IMPLEMENTACIÒN DE ALGORITMOS GENÉTICOS EN EL PROBLEMA DE RUTEO DE VEHÍCULOS CON CAPACIDAD LIMITADA

MAURICIO RANGEL SALAZAR

CODIGO 199811899

UNIVERSIDAD DE LOS ANDES

FACULTAD DE INGENIERÍA

DEPARTAMENTO INGENIERÍA INDUSTRIAL

BOGOTÁ

DICIEMBRE 2003

IMPLEMENTACIÒN DE ALGORITMOS GENÉTICOS EN EL PROBLEMA DE RUTEO DE VEHÍCULOS CON CAPACIDAD LIMITADA

MAURICIO RANGEL SALAZAR

CODIGO 199811899

TESIS DE GRADO

ASESOR DE TESIS

Dr. FIDEL TORRES

Profesor Asociado Universidad de Los Andes

UNIVERSIDAD DE LOS ANDES

FACULTAD DE INGENIERÍA

DEPARTAMENTO INGENIERÍA INDUSTRIAL

BOGOTÁ

DICIEMBRE 2003

A mi familia,

que me apoyó durante estos años.

INDICE

INTRODUCCIÓN

Debido al mercado tan competitivo que existe hoy en día, bajar los costos es una necesidad para sobrevivir. Unas de las áreas donde se pueden reducir costos es en el área de distribución de los productos. Se busca alcanzar un proceso de distribución eficiente que maximice el uso de los recursos disponibles de la empresa, minimice el costo total y mantenga unos niveles de calidad competitivos en el mercado. Con el avance tecnológico y el bajo costo de los computadores, se están empezando a usar diferentes algoritmos matemáticos que necesitan de mucho poder de cómputo para resolver este problema. Uno de estos algoritmos es el Algoritmo Genético.

En esta investigación se hizo una implementación de los Algoritmos Genéticos en el lenguaje de programación JAVA. Se desarrolló una aplicación que implementa un algoritmo genético elitista y tres operadores genéticos para resolver varios problemas de ruteo. Se resolvieron diferentes tipos de problemas de diferentes tamaños y complejidades. Además, se estudió la relación entre algunos de los parámetros que componen el algoritmo genético.

La estructura del trabajo es la siguiente, en los dos primeros capítulos se presenta el marco teórico del trabajo. En el capítulo 1 se explica el Problema de Ruteo de Vehículos. En el capítulo 2 se explican los Algoritmos Genéticos. En el tercer capítulo se explica detalladamente las decisiones que se tomaron para poder implementar el algoritmo genético. En el cuarto capítulo se desarrollan experimentos estadísticos para medir los diferentes parámetros del algoritmo.

OBJETIVO GENERAL

Implementar un algoritmo genético aplicado al Problema de Ruteo de Vehículos con Capacidad Limitada.

OBJETIVOS ESPECÍFICOS

1. Plantear y entender el Problema de Ruteo de Vehículos
2. Entender e implementar un algoritmo Genético
3. Analizar los diferentes parámetros que afectan un Algoritmo Genético.

1. EL PROBLEMA DE RUTEO DE VEHÍCULOS (VRP)

La distribución de bienes concierne el servicio en un periodo de tiempo, a un grupo de clientes, por un conjunto de vehículos, que están localizados en uno o más depósitos, que son operados por un grupo de conductores y se movilizan utilizando una red vial.

El problema de ruteo de vehículos (vrp) pide la determinación de un conjunto de rutas, cada una recorrida por un solo vehículo que empieza y termina en el depósito y se cumplen todas las demandas de los clientes, todas las restricciones operacionales son satisfechas y el costo global de transporte es minimizado.

La red vial, utilizada para el transporte de los bienes, es generalmente descrita a través de un grafo, cuyos nodos representan los clientes y el depósito. Y los arcos, el costo de ir de un cliente a otro. Este costo puede representar la distancia, es decir, el tiempo de ir de un nodo a otro. Los arcos pueden ser dirigidos o no, dependiendo en las características de la red vial.

Las características típicas que tienen los clientes son:

- La demanda del producto.
- Una ventana de tiempo en la que pueden recibir productos.
- Subconjunto de vehículos disponibles que pueden ser usados para servir al cliente.

Cada depósito puede estar caracterizado por el número y tipo de vehículos asociados con él y por la cantidad total de productos que puede albergar.

El transporte de los productos se realiza por una flota de vehículos cuyo tamaño puede ser fijo o definido de acuerdo a los requerimientos de los clientes.

Las características de los vehículos son:

- El vehículo puede estar asociado a un depósito, o puede terminar en cualquier depósito.
- El vehículo tiene una capacidad que puede ser expresada en peso máximo, volumen, número de productos que puede transportar.
- Costos asociados a la utilización del vehículo.

Las rutas deben satisfacer restricciones operacionales, que dependen en la naturaleza de los productos transportados y en el nivel de calidad.

Algunas restricciones operacionales típicas son:

- En cada ruta la capacidad del vehículo no puede ser sobrepasada.
- Los clientes servidos en una ruta pueden requerir sólo recepción o entrega de productos, o ambas posibilidades.
- Los clientes pueden ser servidos durante una ventana de tiempo.

Las restricciones de precedencia pueden ser impuestas en el orden en que se deben visitar los clientes. Un tipo de restricciones de precedencia es:

- Un cliente que sea servido en la misma ruta de un conjunto de clientes o que el cliente sea atendido antes de un conjunto de clientes.

La evaluación del costo global de las rutas y el chequeo de las restricciones impuestas sobre ellas, requieren conocimiento del costo y tiempo de transporte entre los clientes y los depósitos. Para facilitar esta evaluación, el grafo vial original es transformado en un grafo completo, con costo infinito entre nodos que no están conectados.

Para cada par de vértices i y j del grafo completo, un arco (i , j) es definido con costo c_{ij} por el costo mínimo que hay para ir de i a j.

Varios objetivos pueden ser considerados para el VRP, como son:

- Minimización del costo total global.
- Minimización del número de rutas utilizadas.

- Balanceo de las rutas para el tiempo de transporte y carga del vehículo.

Además, hay que considerar versiones estocásticas o dependientes del tiempo del problema, en las cuales sólo hay un conocimiento parcial de las demandas de los clientes a priori o de los costos asociados a los arcos del grafo vial.

1.1 VARIANTES DEL VRP

1.1.1 VRP con Capacidad (CVRP

El CVRP, es la variante más sencilla y básica que existe del VRP. En ella las demandas de los clientes son determinísticas, conocidas de antemano y no pueden ser divididas.

Los vehículos son idénticos y están basados en un único depósito. La restricción de capacidad para los vehículos es impuesta.

Los costos de los arcos se miden como la distancia euclidiana entre los nodos.

El objetivo es minimizar la distancia total de todas las rutas, manteniendo las restricciones de capacidad del vehículo y atendiendo a todos los clientes.

El CVRP puede ser descrito como el siguiente problema de Grafos:

Sea G = (V, A) un grafo completo, donde V = {0,..n} son los nodos y A es el conjunto de arcos. Los nodos i = 1,.,.,.,.,n corresponden a los clientes, mientras el nodo 0 corresponde al depósito.

Un costo positivo c_{ij} está asociado a cada arco (i , j) ϵ A y representa el costo de transporte para ir del nodo i al j. Los arcos de ciclo (i , i) no son permitidos definiendo c_{ii} = infinito. Si G es un grafo dirigido, la matriz de costos es asimétrica, y el problema se llama CVRP asimétrico.

Cuando el grafo es no dirigido, la matriz de costos es simétrica. Este es el caso de cuando los nodos representan puntos en un plano cartesiano y el costo entre los nodos es

hallado a través de las distancias euclidianas. Cada cliente i (i = 1, 2 , . , . , . , n) está asociado con una demanda positiva d_i, y el depósito tienen una demanda ficticia do = 0. Un conjunto de vehículos idénticos, cada uno con capacidad C, están disponibles en el depósito. Para asegurar la factibilidad se asume que d_i <= C para todos los clientes i = 1,2, 3. , . , . , n. Cada vehículo puede realizar máximo una ruta .

El CVRP consiste en encontrar un conjunto de rutas que minimicen el costo total, definido como la suma de los costos de los arcos en todas las rutas, tal que:

- Cada ruta empieza y termina en el depósito.
- Cada cliente es visitado solamente una vez.
- La suma de las demandas de una ruta no supera la capacidad del camión.

1.1.2 VRP con Ventanas de Tiempo VRPTW (VRP with Time Windows):

Esta variante es una extensión del CVRP, en la cuál además de las restricciones de capacidad, a cada cliente i se le asocia un intervalo de tiempo [a_i, b_i], llamado ventana de tiempo. También se define el instante de tiempo en que los vehículos salen del depósito, el tiempo de viaje entre los nodos t_{ij} y el tiempo de servicio s_i para cada cliente i. El servicio de cada cliente debe empezar dentro de su ventana de tiempo asociada, incluso en caso de que el vehículo llegue más temprano, se le permite esperar hasta que la ventana de tiempo empiece.

