Controlador Robótico obtenido a través de una metaheurística de población variable

Franco Ronchetti, Laura Lanzarini

III-LIDI (Institute of Research in Computer Science LIDI)
Faculty of Computer Science, National University of La Plata
La Plata, Buenos Aires, Argentina
francoronchetti@gmail.com ; laural@lidi.info.unlp.edu.ar

Abstract. Las metaheurísticas, por su capacidad de adaptación al entorno de información, son herramientas sumamente útiles para obtener controladores robóticos. En general, se trata de una tarea computacionalmente costosa lo que ha motivado el estudio de alternativas para reducir su tiempo de obtención. Este artículo propone una metaheurística de población variable que utiliza especiación para obtener un controlador robótico, basado en una red neuronal de arquitectura mínima, con capacidad para resolver el problema de evasión de obstáculos y alcance de objetivos. Esta solución es novedosa ya que en general se utilizan poblaciones de tamaño fijo. A lo largo de este trabajo se discuten los operadores genéticos utilizados así como los distintos aspectos de implementación que fueron considerados para introducir esta variación poblacional. Las pruebas realizadas tanto en ambientes simulados como sobre el robot real han dado resultados satisfactorios.

Palabras Claves : Evolutiva. Especiación. Población variable.

1 Introducción

La Robótica Evolutiva muestra un fuerte interés en las redes neuronales por considerarlas un modelo artificial de la manera en que los humanos aprenden y por poseer la capacidad de representar el conocimiento adquirido a través de una estructura que, una vez entrenada, es capaz de operar en tiempo real.
Cuando se trata de obtener un controlador neuronal para un robot autónomo, las metaheurísticas poblacionales resultan más adecuadas para lograr la adaptación de la red que un entrenamiento por gradiente. Esto se debe a que generalmente se trata de problemas complejos entendiendo por tales aquellos cuya solución implica aprender alguna estrategia.
Este trabajo presenta una solución basada en una metaheurística poblacional con la particularidad de que dicha población puede cambiar de tamaño durante la adaptación. Se trata de un enfoque novedoso ya que en general las soluciones existentes utilizan un tamaño de población fija. Esto se encuentra relacionado con el costo computacional del proceso de adaptación. Se requiere de un control adecuado, tanto en lo que se refiere al agregado de nuevos individuos como a

su eliminación, a fin de no incrementar excesivamente el tiempo de cálculo ni perder diversidad durante el proceso.
Este trabajo está organizado de la siguiente forma: la sección 2 presenta algunos trabajos relacionados con el problema a resolver, la sección 3 describe el método propuesto, la sección 4 detalla los aspectos considerados durante la implementación, la sección 5 muestra los resultados obtenidos y la sección 6 expone las conclusiones y líneas de trabajo futuras.

2 Trabajos relacionados

Existen distintos trabajos previos dedicados a la obtención de controladores para agentes autónomos utilizando redes neuronales y algoritmos evolutivos.
Investigaciones realizadas han demostrado que la descomposición del problema en tareas más simples facilita el entrenamiento del controlador. En esta dirección, se desarrollaron soluciones basadas en aprendizaje incremental [1, 2] y en aprendizaje por capas [3]. Este tipo de soluciones, si bien permiten obtener buenos resultados, requieren de una gran asistencia. En el caso del aprendizaje incremental es necesario identificar previamente las distintas etapas de complejidad creciente que lo componen y esto no siempre es posible ya que no todo problema puede descomponerse de esta forma. En el caso del aprendizaje por capas es preciso indicar el orden en que cada módulo debe ejecutarse considerando el orden en que fueron aprendidos.
En [4] se desarrolló un mecanismo de integración automática de módulos que permitió combinar comportamientos básicos aprendidos previamente reduciendo de esta forma el costo de entrenamiento.
También se han analizado distintas estrategias elitistas de evolución como forma de mejorar el efecto de los operadores genéticos [6].

