Optimización de problemas dinámicos con algoritmos evolutivos

Enrique Alba y Juan F. Saucedo

Resumen— **El interés en resolver problemas de optimización utilizando algoritmos evolutivos ha sido muy intenso e importante en las últimas décadas. Existe un gran conjunto de problemas que tienen en común un comportamiento no estacionario con gran interés en procesos de fabricación o incluso en sistemas tan cotidianos como el ascensor de un edificio. La optimización de problemas dinámicos representa otra rama de investigación donde muchos de los avances existentes son útiles pero muchos otros no lo son, de forma que es necesario explorar nuevas técnicas.**

En este artículo estudiaremos de forma comparada varias de dichas técnicas, entre las que destacan los genotipos diploides y la selección disruptiva. Como es previsible que la tasa de mutación y el modelo algorítmico tengan un efecto importante en el desempeño de la búsqueda incluimos algunos resultados a este respecto.

Nuestras conclusiones también confirman algunas sugerencias acerca de las deficiencias de los genotipos diploides frente a los genotipos haploides, y muestran cómo la selección disruptiva es simple y eficaz al mismo tiempo (con ciertas connotaciones). Ya que usamos un modelo de algoritmo genético de estado estacionario, comprobaremos que una "elevada" mutación mejora la adaptación continua y que el tipo de reemplazo influye también en dicha adaptación.

Palabras clave— **Algoritmos Evolutivos, Entornos Dinámicos.**

I. Introducción

DESDE la aparición de los algoritmos genéticos (GA's) y evolutivos en general (EA's) el número y tipo de aplicaciones resueltas satisfactoriamente es realmente elevado [BFM97]. Sin embargo, a pesar de existir dominios claramente específicos como la optimización multi-objetivo, sujeta a restricciones, etc., y a pesar del uso en ingeniería, matemáticas, inteligencia artificial, etc., podemos fácilmente concluir que todos ellos tienen un punto en común: el uso de una única función que no cambia durante el tiempo que dura la búsqueda.

Sin embargo, también son de interés los dominios de optimización en que, o bien la función incorpora algún tipo de ruido, o bien la función fluctúa en el tiempo. Estos problemas son denominados No Estacionarios o Dinámicos, y el creciente interés que actualmente suscitan está abriendo las puertas a la investigación en un nuevo dominio: DOP (*Dynamic Optimization Problems*) [Bra01].

Dpto. de Lenguajes y Ciencias de la Computación (LCC). E.T.S.I. Informática, Teatinos, 3.2.12. 29071, Málaga, ESPAÑA. E-mail: eat@lcc.uma.es

Las aplicaciones del mundo real que pueden modelarse como un problema DOP son muy abundantes. Algunos ejemplos son el sistema de semáforos de una ciudad, el proceso de atención de llamadas de un ascensor o la selección de rutas en una red de comunicación. Todos estos sistemas son inherentemente dinámicos. Esto implica que cualquier algoritmo que desee optimizar un sistema dinámico debe considerar que *al mismo tiempo* que se está buscando el óptimo en el espacio de búsqueda, dicho espacio cambia su forma. En el caso de uno o varios ascensores, mientras perseguimos minimizar el tiempo medio de espera de las personas de un edifico, nuevas llamadas pueden producirse en otras plantas.

Nuestra aportación inicial en este terreno consiste en comparar técnicas que se han propuesto por separado, usando para ello un algoritmo único con la intención de obtener conclusiones fundamentadas. Ya que mantener la diversidad parece un factor clave para una rápida adaptación al nuevo óptimo, las propuestas existentes intentan evitar la convergencia prematura y/o la pérdida de información que en el pasado ha resultado útil (en previsión de que pueda volver a serlo). Nuestro análisis incluye mecanismos de distinta naturaleza que afectan al genotipo (cromosomas diploides), al fenotipo (selección disruptiva), e influencia de operadores (mutación y reemplazo).

En este artículo primero caracterizaremos los problemas dinámicos en la Sección II, presentando asimismo nuestro caso de estudio. En la Sección III revisamos las técnicas existentes para estos problemas. La Sección IV presenta el algoritmo utilizado. La Sección V contiene un estudio sobre genotipos diploides, mutación y selección disruptiva; cada uno representante de las diferentes técnicas existentes. La Sección VI estudia la importancia del reemplazo en nuestro algoritmo base, y finalmente la Sección VII resume las conclusiones más importantes y discute las líneas de trabajo futuro que tienen interés a la luz de los resultados obtenidos.

