ScienceDirect
Disponible en www.sciencedirect.com
Revista Iberoamericana de Automática e Informática industrial 13 (2016) 430–437 www.elsevier.es/RIAI

Solución al Problema de Secuenciación de Trabajos mediante el Problema del Agente Viajero

G.E. Anaya Fuentes, E.S. Hernández Gress*, J.C. Seck Tuoh Mora, J. Medina Marín

Universidad Autónoma del Estado de Hidalgo, Área Académica de Ingeniería; Carboneras,CP 42183, Mineral de la Reforma Hidalgo, México

Resumen

En este trabajo se estudia el Problema de Secuenciación de Trabajos codificado como un Problema de Agente Viajero y resuelto mediante Algoritmos Genéticos. Se propone un Algoritmo Genético en donde se comparan dos tipos de selección: por torneo y por ruleta. Se realizan diferentes pruebas para la solución del Problema del Agente Viajero con los dos tipos de selección bajo diferentes parámetros: número de individuos, número de iteraciones, probabilidad de cruce y probabilidad de mutación; a partir de estos se seleccionan los parámetros y el tipo de selección. Posteriormente se codifica al Problema de Secuenciación como un Problema del Agente Viajero. La propuesta se presenta mediante la aplicación a diferentes ejemplos del Problema de Secuenciación de Trabajos y la comparación con los resultados obtenidos en la literatura.

Palabras Clave:
algoritmos eficientes, sistemas industriales de producción, problemas de optimización, problema de agente viajero.

1. Introducción

El problema del agente viajero (PAV) es un problema de optimización combinatoria en la que una persona visita sólo en una ocasión cada una de las ciudades y se regresa al punto de partida; se deberá localizar la ruta que presente la distancia más corta y a ésta se la conoce como la ruta óptima (Moon, 2002). A medida que crece el número de ciudades a visitar por el agente viajero, no es factible resolverlo por programación entera debido a que el tiempo de solución computacional crece de forma exponencial de acuerdo con el número de ciudades visitadas; a este tipo de problemas se los conoce como no polinomiales (NP) (Maldonado, 2010).

En esta investigación se modeló el PAV a través de programación entera analizando hasta dónde fue factible de resolverse por este método. Posteriormente el problema se resolvió mediante algoritmos genéticos (AG), el cual fue probado con algunos ejemplos en donde la solución fue encontrada a través de programación entera. También se utilizó el AG para resolver algunos ejemplos en donde la solución no pudo ser encontrada por programación entera debido a que el número de restricciones crece de forma exponencial al aumentar el número de ciudades visitadas.

En el PAV resuelto con AG se realizan pruebas para determinar los mejores operadores y parámetros; los operadores son cruce, mutación, selección etc. y los parámetros son: tamaño de la población, criterio de paro, entre otros. Existen muchos trabajos en la literatura que han logrado resolver el PAV con AG, sin embargo, no especifican la codificación utilizada. La primera aportación de este trabajo es la codificación del PAV a través del AG, y la selección de los mejores parámetros y operadores, especialmente la comparación del operador selección en dos tipos: torneo y ruleta.

La segunda aportación de esta investigación es utilizar los resultados del PAV resuelto por AG para resolver el Problema Secuenciación de Trabajos (PST), un problema combinatorio que se sigue estudiando hasta la fecha y tiene utilidad práctica en la industria manufacturera (Tamilarasi y Anantha, 2010). Las industrias requieren métodos sencillos que les permitan tener buenas soluciones, en esta investigación se especifica el proceso para transformar un PST a un PAV y tener buenas soluciones que les permitan tomar decisiones a las empresas en tiempos razonables.

La organización del trabajo es como sigue, en la sección 2 se habla del PAV, sus antecedentes, y como se resuelve el PAV mediante AG, resaltando los parámetros y operadores encontrados. En la sección 3 se aborda el PST y se explica su modelación a través de un PAV y su optimización con AG**,** mostrando los resultados obtenidos. Por último en la sección 4 se muestran las conclusiones.

2. El Problema del Agente Viajero

En la actualidad los métodos convencionales para la solución del PAV, tales como la programación entera, reportan una

* Autor en correspondencia.
Correos electrónicos: gustavoerick_anay@hotmail.com (G.E. Anaya Fuentes), evaselenehg@hotmail.com (E.S. Hernández Gress), juanseck@gmail.com (J.C. Seck Tuoh Mora), jmedina@uaeh.edu.mx (J. Medina Marín)

http://dx.doi.org/10.1016/j.riai.2016.07.003

frontera en cuanto al tiempo computacional, que permitan reducir tiempos de producción en la industria manufacturera mediante la secuenciación de trabajos (Tamilarasi y Anantha, 2010). A través de la resolución del PAV con AG se busca un medio para solucionar el PST.

2.1. Antecedentes del PAV

El Agente Viajero ha sido estudiado ampliamente con heurísticos, ver por ejemplo, los trabajos de Dorigo (1997) con colonia de hormigas, Cerny (1985) con el Método Montecarlo; Jog et al. (1991), Chattarjee et al. (1996), Larrañaga et al. (2000), Moon et al. (2002), Fogel (1998), entre otros con AG obteniendo muy buenos resultados. Cabe resaltar, que el objetivo principal de este trabajo no es encontrar un método mejor para resolver el PAV, sino uno que pueda ayudarnos a resolver posteriormente el PST en un tiempo computacional razonable.

