ANELAR. Arreglos Neuronales Evolutivos de Longitud Adaptable Reducida

Leonardo C. Corbalan[1], Germán L. Osella Massa[2], Laura Lanzarini[3] y Armando E. De Giusti[4]

III - LIDI - Instituto de Investigación en Informática LIDI
Facultad de Informática, Universidad Nacional de La Plata.

RESUMEN

Los arreglos neuronales evolutivos (ANE) han demostrado ser capaces de aprender comportamiento complejo y han sido aplicados satisfactoriamente en la resolución de problemas en áreas tales como robótica y control de procesos. A diferencia de los métodos convencionales, basados en una única red neuronal, los ANE están conformados por un conjunto de redes que se organizan en forma de arreglo.

Por otro lado, los arreglos neuronales evolutivos de longitud adaptable (ANELA) poseen las mismas características de los ANE y además son capaces de ajustar automáticamente su longitud durante el proceso evolutivo. Sin embargo, si bien han demostrado ser muy eficientes, no están diseñados para minimizar el tamaño de la estructura sino para maximizar su rendimiento.

En este artículo se presenta ANELAR que, si bien comparte la definición de la arquitectura con ANELA, mejora el método de aprendizaje para obtener arreglos neuronales de longitud reducida con la menor pérdida de eficiencia posible.

Se ha aplicado ANELA y ANELAR a problemas de evasión de obstáculos y recolección de objetos, evidenciando su superioridad con respecto a los métodos tradicionales que manejan poblaciones simples de redes neuronales.

Finalmente se presentan las conclusiones y se plantean algunas líneas de trabajo futuras.

Keywords: Redes Neuronales Evolutivas. Arreglos Neuronales Evolutivos. Aprendizaje. Algoritmos Genéticos. Subpoblaciones. Migraciones.

Workshop al que está dirigido: Workshop de Agentes y Sistemas Inteligentes (WASI)

1. Introducción

La aplicación de algoritmos evolutivos en el área de las Redes Neuronales Artificiales ha sido utilizada para conseguir los pesos de conexión, el diseño de la arquitectura, el valor de los parámetros iniciales, las reglas de aprendizaje, etc. [12].

Freeman y Skapura [7] argumentaron la necesidad de aprender a combinar redes neuronales pequeñas, poniéndolas bajo el control de otras redes para resolver el problema del escalado. Xin Yao y Yong Liu [11] estudiaron los beneficios de utilizar como solución la población completa de redes neuronales de la última generación y, más recientemente, Bruce y Miikkulainen [1], demostraron que todas las redes neuronales de una población, combinado con una técnica efectiva de especialización, pueden responder mejor de manera colectiva que cualquiera de ellas individualmente.

[1] corbalan@lidi.info.unlp.edu.ar - Becario de Iniciación UNLP
[2] gosella@lidi.info.unlp.edu.ar - Becario de Iniciación UNLP
[3] laural@lidi.info.unlp.edu.ar - Profesora Titular UNLP
[4] degiusti@lidi.info.unlp.edu.ar - Profesor Titular UNLP. Investigador Principal del CONICET

Bajo el lineamiento general de explorar soluciones basadas en múltiples redes neuronales, se presentaron en [4] los arreglos neuronales evolutivos (ANE). Este método demostró ser más eficiente que otras estrategias neuroevolutivas al combinar los beneficios de la evolución incremental con la potencia de varias redes neuronales integradas en un único controlador. Una nueva versión del método [5] consiguió erradicar la necesidad de definir explícitamente los subobjetivos en cada etapa, herencia de la evolución incremental [9], lo que dificultaba su generalización.

Los arreglos neuronales evolutivos de longitud adaptable (ANELA) [6] han demostrado ser, hasta el momento, el método más potente de esta familia de algoritmos. La capacidad para ajustar automáticamente su longitud durante el proceso evolutivo lo convierte en un método sumamente eficiente para hallar soluciones cercanas a la óptima en pocas generaciones. Debido a que no es necesario determinar a priori la longitud de los arreglos, este método resulta fácilmente adaptable a distintos tipos de problemas. Se comienza con arreglos de longitud 1, que van creciendo con la incorporación de una nueva componente cada vez que el progreso de la evolución se detiene.

No obstante, en ocasiones es importante minimizar la longitud de los arreglos para economizar espacio de almacenamiento. Si bien, a medida que avanzan las generaciones, ANELA es capaz de reducir la cantidad de componentes de los arreglos evolucionados, el método no está diseñado para minimizar la longitud de los mismos sino para maximizar su rendimiento.

En este artículo se propone un nuevo método de evolución, tendiente a conseguir arreglos de longitud reducida con la menor pérdida de eficiencia posible.

Las mediciones del método propuesto se realizaron sobre el problema de evasión de obstáculos y recolección de objetos que se describe más adelante.

2. Controlador Neuronal de ANELA y ANELAR

ANELA y ANELAR permiten obtener controladores formados por arreglos neuronales que resuelven problemas de control de procesos en forma más eficiente que las soluciones tradicionales. Las redes integrantes provenientes de la evolución de distintas subpoblaciones, aprenden a especializarse en distintas subtareas del proceso total a controlar. Así, del accionar coordinado de estas redes surge la resolución de un problema complejo.

2.1. Organización interna del controlador

El controlador está formado por un arreglo neuronal, una *n*-upla de redes neuronales de la forma $C=(rn_1,rn_2,..,rn_n)$. Al igual que una red, un arreglo neuronal acepta una entrada de datos, se evalúa y produce la salida correspondiente. En cada instante t, la salida será provista por alguna de las redes integrantes. Sólo una de ellas permanece activa a la vez, por lo tanto el tiempo de procesamiento del controlador será igual al de la red neuronal que se evalúa en dicho instante.

