Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

Análisis comparativo de los Operadores Genéticos de Cruce Puntual SPX y Uniforme UX aplicados a la resolución del problema de optimización PathFinder

Cristian Zambrano Vega[1], Joel A. Cedeño Muñoz[2], Jefferson Bravo Salvatierra[3], Roberto Pico Saltos[4]

1 Universidad Técnica Estatal de Quevedo, *czambrano@uteq.edu.ec*

2 Universidad Técnica Estatal de Quevedo, *jacedeno@uteq.edu.ec*

3 Universidad Técnica Estatal de Quevedo, jbravo@uteq.edu.ec

4 Universidad Técnica Estatal de Quevedo, rpico@uteq.edu.ec

RESUMEN

En el presente trabajo se emplea un estudio experimental de dos variantes de un algoritmo evolutivo estándar para la exploración de un laberinto (*PathFinder*) que le permita a un robot encontrar el mejor camino de salida a partir de su ubicación dentro del mismo. La primera variante del algoritmo emplea el Operador Genético de Cruce Puntual SPX ("point crossover") y la segunda el Operador Uniforme UX ("uniform crossover"). Los laberintos de prueba están formados por un tamaño de *NxM* posiciones con un 20% de ellas representadas como obstáculos para el robot, la representación de un camino, básicamente está compuesta de cuatro coordenadas internas absolutas (Norte, Sur, Este y Oeste), pero se agregan coordenadas relativas como: Avanzar, Girar-Derecha y Girar-Izquierda, para realizar un análisis adicional al caso de estudio. Para confirmar los resultados se identifican si existen o no diferencias estadísticamente significativas entre los resultados brindados por estos dos operadores. Los resultados obtenidos indican que la mejor variante a implementarse del algoritmo es la que usa el operador UX con coordenadas relativas, ya que siempre genera una solución factible para la salida del laberinto o una solución más cercana a la salida en comparación a las otras variantes.

Palabras Claves: Algoritmos Evolutivos, Operadores Genéticos, Cruce SPX, Cruce UX PathFinder.

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

A Comparative study of the Genetic Operators Single Point Crossover (SPX) and Uniform Crossover UX applied to the Optimization problem PathFinder

ABSTRACT:

In this paper we have carried out an experimental study of two variants of an evolutionary algorithm applied to exploration of a maze (PathFinder) to allows to a robot find the best way out from its location within. The first variant applies the Genetic Operator Single Point Crossover (SPX) and the second the Uniform Crossver. The test mazes are formed by NxM positions with 20% of obstacles for the robot, the representation of the solution (set of steps) is based on internal absolute coordinates (North, South, East and West), but we have added some relatives coordinates (Move, Rotate Right and Rotate –Left) with the aim of perform an additional analysis. To confirm the results we have identified if exist or not statistically significant differences between the results provided by these two operators. The obtained results indicate that the best variant of the evolutionary algorithm to implement is which uses the UX crossover operator with relative coordinates, because always generates a feasible solution to the robot or generates solutions closer to the exit than the other variants in study.

Keywords: Evolutionary algorithms; Genetic Operators; PathFinder; Single Point Crossover; SPX; Uniform Crossover; UX.

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

1 INTRODUCCIÓN

Actualmente los operadores genéticos de cruce tienen una alta incidencia en la resolución de problemas de optimización o búsqueda basados en un ordenador. El presente trabajo tiene como objetivo optimizar la solución de búsqueda del mejor camino de salida mediante la exploración de un laberinto, empleando como referencia los Operadores Genéticos de Cruce Puntual SPX ("point crossover") (Davis, 1991) y el Operador Uniforme UX ("uniform crossover") (Goldberg, 1989), para conocer si existen o no diferencias significativas estadísticamente entre los resultados brindados por estos dos operadores. La implementación de estos dos operadores genéticos de cruce, permitirá obtener mejores resultados de diferenciación, que determinaran la mejor salida según el caso de estudio de adición de coordenadas relativas para un mejor análisis y por ende verificar la mejor solución de salida del laberinto.

2 ALGORITMOS EVOLUTIVOS

Este término es empleado para describir sistemas de resolución de problemas de optimización o búsqueda basados en el ordenador, empleando modelos computacionales de algún mecanismo de evolución conocido como elemento clave en su diseño e implementación. Los Algoritmos Evolutivos (Back, 1996) basan su funcionamiento en la simulación del proceso de evolución natural. Consiste en una técnica iterativa que aplica operadores estocásticos sobre un conjunto de individuos de la población con el propósito de mejorar su fitness, una medida relacionada con la función objetivo del problema en cuestión. Cada individuo de la población representa una solución potencial del problema, codificada de acuerdo a un esquema de representación, generalmente basado en números binarios o reales. Inicialmente la población se genera de forma aleatoria, y luego evoluciona mediante la aplicación iterativa de interacciones denominadas operadores de reproducción, que incluyen combinaciones de individuos y modificaciones aleatorias de las mutaciones. Esta evolución es guiada por una estrategia de selección de los individuos más adaptados a la resolución del problema, de acuerdo a sus valores de fitness.

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

2.1 Estructura de un Algoritmo Evolutivo Simple

Algoritmo Básico

```
1: t ← 0;
2: Inicializar P(t);
3: Evaluar P(t);
4: while! Termination Condition do
5:      t ← t+1;
6:      P'(t) ← Seleccionar [P(t-1)];
7:      P"(t) ← Cruce [P'(t)];
8:      P"'(t) ← Mutación [P"(t)];
9:      P(t) ← Reemplazo(P(t-1),P"'(t));
10:     Evaluar [P(t)];
11: end while
```

2.2. Implementación de la Solución Evolutiva

Un algoritmo evolutivo debe tener los siguientes elementos:

- **Representación**: estructura de datos que codifica los parámetros de una posible solución a un problema.
- **Población**: Conjunto de individuos que representan las variables de decisión de la función objetivo. Dichos individuos deben estar configurados en forma de cromosomas.
- **Técnicas de selección**: Sirve para determinar la supervivencia de un individuo en la próxima generación. Depende del grado de aptitud (fitness) que obtiene un individuo al ser analizado por el resultado de las funciones objetivo.
- **Técnicas de cruce**: En los sistemas biológicos, el cruce es un proceso complejo que ocurre entre parejas de cromosomas. Estos cromosomas se alinean, luego se fraccionan en ciertas partes y posteriormente intercambian fragmentos entre si.
- **Mutación**: Variación de uno o más alelos del gen. Su aplicación en forma aleatoria a diferentes puntos de la cadena cromosómica produce individuos con pequeñas variaciones con respecto al individuo original.
- **Ajuste de parámetros**: a) Tamaño de población, b) Porcentaje de cruza y c) Porcentaje de Mutación.