2.2. Definición del PAV

El problema del agente viajero puede ser definido formalmente como sigue (Buthainah, 2008). Sea una red $G = [N, A, C]$ que está definida por un conjunto de N nodos, A el conjunto de arcos, y $C = [c_{ij}]$ la matriz de costos. Eso es, c_{ij} es el costo de moverse desde el nodo i al nodo j. El PAV requiere un ciclo Halmiltoniano en G de mínimo costo, siendo un ciclo Hamiltoniano, uno que pasa a través de cada nodo i exactamente una vez. El PAV es un problema de permutación que tiene como objetivo encontrar el camino de menor longitud o de costo mínimo en un grafo no dirigido que representa las ciudades o nodos a ser visitados. El PAV inicia en un nodo, visitando a todos los nodos uno a uno para finalmente regresar al nodo inicial. De tal forma que se deben formar rutas y no subrutas, por ejemplo, si existieran 6 ciudades una ruta de ciclo Hamiltoniano seria *2-4-5-3-6-1-2*, regresando nuevamente al nodo 2 y una subruta seria *2-4-5-2* y *3-6-1-3*, en esta última no se han visitado todos los nodos cuando se regresa al punto de partida.

El PAV puede resolverse mediante programación entera (Wagner, 1975) en este siendo $1,2,3,...,N$ el conjunto de ciudades a visitar . Para $i \neq j$, C_{ij} es la distancia desde la ciudad i hasta la ciudad j y sea $C_{ii} = M$, donde M es un número muy grande en relación con las distancias del problema. Las variables de decisión son $X_{ij} = 1$, si la solución al PAV va de la ciudad i a la ciudad j, $X_{ij} = 0$, en caso contrario. Entonces la solución del problema se encuentra al resolver

$$\min Z = \sum_i \sum_j C_{ij} X_{ij} \tag{1}$$

s.a.

$$\sum_{j=1}^{N} X_{ij}, \forall \quad j = 1,2,...,N \tag{2}$$

$$\sum_{i=1}^{N} X_{ij}, \forall \quad i = 1,2,...,N \tag{3}$$

$$\begin{aligned} &U_i - U_j + NX_{ij} \leq N-1 \quad \forall \quad i \neq j, \\ &i, j = 1,2,...,N \quad U_i, U_j \geq 0 \quad X_{ij} = \{1,0 \end{aligned} \tag{4}$$

La función objetivo (1) proporciona la distancia total de los arcos incluidos en un tour. Las restricciones en (2) dan certeza de que al salir de una ciudad sólo es posible llegar a una sola ciudad o nodo. Las restricciones en (3) aseguran que abandona una sola ciudad o nodo cada vez. Las restricción (4) es clave para el planteamiento, mediante ésta, se asegura que cualquier conjunto de X_{ij} que forma un subruta será no factible, y cualquier conjunto de X_{ij} que forma una ruta será factible.

2.3. Antecedentes de los AG

Los AG buscan proporcionar una calificación para la supervivencia en la búsqueda de un buen resultado. En la naturaleza, los individuos más aptos tienen más probabilidades de sobrevivir y aparearse, por lo que la próxima generación debe ser mejor. Esta misma idea se aplica al PAV, en donde primero se tratan de encontrar las regiones de soluciones y luego combinar a las más aptas para crear una nueva generación de soluciones que deben ser mejor que la generación anterior (Holland, 1992).

También incluyen un elemento mutación al azar para evitar la degradación entre los elementos, evitando caer en el óptimo local. Una codificación adecuada encuentra la solución al problema de manera aleatoria utilizando el concepto de que cada posible solución tiene una codificación única (Chambers, 1998). La población inicial se genera de manera aleatoria o introduciendo en esta generación inicial una o varias soluciones encontradas previamente con otras técnicas. Para la evaluación se toma una simple solución como parámetro y devuelve un número que indica lo buena que la solución es. Por sí mismo el número devuelto no significa nada, solamente al ser comparado con otro valor podemos seleccionar a la mejor solución. Los operadores utilizados por los algoritmos genéticos son selección, cruce y mutación y la estrategia consiste en encontrar los parámetros más adecuados para solucionar el problema (Goldberg, 1989).

2.4. Experimentación del PAV

A continuación se explica cómo se realizó la experimentación con la matriz de distancias del problema propuesto por Winston (2005), que se muestra a continuación, **C1** representa la ciudad 1, **C2** es ciudad 2, **C3** es ciudad 3, y así sucesivamente:

Tabla 1: Matriz de distancias

	C1	C2	C3	C4	C5
C1	M	132	217	164	58
C2	132	M	290	201	79
C3	217	290	M	113	303
C4	164	201	113	M	196
C5	58	79	303	196	M

Se soluciona el problema de la Tabla 1, debido a que al tener pocos nodos o ciudades es sencillo de explicar, este ejemplo se soluciona con programación entera, y con algoritmos genéticos utilizando dos tipos de selección. En la búsqueda de la solución a través de programación lineal entera se tiene que la función objetivo específica para este problema está en (5) las restricciones en (6), (7) y (8) y las de no negatividad en (9).

$$\begin{aligned}\min Z = {} & MX_{11} + 132X_{12} + 164X_{14} + 58X_{15} \\ & + 132X_{21} + MX_{22} + 290X_{23} + 290X_{32} + \\ & + MX_{33} + 113X_{34} + 305X_{35} + 164X_{41} \\ & + 201X_{42} + 113X_{43} + MX_{44} + 196X_{45} \\ & + 58X_{51} + 79X_{52} + 303X_{53} + 196X_{54} \\ & + MX_{55}\end{aligned} \tag{5}$$

s.a.

