TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

TÉCNICAS AVANZADAS DE HIBRIDACIÓN PARA METAHEURÍSTICAS

Autora
Andrea Morales Garzón

Director
Daniel Molina Cabrera

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Granada, Junio de 2018

Advanced Hybridization Techniques of Metaheuristics

Andrea Morales Garzón

Palabras clave: hibridación, evolución, poblaciones, islas, heterogéneo, migración, genético

Resumen

En este proyecto estudiaremos el comportamiento de modelos híbridos formados por diferentes combinaciones de algoritmos evolutivos, concretamente, algoritmos genéticos básicos. Para dicho estudio utilizaremos distintos problemas de codificación real obtenidos de CEC'14 (*Test Function Suite Optimization*). Se analizará el comportamiento de dichos algoritmos para funciones de distintas características: funciones unimodales, multimodales, funciones híbridas y otro tipo de funciones que son resultado de una composición de funciones previas.

El objetivo principal en este proyecto, se puede en como un análisis comparativo de diferentes tipos de modelos de hibridación. En primer lugar, implementaremos los distintos algoritmos genéticos, variando los operadores y parámetros de los operadores en dos versiones diferentes; por una parte tendremos algoritmos en la versión generacional y por otra parte en la estacionaria. En el caso de la versión generacional, utilizaremos algoritmos que se diferencian en los parametros para el operador de cruce, generando así técnicas orientadas tanto a la exploración como a la explotación, porque la combinación de estas características es interesante para una hibridación Por otro lado, para la versión estacionaria, implementaremos diferentes operadores de selección y reemplazo que nos permitan explorar distintas zonas del espacio de búsqueda.

Por otra parte, generaremos un Modelo de Islas clásico que incorporará diferentes combinaciones de los algoritmos anteriores, contemplando tres versiones del mismo: una versión generacional, una versión estacionaria y finalmente una versión que combina los dos mejores algoritmos de cada versión anterior. Una vez que tengamos todos estos modelos, los ejecutaremos para diferentes problemas y parámetros, obteniendo los resultados para las diferentes ejecuciones y así hacer un análisis detallado de los mismos. Por último, vamos a implementar otros modelos de hibridación más innovadores, que se caracterizan por ser modelos que involucran una componente de adaptación durante la ejecución. Con estos modelos procederemos de igual forma, y tras hacer el análisis individual de estos modelos,

compararemos los resultados de cada implementación híbrida con el objetivo de comprender el comportamiento tanto a nivel global como al nivel de los algoritmos que participan. Para analizar las diferentes opciones que tenemos, vamos a apoyarnos en gráficas de convergencia para sustentar nuestras conclusiones, y gráficas que representan el funcionamiento interno de los modelos híbridos, con el objetivo de entender mejor cómo implementan la adaptación para mejorar los resultados. El objetivo aquí son principalmente dos aspectos. En primer lugar, conocer las ventajas y desventajas de cada modelo y, en segundo lugar, decidir cuál de los modelos, creemos que es el mejor, explicando por qué y cuándo usaríamos cada modelo.

En esencia, con este proyecto, veremos que los Modelos de Islas son una opción muy poderosa, pero una de sus deficiencias es la necesidad de adaptación en ejecución que se condicione a los algoritmos que estamos utilizando. Podremos observar que, la utilización de modelos que involucren dicha adaptación, tendrá sus ventajas y desventajas, ya que, al fin y al cabo, todo depende del uso que hagamos de este tipo de metodologías y del problema específico y algoritmos que implementemos

Advanced Techniques of Metaheuristics Hibridation

Andrea Morales Garzón

Keywords: hybridization, evolution, populations, islands, heterogeneous, migration, genetic

Abstract

In this project we will study the behavior of hybrid models which are composed of different combinations of evolutionary algorithms, mainly, basic genetic algorithms. For this study we will use different real-coding problems obtained from CEC'14 (Test Function Suite Optimization). We will analyze the behavior of these algorithms for some of the functions of different characteristics: unimodal and multimodal functions, hybrid function, and other type of functions as result of a composition of previous functions working together.

The main objective can be summarized as a comparative analysis of different kind of hybridization evolutive models. Firstly, we are going to implement different kinds of Genetic Algorithms, changing the operators and parameters of the functions in two different versions: generational and steady-state genetic versions. For the generational implementation, we are going to use different algorithms depending on the crossover operator in order to generate explorative and explotative techniques that can be interesting for our study. By the other way, in the case of the steady-state version, we'll implement different Selection and Replacement operators to explore other options.

In a second place, we are going to generate a classic Island Model that incorporates different combinations of the algorithms that have been evaluated individually. We are going to study three versions; a generational version, a steady-state version and finally a version that blends the two best algorithms of each prior version. Once we have all this models implemented, we are going to execute it for different problems and parameters, and we will obtein the results of the differents executions, with the goal of doing a detailed analysis of the results.

For last, we are going to implement other Hybrid Models, which are characterized for being innovative and adaptative models. Then, we will repeat the previous procedure, and we'll obtaing the model's results for the three versions that we have mentioned earlier. After doing an individual analysis of these hybrid models, we'll compare the results of every hybrid implementation with the aim of understanding the behavior of genetic algorithms in hybrid models. In order to analyze the different options we have, we are going to create convergence graphs to support our conclussions, and also we are going to use graphs which represents the internal working of the hybrid models, with the aim of knowing better how they implement adaptation to improve the results.

The objetive here are mainly two things. In a first place, knowing about the advantages and disadvantages of each model, and in a second place, deciding which of the models, we think is the best, explaining why and when we'd use each model.

In essence, with this project, we can see that Island Models are a very powerful option, but one of their shortcomings is the need to do a model adaptation during execution, depending on the algorithms that we are using. We will see that, using models with these adaptation that we are mentioned before, will have its advantages and disadvantages, since, after all, everything depends on the use we make of this type of methodologies and the specific problem and algorithms that we have implemented.

Yo, **Andrea Morales Garzón**, alumna de la titulación GRADO EN INGENIERÍA INFORMÁTICA de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77147632C, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Andrea Morales Garzón

Granada a 10 de Junio de 2018 .

Otros

D. **Daniel Molina Cabrera**, Profesor del Departamento Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Técnicas Avanzadas de Hibridación de Metaheurísticas***, ha sido realizado bajo su supervisión por **Andrea Morales Garzón**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 18 de Mayo de 2018 .

El director:

Daniel Molina Cabrera

Agradecimientos

A todo mi circulo cercano durante el desarrollo de este proyecto, por sus consejos y apoyo continuo, fundamentales para intentar aportar lo mejor de mí misma en este proyecto.

Índice

Índice de tablas

Introducción y Motivación

RESUMEN: En este capítulo se va a introducir, desde una forma más global y general, el tema que vamos a tratar en este proyecto. Buscamos, con esta sección, crear un foco de interés en los distintos aspectos que vamos a estudiar, los cuáles están directamente relacionados con el estudio y comparación de distintas técnicas de hibridación, su funcionamiento, y su idoneidad a la hora de llevarlas a la práctica unas frente a otras.

1. Introducción y Motivación

A lo largo de los años, se ha puesto un foco de interés en el diseño de modelos híbridos para resolver problemas de codificación real. Hay una gran cantidad de Metaheurísticas, cada una, con sus propiedades y características, da unos resultados para los distintos problemas que evaluemos. Incluso realizando modificaciones en los parámetros utilizados, la variabilidad que se puede producir en las soluciones es enorme. Por ello, puede ser una buena idea realizar combinaciones de los algoritmos con vistas a potenciar las características que muestran cada uno por separado. Tal y como se ha observado en diversos proyectos [29] [34] [31], se ha visto que utilizar modelos de hibridación supone una mejora cuantitativa en las soluciones respecto a los resultados obtenidos de la ejecución, de manera individual, de los algoritmos que forman parte de dicho modelo. En otras palabras, el resultado de combinar distintos algoritmos permite superar algunas de las dificultades que se puedan encontrar los algoritmos de manera individual, favoreciendo el desarrollo de nuevas soluciones.

Desde sus inicios, las técnicas de hibridación han avanzado y han ido surgiendo nuevos modelos, generando alternativas a los modelos más clásicos (como pueden ser los Modelos de Islas). Sin embargo, hasta el momento no se ha realizado una comparativa de estas técnicas más innovadoras en comparación a los modelos tradicionales. Por ello, en este proyecto se realizará una comparativa de dichas técnicas, para hacer un análisis de cómo funcionan unas respecto a otras y ver cuestiones como su potencial a la hora de alcanzar resultados competitivos, el funcionamiento de los algoritmos internamente en el modelo y el comportamiento en distintos problemas que contemplaremos. En nuestro caso, vamos a estudiar los modelos híbridos que combinan algoritmos evolutivos al mismo nivel. Con ello, estamos trabajando con algoritmos que se ejecutan cada uno por separado, y no de manera complementaria unos a otros con objetivos concretos como puede ser la incorporación de mayor diversidad o mayor capacidad de explotación a un modelo. Nuestro objetivo, para el final de este proyecto, es por tanto hacernos una idea más concreta de los aspectos competitivos de cada modelo y compararlos en vista a dos aspectos. El primero de ellos

sería estudiar qué modelo híbrido parece ir mejor, y ver así si los estudios realizados en este ámbito han ido en los últimos años por buen camino. Sin embargo, un modelo no tiene por qué ser el óptimo siempre, por lo que también veremos en qué tipo de casos utilizaríamos cada una de las técnicas estudiadas.

Para ello, vamos a centrar el estudio en un tipo concreto de algoritmos dentro de los Algoritmos Evolutivos: los **Algoritmos Genéticos**. Realizaremos distintas versiones, tanto de forma individual como de forma híbrida para ver ambos comportamientos en las distintas pruebas que se realizarán. Realizaremos pruebas individuales de los algoritmos genéticos (en las versiones generacionales y estacionarios) modificando parámetros del operador de cruce, y combinando distintos operadores de selección y reemplazo.

Para las pruebas de modelos híbridos, contemplaremos tres modelos. En primer lugar, implementaremos un **modelo clásico de islas** con modelo de vecindario aleatorio. Este tipo de técnicas, se caracteriza por combinar diferentes subpoblaciones, cada una en base a un algoritmo, con el objetivo de retroalimentar las subpoblaciones entre sí, para escapar de zonas no prometedoras y mejorar.

Por otra parte, compararemos el modelo anterior con dos modelos que involucran técnicas de adaptación en tiempos de ejecución, aspecto que puede dar muy buenos resultados a la hora de combinar algoritmos. Tendremos por tanto **un segundo modelo que implementa dos fases de ejecución**, una previa para todos los algoritmos y una posterior para el más prometedor. Este algoritmo es digno de estudiar porque permite detectar en tiempo de ejecución el algoritmo que mejor resultado está dando para ese caso, y así personalizar el problema para intentar dar buenas soluciones [18]. Por último, implementaremos **un tercer modelo híbrido que involucre probabilidades adaptativas** para determinar dinámicamente el camino que debe tomar el modelo a la hora de ejecutar los algoritmos involucrados en él [22], repartiendo la participación de los algoritmos en el modelo en función de los resultados que estén dando cada uno de ellos.

Todos estos modelos, serán probados para distintas combinaciones de algoritmos genéticos, dando lugar a tres focos de estudio para cada uno de ellos. Estos tres focos serán, en primer lugar una versión para algoritmos genéticos generacionales, una segunda versión que combine los algoritmos genéticos estacionarios implementados, y por último, una combinación de los dos algoritmos genéticos estacionarios y los dos generacionales con los que se hayan obtenido mejores resultados.

Desde el punto de vista de las realización de las pruebas, lanzaremos los algoritmos en sus distintas versiones para las funciones proporcionadas en **CEC'14 Benchmark Suite** [17] para dos dimensiones distintas del problema (Dimensión 10 y Dimensión 30).

Planificación

RESUMEN: En este apartado, se expone la planificación creada para llevar a cabo el proyecto, con los plazos esperados y coordinados para poder compaginar la planificación, tanto de la parte teórica como de la parte relativa a la documentación.

2. Planificación

Hemos dividido las distintas tareas en bloques para facilitar mejor la organización. Aunque se ha intentado seguir los ritmos de planificación previstos, al final no se han podido lograr de forma estricta, por las distintas complicaciones y los añadidos al proyecto que se han realizado sobre la marcha. Por tanto, exponemos tanto la planificación inicial, como la planificación actualizada que se ha llevado finalmente.

2.1. Planificación Inicial

Implementación de los Algoritmos

En primer lugar, vamos a ver la organización acerca de las cuestiones de implementación del proyecto. Se implementarán los algoritmos individuales, y posteriormente los modelos que involucren dichos algoritmos. Para cada una de las implementaciones realizadas, primero habrá una fase de comprensión del algoritmo previa a la programación del mismo. Una vez finalizada dicha implementación, se repasará cada uno antes de pasar a la siguiente tarea. A continuación podemos ver una tabla con la organización de dichas actividades.

Tabla 1: Planificación: Implementación de los Algoritmos

Tarea	**Duración**	**Fecha Tope**
Algoritmo Genético (Comprensión)	1 día	31/11/17
Algoritmo Genético (Estudio operadores)	1 día	31/11/17
Algoritmo Genético Estacionario	1 día	31/11/17
Repaso Algoritmos Genéticos	1 día	01/05/18
Modelo Híbrido 1 (Comprensión)	1 día	01/04/18
Modelo Híbrido 1 (Implementación)	2 días	01/05/18
Modelo Híbrido 2 (Comprensión)	1 día	01/04/18
Modelo Híbrido 2 (Implementación)	2 días	01/05/18
Modelo Híbrido 3 (Comprensión)	1 día	01/04/18
Modelo Híbrido 3 (Implementación)	2 días	01/05/18
Repaso Modelos Híbridos	1 día	01/06/18

Planificación: Ejecución de los Algoritmos

En este bloque podemos ver la organización llevada a cabo para las ejecuciones y recogidas de datos de los modelos y algoritmos ejecutados. Debido a su larga duración, se podrá simultanear con los objetivos a cumplir en la tabla 3

Tabla 2: Planificación: Obtención de Datos

Tarea	**Duración**	**Fecha Tope**
Implementación para fichero CSV	3 días	12/05/18
Tomar datos y CSV	5 días	17/05/18
Crear gráficas de los datos tomados	1 día	31/05/18

Planificación: Redacción y desarrollo de la Memoria

Una vez finalizada la fase de implementación, procedemos a realizar ejecuciones y recoger los datos del algoritmo. Para ello, llevaremos a cabo muchas de las tareas de forma simultánea a las especificadas en la tabla 2 ya que, debido al tiempo de ejecución que necesita cada algoritmo, las secciones de redacción del diseño e implementación del proyecto se pueden llevar a cabo a la vez que ejecutamos.

Como se puede observar en la tabla 3, en primer lugar se llevará a cabo la redacción del grueso del proyecto, dejando para el final las conclusiones obtenidas a partir de los datos.

Tabla 3: Redacción de la Memoria

Tarea	**Duración**	**Fecha Tope**
Memoria: Introducción	1 día	01/05/18
Repaso Introducción	1 día	01/06/18
Memoria: Planificación	1 días	01/06/18
Repaso Planificación	1 día	01/06/18
Memoria: Diseño	1 días	01/05/18
Repaso Diseño	1 día	01/06/18
Memoria: Implementación	2 días	01/05/18
Repaso Implementación	1 día	01/06/18
Memoria: Pruebas	1 días	01/06/18
Repaso Pruebas	1 día	04/06/18
Memoria: Conclusiones	1 días	05/06/18
Memoria: Repaso Conclusiones	1 día	05/06/18
REPASO FINAL	2 días	07/06/18

2.2. Planificación Final llevada a cabo

Implementación de los Algoritmos

Tabla 4: Planificación: Implementación de los Algoritmos

Tarea	**Duración**	**Fecha Tope**
Algoritmo Genético (Comprensión)	1 día	31/11/17
Algoritmo Genético (Estudio operadores)	1 día	31/11/17
Algoritmo Genético Estacionario	1 día	31/11/17
Repaso Algoritmos Genéticos	1 día	01/05/18
Modelo Híbrido 1 (Comprensión)	1 día	01/04/18
Modelo Híbrido 1 (Implementación)	2 días	01/05/18
Modelo Híbrido 2 (Comprensión)	1 día	01/04/18
Modelo Híbrido 2 (Implementación)	2 días	01/05/18
Modelo Híbrido 3 (Comprensión)	1 día	01/04/18
Modelo Híbrido 3 (Implementación)	2 días	01/05/18
Repaso Modelos Híbridos	1 día	01/06/18
MOS Comprender Algoritmo	2 días	20/05/18
MOS Realizar Pruebas	15 días	12/06/18

Planificación: Ejecución de los Algoritmos

Tabla 5: Planificación: Obtención de Datos

Tarea	**Duración**	**Fecha Tope**
Implementación para fichero CSV	3 días	12/05/18
Tomar datos y CSV	7 días	25/05/18
Crear gráficas de los datos tomados	2 días	31/05/18

Planificación: Redacción y desarrollo de la Memoria

Tabla 6: Redacción de la Memoria

Tarea	**Duración**	**Fecha Tope**
Memoria: Introducción	2 días	01/05/18
Repaso Introducción	1 día	01/06/18
Memoria: Planificación	2 días	01/06/18
Repaso Planificación	1 día	01/06/18
Memoria: Diseño	2 días	01/05/18
Repaso Diseño	1 día	01/06/18
Memoria: Implementación	2 días	01/05/18
Repaso Implementación	1 día	01/06/18
Memoria: Pruebas	2 días	01/06/18
Repaso Pruebas	1 día	04/06/18
Memoria: Conclusiones	5 días	05/06/18
Implementación Script Gráficas	2 días	07/06/18
Creación Gráficas	2 días	08/06/18
Memoria: Incorporación Gráficas	1 día	09/06/18
Memoria: Ideas y conclusiones finales	1 día	10/06/18
Memoria: Repaso Conclusiones	1 día	11/06/18
REPASO FINAL	2 días	16/06/18

Contexto Bibliográfico

RESUMEN: En esta sección se lleva a cabo una revisión bibliográfica sobre el tema en el que hemos centrado el proyecto. Buscamos con ello, llevar a cabo una pequeña vista hacia atrás para poder entender los distintos ámbitos que más se han tratado y centrado los estudios en cuanto a la hibridación de técnicas evolutivas, las cuáles han evolucionado y derivado en nuevos modelos, no tan estudiados, los cuáles estudiamos en este proyecto.

3. Contexto Bibliográfico

La optimización en problemas de codificación real ha sido objeto de estudio durante muchos años, siendo foco de investigación junto con este tipo de algoritmos por diversos autores. Se ha probado el uso de Algoritmos Evolutivos para los problemas de codificación real, como es el caso de los algoritmos utilizados en [4] y en [26]. En nuestro caso, nos vamos a centrar únicamente en un tipo concreto de algoritmos evolutivos: los **Algoritmos Genéticos**. Los algoritmos genéticos (GA) son algoritmos basados en poblaciones que se pueden describir como la combinación de dos procesos, la generación de elementos del espacio de búsqueda (con la aplicación de recombinación o mutación a la población anterior) y la actualización (con la selección y el redimensionamiento) para producir nuevas poblaciones basadas en el conjunto de puntos que se crean junto con los de la anterior población [30][33]. Además, vamos a estudiarlos también realizando una implementación de su versión `estacionaria` (*Steady-State*), de forma que apliquemos distintas estrategias de control en cuanto a la reproducción [7].

Por tanto, es de esperar que la estrategia de obtención de nuevas soluciones del espacio de búsqueda sea determinante a la hora de optimizar los resultados, ya que cada operador de cruce va a dirigir la búsqueda a una zona u otra del espacio en función de su procedimiento. Esto significa que cada operador de cruce se comportará de manera distinta al resto, y también de manera diferente en cada problema en el que lo apliquemos [13]. En nuestro caso, hemos escogido utilizar el Operador de Cruce *BLX-*α [27][12], aplicando distintos valores para α. Este operador combina una parte explotativa junto con una explorativa (factor que depende del valor que tome α), permitiendo así controlar el nivel de exploración que se aplica en el problema en cuestión [11].

Hay otros factores, también esenciales, que influyen directamente en la convergencia del algoritmo. Por un lado tenemos el tamaño de la población, ya que si este es demasiado

pequeño, la población convergerá rápidamente debido a la poca diversidad [3], pero si es demasiado grande podemos estar sobrecargando el esfuerzo que se realiza con el algoritmo de forma innecesaria. Por otra parte, también repercute el **proceso de selección** que se elija [5].

Sin embargo, a pesar de la efectividad de este tipo de algoritmos, que han mostrado su robustez y su eficacia en el campo que estudiamos, se ha llegado a la conclusión de que para problemas concretos, hay unos algoritmos que son más efectivos que otros. Por esta razón, la combinación de distintas estrategias parece una opción a considerar a la hora de explorar los puntos fuertes de cada una de ellas y obtener mejores resultados globales debido a la cooperación de unas técnicas con otras [29]. El objetivo principal a la hora de utilizar este tipo de técnicas ha sido mejorar el rendimiento en la búsqueda en términos de la calidad de la solución encontrada o de la reducción de esfuerzos en cuanto a la cantidad de evaluaciones ya que a medida que utilizamos un mayor número de islas, se reduce el esfuerzo necesario para poder encontrar una solución [2]. Este concepto de **cooperación** ha sido estudiado en una gran cantidad de trabajos, donde hemos podido ver que la hibridación de distintos algoritmos evolutivos ha dado buenos resultados respecto a los que se conseguían de forma individual con dichos algoritmos [15] [18]. Existen numerosos artículos donde se propone la cooperación entre distintas poblaciones para mejorar los resultados obtenidos con algoritmos evolutivos, como podemos ver en [24] o [23] para Algoritmos Genéticos, en [28] haciendo uso de *Differential Evolution*, o en [6] y [16] donde se lleva a cabo *MultiSwarm*.

Dependiendo de las características reproductivas de las estrategias que combinemos, estaremos tratando con un modelo *homogéneo* o un modelo *heterogéneo*. Si todas las estrategias que intervienen tienen la misma capacidad reproductiva, estaremos hablando de un modelo homogéneo, como es el caso de [10] y al contrario. Si la estrategia de reproducción es diferente entre los algoritmos utilizados, la isla será de carácter heterogéneo como ocurre con [1] [32] [18]. En nuestro caso contemplaremos las dos variantes, realizando diferentes versiones para cada uno de los modelos híbridos que realizaremos.

A su vez, dependiendo de la forma en la que combinemos los algoritmos, se contemplan dos corrientes principales en los problemas que incluyen hibridación de algoritmos evolutivos:

1. Combinación de estrategias donde existe poca comunicación, y cada una de las técnicas cumple roles muy específicos, tal y como podemos ver en [19], [4], [15] y en [25]. En estos casos, normalmente hay un componente que se caracteriza por ser potencialmente*explotativo* y otro *explorativo*.

2. Combinación de estrategias al mismo nivel, donde todos los algoritmos tienen igual papel e importancia en la ejecución [20] [10] [18].

Durante este proyecto nos centraremos en el segundo tipo mencionado, utilizando diferentes variantes de Algoritmos Genéticos como componentes del modelo de hibridación. Con ello, se mantiene un grado de independencia y se exploran zonas del espacio de búsqueda, cooperando unos algoritmos con otros mediante la trasmisión de información mediante la **migración** de soluciones entre las poblaciones de dichos algoritmos [20]. En nuestro caso, estudiaremos, por una parte, el modelo clásico de hibridación, el cuál compararemos con dos técnicas más innovadoras: el *Modelo de Ejecución en 2 fases*, y el *Modelo con Probabilidades Adaptativas*.

Los Modelos de Islas, como técnica clásica que es, ha sido estudiada en profundidad desde una gran cantidad de enfoques. En la literatura, podemos encontrar numerosas variantes en cuanto a cómo se debe realizar la comunicación entre subpoblaciones en los Modelos de Islas, y es que la clave en la mejora de soluciones con estos modelos es el trabajo **cooperativo** [18] que realizan los algoritmos para poder llegar a soluciones que de forma individual no conseguirían. Principalmente, hay una gran diferenciación en cuanto a la sincronización existente en esta comunicación. Por una parte encontramos las comunicaciones *síncronas*, con las cuáles trabajaremos a lo largo del proyecto. Uno de los ejemplos más típicos es aquel donde la comunicación entre las islas funciona como un anillo unidireccional [10] [35]. En cuanto al criterio que se utiliza para que un individuo de una población concreta realice una migración, encontramos trabajos donde son los peores individuos de cada subpoblación los que migran en busca de un entorno más adecuado [14], aunque lo más común es que la migración provoque que los mejores individuos de unas islas viajen a otras. También encontramos proyectos en los que el individuo que migra es elegido aleatoriamente, aunque solo se sustituya en otra población si es mejor que el peor de a la que migra [2]. Encontramos también otros trabajos donde la migración se realiza en base al *Concepto de Especie*: los individuos atípicos de cada subpoblación se desplazan a otras Islas [9] [8]. También hay proyectos que utilizan funciones concretas para pasar soluciones de unas zonas prometedoras del espacio de búsqueda a otras [34].

Por otra parte, hay trabajos que defienden la conveniencia de usar la comunicación *asíncrona*, debido a sus mejoras en cuestiones de rendimiento, ya que se ahorra tiempo durante la ejecución del modelo [10].

Hasta ahora, hemos hablado de modelos más tradicionales pero han ido surgiendo variantes en las cuales se produce una adaptación interna durante su propia ejecución para optimizar sus resultados, similar a la idea vista en [13]. Como dice [30] parece que, en general, los algoritmos que se basan en la observación del rendimiento de diferentes estrategias parecen ser los que mejores resultados obtienen en el campo, por lo que estudiaremos en este proyecto dos Modelos Híbridos los cuáles llevan a cabo una `autoadaptación` (*Self-Adaptation*) durante su ejecución, entendiendo por `autoadaptación` la "*capacidad de utilizar los mecanismos internos del algoritmo para determinar la trayectoria del mismo*", a diferencia de los algoritmos `adaptativos` los cuáles utilizan herramientas externas al algoritmo, como pueden ser estudios acerca de la distribución del *Fitness* en la población. La primera de las variantes se trata de un Modelo Híbrido con un nivel de acoplamiento muy bajo, donde se lleva a cabo una ejecución por fases dentro del mismo modelo. Tendríamos una primera fase de competición de los algoritmos, con el objetivo de determinar cuál de ellos es el que mejor se adapta a la población de la que partimos, y una segunda fase donde únicamente se ejecutaría este mejor algoritmo. Este modelo está contemplado en [18]. La segunda de las variantes está basada en la propuesta de adaptación de los algoritmos de Búsqueda Local expuesto en [22]. De esta forma, todos los algoritmos tienen de partida las mismas oportunidades de ejecutarse, pero a medida que se va avanzando en la ejecución, aquellos que obtienen mejores resultados se ven compensados de cara a la posibilidad de ejecutarse mayor número de veces en las siguientes iteraciones del modelo, dándoles así una oportunidad mayor para desarrollarse, respecto a la que recibirían en un modelo más clásico.

Con ambas técnicas, se obtuvieron resultados muy competitivos en comparación a sus componentes analizados de manera individual, llegando a la conclusión de que los modelos híbridos que involucran autoadaptación forman una técnica competitiva en el campo de

la resolución de problemas de codificación real. En las siguientes secciones, estudiaremos en mayor profundidad estos modelos evaluando cuestiones como su eficacia y su idoneidad en la práctica.

Diseño de la parte experimental

RESUMEN: En esta sección, se explica de forma detallada los distintos algoritmos que se han llevado a cabo, y cómo se ha estructurado y llevado a cabo dicha implementación. Veremos por tanto, un esquema de funcionamiento de los distintos modelos y técnicas utilizadas, además del diseño llevado a cabo para su puesta en ejecución.

4. Diseño de la parte experimental

En cuanto al diseño del proyecto, vamos a distinguir varias partes diferenciadas en relación a la propia funcionalidad de cada uno de los algoritmos que se van a implementar e utilizar en nuestro problema.

Estas partes serían las siguientes:

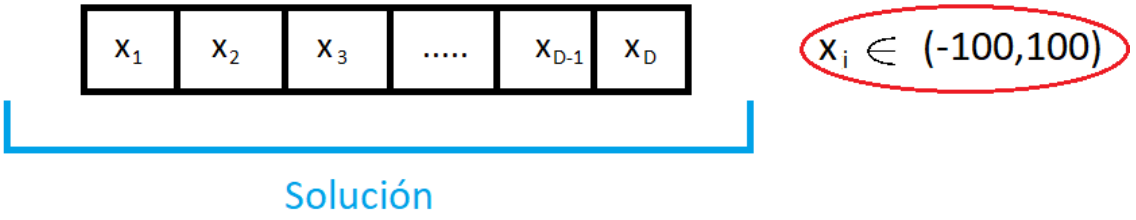
- *Representación de las soluciones* del problema, en la cuál se explicará cómo son las soluciones, de cara a orientar el desarrollo del modelo adaptado a las mismas
- *Función Objetivo.* En nuestro caso, tendremos 30 funciones objetivo diferentes, puesto que hemos utilizado un *Benchmark* que incorpora diferentes casos.
- Algoritmos Genéticos
    1. Algoritmo Genético *generacional.*
    2. Algoritmo Genético *estacionario.*
    3. Operadores y métodos.
- Modelos Híbridos
    1. *Modelo de Islas clásico.*
    2. *Modelo Híbrido Ejecución por fases.*
    3. *Modelo Híbrido con Probabilidades Adaptativas.*

A continuación vamos a proceder a profundizar en cada una de las fases que hemos diferenciado.

4.1. Representación de la Solución

En primer lugar, la primera cuestión de diseño que hay que barajar es cómo se van a representar las posibles soluciones del problema.

En este caso, estamos tratando problemas de codificación real, por lo que lo más indicado será utilizar un vector de tantos elementos como sea la dimensión escogida, todos dentro del rango establecido en la documentación de **CEC '14**.



En nuestro caso, hemos escogido trabajar con soluciones de dimensión 10 y dimensión 30, con valores comprendidos en el rango (-100.0, 100.0), siguiendo las especificaciones del Benchmark.

4.1.1. Representación de una población

En este ámbito, llamamos *población* a un conjunto de soluciones, por lo que para su representación, lo haremos utilizando un vector de soluciones.

4.2. Función Objetivo

Otra cuestión importante es el cálculo del valor *Fitness* de cada una de las soluciones. En nuestro caso, no estamos tratando un único problema (que por tanto esté asociado a una única función objetivo), sino que estamos haciendo pruebas de un Benchmark con 30 problemas diferentes. Con el código de dicho Benchmark, se nos facilita una función que nos permite el cálculo del *Fitness* de una solución para uno de los problemas contenidos en él.

Por tanto, para saber el *Fitness* de una solución, haremos una llamada a dicha función, la cuál nos permitirá saber el valor asociado a la solución y problema en cuestión.

En la imagen que se muestra a continuación podemos ver una representación gráfica del funcionamiento de este proceso. Partimos de un vector que representa una solución (s1), donde, en un primer momento no tiene asociado ningún valor *Fitness* porque no se le ha consultado, por lo que dicha variable tiene un valor desconocido. Para averiguar su valor, hacemos una llamada a la función que lo calcula, pasándole la solución, la variable *Fitness*, sus tamaños y el número de problema a evaluar. Una vez que se lleva a cabo dicho procedimiento, en la variable *Fitness* se guarda un número real f, que se corresponde con el *Fitness* correspondiente.

4.3. Algoritmos Genéticos

Para los algoritmos genéticos, vamos a distinguir entre dos enfoques: *generacional* y *estacionario*. La mayor parte del procedimiento es común en los dos casos, ya que ambos enfoques llevan a cabo las distintas etapas generales del algoritmo: *selección*, *cruce* y *reemplazo*, por lo que muchos de los métodos utilizados serán comunes.

Inicialización del Algoritmo

Para ambos enfoques, la población de partida se genera de manera aleatoria, y se va modificando a lo largo de las iteraciones del bucle interno del algoritmo.

Condición de Parada

La condición de parada de todos los algoritmos va a ser el número de llamadas que se realicen a la función objetivo con el fin de obtener el valor *fitness* de las soluciones. A esta función la llamaremos cada vez que generamos un nuevo hijo durante el proceso de ejecución.

4.3.1. Algoritmo Genético Generacional

Como hemos comentado previamente, vamos a implementar dos enfoques de algoritmos genéticos. En este apartado vamos a tratar la versión *Generacional* del algoritmo. En esta versión se genera, en cada iteración del algoritmo, una nueva generación, la cuál es utilizada en la posterior vuelta en la ejecución.

Para ello, en primer lugar partimos de una población inicial del tamaño que se haya establecido (P_{actual}) y se llevarán a cabo las siguientes fases en las distintas vueltas que dé el algoritmo:

1. **Proceso de Selección** donde determinaremos qué individuos de la población participarán en los cruces para determinar la siguiente generación. Todos estos individuos serán incluidos en una población, la cuál será la que contiene los individuos a reproducirse: *Población de los Padres* (P_{padres}).

2. **Proceso de Cruce**, donde se crearán los descendientes a partir de los padres seleccionados para el cruce. Se darán dos posibles casos:

a) Habrán parejas candidatas que se cruzarán y generarán dos hijos (los cuáles automáticamente pasarán a formar parte de la población de los hijos (P_{hijos}).

b) También tendremos el caso de parejas candidatas que no llegarán a cruzarse, las cuáles se copiarán automáticamente en la población de los hijos (P_{hijos}).

De esta forma, se generará una nueva población del mismo tamaño de la de partida, formada por individuos de la población, pero también por hijos resultantes de cruces entre individuos de la población.

3. **Proceso de Reemplazo**: para mantener en el algoritmo el mejor individuo generado hasta el momento (Elitismo). Cuando hemos generado la población que pasa a la siguiente iteración, nos aseguramos de que el mejor individuo generado en la nueva población supera o iguala al generado en iteraciones anteriores. Si esto no ocurre, copiamos este mejor individuo en la nueva generación, para así no perder nunca la mejor solución encontrada.

A continuación podemos ver de una forma gráfica el proceso que realiza este algoritmo en la imagen 1.

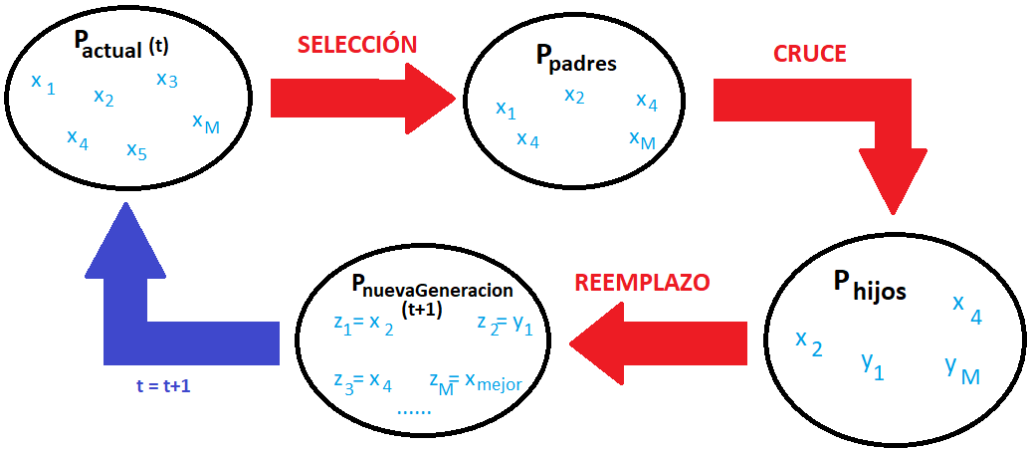


Figura 1: Funcionamiento del Algoritmo Genético Generacional

4.3.2. Algoritmo Genético Estacionario

También se llevará a cabo una versión Estacionaria del **Algoritmo Genético**. A diferencia de la versión generacional, en este caso no se genera una nueva generación por iteración, sino que únicamente se crea un hijo por iteración del algoritmo, el cuál compite por formar parte de la población. A pesar de dicha diferencia, las fases del Algoritmo Genético se mantienen.

1. **Proceso de Selección** donde determinaremos qué dos individuos de la población participarán en el cruce a realizar. En este caso no necesitamos una población para determinar qué padres se van a reproducir, por lo que prescindiremos de P_{padres}.

2. **Proceso de Cruce**, donde se creará un descendiente a partir de los padres seleccionados para el cruce. En este caso, no existe una probabilidad de cruce, sino que la pareja elegida se va a reproducir siempre.
3. **Proceso de Reemplazo**, donde el hijo generado compite para entrar o no a la población actual en función de su valor *Fitness*. En este caso, a diferencia de la versión Estacionaria, contemplaremos un criterio Elitista pero también habrá casos donde utilicemos otro criterio diferente (*Crowding Determinístico*).

A continuación podemos ver una representación gráfica del proceso que sigue esta versión en la imagen 2.

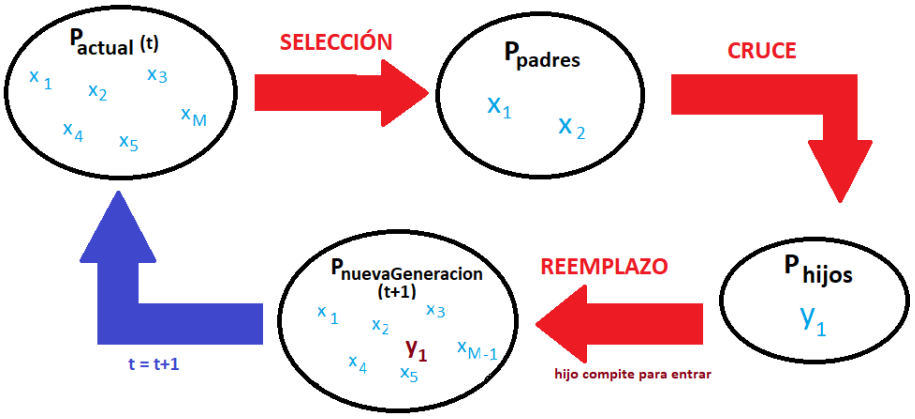


Figura 2: Funcionamiento del Algoritmo Genético Estacionario

4.3.3. Operadores y métodos

En primer lugar vamos a ver los operadores y métodos que vamos a utilizar para el desarrollo del algoritmo generacional, los cuales vamos a exponer a continuación. Habrá métodos que será comunes para ambas versiones, como puede ser la inicialización de la población o el operador de cruce, y otros que utilicemos específicamente para unas versiones u otras del algoritmo, como puede ser el caso de los distintos operadores de *Selección*.

Inicialización de la Población Para poder comenzar a ejecutar los distintos algoritmos, hay que partir de una población inicial, desde la cuál se irán generando descendientes para formar u actualizar las poblaciones de las siguientes iteraciones. Por ello, crearemos una serie de vectores aleatorios que formen la población de partida.

Parámetros

En nuestro caso, hemos trabajado con poblaciones de tamaño 60, tanto para los algoritmos individuales, como para los algoritmos incluidos dentro de los modelos híbridos.

Cabría la posibilidad de que los resultados se vean afectados por el hecho de que comparemos algoritmos individuales de poblaciones de tamaño 60 con modelos híbridos donde cada uno de sus cuatro algoritmos utiliza tamaño 60 de población, por

lo que también se ha realizado un estudio con poblaciones de tamaño 240 para los algoritmos individuales. Con ello se intenta comprobar si afecta o no el aumento del tamaño de la población en los resultados que estamos obteniendo.

Operador de selección Se van a contemplar dos posibles operadores de selección a utilizar: `Torneo Binario` y `Emparejamiento Variado Inverso`.

▷ *Torneo Binario*

La idea principal de este operador se basa en seleccionar los individuos que se van a cruzar mediante una comparación previa de los individuos.

En este proyecto se ha utilizado la versión **determinística** del Torneo, donde escogeremos dos elementos de la población de forma aleatoria de entre todos los individuos de la población, de entre los cuáles obtendremos el padre seleccionado por el método, que se corresponderá con el que sea más prometedor de los dos elegidos.

El criterio que hemos elegido va a provocar una presión de selección alta, ya que estamos teniendo en cuenta a todos los individuos de la población, y los individuos menos competitivos tendrán menos oportunidades que si hubiésemos elegido un subconjunto de la población donde aplicar el torneo.

Cuando nos decantemos por utilizar este operador, se realizará el proceso dos veces (una vez para cada uno de los padres con los que realizar el posterior cruce).

▷ *Emparejamiento Variado Inverso*

Este método, al contrario que el *Torneo Binario*, no genera los dos padres de igual forma. En primer lugar, se obtiene un padre de la población de manera aleatoria. Para ello, de manera aleatoria, escogemos un número **k** en el intervalo [0,`tamPoblacion`), donde `tamPoblacion` representa el número de individuos que forman parte de la población. Este número **k** que obtenemos representa al individuo en la posición número k en el vector de individuos que representa la población.

En un segundo lugar, y de igual manera al primer paso, se escogen N_{NAM} padres, y de entre ellos nos quedamos con el que sea más distante al primer padre generado.

Por último, quedaría por determinar el criterio para considerar un individuo el más distante al primer padre obtenido. Para ello, a diferencia del método *NAM* clásico, que utiliza un criterio de distancia de fenotipo, nosotros hemos considerado un criterio de distancia *Fitness*. Para ello, hacemos una resta de valores absolutos del *Fitness* del primer padre contra los valores *Fitness* de los candidatos a ser el segundo padre, y aquel cuya diferencia sea la mayor, será considerado el candidato más distante y, por tanto, el que nos quedaremos.

Con este método, a diferencia del anterior, conseguimos una mayor diversidad en las poblaciones, ya que estamos propiciando que los padres sean diferentes dentro de lo posible. **Operador de Cruce** Para el cruce de los individuos de la población, se ha hecho uso del operador de cruce para codificación real **BLX-α**.

Dados dos individuos de la población, se van a generar sus descendientes. Para ello, vamos a ir comparando el valor en las posiciones iésimas de los individuos con el

fin de determinar cuál es el máximo y mínimo de entre los dos valores de dichos vectores (C_{max} y C_{min}) respectivamente.

A partir de dichos valores, determinaremos un tercer parámetro al que denominamos I, el cual se define como:

$$I = C_{max} - C_{min}$$

Con I, C_{max} y C_{min} definidos, ya podemos calcular el valor que va a tomar el hijo en su posición iésima generando un valor real aleatorio que se encuentre en el siguiente intervalo:

$$[I = C_{min} - I \cdot \alpha, C_{max} + I \cdot \alpha]$$

Por último, faltaría por determinar el valor del parámetro α, pero este se lo pasaremos por parámetro al algoritmo, ya que probaremos el método para distintos valores de ese parámetro.

Este procedimiento se realiza con cada uno de los elementos de los vectores de los individuos hasta completar el vector correspondiente al hijo, y una vez calculado, calculamos su valor *Fitness* para poder almacenarlo.

Parámetros

- Para este operador, únicamente es necesario determinar el valor que va a tomar α, lo cuál será determinante para formar el descendiente, ya que si utilizamos valores de α más altos estaremos considerando un intervalo mayor (y una mayor posibilidad de valores) que si estuviésemos considerando valores de α más pequeños.