II. Problemas DOP

Un problema de optimización dinámica puede definirse como:

Definición 1: **(Problema Dinámico de Optimización)**. Un problema dinámico de optimización requiere resolver un problema en el cual hay que encontrar un extremo absoluto (máximo o mínimo) de cierta función la cual varía con el tiempo de forma continua o discreta. La función de adecuación sería:

$$f(x,t) \rightarrow \{max|min\} \quad t \in T \tag{1}$$

Usualmente, el cambio se produce de forma discreta, manteniéndose constante en intervalos de tiempo, e incluso el cambio se produce de forma cíclica entre varias funciones (usualmente entre dos), es decir,

$$f(x,t) = \begin{cases} g_1(x) & \text{si } t \in U_1 \subset T \\ g_2(x) & \text{si } t \in U_2 \subset T \\ \vdots & \vdots \\ g_n(x) & \text{si } t \in U_n \subset T \end{cases} \tag{2}$$
$$\text{con } U_1 \cap U_2 \cap \cdots \cap U_n = \emptyset$$

donde los U_i pueden ser no conexos. A cada parte conexa y cíclica de los U_i, t_i, se le llama época. Se define el período de oscilación p_i como la duración de una época t_i. Además de ser usual utilizar sólo dos funciones, también es usual usar un período único, tal que $\forall i, p_i = p$. Sin embargo, debemos aclarar que los períodos de duración no tienen por qué ser iguales, incluso cada evaluación de un individuo podría ser distinta dentro de la misma generación (!), con lo que p_i podría identificar el número de evaluaciones y no de generaciones de una época.

$$f\left(\vec{x}, t_i\right) = \begin{cases} f_1\left(\vec{x}\right) & i \text{ mód } 2 = 0 \\ f_2\left(\vec{x}\right) & i \text{ mód } 2 \neq 0 \end{cases} \tag{3}$$

Como hemos dicho lo normal es usar una definición como la ecuación (3) en la que dos funciones se alternan con un cierto período único de p generaciones. El proceso de optimización se asemeja así a un proceso de aprendizaje de los dos óptimos. La mejor técnica para un problema dado será aquella que más rápidamente localice al nuevo óptimo tras cada cambio.

Podemos considerar que los casos extremos en los problemas dinámicos son: (1) el estacionario, en el cual la función no cambia durante el tiempo de búsqueda, y (2) la funciones con ruido, en las cuales el cambio se puede producir en cada evaluación de un individuo cualquiera.

De entre el elevado número de problemas dinámicos hemos seleccionado uno conocido como el problema de la *mochila dinámica* [GS87], consistente en maximizar beneficios (incluyendo muchos objetos) pero minimizando su coste o peso total. Básicamente se trata de un problema estático que modificamos para ser dinámico usando dos valores de peso máximo distintos para la capacidad de la mochila $W_1 = 60$ y $W_2 = 104$. Por tanto, con cierto período, $p = 50$ generaciones, conmutamos la capacidad de la mochila, consiguiendo el dinamismo.

Hemos elegido este problema por pertenecer a la importante clase de problemas conocida como NP-Completos, así como por incorporar restricciones. Ambas características son muy usuales en problemas reales. Asimismo, es un problema absolutamente clásico en la optimización DOP usando algoritmos evolutivos. Aunque es interesante abordar problemas de mayor dificultad y aplicabilidad en el presente trabajo, nos centraremos en estudiar los algoritmos evolutivo de forma unificada por cuestiones de espacio y diferimos a otro trabajo futuro un estudio enfocado a los problemas en vez de a las técnicas usadas para resolverlos.

TABLA I
Problemas de la mochila dinámica. 17 Objetos.

Objeto #	**Valor**	**Peso**	**Óptimo 71** ($W_1 = 60$)	**Óptimo 87** ($W_2 = 104$)
1	2	12	0	0
2	3	5	1	1
3	9	20	0	1
4	2	1	1	1
5	4	5	1	1
6	4	3	1	1
7	2	10	0	0
8	7	6	1	1
9	8	8	1	1
10	10	7	1	1
11	3	4	1	1
12	6	12	1	1
13	5	3	1	1
14	5	3	1	1
15	7	20	0	1
16	8	1	1	1
17	6	2	1	1
Total:	**91**	**122**	**13 Objetos**	**15 Objetos**

$$f(\vec{x}) = \begin{cases} v - \lambda\left(w - W\right)^2 & \text{si } w > W \\ 0 & \text{si } v - \lambda\left(w - W\right)^2 < 0 \\ v & \text{si } w \leq W \end{cases} \tag{4}$$

La función de adecuación es la presentada en la ecuación (4) siendo $v = \sum_{l=1}^{n} x_l v_l$ y $w = \sum_{l=1}^{n} x_l w_l$. Existen 17 objetos en total (ver tabla I) y su inclusión en la mochila viene dictada por un valor lógico binario x_i; v_i y w_i representan el valor y el peso del objeto i-ésimo, respectivamente. El parámetro $\lambda = 20$ (ver [GS87]) es un coeficiente de penalización, y W es la capacidad máxima, cuyos valores alternados son $W_1 = 60$ y $W_2 = 104$.

