Programación Matemática y Software (2009)
Vol. 1. No 1. ISSN: 2007-3283

Recibido: 30 de Abril de 2008/Aceptado: 3 de Agosto de 2008.
Publicado en línea: 26 de junio de 2009

Un Mecanismo de Vecindad con Búsqueda Local y Algoritmo Genético para el Problema de Transporte con Ventanas de Tiempo

Marco Antonio Cruz-Chávez, Ocotlán Díaz-Parra[A]

CIICAp, Universidad Autónoma del estado de Morelos, Avenida Universidad 1001. Col. Chamilpa, C.P. 62210. Cuernavaca, Morelos, México.
{mcruz, odiazp }@uaem.mx

Resumen. Las técnicas de búsqueda por vecindad han resultado medios útiles para encontrar soluciones aproximadas a problemas de optimización combinatoria. Una vecindad está definida como el conjunto de soluciones cercanas a una solución inicial dada. En este artículo se presenta un mecanismo de vecindad combinado con un algoritmo genético, mostrando la etapa de análisis y diseño de la estructura de vecindad con búsqueda local combinada con un algoritmo genético para el problema de transporte con ventanas de tiempo. Este diseño hibrido se propone con la finalidad de explotar el espacio de soluciones del problema del transporte con ventanas de tiempo. La vecindad se propone con movimientos tipo uno-óptimos.

1 Introducción

Las técnicas de búsqueda por vecindad comúnmente se llaman algoritmos de búsqueda local. Una vecindad es un conjunto de todas aquellas soluciones que pueden ser alcanzables a partir de una solución inicial, por medio de un movimiento que puede ser un intercambio entre elementos que conforman la solución inicial. La parte medular en el éxito de una búsqueda en vecindad es su estructura y tamaño. En cuanto al tamaño de la vecindad mientras más grande sea mayor será la calidad de las soluciones. Una vecindad amplia produce una heurística más eficaz. La tendencia de utilizar procedimientos de búsqueda local en combinación con otras estrategias se refleja en los trabajos encontrados en la literatura especializada, por ejemplo:

[A] Autor de correspondencia

1. Búsqueda Local Guiada de Vondouris y Tsang [1].- La idea consiste básicamente en dividir el vecindario de la solución actual en subvecindarios mas pequeños y asociar a cada uno de estos subvecindarios una variable binaria que indique si están activos o inactivos, de tal forma que en cada paso solamente se exploran los subvecindarios activos. En cada iteración si al explorar el subvecindario este no contiene solución mejor a la actual se pone inactivo, en caso contrario permanece activo. A medida que avanza la solución y existe mejora habrá menor número de vecindarios activos y por tanto se gastara menos tiempo de computación.
2. GRASP de Feo y Resende [2] [3].- Se implementó originalmente para estudiar un problema de cobertura de alta complejidad. La palabra GRASP es el acrónimo de Greedy Randomized Adaptive Search Procedure (procedimiento de búsqueda miope, aleatorizado y adaptativo). El GRASP es una metaheurística para encontrar soluciones aproximadas (es decir, sub-óptimas de buena calidad, pero no necesariamente óptimas) a problemas de optimización combinatoria. Se basa en la premisa de que soluciones iníciales diversas y de buena calidad juegan un papel importante en el éxito de métodos locales de búsqueda. La búsqueda local juega un papel importante en GRASP ya que sirve para buscar soluciones localmente óptimas en regiones prometedoras del espacio de soluciones [26].
3. Concentración Heurística de Rosing y Revelle [4]. - La técnica se desarrolla en dos fases básicas: a) Genera un conjunto de óptimos locales a partir de soluciones aleatorias y registrar mejores. b) Forma el conjunto de CS (conjunto de concentración) de elementos que aparecen en esas soluciones y ejecuta un algoritmo exacto o heurístico del problema original pero restringiendo o concentrando la búsqueda de elementos al conjunto de concentración.
4. Algoritmos Meméticos según Moscato [5].- Los algoritmos meméticos (MA) son técnicas de optimización que combinan sinérgicamente conceptos tomados de otras metaheurísticas, tales como la búsqueda basada en poblaciones (como en los algoritmos evolutivos), y la mejora local. Con los algoritmos meméticos se añade a las operaciones de los algoritmos genéticos la búsqueda local.
5. Búsqueda Dispersa o SS (scatter search) de Laguna [8].- Scatter search es un ejemplo de lo que se conoce como métodos evolutivos. Pero a diferencia de otros métodos evolutivos no utiliza la aleatoriedad como principal mecanismo para buscar soluciones [24].
6. Búsqueda Tabú básica de Glover [6] [7].- Es una técnica iterativa de la búsqueda local que trata de evitar que las soluciones caigan en óptimos locales. Para esto utiliza unas estructuras de memorias de corto y largo plazo. En cada iteración se pretende pasar de una solución a la mejor solución vecina sin importar si esta es mejor o peor que la solución actual. El criterio de terminación puede ser un

cierto número máximo de iteraciones o un valor de la función a optimizar [21].

