# Auto-Adaptación en Algoritmos Evolutivos Celulares. Un Nuevo Enfoque Algorítmico

Enrique Alba, Bernabé Dorronsoro

*Resumen*— **En este trabajo presentamos una nueva extensión del modelo de búsqueda descentralizado conocido como algoritmo evolutivo celular, donde los individuos residen en una cierta topología e interaccionan únicamente con sus vecinos. Las variaciones en la forma de la topología y la definición de vecindad pueden dar lugar a infinidad de algoritmos distintos, pero es posible un estudio metodológico mediante la definición de un radio para la forma de la población y el vecindario. Como la relación entre ambos define las características de la búsqueda proponemos profundizar en la influencia que dicha ratio tiene sobre el equilibrio exploración/explotación. Para ello extenderemos resultados existentes a nuevos tipos de problemas. Veremos que es difícil decidir cuál es la mejor ratio según la familia de problemas; proponemos un cambio pre-programado en dicha ratio durante la evolución como posible mejora adicional seguido de un posterior refinamiento que dará lugar a la primera solución auto-adaptativa de un modelo celular en la actualidad. Concluiremos que dicho algoritmo auto-adaptativo tiene el mejor comportamiento de entre todos los evaluados, y validaremos nuestros resultados sobre un conjunto de siete tipos de problemas de complejidad considerable que sustentarán nuestras conclusiones.**

*Palabras clave*— **Algoritmo Evolutivo Celular, Algoritmo Evolutivo Descentralizado, Auto-Adaptación, Ratio Vecindario-Población**

## I. Introducción

La aplicación de Algoritmos Evolutivos (EAs) a problemas de optimización ha sido muy importante durante esta última década [1]. Es posible encontrar este tipo de algoritmos en problemas complejos tales como tareas de optimización con restricciones, con ruido, o con elevada epistasis y/o multimodalidad. Normalmente, la mayoría de EAs utilizan una única población (panmícticos), aunque también existe una cierta tradición en el uso de poblaciones estructuradas, especialmente en el caso de implementaciones paralelas.

Entre los EAs estructurados más conocidos figuran los distribuidos y los celulares [2]. En el caso de los distribuidos la población se divide en un conjunto de islas en las que se ejecutan EAs aislados. Entre estas islas se producen intercambios de individuos esporádicamente para introducir diversidad en cada una de ellas y guiar la búsqueda hacia las porciones más prometedoras del espacio.

En el caso de un EA celular (cEA) se introduce el concepto de *vecindario*, de forma que un individuo sólo pueda relacionarse con sus vecinos en el bucle reproductor que aplica los operadores de variación. La topología más popular es una rejilla toroidal en la que residen todos los individuos; además puede demostrarse que usar esta topología no restringe el ámbito de las conclusiones obtenidas [3]. El solapamiento de vecindarios en la topología provoca una suave difusión de las soluciones ya que un mismo individuo pertenece a varios vecindarios al mismo tiempo. En la Fig. 1 podemos ver representados estos tres tipos de EAs. Los EAs celulares y distribuidos representan de forma natural regiones distintas del espacio de búsqueda en su población descentralizada, mejorando en muchos casos el comportamiento numérico y las prestaciones de un algoritmo equivalente en panmixia [4].

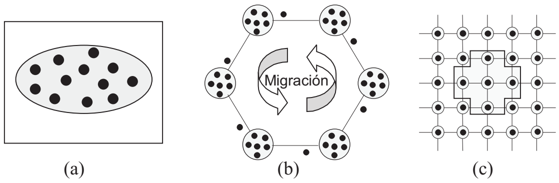


Fig. 1. EA panmíctico con todos sus individuos -puntos negros- en la misma población (a). Estructurando la población podemos distinguir entre EAs distribuidos (b) y EAs celulares (c).

Los cEAs están ganando en importancia por varias razones. Primero, su estructura espacial interna permite conservar la diversidad de fenotipo y genotipo durante más iteraciones que uno tradicional [5]. Además, algunos trabajos establecen las ventajas de utilizar cEAs para resolver complejas tareas de optimización, donde se necesita alta eficacia y reducido número de pasos [6] [7]. Por último, la similitud existente entre cEAs y autómatas celulares formalmente demostrada en [8] [9], sus aplicaciones potenciales, y su disponibilidad para ser implementados en máquinas SIMD les hacen dignos de estudio.

Existen trabajos que sugieren (pero no estudian) que la forma de la rejilla de la población influye en la calidad de la búsqueda. Ejemplos

Departamento de Lenguajes y CC.CC., Universidad de Málaga. E-mails: eat@lcc.uma.es; dorronsoro@uma.es

de esto son [7], que estudia instancias grandes del problema TSP, o [6] y [10] en el caso de optimización de funciones. Ha sido en un trabajo más reciente [11] donde se ha estudiado cuantitativamente el incremento en la eficacia obtenido al utilizar cEAs en los cuales la rejilla no es perfectamente cuadrada. El uso de rejillas estrechas ha provocado algoritmos muy eficientes. En dicho trabajo podemos encontrar un estudio comparativo entre diferentes formas de rejilla para tres problemas.

Nuestro primer objetivo en este artículo es comprobar si se mantienen dichos resultados al ampliar el conjunto de problemas con tareas más complejas. Así mismo, el segundo objetivo será comprobar si en este conjunto extendido de problemas se dan situaciones donde un cambio preprogramado en la forma de la rejilla es ventajoso frente a utilizar una única rejilla estática durante la búsqueda, como ocurre en [12] para el núcleo de tres problemas.

Sin embargo, la contribución más importante de este trabajo consiste en profundizar sobre la importancia de la relación entre la topología y el vecindario (*ratio*). Utilizando el valor cuantitativo de ratio definiremos un nuevo modelo algorítmico auto-adaptativo [13] [14] que tendrá la ventaja de elegir de manera dinámica la ratio más conveniente para el problema en cuestión, de manera que el diseñador no tenga que hacer un estudio de la mejor ratio para su problema concreto.

Este documento se organiza como sigue. En la sección siguiente se expone el modelo canónico de cEA, así como la caracterización del concepto de ratio. La Sección III presenta las distintas variantes de cEA analizadas. En la Sección IV se discute y justifica la elección del conjunto de problemas elegido para realizar las pruebas. Los resultados de estas pruebas se presentan en la Sección V. Por último, en la Sección VI se resumen algunas conclusiones y se sugieren futuras líneas de investigación.

