕ es ahora un parámetro local y cada individuo tiene un comportamiento independiente en la fase de reemplazo.

En el Agrupamiento Generalizado con control autoadaptativo, la regla de reemplazo para el padre $p \equiv \langle x_{p,1}, \ldots, x_{p,N}, \phi_p\rangle$ y el hijo $c \equiv \langle x_{c,1}, \ldots, x_{c,N}, \phi_c\rangle$ opera de tal forma que la probabilidad P_c de que el hijo c reemplace al padre p es la siguiente:

$$P_c = \begin{cases} \frac{f(c)}{f(c)+\phi_{\text{self}}\times f(p)} & \text{if } f(c) > f(p) \\ 0{,}5 & \text{if } f(c) = f(p) \\ \frac{\phi_{\text{self}}\times f(c)}{\phi_{\text{self}}\times f(c)+f(p)} & \text{if } f(c) < f(p) \end{cases},$$

donde ϕ_{self} es definido de la siguiente forma:

$$\phi_{\text{self}} = \begin{cases} \phi_p & \text{if } f(c) > f(p) \\ \phi_c & \text{if } f(c) < f(p) \end{cases}.$$

Nótese que, en el control autoadaptativo, el factor de escalado aplicado en la regla de reemplazo es aquel del individuo que posee menor *fitness* entre p y c. Esto está en consonancia con el Agrupamiento Generalizado, donde el *fitness* del peor individuo entre p y c es el *fitness* multiplicado por el factor de escalado.

En la etapa de inicialización, a cada factor de escalado se le asigna un número aleatorio distribuido uniformemente en el rango $[0, \phi_{\max}]$, donde $\phi_{\max}$ es definido por el usuario. Durante la recombinación, el cruce del factor de escalado puede asignarse de varias maneras: se puede emplear la media de los valores de los padres o permitir que los hijos hereden el valor de los padres, entre otras posibilidades. Este trabajo utiliza la herencia como opción de recombinación del factor de escalado, ya que hemos encontrado que produce mejores resultados experimentales.

La mutación del factor de escalado consiste en la suma a su valor de una cantidad procedente de una distribución gaussiana de media cero y desviación estándar $\phi_{\max} \times 0.1$. En el caso de que el factor de escalado se salga de los límites definidos $[0, \phi_{\max}]$, se mantiene el valor original.

En el capítulo 5 Evaluación se muestra empíricamente que los métodos de control adaptativo y de control autoadaptativo son buenos candidatos para lograr un control efectivo del factor de escalado de una manera robusta.

Capítulo 4: DESARROLLO DEL SISTEMA INFORMÁTICO

4.1. Introducción

El esquema propuesto por David Marr [12] distingue tres niveles de descripción. El primer nivel, denominado "Teoría Computacional", que describe el objetivo de la computación. El segundo nivel es "La Representación y el Algoritmo" que describe cómo alcanzar el objetivo descrito en el primer nivel. El último nivel es "La Implementación del Soporte Físico", que define lo que es usado por el algoritmo para alcanzar la meta.

En primer lugar hemos especificado en los capítulos 2 y 3 Computación Evolutiva y Nuevas Técnicas Robustas de Control de la Fase de Agrupamiento respectivamente, la Teoría Computacional abstracta del mecanismo, en la que se ha explicado el objetivo de maximizar las funciones de Ballester y Carter, junto con la estrategia a seguir. El segundo nivel consistente en la representación del algoritmo, se ha descrito también en capítulos previos, en los que se hace referencia a la técnica del Agrupamiento en el contexto de los AGs. Por tanto, los dos primeros niveles han sido contemplados en los capítulos 2 Computación Evolutiva y 3 Nuevas Técnicas Robustas de Control de la Fase de Agrupamiento.

Finalmente, el tercer nivel consiste en la descripción más detallada y a bajo nivel de la solución. Este último punto se denomina nivel de implementación y es descrito en este capítulo. Presentamos en primer lugar la especificación de requisitos del producto software, seguida del análisis arquitectónico y finalizamos con un apartado de pruebas.

4.2. Especificación de Requisitos

A continuación se presenta la especificación de requisitos del software, formada por una descripción completa del comportamiento del sistema desarrollado. Además, incluimos un conjunto de casos de uso que describen todas las interacciones que los usuarios tienen con el software. También contiene requisitos no funcionales. Para la descripción de este apartado se han seguido las recomendaciones de la IEEE 830-1998 [11], omitiendo las secciones no aplicables.

4.2.1. Introdución

4.2.1.1. Propósito

El objetivo de la especificación es definir de manera clara y precisa todas las funcionalidades y restricciones del sistema que se ha construido, orientadas a un público técnico.

4.2.1.2. Alcance

El nombre del producto software desarrollado es *Generalized Crowding* (GC), debido a que implementa dicha técnica. La aplicación implementa un AG que aplica la técnica del Agrupamiento Generalizado. Es capaz de llevar a cabo la optimización de diferentes funciones, utilizando diferentes configuraciones, asignadas en un fichero de propiedades. Por tanto, a través de un fichero de parejas clave-valor, actualiza la configuración de ejecución del algoritmo. Las propiedades configurables a través de este recurso se enumeran a continuación.

- Función a optimizar
- Número de variables reales de la función a optimizar
- Número de ejecuciones del AG
- Número de generaciones
- Tamaño de la población

- Máximo valor de cada una de las variables reales (define un intervalo simétrico entorno a 0)
- Tipo de cruce
- Probabilidad de cruce
- Tipo de mutación
- Probabilidad de mutación
- Valor del factor de escalado
- Tipo de control del factor de escalado

El objetivo de la aplicación es obtener una colección de resultados empíricos organizados para poder posteriormente estudiar los mismos. A partir del parámetro "Número de ejecuciones del AG", se define el número de veces que será ejecutado el AG. El AG se ejecuta más de una vez para poder obtener resultados experimentales más fiables, hallando la media, ya que entran en juego constantemente variables aleatorias y probabilidades.

La aplicación vuelca a dos ficheros de texto plano cada ejecución de la aplicación. Uno de ellos contiene los datos del mejor individuo encontrado en cada ejecución del AG. El otro archivo contiene el valor de adaptación medio (sobre las diferentes ejecuciones del AG) del mejor individuo de la población en cada generación.

4.2.1.3. Definiciones, Siglas y Abreviaciones

- AG: Algoritmo Genético
- GC: Generalized Crowding

4.2.1.4. Apreciación Global

El resto de la especificación de requisitos se organiza de la siguiente manera. La sección 4.2.2

presenta la descripción general del producto software, especificando una amplia perspectiva de las funcionalidades del mismo a alto nivel. Esta misma sección indica también a qué tipos de usuarios se orienta la aplicación. La sección Restricciones describe las obligaciones del sistema. La sección 4.2.3 especifica el prorrateo de los requisitos. La sección 4.2.4 describe los requisitos específicos descompuestos en requisitos funcionales, requisitos del desarrollo, requisitos del diseño y atributos del software.

4.2.2. Descripción Global

4.2.2.1. Perspectiva del Producto

Se ha desarrollado un producto independiente y totalmente autónomo. El sistema se ha construido de tal forma que permita una fácil ampliación. El producto se define como un sistema completo, que puede ser dotado de nuevas funcionalidades fácilmente, como nuevos operadores de cruce o mutación. Por lo que dichos componentes se organizan en paquetes dependiendo de su área de aplicación, como muestra la figura 1.

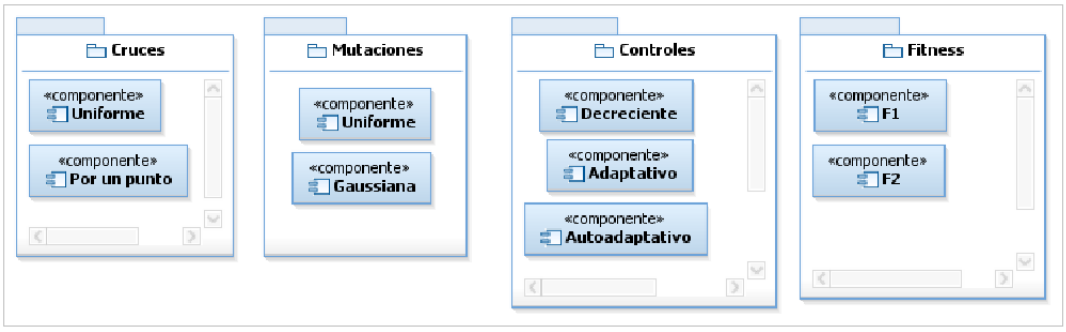


Figura 1: Organización

El sistema no se comunica con otros sistemas, por lo que no define ni importa interfaces externas. La forma de uso y ejecución es por línea de comandos. Su ejecución requiere de dos atributos, el primero de ellos indica el fichero de configuración y el segundo de ellos si la ejecución

se realiza a través de un fichero .jar. Dicha ejecución requiere permisos de escritura en el directorio de ejecución, ya que creará dos ficheros con los resultados obtenidos, cuyos nombres indicarán los valores de la prueba a la que referencian.

4.2.2.2. Funciones del Producto

La función principal del producto consiste en permitir diferentes ejecuciones de un AG, variando con facilidad su configuración. Las funcionalidades del producto son las siguientes:

- Ejecución del AG.
- Modificación intuitiva de los parámetros de configuración para permitir diferentes ejecuciones variando parámetros como la función de *fitness*, el operador de cruce, el operador de mutación y el control del factor de escalado aplicado.
- Recogida de resultados en forma de ficheros de texto plano.

4.2.2.3. Características del Usuario

El sistema está enfocado a un usuario con nivel de estudios alto, experiencia en informática y una especialización media en el diseño de AGs.

4.2.2.4. Restricciones

No existen restricciones en cuanto a políticas reguladoras, interfaces a otras aplicaciones, ni protocolos señalados. Sin embargo, el usuario debe de tener en cuenta las limitaciones del hardware en el que ejecute la aplicación. Dependiendo del tamaño de la población, del número de generaciones y del número de ejecuciones para calcular las medias la aplicación tendrá diferentes consumos de recursos. La aplicación puede ser ejecutada de manera independiente tantas veces como recursos disponga la máquina sobre la que se realizan las ejecuciones. Como norma general no se deberá superar el número de ejecuciones paralelas al número de procesadores, ya

que cada ejecución correrá en un hilo. El desarrollo de la aplicación tuvo como restricción el uso del lenguaje Java por su característica multiplataforma y su amplia extensión en el mundo entero.

4.2.3. Prorratear los requisitos

Aunque inicialmente se desarrolló un producto completo, con más técnicas de cruce y mutación para los estudios iniciales, la versión final sólo cuenta con el cruce uniforme y la mutación uniforme, por ser el objeto de los experimentos finales.

4.2.4. Requisitos funcionales

A continuación se detallan los requisitos funcionales del sistema:

- **[R01] Optimización de las seis funciones de Ballester y Carter**

  La funcionalidad principal de la aplicación será la optimización de las seis funciones de Ballester y Carter mediante un AG, que implementa el mecanismo de Agrupamiento Generalizado.

- **[R02] Carga del fichero de propiedades**

  La ejecución del algoritmo quedará determinada por la configuración especificada en un fichero de propiedades. En dicho fichero se configurarán las siguientes características:

  - Función a optimizar (F1-F6 funciones de Ballester y Carter)
  - Número de variables (Se usará con valor 2)
  - Número de ejecuciones (El número de ejecuciones que se realizan)
  - Número de generaciones (El número de generaciones por cada ejecución)
  - Tamaño de la población (El número de individuos en la población)

  - Máximo valor de cada una de las variables (Define un intervalo simétrico entorno a 0)
  - Tipo de cruce El tipo de cruce (En nuestro caso hará referencia al cruce uniforme, sin embargo dará cabida a otros tipos en función de otros valores)
  - Probabilidad de cruce (La tasa de recombinación)
  - Tipo de mutación (Hará referencia a la mutación uniforme)
  - Tasa de mutación (Probabilidad de mutación)
  - Valor de factor de escalado (Número real comprendido entre [0 - 1.25]
  - Tipo de control del factor de escalado (Entero que define el tipo de control de escalado)

- **[R03] Control fijo del factor de escalado**

  Funcionalidad que mantiene fijo el factor de escalado durante todas las generaciones de una ejecución.

- **[R04] Control linealmente decreciente del factor de escalado**

  Funcionalidad que decrece el factor de escalado de generación en generación.

- **[R05] Control adaptativo del factor de escalado**

  Funcionalidad que adapta el valor del factor de escalado en función de la entropía.

- **[R06] Control autoadaptativo del factor de escalado**

  Funcionalidad que lleva a cabo el control autoadaptativo, basado en que cada individuo posee su propio factor de escalado.

- **[R07] Presentación de los resultados por pantalla**

  La aplicación muestra por pantalla los resultados obtenidos al analizar su ejecución. Dichos resultados están formados por la media del mejor *fitness* encontrado para cada iteración en el conjunto de las ejecuciones y por el mejor *fitness* encontrado de cada ejecución.

- **[R08] Presentación de los resultados en ficheros de texto plano**

  La aplicación almacena los resultados obtenidos, expresados en el requisito anterior, en dos ficheros de texto plano.

4.2.5. Requisitos del desarrollo

Se presenta el único requisito del desarrollo:

- **[R09] El desarrollo es iterativo e incremental**

  El desarrollo del producto se ha realizado de manera iterativa en base a incrementos. De esta forma el sistema se desarrolló para soportar todas las funcionalidades desde el principio. Con cada incremento se implementaba una nueva funcionalidad, operador o control.

4.2.6. Restricciones del diseño

Se presenta a continuación la única restricción de diseño:

- **R[10] Aplicación de consola**

  El sistema es una aplicación de consola, sin interfaz gráfica de usuario, que recibe por línea de comandos el nombre del fichero de propiedades junto con la opción que indica si se ejecuta desde un fichero jar o no.

4.2.7. Atributos del software

Los principales atributos del software son enumerados a continuación:

- **Fiabilidad**. El sistema ha pasado pruebas unitarias, funcionales y además un estudio de resultados con otro sistema desarrollado por el tutor para verificar y validar los datos obtenidos.

- **Seguridad**. En caso de que se intente sobrescribir un fichero de resultados existente, se lanza un error e impide la pérdida de información mostrando por pantalla los resultados obtenidos tras la ejecución.
- **Mantenimiento**. Los principios de alta cohesión, bajo acoplamiento, junto con el uso de interfaces permiten la facilidad de mantenimiento del producto, minimizando la complejidad.
- **Portabilidad**. Toda la aplicación esta desarrollada con un lenguaje totalmente portable como es Java. Además, el desarrollo ha sido orientado a la reutilización y extensión de la aplicación gracias a su estructura de paquetes que favorecen la alta cohesión y el bajo acoplamiento.

4.2.8. Casos de uso

La figura 2 presenta los casos de uso del sistema y a continuación se describen.

[CU01] Configurar ejecución

Descripción: Representa la configuración de los parámetros de ejecución en el fichero de propiedades.

Pasos:

1. El actor Usuario actualiza los valores de configuración en el fichero de propiedades a su gusto.
2. El actor Usuario guarda dicho fichero.

[CU02] Realizar ejecución

Descripción: Se ejecuta la aplicación con una configuración determinada y se obtienen los resultados a su salida.