7. Recocido Simulado por Kirkpatrick [9] [10] esta técnica está basada en el proceso físico de tratamiento térmico de los metales denominado recocido, en donde un metal es llevado a altas temperaturas alcanzando altos niveles energéticos llegando al punto de fusión y luego es enfriado gradualmente, en un proceso por fases en donde el sólido puede alcanzar el equilibrio térmico para cada fase hasta volver de nuevo al estado sólido obteniendo un estado de energía mínimo que es definido previamente. De esta manera se puede construir el modelo de optimización comparando las posibles soluciones con los estados del sistema físico y el costo de la solución con la energía del estado, teniendo en cuenta que el proceso de recocido debe tener una configuración en el recocido simulado, se debe tener una solución factible; la solución optima en el problema de optimización está relacionada con la configuración que se debe tener para alcanzar el mínimo de energía nombrado anteriormente y finalmente el manejo de la temperatura en el proceso de recocido se compara con un parámetro dado en el modelo de optimización.

Estas estrategias son en realidad repeticiones de procedimientos de Búsqueda Local (donde se modifica la función objetivo y la solución inicial se reemplaza por una mejor).

La tendencia es aplicar búsqueda local en vecindades junto con otras técnicas de solución a problemas de optimización combinatoria [17].

En este artículo se propone un mecanismo de vecindad inmerso en un algoritmo genético con la finalidad de realizar una explotación del espacio de soluciones del problema de transporte con ventanas de tiempo.

El problema del transporte o VRP (Vehicle Routing Problem) como se conoce en literatura [12], es un problema que se le da principal importancia en las áreas de transportación, distribución y logística. En algunos sectores de la industria, la transportación significa un alto porcentaje de valor agregado a los productos. Por eso la utilización de métodos computacionales para que la

transportación ofrezca buenos resultados es de gran utilidad, los ahorros van desde un 5% a un 20% en el total de costos, como lo reporta Toth & Vigo [12].

El problema del transporte tiene diferentes variantes. Por ejemplo, que cada vehículo tenga una capacidad limitada, que cada cliente tenga que ser atendido dentro de un margen de tiempo (ventana de tiempo), que los puntos de suministro sean varios, que los clientes deban ser atendidos por varios vehículos, que algunos datos del problema sean aleatorios, que las entregas se hayan de realizar en determinados días. En el presente documento se trabajará con el problema de transporte con ventanas de tiempos y flota de vehículos fija.

El problema de transporte con ventanas de tiempo VRPTW (Vehicle Routing Problem with Time Windows) es una variante del VRP y consiste básicamente en minimizar los costos de transportación sujeto a restricciones de tiempo de cada ruta y de capacidad en base a la demanda de cada cliente [12]. La función objetivo que representa a este problema se describe en la ecuación (1) y las restricciones de ventana de tiempo en la ecuación (2).

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ijk} \tag{1}$$

$$a_i \sum_{j \in \nabla^+(i)} x_{ijk} \leq w_{ik} \leq b_i \sum_{j \in \nabla^+(i)} x_{ijk}, \forall k \in K, i \in N \tag{2}$$

Dónde: k = vehículo, i = nodo origen, j = nodo destino, x = ruta, w_i = ventana de tiempo del nodo i, b_i = extremo final de la ventana perteneciente al nodo i, a_i = extremo inicial de la ventana perteneciente al nodo i, $\nabla^+(i)$ = conjunto de vértices j tales que $(i,j) \in A$.

El problema de transporte con ventanas de tiempo se considera dentro de la teoría de la complejidad un problema NP-Completo [18] [25] [27] [28]. Solomon [30] y Savelsbergh [29] indican que las restricciones de tiempo del problema y una flota de vehículos fija es lo que lo hace más difícil que el problema VRP simple. Savelsbergh muestra que encontrando

una solución factible con una flota de vehículos fija el problema VRPTW se convierte en un problema NP-Completo [29].

Existen diferentes tipos de instancias para VRPTW. Las instancias están clasificadas por tipo y por clase, se tiene dos tipos; el tipo1 maneja ventanas estrechas de tiempo y vehículos con capacidad pequeña, las de tipo2 manejan ventanas grandes de tiempo y capacidad grande de vehículos. Se tienen tres clasificaciones C, R y RC, la clasificación C son las instancias cuya distribución territorial por cliente es arracimado, en las instancias de clase R los clientes están uniformemente distribuidos en un área territorial, las instancias de clase RC son la combinación de distribución territorial arracimada y distribuida. En total las instancias que se utilizan comúnmente en el benchmark de Solomon para VRPTW son C1, R1, RC1, C2, R2, RC2 la figura1 muestra la caracterización de estas instancias.

