INGENIERÍA Y DESARROLLO

Algoritmo basado en autómatas finitos para la obtención de óptimos globales en problemas combinatorios

Algorith based on finite automata for obtaining global optimum combinatorial problems

Elías D. Niño*, Carlos J. Ardila**

* Magíster en Ingeniería de Sistemas y Computación, Universidad del Norte, docente catedrático, Departamento de Ingeniería de Sistemas. **Correspondencia**: Universidad del Norte, km 5 vía a Puerto Colombia, Barranquilla (Colombia). *enino@uninorte.edu.co*

** Magíster en Ingeniería Industrial, Universidad del Norte, docente de tiempo completo, Departamento de Ingeniería de Sistema. *cardila@uninorte.edu.co*

Número 25
Enero-Junio, 2009
ISSN: 0122-3461

Resumen

En este artículo se propone un Autómata Finito Determinista de Intercambio (AFD - I) que permite modelar el espacio de soluciones factibles a problemas de naturaleza combinatoria, específicamente a problemas asociados con el orden de elementos. Con la estructura AFD - I definida, se diseña e implementa un algoritmo con cuyo uso se obtiene un óptimo global a problemas combinatorios.

El problema que aquí se trata puede ser extrapolado a cualquiera de los siguientes casos: asignación de n procesos a n máquinas que trabajan en paralelo, selección de la ruta óptima en el problema del agente viajero y el problema del bin packing.
Palabras claves: autómata, grafo, optimización combinatoria, óptimo global.

Abstract

This article states a Deterministic Finite Automaton of Exchange (DFA - E). It allows modeling of the space of feasible solutions to combinatorial problems, specifically, the problems associated with the order of elements. With the structure DFA - E defined, we designed and implemented an algorithm that uses it for obtaining a global solution of combinatorial problems.

The problem we treat here can be extrapolated to any of the following: an allocation of n processes machines working in parallel, selecting the optimal route in the traveling salesman problem (TSP) and the problem of Bin Packing.
Key words: automaton, graph, combinatorial optimization, global optimum.

Fecha de recepción: 16 de enero de 2009
Fecha de aceptación: 24 de marzo de 2009

1. INTRODUCCIÓN

Los problemas de naturaleza combinatoria son encontrados a diario: distribución de recursos a procesos, selección de la mejor ruta para la entrega de paquetes, organización de paquetes en bodegas; estos son solo algunos ejemplos de la alta gama de problemas asociados a este género.

La gran dificultad asociada a los problemas anteriormente mencionados es que a medida que crece el número de elementos al momento de la toma de decisiones, el espacio solución crece en orden de factorial, lo cual dificulta la obtención del óptimo global (o los óptimos globales).

1.1. **Autómata finito determinista**

Un AFD (autómata finito deterministas) es una quíntupla [5]:

$$M = (Q, \sum, \delta, q_0, F) \quad (1)$$

Donde:

Q es un conjunto finito de estados.
$\sum$ es el alfabeto finito de entrada.
δ es la función de transición la cual toma un estado y una entrada del alfabeto y determina un nuevo estado.
q_0 es el estado inicial, $q_0 \in Q$
F es el conjunto de estados de finalización.

Una característica de los AFD es que todos los estados que pertenecen a Q, obligatoriamente, tienen transiciones con todos los elementos del alfabeto de entrada $\sum$.

2. AUTÓMATAS FINITOS DETERMINISTAS DE INTERCAMBIOS (AFD – I):

Un **Autómata Finito Determinista de Intercambio** (AFD – I) es un modelo que mediante un grafo no dirigido representa el conjunto de soluciones factibles asociadas a un problema de naturaleza combinatoria. Este modelo soporta los problemas en los cuales el orden de los elementos es importante.

Alrededor de los problemas combinatorios existen estructuras que permiten modelar su espacio de soluciones factibles. Las representaciones mediante redes neuronales para problemas de naturaleza combinatoria con funciones de costo lineales [8] y las que corresponden a representaciones neuronales solo [7] son las comúnmente utilizadas.

Uno de los aportes del modelado mediante un AFD – I es la sencillez y rapidez con que puede ser representado el conjunto de soluciones factibles a un problema combinatorio.