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

3 METODOLOGÍA

3.1 Laberinto

Para este caso de estudio se implementó un laberinto de tamaño *N x M* posiciones de las cuales, la posición (1,1) representa su entrada y la posición *(N, M)* su salida, siendo esta coordenada el objetivo a alcanzar para cualquier camino trazado sobre el laberinto. Contiene un **ρ** por ciento de las posiciones representadas como obstáculos del Laberinto, y sobre los cuales no puede cruzar ningún camino que desea llegar a la salida. En la figura 1 se muestran cinco Laberintos de prueba, generados bajo estos parámetros.

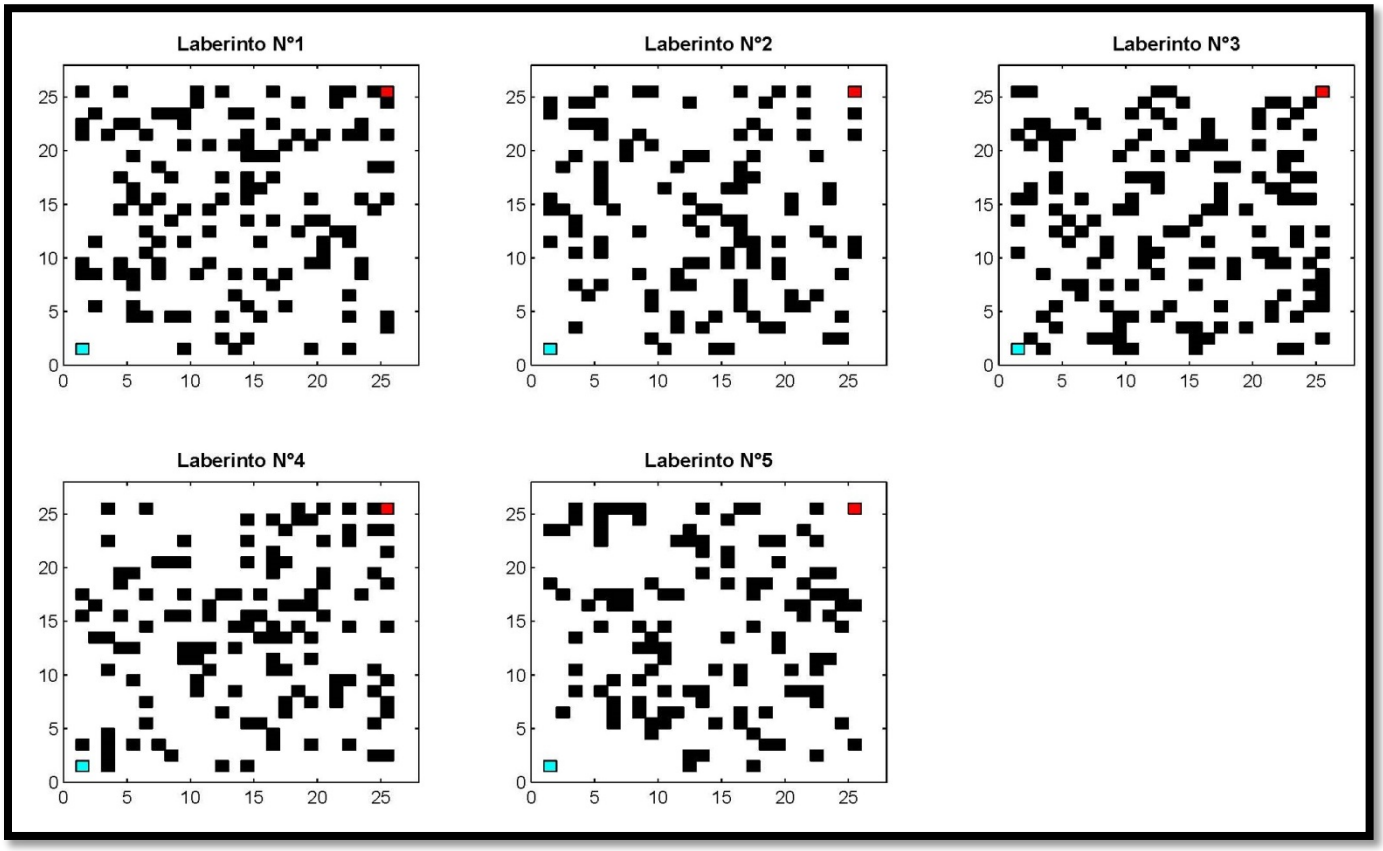


Figura 1: Laberintos de prueba de tamaño (NxM).

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

3.2 Algoritmo genético para resolver el problema del PathFinder

El algoritmo genético implementado en este estudio está compuesto por las siguientes partes:

3.2.1 Representación del individuo

Se empleó una representación numérica, la cual está compuesta por un grupo de $((N + M) * 5)$ valores, de dos conjuntos de dígitos: a) {1, 2, 3,4} para representar las coordenadas internas absolutas y b) {5, 6, 7} para representar las coordenadas internas relativas. Considerando 1: Norte, 2: Sur, 3: Este, 4: Oeste, 5: Avanzar, 6: Girar a la Derecha y 7: Girar a la Izquierda.