El VRPTW consiste en encontrar un conjunto de rutas que minimicen el costo total, tal que:

- Cada ruta empieza y termina en el depósito.
- Cada cliente es visitado solamente una vez.
- La suma de las demandas de una ruta no supera la capacidad del camión.

- Para cada cliente i, el servicio empieza dentro de su ventana de tiempo asociada $[a_i , b_i]$ y el vehículo se demora s_i unidades de tiempo.

1.1.3 VRP con BackHauls (VRPB)

Esta versión es una extensión del CVRP en la cual el conjunto de clientes se divide en dos subconjuntos. El primer subconjunto: L, contiene n clientes, cada uno contiene una demanda de producto conocida. El segundo subconjunto: B contiene m clientes los cuales tienen una oferta del producto, la cual debe ser recogida.

En VRPB existe una restricción de precedencia entre los clientes del subconjunto L y B que consiste en que cuando una ruta sirva a los dos tipos de clientes todos los clientes del subconjunto L deben ser servidos antes que los del subconjunto B.

VRPB consiste en determinar un conjunto de rutas que minimicen el costo total, tal que:

- Cada ruta empieza y termina en el depósito.
- Cada cliente es visitado solamente una vez.
- La suma de las demandas de una ruta no supera la capacidad del camión.
- En un ruta todos los clientes del subconjunto L son atendidos primeros que los del subconjunto B.

1.1.4 VRP con Recolección y Entrega (VRPPD

En la versión básica de VRPPD, a cada cliente i se le asocia dos cantidades d_i y p_i que representan las cantidades de producto a ser entregadas (d_i) y la de ser recogidas (p_i). Algunas veces sólo se usa una demanda $d_i = d_i – p_i$ para cada cliente, por lo que pueden haber demandas negativas.

Se asume que en cada locación del cliente, la entrega se hace antes de la recolección; por lo tanto la carga del camión es definida como la carga inicial, menos todas las entregas, más todas las recolecciones.

VRPPD consiste en determinar un conjunto de rutas que minimicen el costo total, tal que:

- Cada ruta empieza y termina en el depósito.
- Cada cliente es visitado solamente una vez.
- La carga del vehículo en una ruta debe ser positiva en todo momento y no puede superar la capacidad del camión.

2. ALGORITMOS GENÉTICOS (GA)

[1]En la naturaleza el proceso evolucionario, ocurre cuando las siguientes condiciones son satisfechas:

- Una entidad tiene la habilidad de reproducirse.
- Hay una población de tales entidades.
- Existe variedad en estas entidades.
- La diferencia en sobrevivir en el ambiente es asociado a esta variedad.

En la naturaleza, la variedad es manifestada como la variación de los cromosomas de las entidades en la población. Ésta variación es traducida en una variación en la estructura y el comportamiento de las entidades en el ambiente. Ésta variación se ve reflejada en la diferencia de la tasa de reproducción y supervivencia. Las entidades que están más aptas para realizar actividades en su ambiente sobreviven y se reproducen a una tasa más alta que las menos aptas. Éste es el concepto de "La supervivencia del más apto y la selección natural descrita por Charles Darwin en "On the Origin of species by Jeans of natural selection (1859)". Sobre un período de tiempo y muchas generaciones, la población como un todo viene a contener más individuos cuyos cromosomas son traducidos en comportamientos y estructuras que les permiten realizar mejor sus tareas, sobrevivir y reproducirse en el medio ambiente. Por lo tanto, durante el tiempo la estructura y el comportamiento de los individuos cambia debido a la selección natural. Cuando estos cambios son notables, se dice que la población ha evolucionado. En este proceso la estructura nace del "fitness" (puntuación) de la entidad.

En la práctica la presencia de la primera de las cuatro condiciones mencionadas anteriormente, es una condición crucial para empezar el proceso evolutivo.

[1] Genetic Programming, Jhon R. Koza

Cualquier problema puede generalmente ser formulado en términos genéticos. Una vez formulado en éstos términos, los problemas pueden ser solucionados a través de Algoritmos Genéticos.

Los Algoritmos Genéticos simulan el proceso evolucionario de Darwin y las operaciones genéticas en los cromosomas. En la naturaleza los cromosomas están constituidos por una secuencia de nucleótidos compuestos por Adenina, Guanina, Timina y Citosina.

Los cromosomas de los hijos contienen cadenas de nucleótidos de los padres, por lo tanto algunas secuencias pasan de padres a hijos manteniéndose las ventajas competitivas para sobrevivir. Ocasionalmente ocurre mutación en los cromosomas.

2.1 Anatomía de un algoritmo genético simple

[2]Los *algoritmos genéticos* son métodos sistemáticos para la resolución de problemas de búsqueda y optimización que aplican a estos los mismos métodos de la evolución biológica: selección basada en la población, reproducción sexual y mutación.

Los algoritmos genéticos tratan de resolver un conjunto de problemas que tratan de hallar $(x_i,...,x_n)$ tales que F $(x_i,...,x_n)$ sea máximo o mínimo. En un algoritmo genético, tras parametrizar el problema en una serie de variables $(x_i,...,x_n)$ se codifican en un cromosoma. Todos los operadores utilizados por un algoritmo genético se aplicarán sobre estos cromosomas, o sobre poblaciones de ellos. En el algoritmo genético va implícito el método para resolver el problema; son solo parámetros de tal método los que están codificados, a diferencia de otros algoritmos evolutivos como la programación genética.

[2] http://geneura.ugr.es/~jmerelo/ie/ags.htm

Las soluciones codificadas en un cromosoma *compiten* para ver cuál constituye la mejor solución (aunque no necesariamente la mejor de todas las soluciones posibles). El *ambiente,* constituido por las otras soluciones, ejercerá una presión selectiva sobre la población, de forma que sólo los mejor adaptados (aquellos que resuelvan mejor el problema) sobrevivan o hereden su material genético a las siguientes generaciones, igual que en la evolución de las especies. La diversidad genética se introduce mediante mutaciones y reproducción sexual.

En la Naturaleza lo único que hay que optimizar es la supervivencia, y eso significa a su vez maximizar diversos factores y minimizar otros. Un algoritmo genético, sin embargo, se usará para optimizar habitualmente sólo una función, no diversas funciones relacionadas entre sí simultáneamente. Este tipo de optimización denominada optimización multimodal, también se suele abordar con un algoritmo genético especializado.

Por lo tanto, un algoritmo genético consiste en lo siguiente: hallar de qué parámetros depende el problema, codificarlos en un cromosoma, y aplicar los métodos de la evolución: selección y reproducción sexual con intercambio de información y alteraciones que generan diversidad

2.2 Codificación de las variables

Los algoritmos genéticos requieren que el conjunto se codifique en un *cromosoma*. Un cromosoma es una representación de una solución al problema. Cada cromosoma tiene varios genes, que corresponden a parámetros del problema. Para poder trabajar con estos genes en el computador, es necesario codificarlos en una *cadena*, es decir, una secuencia de símbolos (números o letras) que generalmente va a estar compuesta de 0s y 1s.

Hay otras codificaciones posibles, usando alfabetos de diferente cardinalidad; sin embargo, uno de los resultados fundamentales en la teoría de algoritmos genéticos, el *teorema de los esquemas*, afirma que la codificación óptima, es decir, aquella sobre la que los algoritmos genéticos funcionan mejor, es aquella que tiene un alfabeto de cardinalidad 2. Sin embargo, en CVRP debido a la complejidad del problema es necesario utilizar un alfabeto de complejidad n+1, donde n es el número de clientes.

La mayoría de las veces, una codificación correcta es la clave de una buena resolución del problema. Generalmente, la regla heurística que se utiliza es la llamada *regla de los bloques de construcción*, es decir, los parámetros relacionados entre sí deben de estar cercanos en el cromosoma. Por ejemplo, si se quiere codificar los pesos de una red neuronal, una buena elección será poner juntos todos los pesos que salgan de la misma neurona de la capa oculta (también llamada codificación *fregona*). En esta, todos los pesos señalados con trazo doble se codifican mediante grupos de bits o bytes sucesivos en el cromosoma.

En todo caso, se puede ser bastante creativo con la codificación del problema, teniendo siempre en cuenta la regla anterior. Esto puede llevar a usar cromosomas bidimensionales, o tridimensionales, o con relaciones entre genes que no sean puramente lineales de vecindad. En algunos casos, cuando no se conoce de antemano el número de variables del problema, caben dos opciones: codificar también el número de variables, fijando un número máximo, o bien, lo cual es mucho más natural, crear un cromosoma que pueda variar de longitud. Para ello, claro está, se necesitan operadores genéticos que alteren la longitud.

Normalmente, la codificación es estática, pero en casos de optimización numérica, el número de bits dedicados a codificar un parámetro puede variar, o incluso lo que

representen los bits dedicados a codificar cada parámetro. Algunos paquetes de algoritmos genéticos adaptan automáticamente la codificación según van convergiendo los bits menos significativos de una solución.