3 Estrategia Propuesta

El presente trabajo propone un mecanismo para obtener un controlador robótico para un robot Khepera II, basado en una red neuronal, con capacidad para resolver el problema de evasión de obstáculos y alcance de objetivos. La arquitectura de la red a evolucionar es de tamaño mínimo, se define a priori y se mantiene fija durante todo el proceso. Es decir que la evolución sólo afecta a los pesos de la red. De esta forma se reduce el costo computacional y luego del entrenamiento se obtiene un controlador pequeño que puede ser instalado fácilmente en el robot real.
La estrategia utilizada es poblacional y cada individuo tiene:

- Un cromosoma formado por todos los pesos de red neuronal. Este cromosoma es de longitud fija ya que la arquitectura no cambia durante el proceso adaptativo.
- Un identificador de especie que permite reconocer individuos con comportamiento similar.

– Una probabilidad de mutación y un grado de mutación que al estar aplicado al individuo le permite regular la manera en que genera sus descendientes durante el proceso.
– Un tiempo de vida máximo y una edad actual que regulan la permanencia del individuo en la población y sus parámetros de mutación respectivamente.
– Su valor de aptitud que se calcula midiendo el desempeño del individuo en la solución del problema.

Finalmente, se utiliza un pequeño porcentaje de elitismo para retener las mejores soluciones encontradas a lo largo del proceso evolutivo.

3.1 Población de tamaño variable

El algoritmo posee una única población de tamaño variable con especiación. Se utilizan técnicas específicas para mantener vivas a todas las especies durante el proceso, incentivando a que se reproduzcan y de esta forma aumenten el espacio de búsqueda.
Al crear la población inicial se agrupa a los individuos de acuerdo a su cromosoma y se les asigna un identificador de especie. Así, al comenzar, los individuos de la misma especie tendrán comportamientos similares. A medida que las generaciones transcurren, cada especie aumenta su tamaño generando nuevos individuos cada vez más aptos.
Cada individuo de la población recibe, al momento de su creación, un valor numérico correspondiente a su tiempo de vida. Este valor representa la cantidad máxima de iteraciones que permanece dentro la población. Por cada iteración su edad aumenta y al llegar a este tiempo de vida máximo el individuo es eliminado.
Para calcular los tiempos de vida se utilizó el algoritmo de asignación de tiempos de vida fijo por clases definido en [7]. Consiste en dividir a los individuos de la población en grupos diferentes de acuerdo a su valor de aptitud, utilizando un algoritmo del estilo winner-take-all, como por ejemplo k-medias [5]. A cada grupo se le asigna una parte proporcional del rango total de tiempos de vida haciendo corresponder los valores más bajos con el grupo de menor fitness (ver figura 1). Los individuos de un mismo grupo reciben un tiempo de vida proporcional al de su clase según su valor de aptitud. Para más detalles consultar [7]
Los individuos son eliminados de la población por dos motivos: porque su tiempo

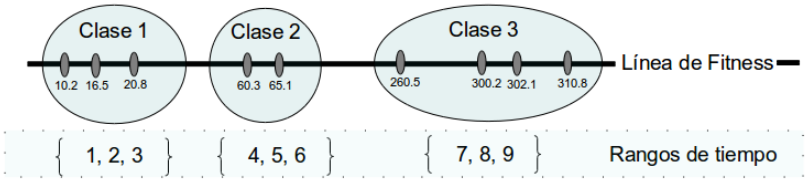

Fig. 1. Asignación de Tiempos de Vida por clase

de vida se terminó o porque se ha producido un crecimiento desmedido. En ambos casos, la eliminación se realiza cuidando de no eliminar individuos de especies que han llegado a una cantidad mínima establecida.
Para evitar el crecimiento desmedido de la población y por consiguiente el aumento de tiempo de convergencia del algoritmo, se establece un máximo posible y se utiliza una técnica de eliminación de individuos. El mecanismo consiste en ordenarlos de acuerdo a su fitness de menor a mayor y eliminar de forma lineal los menos aptos salteando los que no puedan ser eliminados en su especie.

