Descomposición jerárquica no homogenea de nichos basados en regiones

Daniel Molina Francisco Herrera

Resumen—
Es muy importante mantener de forma adecuada la diversidad de la población de los algoritmos evolutivos durante la búsqueda. Para mantener la diversidad es común el uso de estrategia de nichos, como la técnica de *clearing*. En trabajos anteriores se propuso una técnica de nichos basada en regiones que divide el espacio de búsqueda en hipercubos predefinidos de igual tamaño. Los algoritmos propuestos reducían el tamaño de todos los hipercubos durante la búsqueda para aumentar el grado de explotación. En este trabajo proponemos un mecanismo de división del espacio de búsqueda no homogéneo, dividiendo más las regiones más prometedoras. Probamos el nuevo mecanismo de descomposición obteniendo mejores resultados, gracias a una explotación más eficiente.

Palabras clave— **Optimización multimodal, algoritmo memético, estrategia de nichos, landscape.**

I. Introducción

En multitud de problemas reales no existe una única solución óptima, si no múltiples opciones. Para estos problemas, es interesante poder obtener no sólo una solución óptima, si no tantas soluciones óptimas como sea posible. A este objetivo se le denomina optimización multimodal, y es un campo de investigación cada vez más popular.

Los Algoritmos Evolutivos, AEs, inicialmente diseñados para buscar un único óptimo, pueden ser adaptados a este objetivo mediante técnicas que incrementan la diversidad de la población para explorar mejor el espacio de búsqueda. Dichas técnicas, denominadas estrategias de nichos [1], están diseñadas para mantener subgrupos de individuos dentro de la población para permitir localizar distintos óptimos.

Lamentablemente, las técnicas tradicionales, como el *clearing* [2], presentan una excesiva dependencia del tamaño de los nichos, influído por la distancia entre ellos, y y el tamaño de la población [3], [2]. Aunque existen algunos trabajos para calcular los nichos de forma automática [4], para poder fijarlos de forma adecuada suele ser necesario conocer tanto el número de óptimos como la distancia entre ellos. Dado que esos datos no son conocidos en problemas reales, se hace necesario el desarrollo de técnicas menos dependientes de dichos parámetros.

Los Algoritmos Meméticos, MAs [5], al ser hibridaciones entre AEs y métodos de mejora local, es una opción interesantes para abordar problemas de optimización multimodal, ya que son algoritmos muy eficientes y el componente del AE puede modificarse fácilmente para crear subpoblaciones para explorar diferentes áreas del espacio de búsqueda. Además, la gran capacidad de explotación de la búsqueda local, BL, pueden permitirnos identificar múltiples óptimos, como es propuesto en [6], o en [7].

Universidad de Cádiz. E-mail: daniel.molina@uca.es
Universidad de Granada. E-mail: herrera@decsai.ugr.es

En un trabajo anterior [8], diseñamos un AM para optimización continua, denominado RMA-LSCh, que ya usaba una nueva propuesta de nichos, denominados regiones. Dicha técnica particiona el espacio de búsqueda en hipercubos denominados regiones, y se permite únicamente una solución por región. Sin embargo, El algoritmo hacía el uso de regiones se usaba para fomentar la exploración mediante mayor diversidad, separando mejor los esfuerzos de exploración y explotación del AE y la BL, pero mantenía como objetivo la obtención de un único óptimo, no para conseguir múltiples óptimos.

En este trabajo proponemos un nuevo algoritmo, AM basado en regiones usando un archivo, en inglés *Region-based Memetic Algorithm with Archive*, RMAwA, especialmente diseñado para optimización multimodal. Las principales características de RMAwA es el uso combinado de la técnica de nichos basados en regiones, junto con el uso de un mecanismo de identificación de óptimos, mediante una memoria externa o archivo. Dicho archivo almacena los óptimos encontrados para limitar la dependencia del tamaño de la población, como se ha realizado anteriormente en la literatura [9], [10]. Además, este archivo permite al algoritmo eliminar del espacio de búsqueda regiones consideradas suficientemente exploradas. Para ello, el archivo mantiene un índice de dichas regiones, para evitar, de forma sencilla y eficiente, que el AE genere nuevas soluciones en alguna de dichas regiones.

Se han realizado distintos estudios para demostrar que el uso de las regiones combinado con un archivo ofrece una mejora para optimización multimodal, y que RMAwA es un algoritmo competitivo frente a los comparados.

Este trabajo se organiza de la siguiente manera. En la Sección II, presentamos en detalle la propuesta. En la Sección III, planteamos el entorno experimental, y los parámetros de experimentación del algoritmo. En la Section IV presentamos los resultados obtenidos, y comparamos con los resultados obtenidos por otros algoritmos. Finalmente, en la Sección V, planteamos las principales conclusiones del trabajo.

II. Algoritmo memético basado en regiones con archivo

Presentamos un nuevo AM basado en regiones con archivo externo, *region-based MA with archive*, RMAwA, que resulta de adaptar un algoritmo previo, RMA-LSCh-CMA [8], para optimización multimodal.