  En total, vamos a estudiar este algoritmo con los valores 0.1, 0.3, 0.5 y 0.8 en el caso de la versión *generacional* del algoritmo, y con un α de 0.5 para la versión *estacionaria*. Esta última decisión se basa en las buenas características que proporciona dicho parámetro, como se puede ver en [21]

- Por otra parte, no todos los padres seleccionados necesariamente generarán descendientes, sino que existe una probabilidad entre las parejas candidatas a cruzarse. La probabilidad de cruce utilizada tiene un valor de 0.6.

Operador de reemplazo Para llevar a la práctica la fase de reemplazo, hemos tenido en cuenta dos posibles operadores: `Reemplazar Peor (RW)` y `Crowding Determinístico` (DC).

▷ *Reemplazar Peor*

Se trata de un criterio elitista, que tiene como objetivo garantizar que el mejor individuo generado por el algoritmo no se pierda durante la ejecución. De esta forma, si la mejor solución conseguida no se mantiene en la siguiente generación obtenida, ésta se incluye de forma automática sustituyendo al peor individuo generado en dicha nueva población.

▷ *Crowding Determinístico*

Este operador solo lo vamos a utilizar en la versión estacionaria, para determinar si el descendiente va a formar parte o no de la población que pasa a la siguiente iteración. Para ello, en primer lugar se localiza el individuo de la población actual más parecido al descendiente generado. Se comparan los valores *Fitness* de ambas soluciones, y ambas compiten por ocupar esa posición de la población. La solución más prometedora de las dos será la que pase a la siguiente ejecución, ocupando el lugar del individuo más parecido encontrado en la población.

4.4. Modelos Híbridos

Para cada uno de los modelos que se van a implementar, hemos planteado tres combinaciones de los algoritmos genéticos que se han implementado de forma individual. Las combinaciones que se van a ejecutar serán las siguientes:

1. Una versión con **Algoritmos Genéticos Generacionales** con valores de α=0.1, α=0.3, α=0.5 y α=0.8 para el operador de cruce BLX-α.
2. Una versión para **Algoritmos Genéticos Estacionarios**, con las siguientes características:
   - AGE con *Torneo Binario* como operador de selección y *Reemplazar Peor* como operador de reemplazo.
   - AGE con *Emparejamiento Variado Inverso* como operador de selección y *Reemplazar Peor* como operador de reemplazo.
   - AGE con *Torneo Binario* como operador de selección y *Crowding Determinístico* como operador de reemplazo.
   - AGE con *Emparejamiento Variado Inverso* como operador de selección y *Reemplazar Peor* como operador de reemplazo.
3. Una versión combinando los dos mejores Algoritmos Genéticos *Generacionales* y los dos mejores Algoritmos Genéticos *Estacionarios* obtenidos para el problema.

Las combinaciones anteriores se van a ejecutar para cada uno de los modelos de hibridación implementados, los cuales explicaremos en los apartados siguientes.

4.4.1. Modelo de Islas

En primer lugar, hemos implementado un Modelo de Islas clásico que utiliza un modelo de vecindario aleatorio. Podemos dividir el procedimiento del modelo en dos fases principales.

1. **Ejecución de los algoritmos**: en primer lugar se ejecutan los cuatro algoritmos que forman el modelo. Esta ejecución es individual, por lo que cada uno de los algoritmos trabajará con una población propia sobre la que realizará los cambios correspondientes.

2. **Migración de las soluciones**: una vez que se han ejecutado los algoritmos, procedemos a la migración de la mejor solución de cada algoritmo a uno de los tres restantes. Para ello, vamos a seguir el orden impuesto por una permutación que se genera de manera aleatoria en cada iteración.

En la imagen 3 podemos ver el esquema de funcionamiento del modelo de una forma gráfica.

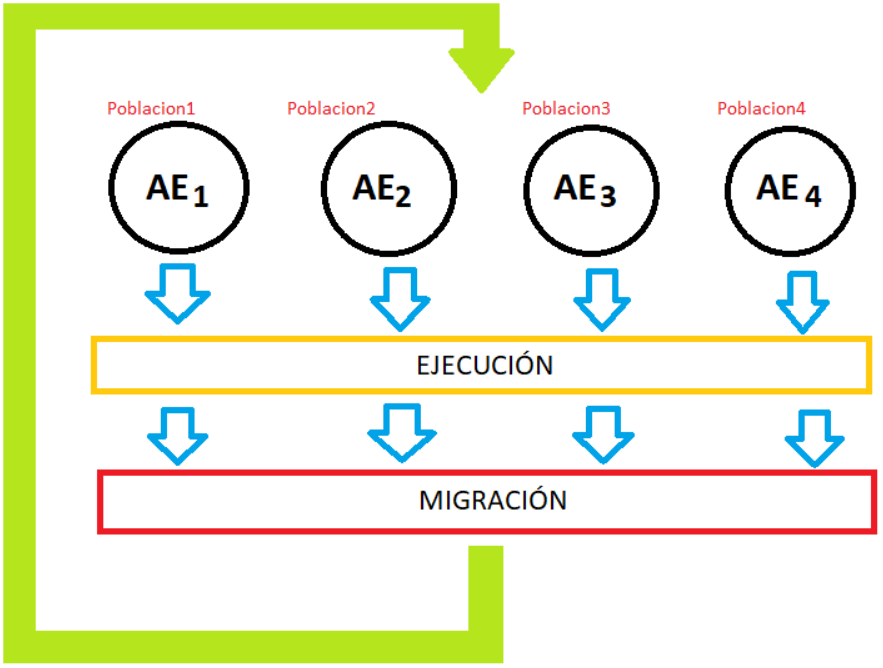


Figura 3: Funcionamiento del Modelo de Islas

Por último, podemos ver a en la imagen 4 el funcionamiento de las migraciones que estamos utilizando en el Modelo de Islas con un ejemplo concreto. Como se observa, cada número de la permutación representa a un algoritmo. Siguiendo dicha permutación, el algoritmo en la posición i le pasará su mejor individuo al algoritmo en la posición i+1 del vector. Para completar el círculo, el individuo en la última posición de la permutación trasmitirá su mejor individuo al individuo en la primera posición.

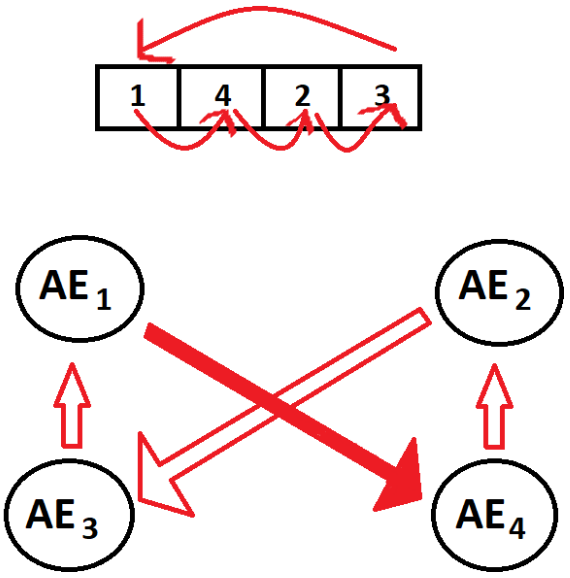


Figura 4: Funcionamiento de la migración del Modelo de Islas

4.4.2. Modelo híbrido: Ejecución en dos fases

Este segunda técnica consiste en implementar un modelo donde la ejecución del algoritmo se va a llevar a cabo en dos fases:

1. Una primera fase donde cada algoritmo se ejecuta durante un 5 % del tiempo total a invertir en el algoritmo. De la forma que lo hemos implementado, cada algoritmo se ejecutará con un número máximo del 5 % de las llamadas que puede realizar el modelo a la función que calcula el valor *Fitness*.
2. Una segunda fase donde el algoritmo que ha obtenido los mejores resultados en la primera ejecución se va a lanzar durante el 80 % del tiempo restante.

En la imagen 5 podemos ver una representación gráfica de este modelo, donde se observa el funcionamiento adaptativo del mismo.

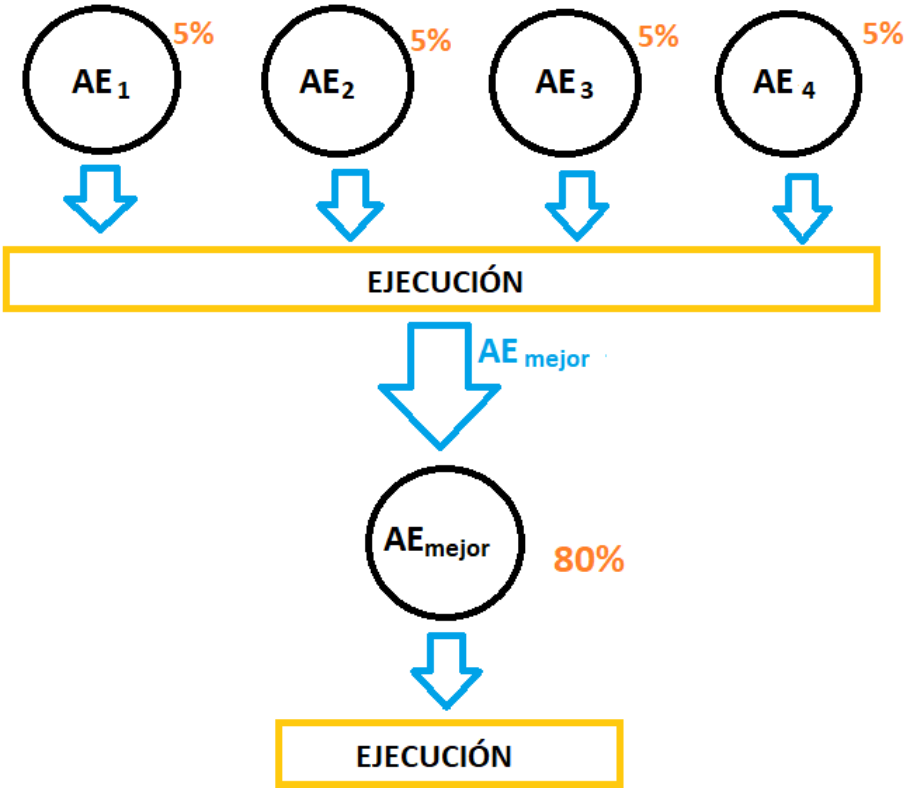


Figura 5: Esquema de funcionamiento del Modelo de Ejecución en 2 Fases

4.4.3. Modelo híbrido: Probabilidades adaptativas

Este tercer modelo vuelve a tener una idea similar a los anteriores en el sentido de que ejecutará los distintos algoritmos que lo contienen, con la diferencia de que no controlaremos de forma exacta qué algoritmo se va a ejecutar en cada iteración del modelo. Esta decisión vendrá dada en función de un valor de probabilidad.

Cada uno de los algoritmos se ejecutará en función de la probabilidad asignada al mismo. Inicialmente, todos los algoritmos tienen la misma probabilidad: como estamos ejecutando 4 algoritmos, cada uno de ellos tendrá un valor igual a 0.25. Por tanto, inicialmente partiríamos de un vector de probabilidades como el siguiente.

$$[0.25, 0.25, 0.25, 0.25]$$

Por otra parte, necesitamos un vector de probabilidades acumuladas, para saber entre qué rango debe salir el valor aleatorio para que se ejecute un algoritmo u otro. De esta forma, el vector de probabilidades acumuladas correspondiente al ejemplo anterior sería el siguiente.

$$[0.25, 0.5, 0.75, 1]$$

Por tanto, en cada iteración del modelo, iremos obteniendo un valor aleatorio, y en función del vector de probabilidades acumuladas ejecutaremos el algoritmo que corresponda al rango que contiene a ese valor aleatorio. Cuando se hayan realizado un número especificado de iteraciones, actualizaremos el vector de probabilidad en función de la mejora que hayan obtenido los distintos algoritmos. De esta forma, se asignará un valor de probabilidad mayor al algoritmo en cuestión cuanto mayor sea la mejora en comparación a la conseguida por el resto. Como es de esperar, cada vez que actualicemos la probabilidad, actualizaremos consecuentemente las probabilidades acumuladas.

A continuación podemos ver en la imagen 6 un esquema acerca de cómo funcionaría este modelo.

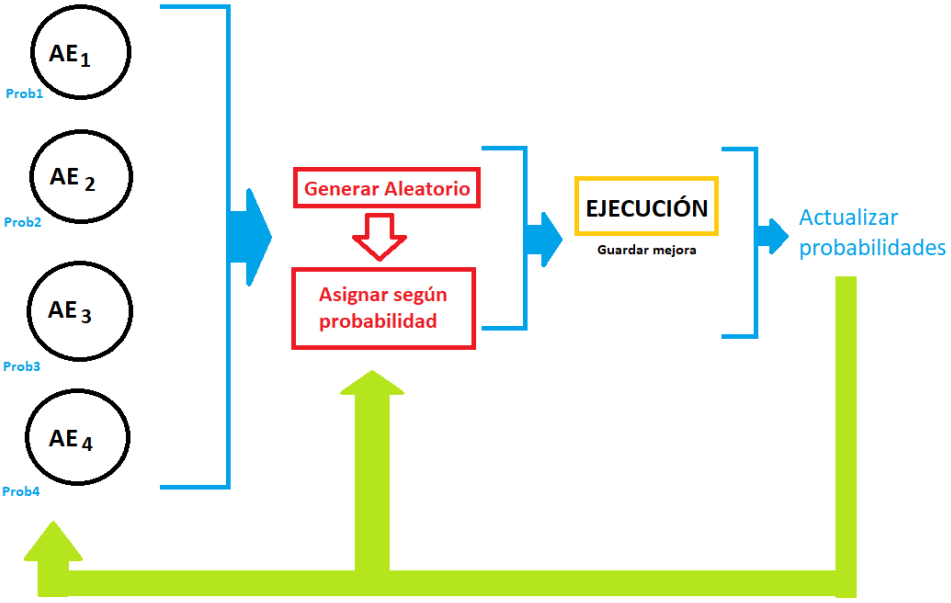


Figura 6: Esquema de funcionamiento del Modelo de Ejecución en 2 Fases

Implementación

RESUMEN: En esta sección, trataremos cuestiones más técnicas de programación que han sido llevadas a cabo a la hora de implementar el proyecto. Veremos por tanto, detalles necesarios para poder obtener los distintos algoritmos y modelos que se tratan en este estudio.

5. Implementación

Al igual que en la sección de *Diseño* de este proyecto, el trabajo de `Implementación` ha sido dividido en varias partes diferenciadas en relación a la propia funcionalidad de cada una de ellas. Vamos a profundizar en estos aspectos desde el punto de vista de implementación.

5.1. Representación de la Solución del problema

Para la representación de las soluciones del problema se ha implementado una clase `Solucion`, que contiene toda la funcionalidad necesaria para poder trabajar con las soluciones del problema.

Como se ha comentado previamente, estamos trabajando con una serie de problemas que se caracterizan por ser de codificación real, por lo que necesitaremos hacer uso de una estructura que nos permita trabajar de esta forma. Por ello, cada solución del problema va a ser representada como un vector de números reales de una dimensión dada.

Para poder trabajar de una forma más cómoda, se han implementado métodos que nos permitirán realizar todas las operaciones necesarias para poder trabajar con la estructura de las soluciones elegidas.

5.1.1. Solución Aleatoria

Con este método, podemos crear una solución de dimensión `tam`, con valores aleatorios comprendidos en el rango `(valorIni, valorFin)`.

Para ello, en primer lugar se redimensiona el vector que guarda la solución al tamaño especificado, y mediante un bucle, vamos asignando valores aleatorios a los elementos contenidos en el vector hasta completarlo. Para generar dichos valores, se realizará mediante otro método al que hemos denominado `valorAleatorioFloat()`.

Podemos ver la implementación en el pseudocódigo que se muestra a continuación.

```
1  begin solucionRandom(tam,  valorInicio, valorFin)
2
3      solucion.resize(tam)
4
5
6      while(cont<tam)
7          solucion[cont] = generarAleatorio(valorInicio, valorFin);
8          cont++;
9      endwhile
10
11     calcularFitness();
12 end
```

5.1.2. Generar valor real aleatorio

El método `valorAleatorioFloat()` se encarga de obtener un valor aleatorio de tipo real, comprendido en un intervalo que hay que especificar por parámetro. Por ejemplo, si llamamos a la función *ValorAleatorioFloat()* con los parámetros 2.0 y 3.5 respectivamente, ésta devolverá un número real comprendido entre 2.0 y 3.5.

Para obtener el valor aleatorio, el método llama a una función incluida en un fichero generador de valores pseudoaleatorios. Este fichero, se encuentra añadido en el proyecto.

5.1.3. Generar valor entero aleatorio

El método `ValorAleatorio()` se encarga de obtener un valor aleatorio de tipo entero, comprendido en un intervalo que hay que especificar por parámetro. El funcionamiento es idéntico al método anterior, únicamente modificando el tipo de dato con el que trabajamos (en este caso, es de tipo entero). Para adquirir el valor aleatorio, el método llama a una función incluida en un fichero, el cuál contiene un generador de valores pseudoaleatorios.

5.1.4. Calcular *Fitness*

Por último, debido a la representación de soluciones que hemos elegido, una función esencial debe ser el poder calcular el valor fitness correspondiente a dicha solución. Como estamos utilizando las funciones de **CEC '14**, para el cálculo del valor de *fitness* podemos realizar una llamada a la Función *Fitness* proporcionada en dicho proyecto. Por tanto, hemos creado un método al cuál se le ha denominado `calcularFitness()`, que se encarga de llamar a la función correspondiente del código facilitado por **CEC '14**.

A continuación se puede ver cómo se hace dicha llamada a la función que calcula el *fitness*. La función `cec14_test_func()` requiere el uso de punteros a los vectores, por lo que se ha adaptado el código para ello. Como se puede observar, a dicha función, además de pasarle los datos necesarios para poder calcular y guardar el valor *Fitness*, también se indica el número de función que se quiere ejecutar. Esto permite, mediante una única función, calcular el *fitness* para cualquiera de las que proporciona el Benchmark.

```
1 begin calcularFitness()
2
3     vector(double) fit[1];
4
5     cec14_test_func(&solucion,&fit,solucion.size(),1,funcion);
6
7     fitness = fit[0];
8 end
```

Para no calcular el *fitness* cada vez que sea necesario consultarlo, se ha definido un atributo de la clase **Solucion** para poder almacenar su valor, y por tanto hacer el cálculo una única vez. Estamos hablando del atributo `fitness`, una variable de tipo real que contendrá el valor *fitness* asociado a la solución a la que representa cada objeto de dicha clase. Este valor lo actualizaremos cada vez que se cree o modifique una solución del problema.

5.2. Algoritmos Genéticos

Como se comentó en el apartado anterior, estamos trabajando con dos enfoques del Algoritmo Genético: **generacional** y **estacionario**. Cada uno de ellos tiene sus particularidades, pero podemos separar las etapas del algoritmo en métodos, para reutilizar todas aquellas partes que sean comunes en ambos enfoques.

Aunque la implementación de las dos versiones es distinta, por motivos de eficiencia en código, hemos utilizado una variable booleana, para que, dependiendo de la versión, se ejecuten los métodos necesarios para la misma.

A continuación vamos a ver todos aquellos métodos implementados.

5.2.1. Operadores y métodos

Inicialización de la Población

Para poder poner en marcha el algoritmo, necesitaremos generar una población inicial de individuos de tamaño N. Habrá casos en los que sea preferible que se genere de manera automática cuando mandemos ejecutar el algoritmo, y otros donde queramos especificarla manualmente, pero en cualquier caso, necesitaremos generarla de alguna forma.

Para ello ha sido implementada una función que genera la población inicial del algoritmo (`crearPoblacionInicial()`). Dicha función, se encarga de almacenar dentro de la estructura utilizada para la población, tantos individuos como elementos de la población queramos. Para generar cada uno de esos algoritmos, se utilizará el método de la clase Solución llamado `solucionRandom()` que, a partir de un número de dimensión `(dimSolucion)`, un valor inicial `(valorIni)` y un valor final `(valorFin)`, se genera un vector de dimSolución elementos, con valores generados aleatoriamente en el rango `(valorIni,valorFin)`. A continuación podemos ver el código correspondiente a esta sección.

```
1 begin crearPoblacionInicial()
2 {
3     for (i=0, i < poblacionInicial.size())
4         poblacionInicial(i) = solucionRandom(dimension,valorInicio,
              valorFin);
5
6 }
7 end
```

Operador de selección

Para la implementación del método de selección se han contemplado dos posibles operadores de selección diferentes. Por una parte se ha implementado el método de selección por `Torneo Binario`, y por otra parte el método de Emparejamiento Variado Inverso (NAM).

En la práctica, vamos a hacer uso de una variable de tipo entero llamada `metodoSeleccion`, la cuál se encarga de representar el operador de selección que se quiere utilizar. En el caso de que el valor de dicha variable sea `1`, se realizará un *Torneo Binario*, y en caso contrario se llevará a a cabo el *Emparejamiento Variado Inverso*. De esta forma podemos reducir el número de líneas de código ya que contemplamos las dos variantes de forma simultánea.

Para decidir el operador que se va a utilizar en la ejecución, se ha implementado un método `seleccion` que ejecuta un criterio u otro en función de la variable `metodoSeleccion` ya mencionada anteriormente. A continuación se muestra el código asociado a dicho método.

```
1 begin seleccion(individuo1, individuo2)
```

```
2
3     if (metodoSeleccion == 1)
4         selectTournament(individuo1, individuo2);
5
6     else
7         selectNAM(individuo1, individuo2);
8 end
```

En los siguientes apartados vamos a ver cómo se han implementado cada uno de los métodos de selección previamente comentados.

Como se comentó en la sección de Diseño, con el proceso de Selección obtenemos una población formada con los posibles individuos a reproducirse (P_{padres}). En este caso, nuestra población de padres va a estar formada por los índices que representan la posición de los individuos en el vector Población. De esta forma, accederemos a los Padres en función de los índices que elijamos.

Torneo Binario

En este caso, utilizamos dos números enteros. que representan los índices de los dos individuos. A dichos valores los que he llamado `individuo1` e `individuo2`. En cada una de estas variables, almacenamos el índice del padre seleccionado, haciendo uso de la función `seleccionTorneoBinario()`.

Para implementar el Torneo Binario se generan dos índices aleatorios, (en la práctica nos tendremos que asegurar de que no coincidan), los cuáles van a representar a dos posibles individuos de la población. Una vez que tenemos estos dos elementos, los comparamos, y nos quedamos con aquel que tenga un menor valor de *Fitness* (y por tanto sea una solución más prometedora). El número de individuo que se corresponda con dicha solución es el que devuelve la función al finalizar el procedimiento.

Emparejamiento Variado Inverso

En este caso, para el primer individuo generamos un valor aleatorio cualquiera. Para el segundo padre, que, tal y como comentamos en la sección de Diseño, se lleva a cabo un procedimiento distinto: dicha tarea se implementa en el método `seleccionNAM()`, de manera que el procedimiento a llevar a cabo será algo como lo que aparece a continuación.

```
1 begin selectNAM(individuo1, individuo2)
2     individuo1 = generarAleatorio(0,tamPoblacion-1);
3     individuo2 = seleccionNAM(individuo1);
4 end
```

En el método `seleccionNAM()` obtenemos tres individuos de la población de manera aleatoria. Estos individuos serán los candidatos a ser el segundo padre del cruce.

Para cada uno de los individuos que obtenemos, nos aseguramos que no sea uno de los seleccionados previamente. Esto quiere decir que cuando obtenemos el primer individuo al azar, nos aseguramos de que no coincida con el padre ya elegido, cuando obtengamos el segundo individuo, que este no coincida ni con el padre, ni con el individuo ya seleccionado, y así sucesivamente. En caso de coincidencia, seguiríamos generando números aleatorios hasta que saliera uno no repetido.

Cuando tenemos los tres candidatos seleccionamos, consultamos sus valores de *Fitness* para saber cuánto se diferencian del padre. Con este cálculo, podremos ver cual es el individuo candidato más distante al padre que ya habíamos seleccionado previamente. En la práctica, se han almacenado la diferencia entre cada valor *Fitness* candidato y el padre seleccionado en un vector de tres elementos, donde el índice en el que se guarda dicho valor representa al candidato en cuestión. Una vez completo dicho vector, hemos buscado el valor más distante para determinar cuál, de entre los candidatos, debe ser el segundo padre elegido en el proceso de selección.

Operador de Cruce

Para la implementación del operador de cruce $BLX-\alpha$, vamos recorriendo elemento a elemento los valores contenidos en ambos padres de forma paralela, generando el descendiente de la forma que ya se explicó anteriormente.

Una vez que se genera el descendiente, realizamos una llamada a la función objetivo para poder conocer su valor *Fitness* asociado y almacenarlo al igual que todos los demás individuos que forman parte de la población. Esta acción se deberá realizar cada vez que se genere un nuevo individuo.

A continuación podemos ver un breve pseudocódigo que resume la implementación del operador.

```
1  begin OperadorCruce(padre1, padre2)
2      for i=0 to hijo.size() do
3      begin
4
5          Cmax = max(padre1(i),padre2(i));
6          Cmin = min(padre1(i),padre2.(i));
7          I=Cmax–Cmin;
8
9          hijo[i] = generarAleatorio(Cmin–I*alfa,Cmax+I*alfa);
10
11     end
12
13     hijo.calcularFitness();
14 end
```

Operador de reemplazo

Como se comentó anteriormente, se han implementado dos operadores de reemplazo, por lo que, para hacer una diferenciación a la hora de realizar las ejecuciones,

haremos uso de una variable llamada `reemplazarPeor`. Esta variable tendrá valor `True` cuando se vaya a ejecutar RW, y valor `False` en caso contrario.

La elección de qué operador utilizar se gestionará mediante un método creado para ello (`Replacement()`). En dicha función, se recibe como parámetro el posible candidato a reemplazar, y, dependiendo del valor que tome la variable `reemplazarPeor`, se llevará a cabo un criterio u otro de reemplazamiento:

- Si la variable está a `True`, devolveremos el peor elemento de la población (que será el que sustituiremos).
- En en caso contrario, llamaremos a la función `solucionMayorParecido` para llevar a cabo el criterio *DC*.

De esta forma, podremos diferenciar entre ambos métodos en tiempo de ejecución.

Para llevar a cabo el criterio de `Crowding Determinístico`, tal y como he mencionado previamente se ha hecho uso una función llamada `solucionMayorParecido()`. Dicha función recibe un individuo por parámetro, y recorre todos los individuos de la población en busca del que tenga la menor diferencia absoluta con el pasado por parámetro. El individuo que cumpla con esta condición se corresponderá con el más parecido de la población, y por tanto el candidato a sustituir en el reemplazo.

Ordenar población

Para trabajar con poblaciones de individuos ordenadas en función del valor *fitness* se ha implementado un método llamado `ordenarPoblacion()`, el cuál realiza una ordenación de los individuos mediante el método de inserción, en función de menor valor *Fitness*. Con ello, se facilita la localización del mejor y peor elemento de cada población.

5.2.2. Algoritmo Genético Generacional

Una vez vistos todos los operadores y métodos que se van a utilizar, procedemos a ver la implementación del algoritmo genético generacional. Recordemos que en esta versión del algoritmo, la población que pasa a la siguiente iteración es una renovación de la anterior. En otras palabras, se genera una nueva generación en cada iteración del algoritmo.

En primer lugar, el algoritmo, recibe dos variables por parámetro.

- El primer parámetro es `poblacionInicial`, y se corresponde con la población de individuos con la que comenzará el algoritmo a ejecutarse. Esta población inicial será la primera generación del algoritmo. La variable `poblacion`, representa en el algoritmo la población actual en la ejecución, por lo que, en la primera iteración, dicha variable se corresponderá con la población inicial.
- El segundo parámetro es `iteraciones`, y se corresponde con el número total de evaluaciones que se realizarán a la función objetivo. Inicialmente, tendremos tantas evaluaciones realizadas como elementos en la población (ya que a

cada uno de estos individuos se les debe haber evaluado para conocer su valor *Fitness*), por lo que el contador de evaluaciones será inicializado con el valor del tamaño de la población (almacenado en `tamPoblacion`.

Como vimos en el diseño, el algoritmo genético en esencia, realizará, un proceso de selección, un cruce en función de un valor de probabilidad para obtener los hijos, y finalmente un reemplazamiento, antes de continuar con la siguiente iteración. Por tanto, implementamos un bucle que se realiza mientras que no se supere el número de llamadas a la función objetivo establecidas.

En este caso, vamos a contemplar los dos posibles **operadores de selección** vistos (Torneo Binario y Emparejamiento Variado Inverso) para elegir los padres.

Para realizar el **cruce**, obtendremos un valor aleatorio, que hemos almacenado en la variable `pos`. En función de que este valor supere o no el valor asignado a la probabilidad de cruce, se realizará una cosa u otra. Si no lo supera, significa que se realiza el cruce, por lo que obtendremos dos hijos a partir de los padres elegidos en el proceso de selección. Estos hijos lo añadiremos a la población de hijos que hemos declarado (`poblacionHijos`), y aumentaremos el número de llamadas a la función objetivo en dos. En caso de que no se tenga que realizar el cruce, copiamos los dos padres en la población de los hijos. Este proceso lo realizamos un número de veces igual al tamaño de población entre 2, que es el número máximo de emparejamientos que se crearán.

Una vez hemos obtenido la nueva generación, almacenada en `poblacionHijos`, ordenamos dicha población, con el objetivo de controlar mejor los individuos más y menos prometedores de la generación. Para terminar el proceso, faltaría llevar a cabo el **reemplazamiento**. En el algoritmo, estamos siguiendo un proceso *elitista*, ya que en esta última etapa, nos aseguramos de que el mejor de la población anterior no se pierda en esta nueva generación. Para ello, comprobamos si dicho elemento se mantiene, y en caso contrario, lo intercambiamos directamente por el mejor de la nueva generación (que en esta situación será menos prometedor que el mejor de la generación de los padres).

5.2.3. Algoritmo Genético Estacionario

Para terminar con esta sección, nos quedaría por ver la segunda versión del Algoritmo Genético que vamos a llevar a cabo.

El algoritmo estacionario comienza de igual modo que la versión anterior: inicializamos la población, la ordenamos en función del coste y actualizamos el número de evaluaciones de la función objetivo. Sin embargo, en este caso, en vez de realizar un bucle con las sucesivas acciones para formar la población de hijo,únicamente seleccionamos dos padres y los cruzamos, generando un hijo. De nuevo, la selección del hijo se podrá llevar a cabo mediante dos criterios (*Torneo Binario* o *Emparejamiento Variado Inverso*).

A la hora de llevar a cabo el reemplazamiento, tendremos en esta versión dos criterios (*Reemplazar Peor* y *Crowding Determinístico*), por lo que utilizamos la función `replacement()` ya mencionada anteriormente para obtener el individuo candidato

a reemplazar en función del criterio que hayamos elegido previamente utilizar. Una vez obtenido el candidato, este competirá con el hijo generado, y aquel que sea más prometedor (o dicho de otra forma, tenga menor valor *fitness*) será el que formará parte de la población que pasará a la siguiente iteración. Por último, una vez agotadas el número de llamadas a la función objetivo que se pueden realizar, el algoritmo finalizará y devolverá la población resultante de la última iteración.

5.3. Modelo de Islas Clásico

Algoritmo para el Modelo de Islas

En primer lugar vamos a definir un vector de poblaciones, porque queremos almacenar las poblaciones de forma individual para cada uno de los algoritmos involucrados, por tanto definimos un vector de poblaciones de tamaño 4 (una para cada algoritmo). Todas las poblaciones inicialmente son la misma, aunque a lo largo de las ejecuciones cada una evolucionará en función del comportamiento de su algoritmo asociado.

A partir de las poblaciones asociadas, lanzamos todos los algoritmos inicializando la población a la correspondiente de cada algoritmo, y guardamos sus mejores resultados.

El siguiente paso a realizar sería la migración de las mejores soluciones de unos algoritmos a otros. Para ello, en cada iteración, definimos una permutación aleatoria que va a determinar el orden en el que va a migrar el mejor individuo de cada algoritmo a otro diferente. Para la determinación de la permutación, se ha hecho uso del método `calcular_permutacion()`.

Una vez que se ha llevado a cabo la migración, se pasa a una nueva iteración, donde se repetirá el proceso anterior, pero esta vez partiendo de la población de la iteración anterior como si de la población inicial del algoritmo se tratara.

Cálculo de la permutación

Tal y como se ha mencionado en el apartado anterior, se ha hecho uso de una función que nos permita obtener una permutación de un tamaño concreto.

Para ello, se ha realizado un método que tiene como parámetro una referencia a un vector vacío del tamaño que se quiere la permutación. Es sobre dicha variable donde se obtiene la permutación.

Para el cálculo de la permutación, vamos a generar, de manera aleatoria, valores enteros que se encuentren en el rango (0,`tam`), donde `tam` es el tamaño del vector (o de la permutación). Hay que tener en cuenta, que en una permutación no hay valores repetidos, por lo que para asignar un valor, debemos asegurarnos de que éste aún no se encuentra en la permutación que estamos generando.

Por último, mencionar que tal y como se ha expuesto en secciones anteriores, este modelo se implementa para varias combinaciones de algoritmos genéticos distintas. Por ello, se ha controlado en el código, diferentes opciones mediante una variable,

para que dependiendo del valor de la misma, se inicialicen los algoritmos `ag01`,`ag02`, `ag03` y `ag04` en función de la versión del Modelo de Islas que se quiera implementar.

5.4. Modelo Híbrido: Ejecución en 2 Fases

Algoritmo para el modelo

En este caso, lanzaremos cada uno de los algoritmos en cuestión con un número de iteraciones igual al 5 % de las totales. Una vez completadas, evaluaremos cuál, de entre todos los algoritmos, ha obtenido el mejor valor de *Fitness* hasta el momento, y el algoritmo correspondiente se ejecutará el resto de iteraciones que faltan hasta el máximo global establecido.

Para ello, en primer lugar volvemos a partir de un vector de poblaciones de tamaño 4, ya que tenemos en cuenta 4 algoritmos. Con ello, podremos almacenar de forma más clara las poblaciones resultantes de la ejecución de cada uno de los algoritmos.

A continuación procedemos a una primera ejecución de todos los algoritmos, donde cada uno va a consumir un número concreto de las evaluaciones (de la función objetivo) totales disponibles para el modelo. Guardaremos el valor *Fitness* del individuo más prometedor, para posteriormente compararlo con el obtenido por el resto de algoritmos. También almacenaremos su población, para poder partir de ella en caso de que volviera a lanzarse dicho el modelo.

Una vez ejecutados todos los algoritmos, se determina cuál es el algoritmo con el individuo más prometedor comprobando los mejores valores de Fitness obtenidos, y el que cumpla con esa condición será el que se ejecute el 80 % del tiempo restante.

5.5. Modelo Híbrido: Probabilidades Adaptativas

Este tercer modelo vuelve a tener una implementación similar a las anteriores, con la diferencia de que no controlaremos de forma exacta qué algoritmo se va a ejecutar en cada iteración del modelo, sino que esta decisión vendrá dada en función de un valor de probabilidad.

Para mayor sencillez, guardaremos estos valores en un vector, donde cada probabilidad i guardada es la que le corresponde al algoritmo iésimo utilizado. Por tanto, inicialmente partiríamos de un vector como el siguiente.

Esta tarea se lleva a cabo mediante una función implementada, la cuál se ha denominado `acumularProbabilidad()`, y se explicará posteriormente. De esta forma, el vector de probabilidades acumuladas correspondiente al ejemplo anterior sería el siguiente.

$$[0.25,\ 0.5,\ 0.75,\ 1]$$

Esta tarea la llevamos a cabo mediante la función `actualizarProbabilidad()`. Como es de esperar, cada vez que actualicemos la probabilidad, actualizaremos consecuentemente las probabilidades acumuladas.

Este proceso se continuará repitiendo hasta que se agoten el número de iteraciones especificadas por el algoritmo.

Vector de probabilidades acumuladas

Como hemos comentado previamente, se ha implementado un método que nos permite obtener el vector de probabilidad acumulada correspondiente a un vector de probabilidades. Para ello, se ha implementado un sencillo método mediante el cuál recorremos el vector acumulando al valor actual, los valores de los elementos previos.

Determinar el algoritmo a ejecutar

Como se ha comentado, en cada iteración se obtiene un valor aleatorio, que determina el algoritmo que se debe ejecutar en función de dicho valor. Por tanto este método recibe, por una parte el valor aleatorio generado, y por otra parte el vector de probabilidades acumuladas.

En el procedimiento, se va recorriendo dicho vector, hasta dar con la posición del mismo que contenga un valor superior al valor aleatorio. Será en este momento cuando hayamos encontrado el rango que lo contiene, y la posición actual se corresponderá con el número de algoritmo a ejecutar.

Actualizar el vector de probabilidades

Como ya hemos visto, cada cierto número de iteraciones es necesario actualizar el vector de probabilidades en función de la mejora obtenida por los algoritmos. Para ello, se ha implementado un método al que hemos llamado `actualizarProbabilidad()`.

Este algoritmo se encarga de obtener estos valores, los cuales se calculan como una proporción, a partir de la mejora media de cada algoritmo respecto del total de mejora conseguida. Además, tenemos en cuenta los casos donde los valores son cero, para evitar valores no válidos.

Obtener el mejor valor de *Fitness*

Para poder conocer el valor de mejora de cada algoritmo, necesitamos saber cuál el mejor valor de la población de partida. Para ello, se ha implementado la función `get_best_fitness`, que va recorriendo todos los elementos de una población almacenando el mejor resultado de *fitness* alcanzado por el momento.

Al igual que los modelos anteriores, este modelo se implementa para varias combinaciones de algoritmos genéticos, por lo que se ha controlado en el código, diferentes opciones mediante una variable, para que dependiendo del valor de la misma, se inicialicen los algoritmos `ag01,ag02, ag03 y ag04` en función de la versión a llevar a cabo.

Análisis y resultados

RESUMEN: En esta sección se realizará un análisis completo de los algoritmos mencionados en secciones anteriores. Realizaremos en ella, diferentes comparaciones para observar el comportamiento de unos algoritmos contra otros, y así sacar conclusiones acerca del funcionamiento de los mismos.

6. Análisis y resultados

Nuestro objetivo principal en este capítulo, es observar el comportamiento de cada uno de los modelos híbridos implementados, para así entender su funcionamiento y cómo se comporta cada uno respecto a los algoritmos que lo componen. Por tanto, es de esperar que se obtengan una gran cantidad de posibilidades a comparar.

En nuestro caso, vamos a comparar cada modelo de isla ejecutado, con los resultados que dan los distintos algoritmos que lo forman, de manera individual. Debido a la gran cantidad de funciones que se ejecutan (en total 30), vamos a utilizar tablas que sinteticen los resultados que hemos obtenido, y que,de esta forma, la información sea más sencilla de comprender.

Tabla 7: Algoritmos y parámetros utilizados en el estudio

	Versión del Modelo	**Versión Algoritmo Genético**	**Tamaño Población**	**Operador Selección**	**Operador Cruce (alfa)**	**Operador Reemplazo**
Modelo Híbrido	**Versión 1**	Generacional	60	Torneo	0.1	RW
		Generacional	60	Torneo	0.3	RW
		Generacional	60	Torneo	0.5	RW
		Generacional	60	Torneo	0.8	RW
	Versión 2	Estacionario	60	Torneo	0.5	RW
		Estacionario	60	NAM	0.5	DC
		Estacionario	60	Torneo	0.5	RW
		Estacionario	60	NAM	0.5	DC

En la tabla 7, podemos ver una tabla que resume las implementaciones finalmente llevadas a cabo, con los parámetros seleccionados para ello. Por generalizar, hemos puesto las dos versiones fijas que se van a utilizar. La tercera versión, estará formada por los dos algoritmos que resulten mejor de cada una de las versiones previas. En

el caso del Modelo de Islas, también se llevarán a cabo los mismos modelos para poblaciones de 60 y 240 individuos.

Los resultados que muestran a lo largo de la memoria, son resultado de realizar varias ejecuciones distintas al proyecto, ya que, como se utilizan valores aleatorios, de esta forma podemos obtener valores medios representativos y precisos. En total, los resultados para Dimensión 10, son resultados medios de 50 ejecuciones distintas a cada uno de los algoritmos y modelos. Para los resultados de problemas con Dimensión 30, se llevan a cabo 20 ejecuciones de cada técnica. Esto se debe al gran aumento de la complejidad en tiempo de procesamiento que conlleva aumentar a tal nivel de dimensión.

Todas las tablas que hemos utilizado, mantienen el mismo formato. Cada columna se corresponde con un algoritmo, y cada fila se corresponde con un caso (función) ejecutado. De esta forma, tendremos para cada algoritmo, su resultado medio alcanzado para cada una de las funciones.

En nuestro caso, buscamos conocer qué algoritmo es el que se comporta de una forma más favorable para las distintas funciones. En otras palabras, buscamos aquellos casos donde, el valor de *Fitness* obtenido sea más bajo que el resto. Debido a la gran cantidad de datos que aparecen en las distintas tablas, es difícil a priori realizar una comparación. Por ello, para mayor visibilidad, en cada tabla hemos resaltado en color rosa el mejor resultado obtenido para cada una de las funciones (y así poder localizarlo rápidamente).

Por último, se han añadido las dos últimas filas, las cuáles resumen la información proporcionada:

1 Hemos utilizado una fila `Recuento`, la cuál contiene, la cuenta del número de veces que cada algoritmo ha obtenido el mejor resultado respecto al resto en las 30 funciones utilizadas.

   Para ello, simplemente se ha realizado la suma de las casillas sombreadas que corresponde a cada uno de los algoritmos. De esta forma, podremos saber de forma rápida, cuál de ellos ha resultado ser el mejor un mayor número de veces.

2 Sin embargo, el recuento del número de veces que cada una de las técnicas ha resultado ganadora, puede no ser una medida muy representativa si dichos valores están muy empatados. Con ello queremos decir que, si un algoritmo resulta ser el mejor 26 veces, y los cuatro algoritmos restantes solo una vez cada uno, es obvio cuál de ellos ha sido el que mejor resultado muestra. Pero esta conclusión puede no ser tan obvia en casos más equilibrados. Para ello vamos a poner un ejemplo.

   "Imaginemos que tenemos los algoritmos A,B,C,D y E. El algoritmo A resulta ganador un total de 12 veces, mientras que en todas las demás ejecuciones proporciona el peor de los resultados. Por otra parte tenemos el algoritmo B, que ha dado el mejor resultado un total de 11 veces, proporcionando en todas las demás ejecuciones el segundo mejor valor de Fitness."

   Como podemos imaginar, en este caso no está tan claro el resultado, porque mientras que utilizando el criterio del recuento, concluiríamos que el mejor re-

sultante es el algoritmo A, claramente es el segundo de los algoritmos (B) el que tiene un mejor comportamiento global. Por tanto, hemos utilizado una medida de `Ranking`, que nos permita determinar con más exactitud la calidad de las distintas técnicas empleadas. La segunda de las filas adicionales, se corresponde con esta medida.