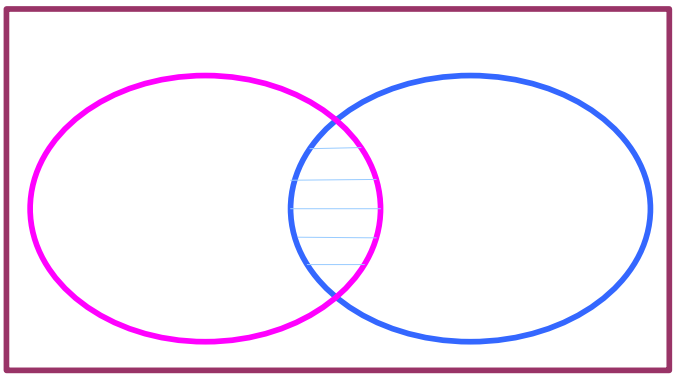

Fig.1. Caracterización de instancias de VRPTW

La representación numérica de las instancias para el problema del transporte con ventanas de tiempo contiene información del número de clientes, la ubicación geográfica de los clientes, la demanda por cliente, el tiempo de inicio de la ventana de tiempo, el tiempo final de la ventana de tiempo y el tiempo de servicio de la ventana de tiempo.

La instancia más pequeña definida para este tipo de problema es la de 25 clientes para los dos tipos y las tres clasificaciones. La instancia más grande para la cual se ha encontrado soluciones es de 100 clientes, existen instancias definidas para 200 clientes para las cuales aún no se conoce una mejor solución o un óptimo.

A continuación se presentan los datos tal cual se presentan en literatura para ser usados en la experimentación de una instancia definida por Solomon para el problema de ruteo de vehículos con ventanas de tiempo tipo 1 clasificación C para 25 clientes con una flotilla homogénea de 25 vehículos con capacidad de 200.

C101

VEHICLE

NUMBER	CAPACITY
25	200

CUSTOMER

CUST NO.	XCOORD.	YCOORD.	DEMAND	READY TIME	DUE DATE	SERVICE TIME
0	40	50	0	0	1236	0
1	45	68	10	912	967	90
2	45	70	30	825	870	90
3	42	66	10	65	146	90
4	42	68	10	727	782	90
5	42	65	10	15	67	90
6	40	69	20	621	702	90
7	40	66	20	170	225	90
8	38	68	20	255	324	90
9	38	70	10	534	605	90
10	35	66	10	357	410	90
11	35	69	10	448	505	90
12	25	85	20	652	721	90
13	22	75	30	30	92	90
14	22	85	10	567	620	90
15	20	80	40	384	429	90
16	20	85	40	475	528	90
17	18	75	20	99	148	90
18	15	75	20	179	254	90
19	15	80	10	278	345	90
20	30	50	10	10	73	90
21	30	52	20	914	965	90
22	28	52	20	812	883	90
23	28	55	10	732	777	90
24	25	50	10	65	44	90
25	25	52	40	169	224	90

Donde: CUSTNO= Número de clientes, XCOORD= Coordenada, YCOORD= Coordenada, DEMAND= Demanda, READYTIME= Tiempo de llegada, DUEDATE= Tiempo de termino, SERVICETIME= tiempo de servicio.

Una de las alternativas de solución a este tipo de problemas son los llamados métodos heurísticos. Un método heurístico es un procedimiento para resolver un problema matemático bien definido mediante aproximaciones en un tiempo de computación razonable sin garantizar la optimalidad [13] [23]. El método heurístico utilizado como alternativa de solución para este trabajo es un algoritmo genético aplicado al problema de transporte con ventanas de tiempo. Un algoritmo genético (GA) es un método adaptativo que puede usarse para resolver problemas de búsqueda y optimización. Estos algoritmos están basados en el proceso genético de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin [22]. Por imitación de este proceso, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. Haciendo una analogía del algoritmo genético con el problema del transporte un individuo formado por cromosomas y genes corresponde a una ruta formada por vehículos y nodos.

El algoritmo genético simple: La forma más simple de un algoritmo genético involucra el uso de 3 tipos de operadores: selección, cruzamiento (de un punto), y mutación [20]. El operador de selección, selecciona un cromosoma dentro de la población para su reproducción. El operador de cruzamiento aleatoriamente selecciona e intercambia dos cromosomas para crear otros dos cromosomas adicionales, es decir recombina biológicamente entre dos simples organismos de cromosomas. El operador de mutación intercambia aleatoriamente dos cromosomas de bits dentro de un cromosoma.