3.2 Operadores genéticos

El método de selección usado para la reproducción es un torneo probabilístico binario en dos etapas. Cada descendiente solo posee un padre. Se toman pares de individuos de forma aleatoria y se los hace competir de acuerdo a su fitness. Luego, los ganadores compiten de la misma forma (esta es la segunda etapa del torneo) hasta obtener una población de padres cuatro veces más chica que la original.
Una vez obtenida la población de padres se aplican los operadores de cruce y mutación. Se utiliza un operador de cruce específicamente diseñado para el problema, que al aplicarlo, se obtiene un controlador que permite al robot comportarse de manera simétrica (espejo) respecto a su antecesor. Por otro lado se utiliza un operador de mutación que transforma algunos pesos del cromosoma aumentándolo o disminuyéndolo en pequeños grados. Cada individuo tiene sus propios parámetros de mutación para sus descendientes. A medida que el individuo envejece decrece linealmente su capacidad de mutar. Es decir que, sus primeros descendientes son menos parecidos a él que los últimos. Una alta capacidad de mutar genera individuos con nuevos comportamientos, mientras que una baja capacidad ayuda a refinar el comportamiento hacia una mejor solución.
La figura 2 muestra el algoritmo descripto en esta sección.

```
crear una población inicial y establecer las diferentes especies
calcular el fitness y el tiempo de vida de los individuos
while no se alcance el objetivo previsto do
   retener a los mejores
   generar una nueva población mediante torneo probabilístico binario
   medir el fitness y el tiempo de vida de la nueva generación
   envejecer la población anterior
   eliminar los individuos con tiempo de vida cero cuidando las especies
   agregar la nueva generación a la población total
   evitar superpoblación cuidando las especies
end while
```

Fig. 2. Pseudocódigo de la estrategia evolutiva propuesta

4 Aspectos de Implementación

La arquitectura de la red neuronal usada para el controlador es mínima (ver Figura 3). Consta de 8 neuronas de entradas que obtienen los valores de los sensores del robot y dos neuronas de salida que especifican las velocidades de los motores.
La principal ventaja de usar una arquitectura tan pequeña es la rapidez con que se consigue una solución evolucionada.

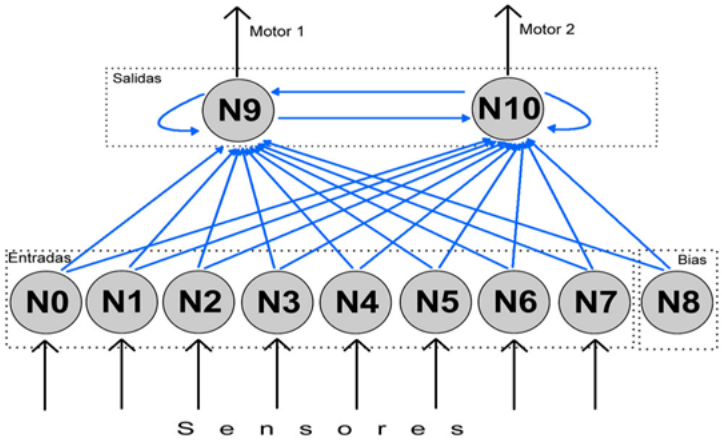

Fig. 3. Arquitectura de la red neuronal

El cromosoma C de cada individuo es un vector de tamaño 22 con todos los pesos de la red interpretados de la siguiente forma:

- C(1..8): pesos de las neuronas de entrada hacia la neurona de salida N9.
- C(9): bias de la neurona N9.
- C(10):peso de la neurona N9 hacia sí misma.
- C(11): peso de la neurona N10 hacia N9
- C(12..19): pesos de las neuronas de entrada hacia la neurona de salida N10.
- C(20): bias de la neurona N10.
- C(21):peso de la neurona N9 hacia N10.
- C(22): peso de la neurona N10 hacia sí misma.

Los cromosomas de los individuos de la población inicial se crean con valores aleatorios entre -0.5 y 0.5 uniformemente distribuidos.
La probabilidad utilizada para el torneo probabilístico binario fue de 0.8.
La cantidad de clases empleadas para agrupar los fitness y calcular los tiempos de vida fue 3.

Se utilizaron 8 especies distintas estableciendo una cantidad mínima de 14 individuos por especie.
El tiempo de vida mínimo permitido es de 1 iteración y el máximo de 9 [9].
El tamaño inicial de la población fue de 80 individuos y el tamaño máximo permitido fue de 300. De esta forma, el valor máximo de individuos nuevos por cada generación es 75.