## II. Caracterización de un EA Celular

En esta sección incluimos una descripción en pseudocódigo del modelo canónico de búsqueda en un cEA. El EA que usaremos como caso de estudio es un algoritmo genético celular (cGA) (vea Algoritmo 1). La población está estructurada en una rejilla 2D toroidal y el vecindario definido -línea 5- contiene siempre 5 individuos: el considerado (posición (x,y)) mas los situados al norte, este, oeste y sur del mismo -NEWS-. Se utiliza dentro del vecindario selección proporcional (*Roulette wheel*) según el valor de adecuación de los individuos (`Local_Select`) -líneas 5, 6 y 7-. Usamos un operador de recombinación de dos puntos DPX1 que crea un sólo hijo (aquel con la mayor parte del mejor padre -línea 8-) y un operador de mutación tradicional (*bit-flip*) -línea 9-.

**Algoritmo 1** Pseudo-Código de un cGA Simple

```
1:  proc Steps_Up(cga)
2:  for s ← 1 to MAX_STEPS do
3:     for x ← 1 to WIDTH do
4:        for y ← 1 to HEIGHT do
5:           nList←Compute_Neigh(cga,position(x,y));
6:           parent1←Local_Select(nList);
7:           parent2←Local_Select(nList);
8:           DPX1(cga.Pc,nList[parent1],nList[parent2],
                   auxInd.chrom);
9:           Mutate(cga.Pm,auxInd.chrom);
10:          auxInd.fit←cga.Fit(Decode(auxInd.chrom));
11:          Insert_New_Ind(position(x,y),auxInd,
                   [if_better|always],cga,auxPop);
12:       end for
13:    end for
14:    cga.pop←auxPop;
15:    Update_Statistics(cga);
16: end for
17: end_proc Steps_Up;
```

En nuestra implementación monoprocesador de este cGA las sucesivas poblaciones reemplazan a las anteriores por completo -línea 14-, luego será necesario situar los nuevos individuos generados por selección local, recombinación y mutación en una población temporal -línea 11-.

El paso de reemplazo -línea 11- puede ser implementado utilizando la antigua población (por ejemplo, reemplazando sólo si la cadena nueva es mejor que la antigua) o no (añadiendo siempre la nueva cadena a la siguiente población). El primer caso (reemplazo por torneo binario) es el que usaremos [4]. El cálculo de las estadísticas básicas -línea 15- es difícil de encontrar en el pseudo-código de otros autores, aunque es necesario para el funcionamiento y monitorización del algoritmo. Como veremos, es posible utilizar algún tipo de valor estadístico para la auto-adaptación de la búsqueda, de ahí la importancia de este cómputo. Algo similar ocurre en el paso que calcula la lista de vecinos de un individuo situado en las coordenadas (x,y) -lineal 5-. Esta operación es necesaria incluirla en el diseño del algoritmo si se ha planificado un cambio en el tipo de vecindario (aunque nosotros no utilicemos dicha capacidad en este trabajo).

Tras presentar el algoritmo, pasaremos a caracterizar la rejilla de nuestro cGA. Usamos una idea derivada de [3] para definir un "radio" [11]. La diferencia básica es que aquí no es necesario que la rejilla sea cuadrada. Consideramos que la rejilla tiene un radio igual a la dispersión de $n^*$ puntos en el círculo centrado en $(\overline{x}, \overline{y})$ (Ecuación 1). Empleando esta definición siempre asignaremos diferentes valores numéricos a rejillas distintas.

$$rad = \sqrt{\frac{\sum (x_i - \overline{x})^2 + \sum (y_i - \overline{y})^2}{n^*}}, \quad (1)$$

$$\overline{x} = \frac{\sum_{i=1}^{n^*} x_i}{n^*}, \; \overline{y} = \frac{\sum_{i=1}^{n^*} y_i}{n^*}$$

Esta definición también proporciona un valor de radio para el vecindario. Como se puede ver en [3] es posible cuantificar la relación entre el vecindario y la población mediante la ratio relativa entre sus radios (Ecuación 2). Los algoritmos con una ratio similar muestran una presión de selección semejante [15].

$$Ratio = \frac{Rad_{Vecindario}}{Rad_{Topología}} \quad (2)$$

Cuando se resuelve un problema dado con un número constante de individuos ($n = n^*$, para realizar comparaciones justas) el radio de la rejilla se incrementará cuanto más estrecha sea ésta (Fig. 2b). Como el vecindario es constante en tamaño y forma en todo nuestro estudio (siempre usamos NEWS, Fig. 2a), entonces cuanto más estrecha sea la rejilla menor será la ratio resultante.

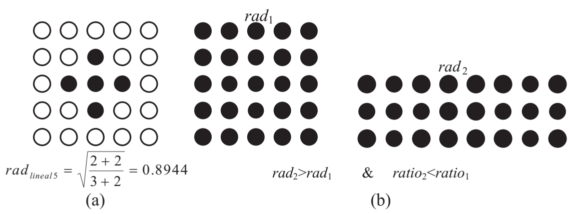


Fig. 2. Radio del vecindario NEWS (a); rejillas de (5×5 = 25 y 3×8≈25) igual número de individuos con radios diferentes (b).

Reducir la ratio es equivalente a reducir la intensidad de selección global, promoviendo la exploración. Esto nos permitirá mantener una diversidad más elevada en el algoritmo, lo que mejora los resultados en problemas difíciles (por ejemplo problemas multimodales y epistáticos). Por otro lado, la búsqueda que se realiza dentro de cada vecindario encamina al algoritmo hacia la explotación de la población. En este documento exploramos cómo influye esta ratio en la eficiencia de la búsqueda sobre una variedad de dominios. Cambiar la ratio durante la búsqueda es una característica única de los cEAs que puede ser utilizada ventajosamente para cambiar de exploración a explotación con una complejidad mínima y sin introducir un nuevo tipo de algoritmo ni hibridaciones en el funcionamiento canónico.

La forma en la que hacemos el cambio de la rejilla de la población durante la ejecución influirá significativamente en el comportamiento del algoritmo. En este estudio hemos utilizado un primer acercamiento que almacena la población de individuos en una lista de longitud $m \cdot n$, de manera que la primera línea de una rejilla de $m \times n$ está compuesta por los $m$ primeros individuos de la lista, la segunda fila por los siguientes $m$ individuos, y así hasta los últimos $m$ individuos de la lista, que formarán la última fila de la rejilla. A este método de redistribución lo llamaremos *contiguo* porque la nueva rejilla se rellena siguiendo el orden en que aparecen los individuos en la lista. Por tanto, en el cambio de una rejilla de dimensiones $m \times n$ a otra $m' \times n'$ (siendo $m \cdot n = m' \cdot n'$), el individuo situado en la posición $(i, j)$ será realojado como sigue:

$$(i, j) \rightarrow ([i * n + j] \; div \; n', \; [i * n + j] \; mod \; n') \quad (3)$$

En la Fig. 3 se puede ver cómo se re-alojan un individuo y sus vecinos cuando cambiamos de una población de 4 × 4 individuos a otra de 2× 8 según la Ecuación 3. Puede observarse cómo en general una parte del vecindario se mantiene mientras otra puede cambiar.