3.2.2 La Función Objetivo

La Función Objetivo (fitness) tiene dos objetivos priorizados:

- Encontrar una solución factible (*Camino que llegue a la salida*) e intentar minimizar su longitud (número de pasos del camino).
- Diferenciar las soluciones NO factibles (Camino que NO llegan a la *salida*), mediante la distancia que existe de la salida. Considerando una coordenada valida, aquella que este dentro del laberinto que no esté sobre ningún obstáculo.

La asignación de valores fitness a los individuos se realiza bajo las siguientes relaciones cualitat ivas:

Sean A y B dos soluciones, entonces

- Si A es una solución factible pero B no, entonces A es mejor que B.
- Si ni A ni son soluciones factibles, entonces la que se quede más cerca de la salida es me jor.
- Si tanto A como B son soluciones factibles, entonces la más corta es mejor.

Por la que la fómula de cálculo puede expresarse de la siguiente manera:

$$\boldsymbol{f(x) = s(x)l(x) + \big(1 - s(x)\big)(K + d(x))}$$

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

donde:

- *s(x)* = 1 si x es solución y 0 en otro caso.
- *l(x)* es la longitud de la solución.
- *d(x)* es la distancia a la que se queja de la salida.
- *K* es una constante que hace de "*offset*" para que cualquier solución no factible más larga tiene longitud (*N.M - 1*) en el peor caso. Por lo que pasa hacer *K= N.M.*

Teniendo en cuenta que es un problema de minimización de distancia como de numero de pasos hacia la salida, el objetivo de la función es conseguir primeramente una solución y luego que esta sea de *N + M* pasos como máximo.

3.2.3 Operadores genéticos

- **Operador de selección**

Se empleó el Operador de Selección Por Torneo Binario (M. Belén, 2003), el cual selecciona el mejor de dos individuos escogidos aleatoriamente, considerando el de menor fitness, puesto que trata de un problema de minimización.

- **Operadores de cruce**

Los operadores de cruce actúan sobre una pareja de individuos (progenitores) para producir un nuevo individuo, combinando características de ambos. Para este caso de estudio se implementaron dos operadores de cruce, en las que si tomamos dos cadenas binarias podemos establecer estas dos operaciones:

- **Puntual SPX ("point crossover"):**

Parent 1:	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**
Parent 2:	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

Hijo: **0 0 1 1 1 0 1 0 0 0**

Punto de Cruce

- **Cruce uniforme UX ("uniform crossover"):**

Parent 1: **0 0 0 1 0 1 0 0 0 1**

Parent 2: **1 1 1 0 1 1 0 1 0 0**

Hijo: **0 0 1 1 1 1 0 1 0 1**

- **Operador de Mutación**

La mutación es un operador básico de alteración genética y es aplicada solo a individuos, realizando una pequeña modificación en alguno de sus genes o en el conjunto de ellos, permitiendo explorar nuevas zonas del espacio de búsqueda. (Holland, 1992).

La probabilidad con la estos cambios se producen suele ser baja, para evitar que el AG realice una búsqueda aleatoria. Sin embargo en ocasiones puede ser necesario aumentar esta probabilidad para recuperar la diversidad de la población. En algunas ocasiones se puede realizar un operador de regeneración para evitar problemas de convergencia prematura.

Para este caso de estudio se implementó una mutación que modifica cada uno de los genes del individuo, reemplazándolos por un valor aleatorio del conjunto de dígitos {1,2,3,4,5,6,7} (dependiendo del tamaño del alfabeto).

3.2.4 Modelo del Algoritmo Genético

Para hacer el estudio del problema se implementó un Modelo de AG ***casi completamente generacional***, ya que solo se conserva al mejor individuo de la población inicial al momento de hacer el reemplazo con la nueva población descendiente, generando así una nueva población con el 99% de individuos nuevos.

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

3.3 Parametrización de los Experimentos

Básicamente la configuración para los dos algoritmos es la misma, siendo el Operador de Cruce el que las diferencia.

Su configuración pormenorizada es la siguiente:

Tamaño de población de 100. Condición de parada de 25,000 evaluaciones. Se usó un operador de selección por Torneo Binario. Se estableció un porcentaje de cruce tanto para SPX como para UX igual a 1/ Tamaño Del Individuo y una probabilidad efectiva de mutación de cada gen del individuo igual al 100%. Siendo un Modelo de AG casi completamente Generacional se definió un Elitismo igual a 1. En la Cuadro 1 se muestran los parámetros usados por cada Algoritmo

Cuadro 1: Parametrización (L = Longitud de Variables = (NxM) + 5)

Tamaño de la Población	**100 Soluciones**
Condición de Parada	**25.000 Evaluaciones**
Operador de Selección	**BinaryTournament**
Operador de Cruce	**Cruce puntual SPX y Cruce uniforme UX, pc = 1/L**
Operador de Mutación	**Uniforme, pm = 1.0**
Elitismo	**1 - Modelo casi completamente Generacional -**

3.3 Metodología aplicada en los experimentos

Una vez implementado el Algoritmo para la exploración de un Laberinto, se desarrolló un Caso de Estudio en el que se comparan las dos Operaciones de Cruce: Puntual SPX y Uniforme UX, para conocer si existen diferencias estadísticamente significativas en sus resultados.

Además se realizó un estudio adicional agregando coordenadas internas relativas con el objetivo de conseguir eliminar movimientos consecutivos Norte - Sur o Este-¿Oeste que lógicamente nos

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

dejarían en el mismo lugar, por lo que no tendría sentirlos ejecutarlos. Se establecieron 5 Laberintos de Prueba con obstáculos ubicados aleatoriamente, y por cada laberinto se realizaron 10 ejecuciones independientes con las dos variantes del Algoritmo, obteniendo de cada ejecución: una traza con información del comportamiento del algoritmo, el mejor fitness de todas las evaluaciones y el mejor camino a la salida.

4 RESULTADOS

4.1 Análisis del problema con coordenadas absolutas

Después de conocer el esquema básico de la metodología aplicada para el estudio (5.2) y con los parámetros mencionados en la sección (3.3), se obtuvieron los siguientes resultados (Tabla 1 y Tabla 2), realizando 10 ejecuciones independientes de cada Algoritmo, resolviendo los 5 Laberintos de pruebas.