4.1 Función de aptitud

En base al problema a resolver, la función de fitness considera más aptos a aquellos individuos capaces de desplazarse largas distancias acercándose al objetivo. Para obtener el valor de aptitud de cada individuo, el mismo es colocado en la posición de partida y evaluado una cantidad de pasos n (donde n es establecido de acuerdo al problema específico, en este caso n=300). La función utilizada es la indicada en la ecuación (1)

$$fitness = \sum_{i=1}^{n} (\alpha_1 * fitVelocidad + \alpha_2 * fitGiro + \alpha_3 * fitObjetivo) \quad (1)$$

siendo

$$fitVelocidad = ((motor_1 + motor_2) + 2)/4 \quad (2)$$

$$fitGiro = ((1 - abs(motor_1 - motor_2)) + 1)/2 \quad (3)$$

$$fitObjetivo = 1/distanciaAlObjetivo \quad (4)$$

donde

- $motor_1$ y $motor_2$ son los valores de los motores y pueden tomar valores enteros entre -10 y 10.
- $distanciaAlObjetivo$ es la distancia actual entre el robot y el objetivo.

Cada expresión retorna un valor entre 0 y 1. De esta forma las constantes α_1, α_2 y α_3 hacen presión en los diferentes aspectos a tener en cuenta. En este caso $\alpha_1 = 0.1$, $\alpha_2 = 0.1$, $\alpha_3 = 2$. La diferencia de magnitud entre α_1 y α_2 respecto de α_3 evita incrementar excesivamente el valor de aptitud de aquellos que avanzan estando muy lejos del objetivo.
Con $fitVelocidad$ el individuo tendrá mayor fitness cuanto mayor sea la velocidad de los motores, llegando a su máximo valor cuando ambos motores avancen a máxima velocidad.
Con $fitGiro$ se intenta evitar que el robot gire. De esta forma tendrá mayor fitness cuanto menor sea la distancia entre el valor de un motor y el otro, obteniendo el mayor fitness cuando ambos motores tienen la misma velocidad.
Con $fitObjetivo$ se premia al individuo de forma inversamente proporcional a su distancia a la salida del laberinto.

En caso que el robot colisione, termina la evaluación y se retorna el fitness acumulado hasta el momento.
En caso que el robot llegue al objetivo se evalúa el fitness con la ecuación (5)

$$fitnessActual = fitnessActual * \beta * (1/cantPasos) \quad (5)$$

Donde β es una constante que incrementa el fitness por haber llegado a la salida. En este caso $\beta = 1000$. La expresión $(1/cantPasos)$ otorga un fitness inversamente proporcional a la cantidad de pasos dados para alcanzar el objetivo.

4.2 Cruce

Es conocido el efecto destructivo que el operador de cruce presenta cuando se evolucionan redes neuronales [8]. Por tal motivo fue aplicado con probabilidad 0.2 y sólo permitiendo cortar el cromosoma en su punto medio con la intención de intercambiar el comportamiento de los motores que controlan las ruedas del robot. Para implementar el operador hace falta analizar la ubicación física de los sensores que dan lugar a los valores de entrada de la red.
El robot presenta una serie de 8 sensores como se muestra en la figura 4. Los sensores 0, 1, 2 y 7 que aparecen del lado izquierdo tienen sus correspondientes sensores del lado derecho numerados como 5,4, 3 y 6 respectivamente. Esta correspondencia es tenida en cuenta a la hora de efectuar el cruce. De la misma forma, los pesos de y hacia los motores deberán ser invertidos.
La figura 5 describe de forma detallada cómo se realiza la modificación en el cromosoma para cambiar el comportamiento del individuo.

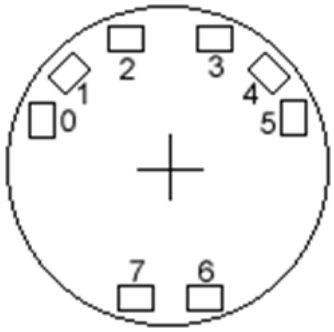

Fig. 4. Vista superior del Robot Khepera II

4.3 Operador de mutación

Cada individuo posee una probabilidad m y un grado s de mutación. Ambos valores sólo tienen incidencia sobre su descendencia. El primero se utiliza para