Formalmente un AFD – I se define como una séptupla:

$$M = (Q, \sum, \delta, q_k, F, X_o, f(X)) \quad (2)$$

Donde:

Q es un conjunto de estados finitos, cada estado $q_k \in Q$ tiene una permutación asociada a los elementos del vector original $\overline{X}_o$, el cual se denota como $\overline{X}_k$

Tabla 1
Muestra de algunas permutaciones de un vector $\overline{X}_0$ cuyos elementos son números enteros y el número de la posición corresponde con el valor del elemento

Vector resultante	Acción
$\overline{X}_0 = (1,2,3,4,5)$	Vector original
$\overline{X}_1 = (3,2,1,4,5)$	Vector $\overline{X}_1$, obtenido al intercambiar las posiciones 1 y 3 del vector original.
$\overline{X}_2 = (1,2,3,5,4)$	Vector $\overline{X}_2$, obtenido al intercambiar las posiciones 4 y 5 del vector original.
$\overline{X}_3 = (1,2,3,4,5)$	Vector $\overline{X}_3$, obtenido al intercambiar las posiciones 4 y 5 y posteriormente 1 y 2, lo que es igual al intercambiar las posiciones 1 y 2 del vector $\overline{X}_2$

La estructura de cada estado viene dada por:

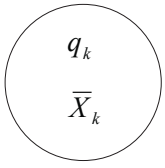


Figura 1. Estructura de un estado de un AFD – I

Debido a que $\overline{X}_k$ tiene asociado un q_k, cada q_k es distinto debido a que proviene de una o más permutaciones de los elementos del vector de entrada $\overline{X}_0$.

El número total de estados en Q será el número total de formas en las que se puedan organizar los elementos del vector de entrada, esto es:

$$\#_{elementos}(Q) = n! \quad \textbf{(3)}$$

Donde n es el número de elementos de $\overline{X}$.

Σ es el alfabeto finito de entrada; contiene todas las posibles combinaciones entre los índices del vector $\overline{X}_0$ agrupándolos en parejas. Las parejas de Σ cumplen la siguiente condición:

$$\sum - \{(k, j) \mid k = 1,2 \ldots n, f = k + 1, k + 2 \ldots k + n\} \quad \textbf{(4)}$$

Donde n es el número de elementos de $\overline{X}$.

Por medio de Σ se representan todos los posibles intercambios que se pueden realizar en un vector de n elementos. Por ejemplo, (1,4) representa intercambiar la posición 1 con la posición 4 del vector. El valor de j siempre es mayor que el valor de k ya que intercambiar 1 con 4 representa lo mismo que intercambiar 4 con 1.

Teorema 1. Dado un AFD – I cuyo vector de entrada es de tamaño n, el número de elementos de Σ viene dado por la expresión:

$$\#_{elementos}(\Sigma) = \frac{n \times (n-1)}{2} \quad \textbf{(5)}$$

Demostración 1:

Por definición se sabe que el número de formas en las que se pueden agrupar n elementos en k subgrupos sin repeticiones es:

$$nC_k = \frac{n!}{(n-k)! \times k!} \quad \textbf{(6)}$$

Por (6), si se agrupan los índices del vector en grupos de a dos se obtiene:

$$nC_2 = \frac{n!}{(n-2)!\times 2!} \quad \textbf{(7)}$$

Aplicando la definición de factorial a 7, se tiene:

$$nC_2 = \frac{(n-2)!\times(n-1)\times n}{(n-2)!\times 2!} \quad \textbf{(8)}$$

Desarrollando operaciones elementales sobre 8, queda:

$$nC_2 = \frac{n\times(n-1)}{2} \quad \textbf{(9)}$$

Y con ello se demuestra el teorema.

δ es la función de transición, que toma un estado q, $q \in Q$, y una tupla de $\sum$, y devuelve un nuevo estado t, $t \in Q$, esto es $\delta(q(i,k)) = t$, en donde $(i,k) \in \sum$.

q_0 es el estado inicial del AFD – I. Este estado contiene al vector inicial.
F es el conjunto de estados de finalización del AFD – I; para estos autómatas, los elementos de Q son los mismos que los elementos de F.
$\overline{X}$ es el vector de entrada; por lo general se asume que $\overline{X}_0$ contiene los elementos en el mismo orden que $\overline{X}$.
$f\left(\overline{X}\right)$ es la función objetivo la cual se desea optimizar.