III. Técnicas para resolver DOPs

La resolución de problemas dinámicos no es una tarea trivial, y por tanto se han propuesto varias estrategias que se pueden clasificar como sigue:

1. **Incorporación de Memoria Histórica.** El EA incorpora información sobre las soluciones que ya antes ha encontrado en situaciones similares. La memoria puede incorporarse a nivel de *algoritmo* o bien a nivel de *genotipo* [NW95]. De la misma manera puede tratarse de una memoria *explícita* (p.ej. almacenando los mejores individuos desde el principio de la evolución) o *implícita* (genotipos n-ploides). Un ejemplo de esta última son los genotipos diploides, donde cada individuo tiene dos alelos para determinar por ejemplo la inclusión de cada objeto en la mochila y debe usarse un esquema de dominancia para decidir el fenotipo de cada individuo (cuál alelo es el dominante).
2. **Preservación de la Diversidad**. Esta técnica es preventiva y consiste en introducir mecanismos en el EA destinados a mantener alta la diversidad. Encontramos muchas técnicas conocidas por ser un requisito no exclusivo de dominios dinámicos. Algunos ejemplos son la selección disruptiva [KH97], usar un EA de población estructurada [SJ99], etc.
3. **Reacción Frente al Cambio**. Se utiliza un EA estándar que realiza acciones específicas para adaptar la búsqueda al nuevo cambio únicamente cuando éste ocurre. Un ejemplo típico es elevar la tasa de mutación al ocurrir el cambio [LHR98].

El primer mecanismo dota al algoritmo de memoria para recordar información que fue útil en pasadas circunstancias, lo que resultará de interés si el nuevo entorno es similar a uno que ya se exploró anteriormente. La preservación de la diversidad es un objetivo en casi cualquier propuesta de algoritmo evolutivo actual. Finalmente, la reacción frente al cambio es un tipo de técnica lógico, si podemos detectar de alguna manera que se ha producido dicho cambio. En las siguientes secciones describiremos las técnicas concretas que vamos a estudiar en este artículo.

A. Genotipo diploide

Los conceptos de genotipo diploide y dominancia son relativamente conocidos en computación evolutiva. Estos dos mecanismos se han utilizado en problemas dinámicos como una manera de evitar que los alelos menos útiles actualmente, pero que en el pasado fueron útiles, desaparezcan en el nuevo entorno actual. Existen muchos trabajos pioneros sobre este tema, tales como la disertación de Bagley en 1967 o la propuesta de genotipo trialélico de Hollstein en 1971. Sin embargo, los resultados eran poco concluyentes, permitiendo únicamente asegurar que estos genotipos diploides (redundantes) permiten mantener una mayor diversidad.

Existen algunos trabajos que sugieren que los genotipos diploides funcionan mejor que los haploides, por ejemplo sobre el problema de la mochila dinámica [GS87]. Desafortunadamente, en la actualidad existe aún mucha controversia sobre este tópico. Puede consultarse en [NW95] un serio análisis de las desventajas de distintos tipos de dominancia usando individuos con dos cromosomas binarios, poniendo en entredicho sus pretendidas ventajas sobre los genotipos haploides.

En el presente trabajo usaremos un esquema de dominancia simple (Figura 1) y cambiaremos la dominancia del 1 por la del 0 (tabla complementaria a la mostrada) cuando las prestaciones del algoritmo caigan drásticamente. Consideraremos una caída drástica cuando la adecuación de la población caiga de una generación a la siguiente más de un 20 %. Este cambio en la dominancia podría aplicarse a nivel de alelo, individuo o población [NW95]. En nuestro caso lo aplicamos a nivel de población, es decir, el mapa es común a toda la población pero sólo afecta a los nuevos individuos que aparezcan tras dicho cambio, y no a los que ya existían.

	0	1									
	0	1	*Cromosoma1*	1	0	1	1	0	0	1	0
0	0	1	*Cromosoma2*	1	0	0	1	0	1	0	0
1	1	1	*Fenotipo*	1	0	1	1	0	1	1	0

Individuo

Tabla de dominancia. Proceso de expresión (genotipo a fenotipo) de un individuo diploide.

Fig. 1. Tabla de dominancia y proceso de expresión.

En la naturaleza también existen ejemplos de cambio de la dominancia como respuesta a un cambio drástico en el ambiente. En nuestro modelo, la dominancia de la población cambiará así:

$$\Delta = \begin{cases} |\overline{f}_t - \overline{f}_{t-1}| & \text{si } t \bmod p \neq 0 \\ \overline{f}_t & \text{si } t \bmod p = 0 \end{cases}$$
$$\text{y si} \begin{cases} \Delta > 0{,}2 & \text{Mantener Dominancia Actual} \\ \Delta \leq 0{,}2 & \text{Cambio de Dominancia} \end{cases} \quad (5)$$

en donde $\overline{f}_t$ representa la media normalizada de la población, t representa la generación actual y p es el período de oscilación.

Por tanto, usamos un mapa de dominancia que cambiamos al mapa complementario cuando las prestaciones caen. Aunque no hacemos un cambio explícito de estrategia cuando hay un cambio, sí que indirectamente cambia la dominancia, ya

que las prestaciones suelen caer al haber un cambio. En este sentido, podríamos decir que nuestra dominancia es un híbrido entre la dominancia tradicional y un esquema de adaptación como los discutidos a continuación.

B. Adaptación al cambio

La adaptación al cambio consiste en introducir en el algoritmo un dispositivo que se active cuando la función objetivo cambie. Parece claro que esto sólo puede hacerse cuando se conozca a priori este hecho, o bien desarrollando algún tipo de detector de cambios en el entorno.

Tradicionalmente, elevar la mutación de alguna manera ha sido la reacción típica al cambio en el entorno. El esquema de elevación y posterior disminución de la tasa de mutaciones, usualmente, suele ser complejo. En nuestro caso, aunque reconocemos la importancia de este operador y de hecho dedicamos a él la Sección V-B, no vamos a incluir resultados con un mecanismo de cambio dinámico de la mutación (únicamente estudiaremos la influencia de su tasa).