Pasos:

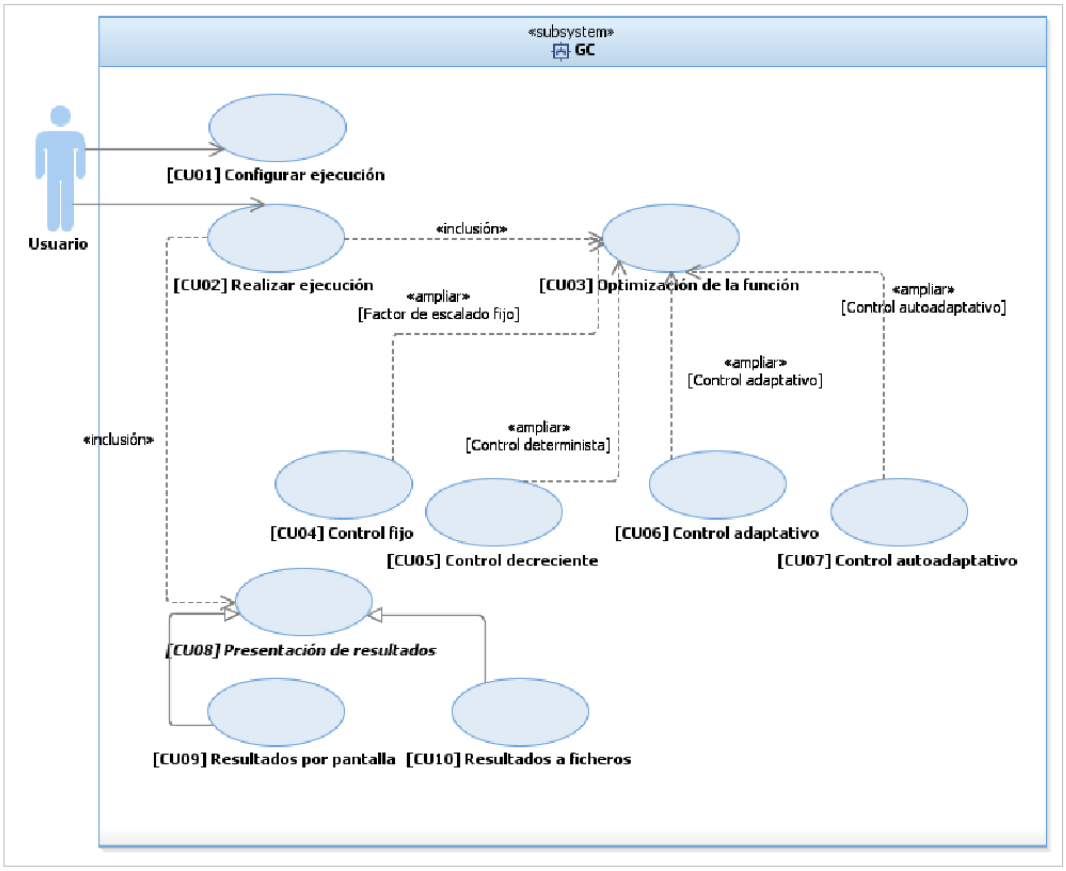


Figura 2: Casos de uso

1. El actor Usuario ejecuta la aplicación.

2. El sistema carga el fichero de propiedades.

3. El sistema ejecuta el algoritmo genético para optimizar la función.

4. El sistema produce la salida por pantalla y guarda los resultados en sendos ficheros.

[CU03] Optimización de la función

Descripción: La optimización de la función se lleva a cabo con la ejecución del AG en base a los parámetros del fichero de configuración previamente cargados.

Pasos:

1. Se realiza la optimización de la función mediante la ejecución del AG.

[CU04] Control fijo

Descripción: El valor del factor de escalado no cambia durante toda la ejecución.

[CU05] Control decreciente

Descripción: El valor del factor de escalado disminuye en cada generación en un factor 1 / nº de generaciones.

Pasos:

1. Al finalizar el bucle de cada generación, el factor de escalado actualiza su valor.

[CU06] Control adaptativo

Descripción: El valor del factor de escalado se modifica en cada generación en función del valor de la entropía de la población, calculada a partir de una discretización en 100 subintervalos del rango (o intervalo) de definición de la función en cada eje.

Pasos:

1. Al finalizar el bucle de cada generación el factor de escalado actualiza su valor.

[CU07] Control autoadaptativo

Descripción: El control autoadaptativo se basa en que cada individuo posee su propio factor de escalado.

Pasos:

1. El valor del factor de escalado forma parte del cromosoma del individuo, por lo que se le da su valor en la fase de cruce y se actualiza con mutación gaussiana en la fase de mutación.

[CU08] Presentación de resultados

Descripción: Caso de uso abstracto que representa la muestra de la información obtenida al finalizar la ejecución del GA.

[CU09] Resultados por pantalla

Descripción: Muestra los resultados por pantalla. En primer lugar muestra el mejor individuo encontrado para cada ejecución. En segundo lugar muestra la media del mejor *fitness* encontrado en cada iteración para todas las ejecuciones.

Pasos:

1. Al finalizar la ejecución del AG, se calculan las medias de los mejores resultados obtenidos.
2. Se muestran los mejores individuos de cada ejecución.
3. Se muestra la media de los mejores resultados de cada generación para todas las ejecuciones.

[CU10] Resultados a ficheros

Descripción: Almacena los resultados en ficheros de texto plano. En primer lugar almacena el mejor individuo encontrado para cada ejecución en un fichero que comienza por "best_means_scaling_factor_" y a continuación el valor del factor de escalado original procedente de la configuración. En segundo lugar almacena la media del mejor *fitness* encontrado en cada iteración para todas las ejecuciones en un fichero, cuyo nombre comienza por "means_scaling_factor_" y a continuación el valor del factor de escalado original, procedente de la configuración.

Pasos:

1. Al finalizar la ejecución del AG, se calculan las medias de los mejores resultados obtenidos.

2. Se almacenan los mejores individuos de cada ejecución.

3. Se almacenan las medias de los mejores resultados de cada generación para todas las ejecuciones.

El contenido de estos ficheros es utilizado para la generación de gráficas por lo que tiene una estructura con comentarios, empleando el carácter (#), que define una línea de comentario.

A continuación se muestra un ejemplo de fichero de mejores resultados de cada ejecución en la tabla 3. El fichero indica el mejor *fitness* encontrado y a continuación imprime el mejor individuo de cada ejecución en dos líneas. Primero indica la ejecución y después el valor de su *fitness*. La siguiente línea contiene el valor de las variables del mejor individuo. Después, en la tabla 4 se muestra un ejemplo de fichero de medias en el que cada fichero contiene una línea por cada generación.

```
#=========================
# Mean Best individual: 408.47230176623515
# Best individual of execution i,fitness,values
#=========================
0 406.8961716578037
#-1.763999696546886,1.6730224309973385
1 410.0484318746666
#1.8970951283992719,0.9996662929198443
...
```

Tabla 3: Fichero de mejores resultados

405.3624648804128
408.47230176623515
...

Tabla 4: Fichero de medias

4.3. Análisis Arquitectónico

4.3.1. Introducción

Este apartado proporciona una visión general de alto nivel de la arquitectura software. Debido a la complejidad para capturar la arquitectura software en un sólo modelo, se hace necesario manejar esta complejidad mediante la representación de diferentes aspectos y características de la arquitectura en múltiples vistas. El modelo más aceptado a la hora de establecer las vistas necesarias para describir una arquitectura software es el modelo 4+1 de Kruchten, representado en la figura 3 [15].

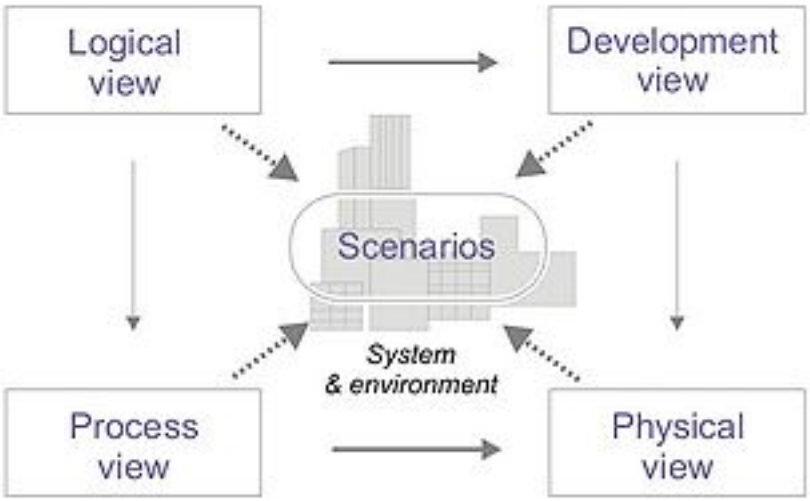


Figura 3: Modelo 4+1

La relación de las vistas utilizadas a las propuestas en el modelo se presenta en la siguiente tabla:

Vista	UML
Escenarios	Casos de Uso
Lógica	Clases, de Estados y Colaboración
Desarrollo	Componentes
Física	Despliegue
Procesos	Actividad, Estados, Secuencia

4.3.2. Organización

La especificación UML del sistema se ha dividido en las siguientes seis vistas, que se desarrollan a continuación:

- Vista de Casos de Uso: Describe los actores y casos de uso del sistema, presenta desde este punto de vista las necesidades del usuario y la funcionalidad del sistema. Presentado en la especificación de requisitos previa.
- Vista Lógica: Describe las abstracciones que forman del dominio del problema, incluye el diagrama de clases del sistema y paquetes.
- Vista de Proceso: Describe los modos de comunicación entre clases, que dan lugar a los procesos. Incluye los diagramas de secuencia.
- Vista de Despliegue: Describe la configuración física del sistema, incluye diagramas de despliegue.
- Vista de Implementación: Describe mediante diagramas todos los componentes del sistema. Incluye los diagramas de componentes del sistema. Los diagramas de componentes son utilizados para modelar la vista estática en un sistema. Además, sirven para modelar el código fuente. Debido a que se considera que la arquitectura y componentes quedan perfectamente definidos para el caso de la herramienta con las otras vistas, se ha omitido esta vista. Otra razón por la que no se ha construido esta vista es que está centrada en la

perspectiva del programador, la cual recoge el modelo a muy bajo nivel y esto queda fuera del alcance de este documento.

- Vista de Datos: Describe como se almacenan los datos de configuración.

4.3.3. Restricciones

La arquitectura está limitada por:

- Los requisitos funcionales que se indican en el apartado de especificación de requisitos software.
- Los requisitos no funcionales que se indican en el apartado de especificación de requisitos software.
- Las restricciones de diseño indicadas en el apartado de especificación de requisitos software.

4.3.4. Vista Lógica

Esta vista presenta en primer lugar el diagrama de clases de análisis del sistema, en segundo lugar el diagrama de clases del diseño y posteriormente se encuentra la arquitectura del sistema en tres niveles.

4.3.4.1. Diagrama de Clases del Análisis

La figura 4 muestra el diagrama de clases del análisis y a continuación se presentan las responsabilidades de dichas clases.

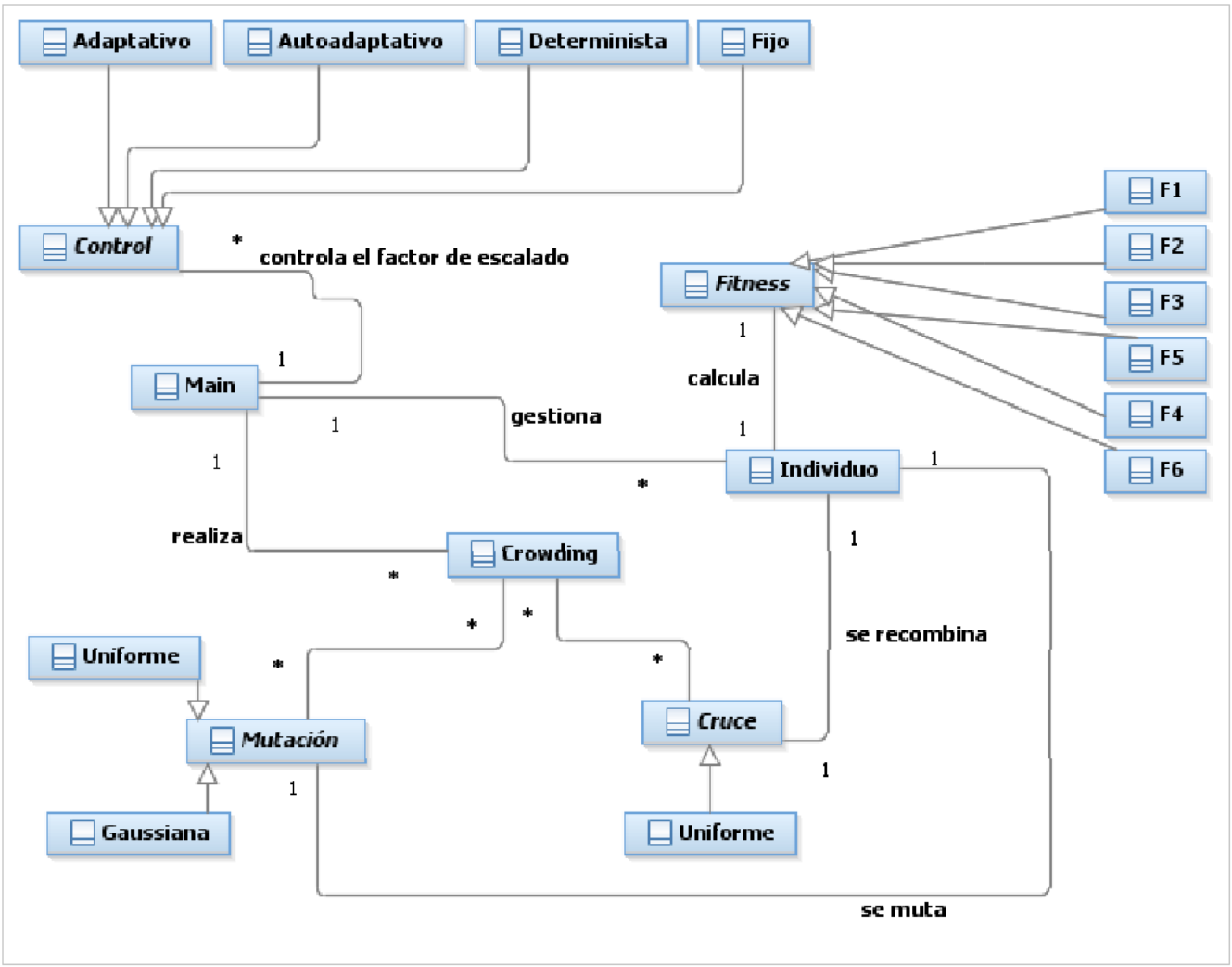


Figura 4: Diagrama de clases del análisis

4.3.4.2. Responsabilidades de las Clases

Main: Clase principal que inicia la ejecución del AG. Además, gestiona una población formada por un conjunto de individuos. También realiza la técnica del Agrupamiento y lleva a cabo el control del factor de escalado.

Control: Clase abstracta que encapsula el comportamiento común de los diferentes tipos de control del factor de escalado.

Adaptativo: Modela el control adaptativo.

Autoadaptativo: Modela el control autoadaptativo.

Determinista: Modela el control determinista.

Fijo: Modela el control fijo.

Crowding: Modela la técnica del Agrupamiento Generalizado.

Individuo: Representa una solución al problema. Contiene los genes, que son las variables y su valor solución al problema. A través de estos valores calcula su *fitness* en base a las funciones F1-F6.