2.2. Funcionamiento del controlador

Al comienzo del proceso, la única red activa del controlador es rn_1 que resuelve todas las entradas hasta su auto-desactivación. Una vez desactivada, el control pasa a rn_2 que continúa evaluándose hasta que "decida" desactivarse, pasando el control a rn_3. Esta delegación de control prosigue hasta que eventualmente se activa rn_n, última componente,

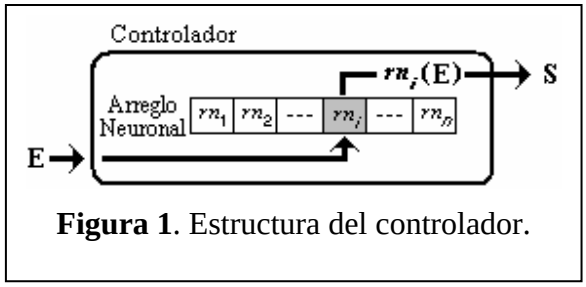

Figura 1. Estructura del controlador.

quien permanece en dicho estado hasta la finalización del proceso. En todos los casos, la desactivación de las redes se efectiviza después de producida la salida del controlador, dejando activa la próxima red neuronal para el siguiente instante de evaluación.

Las redes del arreglo cuentan con una neurona de salida extra a la cantidad definida por el problema a resolver. El valor de dicha neurona no forma parte de la salida del controlador y se utiliza para determinar si la red evaluada debe desactivarse.

Sea $d_i(t)$ el valor de la neurona extra de salida de la red activa rn_i en el instante de evaluación t. La red neuronal rn_i seguirá activa para el instante t+1 si se cumple alguna de las dos siguientes condiciones:

1) $|d_i(t) - d_i(t-1)| < c$, donde c es una constante definida a priori.

2) $i = n$, siendo n la longitud del arreglo.

Por el contrario, si no se cumple ninguna de las condiciones anteriores, rn_{i+1} se activará en el instante t+1.

Así, el mecanismo de activación/desactivación de las redes integrantes queda definido en función de la información suministrada al arreglo en dos instantes de tiempo consecutivos. La sensibilidad de este mecanismo puede ajustarse por la elección adecuada de la constante c que aparece en la primera condición. Para un rango de salida de la neurona extra entre 0 y 1, se han hallado buenos resultados con valores para c comprendidos entre 0.01 y 0.1. Como ocurre con otros parámetros en la teoría de redes neuronales, la experimentación previa puede ser la mejor opción para la determinación adecuada de este valor.

3. ANELA

3.1 Algoritmo Evolutivo

Se distinguen dos fases en el algoritmo evolutivo:

1) *Exploración:* Comprende las primeras generaciones de la evolución. Comenzando por arreglos de una única componente, el algoritmo irá aproximándose a la solución del problema al mismo tiempo que la longitud del arreglo va incrementándose con la incorporación de nuevas redes neuronales.

2) *Explotación*: Una vez que la solución al problema está lo suficientemente próxima (fitness suficientemente alto) la longitud del arreglo queda fija y la solución es alcanzada por medio del ajuste u optimización de las componentes.

Si la longitud del arreglo establecida durante la exploración es n, se obtendrán controladores de la forma $C=(rn_1,rn_2,..,rn_n)$ por evolución concurrente de n subpoblaciones de redes neuronales, una para cada componente rn_i del controlador.

En ambas fases, exploración y explotación, las poblaciones P_i de redes neuronales son evolucionadas de a una por vez, durante períodos de longitud variable, medidos en cantidad de generaciones, y determinados en función del mejor fitness obtenido por generación. A medida que avance el proceso, las evoluciones de las poblaciones se sucederán ordenadamente en una cola circular. Cada P_i eventualmente pasará por varios períodos de evolución.

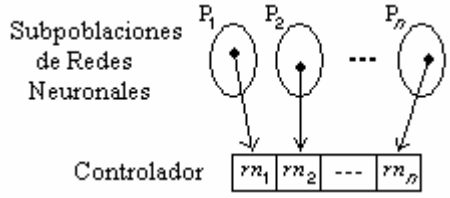

Figura 2. Subpoblaciones de redes neuronales

En la fase de exploración se somete a evolución siempre la última componente del arreglo en formación. Es una etapa de crecimiento de la longitud de los arreglos neuronales. La condición `finExploracion` (ver fig. 3) se relaciona con la apreciación, algo vaga, de "fitness suficientemente alto" ya mencionada. Esta condición debe definirse en función de la implementación particular que se haga de un problema dado. A modo de ejemplo, en el problema de evasión de obstáculos, búsqueda y recolección de objetos presentado en este artículo, `finExploracion` toma valor verdadero cuando se alcanza un fitness igual al 75% del valor máximo alcanzable por un controlador óptimo.

La condición `fitness_Estacionario` será verdadera cuando no se consiga mejorar el fitness a lo largo de cierta cantidad de generaciones. Esta cantidad es un parámetro del algoritmo ANELA. En el presente artículo se muestran resultados que sugieren utilizar un número pequeño para este parámetro (1 ó 2 a lo sumo) si se pretende obtener el mayor rendimiento posible. Por el contrario, un valor alto (20 ó 30) arrojará como resultado arreglos de menor longitud pero también, menor rendimiento.

La condición de `terminación` se hace verdadera al alcanzar una determinada generación o bien un valor de fitness fijados previamente.