Los pasos generales de un algoritmo genético son: Inicializar aleatoriamente la población inicial de soluciones del problema, evaluar cada una de las soluciones (asignando un valor o fitness de acuerdo a la solución factible encontrada), seleccionar la población inicial (aquella que tenga el mejor valor), aplicar el operador de cruzamiento y el operador de mutación a diferentes soluciones de la población seleccionada para crear una nueva población, repetir un número determinado de veces hasta que la solución sea encontrada.

Un GA simple funciona como se muestra a continuación [22]:

1.-Iniciar con una población generada aleatoriamente con n−bit cromosomas (soluciones candidatas a un problema).

2.-Calcular el valor del fitness f(x) para cada cromosoma x de la población.

3.-Repetir los pasos siguientes hasta n generaciones (Cada iteración es llamada generación. Un GA tiene entre 20 y 500 generaciones o más) hayan sido creadas:

a.- Seleccionar cada par de cromosomas padres de la actual población, la probabilidad de selección empieza a incrementar la función del fitness. La selección se termina con el remplazo, lo que significa que el mismo cromosoma puede ser seleccionado más de una vez por un cromosoma padre.

b.- Con la probabilidad pc (crossover probability o crossover rate), el cruzamiento sobre los pares en un punto aleatoriamente seleccionado (tomado con una probabilidad uniforme). Si no existe cruzamiento se copia exactamente.

c.- Mutar con la probabilidad mp (mutation probability o mutation rate), y poner los cromosomas resultantes dentro de la nueva población.

4.- Reemplazar la actual población con la nueva población.

5.- Ir al paso 2.

Una vez puesto el marco de referencia de los elementos a utilizar en este artículo, en la sección 2 se explicará a detalle la definición de vecindad, en la sección 3 se explica la vecindad propuesta, en la sección 4 se muestra un análisis del nivel de aplicación de la vecindad en el algoritmo

genético y en la sección 5 las conclusiones y los trabajos futuros derivados de el análisis presentado en este documento.

2 Definición de Vecindad

La vecindad de una solución se define como el conjunto de todas aquellas soluciones que pueden ser alcanzables a partir de una solución *s'* por medio de un movimiento σ [11], un movimiento puede ser un intercambio entre elementos que conforman la solución *s*.

$$N(s) = \{s' \in S : s \xrightarrow{\sigma} s'\} \tag{3}$$

Donde *N(s)* representa la vecindad con respecto a *s*, *s* representa una solución tomada del espacio total de soluciones *S* y s' representa el vecino de *s* generado a partir de σ movimientos. Un movimiento puede ser una inserción, eliminación o intercambio de componentes en una solución. Para este caso en particular un movimiento estará definido como un intercambio de dos genes en un individuo. En la figura 2 se muestra la representación gráfica de la vecindad generada a partir de una solución *s*.

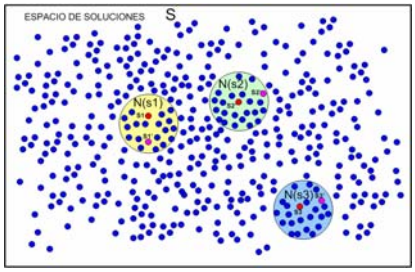

Fig. 2. Vecindad generada a partir de individuo solución *s*

Por ejemplo la vecindad generada por la solución s_1 está representada en la figura 2 como $N(s_1)$ en el círculo amarillo y sus vecinos están representados por s_1' y así sucesivamente para cada una de las soluciones s_i que pertenecen al espacio de soluciones *S*, en notación de conjuntos se expresa $S=\{ s_1 , s_2 , s_3 ,\dots s_i\}$. La estructura de vecindad mediante una búsqueda permite ir haciendo

una explotación en el espacio de soluciones. A continuación se explica a detalle cómo se construye la vecindad, los criterios aplicados en la construcción y su justificación.

3 Mecanismo de Vecindad con búsqueda local y algoritmo genético

La vecindad que se propone implementar al algoritmo genético para el problema del transporte se basa en la idea propuesta por Or [14], para el Problema del Viajante o TSP. El método de intercambio de Or es una variante de los conocidos intercambios *r-óptimos* desarrollados por Lin [15] y Lin & Kernighan [16] para el TSP simétrico. Un intercambio *r-óptimo* consiste básicamente en mover cadenas de *r* elementos. Por ejemplo sea 0-1-2-3-4-0-5-6-7-0-8-9-10-0 una ruta válida para un problema de rutas. Un intercambio 2-óptimo consiste en intercambiar cadenas de dos números; entonces para la cadena inicial dada se intercambiarán las cadenas 3-4 y la cadena 8-9 de tal forma que la siguiente ruta factible estará definida por: 0-1-2-**8-9**-0-5-6-7-0-**3-4**-10-0.