Vecino de un estado: dos estados q_k, $q_j \in Q$ son vecinos sí y solo sí es posible en una sola transición con un elemento de $\sum$ llegar de q_k a q_j y viceversa, esto es: $V\left(q_k, q_j\right) \Leftrightarrow \exists a \in \frac{\Sigma}{\delta(q_k, a)} = q_j \Lambda \delta\left(q_j, a\right) = q_k$

2.1. Construcción de un AFD – I a partir de un vector de entrada de tres posiciones

Ejemplo 1. Se supone el vector de entrada $\overline{X} = (1,2,3)$ y la función objetivo $f\left(\overline{X}\right) = 0,3 * x_1 + 0,2 * x_2 + 0,1 * x_3$, para la construcción del AFD – I equivalente a estos datos; se procede a hacer lo siguiente:

Establecer el estado inicial: en este caso el estado inicial q_0 contendrá el vector de entrada $\overline{X} = (1,2,3)$; por lo tanto, $\overline{X} = \overline{X_0}$.

Establecer el conjunto ∑: por 6 se sabe que el número de elementos de ∑ es igual a 3 x (3–1). Además, por la definición de ∑, se tiene que ∑ – {(1,2), (1,3), (2,3)}.

Establecer la función δ: Debido a que los AFD – I exigen que los estados tengan transiciones con todos los símbolos del alfabeto, se inicia la generación de vecinos por el estado inicial el cual es q_0. Cada estado vecino nuevo encontrado es nombrado como q_k. Los q_k encontrados son utilizados para generar nuevos vecinos hasta que no haya nuevos. La traza de generación de vecinos se aprecia en la tabla 2.

Tabla 2
Función δ para el AFD – I cuyo vector de entrada es $\overline{X} = (1,2,3)$

$\delta(q_0,(1,2)) = (2,1,3) = X_3 = q_3$	$\delta(q_0,(1,3)) = (3,2,1) = X_5 = q_5$	$\delta(q_0,(2,3)) = (1,3,2) = X_4 = q_4$
$\delta(q_1,(1,2)) = (2,1,3) = X_4 = q_4$	$\delta(q_1,(1,3)) = (3,1,2) = X_2 = q_2$	$\delta(q_1,(2,3)) = (1,2,3) = X_0 = q_0$
$\delta(q_2,(1,2)) = (3,2,1) = X_5 = q_5$	$\delta(q_2,(1,3)) = (1,3,2) = X_1 = q_1$	$\delta(q_2,(2,3)) = (2,1,3) = X_3 = q_3$
$\delta(q_3,(1,2)) = (1,2,3) = X_0 = q_0$	$\delta(q_3,(1,3)) = (2,3,1) = X_4 = q_4$	$\delta(q_3,(2,3)) = (3,1,2) = X_2 = q_2$
$\delta(q_4,(1,2)) = (1,3,2) = X_1 = q_1$	$\delta(q_4,(1,3)) = (2,1,3) = X_3 = q_3$	$\delta(q_4,(2,3)) = (3,2,1) = X_5 = q_5$
$\delta(q_5,(1,2)) = (3,1,2) = X_3 = q_3$	$\delta(q_5,(1,3)) = (1,2,3) = X_0 = q_0$	$\delta(q_5,(2,3)) = (2,3,1) = X_4 = q_4$

Establecer todos los estados de $q \in Q$ como estados de finalización, esto es $q \in Q$ y $q \in F$.

Establecer los valores obtenidos al evaluar $\overline{X_t}$ en la función objetivo. El siguiente paso es calcular el valor de evaluar en la función objetivo cada $\overline{X_t}$, con esto se obtiene la tabla 3.