En optimización multimodal es posible utilizar una memoria o archivo externo para mantener las distintas soluciones óptimas encontradas y evitar que se tengan que mantener en la población, permitiendo que el número de óptimos que el algoritmo pueda encontrar no esté limitado por el tamaño de la población [10]. Esto es crucial ya que el número de óptimos posibles suele ser desconocido. Un aspecto novedoso de RMAwA es que usa también dicha memoria como un índice de las regiones ya totalmente exploradas para eliminarlas del espacio de búsqueda.

En esta sección describimos la estrategia de *niching* basado en regiones, y el diseño y uso de la memoria externa, para terminar explicando cómo se integran para componer el algoritmo completo.

A. Estrategia de niching basado en regiones

En [8] redefinimos la noción de nicho de ser el área circundante alrededor de una solución a ser una división fija del espacio de búsqueda. Para ello, el espacio de búsqueda de cada dimensión se divide en ND partes homogéneas creando así una malla de hipercubos o regiones de igual tamaño. Cada región así creada representa un nicho. Lo ideal sería que cada región representase una base de atracción para la BL, pero no se puede garantizar, por lo que el tamaño de las regiones no se mantiene constante, para obtener mayor robustez en el algoritmo. La Figura 1 visualiza la fragmentación, en donde s_n son vectores de números reales que representan las soluciones alcanzadas, y r_n son vectores de números enteros que identifican la regiones asociadas. La ventaja de definir r_n es que nos permite identificar fácilmente los nichos existentes evitando el coste computacional de calcular las distancias entre las soluciones.

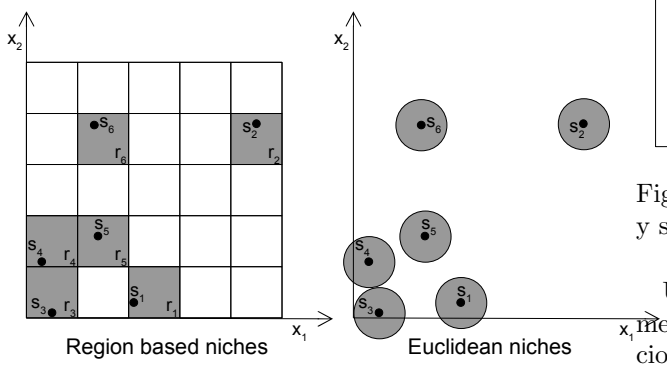

Fig. 1: Distinta estrategia de *niching*

En la estrategia propuesta cada vez que el AE genera una nueva solución se comprueba si existe ya en su región una solución y en ese caso la reemplaza únicamente si posee mejor *fitness*. Este comportamiento es similar a la estrategia de niching *clearing* en el sentido que las soluciones de un mismo nicho compiten entre si, pero se diferencia en la definición del nicho: de una representación de distancia a una representación en regiones.

B. La memoria externa

Como ya se ha indicado, RMAwA utiliza una memoria externa para almacenar las soluciones consideradas como optimizadas (consideramos optimizadas aquellas mejoradas mediante la BL) y que establece un índice de regiones que no se desea seguir explorando.

La memoria se compone de dos colecciones sin tamaño limitado. La primera es una lista de todas las soluciones generadas. La segunda es un índice ordenado de las regiones almacenadas. Este índice permite recuperar de forma eficiente las regiones exploradas, para marcarlas y no intentar explorarlas nuevamente, para evitar desperdiciar esfuerzo. Para hacer muy eficiente tanto la inserción como la búsqueda, dicho índice tiene estructura de un árbol binario balanceado.

En la Figura 2 se muestra un ejemplo de la estructura de la memoria que representa las regiones mostradas en la Figura 1. En la figura se muestra cómo se inserta nueva nueva solución s_n en la región asociada, r_n. La solución se almacena en la lista de soluciones, mientras que su identificador de región se introduce en él arbol binario de regiones (en donde una región sólo se almacena una única vez).

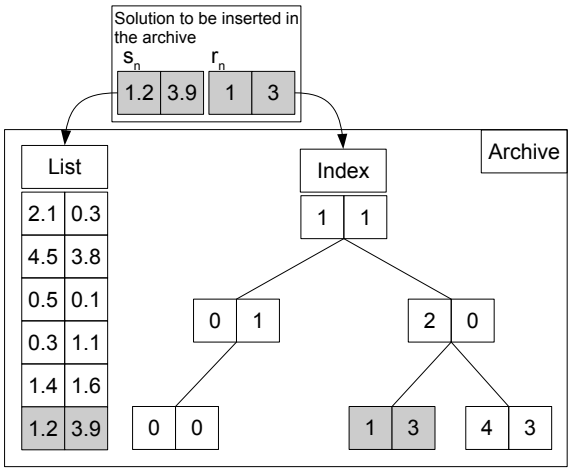

Fig. 2: Ejemplo de la representación de la memoria y sus índices para el problema 2-D de la Figura 1

Un aspecto fundamental del algoritmo es que la memoria debe de usarse para almacenar las soluciones óptimas ya exploradas, para poder marcas sus regiones asociadas para no ser exploradas nuevamente. Sin embargo, conocer si una solución es un

óptimo es complicado si no se conoce el óptimo de la función (algo desconocido en problemas reales). Para ello, nuestro algoritmo hace uso de la BL, al considerar que una solución es un óptimo (local o global) cuando ya no mejora, o no de forma suficiente, al aplicarse la BL. Una mejora no suficiente es cuando la mejora es inferior a δ_{LS}^{min}. las regiones asociadas con óptimos se marcan para no ser exploradas nuevamente. Adicionalmente, para evitar aplicar innecesariamente la BL, la memoria almacena no sólo la solución final de aplicar la BL, si no también la solución que sirve de punto de inicio de cada BL.