Fitness: Clase abstracta que encapsula el comportamiento común de las funciones de adaptación.

F1-F6: Funciones de Ballester y Carter empleadas para calcular el valor de adaptación de un individuo.

Cruce: Clase abstracta que encapsula el comportamiento común a los diferentes operadores de cruce.

Uniforme: Modela el cruce uniforme.

Mutación: Clase abstracta que encapsula el comportamiento común a los diferentes operadores de mutación.

Uniforme: Modela la mutación uniforme.

Gaussiana: Modela la mutación gaussiana.

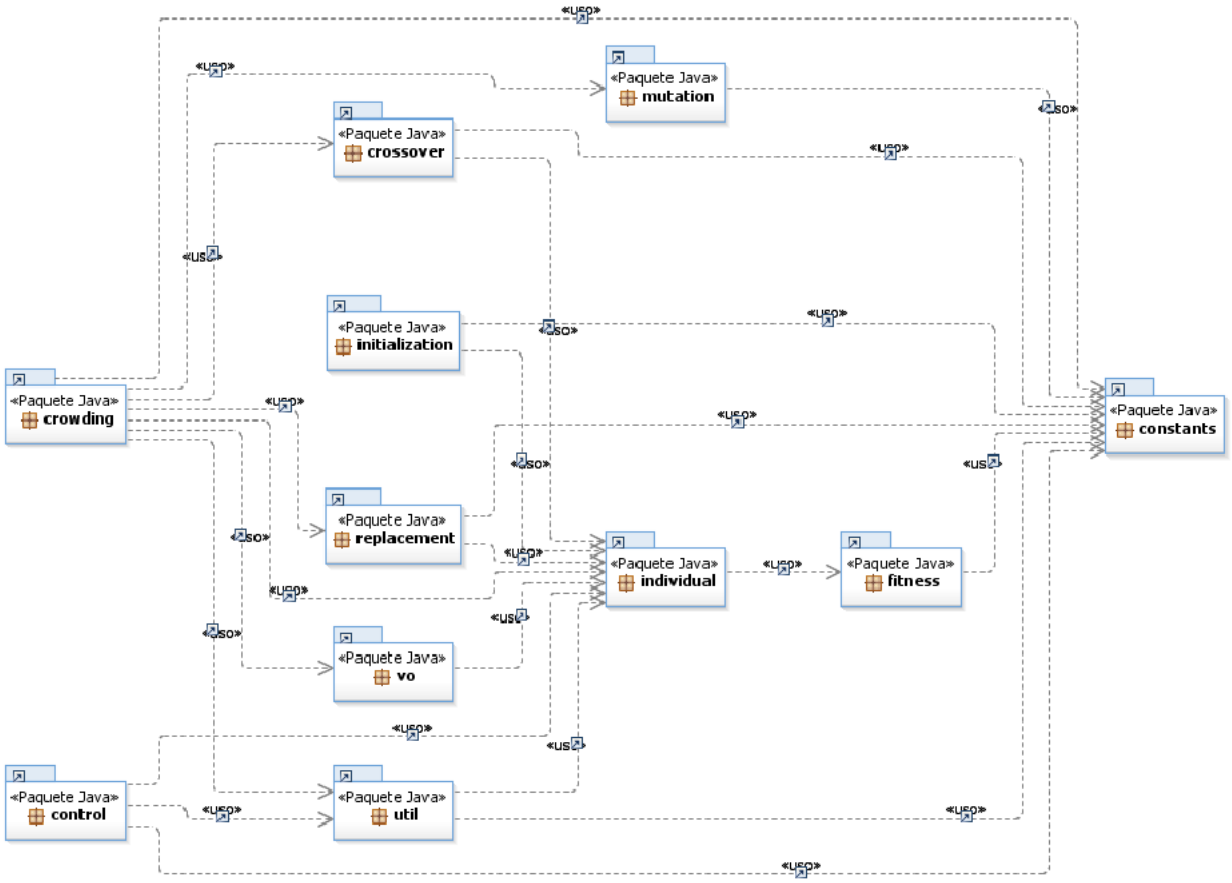


Figura 5: Diagrama de paquetes del diseño

4.3.4.3. Arquitectura del Sistema

Una vez conocidos los requisitos, los objetivos principales y las clases de análisis de la arquitectura software, a continuación se explica la arquitectura con menor nivel de abstracción.

El sistema se divide internamente en diferentes paquetes o subsistemas de manera que los artefactos se reparten entre los mismos paquetes, que colaboran para llevar a cabo el trabajo.

La figura 5 muestra los paquetes de la aplicación y sus dependencias. Dichos paquetes y su contenido son explicados en la siguiente sección.

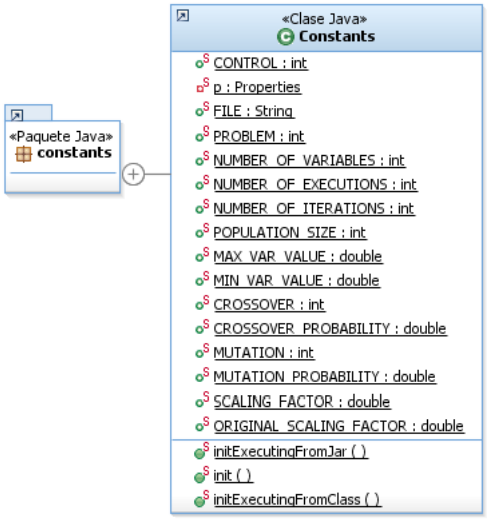


Figura 6: Paquete constants

4.3.4.4. Clases del Diseño

A continuación se presentan para cada paquete cada una de las clases e interfaces que lo forman.

Paquete Constants

Almacena la clase que contiene todas las constantes de ejecución, cuyo contenido se muestra en la figura 6.

Clase Constants

Descripción: Contiene todas las constantes de ejecución del AG. Dichas constantes son cargadas en la inicialización de la aplicación, procedentes del fichero de propiedades.

Atributos principales:

- public static int CONTROL: Indica el tipo de control que se lleva a cabo.
- private static Properties p: Fichero de propiedades.

- public static int PROBLEM: Indica la función a maximizar.
- public static int NUMBER_OF_VARIABLES: Número de variables del problema.
- public static int NUMBER_OF_EXECUTIONS: Número de ejecuciones.
- public static int NUMBER_OF_ITERATIONS: Número de generaciones
- public static int POPULATION_SIZE: Tamaño de la población.
- public static double MAX_VAR_VALUE: Valor máximo que puede tomar una variable.
- public static double MIN_VAR_VALUE: Valor mínimo que puede tomar una variable.
- public static int CROSSOVER: Indica el cruce que se realiza.
- public static double CROSSOVER_PROBABILITY: Probabilidad de cruce.
- public static int MUTATION: Indica la mutación que se realiza.
- public static double MUTATION_PROBABILITY: Tasa de mutación.
- public static double SCALING_FACTOR: Factor de escalado en una determinada generación.
- public static double ORIGINAL_SCALING_FACTOR: Factor de escalado originalmente configurado para una ejecución.

Métodos principales:

- public static void initExecutingFromJar(String file): Carga el fichero de propiedades como si estuviera ejecutándose el algoritmo desde un fichero jar.
- public static void init(String file, int option): Inicializa los parámetros.
- public static void initExecutingFromClass(String file): Carga el fichero de propiedades como si estuviera ejecutándose el algoritmo desde un conjunto de .class.

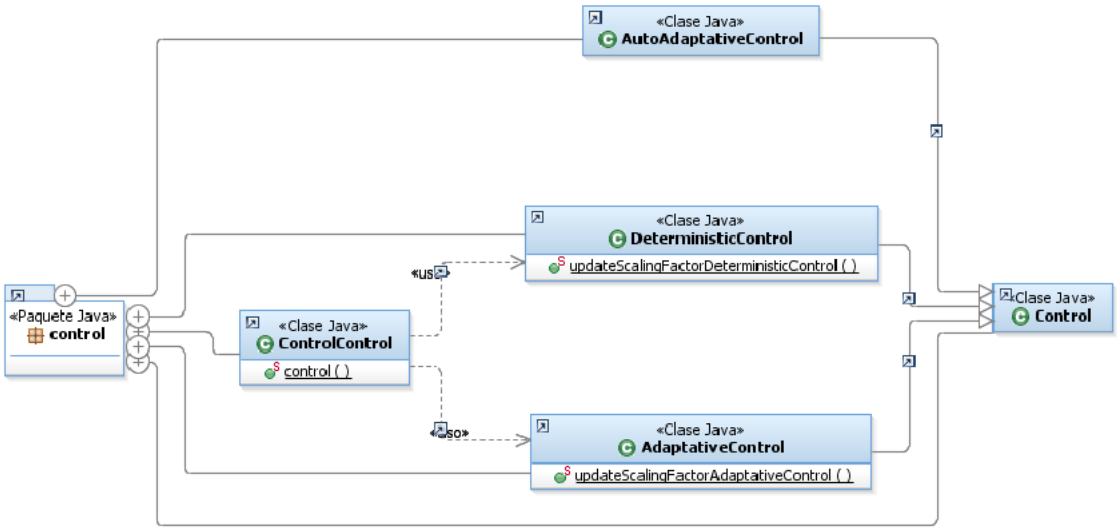


Figura 7: Paquete control

Paquete Control

Contiene la lógica de negocio referente al control del factor de escalado, cuyo contenido se muestra en la figura 7.

Clase ControlControl

Descripción: Determina el control del factor de escalado a aplicar e invoca a dicho control.

Métodos principales:

- public static void control(int iteration, Individual[] population, double initialEntropy): Determina el control a aplicar y lo ejecuta.

Clase AutoAdaptativeControl

Descripción: Modela el control autoadaptativo.

Clase DeterministicControl

Descripción: Implementa el control determinista.

Métodos principales:

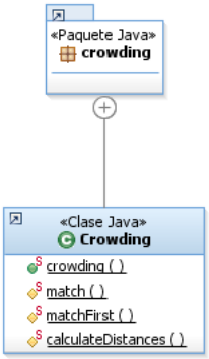


Figura 8: Paquete crowding

- public static void updateScalingFactorDeterministicControl(int iteration): Actualiza el valor del factor de escalado, aplicando la regla del control determinista.

Clase AdaptativeControl

Descripción: Implementa el control adaptativo basado en la entropía de la población.

Métodos principales:

- public static void updateScalingFactorAdaptativeControl(Individual[] population, double initialEntropy): Aplica el control adaptativo al factor de escalado.

Clase Control

Descripción: Clase de la que heredan todas las implementaciones de controles del factor de escalado. Sirve como punto de referencia para ampliar el sistema con atributos o métodos comunes a todos los controles.

Paquete Crowding

Posee la funcionalidad de la técnica del Agrupamiento, cuyo contenido se muestra en la figura 8.

Clase Crowding

Descripción: Implementa la técnica del Agrupamiento Generalizado.

Métodos principales:

- public static Individual[] crowding(Individual[] oldPop, int S): Realiza el agrupamiento sobre una población.
- protected static List¡MatchVO¿match(double[][] distance, Individual[] parent, double[][] child, double sf0, double sf1): Devuelve las parejas padre e hijo que minimizan la función distancia.
- protected static boolean matchFirst(double[][] distance): Detecta si la suma de las distancias entre el primer padre y el primer hijo, y el segundo padre y el segundo hijo, es menor que la suma de las distancias enter el primer padre y el segundo hijo, y el segundo padre y el primer hijo.
- protected static double[][] calculateDistances(Individual[] parent, double[][] child): Calcula las distancias entre padres e hijos.

Paquete Crossover

Agrupa los diferentes artefactos relacionados con el cruce de individuos, cuyo contenido se muestra en la figura 9.

Clase CrossoverControl

Descripción: Determina el cruce a aplicar de entre los existentes.

Atributos principales:

- private static CrossoverInterface uniformCrossover: Cruce uniforme.

Métodos principales:

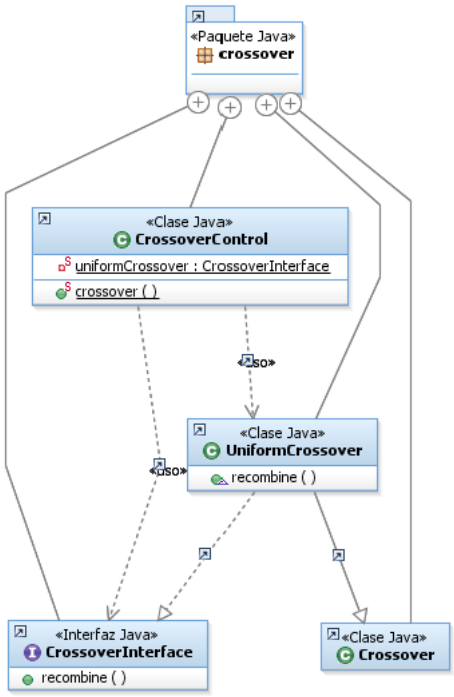


Figura 9: Paquete crossover

- public static double[][] crossover(Individual parent1, Individual parent2): Aplica el cruce seleccionado en el fichero de configuraciones.

Clase UniformCrossover

Descripción: Implementa el operador de recombinación uniforme.

Métodos principales:

- public static double[][] crossover(Individual parent1, Individual parent2): Realiza el cruce entre dos padres pasados como parámetros.

Clase Crossover

Descripción: Clase de la que heredan todos los cruces. Supone un punto de referencia para la posible ampliación de elementos comunes entre los operadores de cruce.

Interfaz CrossoverInterface

Descripción: Interfaz que han de implementar todos los operadores de recombinación.
Métodos principales:

- public double[][] recombine(Individual f1, Individual f2): Recombina dos padres devolviendo los genes de sus hijos.

Paquete Fitness

Almacena los artefactos referentes al cálculo de la función de adaptación, cuyo contenido se muestra en la figura 10.

Clase FitnessControl

Descripción: Determina la función de adaptación a aplicar y la ejecuta.
Atributos principales:

- private static FitnessInterface f1: Implementación de la función F1 de Ballester y Carter.
- private static FitnessInterface f2: Implementación de la función F2 de Ballester y Carter.
- private static FitnessInterface f3: Implementación de la función F3 de Ballester y Carter.
- private static FitnessInterface f4: Implementación de la función F4 de Ballester y Carter.
- private static FitnessInterface f5: Implementación de la función F5 de Ballester y Carter.
- private static FitnessInterface f6: Implementación de la función F6 de Ballester y Carter.

Métodos principales:

- public static double calculateFitness(double[] genes): Devuelve el valor de adaptación de un determinado conjunto de genes de la función configurada en el fichero de propiedades.