$$\begin{aligned}& X_{11} + X_{12} + X_{13} + X_{14} + X_{15} = 1 \\ & X_{21} + X_{22} + X_{23} + X_{24} + X_{25} = 1 \\ & X_{31} + X_{32} + X_{33} + X_{34} + X_{35} = 1 \\ & X_{41} + X_{42} + X_{43} + X_{44} + X_{45} = 1 \\ & X_{51} + X_{52} + X_{53} + X_{54} + X_{55} = 1\end{aligned} \tag{6}$$

$$\begin{aligned}& X_{11} + X_{21} + X_{31} + X_{41} + X_{51} = 1 \\ & X_{12} + X_{22} + X_{32} + X_{42} + X_{52} = 1 \\ & X_{13} + X_{23} + X_{33} + X_{43} + X_{53} = 1 \\ & X_{14} + X_{24} + X_{34} + X_{44} + X_{54} = 1 \\ & X_{15} + X_{25} + X_{35} + X_{45} + X_{55} = 1\end{aligned} \tag{7}$$

$$\begin{aligned}& U_2 - U_3 + 5X_{23} \le 4 \\ & U_2 - U_4 + 5X_{24} \le 4 \\ & U_2 - U_5 + 5X_{25} \le 4 \\ & U_3 - U_2 + 5X_{32} \le 4 \\ & U_3 - U_4 + 5X_{34} \le 4 \\ & U_3 - U_5 + 5X_{35} \le 4 \\ & U_4 - U_2 + 5X_{42} \le 4 \\ & U_4 - U_3 + 5X_{43} \le 4 \\ & U_4 - U_5 + 5X_{45} \le 4 \\ & U_5 - U_2 + 5X_{52} \le 4 \\ & U_5 - U_3 + 5X_{53} \le 4 \\ & U_5 - U_4 + 5X_{54} \le 4\end{aligned} \tag{8}$$

$$U_i, U_j \ge 0 \quad \forall \quad i, j \quad X_{ij} = \{1,0 \tag{9}$$

La función objetivo *(5)* proporciona los costos de los arcos que podrían ser incluidos en la ruta que se trata de minimizar. Las restricciones del tipo *(6)* y *(7)* son simplemente las que tratan de buscar la ruta más corta y se plantean como un problema de asignación; en donde las *(6)* dan certeza de que al salir, sólo es posible llegar a una ciudad cada vez, las *(7)* aseguran que uno abandona la ciudad sólo una vez. Con la finalidad de evitar que una ciudad tenga como destino así misma se asigna como costo a las variables de decisión X_{ii} un valor relativamente grande al que llamamos *M*. Con las restricciones del tipo (*8)* se impide la creación de subrutas para el resultado del PAV. Al resolverlo mediante el método de ramificación y acotamiento se obtuvieron los siguientes resultados:

Tabla 2. Solución a través de ramificación y acotamiento, Costo de 668 unidades

(Nodo i, Nodo j)	Costo
(1,5)	58
(2,4)	201
(3,1)	217
(4,3)	113
(5,2)	79

En donde la ruta óptima es de 668 unidades bajo la secuencia 1-5-2-3-4-1. En la Figura 1 se muestra que el ciclo es cerrado, es decir que la secuencia 5-2-4-3-1-5, la 2-4-3-1-5-2, la 4-3-1-5-2-4, la 3-1-5-2-4-3 son equivalentes, es decir poseen las mismas 668 unidades.

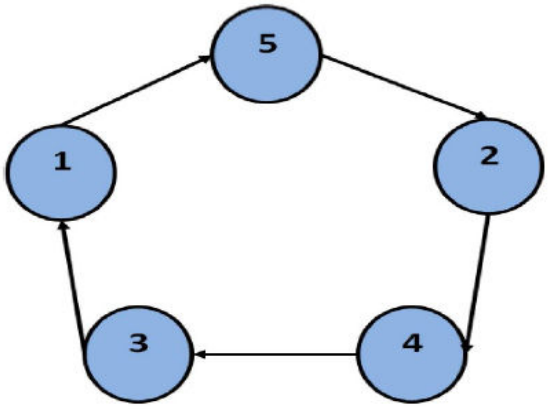

Figura 1. Secuencia para el Problema de 5 ciudades

A medida que el número *n* de ciudades a visitar en el PAV crece, la complejidad computacional lo hace también, hasta un punto en el cual sería necesario esperar demasiado tiempo para obtener un resultado; por esta razón, es conveniente utilizar un heurístico como los AG, que permitan encontrar un buen resultado en un tiempo computacional razonable. Para efecto de explicar cómo funciona el algoritmo, se experimentó nuevamente con el problema de 5 ciudades, a continuación se explica brevemente como trabajan cada una de las operaciones del AG.

Con la finalidad de iniciar el funcionamiento del algoritmo, es necesario la matriz cuadrada que represente el costo de la distancia al viajar de la ciudad *i* a la ciudad *j*. Adicionalmente, se genera una población inicial de un determinado número de individuos con rutas aleatorias, más adelante se específica como se elige el número de individuos. Por ejemplo, en la Tabla 3 se muestran 10 individuos con 5 alelos (cada alelo es una ciudad) y su respectiva aptitud (fitness) o distancia de viajar en la ruta.

Tabla 3. Rutas aleatorias.

Individuo	Ruta	Costos	Aptitud
1	1-2-3-4-5	132+290+113+196+58	789
2	5-3-1-2-4	303+217+132+201+196	1049
3	3-4-5-2-1	113+196+79+132+217	737
4	2-4-5-1-3	201+196+58+217+290	962
5	3-1-4-2-5	217+164+201+79+303	964
6	5-2-3-1-4	79+210+217+164+196	946
7	2-1-5-3-4	132+58+303+113+201	807
8	3-2-1-5-4	290+132+58+196+113	789
9	1-2-5-4-3	132+79+196+113+217	737
10	5-4-1-3-2	196+164+217+290+79	946

Para realizar la selección por torneo se generan dos permutaciones aleatorias de tamaño igual al número de individuos; por ejemplo, la primer permutación *P1=6-3-7-8-5-1-2-4-9-10*, la segunda permutación *P2=2-4-9-10-6-3-7-8-5-1*, lo que significa que el individuo *6* competirá con el *2,* el *3* con el *4* y así sucesivamente, los vencedores son aquellos que presenten una menor aptitud (menor distancia en las rutas). El resultado puede apreciarse en la Tabla 4.