Para este trabajo se propone utilizar la técnica de Or con 1-óptimo tomando solo movimientos hacia adelante como lo propone Pacheco y Delgado [17] y en este caso para VRPTW se establecerá un límite en el tamaño de la cadena a intercambiar; el tamaño será de uno. En VRPTW cada movimiento en la ruta total implica *n*-búsquedas para no violar las restricciones propias del problema, por lo que es conveniente utilizar movimientos con cadenas de uno, a esta nueva adecuación la llamamos vecindad tipo OR modificada.

El mecanismo de vecindad propuesto es la vecindad tipo OR modificada tomando dos nodos con intercambio de cadenas 1-óptimo y el movimiento se realiza hacia adelante. Y la búsqueda en la vecindad es una búsqueda local clásica. El tamaño de la vecindad se establece mediante la razón del tamaño del individuo y el número de genes a intercambiar, dicha razón se conoce en optimización combinatoria como el número de combinaciones sin repetición de *m* elementos con tamaño *n* como se representa en la ecuación (4).

$$c_m^n = \binom{m}{n} = \frac{m!}{n!(m-n)!}$$

(4)

$$T_{N(s)} = c_m^n \tag{5}$$

Generalizando la ecuación (4) en términos de vecindad, surge la ecuación (5), donde $T_{N(s)}$ es el tamaño de la vecindad generada a partir de *s*.

Representando las ecuaciones (4) y (5) en términos de vecindad en la ecuación 6, se tiene que el tamaño de la vecindad $T_{N(s)}$ con respecto a la solución *s* está definido como la razón del tamaño del individuo y del número de genes a mutar esto es:

$$T_{N(s)} = \frac{t_{(s)}!}{n_{(s)}!(t_{(s)} - n_{(s)})!} \tag{6}$$

Donde $t_{(s)}$ representa el tamaño del individuo y $n_{(s)}$ representa el número de genes a mutar.

Por ejemplo para un individuo formado por 10 genes con intercambio de dos genes, $t_{(s)}$ =10 y $n_{(s)}$ =2 por lo tanto $T_{N(s)}$ =45 lo que implica que solo 45 vecinos serán generados.

El criterio de paro de la vecindad se propone de inicio como lo proponen en literatura dos veces el tamaño de la vecindad como se representa en la ecuación 7.

$$P(N(s)) = 2T_{N(s)} \tag{7}$$

Donde $P_{(N(S))}$ representa el criterio de paro de la búsqueda en la vecindad $N(s)$. En cuanto al criterio de paro de la vecindad se pretende, en base a la experimentación, realizar un análisis del comportamiento de los resultados y con base en ese análisis proponer un criterio de paro. Una vez identificado el tipo de vecindad, el tamaño de la vecindad y el criterio de paro el siguiente paso es analizar en que parte del algoritmo genético se aplicará la vecindad.

4 Algoritmo genético con búsqueda local en vecindad

El análisis para la aplicación de la vecindad implica dos partes. La primera consiste en analizar una solución mediante un algoritmo genético para el problema de transporte con ventanas de tiempo y la segunda mediante este análisis proponer el nivel de aplicación de la vecindad. Por la naturaleza del algoritmo genético se hace uso de una población inicial, sometida a una serie de operadores genéticos como es el operador de selección, el operador de cruzamiento y el operador de mutación como lo muestra la Figura3.

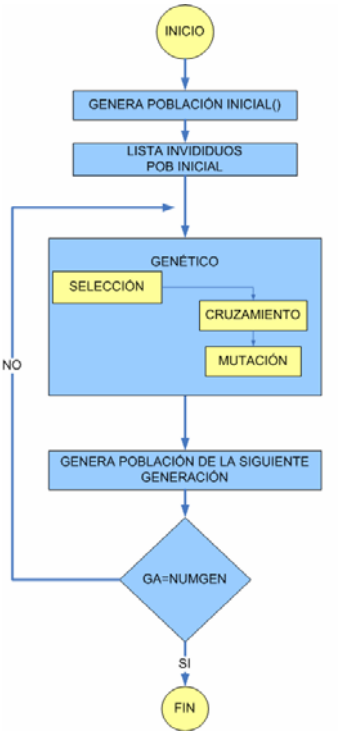

Fig. 3. Diagrama de flujo del algoritmo genético

El algoritmo genético se ejecutó 10 veces obteniendo los tiempos totales del algoritmo mostrados en la tabla1 y 10 veces más tomando los tiempos por función mostrados en la tabla2.