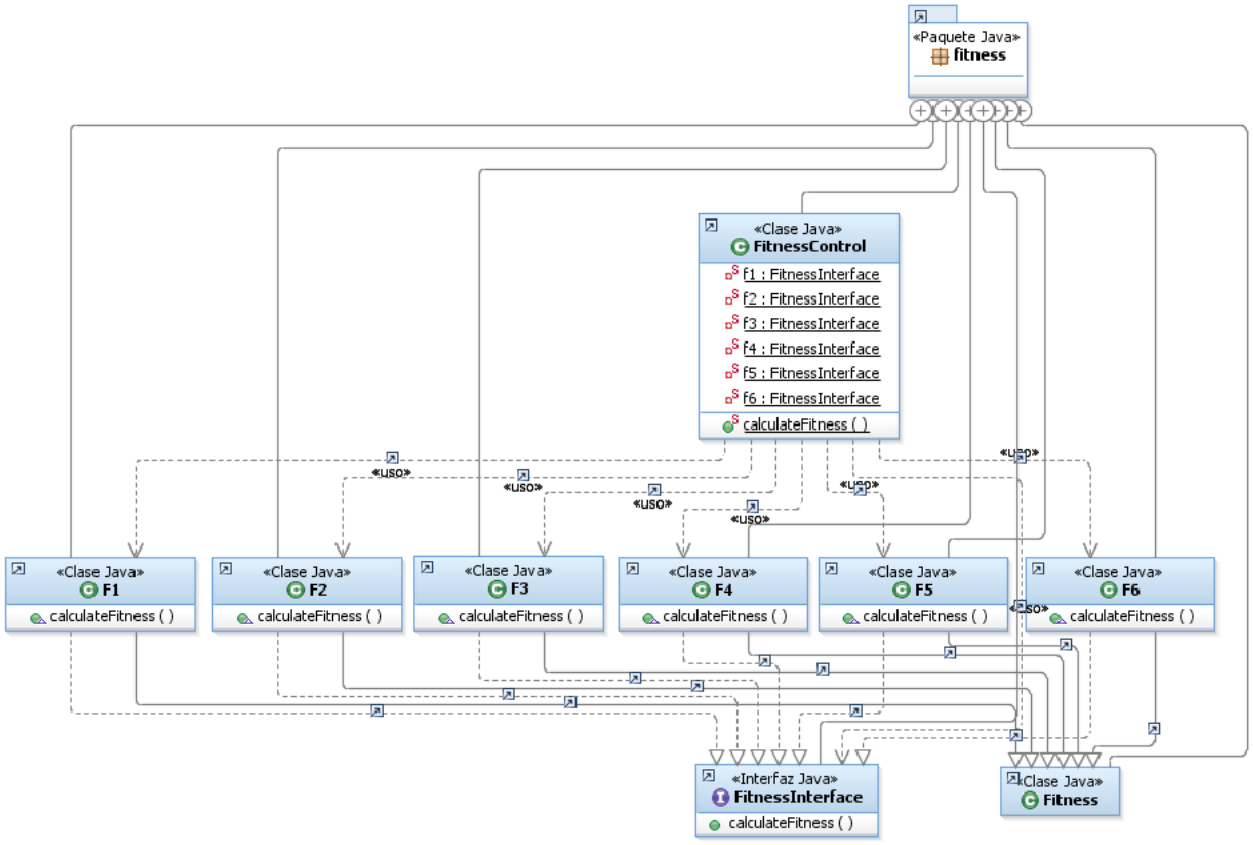


Figura 10: Paquete fitness

Clases F1-F6

Descripción: Implementan las funciones de Ballester y Carter.

Métodos principales:

- public double calculateFitness(double[] x): Calcula el valor de adaptación para un conjunto de genes.

Clase Fitness

Descripción: Clase abstracta de la que heredan todas las clases de cálculo de las funciones de adaptación. Representa un punto de ampliación de comportamiento y atributos comunes a dichas clases.

Interfaz FitnessInterface

Descripción: Define los métodos que las clases que codifican las funciones de adaptación deben implementar.

Métodos principales:

- public double calculateFitness(double[] x): Devuelve el valor de adaptación de un individuo cuyos genes se pasan como parámetro.

Paquete Individual

Contiene la clase que modela a un individuo de la población, cuyo contenido se representa en la figura 11.

Clase Individual

Descripción: Representa un individuo de la población.

Atributos principales:

- private double[] genes: Los genes del individuo.
- private double fitness: Valor de adaptación.

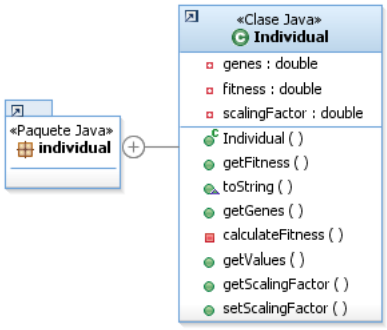


Figura 11: Paquete individual

Figura 12: Paquete initialization

- private double scalingFactor: Factor de escalado utilizado en el control autoadaptativo.

Métodos principales:

- public Individual(double[] genes): Constructor.
- private void calculateFitness(): Calcula el valor de adaptación del individuo.

Paquete Initialization

Agrupa la funcionalidad de inicialización de la población, cuyo contenido se muestra en la figura 12.

Clase PopulationInitialization

Descripción: Clase de utilería que contiene el método de inicialización de poblaciones para las diferentes ejecuciones del AG.

Métodos principales:

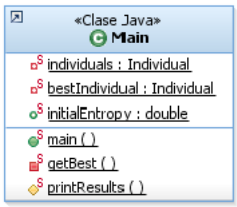


Figura 13: Clase Main

- public static Individual[] initializePoblation(): Devuelve una población inicializada de manera aleatoria, basándose en los parámetros del fichero de configuración.

Clase Main

Descripción: Clase principal del sistema, lleva a cabo la inicialización del sistema, la carga del fichero de propiedades y las diferentes ejecuciones del AG. Además mantiene la población actual en cada iteración.

Atributos principales:

- private static Individual[] individuals: La población.
- private static Individual bestIndividual: Individuo mejor adaptado de cada ejecución.
- public static double initialEntropy: Entropía inicial. Sólo usado en caso de seleccionar el control adaptativo.

Métodos principales:

- public static void main(String[] args): Método principal de ejecución.
- private static Individual getBest(): Devuelve el mejor individuo de la población actual.
- protected static void printResults(Individual[][] results): Imprime los resultados por pantalla y crea los ficheros de salida.

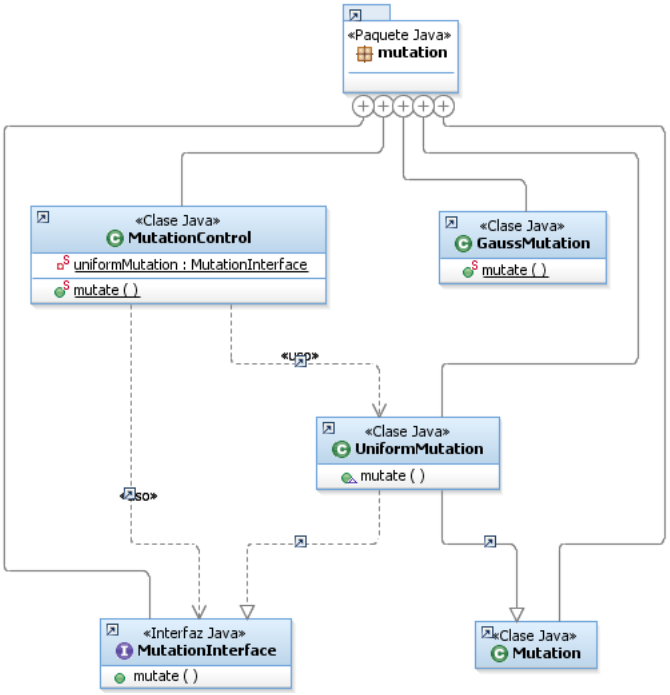


Figura 14: Paquete mutation

Paquete Mutation

Representa el encapsulamiento de la funcionalidad de mutación, cuyo contenido se presenta en la figura 14.

Clase MutationControl

Descripción: Determina el tipo de mutación a aplicar en función del valor en el fichero de propiedades y ejecuta dicha mutación.

Atributos principales:

- private static MutationInterface uniformMutation: Representa la mutación uniforme.

Métodos principales:

- public static void mutate(double[] child): Muta los genes de un individuo según el tipo de mutación seleccionada.

Clase GaussMutation

Descripción: Mutación gaussiana adaptada al método de control autoadaptativo.

Métodos principales:

- public static double mutate(double sf): Devuelve el factor de escalado mutado.

Clase UniformMutation

Descripción: Implementa la mutación uniforme.

Métodos principales:

- public void mutate(double[] child): Método que ejecuta la mutación uniforme.

Clase Mutation

Descripción: Clase abstracta de la que heredan todos los operadores de mutación. Sirve como punto de extensión para atributos y funcionalidades comunes a todos los operadores de mutación.

Interfaz MutationInterface

Descripción: Interfaz común de los operadores de mutación.

Métodos principales:

- public void mutate(double[] child): Método de ejecución de la mutación que codifican las clases que implementan esta interfaz.

Paquete Replacement

Contiene la lógica de negocio del reemplazo, cuyo contenido se modela en la figura 15.

Clase Replacement

Descripción: Modela el torneo utilizado en el mecanismo de Agrupamiento Generalizado.

Métodos principales:

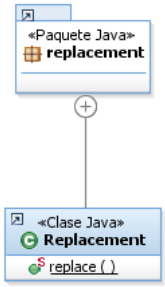


Figura 15: Paquete replacement

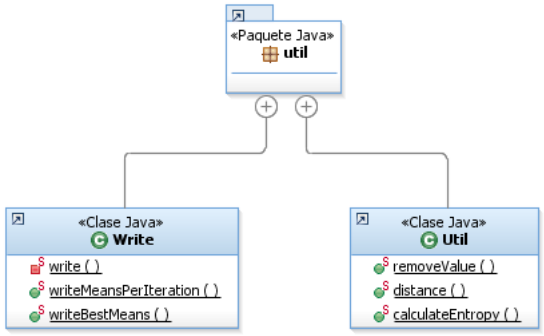


Figura 16: Paquete util

- public static boolean replace(Individual parent, Individual child): Devuelve *true* si el hijo es el ganador del torneo.

Paquete Util

Posee un conjunto de clases de utilería, representadas en la figura 16.

Clase Util

Descripción: Clase con diferentes métodos de utilería.

Métodos principales:

- public static Individual[] removeValue(Individual[] old, Individual value): Elimina una instancia de Individual de un array pasado como parámetro.

- public static double distance(double[] gen1, double[] gen2): Calcula la distancia entre dos genotipos.
- public static double calculateEntropy(Individual[] individuals): Calcula la entropía normalizada de una población.

Clase Write

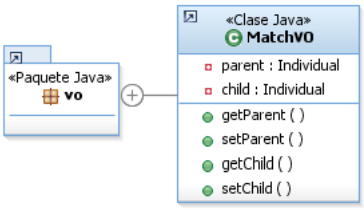
Descripción: Contiene los métodos de escritura en fichero de los resultados de la ejecución de la aplicación.

Métodos principales:

- private static void write(String file, String[] text): Escribe en el fichero pasado como parámetro, las cadenas que también se le suministran.
- public static void writeMeansPerIteration(double[]meansPerGeneration): Escribe las medias del mejor fitnes de cada generación.
- public static void writeBestMeans(Individual[] bestsOfExecutions, double bestMean): Escribe los mejores individuos de cada ejecución del AG.

Paquete VO

Contiene los objetos de valor.



Clase MatchVO

Descripción: Modela un emparejamiento entre un padre y un hijo.

Atributos principales:

- private Individual parent: El padre.
- private Individual child: El hijo.

4.3.5. Vista de Procesos

En esta sección se detalla el comportamiento dinámico del sistema. Se describirá la realización de los casos de uso, mediante diagramas de secuencia. En primer lugar se muestra el diagrama de secuencia del análisis. Posteriormente, dicho diagrama es refinado, con menor nivel de abstracción mostrando las interacciones entre las clases de diseño, que dan lugar a al ejecución de la aplicación.

4.3.5.1. Realización de Casos de Uso (Análisis)

La figura 17 muestra los pasos realizados por la aplicación cuando esta es ejecutada a muy alto nivel. Dichos pasos son descritos con mayor detalle a continuación:

1. Se realiza la inicialización, en la que se validan los parámetros de entrada y se cargan los valores del fichero de propiedades en el sistema.
2. Comienzan las diferentes ejecuciones del AG. Para cada ejecución se realizan los siguientes pasos:
   a) Se inicializan todas las variables temporales y se restauran a sus valores originales.
   b) Se crea una nueva población aleatoriamente distribuida.
   c) Para cada generación a evolucionar la población original se realiza la siguiente secuencia:
      1) Se ejecuta la técnica de Agrupamiento Generalizado produciendo una nueva población, que reemplaza a la anterior.
      2) Se almacena el mejor individuo de dicha población para el cálculo de medias realizado a posteriori.

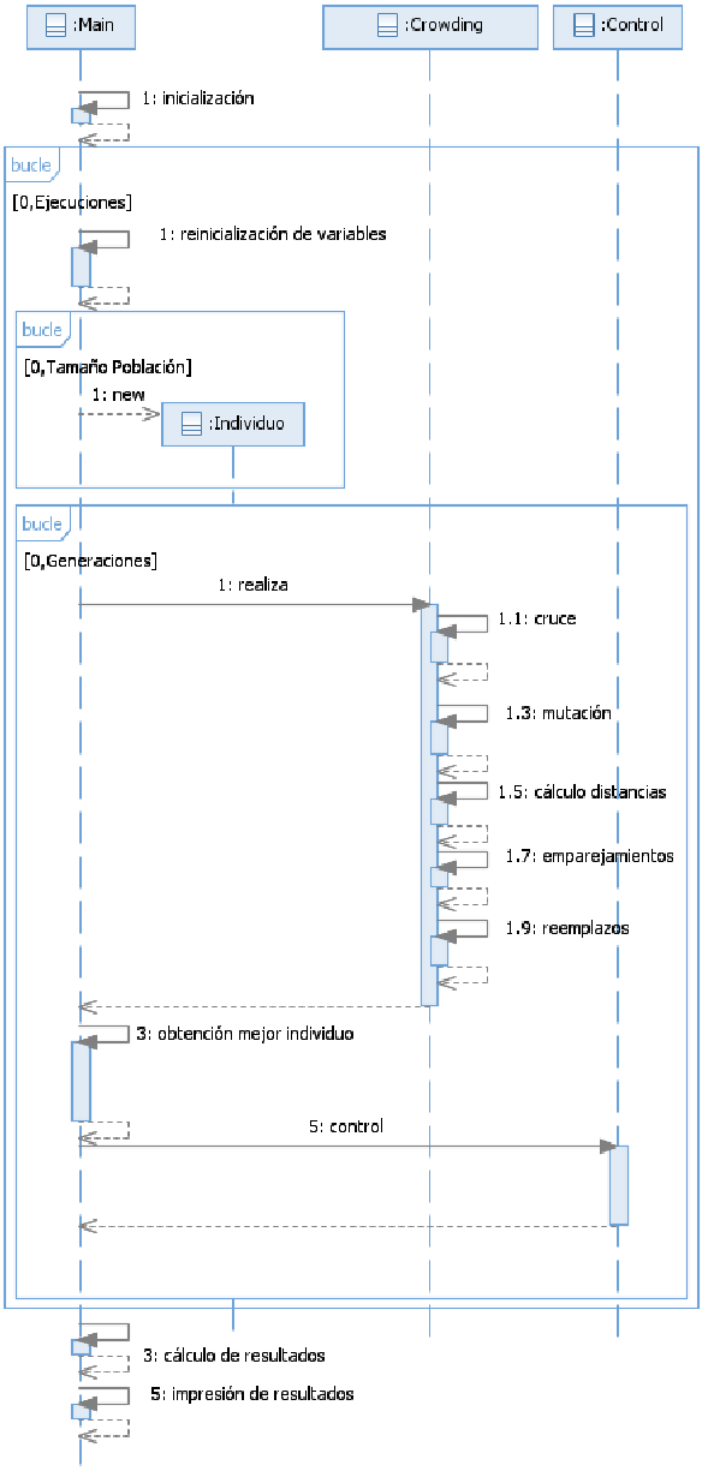


Figura 17: Diagrama de secuencia del análisis

3) Se ejecuta el control del factor de escalado seleccionado en el fichero de propiedades.