Nuestro tipo de reacción frente al cambio (de manera indirecta) es modificar la dominancia atendiendo a la media normalizada de la población. Debemos tener entonces en cuenta que la primera generación tras un cambio en la función objetivo debe mantener, al menos inicialmente, la dominancia existente a partir de la segunda generación (como indica la ecuación (5)).

C. Mantenimiento de la diversidad

Otra de las técnicas utilizadas para resolver problemas DOP es mantener la diversidad de la población a lo largo de la búsqueda. Esto puede conseguirse de varias maneras:

1. Usando mutación alta siempre.
2. Insertando nuevos individuos de contenido aleatorio en la población con cierta frecuencia.
3. Usando un algoritmo de población estructurada [AT00].
4. Limitando el emparejamiento de individuos.
5. Alterando la adecuación de los individuos para favorecer la diversidad.

Entre los métodos que preservan la diversidad, la selección disruptiva [KH97] es una de las técnicas más simples. Como muestra la ecuación (6), la selección disruptiva consiste simplemente en evaluar los individuos y modificar su adecuación bruta restándoles la adecuación media de la población. Cualquiera que sea el posterior método de selección usado (en nuestro caso proporcional) se favorecerá a los individuos distintos de la media, sean los mejores o los peores.

$$f(\vec{x},t) = |f(\vec{x},t) - \overline{f}(t)| \qquad (6)$$

Al aplicar selección disruptiva evitamos homogeneizar la población a la hora de la selección; los individuos mediocres tendrán una adecuación cercana a 0 y por tanto, la presión, para desaparecer sobre los peores adaptados, disminuye.

Debemos resaltar un detalle. Puesto que vamos a mezclar las técnicas que estamos presentando para compararlas, ocurrirá que, usando genotipos diploides, las oscilaciones provocadas al usar disrupción superen fácilmente el 20 %. Por tanto, la dominancia cambiaría constantemente, perdiendo su utilidad. Por ello desactivaremos el cambio de dominancia únicamente cuando se mezclen en el mismo algoritmo un genotipo diploide y selección disruptiva.

En la práctica esto no supone ninguna restricción, ya que se suele optar por usar genotipos diploides o bien por selección disruptiva, sin mezclarlos. Nuestra idea de un algoritmo diploide que además utiliza selección disruptiva tiene un doble objetivo: lograr un mayor grado de diversidad, y aumentar la capacidad de adaptación. La validez de esta propuesta se estudiará en las secciones siguientes.

IV. El algoritmo utilizado

En este trabajo utilizamos un algoritmo genético de estado estacionario (ssGA) [Sys91] en donde cada paso genera un nuevo individuo que reemplaza al peor existente sólo si es mejor. Se trata pues de un algoritmo evolutivo de tipo $(\mu + 1)$ que selecciona dos padres utilizando selección proporcional a la adecuación por ruleta, cruce de un punto ($p_c = 0{,}075$) y mutación por inversión de bits (probabilidad variable). Consulte el pseudocódigo en la Figura 2.

```
[ssGA]
proc Ciclo_Reproductor (ag):
    para s←1 hasta MAX_PASOS hacer
        padre1 ← Seleccionar_RW(ag.pob);
        padre2 ← Seleccionar_RW(ag.pob);
        SPX(ag.p_c, padre1, padre2, ind_aux.crom);
        Mutar(ag.p_m, ind_aux.crom);
        ind_aux.adecuación ←
            ag.Evaluar(Decodificar(ind_aux.crom));
        Insertar_Nuevo_Ind(ag, ind_aux, si_mejor);
        Recolectar_Estadísticas(ag);
    fin_para;
fin_proc Ciclo_Reproductor;
```

Fig. 2. Pseudocódigo del algoritmo genético utilizado.

Este algoritmo difiere de los algoritmos generacionales tradicionales en que se necesitan μ iteraciones para desarrollar una nueva generación completa. Normalmente, un ssGA posee una presión selectiva mayor que la de uno generacional, lo que le hace converger más rápido en muchas ocasiones.

De hecho, si se reemplazara a cualquier cadena, aleatoriamente elegida, tendríamos un algoritmo de igual presión que la dada en un GA generacional. En particular, exploraremos la incidencia que tiene el tipo de reemplazo en problemas dinámicos (Sección VI). En la práctica, los ssGA son muy usados en dominios complejos; algunos ejemplos son GENITOR y GENITOR II [WS90], xxGA [AT00], o en trabajos DOP [VF96].

Los parámetros que utilizaremos incluyen usar una población de 75 individuos, sean estos haploides o diploides. Aunque muchos otros autores no discuten este punto, es importante dedicar el mismo número de individuos para ambos tipos de algoritmo, ya que de otro modo el diploide tendría ventaja por usar el doble de memoria que el haploide. Sin embargo, aquí usamos 75 individuos siempre porque comprobamos que usar 150 haploides y 75 diploides arrojan los mismos resultados que usar 75 en ambos tipos de algoritmo.