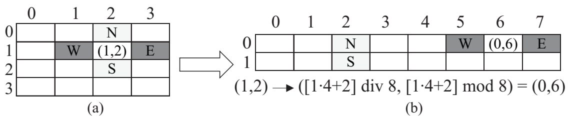


Fig. 3. Re-alojamiento de un individuo y sus vecinos cuando cambia la rejilla de (a) 4 × 4 a otra (b) de forma 2 × 8.

## III. Descripción de los Algoritmos Utilizados

En este trabajo hemos aplicado siempre el mismo cGA canónico cambiando tan sólo la ratio entre el vecindario y la topología de la población. Puesto que puede establecerse la relación entre baja/alta ratio y exploración/explotación del algoritmo [11] nuestro objetivo es primero comprobar la influencia de este parámetro sobre una familia de problemas representativa, extendida de los resultados iniciales incluidos en [11]. En segundo lugar, realizaremos una propuesta de auto-adaptación de dicha ratio para evitar al investigador la definición *ad hoc* de este parámetro, y con la esperanza de obtener aún mejores resultados que los existentes.

En primer lugar abordamos un cGA donde se han utilizado tres ratios estáticas claramente distintas, que son las correspondientes a las rejillas de formas 20 × 20 (*square*), 10 × 40 (*rectangular*), y 4 × 100 (*narrow*). Los valores de la ratio para cada uno de estos casos utilizando vecindario NEWS son 0.110, 0.075, y 0.031.

Además, también hemos utilizado dos criterios con cambio de ratio pre-programado. Estos criterios cambian la ratio en un punto fijo de la ejecución. En nuestro caso particular, el cambio se produce cuando el algoritmo alcanza la mitad de las evaluaciones que se necesitan para resolver el problema utilizando la rejilla *square*, tal como ocurre en [11]. Los dos criterios pre-programados que hemos utilizado son aquellos que cambian de la ratio 0.110 a 0.031 (llamado *SqNar* -*square* a *narrow*-) y de 0.031 a 0.110 (llamado *NarSq*).

Por último, completamos nuestro estudio haciendo una aportación al campo de la auto-adaptación. El mecanismo de auto-guía que proponemos varía la topología de la población durante la búsqueda según la velocidad de convergencia (*AvgFit*). Una ventaja valiosa de este criterio auto-adaptativo consiste en que no es necesario precisar el parámetro que define la forma de la población. En el Algoritmo 2 se especifica cómo funciona este criterio, que está basado en el valor medio de adecuación de los individuos de la población en cada generación ($\overline{f}_t$).

Para controlar la forma de la rejilla intentamos incluir un sesgo hacia rejillas cuadradas (mayor explotación) cuando el algoritmo evoluciona lentamente (i.e. la media de adecuación entre poblaciones consecutivas cambia menos que un cierto valor $\varepsilon$). Esto viene representado por la condición de la línea 1 en el Algoritmo 2. Si por el contrario la búsqueda se considera demasiado rápida entonces la forma de la población se hace más rectangular (estrecha). La condición para detectar esta situación viene expresada por la línea 3 del Algoritmo 2. Puede observarse que esto ocurre cuando se acelera más de $(1-\varepsilon)$ veces la última aceleración conocida $(\overline{f}_t - \overline{f}_{t-1})$.

La función **ChangeTo**(square/rectangular) del Algoritmo 2 cambia la forma de la población a la rejilla inmediatamente más cuadrada/rectangular permitida (los límites son la rejilla completamente cuadrada y la rejilla completamente lineal). En los casos en los que el cambio no sea posible porque el algoritmo está usando una rejilla con forma extrema no se realiza ningún cambio en la forma de la población. Para completar la definición de la auto-adaptación sólo resta especificar cómo se lleva a cabo el cambio: la posición $(i, j)$ de un individuo en la nueva rejilla se calcula como indica la Ecuación 3.

## IV. Problemas Estudiados

Con el fin de analizar el comportamiento de la búsqueda del cGA con diferentes ratios se ha elegido un conjunto de problemas representativos por contener diversas características de interés en optimización, tales como epistasis, multimodalidad, engaño, uso de restricciones, o identificación paramétrica. Inicialmente se han utilizado los problemas abordados en [11], que incluyen el problema engañoso masivamente multimodal (MMDP), el problema de la modulación en frecuencia de sonidos (FMS) y el generador multimodal P-PEAKS; luego hemos extendido esta base de tres problemas con los problemas COUNTSAT (una instancia de MAXSAT), diseño de códigos correctores de errores (ECC), máximo corte de un grafo (MAXCUT), y planificación de tareas con espera mínima (MTTP). La gran representatividad de este conjunto de problemas tanto en tipos de dificultad como en dominios de aplicación (optimización combinatoria, optimización continua, telecomunicaciones, etc.) justifican claramente su elección y permiten asegurar un alto nivel de confianza en los resultados que se obtengan.

**Algoritmo 2** Auto-adaptación según *AvgFit*

1: **if** $\left((\overline{f}_{t+1} - \overline{f}_t) < (1+\varepsilon)(\overline{f}_t - \overline{f}_{t-1})\right)$ **then**
2:     **ChangeTo**(square) //*exploit*
3: **else if** $\left((\overline{f}_{t+1} - \overline{f}_t) > (2-\varepsilon)(\overline{f}_t - \overline{f}_{t-1})\right)$ **then**
4:     **ChangeTo**(rectangular) //*explore*
5: **end if**

A continuación presentamos la descripción de los siete problemas abordados en este trabajo.

