# Nichos y Operadores de Macromutación para Problemas de Optimización con Funciones Multimodales No-Estacionarias

Susana Esquivel, Victoria Aragón

Laboratorio de Investigación y Desarrollo en Inteligencia Computacional*
Universidad Nacional de San Luis
Ejército de los Andes 950
(5700) San Luis, Argentina
{esquivel,vsaragon}@unsl.edu.ar
Workshop de Agentes y Sistemas Inteligentes

**Resumen**

El interés en la optimización de funciones multimultimodales dinámicas se ha acrecentado en los últimos años debido a que muchos de los problemas del mundo real poseen esta característica. En este tipo de problemas, el objetivo es seguir la trayectoria del óptimo global tan cercanamente como sea posible, a medida que éste se desplaza sobre el espacio de búsqueda. En este trabajo se presentan tres métodos de mantenimiento de diversidad, uno de nichos conocido como *fitness sharing* y dos operadores de macromutación: *random inmigrantes* y *recrudescencia*. La efectividad y/o limitaciones de cada técnica y de combinaciones de ellas es discutida y analizada.

**Palabras Claves:** Algoritmos Evolutivos, Funciones Dinámicas Multimodales, Técnicas de Mantenimiento de Diversidad, Métodos de Nichos.

*El LIDIC es financiado por la Universidad Nacional de San Luis y por la ANPCyT (Agencia Nacional para promover la Ciencia y Tecnología).

# 1. Introducción

Un *landscape* dinámico multimodal es un espacio de búsqueda donde una o más características topológicas (altura, pendiente y ubicación) de los picos que lo componen cambian en el tiempo. Ejemplos de problemas que dan lugar a este tipo de *landscapes* incluyen: nuevos jobs que arrivan continuamente y deben ser agregados a un *schedule*, máquinas que deben entrar en mantenimiento, etc.

Los *landscapes* dinámicos multimodales presentan un desafío para cualquier técnica de búsqueda debido a que:

1. El número total de características topológicas en el *landscape* puede cambiar,
2. La pendiente de uno o más picos en el *landscape* puede(n) cambiar,
3. Uno o más picos del *landscape* puede(n) ser realocado(s),
4. Las alturas relativas de uno o más picos en el *landscape* pueden ser alteradas,
5. Las restricciones del problema pueden cambiar.

Los puntos 2, 3 y 4 pueden provocar que nuevos picos emerjan, éstos deben ser localizados rápidamente para evitar que se pierdan soluciones potencialmente buenas. Por lo antedicho, un algoritmo de optimización que busque soluciones en *landscapes* dinámicos multimodales debe ser capaz de enfrentar estos desafíos localizando y/o manteniendo óptimos actuales y nuevos.

Para cada uno de los tipos de cambios planteados en los puntos 1 a 5, una variedad de dinámicas pueden ser aplicadas. Por ejemplo, la magnitud del cambio puede ser pequeña, larga o caótica y la velocidad de los cambios puede ser rápida o lenta. Cuando la función que se optimiza es dinámica, el objetivo no es alcanzar el valor óptimo de la función sino acompañar su trayectoria a lo largo del espacio de búsqueda tan cercanamente como sea posible.

Los métodos de *nichos* han sido desarrollados para mantener diversidad en la población y permitir a los Algoritmos Evolutivos (AE's) explorar varios picos simultáneamente. También previenen que la población sea atrapada por un óptimo local en el espacio de búsqueda [17]. El método de *sharing* es probablemente el más conocido y también el más utilizado entre las técnicas de *nichos*. Este método fue introducido originalmente por Holland [9] y mejorado por Goldberg y Richardson [6].

Para este tipo de problemas de optimización, la Computación Evolutiva (CE) ofrece ventajas sobre las heurísticas que no están basadas en población. La principal ventaja recae en el hecho que, los AE's mantienen una población de soluciones. Consecuentemente, frente a un cambio, ellos tienen la posibilidad de moverse de una solución a otra para determinar si aún una o más de ellas poseen el mérito suficiente para continuar la búsqueda a partir de ellas [3].

Por otro lado, los AE's tienen una desventaja, conforme evoluciona la población, ésta converge hacia alguna zona del espacio de búsqueda, idealmente hacia el óptimo global, entonces cuando el *landscape* se modifica, la población posiblemente ha perdido su diversidad inicial, característica por la cual se considera necesario complementar a los AE's con técnicas que permitan mantener la diversidad de la población.

Entre 1987 y 1992, Goldberg y Smith [7], Cobb [4] y Grefenstette [8] comenzaron la investigación acerca del comportamiento de los AE's sobre funciones de *fitness* dinámicas. Recientemente, el interés en esta área ha crecido drásticamente [14], [10], [2], [20], [11], [19], [13], [21], [15].

El trabajo está organizado de la siguiente manera. La Sección 2 describe los conceptos básicos de las técnicas de mantenimiento de diversidad, la Sección 3 decribe el generador de funciones de prueba empleado para validar las técnicas implementadas. La Sección 4 presenta

las características comunes de los distintos EA's propuestos. En la Sección 5 se detallan los experimentos y parámetros usados por los algoritmos. En la Sección 6 se muestran y analizan los resultados y, finalmente, en la Sección 7 se presentan las conclusiones y las líneas de trabajo presente y futuras.

