Pontificia Universidad Católica de Valparaíso

Facultad de Ingeniería

Escuela de Ingeniería Informática

Ingeniería Civil en Informática

ESTUDIO DE ALGORITMOS GENÉTICOS PARA EL PROBLEMA DE TRANSPORTE DE PASAJEROS (DARPTW)

Autor:

Enrique Andrés Urra Coloma

Informe final del Proyecto para optar al Título profesional de

Ingeniero Civil en Informática

Profesor guía:

Claudio Cubillos Figueroa

Julio del 2007

Este trabajo va dedicado a mis padres, quienes siempre me otorgaron su apoyo para terminar este importante trabajo en mi carrera…

Lista de abreviaturas

DARPTW: Dial-a-ride problem with time windows.

DARP: Dial-a-ride problem.

GA: Genetic algorithm.

LLGA: Linkage learning genetic algorithm.

DRTS: Demand responsive transport system.

VRP: Vehicle routing problem.

TW: Time window.

PDP: Pickup and delivery problem.

TSP: Travelling salesman problem.

mGA: Messy genetic algorithm.

cGA: Compact genetic algorithm.

PMX: Partially matched crossover.

MRT: Maximum ride time.

DRT: Direct ride time.

Resumen

En el problema de transporte *Dial-a-Ride Problem with Time Windows*, se tiene un conjunto de solicitudes de transporte de personas desde un lugar de origen a un lugar de destino a través de una red de locaciones, bajo distintos tipos de restricciones, entre las que destacan principalmente las ventanas de tiempo y la capacidad limitada de los vehículos. La dificultad del problema (*NP-Hard*) ha impulsado la aplicación de heurísticas para su resolución. En este contexto, el uso de Algoritmos Genéticos en DARPTW no ha sido mayormente abordado, salvo estudios puntuales.

En el contexto de un DARPTW altamente restrictivo, en este trabajo se desarrollaron un par de modelos de GA: Uno está basado en la adaptación de un modelo genérico presentado en la literatura, llamado Linkage Learning Genetic Algorithm, mientras que el otro corresponde a una propuesta basada en operadores personalizados. De forma transversal a ambos modelos, se han aplicado técnicas de pre-procesamiento en los datos de entrada, con el fin de obtener información útil en lo que se refiere al manejo de restricciones del problema, y facilitando con ello la búsqueda de soluciones y utilización de operadores en un contexto factible. Los modelos fueron implementados, calibrados y testeados de forma adecuada al número de parámetros relevantes usados, junto con la utilización de pautas de comparación acorde a resultados presentados en otros estudios en la literatura.

Los resultados mostraron que el modelo basado en LLGA entrega mejoras interesantes en lo que se refiere al tiempo de ejecución del algoritmo, además que los dos modelos desarrollados en este estudio mejoraron en cierto nivel las prestancias de la calidad de servicio en las instancias testeadas. Estas comparaciones fueron hechas pensando en las diferencias respecto a los estudios comparados, ofreciendo nuevas alternativas de investigación: por un lado la posibilidad de tener más puntos de comparación en lo que se refiere a escenarios altamente restrictivos, y por otro la posibilidad de modificar los

modelos de forma que las comparaciones puedan realizarse en un conjunto más homogéneo en lo que se refiere a las características del problema.

Palabras-claves: *Algoritmos Genéticos, Dial-a-Ride Problem.*

Abstract

On the Dial-a-Ride Problem with Time Windows, there are a set of requests from customers to be transported from a pickup point to a delivery point through a locations network, considering diverse constraints, including time windows and the limited vehicle capacity. The problem complexity (NP-Hard) has promoted the application of diverse heuristics to solve the problem. On this context, the use of Genetic Algorithms for DARPTW has not been largely considered, excepting for few researches.

On the context of a highly restricted DARPTW, two GA models have been developed for the problem: The first is adapted from a generic model presented in the literature: the Linkage Learning Genetic Algorithm. The second is based on a new set of proposed operators. Transversally to both models, a set of pre-processing techniques have been applied to the input data, to generate useful information in constraints manipulation, facilitating the solutions search and the application of the operators on a feasible context. The models were implemented, tuned and tested according to the relevant parameters considered, along with the use of evaluation patterns according to the results exposed on other researches from the literature.

The results reported that the LLGA based model offers interesting improvements in the execution times, also the two models developed in this research have improved the quality of service factors for the tested instances. Anyway, these comparisons consider the differences between the compared researches, offering new investigation approaches: on the one hand, a bigger comparison spectrum for highly restrictive instances of the problem, and on the other, the possibility of modifying the models to a more homogeneous approach in the problem characteristics.

Key-words: *Genetic Algorithms, Dial-a-Ride Problem*.

1. Presentación del tema

1.1. Introducción

El estudio de sistemas de transporte, tanto en sus características como en su planificación, ha sido de gran relevancia desde hace muchos años. Algunos de los que han ofrecido mayores desafíos en el contexto de la investigación son los que se pueden clasificar como "en respuesta a la demanda" (*Demand Responsive Transport Systems*). Estos se caracterizan por el hecho que se realiza una solicitud previa a la ejecución del servicio, lo que se conoce como una "solicitud avanzada", por ejemplo cuando un cliente solicita el servicio por medio de una llamada telefónica. Se trata de servicios cuyas rutas y horarios son variables, además que funcionan principalmente de puerta a puerta a diferencia de los sistemas de transporte más tradicionales que lo hacen de parada a parada. Desde los años 70 que son usados, en particular para el transporte de adultos mayores y personas con discapacidad, sin embargo, gracias a nuevas tecnologías que han ido apareciendo en los últimos años, su consideración y alcances han aumentado importantemente.

Las principales dificultades al abordar este tipo de sistemas se generan en virtud de la naturaleza de las restricciones y parámetros críticos que ellos presentan. Por ejemplo, en [1] se encuentra un estudio enfocado a comprender y predecir las consecuencias que generan distintas políticas de administración en estos sistemas de transporte, en base al análisis de los requerimientos de los pasajeros y el costo de los operadores asociados al problema. En [2] se realiza un estudio vía simulación que muestra de forma cuantitativa la dependencia entre la productividad y costos de estos sistemas con dos factores importantes como son el tamaño de las ventanas de tiempo (mencionadas en lo que sigue) y el tipo de estrategia (centralizada o descentralizada) que se utilice.

En términos de problema de optimización, un DRTS puede representarse en el *Dial-a-Ride Problem* o el problema de transporte de pasajeros. Consiste en buscar la mejor forma de transportar un conjunto de pasajeros, distribuidos espacialmente en una red de

locaciones, en base a una serie de restricciones de distinto tipo. Entre estas destacan la capacidad limitada de cada vehículo y la consideración de ventanas de tiempo o time windows, que corresponden a un intervalo válido de tiempo en el cual un usuario puede ser recogido o entregado en la locación origen o destino respectivamente que especifique. El objetivo es optimizar los factores vinculados al sistema de transporte (número de vehículos requeridos, costos de viaje) y al servicio entregado a los usuarios (tiempo de espera, tiempo de transporte). DARP es una versión particular de una familia de problemas de transporte, en los que se consideran (en orden de generalidad respectivo) el *travelling salesman problem, el vehicle routing problem y el pickup and delivery problem.* DARPTW es considerado un problema de tipo NP-Hard, principalmente por la restricción que imponen las TW ([1], [4]). Por ello, en términos prácticos, este problema suele ser abordado con la utilización de heurísticas para encontrar soluciones aproximadas, bajo las distintas variantes que pueda presentar.

Una de las herramientas usadas ampliamente en problemas de transporte son los Algoritmos Genéticos. Tras su aparición en el escenario científico gracias a John Holland en los años 70 [5], 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. Hasta nuestros días, son numerosos los estudios que se han realizado en el área respecto a las dificultades que el GA básico puede tener al abordar problemas de optimización de alta complejidad. La base de toda la investigación en este contexto nace justamente en los inicios de los GAs, cuando Holland establece un modelo formal para mostrar su efectividad en procesos de búsqueda, presentando el *schema theorem.* Este teorema es el pilar fundamental en el funcionamiento de los GAs, desde él se desprende la idea que pequeños "esquemas" o *schematas* (patrones que sirven para analizar similaridades entre cromosomas) de alta calidad, tienden a subsistir y ser más frecuentes en generaciones subsecuentes de un GA. Posteriormente Goldberg [6] formaliza la hipótesis de los bloques de construcción o *building blocks,* señalando que los GAs dirigen una búsqueda de resultados casi óptimos a través de la yuxtaposición de schematas pequeños, de bajo orden y de alto rendimiento llamados building blocks. También se han estudiado problemáticas particulares en las que la utilización de esta estrategia no siempre conduce a

óptimos globales, si no que lleva a óptimos locales. A este tipo de problemas se les ha catalogado de "engañosos" o *deceptive problems* [7]. Por último, se debe mencionar la importancia (en problemas engañosos especialmente) que ha adoptado el nivel de acoplamiento o *linkage* que los *building blocks* posean dentro de un cromosoma y como éste se ve afectado cuando se mezclan entre distintos individuos. En [8] y [9] Harik profundiza en el tema y propone un nuevo tipo de GA que a través de su representación y operadores, pretende abordar este aspecto que hasta la fecha había sido muy descuidado.

En general, todos estos conceptos han sido aplicados en distintos tipos de nuevos GAs que pretenden de una u otra forma mejorar el rendimiento y las prestaciones en la búsqueda de soluciones para distintos tipos de problemas. En [10] se mencionan tres aproximaciones que los GAs han utilizado para abordar particularmente el problema de *linkage*, que en la práctica considera los demás conceptos mencionados:

- **Métodos basados en perturbación:** buscan el acoplamiento perturbando un individuo y midiendo el cambio en su calidad.

- **Técnicas para adaptación del acoplamiento:** se basan en la representación, operadores y mecanismos del propio GA para adaptar el acoplamiento en el proceso evolutivo del algoritmo.

- **Constructores de modelos probabilísticos:** Adquieren el acoplamiento a través de la construcción de modelos probabilísticos.

Los GAs han sido aplicados en VRP, allí tenemos ejemplos como [11] donde se usa una estrategia *cluster-first route-second*, que consiste en primero hacer una asignación de pasajeros a los vehículos (lo que se realiza con GAs) y posteriormente se optimizan las rutas para los vehículos utilizando una heurística de optimización local. En [12] se utiliza una estrategia completa con GAs que arroja resultados experimentales interesantes. En PDP con [13] se utilizó un mecanismo de selección especial para resolver el problema.

En el caso de DARP, es poco lo que se puede encontrar respecto a la utilización de GAs. En [1] y [14], al igual que el caso de [11], se usa la estrategia *cluster-first route-second* para resolver el problema, considerando el uso de GA para la asignación. En [15] se utilizan los

GAs para resolver el problema de forma completa. Se debe considerar que por la ausencia de datos de testing o *benchmarks* conocidos para DARPTW, estos estudios se basaron en datos muy particulares para realizar la experimentación, cosa que no ocurre en VRP por ejemplo donde se tienen las instancias de Solomon, más estandarizadas. La razón de que no haya estudios más acabados en esta materia, va por la complejidad que implica la implementación de todas las variables influyentes de DARPTW en un GA, que a diferencia de otros problemas, van muy relacionadas con el factor de calidad de servicio que se entrega al cliente. De hecho en [15] se considera directamente como un problema engañoso.

En esta investigación se profundizó en el estudio de GAs para DARPTW considerando un escenario restrictivo, similar al presentado en [15]. Esto se realizó desde una perspectiva experimental, es decir, en base al desarrollo de una representación adecuada y operadores acorde a las necesidades del problema, junto con la comparación tanto entre los algoritmos implementados como de cada algoritmo respecto a otros estudios abordados previamente en la literatura. Se desarrollaron dos modelos, uno basado en un GA genérico de la literatura, denominado *Linkage Learning Genetic Algorithm* y el otro basado en operadores y estructuras personalizadas.

En lo que resta de este capítulo se analizarán los objetivos planteados en esta investigación, tanto el general como los específicos con sus respectivas justificaciones, además que se describirá la metodología que se usará en la investigación y desarrollo, junto con un plan de trabajo.

En el capítulo 2 se hará una revisión teórica del problema de transporte de pasajeros DARPTW, mostrando las restricciones más relevantes que presenta, en especial las que lo convierten en un problema difícil de abordar con GAs. Se mostrará una formalización matemática de DARPTW como modelo de referencia en este trabajo.

En el capítulo 3 se detallarán los aspectos más importantes de los GAs, profundizando en los conceptos descritos en esta introducción y detallando algunos de los GAs (modificaciones) que en la clasificación mencionada antes fueron considerados en esta investigación.

En el capítulo 4 se repasarán algunos estudios que hasta el momento se han realizado y publicado para el uso de GAs en DARPTW y problemas relacionados, en especial [1] y [15], destacando el aporte en esta investigación.

En el capítulo 5 se hará una descripción detallada, en el contexto de un marco de trabajo, de los modelos implementados y evaluados en el transcurso de esta investigación.

En el capítulo 6 se presentarán algunos resultados experimentales obtenidos con los modelos propuestos, además se mencionarán distintas consideraciones previas de implementación, enfocadas a los datos de prueba a utilizar y a las condiciones influyentes de los parámetros más importantes del problema tratado.

Conclusiones y futuro trabajo se detallarán en el capítulo 7.

1.2. Objetivos del proyecto

1.2.1. Objetivo general

Desarrollar y evaluar comparativamente un par de modelos de Algoritmos Genéticos para el problema de transporte de pasajeros con ventanas de tiempo (DARPTW).