Tabla 4: Torneo

Competidores	Vencedor	Ruta	Aptitud
6,2	6	5-2-3-1-4	946
3,4	3	3-4-5-2-1	737
7,9	9	1-2-5-4-3	737
8,10	8	3-2-1-5-4	789
5,6	6	5-2-3-1-4	946
1,3	3	3-4-5-2-1	737
2,7	7	2-1-5-3-4	807
4,8	8	3-2-1-5-4	789
9,5	9	1-2-5-4-3	737
10,1	1	1-2-3-4-5	789

Se ordenan estos individuos de menor a mayor de acuerdo a su aptitud (ver Tabla 5).

Tabla 5: Población ordenada

Individuo	Ruta	Aptitud
1	1-2-5-4-3	737
2	1-2-5-4-3	737
3	3-4-5-2-1	737
4	3-4-5-2-1	737
5	3-2-1-5-4	789
6	3-5-2-1-4	791
7	1-2-3-4-5	789
8	2-1-5-3-4	807
9	5-2-3-1-4	946
10	5-2-3-1-4	946

Para el cruce se genera aleatoriamente un arreglo, los renglones son el número de individuos entre dos, las columnas siempre son 2. En el ejemplo son 5 x 2. La columna 1 son los *Posibles Padres 1* y la columna 2 los *Posibles Padres 2*. Se genera una probabilidad de cruce, por ejemplo, 0.6 y se genera un aleatorio para cada posibilidad, si este aleatorio es menor al valor predeterminado (0.6) se realiza el cruce entre la pareja formada por el primer renglón de la columna *Posibles Padres 1* y el primer elemento de la segunda columna los *Posibles Padres 2;* en otro caso no. Para realizar el ejemplo específico, se supone un aleatorio es de 0.5 y los elementos guardados en las columnas son el 9 y 1. Una vez seleccionados, se generan dos puntos de corte para dividir a ambos padres en tres segmentos, por ejemplo, la posición dos y cuatro. En donde el *Hijo 1* se forma con el primer segmento del *Padre1* y segundo segmento del *Padre 2*. Finalmente el tercer segmento del *Hijo 1* se forma con el tercer segmento del *Padre 1*. El *Hijo 2* es formado con los segmentos restantes como se muestra en la Figura 2.

Padre 1=	5	2	3	1	4
Padre 2=	1	2	5	4	3
Hijo 1=	5	2	5	4	4
Hijo 2=	1	2	3	1	3

Figura 2. Cruce.

Para terminar con el cruce, se identifica la existencia de valores faltantes en cada individuo debido a que esta condición genera subrutas. El *Hijo 1* tiene los elementos faltantes *1-3* y el *Hijo 2* los elementos *5-4*. Esto se corrige al colocar los valores faltantes en las posiciones en las que se encuentran los valores repetidos, obteniéndose el cruce final.

Hijo 1=	5	2	3	1	4	=Individuo 9
Hijo 2=	1	2	5	4	3	=Individuo1

Figura 3. Cruce, corrección de subrutas

El proceso se repite para cada una de las parejas del mismo renglón guardadas en las columnas *Posibles Padres 1* y *Posibles Padres 2*, generándose la siguiente población (Tabla 6*)*:

Tabla 6: Población después del Cruce

Individuo	Ruta	Aptitud
1	1-2-5-4-3	737
2	4-2-1-5-3	807
3	3-4-5-2-1	737
4	3-4-5-2-1	737
5	3-2-1-5-4	789
6	3-5-2-1-4	791
7	1-2-3-4-5	789
8	2-1-5-3-4	807
9	5-2-3-1-4	946
10	5-2-3-1-4	946

El último operador es la mutación, la cual se realiza seleccionando una probabilidad de mutación, en este caso de 0.1, y generando aleatorios para cada individuo. Después de generar aleatorios el único que resultó ser menor a 0.1 fue el individuo *2*. Para este individuo en particular, se genera puntos de corte por ejemplo la posición *2* y *5* que se intercambian.

Individuo2=	4	2	1	5	3
Mutación=	4	3	1	5	2

Figura 4. Mutación

Este elemento se inserta en la población resultante del cruce y esto se hace durante el número de iteraciones predefinido. En este caso el elemento obtenido es la misma solución óptima a través de la programación entera *4-3-1-5-2* que es de 668 unidades.

La selección por ruleta se explica a partir los datos de la Tabla 3, y se realiza ordenando primeramente la población con respecto a la aptitud, de menor a mayor, el mejor individuo se pasa directamente a la nueva población. Para buscar los otros individuos de la población es necesario hacer 3 cálculos (ver Tabla 7); A1 es la resta de la mayor aptitud menos la aptitud de cada individuo, estas cantidades de A1 se suman y para formar A2, cada aptitud es dividida entre esa suma; A3 es simplemente la función acumulada de A2.

Tabla 7: Selección por Ruleta

Individuo	Ruta	Aptitud	A1	A2	A3
3	3-4-5-2-1	737	312	0.17687	0.17687
9	1-2-5-4-3	737	312	0.17687	0.35374
1	1-2-3-4-5	789	260	0.14739	0.50113
8	3-2-1-5-4	789	260	0.14739	0.64852
7	2-1-5-3-4	807	242	0.13718	0.78571
6	5-2-3-1-4	946	103	0.05839	0.84410
10	5-4-1-3-2	946	103	0.05839	0.90249
4	2-4-5-1-3	962	87	0.04932	0.95181
5	3-1-4-2-5	964	85	0.04818	1
2	5-3-1-2-4	1049	0	0	1

Para cada individuo de la nueva población se genera un aleatorio y se compara con respecto a A3, por ejemplo si el aleatorio es 0.9 se elige el individuo 6, *5-2-3-1-4* porque 0.9 es menor que 0.90249. Generando los aleatorios 0.0298, 0.9419, 0.4028, .0733, 0.6490, 0.7075, 0.2942, 0.5115, 0.3583 se genera la población que se aprecia en la Tabla 8.