Tabla 1: Mejor Fitness obtenido con Operador de Cruce SPX.

Laberinto	Ejecuciones										Mediana
	1	2	3	4	5	6	7	8	9	10	
1	630	**626**	645	627	629	651	631	627	639	633	**630.5**
2	**105**	634	109	630	638	648	639	115	633	630	**631.5**
3	628	**626**	644	646	628	640	628	653	628	639	**633.5**
4	661	640	640	633	650	633	633	633	644	**631**	**636.5**
5	655	99	99	97	635	**91**	642	657	632	632	**632**

Tabla 2: Mejor Fitness obtenido con Operador de Cruce UX.

Laberinto	Ejecuciones										Mediana
	1	2	3	4	5	6	7	8	9	10	
1	630	627	627	634	627	627	641	627	627	**627**	**627**
2	628	**59**	637	59	632	57	55	51	67	57	**59**
3	**53**	627	63	630	73	639	639	53	63	639	**350**
4	57	633	627	633	**55**	633	633	627	633	633	**633**

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

5	55	636	635	635	635	69	53	**51**	635	61	**352**

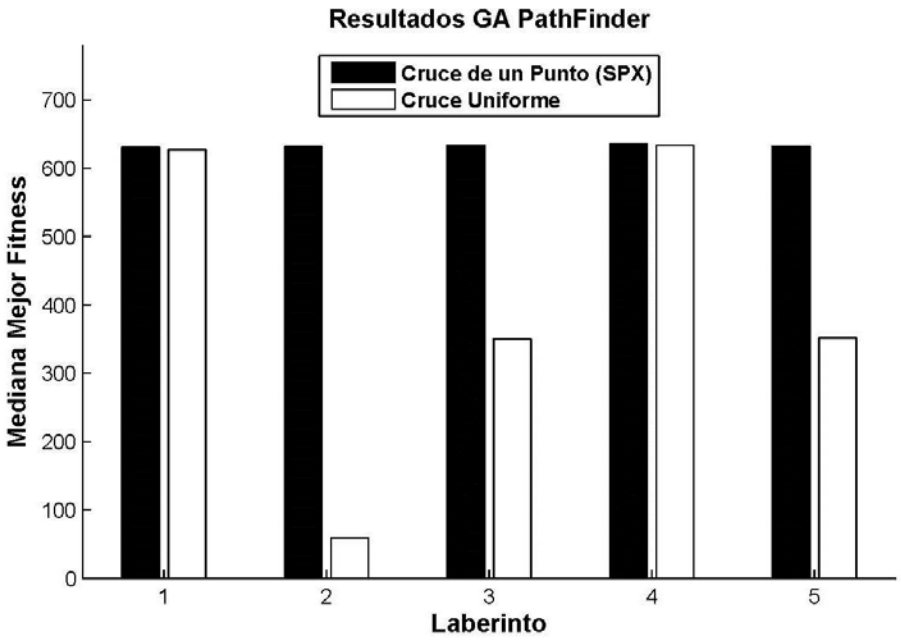


Figura 2: Resultado de las 10 Ejecuciones del algoritmo para cada Laberinto.

Sobre estos resultados se aplicó el test estadístico de *Wilcoxon ranksun*, para analizar comparativamente diferencias significativas entre las soluciones obtenidas de ambos Operadores. En la Tabla 3 se muestran los resultados de este test, donde la columna ***h*** nos indica si se debe o no rechazar la Hipótesis Nula, la cual indica que las medianas de los resultados son iguales, y la columna ***p*** representa el *p-value* retornado del test, el cual es equivalente al *U-test* de *Mann-Whitney*.

Tabla 3: Resultados del test de Wilcoxon.

Test de Wilcoxon Rank sum		
Laberinto	***h***	***p***
1	**0.1151**	**0**
2	**0.00903**	**1**
3	**0.03958**	**1**
4	**0.00427**	**1**
5	**0.20849**	**0**

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

Entonces según estos resultados podemos concluir que para los laberintos: 1 y 5, los resultados de los dos algoritmos serán iguales, no habrá diferencia significativa al usar cualquiera de los dos operadores. Pero en el caso de los laberintos: 2, 3 y 4, si existe una diferencia en los resultados, ya que según los datos del test, se rechaza la hipótesis Nula con un nivel de significancia mayor al 5%, por lo tanto, para obtener mejores soluciones **deberíamos escoger el Algoritmo B con Cruce Uniforme (UX),** puesto que genera más soluciones factibles (que llegan a la salida) y/o soluciones no factibles pero que están mucho más cerca a la salida, en relación a las generadas por el Algoritmo A Cruce Puntual SPX.

Una representación gráfica de estos resultados, puede verse en detalles en la Figura 3, en la que se muestran los mejores fitness obtenidos por cada una de las 10 ejecuciones, y en la que podemos ver que el Algoritmo B con Cruce Uniforme genera, por lo menos en una ejecución, una solución factible que llega a la salida (exceptuando Laberinto 1), ya que el fitness de esta solución está por debajo del valor del "*offset*" (N.M), el cual representa la diferencia entre las soluciones factibles y No factibles. En el caso del Algoritmo A con cruce Puntual, la mayoría de soluciones que genera, principalmente en los laberintos de prueba 3 y 4, tiene un fitness que están por arriba del "offset", lo que indica que mayormente genera soluciones que no llegan a la salida, pero que estas logran estar a una corta distancia de la misma.