2.3 Algoritmo

Para comenzar la competición, se generan aleatoriamente una serie de cromosomas. El algoritmo genético procede de la forma siguiente:

1. Evaluar la puntuación (*fitness*) de cada uno de los genes.
2. Permitir a cada uno de los individuos reproducirse, de acuerdo con su puntuación.
3. Emparejar los individuos de la nueva población, haciendo que intercambien material genético, y que alguno de los bits de un gen se vea alterado debido a una *mutación* espontánea.

Cada uno de los pasos consiste en una actuación sobre las cadenas de bits, es decir, la aplicación de un *operador* a una cadena binaria. Se les denominan, por razones obvias, *operadores genéticos*, y hay tres principales: *selección, crossover* o recombinación y *mutación;* aparte de otros operadores genéticos no tan comunes, todos ellos se verán a continuación.

Un algoritmo genético tiene también una serie de parámetros que se tienen que fijar para cada ejecución, como los siguientes:

- **Tamaño de la población**: debe ser suficiente para garantizar la diversidad de las soluciones, y, además, tiene que crecer más o menos con el número de bits

del cromosoma, aunque nadie ha aclarado cómo tiene que hacerlo. Por supuesto, depende también del computador en el que se esté ejecutando.

- **Condición de terminación:** lo más habitual es que la condición de terminación sea la convergencia del algoritmo genético o un número prefijado de generaciones.

2.4 Evaluación y selección

Durante la evaluación, se decodifica el gen, convirtiéndose en una serie de parámetros de un problema, se halla la solución del problema a partir de esos parámetros, y se le da una puntuación a esa solución en función de lo cerca que esté de la mejor solución. A esta puntuación se le llama *fitness*. Por ejemplo, se quiere hallar el máximo de la función, una parábola invertida con el máximo en x=1. En este caso, el único parámetro del problema es la variable x. La optimización consiste en hallar un x tal que $F(x)$ sea máximo. Se creará, pues, una población de cromosomas, cada uno de los cuales contiene una codificación binaria del parámetro x. Se hará de la forma siguiente: cada byte, cuyo valor está comprendido entre 0 y 255, se transformará para ajustarse al intervalo [-1,1], donde se quiere hallar el máximo de la función.

Valor binario	Decodificación	Evaluación f(x)
10010100	21	0,9559
10010001	19	0,9639
101001	-86	0,2604
1000101	-58	0,6636

El “fitness” determina siempre los cromosomas que se van a reproducir, y aquellos que se van a eliminar, pero hay varias formas de considerarlo para seleccionar la población de la siguiente generación:

- Usar el orden, o rango, y hacer depender la probabilidad de permanencia o evaluación de la posición en el orden.
- Aplicar una operación al fitness para escalarlo..

Una vez evaluado el "fitness", se tiene que crear la nueva población teniendo en cuenta que los *buenos* rasgos de los mejores se transmitan a ésta. Para ello, hay que seleccionar una serie de individuos encargados de tan ardua tarea. Y esta selección, y la consiguiente reproducción, se pueden hacer de dos formas principales:

- *Basado en el rango*: en este esquema se mantiene un porcentaje de la población, generalmente la mayoría, para la siguiente generación. Se coloca toda la población por orden de "fitness", y los *M* menos dignos son eliminados y sustituidos por la descendencia de alguno de los *M* mejores con algún otro individuo de la población. A este esquema se le pueden aplicar otros criterios; por ejemplo, se crea la descendencia de uno de los paladines/amazonas, y esta sustituye al más parecido entre los perdedores. Esto se denomina *crowding*, y fue introducido por **DeJong.** Cuando nace una nueva criatura, se seleccionan CF individuos de la población, y se elimina al más parecido a la nueva criatura. Una variante de este es el muestreo estocástico universal, que trata de evitar que los individuos con más "fitness" copen la población; en vez de dar la vuelta a una ruleta con una ranura, da la vuelta a la ruleta con N ranuras, tantas como la población; de esta forma, la distribución estadística de descendientes en la nueva población es más parecida a la real.
- *Rueda de ruleta*: se crea un pool genético formado por cromosomas de la generación actual, en una cantidad proporcional a su "fitness." Si la proporción hace que un individuo domine la población, se le aplica alguna operación de

escalado. Dentro de este *pool*, (grupo) se cogen parejas aleatorias de cromosomas y se emparejan, sin importar incluso que sean del mismo progenitor (para eso están otros operadores, como la mutación).

- *Elitismo*: Se denomina Elitismo, el proceso por el cual determinados elementos con una adaptación especialmente buena tienen determinados privilegios - nunca mueren, proporción alta de pasos en que se reproduce uno de la élite con otro al azar-. Sin embargo, en fases iniciales es peligroso, ya que puede producirse que una élite de súper individuos acabe con la diversidad genética del problema (Mínimos Locales). Para ello, lo que se puede hacer es escalar la función de adaptación en las primeras fases del algoritmo -de forma que las diferencias entre la élite y el pueblo sean menores- y superescalar la función de adaptación al final del algoritmo, para evitar un bloqueo de la convergencia. Sin embargo, se ha tomado una decisión distinta. Para definir el Elitismo se necesitan dos parámetros que son el Porcentaje de Creación y el Porcentaje de la Población Creada. El primero, hace referencia al porcentaje de la población inicial que va a ser tenida en cuenta para crear la nueva población. Éste porcentaje puede crear toda la población, o sólo una parte de ella. Esta decisión es el segundo parámetro del Elitismo. Así por ejemplo, si se quiere que sólo el 10% de la población actual genere el 60 % de la siguiente población, los parámetros son 10% y 60% respectivamente. Cuando no existe elitismo se dice que el porcentaje de creación es del 100%.

- *Por torneo*: escoge un subconjunto de individuos de acuerdo con una de las técnicas anteriores -habitualmente, aleatoria o estocástica- y de entre ellos selecciona el más adecuado por otra técnica -habitualmente, determinística de tipo «el mejor» o «el peor»-. Esta técnica tiene la ventaja de que permite un

cierto grado de elitismo -el mejor nunca va a morir, y los mejores tienen más probabilidad de reproducirse y de emigrar que los peores- pero sin producir una convergencia genética prematura, si la población es, al menos, un orden de magnitud superior al del número de elementos involucrados en el torneo. La selección por torneo ha sido la técnica empleada en este algoritmo para decidir los cromosomas que van a ser tenidos en cuenta en las siguiente generación.

2.5 Crossover

Consiste en el intercambio de material genético entre dos cromosomas (a veces más, como el *operador orgía* propuesto por Eiben et al.). El *crossover* es el principal operador genético, hasta el punto que se puede decir que no es un algoritmo genético si no tiene *crossover*, y, sin embargo, puede serlo perfectamente sin mutación, según descubrió Holland. El *teorema de los esquemas* confía en él para hallar la mejor solución a un problema, combinando soluciones parciales.

Para aplicar el crossover, entrecruzamiento o recombinación, se escogen aleatoriamente dos miembros de la población. No pasa nada si se emparejan dos descendientes de los mismos padres; ello garantiza la perpetuación de un individuo con buena puntuación (y, además, algo parecido ocurre en la realidad; es una práctica utilizada, por ejemplo, en la cría de ganado, llamada *inbreeding*, y destinada a potenciar ciertas características frente a otras). Sin embargo, si esto sucede demasiado a menudo, puede crear problemas: toda la población puede aparecer dominada por los descendientes de algún gen, que además, puede tener caracteres no deseados. Esto se suele denominar en otros métodos de optimización *estancarse en un mínimo local*, y es uno de los principales problemas con los que se enfrentan los que aplican algoritmos genéticos.

En cuanto al teorema de los esquemas, se basa en la noción de *bloques de construcción*. Una buena solución a un problema está constituido por unos buenos bloques, igual que una buena máquina está hecha por buenas piezas. El crossover es el encargado de mezclar bloques buenos que se encuentren en los diversos progenitores, y que serán los que den a los mismos una buena puntuación. La presión selectiva se encarga de que sólo los buenos bloques se perpetúen, y poco a poco vayan formando una buena solución. El *teorema de los esquemas* viene a decir que la cantidad de *buenos bloques* se va incrementando con el tiempo de ejecución de un algoritmo genético, y es el resultado teórico más importante en algoritmos genéticos.

El intercambio genético se puede llevar a cabo de muchas formas, pero hay dos grupos principales:

- *Crossover n-puntos*: los dos cromosomas se cortan por n puntos, y el material genético situado entre ellos se intercambia. Lo más habitual es un crossover de un punto o de dos puntos.

- *Crossover uniforme*: se genera un patrón aleatorio de 1s y 0s, y se intercambian los bits de los dos cromosomas que coincidan donde hay un 1 en el patrón. O bien se genera un número aleatorio para cada bit, y si supera una determinada probabilidad se intercambia ese bit entre los dos cromosomas.

- *Crossover especializados*: en algunos problemas, aplicar aleatoriamente el crossover da lugar a cromosomas que codifican soluciones inválidas; en este caso hay que aplicar el crossover de forma que genere siempre soluciones válidas. Un ejemplo de estos es el operador de crossover que se va a utilizar en el CVRP.