Esta medida, asigna una puntuación, a cada uno de los algoritmos para cada una de las funciones, de acuerdo a cómo de buena sea su solución respecto a los demás algoritmos. De esta forma, si un algoritmo alcanza para la función X el mejor valor *Fitness*, su puntuación para dicha función será de 1 punto. Si sus resultados son los segundos mejores, su puntuación sería de 2 puntos, y así sucesivamente. Por tanto, el algoritmo con un valor inferior de Ranking, será el mejor resultante. Finalmente, se sumarán los puntos que obtiene cada algoritmo y se obtendrá un valor medio por cada uno de ellos. Dicho valor, será el mostrado en la tabla, en la fila reservada para el Ranking.

Por otra parte, este criterio tiene una doble funcionalidad muy interesante en el tema que estamos tratando, y es que nos permite determinar la diferencia de calidad que existe entre dos técnicas. Para ello, vamos a verlo de nuevo con un ejemplo.

"Imaginemos que tenemos los algoritmos A, B y C. El algoritmo A obtiene en el Ranking un valor de 1.230, el algoritmo B consigue un total de 1.240, y el algoritmo C 3.000 puntos. Podemos concluir que tanto A como B son mejores que C, ya que hay una gran diferencia en el valor de Ranking que lo justifica. Sin embargo, no podemos afirmar que A es mejor que B, porque la pequeña diferencia que las distingue no es determinante. "

Por tanto, con el valor obtenido en el Ranking, podemos ayudarnos para saber cómo de robusta es nuestra decisión a la hora de determinar que un algoritmo es mejor o peor que otro.

Por último, hemos dividido las funciones en subgrupos, como se pueden observar en las tablas. Tenemos por una parte funciones unimodales y multimodales, y por otra parte distintas combinaciones: funciones híbridas formadas a partir de la Función 1, y composición de funciones. Han sido añadidas porque, puede ser, que se planteen comportamientos similares en funciones de un mismo tipo, y de esta forma podremos apreciarlo visualmente.

6.1. Modelo de Islas

6.1.1. Estudio de los Algoritmos Genéticos Generacionales

Poblaciones de tamaño 60

Tabla 8: Comparativa Modelos de Islas *vs* AGG, Dimensión 10, Tamaño Población 60

	Función	ISLAS	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.8$
Funciones unimodales	**1**	6.5026e+05	2.6219e+07	1.9733e+07	1.3739e+07	1.7166e+06
	2	4.4754e+03	8.1579e+08	2.5251e+08	2.1115e+07	5.2363e+03
	3	1.6329e+03	2.0687e+04	2.0828e+04	1.3467e+04	1.7111e+04
Funciones Multimodales Simples	**4**	1.7514e+01	2.5996e+02	9.5283e+01	4.1253e+01	2.8846e+01
	5	2.0105e+01	2.0493e+01	2.0417e+01	2.0398e+01	2.0543e+01
	6	2.2033e+00	6.3108e+00	4.0014e+00	3.3448e+00	3.6400e+00
	7	5.4011e-02	3.9984e+01	8.0940e+00	3.7040e-01	7.4475e-02
	8	7.2696e+00	2.5320e+01	1.4671e+01	1.2338e+01	1.6869e+01
	9	9.8555e+00	3.3245e+01	2.2132e+01	1.6095e+01	2.4467e+01
	10	1.6907e+02	9.7422e+02	5.1843e+02	3.3073e+02	3.8424e+02
	11	5.5017e+02	1.2316e+03	9.6166e+02	7.7194e+02	1.0637e+03
	12	2.4918e-01	9.7918e-01	8.5978e-01	8.4969e-01	1.1069e+00
	13	1.6825e-01	1.5488e+00	3.6344e-01	2.4493e-01	2.7126e-01
	14	2.9645e-01	7.5064e+00	1.0728e+00	4.5416e-01	4.3870e-01
	15	1.0419e+00	3.5108e+01	7.7786e+00	2.2581e+00	1.5979e+00
	16	2.7782e+00	3.5554e+00	3.3121e+00	3.2780e+00	3.5895e+00
Híbridas Función 1	**17**	4.4819e+04	5.7440e+05	5.8414e+05	5.2481e+05	1.5050e+05
	18	2.3651e+03	1.7757e+04	1.1313e+05	1.6438e+04	1.0794e+04
	19	1.5083e+00	9.1000e+00	4.6540e+00	2.6449e+00	2.4317e+00
	20	1.2494e+03	2.3066e+04	6.5040e+03	6.7841e+03	1.0622e+04
	21	2.3658e+03	1.9674e+05	2.1169e+05	1.0274e+05	1.5027e+04
Composición de Funciones	**22**	2.1822e+01	1.4471e+02	1.2290e+02	7.7622e+01	6.1099e+01
	23	3.2945e+02	3.5255e+02	3.3771e+02	3.3118e+02	3.3041e+02
	24	1.2344e+02	1.8107e+02	1.6198e+02	1.3911e+02	1.3759e+02
	25	1.8034e+02	1.9995e+02	1.9757e+02	1.9939e+02	1.9927e+02
	26	1.0013e+02	1.0234e+02	1.0027e+02	1.0025e+02	1.0028e+02
	27	2.0665e+02	4.0829e+02	3.6480e+02	3.3075e+02	3.1702e+02
	28	4.0707e+02	1.0151e+03	6.3040e+02	4.8659e+02	4.2352e+02
	29	4.1250e+02	3.0855e+06	2.8811e+05	4.1881e+04	1.5624e+05
	30	8.5598e+02	1.0322e+04	3.6690e+03	1.4501e+03	8.0089e+02
	Recuento	**29**	**0**	**0**	**0**	**1**
	Ranking	**1.033**	**4.766**	**3.766**	**2.633**	**2.800**

Tabla 9: Comparativa Modelos de Islas *vs* AGG, Dimensión 30, Tamaño población 60

	Función	**ISLAS**	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.8$
Funciones unimodales	**1**	1.0861e+07	7.0858e+08	3.9686e+08	1.3579e+08	5.3139e+07
	2	1.3450e+04	3.1438e+10	1.2581e+10	1.4886e+09	1.3695e+09
	3	1.0403e+04	1.0312e+05	1.1226e+05	6.9242e+04	6.1670e+04
Funciones multimodales Simples	**4**	1.1080e+02	4.4602e+03	1.3222e+03	3.0729e+02	1.6931e+02
	5	2.0580e+01	2.1015e+01	2.0863e+01	2.0825e+01	2.1068e+01
	6	1.6482e+01	3.1775e+01	2.6539e+01	1.7874e+01	1.8268e+01
	7	1.0498e-02	3.4150e+02	1.2445e+02	1.9213e+01	6.1387e+00
	8	5.4164e+01	1.9882e+02	1.0496e+02	6.1901e+01	8.7885e+01
	9	8.0978e+01	2.2636e+02	1.4355e+02	9.1954e+01	1.7095e+02
	10	2.2620e+03	5.2923e+03	3.5439e+03	2.4715e+03	3.1230e+03
	11	3.4862e+03	6.5271e+03	5.0997e+03	4.5446e+03	5.5297e+03
	12	7.3543e-01	1.8463e+00	1.6757e+00	1.4926e+00	2.6792e+00
	13	3.3045e-01	4.9012e+00	3.0698e+00	4.8713e-01	7.8861e-01
	14	2.8790e-01	1.2928e+02	4.7509e+01	1.1406e+00	2.2999e+00
	15	1.2691e+01	4.0715e+04	1.4857e+04	2.6908e+02	6.3507e+04
	16	1.2349e+01	1.3385e+01	1.2896e+01	1.2813e+01	1.3488e+01
Híbridas Función 1	**17**	1.3011e+06	4.3504e+07	3.2020e+07	9.7566e+06	4.2351e+06
	18	2.8326e+03	6.0485e+08	1.2091e+08	4.8786e+06	6.8887e+03
	19	1.6123e+01	2.5172e+02	1.7235e+02	8.4923e+01	2.5122e+01
	20	1.7651e+04	2.2141e+05	1.3282e+05	6.3561e+04	5.0486e+04
	21	4.0684e+05	1.3408e+07	6.7215e+06	1.6784e+06	9.7467e+05
Composición de Funciones	**22**	4.1445e+02	1.2076e+03	7.9064e+02	5.7345e+02	6.7263e+02
	23	3.1524e+02	5.0559e+02	3.9989e+02	3.3350e+02	3.3037e+02
	24	2.3491e+02	2.7820e+02	2.6773e+02	2.4961e+02	2.5019e+02
	25	2.1120e+02	2.2843e+02	2.2852e+02	2.2382e+02	2.0986e+02
	26	1.0041e+02	1.8216e+02	1.4941e+02	1.1604e+02	1.1209e+02
	27	7.1579e+02	1.2427e+03	1.0208e+03	7.8389e+02	8.0668e+02
	28	9.8711e+02	6.2414e+03	3.3967e+03	1.4937e+03	1.1413e+03
	29	3.7113e+03	2.9792e+08	3.4274e+07	1.1281e+06	1.3776e+06
	30	4.6512e+03	1.4105e+06	3.2001e+05	3.2739e+04	1.3471e+04
	Recuento	**29**	**0**	**0**	**0**	**1**
	Ranking	**1.033**	**4.800**	**3.866**	**2.500**	**2.800**

En este caso, podemos observar dos tablas, para dos dimensiones diferentes. En ellas, podemos ver el algoritmo de un modelo de islas homogéneo, formado por cuatro algoritmos genéticos generacionales que se diferencian en los parámetros utilizados en su operador de cruce.

En la tabla 8, tendremos los resultados obtenidos para Dimensión 10. En ella, podemos observar cómo los resultados obtenidos por el modelo de Islas son realmente buenos, en comparación con los algoritmos que lo componen. Hay varias cuestiones que destacar.

En primer lugar, vamos a hacer una pequeña reflexión sobre los resultados individuales de los algoritmos que están involucrados en la hibridación. Como podemos ver, la diferencia entre los distintos algoritmos está basada en el parámetro del valor

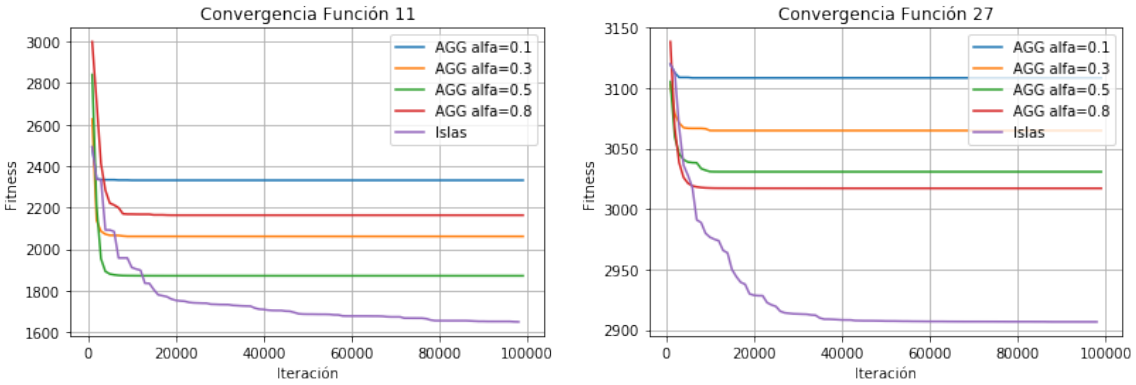
α. Estamos utilizando por tanto algoritmos genéticos cuyo cambio en el operador de cruce nos permite obtener una mayor o una menor diversidad en el cruce, y por tanto distintos resultados.

En el caso de utilizar un valor de $\alpha = 0.1$, estaremos limitando mucho la diversidad del algoritmo, ya que la exploración y el alcance de los valores es mínimo (son valores en un intervalo de valores muy cercano a los que ya tenemos). Ello conllevará, a que con este algoritmo, realicemos cruces que explotan en cantidad la zona del espacio de búsqueda en la que nos encontramos. El problema en este caso, es que si se da una situación en la que estemos en una zona muy poco prometedora del espacio de búsqueda o en una zona en la que haya un gran máximo local, debido a la cualidad explotadora mencionada será muy difícil escapar de dicha área, lo que dificulta que este algoritmo de realmente buenos resultados.

En el caso de utilizar un valor de $\alpha = 0.8$, nos encontraremos en la situación justo contraria. De esta forma, ampliamos mucho los posibles valores que puede tomar cada hijo, ya que en este caso, la zona en la cuál se explora será mucho mayor. Con ello, estamos aumentando en gran medida la diversidad en el cruce del algoritmo, y como cualquier límite, tampoco es deseable. Lo ideal será por tanto, utilizar un valor de α que combine ambos extremos, y como podemos ver en los resultados de la 8, utilizar un valor de $\alpha = 0.5$ proporciona los mejores resultados en comparación. Este hecho ratifica el comportamiento esperable de los algoritmos en función del valor de α que comentamos. Estas conclusiones, se pueden contrastar con un proyecto que justifica justamente el uso de 0.5 como parámetro ideal para α, de acuerdo a lo que hemos venido comentando [21].

En segundo lugar, es de esperar por tanto que a priori, **puede ser una buena idea combinar algoritmos potentes en soluciones caracterizadas por su diversidad, con algoritmos cuyas nuevas generaciones llevarán a una mayor explotación**, ya que como hemos visto, es una forma de intentar escapar de malas zonas o áreas del espacio de búsqueda, y tener más oportunidades para dar lugar a mejores resultados. Este hecho se ve reflejado en la tabla 8, que, tal y como se aprecia en la misma, el resultado que obtiene el modelo de islas respecto a los algoritmos utilizados es aplastante. Podemos contrastar este hecho tanto viendo los resultados del recuento, (donde vemos que ha sido el mejor modelo en comparación con todos los demás), como con los resultados del Ranking, donde se puede observar que los resultados alcanzados por este son superiores del doble de los que consigue el segundo algoritmo mejor en la clasificación.

Podemos visualizar este comportamiento en la imagen 7. En estas imágenes, vemos algunas de las gráficas de convergencia que se obtienen con el Modelo de Islas, en este caso para la Función 27 y la Función 11. Como podemos observar, mientras que los algoritmos genéticos se estancan, y no consiguen mejorar sus soluciones desde iteraciones relativamente tempranas, el Modelo de Islas sí que sigue convergiendo durante mayor parte de la ejecución. El hecho de que se retroalimente el siguiente algoritmo con soluciones de otro de los que forman el modelo, permite escapar de las zonas en las que el algoritmo se estanca, y así continuar mejorando.



(a) Función 11 (b) Función 27

Figura 7: Convergencia Modelos de Islas I

También podemos ver casos realmente destacables, como pueden ser los resultados para la primera función, donde la diferencia de *Fitness* alcanzada por el Modelo de Islas es muy superior a la alcanzada por los algoritmos individuales que se ejecutan. Es en este tipo de casos, donde se puede ver la potencia de este tipo de hibridaciones.

Como última referencia a los resultados de la tabla 8, vamos a hacer una especial mención a la función número 30, la cuál es la única donde el modelo de Islas no ha resultado el mejor parado. Respecto a este hecho, consideramos que no se puede sacar ningún tipo de conclusión, ya que la diferencia en el Valor de *Fitness* obtenida con el Modelo de Islas y con el mejor algoritmo para esa función es mínimo. Al tratarse de una diferencia tan pequeña, no existe base para pensar con firmeza que existe una razón concreta por la que en ese caso no ha funcionado. También podría haberse tratado de, una falta de precisión, o de una necesidad de mayor número de ejecuciones a los problemas para demostrarlo (en nuestro caso, hemos realizado 50 ejecuciones, pero quizá con 100 ese resultado hubiera salido en armonía con todos los demás).

En la imagen 8 podemos ver la gráfica de convergencia obtenida para la función que estamos comentando (Función 30). Como se puede ver, realmente el Modelo de Islas realmente tiene un buen comportamiento, ya que es capaz de mejorar de forma considerable en comparación con el algoritmo ganador en este caso. Por tanto, como decíamos, realmente no podemos concluir que se comporte de forma mala en esta situación.

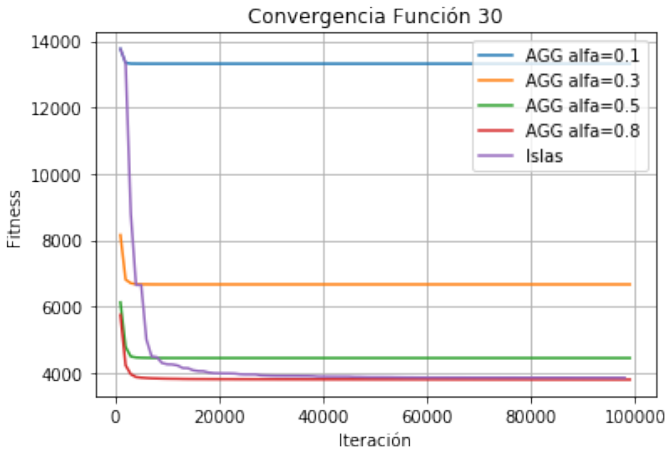


Figura 8: Convergencia de la Función Número 30

En el caso de la tabla 9, estamos tratando el mismo caso anterior, con la diferencia de que en este caso utilizamos soluciones de dimensión 30. De nuevo, podemos ver ratificadas todas las conclusiones anteriores, que se mantienen en la misma dirección que las alcanzadas para la dimensión 10 (tabla 8).

Podemos observar, cómo el Modelo de Islas sigue alcanzando los mejores resultados prácticamente en todas las soluciones, y con una superioridad bastante representativa, como se puede observar en los resultados del Ranking (donde la puntuación alcanzada para las Islas es menos de la mitad de la puntuación del segundo algoritmo mejor).

De nuevo, el algoritmo individual con el que se obtienen mejores resultados, es el que utiliza un valor de α que equilibra tanto la diversidad como la explotación de nuevas soluciones. Concretamente, podemos ver en esta tabla los resultados de una mejor forma, ya que la diferencia de resultados obtenidos con dicho algoritmo destaca en mayor cantidad.

Por último, mencionar que de nuevo no podemos sacar conclusiones respecto a la única función donde el Modelo de Islas no obtiene mejor resultado, puesto que reiteradamente, nos encontramos en una situación donde la diferencia entre dichos valores de *Fitness* es muy pequeña como para extraer de la misma conclusiones fiables.

Poblaciones de tamaño 240

Hemos visto en el apartado anterior, la superioridad que han resultado tener los Modelos de Islas respecto a los resultados que ofrecen cada uno de sus algoritmos de forma individual. Sin embargo, ahora nos surge la duda de si pueden estar los resultados ligados al aumento de diversidad que se ha incorporado. Con ello, nos referimos, a que los modelos de Islas están combinando los cuatro algoritmos individuales de poblaciones de tamaño 60, resultando una población global de 240 individuos en el modelo.

Este gran número de soluciones puede provocar, como hemos mencionado, un aumento de la diversidad, ya que al haber tal cantidad de soluciones, se facilita la tarea de alcanzar más cantidad de zonas del espacio de búsqueda, y aumentar la probabilidad de alcanzar mejores resultados. Por tanto, en esta sección vamos a comparar el resultado del Modelo, con el resultado de los individuos individuales con un tamaño de 240, y de esta forma equilibrar las poblaciones. Con ello, podremos determinar si existe una razón ligada a la diversidad, por la cuál anteriormente se han obtenido resultados tan destacados.

De nuevo vamos a contemplar dos dimensiones. En la tabla 10 podemos ver los resultados obtenidos para soluciones de dimensión 10, y en la tabla 11 para problemas de dimensión 30.

Si observamos la tabla 10, podemos ver cómo ya los resultados no son tan representativos y claros como sucedía en el caso anterior, donde se comparaba el Modelo de Islas a los algoritmos individuales de tamaño 60. Si observamos tanto el recuento como los resultados obtenidos en el Ranking, es obvio que **existe una relación entre los resultados alcanzados anteriormente con la diversidad y el tamaño de la población**.

Sin embargo, hay dos aspectos destacados y dignos de mención.

1. El Modelo de Islas no da los mejores resultados, pero sí los segundos mejores resultados. Si además observamos las diferencias en los valores de *Fitness* alcanzados, en la mayor parte de los casos no hay prácticamente diferencia en los valores, y sigue superando a tres de los cuatro algoritmos que involucra.

2. A pesar de no dar los mejores resultados globales, estos no se deben en su totalidad a la relación de divergencia.

   En primer lugar, porque los algoritmos individuales han seguido mejorando sus resultados con el aumento de las poblaciones al tamaño 240 , por lo que no se han visto atascados con este aumento de diversidad, sino que son aún con ese tamaño todavía competitivos. Al ser competitivos con un tamaño alto, en un principio también debería seguir mejorando el modelo de islas, y el hecho es, que en una cantidad alta de casos obtiene buenos resultados. Si observamos detenidamente los casos en los que destaca y los comparamos con los que no alcanzan mejores resultados, podemos extraer dos conclusiones:

   a) En general, los casos en los que obtiene mejores resultados son aquellos en los que todos los algoritmos de manera individual dan resultados similares, y al combinarse resultan en una mejoría. Por tanto, el modelo de Islas sigue siendo efectivo, ya que cuando los algoritmos son malos (o bien no son capaces de escapar de sus dificultades), no dan buenos resultados que destaquen, con el Modelo de Islas somos capaces de compensar los aspectos buenos de unos y otros y alcanzar así resultados competitivos.

   Por otra parte se puede dar otra situación, y es que cuando los algoritmos individuales se van comportando más o menos de forma similar, excepto uno de ellos, que es bastante malo en comparación con los demás. En estos casos, como el modelo de islas se ve influido por el comportamiento de todos los algoritmos, la mejora que se consigue por una parte, se pierde

con otra, ya que el algoritmo que es peor (de forma destacada) impide al modelo de islas evolucionar tan bien como pudiese. Este hecho podemos verlo en el caso de la Función 2, por ejemplo.

b) Es destacable el comportamiento en las funciones híbridas. Estas funciones, son aquellas cuyo espacio de búsqueda está formado por uniones de distintas zonas del espacio de la función 1. Al haber distintas zonas, unas serán laderas, otras serán más llanas, y otras representarán mayores dificultades, y es más fácil con las islas acercarse a ellas. En la imagen 9 podemos ver, de una forma visual, el tipo de función que estamos tratando.

Como ya sabemos, los modelos de islas nos permiten viajar de unas zonas a otras del espacio de búsqueda, por lo que este comportamiento ayudaría a escapar de los mínimos locales y zonas no prometedoras, y así poder dar buenos resultados.

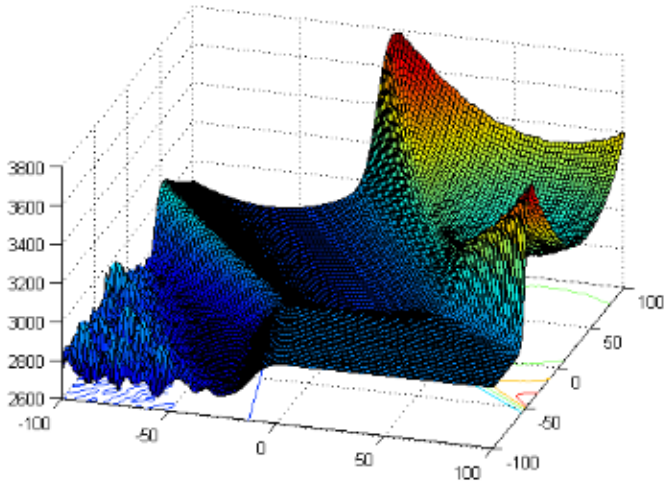


Figura 9: Representación de la Función Número 20

Este comportamiento, es el que podemos ver notablemente en las funciones híbridas, ya que como podemos observar, en una cantidad de estas funciones, es el modelo híbrido el que obtiene mejores resultados, y en los casos en los que no es así, la diferencia es muy pequeña, por lo que tampoco se puede concluir nada.

Tabla 10: Comparativa Modelos Islas *vs* AGG, Dimensión 10, Tamaño Población 240

	Función	**ISLAS**	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.8$
Funciones unimodales	**1**	6.5026e+05	1.5061e+07	9.0084e+06	2.1733e+06	2.2745e+05
	2	4.4754e+03	7.0289e+07	2.2971e+04	1.3810e+03	5.5215e+03
	3	1.6329e+03	1.0493e+04	4.2353e+03	2.2691e+03	5.5495e+03
Funciones Multimodales Simples	**4**	1.7514e+01	8.7790e+01	4.6495e+01	3.0481e+01	3.3704e+01
	5	2.0105e+01	2.0132e+01	1.9612e+01	1.8600e+01	2.0308e+01
	6	2.2033e+00	3.9352e+00	9.7020e-01	1.6552e-01	1.3404e+00
	7	5.4011e-02	1.1171e+01	9.5133e-02	3.9416e-03	5.5262e-02
	8	7.2696e+00	1.2547e+01	4.3653e+00	3.8604e+00	1.0317e+01
	9	9.8555e+00	1.3139e+01	5.2764e+00	3.5049e+00	2.0699e+01
	10	1.6907e+02	4.8904e+02	1.3549e+02	7.2378e+01	1.1952e+02
	11	5.5017e+02	8.0281e+02	2.5610e+02	8.8974e+01	7.0033e+02
	12	2.4918e-01	3.6477e-01	8.2700e-02	7.9644e-01	4.0755e-01
	13	1.6825e-01	2.4730e-01	6.8969e-02	3.2182e-02	1.6965e-01
	14	2.9645e-01	7.8302e-01	4.0135e-01	3.8735e-01	3.4250e-01
	15	1.0419e+00	1.7742e+00	1.0977e+00	1.0099e+00	2.1946e+00
	16	2.7782e+00	3.3550e+00	2.7572e+00	2.0125e+00	3.3862e+00
Híbridas Función 1	**17**	4.4819e+04	2.7499e+05	2.0781e+05	1.0340e+05	8.2781e+04
	18	2.3651e+03	8.1286e+03	7.6514e+03	7.8739e+03	8.9739e+03
	19	1.5083e+00	3.8970e+00	1.6408e+00	6.1893e-01	1.2340e+00
	20	1.2494e+03	3.8788e+03	4.3087e+03	3.7767e+03	4.7813e+03
	21	2.3658e+03	1.9625e+04	7.7430e+03	6.1609e+03	6.3551e+03
Composición de funciones	**22**	2.1822e+01	1.1880e+02	5.8198e+01	1.3068e+01	1.1712e+01
	23	3.2945e+02	3.3309e+02	3.2989e+02	3.2945e+02	3.2945e+02
	24	1.2344e+02	1.5946e+02	1.3237e+02	1.1338e+02	1.2598e+02
	25	1.8034e+02	1.9815e+02	1.9751e+02	1.9479e+02	1.9026e+02
	26	1.0013e+02	1.0012e+02	1.0005e+02	1.0003e+02	1.0014e+02
	27	2.0665e+02	3.3048e+02	2.6627e+02	2.6309e+02	2.9265e+02
	28	4.0707e+02	6.6860e+02	4.6920e+02	3.7649e+02	3.8118e+02
	29	4.1250e+02	5.0252e+05	4.8687e+02	4.6958e+02	9.5652e+02
	30	8.5598e+02	2.6904e+03	1.5096e+03	7.0842e+02	5.2901e+02
	Recuento	**12**	**0**	**1**	**14**	**5**
	Ranking	**2.200**	**4.633**	**3.133**	**1.766**	**3.266**

A continuación, vamos a ver qué cambios percibimos con el cambio de dimensión, y cuáles de nuestras conclusiones se mantienen con dicho cambio. En la tabla 11, podemos ver los resultados obtenidos para los problemas de dimensión 30 y los algoritmos individuales con un tamaño de población 240, de nuevo para compensar el aumento de diversidad que supone mantener un modelo de islas con subproblemas de 60 individuos.

Como podemos observar en un primer lugar, de nuevo ocurre que el modelo parece no dar tan buenos resultados cuando igualamos los tamaños de poblaciones. En este caso concreto, podemos verlo mucho más destacado en el anterior, ya que la diferencia en los recuentos es inmensa. Podemos ver que en una gran cantidad de casos, la

diferencia entre los valores medios de *Fitness* es muy pequeña (véase funciones 3,5,8 o 9 como ejemplo). También ocurre esta diferencia al revés, donde el modelo de islas gana con muy poca diferencia (como puede ser el caso de la `Función 3`). Sin embargo, nos volvemos a encontrar con la misma situación anterior, y es que hay muchos casos, donde un algoritmo es malo, (o más de uno), y, aunque haya resultados buenos, estos se vean perjudicados al implementar una hibridación. Como ya hemos comentado, esto se debe a que al fin y al cabo, el modelo de islas combina las soluciones de todos, resultando al final una fusión de los comportamientos individuales. Si hay comportamientos malos, estos repercutirán en los resultados negativamente, pero a su vez, un comportamiento muy bueno de un algoritmo puede verse afectado por los malos comportamientos de otros, y no dar buenos resultados globales. Por ejemplo, fijémonos en la `Función 11`. Los algoritmos que incluyen parámetros de $\alpha = 0.1$,$\alpha = 0.3$ y $\alpha = 0.8$ son malos en comparación con el resultado para un $\alpha = 0.5$. Es de esperar por tanto, que el valor resultante del modelo híbrido sea un valor que se encuentre entre el mejor encontrado y los resultados de los peores, y como podemos observar, efectivamente eso es lo que ocurre. Este mismo hecho se repite en muchas otras situaciones; véase por ejemplo la `Función 10`. Por tanto, **en casos donde un algoritmo sea realmente bueno para un problema, su combinación en un modelo de islas no tiene por qué ser prometedora también, sino más bien al contrario, ya que el resultado final tenderá más a equipararse a un resultado *medio* de entre los resultados obtenidos de forma individual**.

Al fin y al cabo, este hecho dependerá también mucho de la función. Sin embargo, tal y como vemos en la tabla 11, en los resultados para las funciones compuestas sí que destaca el Modelo de Islas, ya que en 7 de los 9 casos, dicho modelo obtiene el mejor resultado, y en los dos restantes la diferencia es tan pequeña que no tiene razón de justificar que el modelo no es bueno para ese caso. Destacamos dos comportamientos esenciales en este bloque, ambos positivos:

1. En el caso de problemas como la `Función 23`, `Función 24`, `Función 26` o `Función 28`, el resultado obtenido en los distintos algoritmos genéticos es muy similar. Con el modelo de islas, se ven retroalimentados, y se permite llegar a un mejor resultado, ya que existe una cooperación entre los mismos, y el hecho de estar tan equiparados unos a otros no resulta perjudicial, al contrario que ocurría en otros ejemplos anteriormente mencionados.

2. En el caso de la `Función 29` o la `Función 30`, vemos el resultado justo contrario, donde todos los resultados son muy distintos entre sí. En este caso, se da otra de las situaciones muy favorecedoras para que un Modelo de Islas de buenos resultados, ya que existe una **sinergia** entre los distintos algoritmos, ya que dan resultados muy diferentes entre sí. El algoritmo genético con $\alpha = 0.1$ explota demasiado, mientras que al algoritmos genético con $\alpha = 0.8$ le ocurre lo contrario, dando así resultados dispares y resultando la sinergia ya comentada anteriormente.

Tabla 11: Comparativa Modelos de Islas *vs* AGG, Dimensión 30, Tamaño Población 240

	Función	**ISLAS**	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.8$
Funciones unimodales	**1**	1.0861e+07	3.5992e+08	1.0414e+08	1.8311e+06	1.6361e+07
	2	1.3450e+04	1.6230e+10	3.5869e+08	1.0291e+04	2.3184e+07
	3	1.0403e+04	7.5617e+04	2.6060e+04	1.1989e+04	2.6751e+04
Funciones multimodales Simples	**4**	1.1080e+02	1.8752e+03	2.2156e+02	7.5882e+01	1.2273e+02
	5	2.0580e+01	2.0628e+01	2.0182e+01	2.0945e+01	2.0883e+01
	6	1.6482e+01	2.5714e+01	1.1550e+01	2.7550e+00	8.3898e+00
	7	1.0498e-02	1.5538e+02	6.4480e+00	3.6926e-04	9.1030e-01
	8	5.4164e+01	1.1774e+02	3.0608e+01	1.5819e+01	1.3213e+02
	9	8.0978e+01	1.4570e+02	3.9830e+01	1.8569e+01	2.0063e+02
	10	2.2620e+03	3.8433e+03	1.1016e+03	3.3154e+02	1.9729e+03
	11	3.4862e+03	5.0477e+03	2.7156e+03	7.2590e+02	4.1644e+03
	12	7.3543e-01	6.3974e-01	1.3952e-01	1.3880e+00	1.6689e+00
	13	3.3045e-01	3.5334e+00	2.4683e-01	1.2183e-01	6.1840e-01
	14	2.8790e-01	6.3526e+01	2.3977e-01	3.9427e-01	8.8237e-01
	15	1.2691e+01	1.0553e+03	1.5402e+01	3.1837e+00	3.2170e+01
	16	1.2349e+01	1.2829e+01	1.1685e+01	1.1702e+01	1.3026e+01
Híbridas Función 1	**17**	1.3011e+06	2.0916e+07	6.5968e+06	6.4173e+05	9.8933e+05
	18	2.8326e+03	9.8094e+07	8.7953e+02	1.4282e+03	3.4471e+03
	19	1.6123e+01	1.9328e+02	6.8026e+01	8.6339e+00	8.3061e+00
	20	1.7651e+04	5.7003e+04	2.7414e+04	2.0329e+04	4.6772e+04
	21	4.0684e+05	5.3755e+06	7.6873e+05	2.8865e+05	5.2825e+05
Composición de Funciones	**22**	4.1445e+02	7.1757e+02	4.5780e+02	1.9404e+02	2.4715e+02
	23	3.1524e+02	5.0559e+02	3.9989e+02	3.3350e+02	3.3037e+02
	24	2.3491e+02	2.7820e+02	2.6773e+02	2.4961e+02	2.5019e+02
	25	2.1120e+02	2.2843e+02	2.2852e+02	2.2382e+02	2.0986e+02
	26	1.0041e+02	1.8216e+02	1.4941e+02	1.1604e+02	1.1209e+02
	27	7.1579e+02	1.2427e+03	1.0208e+03	7.8389e+02	8.0668e+02
	28	9.8711e+02	6.2414e+03	3.3967e+03	1.4937e+03	1.1413e+03
	29	3.7113e+03	2.9792e+08	3.4274e+07	1.1281e+06	1.3776e+06
	30	4.6512e+03	1.4105e+06	3.2001e+05	3.2739e+04	1.3471e+04
	Recuento	**9**	**0**	**5**	**14**	**2**
	Ranking	**2.550**	**4.700**	**2.966**	**1.550**	**3.230**

Por último, vamos a estudiar un poco el funcionamiento interno que se da en los modelos de islas para las dos dimensiones mencionadas. A continuación podemos observar gráficamente en la imagen 10 y en la imagen 11, cómo se han comportado,para dimensiones 10 y 30 respectivamente, el Modelo de Islas para cada uno de los algoritmos por cada función.

Se ve, de forma sintetizada, la proporción de veces donde la participación de los distintos algoritmos que componen el modelo ha llevado a obtener los mejores resultados. En otras palabras, estamos representando de forma gráfica, qué algoritmos consiguen más mejorías durante la ejecución del Modelo de Islas.

Como podemos observar, la retroalimentación que se produce de unos algoritmos a otros, es una de las razones que hacen más potentes a los algoritmos que componen

el modelo. De esta forma podemos ver, cómo el AGG con $\alpha = 0.8$, que no es el mejor de los algoritmos individuales, es el que obtiene mejores resultados internamente, probablemente debido a que la combinación de soluciones más explotadoras con la mayor exploración que proporciona el mismo, permite llegar a zonas más prometedoras. De igual forma, es importante destacar, que aquel que más veces da mejores resultados no tiene por qué ser el mejor algoritmo para ese problema concreto, puesto que éste puede ser el mejor por mejoras muy pequeñas, y otro algoritmo podría realizar de una sola vez una mejora cuantitativa (no quedando reflejado en la figura).

Sin embargo, este no es el aspecto más importante a destacar en las imágenes mostradas, y es que, como se puede observar, hay casos en los que, para determinadas funciones, alguno de los algoritmos no consigue producir ningún tipo de mejoría durante la ejecución, lo cuál se puede deber a que es un algoritmo que en la combinación híbrida realizada, no resulta beneficioso. En otras palabras, podemos apreciar cómo hay algoritmos que no están contribuyendo al modelo híbrido de forma positiva, por lo que en su lugar, podrían estar entorpeciendo el desarrollo del mismo. Véase por ejemplo, la `Función 1`, `Función 2` y `Función 4` para dimensión 30 (imagen 11), donde como se puede ver en sus resultados de la tabla 11, efectivamente no dan buenos resultados en la hibridación.

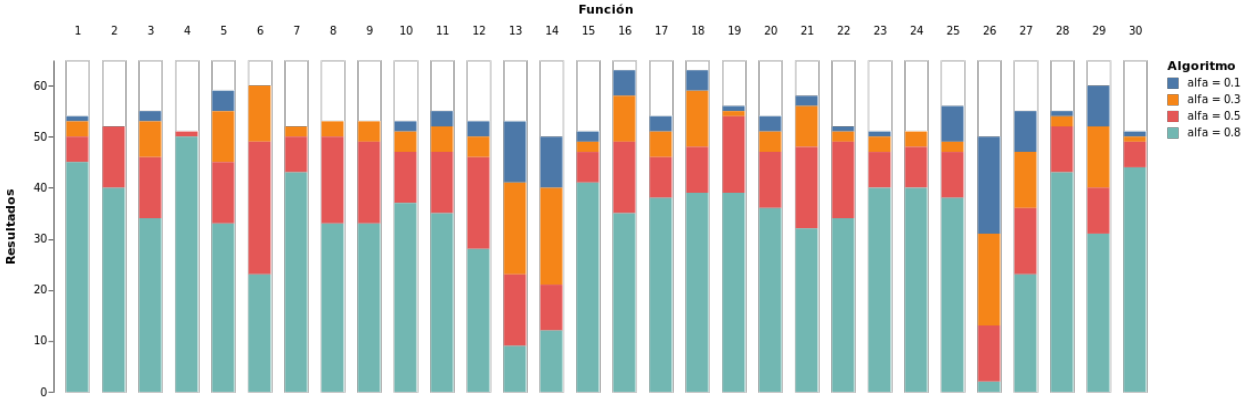


Figura 10: Estudio del comportamiento del Modelo de Islas con AGG, Dim=10

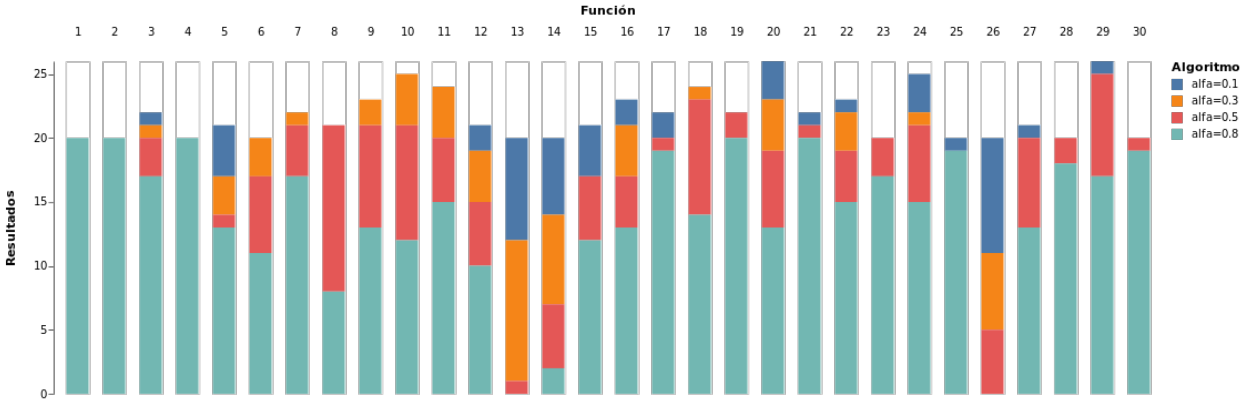


Figura 11: Estudio del comportamiento del Modelo de Islas con AGG, Dim=30

6.1.2. Estudio de los Algoritmos Genéticos Estacionarios

Ahora vamos a estudiar el comportamiento del Modelo de Islas para otra combinación homogénea: algoritmos genéticos generacionales. En este caso, no realizaremos cambios en el operador de cruce, sino que utilizaremos distintas combinaciones de operadores de selección y reemplazo en algoritmos estacionarios.

De nuevo vamos a proceder de la misma forma que antes. En primer lugar, vamos a estudiar las poblaciones de tamaño 60 para ambas dimensiones, y posteriormente las poblaciones de tamaño 240. De esta forma, veremos si las conclusiones que hemos sacado anteriormente acerca del Modelo de Islas son contradictorias, o si por el contrario, nos reafirmamos en nuestras conclusiones.

Antes de comenzar a analizar los resultados obtenidos, vamos a repasar un poco las características de los algoritmos que estamos involucrando, ya que son determinantes en los resultados.

1. En primer lugar hablaremos del algoritmo genético estacionario que utiliza el operador de selección *Torneo Binario*, combinado con el operador de reemplazo de *Sustituir Peor*. Este algoritmo, al que hemos denominado como `TB-RW` en las tablas, es altamente explotador, con lo cuál, como mencionábamos anteriormente, no obtendrá en general buenos resultados, puesto que debido a este exceso de explotación, rápidamente caerá en mínimos locales o en zonas no prometedoras de las que el algoritmo no sea capaz de salir. Sin embargo, por esta misma característica de exceso de explotación, es una de las razones por la que es interesante su combinación en un modelo híbrido.
2. En segundo lugar, mencionar el algoritmo genético estacionario que combina el método de selección de *Emparejamiento Variado Inverso* con el operador de reemplazo conocido como *Crowding Determinístico* (`NAM-DC` en las tablas). En este caso, ocurre justamente lo contrario, puesto que sus operadores se caracterizan por tener una componente alta en cuanto a exploración. Esta característica, hace a este algoritmo muy interesante a la hora de combinarlo en un modelo híbrido, al igual que ocurría en el caso anterior.
3. Los otros dos algoritmos estacionarios utilizados (`NAM-RW` y `TB-DC`), son combinaciones menos extremas en cuanto a exploración y explotación, las cuáles pueden dar buenos resultados de forma individual, aunque a continuación veremos cómo se comportan dentro de un modelo híbrido.

Poblaciones de tamaño 60

En la tabla 12, podemos encontrar los resultados que hemos recopilado acerca del Modelo de Islas con algoritmos estacionarios, y también de los resultados de dichos algoritmos genéticos ejecutados de forma separada al modelo.

A diferencia de lo que sucedía con la tabla 8, en este caso no es tan exagerada la superioridad del Modelo de Islas en comparación con los algoritmos estacionarios. A pesar de ello, el Modelo de Islas sigue obteniendo resultados muy competitivos y superiores al resto de los algoritmos. Si consultamos los resultados del Ranking, podemos ver cómo el resultado del mismo se acerca a ser el doble del siguiente mejor.