V. Resultados

En esta sección presentamos los resultados de comparar las distintas técnicas explicadas en la Sección III. Nuestro objetivo primario es delimitar las ventajas e inconvenientes del genotipo diploide y de la selección disruptiva. En segundo lugar nos interesa estudiar el papel de la mutación y del reemplazo en cualquier algoritmo usado para resolver un problema DOP. Presentamos medias de 30 ejecuciones independientes de todos los algoritmos.

Debemos recalcar que utilizamos siempre una selección proporcional a la adecuación de ambos padres. Asimismo, reemplazamos al peor individuo si el nuevo es mejor y únicamente usamos el aleatorio (para comparar) en la Sección VI.

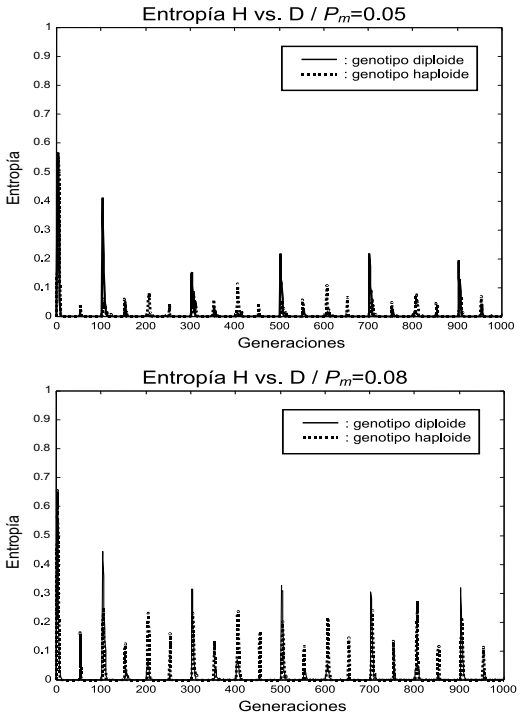
A. Genotipo diploide vs. haploide

A priori se podría pensar que los genotipos haploides están en desventaja frente a los diploides, pues se supone que los últimos mantienen mayor diversidad en la población al poseer un mayor número de cromosomas en la población, sin embargo, no está claro que este hecho permita una adaptación más rápida al nuevo entorno. Una medida típica de la diversidad es la *entropía* H de la población $P(t)$:

$$H\left[P(t)\right] = -\frac{1}{l}\sum_{i=1}^{l}\left(P_0^i \log_2 P_0^i + P_1^i \log_2 P_1^i\right) \tag{7}$$

donde P_0^i/P_1^i es la proporción de 0's/1's en el locus i, y l es la longitud del cromosoma.

Estudiando la entropía para distintos valores de mutación se observa en la Figura 3 que la representación diploide ni siquiera es capaz de mantener diversidad fenotípica (resultado de aplicar la dominancia a ambos cromosomas) a las pocas generaciones de encontrar uno de los óptimos. Los picos de diversidad diploide sí son más altos que los haploides, pero igualmente efímeros. Podemos observar que aumentando la probabilidad de mutación de 0,05 (Figura 3 arriba) a 0,08 (Figura 3 abajo) mejoramos la diversidad algunas generaciones más tras un cambio.



Tasas de mutación 0,05 (arriba) y 0,08 (abajo).

Fig. 3. Entropía Haploide vs. Diploide.

Puede notarse en la Figura 4 cómo el algoritmo haploide localiza muy rápidamente los óptimos 87 y 71, mientras que el algoritmo diploide a veces no los localiza. Este resultado es decepcionante porque ni siquiera un cambio de dominancia parece mejorar un genotipo diploide simple.

