PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

DESARROLLO DE UN ALGORITMO GENÉTICO-CULTURAL PARA EL PROBLEMA DE ENRUTAMIENTO DE VEHÍCULOS CON VENTANAS DE TIEMPO

DANIEL RODOLFO QUAGLIAROLI ZAPATA

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO CIVIL EN INFORMÁTICA

DICIEMBRE, 2007

Pontificia Universidad Católica de Valparaíso
Facultad de Ingeniería
Escuela de Ingeniería Informática

DESARROLLO DE UN ALGORITMO GENÉTICO-CULTURAL PARA EL PROBLEMA DE ENRUTAMIENTO DE VEHÍCULOS CON VENTANAS DE TIEMPO

DANIEL RODOLFO QUAGLIAROLI ZAPATA

Profesor Guía: **Claudio Cubillos Figueroa**

Profesor Co-referente: **Broderick Crawford Labrín**

Carrera: **Ingeniería Civil en Informática**

DICIEMBRE, 2007

Dedicatoria

Dedicado a mi madre, quien es la responsable de mis logros

Agradecimientos

A Dios, por el camino recorrido.

A mi madre, por su esfuerzo y amor inagotable.

A Natalia por su apoyo incondicional.

A mi profesor guía por su paciencia y dedicación.

A mis compañeros, amigos y profesores con quienes compartí los mejores años de mi vida.

Resumen

El problema de enrutamiento de vehículos con ventanas de tiempo (VRPTW), consiste en determinar un conjunto de rutas para una flota de vehículos que parten desde un depósito para satisfacer, dentro de un intervalo de tiempo, la demanda de clientes dispersos geográficamente. Actualmente las investigaciones que tratan el VRPTW y los problemas combinatorios en general, tienden a dirigirse hacia el campo de las metaheurísticas, que entregan resultados suficientemente buenos con recursos razonables. No obstante, la gran cantidad de trabajos relacionados en esta área, aún se trabaja en el desarrollo de avances que permitan mejorar el funcionamiento de estas técnicas. El objetivo principal del proyecto fue desarrollar un modelo híbrido utilizando los conceptos de algoritmos genéticos y culturales para el problema descrito.

Abstract

The Vehicle Routing Problem with time windows (VRPTW) consists on finding a set of routes for a fleet of vehicles that start from a central depot to satisfy customer demands that are dispersed geographically in a time interval. Nowadays the researches of VRPTW and combinatorial problems in general, are addressed to Meta-heuristics field, presenting important results with reasonable resources. Despite several works in this area, researchers still work to improve the functioning of these techniques. The main object of the project was to develop a hybrid model using concepts from genetic and cultural algorithms for the proposed problem.

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1

1 Presentación del tema

1.1 Introducción

El problema del enrutamiento de vehículos (VRP) es uno de los problemas más conocidos en optimización combinatoria. El problema consiste en minimizar los costos de una flota de vehículos que deben satisfacer la demanda de varios clientes distribuidos geográficamente, partiendo y regresando a un depósito central.

Dentro de la gama de variantes que tiene el VRP se encuentra el problema de enrutamiento de vehículos con ventanas de tiempo (VRPTW), en este caso al problema mencionado anteriormente se le agrega la restricción de visitar a los clientes dentro de un intervalo de tiempo establecido.

La forma del problema no permite llegar a una solución óptima con el uso de una cantidad razonable de recursos, dado que a medida que aumenta la cantidad de clientes la cantidad de posibles rutas crece en forma exponencial. Debido a esto se utilizan ciertas técnicas que permiten hallar una solución confiable, cercana a la óptima y en tiempos prudentes.

En la actualidad las investigaciones que tratan el VRP y los problemas combinatorios en general, tienden a dirigirse hacia el campo de las metaheurísticas [1], ya que estas permiten llegar a buenos resultados en tiempos bastante razonables. Por esto los trabajos tienden hacia el desarrollo de avances que permitan mejorar el funcionamiento de estas técnicas y esto es algo que tiene bastante campo de trabajo, ya que en estas metaheurísticas se deben definir una gran cantidad de parámetros para lo cual no hay un método definido en el que se garantice el óptimo funcionamiento de las técnicas.

Una de las herramientas usadas ampliamente en problemas de optimización como el VRP, son los algoritmos genéticos, este tipo de algoritmos ha recibido una gran aceptación por su eficiencia para resolver problemas de distinta complejidad. Sin embargo, esta eficiencia depende mucho del tipo de problema planteado, junto con la estrategia que adopte el algoritmo para abordarlo. Por lo tanto es importante considerar la búsqueda de nuevas representaciones al igual que la definición de operadores de cruce adecuados que cumplan con las restricciones del VRP.

Entre las heurísticas recientes que utilizan los conceptos de la computación evolutiva están los algoritmos culturales propuestos por Reynolds [9]. En este tipo de algoritmos se pretende que la información más importante del problema que se haya extraído durante el

proceso evolutivo sea accesible a toda la población. Los algoritmos culturales, tal vez por lo reciente de su propuesta, han sido poco estudiados y desarrollados.

Considerando los puntos anteriores, en este trabajo se pretende profundizar el estudio de los algoritmos genéticos y culturales con el fin de lograr una propuesta de solución para el problema de enrutamiento de vehículos con ventanas de tiempo, considerando la existencia de un solo depósito y capacidad constante para los vehículos.

1.2 Objetivo general

Desarrollar un modelo de algoritmo Genético-Cultural para el problema de enrutamiento de vehículos con ventanas de tiempo.

1.3 Objetivos específicos

- Estudiar y comprender el problema del enrutamiento de vehículos con ventanas de tiempo y propuestas de solución que involucren algoritmos genéticos y culturales.

- Definir los elementos de algoritmos genéticos y culturales que se utilizarán para construir un modelo híbrido, el lenguaje de programación para la implementación de un programa de experimentación y los datos de entradas para su validación.

- Validar el modelo a desarrollar utilizando los datos de entrada definidos en el programa de experimentación y comparar los resultados obtenidos.

1.4 Metodología de trabajo

Para el desarrollo del proyecto, se utilizará una metodología de investigación científica clásica. La cual puede ser descrita bajo las siguientes etapas:

- Se inicia la investigación con la concepción de la idea central, con la cual se puede plantear el problema a través de los objetivos, cuestionamientos asociados y justificaciones referentes su viabilidad.

- Se desarrolla un marco teórico, lo que contempla la revisión de la literatura desde un punto de vista de la obtención, selección y la extracción de información relevante. Esto se asocia directamente con todas las referencias utilizadas para la investigación y los contenidos teóricos que se verán a través de los informes desarrollados en el proyecto.

- Se desarrollan hipótesis o planteamientos previos respecto a la problemática, se especifican variables vinculadas al problema de manera conceptual y operacional, además se define si el problema se abordará experimentalmente. Las hipótesis son

resultado directo del desarrollo del marco teórico y se puede ver reflejadas en las distintas conclusiones que se puedan obtener de las primeras etapas del proyecto.

- Se definen las herramientas de experimentación, en donde se contempla tanto el instrumento utilizado como las muestras para realizar los experimentos. Las muestras para esta investigación corresponderán a datos de entrada para el problema VRPTW, los cuales pueden ser obtenidas de datos de prueba estandarizados.

- Se elaboran y presentan los reportes de investigación adecuados.

1.5 Plan de trabajo

Para cumplir los objetivos establecidos, se propone dividir la realización del proyecto en tres etapas:

- **Recolección de antecedentes y análisis investigativo**

En esta etapa se realizará una revisión detallada de las fuentes y referencias encontradas, para rescatar los elementos más importantes y útiles que tengan relación con las características del VRPTW y los aspectos teóricos de los algoritmos genéticos y culturales. Tiempo de duración propuesto: 2 meses (Marzo – Abril 2007).

- **Desarrollo y construcción de modelos**

Con la información analizada en la etapa anterior, se procederá a definir y desarrollar el modelo de algoritmo de solución para el VRPTW. Este desarrollo contempla el estudio de los conceptos teóricos de algoritmos genéticos y culturales encontrados en la literatura. En la construcción se contemplan distintos submodelos para el algoritmo, es decir, población inicial, representación, recombinación, selección, mutación, etc. Para validarlos se desarrollará un prototipo de software, en el que se usará datos de prueba preliminares. Tiempo de duración propuesto: 2 meses (Mayo – Junio 2007).

- **Implementación formal de los modelos y etapa de pruebas finales**

En base al modelo y prototipo obtenidos en la etapa anterior, se desarrollará un software para generar soluciones en base a datos de entrada a definir, no necesariamente los mismos usados anteriormente. Se realizarán distintas ejecuciones para medir la eficiencia en la implementación del modelo, realizando comparaciones con otras soluciones encontradas en la literatura. Tiempo de duración propuesto: 4 meses (Agosto - Noviembre 2007).

En la Tabla 1.1 y Figura 1.1 se presenta un esquema gráfico de la planificación del proyecto.

Tabla 1.1 Planificación detallada del proyecto.

	Tareas	Duración
1	Desarrollo del marco teórico del VRPWT	15 días
2	Desarrollo del marco teórico de algoritmos genéticos	20 días
3	Desarrollo del marco teórico de algoritmos culturales	15 días
4	Investigación de trabajos relacionados	15 días
5	Desarrollo del marco teórico del modelo	20 días
6	Construcción del modelo	15 días
7	Implementación prototipo	12 días
8	Pruebas del prototipo	10 días
9	Definición de datos de pruebas finales	20 días
10	Refinamiento del modelo	25 días
11	Construcción del programa de experimentación	15 días
12	Desarrollo de pruebas	30 días
13	Análisis de resultados y presentación de resultados	20 días

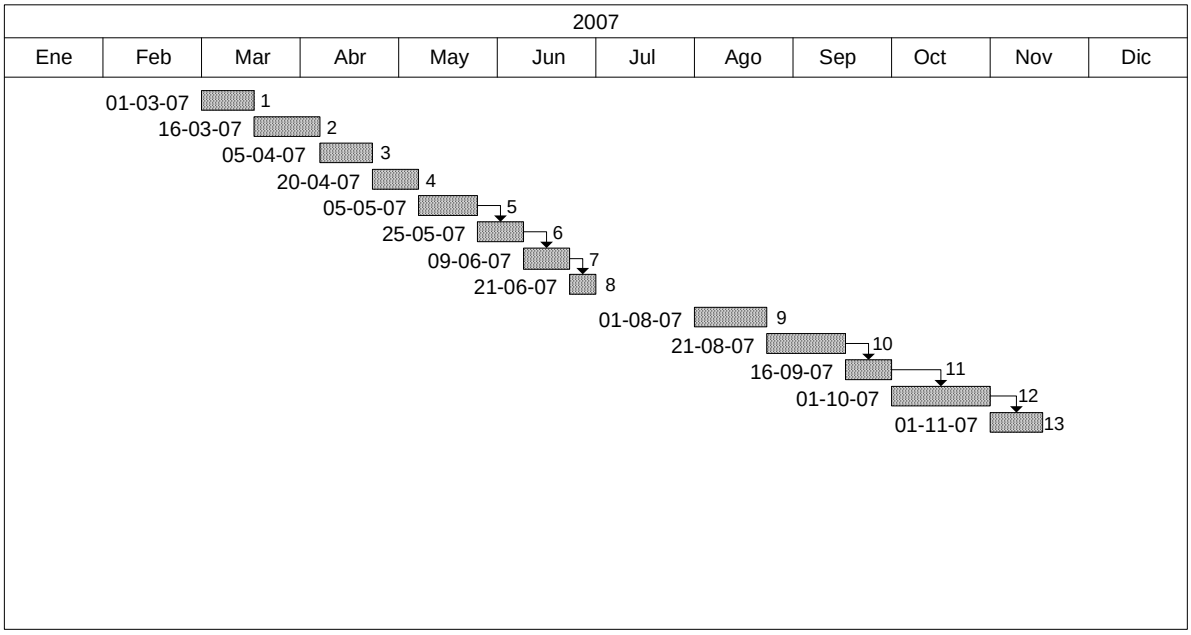


Figura 1.1 Planificación detallada del proyecto.

Capítulo 2

2 Problema de enrutamiento de vehículos

2.1 VRP clásico

El problema del enrutamiento de vehículos es uno de los problemas más conocidos de optimización combinatoria, consiste en determinar un conjunto de rutas para una flota de vehículos que parten desde uno o más depósitos o almacenes para satisfacer la demanda de clientes dispersados geográficamente. El término cliente puede ser utilizado para indicar las paradas o entregas realizadas por los vehículos, en donde cada cliente debe ser asignado exactamente a un vehículo en un orden establecido. El objetivo es satisfacer la demanda de los clientes minimizando el costo total que se incurre en las rutas. En la figura 2.1 se muestra una representación de una solución para VRP, en donde el nodo cero es el depósito y los restantes los clientes.

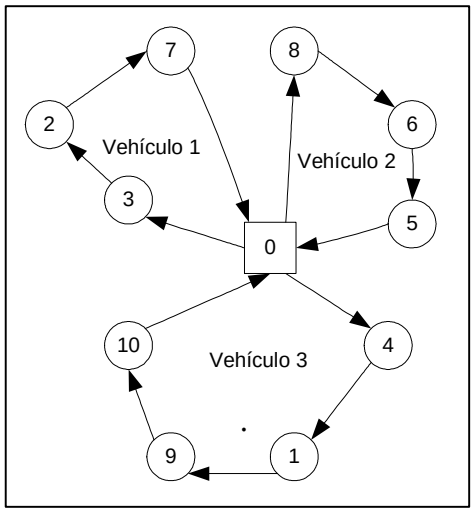


Figura 2.1 Una solución para el problema de enrutamiento de vehículos.

Desde que el problema de enrutamiento de vehículos fue introducido por Dantzig y Ramser [5] en 1959, ha surgido en torno a él toda una línea de investigación y desarrollo. Este interés se deriva por un lado en el sentido práctico de su aplicación, como por ejemplo en transporte de personas y productos, servicios de reparto, recolección de basura, etc., y por otro lado en la complejidad que entrega éste problema, considerado como un problema de complejidad NP-Completo [3], dado que el número de posibles soluciones crece exponencialmente con respecto al número de clientes.

2.2 Modelo matemático

El modelo matemático del problema se presentará según [3] mediante teoría de conjuntos y funciones.

Se establece que existen *n* clientes y un depósito representados por un conjunto *P*:

$$P = \{p_0, p_1, p_2, ..., p_n\}$$

En donde p_0 representa el depósito y los demás puntos representan a cada uno de los *n* clientes. Se tiene en cuenta la relación *R*:

$$R \subset P \times P$$

En donde $(p_i, p_j) \in R$ implica que hay una conexión entre el punto p_i y el punto p_j.

Existen *m* vehículos representados por el conjunto *V:*

$$V = \{v_1, v_2, ...v_m\}$$

En donde *m* puede ser constante o variable no mayor que *n*, cada vehículo tiene una capacidad asociada representada por la función *Q*:

$$Q : V \rightarrow \boldsymbol{N}^+$$

Existe un costo asociado representado por la función *C*:

$$C : R \rightarrow \mathbf{R}^+$$

En donde $C(p_i, p_j)$ es el costo de ir de un punto p_i a un punto p_j y $C(p_i, p_j) = C(p_j = p_i)$.

Cada cliente tiene una demanda asociada representada por la función *D*:

$$D : p_1..p_n \rightarrow \boldsymbol{N}^+$$

Finalmente en algunos casos se presentan situaciones en donde la ruta de cada vehículo no puede exceder un límite de tiempo *TL*, entonces se tiene en cuenta un tiempo representado por la función *T*:

$$T : R \rightarrow \mathbf{R}^+$$

En donde $T(p_i, p_j)$ es el tiempo que un vehículo tarde en viajar desde el punto p_i al punto p_j.

Se tienen en cuenta las variables de número de clientes que visita cada vehículo representadas por *U*:

$$U : V \rightarrow 1..n$$

Se representan las rutas de cada vehículo mediante las funciones F_i, $i = 1, 2, ..., m$

$$F_i : 1..U(v_i) \rightarrow p_1..p_n$$

En donde F_i es la función que representa la ruta del vehículo v_i y $F_i(j)$ representa el j-ésimo punto que recorre el vehículo v_i, esta función a diferencia de las anteriores es inyectiva ya que cada cliente sólo puede ser visitado una vez.

Se plantea entonces el problema de optimización de la siguiente manera:

Minimizar:

$$\sum_{i=1}^{m}\left(\left(\sum_{j=1}^{U(v_i)-1} C(F_i(j), F_i(j+1))\right) + C(p_0, F_i(1)) + C(F_i(U(v_i)), p_0)\right) \tag{2.1}$$

Sujeto a:

$$\bigcap_{i=1}^{m} ran(F_i) = \varnothing \tag{2.2}$$

$$p_1..p_n - \left(\bigcup_{i=1}^{m} ran(F_i)\right) = \varnothing \tag{2.3}$$

$$Q(i) > \sum_{j=1}^{U(v_i)} D(F_i(j)) \quad \textit{para todo } i = 1, 2, ..., m \tag{2.4}$$

$$TL > \left(\sum_{j=1}^{U(v_i)-1} T(F_i(j), F_i(j+1))\right) + T(p_0, F_i(1)) + T(F_i(U(v_i)), 0) \quad \textit{para todo } i = 1, 2, ..., m \tag{2.5}$$

Nótese que en la ecuación (2.2) se establece que no puede haber clientes asignados a más de un vehículo, en la ecuación (2.3) se establece que no queden clientes sin ser atendidos, la ecuación (2.4) determina que la demanda de los clientes asignados a un vehículo no exceda la capacidad de este y en la ecuación (2.5) se determina que cada vehículo no se tarde más de un tiempo límite *TL* haciendo el recorrido. Existen variantes de VRP en donde no se tiene en cuenta un *TL* y/o se tiene capacidad infinita para cada vehículo, dependiendo de estos dos aspectos las restricciones (2.4) y/o (2.5) se tendrán en cuenta.

2.3 Variantes de VRP

El VRP más que un problema individual, es un amplio conjunto de variantes y caracterizaciones de problemas. A continuación se describirán las variantes más comunes; VRP capacitado (CVRP), VRP con múltiples depósitos (MDVRP), VRP con entregas y devoluciones (VRPPD) y VRP periódico (PVRP). La variante VRP con ventanas de tiempo (VRPTW) que será con la que se trabajará en el presente proyecto y será presentada en detalle en la sección 2.4.

- **VRP Capacitado (CVRP)**

Este problema consiste en un VRP con la característica que cada uno de los vehículos de la flota tiene la misma capacidad. Esto quiere decir que se tendría en cuenta la restricción (2.13) planteada en 2.2, por lo tanto se debe tener en cuenta que:

$$Q(v_i) = Q(v_j) \quad para\ todo\ \ i, j = 1, 2, ..., m$$

- **VRP con múltiples depósitos (MDVRP)**

En este problema existen varios depósitos y cada cliente está asignado a uno éstos, teniendo cada depósito una flota de vehículos disponibles, los vehículos deben partir del respectivo depósito visitando clientes asociados a este y llegando de nuevo al mismo depósito.

- **VRP con entregas y devoluciones (VRPPD)**

El VRPPD consiste en que determinados clientes pueden no tener una demanda definida sino que en lugar de eso tienen algún tipo de cantidad de mercancía para devolución. En el caso de transporte de personas se puede plantear de manera que hay puntos en los que se recogen personas y hay puntos en los que se dejan personas. De esta manera la demanda asociada a cada punto podría ser positiva o negativa.

- **VRP Periódico(PVRP)**

En esta variación del VRP los vehículos no deben visitar todos los clientes el mismo día por lo que los vehículos pueden visitar solo algunos de los clientes asignados y luego regresar al depósito para visitar los demás clientes otro día. En éste problema se tiene en cuenta que se debe asignar un día para visitar determinado cliente.

- **VRP de entregas divididas (SDVRP)**

El SDVRP es una relajación del VRP en donde se permite que el mismo cliente pueda ser servido por diversos vehículos si se reduce los costes totales. Esta relajación es muy importante si los tamaños de los pedidos de los clientes son más grandes que la capacidad de los vehículos.

2.4 VRP con ventanas de tiempo (VRPTW)

El problema de enrutamiento de vehículos con ventanas de tiempo, agrega al problema original un intervalo de tiempo en el cual cada cliente puede ser visitado. A continuación se presentan las expresiones matemáticas que permiten extender para VRPTW el modelo matemático presentado en le sección 2.2 para el VRP clásico:

Se define la hora más temprana en la que puede ser atendido cada cliente, mediante la función *EH*:

$$EH : P \rightarrow \mathbf{R}^{+}$$

Y la hora más tardía en la que puede ser atendido cada cliente mediante *LH*:

$$LH : p \rightarrow \mathbf{R}^{+}$$

Teniendo en cuenta también una hora de partida para cada vehículo representada por la función *HV*:

$$HV : V \rightarrow \mathbf{R}^{+}$$

Luego, para el VRPTW se considera la restricción (2.5) expuesta en la sección 2.2 junto con las siguientes restricciones:

$$F_i(k) = p_j \Leftrightarrow HV(v_i) + T(p_0, F_i(1)) \sum_{z=1}^{k-1} T(F_i(z), F_i(z+1)) > EH(p_j) \tag{2.6}$$

$$F_i(k) = p_j \Leftrightarrow HV(v_i) + T(p_0, F_i(1)) \sum_{z=1}^{k-1} T(F_i(z), F_i(z+1)) < LH(p_j) \tag{2.7}$$

$$\text{para todo } k = 1, 2, .., U(i) \quad , i = 1, 2, .., m \ , \ j = 1, 2, .., n$$

2.5 Técnicas de solución para VRP

A continuación se indican las técnicas más comunes de resolución del VRP, clasificadas en exactas, heurísticas y metaheurísticas. En su mayoría las técnicas utilizadas son del tipo heurísticas y metaheurísticas, ya que los métodos exactos de resolución no garantizan encontrar la solución óptima en un tiempo razonable de computación cuando el número de clientes es de gran tamaño.

2.5.1 Técnicas exactas

- **Branch and Bound**

La técnica de Branch and Bound [8] consiste en ir construyendo un árbol con todas las posibles soluciones pero en el momento que una rama ya no sea la mejor, se deja de construir el árbol por esa rama, para ahorrar recursos computacionales, de esta manera se puede llegar a la solución óptima sin necesidad de explorar todas y cada una de las posibles soluciones.

Para el caso del VRP se debe tener una solución factible inicial con una distancia total recorrida asociada y así realizar el árbol cortando las ramas que superen esta distancia.

- **Branch and Cut**

Branch and Cut [8] es una técnica derivada de Branch and Bound en donde se tratan los problemas combinatorios sin tener en cuenta la restricción de soluciones enteras, esto lo hace mediante el método simplex tradicional. Cuando una solución óptima es encontrada, se utiliza un algoritmo de plano de corte para encontrar restricciones que satisfagan los puntos enteros factibles que fueron violados en la solución que fue hallada sin tener en cuenta las restricciones de variables enteras. Si se hallan desigualdades, éstas son agregadas al problema de programación lineal esperando que la solución sea lo más cercana a valores enteros. De esta manera se repite el proceso hasta encontrar una solución con valores enteros o hasta que no se encuentren más planos de corte.

2.5.2 Técnicas heurísticas

Realizan una exploración limitada en el espacio de soluciones, produciendo soluciones razonablemente buenas con tiempos de computación modestos. Se pueden clasificar en métodos de construcción, que construyen de forma gradual una solución factible a la vez que tratan de minimizar el costo, pero de por sí no tienen una fase de mejora. Ejemplo de método de construcción es el algoritmo de ahorro de Clarke and Wright. Y algoritmos de dos fases, que consisten en agrupar los clientes por zonas adecuadamente escogidas que van a ser atendidas cada una por un vehículo, y luego aplicar algún otro procedimiento para resolver cada zona por separado fijando la ruta de cada

vehículo. Dentro de los algoritmos de dos fases se encuentran el algoritmo de barrido y el algoritmo de pétalo.