# 2. Técnicas de Mantenimiento de la Diversidad

## 2.1. Fitness Sharing

La función de *Fitness Sharing* determina una degradación en el beneficio (*fitness*) de un individuo debido a la presencia de vecinos a una cierta distancia, medida como alguna similitud del espacio. Típicamante, el *fitness* compartido $f_i^{'}$ de un individuo $i$ con *fitness* $f_i$ es simplemente: $f_i^{'} = \frac{f_i}{m_i}$, donde $m_i$ es el contador del nicho, éste mide el número aproximado de individuos con los cuales el individuo i comparte su *fitness*. El contador de nicho $m_i$ se calcula sumando una función de compartición, *sharing*, sobre todos los miembros de la población: $m_i = \sum_{j=1}^{N} sh(d_{ij})$, donde N denota el tamaño de la población y $d_{ij}$ representa la distancia entre el individuo $i$ y el individuo $j$. Así, la función de *sharing* $sh$ mide el nivel de similitud entre dos elementos de la población. Esta retorna 1 si los elementos son idénticos, 0 si su distancia es mayor a un umbral de disimilitud y un valor intermedio acorde al grado de disimilitud de los elementos. La función de *sharing* más utilizada está definida:

$$sh(d_{ij}) = \begin{cases} 1 - (d_{ij}/\sigma_{share})^{\alpha} & d_{ij} < \sigma_{share} \\ 0 & en\ otro\ caso \end{cases}$$

donde $\sigma_{share}$ determina el umbral de disimilitud, como así también el radio de los nichos, y $\alpha$ es un paramétro constante que regula la forma de la función de compartición. El valor de $\alpha$ comúnmente es 1 y, en este caso, la función de *sharing* resultante se conoce con el nombre de función de *sharing* triangular. La distancia $d_{ij}$ entre dos individuos i y j está caracterizada por una métrica de similitud basada en la similitud genotípica o fenotípica. La similitud genotípica está relacionada con la representación de cadenas de bits y, generalmente, es la distancia de *Hamming*, mientras que la similitud fenotípica está directamente relacionada a los paramétros reales del espacio de búsqueda, aquí la distancia más utilizada es la distancia *Euclideana*. La similitud fenotípica parece mostrar mejores resultados que la similitud genotípica [5].

## 2.2. Operadores de Macromutación

*Recrudescencia* [12]: este operador incrementa las probabilidades de recombinación y mutación de una porción de la población y produce una reorganización radical genotípica de los individuos a los cuales se les aplica. Este operador se aplica en cada generación con una cierta probabilidad $P_{Recru}$.
*Random Inmigrantes* [8]: individuos generados aleatoriamente reemplazan a un porcentaje de la población. Estos se insertan en la población una vez que se determina que ha ocurrido un cambio en la función.

# 3. Funciones de Prueba Dinámicas

En esta Sección se describe al Generador de Problemas de Prueba DF1 propuesto por Ronald Morrison y Kenneth De Jong [16] empleado para validar los AE's propuestos.

La función estática base genera una morfología de *landscapes* conocida con el nombre de *campos de conos* y se define como:

$$f(x_1, \ldots, x_n) = \max_{i=1,M} [H_i - R_i * \sqrt{(x_1 - x_{1i})^2 + (x_2 - x_{2i})^2 + \ldots + (x_n - x_{ni})^2}]$$

donde ($x_1$,...,$x_n$) son los genes de un individuo, M especifica el número de conos en el ambiente, ($x_{1i}$,...,$x_{ni}$) son las coordenadas del cono $i$, $x_{ki} \in$ [-1.0,1.0] con $k$ = 1,...,$n$. $H_i$ es la altura del cono $i$ y $R_i$ es la pendiente del cono $i$. Cada cono se especifica independientemente por su ubicación ($x_{1i}$,...,$x_{ni}$), su altura $H_i$ y su pendiente $R_i$. La función máx es la encargada de combinar todos los conos. Cada vez que se invoca al generador este produce una morfología generada aleatoriamente del tipo descripto, en la cual los valores para cada cono se asignan basándose en dos rangos especificados por el usuario: $H_i \in$ [ $Hbase$, $Hbase + Hrango$] y $R_i \in$ [ $Rbase$, $Rbase + Rrango$].

## 3.1. Cambios Permitidos por el Generador

Una vez definida la morfología estática deseada, el usuario puede efectuar distintos tipos de cambios sobre ella, de todos ellos en este trabajo sólo se considera al cambio en la ubicación del cono que contiene al valor óptimo.

La dinámica de los cambios permite controlar cómo cambian las características topológicas del *landscape*. El generador DF1 propone controlar la dinámica a través del uso de la función logística que es una función no lineal, unidimensional que genera bifurcaciones progresivamente más complejas y que se define: $Y_i = A * Y(i-1) * (1 - Y(i-1))$, donde A es una constante e $Y_i$ es el valor de la función en la iteración $i$. A medida que la constante A se incrementa, de la función emergen comportamientos dinámicos más complicados. Los valores producidos por Y los usa DF1 sobre cada iteración para seleccionar el tamaño de paso para la porción dinámica del *landscape*. Se liga a la función un valor específico de A para cada característica topológica. Las dinámicas complejas se obtienen con sólo especificar:

1. El número de conos que se moverán en el *landscape*.
2. Si el cambio será aplicado a la altura de los conos, a su pendiente, a una coordenada particular y/o a cualquier combinación de estas características.