Tabla 3
Evaluación de los vectores en la función objetivo

Estado	Vector $\overline{X_t}$	Valor $f(\overline{X_v})$
	(1, 2, 3)	1
	(1, 3, 2)	1.1
	(3, 2, 1)	1.4
	(2, 1, 3)	1.1
	(2, 3, 1)	1.3
	(3, 2, 1)	1.4

El AFD – I de la figura 2 representa el autómata generado por este ejemplo. Note que cada estado tiene transición con todos los elementos de $\sum$. Estos representan el intercambio utilizado en el vector $\overline{X_t}$ para alcanzar el nuevo estado $\overline{X_t}$

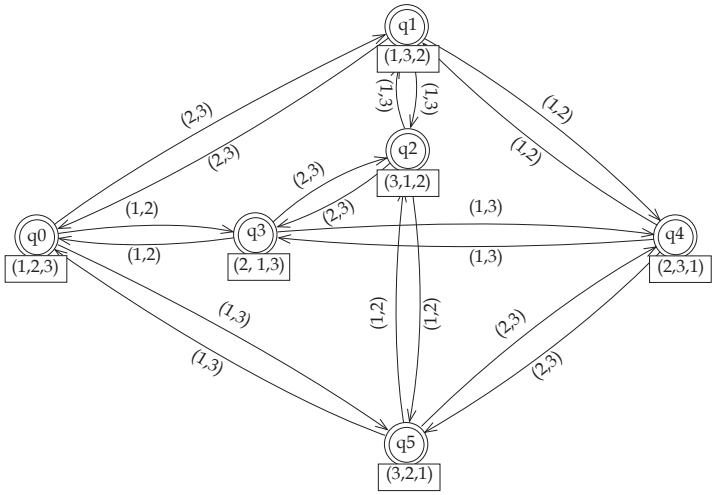


Figura 2. Representación del AFD – I para el vector de entrada $\overline{X} = (1, 2, 3)$

3. ALGORITMO PROPUESTO

Aún cuando los AFD – I facilitan el modelado del espacio de soluciones factibles de un problema combinatorio, no especifica ninguna técnica que permita obtener un óptimo global. Debido a esto, se propone el Algoritmo

Elías D. (AED); es una técnica que obtiene un óptimo global de un problema combinatorio a través del uso de un AFD – I.

La estrategia consiste en encontrar cuál es la permutación adecuada de los elementos del vector de entrada $\overline{X}$ que optimiza la función $f(\overline{X})$. Debido a que Q contiene todas las soluciones del problema asociado, en alguno de los estados $q \in Q$ se encuentra el óptimo global para $f(\overline{X})$. Se podría pensar entonces en la construcción de todo el AFD – I y explorar los estados para ver cuál optimiza la función objetivo. Lo anterior sería viable si la cantidad de elementos a considerar es pequeña. Por otra parte, para grandes cantidades de elementos explorar todos los estados sería algo intratable. Esto es debido a que la proporción de crecimiento del espacio de soluciones factibles (número de estados) respecto a la cantidad de elementos del vector de entrada, es del orden factorial. En la tabla 4 puede apreciarse la proporción para algunos valores de n (número de elementos del vector de entrada).

Tabla 4
Algunos espacios soluciones de acuerdo al tamaño del vector de entrada $\overline{X}$

Tamaño del vector (n)	Tamaño del espacio solución (n!)
5	120
6	720
7	5040
8	40320
9	362880
10	3628800

Pensar en explorar espacios soluciones que contienen 100!, 200! y 500! estados suena casi imposible; para solucionar el problema se proponen los siguientes pasos conocidos como AED:

Parámetros: y $f(\overline{X_\tau})$

σ es el estado actual.
$\overline{X_\tau}$ es el vector asociado a un estado τ.
$f(\overline{X_\tau})$ es el valor de evaluar el vector $\overline{X_\tau}$ en la función objetivo.

Paso 1. $\sigma - q_0$.

Paso 2. $\varphi = f(\overline{X_\tau})$ y $\theta = vac'o$.

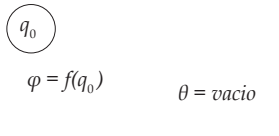
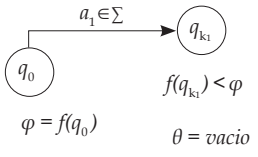
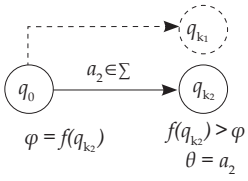
Paso 3. Hacer V=$\delta(\sigma,a_i)$, $a_i \in \sum$. Si $f(\overline{X_v})$ es mejor ϕ entonces $\phi = f(\overline{X_v})$ y $\theta = a_i$

Paso 4. Si existen elementos si $\in \sum$ sin utilizar, volver al paso 3.