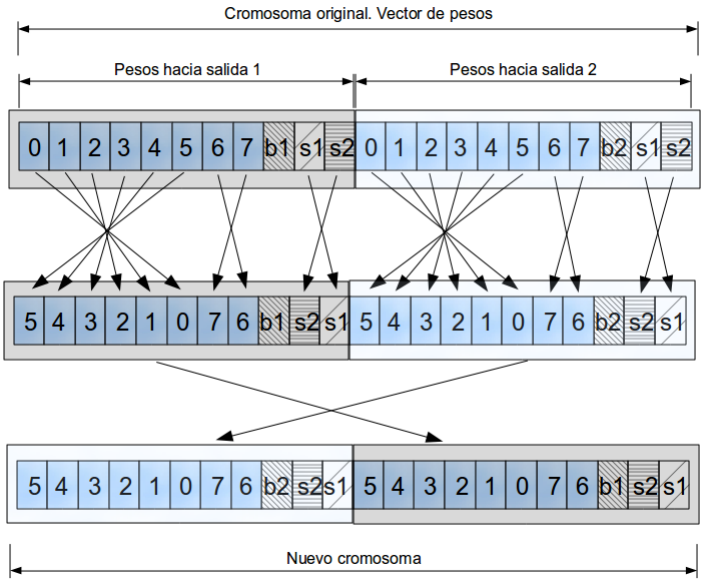

Fig. 5. Operador de cruce. Se aplica sobre un único padre y da lugar a un único hijo

decidir uniformemente si se muta cada uno de los pesos del cromosoma del hijo y en caso de producirse la mutación, el segundo permite calcular el incremento que se adicionará al valor del peso actual. Este incremento se calcula con una probabilidad gaussiana con media 0 y desviación estándar s.
A medida que envejece, cada individuo disminuye linealmente su probabilidad de mutar descendientes, comenzando en 0.5 hasta llegar al mínimo establecido en 0.05; valor que equivale a mutar aproximadamente un solo peso de la red.
Al igual que la probabilidad, cada individuo disminuye su grado de mutación a medida que envejece, comenzando con un grado de 0.15 y terminando en 0.05.
Definido de esta forma, el operador de mutación permite que los individuos posean una gran capacidad de exploración inicial para pasar luego a una etapa de especialización.

5 Resultados

Se realizaron 50 ejecuciones independientes del algoritmo propuesto midiendo el desempeño del robot Khepera II para salir de un laberinto.
En la figura 6 se observan dos gráficos comparativos de la aptitud de los individuos a lo largo del tiempo. A izquierda (figura 6.a) se observa el promedio de cincuenta ejecuciones independientes con el método propuesto y los parámetros antes descriptos. A la derecha (figura 6.b) se aprecia el promedio de otras 50 ejecuciones independientes sin usar especiación y sin el operador de cruce im-

plementado. En ambos gráficos se ha representado el fitness promeido de las 50 ejecuciones del mejor individuo y de la población completa.
Puede verse en la Figura 6 que el método propuesto (figura 6.a) consigue buenos individuos a partir de la generación 60 mientras que el método evolutivo que no utiliza especiación ni cruce (figura 6.b) queda estancado en un óptimo local. Esto último se debe a que los individuos obtienen un buen valor de aptitud por recorrer grandes distancias a velocidades altas sin necesidad de acercarse a la salida del laberinto. Al no haber especiación ni cruce, estos individuos serán los mejores y tendrán más posibilidades de obtener descendientes, achicando el espacio de búsqueda.
La función de aptitud descripta en la subsección 4.1. posee dos etapas diferentes. En la primera los individuos adquieren un mejor puntaje a medida que avanzan y aprenden a evitar obstáculos. Una vez que los individuos consiguen llegar al objetivo comienza la segunda etapa ya que la población posee individuos con fitness alto y la función deberá ser capaz de compararlos según su capacidad para resolver el laberinto. Esto se nota claramente en el gráfico 6.a, donde el fitnes aumenta considerablemente a partir de la generación 60. En el gráfico 6.b, en cambio, se observa que el problema queda estancado en la primera etapa por los motivos antes expuestos.
En el gráfico 6.b se aprecia que la aptitud media es mejor durante las primeras 60 generaciones que en 6.a. Esto es porque la población promedio representada en 6.b) comienza a perder diversidad, obteniendo todos un buen valor de aptitud y quedando atrapados en un máximo local.