En resumen, la memoria externa tiene dos funcionalidades: Por un lado, almacena las soluciones óptimas como aquellas no mejorables por la BL, permitiendo no perderlas; y, por el otro, gestiona un índice de regiones para identificar rápidamente cuando ya existe una solución en una región, permitiendo marcar las regiones con óptimos para no explorarlas nuevamente y evitar esfuerzos innecesarios.

C. Esquema general

RMAwA utiliza un esquema similar a RMA-LSCh-CMA propuesto en [8], en donde se aplica de forma alternada el AE y la BL. El AE se aplica durante I_{AE} evaluaciones, mientras que la BL se aplica al mejor individuo de la población que no se aplicase anteriormente, hasta que la BL no obtenga mejoras significativas. El no limitar la BL (más allá del número máximo de evaluaciones permitido) es la principal diferencia con respecto a RMA-LSCh-CMA.

Al igual que en RMA-LSCh-CMA, el tamaño de la región va disminuyendo durante la búsqueda para evitar que sea un parámetro crítico del algoritmo. Al incrementar el número de regiones la memoria se reestructura, no se borra. Para incrementarlo, el número de división por dimensión, ND, se incrementa u veces durante la búsqueda. Por tanto, dado que el algoritmo se ejecuta durante Max_{FEs} evaluaciones, cada $Max_{FEs}/(u+1)$ evaluaciones $ND_i = m_u \cdot ND_{i-1}$ donde m_u es el multiplicador. El modelo general puede verse en Algoritmo 1.

Algorithm 1 Pseudo-código de RMAwA

```
Inicializa población aleatoriamente de forma uniforme.
while Max_FEs no se alcance do
Aplica el AE durante i_EA evaluaciones
s_best ← mejor individuo de la población
Aplica la BL sobre s_best
if se debe incrementar el número de regiones then
ND_i = m_u · ND_{i-1}
Actualiza los índices de la memoria
end if
end while
```

D. Componente evolutivo (AE)

El AE utiliza es el mismo que en el RMA-LSCh-CMA, un algoritmo genético generacional con tamaño NP y cruce $BLX-\alpha$. Consulte el [8] para conocer más detalles. Existe, sin embargo, una gran diferencia. Por cada nueva solución se calcula su región r_n y se consulta el índice de la memoria. Si ya existe dicha región representada por otra solución, ambas se comparan y la nueva solución se introduce únicamente si mejora a la existente. Si r_n no existía entonces s_n compite con la peor solución de la población para reemplazarla. Este proceso se describe en el Algoritmo 2.

Algorithm 2 Pseudo-código del AE en RMAwA

```
i = 0
while i < i_EA do
Selecciona dos padres
repeat
Crea una solución s_n mediante cruce y mutación
Calcula la región r_n en donde pertenece s_n
until r_i se deba explorar
Evalúa s_n, i = i + 1
Recupera de la población conjunto de soluciones S_{r_n} de la región r_n
if S_{r_n} ≠ ∅ then
S_{r_n} ← S_{r_n} ∪ s_n
Borra peor individuo de S_{r_n}
else
Reeplaza el peor individuo s_worst de la población si f(s_worst) > f(s_n)
end if
end while
```

E. El método de BL

El método de BL aplicado es el popular CMA-ES [11], ya que obtiene muy buenos resultados con una convergencia muy rápida. A diferencia de RMA-LSCh-CMA, RMAwA aplica la BL hasta que converge por lo que no aplica un encadenamiento de BLs (no tiene sentido aplicarla de nuevo si ha convergido). La BL considera que converge si no mejora significativamente en las últimas i_{BL} evaluaciones.

La BL se aplica sobre la mejor solución s_{best} de la población, y una vez que se ha aplicado la BL la solución se introduce en la memoria externa y se elimina de la población, reemplazándose por una solución aleatoria.

III. Entorno experimental

Los experimentos se han realizado usando el *benchmark* propuesto por la sesión especial de optimización multimodal del congreso de IEEE de Computación Evolutiva de 2013 (CEC'2013) [12].

A continuación describimos las principales características del *benchmark* usado, puede consultar [12] para obtener más información.

El *benchmark* está compuesto por 20 problemas a optimizar, compuesta por 12 funciones distintas, para los cuales se desea obtener el mayor número de óptimos globales. Para abordarlos el algoritmo debe de usar los mismos parámetros para todos ellos. La Tabla I muestra para cada problema su función, dimensión, y número de óptimos. Como se ve, es un *benchmark* bastante heterogéneo en el número de óptimos por problema.