Los controladores se construyen con cada una de las redes de la población sometida a evolución y la mejor red ranqueada según su fitness, en cada una de las restantes poblaciones (ver fig. 4). De esta forma, los controladores evaluados difieren entre sí, sólo en la i-ésima componente.

$$P_c(i) = \{(rn_1, rn_2, ...rn_i, ...rn_n): rn_i \in P_i \text{ y } rn_J = mejor(P_J) \text{ para } i \neq J\}$$

De esta manera, en todo momento, existe una población P_i que evoluciona para optimizar la integración de la *i*-ésima componente del controlador con las restantes redes del mismo.

```
Begin Program
// exploración
n:=0
while not finExploracion
  n:= n + 1
  Generar aleatoriamente la subpob. Pn
  Repeat
    pasoDeEvolucion(Pn)
  Until fitness_Estacionario
End while
//ahora n guarda la long. del arreglo

// explotación
i:=0
While not terminación
  i:= i mod n + 1
  Repeat
    PasoDeEvolucion(Pi)
  Until fitness_Estacionario
End while

End Program.

//rutina PasoDeEvolucion
PasoDeEvolucion(Pi)
Begin
 Armar pob. de controladores Pc(i)
 Evaluar cada controlador de Pc(i)
 Asignar fitness a redes de Pi
 If not fitness_estacionario then
   Pi:= next_generation(Pi)
End
```

Figura 3. Pseudocódigo ANELA

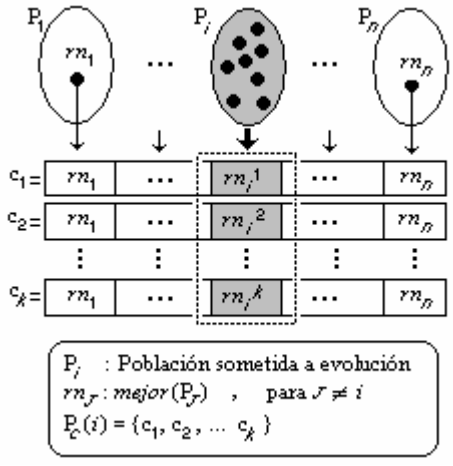

Figura 4. Evolución de las subpoblaciones

El fitness alcanzado por cada controlador durante la evaluación, es asignado a la *i*-ésima componente del mismo (individuos de la población P_i).

Para obtener la próxima generación en la población de redes P_i, no se asume ningún algoritmo evolutivo específico. Puede utilizarse desde un algoritmo genético simple como el presentado por Goldberg [8] hasta otros más sofisticados propios del paradigma de la neuroevolución. En particular, en el presente trabajo se ha empleado SANE [10].

No se asume ninguna restricción sobre los parámetros de la red sometidos a evolución (pesos de conexión, arquitectura, función de transferencia, etc.). Puede hallarse un conjunto de variantes en [12] donde se citan varias investigaciones en neuroevolución incluyendo hibridaciones con algoritmos de aprendizaje tradicionales.

Debido a que la evaluación de un arreglo neuronal utiliza los mismos recursos de cómputo que una única red, y a que las subpoblaciones se evolucionan por turnos de a una a la vez, la carga computacional de ANELA es similar a la de cualquier método convencional.

3.2. Más sobre las fases de exploración y explotación

Durante la fase de exploración del algoritmo, es importante que los controladores parcialmente construidos tiendan a desactivar hasta la última de sus redes neuronales durante la evaluación de los mismos. Así, la futura siguiente componente tendrá chance de ejecutarse. Para ello, se penaliza con un fitness bajo a aquellos controladores que no hayan desactivado todas sus redes en el último instante de evaluación (sólo durante la exploración) y se extiende el tiempo de evaluación cada vez que el algoritmo agrega una componente a los arreglos neuronales.

Si bien ANELA no se propone obtener arreglos neuronales de longitud reducida, en la fase de explotación, la optimización de las componentes puede dar como resultado la disminución de la longitud del arreglo. No es infrecuente que mientras el algoritmo ajusta la *i*-ésima componente, sometiendo a evolución la subpoblación P_i, se encuentre algún controlador que resuelve el problema utilizando las primeras *k* redes neuronales con $i \leq k < n$, siendo *n* la longitud actual de los arreglos. En este caso se eliminan las subpoblaciones P_J para $k < J \leq n$ y se truncan los arreglos estableciendo su nueva longitud en *k*.

4. ANELAR

En la fase de exploración de ANELA se tiende a aumentar rápidamente la longitud de los arreglos para obtener así el mejor desempeño posible. La capacidad de ANELA para disminuir la longitud de los arreglos en la fase de explotación no es suficiente para hallar arreglos de longitud reducida. Ello se debe a que en dicha fase las componentes rn_i del arreglo de longitud *n* se someten por turnos a evolución para optimizar su integración con las restantes rn_j ($i<>j$) del controlador, no existiendo ninguna presión para que rn_i , con $i < n$, complete por sí misma la tarea a resolver y reduzca así la longitud del arreglo. Si la evolución consigue una rn_i que es potencialmente buena como última componente, seguramente no recibirá un fitness alto si las redes rn_k ($k>i$) del arreglo al que pertenece no se adecuan a la nueva situación. Ello habitualmente no sucede pues la única componente sometida a evolución en ese momento es rn_i y quizás ya no exista en la próxima generación.

Por el contrario, en la fase de exploración cada rn_i sometida a evolución es siempre la última componente del arreglo en formación. Por lo tanto, el valor de fitness obtenido por la red es directamente proporcional a su capacidad para completar la tarea. Si se consigue un fitness suficientemente alto, se fija la longitud del arreglo pasando a la fase de explotación.

Por estas razones, ANELAR modifica la fase de exploración de ANELA para obtener arreglos de menor longitud. El resto del algoritmo es idéntico a ANELA. A continuación se describe su implementación.