3. Una vez finalizadas las ejecuciones del AG, se calculan las medias, se imprimen por pantalla y se almacenan dichos valores en los ficheros de texto plano.

4.3.5.2. Realización de Casos de Uso (Diseño)

La figura 18 muestra los pasos realizados por la aplicación cuando esta es ejecutada. Dichos pasos son descritos a continuación con mayor grado de detalle:

1. En primer lugar se validan los argumentos de entrada. El primer parámetro indica el fichero de configuración. El segundo parámetro indica el modo de ejecución. Si el segundo parámetro es un 0, indica que la ejecución se realiza desde ficheros .class. Si el segundo parámetro es un 1, indica que la ejecución se realiza desde un fichero .jar.

2. El sistema lee el fichero de propiedades y carga los valores de dichos registros en los atributos de la clase Constants.

3. A continuación comienza un bucle que se ejecutará tantas veces como se haya definido en el fichero de configuración. Este bucle incial representa el número de ejecuciones del AG que se llevarán a cabo de cara a calcular las medias de los resultados obtenidos en cada ejecución y así obtener resultados más fiables.

   a) Inicializa el valor del factor de escalado al valor original. De esta forma se reinicia dicho valor ya que podía haber sido alterado por los diferentes controles.

   b) Inicializa el contador de generaciones junto con el valor de entropía inicial por si se usa el control adaptativo basado en entropía.

   c) Inicializa una nueva población y obtiene el mejor individuo de la misma.

   d) Si se emplea el control adaptativo se calcula la entropía inicial.

e) Comienza el bucle que se ejecutará tantas veces como generaciones se desee evolucionar la población inicial.

1) Se realiza el Agrupamiento Generalizado. En dicha técnica se realiza la recombinación, mutación, calculo de distancias, emparejamientos padre-hijo y aplicación de la regla de reemplazo.
2) Se obtiene el mejor individuo de la nueva población.
3) Se aplica la estrategia de control del factor de escalado seleccionada en caso de ser determinística o adaptativa. Para el caso del control autoadaptativo, cada individuo ya posee su propio factor de escalado por lo que ya ha sido concebido y alterado mediante las fases de recombinación y mutación respectivamente.

4. Se calculan las medias, se imprimen los resultados y se almacenan en ficheros de texto plano.

4.3.6. Vista de Despliegue

4.3.6.1. Introducción

La vista de despliegue presenta la infraestructura necesaria para la instalación y ejecución de la herramienta.

Se presenta aquí, mediante el modelo de despliegue, la arquitectura técnica de la aplicación indicando los nodos presentes en la infraestructura tecnológica esperada y la localización de los componentes de dichos nodos. Las máquinas físicas, componentes y procesadores son representados mediante nodos, mientras que las construcciones internas tales como subsistemas, etc. son reflejados mediante componentes y artefactos o nodos adicionales.

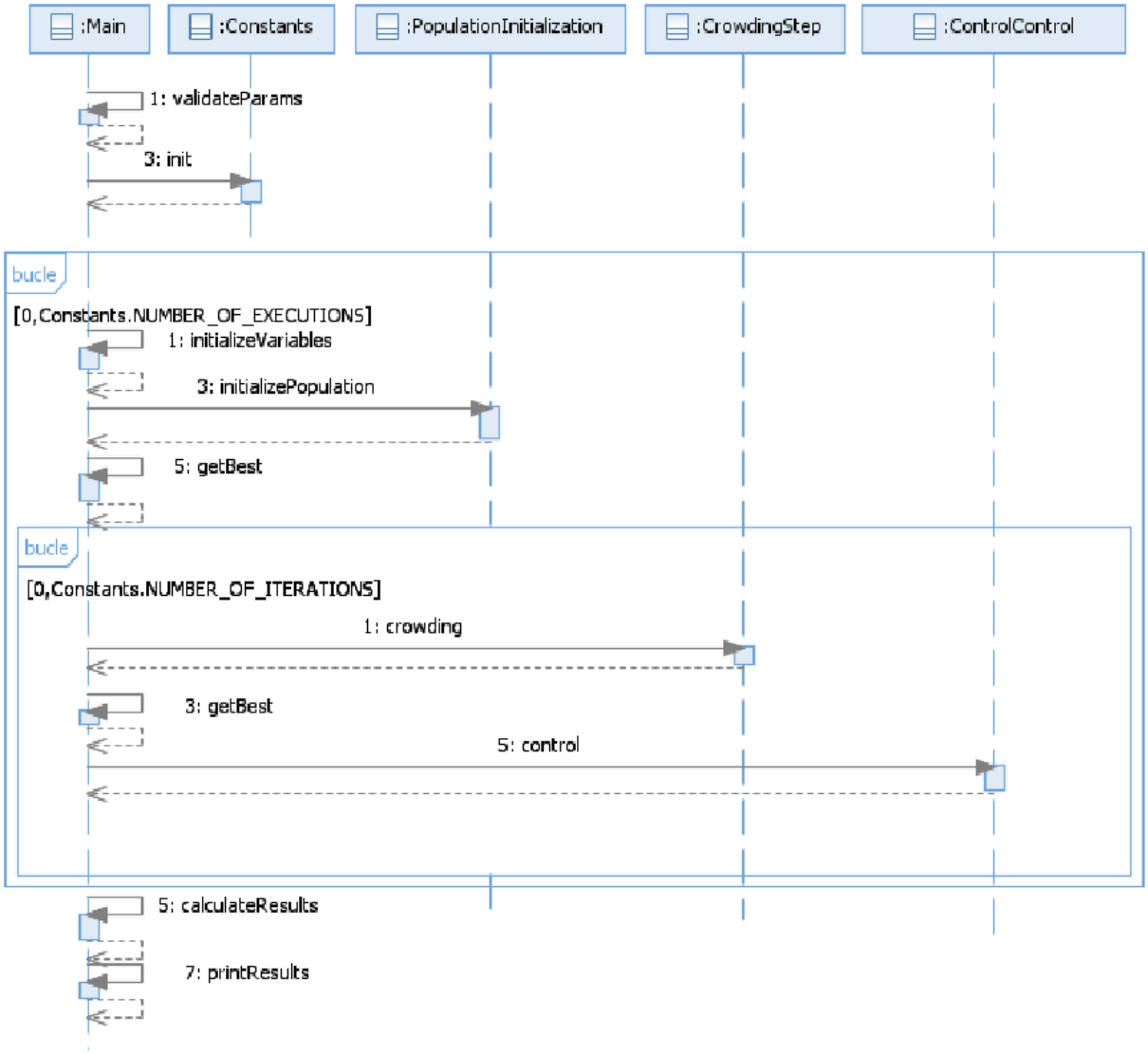


Figura 18: Diagrama de secuencia de Diseño

4.2.6.2. Arquitectura Técnica

El modelo de la figura 19 representa la estructura en la cual el sistema funciona en producción en una única máquina. Dispone de un ejecutable denominado GC.jar. El ejecutable contiene el *bytecode* de Java listo para su ejecución a través de línea de comandos. Dicho ejecutable tiene como dependencia el fichero de configuración mostrado también en la figura.

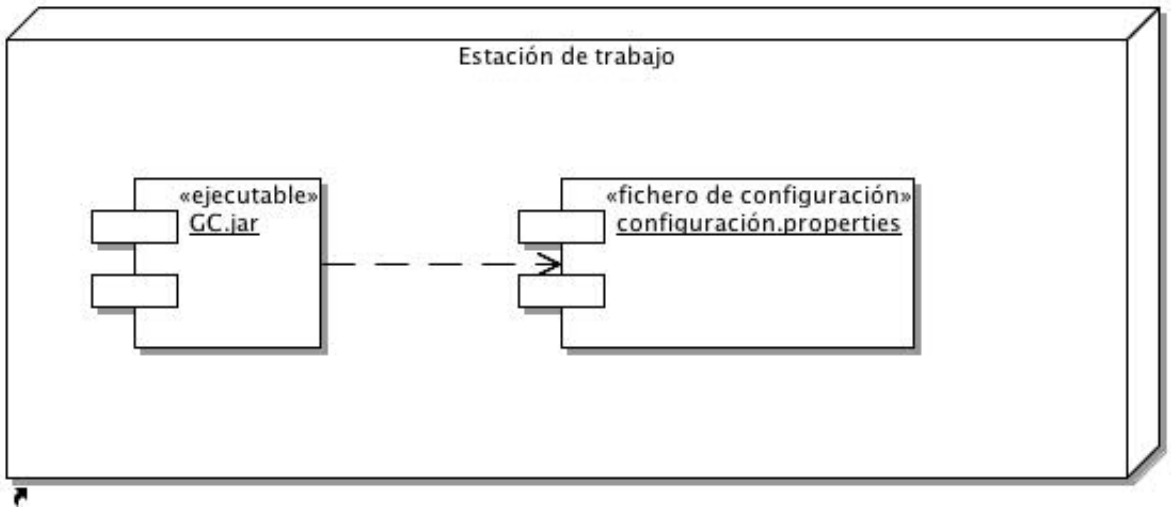


Figura 19: Diagrama de despliegue

La figura 20 muestra la secuencia de pasos durante la ejecución de la aplicación. Dicha secuencia sigue los siguientes pasos:

1. El usuario configura el fichero de propiedades para ejecutar la aplicación con los parámetros deseados.

2. El usuario ejecuta la aplicacióna través de la línea de comandos.

3. La aplicación es ejecutada.

4. La aplicación genera los ficheros de salida con los resultados.

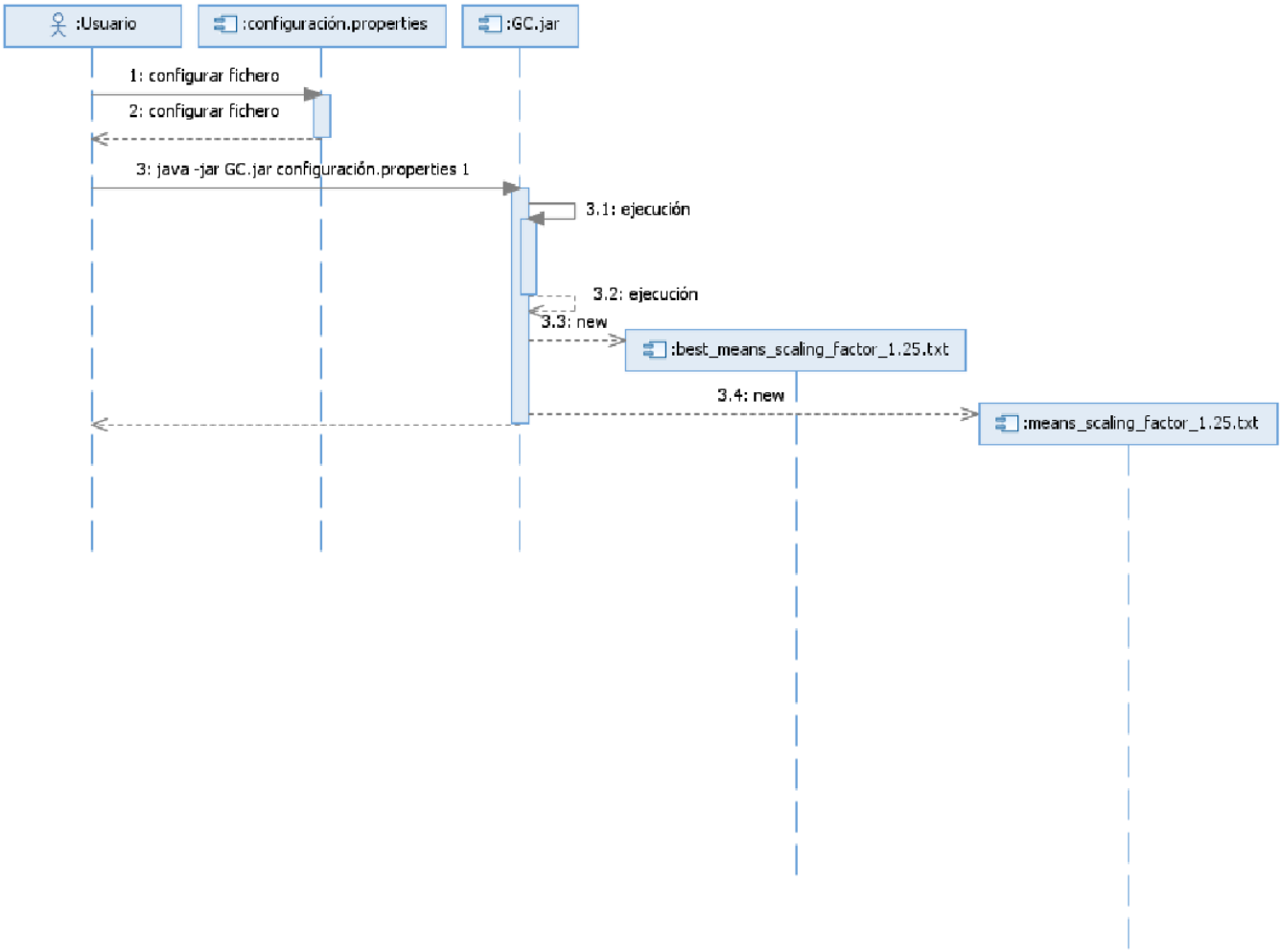


Figura 20: Diagrama de despliegue en ejecución

4.3.7. Vista de Datos

Esta sección describe el fichero de configuración de propiedades.

4.3.7.1. Introducción

En esta sección se detalla el fichero de configuración, que sirve como entrada a la ejecución de la aplicación. A partir del mismo se definen los parámetros descritos en la subsección Alcance de la sección Especificación de requisitos del presente capítulo.

4.3.7.2. Fichero de Configuración

El fichero de configuración contiene los parámetros que determinan la ejecución de la aplicación y es tomado como entrada en la inicialización de la aplicación. Posee un conjunto de parejas clave-valor que son leídas y cargadas en memoria una única vez durante toda la ejecución de la aplicación. De esta forma se evita tener que realizar una lectura continua del fichero de propiedades lo que en recursos computacionales es más costoso que un acceso a memoria principal.

A continuación se describe cada una de las propiedades de dicho fichero.

- La función de adaptación se determina a través del parámetro *problem*, cuyos valores son presentados en la tabla 5.
- El número de variables empleadas en cada problema se designa a través de la clave *number.of.variables* como muestra la tabla 6.
- El número de ejecuciones del AG se designa a través de la clave *number.of.executions* como muestra la tabla 7.
- El número de generaciones a evolucionar la población inicial del AG se designa a través de la clave *number.of.iterations* como muestra la tabla 8.