- **Algoritmo de ahorro de Clarke and Wright**

El algoritmo de ahorro de Clarke and Wright [1] es una de las más conocidas heurísticas para VRP, es utilizada cuando el número de vehículos es variable. Cuando las rutas de dos vehículos F_1 y F_2 se pueden unir de manera factible en una sola ruta F_k:

$$F_k(i) = F_1(i) \quad \textit{para todo } I = 1, 2, .., U(v_1)$$

$$y \quad F_k(i + (U(v_2))) = F_1(i) \quad \textit{para todo } I = 1, 2, .., U(v_2)$$

Se establece una distancia de ahorro δ_{ij}:

$$\delta_{ij} = C(F_k(U(v_1)), p_0) + C(p_0, F_2(1)) - C(F_1(U(v_1), F_2(1)))$$

Teniendo en cuenta esto, se pueden aplicar dos tipos de algoritmos, el secuencial y el paralelo, en ambos algoritmos el primer paso es el proceso mostrado en el Algoritmo 1.

Algoritmo 1 Paso 1 Clarke and Wright

Desde i=1 hasta n−1 hacer

  Desde j=i+1 hasta n hacer

    $\delta_{ij} = C(F_k(U(v_1)), p_0) + C(p_0, F_2(1)) - C(F_1(U(v_1), F_2(1)))$

  Fin desde

Fin desde

Desde k=1 hasta n hacer

$F_k(1) = k$

Fin desde

El algoritmo paralelo consiste en lo siguiente:

Algoritmo 2 Paso 2 paralelo Clarke and Wright

Aplicar Algoritmo 1

Ordenar en orden Descendente cada δ_{ij}

k = 0

k = k + 1

Para el k-ésimo δ_{ij} ordenado determinar si existen dos rutas *Fq y Fp* tal que *Fp(1) = pi* y *Fq(U(vq)) = pj* . Si existen unirlas en una

Si *k = ((n2 −n)/2)* finalizar, de lo contrario pasar a 4

En el algoritmo secuencial se siguen los siguientes pasos:

Algoritmo 3 Paso 2 secuencial Clarke and Wright

1. Aplicar Algoritmo 1
2. Ordenar en orden Descendente cada δ_{ij}
3. Escoger una ruta *Fq* que no haya sido escogida. Si no existe, terminar el proceso
4. Determinar cuál es el primer δ_{ij} ordenado con el cual exista otra ruta *Fp* tal que *Fp(1) = pi* y *Fq(U(vq)) = pj* o tal que *Fp(U(vp)) = pi y Fq(1) = pj* .(Si existe, unir las rutas y repetir este paso, de lo contrario ir al paso 4)

La aplicación de este algoritmo busca minimizar la distancia recorrida para satisfacer la demanda y al mismo tiempo busca minimizar el número de vehículos usados. En esta técnica no se han hecho muchos nuevos desarrollos, sin embargo puede llegar a ser usada en aplicaciones de la vida real obteniendo resultados rápidos y prácticos.

- **Algoritmo de barrido**

El proceso en un algoritmo de barrido [1] consiste crear las zonas mediante rayos que tienen como centro el depósito y donde los ángulos que hay entre éstos, definen las zonas.

Algoritmo 4 Algoritmo de barrido

1. Fijar el rayo con ángulo 0
2. Escoger un vehículo *k* no usado
3. Crear otro rayo aumentando el ángulo y a medida que la zona vaya abarcando clientes verificar si se cumplen las restricciones (2.4) y (2.5) de tiempo y capacidad para cada vehículo, cuando el ángulo no se pueda aumentar debido a esto, asignar los clientes de la zona al vehículo *k* escogido. Si faltan clientes por atender ir al paso 2.
4. Optimizar cada zona por separado resolviendo el problema del vendedor viajero para cada vehículo.

- **Algoritmo de pétalo**

El algoritmo de pétalo [1] es un algoritmo derivado del Algoritmo de Barrido, en el cual se generan varias rutas denominadas pétalos y después se hace una selección final planteando el problema de la siguiente manera:

Minimizar: $\sum_{k \in S} d_k X_k$

Sujeto a: $\sum_{k \in S} a_{ik} x_k = 1 \quad para\ i = 1, 2, ..., n$

Donde *S* es el grupo de rutas, X_k es la variable binaria que define si la ruta *k* pertenece al conjunto solución, a_{ik} es la variable binaria que define si el punto *i* pertenece a la ruta *k* y d_k es el costo del pétalo *k*.

2.5.3 Técnicas metaheurísticas

Las metaheurísticas son métodos que realizan una exploración intensiva del espacio de soluciones, obteniendo una mayor calidad de las soluciones que las entregadas mediante métodos heurísticos clásicos. Dentro de las principales metaheurísticas usadas tanto en VRP como en otros problemas de optimización combinatoria se encuentran: la búsqueda tabú, recosido simulado, colonias de hormigas y algoritmos genéticos. A continuación se hace una descripción de las técnicas metaheurísticas mencionadas, dejando la explicación de algoritmos genéticos para el siguiente capítulo, en donde se describirá con mayor detalle.

- **Búsqueda Tabú (TS)**

La Búsqueda Tabú [2] es una técnica iterativa de búsqueda local que trata de evitar que las soluciones caigan en óptimos locales. Para esto utiliza unas estructuras de memoria de corto y largo plazo. Dentro de una 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 término al igual que en Algoritmos Genéticos puede ser un cierto número máximo de iteraciones o un valor de la función a optimizar.

El uso de memoria consiste en almacenar en una lista tabú soluciones que ya han sido visitadas o almacenar atributos de soluciones ya exploradas. De esta manera la solución actual va a pasar o no a otra solución dependiendo de los datos almacenados en la memoria. Esta lista tabú puede ser variable dentro de las iteraciones.

Las memorias se pueden clasificar en memorias de corto y largo plazo. La memoria de corto plazo consiste en almacenar los últimos movimientos realizados, lo cual permite hacer una mejor explotación de la zona en la que se está buscando. La memoria de largo plazo consiste en guardar los atributos de un conjunto de soluciones para identificar regiones de búsqueda, y así generar una mejor exploración. Existen unos niveles de aspiración que consisten en suavizar la prohibición de pasar a una solución que esté en la lista tabú, de esta manera se evita que por ejemplo el uso de la memoria atributiva lleve a un proceso cíclico.

En la figura 2.2 se muestra el proceso básico de la búsqueda Tabú.

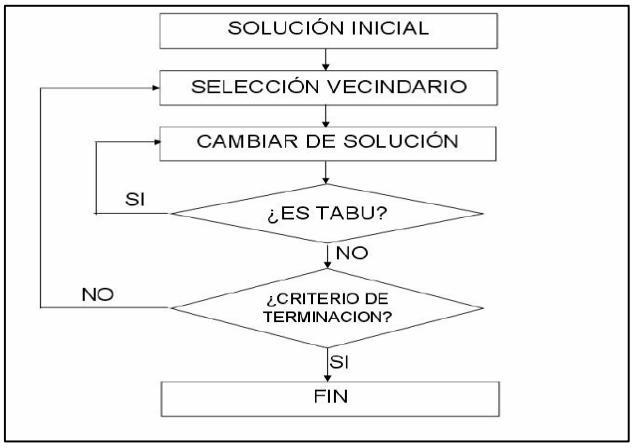


Figura 2.2: Algoritmo básico búsqueda tabú.

- **Recosido Simulado (SA)**

La técnica de recosido simulado [4] 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 óptima 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.

El proceso de recocido simulado al igual que el de otras metaheurísticas es iterativo y cada iteración corresponde a una fase de enfriamiento, en cada iteración se realiza una perturbación de la solución, el reemplazo que produce esta perturbación se hará si es mejor que la solución actual o si cumple cierta regla de aceptación que es probabilística y está en función de la temperatura. En la Figura 2.3 se muestra el proceso básico que se sigue en un algoritmo de recocido simulado.

Para la aplicación de la técnica al VRP es importante establecer de que manera va a ser perturbada la solución, para esto se pueden utilizar métodos como los usados en Algoritmos Genéticos para hacer mutación y en Búsqueda Tabú para la escogencia de la vecindad.

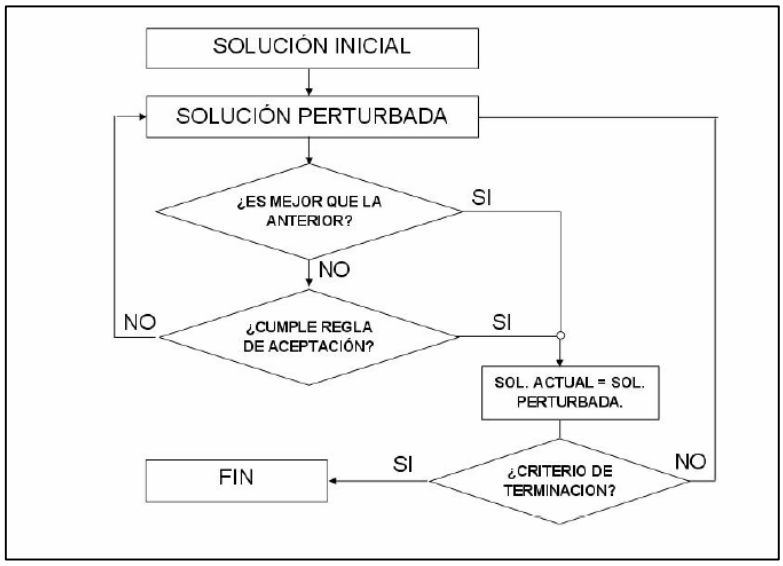


Figura 2.3: Algoritmo básico de recosido simulado.

- **Colonias de hormigas (ACO)**

La Optimización Colonia de Hormigas [7], es un método de optimización que está basado en el comportamiento de las hormigas en cuanto a la forma en que estas buscan su alimento. Es un hecho que las hormigas encuentran los caminos más cortos entre el

hormiguero y las fuentes de alimento aunque no puedan utilizar la visión como una forma para hacerlo debido a su casi completa falta de este sentido. Las hormigas empiezan a buscar su alimento moviéndose aleatoriamente y en este proceso van expulsando una sustancia llamada feromona la cual puede ser muy susceptible de olor para las otras hormigas, de esta manera las hormigas que encuentren primero el alimento van a regresar más rápido al hormiguero y van a dejar en el camino una concentración mayor de feromona la cual va a ser seguida por las demás hormigas dejando la búsqueda aleatoria, de esta manera todas las hormigas empiezan a seguir el mejor camino hacia la fuente de alimentación. De esta manera para la implementación de un modelo de optimización basado en lo anterior, se crea un modelo artificial de colonia de hormigas en donde se definen parámetros tales como niveles de feromona, tendencia de las hormigas a seguir a un nodo más cercano o a un nodo donde la arista tiene un mayor nivel de feromona, entre otros.

En la implementación de ACO para VRP cada vehículo va a ser representado por una hormiga que va a partir del depósito y va a ir visitando clientes hasta que su capacidad esté completa o hasta que no haya más clientes por visitar. Las hormigas (vehículos) van a realizar el procedimiento una después de otra hasta que todos los clientes hayan sido visitados.

Un aspecto tenido en cuenta en ACO es el hecho de que el nivel de feromona se va evaporando, ya que una ruta se puede volver muy predominante haciendo caer la solución a óptimos locales.

De esta manera se pueden diseñar algoritmos definiendo parámetros tales como número de iteraciones y nivel de feromona inicial, entre otros, que permitan obtener una buena solución para los diferentes tipos de VRPs.

Capítulo 3

3 Algoritmos genéticos

3.1 Conceptos generales

Los algoritmos genéticos (GA) fueron inventados y desarrollados por John Holland y sus estudiantes de la Universidad de Michigan en los inicios de la década de 1970, inspirado en la Teoría de la evolución de las especies de Charles Darwin [6]. Holland presenta a los GAs como una abstracción de la evolución biológica y entrega un marco de trabajo teórico para la adaptación natural bajo los GAs.

La idea central de los GAs es modelar la evolución natural por medio del uso de herencia genética en conjunto con la teoría de Darwin. En los GAs la población, que consiste en un conjunto de *soluciones* o *individuos* en lugar de cromosomas. Un operador de cruzamiento (*crossover*) juega el rol de reproducción, un operador de mutación es asignado para hacer cambios aleatorios en las soluciones y un operador de inversión revierte el orden de una sección continua en un cromosoma, cambiando el orden de sus genes. Mediante un procedimiento selectivo se escoge desde la población las soluciones que están más capacitadas para reproducirse, logrando que en promedio las soluciones de mayor calidad generen nuevos individuos para la población.

El proceso de un algoritmo genético básico se muestra en la figura 3.1. El proceso en el que cada vez que se iteran los pasos de evaluación, selección y operadores genéticos es llamado una generación y se itera creando nuevas generaciones hasta que algún criterio de término sea cumplido.

La representación de las soluciones se hace mediante cadenas de genes llamadas cromosomas en donde se define claramente la solución, un aspecto muy importante en GAs es definir cómo se van a representar las soluciones en estos cromosomas ya que esto es lo que va a permitir un proceso de evaluación más sencillo y con mejores resultados.

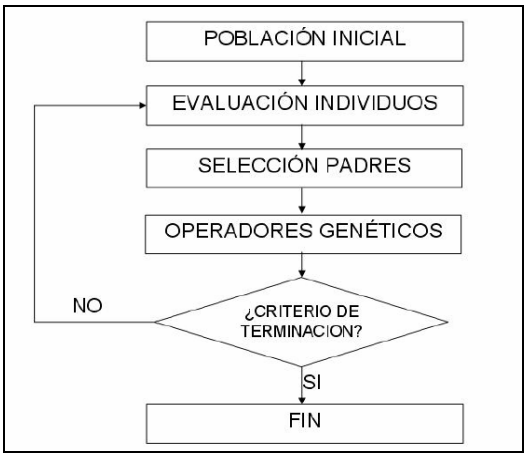


Figura 3.1 Algoritmo genético básico.

A los GAs generalmente se les refiere en plural, ya que existen varias versiones ajustadas para diferentes problemas. A continuación se mencionarán características que los GAs tienen en común y que las distinguen de otras heurísticas [10]:

- Los GAs trabajan con una codificación de las soluciones en vez de ellas mismas, por lo tanto se requiere contar con una buena y eficiente representación de las soluciones en forma de cromosomas.

- Para la búsqueda los GAs utilizan un conjunto de soluciones, a diferencia de otras metaheurísticas como SA o TS que comienzan la búsqueda con una solución individual y se mueven a otra solución mediante transiciones. De esta manera se logra una búsqueda multidireccional en el espacio de solución, reduciendo la probabilidad de terminar en un óptimo local.

- Los GAs sólo necesitan los valores de la función objetivo, no sus derivadas ni otros datos auxiliares.

- Los GAs son no deterministas, es decir sus decisiones son estocásticas, lo que las hacen ser más robustas.

- Los GAs son ciegos porque no saben cuando han encontrado una solución óptima.

3.2 Población inicial

La población inicial es la principal fuente de material genético para el algoritmo. En la población inicial los individuos deben estar bien dispersos por el espacio de soluciones.

La manera más simple de cumplir con este objetivo es elegir individuos al azar, pero el uso de alguna técnica de indagación y descubrimiento, puede ayudar a generar una

población inicial compuesta de soluciones de mediana calidad, ahorrando tiempo al proceso de evolución, siempre y cuando se garantice una dispersión suficiente para la búsqueda.

3.3 Evaluación de los individuos

Para la evaluación de los individuos simulando la selección natural, se utiliza una variable llamada *fitness*, que representa que tan buenos son los individuos que hay en la población. El valor fitness mide la calidad de las soluciones y permite que sean comparadas.

La selección de los individuos tanto para la reproducción como para la supervivencia es crucial para la eficiencia del GA, una selección muy ambiciosa induce a convergencias prematuras, que es uno de los grandes problemas de los GAs. Dado que los métodos de selección se basan en el valor fitness, es importante elegir cuidadosamente una función de evaluación a utilizar.

La función de evaluación ha de medir la adaptación de cada uno de los individuos, es decir, la calidad de cada solución del problema, y depende fundamentalmente de la representación elegida, puesto que la única información que puede evaluar esta función va a ser la contenida en los genes según la representación utilizada.

Otro punto a tener en cuenta para la evaluación de los individuos es el *escalado*, cuando el valor fitness de los elementos de la población varía mucho, es común que la población contenga unas pocas soluciones cercanas al óptimo, rodeadas de otras mediocres. Los individuos más aptos obtendrán valores fitness exageradamente superiores al promedio, y el algoritmo genético elegirá solamente a estos para la reproducción. Esto puede ocasionar lo que se denomina *convergencia prematura*, que es que el algoritmo genético encontrará una solución sin haber explorado suficientemente el espacio de búsqueda. Sin embargo, a medida que la aptitud promedio va subiendo, el problema será diferente, la diferencia irá disminuyendo y los individuos más aptos y los más débiles obtendrán valores fitness similares, reduciendo la probabilidad de que el algoritmo genético seleccione a los mejores para la reproducción. Los dos problemas pueden corregirse aplicando una función de escalado al valor fitness. Un ejemplo de función de escalado para el valor fitness es la función de escalado lineal mostrado en la ecuación 3.1. En [10] se pueden revisar más funciones de escalado para el valor fitness.

La función de escalado lineal calcula un nuevo valor fitness *f'* a partir del fitness inicial *f* usando la siguiente transformación lineal:

$$f' = a \times f + b \tag{3.1}$$

Las constantes *a* y *b* se eligen de forma tal que el promedio de las dos funciones sea el mismo, y que el máximo para la función *f'* asegure la diferencia de probabilidades deseada entre los elementos más aptos y el promedio.

3.4 Selección

En los GAs es necesario tener un mecanismo de selección que será el encargado de escoger qué individuos van a disponer de oportunidades de reproducirse y cuáles no.

Un problema común al momento de aplicar mecanismos de selección en los GAs es una prematura o muy rápida convergencia. La *convergencia* es una medida de cuán rápido mejora la población, si la población mejora muy rápido indica que los individuos débiles son separados muy pronto, sin que puedan traspasar sus características.

La *presión de selección* es la medida de con qué frecuencia los individuos con mejores características son seleccionados en comparación con los más débiles. Una alta presión de selección significa que la mayor parte del tiempo los mejores individuos son seleccionados y los más débiles rara vez son escogidos. Por otro lado con una baja presión de selección los individuos débiles tienen una gran oportunidad de ser seleccionados.

Una alta presión de selección estimula a una rápida convergencia, lo que provoca que la búsqueda de soluciones se centre en un entorno próximo a las mejores soluciones iniciales. Por el contrario una baja presión de selección hace la búsqueda más ineficiente, pero deja el camino abierto a la exploración de nuevas regiones del espacio de búsqueda.

Es muy importante conseguir un balance de la presión de selección y de la diversidad de la población para obtener tan buenas soluciones como sea posible.

A continuación se mencionarán algunas de las técnicas de selección más conocidas.

- **Método de selección de la ruleta**

El método de selección de la ruleta (*roulette wheel method*), se basa en asignar una probabilidad de selección a cada individuo de acuerdo a su valor fitness que representa su calidad. Esto permite que aunque los individuos con alta calidad tengan altas posibilidades, no necesariamente serán los seleccionados. Lo mismo en el caso de los individuos con baja calidad que igualmente tienen posibilidades de ser seleccionados, esto favorece la variedad de los individuos y evitar la convergencia prematura.

En la figura 3.2 se ilustra el método de la ruleta para un problema con cinco individuos en la población, en donde el individuo p_i tiene un valor fitness f_i.

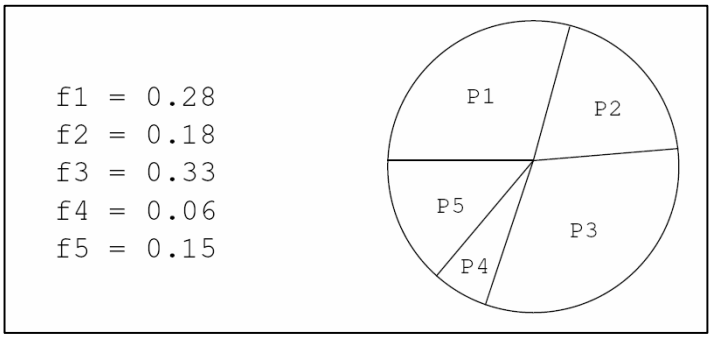


Figura 3.2 Método de selección de la ruleta.

- **Método de selección del ranking**

En este método se construye un *ranking* ordenando a los individuos de acuerdo a su valor fitness. De esta manera si tenemos *n* cromosomas, el individuo con peor fitness se le asignará el valor 1 y el que tenga el mejor fitness se le asignará el valor *n*. Una función dependiendo del ranking es utilizada para seleccionar un individuo, así los individuos son seleccionados proporcionalmente a su ranking en lugar de su valor fitness como en el método de la ruleta. Por ejemplo la selección se puede basar en la siguiente distribución de probabilidad:

$$p(k) = \frac{2k}{M(M+1)} \tag{3.2}$$

La constante *k* indica el *k-ésimo* individuo en el ranking y *M* el tamaño de la población. El mejor individuo *(k = M)* tiene una probabilidad $2/(M+1)$ de ser seleccionado y el peor individuo *(k = 1)* tiene probabilidad $2/M(M+1)$. En este ejemplo la probabilidad es proporcional al tamaño de la población.

La ventaja de este método [10] es su mayor capacidad para controlar la presión de selección que el método de la ruleta. Aunque tiene como inconveniente que trata uniformemente todos los casos y deja de lado la magnitud del problema.

- **Método de selección del torneo**

En el método de selección del torneo un padre es seleccionado por la elección del mejor individuo de un conjunto o un subgrupo de la población.

Inicialmente, dos subgrupos de tamaño *S* son seleccionados aleatoriamente, en el caso que se necesiten dos padres. Si *k* individuos de la población fueran cambiados en cada iteración, el número de subgrupos sería *k*. Cada subgrupo debe contener al menos dos individuos, para permitir una comparación entre ellos. El tamaño de los subgrupos influencia la presión de selección, es decir, más individuos en los subgrupos incrementa las

posibilidades de seleccionar a los mejores individuos. Dentro de cada subgrupo, los individuos compiten por la selección de manera parecida a un torneo, cuando se deben seleccionan los individuos para la reproducción los mejores de cada grupo son elegidos. Por otro lado, se escoge a los individuos más débiles cuando el método es usado para seleccionar quien dejará la población, luego el individuo más débil ya no sería seleccionado para reproducirse y más importante aun el mejor individuo nunca dejaría la población.

3.5 Operadores genéticos

En la presente sección se describirán los operadores genéticos utilizados por los GAs: cruzamiento, mutación e inversión. Estos operadores permiten que las soluciones puedan reproducirse y mejorar sus características, mutar cambiando aleatoriamente sus genes y así poder moverse por el espacio de búsqueda.

A continuación se realiza una descripción de los operadores mencionados:

3.5.1 Cruzamiento

El principal operador genético es el Cruzamiento (*crossover*), el cual simula la reproducción entre dos organismos (los padres). Este operador combina copia de los genes de los padres para generar una o más descendencia que recibe las características de sus padres, de esta manera las características se traspasa a las futuras generaciones.

A continuación de describen dos de los cruzamientos más conocidos.