Tabla 1. Tiempos de ejecución del algoritmo génetico en serie de 10 para intancias C101 para 100 nodos .

Instancia	Tiempo(seg)	Instancia	Tiempo(seg)
C101-100	2223	C101-100	2131
C101-100	2734	C101-100	2048
C101-100	2196	C101-100	2051
C101-100	2515	C101-100	1985
C101-100	2313	C101-100	2133

En la tabla 1, se muestran los tiempos totales de ejecución, en base a estos datos se obtiene el tiempo de ejecución del algoritmo genético en promedio 2232 segundos. La hipótesis que se plantea es minimizar mediante la estructura de vecindad el tiempo total para hacer más eficiente el algoritmo genético.

Tabla 2. Tiempos de ejecución del algoritmo génetico por función en serie de 10 para intancias C101 para 100 nodos.

Instancia	Función	Tiempo(s	Instancia	Función	Tiempo(s

		eg)			eg)
C101-100	Población()	329	C101-100	Población()	345
	Selección()	455		Selección()	490
	Cruzamiento()	543		Cruzamiento()	528
	Mutación()	598		Mutación()	635
	Nva-Generac()	212		Nva-Generac()	215
C101-100	Población()	351	C101-100	Población()	367
	Selección()	523		Selección()	509
	Cruzamiento()	578		Cruzamiento()	554
	Mutación()	667		Mutación()	675
	Nva-Generac()	212		Nva-Generac()	217
C101-100	Población()	357	C101-100	Población()	390
	Selección()	567		Selección()	522
	Cruzamiento()	563		Cruzamiento()	578
	Mutación()	689		Mutación()	656
	Nva-Generac()	213		Nva-Generac()	211
C101-100	Población()	333	C101-100	Población()	346
	Selección()	556		Selección()	512
	Cruzamiento()	575		Cruzamiento()	524
	Mutación()	636		Mutación()	669
	Nva-Generac()	216		Nva-Generac()	218
C101-100	Población()	369	C101-100	Población()	378
	Selección()	515		Selección()	509
	Cruzamiento()	529		Cruzamiento()	548

Mutación()	662	Mutación()	649
Nva-Generac()	216	Nva-Generac()	215

En la tabla 2 se muestra la experimentación realizada por función lo que se pretende es detectar cual de las funciones consume mayor tiempo y tomar ese tiempo como parámetro para analizar en especifico esa función y analizar si es posible implementar la estructura de vecindad, además se presentan los resultados de tiempos de ejecución en serie de 10 por función, notando que en la función de mutación es la que implica mayor tiempo. Esto nos da la pauta para proponer la estructura de vecindad en lugar del operador de mutación del algoritmo genético como lo muestra la Figura 4.

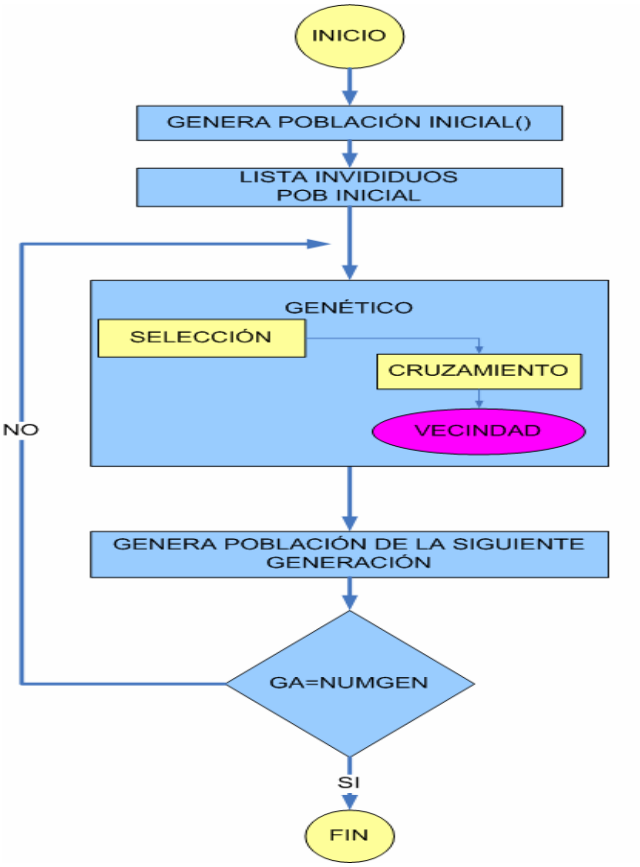

Fig. 4. Nivel de implementación de la estructura de vecindad en el algoritmo genético

Cabe mencionar que en el algoritmo genético la mutación se aplica a nivel de individuo, lo que significa que los movimientos realizados dentro de este operador en el algoritmo genético implica lo que en vecindades se conoce como movimiento vecinal. Entonces lo que se busca es optimizar el tiempo en realizar ese tipo de movimientos. Mediante la propuesta realizada por los autores de este artículo.