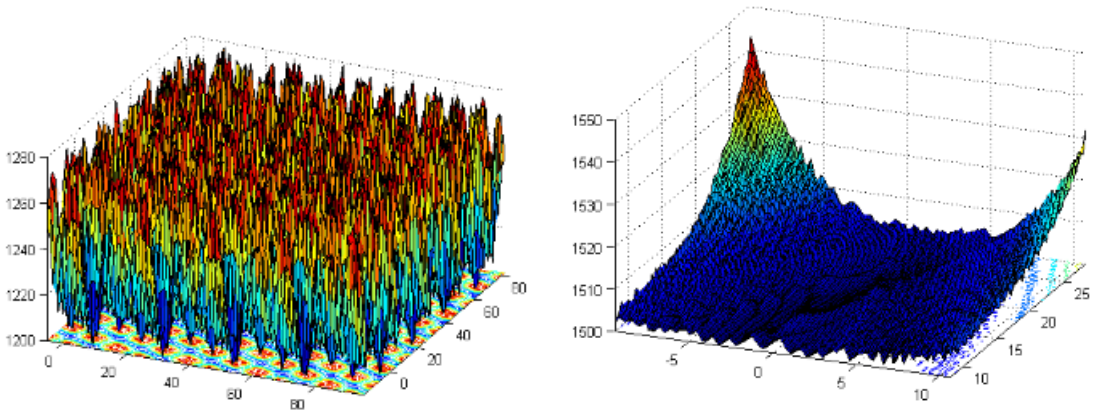
Hay algunos aspectos que podemos destacar de estos resultados.

1. En primer lugar, como hemos comentado, el Modelo de Islas obtiene claramente los mejores resultados de forma global, ya que es el que mejor resultado obtiene en el 70 % de las funciones. Sin embargo, como vimos anteriormente, estamos ejecutando subproblemas de tamaño 60, y comparando posteriormente con algoritmos que tienen el mismo tamaño de población, por lo que los resultados pueden estar ligados a este aumento de población global, que como consecuencia, aumenta la diversidad de soluciones que se pueden dar. Comprobaremos esta cuestión en la siguiente sección.

2. En segundo lugar, hay que mencionar que los algoritmos estacionarios, en este proyecto, obtienen mejores resultados que los algoritmos genéticos. En este sentido, por tanto, son mas potentes de forma individual, por lo que es lógico que los Modelos de Islas se encuentren con una mayor dificultad a la hora de superar los valores de *Fitness* que alcanzan éstos. Además, en este caso estamos combinando algoritmos más diferentes, ya que no cambian en un sólo parámetro de un operador, sino que de por sí, los algoritmos son mucho más distintos, por lo que los resultados son más distantes entre sí. Como comentábamos, los Modelos de Islas realizan combinaciones tanto para bien como para mal, por lo que al final, todos los algoritmos aportan algo, y si las soluciones son muy diferentes, las peores influirán en mayor cantidad a los resultados que devuelva el Modelo de Islas.

3. En tercer lugar, también podemos ver cómo el algoritmo estacionario `NAM-RW` no obtiene buenos resultados, debido a su rapidísima convergencia. Este algoritmo, se verá beneficiado sobretodo cuando tengamos soluciones malas con hijos muy buenos, situación que no tiene por qué darse con regularidad, y que se ve reflejado en los malos resultados que se obtienen. Estas malas soluciones, también influyen en los resultados del Modelo de Islas.

4. Por otra parte, podemos observar que el algoritmo estacionario NAM-DC obtiene realmente buenos resultados en varias de las funciones que estamos contemplando. En concreto, podemos ver que entre las funciones desde la 12 hasta la 16, es realmente competitivo (cuando directamente no es ganador), y es de interés conocer por qué funciona tan bien en estos casos un algoritmo que se caracteriza por añadir mucha diversidad. Para entender la razón, tenemos que conocer primero la representación gráfica de dichas funciones y así poder ver cómo es el espacio de búsqueda de las mismas. En la selección de imágenes 12 podemos ver cómo son algunas de las funciones a las que nos estamos refiriendo; concretamente a la función 12 y a la función 15.



(a) Función 12 (b) Función 15

Figura 12: Funciones 12 y 15

Si buscamos un punto en común entre estas funciones, podemos sacar como conclusión que son funciones que tienen una gran cantidad de máximos y mínimos pronunciados en mayor o en menor medida. Este tipo de funciones, se caracterizan por necesitar algoritmos que diversifiquen en cantidad para poder obtener buenas soluciones, ya que es muy fácil caer en mínimos locales, pero muy difícil salir de ellos. Como consecuencia, los algoritmos fuertes en una componente de diversidad, serán especialmente adecuados para este tipo de casos, tal y como podemos ver en la gráfica de convergencia para la función 12, que se muestra en la figura **??**, donde se puede ver cómo ésta no llega a converger. De nuevo tenemos aquí otro caso en el que el tipo de función con el que estamos tratando es determinante a la hora de obtener buenos resultados con Modelos de Islas, ya que estas funciones en concreto funcionan realmente bien con un tipo de funciones, y el incorporarle otros componentes que no las benefician (como ocurre con el Modelo de Islas), no sería beneficioso en cuanto a resultados. **Por tanto nos reiteramos en la idea de que cuando un algoritmo funciona realmente bien de forma individual respecto al resto de algoritmos, incorporarlo en un Modelo de Islas no nos lleva a mejorar la situación.**

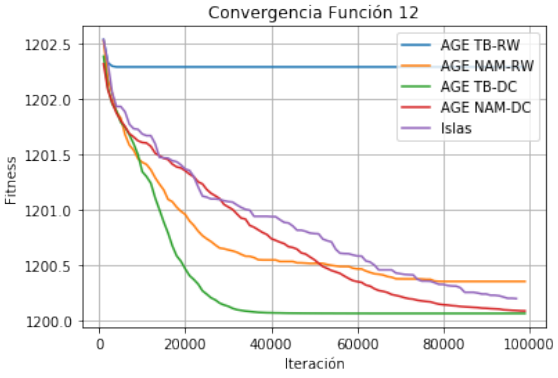


Figura 13: Convergencia de la Función 12 para AGE, Dim=10

Tabla 12: Comparativa ISLAS *vs* AGE, Dimensión 10, Tamaño Población 60

	Función	ISLAS	TB-RW	NAM-RW	TB-DC	NAM-DC
Funciones unimodales	**1**	1.2687e+05	1.5062e+07	4.5445e+06	9.1373e+04	1.2852e+05
	2	1.4547e+03	2.2385e+07	2.3344e+03	3.2162e+03	3.7183e+03
	3	9.8373e+02	1.3982e+04	5.1444e+03	2.9106e+03	2.2521e+03
Funciones Multimodales simples	**4**	2.4256e+01	4.8196e+01	2.6471e+01	2.2436e+01	2.4316e+01
	5	1.8092e+01	2.0700e+01	1.8308e+01	1.9518e+01	1.8705e+01
	6	6.8492e-01	2.4887e+00	8.9220e-01	9.7225e-01	1.3493e+00
	7	2.5634e-02	1.0190e+00	2.7697e-02	5.9555e-02	6.7596e-02
	8	3.6285e+00	1.2589e+01	6.6274e+00	5.8903e+00	6.7062e+00
	9	6.4028e+00	1.6510e+01	9.3395e+00	8.8193e+00	8.5368e+00
	10	6.7845e+01	2.8852e+02	1.1994e+02	1.3100e+02	1.3971e+02
	11	2.1102e+02	1.1878e+03	4.7046e+02	2.5693e+02	2.5919e+02
	12	1.9412e-01	2.2869e+00	3.5016e-01	6.0007e-02	8.0529e-02
	13	8.9394e-02	2.9047e-01	7.5061e-02	9.4233e-02	7.1274e-02
	14	2.7045e-01	4.9721e-01	2.9625e-01	3.0278e-01	2.5294e-01
	15	8.9744e-01	2.8492e+00	1.0124e+00	8.5777e-01	7.6538e-01
	16	1.6813e+00	3.4910e+00	1.8494e+00	2.2432e+00	1.7438e+00
Híbridas Función 1	**17**	4.4129e+03	3.6888e+05	1.6802e+05	1.0080e+04	7.9785e+03
	18	3.2796e+03	8.3484e+03	6.4743e+03	1.1931e+04	1.1616e+04
	19	1.0478e+00	2.3462e+00	1.1198e+00	1.4295e+00	1.2619e+00
	20	1.0811e+03	1.0017e+04	6.7911e+03	4.9414e+03	4.0037e+03
	21	9.9040e+02	1.2894e+05	2.8723e+04	2.9665e+03	1.8785e+03
Composición de Funciones	**22**	1.0174e+01	6.6400e+01	3.7563e+01	3.3007e+01	3.7571e+01
	23	3.2945e+02	3.3298e+02	3.2945e+02	3.2945e+02	3.2945e+02
	24	1.1696e+02	1.4775e+02	1.2625e+02	1.1727e+02	1.1533e+02
	25	1.6253e+02	1.9781e+02	1.9253e+02	1.8447e+02	1.9253e+02
	26	1.0009e+02	1.0027e+02	1.0008e+02	1.0008e+02	1.0007e+02
	27	8.7279e+01	3.4281e+02	2.8240e+02	2.4831e+02	2.0931e+02
	28	3.8157e+02	4.5314e+02	4.1560e+02	4.2631e+02	4.5258e+02
	29	4.5704e+02	6.9891e+04	6.1174e+02	4.2500e+04	4.6736e+02
	30	6.4339e+02	1.2212e+03	8.0306e+02	9.5011e+02	9.1190e+02
	Recuento	**21**	**0**	**1**	**4**	**7**
	Ranking	**1.483**	**4.933**	**3.050**	**2.900**	**2.633**

Tabla 13: Comparativa ISLAS *vs* AGE, Dimensión 30, Tamaño Población 60

	Función	ISLAS	TB-RW	NAM-RW	TB-DC	NAM-DC
Funciones unimodales	**1**	5.1771e+06	1.1916e+08	1.4570e+06	1.4407e+06	2.6206e+06
	2	9.3516e+03	1.2052e+09	1.1501e+04	1.6902e+03	8.6171e+03
	3	2.8612e+03	8.1310e+04	2.1060e+04	3.0364e+03	5.0131e+03
Funciones multimodales Simples	**4**	9.5424e+01	2.6966e+02	8.9712e+01	9.3134e+01	1.1954e+02
	5	2.0677e+01	2.1192e+01	2.0909e+01	2.0789e+01	2.0912e+01
	6	6.9519e+00	1.7514e+01	7.8676e+00	9.2749e+00	7.5959e+00
	7	2.6638e-02	1.8105e+01	1.7852e-02	6.2063e-02	1.2408e-02
	8	2.7774e+01	5.9742e+01	2.7377e+01	4.3681e+01	3.9696e+01
	9	4.1652e+01	1.1170e+02	4.4058e+01	4.9968e+01	5.2751e+01
	10	5.7865e+02	3.0434e+03	1.0648e+03	8.3021e+02	9.2419e+02
	11	1.3906e+03	7.4797e+03	3.1212e+03	1.9024e+03	1.5268e+03
	12	2.1299e-01	4.0723e+00	2.0284e+00	7.8601e-02	5.2923e-02
	13	2.0226e-01	5.9248e-01	2.1503e-01	2.8563e-01	2.3721e-01
	14	2.9372e-01	2.3140e+00	3.0385e-01	3.9394e-01	3.9921e-01
	15	4.0990e+00	1.7903e+02	4.1669e+00	4.8115e+00	3.4723e+00
	16	9.4650e+00	1.3330e+01	9.3609e+00	1.0223e+01	9.2643e+00
Híbridas Función 1	**17**	3.8396e+05	1.1845e+07	7.9535e+05	3.0375e+05	4.4548e+05
	18	4.7465e+02	5.3168e+06	2.7573e+03	4.4653e+03	9.5783e+03
	19	6.6742e+00	7.1090e+01	7.0760e+00	1.3861e+01	1.3805e+01
	20	7.8843e+03	8.4253e+04	2.7974e+04	1.4078e+04	1.8457e+04
	21	1.9271e+05	2.3949e+06	3.5593e+05	1.4936e+05	1.5443e+05
Composición de Funciones	**22**	1.9731e+02	6.7019e+02	3.6703e+02	4.4510e+02	3.2607e+02
	23	3.1524e+02	3.2676e+02	3.1528e+02	3.1534e+02	3.1525e+02
	24	2.3175e+02	2.4959e+02	2.3721e+02	2.3053e+02	2.3249e+02
	25	2.0696e+02	2.2070e+02	2.1034e+02	2.0704e+02	2.0555e+02
	26	1.0021e+02	1.1628e+02	1.0531e+02	1.1027e+02	1.0025e+02
	27	4.2317e+02	7.5335e+02	4.8956e+02	5.3681e+02	4.7603e+02
	28	8.9731e+02	1.3151e+03	9.2209e+02	9.3646e+02	9.8801e+02
	29	1.2977e+03	1.4712e+06	4.3510e+05	1.6584e+03	4.5960e+05
	30	2.7189e+03	5.9619e+04	6.8678e+03	3.0010e+03	2.6889e+03
	Recuento	**17**	**0**	**2**	**5**	**6**
	Ranking	**1.700**	**5.000**	**2.933**	**2.800**	**2.566**

En la tabla 13, encontramos ahora los resultados de los mismos algoritmos, pero para dimensión 30. Como podemos observar, las conclusiones anteriores que hemos obtenido se ven ratificadas, con algunos aspectos notables que pensamos que son dignos de mención.

1. Con el aumento de dimensión, se mantiene una sincronía entre los resultados, ya que los puestos que determina el ranking se compensan.

2. Con el conjunto de funciones que van desde la número 12 a la 16, se mantienen las mismas características que mencionamos anteriormente, por lo que lo consideramos como confirmación de lo que veníamos comentando.

3. En tercer lugar, mencionar que hemos incorporado un algoritmo que es realmente malo por sus amplias capacidades de explotación (AGE TB-RW). Tanto

en la tabla 12 como en la tabla 13 obtiene unos resultados nefastos. Cuando tenemos un algoritmo muy malo en una combinación de Islas, éste la perjudica. De hecho, al aumentar de dimensión, que este algoritmo es empeora también, los resultados para el Modelo de Islas también se ven perjudicados en mayor cantidad, por lo que pensamos que existe una relación clara donde los resultados de TB-RW entorpecen al Modelo de Islas de forma notable.

4. Por último, mencionar que para los algoritmos que son resultado de una composición de funciones, el Modelo de Islas sigue generando buenos resultados. **En funciones que están compuestas unas con otras, y además con posibilidad de que existan zonas muy diversas en el espacio, los modelos de islas pueden dar muy buenos resultados, ya que facilitan el desplazamiento por el espacio.** Pensamos que este sería el motivo por el que en este subconjunto de funciones destaca por sus buenos resultados.

Poblaciones de tamaño 240

En la tabla 14 y en la tabla 15, podemos ver los resultados obtenidos utilizando poblaciones de tamaño 240 para las poblaciones de los algoritmos genéticos. De nuevo, tal y como pudimos ver en el caso de los algoritmos genéticos generacional, debe de haber una relación entre los resultados obtenidos y la diversidad que conlleva utilizar poblaciones de mayor tamaño o menor a la hora de interpretar los resultados.

Si estudiamos Dimensión 10 (disponible en la tabla 14), podemos ver algunos comportamientos destacables. En primer lugar, deberíamos mencionar que a pesar de que el algoritmo que obtiene mejores resultados es el algoritmo genético estacionario con *Emparejamiento Variado Inverso*(NAM) y *Reemplazar Peor*(RW), si observamos los valores proporcionados en el valor de Ranking, podemos ver que la diferencia entre la puntuación de dicho algoritmo en comparación con el Modelo de Islas es muy pequeña, por lo que realmente estamos muy igualados en esta situación. Si nos centramos en analizar los resultados, vemos que la tendencia que se ha ido mostrando hasta ahora se repite.

1. Hay casos, como puede ser el de la `Función 7`, en los que hay algún resultado realmente malo, a pesar de existir alguno muy bueno, lo que afecta negativamente al Modelo de Islas y no le permite desarrollarse lo mejor que se pudiera.

2. Hay casos, como el de la `Función 2`, donde hay resultados muy distintos entre sí, lo que resulta beneficioso para el Modelo de Islas, ya que es capaz de aprovechar la sinergia existente para obtener mejores resultados.

3. Hay casos como el de la `Función 25` o la `Función 12`, donde todos los resultados para una misma función son muy similares probablemente debido a no poder salir de una zona no prometedora, de forma que el Modelo de Islas puede aprovechar esta igualdad para realmente avanzar y tener buenos resultados. Del mismo modo, los casos donde hay tan poca diferencia no son realmente representativos de forma individual, ya que no podemos sacar conclusiones de diferencias pequeñas. De hecho, podemos observar cómo realmente destaca su comportamiento respecto al resto de las técnicas si observamos la convergencia del Modelo de Islas en la imagen 14

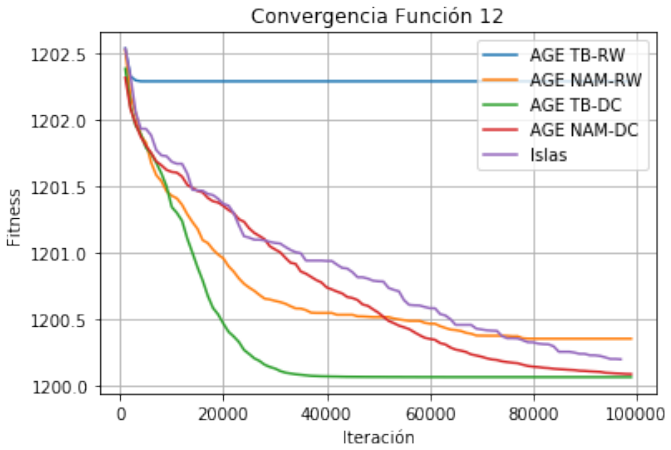


Figura 14: Convergencia de la Función Número 12, Dimensión 10

Si consultamos la tabla 15, podemos ver que la tendencia se mantiene. De hecho, si consultamos los valores del Ranking, observamos que las diferencias entre las puntuaciones alcanzadas en una dimensión y en otra no suponen una gran diferencia, sino que en general los resultados no varían en cantidad.

Podemos destacar algunos comportamientos que se mantienen para ambas dimensiones. Un ejemplo podemos verlo en el caso de la `Función 23`, donde podemos ver lo que con alta probabilidad será un mínimo local, el cuál ninguno de los algoritmos ha sido capaz de superar. Aquí podemos ver un claro ejemplo donde la técnica no es infalible, y que depende mucho de los algoritmos que la contienen. De hecho, observando su gráfica de convergencia, que se puede ver en la imagen 15, podemos ver claramente cómo los algoritmos convergen hacia el mismo valor, con mayor o menor rapidez, lo que nos lleva a confirmar la existencia de un posible mínimo local en ese caso concreto.

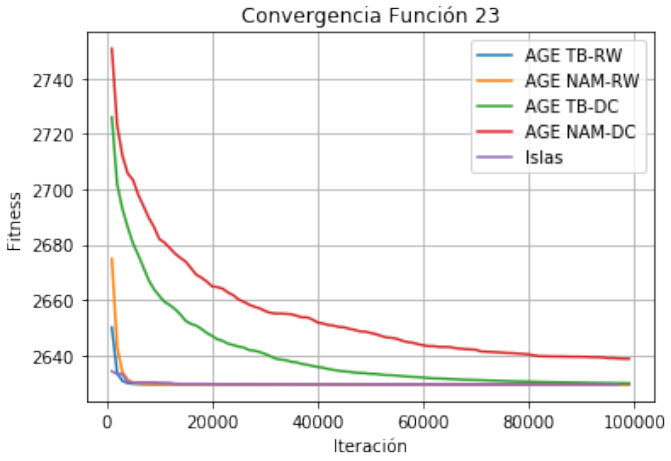
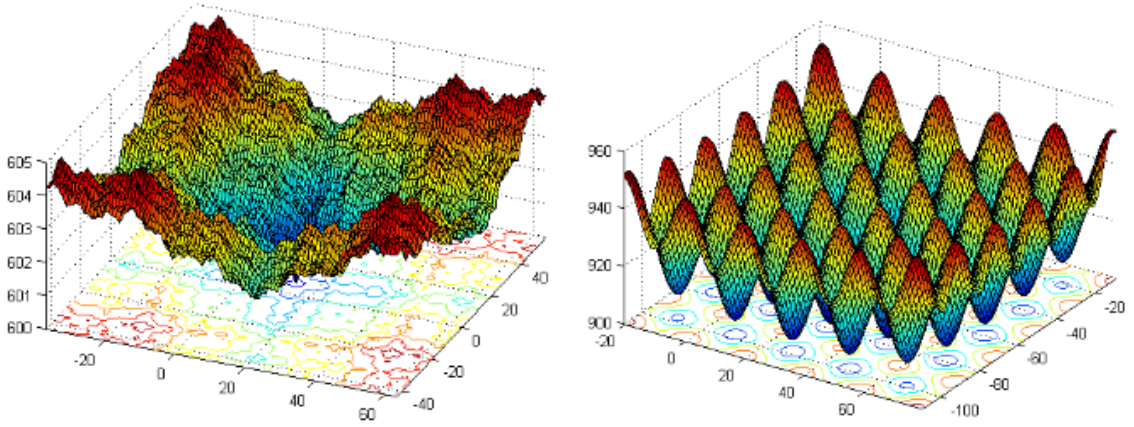


Figura 15: Convergencia de la Función Número 23, Dimensión 10

Otro caso digno de mención puede ser el grupo de funciones que van desde la Función

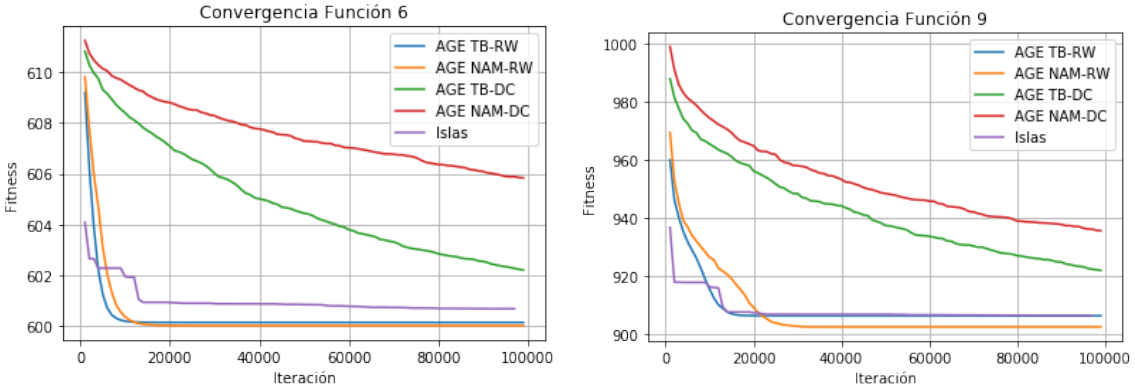
6 hasta la Función 10. Este grupo, como podemos observar, en ambas dimensiones obtiene los mejores resultados con el algoritmo estacionario **NAM-RW**. Esta combinación de operadores, funciona bien cuando estamos trabajando con funciones que no requieren una alta complejidad, ya que, mientras que proporcionan una componente elitista para conservar mejores soluciones, permite obtener soluciones de calidad en función de los padres, y cuando se da esta situación, converge realmente rápido. Recordemos que, con el proceso de selección que lleva a cabo, va cogiendo soluciones distantes y generando hijos, por tanto cada vez quedarán menos soluciones muy separadas, sino que se irán centrando hacia una zona concreta, (siendo por tanto de convergencia rápida). Si además, el espacio de búsqueda no tiene excesiva complejidad, podemos alcanzar buenas soluciones con frecuencia. Podemos ver algunas de las funciones que consiguen buenos resultados con este algoritmo en la imagen 16.



(a) Función 6 (b) Función 9

Figura 16: Funciones 6 y 9

Tal y como hemos comentado, y como podemos ver en las gráficas de convergencia mostradas en la imagen 17, el algoritmo genético estacionario NAM-RW muestra el comportamiento esperado para estas funciones, ya que converge con una rapidez enorme en comparación a otros de los algoritmos utilizados. Esta rápida convergencia, unida a las funciones concretas que estamos mencionando, suponen que este algoritmo sea capaz de ser competitivo en el contexto que estamos estudiando.



(a) Conv. Función 6

(b) Conv. Función 9

Figura 17: Convergencia AGE Funciones 6 y 9, Dimensión 10

Tabla 14: Comparativa Modelos de Islas *vs* AGE, Dimensión 10, Tamaño población 240

	Función	ISLAS	TB-RW	NAM-RW	TB-DC	NAM-DC
Funciones Unimodales	**1**	1.2687e+05	6.0144e+05	7.4428e+04	4.8676e+05	4.0749e+06
	2	1.4547e+03	1.0482e+03	7.2105e+02	1.8222e+07	2.6538e+08
	3	9.8373e+02	3.9867e+03	2.3256e+03	2.2700e+03	1.1065e+04
Funciones Multimodales Simples	**4**	2.4256e+01	3.3391e+01	3.4171e+01	3.5165e+01	6.1783e+01
	5	1.8092e+01	2.0511e+01	1.8680e+01	1.8719e+01	2.0257e+01
	6	6.8492e-01	1.4673e-01	2.1035e-02	2.1692e+00	5.7703e+00
	7	2.5634e-02	5.7653e-03	3.8403e-03	1.2570e+00	4.3618e+00
	8	3.6285e+00	3.3629e+00	2.5669e+00	1.6550e+01	4.5313e+01
	9	6.4028e+00	6.4264e+00	2.5669e+00	2.1822e+01	3.5710e+01
	10	6.7845e+01	6.5344e+01	6.2143e+01	1.7189e+02	8.2647e+02
	11	2.1102e+02	5.6540e+02	4.8549e+01	3.3371e+02	1.0019e+03
	12	1.9412e-01	1.4773e+00	8.7579e-01	1.1036e+00	1.1758e+00
	13	8.9394e-02	1.5675e-01	8.9885e-02	2.4099e-01	5.6857e-01
	14	2.7045e-01	3.4397e-01	3.4761e-01	2.9640e-01	1.9804e+00
	15	8.9744e-01	1.1856e+00	9.7879e-01	3.9297e+00	1.0890e+01
	16	1.6813e+00	2.2980e+00	1.2384e+00	2.8984e+00	3.1017e+00
Híbridas Función 1	**17**	4.4129e+03	2.8500e+04	6.4435e+03	3.1627e+03	1.8713e+04
	18	3.2796e+03	7.1135e+03	8.1343e+03	1.0658e+04	3.4501e+04
	19	1.0478e+00	4.9663e-01	3.5989e-01	2.4597e+00	4.9825e+00
	20	1.0811e+03	4.5025e+03	3.7892e+03	1.9703e+03	2.2611e+03
	21	9.9040e+02	4.4455e+03	1.6056e+03	2.3646e+03	5.2675e+03
Composición de Funciones	**22**	1.0174e+01	9.1653e+00	5.2574e+00	2.6482e+01	5.0143e+01
	23	3.2945e+02	3.2945e+02	3.2945e+02	3.2983e+02	3.3844e+02
	24	1.1696e+02	1.1257e+02	1.0934e+02	1.2726e+02	1.4595e+02
	25	1.6253e+02	1.9183e+02	1.8243e+02	1.6787e+02	1.9104e+02
	26	1.0009e+02	1.0013e+02	1.0008e+02	1.0022e+02	1.0054e+02
	27	8.7279e+01	2.4065e+02	1.9881e+02	6.8228e+01	2.4277e+01
	28	3.8157e+02	3.7946e+02	3.8003e+02	3.9720e+02	5.1697e+02
	29	4.5704e+02	6.2248e+02	5.8943e+02	8.7501e+02	6.4977e+03
	30	6.4339e+02	6.7642e+02	5.7951e+02	7.2413e+02	1.8480e+03
	Recuento	**13**	**2**	**15**	**1**	**1**
	Ranking	**1.930**	**3.100**	**1.900**	**3.400**	**4.666**

Tabla 15: Comparativa Modelos de Islas vs AGE, Dimensión 30 Tamaño Población 240

	Función	ISLAS	TB-RW	NAM-RW	TB-DC	NAM-DC
Funciones unimodales	**1**	5.1771e+06	5.5134e+05	7.8461e+05	8.0368e+06	7.5853e+07
	2	9.3516e+03	1.2748e+04	9.9445e+03	7.1070e+07	9.3563e+09
	3	2.8612e+03	1.3992e+04	1.0528e+04	8.8364e+03	4.4448e+04
Funciones multimodales Simples	**4**	9.5424e+01	6.3184e+01	7.5236e+01	1.4550e+02	4.6364e+02
	5	2.0677e+01	2.1049e+01	2.0912e+01	2.0950e+01	2.0931e+01
	6	6.9519e+00	2.0418e+00	9.0951e-01	7.1233e+00	2.5641e+01
	7	2.6638e-02	1.0864e-03	4.3029e-04	1.6385e+00	5.8333e+01
	8	2.7774e+01	1.9542e+01	1.1442e+01	5.1496e+01	2.1656e+02
	9	4.1652e+01	7.2292e+01	2.1708e+01	1.3101e+02	2.3836e+02
	10	5.7865e+02	1.0786e+03	1.9898e+02	8.0384e+02	4.6364e+03
	11	1.3906e+03	5.1180e+03	2.6044e+03	3.1515e+03	5.7025e+03
	12	2.1299e-01	3.2847e+00	2.4096e+00	2.5213e+00	2.4662e+00
	13	2.0226e-01	3.9735e-01	1.9570e-01	2.4033e-01	1.7300e+00
	14	2.9372e-01	3.8933e-01	3.0355e-01	3.9443e-01	2.4347e+01
	15	4.0990e+00	1.1687e+01	4.0003e+00	1.6909e+01	1.3002e+03
	16	9.4650e+00	1.2835e+01	1.1027e+01	1.1324e+01	1.1941e+01
Híbridas Función 1	**17**	3.8396e+05	3.2417e+05	2.6821e+05	3.7724e+05	1.3699e+06
	18	4.7465e+02	5.4486e+02	7.3707e+02	4.2048e+03	1.0107e+07
	19	6.6742e+00	7.8057e+00	3.7262e+00	8.0522e+00	2.4359e+01
	20	7.8843e+03	2.2892e+04	4.7860e+03	1.0559e+04	9.5148e+03
	21	1.9271e+05	2.5189e+05	2.2956e+05	1.9470e+05	4.2803e+05
Composición de Funciones	**22**	1.9731e+02	3.5326e+02	1.5081e+02	8.9485e+01	4.3008e+02
	23	3.1524e+02	3.1524e+02	3.1524e+02	3.1603e+02	3.4054e+02
	24	2.3175e+02	2.3163e+02	2.2837e+02	2.2651e+02	2.7901e+02
	25	2.0696e+02	2.0862e+02	2.0723e+02	2.0512e+02	2.1504e+02
	26	1.0021e+02	1.0038e+02	1.0019e+02	1.0022e+02	1.0128e+02
	27	4.2317e+02	3.4215e+02	3.2039e+02	4.0717e+02	4.5236e+02
	28	8.9731e+02	8.5735e+02	8.4185e+02	8.7212e+02	1.2704e+03
	29	1.2977e+03	1.5954e+03	1.5960e+03	3.0908e+03	6.5328e+05
	30	2.7189e+03	3.3526e+03	3.2080e+03	2.6512e+03	4.2146e+04
	Recuento	**11**	**3**	**12**	**4**	**2**
	Ranking	**2.133**	**3.133**	**1.766**	**3.200**	**4.766**

Antes de terminar esta sección, vamos a observar el comportamiento interno del Modelo de Islas que involucra los algoritmos estacionarios. De nuevo, la gráfica que estamos utilizando representa, para cada algoritmo, la cantidad de veces cuya solución ha sido la mejor de las 4 resultantes en la iteración del Modelo de Isla. Por tanto, a mayor área ocupada en la barra, mayor será la mejoría que ha proporcionado.

De nuevo, podemos observar, tanto en la tabla 16 como en la tabla 17 algo que ya sabíamos: no todos los algoritmos proporcionan mejoras en todas las funciones. Podemos ver el caso de las funciones 1,2,6,7,8 de la tabla 16, donde no participan todos los algoritmos aportando mejoras.

También podemos verlo de forma más intensificada en dimensión 30: en la tabla 17 podemos ver incluso funciones donde sólo aportan mejores soluciones uno de

los algoritmos incluidos. Esto nos lleva a una necesidad de adaptación para intentar mejorar resultados, en lugar de una mezcla ya preparada de antemano que no debería aplicarse a la ligera.

Por último mencionar dos tendencias muy claras:

1. Al aumentar la dimensión, notamos un cambio claro acerca del protagonismo que toman las distintas funciones. Mientras que en dimensión 10, los algoritmos que más destacan dentro del modelo son aquellos no extremos, que combinan cualidades de explotación y exploración, el algoritmo que más contribuye al modelo en dimensión 30 es aquel que más capacidad de exploración experimenta (NAM-DC). **Por tanto, podemos ver que no existen las mismas necesidades en una dimensión u en otra y no podemos fijar que un algoritmo es mejor que otro en una función a priori.**

2. En todos los casos hay un algoritmo que sobresale. Esto no es más que una de las razones que justifican que para cada función, habrá uno de esos algoritmos que funcione mejor, y por tanto se repita más y ocupe más área de la barra acumulada.

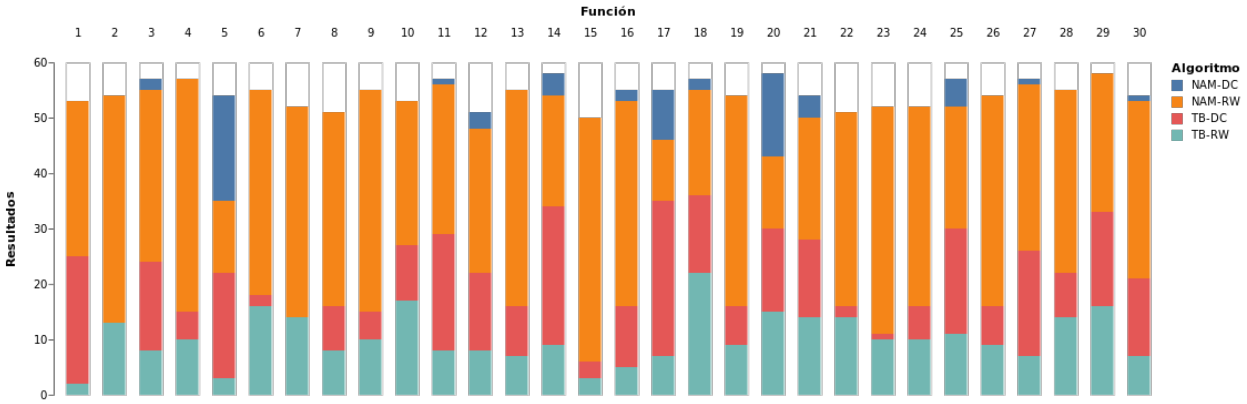


Figura 18: Estudio del comportamiento del Modelo de Islas con AGE, Dim=10

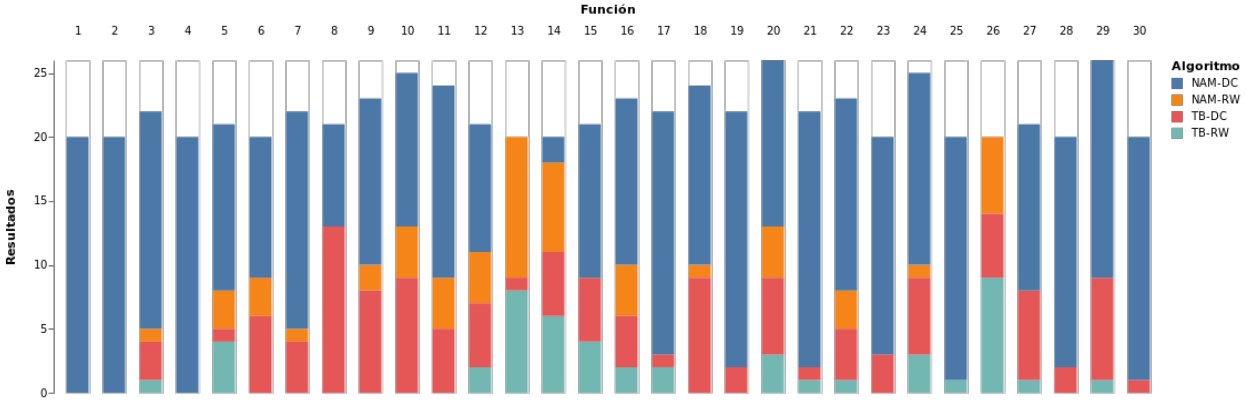


Figura 19: Estudio del comportamiento del Modelo de Islas con AGE, Dim=30

6.1.3. Estudio de los Modelos de Islas con los mejores Algoritmos Genéticos resultantes

En esta sección, vamos a estudiar los resultados que genera el Modelo de Islas **heterogéneo** que hemos implementado. En esta sección, vamos a verlo todo de una forma más genérica, puesto que los resultados reflejan un resumen de todo lo que ya hemos venido comentario anteriormente en las dos versiones anteriores del modelo.

Poblaciones Tamaño 60

Tabla 16: Comparativa Modelos Islas *vs* mejores AGs, Dimensión 10, Tam. Población 60

	Función	ISLAS	AGG 0.5	AGG 0.8	NAM-RW	NAM-DC
Funciones unimodales	**1**	5.0092e+05	1.3739e+07	1.7166e+06	4.5445e+06	1.2852e+05
	2	2.0814e+03	2.1115e+07	5.2363e+03	2.3344e+03	3.7183e+03
	3	1.4915e+03	1.3467e+04	1.7111e+04	5.1444e+03	2.2521e+03
Funciones multimodales Simples	**4**	1.7459e+01	4.1253e+01	2.8846e+01	2.6471e+01	2.4316e+01
	5	1.8235e+01	2.0398e+01	2.0543e+01	1.8308e+01	1.8705e+01
	6	9.3400e-01	3.3448e+00	3.6400e+00	8.9220e-01	1.3493e+00
	7	2.9913e-02	3.7040e-01	7.4475e-02	2.7697e-02	6.7596e-02
	8	4.6105e+00	1.2338e+01	1.6869e+01	6.6274e+00	6.7062e+00
	9	6.8679e+00	1.6095e+01	2.4467e+01	9.3395e+00	8.5368e+00
	10	9.7384e+01	3.3073e+02	3.8424e+02	1.1994e+02	1.3971e+02
	11	1.9769e+02	7.7194e+02	1.0637e+03	4.7046e+02	2.5919e+02
	12	2.3726e-01	8.4969e-01	1.1069e+00	3.5016e-01	8.0529e-02
	13	1.1276e-01	2.4493e-01	2.7126e-01	7.5061e-02	7.1274e-02
	14	2.8986e-01	4.5416e-01	4.3870e-01	2.9625e-01	2.5294e-01
	15	9.9687e-01	2.2581e+00	1.5979e+00	1.0124e+00	7.6538e-01
	16	2.0866e+00	3.2780e+00	3.5895e+00	1.8494e+00	1.7438e+00
Híbridas Función 1	**17**	6.2964e+03	5.2481e+05	1.5050e+05	1.6802e+05	7.9785e+03
	18	3.6020e+03	1.6438e+04	1.0794e+04	6.4743e+03	1.1616e+04
	19	1.2682e+00	2.6449e+00	2.4317e+00	1.1198e+00	1.2619e+00
	20	6.9206e+02	6.7841e+03	1.0622e+04	6.7911e+03	4.0037e+03
	21	1.4470e+03	1.0274e+05	1.5027e+04	2.8723e+04	1.8785e+03
Composición de Funciones	**22**	1.0193e+01	7.7622e+01	6.1099e+01	3.7563e+01	3.7571e+01
	23	3.2945e+02	3.3118e+02	3.3041e+02	3.2945e+02	3.2945e+02
	24	1.1610e+02	1.3911e+02	1.3759e+02	1.2625e+02	1.1533e+02
	25	1.7673e+02	1.9939e+02	1.9927e+02	1.9253e+02	1.9253e+02
	26	1.0010e+02	1.0025e+02	1.0028e+02	1.0008e+02	1.0007e+02
	27	9.0084e+01	3.3075e+02	3.1702e+02	2.8240e+02	2.0931e+02
	28	4.0269e+02	4.8659e+02	4.2352e+02	4.1560e+02	4.5258e+02
	29	4.3699e+02	4.1881e+04	1.5624e+05	6.1174e+02	4.6736e+02
	30	6.9155e+02	1.4501e+03	8.0089e+02	8.0306e+02	9.1190e+02
	Recuento	**17**	**0**	**1**	**3**	**9**
	Ranking	**1.533**	**4.533**	**4.200**	**2.550**	**2.183**

Por tanto, en un primer lugar podemos ver cómo el equiparar las poblaciones a la hora de comparar, es esencial, ya que es determinante a la hora de interpretar re-

sultados. Con este cambio en las poblaciones, podemos ver cómo el Modelo de Islas deja de ser el que obtiene mejores resultados, para dar lugar al algoritmo genético estacionario que utiliza *Emparejamiento Variado Inverso* y *Crowding Determinístico*. Como comentábamos anteriormente, este algoritmo obtiene buenos resultados cuando somos capaces de generar hijos muy buenos, a partir de padres malos. Para que este algoritmo provoque realmente efecto, tenemos que contar con poblaciones con diversidad, donde el propio algoritmo pueda escoger entre una gran variedad de soluciones a la hora de determinar el padre. El hecho de combinar poblaciones tan grandes, y además, soluciones de una dimensión mayor, provoca este efecto, y podría ser una de las causas por las que se obtienen tan buenos resultados con dicho algoritmo. De este modo, como veníamos comentando anteriormente, cuando un algoritmo realmente destaca sobre los demás, el Modelo de Islas lo suele equiparar y obtener un resultado que representa a la totalidad, por lo que los resultados con el algoritmo individual serán superiores.