- **Cruzamiento en dos puntos**

En éste mecanismo se escogen dos puntos en los cromosomas y los segmentos entre los puntos seleccionados se combinan en los nuevos individuos. Por ejemplo, suponiendo que se tienen los siguientes cromosomas representados en una cadena binaria:

1101 | 0111 | 00001
0101 | 0101 | 00011

Al realizar el cruzamiento en dos puntos, para éste caso, en las posiciones 5 y 9, se obtienen los siguientes cromosomas:

Hijo 1: 1101 | 0101 | 00001
Hijo 2: 0101 | 0111 | 00011

- **Cruzamiento uniforme**

En este mecanismo los genes en cada posición de los cromosomas padres se comparan, con una probabilidad constante los genes se intercambian. Una variante a éste método de cruzamiento es el *medio cruzamiento uniforme (half uniform crossover)*, donde exactamente la mitad de los genes distintos se intercambian.

3.5.2 Mutación

Este operador corresponde a la modificación de genes particulares de un individuo bajo una muy baja probabilidad ocurrencia. El objetivo principal de la mutación es mantener la diversidad de la población y evitar que exista convergencia prematura en la ejecución del algoritmo, además de permitir explorar otros sectores en el espacio de búsqueda. Es importante no alterar muy a menudo las soluciones con éste operador, ya que de lo contrario el algoritmo se convertiría en una búsqueda aleatoria. Una versión muy simple del proceso de mutación es ilustrada en la figura 3.3.

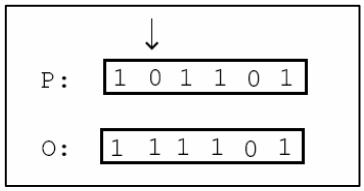


Figura 3.3 Ilustración del procedimiento de mutación.

La cadena binaria *P* representa la solución padre. Aleatoriamente, el segundo bit ha sido seleccionado para mutar. Como resultado la descendencia *O* ha cambiado su segundo bit de 0 a 1.

3.5.3 Inversión

El tercer operador genético es la inversión, el cual revierte el orden de una sección continua de un individuo cambiando el orden de los genes respectivos. De forma similar al operador de mutación, es aplicado a una solución de manera individual. En la figura 3.4 se ilustra éste procedimiento con una cadena de caracteres.

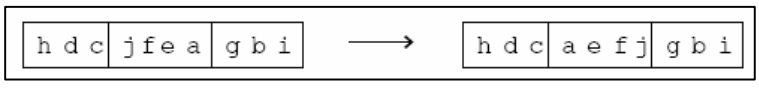


Figura 3.4 Ilustración de procedimiento de inversión.

En la cadena de caracteres, que podría representar una simple ruta, se seleccionan aleatoriamente dos cortes entre el tercer y cuarto, y séptimo y octavo carácter,. Luego se revierte el orden entre los cortes.

3.6 Criterio de término

El criterio de término para los GAs es el encargado de definir el momento en el cual debe interrumpir el ciclo de evolución y adoptar el individuo más apto como la solución encontrada por el algoritmo genético.

A continuación se describen algunos criterios de término comúnmente utilizados.

- **Criterio de Convergencia de Identidad**

Este criterio consiste en detener el algoritmo genético cuando un determinado porcentaje de los individuos representa a la misma solución. Los operadores tienden a preservar y difundir el material genético de los cromosomas más aptos, por lo que es de esperar que luego de un gran número de generaciones, alguna solución con gran valor de aptitud se imponga y domine la población.

- **Criterio de Convergencia de Aptitud**

Puede suceder que existan soluciones equivalentes o casi equivalentes a un problema, que obtengan valores fitness parecidos. En estos casos, es probable que no haya una solución que se imponga en la población (y el criterio de término por convergencia de identidad nunca se cumpla).

Este criterio no espera que la población se componga mayoritariamente de una sola solución, sino que finaliza la ejecución del algoritmo cuando los valores fitness de un determinado porcentaje de las soluciones son iguales o difieren en un porcentaje dado. Por ejemplo, cuando el 90% de las soluciones tenga valores de aptitud que no difieran en más de un 1%.

- **Criterio de Cantidad de Generaciones**

El criterio de término por cantidad de generaciones consiste simplemente en finalizar la ejecución una vez que ha transcurrido un número determinado de generaciones.

Los métodos anteriores apuntan a esperar a que la evolución de la población llegue a su fin. Cuando alguno de ellos se cumple, es probable que las soluciones no sigan mejorando mucho más, no importa cuántas generaciones más se ejecuten. Sin embargo, los GAs pueden necesitar un número de generaciones muy grande para llegar a la

convergencia, dependiendo de las tasas de reproducción y mutación. Utilizando cualquiera de los dos criterios anteriores no puede estimarse un número máximo de generaciones, ya que esto dependerá no solamente de los parámetros del algoritmo genético sino también del azar. Esto puede ser un problema, en el caso que se quisiera comparar los tiempos de resolución de un problema mediante GAs con otros métodos.

Este método permite determinar con precisión los tiempos de ejecución del algoritmo a costa de detener la evolución sin la certeza de que las soluciones no seguirán mejorando.

Capítulo 4

4 Algoritmos culturales

Los algoritmos culturales fueron desarrollados por Robert G. Reynolds [9], como un complemento a la metáfora que usan los algoritmos de computación evolutiva, que se han concentrado en conceptos genéticos, y de selección natural.

La cultura puede verse como un conjunto de fenómenos ideológicos compartidos por una población, pero por medio de los cuales, un individuo puede interpretar sus experiencias y decidir su comportamiento. En estos modelos se aprecia muy claramente la parte del sistema que es compartida por la población: el conocimiento, recabado por miembros de la sociedad, pero codificado de tal forma que sea potencialmente accesible a todos. De igual manera se distingue la parte del sistema que es individual: la interpretación de ese conocimiento codificado en forma de un conjunto de símbolos, y los comportamientos que trae como consecuencia su asimilación; también la parte individual incluye las experiencias vividas, y la forma en que estas pueden aportar algo al conocimiento compartido.

Los algoritmos culturales están basados en las teorías de algunos sociólogos y arqueólogos, que han tratado de modelar la evolución cultural. Tales investigadores indican que la evolución cultural puede ser vista como un proceso de herencia en dos niveles:

- Nivel micro-evolutivo, que consiste en el material genético que es heredado por los padres a sus descendientes.

- Nivel macro-evolutivo, que es el conocimiento adquirido por los individuos a través de las generaciones, y que una vez codificado y almacenado, sirve para guiar el comportamiento de los individuos que pertenecen a una población.

Tomando en cuenta que la evolución se puede ver como un proceso de optimización, Reynolds desarrollo un modelo computacional de evolución cultural que puede tener aplicaciones en optimización, que denominó como algoritmos culturales [9]. Para ello captó el fenómeno de herencia doble con la finalidad de incrementar las tasas de aprendizaje y convergencia.

En este modelo cada uno de los niveles está representado por un espacio. El nivel micro-evolutivo por un espacio de la población, y el macro-evolutivo por un espacio de creencias.

4.1 Espacio de la población

El espacio de la población puede ser adoptado por cualquiera de los paradigmas de computación evolutiva, como los algoritmos genéticos, en los que se tienen un conjunto de individuos donde cada uno de ellos tiene un conjunto de características independientes entre sí, con las que es posible determinar su aptitud (fitness). A través del tiempo, tales individuos podrán ser reemplazados por algunos de sus descendientes, obtenidos a partir de un conjunto de operadores aplicados a la población.

4.2 Espacio de creencias

El espacio de creencias es donde se almacenan los conocimientos que han adquirido los individuos de generaciones anteriores. La información contenida en este espacio debe ser accesible a cualquier individuo, quien puede utilizarla para modificar su comportamiento.

Desde sus inicios se ha trabajado con cinco tipos de conocimiento para ser almacenados en el espacio de creencias, ellos son: Circunstancial, Normativo, Topográfico, Histórico y del Dominio. Los cuales según Reynolds [9] forman un conjunto completo, es decir, que cualquier otro tipo de conocimiento que se desee agregar al espacio de creencias puede ser generado mediante una combinación de dos o más de los tipos de conocimiento existentes.

- **Conocimiento Circunstancial**

Consiste en almacenar el mejor individuo del proceso evolutivo hasta ese momento, con el fin de considerarlo como líder para la siguiente generación. La influencia de este conocimiento se hace efectiva al hacer tender a las variables de decisión de los nuevos individuos a los valores del almacenado en el espacio de creencias.

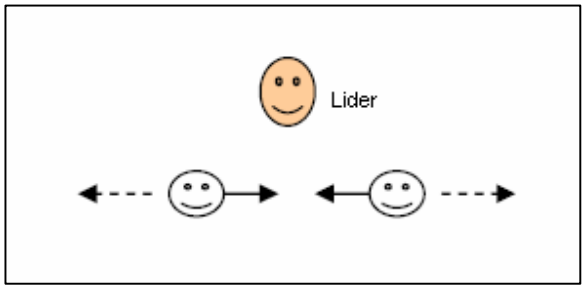


Figura 4.1 Conocimiento Circunstancial.

- **Conocimiento Normativo**

Consiste en almacenar el rango de cada variable de decisión donde se han encontrado los mejores valores, con el fin de hacer tender a las nuevas soluciones a dichos intervalos. La actualización de éste espacio de conocimiento puede ampliar o disminuir los intervalos en el rango.

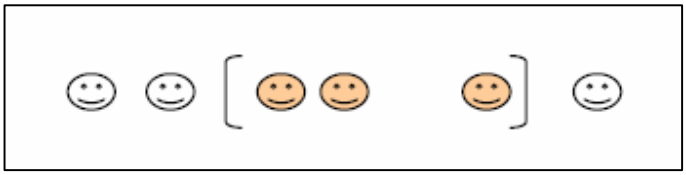


Figura 4.2 Conocimiento Nominativo.

- **Conocimiento Topográfico**

Este tipo de conocimiento consiste en la creación de un mapa del lugar donde se encuentran las soluciones del problema en el proceso evolutivo. Este mapa es dividido en regiones que pueden ser clasificadas en factibles, infactibles, semi-factibles y desconocidas, de acuerdo al contenido de cada una de ellas. La influencia de este conocimiento se hace efectiva al hacer tender a los nuevos individuos a las regiones factibles.

Este tipo de conocimiento se representa como un conjunto de celdas donde se almacenan los mejores individuos encontrados. Cada una de las celdas se encuentra ordenada, la actualización de este espacio de conocimiento se hace cuando un individuo de la nueva generación es mejor que el mejor de los valores que se encuentran en la celda respectiva.

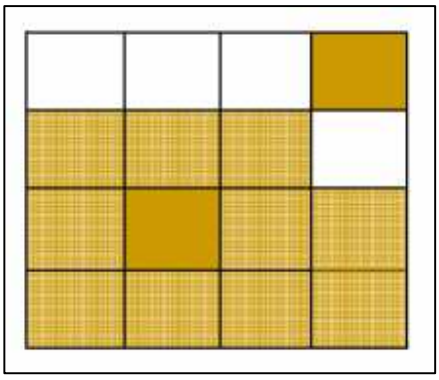


Figura 4.3 Conocimiento Topográfico.

- **Conocimiento Histórico**

En este tipo de conocimiento se guarda una lista de los mejores individuos que aparecen en cada uno de los cambios de entorno, con el fin de no perder los buenos eventos encontrados en las generaciones anteriores.

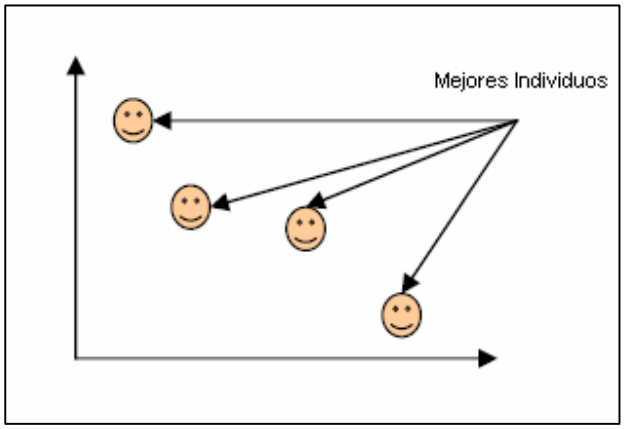


Figura 4.4 Conocimiento Histórico.

- **Conocimiento del Dominio**

Utilizar el conocimiento del dominio del problema para guiar la búsqueda. Depende directamente del tipo de problema y del espacio de búsqueda con que se cuente, es difícil de modelar si no se tiene suficiente conocimiento del problema.

4.3 Protocolo de comunicación

Para unir ambos espacios, se establece un protocolo de comunicación, que dicta reglas respecto al tipo de información que se debe intercambiar entre los espacios. Este protocolo define dos funciones:

- **Función de aceptación**

Esta función se encarga de extraer la información o experiencias que han obtenido los individuos de una generación y llevarlas al espacio de creencias.

- **Función de influencia**

Esta función se encarga de influir en la selección y sobre los operadores de variación de los individuos (como el cruzamiento y mutación en el caso de los GAs). Lo que significa

que esta función ejerce un tipo de presión para que los individuos resultantes de la aplicación de los operadores de variación se acerquen a los comportamientos deseables y se alejen de los indeseables, según la información almacenada en el espacio de creencia.

La interacción entre los espacios de la población y de creencias se muestras en la figura 4.5.

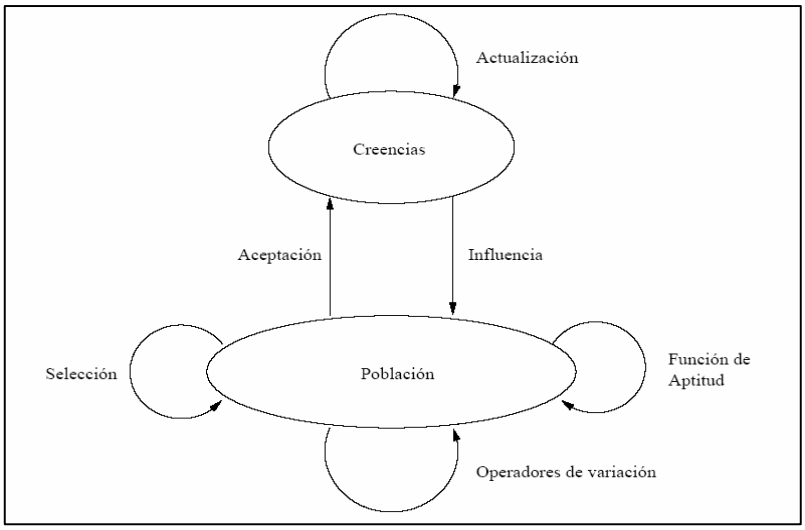


Figura 4.5 interacción entre los espacios de un algoritmo cultural.

4.4 Diseño de un algoritmo cultural

Después de haber mencionado en las secciones anteriores las características básicas de un algoritmo cultural, se presenta su funcionamiento en el algoritmo 5.

Algoritmo 5 Algoritmo cultural

```
Generar población inicial
Iniciar el espacio de creencias
Evaluar población inicial
Repetir
        Actualizar el espacio de creencias (con los individuos aceptados)
        Aplicar operadores de variación (bajo la influencia del espacio de creencias)
        Evaluar cada hijo
        Realizar la selección
Mientras no se cumpla la condición de finalización
```

La mayoría de los pasos de un algoritmo cultural corresponden con los de los algoritmos tradicionales de computación evolutiva, y se puede apreciar que las diferencias están en los pasos que incluyen al espacio de creencias.

En el ciclo principal está la actualización del espacio de creencias. Es en ese momento donde el espacio de creencias incorpora las experiencias individuales de un grupo selecto de individuos. Tal grupo se obtiene entre toda la población con la función de aceptación.

Por otro lado, los operadores de variación de los individuos (como la recombinación o la mutación) son modificados por la función de influencia. La función de influencia ejerce cierta presión, para que los hijos resultantes de la variación se acerquen a los comportamientos deseables y se alejen de los indeseables, según la información almacenada en el espacio de creencias.

Estas dos funciones, la de aceptación y la de influencia, son mediante las cuales se establece la comunicación entre los espacios de la población y de creencias.

Capítulo 5

5 Trabajos relacionados con VRPTW

A continuación se presentarán cuatro trabajos aplicados al problema de enrutamiento de vehículos con ventanas de tiempo basándose en algoritmos genéticos:

5.1 Zhu K. Q.

El trabajo de Zhu [12] abarca el problema VRPTW usando GAs para obtener soluciones aproximadas del problema. Se basa en una representación intuitiva de la solución, operadores especializados de recombinación y la aplicación de técnicas como *hill-climbling* y esquemas de mutación adaptativa usando medidas estadísticas.

Cada individuo se representa como una permutación de los clientes. La representación se esboza del siguiente modo: un cromosoma se estructura como una cadena de enteros de largo *n*, donde *n* corresponde a la cantidad de clientes del problema. Cada gen en el cromosoma representa un nodo asociado a un cliente, y en conjunto, los genes representan un conjunto de rutas, ordenados de acuerdo a su aparición en el cromosoma. Por ejemplo, se tiene la siguiente solución:

Ruta 1: $0 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 0$
Ruta 2: $0 \rightarrow 10 \rightarrow 6 \rightarrow 1 \rightarrow 12 \rightarrow 11 \rightarrow 0$
Ruta 3: $0 \rightarrow 9 \rightarrow 8 \rightarrow 7 \rightarrow 0$

El cromosoma que representa esta solución sería de la forma:

$$3 - 2 - 4 - 5 - 10 - 6 - 1 - 12 - 11 - 9 - 8 - 7$$

Para decodificar las rutas desde el cromosoma, se van revisando los genes y la capacidad que conlleva la inserción del mismo en la ruta que se está conformando. No necesariamente se obtendrán las mismas rutas codificadas, pero por el asunto de la capacidad máxima, las rutas que se obtienen son potencialmente efectivas.

Para la población inicial, se usa una heurística denominada *Push-Forward Intertion Heuristic* (PFIH), la que se basa en tener una solución inicial factible y relativamente buena (obtenida por algún tipo de heurística). De ella y de sus vecinos cercanos aleatorios en el

espacio de soluciones, se obtiene una porción de la población inicial. El resto es obtenido de forma aleatoria, con el fin de explorar otras regiones.

La selección se basa en el mecanismo de torneo explicado en la sección 3.4. Para la recombinación, se usan tres mecanismos adaptados para la operación con cromosomas con algún grado de ordenamiento. Uno es el *PMX Crossover*, que funciona por el intercambio de los valores de los genes en todo el cromosoma, basándose en los que conforman un segmento especifico de él. Otro es el *HeuristicCrossover*, en donde se trata la distancia entre nodos representados en el cromosoma. El último operador propuesto se denomina *Merge Crossover*, el cual opera en base a una precedencia de tiempo predefinida. Esta a menudo es obtenida según las ventanas de tiempo impuestas por cada nodo. Existen dos variantes definidas: *MergeCrossover1* y *MergeCrossover2*, sobre los dos últimos mecanismos, se genera un sólo hijo desde dos padres, por ello el autor se enfoca en realizar mezclas de los mecanismos para obtener más individuos en la recombinación. Para la mutación, utiliza otros métodos convencionales.

En el trabajo de Zhu los experimentos se realizaron con distintas combinaciones de operadores de recombinación, usando 56 instancias de Solomon [11] con 100 nodos. La probabilidad de recombinación se definió como 0.77 y la de mutación como 0.06 mínimo. Se lograron resultados comparativos interesantes respecto a otras técnicas utilizadas

5.2 Thangiah S. R.

En el trabajo de Thangiah [12] se describe *GIDEON*, una heurística basada en GAs para resolver VRPTW. Este mecanismo usa una estrategia del tipo *cluster first–route second*, la cual consiste en primero hacer una asignación de usuarios a los vehículos y luego realizar un refinamiento de la mejor solución obtenida a través de una post-optimización.

Para la implementación del sistema, se utilizó un software para GAs llamado *GENESIS*, en el cual los individuos son representados como cadena de bits.

Los clusters se generan mediante la ubicación de *K* “puntos semilla” en el plano. Desde el depósito se trazan semirrectas hacia cada punto semilla, definiendo sectores que particionan al conjunto de clientes en clusters. El algoritmo se utiliza para determinar la mejor ubicación de los puntos semilla. La ubicación de un punto semilla se codifica, utilizando 5 bits. Un vector de 5K bits representa la ubicación de los K puntos semilla.

Para la etapa de post-optimización se intercambian los clientes entre las rutas obtenidas con el fin de mejorar la solución. Se utiliza un método de optimización local *λ-interchange*, el que consiste básicamente en intercambiar un conjunto de clientes de tamaño máximo λ entre rutas de la solución original para generar nuevas soluciones alternativas, lo que se realiza en un proceso iterativo hasta no encontrar mejoras en las soluciones. Para el sistema aquí descrito, se utilizó $\lambda=2$.

Para calcular el fitness de un individuo se genera una ruta para cada uno de sus clusters utilizando un algoritmo simple de inserción para el problema del vendedor viajero. Se permite que las rutas no sean factibles. El fitness de un individuo es el costo de las rutas obtenidas para cada cluster, más términos que penalizan las violaciones a las restricciones de capacidad, largo máximo de cada ruta y de ventanas de tiempo.

El operador de cruzamiento utilizado en este trabajo es el DPX y como operador de mutación se modifican algunos bits aleatoriamente.

Para los experimentos, los valores de los parámetros correspondientes al tamaño de la población, probabilidad de recombinación y de mutación corresponden a 1000, 0.5 y 0.001 respectivamente. Se testearon un conjunto de 56 instancias del problema usando los datos de entrada desarrollados por Solomon [11], ampliamente conocidos en esta área. Los resultados publicados señalan que de ese número, 41 resultados fueron mejores que otras heurísticas desarrolladas anteriormente por Solomon.

5.3 Tan K. C.

El algoritmo propuesto por Tan K. C. [14] puede ser considerado un método “Asignar Primero - Rutear Después”.

La representación de las soluciones se hace de la siguiente forma: cada individuo codifica una agrupación de clientes mediante una permutación y un vector que indica cuántos clientes tiene cada cluster. Por ejemplo se tiene la siguiente solución:

Ruta 1: $0 \rightarrow 2 \rightarrow 4 \rightarrow 7 \rightarrow 0$
Ruta 2: $0 \rightarrow 1 \rightarrow 6 \rightarrow 0$
Ruta 3: $0 \rightarrow 5 \rightarrow 9 \rightarrow 3 \rightarrow 8 \rightarrow 0$

El cromosoma que representa esta solución sería de la forma:

[2, 4, 7, 1, 6, 5, 9, 3, 8] [3, 2, 4]

Para evaluar el valor fitness de un individuo se construye una ruta para cada cluster mediante una Heurística de Inserción Secuencial de Solomon [11].

Para la selección de los individuos se utiliza el mecanismo de selección por Torneo. En este trabajo se utiliza como operador de cruzamiento PMX y como operador de mutación se intercambian aleatoriamente dos elementos del primer vector.

5.4 Berger, Barkaoui y Bräysy

En esta propuesta [15] se utilizan dos poblaciones que evolucionan en paralelo y operan directamente sobre las soluciones.

En la población P1 el objetivo es minimizar el costo total y siempre se tiene al menos una solución factible, mientras que en la población P2 se intenta minimizar las violaciones de las restricciones de capacidad y de ventanas de tiempo.

Los individuos de una misma población tienen la misma cantidad de rutas; los de P1 tienen *Rmin* y los de P2 tienen *Rmin*−1, siendo *Rmin* la mínima cantidad de rutas para la que se ha conseguido una solución factible. Cuando se encuentra una nueva mejor solución en P2, se actualiza *Rmin* en ambas poblaciones.

Las dos poblaciones evolucionan de la misma manera, utilizando los mismos operadores y la misma función de fitness. La única diferencia radica en la cantidad de rutas impuestas a los individuos.