El algoritmo de generación y búsqueda en vecindad se probó para un individuo de tamaño 10 con un tamaño de vecindad de 45 vecinos generados con el fin de probar la funcionalidad del algoritmo. Una vez probada la funcionalidad se implementará para instancias grandes y por último como trabajo futuro, implementarlo en el algoritmo genético, el algoritmo de generación y búsqueda en vecindad se muestra en la Figura 5.

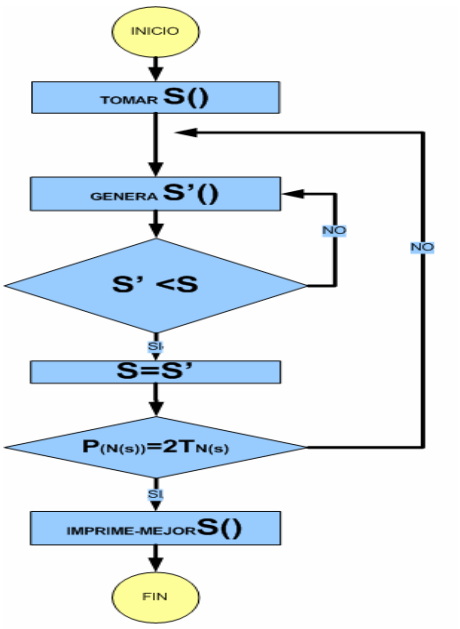

Fig. 5. Algoritmo de generación y búsqueda en vecindad tipo Or-modificada

Como lo muestra la figura5 el algoritmo de vecindad parte de una solución *s* tomada del espacio de soluciones y a partir de esa solución se genera una vecindad de tamaño $T_{N(s)}$ al generar un nuevo vecino *s'* se va comparando si es mejor al actual si lo es entonces lo reemplaza como el mejor vecino y así

sucesivamente hasta terminar con todos los vecinos generados a partir de la solución. En la Figura 6 se muestra el mecanismo de cómo se genera un vecino.

Fig. 6. Generación de un vecino

Como ejemplo para ilustrar se toma un individuo formado por 10 nodos y se inicia con el individuo 0 se generan dos aleatorios y se realiza el cambio, para el siguiente vecino se vuelve a tomar el individuo 0 y se vuelven a generar diferentes par de aleatorios y se realiza el cambio así sucesivamente hasta la condición de paro. En el proceso se va comparando, si el vecino generado es mejor que el anterior, se realiza el cambio; la comparación se realiza en base a la función de evaluación. En este caso la función objetivo a evaluar es la del problema del transporte con ventanas de tiempo. Para el propósito de este documento se propone solo el análisis y el diseño de la estructura.

5 Conclusiones y trabajos futuros

El análisis realizado en este documento sirve de punto de partida para conseguir minimizar el tiempo de ejecución de un algoritmo heurístico aplicado a un problema de optimización combinatoria mediante una vecindad. El análisis de tiempos realizado al algoritmo genético para el problema del transporte con ventanas de tiempo generó resultados que ayudaron a identificar el nivel en que el algoritmo consume mayor tiempo de ejecución. Dicho análisis sirvió para identificar donde se aplicará la estructura de vecindad.

Con miras a probar la propuesta expresada en este documento se pretende realizar la implementación del algoritmo de vecindad al algoritmo genético y

realizar la experimentación y comparación para detectar que tan buena o no resultó la estructura de vecindad aplicada a un algoritmo genético para el problema del transporte con ventanas de tiempo. Con base en el análisis de resultados proponer un criterio de paro de la búsqueda en la vecindad con motivo de aumentar la eficiencia del algoritmo.