4.1. Fase de exploración en ANELAR

Si durante la fase de exploración del método ANELA, la condición `fitness_Estacionario` se hace verdadera cuando la cantidad de "generaciones estancadas" es un número alto, los controladores obtenidos resultarán indudablemente de menor longitud. No obstante, un valor demasiado alto hará que el algoritmo no encuentre la solución buscada, a menos que la evolución se prolongue por un gran número de generaciones, y uno demasiado bajo tenderá a encontrar arreglos de longitud elevada. Determinar el valor óptimo para este parámetro es sólo posible por medio de la experimentación, y dependerá del problema particular a resolver.

ANELAR propone hacer variar la cantidad de generaciones estancadas necesarias para hacer verdadera la condición `fitness_Estacionario` a lo largo de la fase de exploración. Además utiliza un mecanismo de migraciones tendiente a enriquecer las poblaciones de subíndices inferiores con copias de los mejores individuos de las poblaciones de subíndices superiores. Ello se debe a que un buen individuo de la población P_i, con $i>1$, que se desempeña como i-ésima componente de un arreglo neuronal, quizá también sea bueno desempeñándose en la posición j, con $j<i$. Si esto es cierto, una funcionalidad generada en cierta componente podrá "moverse" hacia una posición anterior, facilitando la obtención de arreglos de menor longitud.

Sea e la cantidad de generaciones estancadas necesarias para hacer verdadera la condición `fitness_Estacionario`. La fase de exploración comienza con $e=1$. Se fija una longitud máxima actual de crecimiento de los arreglos en un valor relativamente pequeño ($n=2$ ó $n=3$) y se procede con la exploración hasta alcanzar la longitud n. Si la condición `finExploracion` no es verdadera se incrementa en 1 el valor de e, se realizan las migraciones entre las poblaciones y se reinicia el ciclo de exploración. Esta estrategia se repite hasta que se hace verdadera la condición `finExploracion` o bien e alcanza un límite máximo prefijado. Llegado este punto, si `finExploracion` es falso, se incrementa en 1 el valor de n y se comienza un nuevo ciclo de exploración con $e=1$.

Algoritmo. Fase exploración

```
n=Longitud_Maxima_Actualmente_Permitida
i=1
repeat
   e=1
   repeat //ciclo de exploración
       Proceder con la exploración evolucionando las poblaciones Pi, Pi+1, ... Pn  (*)
       Realizar migraciones entre las poblaciones Pi, Pi+1, ... Pn
       e = e + 1
   until (finExploracion) OR (e > limite)
   Si NOT (finExploracion),
       n=n+1, i=i+1  //se corre la ventana de poblaciones a evolucionar
until (finExploracion)
//Fin fase de exploración

El punto (*) puede refinarse de la siguiente manera:

k=i
repeat
  Si no existe, generar aleatoriamente la población Pk
  repeat
     pasoDeEvolucion(Pk)
  until (cant. de gen. estancadas >= e)
  k=k+1
until (k > n)
```

Obsérvese que se crea una ventana $[i,n]$ de poblaciones que se someten, por turnos, a evolución en cada ciclo de exploración. Dicho intervalo se mueve una unidad hacia la derecha ($i=i+1$, $n=n+1$) cuando se necesita permitir la incorporación de una nueva componente en los arreglos neuronales.

La razón por la cual se desplaza el intervalo descansa en la necesidad de ir excluyendo del proceso de exploración a las componentes de índices inferiores, para que éstas no sean sometidas a evolución un número excesivo de veces comparado con las redes de índices más grandes. Además el efecto benigno de las migraciones pierde eficiencia al realizarse entre poblaciones muy distanciadas entre sí.

En cada ciclo de exploración la población de redes sometidas a evolución o bien es el producto de una inicialización aleatoria, en caso de ser la primera vez que se evoluciona, o bien está conformada por los mismos individuos del ciclo anterior salvo aquellos reemplazados por el esquema de migraciones que se describe a continuación.

4.2. Esquema de migraciones

En el esquema de migraciones implementado, cada población P_i, excepto la última, elimina su cuarta parte peor ranqueada reemplazándola por los mejores individuos provenientes de las poblaciones P_j, con $i < j <= n$. (Fig. 5)

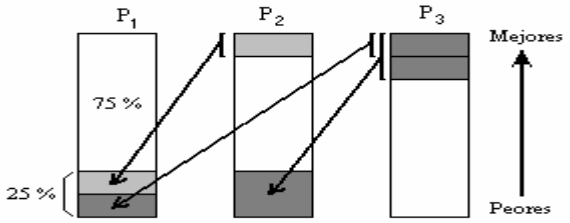

Figura 5. Esquema de migraciones para $i = 1$, $n = 3$

5. Evasión de obstáculos, búsqueda y recolección de objetos

5.1. Definición del problema

Se pretende conseguir comportamiento inteligente en un agente que se mueve libremente en dos dimensiones dentro de los límites de un entorno virtual, interactuando con obstáculos, que debe aprender a evitar, y objetos que debe encontrar y recoger.

Un controlador dirige los movimientos del agente en un intervalo temporal simulado por la sucesión de T instantes discretos de tiempo (pasos de simulación). En cada instante el controlador es estimulado por un conjunto de señales de entrada. La salida está conformada por un par ordenado (α, ρ) que determina el ángulo de giro y desplazamiento que realiza el agente sobre la superficie. No obstante, los obstáculos presentes y los límites del entorno pueden impedir que el movimiento se lleve acabo.

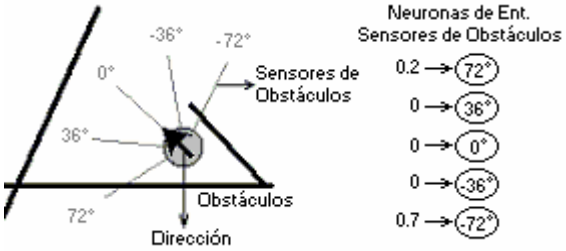

Figura 6. Sensores de obstáculos

El objetivo del problema es controlar al agente para que, partiendo desde un sitio determinado, encuentre y retire del escenario dos objetos de distinta clase, en un orden determinado, en la menor cantidad de pasos posible.