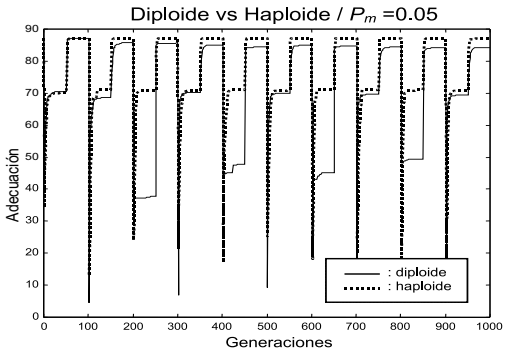


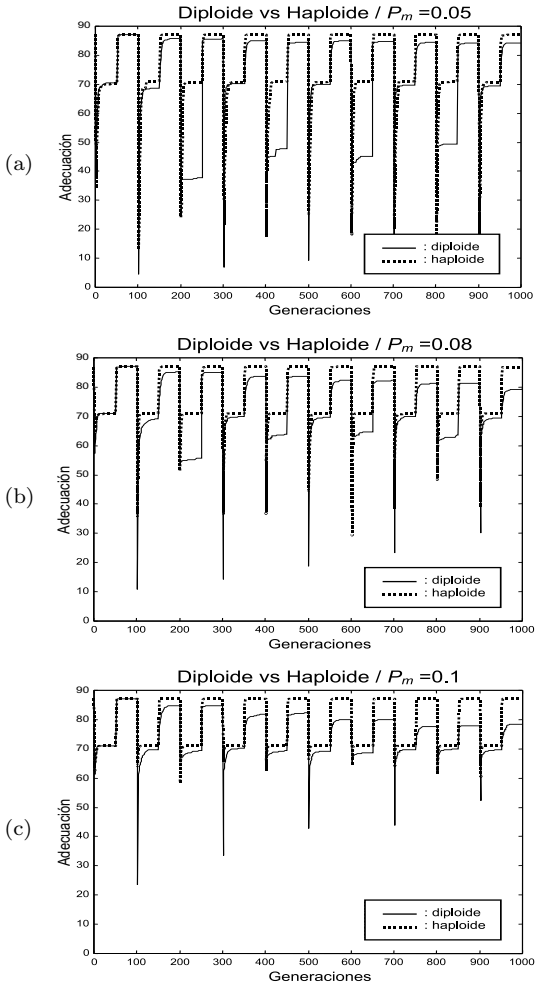
Fig. 4. Mejor adecuación observada, Haplo. vs. Dipl.

B. Importancia de la mutación

En problemas dinámicos es usual encontrar que la adaptación a los cambios de los distintos algoritmos comparados es, o bien lenta, o bien lleva a las cercanías del óptimo, pero no a dicho óptimo.

Este efecto aparece con frecuencia, ya que una vez alcanzado un objetivo es inevitable una convergencia de toda la población hacia los individuos mejor adaptados, disminuyendo así la diversidad necesitada. Esta es la razón por la cual la mutación adquiere una importancia transcendental en los algoritmos para problemas DOP.

Mantener una mutación muy alta podría parecer una solución. Pero en la práctica esta solución es demasiado drástica, pues con una mutación demasiado alta se reduce mucho la eficacia del algoritmo, al asemejarse a un algoritmo de búsqueda aleatoria. Por tanto es importante encontrar una solución de compromiso, ya que aquí no consideramos la posibilidad de variar la mutación de manera dinámica.



Tasas de mutación:
(a) $p_m = 0{,}05$, (b) $p_m = 0{,}08$ y (c) $p_m = 0{,}1$

Fig. 5. Importancia de la mutación.

Note en las tres gráficas de la Figura 5 cómo elevar la mutación (0,05, 0,08 y 0,1) es un remedio eficaz en ssGA. Los algoritmos haploides han resistido muy bien los tres tipos de mutación, tanto en velocidad de adaptación como en exactitud al localizar ambos óptimos. La versión diploide ha mejorado sustancialmente porque, con la subida de la mutación, disminuyen sus caídas de adecuación tras cada cambio. Sin embargo, se observa especialmente bien en la Figura 5(c) que las prestaciones del diploide se degradan a medida que aumenta el número de oscilaciones de una función objetivo a otra. Podemos también concluir que los ssGA son resistentes ante una alta mutación.

C. Selección con disrupción vs. sin disrupción

En esta sección presentamos una comparativa novedosa entre algoritmos que utilizan disrupción y algoritmos sin ella. Nuestro aportación es estudiar este operador, no sólo sobre genotipos haploides (para los que fue propuesto) sino también diploides. Haremos que este estudio incluya varias tasas de mutación puesto que usar una sola tasa podría llevarnos a conclusiones sesgadas.

Obsérvese en la Figura 6 (fila inferior.) que los algoritmos de genotipo diploide se ven beneficiados por el uso de disrupción, evitando ésta la degradación que se ha observado para alta mutación (0,1). Realmente es notable la mayor velocidad de adaptación y exactitud en dicho caso.

Pasemos ahora a analizar el genotipo haploide. Elevar la tasa de mutación no altera su comportamiento al no usarse disrupción más que para obtener una mayor velocidad de adaptación (Figura 6 fila superior). Usar o no disrupción es prácticamente equivalente para una tasa de mutación baja (0,05), con una ligera ventaja al usar disrupción porque las caídas de adecuación son pequeñas. En general, cuando usamos disrupción, elevar la mutación ralentiza la velocidad de adaptación y perjudica la exactitud del algoritmo, que ahora no alcanza los óptimos. Esto se debe a que su mayor diversidad enlentece sin necesitarlo al algoritmo.

VI. Tipo de reemplazo

Para enlazar nuestros resultados con otros trabajos que utilizan algoritmos generacionales [GS87] comparamos en esta sección el comportamiento de nuestro ssGA con dos tipos de reemplazo: reemplazamos al peor individuo (si el nuevo es mejor) o reemplazamos a cualquiera elegido aleatoriamente de la población. La razón para hacer esto último es comprobar el comportamiento de algoritmos de menor presión selectiva como