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

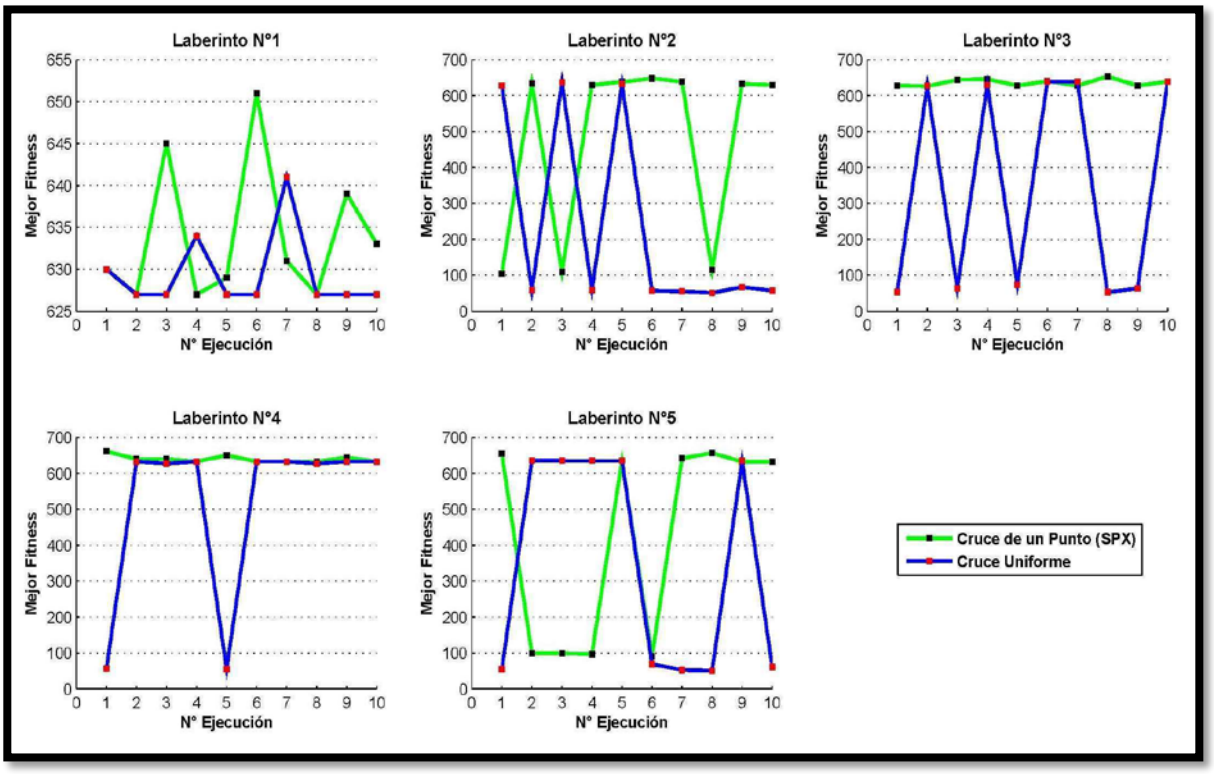


Figura 3: Mejor Fitness encontrado para 10 ejecuciones independientes del Algoritmo.

4.2 Análisis del problema con coordenadas relativas

El mismo análisis y ejecución de resultados de esta Sección, se aplicó con una variante en el conjunto de coordenadas internas, se agregaron tres opcionales que son:

- 5: Avanzar,
- 6: Girar a la Derecha y
- 7: Girar a la Izquierda,

cuyos resultados de ejecución se muestran en las Tablas 4 y 5.

Tabla 4: Mejor Fitness generado con el Operador de Cruce SPX y con Coordenadas relativas.

Laberinto	Ejecuciones	Mediana

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
1	640	643	643	643	632	645	636	**625**	627	627	**638**
2	97	632	648	**89**	119	89	632	630	632	107	**374.5**
3	630	95	641	626	97	641	87	629	**57**	641	**627.5**
4	**627**	633	641	628	632	628	633	633	648	628	**632.5**
5	629	627	629	637	636	**89**	651	629	635	635	**632**

Tabla 5: Mejor Fitness generado con el Operador de Cruce UX y con Coordenadas relativas.

Laberinto	**Ejecuciones**										**Mediana**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
1	627	630	**626**	627	627	627	627	627	627	627	**627**
2	59	65	632	65	57	59	53	**51**	59	67	**59**
3	61	67	**53**	59	628	57	632	53	57	69	**60**
4	633	633	**49**	61	633	59	628	633	633	61	**630.5**
5	51	65	635	635	63	79	**51**	59	71	635	**68**

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

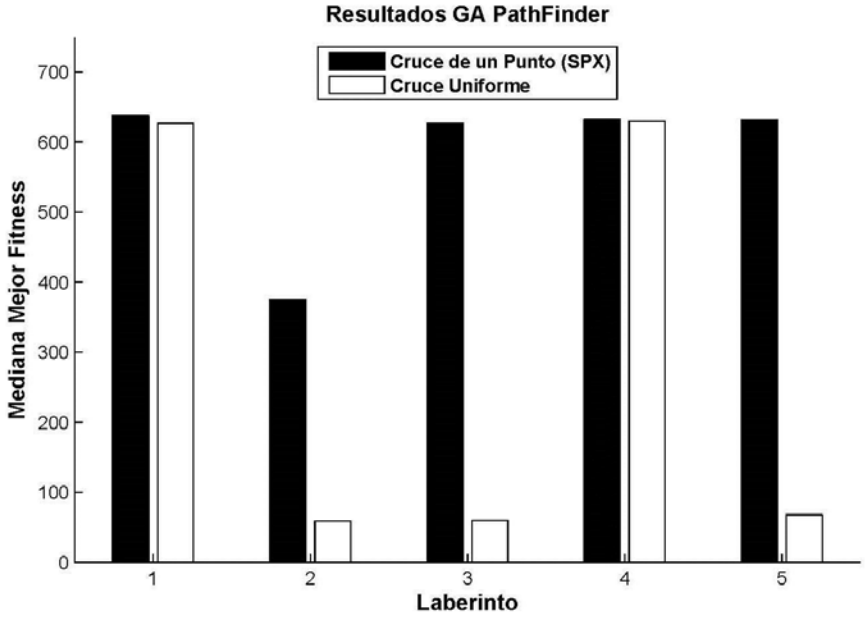


Figura 4: Resultados de las 10 Ejecuciones del algoritmo para cada laberinto, con Coordenadas Relativas.