1.2.2. Objetivos específicos

• Realizar un análisis detallado acerca de distintas soluciones propuestas para DARPTW y problemas de transporte asociados, poniendo especial énfasis en trabajos vinculados con Algoritmos Genéticos, junto con analizar los requerimientos y alternativas de Algoritmos Genéticos para abordar DARPTW.

• Definir para su desarrollo y aplicación en el proyecto, un par de variantes de Algoritmos Genéticos, considerando los submodelos necesarios para cada uno, e implementarlos en un programa de experimentación usando el lenguaje de programación C Sharp.

• Realizar pruebas con el programa previamente mencionado, usando datos de entrada apropiados, junto con presentar los resultados obtenidos de forma comparativa entre los modelos desarrollados y respecto a otros estudios de la literatura.

1.3. Metodología de trabajo

En el desarrollo del proyecto, se utilizó una metodología de investigación científica clásica. La misma puede ser descrita bajo las siguientes etapas genéricas:

• 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 a la viabilidad del problema. Estos elementos se pueden reflejar directamente en los objetivos mismos del proyecto y la introducción bibliográfica donde se resume el estado de arte del problema.

• Se desarrolla un marco teórico, lo que contempla la revisión de la literatura del punto de vista de su 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 reflejado en las distintas conclusiones que se puedan obtener de las primeras etapas del proyecto. Las variables y elementos relacionados en este proyecto estarán enmarcadas en los modelos de GA desarrollados. Para el contexto de este problema, el enfoque utilizado es netamente experimental.

• 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 DARPTW, las que pueden ser obtenidas tanto de estudios anteriores como de la transformación de datos de prueba estandarizados pertenecientes a otros problemas similares, los que en el caso de

DARPTW no se tienen a mano. Además se consideran todos los parámetros relevantes tanto en DARPTW como en GAs. El instrumento utilizado corresponderá al programa de experimentación planteado en los objetivos, el cual será construido, aplicado y validado adecuadamente.

• Se realiza el análisis profundo de los resultados obtenidos, a través de un problema de análisis elaborado convenientemente. Este problema contempla los aspectos más relevantes de DARPTW y el planteamiento comparativo respecto a otros estudios.

• Se elaboran y presentan los reportes de investigación adecuados.

1.4. Plan de trabajo

Para cumplir los objetivos establecidos en 1.2, se dividió la realización del proyecto en tres etapas generales:

• **Recolección de antecedentes y análisis investigativo:** En esta etapa se realiza una revisión detallada de las fuentes y referencias encontradas. De ellas se rescatan los elementos más importantes y útiles que tengan relación con las características de DARPTW, los aspectos teóricos de los GAs y la aplicación en conjunto de ambos. Tiempo de duración propuesta: 2 meses (Agosto-Septiembre 2006).

• **Desarrollo y construcción de modelos:** Con la información analizada en la etapa anterior, se procede a definir y desarrollar cada uno de los modelos de GA. Este desarrollo contempla el estudio de variantes de GAs encontrados en la literatura, que sean útiles para el problema. En la construcción se contemplan los distintos submodelos para los GA, es decir, representación, recombinación, selección, mutación y generación de población inicial. Para validarlos y verificarlos, se desarrollan prototipos de software, en los que se usarán datos de prueba preliminares. Se trata de una etapa iterativa e incremental. Tiempo de duración propuesta: 2 meses (Octubre-Noviembre 2006).

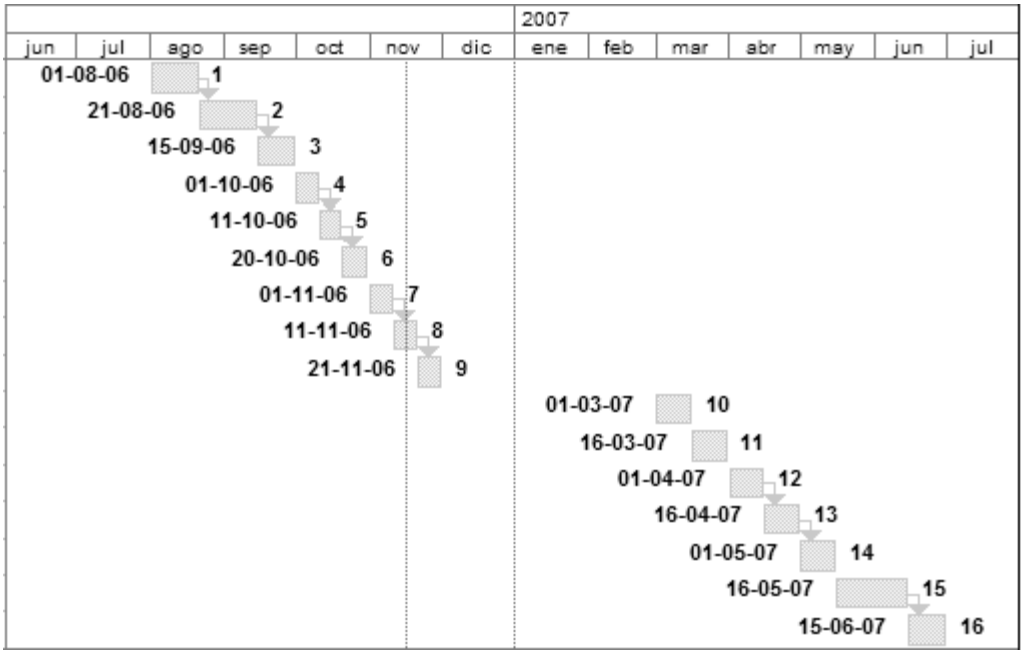
• **Implementación formal de los modelos y etapa de pruebas finales:** En base a los prototipos y a los modelos obtenidos en la etapa anterior, se desarrolla un software para generar soluciones en base a datos de entrada adecuados, no necesariamente los mismos

usados en los prototipos. Se realizan distintas ejecuciones que permiten medir la eficiencia en la implementación del modelo, realizando comparaciones con otras soluciones encontradas en la literatura. Tiempo de duración propuesta: 4 meses (Marzo-Junio 2007).

En la Figura 1.1 se presenta un esquema gráfico detallado de la planificación.

Figura 1.1 Planificación detallada del proyecto.

Id	Nombre de tarea	Duración
1	Desarrollo del marco teórico de DARP	20 días
2	Desarrollo del marco teórico de GAs	25 días
3	Investigación referente a trabajos relacionados	16 días
4	Desarrollo del marco teórico modelo 1	10 días
5	Construcción del modelo 1	9 días
6	Implementación y pruebas prototipo modelo 1	11 días
7	Desarrollo del marco teórico modelo 2	10 días
8	Construcción del modelo 2	10 días
9	Implementación y pruebas prototipo modelo 2	10 días
10	Definición de datos de prueba finales	15 días
11	Refinamiento de modelos	15 días
12	Construcción de SW de experimentación: Core	15 días
13	Construcción de SW de experimentación: Modelo 1	15 días
14	Construcción de SW de experimentación: Modelo 2	15 días
15	Desarrollo de pruebas	30 días
16	Análisis y presentación de resultados	16 días



2. DARPTW: Problema de transporte de pasajeros

En este capítulo se hará una descripción general de los elementos más importantes que se deben considerar para diseñar un modelo de búsqueda de soluciones en DARPTW. El problema en sí tiene una gran cantidad de factores y/o parámetros que pueden ser distintos en varias instancias del problema, por ello es importante especificar desde que enfoque los mismos se abordarán. Muchas de las características del problema se heredan de sus generalizaciones, las cuales serán revisadas a grandes rasgos.

Con el fin de dejar claramente establecidas las características del problema consideradas en este trabajo, se presentará un modelo matemático que muestra la función objetivo y las restricciones establecidas. Este modelo corresponde a la versión básica del que se propone en [16]. En [4] se usa ese modelo como base, pero se generaliza según otros criterios. Gran parte de los conceptos mencionados se extraen de ambas referencias también.

2.1. Características generales

El problema a describir se conoce en la práctica como *Dial-a-Ride Problem with Time Windows*, acá el considerar las ventanas de tiempo dentro de la definición marca diferencias fundamentales, en [4] se muestra de forma general como constituyen el factor que ayuda a justificar en gran parte que se trate de un problema de tipo *NP-Hard*, tal como lo son otros problemas más generales que consideran la misma restricción.

DARPTW es esencialmente un problema de optimización de múltiples objetivos. Esto porque son dos elementos críticos los que están en juego en las soluciones: por un lado se busca minimizar los costos totales de transporte y por otro se busca maximizar el nivel de servicio ofrecido a los usuarios, o equivalentemente, minimizar el grado de insatisfacción de los mismos con el servicio.

Se tiene un conjunto de peticiones de transporte de parte de los clientes. Dicho conjunto se asume como conocido de antemano y no se modifica durante la ejecución del algoritmo, lo

que define el problema como estático. Cada petición establece una ventana de tiempo para la entrega del cliente a su lugar de destino (*outbond customers*) u otra ventana para la recogida del cliente desde su lugar de origen (*inbound customers*). Para la ventana definida, se entrega tanto el rango inferior de la ventana como el superior. Se considera una solución como infactible si la entrega efectiva del cliente se hace en un tiempo fuera de la ventana, esto la caracteriza como ventana de tiempo rígida (*hard time windows*). La ventana de tiempo no definida, se genera en base a los valores de los rangos de la ventana que si lo está. Los detalles de esto último se repasan en 2.6.

Para realizar el servicio, se dispone de una flota de vehículos homogéneos, todos tienen la misma capacidad de carga que naturalmente no puede ser sobrepasada. Los pasajeros son recogidos desde su lugar de origen y entregados en su lugar de destino por el mismo vehículo. Adicionalmente, de dispone de múltiples depósitos para los vehículos, estos son locaciones particulares donde comienzan y terminan su recorrido. El depósito de término del recorrido no necesariamente debe ser el mismo que el de partida.

A continuación se detallarán algunas de las características mencionadas.

2.2. Optimización de múltiples objetivos

Cuando hablamos de un problema de optimización de múltiples objetivos, nos referimos a aquellos que involucran la optimización simultánea de más de una función objetivo, para este caso, minimizar los costos de transporte y minimizar el descontento de los clientes respecto al nivel de servicio. De forma matemática, un problema de este tipo se puede expresar como

$$\text{Minimize } v(h) = \begin{bmatrix} v_1(h) \\ v_2(h) \\ \vdots \\ v_\sigma(h) \end{bmatrix}$$

donde *vi(h), i = 1,...,n* son las n funciones objetivos en el problema de objetivos múltiples. Este esquema es para funciones con los mismos parámetros, en caso de que no sea ese el

caso, se requiere algún tipo de modificación en el criterio de la función objetivo para obtener resultados adecuados.

Este tipo de problemas se puede resolver de dos maneras:

• Estableciendo una escala de prioridades de los objetivos y resolviendo el problema en el orden de las prioridades.

• Multiplicar una constante de "peso" con la función de cada objetivo y agruparlas en una suma.

Para DARPTW la segunda forma es más conveniente [16] puesto que en un sistema de transporte de su tipo, es más práctico tener un control detallado tanto de los factores del costo de operación como del nivel del servicio, que se traduce en captar mayor cantidad de clientes.

2.3. Influencia de la calidad del servicio

En [4] se menciona que existen dos tipos de ventanas de tiempo, las explícitas y las implícitas. Esta clasificación se relaciona directamente con la que se presenta en 2.5. Las explicitas son las más comunes, corresponden a las que se dan como entrada al problema, las que el cliente entrega y que se deben respetar para que el servicio prestado sea válido o adecuado según sea el caso. Las implícitas por otra parte, son más particulares, especialmente de sistemas de transporte como DARPTW. Surgen a causa de la necesidad de controlar las inconveniencias del cliente. Este generalmente proporciona una hora de entrega o de recogida esperada, lo que fija un extremo de una ventana. Como el cliente no quiere llegar tarde a su destino, el sistema que desarrolle el plan de transporte debe establecer el otro extremo de la ventana según corresponda. Con esto se tiene una ventana media abierta. Para prevenir que el transporte comience muy temprano respecto a su hora de entrega, o que la entrega se haga muy tarde respecto a su hora de recogida definida, se construye otra ventana de tiempo en la que el servicio se considera válido o adecuado. En 2.6 se discute cómo se construyen estas ventanas implícitas.

Al tener las ventanas de tiempo regulando la vialidad y calidad de las soluciones, sin duda que la complejidad del problema se incrementa considerablemente. A diferencia de otros tipos de problemas más generales, en DARPTW se trabaja con personas, por ello la calidad de servicio es un aspecto crítico a considerar como parte de los objetivos. En otros casos lo transportado son objetos. Más allá de la carga del vehículo, no es de mayor importancia el tiempo que vayan dentro de un vehículo, salvo excepciones puntuales, donde se transporten productos muy susceptibles al ambiente. Es por esto que en este tipo de problemas, las restricciones vinculadas al tiempo juegan un rol esencial.

2.4. Problema estático versus problema dinámico

Como en otro tipo de problemas, DARPTW puede existir tanto en una versión estática como en una versión dinámica. En este contexto, estático o dinámico depende de la cantidad de información que se conoce de antemano al abordar el problema. El caso en que toda la información que se tenga es conocida, se trata de un problema estático. Basta con que una parte de la información falte para que el problema se transforme en dinámico.

Los problemas estáticos son encontrados principalmente cuando se modela en un nivel de planeamiento táctico o estratégico. En estos casos los datos en el nivel operacional son construidos en base a información histórica o proyectada. El modelo es utilizado para revisar la gama de soluciones bajo en distintos escenarios, usando procesos de simulación especialmente. Para DARPTW, la información más relevante en este aspecto corresponde a las solicitudes de los clientes.