Paso 5. Si θ es vacío, σ es un óptimo global. Si no $\sigma - \delta(\sigma,\theta)$ volver al paso 2.

En la tabla 5 se condensan los pasos realizados en la ejecución de este algoritmo.

Tabla 5
Representación del algoritmo EDNR

Se inicia en el primer estado	**Se genera un vecino con un elemento de $\sum$**	**Se genera otro vecino y se guarda el elemento a_2 que lo genera por ser el mejor**
		

Nótese que aun cuando se comparan contra todos los vecinos, no todos los vecinos se generan al mismo tiempo; solo son creados al momento de la comparación.

4. ANÁLISIS Y PRUEBA

El algoritmo se analizó y se probó mediante la función objetivo:

$$f(\overline{x}) = \sum_{i=0}^{n} \frac{i}{100} \times x_i \qquad \textbf{(10)}$$

El problema consiste en encontrar el orden de los elementos de $\bar{x}$ que maximice a 10. Se utilizaron vectores de entrada $\bar{x}$ de tamaño 2, 3, 4… 1000.

Para obtener los óptimos globales de 10 y comparar con los resultados obtenidos mediante AED, se implementó un algoritmo exhaustivo. Mediante esta técnica fue posible obtener los óptimos globales de espacios soluciones menores a 10!.

El vector de entrada con el que se trabajó es de la forma $\bar{x}$–(n, $n-1$, n–1, ..., 2, 1) en donde n es el número de elementos del vector; en la tabla 6 se puede observar el vector de entrada para algunos valores de n.

Tabla 6
Algunos de los vectores utilizados en la prueba

Tamaño del vector	Vector resultante
3	$\bar{x} = (3,2,1)$
10	$\bar{x} = (10,9,8,7,6,5,4,3,2,1)$
15	$\bar{x} = (15,14,13,12,11,10,9,8,7,6,5,4,3,2,1)$

5. RESULTADOS

5.1. Resultados obtenidos por medio de búsqueda exhaustiva y analítica

Los resultados obtenidos por medio de la exploración exhaustiva, así como el número de iteraciones y el tiempo consumido para encontrar los óptimos globales, se condensan en la tabla 7.

Para espacios soluciones mayores a 9 se obtuvieron los óptimos globales de manera analítica. Simplemente se invirtió el orden de los elementos del vector de entrada, que se presentan en la tabla 8.

Tabla 7
Óptimos globales encontrados realizando búsqueda exhaustiva

Valor de N	Duración (Ms)	Iteraciones	Óptimo global
2	0	2	0,05
3	0	6	0,14
4	1	24	0,3
5	1	120	0,55
6	4	720	0,91
7	22	5040	1,4
8	91	40320	2,04
9	686	362880	2,85
10	ERROR	ERROR	ERROR

Tabla 8
Óptimos globales encontrados por medio de análisis

Tamaño del vector	Espacio solución generado	Óptimo global
100	100!	3383,5
200	200!	26867
300	300!	90450,5
400	400!	214134
500	500!	417917,5
600	600!	721801
700	700!	1145784,5
800	800!	1709868
900	900!	2434051,5
1000	1000!	3338335

5.2. **Resultados obtenidos mediante AED**

En la tabla 9 se comparan los resultados de AED con los obtenidos mediante búsqueda exhaustiva.

Tabla 9
Contraste entre los resultados obtenidos mediante búsqueda exhaustiva y los propuestos por AED

BÚSQUEDA EXHAUSTIVA				ALGORITMO EDNR			
N	**Duración (Ms)**	**Iteraciones**	**Óptimo global**	**N**	**Duración (Ms)**	**Iteraciones**	**Óptimo global**
2	0	2	0,05	2	16	2	0,05
3	0	6	0,14	3	0	2	0,14
4	1	24	0,3	4	1	3	0,3
5	1	120	0,55	5	0	3	0,55
6	4	720	0,91	6	1	4	0,91
7	22	5040	1,4	7	0	4	1,4
8	91	40320	2,04	8	0	5	2,04
9	686	362880	2,85	9	0	5	2,85
10	ERROR	ERROR	ERROR	10	1	6	3,85
11	ERROR	ERROR	ERROR	11	1	6	5,06
12	ERROR	ERROR	ERROR	12	5	7	6,5
13	ERROR	ERROR	ERROR	13	1	7	8,19
14	ERROR	ERROR	ERROR	14	0	8	10,15
15	ERROR	ERROR	ERROR	15	1	8	12,4