2.6 Mutación

En la Evolución, una mutación es un suceso bastante poco común (sucede aproximadamente una de cada mil replicaciones), como ya se ha visto anteriormente. En la mayoría de los casos las mutaciones son letales, pero en promedio, contribuyen a la diversidad genética de la especie. En un algoritmo genético tendrán el mismo papel, y la misma frecuencia (es decir, muy baja).

Una vez establecida la frecuencia de mutación, por ejemplo, uno por mil, se examina cada bit de cada cadena cuando se vaya a crear la nueva criatura a partir de sus padres (normalmente se hace de forma simultánea al crossover). Si un número generado aleatoriamente está por debajo de esa probabilidad, se cambiará el bit (es decir, de 0 a 1 o de 1 a 0). Si no, se dejará como está. Dependiendo del número de individuos que haya y del número de bits por individuo, puede resultar que las mutaciones sean extremadamente raras en una sola generación.

No hace falta decir que no conviene abusar de la mutación. Es cierto que es un mecanismo generador de diversidad, y, por tanto, la solución cuando un algoritmo genético está estancado, pero también es cierto que reduce el algoritmo genético a una búsqueda aleatoria. Siempre es más conveniente usar otros mecanismos de generación de diversidad, como aumentar el tamaño de la población, o garantizar la aleatoriedad de la población inicial.

Este operador, junto con la anterior y el método de selección de ruleta, constituyen un *algoritmo genético simple,* sga.

2.7 Otros Operadores

No se usan en todos los problemas, sino sólo en algunos, y en principio su variedad es infinita. Generalmente son operadores que exploran el espacio de soluciones de una forma más ordenada, y que actúan más en las últimas fases de la búsqueda, en la cual se pasa de soluciones "casi buenas" a "buenas" soluciones.

2.7.1 Cromosomas de longitud variable

Hasta ahora se han descrito cromosomas de longitud fija, donde se conoce de antemano el número de parámetros de un problema. Pero hay problemas en los que esto no sucede. Por ejemplo, el CVRP, donde dado un conjunto de clientes, se quiere agruparlo en conjunto de rutas, y no se puede saber de antemano el número ni la longitud de cada una. O en diseño de redes neuronales, puede que no se sepa (de hecho, nunca se sabe) cuántas neuronas se van a necesitar.

2.7.2 Operadores de nicho (ecológico)

Otros operadores importantes son los operadores de *nicho*. Estos operadores están encaminados a mantener la diversidad genética de la población, de forma que cromosomas similares sustituyan sólo a cromosomas similares, y son especialmente útiles en problemas con muchas soluciones; un algoritmo genético con estos operadores es capaz de hallar todos los máximos, dedicándose cada especie a un máximo. Más que operadores genéticos, son formas de enfocar la selección y la evaluación de la población.

Uno de las formas de llevar esto a cabo ya ha sido nombrada, la introducción del *crowding (apiñamiento)*. Otra forma es introducir una función de división o *sharing*,

que indica cuán similar es un cromosoma al resto de la población. La puntuación de cada individuo se dividirá por ésta función, de forma que se facilita la diversidad genética y la aparición de individuos diferentes.

También se pueden restringir los emparejamientos, por ejemplo, a aquellos cromosomas que sean similares. Para evitar las malas consecuencias del inbreeding que suele aparecer en poblaciones pequeñas, estos períodos se intercalan con otros períodos en los cuales el emparejamiento es libre.

2.7.3 Operadores especializados

En una serie de problemas hay que restringir las nuevas soluciones generadas por los operadores genéticos, pues no todas las soluciones generadas van a ser válidas, sobre todo en los problemas con restricciones. Por ello, se aplican operadores que mantengan la estructura del problema. Otros operadores son simplemente generadores de diversidad, pero la generan de una forma determinada:

1. **Zap**: en vez de cambiar un solo bit de un cromosoma, se cambia un gen completo de un cromosoma.
2. **Creep:** este operador aumenta o disminuye en 1 el valor de un gen; sirve para cambiar suavemente y de forma controlada los valores de los genes.
3. **Transposición:** similar al crossover y a la recombinación genética, pero dentro de un solo cromosoma; dos genes intercambian sus valores, sin afectar al resto del cromosoma. Similar a este es el operador de **eliminación-reinserción**, en el que un gen cambia de posición con respecto a los demás. Éste operador va a ser utilizado en la solución del CVRP.

3. DECISIONES AL IMPLEMENTAR EL ALGORITMO GENÉTICO

3.1. Representación del Problema de Ruteo con Capacidad

Como se había explicado anteriormente, el CVRP puede ser descrito como el siguiente problema de Grafos:

Sea G = (V, A) un grafo completo, donde V = {0,..n} son los nodos y A es el conjunto de arcos. Los nodos i = 1,.,.,.,.,n corresponden a los clientes, mientras el nodo 0 corresponde al depósito.

Un costo positivo c_{ij} está asociado a cada arco (i , j) ϵ A y representa el costo de transporte para ir del nodo i al j. Los arcos de ciclo (i , i) no son permitidos definiendo c_{ii} = infinito.

3.2. Criterio de codificación

Cómo se va a almacenar la información en el Cromosoma. Generalmente los cromosomas son representados por cadenas binarias (1 y 0). Debido a la complejidad de representar el conjunto de rutas en cadenas binarias, se ha decidido representarlo como la secuencia de clientes visitados en cada ruta, que empieza y termina con un depósito. Por ejemplo el siguiente cromosoma:

D 1 3 2 D 4 6 D 5 D

Representa una solución en la cual existen 3 rutas que son:

1-2-3

4-6

5

3.3 Criterio de tratamiento de individuos no factibles.

Cómo se van a tratar a los individuos que no cumplan las restricciones. Existen dos enfoques para el manejo de los cromosomas no factibles:

i) *No Aceptar Cromosomas no Factibles*: En este enfoque se restringe la viabilidad de cromosomas no factibles en la población. Esto se puede realizar creando operadores genéticos que creen sólo cromosomas factibles, o mediante unas funciones reconstructivas, que ”arreglan “ los cromosomas que no son factibles, haciéndolos factibles. La ventaja de este enfoque consiste en que los cromosomas de la población, son todos soluciones implantables y por lo tanto el material genético de ellos es más rico, en cuanto al “fitness”. La desventaja consiste en que al no permitir cromosomas no factibles, se pierden muchas combinaciones genéticas que podrían enriquecer los cromosomas factibles, o por ejemplo en el caso de los operadores binarios, la unión de dos cromosomas no factibles podría resultar en un cromosoma factible y con un fitness aceptable.

ii) *Penalizar la no factibilidad del Cromosoma*: Cada vez que un cromosoma viola una restricción se penaliza la función objetivo. Así la población puede tener cromosomas no factibles, pero los no factibles tendrán un costo muy alto en comparación con los factibles, haciéndolos tender a desaparecer (como en la naturaleza). Sin embargo, existe la posibilidad que éstos cromosomas no factibles, contribuyan en la generación de un mejor cromosoma.

En ésta implementación se utilizó el enfoque mixto de penalización y de reparación del cromosoma. La reparación de los cromosomas se realizaba después de cada operación genética y tiene como fin, evitar la repetición de clientes en un cromosoma.

La penalización se utilizó en las siguientes restricciones:

i. Satisfacer la Demanda de todos los clientes: Si en un cromosoma, falta un cliente, se penaliza con un costo elevado.

ii. Capacidad de los Camiones: Si alguna de las rutas de los cromosomas sobrepasan la capacidad del camión, se penaliza la función objetivo.

3.4 Criterio de inicialización.

Cómo se va a construir la población inicial del algoritmo genético. Para las investigaciones se ha implementado a través de la creación aleatoria de la población inicial, asemejando la Naturaleza. Se implementó a través del siguiente Algoritmo:

1 Se agrega un Depósito a la ruta.

2 Mientras hallan nodos no utilizados.

3 Se selecciona aleatoriamente un nodo de los nodos disponibles (no utilizados).

4 Si la demanda del nodo, sobrepasa la capacidad restante de la ruta:

3.1 Se asigna un depósito.

3.2 Se asigna el nodo.

3.3 Se recalcula la capacidad de la nueva ruta.

5 Si no:

3.1 Se asigna el nodo.

3.2 Se recalcula la capacidad de la ruta.

6 Vuelva al punto 2.

7 Cuando se acaban los nodos se asigna un depósito al final para cerrar la última ruta.

3.5 Criterio de parada.

Determina cuándo el algoritmo ha llegado a una solución aceptable. Desgraciadamente, no se sabe si puede existir una solución mejor que la que se dispone mediante un mecanismo formal, por la complejidad de las ecuaciones. De ahí que sea necesario el control de un experto humano para decidir cuando alguna de las soluciones temporales propuestas por el algoritmo son razonablemente buenas y puede ser interesante un estudio detallado de ellas. En esta implementación, el criterio de parada es el número de iteraciones del algoritmo, y es un parámetro de entrada, asignado por el usuario.