Los problemas dinámicos pueden ser formulados tras una serie discreta de problemas estáticos, donde el factor tiempo va impulsando la definición de nuevos problemas. Esta aproximación no suele ser la más efectiva (en [16] se menciona que en DARPTW efectivamente no lo es), a veces imposible de concretarse, por ello es que otros mecanismos son utilizados. Cuando se busca resolver modelos dinámicos reales y complejos, donde el tiempo computacional es crítico, es factible resolver el problema estático primero, y con el avance del tiempo y el cambio de los datos, aplicarle correcciones con heurísticas de reconstrucción apropiadas. Para DARPTW, las rutas de los vehículos van construyéndose

en tiempo real, acorde a nuevas peticiones de clientes, inconvenientes que se presenten, entre otros.

Para el caso dinámico, los métodos de solución deben ser lo suficientemente rápidos para que un cliente que está realizando una solicitud pueda tener una respuesta concreta y en un tiempo razonable. En el caso estático esto no es tan importante considerando que las peticiones se tienen de antemano.

Como se definió anteriormente, en esta investigación se aborda el caso estático de DARPTW.

2.5. Clientes “de entrada” y clientes “de salida”

Esta clasificación de clientes se vincula directamente a las características de la solicitud del servicio que realizan los clientes. Un problema DARP genérico suele considerar ambos tipos.

Los clientes “de entrada” (*inbound customers*) son aquellos que están en alguna locación, generalmente vinculada al término de alguna jornada en su rutina. Ellos desean ser recogidos en esa locación, a alguna hora específica, para ser trasladados hasta algún otro lugar, usualmente sus hogares. Es aceptable que haya una pequeña demora desde la hora que ellos especifican hasta la hora en que efectivamente llega el transporte a recogerlos, sin embargo no corresponde que este último llegue antes de la hora especificada, pues el cliente no está preparado aún para partir. El cliente no define una ventana de tiempo explicita para la llegada a su destino.

Los clientes “de salida” (*outbound customers*) por otro lado son aquellas personas que requieren llegar a alguna locación, como su lugar de trabajo, de estudio, etc., a una hora específica. En este caso, el cliente puede ser recogido a cualquier hora, es decir, no define una ventana de tiempo de recogida explicita, pero el vehículo que lo transporte debe llegar al lugar de destino necesariamente antes de la hora que el cliente requiere.

Como se mencionó en 2.3, aunque sólo se especifica una ventana de tiempo de parte del cliente, se deben definir ventanas de tiempo implícitas, según sea cliente de entrada o de

salida. En el caso de esta investigación, se trabajó con ambos tipos de clientes, especificando tanto el límite superior como el inferior de la ventana de tiempo correspondiente.

2.6. Ventanas de tiempo o Time Windows

Una ventana de tiempo corresponde a un intervalo de tiempo dentro del cual, del punto de vista de DARPTW, es válido o adecuado que un cliente sea recogido o entregado en una locación, según la operación que esté realizando en el momento el vehículo asociado. Que la situación sea válida o adecuada, dependerá del tipo de ventana de tiempo que se hable, podemos clasificarlas en ventanas de tiempo blandas y rígidas.

Cuando una ventana de tiempo es blanda (*soft time window*), es válido que un vehículo llegue fuera del tiempo que corresponde a la ventana. Esto implica que la solución pertinente no será considerada como inválida. Sin embargo, cuando se usa este tipo de ventanas, se suele agregar en términos de calidad de servicio un factor de influencia importante en la función objetivo. De esta forma, si bien la solución no será desechada, su calidad bajará en la medida que se no se respeten las ventanas de tiempo.

En el otro caso, una ventana de tiempo es rígida (*hard time window*) cuando su incumplimiento implica directamente que la solución asociada sea inválida.

La definición de las ventanas de tiempo, es decir, sus rangos inferiores y superiores, depende del tipo de cliente que se considere para el problema, más otro tipo de variables presentes. En lo que sigue, se considera a i como una solicitud de transporte particular, esto es, un cliente puntual.

2.6.1. Para clientes de entrada

En este caso las ventanas son establecidas en base al horario de recogida establecido por el cliente. Este tiempo a_i corresponderá al rango inferior de la ventana de tiempo de recogida del cliente. El rango superior de esta última, b_i, si no es especificado, se puede obtener con

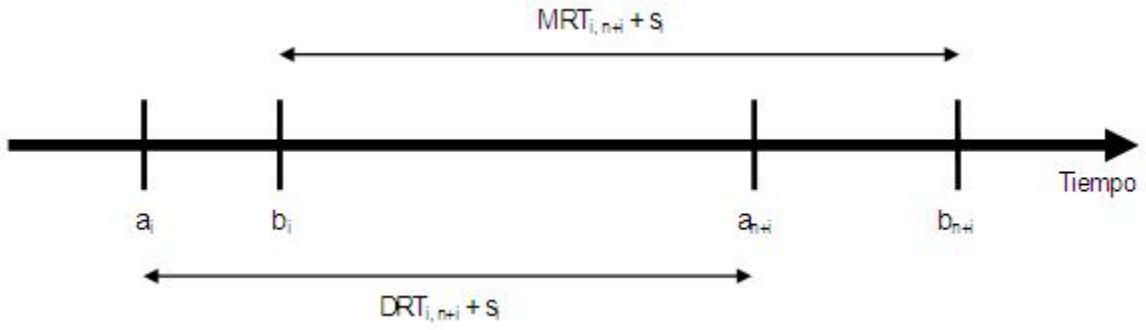
un tiempo límite *dev* establecido en el sistema, como una desviación máxima para la espera del cliente en la recogida, de esta forma, $b_i = a_i + dev$.

Para obtener el límite inferior de la ventana de tiempo de entrega a_{n+i}, se obtiene el tiempo más temprano en el que el cliente podría ser entregado a su lugar de destino, esto equivale al rango inferior de la ventana de recogida a_i, más el tiempo directo entre el punto de recogida y el punto de entrega, lo que se conoce como tiempo directo de viaje (*direct ride time* o *DRT*), y el tiempo de servicio del cliente *i*, s_i. De esta forma $a_{n+i} = a_i + s_i + DRT_{i,n+i}$.

El límite superior de la ventana de tiempo de entrega b_{n+i}, requiere de un parámetro adicional, que en la literatura suele encontrarse de dos formas. Por un lado, podemos considerar el máximo exceso en el tiempo de viaje *E* (*maximum excess ride time)*, el cual se puede obtener a través de una ecuación lineal simple dependiente del *DRT*, por ejemplo $E_{i,n+i} = 5\ min. + 0.5\ DRT_{i,n+i}$. Este valor es sumado a b_i y al tiempo de servicio para obtener el límite superior de la ventana de tiempo de entrega, esto es, $b_{n+i} = b_i + s_i + DRT_{i,n+i} + E_{i,n+i}$. Por otro lado, está el tiempo máximo de viaje *MRT* (*maximum ride time*) que es igual al anterior, pero incluye implícitamente a *DRT*. Para cumplir con esto, en la ecuación lineal *DRT* es multiplicado por un número mayor o igual que 1, por ejemplo $MRT_{i,n+i} = 8\ min. + 2\ DRT_{i,n+i}$. De esta forma, el límite superior de la ventana de tiempo de entrega sería $b_{n+i} = b_i + s_i + MRT_{i,n+i}$.

En la Figura 2.1 se muestra la construcción de las ventanas de tiempo en este caso, considerando el uso de *MRT* para el cálculo de límite superior de la ventana de entrega.

Figura 2.1 Construcción de las ventanas de tiempo para clientes de entrada.



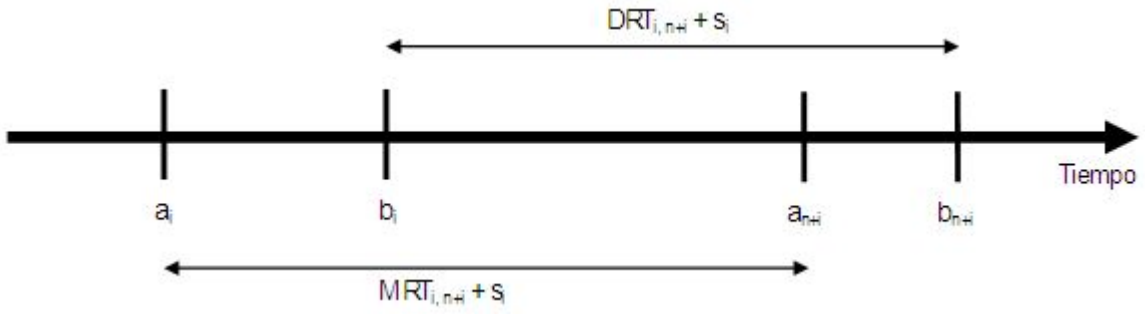
2.6.2. Para clientes de salida

Este caso es análogo al anterior, considerando que los límites se van obteniendo hacia atrás en el tiempo. Acá las ventanas son establecidas en base al horario de entrega establecido por el cliente. Este tiempo b_{n+i} corresponderá al rango superior de la ventana de tiempo de recogida del cliente. El rango inferior de esta última, a_{n+i}, si no es especificado, se puede obtener del valor *dev*, de esta forma, $a_{n+i} = b_{n+i}$ - *dev*.

Para obtener el límite superior de la ventana de tiempo de recogida b_i, se resta *DRT* y el tiempo de servicio a b_{n+i}, esto es $b_i = b_{n+i} - (s_i + DRT_{i,n+i})$. El límite inferior de la ventana de tiempo de recogida a_i, usando tanto *E* como *MRT*, resultar como $a_i = a_{n+i} - (s_i + DRT_{i,n+i} + E_{i,n+i})$ y $a_i = a_{n+i} - (s_i + MRT_{i,n+i})$ respectivamente.

En la Figura 2.2 se muestra la construcción de las ventanas de tiempo en este caso, considerando el uso de MRT para el cálculo de límite superior de la ventana de recogida.

Figura 2.2 Construcción de las ventanas de tiempo para clientes de salida.



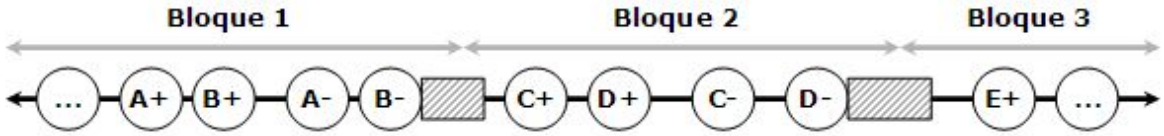
En esta investigación se consideró la construcción de ventanas, usando el factor MRT y con ventanas rígidas.

2.7. Estructura de las rutas

Una consideración adicional en términos de las características de DARPTW, corresponde a la planificación del recorrido de un vehículo. Este aspecto puede influir directamente en consideraciones de implementación de un sistema, por ejemplo, en lo que es la función objetivo. Acá se mencionarán dos aproximaciones al respecto.

Por una parte, la ruta que desarrolla un vehículo puede estar conformada por bloques de planificación. Un esquema de estos bloques se muestra en la Figura 2.3. Estos bloques se caracterizan por contener eventos tanto de recogida como de entrega de un conjunto de clientes en la ruta. Los bloques en sus extremos pueden tener depósitos de vehículos y tiempos de inactividad conocidos como slacks. En este contexto, se dice que un vehículo puede estar en tres estados distintos: En uno, está en un depósito, por lo que no ha comenzado su recorrido, o ya lo completó. El otro estado es de actividad, en el que se encuentra viajando para recoger o entregar pasajeros. Un último estado es de inactividad, en donde el vehículo está detenido, esperando un horario en que volverá a estar activo para servir pasajeros pertenecientes a un bloque posterior. En este contexto, un vehículo jamás estará inactivo con pasajeros a bordo, pues los *slacks* se ubican entre bloques.

Figura 2.3 Estructura de un bloque de planificación. Los cuadros grises son *slacks*.



Otra aproximación ya no considera de forma tan directa los bloques. Se permiten tiempos de inactividad aún con pasajeros a bordo. Esta forma de abordar las rutas es menos restrictiva que la anterior, pero tiene un costo importante en términos de calidad de servicio, que debe ser considerado en la función objetivo. Este costo es mesurado según el tiempo que los pasajeros permanecen dentro de un vehículo detenido.

Los tiempos de inactividad surgen cuando en una ruta, el viaje inmediato desde un evento origen a un evento destino implica que se llegue muy temprano a este último. Si bien estos slacks pueden parecer negativos desde el punto de vista del problema, su existencia agrega cierto grado de flexibilidad a las soluciones, y los *slacks* a su vez pueden ser útiles cuando se quiere reconstruir una ruta, por ejemplo con la inserción de un nuevo pasajero en ella.

En el contexto de esta investigación, se consideró la primera aproximación, en la que los tiempos de inactividad deben existir sólo con los vehículos vacíos.

2.8. Problemas relacionados

Como se ha mencionado, DARP es la especificación de una serie de problemas de transporte que comparten muchas características. A continuación se describirá de forma general los más importantes para considerar en esta investigación. Es importante destacar que para cada uno de ellos, la restricción de las ventanas de tiempo es considerada en versiones más particulares.