En la tabla 9 se aprecia como AED mejora en gran medida la búsqueda exhaustiva (que es la única que garantiza óptimos globales) en tiempo y rendimientos, pues, como puede observarse el número de iteraciones es considerablemente menor; esto no es algo sorprendente ya que, por lo general, las metas heurísticas superan a la primera. La idea al realizar este tipo de comparación es observar que los óptimos arrojados por este algoritmo, son globales ya que son los mismos que se encuentran por medio de una búsqueda. En la figura 3 puede apreciarse la mejora en cuanto a tiempo de respuestas en milisegundos.

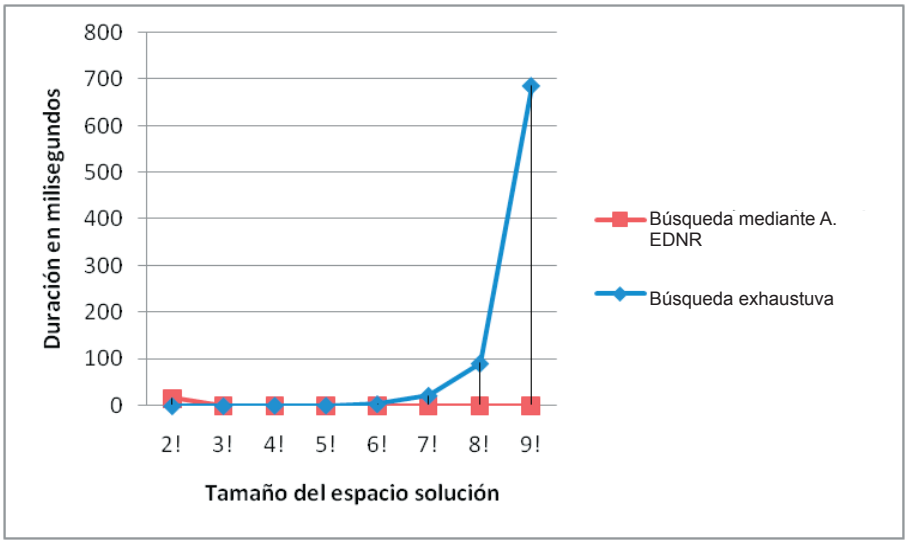


Figura 3. Gráfico de espacio solución contra tiempo consumido para obtener el óptimo global

Se ejecutó el algoritmo con los espacios soluciones de la tabla 8; los resultados obtenidos se sintetizan en la tabla 10: son los óptimos globales como puede apreciarse.

Tabla 10
Óptimos globales encontrados por el algoritmo EDNR

Valor de N	Espacio solución	Duración (Ms)	Duración en segundos	Duración en minutos	Iteraciones	Óptimo global
100	9,3326E+157	95	0,095	0,001583333	51	3383,5
200	200!	1452	1,452	0,0242	101	26867
300	300!	6857	6,857	0,114283333	151	90450,5
400	400!	21160	21,16	0,352666667	201	214134
500	500!	50261	50,261	0,837683333	251	417917,5
600	600!	102892	102,892	1,714866667	301	721801
700	700!	188763	188,763	3,14605	351	1145784,5
800	800!	322899	322,899	5,38165	401	1709868
900	900!	511079	511,079	8,517983333	451	2434051,5
1000	1000!	1053554	1053,554	17,55923333	501	3338335

Obsérvese en la tabla 10 que no fue posible representar numéricamente el espacio soluciones factibles a partir de 200! ya que se necesita toda esta página para poder hacerlo. Por lo anterior se optó por dejar la expresión indicada. En la figura 3 puede apreciarse el tiempo consumido en segundos por el algoritmo para la obtención del óptimo global del problema planteado.