A partir de aquí el cruce y la mutación se hacen de la forma anteriormente explicada. Tanto en este problema de 5 ciudades que se explica, como en otros con más ciudades que se experimentaron la selección por torneo tuvo mejor desempeño que la selección por ruleta encontrando mejores soluciones o la misma pero en menores generaciones, ver Tabla 9. Se muestra en la Figuras 5 y 6 una comparación el promedio y el menor encontrado en ambos tipos de selección.

De estos ejemplos se puede concluir que la selección por torneo es la mejor, ya que es necesario considerar un número grande de individuos para la población, con un número moderado de iteraciones (en una proporción de 10 individuos para una iteración aproximadamente). La probabilidad de cruce que mejor funciona es la de 0.6 y una probabilidad de mutación baja *(1/n)*, donde *n* es el número de individuos (ver Tabla 9).

Tabla 8: Población utilizando la Selección por Ruleta

Individuo	Ruta	Aptitud
3	3-4-5-2-1	737
3	3-4-5-2-1	737
10	5-4-1-3-2	946
1	1-2-3-4-5	789
3	3-4-5-2-1	737
7	2-1-5-3-4	807
7	2-1-5-3-4	807
9	1-2-5-4-3	737
8	3-2-1-5-4	789
1	1-2-3-4-5	789

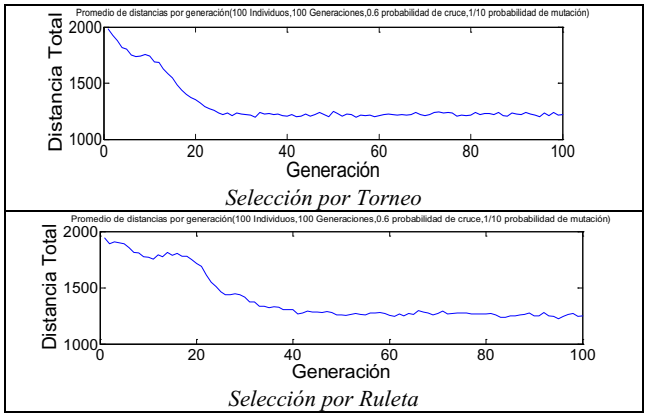

Figura 5. Comparación del promedio de distancias encontrado en la selección por torneo y por ruleta.

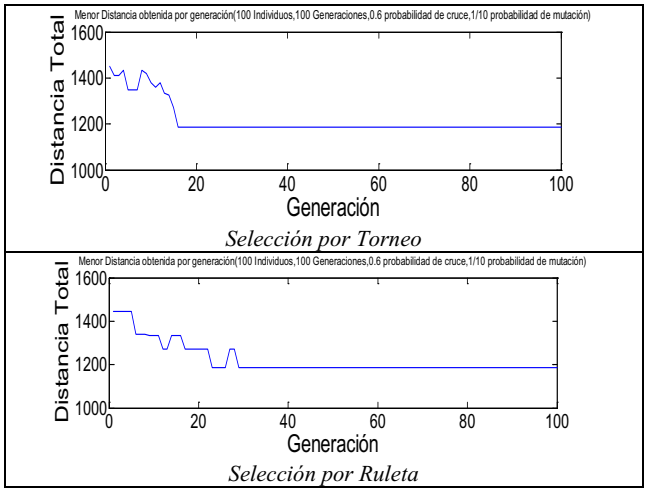

Figura 6. Comparación del menor encontrado en la selección por torneo y por ruleta.

En la Tabla 9, la columna 1 es el número de ciudades, a es para el problema de 5 ciudades (Winston, 2005) y b es para el de 280 (Gerhard,2006); la columna 2 muestra la selección 1 es por torneo y 2 por ruleta; la columna 3 es la probabilidad de cruza, la columna 4 es la mutación utilizada, la columna 5 muestra el tiempo en

segundos en que se obtuvo la solución; por último en las columnas 6 y 7 se muestra el mejor encontrado en las experimentaciones y mejor encontrado en la literatura, respectivamente.

Tabla 9: Experimentación del AG con Probabilidad de cruce de 0.6 y probabilidad de mutación 1/n

1	2	3	4	5	6	7
5[a]	1	0.6	0.1	2	668	668
	2	0.6	0.1	2	668	668
10[b]	1	0.6	0.1	3	1185	1185
10[b]	2	0.6	0.1	3	1185	1185
10[b]	1	0.6	0.1	3	7392	8914
10[b]	2	0.6	0.1	3	7392	8914
15[a]	1	0.6	0.05	5	1692	1513
20[b]	1	0.6	1/20	7	1700	1688
20[b]	2	0.6	1/20	7	1854	1688
21[b]	1	0.6	0.05	8	2042	2042
280[b]	1	0.6	0.1	420	6213.25	2579
280[b]	2	0.6	0.1	900	7968.55	2579
280[b]	1	0.6	0.1	2220	6750.22	2579
280[b]	2	0.6	0.1	13680	8686.33	2579
280[b]	1	0.6	0.1	4200	5656.54	2579
280[b]	2	0.6	0.1	13980	8686.33	2579
280[b]	1	0.6	0.1	8400	6424.03	2579
280[b]	1	0.6	1/280	46810	5325.67	2579
280[b]	2	0.6	1/280	76680	7433.69	2579
280[b]	1	0.6	1/280	3600	7205.13	2579
280[b]	2	0.6	1/280	21600	5509.59	2579
280[b]	1	0.6	1/280	81140	4724.52	2579