Se utiliza un modelo evolutivo de Estado Estacionario: una generación consiste en agregar *np* individuos a la población y luego eliminar los *np* peores individuos. Luego de que ambas poblaciones evolucionan una generación, si P2 contiene una solución factible, se copian los individuos de P2 en P1 y se aplica un operador de mutación llamado RSS_M a los individuos de P2 para reducir la cantidad de rutas en uno. Los operadores de cruzamiento y mutación son bastante complejos; para una descripción de los mismos referirse al artículo original [15].

Capítulo 6

6 Modelo propuesto

6.1 Estructura general del modelo

El modelo propuesto combina los elementos mencionados en los capítulos 3 y 4 sobre algoritmos genéticos y culturales.

La estructura general del modelo se muestra en el algoritmo 6 y en la figura 6.1. Se parte iniciando los espacios de la población y de creencias para luego entrar en el ciclo evolutivo, en donde a los operadores genéticos tradicionales se les complementa con influencia cultural para la evolución de la siguiente generación de individuos.

En las siguientes secciones del presente capítulo se explicarán detalladamente cada uno de los elementos utilizados para el desarrollo del modelo.

Algoritmo 6 Estructura del modelo Propuesto

```
t = 0
Inicializar espacio de Población P(t)
Inicializar espacio de Creencias C(t)
Evaluar P(t)
Repetir
   t = t + 1
   Padres = Selección( P(t-1) , C(t-1) )
   Hijos = Cruzamiento (Padres)
   Hijos = Mutación (Hijos)
   P(t) = Hijos
   Evaluar P(t)
   C(t) = Actualizar (P(t))
Mientras (no se cumpla condición de término)
```

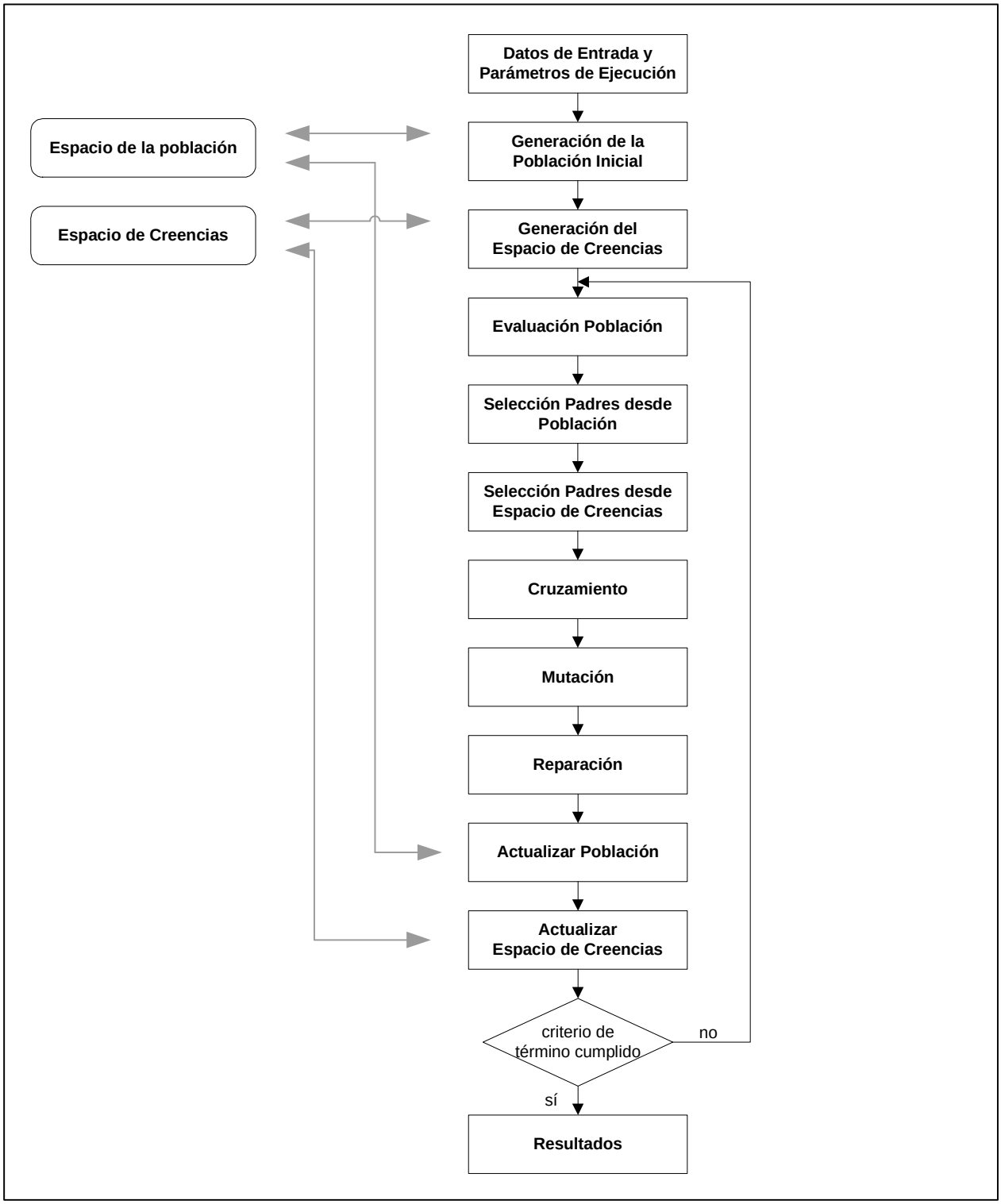


Figura 6.1 Estructura general del modelo propuesto.

6.2 Datos de entrada

Para la implementación del modelo propuesto se ha trabajado bajo el formato de los datos de benchmark propuestos por Solomon [11].

Los datos de entrada son archivos de texto que contienen la información requerida en el siguiente formato: las primeras 5 líneas del archivo corresponden a los datos sobre los vehículos, cantidad y capacidad de estos. A continuación cada fila representa un cliente (con el depósito como cliente cero), mientras que en cada columna se detalla los siguientes datos sobre los clientes: número de cliente, coordenada X, coordenada Y, demanda, tiempo mínimo de visita, tiempo máximo de visita y tiempo de servicio.

En la figura 6.2 se muestra la instancia de Solomon C101.

Archivo Edición Formato Ver Ayuda

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	144	90
25	25	52	40	169	224	90

Figura 6.2 Instancia de Solomon.

6.3 Representación de los individuos

Cada individuo representa una solución al problema, es decir, un conjunto de rutas para visitar a los clientes en un orden determinado.

Para la codificación de los individuos se utiliza un arreglo de pares ordenados con valores enteros, cuyo largo es la cantidad de clientes. En donde el primer miembro del par ordenado representa al cliente y el segundo el vehículo que le presta el servicio.

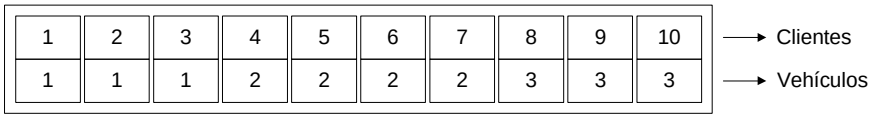


Figura 6.3 Representación del cromosoma de un individuo.

La decodificación del cromosoma considera que todos los clientes involucrados con un mismo vehículo pertenecen a la misma ruta, respetando el orden de aparición en el cromosoma como el orden de visita. Se considera que el primer cliente perteneciente a una ruta es visitado directamente desde el depósito y que el vehículo después de visitar al último cliente aparecido en una ruta regresa al depósito. En la figura 6.4 se ejemplifica las rutas y el orden de visitas que representa el cromosoma de la figura 6.3.

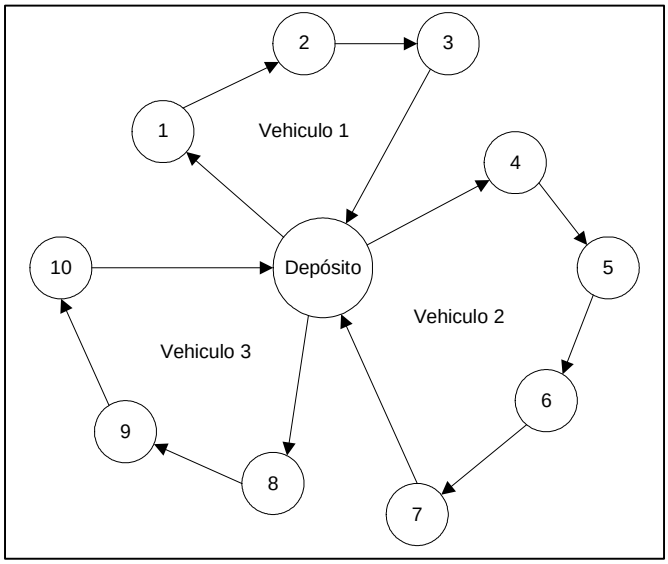


Figura 6.4 Decodificación de un cromosoma.

6.4 Función de evaluación

La función de evaluación calcula el costo de las rutas, en término de la distancia euclidiana entre los puntos visitados.

Como se explicó en la sección 2.2, el costo en que incurren las rutas está determinado por la suma de las distancias recorridas por cada uno de los vehículos existentes. En la ecuación 6.1 se detalla la función de evaluación con las variables explicadas anteriormente en la sección 2.2.

$$f = \sum_{i=1}^{m} \left(\left(\sum_{j=1}^{U(v_i)-1} C(F_i(j), F_i(j+1)) \right) + C(p_0, F_i(1)) + C(F_i(U(v_i)), p_0) \right) \qquad (6.1)$$

6.5 Población inicial

Se generará una población inicial de *n* individuos, cantidad designada por el usuario al momento del ingreso de parámetros. La generación de la población inicial se realiza mediante una heurística constructiva que trabaja formando las rutas necesarias, preocupándose que estas no violen las restricciones de capacidad y de ventanas de tiempo. De esta manera la población inicial con la que trabajará el algoritmo estará formada sólo por soluciones factibles.

- **Orden de precedencia de los clientes**

La heurística constructiva de los cromosomas pertenecientes a la población inicial, comienza formando de manera aleatoria una permutación de los clientes, asignándoles un orden de precedencia con la preocupación que no existan clientes repetidos. En la figura 6.5 se muestra un ejemplo de la permutación inicial para un problema de diez clientes.

Figura 6.5 Permutación aleatoria de clientes.

Una vez que se cuenta con una permutación inicial de los clientes, se forman las rutas ubicando a los clientes uno a uno según el orden establecido en la permutación.

- **Asignación de las Rutas**

Inicialmente se toma al primer cliente de la permutación y se asigna a la ruta 1.

Luego se toman uno a uno los clientes restantes según el orden establecido y se intentan insertar en las rutas existentes. Primero verificando si la inserción de un nuevo cliente en una ruta no viola la restricción de capacidad y luego verificando si se puede insertar el cliente en alguna posición dentro de la ruta que no viole la restricción de ventanas de tiempo. En caso que no sea posible insertar el cliente en ninguna ruta existente manteniendo la factibilidad, se crea una nueva ruta para el cliente en cuestión.

El siguiente ejemplo (ver figura 6.6) describe como se insertan los clientes a las rutas: El primer cliente (4) es ingresado inmediatamente a la ruta 1. El segundo cliente (8) cumple con la restricción de capacidad en la ruta 1 y ubicado en la segunda posición cumple con la restricción de ventanas de tiempo, por lo tanto es ubicado en la ruta 1. El tercer cliente (9) si se ingresara en la ruta 1 no cumpliría con la restricción de capacidad, por lo tanto se crea la ruta 2 para este cliente. El cuarto cliente (2) cumple con la restricción de capacidad en la ruta 1 y ubicado entre los clientes 4 y 8 cumple con la restricción de ventanas de tiempo, por lo tanto es ubicado en la ruta 1.

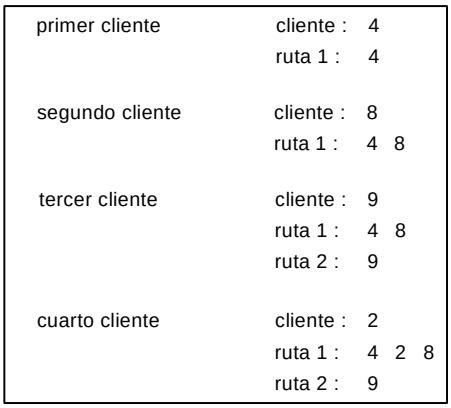


Figura 6.6 Ejemplo de asignación de rutas.

Finalmente se debe construir el cromosoma guardando la secuencia y ruta de cada cliente en el formato de la representación utilizada, en el ejemplo de la figura 6.7 se muestra el cromosoma final.

4	2	8	9	7	5	3	10	1	6
1	1	1	2	2	2	2	3	3	3

Figura 6.7 Asignación de las rutas a un cromosoma.

6.6 Selección

El mecanismo de selección utilizado es el del torneo probabilístico (ver sección 3.4). Como ya se ha mencionado anteriormente se busca minimizar el costo en que incurren las rutas al satisfacer la demanda de los clientes, por lo tanto el ganador del torneo será aquel cromosoma que tenga un menor valor fitness. No obstante, el cromosoma perdedor puede ser seleccionado, según una probabilidad dada.

En el algoritmo 7 se describe el procedimiento utilizado por medio de selección del torneo. Como en este caso se compite de a dos cromosomas, el resultado final será de *n/2* seleccionados.

Algoritmo 7 Selección por Torneo

Se barajan los individuos de la población

Repetir

Se escogen 2 individuos

Se compara su valor fitness

Se genera un valor aleatorio (entre 0 y 100)

Si (aleatorio <= probabilidad de torneo)

El individuo con menor valor fitness gana el torneo (es seleccionado)

Sino

El individuo con mayor valor fitness gana el torneo (es seleccionado)

El individuo que ha perdido el torneo ya no se considera para la selección

Mientras queden individuos sin competir

6.7 Operadores genéticos

6.7.1 Cruzamiento

Se cuenta con tres operadores de cruzamiento diseñados para permutaciones, los cuales trabajan contemplando que la descendencia no debe contener genes clientes repetidos.

Los operadores implementados para su posterior análisis son: PMXCrossover, OrderCrossover, HeuristicCrossover, los cuales serán explicados a continuación.

- **PMXCrossover**

El cruzamiento realizado mediante el operador PMXCrossover, genera un hijo a partir de dos padres de la siguiente manera:

Primero escoge aleatoriamente dos puntos de corte y se los intersecta en ambos padres.

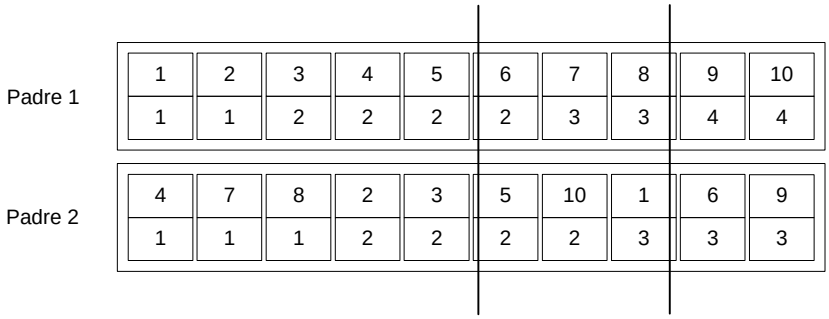


Figura 6.8 PMXCrosover: intersección de cortes.

Se forma el Hijo, clonando el segundo padre y luego intercambiando los genes cliente del primer padre, que se encuentren dentro del corte.

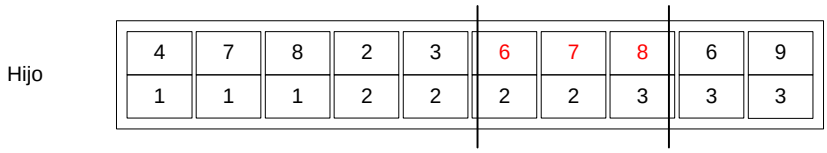


Figura 6.9 PMXCrosover: intercambio de genes de los padres al hijo.

Por último los genes cliente repetidos se intercambian por el valor que tenían antes de ser remplazado, en el ejemplo de la Figura 6.10 el cliente 5 fue reemplazado por 6, luego en la posición que estaba 6 antes del intercambio se ubica 5. Análogamente se sigue el mismo procedimiento con los otros genes repetidos.

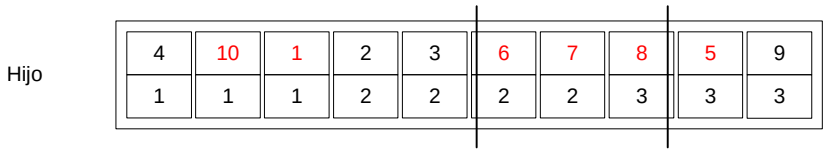


Figura 6.10 PMXCrosover: intercambio de genes.

Nótese que para el intercambio de genes sólo se consideran los genes cliente, mientras los genes vehículos permaneces en sus respectivas ubicaciones.

- **OrderCrossover**

El operador de cruzamiento OrderCrossover introducido en 1985 [16], genera un hijo a partir de dos padres de la siguiente manera:

Primero se escoge aleatoriamente dos puntos de corte y se los intersecta en ambos padres.

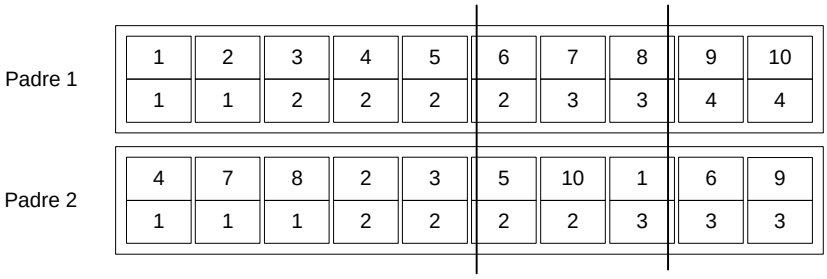


Figura 6.11 OrderCrosover: intersección de cortes.

Se forma el hijo, clonando desde el primer padre al hijo, los genes cliente que se encuentren dentro del corte. Los genes vehículo no se consideran hasta la etapa final del operador.

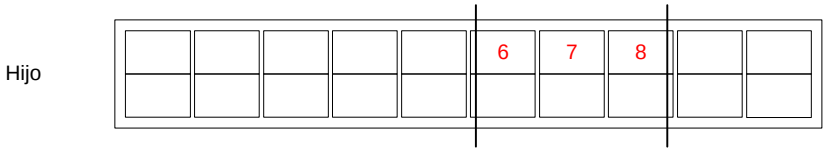


Figura 6.12 OrderCrossover: intercambio de genes del primer padre al hijo.

El siguiente paso es eliminar en el segundo padre los genes anteriormente clonados en el hijo.

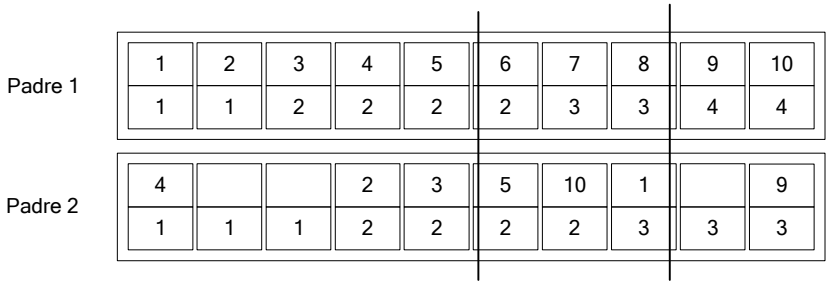


Figura 6.13 OrderCrossover: eliminación de genes repetidos.

Una vez que el segundo padre no contiene genes repetidos con el hijo, se procede a ingresar sus genes restantes en orden de izquierda a derecha en los espacios vacíos del hijo.

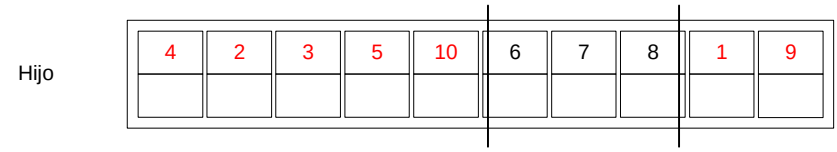


Figura 6.14 OrderCrossover: inserción de los genes restantes.

Finalmente, una vez que se tiene el hijo con todos sus genes clientes generados, se procede a la asignación de rutas, utilizando el algoritmo presentado en el punto 6.7.3.

- **HeuristicCrossover**

El operador de cruzamiento HeuristicCrossover genera un hijo a partir de dos padres, tomando como consideración la distancia entre los nodos, es decir, entre dos genes candidatos para ser el siguiente gen de la descendencia, se elegirá aquél cuya distancia con el último nodo hijo sea menor. HeuristicCrossover funciona de la siguiente manera:

Primero escoge aleatoriamente un punto de corte y se intersecta en ambos padres

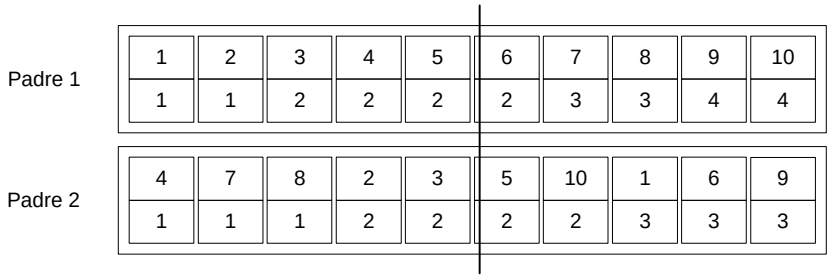


Figura 6.15 HeuristicCrosover: intersección del corte.

A continuación se selecciona el gen cliente que se encuentre inmediatamente después del corte, en cualquiera de los padres, y se inserta como el primer gen del hijo. Luego el gen del padre no utilizado es intercambiado por aquel seleccionado, para no volverlo a utilizar.

En el ejemplo de la figura 6.16 se selecciona el gen 6 proveniente del padre 1 y se agrega como primer gen del hijo. Luego en el padre 2 los genes 5 y 6 son intercambiados. De esta manera el gen 6, ya considerado en el hijo no volverá a aparecer como candidato.

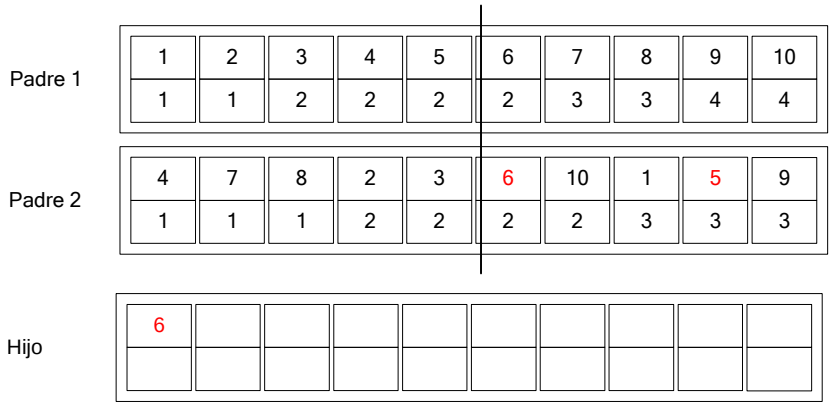


Figura 6.16 HeuristicCrosover: Selección del primer gen del hijo.

Una vez que el hijo cuenta con el primer gen, se debe elegir al siguiente para agregarlo al hijo. Para esto se considera como candidato al gen vecino del último seleccionado en ambos padres, y de entre estos dos se elige a aquel que posea la menor distancia entre ellos y el último gen del hijo.