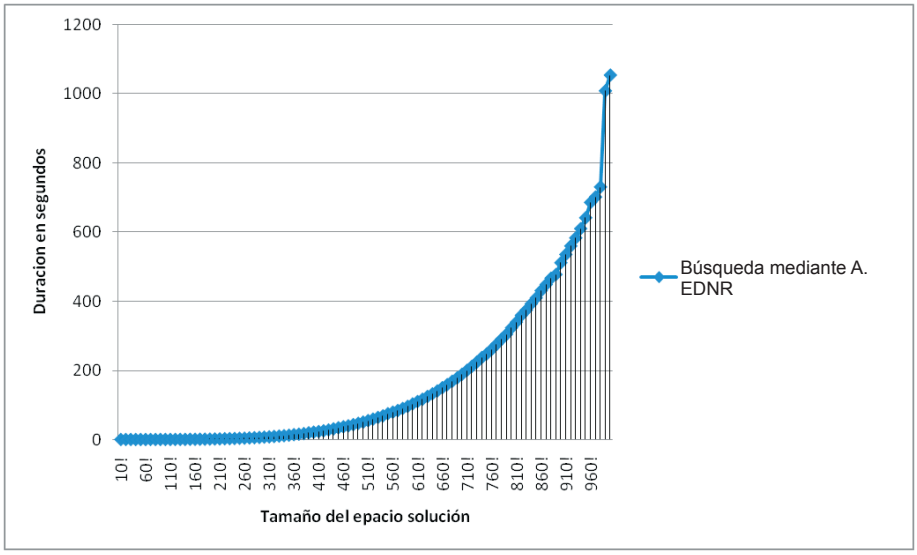


Figura 3. Gráfico de espacio solución contra tiempo consumido en segundos para obtener el óptimo global

6. CONCLUSIONES

AED es un algoritmo que mediante saltos obtiene un óptimo global asociado a un problema combinatorio. Su diseño está basado sobre la teoría de los AFD – I. Los tiempos de respuestas así como el número de iteraciones son cortos en comparación con los enormes espacios de soluciones factibles a los cuales se enfrenta. Su estrategia, principalmente, es obtener un óptimo a partir del cual se nueve saltando hacia donde se encuentre uno mejor que el actual.

Como pudo apreciarse experimentalmente, AED acota un problema combinatorio en orden polinómico, lo cual garantiza que, en un tiempo finito, se

obtendrá una solución a un problema combinatorio sin importar que tan grande sea el espacio de soluciones factibles.

Finalmente, categorizar este algoritmo puede ser un poco complicado. No existe nada similar basado sobre una estructura como un grafo y que esté soportado sobre una teoría como es la de autómatas. Sin embargo, esta estrategia puede considerarse como un método de búsqueda local, que comienza con una solución del problema, y con el cual se va mejorando progresivamente **[9].**

REFERENCIAS

[1] I. Navarrete, *Teoría de autómatas y lenguajes formales*. Murcia: Addison-Wesley Iberoamericana, 2003, pp. 10 – 12.

[2] E. Castillo, *Formulación y resolución de modelos de programación matemática en ingeniería y ciencia*. [E-book] Disponible en: http://departamentos.unican.es/macc/personal/profesores/castillo/descargas.htm.

[3] D. Castro, *Teoría de autómatas, lenguajes formales y gramática*. Alcalá: Addison-Wesley Iberoamericana, 2004, pp. 15 – 17

[4] W. Cook, *Combinatorial Optimization*. United States of America: John Wiley & Sons, 1998.

[5] R. Brena, *Autómatas y lenguajes formales, un enfoque de diseño*. Editorial, 2003. [E-book] Disponible: http://homepages.mty.itesm.mx/rbrena/AyL.html

[6] R. Johnsonbaugh, *Matemáticas discretas*. España: Prentice Hall, 2005.

[7] S. Matsuda, "Yet Another Optimal Neural Representation For Combinatorial Optimization, Neural Networks", *Proceedings of the International Joint Conference*, vol. 2, pp. 873-878, July 2003, Disponible: http://ieeexplore.ieee.org

[8] S. Matsuda, "Optimal Neural Representation For Combinatorial Optimization With Linear Cost Function", *Neural Networks Proceedings*, IEEE World Congress on Computational Intelligence, vol. 2, pp. 1657-1660, May 1998. Disponible: http://ieeexplore.ieee.org

[9] R. Hincapié, C. Ríos y R. Gallego, "Técnicas heurísticas aplicadas al problema del cartero viajante (TSP)", *Sciencia et Técnica*, n.° 24, mar. 2004.