2.8.1. TSP: Problema del vendedor viajero

TSP hace referencia a un vendedor que recorre una serie de ciudades para ofrecer su mercancía personalmente. Parte desde su hogar, visita a todos sus clientes y retorna lo más temprano posible, según cómo terminen sus labores. Qué tan pesado sea su día de trabajo, dependerá del orden en que visite a los clientes en las distintas ciudades. Hay dos supuestos respecto a los lugares de recorrido que deben considerarse: por un lado el tiempo de viaje entre dos puntos no depende del tiempo (hora del día, por ejemplo), por otro lado el tiempo de viaje funciona "simétricamente", es decir, es lo mismo viajar desde A hasta B y desde B hasta A. Una condición que suele imponerse en el problema, es que el vendedor debe recorrer sólo una vez cada cuidad. El problema al final consiste en encontrar una ruta óptima que permita al vendedor completar su trabajo en el menor tiempo posible.

El problema puede ser generalizado a un vehículo que tiene que recorrer las ciudades, sólo una vez cada una y que la velocidad con que se desplaza depende de distintos factores en las rutas, como el estado de las mismas, el clima, etc. En este problema en vez de minimizar el tiempo de viaje, se minimiza la distancia recorrida. Si son N ciudades a recorrer, hay $(N-1)!$ posibles soluciones.

Adicionalmente existe el TSP con ventanas de tiempo (TSPTW), el cual es considerado como un problema *NP-Hard*. Este problema es usado para comprobar que DARPTW es *NP-Hard* también [1].

2.8.2. VRP: Problema del enrutamiento de vehículos

En VRP, se agregan un conjunto de restricciones y consideraciones importantes a TSP. En este problema se tiene un conjunto de vehículos idénticos en capacidad. Ellos inician y terminan su recorrido desde un depósito. Puede haber uno o más depósitos, en los que se encuentra un conjunto del total de vehículos. Estos se encargan de recoger bienes desde vendedores y trasladarlos a clientes. Un cliente puede ser servido por un sólo vehículo, por lo que la entrega debe ser completamente realizada por dicho vehículo. Pueden existir límites superiores en el largo de los recorridos de los vehículos y de los bienes demandados por los clientes o los bienes ofrecidos por los vendedores. De esta manera el problema consiste en buscar la mejor forma de trasladar a los vendedores hacia los clientes, en términos de minimizar la distancia recorrida por cada vehículo tras asignar vendedores a los vehículos y ordenar la forma en que visitan a sus clientes, considerando todas las restricciones presentes.

2.8.3. PDP: Problema de la recolección y entrega

PDP se puede considerar como una versión particular de VRP. De hecho es conocido también como el problema de enrutamiento de vehículos con recolección y entrega (*Vehicle routing problem with pickup and delivery*). La diferencia fundamental es que para cada solicitud de transporte existe una locación de origen y una de destino, por ende se requieren rutas adecuadas para la recolección y entrega de los bienes transportados.

2.9. Modelo matemático

A continuación se mostrará un modelo matemático de DARPTW que sirve para ejemplificar la formalización del mismo. Este modelo trabaja con ventanas de tiempo rígidas, lo que se aproxima a las necesidades que en la presente investigación se contemplaron. Un detalle completo del mismo se puede obtener en [16].

El modelo se basa en los siguientes datos:

- Se tienen n de solicitudes de servicio. Para una petición, i es el punto de recogida y $n+i$ corresponde al punto de entrega. d_i corresponde a la cantidad de pasajeros a transportar en la solicitud i.
- $P = \{1,\ldots, n\}$ y $D = \{n+1,\ldots, 2n\}$ son el conjunto de puntos de recogida y de entrega respectivamente.
- $N = \{P \cup D\}$ es el conjunto total de puntos especificados en las solicitudes.
- Para cada vehículo $k \in K$, se tienen dos depósitos, el de origen $o(k)$ y el de destino $d(k)$.
- $A = N \cup \{o(k), d(k)\}\ \forall k \in K$ es el conjunto total de puntos del problema.
- $V \subset K$ es el conjunto de los vehículos usados efectivamente en la solución y v es su cantidad.
- C^k corresponde a la capacidad máxima del vehículo k.
- $l_i = d_i$ corresponde a la diferencia de carga en el nodo i y $l_{n+i} = -d_i$ corresponde a la diferencia de carga en el nodo $n+i$.
- L_i^k corresponde a la carga del vehículo k después de visitar el nodo i.
- Cada nodo i debe ser servido en la ventana de tiempo $[a_i, b_i]$.
- T_i^k corresponde al tiempo inicial de servicio del vehículo k en el nodo i.
- $t_{i,j}$ corresponde al tiempo de conducción entre el nodo i y el nodo j.
- s_i corresponde al tiempo de servicio en el nodo i.
- $x^k_{i,j}$ es una variable de decisión con valor 1 si el vehículo k sirve al nodo i y después conduce para servir al nodo j. En caso contrario, la variable es 0.

Considerando la técnica de multiplicar constantes mencionada en 2.2 (en este caso α, β y γ), se define la siguiente función objetivo:

$$\min \alpha \sum_{k \in V} \sum_{(i,j) \in A} t_{i,j} x^k_{i,j} + \beta v + \gamma \sum_{k \in V} \sum_{i \in P} (T^k_{n+i} - s_i - T^k_i)$$

Como se ve, son tres los elementos que se miden en esta función. Lo que esta multiplicado con α corresponde al factor del tiempo de viaje de todos los vehículos en la solución. Lo multiplicado con β es simplemente la cantidad de vehículos usados en la solución. Los dos factores anteriores están enfocados a los costos de transporte en sí. El tercer elemento, multiplicado por γ corresponde al tiempo de viaje total de los pasajeros sobre los vehículos. Naturalmente, este es el factor vinculado a la calidad de servicio.

A su vez, la función objetivo está sujeta a las siguientes condiciones:

$$(1)\quad \sum_{k\epsilon V}\sum_{j\epsilon P\cup d(k)} x^k_{o(k),j} = v$$

$$(2)\quad \sum_{k\epsilon V}\sum_{i\epsilon D\cup o(k)} x^k_{i,d(k)} = v$$

$$(3)\quad \sum_{k\epsilon V}\sum_{j\epsilon A} x^k_{i,j} = 1, \forall i \epsilon N$$

$$(4)\quad \sum_{j\epsilon N} x^k_{i,j} - \sum_{j\epsilon N} x^k_{j,n+i} = 0, \forall k \epsilon V, i \epsilon P$$

$$(5)\quad x^k_{i,j}\left(T^k_i + s_i + t_{i,j} - T^k_j\right) \leq 0, \forall k \epsilon V, (i,j) \epsilon A$$

$$(6)\quad a_i \leq T^k_i \leq b_i, \forall k \epsilon V, i \epsilon A$$

$$(7)\quad T^k_i + s_i + t_{i,n+i} \leq T^k_{i+n}, \forall k \epsilon V, i \epsilon P$$

$$(8)\quad x^k_{i,j}\left(L^k_i + l_j - L^k_j\right) = 0, \forall k \epsilon V, (i,j) \epsilon A$$

$$(9)\quad l_i \leq L^k_i \leq C^k, \forall k \epsilon V, i \epsilon P$$

$$(10)\quad L^k_{o(k)} = L^k_{d(k)} = 0, \forall k \epsilon V$$

$$(11)\quad x^k_{i,j} \epsilon \{0,1\}, \forall k \epsilon V, (i,j) \epsilon A$$

(1) y (2) obligan a que el número de vehículos de abandona un deposito y el número de vehículos que llega a ellos sea igual al número de vehículos considerados en la solución.

(3) y (4) son validaciones referentes a los clientes que son servidos por los vehículos. En (3) se asegura que todas las solicitudes de los clientes del problema son atendidas y en (4) se establece que cuando un cliente es recogido por un vehículo, debe ser entregado por el mismo.

(5) es una validación respecto a las rutas que considera la solución. Para cada uno de los desplazamientos desde un nodo a otro en la solución, el tiempo de salida de un nodo de origen debe estar antes que el tiempo de llegada al nodo destino.

(6) es la validación referente a las ventanas de tiempo.

(7) es igual que (5), pero en el contexto de cualquier par de nodos que conforme una solicitud de servicios de parte de un cliente.

(8) asegura que entre dos eventos de una ruta solución, la carga del vehículo sea correctamente calculada.

(9) asegura que la capacidad de un vehículo nunca sea superada.

(10) es una validación respecto a la carga inicial y final de cada vehículo en su recorrido, las que deben ser 0.

(11) son las variables usadas en el modelo.

3. Algoritmos Genéticos

En el presente capítulo se abordará el otro tema importante para esta investigación, el concepto de Algoritmo Genético (GA). Este es un tema muy extenso para ser abordado en profundidad, hay muchos aspectos, teóricos especialmente, que pueden ser analizados en gran detalle, pero ello no se corresponde con los objetivos planteados en este trabajo. Acá se pretende examinar a los GA como herramienta para la solución a problemáticas complejas, para de esta forma, tener un acercamiento más directo a su aplicación en DARPTW.

Se examinarán los GAs desde tres enfoques distintos: Por un lado se analizarán los conceptos más básicos, esto es, su estructura y funcionamiento, lo que lleva a revisar en qué consiste el GA más simple. Por otro lado, se verá una serie de conceptos asociados a la teoría más elemental de los GAs, los que han ido tomando importancia con el tiempo, gracias a la investigación y las aplicaciones impulsadas en distintos tipos de problemas. Estos conceptos, en los estudios más recientes, juegan roles esenciales en el contexto de la construcción de variantes de GAs cada vez más eficientes y flexibles. En última instancia, se revisarán las características principales de algunas de estas variantes.

3.1. Conceptos básicos

3.1.1. Inicios

Los GAs fueron inventados y desarrollados por John Holland y sus estudiantes de la Universidad de Michigan en los años 60 y 70. Más que un tipo de algoritmo que sirviera para resolver problemas específicos, la intención de Holland era estudiar el fenómeno de la adaptación natural y desarrollar vías por las cuales se pudieran importar estos mecanismos a sistemas computacionales. En 1975 Holland realizó una publicación [5] en la que se presentó a los GAs como una abstracción de la evolución biológica y entregó un marco de trabajo teórico para la adaptación natural bajo los GAs.

El algoritmo propuesto por Holland se puede describir como un método para moverse desde una población de "cromosomas" (que pueden considerarse como strings binarios) a una nueva población, usando cierto tipo de "selección natural" con el uso de operadores inspirados en la genética, como son el cruzamiento (*crossover*), mutación (*mutation*) e inversión (*inversion*). Cada cromosoma está compuesto de "genes" (por ejemplo un bit), los que a su vez pueden ser una instancia particular de un "allele" (por ejemplo, un 0 o 1). Un operador de selección escoge en la población los cromosomas que están más capacitados para reproducirse, logrando que en promedio los cromosomas de mayor calidad generen nuevos individuos para la población, en contraste a los menos capaces. El cruzamiento intercambia partes de un cromosoma, imitando de cierto modo la recombinación biológica entre dos organismos cromosómicos simples. La mutación cambia de forma aleatoria el allele de ciertas partes de los cromosomas. La inversión revierte el orden de una sección continua de un cromosoma, cambiando el orden de los genes respectivos.

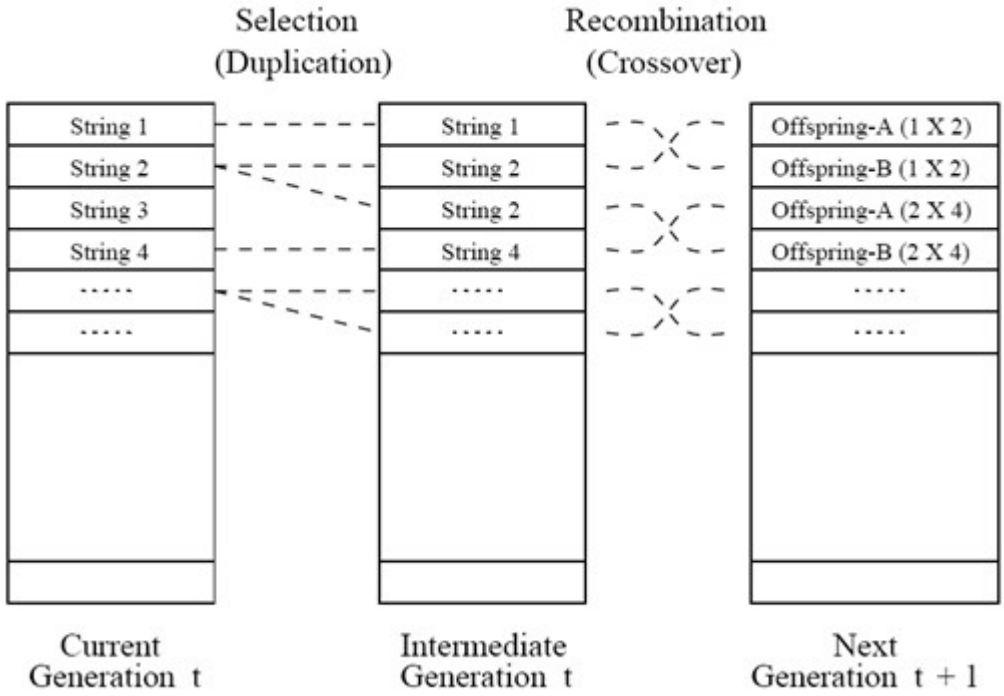
3.1.2. GA canónico

En esta sección se hará una descripción general del funcionamiento del denominado en [17], GA "canónico".