Además, es necesario destacar las funciones que van desde la 6 a la 10, que forman un bloque interesante donde este algoritmo obtiene muy buenos resultados. Para ello, vamos a ver alguna de ellas. En la selección de figuras 16, podemos ver cómo estos algoritmos se caracterizan por tener estructuras también con muchos máximos y mínimos, lo cuál no hace más que ver, que lo expuesto anteriormente, con la necesidad de superar aquellos mínimos locales, hacen al algoritmo genético estacionario de Emparejamiento Variado Inverso con Reemplazar Peor una opción muy potencial en este proyecto. Las características del espacio de búsqueda en dicha función, son determinantes también para explicar que sus resultados se ven gratamente empeorados por la explotación, como podemos observar en la tabla 16, al igual que en la tabla 17

Tabla 17: Comparativa Modelos de Islas *vs* mejores AGs, Dimensión 30, Población 60

	Función	ISLAS	AGG 0.5	AGG 0.8	TB-DC	NAM-DC
Funciones unimodales	**1**	5.3193e+06	1.3579e+08	5.3139e+07	1.4407e+06	2.6206e+06
	2	6.2833e+03	1.4886e+09	1.3695e+09	1.6902e+03	8.6171e+03
	3	2.7764e+03	6.9242e+04	6.1670e+04	3.0364e+03	5.0131e+03
Funciones multimodales Simples	**4**	1.0926e+02	3.0729e+02	1.6931e+02	9.3134e+01	1.1954e+02
	5	2.0354e+01	2.0825e+01	2.1068e+01	2.0789e+01	2.0912e+01
	6	8.7581e+00	1.7874e+01	1.8268e+01	9.2749e+00	7.5959e+00
	7	1.5136e-02	1.9213e+01	6.1387e+00	6.2063e-02	1.2408e-02
	8	3.2938e+01	6.1901e+01	8.7885e+01	4.3681e+01	3.9696e+01
	9	4.9306e+01	9.1954e+01	1.7095e+02	4.9968e+01	5.2751e+01
	10	7.0523e+02	2.4715e+03	3.1230e+03	8.3021e+02	9.2419e+02
	11	1.6933e+03	4.5446e+03	5.5297e+03	1.9024e+03	1.5268e+03
	12	4.1456e-01	1.4926e+00	2.6792e+00	7.8601e-02	5.2923e-02
	13	2.4816e-01	4.8713e-01	7.8861e-01	2.8563e-01	2.3721e-01
	14	3.1666e-01	1.1406e+00	2.2999e+00	3.9394e-01	3.9921e-01
	15	5.0879e+00	2.6908e+02	6.3507e+04	4.8115e+00	3.4723e+00
	16	9.9512e+00	1.2813e+01	1.3488e+01	1.0223e+01	9.2643e+00
Híbridas Función 1	**17**	4.6420e+05	9.7566e+06	4.2351e+06	3.0375e+05	4.4548e+05
	18	2.3388e+03	4.8786e+06	6.8887e+03	4.4653e+03	9.5783e+03
	19	8.4087e+00	8.4923e+01	2.5122e+01	1.3861e+01	1.3805e+01
	20	1.0556e+04	6.3561e+04	5.0486e+04	1.4078e+04	1.8457e+04
	21	1.9287e+05	1.6784e+06	9.7467e+05	1.4936e+05	1.5443e+05
Composición de Funciones	**22**	2.6657e+02	5.7345e+02	6.7263e+02	4.4510e+02	3.2607e+02
	23	3.1524e+02	3.3350e+02	3.3037e+02	3.1534e+02	3.1525e+02
	24	2.3333e+02	2.4961e+02	2.5019e+02	2.3053e+02	2.3249e+02
	25	2.0606e+02	2.2382e+02	2.0986e+02	2.0704e+02	2.0555e+02
	26	1.0028e+02	1.1604e+02	1.1209e+02	1.1027e+02	1.0025e+02
	27	4.5061e+02	7.8389e+02	8.0668e+02	5.3681e+02	4.7603e+02
	28	9.0371e+02	1.4937e+03	1.1413e+03	9.3646e+02	9.8801e+02
	29	1.5914e+03	1.1281e+06	1.3776e+06	1.6584e+03	4.5960e+05
	30	3.3217e+03	3.2739e+04	1.3471e+04	3.0010e+03	2.6889e+03
	Recuento	**14**	**0**	**0**	**7**	**9**
	Ranking	**1.766**	**4.466**	**4.466**	**2.200**	**2.100**

Al igual que ocurría con sus equivalentes en las dos versiones anteriores (sólo AGG o sólo AGE), los resultados del modelo siguen mostrando su superioridad para una población 60. Sin embargo, en este caso, el recuento obtenido, donde podemos ver el número de veces que cada algoritmo ha salido ganando, nos muestra un descenso, tanto en la tabla 16 como en la tabla 17. Esto se puede deber, a que hemos aumentado considerablemente la calidad de las soluciones, es decir, al tratar ahora únicamente con los mejores, es más fácil para uno de estos algoritmos destacar en cuanto a resultados, o incluso verse afectado por los resultados de otros algoritmos. Este hecho lo podemos ver en la tabla 16, por ejemplo en las funciones 11 o 12, y también en la 17 con las funciones 1 o 2. Estas funciones son ejemplos de cómo un buen resultado, al combinarlo con peores no nos permite mejorar en valores *Fitness* obtenidos.

Sin embargo, también por el hecho de estar tratando con los mejores algoritmos se podría dar la situación contraria. Esta sería que todos ellos converjan hacia unos valores de *Fitness* concretos, por causas como puede ser la dificultad de salir de determinadas zonas del espacio. Esta situación la podemos ver reflejada en funciones como la Función 4,5,27 o 28 de la tabla 9, y también en tabla 17 en situaciones como la función 19,20, 27 o 28.

Poblaciones Tamaño 240

Con la relación existente entre la diversidad de la población y el tamaño de la misma, es de esperar que en este caso también empeoren los resultados de los Modelos de Islas. De hecho, ocurre de dicha forma, tal y como se esperaba.

Tabla 18: Comparativa Modelos de Islas *vs* mejores AGs, Dimensión 10, Población 240

	Función	ISLAS	AGG05	AGG 08	TB-RW	NAM-RW
Funciones unimodales	**1**	1.5752e+05	2.1733e+06	2.2745e+05	6.0144e+05	7.4428e+04
	2	1.0672e+03	1.3810e+03	5.5215e+03	1.0482e+03	7.2105e+02
	3	1.5882e+03	2.2691e+03	5.5495e+03	3.9867e+03	2.3256e+03
Funciones multimodales simples	**4**	1.6113e+01	3.0481e+01	3.3704e+01	3.3391e+01	3.4171e+01
	5	1.9846e+01	1.8600e+01	2.0308e+01	2.0511e+01	1.8680e+01
	6	6.8365e-01	1.6552e-01	1.3404e+00	1.4673e-01	2.1035e-02
	7	2.3192e-02	3.9416e-03	5.5262e-02	5.7653e-03	3.8403e-03
	8	5.7001e+00	3.8604e+00	1.0317e+01	3.3629e+00	2.5669e+00
	9	5.9507e+00	3.5049e+00	2.0699e+01	6.4264e+00	2.5669e+00
	10	7.7954e+01	7.2378e+01	1.1952e+02	6.5344e+01	6.2143e+01
	11	3.2684e+02	8.8974e+01	7.0033e+02	5.6540e+02	4.8549e+01
	12	2.2760e-01	7.9644e-01	4.0755e-01	1.4773e+00	8.7579e-01
	13	9.4284e-02	3.2182e-02	1.6965e-01	1.5675e-01	8.9885e-02
	14	2.7804e-01	3.8735e-01	3.4250e-01	3.4397e-01	3.4761e-01
	15	8.5697e-01	1.0099e+00	2.1946e+00	1.1856e+00	9.7879e-01
	16	1.9262e+00	2.0125e+00	3.3862e+00	2.2980e+00	1.2384e+00
Híbridas Función 1	**17**	3.0557e+04	1.0340e+05	8.2781e+04	2.8500e+04	6.4435e+03
	18	2.0985e+03	7.8739e+03	8.9739e+03	7.1135e+03	8.1343e+03
	19	1.0338e+00	6.1893e-01	1.2340e+00	4.9663e-01	3.5989e-01
	20	1.9395e+03	3.7767e+03	4.7813e+03	4.5025e+03	3.7892e+03
	21	1.4486e+03	6.1609e+03	6.3551e+03	4.4455e+03	1.6056e+03
Composición de funciones	**22**	1.2414e+01	1.3068e+01	1.1712e+01	9.1653e+00	5.2574e+00
	23	3.2945e+02	3.2945e+02	3.2945e+02	3.2945e+02	3.2945e+02
	24	1.1681e+02	1.1338e+02	1.2598e+02	1.1257e+02	1.0934e+02
	25	1.7491e+02	1.9479e+02	1.9026e+02	1.9183e+02	1.8243e+02
	26	1.0009e+02	1.0003e+02	1.0014e+02	1.0013e+02	1.0008e+02
	27	2.6215e+02	2.6309e+02	2.9265e+02	2.4065e+02	1.9881e+02
	28	3.7553e+02	3.7649e+02	3.8118e+02	3.7946e+02	3.8003e+02
	29	4.4850e+02	4.6958e+02	9.5652e+02	6.2248e+02	5.8943e+02
	30	6.1567e+02	7.0842e+02	5.2901e+02	6.7642e+02	5.7951e+02
	Recuento	**12**	**4**	**2**	**2**	**14**
	Ranking	**2.433**	**3.033**	**4.300**	**3.200**	**2.033**

Con los resultados obtenidos, no hemos podido determinar dos mejores algoritmos genéticos generacionales, puesto que, el número de funciones que alcanzan el mejor resultado con $\alpha = 0.3$ y $\alpha = 0.8$ es el mismo. Por tanto, vamos a dar el resultado para ambos.

Tabla 19: Comparativa Modelos Islas *vs* mejores AGs I, Dimensión 30, Tamaño Población 240

	Función	ISLAS	AGG 0.3	AGG 0.5	NAM-RW	TB-DC
Funciones unimodales	**1**	4.5819e+06	1.0414e+08	1.8311e+06	7.8461e+05	8.0368e+06
	2	8.4337e+03	3.5869e+08	1.0291e+04	9.9445e+03	7.1070e+07
	3	3.7853e+03	2.6060e+04	1.1989e+04	1.0528e+04	8.8364e+03
Funciones multimodales Simples	**4**	9.8148e+01	2.2156e+02	7.5882e+01	7.5236e+01	1.4550e+02
	5	2.0560e+01	2.0182e+01	2.0945e+01	2.0912e+01	2.0950e+01
	6	7.1581e+00	1.1550e+01	2.7550e+00	9.0951e-01	7.1233e+00
	7	3.9358e-02	6.4480e+00	3.6926e-04	4.3029e-04	1.6385e+00
	8	3.1646e+01	3.0608e+01	1.5819e+01	1.1442e+01	5.1496e+01
	9	3.7551e+01	3.9830e+01	1.8569e+01	2.1708e+01	1.3101e+02
	10	6.1665e+02	1.1016e+03	3.3154e+02	1.9898e+02	8.0384e+02
	11	1.9445e+03	2.7156e+03	7.2590e+02	2.6044e+03	3.1515e+03
	12	3.5584e-01	1.3952e-01	1.3880e+00	2.4096e+00	2.5213e+00
	13	2.0474e-01	2.4683e-01	1.2183e-01	1.9570e-01	2.4033e-01
	14	2.7573e-01	2.3977e-01	3.9427e-01	3.0355e-01	3.9443e-01
	15	4.1642e+00	1.5402e+01	3.1837e+00	4.0003e+00	1.6909e+01
	16	1.0101e+01	1.1685e+01	1.1702e+01	1.1027e+01	1.1324e+01
Híbridas Función 1	**17**	4.3665e+05	6.5968e+06	6.4173e+05	2.6821e+05	3.7724e+05
	18	1.0774e+03	8.7953e+02	1.4282e+03	7.3707e+02	4.2048e+03
	19	6.6356e+00	6.8026e+01	8.6339e+00	3.7262e+00	8.0522e+00
	20	9.5225e+03	2.7414e+04	2.0329e+04	4.7860e+03	1.0559e+04
	21	2.2820e+05	7.6873e+05	2.8865e+05	2.2956e+05	1.9470e+05
Composición de Funciones	**22**	2.6527e+02	4.5780e+02	1.9404e+02	1.5081e+02	8.9485e+01
	23	3.1524e+02	3.2556e+02	3.1524e+02	3.1524e+02	3.1603e+02
	24	2.3198e+02	2.3595e+02	2.3424e+02	2.2837e+02	2.2651e+02
	25	2.0800e+02	2.1226e+02	2.0838e+02	2.0723e+02	2.0512e+02
	26	1.0024e+02	1.1068e+02	1.0011e+02	1.0019e+02	1.0022e+02
	27	4.5748e+02	6.2746e+02	3.6051e+02	3.2039e+02	4.0717e+02
	28	9.0140e+02	1.1422e+03	8.7343e+02	8.4185e+02	8.7212e+02
	29	1.4914e+03	5.1605e+04	1.6046e+03	1.5960e+03	3.0908e+03
	30	3.2738e+03	3.2580e+04	3.5245e+03	3.2080e+03	2.6512e+03
	Recuento	**5**	**4**	**6**	**12**	**5**
	Ranking	**2.666**	**4.300**	**2.800**	**1.866**	**3.366**

Tabla 20: Comparativa Modelos Islas *vs* mejores AGs II, Dimensión 30, Población 240

	Función	ISLAS	AGG 0.5	AGG 0.8	TB-DC	NAM-DC
Funciones unimodales	**1**	2.5406e+06	1.8311e+06	1.6361e+07	7.8461e+05	8.0368e+06
	2	7.0925e+03	1.0291e+04	2.3184e+07	9.9445e+03	7.1070e+07
	3	1.8475e+03	1.1989e+04	2.6751e+04	1.0528e+04	8.8364e+03
Funciones multimodales Simples	**4**	8.5251e+01	7.5882e+01	1.2273e+02	7.5236e+01	1.4550e+02
	5	2.0580e+01	2.0945e+01	2.0883e+01	2.0912e+01	2.0950e+01
	6	7.1369e+00	2.7550e+00	8.3898e+00	9.0951e-01	7.1233e+00
	7	2.2592e-02	3.6926e-04	9.1030e-01	4.3029e-04	1.6385e+00
	8	3.1288e+01	1.5819e+01	1.3213e+02	1.1442e+01	5.1496e+01
	9	4.0098e+01	1.8569e+01	2.0063e+02	2.1708e+01	1.3101e+02
	10	5.6767e+02	3.3154e+02	1.9729e+03	1.9898e+02	8.0384e+02
	11	1.9890e+03	7.2590e+02	4.1644e+03	2.6044e+03	3.1515e+03
	12	4.4145e-01	1.3880e+00	1.6689e+00	2.4096e+00	2.5213e+00
	13	2.1078e-01	1.2183e-01	6.1840e-01	1.9570e-01	2.4033e-01
	14	2.8141e-01	3.9427e-01	8.8237e-01	3.0355e-01	3.9443e-01
	15	4.1306e+00	3.1837e+00	3.2170e+01	4.0003e+00	1.6909e+01
	16	9.5850e+00	1.1702e+01	1.3026e+01	1.1027e+01	1.1324e+01
Híbridas Función 1	**17**	3.5281e+05	6.4173e+05	9.8933e+05	2.6821e+05	3.7724e+05
	18	9.6422e+02	1.4282e+03	3.4471e+03	7.3707e+02	4.2048e+03
	19	6.5924e+00	8.6339e+00	8.3061e+00	3.7262e+00	8.0522e+00
	20	8.9508e+03	2.0329e+04	4.6772e+04	4.7860e+03	1.0559e+04
	21	1.7492e+05	2.8865e+05	5.2825e+05	2.2956e+05	1.9470e+05
Composición de Funciones	**22**	2.4122e+02	1.9404e+02	2.4715e+02	1.5081e+02	8.9485e+01
	23	3.1524e+02	3.1524e+02	3.1582e+02	3.1524e+02	3.1603e+02
	24	2.3212e+02	2.3424e+02	2.3683e+02	2.2837e+02	2.2651e+02
	25	2.0687e+02	2.0838e+02	2.0720e+02	2.0723e+02	2.0512e+02
	26	1.0022e+02	1.0011e+02	1.0057e+02	1.0019e+02	1.0022e+02
	27	4.5848e+02	3.6051e+02	5.1522e+02	3.2039e+02	4.0717e+02
	28	9.1012e+02	8.7343e+02	8.6859e+02	8.4185e+02	8.7212e+02
	29	1.3269e+03	1.6046e+03	1.2746e+06	1.5960e+03	3.0908e+03
	30	2.6090e+03	3.5245e+03	3.6930e+03	3.2080e+03	2.6512e+03
	Recuento	**10**	**7**	**1**	**12**	**2**
	Ranking	**2.350**	**2.766**	**4.466**	**1.933**	**3.483**

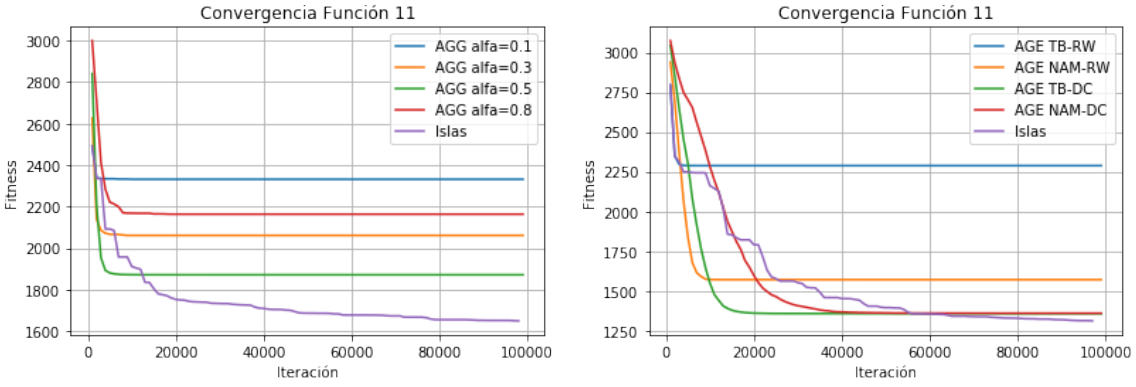
Por tanto, para concluir, podemos ver que para la versión estacionaria se obtiene mejores resultados que en comparación con la versión generacional. Como hemos comentado, el hecho de tener algoritmos más distintos entre sí, es una de las razones que propician esta situación, ya que da posibilidad a que se aprovechen las posibles situaciones en las que exista una sinergia, propiedad que los Modelos de Islas aprovechan muy bien.

Es importante destacar que, las distintas versiones realizadas para el modelo generacional, realmente son muy parecidas entre sí, ya que el cambio se limita a un parámetro en el cruce. Es cierto que este parámetro puede cambiar mucho los resultados que obtengamos en función de si es más bajo o si es un valor más alto [21], pero al fin y al cabo, todo lo demás permanece sin modificaciones. Este aspecto es diferente en la versión estacionaria, puesto que, en este caso, los algoritmos que

la componen son combinaciones de diferentes parámetros y operadores, y el hecho de que los algoritmos sean más diferentes entre sí, hacen al Modelo de Islas más potente, ya que le permite explorar una mayor área del espacio de búsqueda.

Por otra parte, es importante tener en cuenta a vista de los resultados que, en cuestión individual, los algoritmos estacionarios están dando de por sí mejores resultados que las versiones generacionales. Vamos a estudiar este hecho con un ejemplo. En la selección de imágenes 20 se puede ver, para la Función 11, el estudio de la convergencia obtenido para los dos modelos. Si observamos el comportamiento de ambas versiones de algoritmos genéticos, podemos ver que la versión estacionaria, en general, tarda más en converger a una solución estable, además de hacerlo con una pendiente rápida, generando así mejores soluciones. Esta selección de imágenes, es un claro ejemplo donde se va superioridad de dicha técnica.

Sin embargo, por otra parte, podemos ver que al tardar más en converger (caso estacionario), esto puede dificultar al buen funcionamiento del Modelo de Islas. Con ello, nos referimos a que un algoritmo que converja muy rápido, será más propicio a ser utilizado en un Modelo de Islas, ya que al reducir el número de iteraciones, esta reducción puede ni siquiera afectarle, ya que igualmente de forma individual no podría aspirar a más. **Sin embargo, si un algoritmo (como ocurre con el caso estacionario) funciona muy bien, pero se le corta en una iteración en la que pudiera dar más de sí, este hecho le perjudica, y por tanto el Modelo no es capaz de conseguir soluciones tan buenas como las que se esperaría. Aquí podemos ver, por tanto, la necesidad de realizar adaptaciones dentro de los modelos de hibridación, con el objetivo de intentar superar y salir favorecidos de las características que tengan los algoritmos que se involucran en la hibridación.**



(a) Conv. Versión Generacional (b) Conv. Versión Estacionaria

Figura 20: Convergencia para la Funciones 11, Dimensión 10, Población 60

6.2. Modelo Híbrido de Ejecución en 2 Fases

En esta sección, vamos a estudiar el comportamiento de uno de los Modelos de Hibridación más innovadores que hemos implementado para el estudio del proyecto. En este caso, como ya hemos venido comentando, se trata de un modelo que adapta el algoritmo a ejecutar durante su ejecución, basándose en que en las primeras iteraciones del algoritmo ya se puede determinar cuál es el algoritmo que será más prometedor en el caso concreto, y por tanto con el que obtendremos buenos resultados.

Para su estudio, seguiremos el mismo procedimiento que hemos llevado a cabo para el Modelo de Islas. En primer lugar, haremos un estudio previo para los resultados obtenidos con algoritmos genéticos generacionales. En segundo lugar, continuaremos con la versión estacionaria, y por último, el modelo heterogéneo que combina los mejores de ambas versiones anteriores.

6.2.1. Estudio del Modelo de Adaptación en 2 fases con los Algoritmos Genéticos Generacionales

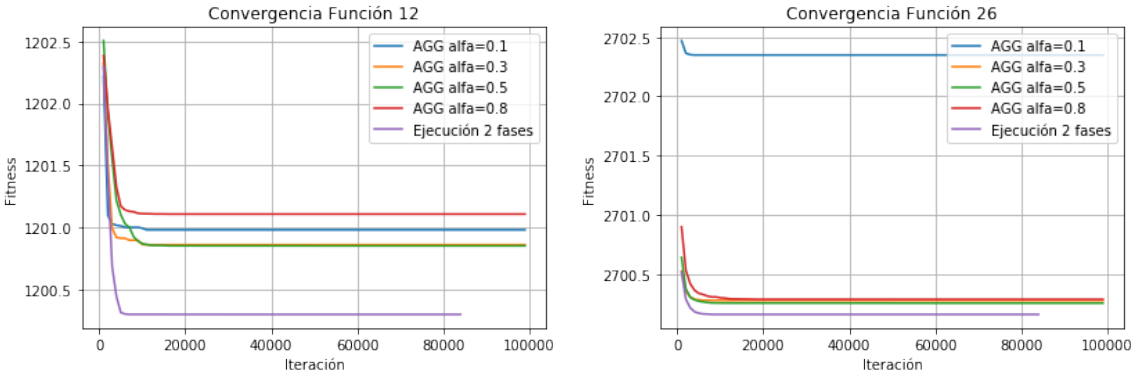
En este caso, no tiene sentido diferenciar entre las distintas dimensiones para las que hemos llevado a cabo el problema, porque como se puede observar tanto en la tabla 21 como en la tabla 22, ambas mantienen el mismo comportamiento para todos los algoritmos. En otras palabras, los puestos conseguidos por cada una de las técnicas en el Ranking, son las mismas para ambas dimensiones.

Lo que sí es realmente destacable, en este caso, es cómo el Modelo de Ejecución en 2 fases muestra una clara superioridad respecto a los algoritmos genéticos utilizados, y es que es ganadora en el 87 % de las veces, y, para los casos en los que no resulta ganadora, la diferencia es muy pequeña como para tomar realmente conclusiones. Por tanto, como primera conclusión, y basándonos en nuestros resultados, pensamos que el adaptar el algoritmo utilizado en tiempo de ejecución realmente da buenos resultados para este tipo de hibridaciones.

Otra de las razones, que en nuestra opinión le permite al modelo ser competitivo, es que se aprovecha de la rápida convergencia de los algoritmos que involucra. Estos algoritmos, le permiten fácilmente deducir, a tempranas iteraciones, cual es el camino o el comportamiento que van a tomar con una alta seguridad, por tanto, el análisis de sus mejoras alcanzadas, le permite llegar a buenos resultados gracias a aprovechar ese dato. De esta forma, **puede hacer uso de ese conocimiento de los algoritmos, y explotarlo, y de esta forma mejorar en cuanto a resultados**.

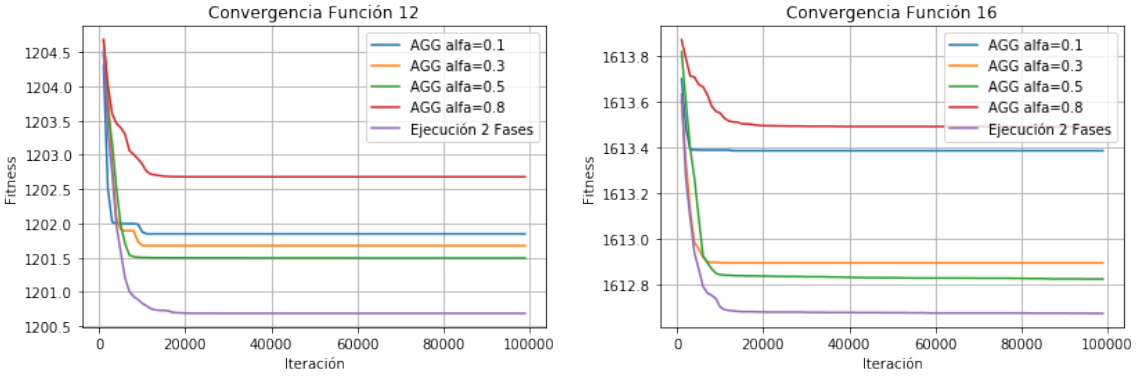
Para entenderlo de una forma más sencilla, vamos a visualizar algunas de las gráficas de convergencia que se generan en estas ejecuciones. Podemos ver este comportamiento reflejado en las selecciones de imágenes 26 y 21. Estas imágenes se corresponden con las gráficas de convergencia para la función 12 y 26 para dimensión 10 y dimensión 30 respectivamente. Como detalle, podemos ver que en este caso, la gráfica de convergencia del modelo, mantiene la misma forma que la de un algoritmo genético, puesto que, como hemos comentado, en realidad lo único que se hace

es decidir qué algoritmo se va a continuar ejecutando por el resto del tiempo de ejecución.



(a) Conv. Función 12 (b) Conv. Función 26

Figura 21: Convergencia AGE Funciones 12 y 26, Dimensión 10



(a) Conv. Función 12 (b) Conv. Función 26

Figura 22: Convergencia AGE Funciones 12 y 26, Dimensión 30

Pero entonces, ¿por qué hay casos, como los que vemos en las **gráficas**, en los que la diferencia entre los resultados del Modelo de Ejecución en 2 fases respecto a los algoritmos genéticos es tan grande? Su justificación se basa precisamente en que, para cada función, hay un algoritmo que normalmente da mejores resultados, debido a las características de ambos son más favorables a dar buenas soluciones que si se ejecutase otro de los algoritmos.

Para poder contrastar este hecho, vamos a analizar los diagramas de barra acumulada que se pueden ver en las gráficas 24 y 25. Estas gráficas representan, de forma proporcional,el número de veces que cada uno de los algoritmos ha sido elegido el mejor para continuar la ejecución, para cada una de las funciones que estamos estudiando. Podemos ver cómo existe un reparto, de forma que en todos los casos existe un algoritmo que parece funcionar mejor que el resto. De estas gráficas podemos extraer dos conclusiones fuertes.

1. Por otra parte, al igual que estas tablas nos permiten conocer el reparto que se lleva a cabo, este reparto tiene una relación directa con el comportamiento de los algoritmos individuales, y es que, cuanto mejor sea el comportamiento individual del algoritmo para un caso concreto, mayor será el área de la gráfica que ocupará. Esta relación, es totalmente lógica, puesto que el algoritmo que mejor funciona para cada función, probablemente será aquel que mayor número de veces se adapte la ejecución hacia él.

   Vamos a verlo, con un ejemplo. La `Función 11`, es una de las funciones para las que este modelo obtiene mejor resultado (en ambas dimensiones). Vamos a mostrar, en la imagen 23, la convergencia de esta función para una dimensión 30. En dicha imagen, podemos ver cómo, los mejores resultados se obtienen para el modelo híbrido, seguido de los algoritmos genéticos con $\alpha = 0.5$, $\alpha = 0.3$, $\alpha = 0.8$ y $\alpha = 0.1$ respectivamente. Si ahora miramos la proporción en el diagrama de barras acumulado, podemos ver que dicho comportamiento de los algoritmos reincide. **Por tanto, en este modelo, para una determinada función, los algoritmos se comportan de forma proporcional a la efectividad que tengan estos algoritmos de forma individual respecto a la misma función.**

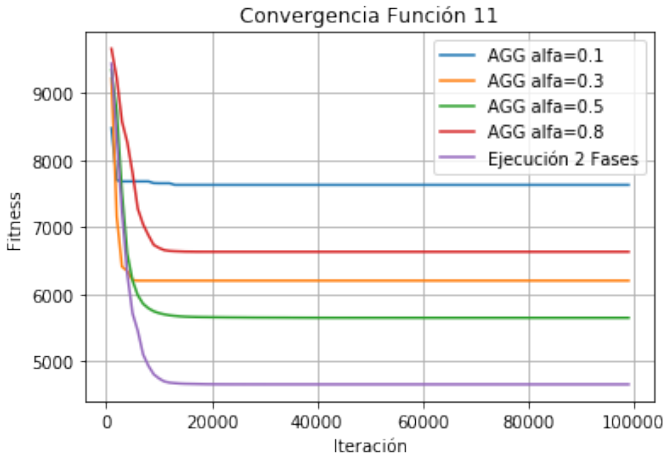


Figura 23: Convergencia Función 11, Dim=30

De igual forma, es de esperar que aunque no se obtengan los mejores resultados, el algoritmo que consiga el resultado ganador, también será el que mayor tiempo de ejecución acapare en el modelo híbrido. Podemos ver este hecho por ejemplo con la `Función 1`. Como podemos ver en la gráfica 24, el algoritmo que mejor comportamiento muestra en el modelo híbrido es aquel que utiliza un valor de $\alpha = 0.8$. Si consultamos la tabla 21, podemos ver que dicho algoritmo es el que proporciona un mejor resultado global para dicha función. Igual comportamiento se puede observar para otras funciones, como la `Función 2` o la `Función 30`.

Este hecho también se aprecia a dimensión 30. Por ejemplo, podemos verlo con la **Función 13**. En este caso, el mejor resultado lo facilita el algoritmo que utiliza $\alpha = 0.5$, y tal y como podemos observar en la gráfica 25, dicho

algoritmo es el que ocupa una mayor parte del área de la barra, y por tanto uno de los que más iteraciones realiza.

2. De hecho, si observamos la gráfica 24 y la comparamos con la gráfica 10 (su equivalente en el Modelo de Islas), podemos ver que hay una cierta tendencia o en los algoritmos que más favorecen. Con ello queremos hacer ver que, **la necesidad de adaptación es existente en las hibridaciones, ya que hay una influencia clara de los algoritmos al respecto. Aprovechar esta propiedad o no, ya dependerá del caso concreto. Este hecho, que no se aprovechaba en el Modelo de Islas, sí que se ve cubierto en este caso.**

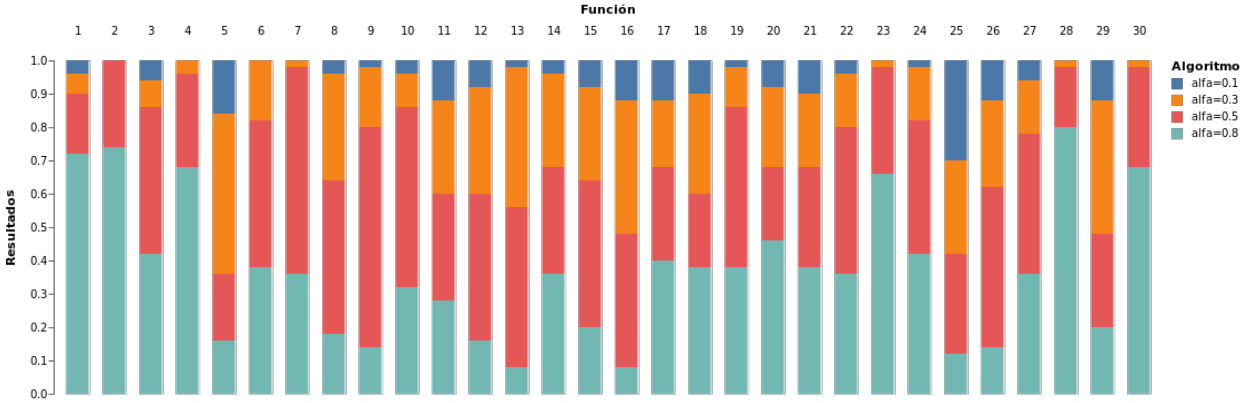


Figura 24: Estudio del comportamiento del Modelo de Ejecución en 2 fases con AGG, Dim=10

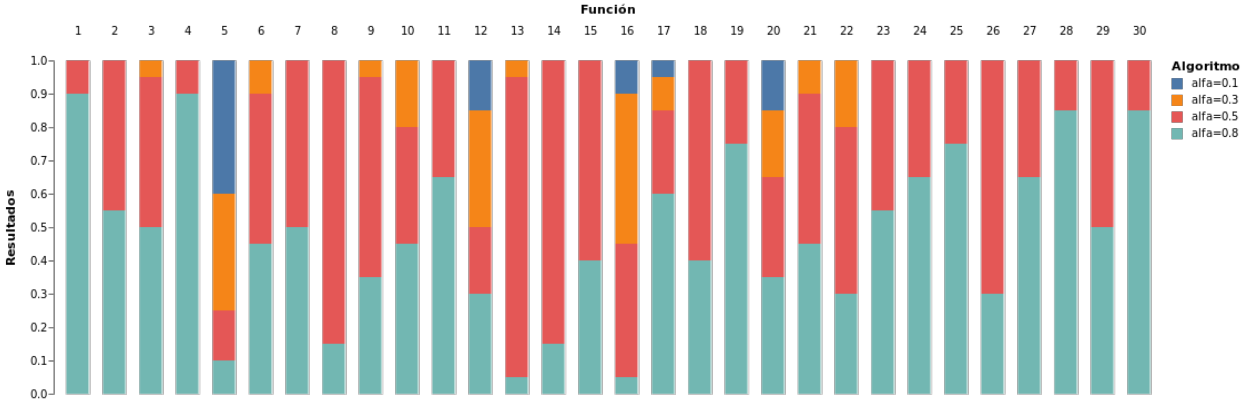
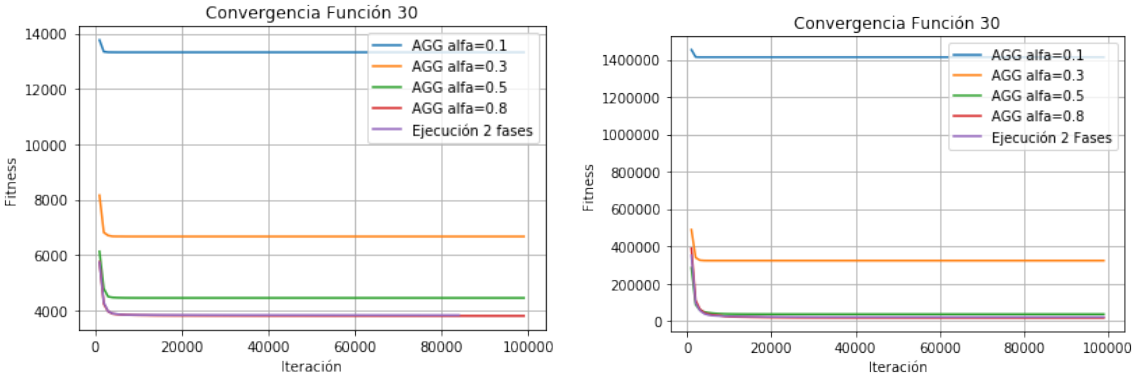


Figura 25: Estudio del comportamiento del Modelo de Ejecución en 2 fases con AGG, Dim=30

Por último, vamos a ver la gráfica de convergencia que se corresponde con la Función 30, una función en la que este modelo híbrido no ha dado un buen resultado para ninguna de las dos dimensiones. Podemos ver, que a pesar de no obtener los mejores resultados, aún así ha sido capaz de adaptarse y ser competitiva respecto al algoritmo que mejor funciona en esos casos concretos (aunque de nuevo, repetimos que, al existir tan poca diferencia, no se pueden sacar conclusiones definitivas de este hecho).

Sin embargo, si es necesario recalcar, que aunque no consiga el mejor resultado, la adaptación ha dado muy buenos resultados, ya que la ha acercado a igualarse al algoritmo con mejor resultado.



(a) Dimensión 10 (b) Dimensión 30

Figura 26: Convergencia Función 30 para el Modelo Ejecución en 2 Fases

Concluimos por tanto, cómo **teniendo en cuenta la necesidad de adaptación que hemos comentado anteriormente podemos obtener muy buenos resultados, y, efectivamente, favorecer a los algoritmos que se comportan mejor para cada función, y de esta forma aprovechar esa ventaja para obtener buenos resultados**.

Tabla 21: Comparativa Modelo Ejecución 2 fases *vs* AGGs, Dimensión 10

	Función	**MODELO 2**	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.8$
Funciones unimodales	**1**	2.7226e+06	2.6219e+07	1.9733e+07	1.3739e+07	1.7166e+06
	2	5.1190e+03	8.1579e+08	2.5251e+08	2.1115e+07	5.2363e+03
	3	5.5671e+03	2.0687e+04	2.0828e+04	1.3467e+04	1.7111e+04
Funciones multimodales Simples	**4**	2.1332e+01	2.5996e+02	9.5283e+01	4.1253e+01	2.8846e+01
	5	2.0156e+01	2.0493e+01	2.0417e+01	2.0398e+01	2.0543e+01
	6	2.2510e+00	6.3108e+00	4.0014e+00	3.3448e+00	3.6400e+00
	7	2.4374e-01	3.9984e+01	8.0940e+00	3.7040e-01	7.4475e-02
	8	9.5883e+00	2.5320e+01	1.4671e+01	1.2338e+01	1.6869e+01
	9	1.0661e+01	3.3245e+01	2.2132e+01	1.6095e+01	2.4467e+01
	10	2.2900e+02	9.7422e+02	5.1843e+02	3.3073e+02	3.8424e+02
	11	5.8306e+02	1.2316e+03	9.6166e+02	7.7194e+02	1.0637e+03
	12	2.9580e-01	9.7918e-01	8.5978e-01	8.4969e-01	1.1069e+00
	13	1.6512e-01	1.5488e+00	3.6344e-01	2.4493e-01	2.7126e-01
	14	3.0076e-01	7.5064e+00	1.0728e+00	4.5416e-01	4.3870e-01
	15	1.6529e+00	3.5108e+01	7.7786e+00	2.2581e+00	1.5979e+00
	16	3.0277e+00	3.5554e+00	3.3121e+00	3.2780e+00	3.5895e+00
Híbridas Función 1	**17**	1.0090e+05	5.7440e+05	5.8414e+05	5.2481e+05	1.5050e+05
	18	2.0763e+03	1.7757e+04	1.1313e+05	1.6438e+04	1.0794e+04
	19	1.7336e+00	9.1000e+00	4.6540e+00	2.6449e+00	2.4317e+00
	20	2.2786e+03	2.3066e+04	6.5040e+03	6.7841e+03	1.0622e+04
	21	6.7484e+03	1.9674e+05	2.1169e+05	1.0274e+05	1.5027e+04
Composición de Funciones	**22**	3.4480e+01	1.4471e+02	1.2290e+02	7.7622e+01	6.1099e+01
	23	3.2976e+02	3.5255e+02	3.3771e+02	3.3118e+02	3.3041e+02
	24	1.2489e+02	1.8107e+02	1.6198e+02	1.3911e+02	1.3759e+02
	25	1.8918e+02	1.9995e+02	1.9757e+02	1.9939e+02	1.9927e+02
	26	1.0016e+02	1.0234e+02	1.0027e+02	1.0025e+02	1.0028e+02
	27	2.2107e+02	4.0829e+02	3.6480e+02	3.3075e+02	3.1702e+02
	28	3.9716e+02	1.0151e+03	6.3040e+02	4.8659e+02	4.2352e+02
	29	5.5509e+02	3.0855e+06	2.8811e+05	4.1881e+04	1.5624e+05
	30	8.2542e+02	1.0322e+04	3.6690e+03	1.4501e+03	8.0089e+02
	Recuento	**25**	**0**	**0**	**0**	**5**
	Ranking	**1.133**	**4.733**	**3.166**	**1.7**	**2.1**

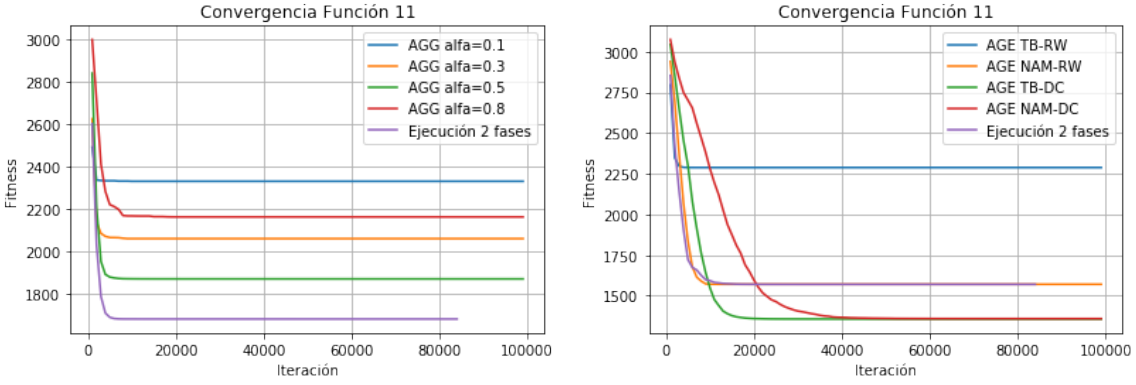
Tabla 22: Comparativa Modelo Ejecución 2 fases *vs* AGGs, Dimensión 30

	Función	MODELO 2	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.8$
Funciones unimodales	**1**	2.2576e+07	7.0858e+08	3.9686e+08	1.3579e+08	5.3139e+07
	2	4.8406e+08	3.1438e+10	1.2581e+10	1.4886e+09	1.3695e+09
	3	3.3825e+04	1.0312e+05	1.1226e+05	6.9242e+04	6.1670e+04
Funciones multimodales Simples	**4**	1.1629e+02	4.4602e+03	1.3222e+03	3.0729e+02	1.6931e+02
	5	2.0818e+01	2.1015e+01	2.0863e+01	2.0825e+01	2.1068e+01
	6	1.6396e+01	3.1775e+01	2.6539e+01	1.7874e+01	1.8268e+01
	7	7.7646e+00	3.4150e+02	1.2445e+02	1.9213e+01	6.1387e+00
	8	6.0486e+01	1.9882e+02	1.0496e+02	6.1901e+01	8.7885e+01
	9	8.9456e+01	2.2636e+02	1.4355e+02	9.1954e+01	1.7095e+02
	10	2.3830e+03	5.2923e+03	3.5439e+03	2.4715e+03	3.1230e+03
	11	3.5528e+03	6.5271e+03	5.0997e+03	4.5446e+03	5.5297e+03
	12	6.8698e-01	1.8463e+00	1.6757e+00	1.4926e+00	2.6792e+00
	13	4.9448e-01	4.9012e+00	3.0698e+00	4.8713e-01	7.8861e-01
	14	4.0220e-01	1.2928e+02	4.7509e+01	1.1406e+00	2.2999e+00
	15	2.4248e+02	4.0715e+04	1.4857e+04	2.6908e+02	6.3507e+04
	16	1.2669e+01	1.3385e+01	1.2896e+01	1.2813e+01	1.3488e+01
Híbridas Función 1	**17**	2.6496e+06	4.3504e+07	3.2020e+07	9.7566e+06	4.2351e+06
	18	6.0032e+04	6.0485e+08	1.2091e+08	4.8786e+06	6.8887e+03
	19	2.1213e+01	2.5172e+02	1.7235e+02	8.4923e+01	2.5122e+01
	20	3.7834e+04	2.2141e+05	1.3282e+05	6.3561e+04	5.0486e+04
	21	7.0658e+05	1.3408e+07	6.7215e+06	1.6784e+06	9.7467e+05
Composición de Funciones	**22**	4.6696e+02	1.2076e+03	7.9064e+02	5.7345e+02	6.7263e+02
	23	3.2100e+02	5.0559e+02	3.9989e+02	3.3350e+02	3.3037e+02
	24	2.4530e+02	2.7820e+02	2.6773e+02	2.4961e+02	2.5019e+02
	25	2.1043e+02	2.2843e+02	2.2852e+02	2.2382e+02	2.0986e+02
	26	1.0058e+02	1.8216e+02	1.4941e+02	1.1604e+02	1.1209e+02
	27	7.2813e+02	1.2427e+03	1.0208e+03	7.8389e+02	8.0668e+02
	28	1.0412e+03	6.2414e+03	3.3967e+03	1.4937e+03	1.1413e+03
	29	9.2152e+05	2.9792e+08	3.4274e+07	1.1281e+06	1.3776e+06
	30	1.7286e+04	1.4105e+06	3.2001e+05	3.2739e+04	1.3471e+04
	Recuento	**26**	**0**	**0**	**1**	**3**
	Ranking	**1.166**	**4.800**	**3.866**	**2.466**	**2.700**

6.2.2. Estudio del Modelo de Adaptación en 2 fases con los Algoritmos Genéticos Estacionarios

Como podemos ver en este caso, las conclusiones acerca del funcionamiento del algoritmo no se pueden sacar de una forma tan clara como ocurría con la versión anterior. Podemos ver en la tabla 23 cómo los resultados que obtiene el algoritmo están muy igualados con uno de los algoritmos genéticos estacionarios que involucra. Sin embargo, aunque en el Ranking podemos ver, que en este caso el Modelo de Ejecución en 2 fases es el que obtiene mejores resultados, la diferencia es tan mínima que nos impide saber si realmente esto es así, o si con un mayor número de ejecuciones a los modelos se podrían haber intercambiado las tornas.