# Number of problem
# 1 = Sphere
# 2 = Schwefel [-500,500]
# 3 = F1 [-10,10]
# 4 = F2 [-10,10]
# 5 = F3 [-10,10]
# 6 = F4 [-10,10]
# 7 = F5 [-10,10]
# 8 = F6 [-10,10]
# 9 = Rastrigin
problem = 8

Tabla 5: Número de problema

# Number of variables
number.of.variables = 2

Tabla 6: Número de variables

# Number of executions
number.of.executions = 1000

Tabla 7: Número de ejecuciones

# Number of iterations
number.of.iterations = 20000

Tabla 8: Número de iteraciones

- El tamaño de la población se designa a través de la clave *population.size* como muestra la tabla 9.

```
# Population size
population.size = 20
```

Tabla 9: Tamaño de la población

- El valor máximo de las variables se designa a través de la clave *max.variable.value* como muestra la tabla 10 y constituye un intervalo simétrico entorno al 0.

```
# Max variable value, the interval will be [-max.variable.value, +max.variable.value]
max.variable.value = 10
```

Tabla 10: Máximo valor de las variables

- El tipo de cruce se designa a través de la clave *crossover* como muestra la tabla 11.

```
# Crossover method
# 1 = One point crossover
# 2 = Uniform crossover
# 3 = Arithmetic crossover
crossover = 2
```

Tabla 11: Tipo de cruce

- La probabilidad de cruce se designa a través de la clave *crossover.probability* como muestra la tabla 12.

```
# Crossover probability
crossover.probability = 1
```

Tabla 12: Probabilidad de cruce

- El método de mutación es designado a través de la clave *mutation* como muestra la tabla 13.

```
# Mutation method
# 1 = Uniform mutation
# 2 = Gaussian mutation
mutation = 1
```

Tabla 13: Mutación

- La probabilidad de mutación es designada a través de la clave *mutation.probability* como muestra la tabla 14.

```
# Mutation probability
mutation.probability = 0.2
```

Tabla 14: Probabilidad de mutación

- El valor del factor de escalado inicial es designado a través de la clave *scaling.factor* como muestra la tabla 15. Para el caso del control autoadaptativo representa el máximo valor del factor de escalado de los individuos.

```
# Scaling factor
# 0 = Deterministic Crowding
# 1 = Probabilistic Crowding
# auto-adaptative control scaling.factor = Max sf
scaling.factor = 1.25
```

Tabla 15: Factor de escalado

- El tipo de control del factor de escalado es designado a través de la clave *control* como muestra la tabla 16.

```
# scaling factor control
# 0 = fixed
# 1 = deterministic control (till sf=0)
# 2 = adaptative control
# 3 = auto-adaptative control
control = 3
```

Tabla 16: Control del factor de escalado

4.4. Pruebas

Los artefactos software producidos han sido probados mediante pruebas unitarias y funcionales. Además, durante todas las fases del desarrollo del mismo se compararon los resultados obtenidos con los calculados por parte del tutor. De esta forma se estableció una metodología de validación y verificación de resultados.

Para las pruebas unitarias y funcionales se utilizó la biblioteca TestNG. TestNG es un marco de trabajo para pruebas que trabaja con Java y está basado en JUnit, pero introduce nuevas funcionalidades, que lo hacen más poderoso y fácil de usar, tales como anotaciones y configuraciones flexibles de pruebas. Además, está soportado por herramientas y plugins importantes como Eclipse.

Capítulo 5: EVALUACIÓN

5.1. Introducción

Esta sección describe los experimentos realizados para el análisis del rendimiento de las tres técnicas de control del factor de escalado en el contexto del Agrupamiento Generalizado:

1. Control determinista del factor de escalado.
2. Control adaptativo del factor de escalado basado en la diversidad de la población.
3. Control autoadaptativo del factor de escalado.

Estas técnicas de control son comparadas con experimentos en los que se aplica el factor de escalado de manera fija para todas las generaciones.

Para cada técnica de control, el usuario determina un factor de escalado original ϕ_0, de manera previa a la ejecución del AG. Cuando se emplea un valor fijo del factor de escalado, se determina $\phi(t) = \phi_0$ para $t \geq 1$; en otras palabras, ϕ_0 permanece constante para todas las generaciones del AG. Cuando se emplea el control determinista lineal, el factor de escalado usado es $\phi(t) = \phi_0 - \{k \times (t-1)\}$. En el caso del control adaptativo basado en diversidad, ϕ_0 es el valor inicial del factor de escalado en la primera generación. Finalmente, en el caso del control autoadaptativo, ϕ_0 es el máximo valor del factor de escalado que puede ser asociado a los individuos.

5.2. Optimización de Funciones Reales de Varias Variables

Dada una función n-dimensional, $f\colon \Re^n \longrightarrow \Re$, la optimización global de dicha función consiste en determinar $x^* \in \Re^n$ tal que $f(x^*) \geq f(x)$ para todo $x \neq x^*$ con $x \in \Re^n$. Esta

definición corresponde a un problema de maximización. En caso de minimización se considera $f(x^*) \leq f(x)$.

Los experimentos de esta sección se llevaron a cabo mediante la codificación de n variables reales como genotipo, sin selección de padres para mantener la diversidad, con emparejamiento de padres aleatorio, con cruce uniforme [16] de probabilidad de cruce igual a la unidad para cada pareja de padres, con mutación uniforme (que escoge un valor uniformemente aleatorio dentro del intervalo en el que la variable real se define) y con selección de supervivientes generacional.

Se llevaron a cabo una serie de experimentos preliminares con Agrupamiento Generalizado, donde se emplearon valores del factor de escalado fijos en funciones multimodales. A través de estos experimentos se identificaron dos grupos de funciones:

- Grupo 1

  Funciones para las que un valor bajo de ϕ obtiene mejor rendimiento. Esto significa que una estrategia más explotativa del Agrupamiento Generalizado obtiene mejores resultados.

- Grupo2

  Funciones para las que $\phi = 0$ no produce el mejor resultado. Esto significa que una estrategia más explorativa del Agrupamiento Generalizado no produce necesariamente peores resultados.

Nuestros experimentos preliminares muestran que las funciones F_1, F_3 y F_6 de Ballester y Carter pertenecen al Grupo 1, mientras que F_2, F_4, y F_5 pertenecen al Grupo 2.

Para cada una de las funciones de Ballester y Carter, se han realizado cuatro casos de prueba:

1. Sin control, valor del factor de escalado fijo.

2. Con control determinista lineal.

3. Con control adaptativo basado en la entropía de la población.

4. Con control autoadaptativo.

Para cada uno de estos cuatro casos se han llevado a cabo mil ejecuciones del AG con diferentes probabilidades de mutación $P_M = \{0, 0.00625, 0.0125, 0.025, 0.05, 0.1, 0.2\}$. Además, para cada valor de dicha probabilidad se han probado los siguientes valores de ϕ_0, $\phi_0 = \{0, 0.25, 0.5, 0.75, 1, 1.25\}$.

Debido a la extensión de los experimentos con las seis funciones, a continuación se explica sólo una función identificativa de cada grupo, ya que el comportamiento se repite para el resto de funciones analizadas, según el grupo al que pertenecen. Del Grupo 1 se toma como ejemplo F_1. Del Grupo 2 se toma como ejemplo F_2. Ambas se definene en el rango $-10 \leq x_1, x_2 \leq 10$ de la siguiente manera:

$$F_1(x_1, x_2) = x_1^2 + 2x_2^2 - 0{,}3\cos(3\pi x_1) - 0{,}4\cos(4\pi x_2) + 0{,}7$$

$$F_2(x_1, x_2) = x_1^2 + 2x_2^2 - 0{,}3\cos(3\pi x_1)\cos(4\pi x_2) + 0{,}3.$$

Debido al objetivo de minimizar estas dos funciones, se transforman para llevar a cabo maximización y se hace que todos sus valores sean positivos en el rango $[-10, 10]$, como requisito de la ecuación 1. Las funciones transformadas quedan de la siguiente manera:

$$F_1^*(x_1, x_2) = -F_1(x_1, x_2) + 301{,}4$$

$$F_2^*(x_1, x_2) = -F_2(x_1, x_2) + 300{,}6.$$

De esta manera, el valor óptimo para F_1^* es 301.4, y el valor óptimo para F_2^* es 300.6. Por simplicidad, en el resto del capítulo se hará referencia a estos óptimos como F_1^* y F_2^*.

A continuación en la sección Discusión se explican los resultados obtenidos. Para cada experimento se utilizó un tamaño de la población empleado de $M = 20$, teniendo cada individuo dos variables reales por genes. El AG se ejecutó mil veces, sin ningún tipo de elitismo, en el intervalo $[-10, 10]$, que fue discretizado en cien subintervalos para calcular la entropía de la población, en el caso del control adaptativo.

5.3. Discusión

Todas las figuras presentadas a continuación representan la evolución, generación a generación, de la media del mejor valor de adaptación (considerando el mejor individuo encontrado

desde la primera generación hasta la actual). Además, cada figura representada consta de cuatro gráficas: factor de escalado fijo (arriba a la izquierda), control determinista lineal (arriba a la derecha), control adaptativo (abajo a la izquierda) y control autoadaptativo (abajo a la derecha). Cada gráfica contiene los casos de prueba para $\phi_0 = \{0, 0.25, 0.5, 0.75, 1, 1.25\}$. Por tanto, todas las gráficas siguen el mismo patrón de representación independientemente de la función que representan, F_1 o F_2.

Por una lado, las figuras 21, 22 y 23 representan las medias del mejor valor de adaptación encontrado para F1, con probabilidades de mutación $P_M = \{0.0125, 0.025, 0.05\}$ respectivamente. Por otro lado, las figuras 24, 25 y 26 representan las medias del mejor valor de adaptación encontrado para F_2, con las mismas probabilidades de mutación que para F_1.

Si se comparan los resultados obtenidos entre el experimento de mantener fijo el factor de escalado, con los obtenidos al aplicar cualquier tipo de control, en cualquiera de las figuras, se observa como los resultados de aplicar cualquier control son mejores que los que aporta el método de mantener ϕ constante. Por ejemplo, para F_1 en la figura 21 se observa cómo el enfoque decreciente lineal (gráfica superior derecha), el enfoque de control adaptativo de ϕ basado en diversidad (gráfica inferior izquierda) y el enfoque autoadaptativo de la misma figura (gráfica inferior derecha), producen mejores resultados que el Agrupamiento Generalizado con valor fijo de ϕ (gráfica superior izquierda). Este efecto puede ser igualmente comprobado en el resto de figuras. Por tanto, los tres métodos de control rinden mejor que el método de mantener ϕ fijo, lo que constituye una importante conclusión.

Otro punto destacable es que para el caso de F_2 con control determinista lineal, se obtienen los mejores valores de adaptación en comparación con los otros tres casos, tal y como se observa en las figuras 24, 25 y 26. Esto se debe a que la constante de decrecimiento aplicada favorece la exploración más que la explotación. En este caso la constante toma el valor 1/20000, que es la unidad dividida el número de generaciones a evolucionar la población inicial, para que de esta forma ϕ sea igual a cero por primera vez en la última generación del AG. Nótese como esto no sucede en el caso de F1, en el que el los resultados obtenidos con el factor de escalado igual a cero son los mejores. Esto es así dado que F1 pertenece al Grupo 1 y, tal como ya hemos comentado,

la constante de decrecimiento utilizada favorece la exploración frente a la exploración.

Tanto para la función F_1 como para la función F_2, los controles adaptativo y autoadaptativo ofrecen mejores resultados que utilizando un factor de escalado fijo. Mientras que el control adaptativo basado en diversidad produce para F_1 mejores resultados que el control autoadaptativo (figuras 21, 22 y 23), ocurre lo contrario para el caso F2 (figuras 24, 25 y 26). Por tanto, la comparación entre éstos dos controles no permite establecer cuál es el mejor, ya que en unos casos es mejor un método y en otros casos es mejor el otro método.

Como conclusión de los dos puntos anteriores, podemos afirmar que la comparación entre los diferentes métodos de control no permite establecer cuál de los tres es el mejor, ya que en unos casos es mejor un método y en otros casos es mejor otro método distinto.

Los métodos adaptativo y autoadaptativo son robustos en el sentido de que no requieren que el usuario introduzca parámetros adicionales y se comportan aceptablemente bien en la mayoría de los experimentos. No ocurre así con el método de decrecimiento lineal, que se comporta peor en el caso en que se necesita más explotación (F_1); es decir, habría que variar la constante de decaimiento para que el método de decaimiento lineal se comportara bien en el caso de F_1. Esta necesidad de determinar previamente a la ejecución del AG qué valor de la constante de decrecimiento sería el mejor, hace que el método de decrecimiento lineal no sea robusto.

Finalmente, cabe destacar el efecto gráfico que producen los métodos más robustos, el control adaptativo para F1 y control autoadaptativo para F2. En este efecto gráfico se observa cómo las líneas obtenidas al variar el valor de ϕ permanecen cercanas unas a las otras. Conforme aumenta la probabilidad de mutación dichas líneas se aproximan unas más a otras como si de un acordeón cerrándose se tratara. Dichos resultados se observan al comparar la figura 21, con $P_M = 0.0125$ con la figura 22, con $P_M = 0.025$, con la figura 23, con con $P_M = 0.05$. Se observa por tanto como a mayor probabilidad de mutación ofrece resultados más similares para diferentes valores de ϕ. Además, a mayor mutación se obtienen mejores valores de adaptación. Este efecto sucede tanto con funciones del Grupo 1 como con funciones del Grupo 2.

Por último, a modo de resumen, la tabla 17 recoge los resultados de las medias de los mejores valores de adaptación encontrados en la última generación del AG ejecutado en los experimentos

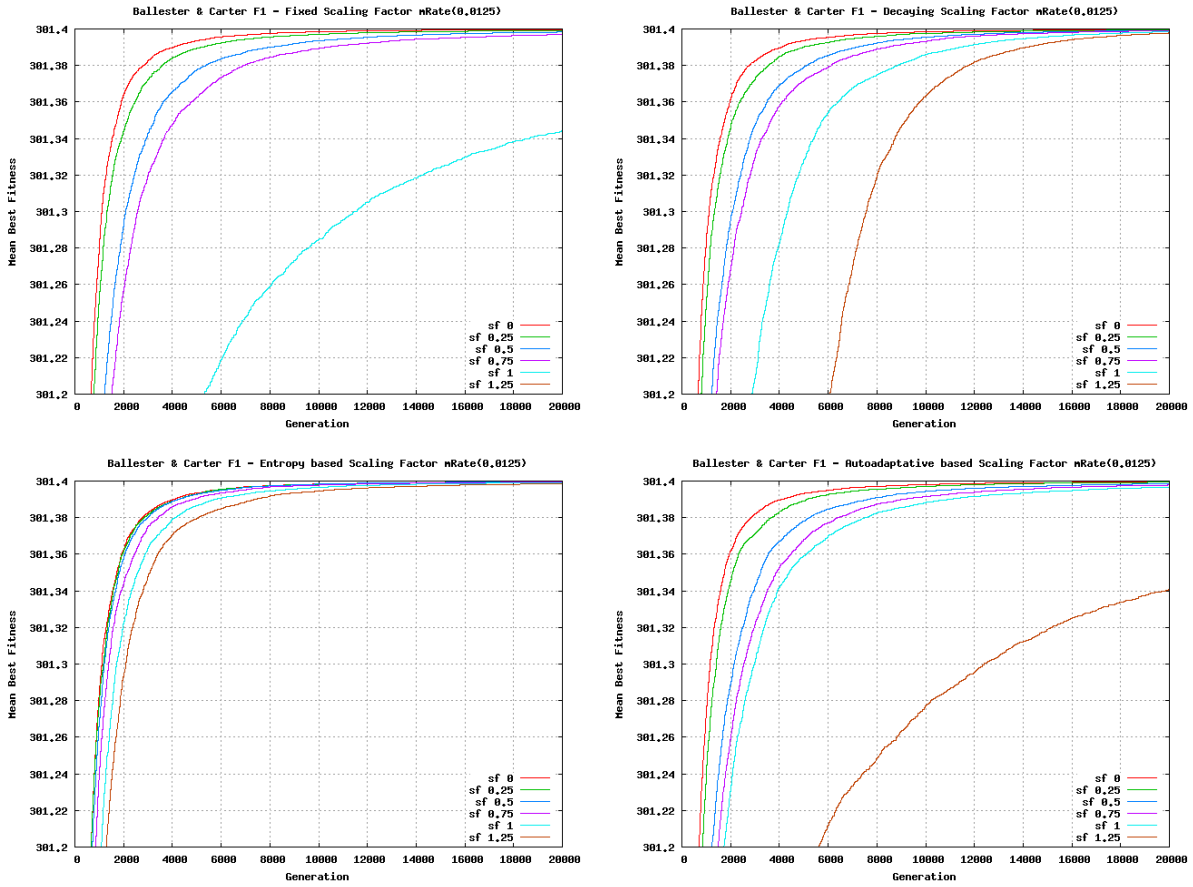


Figura 21: Media de los mejores valores de adaptación sobre 1000 ejecuciones de F_1 con tamaño de población $M = 20$ y probabilidad de mutación: $P_M = 0.0125$. Cada gráfica contiene las pruebas para $\phi_0 = \{0, 0.25, 0.5, 0.75, 1, 1.25\}$.

de las figuras 21 - 26. Dicho de otro modo, la tabla 17 incluye el valor para la última generación de cada gráfica de las figuras 21 a 26.