El problema MMDP fue específicamente diseñado para ser difícil de resolver para un EA [16]. Está compuesto por $k$ subproblemas engañosos ($s$), formado cada uno de ellos por una cadena binaria de longitud seis, y su valor depende del número de unos que contenga dicha cadena -$fitness_{s_i}$- (vea Fig. 4). Es fácil ver (gráfica de la Fig. 4) que estas funciones poseen dos máximos globales y un atractor engañoso en el punto intermedio .

| Nº de Unos | Valor de la Subfunción |
|---|---|
| 0 | 1.000000 |
| 1 | 0.000000 |
| 2 | 0.360384 |
| 3 | 0.640576 |
| 4 | 0.360384 |
| 5 | 0.000000 |
| 6 | 1.000000 |

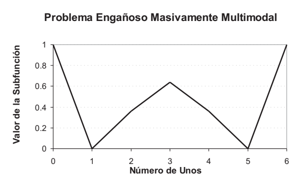


Fig. 4. Función bipolar engañosa básica ($s_i$) para MMDP

En MMDP, al contribuir cada subproblema $s_i$ al valor de adecuación total, el número de óptimos locales es muy grande ($22^k$), mientras que sólo hay $2^k$ soluciones globales. Por tanto, el grado de multimodalidad viene dado por el parámetro $k$. Aquí utilizaremos una instancia considerablemente grande de $k = 40$ subproblemas. La función que trataremos de maximizar para resolver el problema se muestra en la Ecuación 4, y su valor máximo es 40.

$$f_{MMDP}(\vec{s}) = \sum_{i=1}^{k} fitness_{s_i} \quad (4)$$

El problema FMS (*Frequency Modulation Sounds*) [17] consiste en determinar los 6 parámetros reales $\vec{x} = (a_1, w_1, a_2, w_2, a_3, w_3)$ (codificados con 32 bits en el rango $[-6.4, +6.35]$) del modelo de sonido en frecuencia modulada dado en la Ecuación 5 para aproximarla a la onda de sonido dada en la Ecuación 6, donde $\theta = 2\pi/100$.

$$y(t) = a_1 \sin\Big(w_1 t\theta + a_2 \sin\big(w_2 t\theta + a_3 \sin(w_3 t\theta)\big)\Big) \quad (5)$$

$$y_0(t) = 1.0 \sin\Big(5.0 t\theta - 1.5 \sin\big(4.8 t\theta + 2.0 \sin(4.9 t\theta)\big)\Big) \quad (6)$$

El objetivo de este problema consiste en minimizar la suma del error cuadrático medio entre los datos de muestreo (Ecuación 5) y los reales (Ecuación 6), como se detalla en la Ecuación 7. Debido a la dificultad para resolver este problema sin operadores específicos, el algoritmo se detiene cuando el error cae por debajo de $10^{-2}$. La función de adecuación que tendremos que maximizar se corresponde con la inversa de la Ecuación 7, y obtenemos su máximo valor cuando $E_{FMS} = 0.0$.

$$E_{FMS}(\vec{x}) = \sum_{t=0}^{100} \Big(y(t) - y_0(t)\Big)^2 \quad (7)$$

El problema P-PEAKS [18] es un generador de problemas multimodales. Un generador de problemas es una tarea fácilmente parametrizable con un grado de dificultad afinable, de manera que nos permita generar instancias con una complejidad tan grande como queramos. Cada vez que se ejecuta el algoritmo se resuelve una instancia distinta, lo que nos permite obtener conclusiones sobre la clase del problema como tal y no sobre una instancia concreta, por muy difícil que ésta pudiera resultar.

La idea de P-PEAKS consiste en generar $P$ cadenas aleatorias de $N$ bits que representan la localización de $P$ cimas en el espacio de búsqueda. El valor de adecuación de una cadena se corresponde con la distancia de Hamming de esa cadena con respecto a la cima más próxima (vea Ecuación 8). Usando un mayor (o menor) número de cimas obtenemos problemas más (o menos) epistáticos. En nuestro caso hemos utilizado una instancia de $P = 100$ cimas de longitud $N = 100$ bits cada una, representando un nivel de epistasis medio/alto [11]. El máximo valor de fitness que podremos conseguir es 1.0.

$$f_{P-PEAKS}(\vec{x}) = \frac{1}{N} \max_{i=1}^{P} \{N - HammingD(\vec{x}, Peak_i)\} \quad (8)$$

El problema COUNTSAT [19] es una instancia de MAXSAT. En COUNTSAT el valor de la solución se corresponde con el número de cláusulas (de entre todas las posibles cláusulas de Horn de tres variables) satisfechas por una entrada formada por $n$ variables booleanas. Es fácil comprobar que el óptimo se consigue cuando todas las variables tienen el valor 1. En este estudio consideramos una instancia de $n = 20$ variables, y el valor de la solución óptima del problema es 6860 (Ecuación 9).

$$f_{CNTSAT}(s) = s + n(n-1)(n-2) - 2(n-2)\binom{s}{2} + 6\binom{s}{3}$$
$$= s + 6840 - 18s(s-1) + s(s-1)(s-2) \quad (9)$$

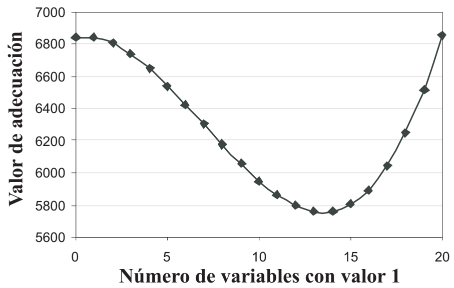


Fig. 5. Función *COUNTSAT* con 20 variables.

La función COUNTSAT está extraída de MAXSAT con la intención de ser muy difícil de resolver con GAs [19]. Las entradas aleatorias generadas uniformemente tendrán $n/2$ unos en término medio. Por tanto, los cambios locales decrementando el número de unos nos conducirán hacia entradas mejores, mientras que si se incrementa el número de unos se decrementará el valor de adecuación. En consecuencia, cabe esperar que un GA encuentre rápidamente la cadena formada por todos sus componentes a cero y tenga dificultades para encontrar la cadena compuesta por unos en todas sus posiciones.

El problema ECC se presentó en [20]. Consideraremos una tupla $(n, M, d)$ donde $n$ es la longitud (número de bits) de cada palabra de código, $M$ es el número de palabras de código, y $d$ es la mínima distancia de Hamming entre dos palabras de código cualesquiera. El objetivo será obtener un código con el mayor valor posible de $d$ (reflejando una mayor tolerancia al ruido y a los errores), y con valores de $n$ y $M$ fijados previamente. La función de adecuación que maximizaremos se corresponde con:

$$f_{ECC}(C) = \frac{1}{\sum_{i=1}^{M} \sum_{\substack{j=1 \\ i \neq j}}^{M} \frac{1}{d_{ij}^2}}, \quad (10)$$

donde $d_{ij}$ representa la distancia de Hamming entre las palabras $i$ y $j$ del código. En este estudio consideramos una instancia donde $C$ está formado por $M=24$ palabras de longitud $n=12$ bits. Lo que forma un espacio de búsqueda de tamaño $10^{87}$ aproximadamente. La mejor solución tendrá el valor de adecuación 0.0674 [21].