5.2. Agente

Cada agente posee 5 sensores para detectar obstáculos, a corta distancia (2 veces su propio diámetro), distribuidos uniformemente hacia el frente del agente, para lograr un ángulo de 144º de visión. (fig. 6).

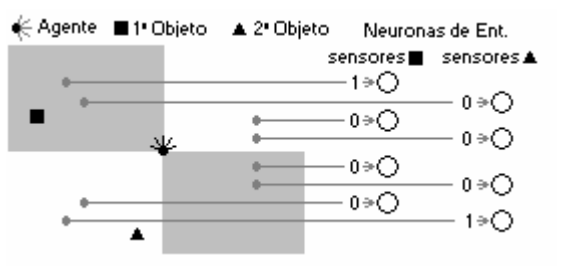

Figura 7. Sensores de objetos

Los sensores de obstáculos pueden considerarse como prolongaciones sensibles al tacto orientadas

hacia delante en cinco direcciones. Cada sensor brinda un valor real perteneciente al intervalo [0,1] directamente proporcional a la cercanía del obstáculo detectado.

Además posee 4 sensores, afectados a la detección del objeto 1, y otros 4 para la detección del objeto 2. Indican ausencia o presencia del objeto en un radio de 90º alrededor del agente, por medio de los valores 0 ó 1. La detección de los objetos se realiza a cualquier distancia, siempre que no se interponga ningún obstáculo que impida su visualización (fig. 7).

Los cuatro sensores afectados a la detección de un objeto, son seteados a 1, en el momento en que el objeto correspondiente es recogido.

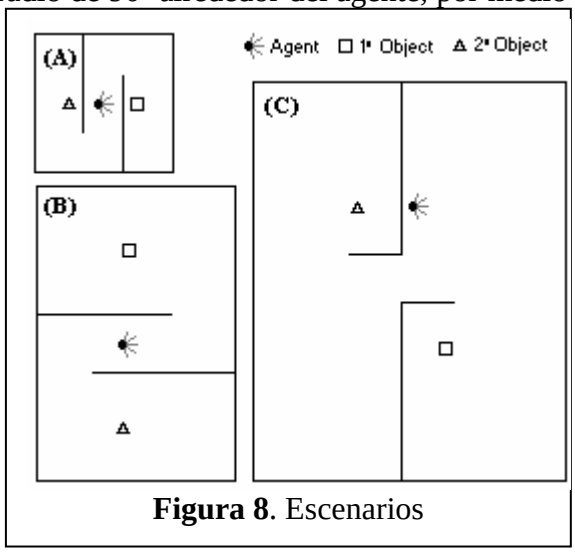

Figura 8. Escenarios

5.3. Escenarios Definidos

La complejidad del problema varía en función de los escenarios definidos (fig. 8)

El escenario A es el más sencillo de los tres. Sus dimensiones reducidas y la disposición de los obstáculos limita la movilidad del agente. Un algoritmo evolutivo hallará sin demasiados inconvenientes una buena solución. Alcanza con evolucionar un controlador que avance hasta encontrar un obstáculo (pared del laberinto) y recorra el escenario sin apartarse del mismo maneniéndolo siempre a mano derecha (conocida heurística para sortear ciertos laberintos).

El escenario B representa un nivel de dificultad intermedio. El controlador debe ser más elaborado que en el caso del escenario A. Aquí es necesario que el agente se aparte de las paredes para alcanzar los objetos. Una vez recogido el primer objeto, el desafío consiste en hallar la dirección más adecuada para proseguir. Debe considerarse que desde ese lugar el segundo objeto no está “a la vista” ni tampoco ninguno de los obstáculos. El agente sólo maneja la información que el primer objeto ha sido recogido (cuatro sensores seteados a 1) y a partir de ella, debe adoptar la estrategia de movimiento más adecuada para resolver el escenario de manera eficiente (menor cantidad de pasos posibles).

El escenario C es el más difícil de los tres. A las dificultades del escenario B debe sumarse la necesidad de realizar varios cambios de dirección para hallar un recorrido eficiente. Este comportamiento no es sencillo de adquirir, sobre todo si, como ocurre en este caso, no existen cambios demasiado significativos en la entrada de datos al controlador.

5.4. Asignación del fitness

El problema de evasión de obstáculos puede considerarse del tipo Sequential decision tasks. Su característica principal es la dificultad para asignar con precisión la bondad de una acción tomada, siendo necesaria una secuencia de decisiones antes de poder medir cuál ha sido el efecto de cualquiera de ellas. Son ejemplos del mundo real: el enrutado de información en los routers de la Internet, el control de flujo en un reactor químico, el control de tráfico aéreo, etc. En todos estos casos, el efecto de una decisión simple se evidencia transcurrido un lapso de tiempo, y aún así, frecuentemente es difícil establecer cuáles decisiones fueron las responsables, y en qué medida, de lo acontecido[10].

Se ha enmarcado a la evasión de obstáculos y búsqueda de objetos dentro de este tipo de problemas y por lo tanto, no se hace ninguna valoración de aptitud de los controladores hasta que no haya concluido la simulación utilizada para su evaluación.

Una vez finalizado el tiempo de simulación se calcula el fitness del controlador que dirige al agente de la siguiente manera: Sea *f(a)* el valor de aptitud asignado al agente *a* por su desempeño en la simulación.