La primera etapa del GA consiste en generar una población inicial (generalmente de forma aleatoria) de individuos, que corresponden a los cromosomas o "genotipos". Cada uno de ellos es evaluado y se le asigna una "calidad" (*fitness*) determinada. En este contexto, la función de evaluación corresponde a una medida de comportamiento que se obtiene en base a una serie de parámetros, mientras que la función de calidad toma dicho valor para transformarlo en una medida que representa posibilidades de reproducción. La función de evaluación es considerada independientemente para cada individuo, no así la de calidad.

De cierto modo, un GA puede considerarse como un proceso de dos etapas. La primera comienza con la población actual y tras un proceso de selección se genera una población intermedia, en la que se realizan distintas operaciones de cruzamiento (o recombinación) y mutación, obteniendo la que es la siguiente población. Una generación es el paso desde una población a otra tras el proceso descrito. El mismo se grafica en la Figura 3.1.

Figura 3.1 Paso desde una generación a otra en el GA canónico [17].



En el GA canónico, la calidad de un individuo se calcula con la expresión f_i / F, donde f_i corresponde al resultado de la evaluación y F corresponde al promedio de los resultados de evaluación en la población actual. Con esta calidad calculada, se realiza la selección considerando que la probabilidad de que un individuo pase a la generación intermedia es proporcional a su calidad. En 3.1.4 se revisarán algunos mecanismos de selección.

Con la población intermedia mezclada adecuadamente, se procede a realizar la recombinación. La operación de cruzamiento se realiza sobre un par de cromosomas elegidos aleatoriamente de la población intermedia, con una probabilidad P_c. Supongamos que se tienen los cromosomas:

1101 | 010111

0100 | 011010

Usando cruzamiento en un punto (One-point crossover), para este ejemplo, en la posición 5, se obtienen dos nuevos cromosomas:

Hijo 1: **1101 | 011010**

Hijo 2: **0100 | 010111**

Estos formarán parte de la nueva generación.

Ya realizada la recombinación, se puede realizar la mutación. Para cada bit en cada miembro de la nueva población, hay una probabilidad P_m de que ocurra mutación, lo que generalmente se traduce en cambiar el bit de 0 a 1 o viceversa.

Con este proceso completo ya se tiene la nueva generación, lista para ser evaluada y recomenzar las etapas de selección, recombinación y mutación.

3.1.3. Codificación

Un elemento esencial en la construcción de GAs es la codificación que se utilice. El cromosoma, más que un simple arreglo de bits, funciona como la representación de una solución para el problema que está abordando el GA. Así como se formula un mecanismo para llevar la información de parámetros, variables o cualquier elemento que tiene alguna relación con el problema en cuestión, debe existir un mecanismo adecuado para extraer esa información.

En la práctica, la codificación va muy de la mano con la función de evaluación. Cuando se consideran los parámetros del problema, su codificación dentro del cromosoma debe ser adecuada para asegurar que al ser decodificados, la evaluación pueda realizarse eficientemente. Por ejemplo, si se quiere representar una variable con un rango posible de 1200 valores, se necesitan al menos 11 bits para cubrir este rango, sin embargo hay otros 848 valores que no se usarán para la evaluación, o se usarán incorrectamente.

No sólo una codificación binaria es posible, otros tipo de representación, ya sea con números binarios, caracteres, etc., se han estudiado y evaluado. En [17] y [18] hay breves reseñas respecto a este punto, donde se destaca que hay opiniones tanto a favor como en contra de usar representaciones distintas a la binaria tradicional. Sin embargo, al final no existen reglas claras para definir cuando una representación distinta puede resultar efectiva o no, o resulta muy dependiente del problema.

3.1.4. Selección

La selección es el proceso mediante el cual un subconjunto de individuos en una generación es escogido en razón de su calidad para formar parte de otra generación y para engendrar nuevos individuos. Se pueden describir tres mecanismos conocidos:

• **Selección proporcional a la calidad** (*fitness proportionate selection*): también conocida como selección de la ruleta (*ruolette-wheel selection*), se basa en asignar una probabilidad de selección a cada individuo de acuerdo a su calidad. Esto permite que aunque los individuos con alta calidad tengan altas posibilidades, no necesariamente serán seleccionados. Lo mismo en el caso de los individuos con baja calidad, igualmente tienen posibilidades de ser seleccionados, esto es favorable para la variedad y para evitar la convergencia prematura.

• **Selección por torneo** (*tournament selection*): se trata de seleccionar un conjunto de individuos en la población y escoger los mejores entre ellos para ser parte de futuras generaciones y procesos de recombinación. En este tipo de selección, el tamaño del conjunto seleccionado es un factor relevante. En el torneo, se va evaluando con una probabilidad p cada individuo en orden de calidad, si es seleccionado o no para recombinación. Una selección por torneo se define determinista si $p = 1$.

• **Muestreo estadístico de resto** (*remainder stochastic sampling*): Considerando la expresión de calidad f_i / F de 3.1.2 cuando este valor es mayor que 1, la porción entera de este número indica cuantas copias del cromosoma respectivo son copiadas a la población intermedia. Todos los cromosomas pondrán copias en la población intermedia con probabilidad equivalente a la parte fraccionaria de f_i / F. Por ejemplo, si este valor es 1.36, se coloca una copia y hay una probabilidad de 0.36 de que coloque otra.

Otros mecanismos de selección pueden ser revisados en [18].

3.1.5. Recombinación o cruzamiento

En la recombinación, dos individuos generan un tercero mezclando copias de sus genes en el nuevo individuo. El mecanismo más tradicional de cruzamiento es el de un punto,

mostrado en 3.1.2. Este mecanismo tiene una desventaja, en el sentido de que dependiendo de cómo estén ordenados los bits en el cromosoma, puede romper la composición de un "esquema" (*schema*) presente en él. Como se verá posteriormente, la mezcla en recombinación de esquemas de bajo orden pueden ayudar a encontrar soluciones de forma más efectiva, por lo que romper su estructura en el cromosoma no es algo conveniente. Bajo este contexto, es que se han desarrollado otro tipo de recombinaciones. Describiremos dos de las más conocidas:

• **Cruzamiento en dos puntos** (*double-point crossover*): De forma análoga al cruzamiento en un punto, en este mecanismo se escogen dos puntos en los cromosomas y los segmentos entre los puntos seleccionados se combinan en los nuevos individuos. Por ejemplo, si se tienen los siguientes cromosomas:

1101 | 0111 | 00001

0101 | 0101 | 00011

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

Hijo 1: **1101 | 0101 | 00001**

Hijo 2: **0101 | 0111 | 00011**

• **Cruzamiento uniforme** (*Uniform crossover*): En este mecanismo, los genes en cada posición (*locus*) de los cromosomas padres se comparan. Con una probabilidad constante, el allele de ambos genes se intercambia. Una variante a este método de cruzamiento es el medio cruzamiento uniforme (*half uniform crossover*), donde exactamente la mitad de los genes distintos se intercambian.

3.1.6. Mutación

Este operador corresponde a la modificación de genes particulares de un cromosoma bajo una probabilidad muy baja de ocurrencia. La función 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.

Pese al mecanismo sencillo con que trabaja la mutación, resulta ser un elemento muy importante en el funcionamiento del GA. Varios estudios se han enfocado en comprobar la influencia de este operador en la resolución de problemas complejos, en [18] se habla de casos comparativos respecto a las capacidades de la recombinación. Sin embargo, se destaca que el balance entre mutación, selección y cruzamiento es lo importante.

3.2. Consideraciones teóricas fundamentales

Una revisión de la literatura en torno a los GAs nos permite ver la gran cantidad de variaciones e improvisaciones que se han desarrollado respecto a las versiones más básicas. Todo este esfuerzo investigativo tiene como gran objetivo el mejorar la efectividad y las prestaciones que los GA entregan para la solución de problemas cada vez más complejos. Desde que Holland presentó los GA al mundo científico, es que se definieron una serie de conceptos y teorías importantes para comprender los caminos por los cuales se debía profundizar para lograr GAs cada vez más competentes. Así es como el *schema theorem* y la notación de *building blocks*, establecen los pilares fundamentales para el entendimiento del comportamiento de los GA, el concepto *deception* ayuda a definir el tipo de problema con que se lidia y el concepto *linkage* nos permite establecer pautas para evaluar la eficiencia de los GA.

En esta sección se revisará de forma general dichos conceptos, se verá la importancia en su consideración al implementar soluciones para problemas de alta complejidad, lo que permitirá, ya en el contexto de esta investigación, tener un acercamiento más claro a los requerimientos de GAs para abordar DARPTW.

3.2.1. Espacios de búsqueda e hiperplanos

El comportamiento computacional de los GAs puede ser explicado como una compleja y robusta búsqueda vía muestreo implícito de particiones de hiperplanos en el espacio de búsqueda. Para entender o visualizar que son las particiones de hiperplanos, podemos

considerar un espacio de búsqueda de tres dimensiones. Se asume que se tiene una codificación de tres bits, lo que se puede representar como un cubo en cuyo origen reside el string 000. En las demás esquinas se tienen otros strings, donde cada esquina adyacente varía en un bit respecto a otra. Esto se grafica en la parte superior de la Figura 3.2. En el frente del cubo se encuentran todos los strings que empiezan con 0. Si "*" es un carácter "neutro", o en otras palabras, reemplazable por cualquier bit, el plano puede representarse como "0**". Un string que contiene * se denomina *schema*. Cada *schema* corresponde a un hiperplano en el espacio de búsqueda. El "orden" del hiperplano corresponde al número de bits que aparecen fijos en el *schema*. De esta forma, "0**" tiene orden 1 y "0***10***" tiene orden 3.

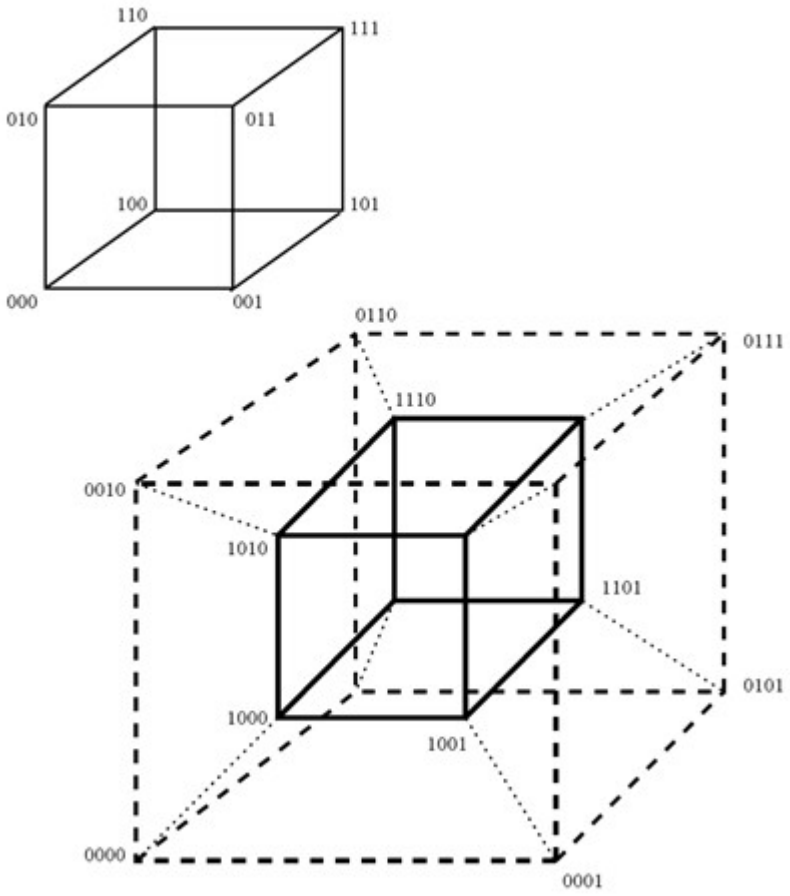
La parte inferior de la **¡Error! No se encuentra el origen de la referencia.** muestra un espacio de cuatro dimensiones, representado por la incrustación de un cubo dentro de otro. Se etiquetan las esquinas del cubo interior y del exterior idénticamente al cubo de tres dimensiones. Después se inserta un 1 al principio de los strings del cubo interior y un 0 en los del exterior. Esto mantiene el formato de variación de un sólo bit entre esquinas adyacentes. De esta forma el cubo interior corresponde al hiperplano 1*** y el exterior al hiperplano 0***. Además el *schema* *0** representa los planos frontales de cada cubo y el hiperplano de orden 2 10** representa la parte frontal del cubo interior.

Un string es parte de un hiperplano (y la correspondiente partición) particular, si el mismo puede obtenerse al reemplazar en el *schema* asociado los valores marcados con "*" con los bits que correspondan. Un cromosoma con representación binaria corresponde a una esquina del hipercubo y es parte de $2^L - 1$ hiperplanos distintos, con L el largo del cromosoma (El *schema* conformado sólo por "*" no se considera, pues representa al espacio de búsqueda en sí). Esto se obtiene al considerar todas las posibilidades de reemplazar uno o más bits en el string con "*".

Adicionalmente, se tiene que para un cromosoma con representación binaria, se tienen $3^L - 1$ posibles particiones de hiperplanos, con L el largo del cromosoma. Para cada elemento del string, se puede tener el valor 0, 1 y "*", que resultan en las 3^L combinaciones.