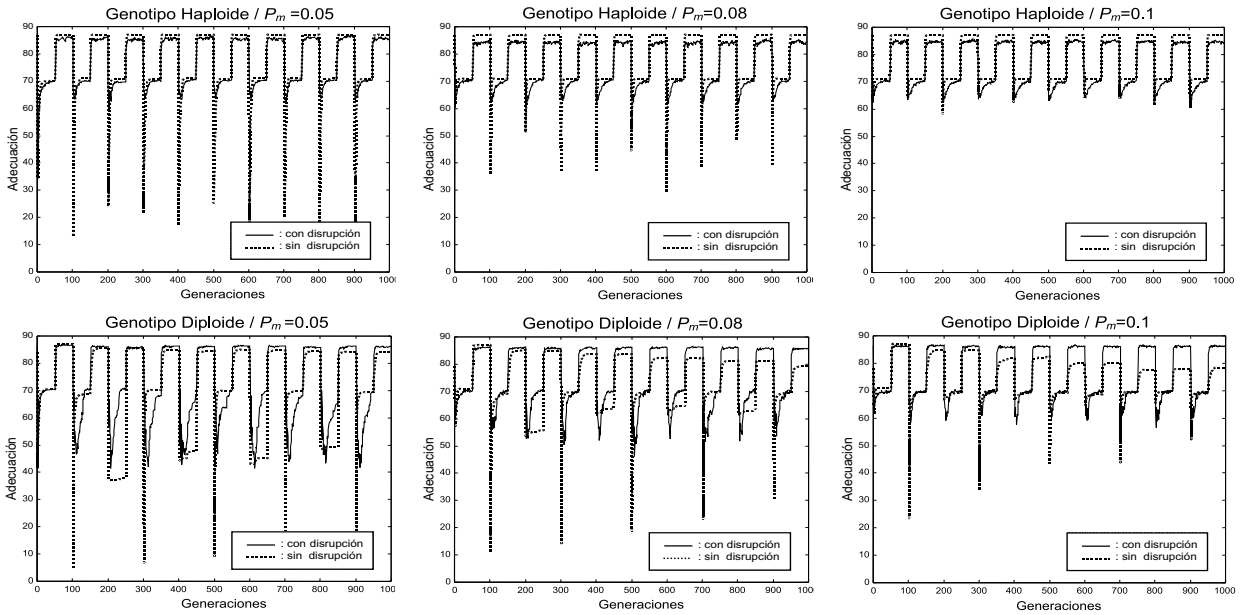
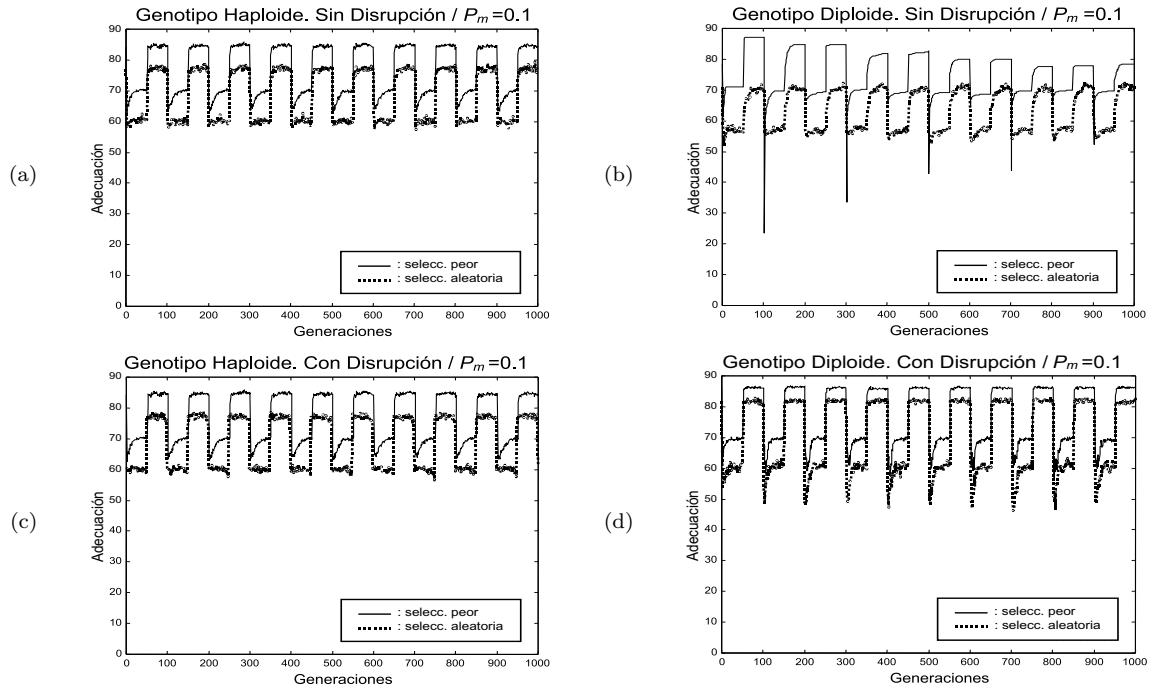


Fig. 6. Ventajas de usar selección disruptiva en genotipos haploides frente a diploides.



Presentamos resultados para genotipos haploides (a,c) y diploides (b,d), tanto sin usar disrupción (a,b) como usando disrupción (c,d). La tasa de mutación es 0,1 en todos los casos.

Fig. 7. Inconvenientes del uso del reemplazo aleatorio frente al del peor individuo.

los generacionales, ya que el reemplazo aleatorio de un ssGA simula el comportamiento generacional a largo plazo [Sys91].

Podemos comprobar en la Figura 7 que el reemplazo aleatorio no afecta a la velocidad de convergencia en ningún caso, pero sí que destruye totalmente la exactitud del algoritmo, ya que nunca consigue alcanzar ninguno de los óptimos.