3. El Problema de Secuenciación de Trabajos

3.1. Antecedentes del PST

La investigación de secuencias en la programación de tareas es muy amplia, ver por ejemplo, la recopilación de trabajos de Delgado (2005), en donde se presentan diferentes métodos como programación matemática, reglas de despacho, sistemas expertos, redes neuronales, algoritmos genéticos, búsqueda tabú, entre otros. El PST es uno de los problemas más difíciles de resolver por lo que los métodos exactos como el método de ramificación y acotamiento y la programación dinámica requieren una gran cantidad de tiempo para resolver el problema (Tamilarasi y Anantha, 2010) por lo que se han buscado otros métodos, como los algoritmos genéticos para resolverlo.

Entre los autores que han utilizado metaheurísticos para su solución destacan Yamada y Nakano (1998), Sivanandam y Deepa (2008) con algoritmos genéticos, Bozejko et al. (2009) con recocido simulado, Juang (2004) y Ge et al. (2007) con algoritmos híbridos entre algoritmos genéticos y optimización de partículas, con un híbrido entre AG y recocido simulado, entre otros.

El presente trabajo tiene por objetivo, resolver el PST como un PAV y utilizar los valores de los parámetros encontrados en la sección anterior para lograrlo. Se diferencia de los trabajos encontrados en la literatura en que se propone un método sencillo para tomar decisiones dentro de la industria, ya que requieren respuestas inmediatas; el AG proporciona un conjunto de buenas soluciones que le sirven a la industria, sin someterla al entendimiento de modelos complejos.

Algunos trabajos similares encontrados son el Chong Wu et al. (2004) que transforman también el PAV en PST, reportando resultados para tres instancias, ellos se basan en la modificación del AG para resolver el PST, en nuestro trabajo el AG puede resolver el PAV y con ello resolvemos el PST. El trabajo de Gao Shang et al. (2007) que resuelven el PAV a través de colonia de hormigas y luego mencionan que se puede utilizar para resolver PST, pero no reportan problemas resueltos.

3.2. Definición del PST

Es posible describir al PST como la disposición de un conjunto de máquinas que realizan diversas tareas cuando se pretende producir un cierto producto a partir de un insumo. Para lo cual se sigue una serie de pasos, donde cada paso consiste en aplicar una máquina determinada durante un período de tiempo. El problema consiste en encontrar una secuencia que minimice el tiempo necesario para completar todas las operaciones a la cual se la denomina como *makespan*. Entendiendo por operación *(O)* a la realización del trabajo *(J_n)* en la máquina (M_n) (Bektas, 2006).

Si se pretende obtener varios productos distintos, tendremos asociado un trabajo con sus correspondientes operaciones para cada uno de esos productos. De tal manera que al tener un conjunto de máquinas y un conjunto de trabajos, una secuencia es una asignación que fija a cada operación con un intervalo de tiempo para ser efectuada.

Para explicar la manera de resolver el PST mediante el PAV se ejemplifica con el caso presentado por Tamilarasi y Anantha (2010), el cual presenta los siguientes tiempos de operación (Tabla 10*)*:

Tabla 10: Operaciones

$O_1=J_1M_1=2$	$O_2=J_1M_3=3$	$O_3=J_1M_2=4$
$O_4=J_2M_2=1$	$O_5=J_2M_1=5$	$O_6=J_2M_3=2$
$O_7=J_3M_3=4$	$O_8=J_3M_1=6$	$O_9=J_3M_2=4$

El resultado obtenido es de 17 unidades. En este trabajo el PST es abordado como un PAV. Por lo que a partir de los tiempos de operación podemos hacer la analogía de cada operación como una ciudad a visitar por el agente viajero; para formar la matriz de distancias considere la $O_1=J_1M_1=2$ *y la* $O_2=J_1M_3=3$, de O_1 a O_2 el trabajo J_1 debe ser procesado en ambas máquinas, de tal forma que la O_2 no puede empezar hasta que termine O_1 como se puede apreciar en la Figura 7, por lo que la distancia entre O_1 y O_2 es cinco unidades. En el caso de que la máquina y el trabajo no sean los mismos, las operaciones pueden realizarse de forma simultánea, por ejemplo O_1 y O_6 (ver Figura 8), así que el tiempo para la matriz de distancias es la mayor de las operaciones, en este caso dos unidades. La matriz de operaciones se muestra en la Tabla 10.

Con lo anterior, la matriz de costos del PAV se forma como se muestra en la Tabla 11.

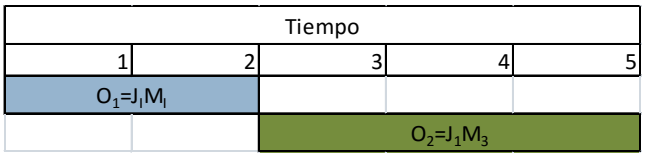

Figura 7. Gantt entre O_1 y O_2.

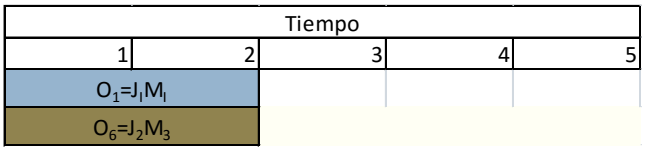

Figura 8. Gantt entre O_1 y O_6.