Puede haber una gran cantidad de razones por las que el comportamiento en este caso sea diferente, al igual que habrá situaciones que sean diferentes y pueda no darse una explicación detallada. Sin embargo, a pesar de todo ello, hay un aspecto importante que distingue el comportamiento de estos algoritmos. Para visualizarlo, vamos a coger una función de ejemplo, que tiene un mal funcionamiento cuando utilizamos el modelo con algoritmos genéticos estacionarios, pero sí que funcione bien el modelo híbrido si utiliza algoritmos genéticos generacionales. Este caso es, por ejemplo, el de la `Función 11`. A continuación, en la selección de imágenes 28, podemos ver la convergencia de dicha función para ambas versiones del modelo.



(a) Versión con AGG,Dim 10

(b) Versión con AGE, Dim 10

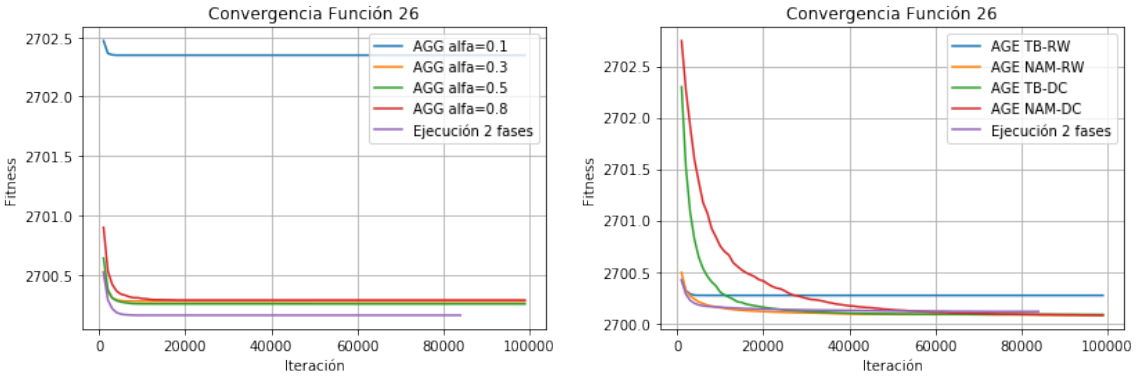
Figura 27: Convergencia Función 11 para el Modelo Ejecución en 2 Fases

Si comparamos detenidamente el comportamiento que se muestra en las dos imágenes 28, podemos sacar una conclusión clara, y es que en la versión estacionaria, el algoritmo converge de una forma más lenta. Esta lentitud, le lleva a que, cuando el modelo realice la adaptación, no sea capaz de determinar, a ese nivel tan temprano, cuál es el mejor algoritmo, y en su lugar se decante por una opción menos prometedora. Podemos ver que en la versión generacional, esto no ocurre, puesto que los algoritmos tienen un comportamiento en el que descienden muy rápido al principio y eso le permite determinar, con mayor precisión, la decisión a tomar.

Vemos por tanto que las características en cuanto a convergencia del algoritmo influyen de forma directa en el comportamiento de este modelo,

puesto que al estar establecida de forma fija la adaptación, es difícil determinar el algoritmo más prometedor de forma correcta.

Este comportamiento, se repite de forma reiterada en las funciones que estamos estudiando. A continuación, en la selección de imágenes 29, podemos ver cómo el hecho planteado se repite, dando lugar a las mismas consecuencias. En este caso, la función que se estudia es la `Función 26`.

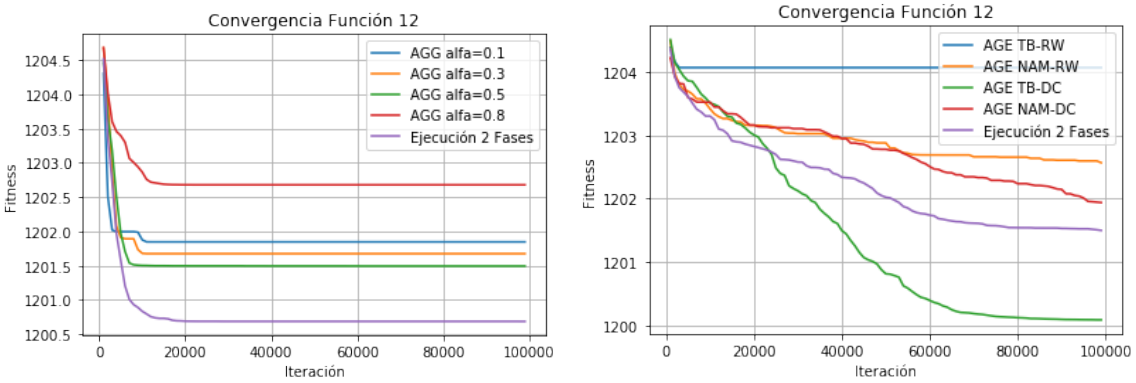


(a) Versión con AGG,Dim 10

(b) Versión con AGE, Dim 10

Figura 28: Convergencia Función 11 para el Modelo Ejecución en 2 Fases

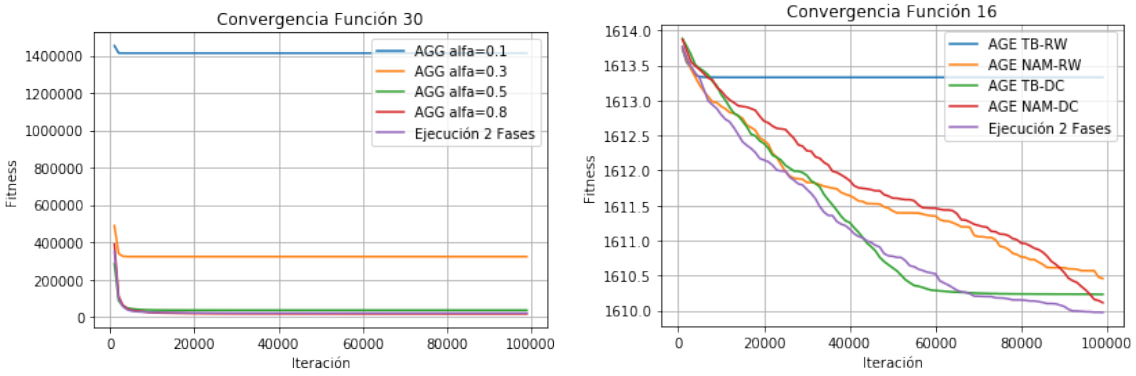
Sin embargo, en la tabla 24, podemos ver los resultados que se obtienen cuando utilizamos una dimensión de tamaño 30, y en este caso, sí que podemos ver que efectivamente no se está obteniendo un buen resultado con el modelo híbrido. Como podemos observar en las selecciones de imágenes 29 y 30, este hecho que anteriormente hemos mencionado, sigue dándose en esta dimensión, y como consecuencia, los resultados del modelo no son buenos.



(a) Versión con AGG,Dim 30

(b) Versión con AGE, Dim 30

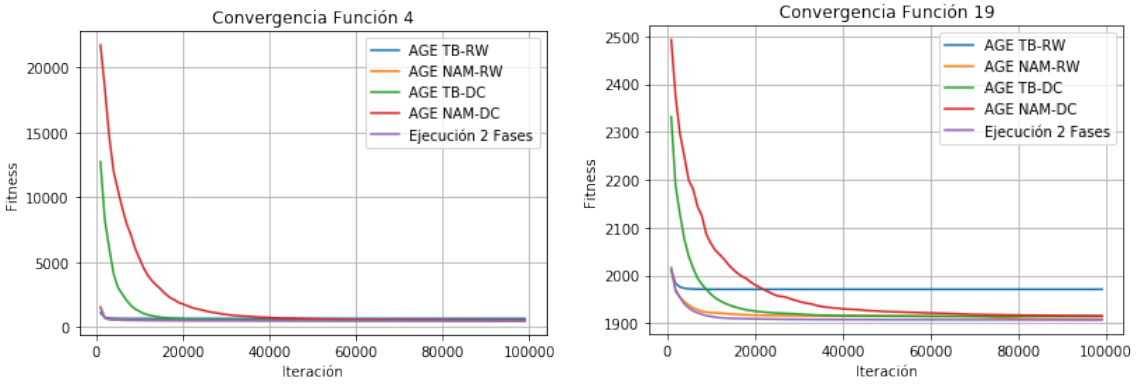
Figura 29: Convergencia Función 12 para el Modelo Ejecución en 2 Fases



(a) Versión con AGG,Dim 30 (b) Versión con AGE, Dim 30

Figura 30: Convergencia Función 16 para el Modelo Ejecución en 2 Fases

Es más, si observamos los resultados de la tabla 24, podemos observar cómo los algoritmos que realmente se ven beneficiados por la situación, son precisamente los algoritmos de versión estacionaria, que son los que ponen los impedimentos al modelo de Ejecución en Fases para que este fuese el algoritmo con mejores resultados. Sin embargo, esto no impediría, que si se detectase un algoritmo realmente bueno al principio, en una pronta iteración, este saliera victorioso en cuanto a resultados en el modelo híbrido. Por tanto,**cuando un algoritmo funciona realmente bien, y es detectado con anticipación, este modelo híbrido da resultados realmente buenos.** Este puede ser el caso de la Función 4 y la Función 19, cuya gráfica de convergencia se puede observar en la selección de imágenes 31



(a) Conv. Función 4, Dim 30 (b) Conv. Función 19, Dim 30

Figura 31: Convergencias Modelo Ejecución en 2 Fases

A pesar de estas claras deficiencias que muestra el algoritmo, que se ven reflejadas en los casos que hemos puesto gráficamente con su respectiva gráfica de convergencia, el Ranking de resultados de los algoritmos no deja al Modelo de Ejecución en 2 Fases en un mal lugar, sino que deja unos resultados muy igualados. **En nuestra opinión, es un modelo que da buenos resultados cuando el algoritmo es realmente bueno y converge rápidamente acorde al momento en el que**

se realiza la adaptación, y debe ser utilizado teniendo en cuenta dicho aspecto.

Tabla 23: Comparativa Modelo Ejecución 2 fases *vs* AGEs, Dimensión 10

	Función	MODELO 2	TB-RW	NAM-RW	TB-DC	NAM-DC
Funciones unimodales	**1**	1.4019e+06	1.5062e+07	4.5445e+06	9.1373e+04	1.2852e+05
	2	5.2433e+03	2.2385e+07	2.3344e+03	3.2162e+03	3.7183e+03
	3	3.9816e+03	1.3982e+04	5.1444e+03	2.9106e+03	2.2521e+03
Funciones Multimodales Simples	**4**	2.6378e+01	4.8196e+01	2.6471e+01	2.2436e+01	2.4316e+01
	5	1.8708e+01	2.0700e+01	1.8308e+01	1.9518e+01	1.8705e+01
	6	8.5966e-01	2.4887e+00	8.9220e-01	9.7225e-01	1.3493e+00
	7	2.2730e-02	1.0190e+00	2.7697e-02	5.9555e-02	6.7596e-02
	8	5.4470e+00	1.2589e+01	6.6274e+00	5.8903e+00	6.7062e+00
	9	8.0846e+00	1.6510e+01	9.3395e+00	8.8193e+00	8.5368e+00
	10	7.7479e+01	2.8852e+02	1.1994e+02	1.3100e+02	1.3971e+02
	11	4.7010e+02	1.1878e+03	4.7046e+02	2.5693e+02	2.5919e+02
	12	2.7543e-01	2.2869e+00	3.5016e-01	6.0007e-02	8.0529e-02
	13	1.1692e-01	2.9047e-01	7.5061e-02	9.4233e-02	7.1274e-02
	14	2.5970e-01	4.9721e-01	2.9625e-01	3.0278e-01	2.5294e-01
	15	1.1314e+00	2.8492e+00	1.0124e+00	8.5777e-01	7.6538e-01
	16	2.1069e+00	3.4910e+00	1.8494e+00	2.2432e+00	1.7438e+00
Híbridas Función 1	**17**	2.3466e+04	3.6888e+05	1.6802e+05	1.0080e+04	7.9785e+03
	18	3.8130e+03	8.3484e+03	6.4743e+03	1.1931e+04	1.1616e+04
	19	1.1479e+00	2.3462e+00	1.1198e+00	1.4295e+00	1.2619e+00
	20	1.6338e+03	1.0017e+04	6.7911e+03	4.9414e+03	4.0037e+03
	21	2.8548e+03	1.2894e+05	2.8723e+04	2.9665e+03	1.8785e+03
Composición de Funciones	**22**	1.4712e+01	6.6400e+01	3.7563e+01	3.3007e+01	3.7571e+01
	23	3.2949e+02	3.3298e+02	3.2945e+02	3.2945e+02	3.2945e+02
	24	1.1832e+02	1.4775e+02	1.2625e+02	1.1727e+02	1.1533e+02
	25	1.7219e+02	1.9781e+02	1.9253e+02	1.8447e+02	1.9253e+02
	26	1.0011e+02	1.0027e+02	1.0008e+02	1.0008e+02	1.0007e+02
	27	1.2146e+02	3.4281e+02	2.8240e+02	2.4831e+02	2.0931e+02
	28	4.0111e+02	4.5314e+02	4.1560e+02	4.2631e+02	4.5258e+02
	29	5.5474e+02	6.9891e+04	6.1174e+02	4.2500e+04	4.6736e+02
	30	7.3145e+02	1.2212e+03	8.0306e+02	9.5011e+02	9.1190e+02
	Recuento	**10**	**3**	**3**	**6**	**10**
	Ranking	**2.233**	**4.933**	**2.866**	**2.683**	**2.283**

Tabla 24: Comparativa Modelo Ejecución 2 fases *vs* AGEs, Dimensión 30

	Función	MODELO 2	AGG 0.5	AGG 0.8	TB-DC	NAM-DC
Funciones unimodales	**1**	3.0227e+06	1.1916e+08	1.4570e+06	1.4407e+06	2.6206e+06
	2	1.4876e+05	1.2052e+09	1.1501e+04	1.6902e+03	8.6171e+03
	3	1.9210e+04	8.1310e+04	2.1060e+04	3.0364e+03	5.0131e+03
Funciones multimodales Simples	**4**	7.4410e+01	2.6966e+02	8.9712e+01	9.3134e+01	1.1954e+02
	5	2.0918e+01	2.1192e+01	2.0909e+01	2.0789e+01	2.0912e+01
	6	7.6757e+00	1.7514e+01	7.8676e+00	9.2749e+00	7.5959e+00
	7	1.4230e-02	1.8105e+01	1.7852e-02	6.2063e-02	1.2408e-02
	8	2.8306e+01	5.9742e+01	2.7377e+01	4.3681e+01	3.9696e+01
	9	4.0370e+01	1.1170e+02	4.4058e+01	4.9968e+01	5.2751e+01
	10	9.2252e+02	3.0434e+03	1.0648e+03	8.3021e+02	9.2419e+02
	11	2.6320e+03	7.4797e+03	3.1212e+03	1.9024e+03	1.5268e+03
	12	3.9633e-01	4.0723e+00	2.0284e+00	7.8601e-02	5.2923e-02
	13	2.5468e-01	5.9248e-01	2.1503e-01	2.8563e-01	2.3721e-01
	14	3.1390e-01	2.3140e+00	3.0385e-01	3.9394e-01	3.9921e-01
	15	3.8618e+00	1.7903e+02	4.1669e+00	4.8115e+00	3.4723e+00
	16	9.3929e+00	1.3330e+01	9.3609e+00	1.0223e+01	9.2643e+00
Híbridas Función 1	**17**	4.9033e+05	1.1845e+07	7.9535e+05	3.0375e+05	4.4548e+05
	18	1.6625e+03	5.3168e+06	2.7573e+03	4.4653e+03	9.5783e+03
	19	6.7155e+00	7.1090e+01	7.0760e+00	1.3861e+01	1.3805e+01
	20	1.1638e+04	8.4253e+04	2.7974e+04	1.4078e+04	1.8457e+04
	21	2.8349e+05	2.3949e+06	3.5593e+05	1.4936e+05	1.5443e+05
Composición de Funciones	**22**	2.6967e+02	6.7019e+02	3.6703e+02	4.4510e+02	3.2607e+02
	23	3.1562e+02	3.2676e+02	3.1528e+02	3.1534e+02	3.1525e+02
	24	2.3677e+02	2.4959e+02	2.3721e+02	2.3053e+02	2.3249e+02
	25	2.1013e+02	2.2070e+02	2.1034e+02	2.0704e+02	2.0555e+02
	26	1.0027e+02	1.1628e+02	1.0531e+02	1.1027e+02	1.0025e+02
	27	4.4036e+02	7.5335e+02	4.8956e+02	5.3681e+02	4.7603e+02
	28	9.4241e+02	1.3151e+03	9.2209e+02	9.3646e+02	9.8801e+02
	29	2.4154e+03	1.4712e+06	4.3510e+05	1.6584e+03	4.5960e+05
	30	5.3866e+03	5.9619e+04	6.8678e+03	3.0010e+03	2.6889e+03
	Recuento	**7**	**0**	**4**	**9**	**10**
	Ranking	**2.400**	**5.000**	**2.800**	**2.530**	**2.266**

6.2.3. Estudio del Modelo de Adaptación en 2 fases con los mejores Algoritmos Genéticos resultantes

Por último aspecto relativo al Modelo Híbrido de Ejecución en 2 fases, vamos a analizar los resultados relativos a una versión heterogénea del mismo. En este caso, lo que vamos a estudiar, es cómo de efectivo es el modelo si los algoritmos que lo contienen son los dos mejores algoritmos genéticos generacionales y estacionarios respectivamente. Para poder sacar conclusiones hay varias cuestiones por subrayar.

1. Como vimos en el apartado anterior, una de las razones por las que el Modelo de Adaptación en 2 fases no era efectivo, era debido a las características de convergencia que mostraban los algoritmos que combinaba dicho modelo, y la falta de adaptación en función de la convergencia que realiza este algoritmo.

2. Tal y como se expuso en el Modelo de Islas, el algoritmo genético generacional tiene peores propiedades que la versión estacionaria, por lo que, aun cogiendo los mejores genéticos, existe una desigualdad grande entre ambas versiones. Como podemos observar en la tabla 25 (resultados para dimensión 10) y en la tabla 26 (resultados para dimensión 30), los peores resultados son los conseguidos por los algoritmos genéticos generacionales.

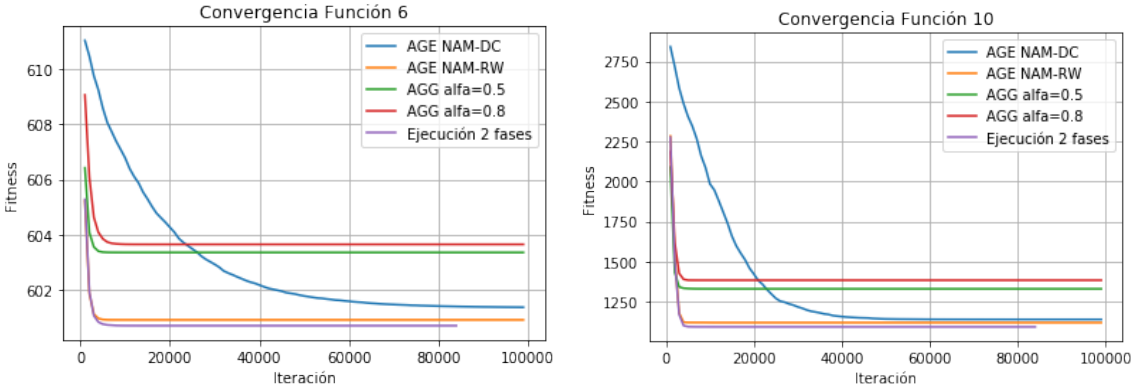
   Además, como vimos en la versión generacional del Modelo de Adaptación en 2 fases, éste solía superar a los algoritmos generacionales en su mayor parte, por lo que es esperar, que en los casos que sean buenos los algoritmos genéticos, el modelo de Islas lo superará en muchos de ellos.

   Por otra parte, vimos en la versión estacionaria, que en una gran cantidad de casos, los algoritmos genéticos estacionarios daban mejores resultados en comparación al Modelo de Adaptación en 2 fases en un gran número de veces.

   Una vez expuestas ambos aspectos, es de esperar que la cuestión vuelva a estar igualada entre los algoritmos estacionarios (fundamentalmente NAM-DC, al igual que antes) y el Modelo de Adaptación en 2 fases.

Respecto a los resultados de **Dimensión 10** (tabla 25), hay algunos bloques de cuestiones dignos de atención.

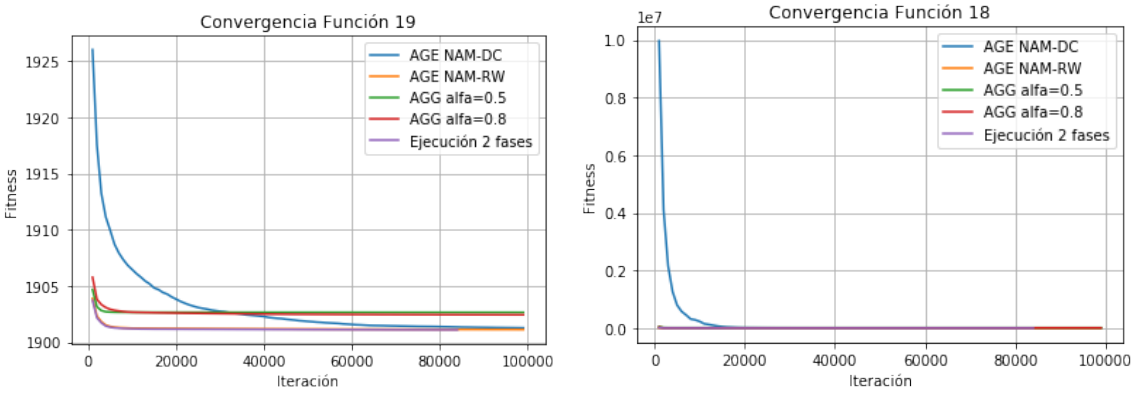
1. En primer lugar, las funciones 6 a 10, tienen realmente buenos resultados en esta dimensión. Este bloque tiene tan buenas características porque los resultados que observan en sus gráficas de convergencia, hay algoritmos que comienzan con resultados realmente buenos desde un principio, por tanto la adaptación se hace de una forma adecuada y así terminamos con buenas soluciones. A continuación en las imágenes de la selección de imágenes 32, podemos ver cómo realmente este comportamiento se ve reflejado en ellas. Se puede apreciar en ellas, cómo hay algoritmos que comienzan con soluciones realmente competitivas (inferiores al resto), y este hecho nos permite detectarlas y así poder conseguir buenos resultados.



(a) Función 6 (b) Función 10

Figura 32: Convergencia Función 6 y Función 10, Dimensión 10

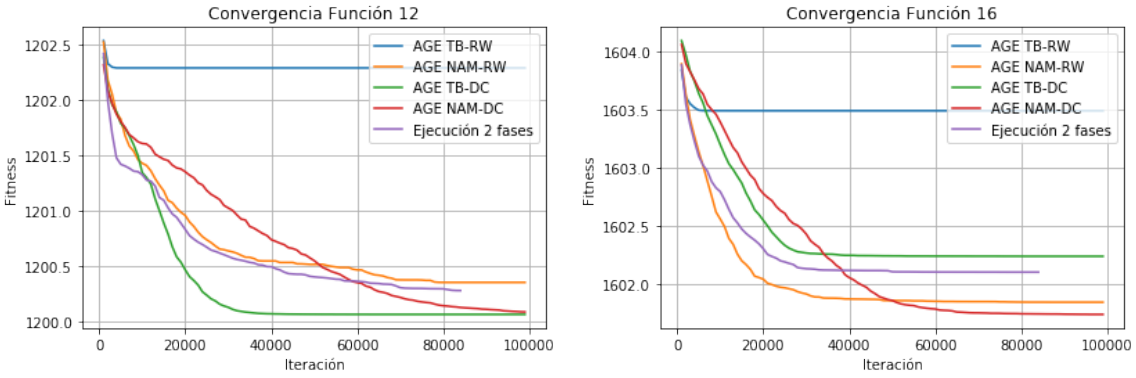
Este mismo efecto es el que ocurre en el grupo de las Funciones Híbridas, donde podemos ver, en la selección de imágenes 34 algunas de las gráficas de convergencia que se generan.



(a) Función 19 (b) Función 20

Figura 33: Convergencia Función 19 y Función 20, Dimensión 10

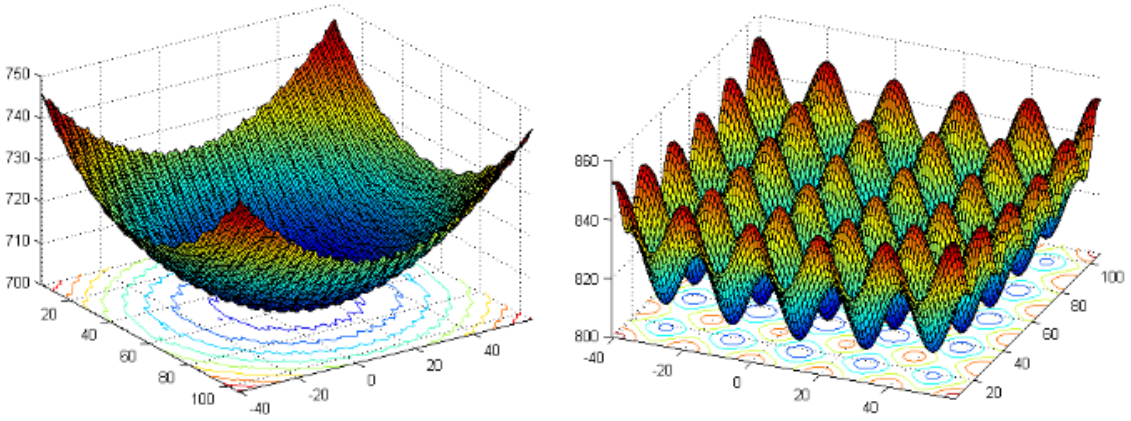
2. En segundo lugar, con el bloque de funciones que va desde la `Función 11` a la `Función 17`, ocurre exactamente lo contrario. En estos casos, donde la convergencia no se produce rápidamente y no es fácil detectar el algoritmo que mejor se comportará en un futuro, podemos ver que los resultados dejan ver, de nuevo, que el Modelo de Ejecución en 2 fases no es el más apropiado en este caso. Podemos ver este comportamiento reflejado en las imágenes 34a y 34b



(a) Función 12 (b) Función 16

Figura 34: Convergencia Función 12 y Función 16, Dimensión 10

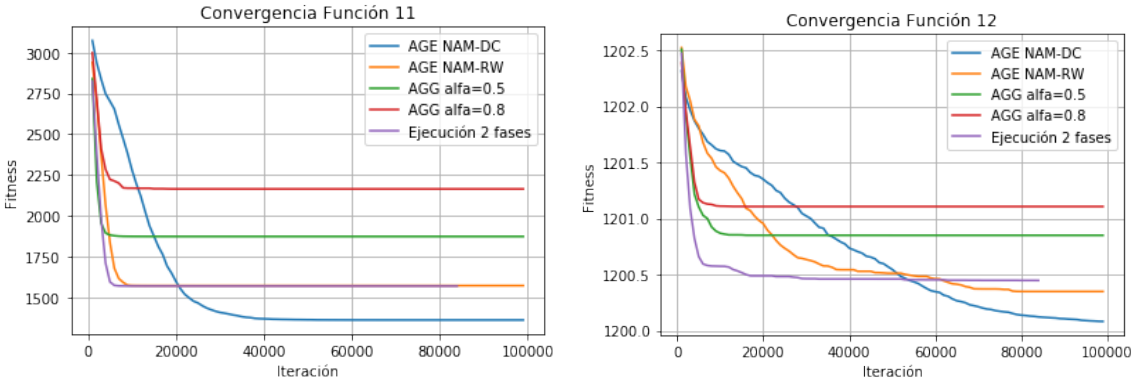
Esto se debe en parte, al tipo de zona de búsqueda que representan estas funciones. Ya vimos en la selección de imágenes 16 la forma que tienen las funciones 6 o 9. Vamos a mostrar en esta ocasión, otras funciones del mismo grupo. En este caso, se trata de la Función 7 y la Función 8, y como podemos ver en la selección de imágenes 35, la gran cantidad de máximos y mínimos contenidos en ellas, hacen que las propiedades de exploración de los algoritmos estacionarios con los que estamos trabajando den resultados muy potenciales en esta situación.



(a) Función 7 (b) Función 8

Figura 35: Representación Función 7 y Función 8

Del mismo modo, el escape de mínimos locales puede ser una de las consecuencias por las que la convergencia es más lenta y tiene pequeñas paradas en el camino, y como ya vimos en el apartado anterior, cuando un algoritmo que funciona bien, tarda en converger, no se ve representado por el Modelo de Ejecución en 2 Fases. Podemos ver este efecto en las gráficas de convergencia correspondientes a la Función 11 y a la Función 12 en la selección de imágenes 36

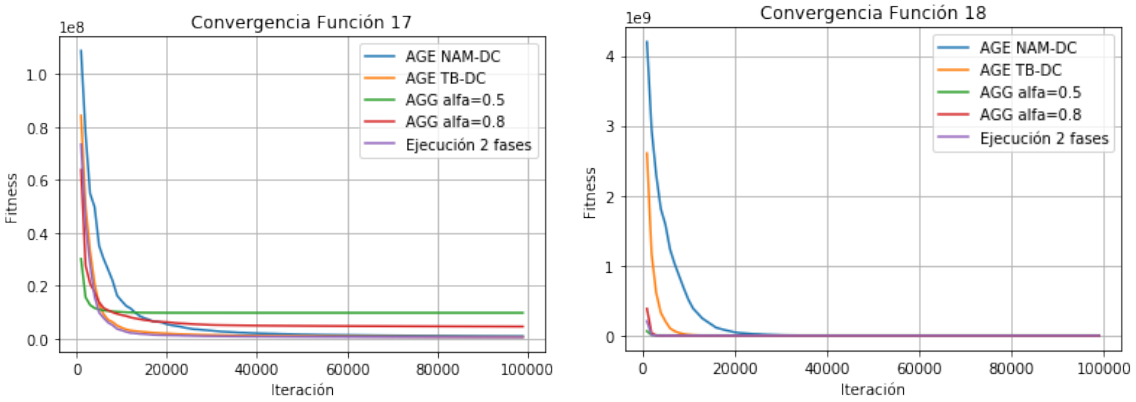


(a) Función 11 (b) Función 12

Figura 36: Convergencia Función 11 y Función 12, Dim 10

Por último, respecto a los resultados en **Dimensión 30** (tabla 26), destacamos también algunos grupos de funciones.

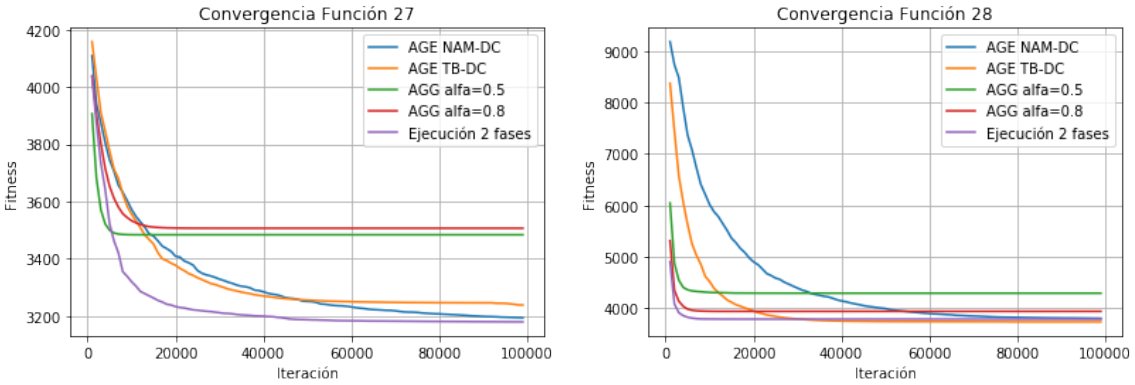
1. De nuevo, las híbridas forman un grupo con características similares, donde los resultados del Modelo de Ejecución en 2 Fases son, los mejores en un 60 % de las veces, mientras que, en las ocasiones que no resulta ganador, sus resultados son muy competitivos. De nuevo, este comportamiento se debe a que hay un algoritmo (o más de uno), que da resultados muy competitivos desde el principio de la ejecución, lo que le permite adaptarse de forma óptima. Lo observamos en la selección de figuras 37.



(a) Función 17 (b) Función 18

Figura 37: Convergencia Función 17 y Función 18, Dim 30

2. Por otra parte, hay una cierta tendencia en la Composición de Funciones a dar resultados muy favorecedores en las funciones individuales. De nuevo, esto se debe a la misma conclusión que venimos comentando anteriormente: los buenos resultados que dan inicialmente algunos algoritmos. Lo observamos en la selección de figuras 38.



(a) Función 27

(b) Función 28

Figura 38: Convergencia Función 27 y Función 28, Dim 30

Tabla 25: Comparativa Modelo Ejecución 2 fases *vs* mejores AGs, Dimensión 10

	Función	MODELO 2	AGG 0.5	AGG 0.8	NAM-RW	NAM-DC
Funciones unimodales	**1**	1.7231e+06	1.3739e+07	1.7166e+06	4.5445e+06	1.2852e+05
	2	1.6996e+03	2.1115e+07	5.2363e+03	2.3344e+03	3.7183e+03
	3	2.7828e+03	1.3467e+04	1.7111e+04	5.1444e+03	2.2521e+03
Funciones Multimodales Simples	**4**	1.8674e+01	4.1253e+01	2.8846e+01	2.6471e+01	2.4316e+01
	5	1.9127e+01	2.0398e+01	2.0543e+01	1.8308e+01	1.8705e+01
	6	6.8201e-01	3.3448e+00	3.6400e+00	8.9220e-01	1.3493e+00
	7	2.6540e-02	3.7040e-01	7.4475e-02	2.7697e-02	6.7596e-02
	8	5.7764e+00	1.2338e+01	1.6869e+01	6.6274e+00	6.7062e+00
	9	6.8298e+00	1.6095e+01	2.4467e+01	9.3395e+00	8.5368e+00
	10	9.3402e+01	3.3073e+02	3.8424e+02	1.1994e+02	1.3971e+02
	11	4.6752e+02	7.7194e+02	1.0637e+03	4.7046e+02	2.5919e+02
	12	4.4881e-01	8.4969e-01	1.1069e+00	3.5016e-01	8.0529e-02
	13	1.0874e-01	2.4493e-01	2.7126e-01	7.5061e-02	7.1274e-02
	14	2.9991e-01	4.5416e-01	4.3870e-01	2.9625e-01	2.5294e-01
	15	1.0292e+00	2.2581e+00	1.5979e+00	1.0124e+00	7.6538e-01
	16	2.0766e+00	3.2780e+00	3.5895e+00	1.8494e+00	1.7438e+00
Híbridas Función 1	**17**	1.8296e+04	5.2481e+05	1.5050e+05	1.6802e+05	7.9785e+03
	18	3.3972e+03	1.6438e+04	1.0794e+04	6.4743e+03	1.1616e+04
	19	1.0923e+00	2.6449e+00	2.4317e+00	1.1198e+00	1.2619e+00
	20	2.0219e+03	6.7841e+03	1.0622e+04	6.7911e+03	4.0037e+03
	21	3.6774e+03	1.0274e+05	1.5027e+04	2.8723e+04	1.8785e+03
Composición de Funciones	**22**	2.1894e+01	7.7622e+01	6.1099e+01	3.7563e+01	3.7571e+01
	23	3.2947e+02	3.3118e+02	3.3041e+02	3.2945e+02	3.2945e+02
	24	1.1645e+02	1.3911e+02	1.3759e+02	1.2625e+02	1.1533e+02
	25	1.7771e+02	1.9939e+02	1.9927e+02	1.9253e+02	1.9253e+02
	26	1.0011e+02	1.0025e+02	1.0028e+02	1.0008e+02	1.0007e+02
	27	1.2815e+02	3.3075e+02	3.1702e+02	2.8240e+02	2.0931e+02
	28	3.9276e+02	4.8659e+02	4.2352e+02	4.1560e+02	4.5258e+02
	29	5.1985e+02	4.1881e+04	1.5624e+05	6.1174e+02	4.6736e+02
	30	6.7396e+02	1.4501e+03	8.0089e+02	8.0306e+02	9.1190e+02
	Recuento	**15**	**0**	**0**	**2**	**14**
	Ranking	**1.800**	**4.530**	**4.166**	**2.500**	**2.000**

Tabla 26: Comparativa Modelo Ejecución 2 fases *vs* mejores AGs, Dimensión 30

	Función	MODELO 2	AGG 0.5	AGG 0.8	TB-DC	NAM-DC
Funciones unimodales	**1**	1.5295e+07	1.3579e+08	5.3139e+07	1.4407e+06	2.6206e+06
	2	3.3774e+08	1.4886e+09	1.3695e+09	1.6902e+03	8.6171e+03
	3	1.9212e+04	6.9242e+04	6.1670e+04	3.0364e+03	5.0131e+03
Funciones multimodales Simples	**4**	1.1723e+02	3.0729e+02	1.6931e+02	9.3134e+01	1.1954e+02
	5	2.0899e+01	2.0825e+01	2.1068e+01	2.0789e+01	2.0912e+01
	6	1.4156e+01	1.7874e+01	1.8268e+01	9.2749e+00	7.5959e+00
	7	6.8299e+00	1.9213e+01	6.1387e+00	6.2063e-02	1.2408e-02
	8	6.7272e+01	6.1901e+01	8.7885e+01	4.3681e+01	3.9696e+01
	9	1.0268e+02	9.1954e+01	1.7095e+02	4.9968e+01	5.2751e+01
	10	1.9526e+03	2.4715e+03	3.1230e+03	8.3021e+02	9.2419e+02
	11	2.7614e+03	4.5446e+03	5.5297e+03	1.9024e+03	1.5268e+03
	12	7.8742e-01	1.4926e+00	2.6792e+00	7.8601e-02	5.2923e-02
	13	4.4471e-01	4.8713e-01	7.8861e-01	2.8563e-01	2.3721e-01
	14	4.4572e-01	1.1406e+00	2.2999e+00	3.9394e-01	3.9921e-01
	15	9.1687e+01	2.6908e+02	6.3507e+04	4.8115e+00	3.4723e+00
	16	1.1137e+01	1.2813e+01	1.3488e+01	1.0223e+01	9.2643e+00
Híbridas Función 1	**17**	2.9300e+05	9.7566e+06	4.2351e+06	3.0375e+05	4.4548e+05
	18	3.4957e+03	4.8786e+06	6.8887e+03	4.4653e+03	9.5783e+03
	19	1.0706e+01	8.4923e+01	2.5122e+01	1.3861e+01	1.3805e+01
	20	1.5206e+04	6.3561e+04	5.0486e+04	1.4078e+04	1.8457e+04
	21	1.5842e+05	1.6784e+06	9.7467e+05	1.4936e+05	1.5443e+05
Composición de Funciones	**22**	3.8138e+02	5.7345e+02	6.7263e+02	4.4510e+02	3.2607e+02
	23	3.1936e+02	3.3350e+02	3.3037e+02	3.1534e+02	3.1525e+02
	24	2.4323e+02	2.4961e+02	2.5019e+02	2.3053e+02	2.3249e+02
	25	2.0660e+02	2.2382e+02	2.0986e+02	2.0704e+02	2.0555e+02
	26	1.0054e+02	1.1604e+02	1.1209e+02	1.1027e+02	1.0025e+02
	27	4.7673e+02	7.8389e+02	8.0668e+02	5.3681e+02	4.7603e+02
	28	9.8953e+02	1.4937e+03	1.1413e+03	9.3646e+02	9.8801e+02
	29	1.1302e+04	1.1281e+06	1.3776e+06	1.6584e+03	4.5960e+05
	30	1.0923e+04	3.2739e+04	1.3471e+04	3.0010e+03	2.6889e+03
	Recuento	**3**	**0**	**0**	**13**	**14**
	Ranking	**2.666**	**4.366**	**4.433**	**1.733**	**1.800**

6.3. Modelo Híbrido con Probabilidades Adaptativas

A continuación vamos a observar los resultados que hemos obtenido para el tercer modelo híbrido implementado, el cuál se denomina *Algoritmo Híbrido con Probabilidades Adaptativas.*

Debido a la altísima similitud que existe en los resultados obtenidos para Dimensión 10 y Dimensión 30, nos vamos a limitar a compararlos de forma simultánea.

6.3.1. Estudio del Modelo de Probabilidades Adaptativas con Algoritmos Genéticos Generacionales

Se pueden apreciar los resultados obtenidos para este modelo en la tabla 27 (para Dimensión 10) y en la tabla 28 (para Dimensión 30). De los datos disponibles, podemos sacar las siguientes conclusiones.