Para una mejor comprensión del ejemplo de HeuristicCrossover se considerará que si se tiene tres clientes *x, y, z* en donde si x < *y* < *z* las distancias entre *x-y* < *x-z.*

Continuando con el ejemplo se requiere seleccionar al segundo gen del hijo, que ya posee al cliente 6. Para ello se toman como candidatos los clientes 7 y 10 provenientes de los padres 1 y 2 respectivamente, como estamos considerando que la distancia entre 6 y 7 es menor que la distancia entre 6 y 10, se selecciona a 7 como siguiente gen del hijo, y en el padre 2 se reemplaza el gen 10 por el 7, quedando los cromosomas como se observa en la figura 6.17

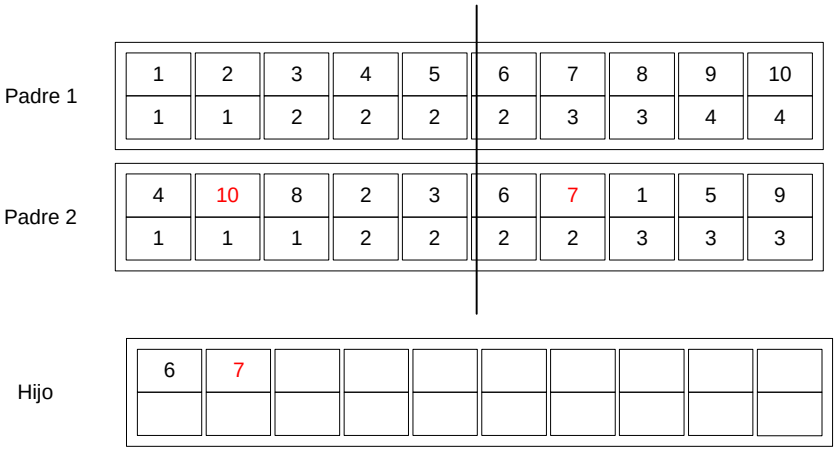


Figura 6.17 HeuristicCrosover: Selección del segundo gen del hijo.

Se continúa el mismo procedimiento hasta completar todos los genes del hijo. Finalmente, se procede a la asignación de rutas, utilizando el algoritmo presentado en el punto 6.7.3.

Considerando el ejemplo expuesto, el estado final de los cromosomas padres e hijos después de ser operados por HeuristicCrossover es el que se muestra en la figura 6.18.

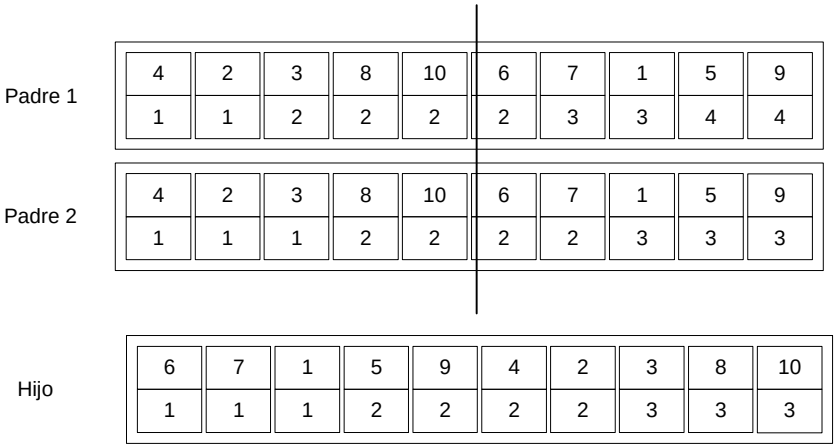


Figura 6.18 HeuristicCrosover: Estado final de los cromosomas.

6.7.2 Mutación

El operador de mutación es aplicado, según una probabilidad dada, a cada hijo después del cruzamiento.

Se han implementado dos variantes en las que se mutan tanto los genes que representan los clientes visitados cambiando el orden de servicio, y los genes que representan los vehículos que prestan el servicio, modificando las rutas y cantidad de vehículos utilizados.

Para cada variante de mutación implementada se debe tener en cuenta que se está trabajando con permutaciones, es decir, que cada gen que representa a un cliente es único y no puede repetirse dentro de un cromosoma.

- **Mutación de los genes Cliente**

Se escogen aleatoriamente dos genes y luego se intercambian sus posiciones.

Este operador de mutación sólo modifica la ubicación de los clientes de una ruta a otra, sin modificar la cantidad de clientes visitados por cada vehículo.

En el ejemplo de la figura 6.19 se seleccionaron aleatoriamente los genes ubicados en la tercera y sexta posición del cromosoma y se intercambiaron de ubicación.

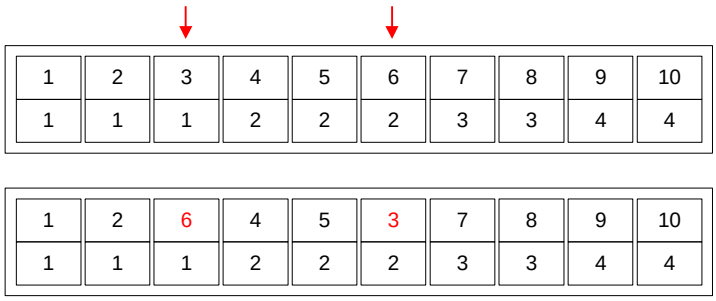


Figura 6.19 Ejemplo de mutación de genes cliente.

- **Mutación de modificación de Rutas**

Se escoge aleatoriamente uno o varios genes y se cambian los valores de los vehículos que los visitan por el valor siguiente del mayor vehículo existente en el cromosoma. Luego los genes modificados se desplazan al final del cromosoma para conservar el orden de las rutas.

Este operador de mutación incrementa el número de vehículos participantes, por lo cual antes de operarlo se debe verificar que no se viole el límite máximo de vehículos permitidos.

En el ejemplo mostrado en la figura 6.20 se seleccionaron aleatoriamente los genes ubicados en la tercera y séptima posición del cromosoma y sus valores de vehículos mutaron desde 1 y 3 a 5. De esta manera los vehículos 1 y 3 disminuyeron sus clientes y se formó una nueva ruta recorrida por el vehículo 5.

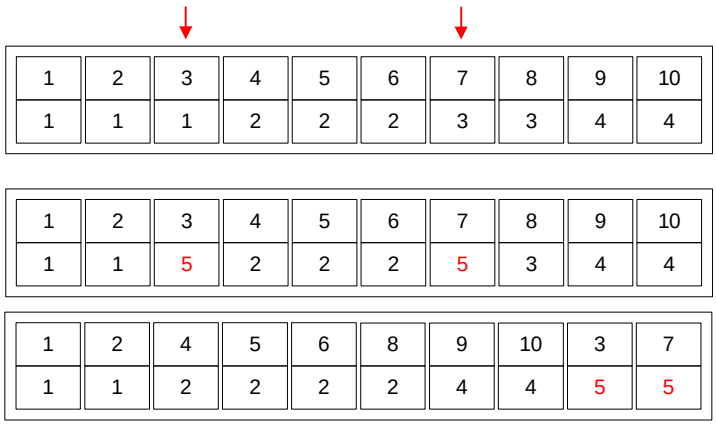


Figura 6.20 Ejemplo de mutación de modificación de rutas.

6.7.3 Reparación

Para enfrentar posibles infactibilidades que puedan aparecer en los cromosomas generados aplicando los operadores de cruzamiento y mutación, interviene un operador heurístico que tiene como fin el reparar las violaciones que pudiera haber con respecto a las capacidades y ventanas de tiempo.

El procedimiento de reparación comienza analizando si un cromosoma viola alguna de las restricciones de factibilidad. En caso de encontrar infactibilidad en cuanto a ventanas de tiempo se procede a reordenar los clientes de una misma ruta de forma ascendente en cuanto a sus ventanas de tiempo.

En el ejemplo de la figura 6.21 se muestran 10 clientes con sus rutas y ventanas de tiempo correspondientes, y el cromosoma que los representa. Si observamos la ruta del vehículo 1 podemos ver que este no puede visitar al cliente 1 antes del tiempo 30, lo que desencadena que no podrá llegar a tiempo a atender al cliente 2; ya que este tiene como límite máximo de visita el tiempo 25. De esta misma manera se puede observar situaciones similares en las rutas restantes.

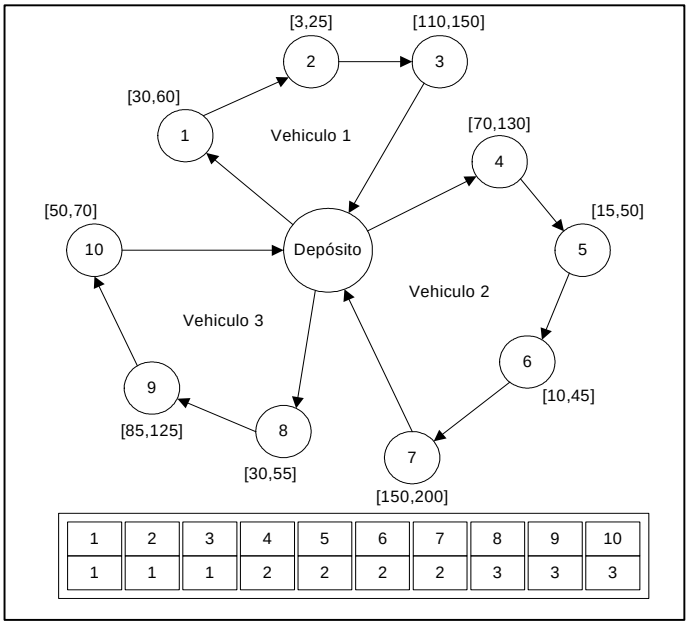


Figura 6.21 Ejemplo de rutas infactibles en relación a las ventanas de tiempo.

Para solucionar la situación descrita en el ejemplo anterior el operador de reparación de ventanas de tiempo ordena los genes cliente, comenzando por aquel que cuente con el promedio de ventana de tiempo menor (promedio entre el límite inferior y superior del intervalo) y continuando con el mismo criterio para cada una de las rutas.

En la figura 6.22 se muestra como quedan las rutas y el cromosoma que las representa del ejemplo anterior después de haber sido ejecutado el operador de reparación.

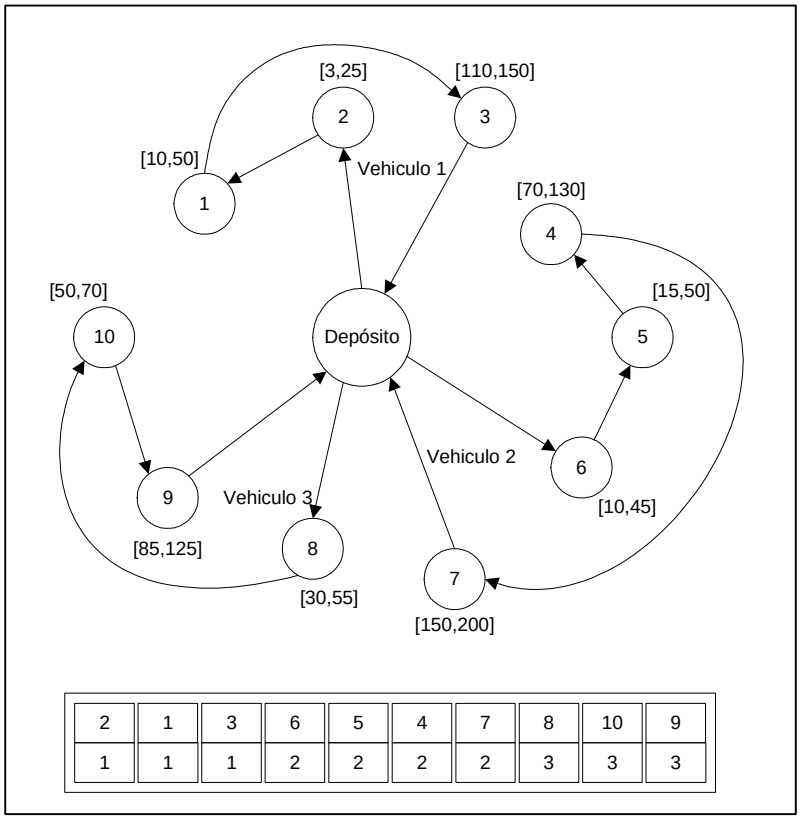


Figura 6.22 Reparación de violación de ventanas de tiempo.

Si aún persisten infactibilidades de ventanas de tiempo luego del reordenamiento de clientes, se eliminan de cada una de las rutas aquellos clientes que no cumplan con la restricción de ventanas temporales. Posteriormente, y si alguna ruta no respetara la restricción de capacidad, se elimina de aquellas rutas los clientes necesarios para dejar todas las rutas factibles.

Una vez que se tienen separados los elementos que provocaban la infactibilidad se intentan insertar entre las demás rutas sin provocar nuevas violaciones. En caso que no sea posible la reubicación de los clientes se les crea una ruta nueva.

En el ejemplo de la figura 6.23 se muestra un conjunto de rutas, en las cuales los clientes 2, 5 y 10 no cumplen con las restricciones. En la figura 6.24 se eliminan de las rutas todos los clientes que provocan la infactibilidad. En la figura 6.25 se muestra que el cliente 2 fue reubicado en la ruta 3, mientras que a los clientes 5 y 10 que no pudieron ser reubicados en ninguna posición se les creó una ruta nueva.

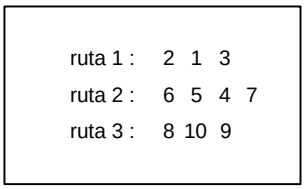


Figura 6.23 Conjunto de rutas infactibles.

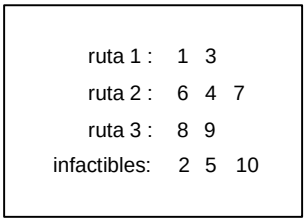


Figura 6.24 Separación de clientes infactibles.

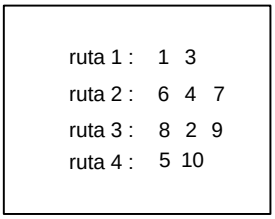


Figura 6.25 Reparación de las rutas.

6.8 Influencia cultural

Para el desarrollo del siguiente modelo se han considerado las características de los algoritmos culturales explicadas en el capítulo 4.

Para la implementación se creó un espacio de creencias que permite almacenar el conocimiento que han adquirido los individuos de generaciones anteriores y dejarlo a disposición del resto de la población para que cualquier individuo pueda utilizarlo para modificar su comportamiento.

La interacción de este espacio de conocimiento con el espacio de la población se realiza mediante dos funciones. Una de ellas es la función de aceptación que extrae las experiencias que han obtenido los individuos de una generación y las almacena en el espacio de creencias. La otra función es de influencia que utilizando los operadores de variación de los individuos (cruzamiento y mutación), presiona a la población para que los individuos resultantes se acerquen a comportamientos deseables, según la información almacenada en el espacio de creencia.

6.8.1 Espacios de creencias

El espacio de creencias utilizado en el modelo es del tipo circunstancial explicado en la sección 4.2, en este espacio de creencia se almacenará un conjunto de buenas soluciones, llamadas líderes.

Entre todas las soluciones que cuenten con la misma cantidad de rutas se selecciona aquella con menor valor fitness como líder, es decir, la mejor solución que utilice n rutas será considerada como líder(n).

Los elementos del conjunto de líderes serán como mínimo uno, en el caso que todas las soluciones cuenten con la misma cantidad de vehículos, y como máximo el límite de vehículos permitidos por el problema en los datos de entrada (explicado en el capítulo 7).

En consecuencia, a medida que aparezcan soluciones con distintos números de rutas, se irán añadiendo nuevos líderes al espacio de creencias, aportando una mayor diversidad a este.

6.8.2 Función de aceptación

La función de aceptación tiene la tarea de seleccionar a aquellos individuos que cumplan con el criterio que determina si merecen ser parte del espacio de creencias.

La inicialización del espacio de creencia se lleva a cabo una vez generada la población inicial. Se agrupan los individuos según la cantidad de vehículos utilizados y para cada uno de estos grupos de soluciones se selecciona la que cuente con menor valor fitness como un líder.

Los individuos líderes, son actualizados cada vez que la población evoluciona a una nueva generación, en donde se debe verificar para cada uno de los líderes si dentro de la nueva población existe algún individuo que utilice la misma cantidad de vehículos y que tenga un valor fitness menor, en tal caso se realizará el reemplazo correspondiente. Además se debe verificar si dentro de la nueva población existen soluciones con otras cantidades de vehículos, en este caso se debe añadir un nuevo líder a espacio de creencias.

6.8.3 Función de influencia

La influencia del espacio de creencias en la población se realiza mediante los operadores de cruzamiento y mutación.

- **Influencia de los Espacios de Creencias en el operador de Cruzamiento**

La influencia inicialmente se presenta al momento de la selección de los padres, pero actúa en el operador de cruzamiento, como se describió en la sección 6.6. Para la selección se utiliza el mecanismo del torneo.

Para que la cultura pueda influir sobre la población, los padres de las nuevas generaciones se cruzarán, según una probabilidad dada, con los individuos almacenados en los espacios de creencias. El objetivo de realizar la recombinación con los individuos líderes es aumentar la probabilidad que después de un cruce el hijo sea mejor que su padre. El motivo que se cuente con más de un individuo líder es aportar un mayor grado de diversidad y permitir realizar búsqueda de soluciones sobre otros espacios que no hayan sido considerados.

- **Influencia de los Espacios de Creencias en el operador de Mutación**

La influencia que se realiza en el operador de mutación consiste en acercar al individuo mutado hacia uno de los individuos líderes que se encuentran almacenados en el espacio de creencias.

Para lograr esto el algoritmo selecciona de forma aleatoria una ruta de algún individuo líder y la intenta colocar en el individuo a mutar. En la figura 6.26 se muestra como el individuo líder pretende influenciar la ruta del vehículo 1 sobre el individuo a mutar.

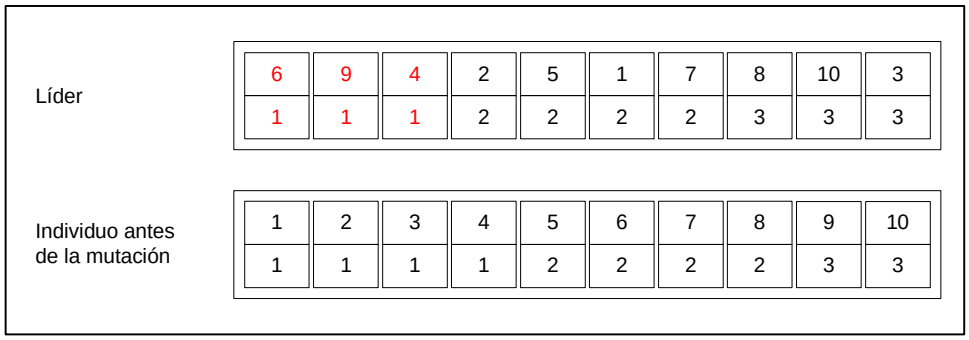


Figura 6.26 Influencia de un líder.

Una vez mutado el individuo con la ruta del líder, se reemplazan los genes clientes repetidos del cromosoma mutado, de manera análoga que la del operador PMXCrosover (ver sección 6.7.1). En la figura 6.27 se muestra la continuación del ejemplo anterior, en

donde la ruta 1 del líder fue clonada en el individuo mutado, luego de esto los genes cliente repetidos 6, 9 y 4, se remplazan.

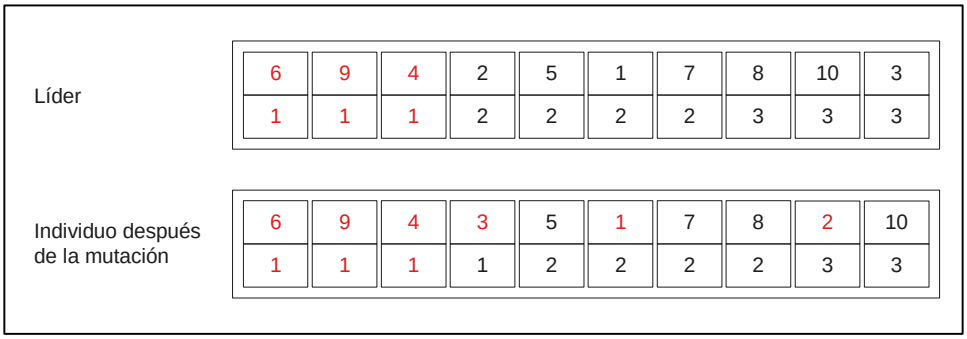


Figura 6.27 Influencia del espacio de creencias en la mutación.

Por último se reasignan las siguientes rutas del cromosoma, de manera que no se repitan los vehículos utilizados en la ruta influenciada por el líder, con las rutas restantes. En la figura 6.28 se muestra el resultado final del cromosoma del ejemplo anterior después de la reasignación de rutas.

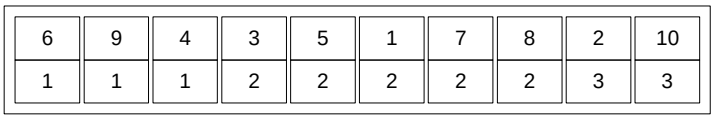


Figura 6.28 Reasignación de rutas al individuo mutado.

Capítulo 7

7 Plan de pruebas

7.1 Datos de Benchmark

Para Realizar las distintas pruebas y calibración de parámetros se utiliza como datos de benchmark las instancias de Solomon [11], ampliamente utilizadas por diversas investigaciones sobre el problema de enrutamiento de vehículos con ventanas de tiempo.

Las instancias de Solomon son 56 archivos de textos que contienen la siguiente información: cantidad de vehículos máximos que se pueden utilizar, capacidad de cada vehículo, y cantidad de clientes. Para cada cliente se indica su posición espacial en términos de coordenadas *x* e *y*, demanda requerida, tiempo mínimo y máximo en que puede ser visitado y el tiempo que demora en ser servido. El formato de estas instancias se detalla en la sección 6.2.

Las instancias de Solomon están agrupadas en 6 diferentes tipos de problemas nombrados *C1, C2, R1, R2, RC1, RC2*. Cada set de datos contiene entre ocho a doce problemas de cien clientes cada uno. Los nombres de los tipos de problemas tienen el siguiente significado. Los problemas del tipo *C* tienen agrupados clientes cuyas ventanas de tiempo fueron generadas en base a soluciones conocidas. Los problemas del tipo *R* tienen clientes cuya localización fue generada de manera aleatoria. Los problemas *RC* cuentan con una combinación de ubicación aleatoria y clientes seleccionados. Los problemas del tipo 1 tienen ventanas de tiempo estrechas y vehículos con pequeña capacidad. Mientras que los problemas del tipo 2 tienen ventanas de tiempo más largas y vehículos con mayor capacidad.

Para el desarrollo de las pruebas del presente proyecto se utilizará una instancia de cada tipo de problema. Las instancias utilizadas se describen en la tabla 7.1.

Tabla 7.1 Instancias utilizadas en el plan de pruebas.

Problema	Cantidad de clientes	Número máximo de Vehículos	Capacidad de los vehículos
C101	100	25	200
C201	100	25	700
R101	100	25	200
R201	100	25	1000
RC101	100	25	200
RC201	100	25	1000

7.2 Comparación de operadores

Dentro de la variedad de operadores disponibles para la implementación, se deben analizar el comportamiento de modelos que utilicen distintas combinaciones de estos operadores.

- PMXCrossover.
- OrderCrossover.
- HeuristicCrossover.

7.3 Comparación de la influencia cultural

Se debe analizar el comportamiento de los modelos sin la influencia cultural, es decir, sólo con los con los operadores y funciones del modelo genético y el modelo que además incluye la influencias del espacio de creencias.

7.4 Calibración de parámetros.

Dentro del funcionamiento del algoritmo existe una diversidad de parámetros que afectan a los distintos procedimientos y operadores. Estos parámetros normalmente interactúan entre sí de forma no lineal, por lo que no pueden optimizarse de manera independiente.