El problema MAXCUT consiste en dividir un grafo ponderado $G=(V,E)$ en dos subgrafos disjuntos $G_0=(V_0,E_0)$ y $G_1=(V_1,E_1)$ de manera que la suma de los pesos de los ejes que posean un extremo en $V_0$ y el otro en $V_1$ sea máxima. Para codificar una partición de los vértices utilizaremos una cadena binaria $(x_1,x_2,\ldots,x_n)$ donde cada dígito se corresponde con un vértice. De manera que si un dígito tiene valor 1, su vértice correspondiente estará en el conjunto $V_1$, y si tuviera valor 0 el vértice respectivo pertenecería a $V_0$. La función que maximizaremos es:

$$f_{MAXCUT}(\vec{x})=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{ij}\left[x_i(1-x_j)+x_j(1-x_i)\right] \quad (11)$$

Téngase en cuenta que $w_{ij}$ contribuye a la suma sólo en el caso de que los nodos $i$ y $j$ se encuentren en distintas particiones. Hemos estudiado el problema con tres instancias distintas tomadas de [22]. Dos de ellas se corresponden con grafos generados aleatoriamente: uno disperso "cut20.01" y otro denso "cut20.09"; ambos formados por 20 vértices. La otra instancia es un grafo escalable de 100 vértices. Las soluciones óptimas para estas instancias son 10.119812 para el caso de "cut20.01", 56.740064 para "cut20.09" y 1077 para "cut100".

El problema MTTP [23] es un problema de planificación de tareas en el que cada tarea $i$ del conjunto de tareas $T=\{1,2,\ldots,n\}$ tiene una longitud $l_i$ -el tiempo que tarda en ejecutarse- un límite de tiempo (o *deadline*) $d_i$ -antes del cual se debe ejecutar- y un peso $w_i$.

Definimos una función de planificación $g$ sobre $S\subseteq T:\ S\mapsto \mathbb{Z}^{+}\bigcup\{0\}\ /\ \forall\ i,\ j\in S$ con estas dos propiedades:

1. Una tarea no puede ser planificada antes de que haya terminado alguna otra ejecutada previamente: $g(i)<g(j)\Rightarrow g(i)+\ l_i\leq g(j)$.
2. Toda tarea tiene que finalizar antes que su *deadline*: $g(i)+l_i\leq d_i$.

El objetivo del problema es minimizar la suma de los pesos de las tareas no planificadas (vea Ecuación 12). Por tanto, la planificación óptima minimiza el valor $W$:

$$W_{MTTP}=\sum_{i\in T-S} w_i \quad (12)$$

La planificación $S$ está representada mediante un vector $\vec{x}=(x_1,x_2,\ldots,x_n)$ que contiene todas las tareas ordenadas según su *deadline*. Cada $x_i\in\{0,1\}$, donde si $x_i=1$ entonces la tarea $i$ está contenida en la planificación $S$, mientras que si $x_i=0$ la tarea $i$ no está incluida en $S$. La función que optimizaremos, descrita en [22], es la inversa de Ecuación 12. Hemos utilizado para el estudio del comportamiento del cGA con esta función tres diferentes instancias [22]: "mttp20", "mttp100" y "mttp200", de tamaños 20, 100 y 200, y máximo valor de adecuación de 0.02439, 0.005 y 0.0025, respectivamente.

## V. Experimentación

En esta sección presentamos y analizamos los resultados obtenidos por las distintas variantes de cGA al resolver los siete problemas propuestos. Para ello procederemos en tres etapas, abordando el estudio de una ratio estática, seguido de un análisis del comportamiento de una ratio dinámica pre-programada, para finalizar con la discusión de los resultados de aplicar auto-adaptación a la forma de la rejilla de un algoritmo celular. En las dos primeras etapas utilizamos rejillas cuadradas y rectangulares en momentos de la evolución determinados a priori, i.e., siempre la misma forma de rejilla (estático) o bien un cambio de tipo de rejilla establecido *off-line* antes de optimizar (pre-programado). En la última y tercera etapa aplicamos un mecanismo basado en la aceleración de la convergencia medido a través de la media de adecuación de la población como motor para la auto-adaptación dinámica de la forma de rejilla más apropiada para la búsqueda en cada instante. La rejilla de inicio será la que tenga un mayor compromiso entre la exploración y explotación: la de valor de ratio más intermedio.

Hemos organizado esta sección en dos subsecciones. En la primera expondremos los parámetros utilizados para la ejecución de los algoritmos, mientras que en la segunda estudiaremos los resultados obtenidos por los diferentes algoritmos sobre todos los problemas.

### A. Parametrización

Con el fin de hacer una comparación justa entre todos los algoritmos hemos utilizado la misma parametrización con todos los problemas; los detalles se describen en la Tabla I, donde $L$ es la longitud del cromosoma del individuo. Para obtener resultados estadísticamente significativos, realizamos $t$-tests sobre los valores de todas las tablas. Consideraremos el valor 0.05 como nivel significativo, y valores ligeramente superiores (hasta 0.08) se considerarán como tendencias en los datos.

La probabilidad de mutación de un bit en el caso de FMS es, a diferencia de los demás proble-

| | |
|---|---|
| *Tamaño Población* | 400 individuos |
| *Selección de Padres* | Ruleta+Ruleta |
| *Cruce* | DPX1, $p_c = 1.0$ |
| *Mutación* | $p_m = 1/L$ ($p_m = 10/L$ en FMS) |
| *Reemplazo* | Reemp_si_Mejor |

TABLA I

Parametrización utilizada en los algoritmos.

mas, $p_m$=$10/L$. Hemos incrementado este valor para FMS porque es muy difícil en este problema obtener la solución con una probabilidad de mutación de $1/L$ sin el uso de búsqueda local u operadores especializados en optimización continua.

Reemplazaremos al individuo considerado en cada instante sólo si su descendiente tiene mejor valor de adecuación (Reemp_si_Mejor). Analizamos el coste de resolver un problema midiendo el número de evaluaciones de la función objetivo. La condición de parada para todos los algoritmos consiste en encontrar una solución o, en su defecto, llegar a realizar un millón de evaluaciones.

En resumen, proponemos seis algoritmos diferentes para resolver cada problema, todos ellos implementando el mismo cEA canónico. Tres utilizan ratios estáticas, dos más emplean ratios dinámicas *pre-programadas* y el último usa un nuevo criterio dinámico *auto-adaptativo*.