El valor particular de A, escogido para cada una de las características seleccionadas, especifica si el movimiento será pequeño, largo o caótico. Se require que A esté entre 1 y 4, los cambios más suaves se producen con A cercano a 1 y los drásticos o caóticos con A mayor que 3.5.

# 4. Algoritmos Evolutivos Propuestos

En este trabajo se comparan 6 AE's que implementan las distintas técnicas de mantenimiento de diversidad y combinaciones de ellas: dos operadores de macromutación y *fitness sharing*. Los 6 AE's comparten tanto la representación de las soluciones como los operadores convencionales, estas dos características de describen a continuación:

## 4.1. Representación

La población $P$ está compuesta por un número constante $N$ de cromosomas, $N$ depende de la dimensionalidad de la función que se esté optimizando. Cada individuo consiste de un único cromosoma, donde cada gen es un valor real en el intervalo [-1.0, 1.0] representando una coordenada en el espacio de búsqueda. El i-ésimo individuo en la población $P$, está representado por el cromosoma:

$$P^i =< x_{i1}, x_{i2}, \ldots, x_{ir} >$$

donde $x_{ij}$ es la $j$-ésima coordenada del $i$-ésimo individuo, con $j$=1,...,$r$ y $r$ es la longitud del cromosoma la cual está determinada por la dimensionalidad del problema.

## 4.2. Operadores Convencionales

*Selección*: para seleccionar a los padres que formarán parte del conjunto de apareamiento se empleó la selección por torneo binario.
*Recombinación*: para el intercambio de material genético se utilizó el *crossover* en un punto, con una probabilidad de aplicación $P_{Recomb}$.
*Mutación*: a cada gen, con una probabilidad $P_{Mut}$ se le aplica la mutación *Gaussiana* [1]. Esta se define como: $x'_{ji} = x_{ji} + \gamma * N(0,1)$, donde $x_{ji}$ es la $i$-ésima componente del individuo $j$-ésimo, $N(0,1)$ es una variable aleatoria *Gaussiana* con media 0 y desviación 1 y $\gamma$ es un factor de escala.

## 4.3. Descripción de los AE's

Los AE's constan de una función *F_Cambia* la cual es responsable de detectar si debe ocurrir un cambio en la función dinámica. Los cambios se generan en intervalos constantes de generaciones, así, dicha función sólo verifica si la generación actual correponde a una en la cual la función dinámica debe ser modificada. A través de una función denominada *Ubicación_Cambia*, se modifica la ubicación del cono que contiene al valor óptimo. Los AE's usan una función, *Ocurrió_un_Cambio*, para testear si el *landscape* ha sido modificado, en cuyo caso se toman las acciones pertinentes: mantener coherencia entre el *fitness* de los individuos y la función a optimizar y la inserción de los *random inmigrantes*, si así correspondiere.

# 5. Descripción de los Experimentos

El seteo de los paramétros para todos los EA's utilizados y las características de las funciones de prueba se describen a continuación.

## 5.1. Paramétros del Algoritmo Evolutivo

Los parámetros para los AE's son los mismos para todos los experimentos en todos los escenarios, y fueron los mejores de una serie de pruebas iniciales. El tamaño de la población fue de 100 individuos. El valor de $\sigma_{share}$ fue 0.05. Las probabilidades de recombinación $P_{Recomb}$ y mutación $P_{Mut}$ se fijaron en 0.25 y 0.5, respectivamente. Para el operador de recrudescencia, $P_{Recru}$ fue 0.2 y las probabilidades de recombinación y mutación se elevaron a 0.5 y 0.8, respectivamente. Para el operador de mutación $\gamma$=0.25. Los porcentajes de *Random Inmigrantes*

Cuadro 1: Datos de las Funciones de Prueba

| Función | Hbase | Hrango | Rbase | Rrango | MaxH | MaxR |
|---|---|---|---|---|---|---|
| F1 | 60.0 | 10.0 | 50.0 | 10.0 | 80.0 | 80.0 |
| F2 | 9.0 | 1.0 | 8.0 | 4.0 | 15.0 | 25.0 |
| F3 | 5.0 | 0.0 | 4.0 | 0.0 | 10.0 | 17.0 |

Cuadro 2: Funciones utilizadas en los Experimentos

| Función | #d-#c | #d-#c | #d-#c |
|---|---|---|---|
| F1 | 2-5 | 2-10 | 2-20 |
| F2 | 2-5 | 2-10 | 2-20 |
| F3 | 2-5 | 2-10 | 2-20 |

fueron 10 %, 30 % y 50 % del tamaño de la población. Los individuos ha ser reemplazados por los *random inmigrantes* se seleccionan aleatoriamente con igual probabilidad. Un número de experimentos ha sido diseñado que difieren en la función de prueba y la severidad del cambio que sufrirá. Para cada uno de estos experimentos se efectuaron 30 corridas con distintas poblaciones iniciales.