- Si el objeto 1 no fue alcanzado, *f(a)* tomará un valor del intervalo [0,40) calculado proporcionalmente al camino recorrido hacia dicho objeto.
- Si, por el contrario, el objeto 1 fue recogido pero aún el objeto 2 no lo fue, *f(a)* tomará un valor del intervalo [40,50) calculado proporcionalmente al camino recorrido hacia el objeto 2.
- Finalmente en caso de haber recogido ambos objetos *f(a)* pertenecerá al intervalo [50,60). Si *s* es el número de pasos utilizados para completar la tarea, *f(a)*=50+10(*o*/*s*) siendo *o* una estimación de la cantidad de pasos necesaria para resolver el escenario por un controlador óptimo.

5.5. Arquitectura de las redes neuronales artificiales

Para construir los arreglos neuronales se utilizan redes feedforward de una sóla capa oculta, con un esquema de conexión libre (no completamente conectado), con evolución de término de tendencia y función de transferencia, pudiendo cada nodo poseer una de las siguientes cuatro sigmoides: $f_1(x)=1/(1+\exp(-0.5x))$, $f_2(x)=1/(1+exp(-x))$, $f_3(x)=1/(1+\exp(-1.5x))$, $f_4(x)=1/(1+exp(-2x))$. La evolución de función de transferencia ha demostrado tener un buen desempeño aplicada a problemas de evasión de obstáculos y alcance de objetivos [2][3], razón por la cual se utiliza en este trabajo.

6. Experimentación

El rendimiento de ANELA fue medido y comparado con SANE. Se ha elegido este último método como representante de las soluciones basadas en una única red neuronal por su alto rendimiento demostrado [10]. A su vez, ANELA y ANELAR fueron comparados entre sí.

Puede consultarse [6] para obtener mayores detalles sobre la experimentación que compara los rendimientos de ANELA y SANE. En el presente trabajo cobran relevancia los resultados obtenidos en relación a la longitud de los arreglos neuronales de los métodos ANELA y ANELAR.

En todas las pruebas realizadas se evolucionaron redes feedforward con 13 neuronas de entrada, 2 de salida (3 en el caso de los arreglos neuronales), y 8 neuronas ocultas, conexión de tendencia y evolución de función de transferencia. El tamaño de las poblaciones de redes neuronales fue establecido en 80 individuos.

Se definió como condición de finalización de la fase de exploración la obtención de un fitness igual al 75% del valor máximo alcanzable por un controlador óptimo (estimado sobre cada escenario).

Se realizaron 30 evoluciones por cada caso de prueba planteado. Para analizar el rendimiento de los métodos testeados se utilizan las funciones *fitnessAvrg* y *HitRatio* definidas de la siguiente manera:

$$FitnessAvrg(g) = \frac{1}{30}\sum_{i=1}^{30} F_i(g) \qquad HitRatio(g) = \frac{100}{30}\sum_{i=1}^{30} H_i(g)$$

Siendo $F_i(g)$ el valor de fitness del mejor individuo de la generación *g*, en la evolución número *i*, y $H_i(g)$ una función que devuelve 1 ó 0, de acuerdo al éxito o fracaso de la evolución número *i*, en hallar un controlador que resuelva la tarea (recogiendo ambos objetos) en a lo sumo *g* generaciones.

6.1. Resultados obtenidos

Sea *e* la cantidad de generaciones estancadas necesarias para hacer verdadera la condición `fitness_Estacionario`. *e* es un parámetro del método ANELA. Las pruebas sobre los tres escenarios arrojaron resultados muy similares determinando que el valor más pequeño para *e* maximiza el rendimiento del método ANELA.

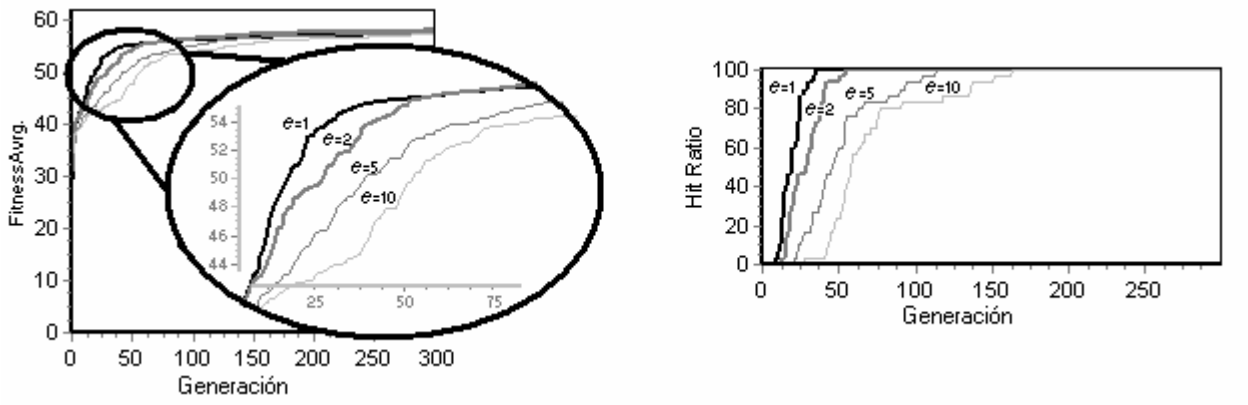

Figura 9. Fitness promedio y Hit ratio para distintos valores de *e*

A continuación se presentan los gráficos comparativos entre los métodos ANELA y SANE sobre los diferentes escenarios. Para el método ANELA se ha establecido en 1 el valor del parámetro *e*.