### *B. Resultados Experimentales*

A continuación, abordaremos la optimización de todos los problemas usando los seis algoritmos propuestos en la Sección II. Se han realizado para cada algoritmo y problema 30 ejecuciones independientes. Los resultados se resumen en las tablas II a IV, donde los valores que aparecen se corresponden con la media del número de evaluaciones realizado sólo en las ejecuciones en las que se encontró la solución. Con letra **negrita** aparecen los mejores valores globales obtenidos para cada problema.

En las Tablas II y III, se presenta el número medio de evaluaciones necesarios por cada algoritmo estático y pre-programado para resolver cada uno de los problemas.

Nuestro primer resultado confirma los valores arrojados por el estudio de Alba y Troya [11], que concluye tras analizar FMS, MMDP y P-PEAKS, que una ratio estrecha (*narrow*) es más eficiente en el caso de problemas irregulares (FMS) y, especialmente, problemas altamente epistáticos (P-PEAKS), donde la reducida presión selectiva favorece la exploración. Por otro lado, la elevada presión de la ratio *square* es más eficiente en el caso de un problema multimodal engañoso como MMDP. Además, los resultados del problema epistático COUNTSAT también confirman las conclusiones anteriores en relación a la mayor eficiencia de la ratio *narrow*.

| *Probl.\Ratio* | *square* | *rectangular* | *narrow* |
|---|---|---|---|
| MMDP | 170637.90 | 237003.40 | 331719.60 |
| FMS | 609949.70 | 574145.10 | **540038.00** |
| P-PEAKS | 50458.10 | 50364.60 | 48653.60 |
| COUNTSAT | 24326.33 | 25214.82 | 21736.54 |
| ECC | 159102.40 | **144332.30** | 151724.00 |
| “cut20.01” | 18364.80 | 18899.47 | 17375.67 |
| “cut20.09” | 41703.00 | 36610.30 | 37679.63 |
| “cut100” | 256826.10 | 232141.50 | 357101.30 |
| “mttp20” | 18017.27 | 17923.70 | 19514.33 |
| “mttp100” | 294613.70 | 274162.70 | 303636.20 |
| “mttp200” | 673598.80 | 688850.20 | 566999.60 |

TABLA II

Resultados con ratios estáticas

Un segundo resultado interesante que también puede observarse en la Tabla II sugiere que la ratio *rectangular*, que no había arrojado resultados competitivos sobre MMDP, FMS ni P-PEAKS, es la más apropiada para resolver los problemas de optimización combinatoria con que hemos extendido la base de pruebas.

| *Problema\Ratio* | *SqNar* | *NarSq* |
|---|---|---|
| MMDP | 1002900.00 | 1002900.00 |
| FMS | 615634.30 | 619971.70 |
| P-PEAKS | 57769.70 | **48012.10** |
| COUNTSAT | 22637.27 | 19720.91 |
| ECC | 163018.90 | 175891.00 |
| “cut20.01” | 15851.87 | 24206.03 |
| “cut20.09” | 42825.80 | 40045.53 |
| “cut100” | 415788.80 | 284057.40 |
| “mttp20” | 15838.50 | 19420.77 |
| “mttp100” | 284829.30 | 303061.40 |
| “mttp200” | 521419.30 | 572132.40 |

TABLA III

Resultados con ratios pre-programadas

Si consulta la Tabla III puede comprobar que la ratio *NarSq* mejora los resultados obtenidos respecto al uso de las rejillas estáticas para el problema COUNTSAT (confirmando también así los resultados obtenidos en [11]).

La Tabla III demuestra también que la ratio *SqNar* (que representa el cambio de explotación a exploración) proporciona mayor eficiencia en los problemas de optimización combinatoria, superando a la estrategia *NarSq*.

Finalmente, en la Tabla IV mostramos los resultados obtenidos con el criterio auto-adaptativo *AvgFit*. Para profundizar un poco más en nuestro estudio hemos realizado pruebas para este criterio con distintos valores de $\varepsilon$. Observando dicha tabla vemos que, salvo en los casos de ECC, FMS y P-PEAKS, al usar *AvgFit* mejoramos los resultados de ratios estáticas y pre-programadas.

En la Fig. 6 trazamos los cambios en la forma de la rejilla durante la resolución de los proble-

| $Problema \backslash \varepsilon$ | 0.05 | 0.15 | 0.25 | 0.3 |
|---|---|---|---|---|
| MMDP | 185261.0 | 182771.5 | **167334.8** | 190822.7 |
| FMS | 676631.8 | 689146.1 | 687530.2 | 716132.7 |
| P-PEAKS | 55804.8 | 54254.3 | 54535.0 | 55283.5 |
| COUNTSAT | 16124.9 | 16894.5 | 19859.1 | **16039.0** |
| ECC | 160786.6 | 153983.0 | 169822.5 | 163099.1 |
| “cut20.01” | 15624.6 | **15303.8** | 17161.8 | 20369.8 |
| “cut20.09” | 30849.3 | 36490.0 | **26251.1** | 32052.3 |
| “cut100” | **226263.3** | 346555.5 | 320548.4 | 304037.2 |
| “mttp20” | 8807.6 | **7577.9** | 10291.3 | 7952.2 |
| “mttp100” | **176011.3** | 245985.8 | 247469.5 | 244141.2 |
| “mttp200” | **492186.4** | 508146.2 | 506020.9 | 493977.5 |

TABLA IV

Resultados con *AvgFit* (auto-adaptativo)

mas propuestos con *AvgFit*. En ella se aprecia una clara tendencia hacia la explotación de la población (ratio mayor que 0.1). Esta evolución hacia la rejilla cuadrada se produce de la misma manera en todos los problemas, excepto en los casos más complejos: FMS y MTTP. En estos dos casos hay breves y periódicas visitas a ratios menores (por debajo de 0.08) para mejorar la exploración del algoritmo. Esta alternancia en problemas complejos es una característica notable del mecanismo auto-guiado que proponemos, por cuanto él decide por sí mismo cuándo explotar o cuándo explorar.

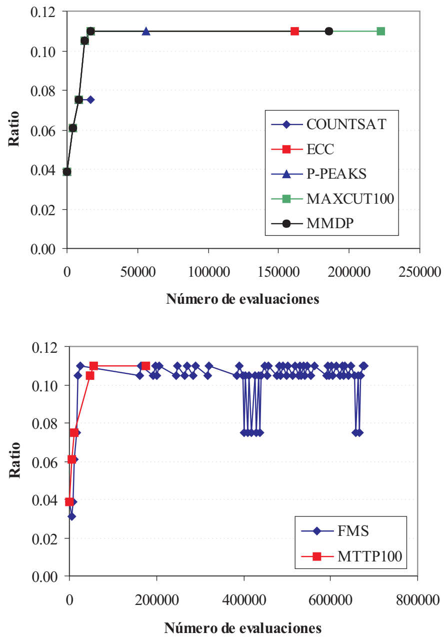


Fig. 6. Cambios en la rejilla producidos al usar AvgFit.