## 5.2. Paramétros del Generador de Problemas de Prueba

El Cuadro 1 muestra los parámetros generales para cada una de las funciones de prueba, altura $H$, pendiente $R$, valores máximos para la altura y pendiente MaxH y MaxR, respectivamente. Para F1 y F2, independientemente del núnero de conos que posean, uno de los conos es inicializado como máximo global empleando para ello los valores de MaxH y MaxR correspondientes a cada función, el resto de los conos se generan como se indicó en la Sección 3 dentro de los rango de valores permitidos para cada función con respecto a la altura y pendiente. F3 además de cumplir con estas restricciones posee un cono con altura igual a 7.0 y pendiente igual a 7.0. La Figura 1 muestra las funciones F1, F2 y F3.

Los parámetros para la severidad de los cambios fueron los siguientes A=1.5 y $Cstepscale$=0.99 para los cambios largos y A=3.8 y $Cstepscale$=0.5 para los cambios caóticos.

La constante A es utilizada por la función logística para determinar la severidad del cambio. Los valores seleccionados para A crean una grado de severidad el cual produce cambios largos y caóticos [15]. La constante *Cstepscale* se emplea para mover cada coordenada sobre el rango especificado por el usuario. Los cambios pequeños y frecuentes no fueron considerados puesto que las autoras coinciden con Branke en que dichos tipos de cambios deben ser contemplados a través de la creación de soluciones robustas mientras que los cambios largos y poco frecuentes deben ser manejados por medio de la adaptación [3]. Se trabajó sobre distintas funciones cuyas características con respecto a la multimodalidad se indican en el Cuadro 2, aquí "#d-#c"indica número de dimensiones y número de conos, respectivamente. Para cada una de estas funciones se testeó que ninguno de los conos estuviera cubierto por otro cono. Los cambios se provocaron cada 10, 50 y 70 generaciones para todas las funciones y escenarios. El objetivo principal fue

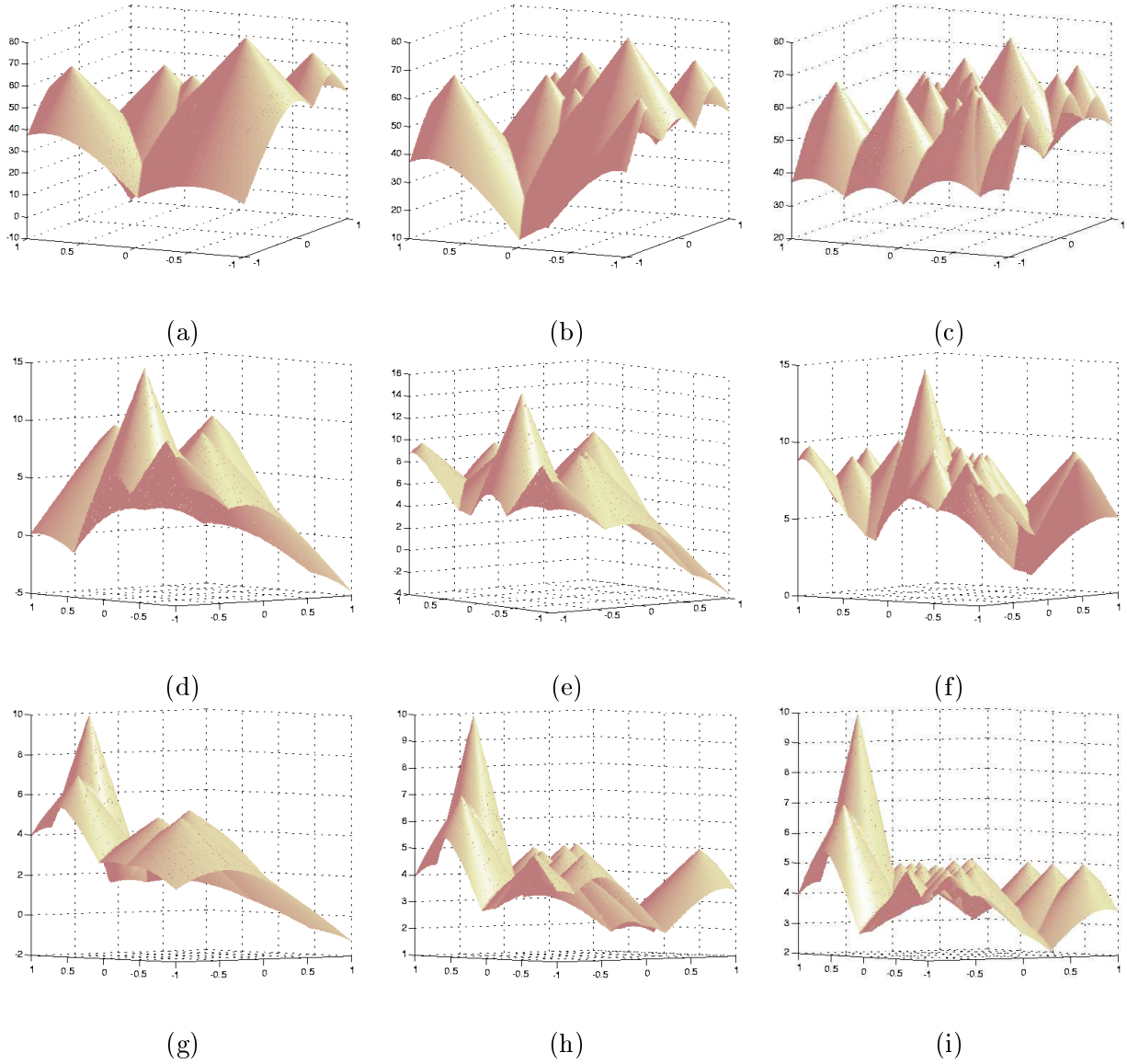

Figura 1: (a)F1 - 5 Conos, (b)F1 - 10 Conos, (c)F1 - 20 Conos, (d)F2 - 5 Conos, (e)F2 - 10 Conos, (f)F2 - 20 Conos, (g)F3 - 5 Conos, (h)F3 - 10 Conos, (i)F3 - 20 Conos