3.6 Función de adaptación.

Corresponde a la función de costo de la investigación operativa tradicional. En este caso la función de adaptación será el costo de las rutas del Cromosoma. Esta función se realizó siguiendo el siguiente algoritmo:

1 Mientras queden nodos en el cromosoma.

2 Se selecciona el nodo en la posición i.

3 Se suma al costo total el costo de ir de i-1 a i.

4 Se aumenta i en 1.

3.7 Operadores genéticos.

Se emplean para determinar cómo va a ser la nueva generación. Básicamente son los operadores unitarios y binarios, es decir, que operan a partir de un cromosoma o dos. En este caso se implementaron los siguientes:

SWAP: Es un operador unitario que intercambia la posición de dos genes dentro de un cromosoma. Se generan aleatoriamente dos números, entre el rango definido por el número de genes del cromosoma. Luego se ingresa el gen i en la posición j y el gen j en la posición i cambiándolos de posición. Por ejemplo en esta implementación del CVRP,

cada gen corresponde a un cliente o al depósito, el cual después del SWAP, genera una nueva ruta, la cuál puede o no ser factible.

Si se tiene el siguiente cromosoma:

D 1 **2** D **3** 5 D 4 7 D

Y se generan dos números i , j cuyos valores son :

i = 3 y j = 5 , el nuevo cromosoma quedaría así después de aplicar el operador SWAP:

D 1 **3** D **2** 5 D 4 7 D

REINSERT: Es un operador unitario parecido al SWAP que elimina el gen en la posición i y lo coloca en la posición j+1. Se generan aleatoriamente dos números i, j, entre el rango definido por el número de genes del Cromosoma. Pero a diferencia del SWAP, el gen de la posición i se ingresa en la posición j + 1. Por ejemplo si se tiene el siguiente cromosoma:

D 1 **2** D **3** 5 D 4 6 7 D

Y se generan dos números i , j cuyos valores son :

i = 3 y j = 5, el nuevo cromosoma quedaría así después de aplicar el operador REINSERT:

D 1 D **3 2** 5 D 4 6 7 D

RELINK: Es un operador unitario que implementa al SWAP varias veces. Es decir, se generan aleatoriamente dos números i, j, entre el rango definido por el número de genes del Cromosoma. Dentro del rango generado, se empiezan a intercambiar los genes haciendo SWAP entre los genes i y j, i+1 y j+1 hasta que se encuentren los dos. Por ejemplo si se tiene el siguiente cromosoma:

D 1 **2** D 3 5 D **4** 6 7 D

Y se generan dos números i , j cuyos valores son :

i = 3 y j = 8, el nuevo cromosoma quedaría así después de aplicar el operador RELINK:

D 1 **4 D 5 3 D 2** 6 7 D

XCROSSOVER: Es un operador binario que realiza el cruce entre dos cromosomas generando un cromosoma hijo. El cruce se realiza de la siguiente manera: Se copia en el hijo el cromosoma Madre. Se generan aleatoriamente dos números i, j, entre el rango definido por el número de genes del cromosoma Padre. Después se inserta la cadena definida por estos dos rangos generados en el cromosoma padre, en el hijo, teniendo en cuenta que no se dupliquen genes diferentes del depósito y que cada ruta cumpla con las restricciones de capacidad. Así se garantiza que el cromosoma resultante va a ser factible. Por ejemplo si se tienen los siguientes cromosomas:

M : D 7 2 D 3 5 D 4 6 1 D

P: D 4 6 3D 5 D 1 2 7 D

1. Se copia el cromosoma de la madre en el hijo

H : D 7 2 D 3 5 D 4 6 1 D

2. Se generan dos números i , j sobre el rango de genes del padre cuyos valores son :

i = 3 y j = 8

P: D 4 **6** 3 D 5 D **1** 2 7 D

El resultado es:

H: D 7 **6 3 D 5 D 1** 2 D 4 D

3.8 Criterios de reemplazo.

Son los criterios que determinan quiénes se van a cruzar. No tienen que ser obligatoriamente los mismos de los padres.

En este caso se implementó un enfoque elitista. El mejor cromosoma de la población pasa automáticamente a la siguiente generación.

Además se genera un porcentaje de la población nueva a partir de los n mejores de la población actual. Los demás cromosomas de la población nueva son generados a partir del 100% de la población actual, que son escogidos aleatoriamente con lo misma probabilidad.

Este algoritmo se corrió con diferentes parámetros de elitismo. Los resultados se encuentran más adelante, en la sección de Realización de Experimentos.

3.9 Generación Nueva Población

El algoritmo para la generación de la nueva población es el siguiente:

1 El mejor cromosoma pasa a la siguiente generación automáticamente sin ninguna clase de operación genética.

2 Primero se crea la población a partir de los parámetros del elitismo.

2.1 Se calcula el número de cromosomas que se van a tener en cuenta para crear la nueva población a partir del porcentaje de creación (cromosomas elitistas).

2.2 Se calcula el número de cromosomas que van a ser creados a partir de los Cromosomas Elitistas a partir del Porcentaje de Población (m).

2.3 Mientras no hayan sido creados los m cromosomas:

2.3.1 se genera un número aleatorio entre 0 y número de cromosomas elitistas -1, para seleccionar un cromosoma.

2.3.2 Se genera un número aleatorio entre 0 y 1 para seleccionar el operador genético a utilizar.

2.3.3 Dependiendo de la probabilidad acumulada de cada operador, se selecciona el operador y se le aplica al cromosoma.

2.3.4 El cromosoma resultante es asignado a la nueva generación

3. Cuando se hayan terminado de generar los m cromosomas.

3.1 Mientras no se hayan generado toda la población nueva.

3.1 Se genera un número aleatorio entre 0 y número de la población para seleccionar un cromosoma.

3.2 Se genera un número aleatorio entre 0 y 1 para seleccionar el operador genético a utilizar.

3.3 Dependiendo de la probabilidad acumulada de cada operador, se selecciona el operador y se le aplica al cromosoma seleccionado.

3.4 El cromosoma resultante es adicionado a la nueva generación.

3.10 Parámetros de funcionamiento.

Determinados parámetros que sin poder ser englobados en ninguno de los anteriores, son fundamentales para el funcionamiento de un algoritmo genético. Todos estos parámetros se pueden cambiar fácilmente en un archivo de configuración que utiliza el programa generado para resolver el CVRP. Los parámetros más importantes son:

Tamaño de la población: Para el tamaño de las poblaciones se utilizó 2n, donde n es el número de clientes + 1.

Probabilidad de la aplicación de los operadores genéticos: Probabilidad de seleccionar uno de los operadores para mutar un cromosoma.

3.11 Algoritmo Genético.

El algoritmo implantado corre de la siguiente manera:

1. Se inicializa la red: Se carga del archivo de configuración los nodos, con sus demandas y posiciones x,y., además, se inicializan los parámetros de entrada como la capacidad del camión, el número de iteraciones, la probabilidad de los operadores genéticos y del elitismo.
2. Se crea la población aleatoria, siguiendo el algoritmo descrito en 4.4.
3. Mientras halla más iteraciones por hacer.
   - 3.1 Se halla la función de costo para cada integrante de la población.
   - 3.2 Se ordenan la población ascendentemente según el costo.
   - 3.3 Se genera la población de la generación i +1 a partir del algoritmo descrito en 4.10 para la generación de la nueva población.
4. Al final de las iteraciones, el primer cromosoma es la solución al problema.

3.12 Decisiones Computacionales

Para implementar este algoritmo genético se decidió utilizar el lenguaje de programación orientado a objetos JAVA.. El cual tiene la ventaja ser independiente del sistema operativo, es decir, corre en cualquier máquina con JAVA instalado, es libre, por lo que no hay que preocuparse de licencias y está muy bien documentada la ayuda.

La desventaja de este lenguaje es que tiene un costo computacional más alto debido a su independencia del sistema operativo, porque corre sobre una máquina virtual.

La complejidad de esta implementación tiene cota superior de r*(n2+n) donde r es el número de iteraciones y n el número de nodos en el grafo.

Se creó una clase nodo la cual tiene como atributos la demanda del nodo, la posición x , la posición Y y el número del nodo. Para el manejo de los cromosomas se utilizó la clase Vector, que maneja los arreglos en JAVA.. Por lo tanto un cromosoma es un vector con nodos en cada posición. Además, esta clase de cromosoma tiene como atributos el costo y la posición en el ranking. Ésta clase tiene además de las funciones naturales “get” y “set”, las funciones referentes a los operadores genéticos. Es en esta clase donde se implementaron los diferentes operadores genéticos, las funciones de evaluación de costo y “arreglar” los cromosomas resultantes.

Se creó una clase Grafo que maneja todo lo referente a la red de nodos y arcos. Los nodos se implementaron en un vector de Nodos y los Arcos se implementaron en una matriz de enteros que contienen el costo de ir de i a j.

La clase genéticos se encarga de crear las poblaciones aleatorias, la generación de la nueva población y el ranking de la población.

4. REALIZACIÓN EXPERIMENTOS

Debido al gran número de parámetros del Algoritmo , es necesario realizar pruebas con éstos para definir los mejores rangos de valores para obtener la mejor solución.