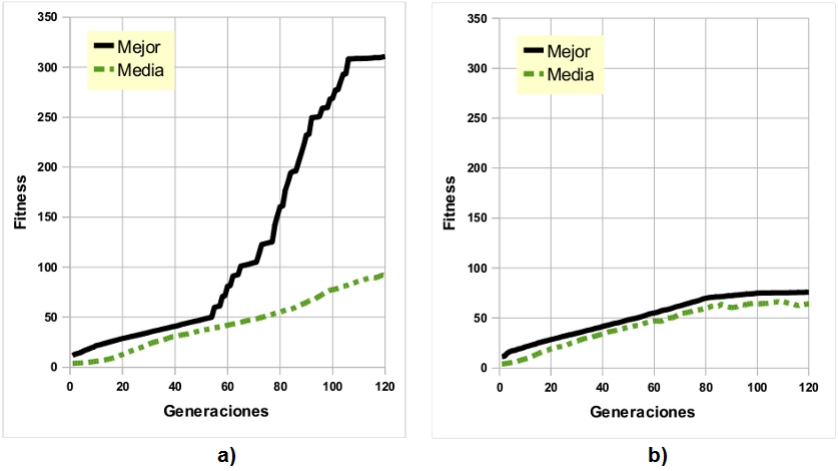

Fig. 6. Valor de la función de fitness obtenida al promediar el resultado de 50 corridas independientes para: a) Método propuesto, b) Método sin especiación ni cruce (sólo usa mutación y población variable)

6 Conclusiones y trabajos futuros

Se presentó un método evolutivo rápido y eficaz para obtener un controlador robótico pequeño y veloz, que puede ser instalado fácilmente en un robot real.
Los experimentos realizados demostraron la eficacia y efectividad de los operadores genéticos utilizados, así como del criterio de especiación basado en el concepto de familia con tamaño mínimo.
La red neuronal obtenida a partir del método descripto en este trabajo podría servir como parte de la solución de un problema más complejo ya que, luego de haber sido entrenada podría combinarse con otros módulos utilizando la estrategia definida en [4].
Actualmente se está trabajando para mejorar el criterio de especiación buscando equiparar la capacidad de reproducción de cada especie con el objetivo de facilitar el operador de"poda" utilizado para evitar el crecimiento desmedido de la población.

References

1. Corbalán, Lanzarini. Evolving Neural Arrays A new mechanism for learning complex action sequences. Special Issue of Best Papers presented at CLEI'2002. Volumen 6. Number 1, December 2003. Montevideo, Uruguay.
2. Corbalán, Lanzarini, De Giusti A. ALENA. Adaptive-Legth Evolving Neural Arrays". Journal of Computer Science and Technology. Vol 5, nro. 4. 2005. Pags. 59-65.
3. Olivera y Lanzarini. Cyclic Evolution. A new strategy for improving controllers obtained by layered evolution. Journal of Computer Science and Technology. Vol 4, nro. 1. 2005. Pags 211-217.
4. Osella Massa, Vinuesa, Lanzarini. Modular Creation of Neuronal Networks for Autonomous Robot Control. Ibero-American Journal of Artificial Intelligence. Vol. 11, No. 35 (2007), pp. 43-53.
5. Darío Maravall Gomez-Allende. Reconocimiento de Formas y Vision Artificial. Addison-Wesley Iberoamericana. ISBN 0-201-64183-6. 1994
6. Vinuesa, Lanzarini. Neural Networks Elitist Evolution. 29th Internacional Conference Information Technology Interfaces (ITI 2007). Dubrovnik. Croatia. 2007.
7. Lanzarini L.,Sanz C. Naiouf M., Romero F. Mixed alternative in the assignment by classes vs. conventional methods for calculation of individuals lifetime in GAVaPS. Proceedings of the 22nd International Conference on Information Technology Interfaces, 2000. ITI 2000. pp. 383- 389. ISBN: 953-96769-1-6. June 2000
8. 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, pp.1423-1447. September 1999.
9. Walker J. Garrett, S. Wilson, M. The Balance Between Initial Training and Lifelong Adaptation in Evolving Robot Controllers. IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics, Vol. 36, No. 2. April 2006.