determinar si los algoritmos tienen éxito en seguir la pista del cono que contiene al valor óptimo conforme cambia el *landscape*. Para todos los experimentos el número de cambios se fijó en 20, por lo tanto si los cambios se producen cada 10 generaciones el algoritmo correrá por 200 generaciones.

## 5.3. Métricas de Performance

Para analizar y comparar la *performance* de los AE's en *landscapes* dinámicos multimodales, se usaron dos tipos de métricas: una que toma en cuenta la calidad de las soluciones encontradas y otra, que permite conocer a qué cantidad de cambios lograron adaptarse los AE's estudiados. *Precisión*: Esta métrica, definida por Trojanoswky [18], fue diseñada específicamente para ambientes no estacionarios y mide la diferencia promedio entre el mejor individuo de la población

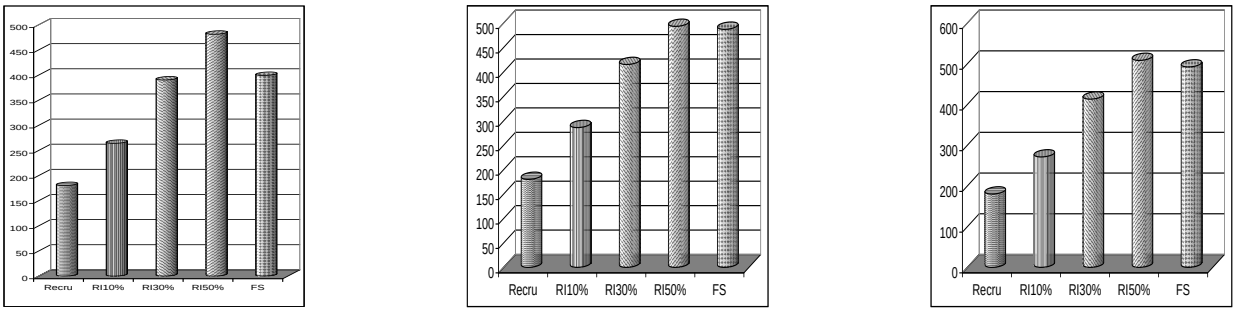


Figura 2: F3 - 10 Conos - Cambios Largos. Izquierda: Intervalo entre Cambios 10 Generaciones. Medio: Intervalo entre Cambios 50 Generaciones. Derecha: Intervalo entre Cambios 70 Generaciones

en la generación "justo antes del cambio" y el valor óptimo. Se define como:

$$Precisión = \frac{1}{k}\sum_{i=1}^{k}(opt - me_i)$$

donde $k$ es el número de cambios que sufrirá la función, *opt* es el valor medio de los valores óptimos en cada cambio y $me_i$ es el mejor valor encontrado antes del $i$-ésimo cambio. De la definición se deduce que pequeños valores para esta métrica indican mejores resultados. Un valor 0 significa que el mejor individuo de la población se encuentra en el óptimo global antes de que la función se modifique.

*Cantidad de cambios*

Para asegurar que los AE's llevan la pista del cono que contiene al valor óptimo y no se malinterpreten los resultados, para cada uno de los conos de las funciones de prueba el generador permite conocer su ubicación, altura y pendiente permitiendo determinar si el mejor individuo de la población pertenece al cono que contiene al valor óptimo. Si antes de que el *landscape* se modifique el mejor individuo de la población pertenece al cono que contiene al valor óptimo se considera que el algoritmo se ha adaptado a dicho cambio. Por lo tanto, en cada experimento se puede detectar un máximo de 600 cambios (20 cambios por corrida por 30 corridas).

# 6. Resultados

En primera instancia, sólo se probaron 3 AE's sobre el conjunto de funciones de prueba: cada uno de ellos implementó, en forma aislada, una de las técnicas de mantenimiento de diversidad, estos AE's son:

*Recru*: Algoritmo Evolutivo + *Recrudescencia.*

*RI*: Algoritmo Evolutivo + *Random Inmigrantes.*

*FS*: Algoritmo Evolutivo + *Fitness Sharing.*

En la Figura 2 sólo se muestra la cantidad de cambios detectados para la función F3 (por ser la más difícil para los AE's) con 10 conos, cuando los cambios son largos, para cada uno de los AE's. Cada barra muestra la cantidad de cambios detectados por un algoritmo. Del análisis de los resultados mostrados en la Figura 2 se deduce que estos AE's, no presentan una buena *performance* dado que la cantidad de cambios a los cuales se adapatron oscila entre 190 y 500, considerándo estos resultados como no satisfactorios. Resultados similares se obtuvieron para F1 y F2. Es por ello que, se decidió usar distintas combinaciones de las técnicas de mantenimiento