Los parámetros a analizar son:

4.1 Tamaño de Población

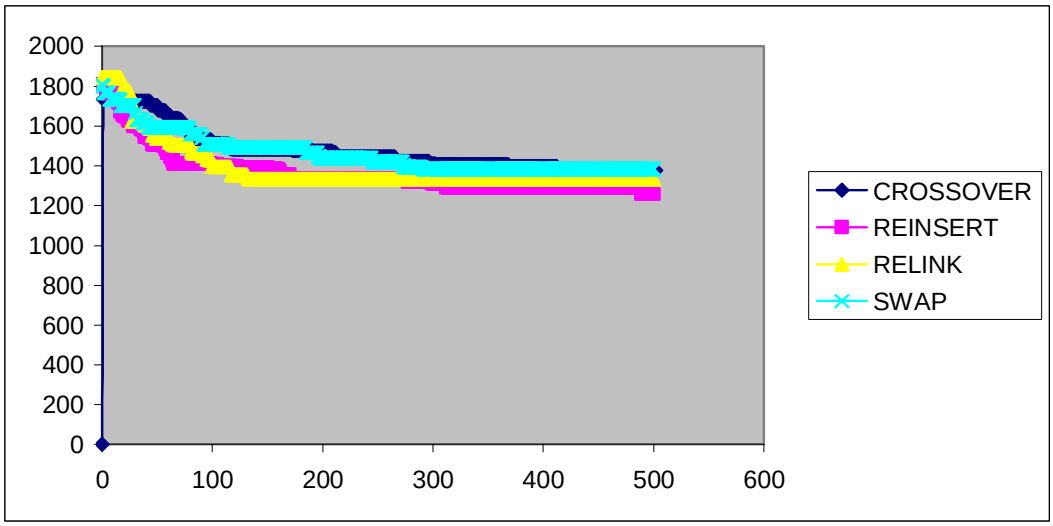
Para este parámetro, se basó en la investigación de Tim Duncan en su paper Experiments in the use of Neighborhood techniques for VRP, en donde se definió el tamaño de la población como 2n

4.2 Número de Iteraciones

El número de iteraciones que se utilizó en todos los casos fue de 1000. Después de varios experimentos se llegó a la conclusión de que el número de iteraciones influye de una manera más importante que el número de cromosomas de la Población. Por lo tanto el número de iteraciones siempre es grande.

4.3 Cálculo de la efectividad de los operadores genéticos desarrollados

Primero se necesita probar que los operadores son los suficientemente buenos como para ir mejorando la función Objetivo, a través de las iteraciones, por lo que se corrieron 30 iteraciones con cada Operador, arrojando el siguiente resultado:



Como se puede observar en la gráfica anterior, todos los operadores van mejorando la

función objetivo a través del tiempo, lo cual es una característica que se espera de cada operador.

Para definir cuál de los operadores genéticos implementados en éste algoritmo resuelve mejor el CVRP, se desarrolló un experimento en el cual se ejecutan varias veces el algoritmo para tres problemas de diferentes tamaños (21, 51, 100),y con la solución de cada ejecución se realizan pruebas estadísticas para comprobar si son o no iguales las medias de los datos. El experimento hizo 40 corridas para cada problema y para cada operador. Cada corrida hace 1000 iteraciones, tiene un porcentaje de creación del 2% ,un Porcentaje de Población del 6% y el tamaño de la población es 2n. Los resultados fueron los siguientes:

Con el siguiente problema de 21 nodos:

Nodo	X	Y	Demanda	Nodo	X	Y	Demanda
Depósito	151	264	0	**11**	156	217	1300
1	159	261	700	**12**	129	214	1300
2	130	254	800	**13**	146	208	300
3	128	252	1400	**14**	164	208	900
4	163	247	2100	**15**	141	206	2100
5	146	246	400	**16**	147	193	1000
6	161	242	800	**17**	164	193	900
7	142	239	100	**18**	129	189	2500
8	163	236	500	**19**	155	185	1800
9	148	232	600	**20**	139	182	700
10	128	231	1200				

Los resultados de las corridas fueron:

Grupos	Cuenta	Suma	Promedio	Varianza
XCrossOver	40	34446,99342	861,1748355	1060,839143
Reinsert	40	30796,92399	769,9230998	1123,498317
Relink	40	30618,02672	765,450668	2101,314699
Swap	40	31499,47358	787,4868395	1279,739503

Primero se prueban si los resultados generados utilizando cada Operador, son iguales. En tal caso el Algoritmo Genético no dependería de la utilización de uno u otro operador. La medida que se está utilizando es el costo de cada solución.

Para probar esto, se hace una prueba F de medias de los costos con las siguientes Hipótesis:

Ho: $u_{Relink} = u_{Swap} = u_{Reinsert} = u_{XCrossOver}$

Ha: Al menos 1 no es igual

El resultado es el siguiente:

Suma Errores	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F	P Value	Valor crítico para F
SCTR	237339,9673	3	79113,32243	56,86091994	6,83E-25	2,662567056
SCE	217050,2748	156	1391,347916			
STC	454390,2421	159				

Después de realizar la prueba F el resultado es 56,86091994, que es mayor a la $F_{0.95.3,156}$ (2,662567056), por consiguiente, se rechaza Ho. Es decir, existe una razón para creer que los operadores genéticos tienen una incidencia en la función objetivo del algoritmo implantado.

Como se quiere saber cuál es el mejor Operador de los cuatro implementados, se hace una prueba F , entre los 3 más pequeños que son: SWAP, REINSERT y RELINK.

Los resultados son:

Suma Cuadrados	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F	P-Value	Valor crítico para F
SCTR	10854,40606	2	5427,203028	3,614478688	0,029980354	3,073765242
SCE	175677,5483	117	1501,517507			
STC	186531,9543	119				

Debido a que el resultado del experimento (3,073765242) es mayor que la $F_{.95,2,117}$ (0.614478688) se rechaza Ho. Es decir, existe una razón para creer que los tres operadores genéticos infieren diferentemente en la función objetivo del algoritmo implantado.

Cómo se quiere saber cuál es el mejor Operador de los cuatro, se hace una prueba T de diferencia de medias, entre los dos operadores más pequeños son REINSERT y

RELINK, para probar cual de los dos es menor. La siguiente hipótesis trata de comprobar si RELINK es mayor que REINSERT.

Ho: $u_{Relink} - u_{Reinsert} = 0$

Ha: $u_{Relink} - u_{Reinsert} > 0$

Debido a que el número de datos es mayor a 30, se puede afirmar por el Teorema de Límite Central que la suma de los datos se distribuye Normal. Como las varianzas no se conocen, pero se asumen iguales , se puede hacer la siguiente prueba T de variables Normales

$$T = \frac{X - Y - 0}{Sp(raiz\ (1/nx + 1/ny)}$$

El resultado es:

Sp	**1612,406508**
Xbarra	**765,450668**
Ybarra	**769,9230998**
T tabla	**1.664**
T	**763,3152259**

Como el resultado de la operación es mayor que el teórico(**763,3152259>1.664)**, se puede afirmar que el operador REINSERT es mejor que el RELINK, es decir, su función objetivo es menor.

Con el problema de 51 nodos:

Nodo	X	Y	Demanda	Nodo	X	Y	Demanda
0	30	40	0	25	7	38	28
1	37	52	7	26	27	68	7
2	49	49	30	27	30	48	15
3	52	64	16	28	43	67	14
4	20	26	9	29	58	48	6
5	40	30	21	30	58	27	19
6	21	47	15	31	37	69	11
7	17	63	19	32	38	46	12
8	31	62	23	33	46	10	23
9	52	33	11	34	61	33	26

10	51	21	5	35	62	63	17
11	42	41	19	36	63	69	6
12	31	32	29	37	32	22	9
13	5	25	23	38	45	35	15
14	12	42	21	39	59	15	14
15	36	16	10	40	5	6	7
16	52	41	15	41	10	17	27
17	27	23	3	42	21	10	13
18	17	33	41	43	5	64	11
19	13	13	9	44	30	15	16
20	57	58	28	45	39	10	10
21	62	42	8	46	32	39	5
22	42	57	8	47	25	32	25
23	16	57	16	48	25	55	17
24	8	52	10	49	48	28	18
				50	56	37	10

Los resultados de las corridas fueron:

Grupos	Cuenta	Suma	Promedio	Varianza
XCrossOver	40	46356,6575	1158,916438	4311,553158
Reinsert	40	34473,26937	861,8317343	3856,089287
Relink	40	33228,17623	830,7044058	4391,834318
Swap	40	44445,0923	1111,127308	7264,86323

Primero se prueban si los resultados generados utilizando cada Operador, son iguales. Para probar esto, se hace la siguiente prueba F de medias :

Ho: $u_{\text{Relink}} = u_{\text{Swap}} = u_{\text{Reinsert}} = u_{\text{XCrossOver}}$

Ha: al menos una es diferente

El resultado es el siguiente:

Origen de las variaciones	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F	Probabilidad	Valor crítico para F
SCTR	3400204,569	3	1133401,523	228,6888791	6,97284E-57	2,662567056
SCE	773149,2598	156	4956,084999			
STC	4173353,829	159				

Debido a que el resultado del experimento es mayor que la $F_{.95,3,156}$ (228,6888791>2,662567056) , se concluye que se puede rechazar Ho, por consiguiente,

existe una razón para creer que los operadores genéticos tienen una incidencia en la función objetivo del algoritmo implantado.