Tabla 11: Matriz de distancias

	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9
O_1	M	5	6	2	7	2	4	8	4
O_2	5	M	7	3	5	5	7	6	4
O_3	6	7	M	5	5	4	4	6	8
O_4	2	3	5	M	6	3	4	6	5
O_5	7	5	5	6	M	7	5	11	5
O_6	2	5	4	3	7	M	6	6	4
O_7	4	7	4	4	5	6	M	10	8
O_8	8	6	6	6	11	6	10	M	10
O_9	4	4	8	5	5	4	8	10	M

Utilizando AG en Matlab se encontró la siguiente ruta de la Figura 9:

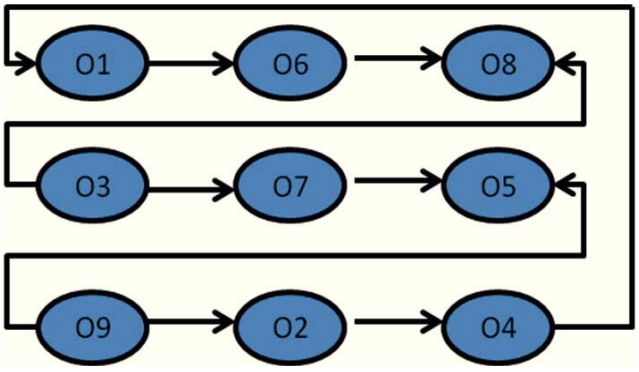

Figura 9. Secuencia del PAV.

Esta ruta puede convertirse al PST, por ejemplo la O_1 representa el J_1 en la M_1 y tiene un tiempo de dos y es la primera que se dibuja, la siguiente es la O_6 que representa el J_2 en la M_3 y tiene un tiempo de dos unidades, al no estar relacionadas estas operaciones pueden hacerse de forma simultánea como puede apreciarse en la figura 10. Posteriormente se dibuja la O_8 que es la J_3 en la M_1 y esta es igual a 6 unidades, como la O_1 utilizó a la M_1 se debe esperar a que termine esta operación, así que O_6 se dibuja al terminar O_1. Esto se realiza hasta terminar cada una de las operaciones. El tiempo en el que termina el último trabajo ser procesado es de 17 unidades (ver Figura 10).

En la Tabla 12 se muestra una comparación de resultados de los experimentos del Problema de Secuenciación de Trabajos resuelto mediante AG. Estos problemas son conocidos en la literatura, ver Tamilarasi y Anantha (2010) y Ruiz (2011) y son parte del banco de problemas disponibles sobre el problema de secuenciación de trabajos proporcionado por Beasley (1990). Como se puede observar se encontraron resultados muy cercanos a los de ellos. En la primera columna se muestra el nombre del experimento, el número de trabajos y de máquinas, su referencia siendo a (Tamilarasi y Anantha ,2010) y b (Ruiz 2011); la columna 2 muestra el número de individuos, la 3 el número de generaciones, la columna 4 el tiempo en el que corrió en segundos; por último se muestra en la columna 5 la mejor distancia encontrada (makespan) y en la 6 se proporciona el mejor resultado encontrando en la literatura. Si bien es cierto, que los resultados alcanzados con la presente investigación son sólo cercanos al mejor encontrado, nuestro objetivo era encontrar un método alternativo que les permitiera a las empresas tener un conjunto de alternativas de decisión para el PST, y esto lo proporcional el PAV resuelto con AG.

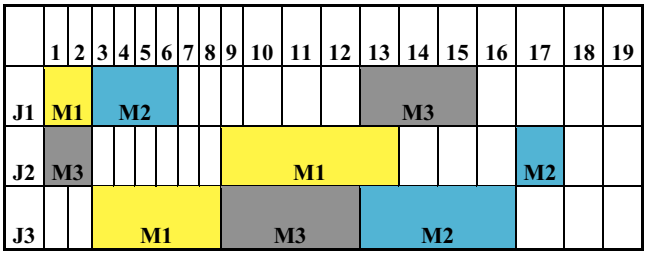

Figura 10. Gantt.

Tabla 12: Resultados obtenidos en PST.

1	2	3	4	5	6
3x3[a]	1000	100	120	17	17
FT06[b] 6x6	1000	100	180	55	55
LA04[b] 10x15	60000	300	64800	621	590
FT10[b] 10x10	50000	1000	41230	1156	930
LA02[b] 10x5	50000	1000	40120	768	655
LA03[b] 10x10	50000	1000	40100	699	597
LA12[b] 10x5	70000	1250	110801	1412	1039
LA13[b] 20x5	50000	1000	75900	1711	1150

4. Conclusiones

En la presente investigación se explicaron algunas metodologías que permiten resolver al PAV, las que posteriormente sirvieron para resolver un PST al codificarlo como un PAV. Estos problemas son bastante comunes en diversas disciplinas como en la ingeniería industrial, sin embargo, al crecer el número de operaciones, el tiempo computacional aumenta hasta un punto en que no es factible su utilización para tomar decisiones dentro de la industria, ya que requieren respuestas inmediatas. Es por ello que los metaheurísticos muestran ser una herramienta muy útil para la búsqueda de respuestas cercanas al óptimo en problemas de poca complejidad y tamaño en un tiempo relativamente corto.

Por ello se recurre a los AG, los cuales son heurísticos que permitieron encontrar buenos resultados a los PAV presentados en esta investigación, en donde logramos encontrar parámetros que registraron los mejores resultados en los diferentes problemas. Estos parámetros fueron un número de individuos mayor al número de iteraciones, en una proporción de 10 individuos para 1 iteración aproximadamente. La probabilidad de cruce muestra los mejores resultados en 0.6 y la probabilidad de mutación utilizada fue relativamente baja de *1/n*; en donde *n* representaba el número de ciudades del problema.