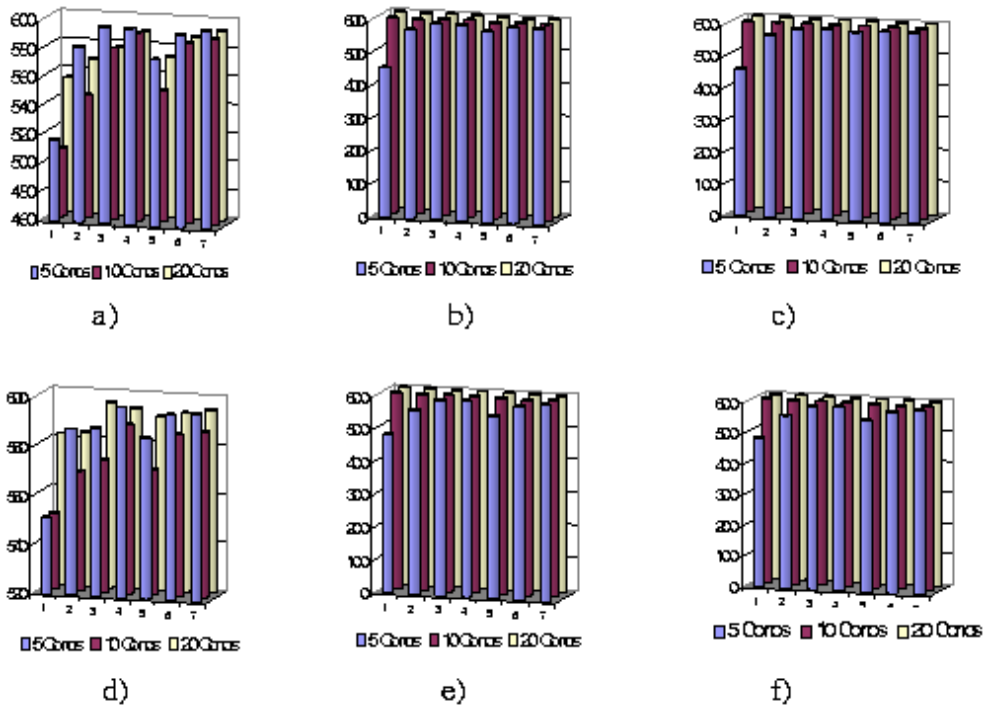


Figura 3: F2 - a)b)c) Cambios Largos - Intervalo entre Cambios 10, 50 y 70 Generaciones, respectivamente d)e)f) Cambios Caóticos - Intervalo 10, 50 y 70 Generaciones, respectivamente

de diversidad antes mencionadas para poder determinar el aporte que cada una realizaba al ir agregándose a otra. Estas combinaciones fueron:

*FSRecru*: Algoritmo Evolutivo + *Fitness Sharing* + *Recrudescencia*.

*FSRI*: Algoritmo Evolutivo + *Fitness Sharing* + *Random Inmigrantes*.

*FSRecruRI*: Algoritmo Evolutivo + *Fitness Sharing* + *Recrudescencia* + *Random Inmigrantes*.

Del análisis de los resultados de los distintos AE's, para el caso de la función de prueba F1, para los cambios largos y caóticos se deduce que la *performance* de los algoritmos es muy satisfactoria, detectando todos los cambios provocados. Cuando los cambios se producen cada 10 generaciones los valores de precisión oscilan entre 2.1 y 3.5. Para cambios cada 50 y 70 generaciones los valores de precisión mejoran oscilando entre 1.9 y 2.4. La buena *performance* de los AE's bajo F1 puede deberse a las características que ésta posee (ver Figura 1(a)(b)(c)), la distancia entre la altura del cono que contiene al valor óptimo con respecto a los demás conos es de 10 unidades, por lo cual, parece ser que los óptimos locales no desorientan, fuertemente, al proceso de búsqueda. Debido a que todos los AE's, para esta función, se adaptaron a todos los cambios, se consideró innecesario mostrar las gráficas correspondientes.

A continuación se analizan los resultados para las funciones F2 y F3 y las gráficas de resultados se muestran en las Figuras 3 y 4, respectivamente. Las Figuras 3 y 4 deben interpretarse de la siguiente manera: cada punto sobre el eje de la abscisa identifica a un tipo de algoritmo y los puntos de la ordenada la cantidad de cambios detectados por ese algoritmo. Así, 1 corresponde a *FSRecru*, 2, 3 y 4 a *FSRI* con un 10 %, 30 % y 50 % de *random inmigrantes*, respectivamente y 5, 6 y 7 a *FSRIRecru* con un 10 %, 30 % y 50 % de *random inmigrantes*, respectivamente.