La calibración de estos parámetros es una tarea de alta importancia, dado que influye directamente con el rendimiento final que se desea obtener. La forma adecuada de definir los parámetros de un algoritmo genético ha sido motivo de investigación desde los orígenes mismos de la técnica, y no existe hasta la fecha una solución satisfactoria a este problema.

Considerando la cantidad de operadores y la gran cantidad de posibilidades de combinación de parámetros existentes, se ha implementado un programa que automatiza la ejecución del algoritmo con distintas configuraciones y parámetros, entregando los resultados estadísticos de cada ejecución.

Dentro de la variedad de parámetros que deben ser contemplados y analizados se pueden mencionar los siguientes:

- Número de individuos de la población inicial
- Probabilidad de selección en el mecanismo del torneo.
- Probabilidad de mutación aleatoria.
- Probabilidad de cruzamiento con los individuos líderes del espacio de creencias.
- Probabilidad de mutación de influencia cultural.

Capítulo 8

8 Resultados

En este capítulo se presentan los resultados obtenidos al aplicar los modelos implementados al problema de enrutamiento de vehículos con ventanas de tiempo.

8.1 Comparación de operadores de cruzamiento.

A continuación se describen los resultados obtenidos para la ejecución del algoritmo utilizando únicamente conceptos de algoritmos genéticos.

En la siguiente tabla se presentan los mejores resultados obtenidos para las distintas alternativas de operadores de cruzamiento, ejecutados con diferentes valores de probabilidad de selección y mutación, utilizando como problema de referencia la instancia C101 la cual consta de 100 clientes y capacidad 200.

Tabla 8.1 comparación de operadores de cruzamiento.

Operador de Cruzamiento	Probabilidad de Selección	Probabilidad de Mutación	Resultado
PMX Crossover	75%	7.5%	1991
OrderCrossover	70%	7.5%	2202
HeuristicCrossover	70%	5%	1043

De la comparación de resultados se puede observar que el operador de cruzamiento HeuristicCrossover es quien entrega los mejores valores, por este motivo este operador se seguirá utilizando para las siguientes etapas de ejecución del algoritmo.

En el Anexo A1 se describen detalladamente los resultados obtenidos para cada uno de los operadores de cruzamiento con distintos valores de probabilidad de selección y de mutación.

8.2 Modelo genético

A continuación se mostrarán los mejores resultados obtenidos para los seis tipos de problemas de las instancias de Solomon aplicados al modelo de algoritmo genético utilizando el operador HeuristicCrossover.

En la Tabla 8.2 se presentan los mejores resultados obtenidos para cada uno de los seis problemas de Solomon utilizados, detallando la cantidad de población, probabilidad de selección y mutación utilizada.

Tabla 8.2 mejores resultados del modelo genético.

Problema	Población	Probabilidad de Selección	Probabilidad de Mutación	Resultado
C101	200	60%	2.5%	965
C201	100	60%	5%	616
R101	200	80%	5%	1774
R201	200	70%	2.5%	2008
RC101	200	80%	2.5%	1804
RC201	200	70%	5%	2222

En el Anexo A2 se detallan los resultados de la calibración de parámetros obtenidos utilizando distintas combinaciones de cantidad de población, probabilidad de selección y probabilidad de mutación. También se presenta gráficamente los resultados obtenidos tanto del promedio de fitness de la población, como de la mejor solución obtenida en cada generación.

8.3 Modelo cultural

A continuación se mostrarán los mejores resultados obtenidos para los seis tipos de problemas de las instancias de Solomon, aplicados al modelo de algoritmo genético con influencia cultural, utilizando el operador HeuristicCrossover.

En la Tabla 8.3 se presentan los mejores resultados obtenidos para cada uno de los seis problemas de Solomon utilizados, detallando la cantidad de población, probabilidad de selección, probabilidad de mutación y probabilidad de seleccionar al líder.

Tabla 8.3 mejores resultados del modelo cultural.

Problema	Población	Probabilidad de Selección	Probabilidad de Mutación	Probabilidad del Líder	Resultado
C101	200	75%	5%	30%	879
C201	100	70%	5%	30%	591
R101	200	70%	2.5%	15%	1684
R201	200	65%	5%	30%	1679
RC101	200	70%	2.5%	15%	1786
RC201	200	90%	5%	30%	1926

En el Anexo A3 se detallan los resultados de la calibración de parámetros obtenidos utilizando distintas combinaciones de cantidad de población, probabilidad de selección, probabilidad de mutación y probabilidad de seleccionar al líder. También se presenta gráficamente los resultados obtenidos tanto del promedio de fitness de la población, como de la mejor solución obtenida en cada generación.

8.4 Comparación de los resultados

A continuación se presenta la comparación del algoritmo genético con y sin influencia cultural. En la tabla 8.4 se presentan comparativamente los resultados obtenidos por ambos modelos. Posteriormente desde la figura 8.1 hasta la figura 8.6 se presenta de manera gráfica la comparación de los resultados obtenidos en cada generación.

Tabla 8.4 Comparación de los resultados obtenidos.

Problema	Mejor Resultado Conocido	Modelo sin Influencia Cultural	Modelo con Influencia Cultural
C101	828	965	887
C201	591	616	591
R101	1648	1774	1684
R201	1080	2008	1679
RC101	1252	1804	1786
RC201	1642	2222	1916

Problema C101

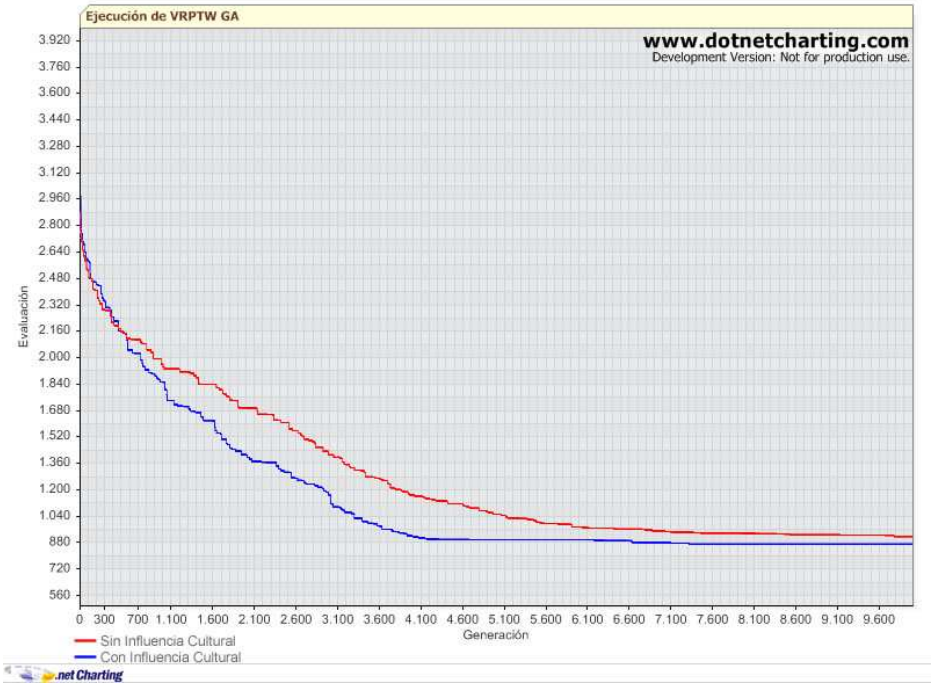


Figura 8.1 Comparación influencia cultural para el problema C101.

Problema C201

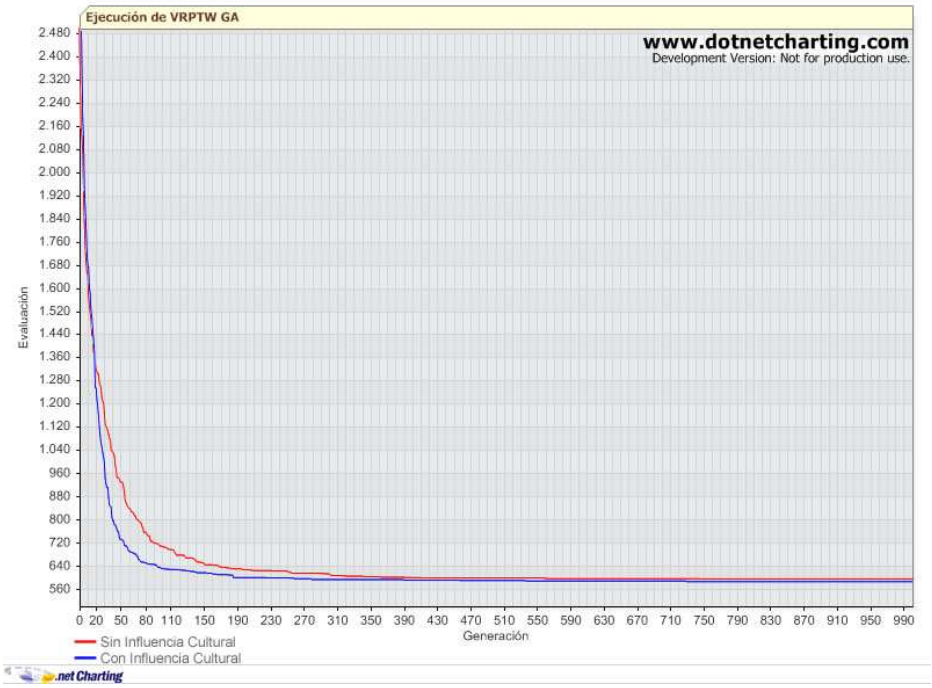


Figura 8.2 Comparación influencia cultural para el problema C201.

Problema R101

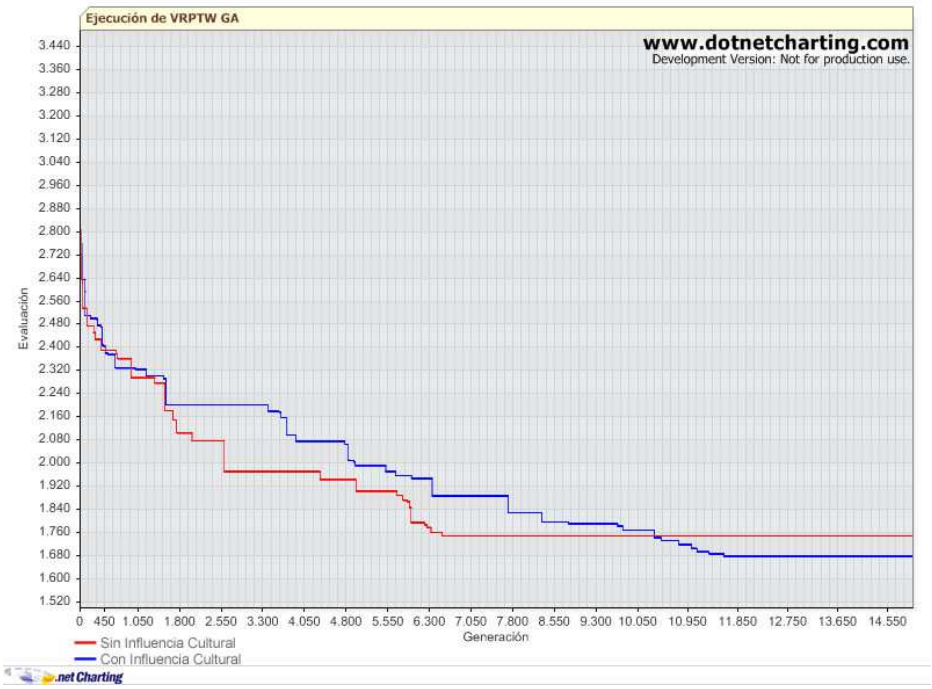


Figura 8.3 Comparación influencia cultural para el problema R101.

Problema R201

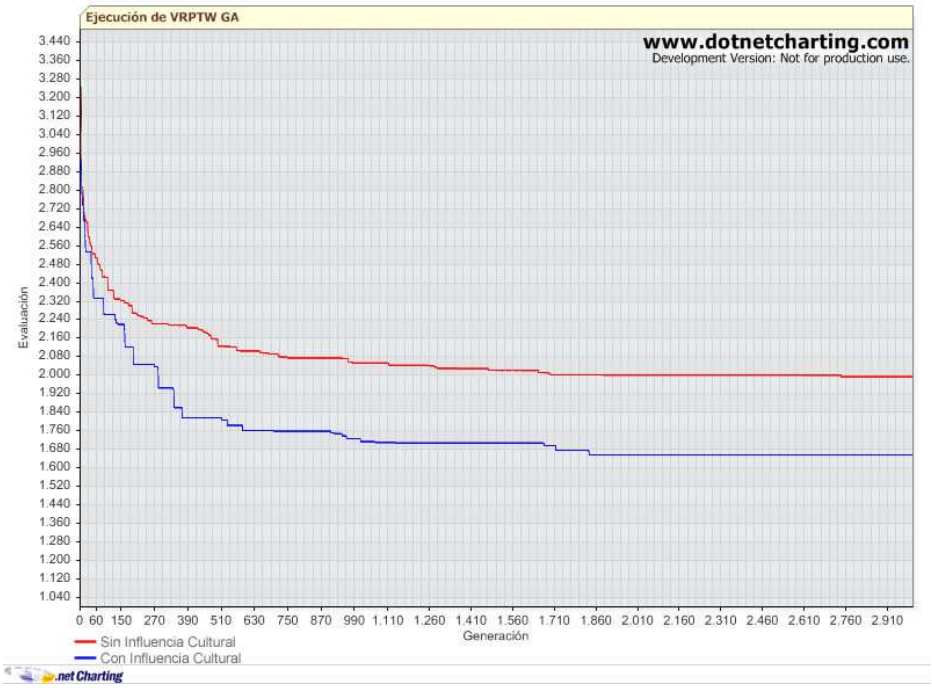


Figura 8.4 Comparación influencia cultural para el problema R201.

Problema RC101

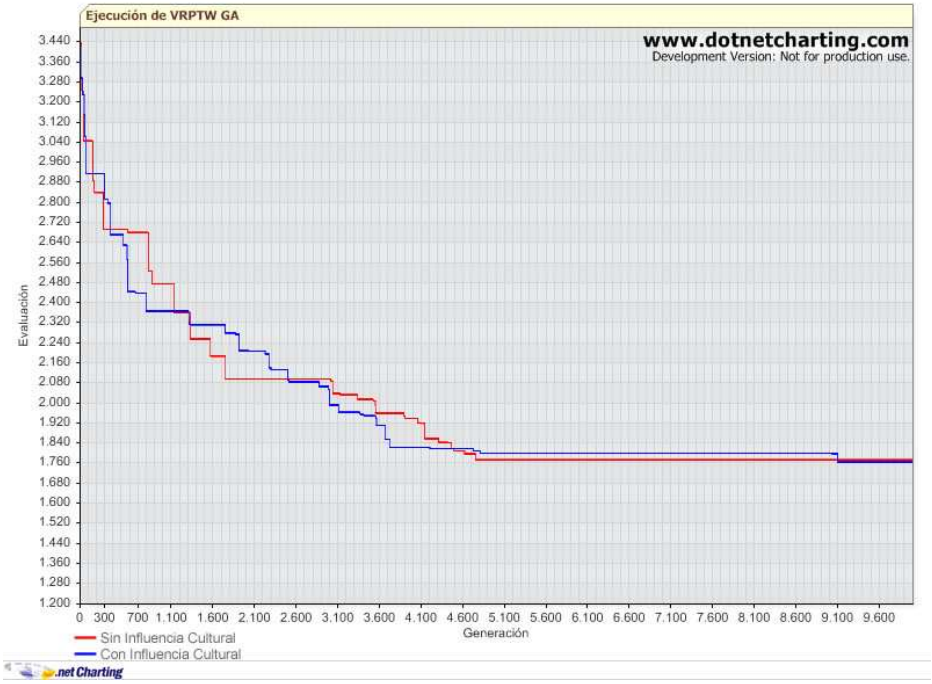


Figura 8.5 Comparación influencia cultural para el problema RC101.

Problema RC201

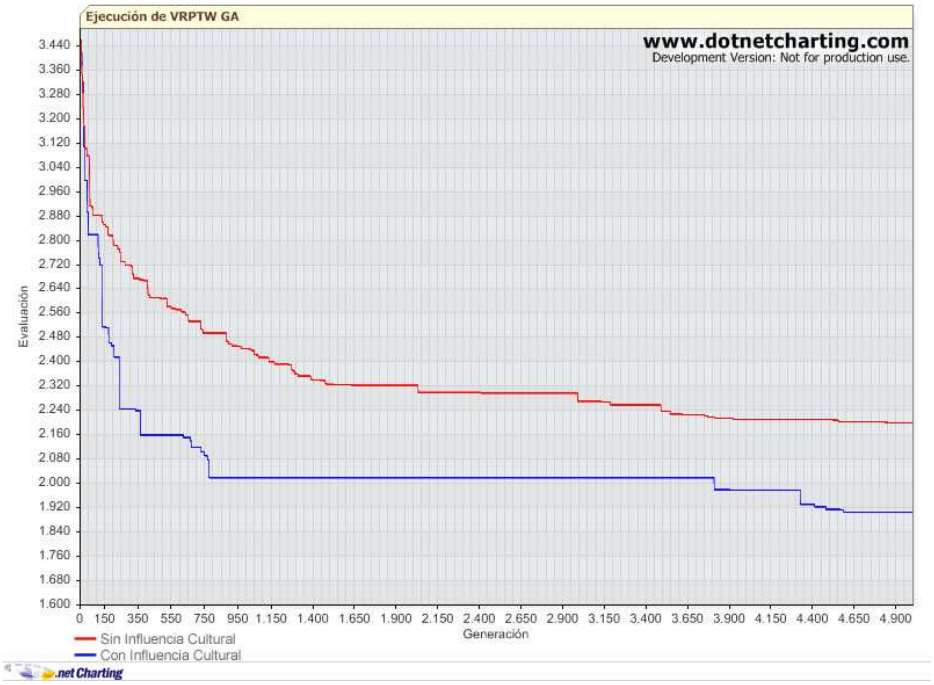


Figura 8.6 Comparación influencia cultural para el problema RC201.

Capítulo 9

9 Conclusiones

9.1 Referentes al proyecto

El problema de enrutamiento de vehículos desde su introducción ha adquirido gran importancia tanto por su aplicabilidad como por la complejidad de resolución, motivo por el cual ha sido extensamente estudiado y experimentado por diversas técnicas de solución, las cuales han intentado entregar buenos resultados con un costo razonable de recursos.

Una de las técnicas que ha demostrado gran eficiencia para este tipo de problemas, en especial para el VRPTW que es la variante abordada en el presente proyecto, es la de los algoritmos genéticos. Sin embargo, esta eficiencia depende del grado de intensidad y diversidad, los cuales permiten explorar de una manera adecuada el campo de soluciones y a la vez explorar determinadas zonas específicas de forma exitosa, dejando abierto el camino al desarrollo de avances que permitan mejorar su funcionamiento.

Por otra parte, los algoritmos culturales, por lo reciente de su propuesta, han sido poco estudiados y desarrollados para VRPTW en comparación con los algoritmos genéticos. Sin embargo se pueden encontrar varios algoritmos bastantes competitivos para problemas mono-objetivos con restricciones.

En cuanto al trabajo realizado, se presentó lo correspondiente al desarrollo total del proyecto, esto es, se revisaron las referencias y bibliografías enfocándose en comprender el problema VRPTW, sus características, variantes, restricciones y técnicas utilizadas para su solución. Se estudiaron los aspectos teóricos fundamentales de los algoritmos genéticos y se abordaron las características fundamentales de los algoritmos culturales. Se construyó un programa de experimentación en lenguaje C Sharp, el cual permitió automatizar las pruebas necesarias para calibrar los parámetros y analizar el rendimiento de los distintos operadores y técnicas propuestas. Se diseñaron dos modelos de algoritmo genético, uno sin influencia cultural y otro con dicha influencia. Se realizaron las pruebas establecidas y se presentaron los resultados obtenidos.

9.2 Referente a los resultados obtenidos

En base a los resultados obtenidos (ver tabla 8.1), se puede concluir que de los tres operadores de cruzamiento implementados, el operador HeuristicCrossover es el que presentó el mejor rendimiento.

El modelo propuesto con influencia cultural presenta mejores resultados al compararlo con el modelo sin la influencia cultural. En las figuras: 8.1, 8.2, 8.3, 8.4, 8.5 y 8.6 que muestran los gráficos comparativos de los mejores resultados obtenidos para cada uno de los problemas de prueba, se observa que el modelo cultural obtiene menores valores fitness y mayor velocidad de convergencia que el modelo genético puro.

En la tabla 9.1 se detallan las mejores configuraciones para cada uno de los problemas de prueba utilizados, en base a la cantidad de población utilizada, probabilidad de selección de torneo, probabilidad de mutación y probabilidad de seleccionar un líder.

Tabla 9.1 Mejores configuraciones obtenidas.

Parámetro						
		Mejor Resultado	**Población**	**Selección**	**Mutación**	**Líder**
Problema	**C101**	879	200	75%	5%	30%
	C201	591	100	70%	5%	30%
	R101	1684	200	70%	2,5%	15%
	R201	1679	200	60%	5%	30%
	RC101	1786	200	70%	2,5%	15%
	RC201	1926	200	90%	5%	30%

9.3 Referentes al trabajo futuro

Como trabajo futuro para mejorar el rendimiento de los modelos desarrollados, se propone implementar las siguientes modificaciones:

- Propuesta de nuevos operadores genéticos que mejoren los resultados del modelo.
- Creación de un nuevo espacio de creencias que permita entregar un incremento en la diversidad de la población, almacenando él o los individuos más diversos según alguna métrica de diversidad propuesta.
- Implementación de los operadores necesarios que permitan al algoritmo variar los parámetros de la influencia cultural de manera adaptativa según vaya evolucionando la diversidad de la población.

Referencias

[1] Barajas N. “Estado del Arte del Problema de Ruteo de Vehículos”. Manuscrito preparado para Seminario de Investigación de la Maestría en Ingeniería de Sistemas y Computación de la Universidad Nacional de Colombia. 2006.

[2] Bräysy O. and Gendreau M.: “Tabu Search Heuristics for the Vehicle Routing Problem with Time Windows”. Internal Report STF42 A01022, SINTEF Applied Mathematics, Department of Optimisation, Oslo, Norway. 2001.

[3] Cook S.: “The P versus NP Problem”. Manuscript prepared for the Clay Mathematics Institute for the Millennium Prize Problems. April, 2000.

[4] Czech Z. and Czarnas P.: “Parallel simulated annealing for the vehicle routing problem with time Windows”. 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, Canary Islands - Spain, 376-383. January 9-11, 2002.

[5] Dantzig G. and Ramser R.: “The Truck Dispatching Problem”. Management Science 6: 80–91. 1959.

[6] Darwin C. Britannica concise encyclopedia from encyclopedia britannica, 2004. URL=http://concise.britannica.com/ebc/article?eu=387589.

[7] Gambardella L, Taillard E, Agazzi G.: “MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows”, In D. Corne, M. Dorigo and F. Glover, editors, New Ideas in Optimization. McGraw-Hill, 1999.

[8] Ladányi L., Ralphs T, and Trotter L.: “Branch, Cut, and Price: Sequential and Parallel, in Computational Combinatorial Optimization”. D. Naddef and M. Juenger, eds., Springer, Berlin . 2001.

[9] Reynolds R.: “An Introduction to Cultural Algorithms”. In A. V. Sebald and L. J. Fogel, editors, roceedings of the Third Annual Conference on Evolutionary Programming, pages 131–139.World Scientific, River Edge, New Jersey, 1994.

[10] Sóley A.: “Solving the Vehicle Routing Problem with Genetic Algorithms”, Abril 2004.