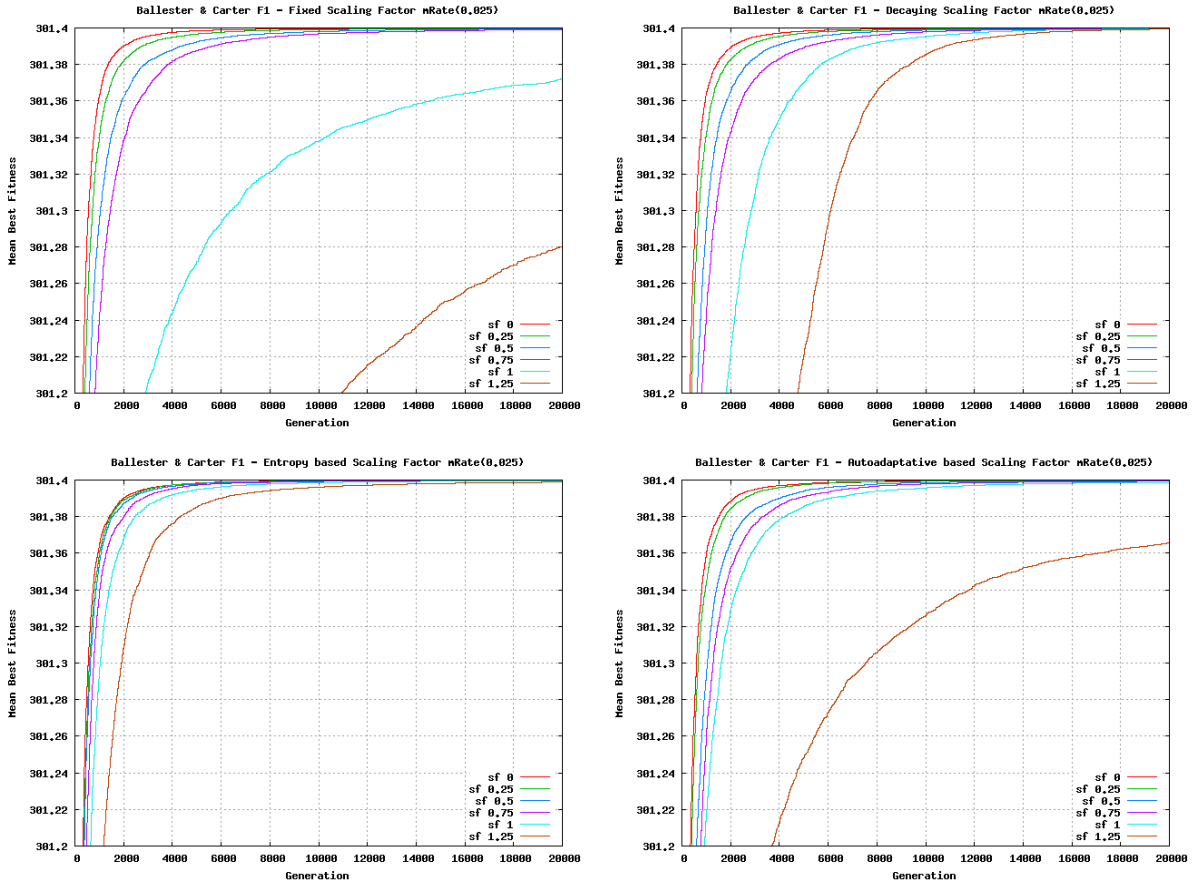


Figura 22: Media de los mejores valores de adaptación sobre 1000 ejecuciones de F_1 con tamaño de población $M = 20$ y probabilidad de mutación: $P_M = 0.025$. Cada gráfica contiene las pruebas para $\phi_0 = \{0, 0.25, 0.5, 0.75, 1, 1.25\}$.

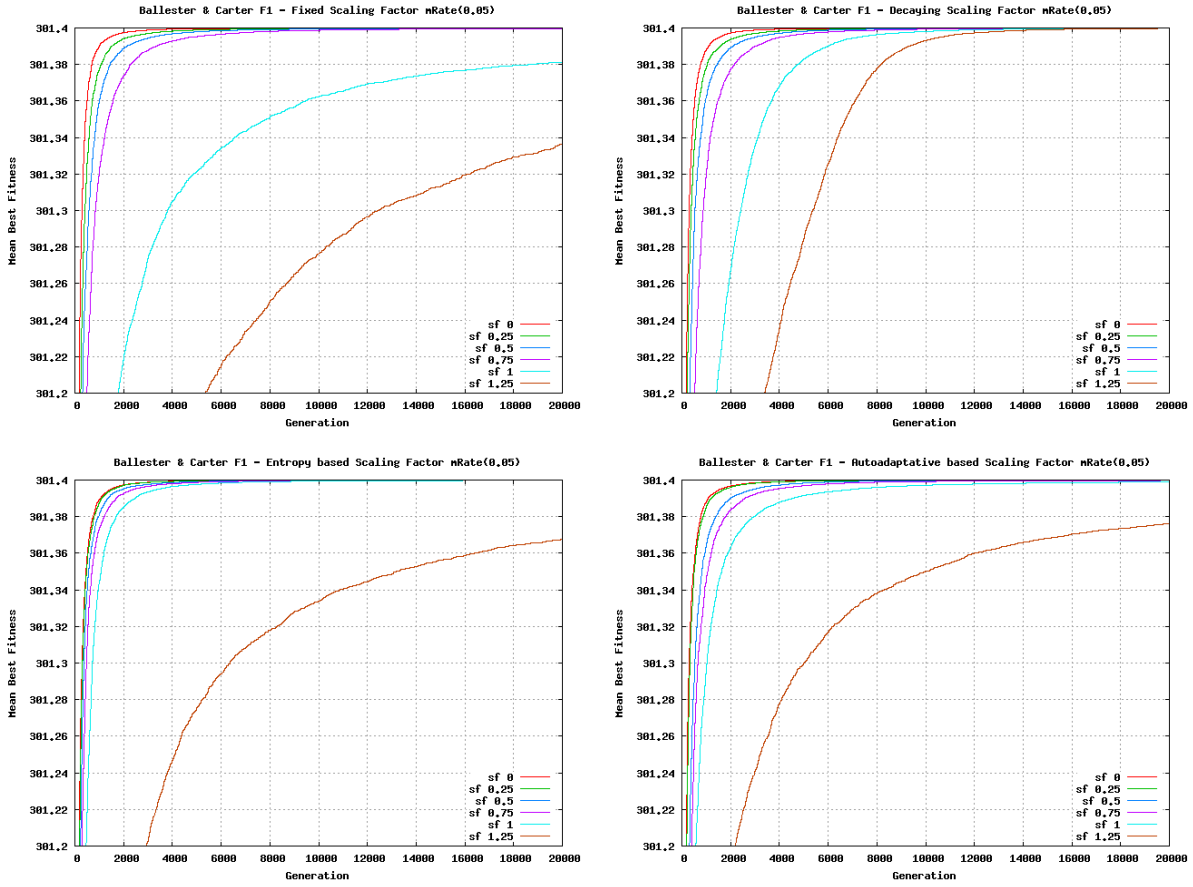


Figura 23: Media de los mejores valores de adaptación sobre 1000 ejecuciones de F_1 con tamaño de población $M = 20$ y probabilidad de mutación: $P_M = 0.05$. Cada gráfica contiene las pruebas para $\phi_0 = \{0, 0.25, 0.5, 0.75, 1, 1.25\}$.

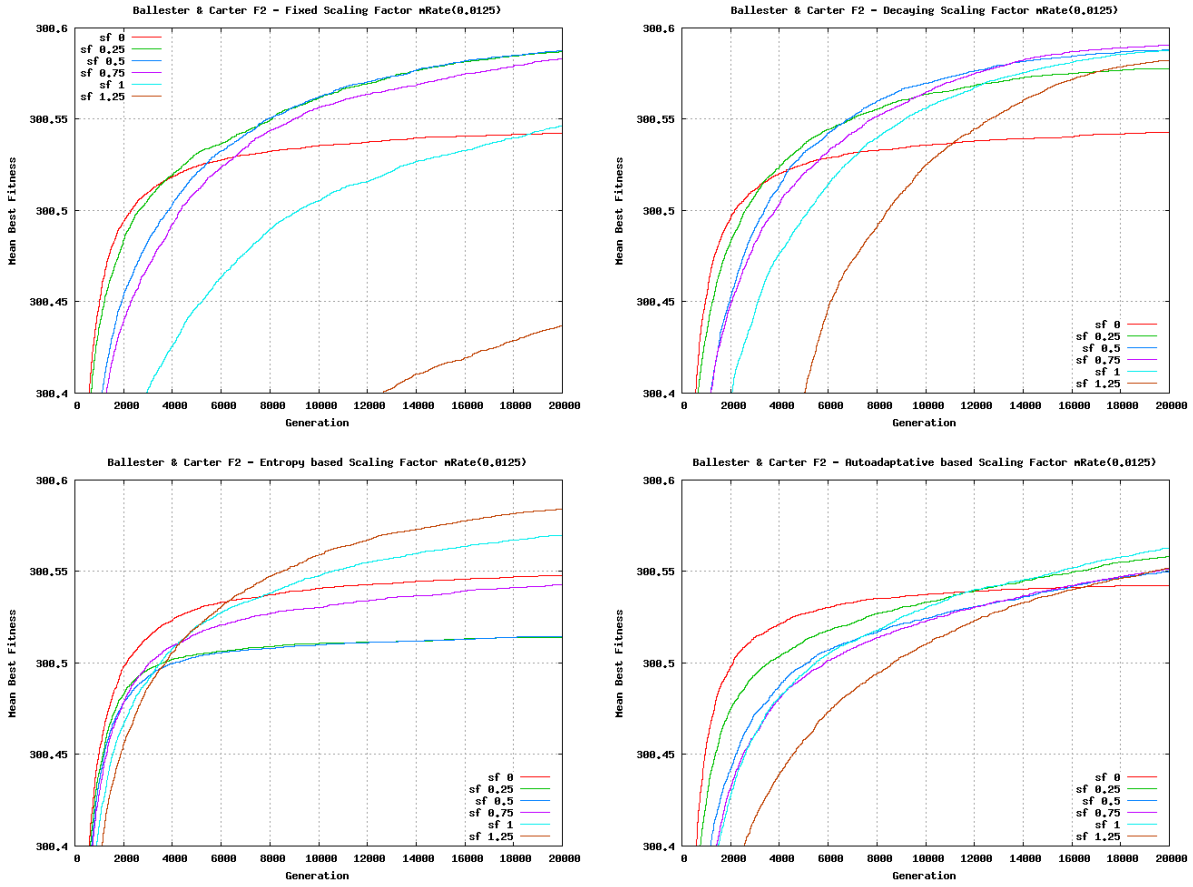


Figura 24: Media de los mejores valores de adaptación sobre 1000 ejecuciones de F_2 con tamaño de población $M = 20$ y probabilidad de mutación: $P_M = 0.0125$. Cada gráfica contiene las pruebas para $\phi_0 = \{0, 0.25, 0.5, 0.75, 1, 1.25\}$.

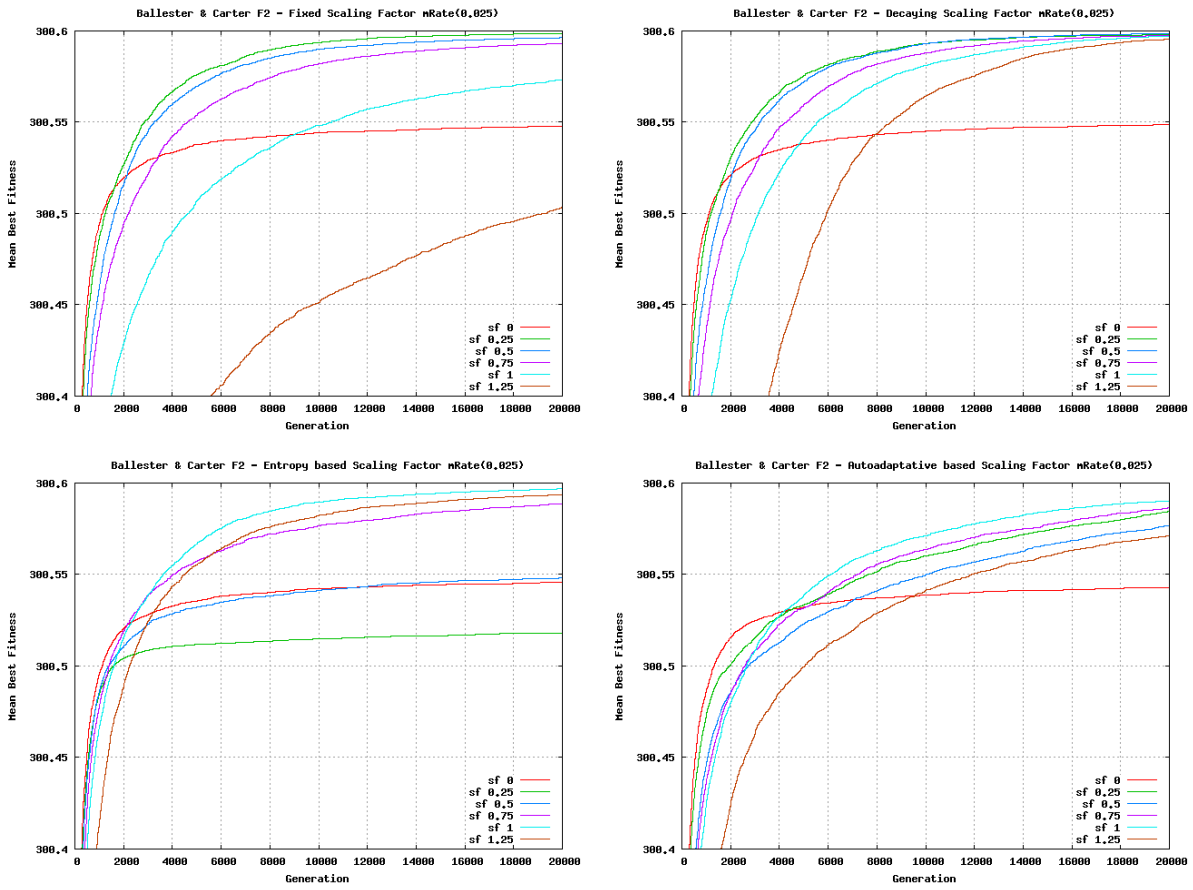


Figura 25: Media de los mejores valores de adaptación sobre 1000 ejecuciones de F_2 con tamaño de población $M = 20$ y probabilidad de mutación: $P_M = 0.025$. Cada gráfica contiene las pruebas para $\phi_0 = \{0, 0.25, 0.5, 0.75, 1, 1.25\}$.