TABLA I: Problemas del *benchmark CEC'2013*

Problema	Función	D	Óptimos	Max_{FEs}
F_1	f_1	1	2	$5 \cdot 10^4$
F_2	f_2	1	5	$5 \cdot 10^4$
F_3	f_3	1	1	$5 \cdot 10^4$
F_4	f_4	2	4	$5 \cdot 10^4$
F_5	f_5	2	2	$5 \cdot 10^4$
F_6	f_6	2	18	$2 \cdot 10^5$
F_7	f_7	2	36	$2 \cdot 10^5$
F_8	f_6	3	81	$4 \cdot 10^5$
F_9	f_7	3	216	$4 \cdot 10^5$
F_{10}	f_8	2	12	$2 \cdot 10^5$
F_{11}	f_9	2	6	$2 \cdot 10^5$
F_{12}	f_{10}	2	8	$2 \cdot 10^5$
F_{13}	f_{11}	2	6	$2 \cdot 10^5$
F_{14}	f_{11}	3	6	$4 \cdot 10^5$
F_{15}	f_{12}	3	8	$4 \cdot 10^5$
F_{16}	f_{11}	5	6	$4 \cdot 10^5$
F_{17}	f_{12}	5	8	$4 \cdot 10^5$
F_{18}	f_{11}	10	6	$4 \cdot 10^5$
F_{19}	f_{12}	10	6	$4 \cdot 10^5$
F_{20}	f_{12}	20	8	$4 \cdot 10^5$

Cada algoritmo se ejecuta 50 veces para cada problema, y se calculará el porcentaje de óptimos alcanzado, *peak ratio* (PR) (calculado por el código del propio *benchmark*). PR es la media del porcentaje de óptimos obtenidos durante las distintas evaluaciones, calculado mediante la expresión:

$$PR = \frac{\sum_{i=1}^{NR} NPF_i}{NKP \cdot NR} \quad (1)$$

en donde NPF_i es el número de óptimos globales obtenidos en la ejecución *ith*, NKP es el número de óptimos globales, y NR es el número de ejecuciones ($NR = 50$).

Este valor PR es calculado considerando cinco valores distintos de precisión $\epsilon = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. Dicho valor de precisión es la diferencia que se admite que tenga el fitness de una solución respecto al óptimo para considerar que dicha solución es una solución óptima.

Las comparaciones entre algoritmos se han realizado de forma independiente para los distintos valores de precisión, y utilizando tests estadísticos no-paramétricos [13]. En particular, usamos el tests de Wilcoxon para comparar dos algoritmos.

Dado que obtener los valores adecuados para los parámetros de una nueva propuesta es una labor muy tediosa, hemos utilizado para ello una herramienta de configuración automática, IRACE [14]. En la Tabla II definimos para cada parámetro el rango de búsqueda indicada a IRACE y el valor final recomendado por éste.

TABLA II: Parámetros del algoritmo

Parámetro	**Rango**	**Obtenido**
i_{EA}	$[100, 1000]$	550
i_{LS}	$[100, 1000]$	150
ND_0	$[2, 10]$	2
u	$[2, 5]$	4
m_u	$[1, 5]$	1.7
NP	$[40, 120]$	70
α	$[0.1, 0.9]$	0.9

Se puede observar que el número de regiones aumenta multiplicando cinco veces por 1.7, por lo que la secuencia de división es $[2, 4, 7, 13, 23, 40]$. Otro aspecto reseñable es el alto valor del parámetro α (0.9) por lo que se fomenta mucha exploración durante el AE.

Para los parámetros definidos en la Tabla III se han usado los valores por defecto de los trabajos respectivos. δ_{LS}^{min} define la precisión final de la búsqueda y le es asignado 10^{-6}.

TABLA III: Otros Parámetros

Parámetro	**Descripción**	**Valor**
λ	Tamaño de población de CMA-ES $p = 4 + \lambda ln(D)$	3 [11]
μ	Número de padres usado por CMA-ES p/μ	2 [11]
α	Parámetro del cruce $BLX - \alpha$	0.5 [8]
NAM_{size}	Selección NAM	3 [8]
δ_{LS}^{min}	Umbral de parada de BL	10^{-6}

IV. Resultados experimentales

Una vez definido el entorno experimental, estudiamos el comportamiento de nuestra propuesta y comparamos sus resultados con los obtenidos por otros algoritmos de la literatura. Primero, vamos a estudiar cómo el uso de regiones y de la memoria externa permiten mejorar tanto los resultados como el rendimiento. Luego comparamos la propuesta con otros algoritmos de la literatura para valorar sus resultados.

A. Influencia de las regiones y la memoria

Vamos a comparar si la nueva definición de nichos usando regiones (Region-MA) mejora la definición de nichos usando distancia euclídea (Euclídea-MA). Para ello, comparamos sin usar la memoria externa. Para comparar nichos de igual tamaño, en Euclídea-MA cada solución compara con las soluciones den-

tro del radio de nicho σ definido como la mitad del tamaño de la región.