También se utilizaron dos tipos de selección. La primera por torneo y la segunda por ruleta. Como se observó en los resultados los mejores fueron aquellos problemas resueltos a través de torneo. Resolver el PST a través de un PAV permite tener otra alternativa para abordar este problema a través de metaheurísticos. En la experimentación realizada para la Secuenciación de Trabajos, logramos igualar ciertos resultados aunque en su mayoría tan sólo logramos aproximarnos a la solución encontrada en los artículos científicos. Con ello podemos concluir que los PST que se presentan en la industria, pueden ser resueltos en tiempos adecuados, como lo demandan las empresas, ajustando los tiempos para la realización de tareas a un PAV, el cual al resolverlo mediante un metaheurístico como lo son los AG obtiene resultados próximos al óptimo.

English Summary

Solution of the Job-Shop Scheduling Problem through the Traveling Salesman Problem.

Abstract

In this paper we proposed a solution to the Job-Shop Scheduling Problem using the Traveling Salesman Problem solved by Genetic Algorithms. We proposed a genetic algorithm where we compare two types of selection: tournament and roulette. Different tests are performed to solve the Traveling Salesman Problem with the two types of selection under different parameters: number of individuals, number of iterations, crossover probability and mutation probability. Then the best type of selection and the best parameters are used to solve the Job-Shop Scheduling Problem with Genetic Algorithms for the Traveling Salesman Problem. The proposal is presented solving different examples of Job Sequencing Problem and compare them with the results obtained in the literature.

Keywords:

Efficient algorithms, industrial production systems, optimization problem, traveling salesman problem.

Referencias

Beasley, J., 1990.OR-Library: Distributing test problems by electronic mail. Journal of the Operational Research Society, 11, 1069-1072.

Bektas, T., 2006. The multiple traveling salesman problem: an overview of formulations and solution procedures, 34,209-219.

Bozejko, W., Pempera J. and Smuntnicki C., 2009. Parallel simulated annealing for the Job Shop Scheduling problem. Biological Cybernetics, 60, 139-144.

Buthainah F. and Hamza A., 2008. Enhanced Traveling Salesman Problem solving by Genetic Algorithm Technique. World Academy of Science, Engineering and Technology,38, 298-302.

Cerny, V., 1985. Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm. Lecture note in computer science Proceedings of the 9th Intenational Conference on Computational Science, 5544, 631-640.

Chambers, L., 1998. Genetic Algorithms. University Western Press, Australia.

Chatterjee, S., Carrera C., and Linch L., 1996. Genetic Algorithms and traveling salesman problems. Siam Journal of Optimation, 515-529.

Chunguo, W., Wei X., and Yanchun L., 2004. A Modified Integer-Coding Genetic Algorithm for Job Shop Scheduling Problem, Trends in Artificial Intelligence. Lecture Notes in Computer Science Volum 3157,373-380.

Delgado E., 2005. Aplicación de Algoritmos Genéticos para la Programación de Tareas de una celda de manufactura. Ingeniería e Investigación: Universidad Nacional de Colombia, 24-31.

Dorigo, M., 1997. Ant colonies for the traveling salesman problem. Universidad Libre de Bruselas, Bélgica.

Fogel, D., 1998. An evolutionary approach to the traveling salesman problem. Biological Cybernetics, 60, 139-144.

Gao, S., Zhang L, and Zhang F. 2007. g. **S**olving Traveling Salesman Problem by Ant Colony Optimization Algorithm with Association Rule Proceedings of the Third International Conference on Natural Computation, 3, 693-698.

Ge, H., Du W., y Quian F., 2007. A hybrid algorithm based on swarm optimization and simulated annealing for job shop scheduling. Proceedings of the Third International Conference on Natural Computation, 3, 715-719.

Gerhard, R., 2006. Discrete and Combinatorial Optimization. Universidad de Heidelberg-Instituto de Ciencias de la Computación, Alemania.

Goldberg, D., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Corporation, Estados Unidos de América.

Holland, J., 1992. Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, Estados Unidos de América.

Jog, P., Kim J., Suh J., y Gucht D., 1991. Parallel Genetic Algorithms Applied to the Traveling Salesman Problem. European Journal of Operational Research, 490-510.

Juang, C., 2004. A hybrid genetic algorithm and particle swarm optimization for recurrent network design. IEEE Transactions on Evolutionary Computation, 34, 997-1006.

Larrañaga, P., Kuijpers C., Murga R., Inza I., y Dizdarevic S., 1999. Genetic Algorithms for the Traveling Salesman Problem: A review of Representations and Operators. Artificial Intelligence Review, 129–170.

Maldonado C. 2010. El mundo de las ciencias de la complejidad: Una investigación sobre qué son, su desarrollo y sus posibilidades. Universidad del Rosario.

Moon, C., Kim J., Choi G., y Seo Y., 2002. An efficient genetic algorithm for the traveling salesman problem with precedent constraints. European Journal of Operational Research, 606-617.

Ruiz, J., 2011. Complexity Indicators Applied to the Job Shop Scheduling Problem. International Journal of Combinatorial Optimization Problems and Informatics, 25-31.

Sivanamdam, S. y Deepa S, 2008. Introduction to Genetic Algorithms. Springer, Estados Unidos de América.

Tamilarasi, A. y Anantha K., 2010. An enhanced genetic algorithm with simulated annealing for jobshop scheduling. International Journal of Engineering Science and Technology 2, 144–151.

Wagner, H, 1975. Principles of Operations Research. 2ª ed. Englewood Cliffs, N.J. Prentice Hall, Estados Unidos de América.

Winston, W., 2005. Investigación de Operaciones: Aplicaciones y Algoritmos. Indiana University Press, Estados Unidos de América.

Yamada T., y Nakano R., 1997. Genetic Algorithms for Job Shop Scheduling. Proceedings of Modern Heuristic for Decision Support, UNICOM Seminar London, 67-81.