La noción de población en el mecanismo de búsqueda de los GAs, se vuelve importante al considerar que no es muy práctico examinar un individuo de forma aislada, que es parte de $2^L - 1$ particiones de hiperplanos distintas. Una población de distintas muestras de individuos entrega información acerca de numerosos hiperplanos. Además, hiperplanos de bajo orden son representados por un gran número de individuos. En este contexto, el concepto de "paralelismo implícito" (*implicit parallelism*) nace al considerar el hecho que numerosos hiperplanos se muestrean al evaluar una población de individuos, de hecho son mucho más los hiperplanos en muestreo que la cantidad de individuos. La evaluación de un sólo individuo ya implica que se evalúen implícita y paralelamente un conjunto de hiperplanos, sin embargo el efecto acumulativo de evaluar una población de individuos es el que permite obtener información estadística acerca de algún conjunto de hiperplanos.

Figura 3.2 Representación de dos espacios de búsqueda [17].



El concepto de paralelismo implícito sugiere que numerosas competencias entre hiperplanos son resueltas en paralelo. A través de los procesos de reproducción y recombinación, la representación del *schema* de hiperplanos en competencia aumenta o disminuye de acuerdo a la calidad relativa de los individuos pertenecientes a esas particiones de hiperplanos. De esta forma, en una población se puede ir evaluando la representación proporcional de un *schema* correspondiente a una partición de hiperplanos particular e indicar si esta aumenta o disminuye en el tiempo, cuando la selección basada en calidad se combina con la recombinación realizada entre individuos para generar nuevos miembros en la población.

3.2.2. El teorema del esquema (*Schema theorem*)

Sea *M(H,t)* el número de strings en una población que forman parte del hiperplano *H* en la generación *t*. Además *(t + intermediate)* indica la generación después de la selección, pero antes de mutación y cruzamiento, y *f(H,t)* es la evaluación promedio de la muestra de strings en *H* para la población actual. $\bar{f}$ corresponde a la evaluación promedio de toda la población. Formalmente, el cambio en la representación asociada a los strings que se obtienen del hiperplano *H* se expresa como:

$$M(H, t + intermediate) = M(H, t)\frac{f(H, t)}{\bar{f}}.$$

De esta fórmula, se puede observar que la cantidad de strings en *H* en el tiempo depende del ratio formado entre el valor promedio de la evaluación de dichos strings respecto al valor promedio de la evaluación de todos los strings en la población. Al realizar una mera copia de los strings, no se muestrean nuevos hiperplanos pues no se generan nuevas muestras. Si bien, en teoría se pretende tener una muestra de nuevos individuos con la misma distribución, en la práctica no es posible. Sin embargo la recombinación y mutación proveen modos de generar nuevas muestras, preservando parcialmente la distribución de strings entre los hiperplanos de la generación intermedia.

Ahora consideraremos los efectos de la recombinación para calcular *M(H,t)*. En teoría, este operador tiene efectos tanto constructivos (*gains*) como destructivos o disruptivos

(*losses*) respecto a la representación de los individuos asociados a *H*. Como después se mencionará, un cruzamiento eventualmente puede destruir la estructura del *schema* asociado a un hiperplano. Debemos considerar que la recombinación es aplicada a una parte de la población, para la otra parte, la representación se mantiene. De esta manera, la formula anteriormente planteada se puede expresar como:

$$M(H,t+1) = (1-p_c)M(H,t)\frac{f(H,t)}{\bar{f}} + p_c\left[M(H,t)\frac{f(H,t)}{\bar{f}}(1-losses) + gains\right]$$

P_c corresponde a la probabilidad de que un individuo sea sometido a recombinación. A estas alturas, se realizan dos supuestos algo conservadores:

• Se asume que un cruzamiento dentro del "largo definitorio" (*defining length*) de un *schema* es siempre disruptivo. El largo definitorio (ΔH) corresponde en términos cuantitativos a la diferencia entre la posición de la última instancia de un bit respecto a la primera en el *schema*. Por ejemplo, el largo definitorio del *schema* ***01*01*1** sería $\Delta H = 6$. En la práctica, este supuesto no es siempre efectivo. Consideremos el schema 11*****. Si un string como 1110101 se recombina entre el primero y el segundo bit con un string como 1000000 o 0100000, no se genera una disrupción en el schema 11*****, pues uno de los individuos que resultan se mantiene dentro de ese hiperplano.

• Los efectos constructivos se asumen como nulos. Este supuesto también es impreciso. En base al mismo ejemplo anterior, para el caso de los strings 1000000 y 0100000, si se recombinaran entre el primer y segundo bit, se obtendría un nuevo individuo perteneciente al *schema* 11*****.

De esta forma, tenemos la siguiente expresión (desigualdad):

$$M(H,t+1) \geq (1-p_c)M(H,t)\frac{f(H,t)}{\bar{f}} + p_c\left[M(H,t)\frac{f(H,t)}{\bar{f}}(1-disruptions)\right]$$

donde el efecto de la disrupción esta sobreestimado. También se considera una excepción en este contexto: dos strings pertenecientes al mismo hiperplano *H*, en recombinación no generan disrupción. Sea *P(H,t)* la representación proporcional de H obtenida al dividir

M(H,t) por el tamaño de la población. La probabilidad que un individuo aleatoriamente elegido pertenezca a *H* es *P(H,t)*. Para el cruzamiento en un punto, $\Delta H / L - 1$ es una medida directa de cómo el punto puede estar dentro del largo definitorio de un *schema*, con *L* el largo del cromosoma. Considerando esto, la disrupción se puede obtener por:

$$\frac{\Delta(H)}{L-1}(1 - P(H,t)).$$

A estas alturas, la desigualdad anterior puede simplificarse respecto a P_c. Ambos lados pueden dividirse por el tamaño de la población, para obtener el término *P(H,t+1)*, la representación proporcional de *H* en la generación *t+1*. De este modo se obtiene:

$$P(H,t+1) \geq P(H,t)\frac{f(H,t)}{\bar{f}}\left[1 - p_c\frac{\Delta(H)}{L-1}(1 - P(H,t))\right]$$

En general ambos padres son elegidos en base a su nivel de calidad. Eso se puede agregar a la expresión, indicando que el padre alternativo es escogido desde la población intermedia después de la selección:

$$P(H,t+1) \geq P(H,t)\frac{f(H,t)}{\bar{f}}\left[1 - p_c\frac{\Delta(H)}{L-1}(1 - P(H,t)\frac{f(H,t)}{\bar{f}})\right]$$

Por último, se debe considerar la mutación. Sea *o(H)* la función que nos indica el orden del hiperplano *H*. Esto último es la cantidad de 0 o 1 que en el *schema* asociado a *H* se encuentren. Sea P_m para probabilidad de mutación, donde está siempre cambia el bit correspondiente. De este modo, la probabilidad de que la mutación afecte al *schema* asociado a *H* es $(1-P_m)^{o(H)}$. Con esto, el *schema theorem* puede definirse finalmente con la siguiente expresión:

$$P(H,t+1) \geq P(H,t)\frac{f(H,t)}{\bar{f}}\left[1 - p_c\frac{\Delta(H)}{L-1}(1 - P(H,t)\frac{f(H,t)}{\bar{f}})\right](1 - p_m)^{o(H)}$$

El *schema theorem* representa el teorema fundamental de los GAs. Básicamente entrega un límite inferior respecto a lo que es el cambio del muestreo para un hiperplano simple desde una generación *t* a otra *t+1*. Permite comprender el efecto de la mejora en la evaluación de los individuos desde poblaciones más antiguas a las más nuevas. Cuando un hiperplano cumple con $f(H,t) > \overline{f}$, es decir, corresponde a una partición con un promedio de evaluación (por ende calidad) alto en la población, los individuos que sean parte de ella proliferarán mejor con el avance de las generaciones respecto a los que tienen un nivel más bajo.

Pese a la trascendencia del teorema, es importante recalcar ciertas inexactitudes que presenta:

• Al no considerar las consecuencias constructivas de la recombinación y sobreestimar las disruptivas, gran cantidad de información relevante puede perderse. Por ende su aplicación en la observación del muestreo de los hiperplanos no siempre resulta muy efectiva.

• La calidad observada (efectiva) de un hiperplano *H* en el tiempo *t*, puede cambiar dramáticamente si la población concentra sus nuevas muestras en más subparticiones especializadas del hiperplano. Por ello, las observaciones del hiperplano en este aspecto son sólo relevantes en las primeras generaciones, mientras más avance el tiempo, el teorema reflejará más inexactitudes.

En [17] se mencionan estas problemáticas y se describen algunos modelos más exactos que las abordan.

3.2.3. Bloques de construcción (*building blocks*) y acoplamiento (*linking*)

En el contexto de la representación en un GA, se conoce como "bloque de construcción" (*building block*) a la subestructura del cromosoma que le permite ser asociado a un *schema*, o formar parte como individuo del hiperplano correspondiente. Un GA opera en torno a los bloques, permitiendo que su número aumente y mezclando unos con otros, para poder obtener soluciones del problema abordado.

En este sentido, como la estructura de un bloque de construcción está directamente asociada a un *schema*, en el proceso de recombinación se da la posibilidad que un bloque sea

destruido. Esta posibilidad está altamente influenciada por el tipo de operador usado para la recombinación. Supongamos que tenemos dos *schemas* de orden 2 con estructuras del siguiente tipo: 11********** y 1**********1. En el caso de cruzamiento en un punto, para el primer *schema* hay una probabilidad de *1 / L – 1* de romper su estructura, con *L* el largo del string. Sin embargo, para el segundo *schema* hay una probabilidad de *L – 1 / L – 1 = 1.0*. Como se aprecia, la posición de los bits fijos en el *schema* afecta mucho el efecto del cruzamiento en un punto sobre su estructura. El cruzamiento en dos puntos funciona mejor en este aspecto: el *schema* se puede representar como un anillo (Figura 3.3).

Figura 3.3 Representación de bits como anillo [17].

Acá las posiciones de los bits en el string van desde b1 hasta b12. El cruzamiento en un punto de este modo se puede considerar como un caso particular del cruzamiento en dos puntos, cuando uno de los puntos escogidos está entre b1 y b12. Las mayores probabilidades de romper el esquema ocurren cuando los bits fijos en el *schema* están en posiciones contrarias del anillo. Sin embargo, se puede apreciar que para los dos *schemas* señalados previamente, la representación en anillo es la misma, al contrario de lo que pasa en el caso del cruzamiento en un punto. Como se ve, la posición de los bits en los *schema* tiene alta influencia en sus probabilidades de ser destruidos en recombinación. Se dice que tienen una "representación compacta" (*compact representation*) en el *schema* si están cerca unos de otros, lo que disminuye claramente la disrupción en la recombinación.

En este contexto, nace el concepto de "acoplamiento" (*linkage*), como el fenómeno en que un conjunto de bits actúan como "alleles coadaptados" y tienden a ser heredados como grupo. El acoplamiento es una generalización de la representación compacta de los bits, en el sentido de que esta última depende de la cercanía que los mismos tengan. Sin embargo, el acoplamiento no necesariamente puede generarse entre bits cercanos. Un bloque de construcción puede estar distribuido en el cromosoma y estar sometido a un grado de acoplamiento. Esto último dependerá de la representación en juego. Por ejemplo, en varias

aplicaciones de GAs se puede encontrar la idea de representación independiente de la posición. En esta se tiene que los genes se representan por un vector (posición, allele). Por ejemplo, se tiene el siguiente cromosoma bajo esta representación:

((9 0) (6 0) (2 1) (7 1) (5 1) (8 1) (3 0) (1 0) (4 0))

El cromosoma representa al string 010010110. En él se pueden aplicar operadores para cambiar el grado de acoplamiento observable en los distintos bloques que lo conforman. Un operador clásico en este contexto es la "inversión" (*inversion*) que consiste en tomar un par de genes e intercambiarlos. En ese caso el nivel de acoplamiento puede cambiar, pero el cromosoma mantiene la misma codificación. En este tipo de operaciones la dificultad natural que nace es que en la recombinación pueden resultar cromosomas con posiciones repetidas u omitidas.

Adicionalmente, es importante considerar como se ve afectada la estructura de un bloque de construcción respecto al orden del *schema* (que es equivalente al orden del bloque) al que este asociado, cuando es manipulado en operaciones de recombinación. Por lo discutido anteriormente, es natural pensar que al tener bloques de orden muy alto, son mayores las probabilidades de que su estructura se vea alterada. Analizando esto del punto de vista del *schema theorem*, el grado de disrupción se ve aumentado directamente pues aumentan las "perdidas" que se generan en el proceso de recombinación, lo que en consecuencia disminuye la representación en la población de hiperplanos cuyos *schemas* presentan niveles altos de calidad en su evaluación. Por esto se puede concluir lo que en muchos puntos de la literatura se define como el criterio clave para asegurar la eficiencia y competencia de un GA: **El diseño del algoritmo debe facilitar la mezcla de bloques de construcción de bajo orden y que presenten altos niveles de calidad en su evaluación**. Así mismo, el grado de acoplamiento en la estructura de estos bloques debe ser asegurado.

3.2.4. Problemas "engañosos" (*deceptive problems*)

El termino *deception* se incorporó al contexto de los GAs para clasificar cierto tipo de problemas que resultan difíciles de abordar bajo la estrategia de utilizar bloques de orden bajo y de alta calidad. En sí, un bloque de estas características representa un largo

hiperplano genérico en el espacio de búsqueda, que en conjunto con otros similares conducen la búsqueda hacia espacios más específicos que suelen mostrar mayor calidad en su evaluación. Un problema se considera como "engañoso" (*deceptive problem*) si ciertos hiperplanos llevan la búsqueda hacia ciertas soluciones o conjunto de bloques que en un contexto global no son competitivos, pese a que en un contexto local si lo puedan ser. En combinación con un grado de acoplamiento bajo en los bloques, este tipo de problemas suelen llamarse *GA-Hard*, es decir, altamente complejos para los GA. Para comprender las características de los problemas engañosos, es importante conocer previamente algunos conceptos que se señalan en [7], vinculados a hiperplanos y a las competencias que se realizan entre ellos cuando un GA opera.