Para simplificar los resultados, comparamos únicamente con una precisión $\epsilon = 10^{-5}$, aplicando el tests de Wilcoxon. La tabla IV muestra el resultado. Se puede observar que se obtienen resultados estadísticamente mejores usando regiones. Además, al no tener que calcular distancias, se consigue una mejora en tiempo media del 17.4%.

TABLA IV: Resultados del test de Wilcoxon comparando Region-MA y Euclídea-MA (para $\epsilon = 10^{-5}$)

R+ Region-MA	R- Euclídea-MA	p-value
189	21	0.0008

También vamos a comprobar que usar la memoria externa para evitar explorar regiones ya exploradas por el AE mejora, y no empeora, los resultados. Para ello vamos a comparar dos versiones del algoritmo. El primero es el descrito en la Sección II, mientras que el segundo es una versión sin la comprobación del AE del paso 6 del Algoritmo 2. Por tanto, comparamos el algoritmo propuesto con una memoria excluyente (RMA with archive, RMAwA) con otra versión del RMA con una simple memoria (RMA with a simple Archive, RMAwSA). Al igual que en la comparativa anterior, analizamos los resultados únicamente para el mayor grado de precisión ($\epsilon = 10^{-5}$).

TABLA V: PRs de RMA con archivo excluyente (RMAwA) y no excluyente (RMAwSA) para $\epsilon = 10^{-5}$

Problema	F_1	F_2	F_3	F_4	F_5
RMAwA	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
RMAwSA	**1.00**	0.31	**1.00**	**1.00**	**1.00**
Problema	F_6	F_7	F_8	F_9	F_{10}
RMAwA	0.00	**0.92**	0.82	**0.51**	**1.00**
RMAwSA	0.00	0.66	**0.91**	0.34	0.98
Problema	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
RMAwA	**1.00**	**1.00**	**0.99**	**0.81**	**0.70**
RMAwSA	0.67	0.93	0.67	0.67	0.65
Problema	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}
RMAwA	**0.67**	**0.66**	**0.23**	**0.13**	**0.12**
RMAwSA	0.67	0.32	0.18	0.12	0.12

La Tabla V muestra los *PRs* obtenidos para cada versión del algoritmo. Se puede observar que usar la memoria para evitar regiones totalmente exploradas hace mejorar los resultados de forma significativa (como muestran los resultados del test de Wilcoxon mostrados en la Tabla VI).

TABLA VI: Tests de Wilcoxon comparando PR del RMA con y sin memoria excluyente (para $\epsilon = 10^{-5}$)

R+ RMAwA	R- RMAwSA	p-value
186.5	23.5	0.00132

B. Comparando con otros algoritmos

En este apartado comparamos los resultados obtenidos por nuestro algoritmo, RMAwA con los obtenidos por otros algoritmos de la literatura presentados en la competición del CEC'2013:

- PNA-NSGAII [15] : este algoritmo aplica la optimización multimodal como si fuese un problema bi-objetivo, en el que se añade como objetivo la diversidad de la población.
- dADE/nrand [9] : un DE con mutación basada en vecindario y una memoria dinámica.
- CrowdingDE [16] : un DE con método de multitud (*crowding*) para evitar convergencia prematura.
- DE/nrand [17] : un DE con mutación basada en vecindario.
- NCDE [18] : un DE con mutación basada en vecindario y mecanismo *crowding*.
- r3pso [19] : un PSO con topología de vecindario en anillo.

Los primeros dos algoritmos participaron en la competición del CEC'2013 mientras que los dos siguientes se dieron como referencia por los organizadores. Los dos últimos son algoritmos de nichos adicionales. Los resultados detallados de cada algoritmo puede consultarse en el Apéndice.

Primero vamos a analizar el rendimiento de cada algoritmo para cada valor de precisión. Para ello, la Tabla VII muestra el ranking medio de cada algoritmo para cada nivel de precisión, y en la tabla VIII la comparativa de Wilcoxon de RMAwA contra cada uno de los demás algoritmos.

TABLA VII: Ranking medio para cada algoritmo y nivel de precisión

Nivel Precisión	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
CrowdingDE	4.3	4.3	4.5	4.7	4.7
DE/nrand	5.1	4.2	4.1	3.6	3.4
r3pso	5.0	5.8	6.0	6.0	6.2
NCDE	4.0	4.4	4.3	4.4	4.5
PNA-NSGAII	3.4	3.7	3.8	4.1	3.9
dADE/nrand	**3.05**	3.00	3.0	3.0	3.1
RMAwA	3.3	**2.7**	**2.3**	**2.2**	**2.2**

Podemos ver que RMAwA es el segundo mejor para $\epsilon = 10^{-1}$ aunque no hay diferencias significativas con el mejor. Luego, la superioridad de RMAwA mejora conforme aumenta el nivel de precisión hasta que es significativamente el mejor para $\epsilon = 10^{-4}, 10^{-5}$, lo cual sugiere que le afecta menos el

TABLA VIII: Test de Wilcoxon comparando el PRs de RMAwA ($R+$) with el resto ($R-$)