Referencias

1. VONDOURIS, C. and TSANG, E.,: "Guided Local Search for the Traveling Salesman Problem". European Journal of Operations Research. Vol. 113, pp 469-499. (1999).
2. FEO, T. A. and RESENDE, M. G. C.: "A Probabilistic heuristic for a computationally difficult Set Covering Problem". Operations Research Letters. Vol, 8, pp 67-71. (1989).
3. FEO, T. A. and RESENDE, M. G. C.: "Greedy Randomized Adaptive Search Procedures". Journal of Global Optimization. Vol. 2, pp 1-27. (1995).
4. ROSING, K. E. and REVELLE, C. S. "Heuristic Concentration: Two Stage solution Construction". European Journal of Operational Research. Vol. 97, pp 75-86. (1997).
5. MOSCATO, P.: "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms". Caltech Concurrent Computation Program, C3P Report 826. (1989).
6. GLOVER, F: "Tabu Search: Part I." ORSA Journal on Computing. Vol. 1, pp.190-206. (1.989).
7. GLOVER,F.: "Tabu Search: Part II." ORSA Journal on Computing. Vol. 2, pp. 4-32. (1990).
8. LAGUNA, M.: "Scatter Search". Aparecerá en Handbook of Applied (1999). Optimization, P. M. Pardalos and M. G. C. Resende (eds). Oxford Academic Press.
9. KIRPATRICK S., GELATT C. D. and VECCHI M. P.: "Optimization by Simulated Annealing". IBM Research Report RC 9355. (1982).
10. KIRPATRICK S., GELATT C. D. and VECCHI M. P.: "Optimization by Simulated Annealing". Science, Vol. 220, pp 671-680. (1983)
11. A. A. MARTÍNEZ MORALES.: "Algoritmo Basado en Tabu Search para el Cálculo del Índice de Transmisión de un Grafo". Departamento de computación Facultad de ciencias y tecnología. FARAUTE de Ciencias y Tecnología, Vol. 1, No. 1, páginas 31-39. Universidad de Carabobo, Valencia, Estado Carabobo, Venezuela (2006).
12. PAOLO TOTH and DANIELLE VIGO.: The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications. Society of Industrial and Applied Mathematics. Philadelphia. USA.(2001).
13. WHITLEY D.: "A genetic Algorithm Tutorial", Tech. Report CS-93-103, Colorado State University. (1993).
14. OR, I.:"Traveling Salesman Type Combinatorial Problems y their Relations to the Logistics of Blood Banking." Ph. Thesis. Dpt. of Industrial Engineering y Management
Sciences, Northwestern Univ. (1.976).

15. LIN, S.: "Computer Solutions to the Traveling Salesman Problem". Bell Syst. Tech. Jou. Vol. 44, pp 2245-2269. (1965).
16. LIN, S. y KERNIGHAN, B. W.: "An Effective Heuristic Algorithm for the Traveling Salesman Problem". Operations Research. Vol. 20, pp 498-516. (1973).
17. PACHECO, J. y DELGADO, C.: "Resultados de Diferentes Experiencias con Búsqueda Local Aplicadas a Problemas de Rutas". Revista Electrónica Rect@. ASEPUMA, vol.2, nº 1, pp. 54- 81. (2000).
18. GAREY, M. R., JOHNSON, D.S.: Computers and intractability, A Guide to the theory of NP-Completeness. W.H.Freeman and Company, New York. USA.ed. (2003).
19. BARAJAS,N.: Estado del arte del problema de ruteo de vehículos (VRP) Colombia.Nov.(2006).
20. BAKER, B.M. and AYECHEW, M.A.: A genetic algorithm for the vehicle routing problem. Computers and operations research, 30(5):787-800, Apr. 2003.
21. BARBAROSOGLU G. and OZGUR D.: A tabu search algorithm for the vehicle routing problem. Computers and operations research, 26(3):255-270, Mar. (1999).
22. GLOVER F.: handbook of metaheuristics. Kluwer Academic plublishers, (2002).
23. LAPORTE, G., GENDREAW, M., POTVIN J. Y. and SEMET, F.: Classical and modern heuristics for the vehicle routing problem. International transactions in operational research, 7(4-5):285-300, sept. (2000).
24.SMITH, J.E. and EIBEN, A.E.: Introduction to evolutionary computing. (2003).
25.TOTH, P.: Optimization engineering techniques for the exact solution of np-hard combinatorial optimization problems. European journal of operational research, 125(2):222-238, sept. (2000).
26. DE-ALBA Romenus. K.: Un procedimiento heurístico para un problema de diseño de redes multiproducto con capacidad finita y cargos fijos, Facultad de ingeniería mecánica y eléctrica. Sn Nicolás de los Garza, N.L.(2004).
27. CHRISTOFIDES, N., MINGOZZI, A., and TOTH, P.: the vehicle routing problem. In combinatorial optimization, P. Toth, N. Christofides, R. Mingpzzi and C. Sandi (Eds.), John Wiley, New York, 315-338, 1989.
28. DESROCHERS, M., DESROCHERS, J. and SOLOMON, M.: A new optimization algorithm for the vehicle routing problem with time windows, operations research 40(2), 1992.
29. SAVELSBERGH, M.W.P.:Local Search for Routing Problems with time windows. Annals for operations research 4, 285-305, 1985.
30. SOLOMON, M.M. and DESROISERS, J.: Time window constrained routing and scheduling problems: A survey. Transportation science 22(1), 1-11, 1986.