1. El Modelo Híbrido con Probabilidades Adaptativas no muestra un buen comportamiento en vista a los resultados. Como se puede observar, consultando tanto el Recuento, como su posición en el Ranking, las posiciones obtenidas no son nada buenas en comparación a como actúan los algoritmos genéticos de manera individual. Podemos apreciar en estas tablas la superioridad del Algoritmo Genético con un valor de $\alpha = 0.5$ y $\alpha = 0.8$. Este hecho, ya lo hemos podido ir viendo en situaciones anteriores, y justificándolo con otros resultados, pero con una diferencia notable. En este caso, los resultados del Modelo Híbrido, además de ser malos, están muy separados en cuanto a puntuación a estos mejores algoritmos genéticos, por lo que podemos concluir que el método, definitivamente, no funciona bien.
2. **A pesar de los malos resultados, hay que mencionar que la adaptación que se lleva a cabo surge un mínimo efecto**, por dos razones fundamentales:
   a) No es el peor de los algoritmos en cuanto a funcionamiento. Si consultamos el valor correspondiente en la fila Ranking, tanto en la tabla 27 como en la tabla 28 queda por encima de la última posición. De hecho, como podemos observar en los resultados de dimensión 30, el Modelo de Probabilidades Adaptativas terminas siendo el tercer algoritmo con mejores resultados del total, por lo que es obvio que la adaptación realizada internamente está mejorando resultados.
   b) En una gran cantidad de funciones, podemos ver cómo el resultado conseguido por el Modelo de Probabilidades Adaptativas no se aleja mucho de los mejores alcanzados. Por tanto, **a pesar de que internamente, hay algo que está impidiendo aspirar a mejores soluciones, se observa una mejoría y un intento de alcanzar a los algoritmos más exitosos en cuanto a resultados.**

Para ilustrar el comportamiento del Modelo, vamos a ver, algunas de las gráficas de convergencia, y así poder analizar mejor qué está ocurriendo.

En primer lugar vamos a ver la gráfica de convergencia del único caso en el que este algoritmo ha conseguido superar (por poca diferencia), al resto de algoritmos de la comparativa. Podemos ver en la tabla 27, que se trata de la `Función 25`, ejecutada con soluciones de Dimensión 10. En la imagen 39. En ella podemos observar el comportamiento ideal para este tipo de modelos, aunque por desgracia, no es el más recurrente, sino que en el análisis que llevamos a cabo, supone la excepción. En dicha imagen, podemos observar cómo, cuando en una determinada iteración, se detecta que las soluciones van por otro camino, esta función adopta el comportamiento más adecuado. De esta forma, aprovecha la información disponible, que puede consultar de sus algoritmos internos, para adaptarse en función de los resultados y experiencia que están logrando estos.

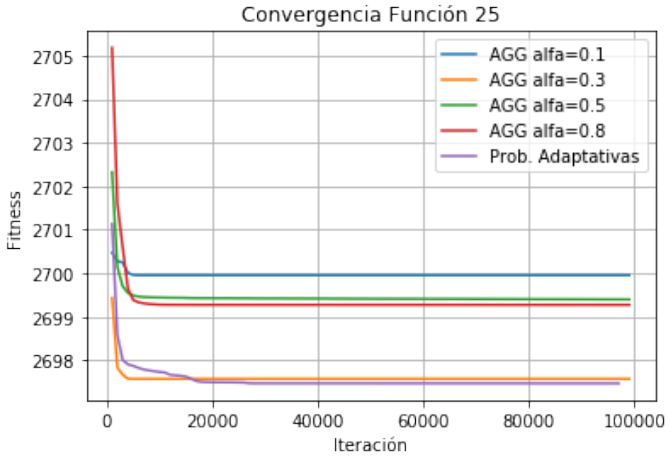
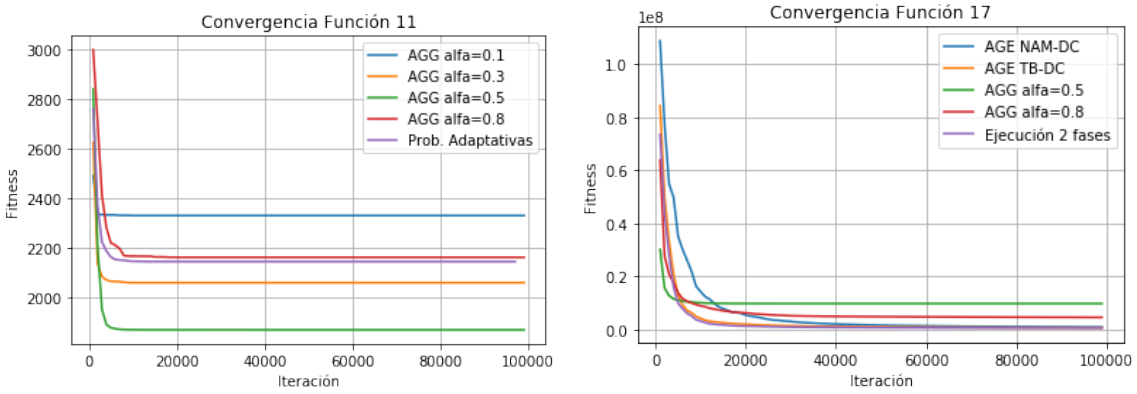


Figura 39: Convergencia Función 25, Dim=10

Sin embargo, tal y como hemos comentado, este hecho es la excepción, y es que como podemos observar en la selección de imágenes 40, realmente el Modelo de Probabilidades Adaptativas no consigue grandes resultados.



(a) Función 11 (b) Función 17

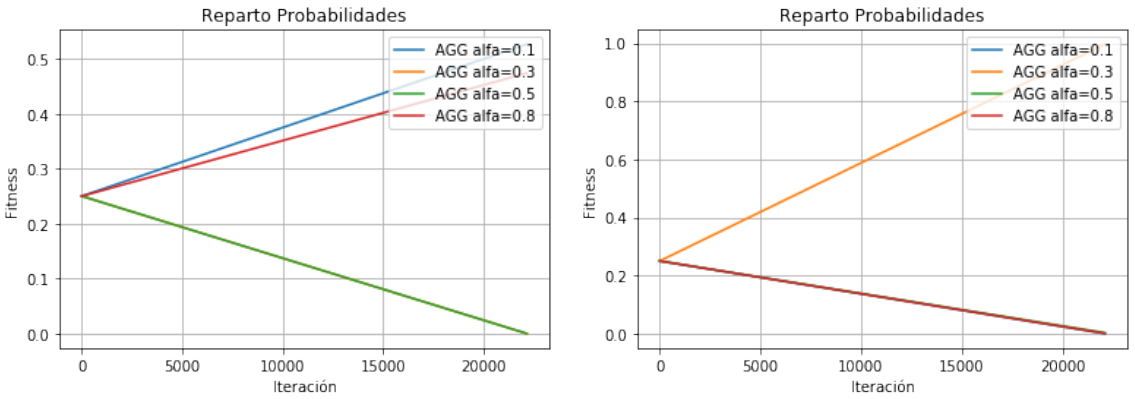
Figura 40: Convergencia Función 11 y Función 17, Dim 10

Esta falta de buenos resultados se debe principalmente a la rápida convergencia que

sufren los algoritmos utilizados. Si nos fijamos en las imágenes anteriores, podemos ver que a partir de una iteración muy temprana, los algoritmos dejan de producir mejora. La mejora, es la base principal para la adaptación del modelo, y, si no se produce ningún tipo de mejora, el algoritmo proporcionará la totalidad de las ejecuciones al último algoritmo que sí le proporcionó algún tipo de mejoría. Como resultado, tenemos el comportamiento que podemos observar en las gráficas de convergencia anteriores, donde se aprecia que el Modelo de Probabilidades Adaptativas también se ve estancado como consecuencia de sus integrantes.

Con dicho comportamiento, al final lo que se consigue es que el algoritmo que el Modelo de Probabilidades Adaptativas esté ejecutando cuando el resto de algoritmos se estanque, probablemente sea el único que vuelva a ejecutarse el resto del tiempo, pues es el último en adquirir algún tipo de mejora, por muy lejana (en cuanto a iteraciones pasadas) que se produjese la misma. Si cualquier algoritmo puede adquirir la totalidad de las ejecuciones, al final lo que estamos consiguiendo es un equilibrio en cuanto a protagonismo de los algoritmos, ya que, en las distintas ejecuciones que se realicen para probar el modelo, no coincidirá siempre el mejor algoritmo, sino que en gran parte de los casos se deberá al azar y como resultado, al final tendremos un reparto equitativo. Precisamente, **este equilibrio es lo que intentamos evitar con los modelos que involucran adaptación en tiempo de ejecución, por lo que podemos confirmar que este modelo no funciona de manera adecuada en los casos que hemos estudiado.**

A continuación, en la selección de imágenes 41, podemos ver el reparto de la probabilidad asociada durante la ejecución de la Función 1, en dos ejecuciones diferentes. Como podemos observar, el hecho que estábamos comentando se ve totalmente reflejado, ya que no se refleja, en la evolución de la gráfica, atisbos de adaptación, sino que como podemos ver, comienza con un equilibro, pero rápidamente hay un algoritmo que se despliega y se atribuye todas las ejecuciones, dejando al resto de algoritmos inutilizados.



(a) Función 1, Ejecución I

(b) Función 1, Ejecución II

Figura 41: Reparto de las probabilidades en ejecución, Función 1, Dim 10

Es más, si ahondamos un poco en la interpretación de la selección de imágenes 41, podemos ver dos cosas:

1. El algoritmo que consigue los mejores resultados para esa función, no tiene una participación que sea de acuerdo a ello.
2. En cada uno de los ejemplos, se atribuyen mayores probabilidades a algoritmos diferentes.

Los dos aspectos mencionados, pueden dar una explicación al equilibrio en la participación de los algoritmos, ya que no se aprecia ninguna tendencia clara al respecto. En la imagen 42 podemos ver el hecho comentado. Esta gráfica representa el valor medio de las veces que se ha ejecutado cada uno de los algoritmos de forma interna en el Modelo de Probabilidades Adaptativas.

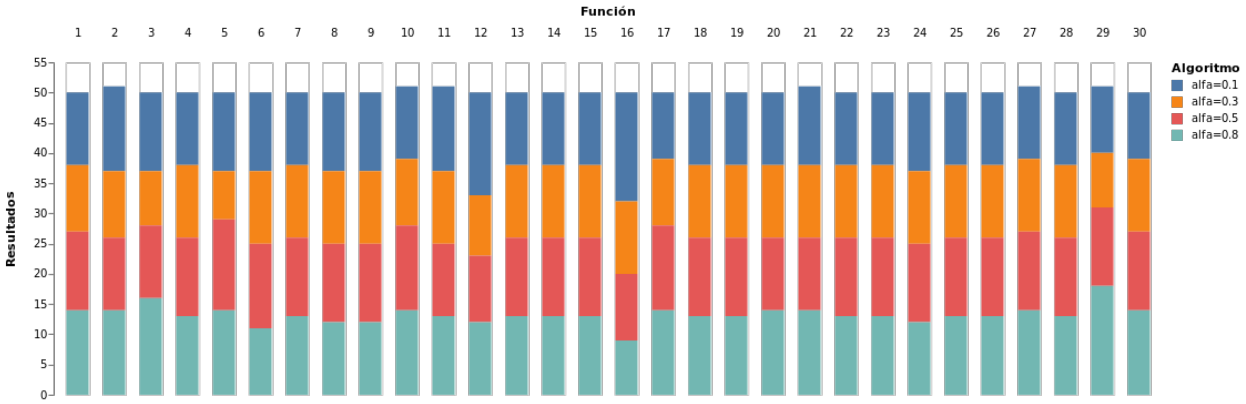


Figura 42: Reparto Ejecuciones en el Modelo para AGG, Dim=10

Tabla 27: Comparativa Modelo Probabilidades Adaptativas *vs* AGG, Dimensión 10

	Función	**Modelo 3**	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.8$
Funciones unimodales	**1**	2.0392e+07	2.6219e+07	1.9733e+07	1.3739e+07	1.7166e+06
	2	2.6328e+08	8.1579e+08	2.5251e+08	2.1115e+07	5.2363e+03
	3	1.5530e+04	2.0687e+04	2.0828e+04	1.3467e+04	1.7111e+04
Funciones Multimodales Simples	**4**	8.6025e+01	2.5996e+02	9.5283e+01	4.1253e+01	2.8846e+01
	5	2.0414e+01	2.0493e+01	2.0417e+01	2.0398e+01	2.0543e+01
	6	4.4434e+00	6.3108e+00	4.0014e+00	3.3448e+00	3.6400e+00
	7	9.4404e+00	3.9984e+01	8.0940e+00	3.7040e-01	7.4475e-02
	8	1.3647e+01	2.5320e+01	1.4671e+01	1.2338e+01	1.6869e+01
	9	2.4704e+01	3.3245e+01	2.2132e+01	1.6095e+01	2.4467e+01
	10	5.1954e+02	9.7422e+02	5.1843e+02	3.3073e+02	3.8424e+02
	11	1.0465e+03	1.2316e+03	9.6166e+02	7.7194e+02	1.0637e+03
	12	9.6813e-01	9.7918e-01	8.5978e-01	8.4969e-01	1.1069e+00
	13	4.5885e-01	1.5488e+00	3.6344e-01	2.4493e-01	2.7126e-01
	14	2.0710e+00	7.5064e+00	1.0728e+00	4.5416e-01	4.3870e-01
	15	1.2973e+01	3.5108e+01	7.7786e+00	2.2581e+00	1.5979e+00
	16	3.4046e+00	3.5554e+00	3.3121e+00	3.2780e+00	3.5895e+00
Híbridas Función 1	**17**	4.9446e+05	5.7440e+05	5.8414e+05	5.2481e+05	1.5050e+05
	18	3.0441e+05	1.7757e+04	1.1313e+05	1.6438e+04	1.0794e+04
	19	4.6037e+00	9.1000e+00	4.6540e+00	2.6449e+00	2.4317e+00
	20	9.0288e+03	2.3066e+04	6.5040e+03	6.7841e+03	1.0622e+04
	21	1.1892e+05	1.9674e+05	2.1169e+05	1.0274e+05	1.5027e+04
Composición de Funciones	**22**	9.8799e+01	1.4471e+02	1.2290e+02	7.7622e+01	6.1099e+01
	23	3.3657e+02	3.5255e+02	3.3771e+02	3.3118e+02	3.3041e+02
	24	1.6273e+02	1.8107e+02	1.6198e+02	1.3911e+02	1.3759e+02
	25	1.9746e+02	1.9995e+02	1.9757e+02	1.9939e+02	1.9927e+02
	26	1.0037e+02	1.0234e+02	1.0027e+02	1.0025e+02	1.0028e+02
	27	3.6467e+02	4.0829e+02	3.6480e+02	3.3075e+02	3.1702e+02
	28	5.7886e+02	1.0151e+03	6.3040e+02	4.8659e+02	4.2352e+02
	29	1.1448e+06	3.0855e+06	2.8811e+05	4.1881e+04	1.5624e+05
	30	4.4203e+03	1.0322e+04	3.6690e+03	1.4501e+03	8.0089e+02
	Recuento	**1**	**0**	**1**	**13**	**15**
	Ranking	**3.300**	**4.733**	**3.166**	**1.700**	**2.100**

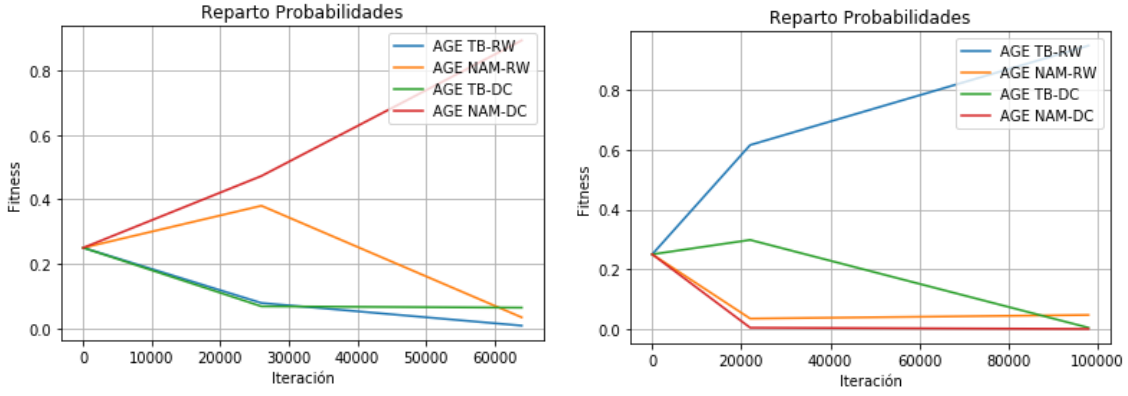
Tabla 28: Comparativa Modelo Probabilidades Adaptativas *vs* AGG, Dimensión 30

	Función	**Modelo 3**	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.8$
Funciones unimodales	**1**	2.2747e+08	7.0858e+08	3.9686e+08	1.3579e+08	5.3139e+07
	2	7.0254e+09	3.1438e+10	1.2581e+10	1.4886e+09	1.3695e+09
	3	7.3947e+04	1.0312e+05	1.1226e+05	6.9242e+04	6.1670e+04
Funciones multimodales Simples	**4**	1.1709e+03	4.4602e+03	1.3222e+03	3.0729e+02	1.6931e+02
	5	2.0938e+01	2.1015e+01	2.0863e+01	2.0825e+01	2.1068e+01
	6	2.3216e+01	3.1775e+01	2.6539e+01	1.7874e+01	1.8268e+01
	7	8.4085e+01	3.4150e+02	1.2445e+02	1.9213e+01	6.1387e+00
	8	1.0737e+02	1.9882e+02	1.0496e+02	6.1901e+01	8.7885e+01
	9	1.3908e+02	2.2636e+02	1.4355e+02	9.1954e+01	1.7095e+02
	10	3.3953e+03	5.2923e+03	3.5439e+03	2.4715e+03	3.1230e+03
	11	5.2612e+03	6.5271e+03	5.0997e+03	4.5446e+03	5.5297e+03
	12	2.0002e+00	1.8463e+00	1.6757e+00	1.4926e+00	2.6792e+00
	13	1.7247e+00	4.9012e+00	3.0698e+00	4.8713e-01	7.8861e-01
	14	2.6848e+01	1.2928e+02	4.7509e+01	1.1406e+00	2.2999e+00
	15	4.6087e+03	4.0715e+04	1.4857e+04	2.6908e+02	6.3507e+04
	16	1.3033e+01	1.3385e+01	1.2896e+01	1.2813e+01	1.3488e+01
Híbridas Función 1	**17**	1.7936e+07	4.3504e+07	3.2020e+07	9.7566e+06	4.2351e+06
	18	7.6977e+07	6.0485e+08	1.2091e+08	4.8786e+06	6.8887e+03
	19	1.0567e+02	2.5172e+02	1.7235e+02	8.4923e+01	2.5122e+01
	20	7.2387e+04	2.2141e+05	1.3282e+05	6.3561e+04	5.0486e+04
	21	4.2400e+06	1.3408e+07	6.7215e+06	1.6784e+06	9.7467e+05
Composición de Funciones	**22**	8.4628e+02	1.2076e+03	7.9064e+02	5.7345e+02	6.7263e+02
	23	3.6163e+02	5.0559e+02	3.9989e+02	3.3350e+02	3.3037e+02
	24	2.5852e+02	2.7820e+02	2.6773e+02	2.4961e+02	2.5019e+02
	25	2.2464e+02	2.2843e+02	2.2852e+02	2.2382e+02	2.0986e+02
	26	1.2663e+02	1.8216e+02	1.4941e+02	1.1604e+02	1.1209e+02
	27	9.4192e+02	1.2427e+03	1.0208e+03	7.8389e+02	8.0668e+02
	28	2.4785e+03	6.2414e+03	3.3967e+03	1.4937e+03	1.1413e+03
	29	4.6336e+07	2.9792e+08	3.4274e+07	1.1281e+06	1.3776e+06
	30	2.2552e+05	1.4105e+06	3.2001e+05	3.2739e+04	1.3471e+04
	Recuento	**0**	**0**	**0**	**15**	**15**
	Ranking	**3.006**	**4.766**	**3.633**	**1.500**	**2.033**

6.3.2. Estudio del Modelo de Probabilidades Adaptativas con Algoritmos Genéticos Estacionarios

Como podemos comprobar, en este caso se repiten todas las conclusiones que hemos obtenido anteriormente para la versión generacional del Modelo con Probabilidades Adaptativas. En este caso sin embargo, ni en Dimensión 10 (tabla 29) como en Dimensión 30 (tabla 30 existe una sola función en la que el modelo híbrido de mejor resultados, lo que es totalmente coherente, ya que como hemos venido viendo hasta ahora, la versión estacionaria es mucho más potente que la generacional.

De nuevo, si estudiamos el comportamiento interno que se da en el reparto de probabilidades que define la participación de cada uno de los algoritmos que forman el modelo, vemos de nuevo que no se está llevando una adaptación propia del modelo. En la selección de imágenes 43 podemos observar claramente este hecho. En estas imágenes, estamos viendo, de nuevo, para la Función 1, el reparto de la participación de los algoritmos en dos ejecuciones distintas de la misma. Como podemos observar, se reiteran las conclusiones sacadas anteriormente. Aunque al principio parezca que existe un equilibrio, y que se va a realizar una adaptación, esta adaptación se pierde cuando se llega a los extremos, dándole, en una iteración muy temprana, todas la participación a un algoritmo, el cuál ni siquiera es el que obtiene los mejores resultados para el algoritmo.



(a) Función 1, Ejecución I

(b) Función 1, Ejecución II

Figura 43: Reparto de las probabilidades en ejecución, Función 1, Dim 10

Sin embargo, de la selección de imágenes 43, sí que podemos sacar otra conclusión. Como hemos venido anticipando en casos anteriores, la versión estacionaria está dando resultados en los que la convergencia tarda más en producirse, respecto a lo que ocurre con la versión generacional. Si comparamos el reparto de probabilidades llevado a cabo en esta sección (imágenes 43) con el reparto ocurrido en la versión generacional (imágenes 41), podemos ver que en este caso, hay una mayor tendencia a adaptarse, y las probabilidades de los algoritmos no se disparan hacia los extremos tan rápidamente. **Este hecho prueba, la relación existente entre la convergencia y el Modelo de Hibridación con Probabilidades Adaptativas, donde una rápida convergencia entorpece el buen funcionamiento del algoritmo.**

Por último, vamos a visualizar de nuevo el reparto medio de las iteraciones que se da internamente en el algoritmo. En este caso, vamos a visualizar los resultados para esta versión en sus dos dimensiones. Podemos ver estos resultados en la imagen 44 (dimensión 10) y en la imagen 45 (dimensión 30). De nuevo, tal y como se puede observar, no existe ninguna tendencia clara que nos permita determinar que se está realizando un ajuste (sino todo lo contrario), **por lo que concluimos que el modelo no está dando buenos resultados para las funciones y algoritmos utilizadas en este proyecto.**

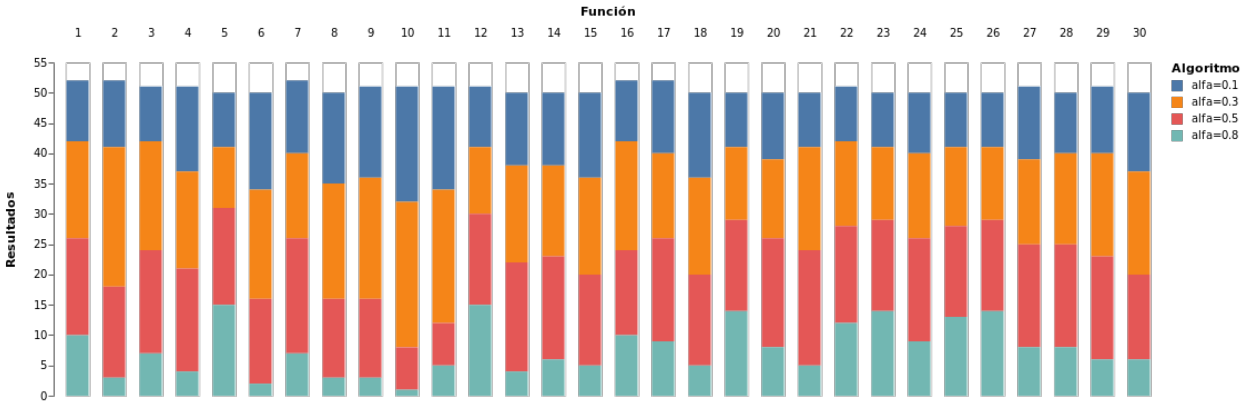


Figura 44: Participación media de los algoritmos en el modelo, Dim=10

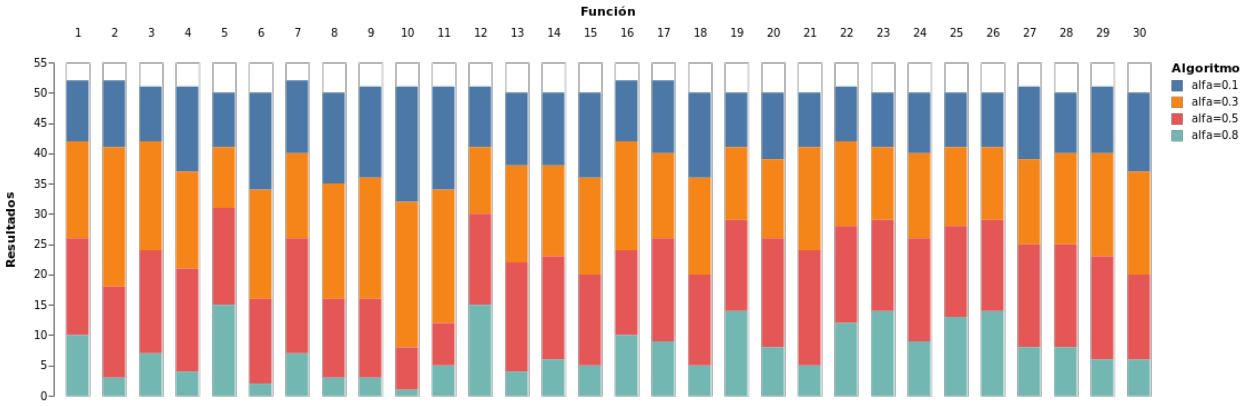


Figura 45: Participación media de los algoritmos en el modelo, Dim=30

De nuevo, podemos observar algunos grupos de funciones de los que sacar algún tipo de conclusión, pero dichos grupos los hemos comentado con anterioridad, por lo que no vamos a entrar a repetir la misma justificación, la cuál es aplicable a este caso. No vemos por tanto, ninguna novedad digna de mención.

Tabla 29: Comparativa Modelo Probabilidades Adaptativas *vs* AGE, Dimensión 10

	Función	MODELO 3	TB-RW	NAM-RW	TB-DC	NAM-DC
Funciones unimodales	**1**	4.2716e+06	1.5062e+07	4.5445e+06	9.1373e+04	1.2852e+05
	2	4.2071e+06	2.2385e+07	2.3344e+03	3.2162e+03	3.7183e+03
	3	5.4611e+03	1.3982e+04	5.1444e+03	2.9106e+03	2.2521e+03
Funciones Multimodales Simples	**4**	3.1670e+01	4.8196e+01	2.6471e+01	2.2436e+01	2.4316e+01
	5	2.0580e+01	2.0700e+01	1.8308e+01	1.9518e+01	1.8705e+01
	6	1.6335e+00	2.4887e+00	8.9220e-01	9.7225e-01	1.3493e+00
	7	2.7812e-01	1.0190e+00	2.7697e-02	5.9555e-02	6.7596e-02
	8	1.0908e+01	1.2589e+01	6.6274e+00	5.8903e+00	6.7062e+00
	9	1.3081e+01	1.6510e+01	9.3395e+00	8.8193e+00	8.5368e+00
	10	2.4223e+02	2.8852e+02	1.1994e+02	1.3100e+02	1.3971e+02
	11	6.0324e+02	1.1878e+03	4.7046e+02	2.5693e+02	2.5919e+02
	12	1.5452e+00	2.2869e+00	3.5016e-01	6.0007e-02	8.0529e-02
	13	2.3872e-01	2.9047e-01	7.5061e-02	9.4233e-02	7.1274e-02
	14	4.0048e-01	4.9721e-01	2.9625e-01	3.0278e-01	2.5294e-01
	15	1.8145e+00	2.8492e+00	1.0124e+00	8.5777e-01	7.6538e-01
	16	2.9716e+00	3.4910e+00	1.8494e+00	2.2432e+00	1.7438e+00
Híbridas Función 1	**17**	2.4045e+05	3.6888e+05	1.6802e+05	1.0080e+04	7.9785e+03
	18	8.4848e+03	8.3484e+03	6.4743e+03	1.1931e+04	1.1616e+04
	19	1.9442e+00	2.3462e+00	1.1198e+00	1.4295e+00	1.2619e+00
	20	8.5527e+03	1.0017e+04	6.7911e+03	4.9414e+03	4.0037e+03
	21	2.8736e+04	1.2894e+05	2.8723e+04	2.9665e+03	1.8785e+03
Composición de Funciones	**22**	5.2381e+01	6.6400e+01	3.7563e+01	3.3007e+01	3.7571e+01
	23	3.3051e+02	3.3298e+02	3.2945e+02	3.2945e+02	3.2945e+02
	24	1.2666e+02	1.4775e+02	1.2625e+02	1.1727e+02	1.1533e+02
	25	1.9530e+02	1.9781e+02	1.9253e+02	1.8447e+02	1.9253e+02
	26	1.0023e+02	1.0027e+02	1.0008e+02	1.0008e+02	1.0007e+02
	27	2.8582e+02	3.4281e+02	2.8240e+02	2.4831e+02	2.0931e+02
	28	4.4835e+02	4.5314e+02	4.1560e+02	4.2631e+02	4.5258e+02
	29	7.4464e+04	6.9891e+04	6.1174e+02	4.2500e+04	4.6736e+02
	30	1.0373e+03	1.2212e+03	8.0306e+02	9.5011e+02	9.1190e+02
	Recuento	**0**	**0**	**9**	**9**	**14**
	Ranking	**3.933**	**4.866**	**2.166**	**2.116**	**1.916**

Tabla 30: Comparativa Modelo Probabilidades Adaptativas*vs* AGE, Dimensión 30

	Función	Modelo 3	TB-RW	NAM-RW	TB-DC	NAM-DC
Funciones unimodales	**1**	2.8377e+07	1.1916e+08	1.4570e+06	1.4407e+06	2.6206e+06
	2	4.6875e+08	1.2052e+09	1.1501e+04	1.6902e+03	8.6171e+03
	3	2.3287e+04	8.1310e+04	2.1060e+04	3.0364e+03	5.0131e+03
Funciones multimodales Simples	**4**	1.6173e+02	2.6966e+02	8.9712e+01	9.3134e+01	1.1954e+02
	5	2.1107e+01	2.1192e+01	2.0909e+01	2.0789e+01	2.0912e+01
	6	1.1470e+01	1.7514e+01	7.8676e+00	9.2749e+00	7.5959e+00
	7	4.2070e+00	1.8105e+01	1.7852e-02	6.2063e-02	1.2408e-02
	8	5.4801e+01	5.9742e+01	2.7377e+01	4.3681e+01	3.9696e+01
	9	7.9479e+01	1.1170e+02	4.4058e+01	4.9968e+01	5.2751e+01
	10	1.5969e+03	3.0434e+03	1.0648e+03	8.3021e+02	9.2419e+02
	11	4.6133e+03	7.4797e+03	3.1212e+03	1.9024e+03	1.5268e+03
	12	3.4170e+00	4.0723e+00	2.0284e+00	7.8601e-02	5.2923e-02
	13	5.6169e-01	5.9248e-01	2.1503e-01	2.8563e-01	2.3721e-01
	14	4.2483e-01	2.3140e+00	3.0385e-01	3.9394e-01	3.9921e-01
	15	1.4657e+02	1.7903e+02	4.1669e+00	4.8115e+00	3.4723e+00
	16	1.2741e+01	1.3330e+01	9.3609e+00	1.0223e+01	9.2643e+00
Híbridas Función 1	**17**	4.7702e+06	1.1845e+07	7.9535e+05	3.0375e+05	4.4548e+05
	18	3.9003e+04	5.3168e+06	2.7573e+03	4.4653e+03	9.5783e+03
	19	2.4999e+01	7.1090e+01	7.0760e+00	1.3861e+01	1.3805e+01
	20	3.8391e+04	8.4253e+04	2.7974e+04	1.4078e+04	1.8457e+04
	21	7.6119e+05	2.3949e+06	3.5593e+05	1.4936e+05	1.5443e+05
Composición de Funciones	**22**	5.3761e+02	6.7019e+02	3.6703e+02	4.4510e+02	3.2607e+02
	23	3.1876e+02	3.2676e+02	3.1528e+02	3.1534e+02	3.1525e+02
	24	2.3999e+02	2.4959e+02	2.3721e+02	2.3053e+02	2.3249e+02
	25	2.1364e+02	2.2070e+02	2.1034e+02	2.0704e+02	2.0555e+02
	26	1.0051e+02	1.1628e+02	1.0531e+02	1.1027e+02	1.0025e+02
	27	5.8582e+02	7.5335e+02	4.8956e+02	5.3681e+02	4.7603e+02
	28	1.0280e+03	1.3151e+03	9.2209e+02	9.3646e+02	9.8801e+02
	29	4.3628e+03	1.4712e+06	4.3510e+05	1.6584e+03	4.5960e+05
	30	1.3362e+04	5.9619e+04	6.8678e+03	3.0010e+03	2.6889e+03
	Recuento	**0**	**0**	**8**	**10**	**12**
	Ranking	**3.866**	**5.0**	**2.166**	**2.066**	**1.900**

6.3.3. Estudio del Modelo de Probabilidades Adaptativas con los mejores Algoritmos Genéticos resultantes

En este caso, podemos ver una mejoría clara respecto al caso anterior, ya que, por un lado, las puntuaciones obtenidas en el Ranking son las mejores hasta el momento por el Modelo de Probabilidades Adaptativas, pero aun así el comportamiento global del Modelo sigue sin ser atractivo, ya que en general, el modelo no destaca por mejorar los resultados individuales. De hecho, como podemos ver en la imagen 46 , no se ha realizado adecuadamente ningún tipo de adaptación, por lo que no es esa la causa de la mejoría. Esta pequeña mejoría que se observa, se puede deber al aumento en la calidad de los algoritmos utilizados, ya que son los mejores algoritmos de cada una de las dos versiones.

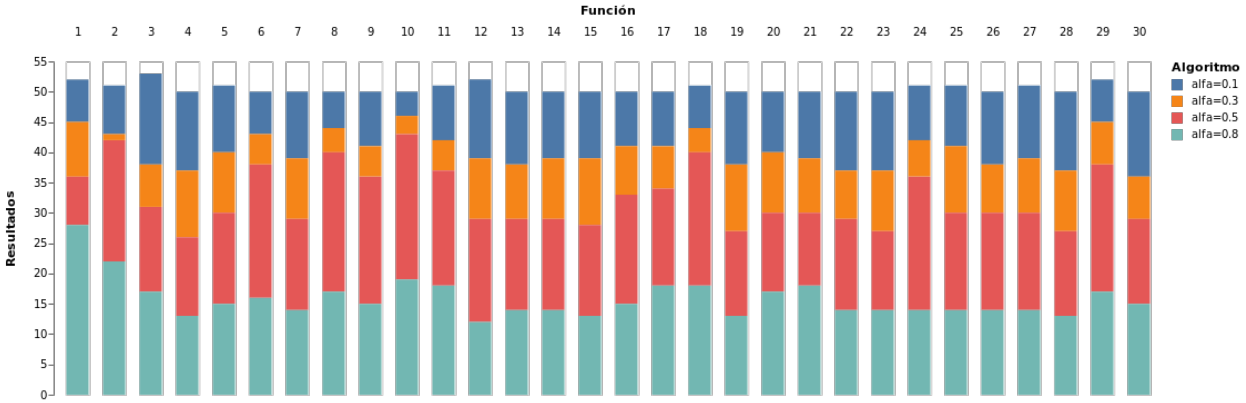


Figura 46: Participación media de los algoritmos en el modelo, Dim=30

Por otro lado, el algoritmo consigue proporcionar la mejor solución en uno de los casos para ambas dimensiones. Pero, tal y como podemos observar en las tablas que recogen los resultados (tabla 31 para dimensión 10 y tabla 32 para dimensión 30), este único mejor resultado no es representativo, ni a nivel global (por la escasez de su aparición) ni a nivel comparativo con los algoritmos genéticos individuales (por el valor tan similar que proporciona). Por tanto, concluimos que los resultados obtenidos para esta versión reiteran las conclusiones que hemos sacado del Modelo en el resto de estudios realizados al respecto, y por tanto, en la versión heterogénea del Modelo tampoco funciona de forma adecuada.

Tabla 31: Comparativa Modelo Prob. Adaptativas *vs* mejores AGs, Dimensión 10

	Función	MODELO 3	AGG 05	AGG 08	NAM-RW	NAM-DC
Funciones unimodales	**1**	1.3989e+06	1.3739e+07	1.7166e+06	4.5445e+06	1.2852e+05
	2	2.9261e+03	2.1115e+07	5.2363e+03	2.3344e+03	3.7183e+03
	3	8.5883e+03	1.3467e+04	1.7111e+04	5.1444e+03	2.2521e+03
Funciones Multimodales Simples	**4**	2.2752e+01	4.1253e+01	2.8846e+01	2.6471e+01	2.4316e+01
	5	2.0369e+01	2.0398e+01	2.0543e+01	1.8308e+01	1.8705e+01
	6	2.7009e+00	3.3448e+00	3.6400e+00	8.9220e-01	1.3493e+00
	7	9.1848e-02	3.7040e-01	7.4475e-02	2.7697e-02	6.7596e-02
	8	1.1385e+01	1.2338e+01	1.6869e+01	6.6274e+00	6.7062e+00
	9	1.5860e+01	1.6095e+01	2.4467e+01	9.3395e+00	8.5368e+00
	10	2.8547e+02	3.3073e+02	3.8424e+02	1.1994e+02	1.3971e+02
	11	8.6690e+02	7.7194e+02	1.0637e+03	4.7046e+02	2.5919e+02
	12	8.2951e-01	8.4969e-01	1.1069e+00	3.5016e-01	8.0529e-02
	13	2.1155e-01	2.4493e-01	2.7126e-01	7.5061e-02	7.1274e-02
	14	4.1729e-01	4.5416e-01	4.3870e-01	2.9625e-01	2.5294e-01
	15	1.4414e+00	2.2581e+00	1.5979e+00	1.0124e+00	7.6538e-01
	16	3.1400e+00	3.2780e+00	3.5895e+00	1.8494e+00	1.7438e+00
Híbridas Función 1	**17**	2.1420e+05	5.2481e+05	1.5050e+05	1.6802e+05	7.9785e+03
	18	1.0545e+04	1.6438e+04	1.0794e+04	6.4743e+03	1.1616e+04
	19	2.1808e+00	2.6449e+00	2.4317e+00	1.1198e+00	1.2619e+00
	20	8.9192e+03	6.7841e+03	1.0622e+04	6.7911e+03	4.0037e+03
	21	2.5209e+04	1.0274e+05	1.5027e+04	2.8723e+04	1.8785e+03
Composición de Funciones	**22**	5.2841e+01	7.7622e+01	6.1099e+01	3.7563e+01	3.7571e+01
	23	3.2965e+02	3.3118e+02	3.3041e+02	3.2945e+02	3.2945e+02
	24	1.3376e+02	1.3911e+02	1.3759e+02	1.2625e+02	1.1533e+02
	25	1.9424e+02	1.9939e+02	1.9927e+02	1.9253e+02	1.9253e+02
	26	1.0023e+02	1.0025e+02	1.0028e+02	1.0008e+02	1.0007e+02
	27	2.9922e+02	3.3075e+02	3.1702e+02	2.8240e+02	2.0931e+02
	28	4.2349e+02	4.8659e+02	4.2352e+02	4.1560e+02	4.5258e+02
	29	7.4443e+04	4.1881e+04	1.5624e+05	6.1174e+02	4.6736e+02
	30	1.0670e+03	1.4501e+03	8.0089e+02	8.0306e+02	9.1190e+02
	Recuento	**1**	**0**	**1**	**12**	**18**
	Ranking	**3.000**	**4.433**	**4.060**	**1.866**	**1.633**

Tabla 32: Comparativa Modelo Prob Adaptativas *vs* mejores AGs, Dimensión 30

	Función	ISLAS	AGG 0.5	AGG 0.8	TB-DC	NAM-DC
Funciones unimodales	**1**	1.2235e+07	1.3579e+08	5.3139e+07	1.4407e+06	2.6206e+06
	2	8.8313e+08	1.4886e+09	1.3695e+09	1.6902e+03	8.6171e+03
	3	4.3356e+04	6.9242e+04	6.1670e+04	3.0364e+03	5.0131e+03
Funciones multimodales Simples	**4**	1.4900e+02	3.0729e+02	1.6931e+02	9.3134e+01	1.1954e+02
	5	2.0766e+01	2.0825e+01	2.1068e+01	2.0789e+01	2.0912e+01
	6	1.6990e+01	1.7874e+01	1.8268e+01	9.2749e+00	7.5959e+00
	7	3.9576e+00	1.9213e+01	6.1387e+00	6.2063e-02	1.2408e-02
	8	7.8292e+01	6.1901e+01	8.7885e+01	4.3681e+01	3.9696e+01
	9	1.2591e+02	9.1954e+01	1.7095e+02	4.9968e+01	5.2751e+01
	10	2.3813e+03	2.4715e+03	3.1230e+03	8.3021e+02	9.2419e+02
	11	4.4359e+03	4.5446e+03	5.5297e+03	1.9024e+03	1.5268e+03
	12	2.4028e+00	1.4926e+00	2.6792e+00	7.8601e-02	5.2923e-02
	13	6.3069e-01	4.8713e-01	7.8861e-01	2.8563e-01	2.3721e-01
	14	7.7631e-01	1.1406e+00	2.2999e+00	3.9394e-01	3.9921e-01
	15	2.6328e+02	2.6908e+02	6.3507e+04	4.8115e+00	3.4723e+00
	16	1.2874e+01	1.2813e+01	1.3488e+01	1.0223e+01	9.2643e+00
Híbridas Función 1	**17**	1.8014e+06	9.7566e+06	4.2351e+06	3.0375e+05	4.4548e+05
	18	5.3439e+03	4.8786e+06	6.8887e+03	4.4653e+03	9.5783e+03
	19	3.0533e+01	8.4923e+01	2.5122e+01	1.3861e+01	1.3805e+01
	20	3.8641e+04	6.3561e+04	5.0486e+04	1.4078e+04	1.8457e+04
	21	6.2263e+05	1.6784e+06	9.7467e+05	1.4936e+05	1.5443e+05
Composición de Funciones	**22**	6.5347e+02	5.7345e+02	6.7263e+02	4.4510e+02	3.2607e+02
	23	3.1799e+02	3.3350e+02	3.3037e+02	3.1534e+02	3.1525e+02
	24	2.4435e+02	2.4961e+02	2.5019e+02	2.3053e+02	2.3249e+02
	25	2.1539e+02	2.2382e+02	2.0986e+02	2.0704e+02	2.0555e+02
	26	1.0562e+02	1.1604e+02	1.1209e+02	1.1027e+02	1.0025e+02
	27	7.1751e+02	7.8389e+02	8.0668e+02	5.3681e+02	4.7603e+02
	28	1.1479e+03	1.4937e+03	1.1413e+03	9.3646e+02	9.8801e+02
	29	4.2708e+05	1.1281e+06	1.3776e+06	1.6584e+03	4.5960e+05
	30	8.1829e+03	3.2739e+04	1.3471e+04	3.0010e+03	2.6889e+03
	Recuento	**1**	**0**	**8**	**14**	**15**
	Ranking	**3.133**	**4.266**	**4.366**	**1.566**	**1.666**

6.4. Comparativa entre los Modelos Híbridos

En esta última sección del proyecto, vamos a hacer una comparativa generalizada de los tres modelos anteriores entre sí. Esta comparativa la vamos a hacer en base a los resultados de las funciones estudiadas que se han obtenido para cada modelo híbrido. En las tablas 34, 35, 36, 37, 38 y 39 podemos ver los resultados alcanzados por dichos modelos para cada una de las versiones y dimensiones estudiadas: versión generacional, estacionaria y una que combina lo mejor de ambas.