[11] Solomon, M.: “Algorithms for Vehicle Routing and Scheduling Problems with time window constraints”. Northeastern University, Boston, Massachusetts, (December, 1985).

[12] Thangiah S.: “Vehicle Routing with Time Windows using Genetic Algoriths”, Application Handbook of Genetic Algorithms: New Frontiers, Volume II. Lance Chambers (Ed.). CRC Press, 1995.

[13] Zhu K. Q.: “A New Genetic Algorithm for VRPTW”. IC-AI 2000, Las Vegas, USA.

[14] Kay Chen Tan, Loo Hay Lee, Ke Ou.: “Hybrid Genetic Algorithms in Solving Vehicle Routing Problems with Time Windows”. Asia-Pacific Journal of Operational Research 18 (2001) 121–130.

[15] Berger, J., Barkaoui, M., Bräysy, O.: “A route-directed hybrid genetic approach for the vehicle routing problem with time windows”. INFOR 41 (2003) 179–194

[16] Whitley D. and Wook Yoo N.: Modeling Simple Genetic Algorithms for Permutation Problems

Anexo

Calibración de parámetros

A.1 Operadores de cruzamiento

En el siguiente anexo se detallan los resultados de la calibración de parámetros obtenidos para los distintos operadores de cruzamiento utilizados.

En las siguientes tabla se presentan los mejores resultados obtenidos para las distintas alternativas de operadores de cruzamiento, ejecutados con diferentes valores de probabilidad de selección y mutación, utilizando como problema de referencia la instancia C101 la cual consta de 100 clientes y capacidad 200.

- **PMX Crossover**

Tabla A.1 Resultados del modelo genético, utilizando operador PMXCrossover.

		Probabilidad de mutación				
Probabilidad de selección		**0%**	**2,5%**	**5%**	**7,5%**	**10%**
	60%	2129	2138	2013	2161	2038
	70%	2165	2080	2092	2112	2031
	75%	2167	2152	2053	**1991**	2124
	80%	2121	2164	2099	2175	2064
	85%	2136	2168	2122	2170	2123
	90%	2240	2252	2126	2221	2160
	100%	2209	2167	2271	2256	2267

- **OrderCrossover**

Tabla A.2 Resultados del modelo genético, utilizando operador OrderCrossover.

		Probabilidad de mutación				
		0%	**2,5%**	**5%**	**7,5%**	**10%**
Probabilidad de selección	**60%**	2342	2391	2307	2278	2255
	70%	2259	2342	2248	**2202**	2259
	75%	2367	2336	2275	2323	2383
	80%	2381	2410	2297	2381	2308
	85%	2280	2389	2325	2297	2336
	90%	2445	2387	2289	2352	2378
	100%	2378	2506	2369	2422	2457

- **HeuristicCrossover**

Tabla A.3 Resultados del modelo genético, utilizando operador HeuristicCrossover.

		Probabilidad de mutación				
		0%	**2,5%**	**5%**	**7,5%**	**10%**
Probabilidad de selección	**60%**	1221	1113	1094	1104	1244
	70%	1114	1071	**1043**	1095	1160
	75%	1104	1053	1065	1169	1142
	80%	1128	1130	1087	1144	1140
	85%	1192	1172	1146	1190	1180
	90%	1187	1177	1159	1247	1244
	100%	1255	1167	1225	1198	1237

A.2 Modelo genético

En el siguiente anexo se detallan los resultados de la calibración de parámetros obtenidos para los distintos problemas de Solomon utilizando el modelo genético, ejecutando distintas combinaciones de cantidad de población, probabilidad de selección y probabilidad de mutación.

- **Problema C101**

En las tablas A4 y A5 se presentan los resultados obtenidos al ejecutar el problema C101 con el modelo genético, utilizando distintos valores de parámetros.

Tabla A4 Modelo genético, problema C101, población = 100.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	1113	1094	1104	1244
	70%	1071	1043	1095	1160
	75%	1053	1065	1169	1142
	80%	1130	1087	1144	1140
	85%	1172	1146	1190	1180
	90%	1177	1159	1247	1244
	100%	1167	1225	1198	1237

Tabla A5 Modelo genético, problema C101, población = 200.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	965	995	997	1003
	70%	971	969	1005	993
	75%	976	972	986	1015
	80%	971	992	978	1004
	85%	977	988	983	1008
	90%	984	1002	988	1018
	100%	993	993	982	1029

Para el problema C101 utilizando el modelo genético, el mejor resultado obtenido fue de 965, para el cual se utilizó 200 individuos de población, 60% de probabilidad de selección y 2,5% de probabilidad de mutación.

En la figura A.1 se muestran gráficamente los resultados obtenidos tanto del promedio de fitness de la población, como de la mejor solución obtenida en cada generación.

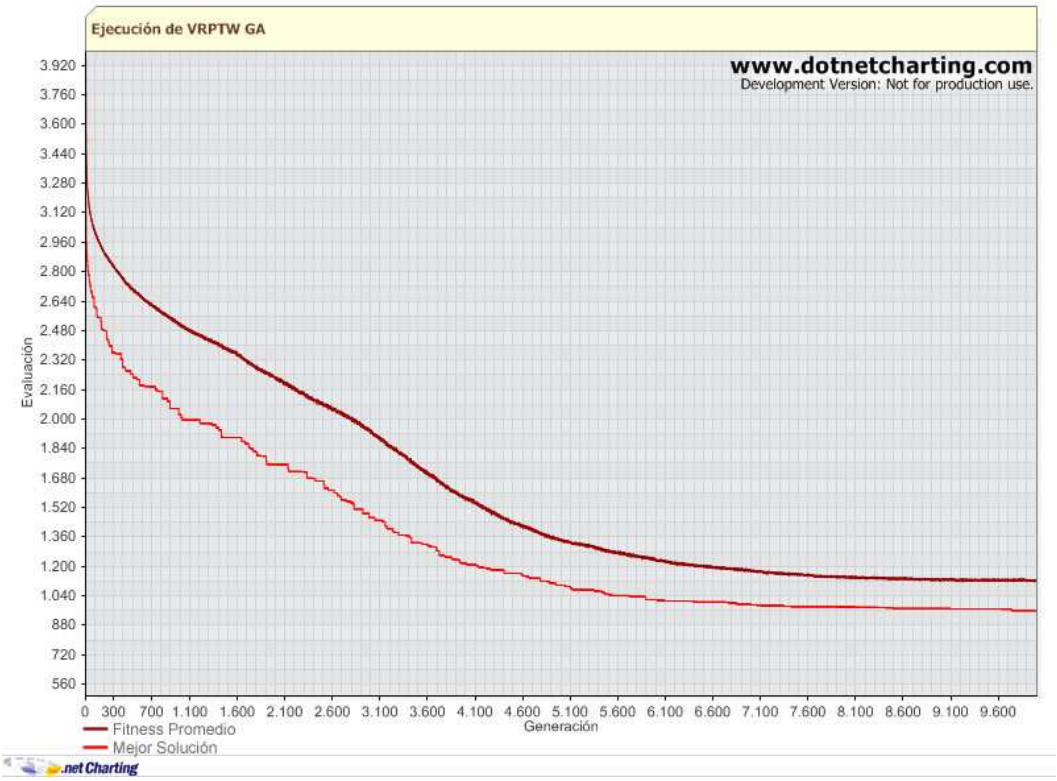


Figura A.1 Resultado algoritmo genético para el problema C101.

- **Problema C201**

En la tabla A6 se presentan los resultados obtenidos al ejecutar el problema C201 con el modelo genético, utilizando distintos valores de parámetros.

Tabla A6 Modelo genético, problema C201, población = 100.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	618	616	621	621
	70%	619	618	623	621
	75%	618	618	617	619
	80%	621	617	627	622
	85%	618	618	619	623
	90%	619	623	618	634
	100%	619	621	623	627

Para el problema C201 utilizando el modelo genético, el mejor resultado obtenido fue de 616, para el cual se utilizó 100 individuos de población, 60% de probabilidad de selección y 5% de probabilidad de mutación.

En la figura A.2 se muestran gráficamente los resultados obtenidos tanto del promedio de fitness de la población, como de la mejor solución obtenida en cada generación.

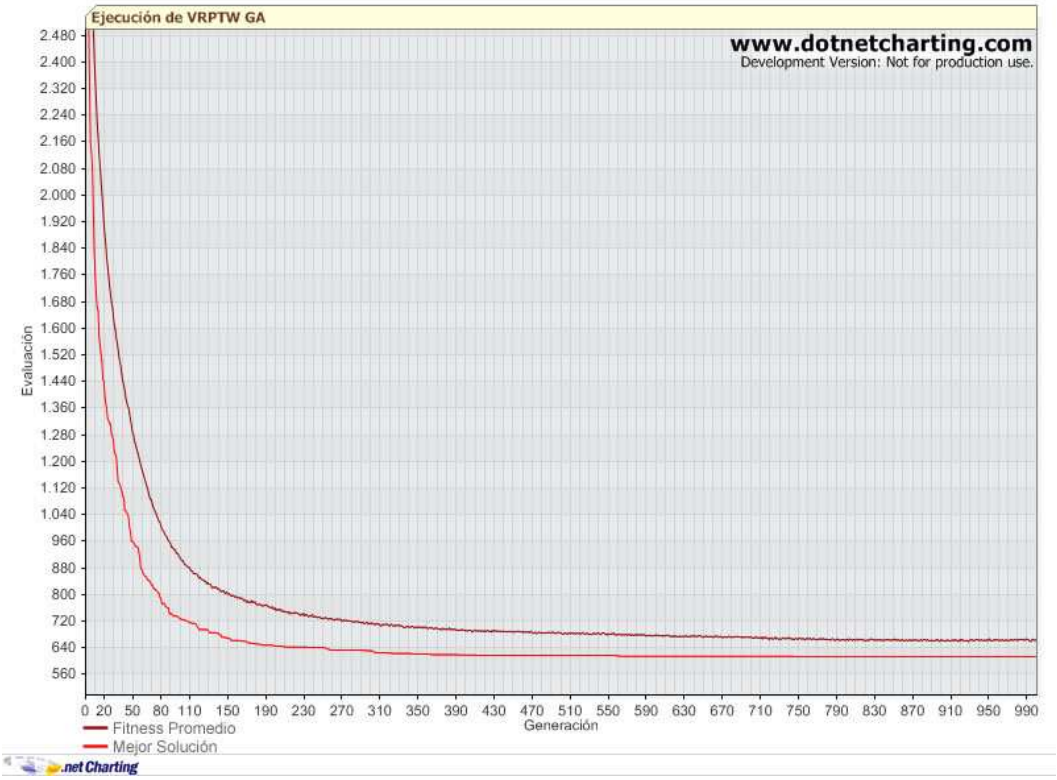


Figura A.2 Resultado algoritmo genético para el problema C201.

- **Problema R101**

En las tablas A7 y A8 se presentan los resultados obtenidos al ejecutar el problema R101 con el modelo genético, utilizando distintos valores de parámetros.

Tabla A7 Modelo genético, problema R101, población = 100.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	1986	1993	1972	1996
	70%	1978	1936	1969	1985
	75%	1979	1948	1967	1992
	80%	1967	1946	1976	1982
	85%	1978	1954	1985	1998
	90%	1987	1989	1996	2023
	100%	1995	2012	1992	2017

Tabla A8 Modelo genético, problema R101, población = 200.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	1973	1924	1893	1893
	70%	1933	1820	1898	1943
	75%	1894	1879	1824	1884
	80%	1870	1774	1798	1892
	85%	1902	1786	1805	1923
	90%	1855	1798	1943	1949
	100%	1929	1844	1982	1982

Para el problema R101 utilizando el modelo genético, el mejor resultado fue de 1774, para el cual se utilizó 200 individuos de población, 80% de probabilidad de selección y 5% de probabilidad de mutación.

En la figura A.3 se muestran gráficamente los resultados obtenidos tanto del promedio de fitness de la población, como de la mejor solución obtenida en cada generación.

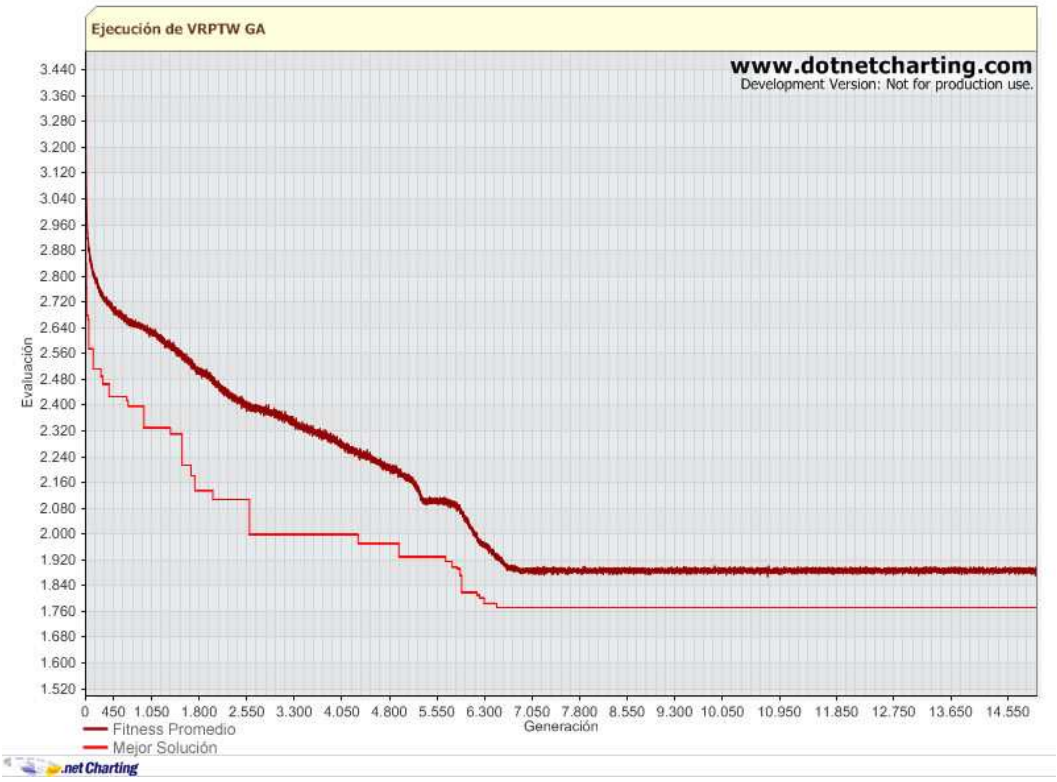


Figura A.3 Resultado algoritmo genético para el problema R101.

- **Problema R201**

En las tablas A9 y A10 se presentan los resultados obtenidos al ejecutar el problema R201 con el modelo genético, utilizando distintos valores de parámetros.

Tabla A9 Modelo genético, problema R201, población = 100.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	2198	2156	2224	2198
	70%	2154	2114	2185	2204
	75%	2137	2123	2174	2176
	80%	2145	2210	2138	2187
	85%	2164	2188	2187	2169
	90%	2180	2190	2212	2219
	100%	2197	2258	2198	2240

Tabla A10 Modelo genético, problema R201, población = 200.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	2097	2012	2156	2143
	70%	2008	2058	2045	2086
	75%	2019	2063	2026	2075
	80%	2022	2055	2069	2069
	85%	2037	2109	2087	2084
	90%	2044	2135	2078	2138
	100%	2108	2157	2112	2152

Para el problema R201 utilizando el modelo genético, el mejor resultado obtenido fue de 2008, para el cual se utilizó 200 individuos de población, 70% de probabilidad de selección y 2,5% de probabilidad de mutación.

En la figura A.4 se muestran gráficamente los resultados obtenidos tanto del promedio de fitness de la población, como de la mejor solución obtenida en cada generación.

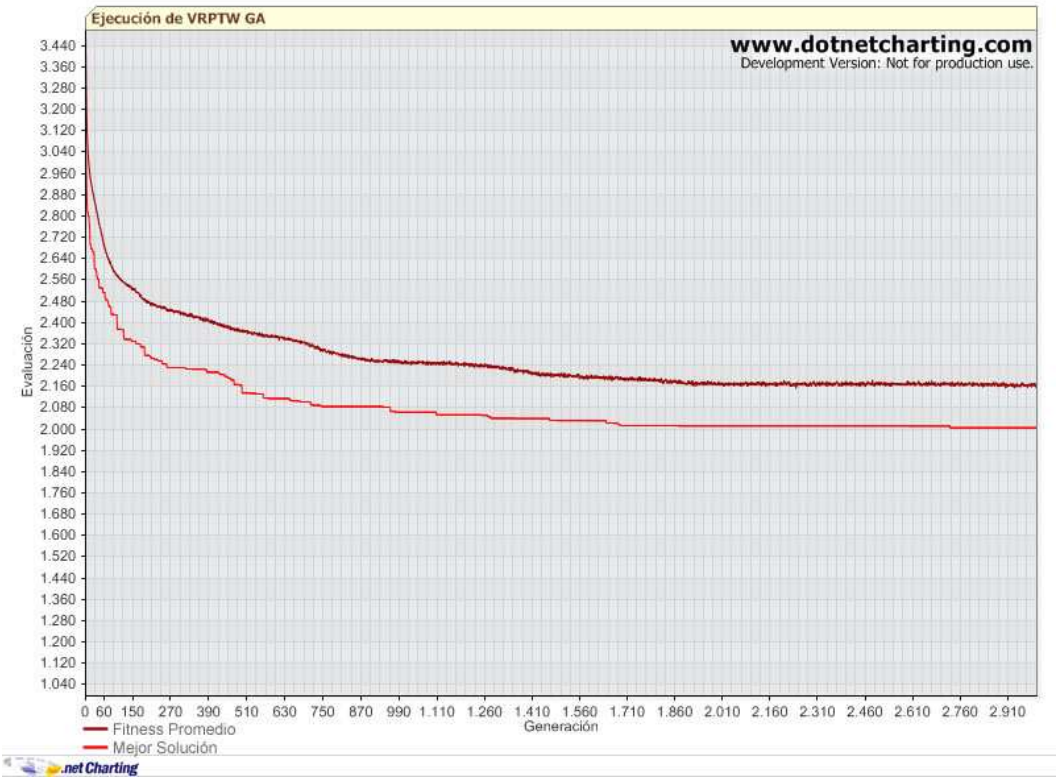


Figura A.4 Resultado algoritmo genético para el problema R201.

- **Problema RC101**

En las tablas A11 y A12 se presentan los resultados obtenidos al ejecutar el problema RC101 con el modelo genético, utilizando distintos valores de parámetros.

Tabla A11 Modelo genético, problema RC101, población = 100.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	2098	2012	2108	2087
	70%	1987	1976	1998	2022
	75%	1977	1965	1988	2013
	80%	1990	1978	2001	2017
	85%	2014	1993	2065	1993
	90%	2008	2015	2034	2025
	100%	2056	2042	2150	2104

Tabla A12 Modelo genético, problema RC101, población = 200.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	1936	1843	1906	1933
	70%	1887	1866	1874	1901
	75%	1852	1816	1843	1876
	80%	1804	1812	1858	1922
	85%	1814	1841	1878	1913
	90%	1845	1860	1921	1965
	100%	1898	1887	1913	1959

Para el problema RC101 utilizando el modelo genético, el mejor resultado obtenido fue de 1804, para el cual se utilizó 200 individuos de población, 80% de probabilidad de selección y 2,5% de probabilidad de mutación.

En la figura A.5 se muestran gráficamente los resultados obtenidos tanto del promedio de fitness de la población, como de la mejor solución obtenida en cada generación.

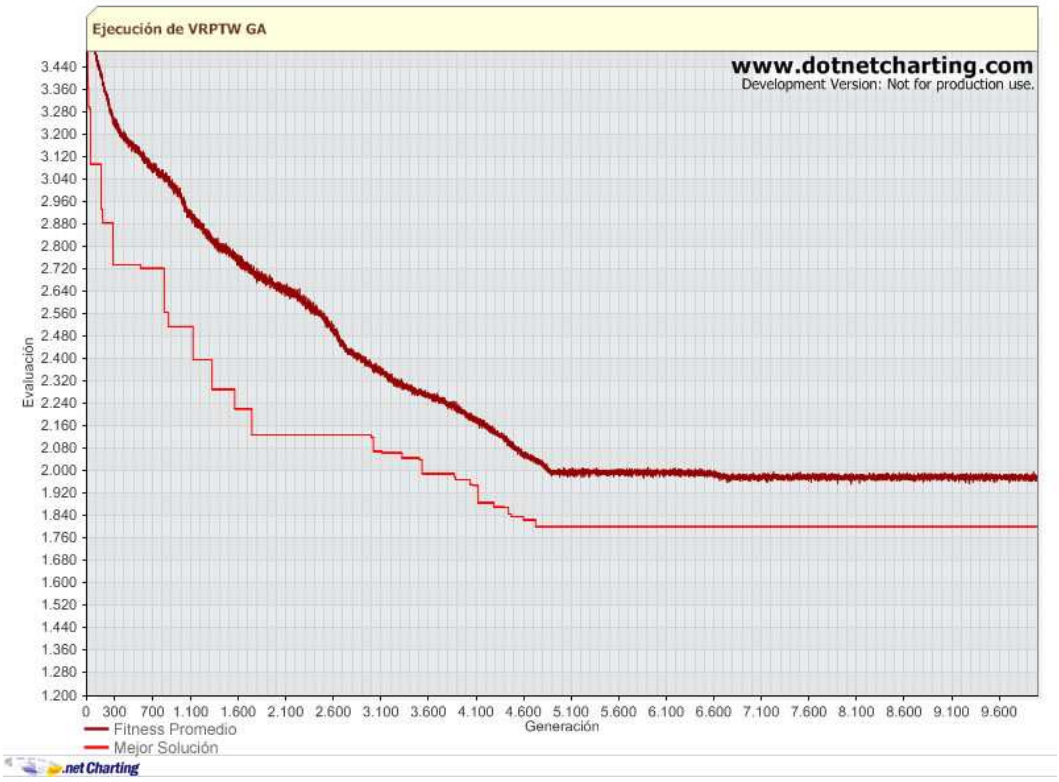


Figura A.5 Resultado algoritmo genético para el problema RC101.

Problema RC201

En las tablas A13 y A14 se presentan los resultados obtenidos al ejecutar el problema RC201 con el modelo genético, utilizando distintos valores de parámetros.

Tabla A13 Modelo genético, problema RC201, población = 100.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	2487	2423	2493	2564
	70%	2425	2398	2421	2481
	75%	2410	2401	2467	2490
	80%	2415	2415	2459	2507
	85%	2487	2487	2480	2485
	90%	2479	2536	2517	2542
	100%	2521	2497	2543	2532

Tabla A14 Modelo genético, problema RC201, población = 200.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	22349	2248	2347	2459
	70%	2387	2222	2319	2417
	75%	2413	2305	2421	2435
	80%	2447	2341	2569	2458
	85%	2456	2378	2461	2489
	90%	2436	2345	2455	2531
	100%	2518	2494	2567	2455

Para el problema RC201 utilizando el modelo genético, el mejor resultado obtenido fue de 2222, para el cual se utilizó 200 individuos de población, 70% de probabilidad de selección y 5% de probabilidad de mutación.

En la figura A.6 se muestran gráficamente los resultados obtenidos tanto del promedio de fitness de la población, como de la mejor solución obtenida en cada generación.

Figura A.6 Resultado algoritmo genético para el problema RC201

A.3 Modelo cultural

En el siguiente apéndice se detallan los resultados de la calibración de parámetros obtenidos para los distintos problemas de Solomon utilizando el modelo genético con influencia cultural, ejecutando distintas combinaciones de cantidad de población, probabilidad de selección, probabilidad de mutación y probabilidad de seleccionar un líder.