Como se quiere saber cuál es el mejor Operador de los cuatro, se hace la prueba F, entre los 3 menores que son: XCROSSOVER, REINSERT y RELINK.

Los resultados son:

Origen de las variaciones	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F	Probabilidad	Valor crítico para F
SCTR	2626018,257	2	1313009,128	313,6298955	9,8398E-48	3,073765242
SCE	489819,5938	117	4186,492255			
SCT	3115837,851	119				

Debido a que el resultado del experimento es mayor que la $F_{.95,2,117}$ (313,6298955>3,073765242) se concluye que se rechaza Ho, por consiguiente, existe una razón para creer que los tres operadores genéticos infieren diferentemente en la función objetivo del algoritmo implantado.

Cómo se quiere saber cuál es el mejor Operador de los cuatro, se hace una prueba T, entre los dos menores que son REINSERT y RELINK, con las siguientes Hipótesis:

Ho: $u_{\text{Relink}} - u_{\text{Reinsert}} = 0$

Ha: $u_{\text{Relink}} - u_{\text{Reinsert}} < 0$

El resultado es:

X BARRA	**830,70**
Y BARRA	**861,83**
DESV X	**66,27091608**
DESV Y	**62,09741772**
Sp	**64,1841669**
T tabla	**-1.664**
T	**0,014539763**

Como el resultado del experimento T es mayor que la $T_{0.95,78}$ **(0,014539763>-1.664) y menor que la** $T_{0.05,78}$ **(0,014539763<1.664)** se puede afirmar que el operador RELINK es estadísticamente igual al REINSERT, por lo que para el problema de 51 nodos cualquiera de los dos operadores incide de la misma manera en la función de Costo.

Para 100 nodos:

Nodo	x	y	D/da	Nodo	x	y	D/da	Nodo	x	y	D/da	Nodo	x	y	D/da
0	41	49	0	**26**	35	40	16	**52**	37	31	14	**78**	57	48	23
1	35	17	7	**27**	41	37	16	**53**	57	29	18	**79**	56	37	6
2	55	45	13	**28**	64	42	9	**54**	63	23	2	**80**	55	54	26
3	55	20	19	**29**	40	60	21	**55**	53	12	6	**81**	15	47	16
4	15	30	26	**30**	31	52	27	**56**	32	12	7	**82**	14	37	11
5	25	30	3	**31**	35	69	23	**57**	36	26	18	**83**	11	31	7
6	20	50	5	**32**	53	52	11	**58**	21	24	28	**84**	16	22	41
7	10	43	9	**33**	65	55	14	**59**	17	34	3	**85**	4	18	35
8	55	60	16	**34**	63	65	8	**60**	12	24	13	**86**	28	18	26
9	30	60	16	**35**	2	60	5	**61**	24	58	19	**87**	26	52	9
10	20	65	12	**36**	20	20	8	**62**	27	69	10	**88**	26	35	15
11	50	35	19	**37**	5	5	16	**63**	15	77	9	**89**	31	67	3
12	30	25	23	**38**	60	12	31	**64**	62	77	20	**90**	15	19	1
13	15	10	20	**39**	40	25	9	**65**	49	73	25	**91**	22	22	2
14	30	5	8	**40**	42	7	5	**66**	67	5	25	**92**	18	24	22
15	10	20	19	**41**	24	12	5	**67**	56	39	36	**93**	26	27	27
16	5	30	2	**42**	23	3	7	**68**	37	47	6	**94**	25	24	20
17	20	40	12	**43**	11	14	18	**69**	37	56	5	**95**	22	27	11
18	15	60	17	**44**	6	38	16	**70**	57	68	15	**96**	25	21	12
19	45	65	9	**45**	2	48	1	**71**	47	16	25	**97**	19	21	10
20	45	20	11	**46**	8	56	27	**72**	44	17	9	**98**	20	26	9
21	45	10	18	**47**	13	52	36	**73**	46	13	8	**99**	18	18	17
22	55	5	29	**48**	6	68	30	**74**	49	11	18				
23	65	35	3	**49**	47	47	13	**75**	49	42	13				
24	65	20	6	**50**	49	58	10	**76**	53	43	14				
25	45	30	17	**51**	27	43	9	**77**	61	52	3				

Los resultados de las corridas del experimento son:

Grupos	Cuenta	Suma	Promedio	Varianza
XCrossOver	40	93344,7827	2333,619568	9755,986979
Reinsert	40	77792,064	1944,8016	4143,044044
Relink	40	73429,0864	1835,72716	3366,350513
Swap	40	91795,4627	2294,886568	11273,42775

Para probar si los operadores son iguales, se hace una prueba F de medias con la siguientes hipótesis:

Ho: $u_{\text{Relink}} = u_{\text{Swap}} = u_{\text{Reinsert}} = u_{\text{XCrossOver}}$

Ha: *al menos una es diferente*

El resultado es el siguiente:

Origen de las variaciones	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F	Probabilidad	Valor crítico para F
SCTR	7458605,86	3	2486201,953	348,4661085	6,59052E-69	2,662567056
SCE	1113013,562	156	7134,702321			
STE	8571619,422	159				

Debido a que el resultado del experimento es mayor que la $F_{.95,3,156}$ (348,4661085>2,662567056) , se rechaza H_o, por lo tanto existe una razón para creer que los operadores genéticos tienen una incidencia en la función objetivo del algoritmo implantado.

Debido a que los promedios son muy diferentes entre sí, y las varianzas muy grandes, se realiza una prueba T para probar cual de los dos operadores es menor. Si el resultado es mayor que 1.664 el REINSERT es mejor que RELINK, si el resultado es menor que -1.664 pasa lo contrario y si el resultado está entre -1.664 y 1.664, los dos operadores afectan igual al resultado.

Las hipótesis son:

Ho: $u_{Relink} - u_{Reinsert} = 0$

Ha: $u_{Relink} - u_{Reinsert} < 0$

El resultado es:

X BARRA	1835,73
Y BARRA	1944,80
DESV X	58,0202595
DESV Y	64,3664823
Sp	61,1933709
T Tabla	-1.664
T	-0,02318974

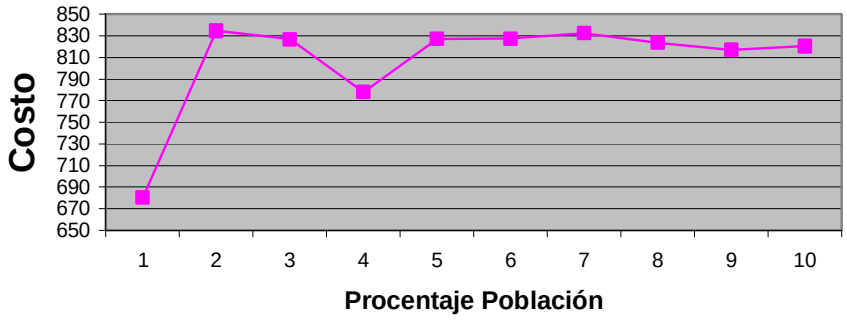
Como el resultado de la prueba está en el rango (-1.664, 1.664) no existe evidencia estadística para decir cual de los dos Operadores incide mejor en el resultado. Por lo tanto para 100 nodos el uso del operador RELINK o REINSERT es indiferente para la obtención de la función de costo.

Después de haber analizado los operadores en los diferentes problemas, se puede concluir que a medida que aumenta el número de nodos en el problema, el operador RELINK se asemeja más al operador REINSERT. Tanto así que en un problema de 51 nodos, el experimento estadístico arroja como resultado la indiferencia de los promedios de los costos generados por el algoritmo genético después de haber sido utilizado por cada operador.