Una "competencia primaria entre hiperplanos" de orden N, involucra a todos los competidores primarios, los que corresponden al conjunto de 2^N hiperplanos asociados a un *schema* de orden N. Por ejemplo 0*0*, 1**0 y **10 son competidores en una competencia primaria entre hiperplanos.

El "ganador global" en una de estas competencias, es el hiperplano que tiene un mayor nivel de calidad en el conjunto de competidores, donde la calidad se mide como el promedio obtenido entre todos los strings en dicho hiperplano.

Se dice que un hiperplano X contiene a otro Y, cuando el orden de X es menor que el de Y, junto con el hecho que este último tiene exactamente los mismos bits en las posiciones respectivas en X, salvo los correspondientes al símbolo "*" en X. De esta manera, un hiperplano representado por el *schema* "0** contiene al hiperplano con *schema* 0*1 o 00*. Esta definición lleva a otro concepto importante vinculado a las competencias: dos competencias primarias entre hiperplanos con orden N *y* K, con $N > K$, son "relevantes" una con otra si en conjunto los competidores de orden K contienen a los de orden N. Por ejemplo, una competición primaria entre los hiperplanos 11***, 10***, 01*** y 00*** consideran un grupo de hiperplanos que en conjunto contiene la competencia primaria entre los hiperplanos 111**, 110**, 101**, 100**, 011**, 010**, 001** y 000**, por ello ambas competencias son relevantes una con otra.

En base a estos conceptos, el término "engaño" (*deception*) implica que el ganador global de una competencia entre hiperplanos de orden *N*, tiene un patrón de bits distinta a la del ganador global de una competencia entre hiperplanos "relevante" de orden menor que *N*. El nivel de engaño en un problema se puede clasificar del siguiente modo:

• **Problema engañoso (*deceptive problem*):** Son problemas específicos de orden N que involucran "engaño" en una o más competiciones primarias relevantes de orden más bajo. Esta clasificación es relativamente amplia, pues en la práctica todos los problemas conservan algún grado de "engaño", en otros términos, siempre se espera que al menos una competencia relevante de mayor orden conduzca a un ganador que no tiene la misma estructura que el de orden N.

• **Problema totalmente engañoso (*fully deceptive problem*):** Son problemas (o subproblemas) de orden N en los que todas las competencias relevantes de mayor orden apuntan a obtener un ganador global correspondiente a un "atrayente engañoso" (*deceptive attractor*). Este último corresponde a un hiperplano de orden *N* distinto al ganador global apoyado por las competencias relevantes entre hiperplanos de orden menor. Se puede demostrar según [7] que el atrayente engañoso puede corresponder al hiperplano representado por el schema que se tiene el patrón de bits complementario del ganador global en la competencia de orden N. Por ejemplo, para un ganador global del tipo **1**1***1, el potencial atrayente engañoso sería **0**0***0.

• **Problema consistentemente engañoso *(consistently deceptive problem):*** Son problemas (o subproblemas) de orden *N* en los cuales ninguna de las competencias primarias de hiperplanos relevantes de mayor orden conducen al ganador global de la competencia de orden *N* señalada, sin embargo, no se asegura que ellas vayan a conducir a un ganador global que corresponda al atrayente engañoso, excepto para las competencias de orden 1, las que guían la competencia o hacia el ganador global o hacia el atrayente engañoso, en dicho orden.

El concepto de *deception* puede ser abordado con mucha más de profundidad, fuera de los límites de esta investigación. Hay estudios que han impulsado la construcción de problemas engañosos de distinto tipo para medir el comportamiento de los GAs ante distintos

escenarios. En [7] varios tópicos se consideran en torno a este tipo de problemas, incluyendo experimentos con distintos modelos de GA aplicados en problemas engañosos.

3.3. Variantes de GAs

En base a los conceptos repasados en 3.2, en la literatura se pueden encontrar numerosas nuevas versiones de GAs que tienden en general a abordar toda la problemática que se genera al considerar la mezcla adecuada de los bloques de construcción y el grado de acoplamiento que los mismos deben presentar, además de considerar las distintas instancias de problemas engañosos en las que se pueda estar trabajando. En esta sección se repasará, en base a una clasificación que se definirá, algunos de los GAs que recaen en las instancias de esta clasificación, con el fin de tener una visión clara acerca de adonde apunta el diseño de nuevos GAs y que esto sirva de referencia para la aplicación de los mismos en DARPTW.

3.3.1. Una clasificación propuesta

Para repasar los tipos de GA que en la literatura se pueden encontrar, se usará la clasificación propuesta en [19] y considerada también en [10] con un poco más de detalle. Esta clasificación se enfoca principalmente en los mecanismos que utiliza el GA para lograr el acoplamiento entre los bloques de construcción que se mezclan por medio de los distintos operadores, en particular, para identificar estos bloques. La clasificación es la siguiente:

- **Métodos basados en perturbación:** Se basan en perturbar la estructura de un individuo y analizar los cambios en la calidad que entrega, para medir el acoplamiento. Consideran la no linealidad o epistasis en el individuo tras las perturbaciones como acoplamiento y extraen dicha información a través de muestreo o enumeración. Algunos de los GAs de esta clasificación son *messy-GA* (Goldberg, Korb, & Deb, 1989; Goldberg, Deb, & Korb, 1990), *fast messy-GA* (Goldberg, Deb, Kargupta, and Harik, 1993), *gene-expression messy-GA* (Kargupta, 1996), *linkage identification by nonlinearity check - LINC* (Munetomo and Goldberg, 1998) y *linkage identification by nonmonotonicity detection - LIMD* (Munetomo and Goldberg, 1999).

• **Técnicas para la adaptación del acoplamiento:** En este tipo de GAs, la identificación del acoplamiento y su aprendizaje en el proceso evolutivo, de logra a través de los mismos mecanismos del algoritmo, ya sea con los operadores y la representación. Estos algoritmos están más cerca que los otros de la metáfora biológica referente a la computación evolutiva. En esta clasificación, el algoritmo más tradicional o representativo es *linkage learning GA - LLGA* (Harik, 1997).

• **Constructores de modelos probabilísticos:** Estos GAs "aprenden" el acoplamiento de los bloques construyendo modelos probabilísticos de la población actual y generando en base a ellos los nuevos individuos para la población futura. Este tipo de GAs tienen una visión más computacional que biológica, y como se señala en **¡Error! No se encuentra el origen de la referencia.**, han tendido a mostrar el mejor rendimiento. En este conjunto se puede destacar *compact genetic algorithm - cGA* (Harik, Lobo, Goldberg, 1999), *extended compact genetic algorithm - ECGA* (Harik, 1999) y *Bayesian optimization algorithm - BOA* (Pelikan, Goldberg, & Cantú-Paz, 2000).

En las siguientes secciones se revisará algunos de los GAs mencionados, en particular uno en cada clasificación: el mGA, el LLGA y el cGA.

3.3.2. Messy Genetic Algorithm (mGA)

El mGA presentado por primera vez en [6] por Goldberg, Korb, y Deb, separa dos procesos importantes que en el GA simple se realizan al mismo tiempo: Identificar y recombinar los bloques de construcción. Cuando no se tiene información acerca del grado de acoplamiento de los bloques, realizar ambos procesos simultáneamente dificulta la resolución del problema especialmente si este presenta características de engañoso. El objetivo del mGA es construir de forma explícita e incremental strings largos y bien acoplados, desde bloques de construcción pequeños y bien testeados.

El algoritmo usa una representación independiente de la posición como la revisada en 3.2.3. Sin embargo, en este caso los strings pueden estar "sobre especificados" y/o "poco especificados". Lo primero ocurre cuando hay posiciones repetidas en el string, lo segundo es cuando faltan posiciones para formar una solución completa. Por ejemplo, se tienen los

siguientes strings: **{(1,0) (2,0) (4,1) (4,0)} y {(3,1) (3,0) (3,1) (4,0) (4,1) (3,1)}**. El primero está poco especificado pues carece del gen en la posición 3 y está sobre especificado pues hay 2 instancias del gen 4. Análogamente para el segundo string, faltan los genes de las posiciones 1 y 2 y existen 4 instancias del gen 3 y 2 instancias del gen 4. Para poder evaluar la calidad del schema asociado al string, dependiendo en cuál de las dos situaciones esté (sin dejar de considerar que puede estar en ambas), se realiza lo siguiente:

• En el caso de sobre especificación, simplemente se aplica una lectura de izquierda a derecha, y la primera ocurrencia del gen se considera como la efectiva en la evaluación (descartando las posteriores). Estos strings representarán *schemas* según su nivel de poca especificación, colocando "*" en los genes que estén ausentes. El string 1 representa al *schema* 00*1, mientras que el 2 representa al **10.

• El caso de la poca especificación es más difícil de abordar, pues resulta complejo (si no imposible) medir la calidad de una solución incompleta, generalmente la interacción entre los distintos componentes de la estructura de un cromosoma no es independiente. Por esto, se desarrolló un método para asignar alleles a los genes no especificados llamado "plantillas competitivas". Se trata de analizar si el *schema* asociado al string produce alguna improvisación respecto al óptimo local. Para ello, al principio de la ejecución, se obtiene un óptimo local mediante alguna técnica como *hill-climbing*. Para evaluar los strings sujetos a poca especificación, se reemplazan los genes faltantes por los del óptimo local encontrado. Por definición, un óptimo local no puede ser optimizado al realizar un cambio simple de bits, por esto es que si el string analizado presenta mejoras respecto al óptimo local, vale la pena que siga siendo analizado.

El mGA separa la identificación y recombinación de los bloques mediante dos fases en el proceso: la fase "primordial" y la fase "yuxtaposicional". En la primordial se considera como k el orden más pequeño de los schemas relevantes para el problema en términos de nivel de "engaño" y se genera una población enumerando todos los *schemas* de ese orden. Por ejemplo, para un string de largo 8 y con $k = 3$, se tiene la siguiente población inicial:

$$
\begin{array}{l}
\{(1,0)\ (2,0)\ (3,0)\} \\
\{(1,0)\ (2,0)\ (3,1)\} \\
\quad\vdots \\
\{(1,1)\ (2,1)\ (3,1)\} \\
\{(1,0)\ (2,0)\ (4,0)\} \\
\{(1,0)\ (2,0)\ (4,1)\} \\
\quad\vdots \\
\{(6,1)\ (7,1)\ (8,1)\}.
\end{array}
$$

En este contexto nace una desventaja importante del algoritmo: se requiere conocer el valor de *k*, o por lo menos una aproximación de este valor, cosa que en la práctica no es algo trivial de obtener.

Posteriormente, con la población formada y la calidad de cada individuo calculada (usando plantillas competitivas), se ejecuta un proceso de selección, sin recombinación o mutación. Cada cierto intervalo de tiempo, se desecha la mitad de la población.

En una generación especifica (determinada por un parámetro), se termina la primera fase y se comienza con la yuxtaposicional. En está, se mantiene el tamaño de la población fijo, se continúa con la selección, y se ejecutan dos operadores: El operador de "corte", que divide un string en dos partes respecto a una posición aleatoria, y el operador de "pegado", que une dos strings distintos. Así es como, para un string de la forma **{(2,0) (3,0) (1,1) (4,1) (6,0)}**, al aplicar el operador de corte en la posición 2, se pueden obtener los strings **{(2,0) (3,0)}** y **{(1,1) (4,1) (6,0)}**. Para el operador de pegado, si se tienen los dos siguientes strings: **{(1,1) (2,1) (3,1)}** y **{(1,0) (4,1) (3,0)}**, se obtiene el string **{(1,1) (2,1) (3,1) (1,0) (4,1) (3,0)}**. El hecho que los operadores de pegado y corte funcionen, se basa en que en la etapa primordial se generaron suficientes bloques como para formar strings óptimos, y que hay suficientes individuos en la población para ello.

Otro de los problemas que surgieron con este algoritmo, es que la enumeración de los individuos en la primera fase, aunque se conozca *k*, conduce a tiempos de ejecución inaceptables para problemas reales donde la representación del cromosoma tiene un largo considerable. Esto llevo al desarrollo del *fast mGA*, en donde la enumeración fue reemplazada por un proceso denominado "completación de iniciación probabilística". En él, las combinaciones pueden ser abordadas por la generación de strings mucho más largos

que k. Para esto, si el largo inicial del string es l y el tamaño de la población inicial es n_0, es posible calcular ambos valores en base a k, tal que se asegure en promedio que cada *schema* de orden k estará presente en la población inicial. Si l incrementa, n_0 puede ser reducido en gran parte para la primera etapa. Adicionalmente los strings se someten a un proceso de filtrado y borrado, lo que en conjunto con selección permite enriquecer la población con los mejores bloques de construcción. Este proceso se repite hasta tener toda la población con orden k, con lo que la etapa yuxtaposicional puede inicializarse tal como se describió antes. Estos cambios permitieron la ejecución del algoritmo mucho más rápida y efectivamente.

Si bien este algoritmo se presentó como una buena alternativa para abordar problemas que el GA simple no podía, la dificultad en su utilización causó que su adopción fuera relativamente lenta. Algunas razones son la dificultad de ajustar el mecanismo de filtrado usado en la inicialización y el aumento exponencial de la población inicial con el aumento de k, aun usando el método de completación de iniciación probabilística. De todas formas, este algoritmo representa uno de los primeros intentos efectivos de abordar el problema del identificar el acoplamiento entre bloques de construcción.