$\epsilon = 10^{-1}$			
RMAwA vs	$R+$	$R-$	p-value
CrowdingDE	150.5	59.5	**0.0935**
DE/nrand	179.5	30.5	**0.0039**
r3pso	192.5	17.5	**0.0004**
NCDE	128.5	65	0.2273
PNA-NSGAII	97	95.5	0.9839
dADE/nrand	68.5	125	0.2862
$\epsilon = 10^{-2}$			
RMAwA vs	$R+$	$R-$	p-value
CrowdingDE	172	20.5	**0.0016**
DE/nrand	173.5	36.5	**0.0089**
r3pso	205	5	$\mathbf{1.910^{-5}}$
NCDE	199.5	10.5	**0.0001**
PNA-NSGAII	135.5	74.5	0.2549
dADE/nrand	125.5	84.5	0.4441
$\epsilon = 10^{-3}$			
RMAwA vs	$R+$	$R-$	p-value
CrowdingDE	185	7.5	**0.0001**
DE/nrand	162	30.5	**0.0077**
r3pso	188.5	3	**0.0000**
NCDE	185	7.5	**0.0001**
PNA-NSGAII	141	51.5	**0.0837**
dADE/nrand	146.5	63.5	0.1279
$\epsilon = 10^{-4}$			
RMAwA vs	$R+$	$R-$	p-value
CrowdingDE	205	5	$\mathbf{1.910^{-5}}$
DE/nrand	162.5	31	**0.0082**
r3pso	188.5	3	$\mathbf{1.910^{-5}}$
NCDE	185	7.5	**0.0001**
PNA-NSGAII	158	52	**0.0484**
dADE/nrand	165.5	44.5	**0.0227**
$\epsilon = 10^{-5}$			
RMAwA vs	$R+$	$R-$	p-value
CrowdingDE	185	7.5	**0.0001**
DE/nrand	160.5	33	**0.0108**
r3pso	188.5	3	$\mathbf{1.910^{-5}}$
NCDE	199.5	10.5	**0.0001**
PNA-NSGAII	152	40.5	**0.0274**
dADE/nrand	151.5	42	**0.0323**
$\epsilon = *$			
RMAwA vs	$R+$	$R-$	p-value
CrowdingDE	4531	431.5	**0**
DE/nrand	4197.5	768	**2.55E−9**
r3pso	4844.5	115	**0**
NCDE	4502	462.5	**0**
PNA-NSGAII	3427	1535.5	**0.0010**
dADE/nrand	3288	1762	**0.0087**

valor de precisión exigido que a los demás algoritmos. Por tanto, la propuesta de este trabajo, RMAwA ofrece un rendimiento significativamente mejor que el resto de algoritmos.

V. Conclusiones

En este trabajo hemos presentado un AM basado en regiones con una memoria externa. Dicha memoria permite evitar seguir explorando regiones con un óptimo local, al considerarse suficientemente exploradas.

Hemos demostrado no sólo que el uso de regiones para *niching* es más eficiente que usando la distancia euclídea, si no también que el uso de la memoria externa para no re-explorar regiones permite mejorar la capacidad de identificar múltiples óptimos reduciendo el coste computacional.

Finalmente, hemos comparado el algoritmo propuesto con otros algoritmos de referencia, observando que nuestro algoritmo era poco dependiente del nivel de precisión exigido, obteniendo resultados estadísticamente mejores que el resto de algoritmos comparados.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Ciencia e Innovacion por el proyecto TIN2011-28488. Benjamin Lacroix agradece el apoyo de la red MIBISOC, la Initial Training Network financiada por la Unión Europea bajo el proyecto PITN-GA-2009-238819.

Referencias

[1] S. Das, S. Maity, Q.Y. Qu, and P.N. Suganthan, "Real-parameter evolutionary multimodal optimization — a survey of the state-of-the-art," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 71 – 88, 2011.

[2] A. Petrowski, "A clearing procedure as a niching method for genetic algorithms," in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, 1996, pp. 798–803.

[3] D.E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, Hillsdale, NJ, USA, 1987, pp. 41–49, L. Erlbaum Associates Inc.

[4] M.M.H. Ellabaan and Y.S. Ong, "Valley-adaptive clearing scheme for multimodal optimization evolutionary search," in *Intelligent Systems Design and Applications, 2009. ISDA '09. Ninth International Conference on*, 2009, pp. 1–6.

[5] P Moscato, "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Toward Memetic Algorithms," Tech. Rep., 1989.

[6] J.E. Vitela and O. Castaños, "A sequential niching memetic algorithm for continuous multimodal function optimization," *Applied Mathematics and Computation*, vol. 218, no. 17, pp. 8242 – 8259, 2012.

[7] B.Y. Qu, J.J. Liang, and P.N. Suganthan, "Niching particle swarm optimization with local search for multimodal optimization," *Information Sciences*, vol. 197, no. 0, pp. 131 – 143, 2012.

[8] B. Lacroix, D. Molina, and F. Herrera, "Region based memetic algorithm for real-parameter optimisation," *Information Sciences*, vol. 262, no. 0, pp. 15 – 31, 2014.

[9] M.G. Epitropakis, X. Li, and E.K. Burke, "A dynamic archive niching differential evolution algorithm for multimodal optimization," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, 2013, pp. 79–86.