En la tabla 6 se muestran los resultados del test de *Wilcoxon Ranksum* aplicados sobre los resultados de esta variante del Algoritmo, y en la que podemos ver, que con esta variante SI existen diferencias en los resultados que ofrecen los dos algoritmos, por lo que para obtener mejores soluciones deberíamos escoger el que mejor resultados nos genere, donde según las medianas de 10 ejecuciones independientes para cada laberinto, deberíamos **preferir el empleo del Operador de Cruce Uniforme UX**, puesto que siempre genera soluciones factibles para casos de los Laberintos 2, 3 y 5, y soluciones no factibles pero que llegan mucho más cerca a la salida en el caso del Laberinto 1. Una representación gráfica de estos resultados, puede verse en detalles en la Figura 5.

Tabla 6: Resultados del test de Wilcoxon, con Coordenadas relativas.

Test de Wilcoxon Rank sum		
Laberinto	**h**	**p**
1	**0.00479**	**1**
2	**0.0014**	**1**
3	**0.0137**	**1**
4	**0.25534**	**0**

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

5	**0.01622**	**1**

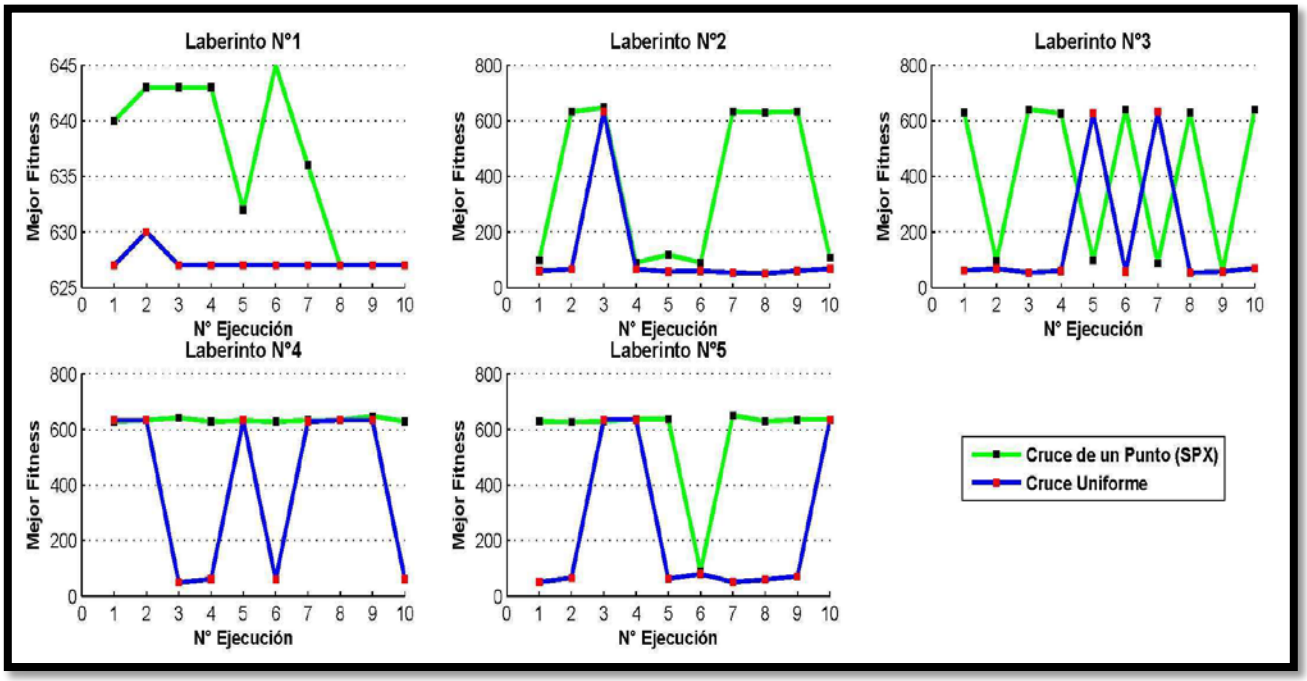


Figura 5: Mejor Fitness encontrado para ejecuciones independientes del Algoritmo, con Coordenadas relativas.

En las siguientes Figuras 6 y 7 se muestra la traza y la Solución más factible obtenida en 10 ejecuciones de los algoritmos, resolviendo cada uno de los 5 laberintos de prueba.

Donde el empleo de Coordenadas Internas Relativas, para la representación de un Camino-Solución, da mejores resultados que los obtenidos con las Coordenadas absolutas.