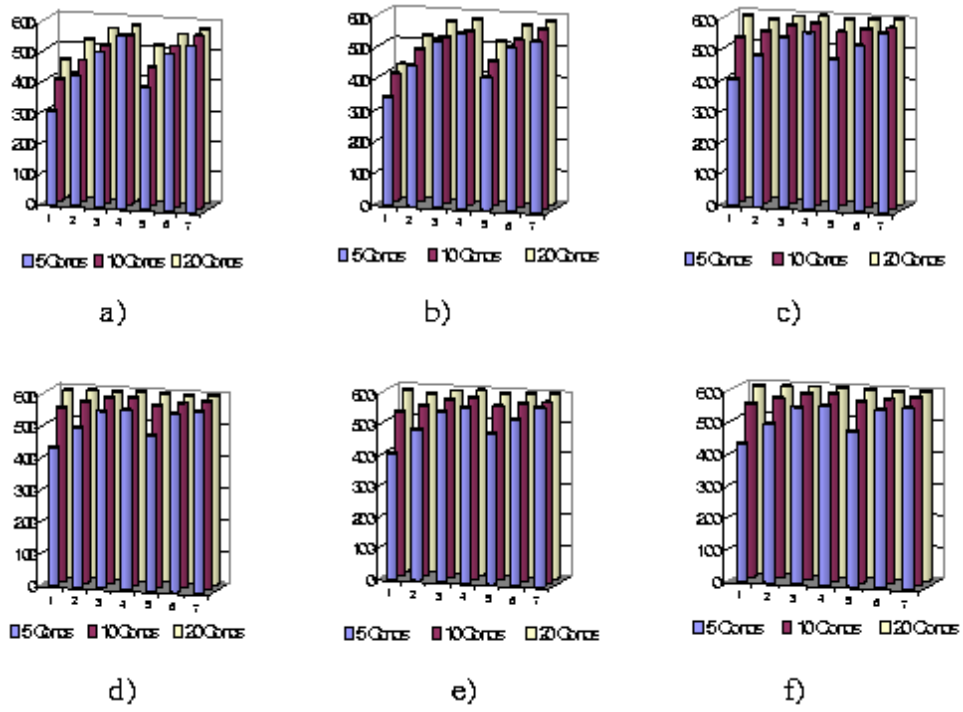


Figura 4: F3 - a)b)c) Cambios Largos - Intervalo 10, 50 y 70 Generaciones, respectivamente d)e)f) Cambios Caóticos - Intervalo 10, 50 y 70 Generaciones, respectivamente

Del análisis de los AE's para el caso de la función de prueba F2 para los cambios largos y caóticos se observa: Cuando los cambios se producen cada 10 generaciones, los mejores resultados se obtienen con *FSRI* con un 30 % y 50 % de *random inmigrantes*, siendo no significativa la diferencia de *performance* con *FSRIRecru*, salvo en el caso de 10 y 20 conos para cambios largos, con un valor de precisión que oscila entre 0.8 y 1.8, detectándose entre 508 y 599 de los cambios provocados. Para cambios cada 50 y 70 generaciones no existe una dificulad notable para adaptarse a los cambios provocados con un valor de precisión que oscila entre 0.6 y 0.8, detectándose entre 456 y 600 de los cambios provacados. Esta función parece ser más compleja que F1. Aquí, la cantidad de unidades de diferencia que hay entre el cono que contiene al valor óptimo y el resto es de 5 unidades, razón por lo cual, los óptimos locales estarían afectando en mayor medida a la *performance* de los AE's (ver Figura 1(d)(e)(f)).

Del análisis de los AE's para el caso de la función de prueba F3 para los cambios largos y caóticos se observa: Cuando los cambios se producen cada 10 generaciones, en general, los mejores resultados se obtienen con un 50 % de *random inmigrantes* de *FSRecruRI FSRI*, con un valor de precisión que oscila entre 0.7 y 0.8, detectándose entre 307 y 584 de los cambios provacados. Cuando los cambios se producen cada 50 y 70 generaciones, los mejores resultados se obtienen con 30 % y 50 % de *random inmigrantes* de *FSRecruRI* y *FSRI*, con un valor de precisión que oscila entre 0.5 y 0.8, detectándose entre 409 y 599 de los cambios provacados. Esta función resulta más compleja, para los AE's, que F1 y F2. El cono inicializado con altura=7.0 y pendiente=7.0, que es un óptimo local distinto del resto, desorienta la búsqueda, actuando en algunas corridas como un atractor para el mejor individuo de la población (ver Figura 1(g)(h)(i)).

# 7. Conclusiones

Los intervalos entre cambios, como es esperable, afectan la *performance* de los algoritmos. A mayor cantidad de generaciones entre cambios menor el valor de la precisión y mayor la cantidad de cambios detectados por todos los AE's estudiados. Esto tiene sentido dado que, los algoritmos tienen más tiempo para evolucionar la población. En general la *performance* de *FS* y *FSRecru* es muy similar y es superada por cualquier algoritmo que incluya *random inmigrantes*. El incremento en el porcentaje de *random inmigrantes* aumenta la cantidad de cambios detectados y la cantidad de corridas donde se detectan todos los cambios, bajando o manteniendo el valor de precisión. Parece ser que los algorimos son más sensibles a la cantidad de generaciones entre cambios que a la cantidad de conos en el *landscape*. Los valores de precisión más elevados y la menor cantidad de cambios detectados se producen cuando los cambios se generan cada 10 generaciones independientemente de la cantidad de conos de las funciones de prueba.