Nótese en el genotipo diploide sin disrupción (Figura 7b) la ya discutida degradación progresiva cuando el reemplazo es del peor; en el caso de reemplazo aleatorio no aparece esta degradación, pero los resultados son incluso peores que en el resto de algoritmos. Fíjese que la disrupción mejora el comportamiento del genotipo diploide, especialmente si el reemplazo es aleatorio (Figu-

ras 7b y 7d). Esto explica por qué en algoritmos generacionales la disrupción es más efectiva que en ssGA, ya que en el caso de reemplazo aleatorio es cuando únicamente se perciben sus ventajas.

Como conclusión, parece claro que es preferible usar en cualquier caso (compruebe la Figura 7 al completo) un reemplazo de la peor solución en un ssGA, tal como hemos venido haciendo a lo largo de todo este trabajo.

VII. Conclusiones

En este artículo hemos comparado exhaustivamente dos técnicas propuestas para resolver problemas DOPs: genotipos diploides y selección con disrupción. También hemos estudiado la importancia de la mutación y del reemplazo en el algoritmo usado.

Los resultados demuestran que un mapa de dominancia simple, aún mejorado con un cambio de dominancia al degradarse el comportamiento, no es suficiente para asegurar que un genotipo diploide es preferible a otro haploide. Existen resultados similares utilizando mapas de dominancia más sofisticados pero que aún no han sido chequeados en ssGA.

La selección disruptiva es un mecanismo interesante que puede utilizarse con bajo costo computacional para mantener la diversidad. Hemos comprobado que la selección disruptiva es útil en el caso de un algoritmo de baja presión selectiva (generacionales o ssGA con reemplazo aleatorio) y con genotipos diploides, cualquiera que sea el tipo de reemplazo (nuestra contribución). En un ssGA con reemplazo del peor su efecto es únicamente una ralentización del proceso de convergencia que en general es perjudicial.

En un ssGA con reemplazo del peor podemos usar libremente un rango de mutación superior que en algoritmos de otro tipo, con lo que resulta una plataforma ideal para proponer técnicas basadas en cambios de la tasa de mutación. De acuerdo con los resultados publicados en [VF96], [WS90], a nuestra Sección VI y a las características generales publicadas en [Sys91], preferimos, para el problema DOP estudiado, un ssGA con reemplazo del peor frente a uno con reemplazo aleatorio o su equivalente generacional.

Finalmente, aunque hemos intentado reproducir los resultados existentes usando sus mismas parametrizaciones, descubrimos que un ssGA haploide es una herramienta suficientemente potente para resolver problemas DOP, al menos las versiones más comunes de éstos. Como línea de trabajo futuro estamos estudiando directamente algoritmos generacionales y celulares [SJ99], encontrando grandes ventajas en estos últimos para resolver problemas dinámicos de dimensiones reales. Igualmente, parece necesario extender nuestros resultados para abarcar el resto de propuestas DOP existentes de forma que podamos tomar conciencia de sus prestaciones.

VIII. Agradecimientos

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 (el proyecto TRACER). Agradecemos la colaboración de José Manuel López Muñoz.

Referencias

[AT00] E. Alba and J. M. Troya. Influence of the migration policy in parallel distributed GAs with structured and panmictic populations. *Applied Intelligence*, 12(3):163–181, 2000.

[BFM97] T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Oxford University Press, 1997.

[Bra01] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Klüwer Academic Publishers, 2001.

[GS87] D. E. Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 59–68. Lawrence Erlbaum Associates, 1987.

[KH97] T. Kuo and S.-Y. Hwang. Using disruptive selection to maintain diversity in genetic algorithms. *Appl. Intell., Int. J. Artif. Intell. Neural Netw. Complex Probl.-Solving Technol (Netherlands)*, 7(3):257–267, 1997.

[LHR98] J. Lewis, E. Hart, and G. Ritchie. A comparison of dominance mechanisms and simple mutation on non-stationary problems. In Agoston E. Eiben, Thomas B.ªck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature – PPSN V*, pages 139–148, Berlin, 1998. Springer. Lecture Notes in Computer Science 1498.

[NW95] K. P. Ng and K. C. Wong. A new diploid scheme and dominance change mechanism for non-stationary function optimization. In Larry Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 159–166, San Francisco, CA, 1995. Morgan Kaufmann.

[SJ99] J. Sarma and K. A. De Jong. The behavior of spatially distributed evolutionary algorithms in non-stationary environments. In Banzhaf W. et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 572–578. Morgan Kaufmann, 1999.

[Sys91] G. Syswerda. A study of reproduction in generational and steady-state genetic algorithms. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 94–101. Morgan Kaufmann, 1991.

[VF96] F Vavak and T C Fogarty. Comparison of steady state and generational gas for use in nonstationary environments. In *Proceedings of the 1996 IEEE Conference on Evolutionary Computation*, pages 192–195. IEEE Press, 1996.

[WS90] D. Whitley and T. Starkweather. GENITOR II: a distributed genetic algorithm. *J. Expt. Theor. Artificial Intelligence*, 2:189–214, 1990.