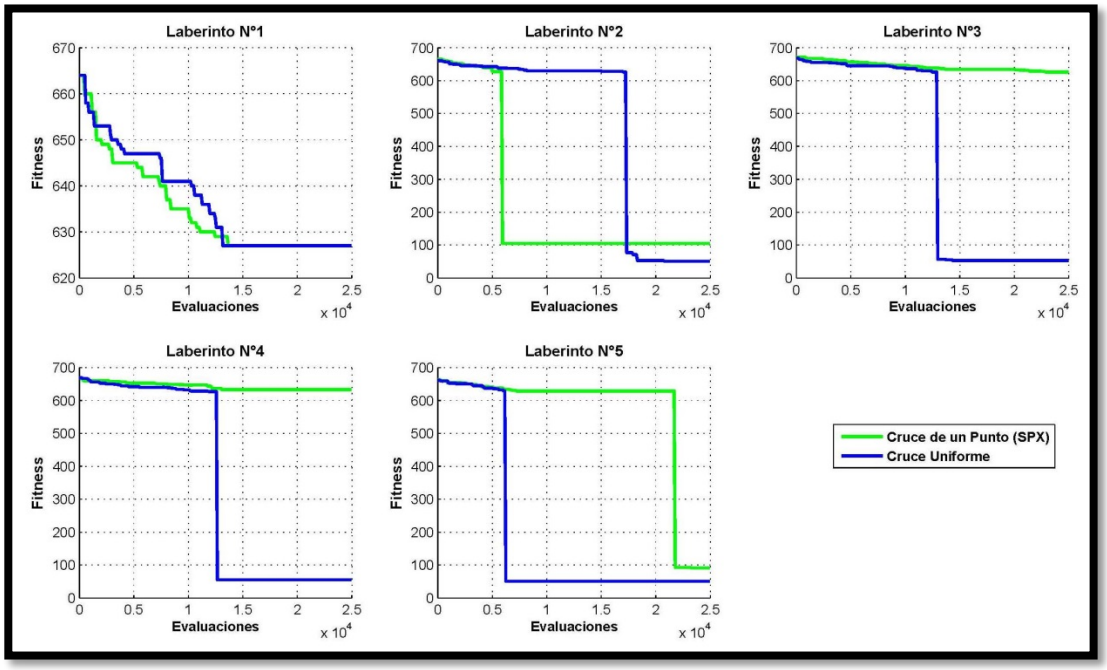


Figura 6: La Mejor Traza de comportamiento del Algoritmo en 10 ejecuciones Independiente, resolviendo los 5 Laberinto de prueba, con Coordenadas absolutas.

Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

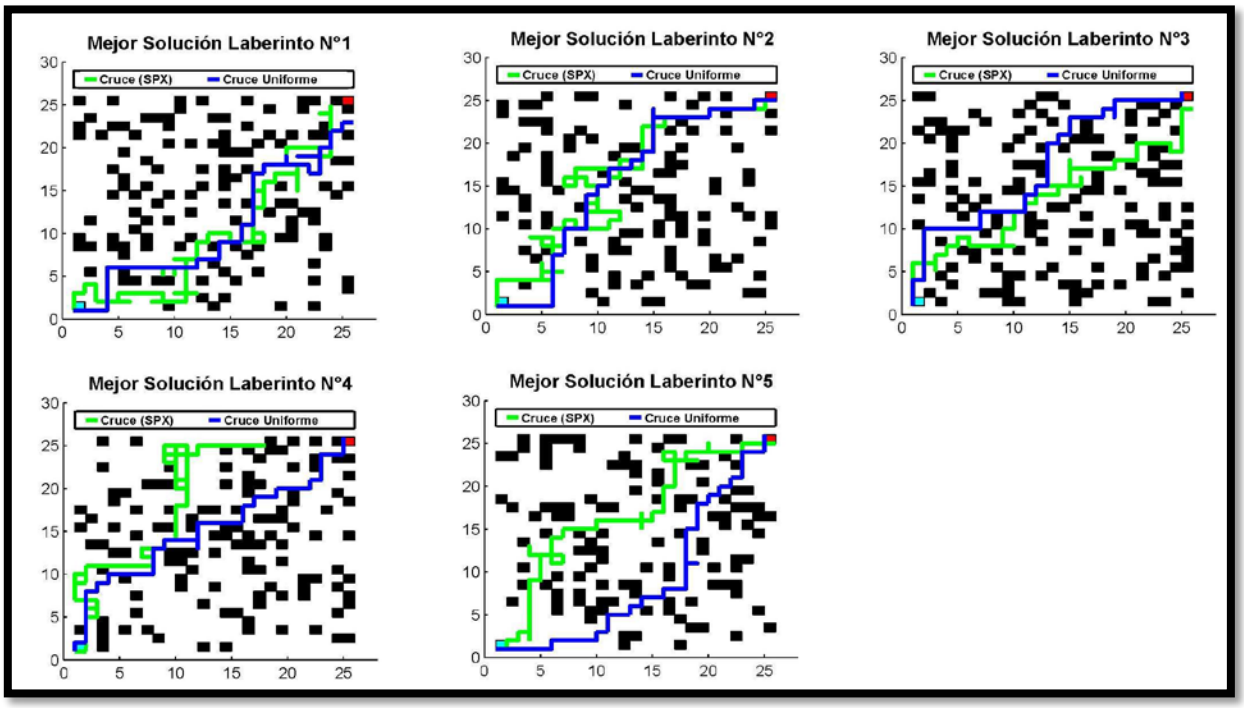
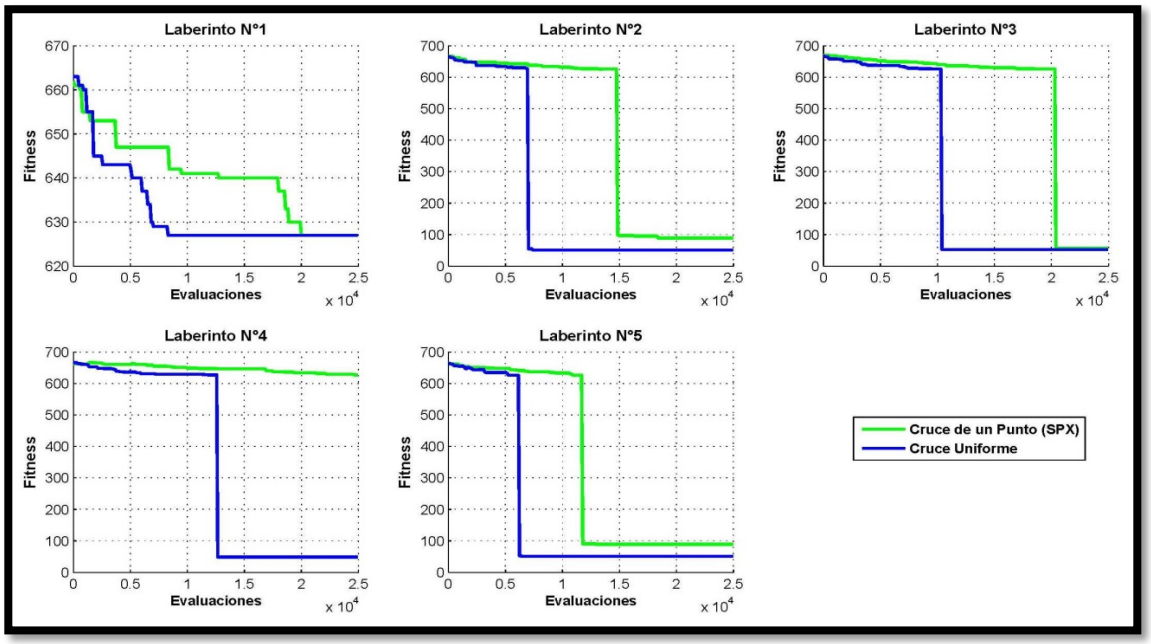


Figura 7: Mejor Solución del Algoritmo en 10 ejecuciones independientes, resolviendo los 5 Laberintos de prueba, con Coordenadas absolutas.