4.4 Parámetros del Elitismo

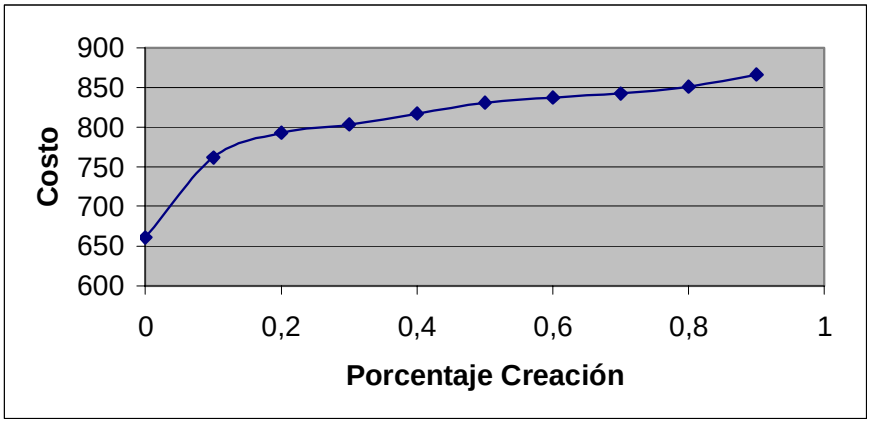
Para analizar el comportamiento de los parámetros del Elitismo (Porcentaje Creación y Porcentaje Población) con el costo, se ha decidido diseñar un experimento en el cual se hagan diferentes corridas con las diferentes combinaciones posibles con los problemas de 21 y 51 nodos. El experimento consiste en ejecutar 20 veces cada problema con todas las combinaciones posibles entre (0 ,0) hasta (0.9,0.9) aumentando cada parámetro en 0.1.Los resultados se muestran individualmente para cada parámetro:

Porcentaje de Población:



Como se puede apreciar en la gráfica, los mejores costos se obtuvieron con el parámetro en 0.1 y 0.3

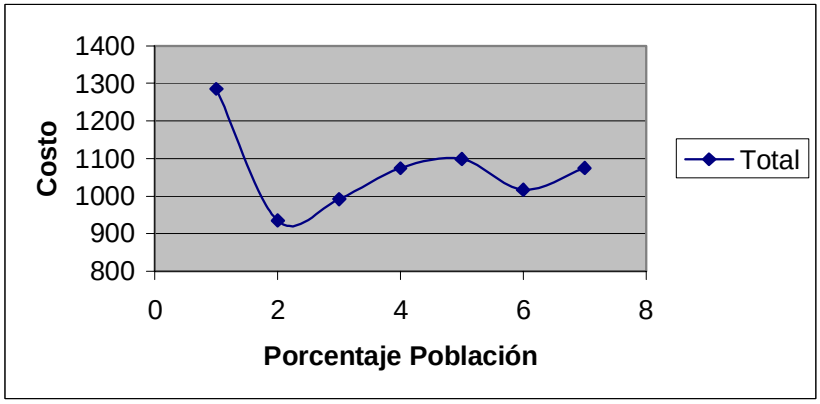
Porcentaje Creación



En esta gráfica se puede apreciar claramente, cómo a medida que aumenta el porcentaje de Creación, el valor de la función objetivo va disminuyendo. Por lo tanto se concluye que es mejor no tener elitismo en este tipo de problemas. Debido a que el problema en que se desarrollaron los experimentos es muy pequeño (21 nodos), el elitismo disminuye la posibilidad de encontrar óptimos generales porque cae en óptimos locales, disminuyendo el espacio de búsqueda del algoritmo. Así, es importante tener en cuenta el tamaño del problema, cuando se estudie la posibilidad de usar elitismo.

Por esta razón, se realizó el mismo experimento, sobre el problema de 51 nodos. Los resultados de los experimentos se exponen a continuación:

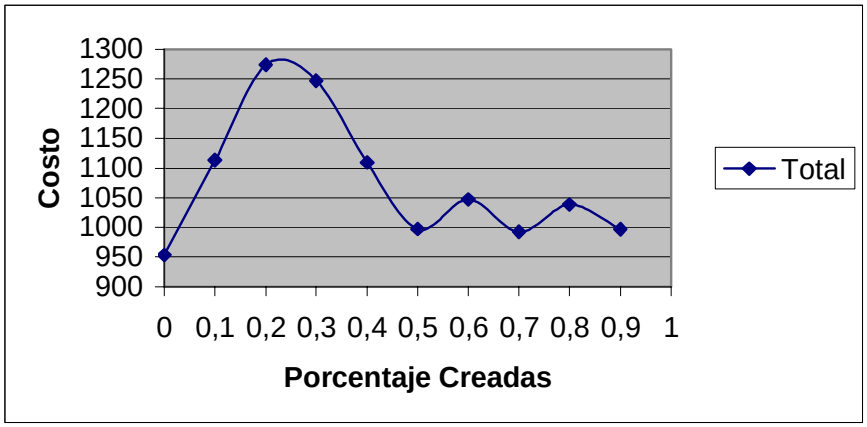
Porcentaje Población



Ahora, la función objetivo es mínima cuando el porcentaje de población tiene el valor de 2%, a diferencia del caso anterior, que el mejor costo se hallaba en 0%.

Se puede apreciar claramente que el elitismo mejora considerablemente la función objetivo. A diferencia del problema de 21 nodos, éste disminuye el costo con el uso del elitismo.

Porcentaje Creadas



Aunque el mínimo sigue estando en 0%, el segundo mínimo está muy cercano y está en 7%. Se empieza a ver un comportamiento diferente de los costos a medida que aumenta el porcentaje de creadas. Sigue estando el mejor resultado en 0% pero después del 5% los costos se emparejan.

Este resultado se puede explicar porque el tamaño del problema es más grande que el anterior, ayudando a evitar caer tan fácilmente en óptimos locales. Así, entre más grande sea el problema, mayor impacto tendrá un porcentaje de creadas alto.

4.5 Comparación de los resultados obtenidos con los problemas Benchmark

Los problemas que se tuvieron en cuenta son VRPNC1, VRPNC2 Y VRPNC3, que se encuentran en http://www.ms.ic.ac.uk/jeb/pub/, los resultados son:

	Relink	Reinsert	CrossOver	SWAP	Combinación Operadores	Mejor Solución Conocida
vrpnc1	715	720	800	820	610	554
vrpnc2	1070	1050	1200	1250	960	909
vrpnc3	1025	1009	1156	1142	950	901

Como se puede apreciar, la combinación de los operadores brinda mejores resultados que los operadores por sí solos. Sin embargo, ninguna de las corridas pudo superar la mejor solución conocida hasta ahora. El mejor resultado fue obtenido a través del algoritmo de Tabu Search. Cabe preguntarse si una mejor redistribución de los pesos de los operadores pueda conseguir una mejor solución. Este análisis queda como propuesta para un trabajo futuro.

5. CONCLUSIONES

Es importante sobresaltar el poder tan grande de aplicación de los algoritmos genéticos. Tienen la ventaja que son fáciles de implementar y de aplicar a problemas complejos. Debido a que el problema se representa en los cromosomas, para cambiar de un problema a otro, es necesario normalmente, cambiar sólo la representación de los cromosomas y a veces los operadores genéticos. Pero, el algoritmo sigue siendo el mismo. Esta es la libertad que brindan los algoritmos genéticos para aplicarse a cualquier tipo de problemas.

Las desventajas de este algoritmo consisten en :

1. Es computacionalmente muy costoso, es decir, necesita de mucho tiempo de máquina para resolver un problema.
2. La cantidad de parámetros que hay que afinar para mejorar el algoritmo es muy grande. Debido a que son tantos parámetros (8 en este caso), es difícil poder medir las interrelaciones entre éstos para definir la mejor combinación posible. Sin embargo, esta característica es la que le da mayor poder al algoritmo para adaptarse a cualquier problema.

En conclusión se cumplieron los objetivos propuestos, para esta tesis. Aunque no se pudo mejorar el costo óptimo de los problemas estudiados, se estuvo cerca y por lo tanto se obtuvieron resultados muy positivos.

6. PROPUESTAS FUTURAS

Este trabajo a cubierto las bases del diseño a través de Algoritmos genéticos para resolver problemas NP-Hard. Los trabajos propuestos a partir de esta tesis son:

1. Estudio más avanzado de las relaciones entre las variables que hacen parte de un algoritmo genético en el CVRP.
2. Implementación de Algoritmos genéticos para las variantes del VRP.
3. Solución del CVRP a través de la Búsqueda TABÚ y Colonia de Hormigas para comparar las diferencias en los resultados y características de implementación de los algoritmos.
4. Implementación de los Algoritmos Genéticos para resolver otros problemas NP-Hard, en otras áreas del saber.

7. BIBLIOGRAFÍA

1) *Experiments in the use of Neighbourhood Search techniques for Vehicle Routing, Tim Duncan*

   Artificial Intelligence Applications Institute,University of Edinburgh, Edinburgh EH1 1HN AIAI-TR-176, June 1995

2) *J. E. Beasley. OR-Library: Distributing Test Problems by Electronic Mail. Journal of the Operational Research Society, 41(11):1069{1072, 1990. (http://mscmga.ms.ic.ac.uk/info.html).*

3) *The vehicle routing problem / edited by Paolo Toth, Daniele VigoPhiladelphia : Society for Industrial and Applied Mathematics, c2002.*

4) *Genetic programming : on the programming of computers by means of natural selection Koza, John*

5) *Genetic algorithms + data structures = evolution programs Michalewicz , Zbigniew 1996*

6) *Design and analysis of experiments Montgomery, Douglas C. 1996*

7) http://osiris.tuwien.ac.at/~wgarn/VehicleRouting/vehicle_routing.html

8) http://osiris.tuwien.ac.at/~wgarn/VehicleRouting/GECCO02_VRPCoEvo.pdf

9) http://geneura.ugr.es/~jmerelo/ie/ags.htm

10) T. Ralphs, J. Hartman and M. Galati. "Capacitated Vehicle Routing and Some Related Problems". Some CVRP Slides. Rutgers University. 2001

11) T. Ralphs, J. Hartman and M. Galati. "Capacitated Vehicle Routing and Some Related Problems". Some CVRP Slides. Rutgers University. 2001