[10] Z. Zhai and X. Li, "A dynamic archive based niching particle swarm optimizer using a small population size," in *Proceedings of the Thirty-Fourth Australasian Computer Science Conference - Volume 113*, Darlinghurst, Australia, Australia, 2011, ACSC'11, pp. 83–90, Australian Computer Society, Inc.

[11] N Hansen, S D Müller, and P Koumoutsakos, "Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)," *Evolutionary Computation*, vol. 1, no. 11, pp. 1–18, 2003.

[12] X. Li, A. Engelbrecht, and M Epitropakis, "Benchmark functions for cec 2013 special session and competition on niching methods for multimodal function optimization," Tech. Rep., Royal Melbourne Institute of Technology, 2013.

[13] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[14] M. Lopez-Ibañez, J. Dubois-Lacoste, T. Stüetzle, and M. Birattari, "The irace package, iterated race for automatic algorithm configuration," Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.
[15] S. Bandaru and K. Deb, "A parameterless-niching-assisted bi-objective approach to multimodal optimization," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, 2013, pp. 95–102.
[16] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *Evolutionary Computation, 2004. CEC2004. Congress on*, 2004, vol. 2, pp. 1382–1389 Vol.2.
[17] M.G. Epitropakis, V.P. Plagianakos, and M.N. Vrahatis, "Finding multiple global optima exploiting differential evolution's niching capability," in *Differential Evolution (SDE), 2011 IEEE Symposium on*, 2011, pp. 1–8.
[18] B. Y. Qu, P.N. Suganthan, and J.J. Liang, "Differential evolution with neighborhood mutation for multimodal optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 16, no. 5, pp. 601–614, 2012.
[19] X. Li, "Niching without niching parameters: Particle swarm optimization using a ring topology," *Evolutionary Computation, IEEE Transactions on*, vol. 14, no. 1, pp. 150–169, 2010.

Apéndices

I. Valores PR de los algoritmos

Esta sección muestra los valores PR del benchmark de CEC'2013 obtenidos por:

- RMAwA (Tabla IX)
- DE/nrand (Tabla X)
- CrowdingDE (Tabla XI)
- dADE/nrand (Tabla XII).
- PNA-NSGAII (Tabla XIII)
- NCDE (Tabla XIV)
- r3PSO (Tabla XV)

TABLA IX: RMAwA

	Nivel de precisión				
Prob.	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
F_1	1	1	1	1	1
F_2	1	1	1	1	1
F_3	1	1	1	1	1
F_4	1	1	1	1	1
F_5	1	1	1	1	1
F_6	0.99	0.99	0.99	0.99	0
F_7	1	0.92	0.92	0.92	0.92
F_8	0.82	0.82	0.82	0.82	0.82
F_9	1	0.52	0.52	0.51	0.51
F_{10}	1	1	1	1	1
F_{11}	1	1	1	1	1
F_{12}	1	1	1	1	1
F_{13}	1	1	1	1	1
F_{14}	0.82	0.81	0.81	0.81	0.81
F_{15}	0.71	0.7	0.7	0.7	0.7
F_{16}	0.68	0.67	0.67	0.67	0.67
F_{17}	0.67	0.66	0.66	0.66	0.66
F_{18}	0.38	0.24	0.24	0.23	0.23
F_{19}	0.13	0.13	0.13	0.13	0.13
F_{20}	0.25	0.13	0.13	0.13	0.13

TABLA X: DE/nrand

	Nivel de precisión				
Prob.	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
F_1	1	1	1	1	1
F_2	1	1	1	1	1
F_3	1	1	1	1	1
F_4	1	1	1	1	1
F_5	1	1	1	1	1
F_6	0.45	0.44	0.44	0.43	0
F_7	0.35	0.35	0.35	0.34	0.33
F_8	0.11	0.11	0.11	0.11	0.11
F_9	0.1	0.1	0.1	0.1	0.09
F_{10}	1	1	1	1	1
F_{11}	0.68	0.67	0.68	0.67	0.67
F_{12}	0.86	0.84	0.82	0.82	0.78
F_{13}	0.67	0.67	0.67	0.67	0.67
F_{14}	0.67	0.67	0.67	0.67	0.67
F_{15}	0.52	0.54	0.51	0.5	0.51
F_{16}	0.68	0.66	0.66	0.66	0.66
F_{17}	0.35	0.33	0.3	0.29	0.29
F_{18}	0.4	0.34	0.32	0.27	0.25
F_{19}	0.3	0.22	0.2	0.17	0.17
F_{20}	0.13	0.13	0.13	0.13	0.12