Revista Publicando, 3(8). 2016, 4-23. ISSN 1390-9304

Figura 8: La Mejor Traza de comportamiento del Algoritmo en 10 ejecuciones Independiente, resolviendo los 5 Laberinto de prueba, con Coordenadas Relativas.

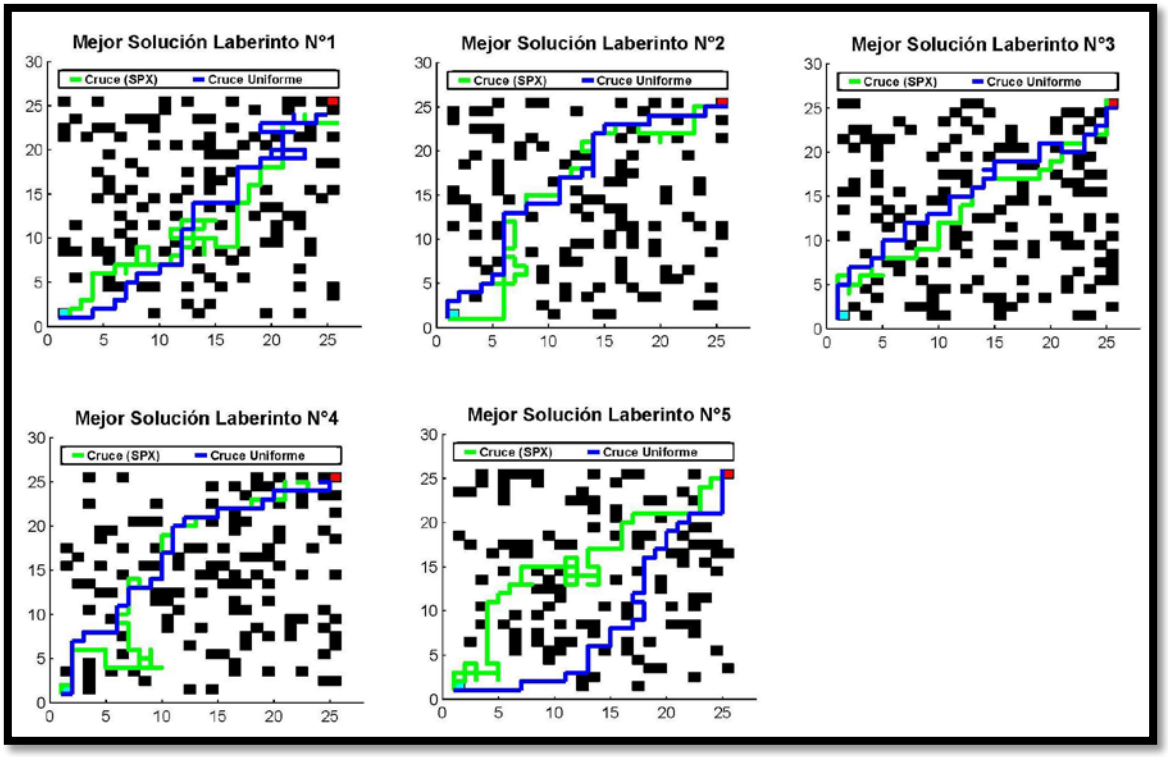


Figura 10: Mejor Solución del Algoritmo en 10 ejecuciones independientes, resolviendo los 5 Laberintos de prueba, con Coordenadas Relativas.

6 CONCLUSIONES

En el presente estudio experimental hemos implementado un algoritmo evolutivo estándar con dos variantes de operadores genéticos de cruce, una con el operador de cruce **Puntual SPX** ("point crossover") y la otra con el operador de cruce **Uniforme UX** ("uniform crossover"), para la resolución del problema de optimización *PathFinder* (Laberinto), con dos tipos de coordenadas que pueden aplicarse para recorrer el camino de salida. Hemos realizado un conjunto de pruebas con 10 ejecuciones independientes de cada algoritmo con estas variantes, y según los resultados obtenidos, las medianas de los fitness de las mejores de las soluciones de cada algoritmo, podemos concluir que la mejor alternativa para resolver el problema será usar la versión del algoritmo

evolutivo con el operador de Cruce Uniforme empleando coordenadas relativas para indicar los movimientos del robot para llegar a la salida. Además los resultados del test estadístico de *Wilcoxon* confirman diferencias significativas a favor de esta variante del algoritmo.

7. BIBLIOGRAFÍA

Alba, E., & Dorronsoro, B. (2008). Introduction to Cellular Genetic Algorithms. In *Cellular Genetic Algorithms* (pp. 3–20). Boston, MA: Springer US. http://doi.org/10.1007/978-0-387-77610-1_1

Back, T., 1996. Evolutionary Algorithms in Theory and Practice, Oxford Press,.

Davis, L., 1991. Handbook of Genetic Algorithms, Van Nostrand Reinhold.

Fogel, D. B. (1994). An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, *5*(1), 3–14. http://doi.org/10.1109/72.265956

Larrañaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., & Dizdarevic, S. (1999). Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. *Artificial Intelligence Review*, *13*(2), 129–170. http://doi.org/10.1023/A:1006529012972

M. Belén, M. José A., & M., J. M. (2003). Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, *7*, 0.

Sumathi, S., & Paneerselvam, S. (2010). *Computational intelligence paradigms: theory & applications using MATLAB*. CRC Press.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning, Addison- Wesley.

Holland, J.H., 1992. Adaptation in Natural and Artificial Systems, MIT Press, Second Edition.