- **Problema C101**

En las tablas A15 y A16 se presentan los resultados obtenidos al ejecutar el problema C101 con el modelo cultural, utilizando distintos valores de parámetros.

Tabla A15 Modelo cultural, problema C101, población = 200, probabilidad líder = 15%.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	1139	1125	1234	1211
	70%	1010	1005	1176	1149
	75%	954	1131	1087	1109
	80%	986	1076	1176	1148
	85%	1045	1128	1097	1187
	90%	1076	1116	1184	1154
	100%	1165	1187	1212	1198

Tabla A16 Modelo cultural, problema C101, población = 200, probabilidad líder = 30%.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	986	1017	1014	1120
	70%	894	887	986	1038
	75%	919	879	943	1007
	80%	990	1006	1001	1011
	85%	1042	955	1029	1044
	90%	1012	976	980	1125
	100%	1009	1008	1018	1142

Para el problema C101 utilizando el modelo genético con influencia cultural, el mejor resultado obtenido fue de 879, para el cual se utilizó 200 individuos de población, 75% de probabilidad de selección, 5% de probabilidad de mutación y 30% de probabilidad de seleccionar un líder.

En la figura A.7 se muestran gráficamente los resultados obtenidos tanto del promedio de fitness de la población, como de la mejor solución obtenida en cada generación.

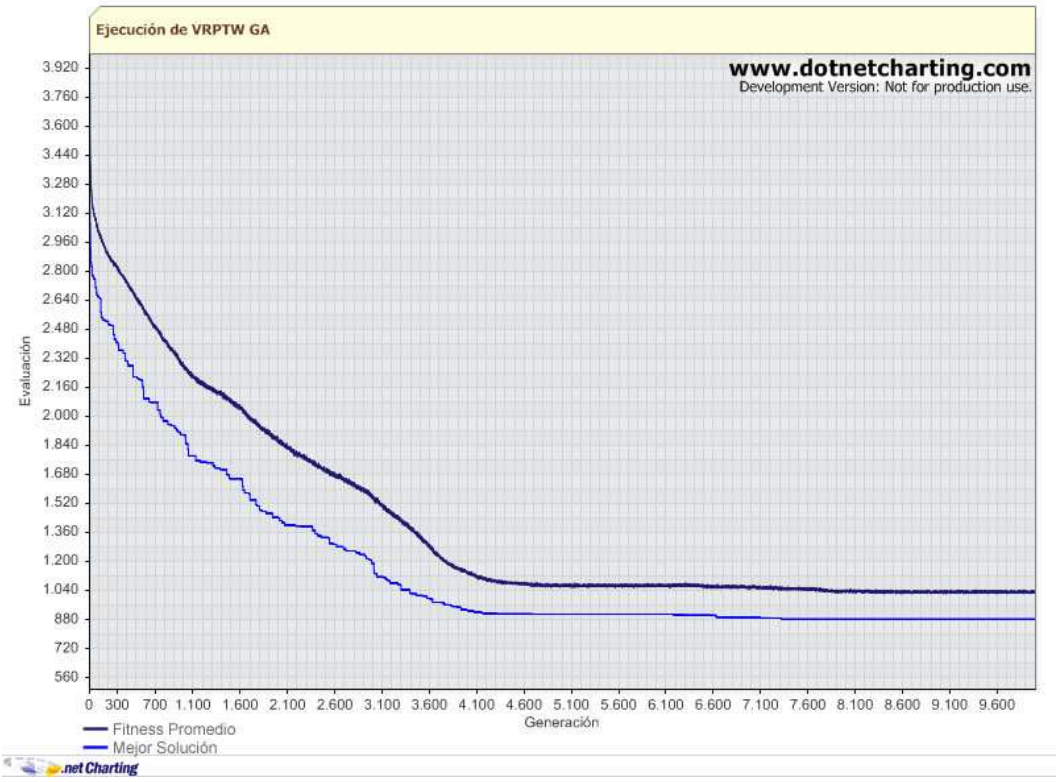


Figura A.7 Resultado algoritmo cultural para el problema C101.

- **Problema C201**

En las tablas A17 y A18 se presentan los resultados obtenidos al ejecutar el problema C201 con el modelo cultural, utilizando distintos valores de parámetros.

Tabla A17 Modelo cultural, problema C201, población = 100, probabilidad líder = 15%.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	598	593	598	604
	70%	602	591	593	602
	75%	602	593	598	604
	80%	593	593	602	598
	85%	598	593	602	602
	90%	593	598	593	604
	100%	602	593	602	602

Tabla A18 Modelo cultural, problema C201, población = 100, probabilidad líder = 30%.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	598	593	598	593
	70%	598	591	598	598
	75%	593	591	593	598
	80%	598	593	593	598
	85%	598	593	593	602
	90%	598	598	598	598
	100%	593	598	598	598

Para el problema C201 utilizando el modelo genético con influencia cultural, el mejor resultado obtenido fue de 591, para el cual se utilizó 100 individuos de población, 70% de probabilidad de selección, 5% de probabilidad de mutación y 30% de probabilidad de seleccionar un líder.

En la figura A.8 se muestran gráficamente los resultados obtenidos tanto del promedio de fitness de la población, como de la mejor solución obtenida en cada generación.

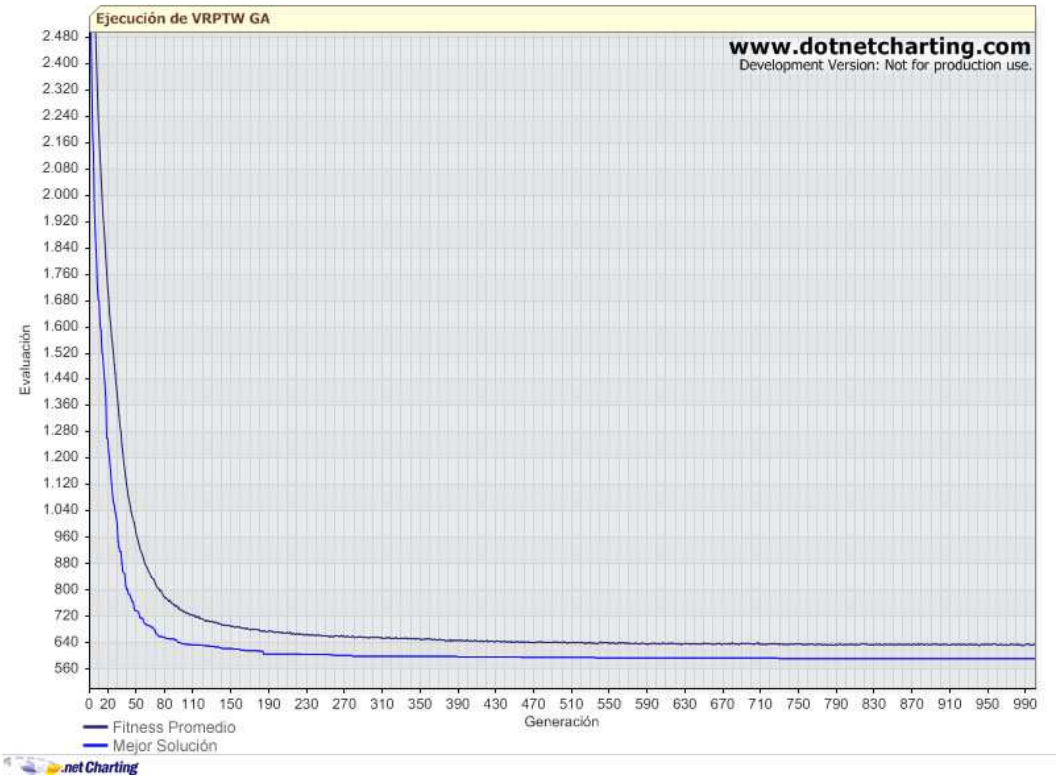


Figura A.8 Resultado algoritmo cultural para el problema C201.

Problema R101

En las tablas A19 y A20 se presentan los resultados obtenidos al ejecutar el problema R101 con el modelo cultural, utilizando distintos valores de parámetros.

Tabla A19 Modelo cultural, problema R101, población = 200, probabilidad líder = 15%.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	1690	1768	1712	1788
	70%	1684	1693	1709	1723
	75%	1687	1701	1714	1710
	80%	1690	1707	1739	1754
	85%	1712	1734	1754	1733
	90%	1704	1755	1788	1764
	100%	1738	1746	1749	1732

Tabla A20 Modelo cultural, problema R101, población = 200, probabilidad líder = 30%.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	1756	1732	1759	1789
	70%	1704	1717	1729	1754
	75%	1721	1702	1712	1758
	80%	1714	1710	1711	1765
	85%	1717	1743	1720	1758
	90%	1765	1735	1737	1789
	100%	1769	1771	1762	1782

Para el problema R101 utilizando el modelo genético con influencia cultural, el mejor resultado obtenido fue de 1684, para el cual se utilizó 200 individuos de población, 70% de probabilidad de selección, 2,5% de probabilidad de mutación y 15% de probabilidad de seleccionar un líder.

En la figura A.9 se muestran gráficamente los resultados obtenidos tanto del promedio de fitness de la población, como de la mejor solución obtenida en cada generación.

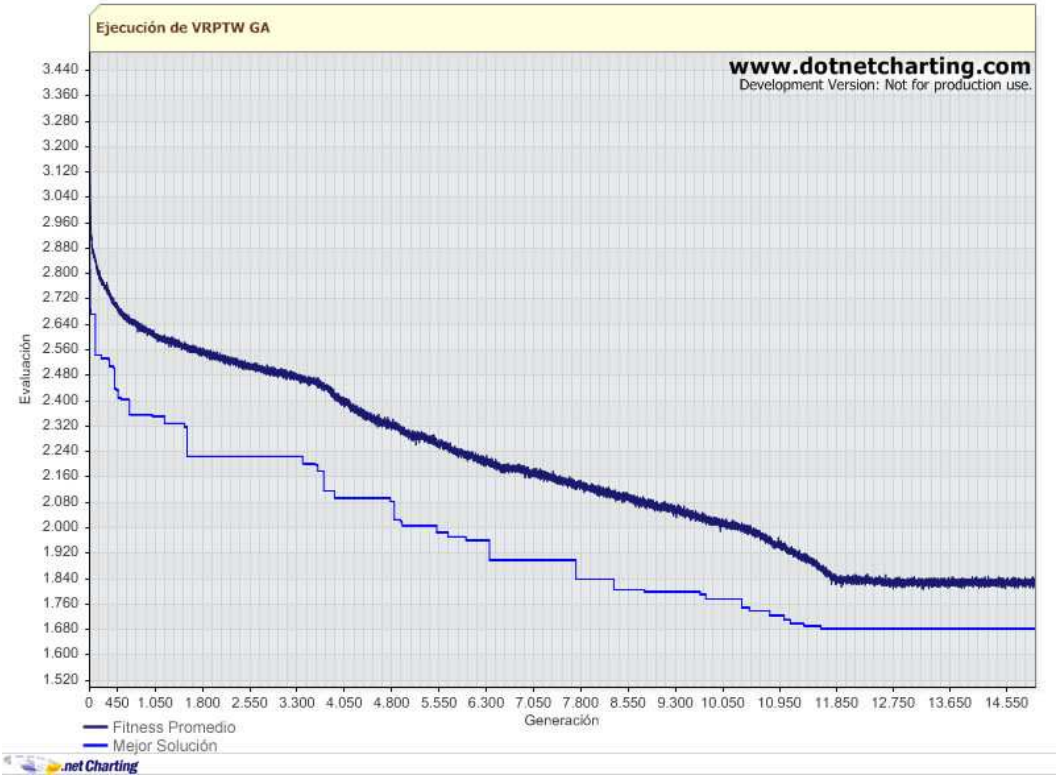


Figura A.9 Resultado algoritmo cultural para el problema R101.

- **Problema R201**

En las tablas A21 y A22 se presentan los resultados obtenidos al ejecutar el problema R201 con el modelo cultural, utilizando distintos valores de parámetros.

Tabla A21 Modelo cultural, problema R201, población = 200, probabilidad líder = 15%.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	1754	1762	1779	1780
	70%	1719	1697	1742	1764
	75%	1698	1704	1729	1751
	80%	1730	1714	1754	1740
	85%	1731	1720	1764	1744
	90%	1756	1741	1740	1763
	100%	1743	1732	1773	1766

Tabla A22 Modelo cultural, problema R201, población = 200, probabilidad líder = 30%.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	1701	1679	1698	1714
	70%	1692	1688	1693	1737
	75%	1687	1684	1701	1721
	80%	1698	1692	1713	1744
	85%	1714	1703	1743	1714
	90%	1715	1714	1721	1754
	100%	1741	1755	1738	1740

Para el problema R201 utilizando el modelo genético con influencia cultural, el mejor resultado obtenido fue de 1679, para el cual se utilizó 200 individuos de población, 65% de probabilidad de selección, 5% de probabilidad de mutación y 30% de probabilidad de seleccionar un líder.

En la figura A.10 se muestran gráficamente los resultados obtenidos tanto del promedio de fitness de la población, como de la mejor solución obtenida en cada generación.

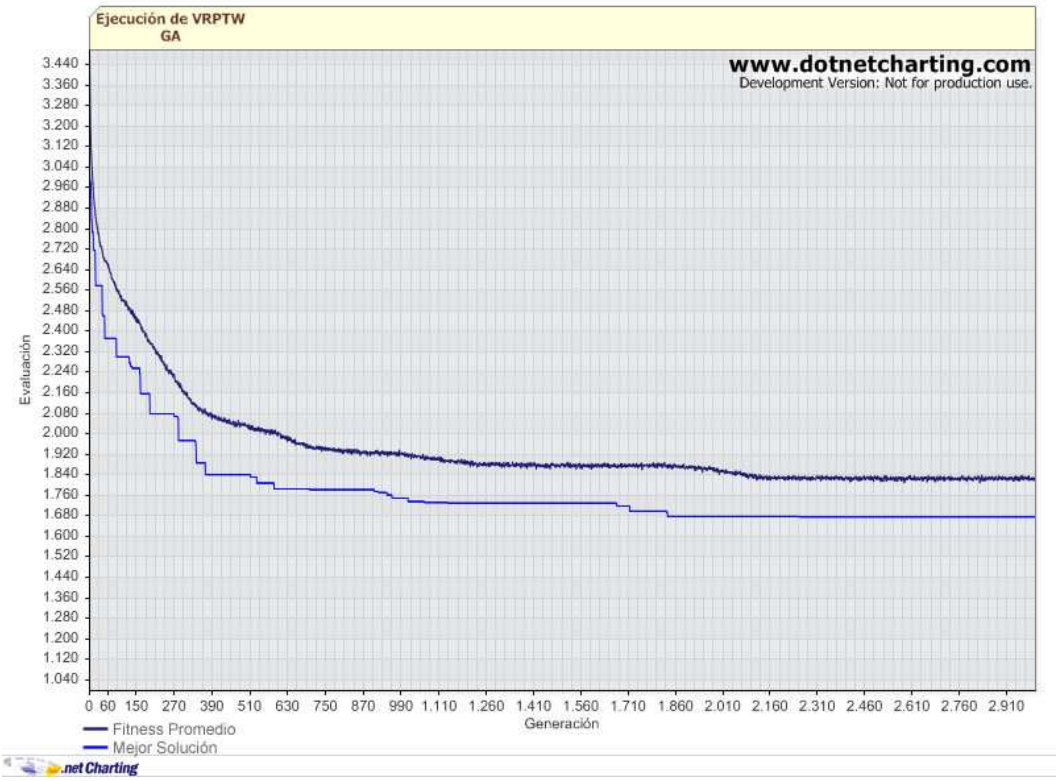


Figura A.10 Resultado algoritmo cultural para el problema R201.

- **Problema RC101**

En las tablas A23 y A24 se presentan los resultados obtenidos al ejecutar el problema RC101 con el modelo cultural, utilizando distintos valores de parámetros.

Tabla A23 Modelo cultural, problema RC101, población = 200, probabilidad líder = 15%.

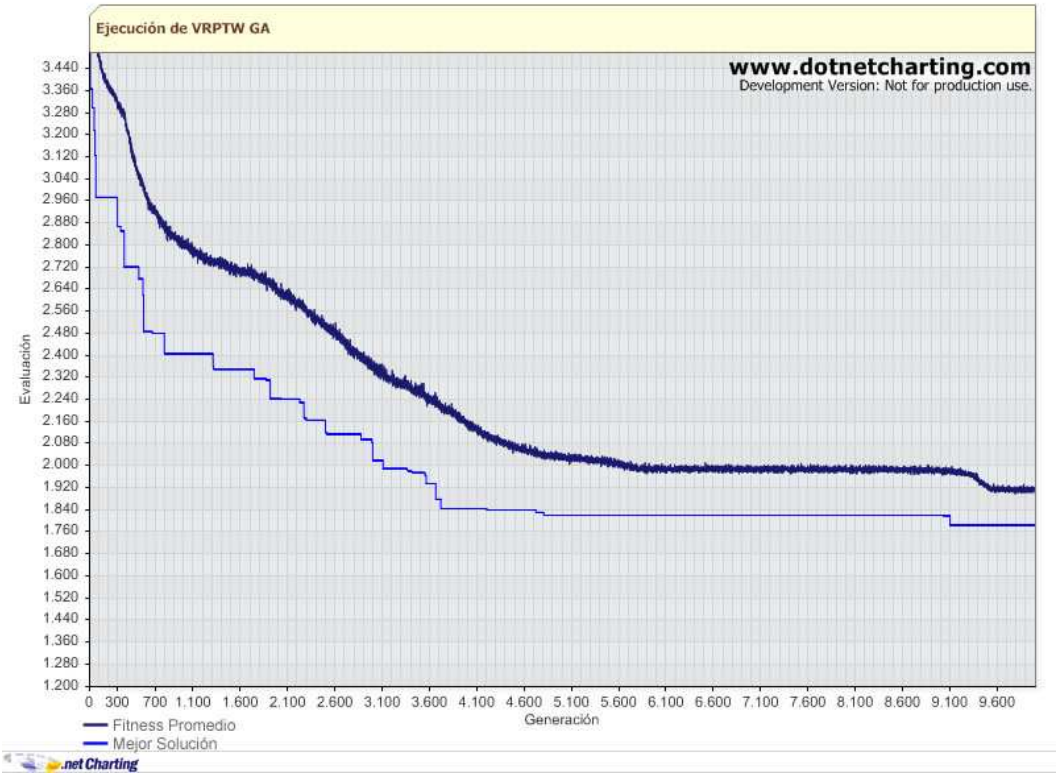
		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	1843	1872	1835	1876
	70%	1795	1808	1813	1880
	75%	1786	1794	1845	1821
	80%	1804	1827	1850	1862
	85%	1796	1834	1867	1883
	90%	1812	1854	1869	1847
	100%	1835	1840	1881	1870

Tabla A24 Modelo cultural, problema RC101, población = 200, probabilidad líder = 30%.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	1840	1844	1856	1862
	70%	1816	1804	1832	1821
	75%	1824	1815	1820	1835
	80%	1853	1836	1855	1869
	85%	1839	1856	1871	1865
	90%	1840	1870	1875	1882
	100%	1844	1867	1884	1889

Para el problema RC101 utilizando el modelo genético con influencia cultural, el mejor resultado obtenido fue de 1786, para el cual se utilizó 200 individuos de población, 70% de probabilidad de selección, 2,5% de probabilidad de mutación y 15% de probabilidad de seleccionar un líder.

En la figura A.11 se muestran gráficamente los resultados obtenidos tanto del promedio de fitness de la población, como de la mejor solución obtenida en cada generación.



FiguraA.11 Resultado algoritmo cultural para el problema RC101.

- **Problema RC201**

En las tablas A25 y A26 se presentan los resultados obtenidos al ejecutar el problema RC201 con el modelo cultural, utilizando distintos valores de parámetros.

Tabla A25 Modelo cultural, problema RC201, población = 200, probabilidad líder = 15%.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	2012	2018	2108	2143
	70%	2005	2021	2099	2106
	75%	2047	2054	2012	2126
	80%	2019	2068	2127	2076
	85%	2065	2075	2114	2148
	90%	2103	2037	2198	2156
	100%	2069	2015	2143	2128

Tabla A26 Modelo cultural, problema RC201, población = 200, probabilidad líder = 30%.

		Probabilidad de selección			
		2,5%	**5%**	**7,5%**	**10%**
Probabilidad de mutación	**60%**	2013	2094	2120	2154
	70%	1990	1989	2065	2045
	75%	2013	2076	2115	2092
	80%	2020	2092	2120	2169
	85%	2051	2041	2080	2135
	90%	2044	1928	2057	2134
	100%	2102	2103	2098	2154

Para el problema RC201 utilizando el modelo genético con influencia cultural, el mejor resultado obtenido fue de 1926, para el cual se utilizó 200 individuos de población, 90% de probabilidad de selección, 5% de probabilidad de mutación y 30% de probabilidad de seleccionar un líder.

En la figura 8.12 se muestran gráficamente los resultados obtenidos tanto del promedio de fitness de la población, como de la mejor solución obtenida en cada generación.

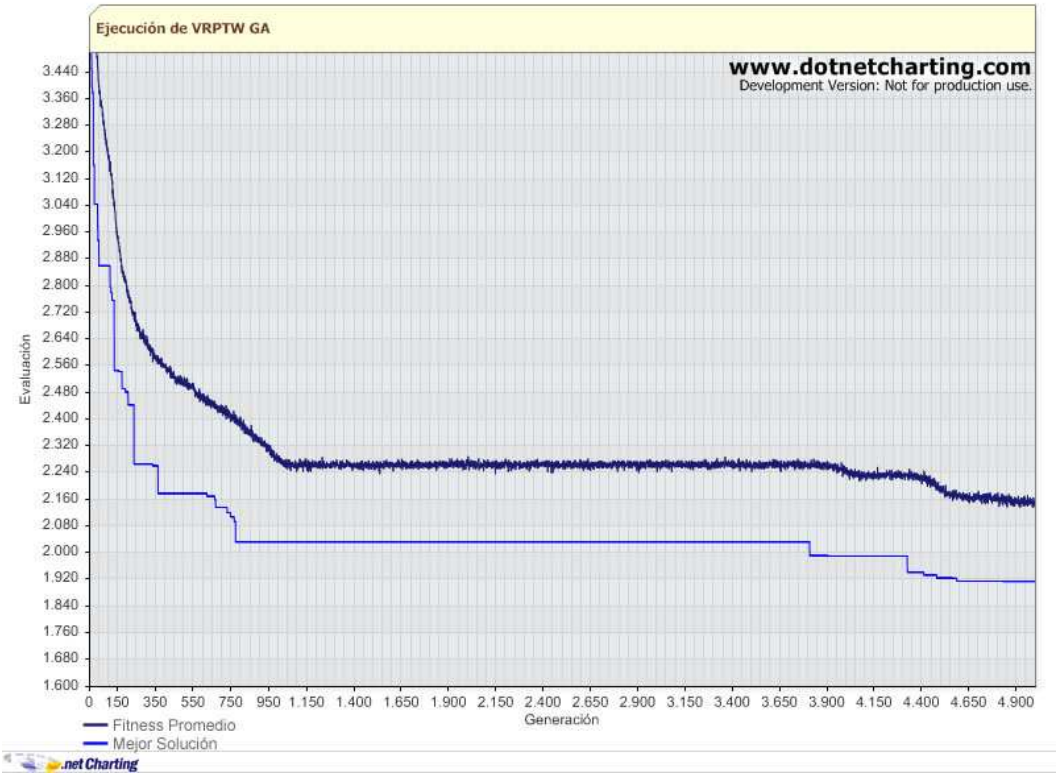


Figura 8.12 Resultado algoritmo cultural para el problema RC201.