Ahora estudiaremos cada problema por separado. En el caso de COUNTSAT la diferencia en el rendimiento entre algoritmos estáticos y pre-programados no es estadísticamente significativa, aunque existe una cierta tendencia hacia *NarSq* como mejor ratio. Parece claro que *NarSq* funcionará mejor en este problema que los demás algoritmos, ya que utilizar la rejilla estrecha al principio permite al algoritmo escapar del óptimo local. Con *AvgFit* obtenemos los mejores resultados globales.

En un problema de optimización combinatoria como ECC hemos visto que las ratios más rápidas en encontrar solución parecen ser las estáticas. Además, *rectangular* es significativamente más eficiente que cualquier otro criterio (seguido de cerca por *AvgFit* con $\varepsilon = 0.15$).

Para FMS comprobamos que, en concordancia estadística con [11], la aplicación de ratios más pequeñas (rejillas estrechas) parece funcionar mejor que el caso de otras mayores, aunque no hay significancia estadística, es sólo una tendencia. En este problema los criterios dinámicos no consiguen superar al uso de una ratio estática.

En el caso de MAXCUT, otro problema de optimización combinatoria, se observa que el criterio auto-adaptativo suele aportar los mejores resultados, y muchas veces con valores estadísticamente significativos. Con los criterios estáticos y pre-programados, hay una cierta tendencia en las tres instancias hacia una mayor eficiencia al utilizar ratios estáticas, en particular con *rectangular*.

En un problema multimodal y engañoso como es el caso de MMDP, es claro que el uso de una ratio que ejerza una elevada presión sobre la población mejora el rendimiento sobre el resto de las ratios. La estrategia *Narrow* y pre-programada son, significativamente, criterios muy ineficientes para este problema. Atrae nuestra atención de manera especial el hecho de que las ratios pre-programadas no sean capaces de encontrar ninguna solución antes de alcanzar el millón de evaluaciones realizadas. Esto podría hacernos pensar que cambiar la ratio durante la ejecución podría no ser una buena idea en el caso de este problema. Sin embargo, nuestro criterio auto-adaptativo consigue los mejores resultados globalmente para este problema utilizando $\varepsilon = 0.25$, además de resultados muy competitivos para el resto de valores de $\varepsilon$ en relación a la rejilla cuadrada, que era la mejor hasta el momento (estudie las filas MMDP comparativamente en las tres tablas de resultados).

El problema de optimización combinatoria MTTP permite concluir claramente la superioridad del criterio auto-adaptativo *AvgFit*. Este criterio es, en cualquiera de las tres instancias estudiadas, significativamente mejor (con todos los $\varepsilon$ propuestos) que cualquiera de los criterios estáticos y pre-programados. También puede observarse una cierta tendencia de este problema a trabajar mejor con ratios pre-programadas con respecto a las estáticas.

Por último, estudiaremos el problema multimodal P-PEAKS, que además es altamente epistático. Los resultados confirman la tendencia detectada para FMS (el otro problema epistático estudiado), según la cual las ratios pequeñas proporcionan mejores resultados. La excepción se encuentra en el caso de *SqNar* que es significativamente peor que cualquier otra ratio estática o pre-programada, a diferencia del caso de FMS en el que las dos ratios preprogramadas ofrecían resultados similares. En el caso de P-PEAKS *NarSq* es significativamente mejor que cualquier otro criterio, mientras que *AvgFit* es significativamente peor que los demás criterios estudiados.

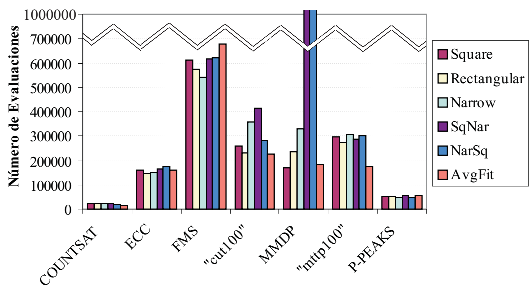


Fig. 7. Resultados obtenidos con los distintos criterios.

En la Fig. 7 se han resumido los resultados obtenidos por los distintos problemas propuestos con todos los criterios estáticos y preprogramados, además del que se ha considerado mejor de entre los auto-adaptativos estudiados: el correspondiente al caso en que $\varepsilon = 0.05$. En los casos de MAXCUT y MTTP sólo se han incluido las instancias de longitud 100. Puede observarse incluso una relación de escala de dificultad comparativa para cada problema resuelto con nuestro cEA, aunque el objetivo de la figura es comparar el coste de cada criterio para un problema dado (las barras entre sí para cada problema).

## VI. Conclusiones y Trabajo Futuro

En este documento hemos estudiado el comportamiento de varias extensiones de un EA celular sobre siete problemas. La ratio, definida como la relación entre el radio del vecindario y el de la población, representa un único parámetro rico en posibilidades para aumentar la eficiencia en un cEA. Puesto que la población está distribuida sobre una rejilla 2D de forma toroidal, el modelo estudiado está incluido en muchos otros algoritmos similares, así que esperamos que los actuales resultados sean realmente de utilidad a otros investigadores.

Nuestra motivación para este trabajo ha sido desarrollar un estudio más profundo al realizado en [11] sobre *cómo* influye la ratio sobre la búsqueda en poblaciones estructuradas. Hemos extendido dicho trabajo añadiendo al análisis más problemas de distintas características, como COUNTSAT, ECC, MAXCUT y MTTP, y resolviéndolos con los mismos algoritmos. Una vez hecho esto, hemos introducido una nueva característica que, a nuestro entender, se utiliza en este documento por primera vez. Se trata de hacer auto-adaptativo el cambio de la ratio en un cEA, de manera que el algoritmo sea capaz de cambiar por él mismo la forma de la población durante la búsqueda. Esto permite favorecer la exploración o la explotación de la población según sea necesario, con la ventaja de manejar un único parámetro.