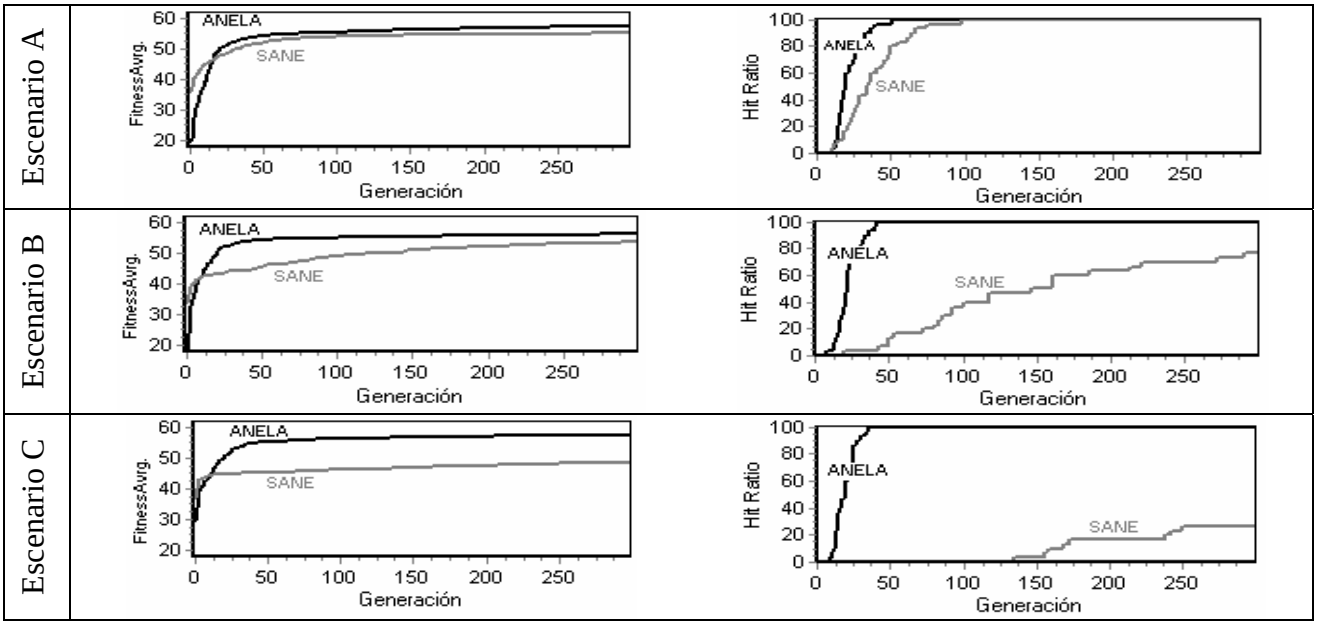

Figura 10. Fitness promedio y Hit Ratio sobre los escenarios A, B y C

Los datos recogidos en la experimentación evidencian que a medida que el problema se hace más complejo se acentúan las diferencias de rendimiento entre ambos métodos. En las pruebas realizadas sobre los tres escenarios se observa además que el rendimiento de SANE es afectado notoriamente por el escenario sobre el que se realiza la prueba. Por el contrario, ANELA se comporta de manera muy similar en todos ellos.

El rendimiento de ANELAR fue comparado con el del método ANELA para distintos valores del parámetro *e*. También se ha tenido en cuenta la longitud de los arreglos evolucionados en ambos métodos, ya que esta es la motivación principal de ANELAR.

Al igual que ANELA, ANELAR no mostró diferencias significativas en los distintos escenarios probados. Las figuras 11 y 12 están basadas en los datos recogidos durante la experimentación en el escenario C.

El rendimiento de ANELAR resultó ser muy similar al de ANELA con parámetro $e = 10$. Para $e > 10$ ANELAR obtuvo los mejores fitness en menor cantidad de generaciones. Lo opuesto ocurrió para $e < 10$ donde ANELA se mostró con mejor desempeño (ver fig. 11). Sin embargo, si se pretende hallar arreglos neuronales de longitud reducida, ANELAR ha demostrado ser siempre la mejor opción (ver Fig. 12).

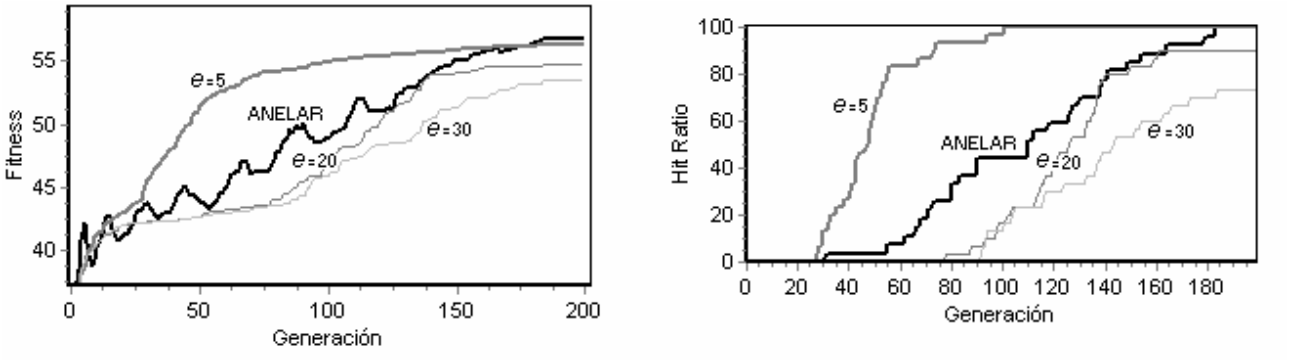

Figura 11. Fitness promedio y Hit ratio para ANELAR y ANELA con distintos valores de *e*

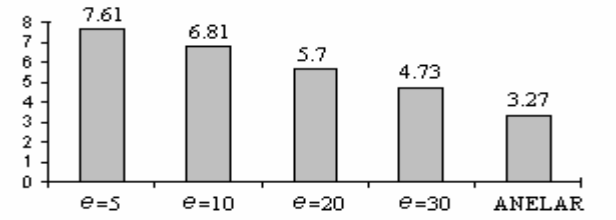

Figura 12. Longitud promedio de los arreglos neuronales

Finalmente debe notarse que el rendimiento obtenido por ANELAR, aunque ciertamente inferior a ANELA con parámetro *e* pequeño, sigue siendo alto (ver Fig. 13 donde se compara con SANE).