Sin embargo, como las conclusiones a extraer de las distintas tablas coinciden en todas ellas, para mayor simplicidad y claridad se puede ver en la tabla 33 un resumen de dicha información. En esta tabla resumen, tenemos los resultados del Ranking obtenidos por los distintos modelos híbridos.

Podemos ver en dicha tabla, una primera columna denominada `DIM`. Esta primera columna hace referencia a la dimensión del problema, y es la que nos permite distinguir los resultados de cada versión para Dimensión 10 y Dimensión 30. Las filas de la tabla, harán por tanto referencia al Ranking de los modelos para una determinada versión y dimensión.

Además, en estas tablas, tendremos por una parte, los tres modelos estudiados hasta el momento: Modelo de Islas con tamaño de subpoblación 60, Modelo de Ejecución en 2 Fases, y Modelo con Probabilidades Adaptativas.

Se le ha añadido a las tablas una última columna, en la cuál contemplamos los resultados de los Modelos de Islas para subpoblaciones con 15 individuos. La razón de este añadido es, que un Modelo de Islas con cuatro subpoblaciones de tamaño 15, formarían un modelo que se ejecuta globalmente con una población de tamaño 60.

Este hecho puede ser interesante ya que, a pesar de que en todos los modelos estamos utilizado poblaciones de 60 para cada uno de sus algoritmos involucrados, habiendo una diferencia clara:

- En el *Modelo de Islas*, **las cuatro subpoblaciones están activas** en todas las iteraciones, funcionando y aportando información unas a otras.
- En los modelos de *Ejecución en 2 Fases* y *Probabilidades Adaptativas* **sólo hay funcionando una única subpoblación de 60 a nivel global**, (aunque cada población funciona con un tamaño de 60).

Por tanto, gracias a este último modelo añadido, queremos ver, si reduciendo la diversidad del Modelo de Islas de esta forma, podríamos seguir obteniendo buenos resultados.

Tabla 33: Tabla Resumen Comparación Modelos Híbridos

	DIM	ISLAS(60)	EJ. FASES	P. ADAPTATIVAS	ISLAS(15)
Generacional	**10**	1.133	1.866	3.233	3.766
	30	1.100	1.900	3.033	3.966
Estacionaria	**10**	1.183	2.166	3.766	2.883
	30	1.233	1.866	3.533	3.366
Combinada	**10**	1.200	2.200	3.400	3.400
	30	1.066	2.300	3.230	3.400

Modelos de Islas

Tal y como podemos ver en la tabla 33, si disminuimos el tamaño de población del Modelo de Islas, para intentar equiparar el tamaño de la población **global** con la de los otros modelos híbridos, este modelo pierde su efectividad.

Por tanto, nos reafirmamos en la conclusión que sacamos anteriormente respecto al tamaño de las subpoblaciones en los Modelos de Islas, y es que existe una relación directa entre la diversidad en el modelo y los resultados que se obtienen. Por tanto, **las mejoras en este tipo de modelos, y la superioridad de la técnica respecto a otros algoritmos, debería plantearse en función a los tamaños de subpoblaciones**.

Respecto a nuestro problema, podemos ver que si mantenemos en igualdad el tamaño de las subpoblaciones de los algoritmos en todos los modelos estudiados, el Modelo de Islas es el que consigue los mejores resultados con diferencia respecto al resto de técnicas.

Modelo Híbrido de Adaptación en 2 Fases

Este modelo de híbridación, es con el que se consiguen los segundos mejores resultados en el estudio que hemos realizado. Podemos sacar, de los resultados vistos en la tabla 33 dos hechos claros:

1. Si el estudio que realizamos, se hace en base al mismo tamaño de población activa (en nuestro caso, sería una población global de 60), un modelo de adaptación es la mejor opción a seguir, ya que permite adaptarse a las circunstancias de los distintos algoritmos que lo componen, y obtener por tanto resultados destacados.

2. Si mantenemos en igualdad los tamaños de poblaciones a nivel de los algoritmos dentro del modelo, sigue obteniendo muy buenos resultados, ya que la adaptación es un potencial para los modelos híbridos, pero vemos sin embargo que el aumento de diversidad que ofrecería un Modelo de Islas superaría a este modelo adaptativo. Como comentamos anteriormente, este modelo sería una buena opción en este caso, pero solo cuando sepamos que alguno de los algoritmos que lo contiene da buenos resultados. En caso contrario, el modelo tradicional es la mejor opción.

Modelo Híbrido con Probabilidades Adaptativas

Como podemos ver en la tabla 33, es que los resultados que proporciona este algoritmo no son buenos resultados, ya que la adaptación en base a la mejora obtenida durante el proceso completo de la ejecución no es la medida más adecuada para el tipo de problemas que contemplamos. Prueba de ello es, que en el caso del Modelo de Ejecución en 2 Fases, obtengamos mejores resultados que con este tercero.

Sin embargo, si lo comparamos respecto al Modelo de Islas, podemos sacar dos conclusiones.

1. Si utilizamos en el Modelo de Islas, los mismos tamaños de subpoblación respecto a los del Modelo de Probabilidades Adaptativas, el Modelo de Islas es claramente superior.
2. Si por otra parte, lo que igualamos es el tamaño de la población global que utiliza el algoritmo en ejecución, podemos ver que los resultados están bastante igualados, y en este caso, podría ser una opción competitiva (respecto el modelo clásico).

Tabla 34: Comparativa de los Modelos Híbridos, Versión Generacional, Dimensión 10

	Función	ISLAS(60)	EJ. FASES	P. ADAPTATIVAS	ISLAS(15)
Funciones Unimodales	**1**	6.5026e+05	2.7226e+06	2.0392e+07	3.3134e+07
	2	4.4754e+03	5.1190e+03	2.6328e+08	1.0350e+08
	3	1.6329e+03	5.5671e+03	1.5530e+04	3.6775e+04
Funciones multimodales Simples	**4**	1.7514e+01	2.1332e+01	8.6025e+01	7.6859e+01
	5	2.0105e+01	2.0156e+01	2.0414e+01	2.0811e+01
	6	2.2033e+00	2.2510e+00	4.4434e+00	7.0298e+00
	7	5.4011e-02	2.4374e-01	9.4404e+00	3.5810e+00
	8	7.2696e+00	9.5883e+00	1.3647e+01	3.8620e+01
	9	9.8555e+00	1.0661e+01	2.4704e+01	5.2224e+01
	10	1.6907e+02	2.2900e+02	5.1954e+02	1.2130e+03
	11	5.5017e+02	5.8306e+02	1.0465e+03	1.7905e+03
	12	2.4918e-01	2.9580e-01	9.6813e-01	2.0950e+00
	13	1.6825e-01	1.6512e-01	4.5885e-01	6.3526e-01
	14	2.9645e-01	3.0076e-01	2.0710e+00	5.9613e-01
	15	1.0419e+00	1.6529e+00	1.2973e+01	2.1500e+01
	16	2.7782e+00	3.0277e+00	3.4046e+00	3.8021e+00
Híbridas Función 1	**17**	4.4819e+04	1.0090e+05	4.9446e+05	6.3868e+05
	18	2.3651e+03	2.0763e+03	3.0441e+05	2.0777e+05
	19	1.5083e+00	1.7336e+00	4.6037e+00	7.5559e+00
	20	1.2494e+03	2.2786e+03	9.0288e+03	1.7022e+04
	21	2.3658e+03	6.7484e+03	1.1892e+05	5.7983e+05
Composición de Funciones	**22**	2.1822e+01	3.4480e+01	9.8799e+01	1.7035e+02
	23	3.2945e+02	3.2976e+02	3.3657e+02	3.3482e+02
	24	1.2344e+02	1.2489e+02	1.6273e+02	1.6922e+02
	25	1.8034e+02	1.8918e+02	1.9746e+02	2.0081e+02
	26	1.0013e+02	1.0016e+02	1.0037e+02	1.0069e+02
	27	2.0665e+02	2.2107e+02	3.6467e+02	4.0539e+02
	28	4.0707e+02	3.9716e+02	5.7886e+02	7.0582e+02
	29	4.1250e+02	5.5509e+02	1.1448e+06	2.4973e+05
	30	8.5598e+02	8.2542e+02	4.4203e+03	8.2989e+03
	Recuento	**26**	**4**	**0**	**0**
	Ranking	**1.133**	**1.866**	**3.233**	**3.766**

Tabla 35: Comparativa de los Modelos Híbridos, Versión Generacional, Dimensión 30

	Función	ISLAS(60)	EJ. FASES	P. ADAPTATIVAS	ISLAS(15)
Funciones Unimodales	**1**	1.0861e+07	2.2576e+07	2.2747e+08	5.2894e+08
	2	1.3450e+04	4.8406e+08	7.0254e+09	1.0895e+10
	3	1.0403e+04	3.3825e+04	7.3947e+04	1.4896e+05
Funciones multimodales Simples	**4**	1.1080e+02	1.1629e+02	1.1709e+03	2.1875e+03
	5	2.0580e+01	2.0818e+01	2.0938e+01	2.1220e+01
	6	1.6482e+01	1.6396e+01	2.3216e+01	3.6711e+01
	7	1.0498e-02	7.7646e+00	8.4085e+01	9.5861e+01
	8	5.4164e+01	6.0486e+01	1.0737e+02	2.6471e+02
	9	8.0978e+01	8.9456e+01	1.3908e+02	3.1541e+02
	10	2.2620e+03	2.3830e+03	3.3953e+03	6.2754e+03
	11	3.4862e+03	3.5528e+03	5.2612e+03	8.2489e+03
	12	7.3543e-01	6.8698e-01	2.0002e+00	4.2148e+00
	13	3.3045e-01	4.9448e-01	1.7247e+00	2.5160e+00
	14	2.8790e-01	4.0220e-01	2.6848e+01	3.4186e+01
	15	1.2691e+01	2.4248e+02	4.6087e+03	1.0392e+05
	16	1.2349e+01	1.2669e+01	1.3033e+01	1.3722e+01
Híbridas Función 1	**17**	1.3011e+06	2.6496e+06	1.7936e+07	3.7637e+07
	18	2.8326e+03	6.0032e+04	7.6977e+07	1.1914e+08
	19	1.6123e+01	2.1213e+01	1.0567e+02	1.5890e+02
	20	1.7651e+04	3.7834e+04	7.2387e+04	3.5490e+05
	21	4.0684e+05	7.0658e+05	4.2400e+06	8.9406e+06
Composición de Funciones	**22**	4.1445e+02	4.6696e+02	8.4628e+02	1.1379e+03
	23	3.1524e+02	3.2100e+02	3.6163e+02	4.1726e+02
	24	2.3491e+02	2.4530e+02	2.5852e+02	2.8202e+02
	25	2.1120e+02	2.1043e+02	2.2464e+02	2.4837e+02
	26	1.0041e+02	1.0058e+02	1.2663e+02	1.5813e+02
	27	7.1579e+02	7.2813e+02	9.4192e+02	1.1646e+03
	28	9.8711e+02	1.0412e+03	2.4785e+03	3.8773e+03
	29	3.7113e+03	9.2152e+05	4.6336e+07	1.3451e+07
	30	4.6512e+03	1.7286e+04	2.2552e+05	2.7559e+05
	Recuento	**27**	**3**	**0**	**0**
	Ranking	**1.100**	**1.900**	**3.033**	**3.966**

Tabla 36: Comparativa de los Modelos Híbridos, Versión Estacionaria, Dimensión 10

	Función	ISLAS(60)	EJ. FASES	P. ADAPTATIVAS	ISLAS(15)
Funciones Unimodales	**1**	1.2687e+05	1.7231e+06	4.2716e+06	2.4020e+05
	2	1.4547e+03	1.6996e+03	4.2071e+06	3.3282e+03
	3	9.8373e+02	2.7828e+03	5.4611e+03	1.7934e+03
Funciones multimodales Simples	**4**	2.4256e+01	1.8674e+01	3.1670e+01	1.6238e+01
	5	1.8092e+01	1.9127e+01	2.0580e+01	2.0040e+01
	6	6.8492e-01	6.8201e-01	1.6335e+00	3.7625e+00
	7	2.5634e-02	2.6540e-02	2.7812e-01	1.9765e-01
	8	3.6285e+00	5.7764e+00	1.0908e+01	1.2492e+01
	9	6.4028e+00	6.8298e+00	1.3081e+01	1.9384e+01
	10	6.7845e+01	9.3402e+01	2.4223e+02	3.3449e+02
	11	2.1102e+02	4.6752e+02	6.0324e+02	6.9847e+02
	12	1.9412e-01	4.4881e-01	1.5452e+00	3.2282e-01
	13	8.9394e-02	1.0874e-01	2.3872e-01	1.8256e-01
	14	2.7045e-01	2.9991e-01	4.0048e-01	3.1859e-01
	15	8.9744e-01	1.0292e+00	1.8145e+00	1.6638e+00
	16	1.6813e+00	2.0766e+00	2.9716e+00	2.7604e+00
Híbridas Función 1	**17**	4.4129e+03	1.8296e+04	2.4045e+05	2.8608e+04
	18	3.2796e+03	3.3972e+03	8.4848e+03	4.0718e+03
	19	1.0478e+00	1.0923e+00	1.9442e+00	2.0303e+00
	20	1.0811e+03	2.0219e+03	8.5527e+03	9.6252e+02
	21	9.9040e+02	3.6774e+03	2.8736e+04	2.4449e+03
Composición de Funciones	**22**	1.0174e+01	2.1894e+01	5.2381e+01	4.8718e+01
	23	3.2945e+02	3.2947e+02	3.3051e+02	3.2945e+02
	24	1.1696e+02	1.1645e+02	1.2666e+02	1.2595e+02
	25	1.6253e+02	1.7771e+02	1.9530e+02	1.8790e+02
	26	1.0009e+02	1.0011e+02	1.0023e+02	1.0019e+02
	27	8.7279e+01	1.2815e+02	2.8582e+02	2.8573e+02
	28	3.8157e+02	3.9276e+02	4.4835e+02	4.4664e+02
	29	4.5704e+02	5.1985e+02	7.4464e+04	4.5880e+02
	30	6.4339e+02	6.7396e+02	1.0373e+03	1.1580e+03
	Recuento	**26**	**2**	**0**	**3**
	Ranking	**1.183**	**2.166**	**3.766**	**2.883**

Tabla 37: Comparativa de los Modelos Híbridos, Versión Estacionaria, Dimensión 30

	Función	ISLAS(60)	EJ. FASES	P. ADAPTATIVAS	ISLAS(15)
Funciones Unimodales	**1**	5.1771e+06	3.0227e+06	2.8377e+07	4.7377e+07
	2	9.3516e+03	1.4876e+05	4.6875e+08	9.0187e+06
	3	2.8612e+03	1.9210e+04	2.3287e+04	1.0367e+04
Funciones multimodales Simples	**4**	9.5424e+01	7.4410e+01	1.6173e+02	2.0490e+02
	5	2.0677e+01	2.0918e+01	2.1107e+01	2.0562e+01
	6	6.9519e+00	7.6757e+00	1.1470e+01	2.7190e+01
	7	2.6638e-02	1.4230e-02	4.2070e+00	1.0714e+00
	8	2.7774e+01	2.8306e+01	5.4801e+01	1.0487e+02
	9	4.1652e+01	4.0370e+01	7.9479e+01	1.5615e+02
	10	5.7865e+02	9.2252e+02	1.5969e+03	3.6160e+03
	11	1.3906e+03	2.6320e+03	4.6133e+03	4.8571e+03
	12	2.1299e-01	3.9633e-01	3.4170e+00	8.1056e-01
	13	2.0226e-01	2.5468e-01	5.6169e-01	3.6592e-01
	14	2.9372e-01	3.1390e-01	4.2483e-01	3.9532e-01
	15	4.0990e+00	3.8618e+00	1.4657e+02	2.5273e+01
	16	9.4650e+00	9.3929e+00	1.2741e+01	1.2016e+01
Híbridas Función 1	**17**	3.8396e+05	4.9033e+05	4.7702e+06	2.4978e+06
	18	4.7465e+02	1.6625e+03	3.9003e+04	3.0289e+04
	19	6.6742e+00	6.7155e+00	2.4999e+01	3.3234e+01
	20	7.8843e+03	1.1638e+04	3.8391e+04	1.2069e+04
	21	1.9271e+05	2.8349e+05	7.6119e+05	6.7911e+05
Composición de Funciones	**22**	1.9731e+02	2.6967e+02	5.3761e+02	3.8305e+02
	23	3.1524e+02	3.1562e+02	3.1876e+02	3.1634e+02
	24	2.3175e+02	2.3677e+02	2.3999e+02	2.4118e+02
	25	2.0696e+02	2.1013e+02	2.1364e+02	2.1916e+02
	26	1.0021e+02	1.0027e+02	1.0051e+02	1.1587e+02
	27	4.2317e+02	4.4036e+02	5.8582e+02	6.7494e+02
	28	8.9731e+02	9.4241e+02	1.0280e+03	1.6793e+03
	29	1.2977e+03	2.4154e+03	4.3628e+03	1.0053e+04
	30	2.7189e+03	5.3866e+03	1.3362e+04	7.4640e+03
	Recuento	**25**	**4**	**0**	**1**
	Ranking	**1.233**	**1.866**	**3.533**	**3.366**

Tabla 38: Comparativa de los Modelos Híbridos, Versión Combinada, Dimensión 10

	Función	ISLAS(60)	EJ. FASES	P. ADAPTATIVAS	ISLAS(15)
Funciones Unimodales	**1**	1.5752e+05	1.7231e+06	1.3989e+06	9.2272e+05
	2	1.0672e+03	1.6996e+03	2.9261e+03	2.6597e+04
	3	1.5882e+03	2.7828e+03	8.5883e+03	4.6951e+03
Funciones multimodales Simples	**4**	1.6113e+01	1.8674e+01	2.2752e+01	2.3633e+01
	5	1.9846e+01	1.9127e+01	2.0369e+01	2.0089e+01
	6	6.8365e-01	6.8201e-01	2.7009e+00	4.5951e+00
	7	2.3192e-02	2.6540e-02	9.1848e-02	3.8099e-01
	8	5.7001e+00	5.7764e+00	1.1385e+01	1.3454e+01
	9	5.9507e+00	6.8298e+00	1.5860e+01	2.0260e+01
	10	7.7954e+01	9.3402e+01	2.8547e+02	4.3488e+02
	11	3.2684e+02	4.6752e+02	8.6690e+02	9.0062e+02
	12	2.2760e-01	4.4881e-01	8.2951e-01	3.9547e-01
	13	9.4284e-02	1.0874e-01	2.1155e-01	2.1959e-01
	14	2.7804e-01	2.9991e-01	4.1729e-01	3.0964e-01
	15	8.5697e-01	1.0292e+00	1.4414e+00	2.9398e+00
	16	1.9262e+00	2.0766e+00	3.1400e+00	3.0727e+00
Híbridas Función 1	**17**	3.0557e+04	1.8296e+04	2.1420e+05	6.2146e+04
	18	2.0985e+03	3.3972e+03	1.0545e+04	6.6389e+03
	19	1.0338e+00	1.0923e+00	2.1808e+00	2.5975e+00
	20	1.9395e+03	2.0219e+03	8.9192e+03	1.6641e+03
	21	1.4486e+03	3.6774e+03	2.5209e+04	2.6070e+03
Composición de Funciones	**22**	1.2414e+01	2.1894e+01	5.2841e+01	7.3412e+01
	23	3.2945e+02	3.2947e+02	3.2965e+02	3.2956e+02
	24	1.1681e+02	1.1645e+02	1.3376e+02	1.4753e+02
	25	1.7491e+02	1.7771e+02	1.9424e+02	1.8772e+02
	26	1.0009e+02	1.0011e+02	1.0023e+02	1.0022e+02
	27	2.6215e+02	1.2815e+02	2.9922e+02	3.2854e+02
	28	3.7553e+02	3.9276e+02	4.2349e+02	4.8308e+02
	29	4.4850e+02	5.1985e+02	7.4443e+04	9.8980e+04
	30	6.1567e+02	6.7396e+02	1.0670e+03	1.4292e+03
	Recuento	**24**	**5**	**0**	**1**
	Ranking	**1.200**	**2.200**	**3.400**	**3.400**

Tabla 39: Comparativa de los Modelos Híbridos, Versión Combinada, Dimensión 30

	Función	ISLAS(60)	EJ. FASES	P. ADAPTATIVAS	ISLAS(15)
Funciones Unimodales	**1**	5.3193e+06	1.5295e+07	1.2235e+07	6.5283e+07
	2	6.2833e+03	3.3774e+08	8.8313e+08	7.7868e+07
	3	2.7764e+03	1.9212e+04	4.3356e+04	1.9340e+04
Funciones multimodales Simples	**4**	1.0926e+02	1.1723e+02	1.4900e+02	2.1938e+02
	5	2.0354e+01	2.0899e+01	2.0766e+01	2.0492e+01
	6	8.7581e+00	1.4156e+01	1.6990e+01	3.0531e+01
	7	1.5136e-02	6.8299e+00	3.9576e+00	1.5895e+00
	8	3.2938e+01	6.7272e+01	7.8292e+01	1.4626e+02
	9	4.9306e+01	1.0268e+02	1.2591e+02	2.2216e+02
	10	7.0523e+02	1.9526e+03	2.3813e+03	3.6537e+03
	11	1.6933e+03	2.7614e+03	4.4359e+03	4.9810e+03
	12	4.1456e-01	7.8742e-01	2.4028e+00	9.7632e-01
	13	2.4816e-01	4.4471e-01	6.3069e-01	4.5236e-01
	14	3.1666e-01	4.4572e-01	7.7631e-01	3.4334e-01
	15	5.0879e+00	9.1687e+01	2.6328e+02	5.3661e+01
	16	9.9512e+00	1.1137e+01	1.2874e+01	1.2687e+01
Híbridas Función 1	**17**	4.6420e+05	2.9300e+05	1.8014e+06	3.2842e+06
	18	2.3388e+03	3.4957e+03	5.3439e+03	7.3053e+05
	19	8.4087e+00	1.0706e+01	3.0533e+01	3.6856e+01
	20	1.0556e+04	1.5206e+04	3.8641e+04	1.6533e+04
	21	1.9287e+05	1.5842e+05	6.2263e+05	7.6120e+05
Composición de Funciones	**22**	2.6657e+02	3.8138e+02	6.5347e+02	5.8582e+02
	23	3.1524e+02	3.1936e+02	3.1799e+02	3.1786e+02
	24	2.3333e+02	2.4323e+02	2.4435e+02	2.4802e+02
	25	2.0606e+02	2.0660e+02	2.1539e+02	2.2343e+02
	26	1.0028e+02	1.0054e+02	1.0562e+02	1.1069e+02
	27	4.5061e+02	4.7673e+02	7.1751e+02	8.5730e+02
	28	9.0371e+02	9.8953e+02	1.1479e+03	2.0551e+03
	29	1.5914e+03	1.1302e+04	4.2708e+05	4.6868e+05
	30	3.3217e+03	1.0923e+04	8.1829e+03	1.2756e+04
	Recuento	**29**	**1**	**0**	**0**
	Ranking	**1.066**	**2.300**	**3.230**	**3.400**

6.5. Estudio con un framework de hibridación dinámica (MOS)

Como hemos venido comentando a lo largo del proyecto, no hay una solución fija que determine cuál es el mejor modelo a utilizar en cada caso (lo cuál llevaría un estudio previo de cada una de las posibilidades a tener en cuenta), y la adaptación de un modelo híbrido al comportamiento de los algoritmos que lo componen es un factor que influye directamente en los resultados que se contienen. Por ello, se ha intentado añadir en este proyecto un estudio de los resultados que proporciona un framework de hibridación dinámica de algoritmos evolutivos llamado *Multiple Offspring Sampling* (MOS), y así poder compararlos con los obtenidos con los modelos que hemos implementado.

Para ello, hemos colaborado con el autor, Antonio LaTorre, citado en [15], para intentar incorporar los resultados a este proyecto. Sin embargo, en este frame, faltan muchos de los algoritmos que hemos utilizado en este proyecto, y debido a la complejidad que suponía añadirlos y ejecutarlos, y también a la falta de tiempo para ello, no se ha podido añadir a este proyecto. Sin embargo, sería una opción posible como trabajo futuro a llevar a cabo.

Conclusiones

7. Conclusiones

Hemos podido observar, a lo largo del análisis que hemos realizado a los métodos implementados, que efectivamente, los modelos de hibridación que involucran modelos evolutivos dan buenos resultados, y son competitivos a la hora de alcanzar buenas soluciones y superar distintas dificultades que nos podemos encontrar.

En primer lugar, hemos llevado a cabo un estudio interno de los Modelos de Islas, para ver sus ventajas y sus deficiencias más importantes. Como resultado, hemos podido ver cómo su comportamiento depende, de forma directa y fundamental, del comportamiento que tengan los algoritmos que lo componen respecto al problema concreto que estemos estudiando, sacando diferentes conclusiones.

Respecto a este algoritmo, es importante destacar que, los casos donde uno de los algoritmos que lo componen se comporta tremendamente bien, pero el resto de algoritmos no dan buenos resultados, el Modelo de Islas deja de ser competitivo en este aspecto. Ello se debe a que, al involucrar de forma igualitaria a todos los algoritmos, sin preferencias por ninguno de ellos, los resultados resultan ser una representación de la unión de ellos.

Podemos hacer una analogía de este hecho con lo que ocurre con una carrera de relevos. Si un atleta corre muy rápido, y marca unos tiempos destacables respecto al resto de los atletas de la competición, por supuesto que influye positivamente a su equipo de relevo. Pero si otro de los atletas de su equipo, va muy lento, esto provocará que pierdan la ventaja que había conseguido el primer atleta, equilibrándose (o incluso viéndose perjudicados en el resultado) el nivel del equipo a un nivel medio, ya que se tiene en cuenta tanto los malos resultados como los buenos.

Hemos visto además, la relación interna que existe entre estos resultados y el comportamiento interno del algoritmo, donde hay situaciones en las que hay modelos que no aportan mejorías trascendentales.

Por tanto, cuando un algoritmo involucrado en un Modelo de Islas, es realmente bueno en comparación con todos los demás, y basándonos en los resultados que hemos obtenido, pensamos que este modelo no es el más adecuado.

Como consecuencia, quedaría demostrado que **existe una necesidad de adaptación en la combinación de los algoritmos, que hay que cubrir en los modelos de hibridación**, la cual no puede prever este tipo de modelos.

En este contexto entra uno de los dos algoritmos más novedosos que hemos implementado en este proyecto. Estamos hablando del *Modelo Híbrido de Ejecución en 2 Fases*. Este modelo, se plantea como una solución a los problemas de adaptación que existen en los Modelos Híbridos. Con el análisis de este modelo, hemos podido reiterar, realmente, la necesidad de adaptación, y es que los algoritmos que más participan y más mejoría aportan dentro del modelo híbrido, son aquellos que mejor responden al problema de manera individual. En vista al análisis que hemos realizado, podemos concluir dos comportamientos principales.

1. En primer lugar, **si uno de los algoritmos incorporados en el Modelo Híbrido de Ejecución en 2 Fases, comienza con muy buenos resultados (o más de un algoritmo, pero mínimo debe haber un algoritmo con estas características), el modelo es capaz de identificar al algoritmo más prometedor, y concluir con soluciones realmente competitivas** respecto a los resultados que consiguen los algoritmos genéticos de forma individual.
2. Sin embargo, **el algoritmo fija la adaptación en un número de iteraciones concreto, y esta decisión no será adecuada para los casos en los que los mejores algoritmos tarden mucho en converger hacia soluciones buenas.** Esto conllevaría que, en el momento de realizar la adaptación, el Modelo de Ejecución en 2 Fases no sería capaz de determinar el mejor algoritmo para el problema (puesto que a ese nivel aún no se ha desarrollado lo suficiente), y como consecuencia, se adapta a un algoritmo de calidad inferior a la que se podría alcanzar. Como resultado, en este caso, los propios algoritmos que forman el modelo, pueden superar con creces los resultados obtenidos a partir de la hibridación.

Pensamos por tanto que, **con este Modelo, sigue habiendo una dependencia fuerte del algoritmo que se elige, puesto que, solo será adecuado utilizarlo para funciones realmente buenas.** Un proyecto a desarrollar en el futuro podría estar centrado en determinar el reparto de las iteraciones de cara a que cuando se realice la adaptación, ésta se vea menos afectada por la convergencia de los algoritmos que implementa.

Por otra parte, hemos analizado los resultados obtenidos con el Modelo de Probabilidades Adaptativas. Este modelo, a priori, parece una buena idea que va a dar lugar a grandes resultados, pero sin embargo, como hemos podido ver, no ha sido así en la práctica. Este tipo de algoritmos utiliza la mejora obtenida por cada algoritmo para realizar un ajuste proporcional de la participación de los mismos. EL problema viene cuando utilizamos algoritmos, que convergen a iteraciones tempranas, y por tanto rápidamente empiezan a no mostrar ningún tipo de mejora. Este hecho provoca que el reparto se termine volviendo exclusivo para el último algoritmo que mejorara antes de que el modelo dejase de producir ningún tipo de mejora, no contemplando

tampoco convergencias fuertes que se den una vez avanzada la ejecución), y, como consecuencia, el Modelo no ha resultado efectivo en nuestro proyecto.

Como conclusión, pensamos que en otro tipo de algoritmos, donde realmente se realicen mejoras durante la mayor parte de la ejecución, sería realmente apropiado utilizar este tipo de técnicas. En primer lugar porque como hemos visto, la necesidad de adaptación hay que cubrirla, y ello no significa que con un caso malo esta técnica no sea buena ni de grandes resultados. Lo que pensamos que ocurre en este caso, es que su convergencia rápida, impide que el algoritmo tome decisiones correctas, y que la falta de mejoras lo confunde y por ello muestra ese comportamiento tan eficaz.

Si utilizásemos por tanto este modelo, con algoritmos que convergieran durante una mayor parte de la ejecución, quizá sí sería interesante consultar la efectividad de la técnica. Otra posible solución para este caso, podría haber sido reinicializar los vectores de probabilidades cada cierto tiempo, para evitar limitar la exploración a un único algoritmo. Ambas ideas, se encuentran fuera de los límites de este proyecto, pero se podrían considerar como ideas futuras a llevar a cabo.

Por último, hemos hecho un análisis de los modelos híbridos entre sí, llegando a una conclusión principal. **El estudio comparativo entre modelos híbridos hay que realizarlo en función del tamaño de subpoblación utilizado.**

- Si igualamos el tamaño de población utilizada por el modelo a nivel **global**, los algoritmos adaptativos utilizados son una muy buena opción. El Modelo de Ejecución en 2 Fases superaría al modelo clásico con creces, y el Modelo de Probabilidades Adaptativas podría ser una opción competitiva.
- Si igualamos el tamaño de subpoblación utilizado por los algoritmos que forman los modelos, el modelo clásico sigue siendo la mejor alternativa a utilizar.

7.1. Ideas finales

Como hemos podido observar, cada uno de los algoritmos de hibridación implementados tienen sus ventajas o desventajas, pero a pesar de ello, sí que hemos sido capaces de determinar conductas claras.

1. La comparativa a realizar entre modelos híbridos evolutivos se debe plantear en función del tamaño de las subpoblaciones que intervienen en cada uno de los modelos.
2. **El Modelo de Islas, se reafirma como el modelo híbrido por excelencia, ya que es el que mejor resultados ha logrado de forma generalizada.** Sus capacidades de aprovechar la sinergia, y también la facilidad de escape de óptimos locales, lo hacen una técnica destacable, especialmente cuándo no estamos seguros del problema que aplicamos, y queremos una solución buena, asumiendo un poco de riesgo. Sin embargo, si equiparamos la población global que utiliza el algoritmo, con la de los otros modelos, vemos que los modelos adaptativos son opciones realmente competitivas, que incluso

superan los resultados del modelo clásico, y lo más adecuado sería utilizar una técnica que incorpore adaptación.

3. Si sabemos qué tipo de algoritmo funciona realmente bien con nuestro problema, el Modelo de Ejecución por Fases es una opción adecuada, ya que nos permitirá obtener soluciones muy buenas bajo poco esfuerzo de adaptación.

4. El Modelo de Probabilidades Adaptativas ha sido la gran sorpresa en este proyecto, ya que se esperaba que realmente fuese capaz de dotar a las ejecuciones de la adaptación dinámica que necesitábamos en cada caso para dar buenas soluciones. Hemos podido observar, que su uso ligado a algoritmos con poca capacidad de salida de óptimos y convergencias rápidas no da buenos resultados, por lo que realmente no sería una técnica adecuada para los casos estudiados en este proyecto.

Bibliografía

[1] Enrique Alba y Marco Tomassini. "Parallelism and evolutionary algorithms". En: *IEEE Transactions on Evolutionary Computation* 6.5 (2002), págs. 443-462. ISSN: 1089778X. DOI: 10.1109/TEVC.2002.800880.

[2] Enrique Alba y José M. Troya. "Influence of the migration policy in parallel distributed GAs with structured and panmictic populations". En: *Applied Intelligence* 12.3 (2000), págs. 163-181. ISSN: 0924669X. DOI: 10.1023/A:1008358805991.

[3] J. Arabas, Z. Michalewicz y J. Mulawka. "GAVaPS-a genetic algorithm with varying population size". En: *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence* (1899), págs. 73-78. DOI: 10.1109/ICEC.1994.350039. URL: http://ieeexplore.ieee.org/document/350039/.

[4] A. Auger y N. Hansen. "A Restart CMA Evolution Strategy With Increasing Population Size". En: *2005 IEEE Congress on Evolutionary Computation* 2 (2005), págs. 1769-1776. ISSN: 1089-778X. DOI: 10.1109/CEC.2005.1554902. URL: http://ieeexplore.ieee.org/document/1554902/.

[5] Thomas Bäck y Frank Hoffmeister. "Extended Selection Mechanisms in Genetic Algorithms". En: *Proceedings of the 4th International Conference on Genetic Algorithms* (1991), págs. 92-99. DOI: 10.1.1.42.2192. URL: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.2192.

[6] Frans Van Den Bergh y Andries P Engelbrecht. "A Cooperative Approach to Particle Swarm Optimization". En: 8.3 (2004), págs. 225-239.

[7] Kalyanmoy Deb, Ashish Anand y Dhiraj Joshi. "A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization". En: *Evolutionary Computation* 10.4 (2002), págs. 371-395. ISSN: 1063-6560. DOI: 10.1162/106365602760972767. URL: http://www.mitpressjournals.org/doi/10.1162/106365602760972767.

[8] John J. Grefenstette, ed. *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*. Cambridge, Massachusetts, USA: L. Erlbaum Associates Inc., 1987. ISBN: 0-8058-0158-8.

[9] Steven Gustafson y Edmund K Burke. "Speciating Island Model: An Alternative Parallel Evolutionary Algorithm". En: *Journal of Parallel and Distributed Computing - Special issue on parallel bioinspired algorithms* 66.May 2006 (2006), págs. 1025-1036.

[10] W E Hart y col. "Analysis of the Numerical Effects of Parallelism on a Parallel Genetic Algorithm". En: *IEEE Proc. of the Workshop on Solving Combinatorial Optimization Problems in Parallel* (1996), págs. 606-612. ISSN: 10636374.

[11] F. Herrera, M. Lozano y A. M. Sánchez. "Hybrid crossover operators for real-coded genetic algorithms: An experimental study". En: *Soft Computing* 9.4 (2005), págs. 280-298. ISSN: 14327643. DOI: 10.1007/s00500-004-0380-9.

[12] F. Herrera, M. Lozano y A. M. S??nchez. "A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study". En: *International Journal of Intelligent Systems* 18.3 (2003), págs. 309-338. ISSN: 08848173. DOI: 10.1002/int.10091.

[13] Giorgos Karafotias, Mark Hoogendoorn y A. E. Eiben. "Parameter Control in Evolutionary Algorithms: Trends and Challenges". En: *IEEE Transactions on Evolutionary Computation* 19.2 (2015), págs. 167-187. ISSN: 1089778X. DOI: 10.1109/TEVC.2014.2308294. arXiv: 0810.3356.

[14] Mohamed Kurdi. "An effective new island model genetic algorithm for job shop scheduling problem". En: *Computers and Operations Research* 67 (2016), págs. 132-142. ISSN: 03050548. DOI: 10.1016/j.cor.2015.10.005. URL: http://dx.doi.org/10.1016/j.cor.2015.10.005.

[15] Antonio LaTorre, Santiago Muelas y José María Peña. "A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: A scalability test". En: *Soft Computing* 15.11 (2011), págs. 2187-2199. ISSN: 14327643. DOI: 10.1007/s00500-010-0646-3.

[16] Xiaodong Li, Senior Member y Xin Yao. "Cooperatively Coevolving Particle Swarms for Large Scale Optimization". En: 16.2 (2012), págs. 210-224.

[17] J J Liang y col. *Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Learning-based Real-Parameter Single Objective Optimization*. November 2014. 2014. ISBN: 2096874622750. URL: http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/CEC-2015/Learning-Based%20Single%20Objective%20Optimization/Definitions%20of%20CEC2015%20benchmark%20learning-based%2020141228.pdf%7B%5C%%7D5Cnhttp://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/Fo.

[18] Tianjun Liao y Thomas Stutzle. "Benchmark results for a simple hybrid algorithm on the CEC 2013 benchmark set for real-parameter optimization". En: *2013 IEEE Congress on Evolutionary Computation, CEC 2013* (2013), págs. 1938-1944. DOI: 10.1109/CEC.2013.6557796.

[19] Daniel Molina y col. "Memetic algorithms for continuous optimisation based on local search chains." En: *Evolutionary computation* 18.1 (2010), págs. 27-63. ISSN: 1530-9304. DOI: 10.1162/evco.2010.18.1.18102. URL: http://www.ncbi.nlm.nih.gov/pubmed/20064025.

[20] Heinz Mühlenbein. "Evolution in time and space - the parallel genetic algorithm". En: *Foundations of Genetic Algorithms* (1991), págs. 316-337. DOI: 10.1.1.54.6763. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.81.1058.

[21] T. Nomura y K. Shimohara. "An Analysis of Two-Parent Recombinations for Real-Valued Chromosomes in an Infinite Population". En: *Evolutionary Computation* 9.3 (sep. de 2001), págs. 283-308. ISSN: 1063-6560. DOI: 10.1162/106365601750406000.

[22] Yew Soon Ong y Andy J. Keane. "Meta-Lamarckian learning in memetic algorithms". En: *IEEE Transactions on Evolutionary Computation* 8.2 (2004), págs. 99-110. ISSN: 1089778X. DOI: 10.1109/TEVC.2003.819944.

[23] Mitchell A Potter y Kenneth A De Jong. "A Cooperative Coevolutionary Approach to Function Optimization 2 Cooperative Coevolutionary Genetic Algorithms". En: (2004), págs. 249-257.

[24] Mitchell A Potter y Kenneth A De Jong. "Cooperative Coevolution : An Architecture for Evolving Coadapted Subcomponents". En: 8.1 (2000), págs. 1-29.

[25] Mike Preuss. "Niching the CMA-ES via nearest-better clustering". En: *Proceedings of the 12th annual conference comp on Genetic and evolutionary computation - GECCO '10* January 2010 (2010), pág. 1711. DOI: 10.1145/1830761.1830793. URL: http://portal.acm.org/citation.cfm?doid=1830761.1830793.

[26] a.K. Qin y P.N. Suganthan. "Self-adaptive differential evolution algorithm for numerical optimization". En: *2005 IEEE Congress on Evolutionary Computation* 2 (2005), págs. 1785-1791. DOI: 10.1109/CEC.2005.1554904.

[27] Roberto Ruiz, Jesús Aguilar-Ruiz y José Riquelme. "SOAP: Efficient Feature Selection of Numeric Attributes". En: *Advances in Artificial Intelligence—IBERAMIA 2002* 2527.May (2002), págs. 233-242. ISSN: 03029743. DOI: 10.1007/3-540-36131-6. URL: http://www.springerlink.com/content/najxguhp2gr6fn6d/.

[28] Yan-jun Shi, Hong-fei Teng y Zi-qiang Li. "Cooperative Co-evolutionary Differential Evolution for Function Optimization". En: *Advances in Natural Computation.* "Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, págs. 1080-1088.

[29] Ofer M. Shir y Thomas Bäck. "Niche Radius Adaptation in the CMA-ES Niching Algorithm". En: (2006), págs. 142-151. ISSN: 03029743. DOI: 10.1007/11844297_15. URL: http://link.springer.com/10.1007/11844297%7B%5C_%7D15.

[30] JE Jim E Smith y TC C Fogarty. "Operator and parameter adaptation in genetic algorithms". En: *Soft Computing* 1.2 (1997), págs. 81-87. ISSN: 1432-7643. DOI: 10.1007/s005000050009. URL: http://dx.doi.org/10.1007/s005000050009%7B%5C%%7D5Cnhttp://link.springer.com/article/10.1007/s005000050009.

[31] Catalin Stoean y Mike Preuss. "Disburdening the Species Conservation Evolutionary". En: *Differentiation* 13 (2007), págs. 1420-1427.

[32] Ryoji Tanabe y Alex Fukunaga. "Evaluation of a randomized parameter setting strategy for island-model evolutionary algorithms". En: *2013 IEEE Congress on Evolutionary Computation, CEC 2013* (2013), págs. 1263-1270. DOI: 10.1109/CEC.2013.6557710.

[33] Zhenguo Tu y Yong Lu. "A robust stochastic genetic algorithm (StGA) for global numerical optimization". En: *IEEE Transactions on Evolutionary Computation* 8.5 (2004), págs. 456-470. ISSN: 1089778X. DOI: 10.1109/TEVC.2004.831258.

[34] Rasmus K. Ursem. "Multinational evolutionary algorithms". En: *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999* 3 (1999), págs. 1633-1640. DOI: 10.1109/CEC.1999.785470.

[35] Darell Whitley. "An Executable Model of a simple Genetic Algorithm". En: *Foundations of Genetic Algorithms 2* (1993), págs. 45-62.