Nuestra primera aportación ha sido confirmar los resultados de [11], según los cuales una rejilla estrecha (ratio pequeña) está más indicada para ser utilizada con problemas multimodales y/o epistáticos mientras que resulta peor que las rejillas cuadrada y rectangular (ratios elevadas) en el caso de problemas simples y no epistáticos. Nuestra segunda aportación en este artículo ha sido demostrar la utilidad de una rejilla intermedia *rectangular* para problemas de optimización combinatoria.

Finalmente, destacaremos que utilizando el criterio auto-adaptativo *AvgFit* obtenemos resultados competitivos e incluso mejores que los mejores obtenidos con ratios estáticas y pre-programadas. Además, en algunos problemas de optimización combinatoria el buen comportamiento de este criterio auto-adaptativo es importante.

Como trabajo futuro proponemos la inclusión de nuevos criterios más sofisticados para la auto-adaptación. Ejemplos de este tipo de criterio pueden ser el uso de la entropía de la población como medida de la diversidad, otras técnicas basadas en el valor de adecuación, o incluso técnicas basadas en más de un criterio, como por ejemplo la entropía y el valor medio de adecuación de forma combinada. En todo caso, es deseable que se utilicen criterios simples de medir, para no convertir la ventaja de auto-adaptación en una desventaja por la sobrecarga introducida en el tiempo real de ejecución.

Como extensiones estamos trabajando en la importancia del parámetro $\varepsilon$, la comprobación de las conclusiones delineadas sobre otros problemas (incluyendo problemas del mundo real) y las diferencias entre el comportamiento de cEAs y algoritmos con técnicas de restricción de reproducción y de selección.

## Reconocimientos

Este trabajo ha sido realizado con el apoyo del Ministerio de Ciencia y Tecnología (MCYT) y el Fondo Europeo de Desarrollo Regional (FEDER) mediante el contrato TIC2002-04498-C05-02.

## Referencias

[1] T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*, Oxford University Press, 1997.

[2] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 443–462, October 2002.

[3] J. Sarma and K. A. De Jong, "An analysis of the effect of the neighborhood size and shape on local selection algorithms," in *Parallel Problem Solving from Nature (PPSN IV)*, H. M. Voigt, W. Ebeling, I. Rechenberg, and H. P. Schwefel, Eds. 1996, vol. 1141 of *Lecture Notes in Computer Science*, pp. 236–244, Springer-Verlag, Heidelberg.

[4] E. Alba and J. M. Troya, "Improving flexibility and efficiency by adding parallelism to genetic algorithms," *Statistics and Computing*, vol. 12, no. 2, pp. 91–114, 2002.

[5] P. Spiessens and B. Manderick, "A massively parallel genetic algorithm," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, L. B. Booker R. K. Belew, Ed. 1991, pp. 279–286, Morgan Kaufmann.

[6] S. Baluja, "Structure and performance of fine-grain parallelism in genetic search," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, S. Forrest, Ed. 1993, pp. 155–162, Morgan Kaufmann.

[7] H. Mühlenbein, M. Schomish, and J. Born, "The parallel genetic algorithm as a function optimizer," *Parallel Computing*, vol. 17, pp. 619–632, 1991.

[8] M. Tomassini, "The parallel genetic cellular automata: Application to global function optimization," in *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, R. F. Albrecht, C. R. Reeves, and N. C. Steele, Eds. 1993, pp. 385–391, Springer-Verlag.

[9] D. Whitley, "Cellular genetic algorithms," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, S. Forrest, Ed. 1993, p. 658, Morgan Kaufmann, CA.

[10] V. S. Gordon and D. Whitley, "Serial and parallel genetic algorithms as function optimizers," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, S. Forrest, Ed. 1993, pp. 177–183, Morgan Kaufmann.

[11] E. Alba and J. M. Troya, "Cellular evolutionary algorithms: Evaluating the influence of ratio," in *Parallel Problem Solving from Nature, PPSN VI*, M. Schoenauer et al., Ed. 2000, vol. 1917 of *Lecture Notes in Computer Science*, pp. 29–38, Springer-Verlag.

[12] E. Alba, C. Cotta, and J.M. Troya, "On the importance of the grid shape in 2d spatially structured gas," *Journal of Evolutionary Optimization*, vol. 1, no. 3, 2000.

[13] A. Ostermeier, A. Gawelczyk, and N. Hansen, "A derandomized approach to self-adaptation of evolution strategies," *Evolutionary Computation*, vol. 2, no. 4, pp. 369–380, Winter 1994.

[14] H. Kita, "A comparison study of self-adaptation in evolution strategies and real-coded genetic algorithms," *Evolutionary Computation*, vol. 9, no. 2, pp. 223–241, Summer 2001.

[15] J. Sarma and K. A. De Jong, "An analysis of local selection algorithms in a spatially structured evolutionary algorithm," in *Proceedings of the Seventh International Conference on Genetic Algorithms*, T. Bäck, Ed. 1997, pp. 181–186, Morgan Kaufmann.

[16] D. E. Goldberg, K. Deb, and J. Horn, "Massively multimodality, deception and genetic algorithms," in *Proceedings of the Parallel Problem Solving from Nature, PPSN II*, R. Männer and B. Manderick, Eds. 1992, pp. 37–46, North-Holland.

[17] S. Tsutsui, A. Ghosh, D. Corne, and Y. Fujimoto, "A real coded genetic algorithm with an explorer and exploiter populations," in *Proceedings of the Seventh International Conference on Genetic Algorithms*, T. Bäck, Ed. 1997, pp. 238–245, Morgan Kaufmann.

[18] K. A. De Jong, M. A. Potter, and W. M. Spears, "Using problem generators to explore the effects of epistasis," in *Proceedings of the 7th International Conference of Genetic Algorithms*, T. Bäck, Ed. 1997, pp. 338–345, Morgan Kaufman.

[19] S. Droste, T. Jansen, and I. Wegener, "A natural and simple function wich is hard for all evolutionary algorithms," in *Proc. of the Third Asia-Pacific Conference on Simulated Evolution and Learning*, Nagoya, Japan, 2000, pp. 2704–2709.

[20] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1977.

[21] H. Chen, N.S. Flann, and D.W. Watson, "Parallel genetic simulated annealing: A massively parallel SIMD algorithm," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 2, pp. 126–136, 1998.

[22] S. Khuri, T. Bäck, and J. Heitkötter, "An evolutionary approach to combinatorial optimization problems," in *Proceedings of the 22nd ACM Computer Science Conference*, Phoenix, Arizona, 1994, pp. 66–73, ACM Press.

[23] D. R. Stinson, *An Introduction to the Design and Analysis of Algorithms*, The Charles Babbage Research Center, Winnipeg, Manitoba, Canada, 1985 (second edition, 1987).