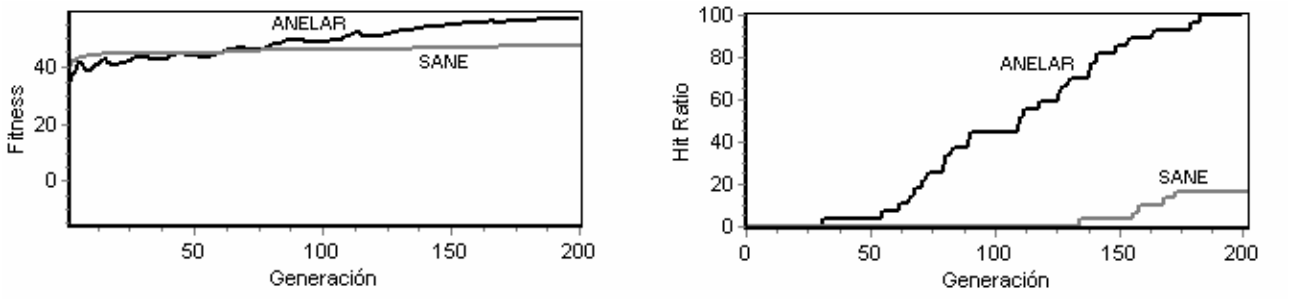

Figura 13. Comparación entre los métodos SANE y ANELAR

7. Conclusiones y líneas de trabajo futuras

ANELA es un método que evoluciona arreglos neuronales para controlar procesos en forma más eficiente que los métodos convencionales con similares requerimientos de procesamiento.

La variante ANELAR ha puesto en evidencia que es posible conseguir por evolución controladores neuronales basados en arreglos de longitud reducida, conservando aún un alto rendimiento. Éste debería ser el método elegido en caso de ser relevante economizar espacio de almacenamiento para el controlador.

Se espera aplicar los resultados obtenidos en el área de la robótica. Entre otras aplicaciones se tiene previsto comandar un brazo robot de cinco grados de libertad a partir de un sistema de adquisición de imágenes externo.

8. Referencias

[1] Bruce, J. and Miikkulainnen, R. Evolving Populations Of Expert Neural Networks. Department of Computer Sciences, The University of Texas at Austin. *Proceedings of the Genetic and Evolutionary Computation Conference*. (GECCO-2001, San Francisco, CA), (2001), pp. 251--257.

[2] Corbalán, L., Pisano, M., Osella Massa, G. y Lanzarini L. Criaturas Virtuales especificadas a través de Redes Neuronales Evolutivas. *VII Congreso Argentino de Ciencias de la Computación*. Argentina, Vol. 2, (Octubre 2001), pp. 1105-1115.

[3] Corbalán Leonardo. Evolución de redes neuronales para comandar criaturas que alcanzan objetivos sorteando obstáculos en un entorno virtual 2D. *Tesis de grado correspondiente a la carrera Lic. en Informática*. UNLP. Marzo 2002.

[4] Corbalán Leonardo y Lanzarini Laura. Arreglos neuronales evolutivos aplicados a la evasión de obstáculos y alcance de objetivos. *VIII Congreso Argentino de Ciencias de la Computación CACIC 2002* (octubre 2002) - *XXVIII Conferencia Latinoamericana de Informática CLEI 2002* (noviembre 2002).

[5] Corbalán Leonardo y Lanzarini Laura. An ENA-Based Strategy Replacing Subobjectives Definition in Incremental Learning. *International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications, (CSITeA'03).* Junio 2003, Río de Janeiro, Brasil.

[6] Corbalán Leonardo, Lanzarini Laura, De Giusti, Armando. "ALENA. Adaptive-Length Evolving Neural Arrays". Journal of Computer Science and Technology (JCS&T). Vol. 4 - No. 1 - April 2004. pp 59-65.

[7] Freeman, J. A. & Skapura, D. M. Redes neuronales Algoritmos, aplicaciones y técnicas de programación. Addison–Wesley, 1991. Versión en español de: Rafael García -Bermejo Giner. Addison–Wesley Iberoamericana 1993.

[8] Goldberg, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison–Wesley, 1989. pág. 10

[9] Gomez, F. and Miikkulainen, R. Incremental Evolution Of Complex General Behavior Department of Computer Sciences, The University of Texas at Austin. *Adaptive Behavior*. Vol 5, (1997), pp.317-342.

[10] Moriarty, D. E. & Miikkulainen, R. Forming Neural Networks Through Efficient and Adaptive Coevolution. Information Sciences Institute, University of Southern California. Department of Computer Sciences, The University of Texas at Austin. *IEEE Transactions on Evolutionary Computation*. Vol.5, (1997), pp.373-399.

[11] Yao, X. and Liu, Y. Ensemble Structure of Evolutionary Artificial Neural networks. *Computational intelligence Group, School of Computer Sciencie University College*. Australian Defence Force Academy, Canberra, ACT, Australia 2600. 1996.

[12] Yao, X. Evolving Artificial Neural networks. School of Computer Sciencie The University of Birmingham Edgbaston, Birmingham B15 2TT. *Proceedings of the IEEE*. Vol.87, No.9, (September 1999), pp.1423-1447.