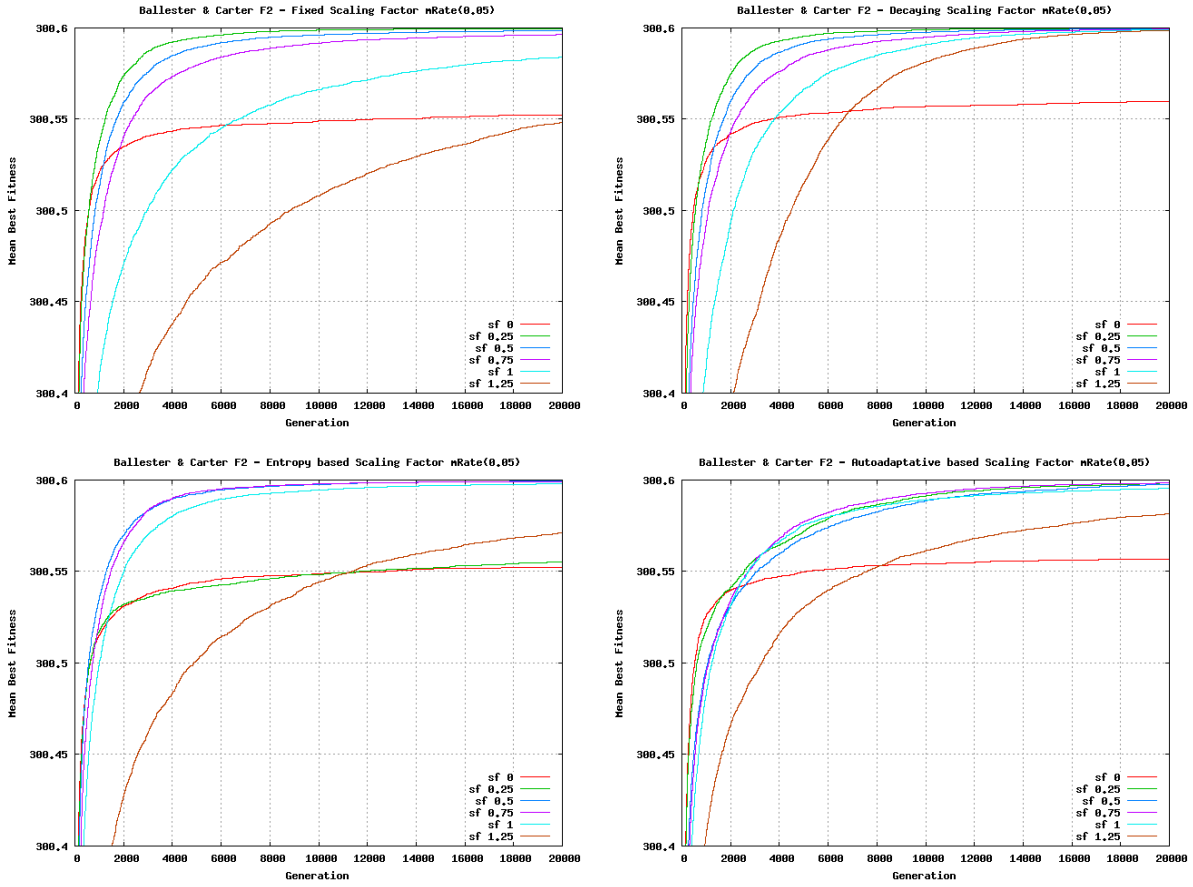


Figura 26: Media de los mejores valores de adaptación sobre 1000 ejecuciones de F_2 con tamaño de población $M = 20$ y probabilidad de mutación: $P_M = 0.05$. Cada gráfica contiene las pruebas para $\phi_0 = \{0, 0.25, 0.5, 0.75, 1, 1.25\}$.

Tabla 17: Resultados de los mejores valores medios de adaptación al final de las generaciones del AG para los experimentos de las figuras 21 a la 26.

			Media del mejor valor de adptación			
			Método de control del factor de escalado:			
Problema de optimización	P_M	ϕ_0	*Fijo*	*Determinista*	*Adaptativo*	*Autoadaptativo*
F_1	0.0125	0	301.39959	301.39956	301.39955	301.39953
		0.25	301.39917	301.39944	301.39958	301.39931
		0.5	301.39828	301.39921	301.39954	301.39843
		0.75	301.39701	301.39903	301.39943	301.39779
		1	301.34444	301.39849	301.39908	301.39661
		1.25	301.17820	301.39776	301.39851	301.34059
	0.025	0	301.39989	301.39988	301.39988	301.39987
		0.25	301.39975	301.39984	301.39989	301.39981
		0.5	301.39948	301.39978	301.39988	301.39961
		0.75	301.39914	301.39971	301.39980	301.39934
		1	301.37196	301.39957	301.39960	301.39875
		1.25	301.27990	301.39935	301.39886	301.36558
	0.05	0	301.39997	301.39997	301.39997	301.39996
		0.25	301.39992	301.39994	301.39997	301.39995
		0.5	301.39985	301.39992	301.39993	301.39989
		0.75	301.39963	301.39990	301.39990	301.39975
		1	301.38121	301.39983	301.39983	301.39892
		1.25	301.33623	301.39978	301.36746	301.37607
F_2	0.0125	0	300.54254	300.54271	300.547853	300.54250
		0.25	300.58691	300.57772	300.514009	300.55800
		0.5	300.58746	300.58794	300.514417	300.55022
		0.75	300.58300	300.59046	300.542590	300.55124
		1	300.54651	300.58768	300.569851	300.56277
		1.25	300.436974	300.58234	300.584207	300.55175
	0.025	0	300.54799	300.54869	300.54567	300.54290
		0.25	300.59861	300.59789	300.51795	300.58440
		0.5	300.59630	300.59849	300.54822	300.57685
		0.75	300.59311	300.59763	300.58885	300.58642
		1	300.57330	300.59707	300.59685	300.59020
		1.25	300.503285	300.59537	300.59350	300.57134
	0.05	0	300.55242	300.55993	300.55229	300.55693
		0.25	300.59961	300.59975	300.55537	300.59834
		0.5	300.59848	300.59960	300.59931	300.59762
		0.75	300.59631	300.59932	300.59924	300.59833
		1	300.58400	300.59914	300.59790	300.59540
		1.25	300.54833	300.59868	300.57133	300.58147

Capítulo 6: CONCLUSIONES Y TRABAJO FUTURO

El presente trabajo ha tratado la optimización de funciones de variable real a través de AGs, mejorando la técnica del Agrupamiento Generalizado, proponiendo tres mecanismos de control del denominado factor de escalado. Dicho factor determina la presión selectiva que el AG realiza sobre la población que gestiona. Por tanto, se han definido e implementado tres técnicas capaces de llevar a cabo un equilibrio entre el grado de explotación y el grado de exploración de tal forma que beneficie al proceso de búsqueda del AG, independientemente de si a la función le favorece un mayor o menor grado de exploración/explotación.

La primera conclusión procedente de los resultados del capítulo 5 de Evaluación es que realizando un control del factor de escalado se obtienen mejores resultados que utilizando un factor de escalado fijo.

El control determinista tiene la desventaja de sufrir una gran dependencia del valor de la constante de decrecimiento, ya sea un control determinista exponencial o lineal. Además, a la vista de los resultados, si dicha constante está bien escogida, los resultados ofrecidos pueden ser mejores que los ofrecidos por los otros dos tipos de control.

Otra importante conclusión procedente de los resultados descritos en el capítulo 5 de Evaluación es que el control de ϕ a través del proceso de búsqueda produce mejores resultados en el rendimiento del AG. Además, en los controles adaptativo y autoadaptativo de ϕ, estos resultados son obtenidos de una manera robusta evitando la introducción de parámetros adicionales a la configuración del AG en la determinación de cómo ϕ debe cambiar. Esto es una consecuencia directa de que el control de ϕ es dirigido por el AG automáticamente, liberando al usuario de

la parametrización dependiente del problema que se trata.

Es interesante comparar los resultados obtenidos por el control adaptativo basado en diversidad frente al control autoadaptativo del capítulo 5 de Evaluación. Para F1, el control adaptativo basado en la entropía de la población obtiene mejores resultados. Sin embargo, para F2 sucede lo contrario. Como consecuencia, en principio es difícil establecer qué método es mejor. Es destacable que el comportamiento del control autoadaptativo sea similar al del adaptativo, a pesar de que el primero de ellos no hace uso del conocimiento heurístico sobre la diversidad de la población que utiliza el control adaptativo. Este conocimiento heurístico se basa en que al principio del proceso de búsqueda se requiere más exploración, mientras que al final del mismo se requiere más explotación.

Se ha desarrollado un amplio estudio experimental que demuestra que las técnicas de control adaptativo y autoadaptativo propuestas obtienen resultados similares. Experimentos futuros podrían determinar bajo qué casos específicos el control adaptativo obtiene mejores resultados que el autoadaptativo y vice versa.

Durante el proceso experimental se ha utilizado selección de supervivientes generacional, aunque se podrían realizar diferentes experimentos con otras técnicas de selección de supervivientes como por ejemplo (μ,λ). Esta técnica de selección de supervivientes se basa en crear λ descencientes (con $\lambda > \mu$), calcular sus valores de adaptación y a posteriori escoger los μ mejores de manera determinista para que pasen a la siguiente generación.

Durante la fase inicial de los experimentos se obtuvieron muy buenos resultados aplicando cruce aritmético y mutación gaussiana sobre todo el cromosoma, por lo que además podría ser otra línea de investigación definir qué tipos de operadores proporcionan una mejor respuesta a la aplicación del Agrupamiento Generalizado y los tres controles propuestos.

Finalmente, destacar que este trabajo ha hecho énfasis en la optimización global de funciones reales utilizando Agrupamiento Generalizado, en lugar de analizar la formación de nichos, que es otra línea de investigación típica para el caso de algoritmos genéticos con fase de agrupamiento. Otra futura e interesante línea de investigación podría ser precisamente estudiar las características en cuanto a formación de nichos que tienen lugar cuando se controla automáticamente el

factor de escalado.

Referencias

[1] J. H. Holland, "Adaptation in Natural and Artificial Systems", Ann Arbor, MI: The University of Michigan Press, 1975, second edition, The MIT Press, 1992.

[2] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Reading, MA: Addison-Wesley, 1989.

[3] K. A. de Jong, "An analysis of the behavior of a class of genetic adaptive systems", Ph.D. dissertation, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, MI, 1975.

[4] S. W. Mahfoud, "Crowding and preselection revisited", in Proceedings of the 2nd International Conference on Parallel Problem Solving from Nature (PPSN II), R. Männer and B. Manderick, Eds. Amsterdam, The Netherlands: Elsevier, 1992, pp. 27–36.

[5] S. W. Mahfoud, "Niching methods for genetic algorithms", Ph.D. dissertation, Department of General Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 1995.

[6] O. J. Mengshoel and D. E. Goldberg, "Probabilistic crowding: Deterministic crowding with probabilistic replacement", in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999), W. Banzhaf, J. M. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. J. Jakiela, and R. E. Smith, Eds. San Francisco, CA: Morgan Kaufmann, 1999, pp. 409–416.

[7] O. J. Mengshoel, "Efficient Bayesian network inference: Genetic algorithms, stochastic local search, and abstraction", Ph.D. dissertation,Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, 1999.

[8] S. F. Galán and O. J. Mengshoel, "Generalized crowding for genetic algorithms", in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2010), M. Pelikan and J. Branke, Eds. New York: ACM Press, 2010, pp. 775–782.

[9] P. J. Ballester and J. N. Carter, "Real-parameter genetic algorithms for finding multiple optimal solutions in multi-modal optimization", in Proceedings of the Genetic and Evolutionary

Computation Conference (GECCO-2003), E. Cantú-Paz, J. A. Foster, K. Deb, L. D. Davis, R. Roy, U. O'Reilly, H. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. A. Dowsland,N. Jonoska, and J. Miller, Eds. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 706–717.

[10] O. J. Mengshoel, S. F. Galán, A. D. San Feliciano, "Adaptive Generalized Crowding for Genetic Algorithms", IEEE Computational Intelligence Society, Transactions on Evolutionary Computation, 2012

[11] IEEE STANDARD 830-1998 - "IEEE Recommended Practice for Software Requirements Specifications".

[12] D. Marr, "Vision: A Computational Investigation into the Human Representation and Processing of Visual Information", New York: Freeman, 1982.

[13] A.E. Eiben, J.E. Smith. "Introduction to Evolutionary Computation", Natural Computation Series, Springer.

[14] S. F. Galán, O. J. Mengshoel, and R. Pinter, "A novel mating approach a for genetic algorithms", Evolutionary Computation, 2012, to be published.

[15] Kruchten, Philippe (1995, November). "Architectural Blueprints — The "4+1" View Model of Software Architecture". IEEE Software 12 (6), pp. 42-50.

[16] G. Syswerda, "Uniform crossover in genetic algorithms", in Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA-89), J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 2–9.

[17] O. J. Mengshoel and D. E. Goldberg, "The crowding approach to niching in genetic algorithms", Evolutionary Computation, vol. 16(3), pp. 315–354, 2008.

[18] S. W. Mahfoud and D. E. Goldberg, "Parallel recombinative simulated annealing: A genetic algorithm", Parallel Computing, vol. 21, pp. 1–28, 1995.

[19] G. R. Harik, "Finding multimodal solutions using restricted tournament selection", in Proceedings of the 6th International Conference on Genetic Algorithms (ICGA-95), L. J. Eshelman, Ed. San Francisco, CA: Morgan Kaufmann, 1995, pp. 24–31.

[20] A. Farhang-Mehr and S. Azarm, "Entropy-based multi-objective genetic algorithm for design optimization, Structural and Multidisciplinary Optimization", vol. 24(5), pp. 351–361, 2002.

[21] Y. Tsujimura and M. Gen, "Entropy-based genetic algorithm for solving TSP", in Proceedings of the 2nd International Conference on Knowledge-Based Intelligent Electronic Systems (KES'98), L. C. Jain and R. K. Jain, Eds. Piscataway, NJ: IEEE Press, 1998, pp. 285–290.

[22] S. H. Liu, M. Mernik, and B. R. Bryant, "To explore or to exploit: An entropy-driven approach for evolutionary algorithms", International Journal of Knowledge-Based and Intelligent Engineering Systems, vol. 13, pp. 185–206, 2009.

[23] I. Rechenberg, "Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution", Ph.D. dissertation, Department of Process Engineering, Technical University of Berlin, Berlin, 1971, reprinted by Fromman-Hozlboog Verlag, 1973.

[24] H. P. Schwefel, "Evolutionasstrategie und numerische optimierung", Ph.D. dissertation, Department of Process Engineering, Technical University of Berlin, Berlin, 1975.