Se puede concluir que para el conjunto de funciones de prueba estudiadas el operador de *recrudescencia* no mejora la *performance* del algoritmo que implementa *fitness sharing* y no aporta lo suficiente cuando se combina con los algoritmos que incluyen *random inmigrantes*. La *performance* de *FS* es altamente superada por los algoritmos que además incluyen *random inmigrantes* al detectarse un cambio. Los algoritmos requieren un alto grado de diversidad cuando el *landscape* cambia, es posible que la mala *performance* de *FS* sea causada por el valor escogido para $\sigma_{share}$ único para todas las funciones. Notar que *FS* como única técnica de mantenimiento de diversidad no produce resultados satisfactorios, pero la incorporación de *random inmigrantes* únicamente, no logra igualar los resultados obtenidos con la combinación de estas dos técnicas (ver Figuras 2, 3, 4). Esto se debe a que la función de *fitness sharing* busca explorar distintos conos en forma simultanéa, sin importar la calidad de las soluciones, y los *random inmigrantes* aportan la diversidad necesaria para crear por un lado nuevos nichos y encontrar por la aplicación de los operadores de recombinación y mutación dentro de cada cono soluciones de calidad superior.

El objetivo ahora es por una parte determinar si estos algoritmos, *FSRI* y *FSRecruRI*, son aptos para trabajar con funciones de mayor dimensionalidad, debido a que el número de conos parece en primer instancia no afectar considerablemente la *performance* en las funciones de 2 dimensiones y, por otra parte buscar funciones incluso en 2 dimensiones con otras morfologías de *landscapes*. Además, los AE's propuestos trabajan sin incluir técnicas de elitismo, en un trabajo futuro se buscará incorporar dichos mecanismos.

# 8. Reconocimientos

En memoria del Dr. Raúl H. Gallard por quien sentimos el mayor respeto y gratitud.

# Referencias

[1] P. Angeline. Tracking extrema in dynamic environments. In J. R. McDonnell R. G. Reynolds and R. Eberhart, editors, *Proceeding of the 6th International Conference on Evolutionary Programming*, pages 335–345, 1997.

[2] T. Bäck. On the behavior of evolutionary algorithms in dynamic fitness landscapes. In *Proceeding of the IEEE International Conference on Evolutionary Algorithms*, pages 446–451, IEEE Service Centre, 1998.

[3] J. Branke. *Evolutionary Optimization in Dynamic Environments.* Kluwer Academic Publishers, 2002.

[4] H. G. Cobb. An investigation into the use of hypernutation as an adaptive operator in genetic algorithms having continuous, time dependent non-stationary environments. Technical Report 6760, Naval Research Laboratory, USA, 1990.

[5] K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In CA: Morgan Kaufmann J. D. Schaffer, San Mateo, editor, *Proceeding 3rd International Conference on Genetic Algorithms*, pages 42–50, 1989.

[6] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum Associates, 1987.

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

[8] J. Grefenstette. Genetic algorithms for changing environments, 1992. In Proceedings of Second Parallel Problem Solving From Nature (PPSN-2), Brussels, 28-30 September, pages 137-144, 1992.

[9] J. Holland. Adaptation in natural and artificial systems, 1975. Ann Arbor, MI: University of Michigan Press.

[10] E. Hart. J. Lewis and G. Ritchie. A comparision of dominance mechanism and simple mutation in non-stationary problems. In Morgan Kaufmann, editor, *Proceeding 7th International Conference on Genetic Algorithms*, pages 138–148, 1997.

[11] Z. Michalewicz K. Trojanoswky. Searching for optima in non-stationary environments. In *Proceeding of the Congress on Evolutionary Computation*, pages 1843–1850, Washington DC, USA, IEEE Service Centre, 1999.

[12] H. Kwasnicka. Redundancy of genotypes as the way for some advanced operators in evolutionary algortihms - simulaton study. In Mumbai, editor, *VIVEK A Quarterly in Artificial Intelligence*, pages 2–11, 1997.

[13] W. Liles and K. De Jong. The usefulness of tag bits in changing environments. In *Proceeding of the Congress on Evolutionary Computation*, pages 2054–2060, Washington DC, USA, IEEE Service Centre, 1999.

[14] N. S. Mori. Adaptation to changing environments by means of the memory based thermodynamic genetic algorithm. In Morgan Kaufmann, editor, *Proceeding 7th International Conference on Genetic Algorithms*, pages 299–306, 1997.

[15] R. Morrison. *Designing Evolutionary algorithms for Dynamic Environments.* PhD thesis, George Mason University, USA, 2002.

[16] R. Morrison and K. De Jong. A test problem generator for non-stationary environments. In *Proceeding of the Congress on Evolutionary Computation*, pages 2047–2053, Washington DC, USA, IEEE Service Centre, 1999.

[17] B. Sareni and L. Krähenbühl. Fitness sharing and niching methods revisited. In *IEEE Transactions on Evolutionary Computation*, pages 97–105, 1998.

[18] K. Trojanoswky. *Evolutionary Algorithms with Redundant Genetic Material for Non-Stationary Environments.* PhD thesis, Institute of Computer Science, Warsaw, Poland, May 1994.

[19] K. Wicker and N. Weicker. On evolutionary strategy optimization in dynamic environments. In *Proceeding of the Congress on Evolutionary Computation*, pages 2039–2046, Washington DC, USA, IEEE Service Centre, 1999.

[20] C. O. Wile. *Evolutionary Dynamics in Time-Dependent Environments.* PhD thesis, Institut für Neuroinformatik, Ruhr-Universität, Bochum, Germany, 1999.

[21] T.Ñanayakkara with et al. Adaptive optimization in a class of dynamic environments using an evolutionary approach, 1999.