TABLA XI: Crowding DE/RAND/bin

	Nivel de precisión				
Prob.	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
F_1	1	0.71	0.09	0.02	0
F_2	1	1	1	1	1
F_3	1	1	1	1	1
F_4	1	1	1	1	0.42
F_5	1	1	1	1	1
F_6	1	1	0.97	0.11	0
F_7	0.7	0.72	0.72	0.71	0.72
F_8	0.85	0.84	0.72	0.29	0.04
F_9	0.27	0.27	0.27	0.27	0.27
F_{10}	1	1	1	1	1
F_{11}	0.94	0.69	0.67	0.67	0.67
F_{12}	0.38	0.06	0.01	0.01	0
F_{13}	0.84	0.68	0.67	0.67	0.67
F_{14}	0.68	0.67	0.67	0.67	0.67
F_{15}	0.73	0.69	0.63	0.49	0.38
F_{16}	0.7	0.67	0.67	0.67	0.67
F_{17}	0.08	0	0	0	0
F_{18}	0.08	0	0	0	0
F_{19}	0	0	0	0	0
F_{20}	0.5	0.01	0	0	0

TABLA XII: dADE/nrand

	Nivel de precisión				
Prob.	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
F_1	1	1	1	1	1
F_2	1	1	1	1	1
F_3	1	1	1	1	1
F_4	1	1	1	1	1
F_5	1	1	1	1	1
F_6	1	1	1	0.98	0
F_7	1	0.96	0.89	0.82	0.73
F_8	0.99	0.98	0.98	0.97	0.95
F_9	0.84	0.6	0.55	0.43	0.36
F_{10}	1	1	1	1	1
F_{11}	0.89	0.67	0.67	0.67	0.67
F_{12}	1	0.89	0.75	0.74	0.73
F_{13}	0.74	0.67	0.67	0.67	0.67
F_{14}	0.92	0.67	0.67	0.67	0.67
F_{15}	1	0.62	0.62	0.63	0.62
F_{16}	0.87	0.67	0.67	0.67	0.67
F_{17}	0.94	0.47	0.42	0.4	0.41
F_{18}	0.68	0.66	0.63	0.63	0.63
F_{19}	0.42	0.14	0.06	0.02	0
F_{20}	0	0	0	0	0

TABLA XIII: PNA-NSGAII

	Nivel de precisión				
Prob.	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
F_1	1	1	1	1	1
F_2	1	1	1	1	1
F_3	1	1	1	1	1
F_4	1	1	1	0.99	0.81
F_5	1	1	1	1	1
F_6	0.56	0.54	0.52	0.47	0
F_7	1	0.74	0.73	0.71	0.68
F_8	0.35	0.33	0.31	0.28	0.25
F_9	0.48	0.33	0.32	0.3	0.28
F_{10}	1	1	1	1	1
F_{11}	0.88	0.68	0.67	0.68	0.66
F_{12}	0.75	0.72	0.67	0.64	0.57
F_{13}	0.7	0.67	0.67	0.66	0.62
F_{14}	0.93	0.67	0.67	0.66	0.61
F_{15}	0.67	0.5	0.49	0.47	0.44
F_{16}	1	0.52	0.52	0.42	0.32
F_{17}	0.92	0.35	0.34	0.3	0.25
F_{18}	0.64	0.12	0.11	0.11	0.09
F_{19}	0.02	0.02	0.04	0.02	0.01
F_{20}	0	0	0	0	0

TABLA XIV: NCDE

	Nivel de precisión				
Prob.	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
F_1	1	1	1	1	1
F_2	1	1	1	1	1
F_3	1	1	1	1	1
F_4	1	1	1	1	0.61
F_5	0	0	0	0	0
F_6	1	0.99	0.8	0.06	0
F_7	1	0.94	0.94	0.94	0.93
F_8	1	1	1	1	0.99
F_9	0.55	0.55	0.55	0.55	0.55
F_{10}	1	1	1	1	1
F_{11}	0.97	0.72	0.68	0.67	0.67
F_{12}	0.72	0.43	0.23	0.09	0.03
F_{13}	0.74	0.67	0.67	0.67	0.66
F_{14}	0.83	0.67	0.67	0.67	0.67
F_{15}	0.64	0.38	0.38	0.37	0.37
F_{16}	1	0.67	0.67	0.67	0.67
F_{17}	0.66	0.25	0.25	0.25	0.25
F_{18}	1	0.45	0.39	0.35	0.33
F_{19}	0.58	0.24	0.2	0.17	0.03
F_{20}	0.35	0.13	0	0	0

TABLA XV: r3PSO

	Nivel de precisión				
Prob.	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
F_1	0.95	0.27	0	0	0
F_2	1	1	1	1	1
F_3	1	1	1	1	1
F_4	1	1	0.97	0.43	0.08
F_5	0	0	0	0	0
F_6	1	0.98	0.88	0.4	0
F_7	1	0.74	0.65	0.56	0.5
F_8	0.02	0	0	0	0
F_9	0.94	0.3	0.19	0.1	0.04
F_{10}	1	1	1	1	1
F_{11}	1	0.67	0.67	0.67	0.66
F_{12}	0.67	0.59	0.5	0.4	0.33
F_{13}	0.96	0.67	0.67	0.65	0.6
F_{14}	0.81	0.26	0.11	0.04	0.01
F_{15}	0.29	0.07	0.02	0.01	0.01
F_{16}	0	0	0	0	0
F_{17}	0	0	0	0	0
F_{18}	0	0	0	0	0
F_{19}	0	0	0	0	0
F_{20}	0	0	0	0	0