3.3.3. Linkage Learning GA (LLGA)

Este algoritmo fue presentado por Harik en [9], donde recalca que hasta los años en los que se realizó dicha publicación, el tema del acoplamiento había sido muy descuidado para la importancia que tiene para la competencia de los GA. En su trabajo, se enfatizan ciertas contrariedades del operador de inversión respecto a buscar y lograr el acoplamiento entre bloques de construcción y propone tanto una definición de la medida de acoplamiento como un nuevo operador de cruzamiento “compatible” con la búsqueda de acoplamiento. En esta sección el enfoque estará en esto último.

En el LLGA, se tiene una representación independiente de la posición, como la mostrada en 3.2.3. Adicionalmente, el operador de recombinación que el algoritmo utiliza es el cruzamiento en dos puntos. Según lo comentado en 3.2.3, para este tipo de operador, el cromosoma en cuestión puede representarse como un anillo.

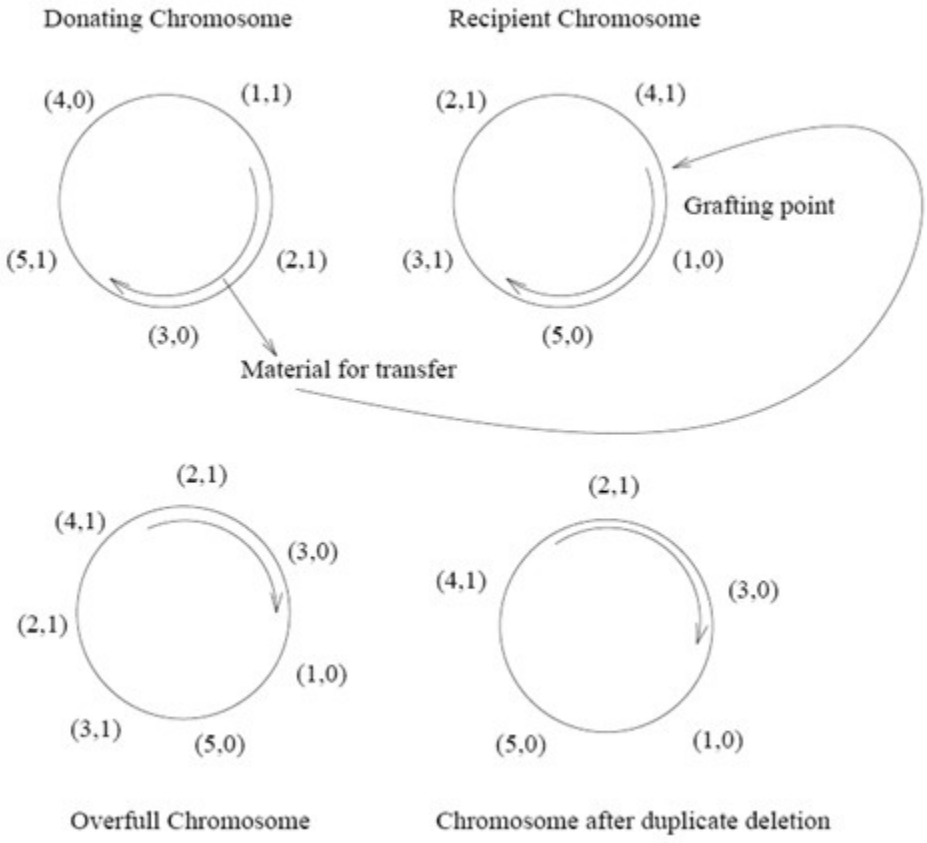
Se definen los dos cromosomas (cuya estructura es arbitraria) que participarán en la recombinación, uno corresponde al "donador" y el otro al "recipiente". En este contexto, del proceso de recombinación usando cruzamiento en 2 puntos, a diferencia de los mecanismos tradicionales, sólo se obtiene un nuevo individuo. Naturalmente, los cromosomas se pueden intercambiar los roles para generar otro individuo. El donador será el que entregue un conjunto de sus genes al recipiente para formar la estructura del nuevo individuo. Al recibir dichos genes, el recipiente tendrá posiciones repetidas en su estructura, de las que se eliminan sólo las que no corresponden al material genético traspasado. Adicionalmente, se asume que ambos cromosomas tienen una orientación implícita, la que sirve para extraer los genes desde el donador al recipiente. Esta orientación debe ser idéntica en ambos cromosomas. Por ejemplo, sea (4,0) (1,1) (2,1) (3,0) (5,1) el donador y (2,1) (4,1) (1,0) (5,0) (3,1) el recipiente. Desde el donador se escoge el segmento (2,1) (3,0) y se inserta en el recipiente después de (4,1), este ultimo termina con la estructura (2,1) (4,1) (2,1) (3,0) (1,0) (5,0) (3,1), la cual contiene genes repetidos. Se eliminan los que estaban originalmente en el cromosoma, esto causa que el recipiente termine con la estructura (4,1) (2,1) (3,0) (1,0) (5,0). El proceso se ilustra en la Figura 3.4.

En [20] se habla de "segmentos no-codificables" denominados "introns", que corresponden a genes en el cromosoma que no son considerados por la función de evaluación para obtener la calidad, aunque los mismos se pueden mover por la recombinación. Este tipo de genes han sido ampliamente estudiados en virtud de su impacto favorable en la recombinación de bloques de construcción.

En [10] se repasa una serie de otras alternativas para este tipo de algoritmo. Otra representación llamada "expresión probabilística" (PE) se propuso para preservar la diversidad al nivel de los bloques de construcción. Se trata básicamente de incluir todos los alleles posibles para una posición en el cromosoma. Bajo el propósito de la evaluación de calidad, un "punto de interpretación" es escogido en el cromosoma, desde el cual se revisan los genes en una dirección de las agujas del reloj, escogiendo el primer gen de cada posición leída como el que se interpretará por la función de evaluación. Como consecuencia de esto, un cromosoma representa una distribución de probabilidades sobre el rango de posibles soluciones, en vez de una simple solución. Una generalización de PE es la

expresión probabilística extendida (EPE), en la cual se puede tener hasta una cantidad k de posiciones repetidas en un cromosoma.

Figura 3.4 Funcionamiento del LLGA básico [9].



También existen los "promotores" en el LLGA. Se trata de puntos no-codificables del cromosoma que son utilizados como "puntos de interpretación obligados", es decir, sólo en ellos se puede definir puntos de interpretación. Estos son utilizados sólo en PE, no en EPE. Además, al existir los promotores, estos pasan a ser el punto de interpretación del nuevo individuo que se obtendrá con el operador de recombinación. Al ser escogido el punto en el que se insertará el material en el recipiente, el punto de interpretación corresponderá al del promotor más cercano después del punto de inserción. En esa instancia, el material genético se inserta en el siguiente orden: el segmento entre el promotor y el punto de inserción, el segmento escogido por el donante, el resto del recipiente.

Existen dos mecanismos básicos mediante los cuales el LLGA "aprende" el acoplamiento:

• ***Linkage Skew:*** Ocurre cuando un bloque óptimo es transferido desde el donante al recipiente. Para esto hay dos condiciones: que el bloque este dentro del segmento cortado y que el bloque óptimo sea expresado antes que otro inferior lo haga. Este mecanismo permite aumentar la distribución del acoplamiento al eliminar individuos de baja calidad.

• ***Linkage Shift:*** Ocurre cuando un bloque óptimo reside en el recipiente y sobrevive a la recombinación. Este mecanismo mantiene los bloques firmes en un cromosoma, al eliminar los genes redundantes resultantes de la recombinación.

El LLGA ha demostrado tener dificultades con problemas escalados uniformemente. Estos corresponden a problemas de múltiples bloques de construcción (donde un cromosoma puede determinarse por la suma de la calidad intrínseca de los distintos genes que lo conforman) en los que la posición de los genes se considera idéntica. Estos tipos de problemas son estudiados a fondo en la versión detallada de la publicación de Harik [8].

3.3.4. Compact GA (cGA)

Este tipo de GA presentado por Harik, Lobo y Goldberg en **¡Error! No se encuentra el origen de la referencia.**, se basa en representar a la población como una distribución de probabilidades sobre el conjunto de soluciones. Opera como el comportamiento de orden uno del GA simple, con cruzamiento uniforme. Al hacer discreta la probabilidad de representación, reduce los requerimientos originales de memoria de los GAs. En este sentido, se aprecia que este tipo de algoritmo está más enfocado a la eficiencia computacional.

En el cGA se maneja un vector V del largo equivalente al del cromosoma a representar. Se tiene además N como el tamaño de la población. Dicho vector es inicializado con valor 0.5 en todos sus índices. Se describe el cGA desde dos puntos de vista: la recombinación y la selección.

La recombinación en cierto modo se basa en la generación un nuevo individuo en base al vector V. En cada índice de este vector se expresa la probabilidad de que el índice respectivo en el hijo sea un 1, y si no se cumple, sería un 0.

En la selección, primero se generan dos cromosomas usando *V*. Para ambos individuos se evalúa la calidad de cada uno y se escoge el ganador. Para cada índice en *V* se consideran las siguientes modificaciones:

• Si en el índice correspondiente, el ganador tiene un 1 y el perdedor un 0, entonces el índice en *V* varía en + *1 / N*.

• Si en el índice correspondiente, el ganador tiene un 0 y el perdedor un 1, entonces el índice en *V* varía en – *1 / N*.

• Si en el índice correspondiente, ambos individuos tienen el mismo bit, no se modifica V en dicho índice.

El algoritmo termina cuando todos los índices de *V* son 0 o 1 (independientemente). Este vector representa la solución.

Como se aprecia, la mecánica de este algoritmo es muy simple. Los autores lo diseñaron como un modo de comprobar la complejidad de un problema. Si el cGA puede solucionarlo con un ratio de selección bajo (el que está dado por el incremento o decremento de los valores en *V*), el problema es sencillo. Para problemas más complejos se debería aumentar los ratios.

También se afirma que el algoritmo no pretende reemplazar el concepto poblacional de un GA tradicional, sin embargo es un acercamiento al tema de los recursos computacionales usados por los GA. En este contexto, el cGA pretende ser la base para nuevos tipos de algoritmos, más eficientes y capaces. Esto quedó demostrado con la aparición de nuevas extensiones, como el cGA extendido [22].

4. Trabajos relacionados

Para entrar en el contexto del estudio de DARPTW y de problemas similares, en esta sección se repasará una serie de trabajos realizados previamente, en los cuales los GAs se han utilizado como herramienta de resolución, ya sea de manera parcial como de manera total, lo que depende principalmente de la estrategia que se haya aplicado. Considerando los alcances y objetivos de esta investigación, no se describirán otras aproximaciones distintas a GA para la resolución de DARPTW, sin embargo cabe destacar que en los trabajos que se señalarán, se mencionan distintas referencias de otras técnicas por medio de las cuales se ha abordado el problema. Primero se verá el trabajo vinculado a VRP y a PDP, luego se abordarán las aproximaciones para DARPTW.

4.1. Vehicle Routing Problem

4.1.1. Thangiah S. R.

En el trabajo de Thangiah [11], se describe GIDEON, una heurística basada en GAs para resolver el VRP con ventanas de tiempo. 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 cromosomas son representados como strings de bits. Los sectores o clusters de clientes se obtienen desde el cromosoma al dividirlo en K divisiones de B bits. Cada subdivisión es usada para computar el tamaño de un sector. La calidad del cromosoma se obtiene por la evaluación de la función de costos de servir a todos los clientes computados, respecto a las divisiones de sectores derivadas desde él. Se señala que se utiliza $B = 3$, valores mayores a eso no entregaban resultados satisfactorios.

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 acá descrito, se utilizó $\lambda=2$.

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, ampliamente conocidos en esta área. Los resultados publicados señalan que de ese número, 41 resultados fueron mejores que otras heurísticas desarrolladas anteriormente por Solomon y Thompson, salvo en cierto set de problemas específicos.

4.1.2. Zhu K. Q.

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

La representación se esboza del siguiente modo: un cromosoma se estructura como un string de enteros de largo N, donde este número 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 set de rutas, ordenados de acuerdo a su aparición en el cromosoma. Por ejemplo, se tiene la siguiente solución: Ruta 1: 0 → 3 → 2 → 4 → 5; Ruta 2: 0 → 10 → 6 → 1 → 12 → 11 → 0; Ruta 3: 0 → 9 → 8 → 7 → 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 Insertion 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 3.1.4.

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 alleles en todo el cromosoma, basándose en los que conforman un segmento especifico de él. Otro es el *Heuristic crossover*, en donde se trata la distancia entre nodos representados en el cromosoma. Para este operador hay dos versiones según su forma de operar, que el autor denominó *HeuristicCrossover1* y *HeuristicCrossover2*. El último operador propuesto se denomina *Merge Crossover*, el que opera en base de una precedencia de tiempo predefinida. Esta a menudo es obtenida según las ventanas de tiempo impuestas por cada nodo. En base a un criterio similar al mecanismo anterior, existen dos variantes definidas: *MergeCrossover1* y *MergeCrossover2*. Sobre los dos últimos mecanismos, estos generan sólo un hijo desde dos padres. Por ello el autor se enfocó en realizar mezclas de los mecanismos para obtener más individuos en la recombinación. Para lo que es mutación, utiliza otros métodos convencionales.

Los experimentos se realizaron con distintas combinaciones de operadores de recombinación, usando 56 instancias de Solomon 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

4.2. Pickup and Delivery Problem

4.2.1. Jih W., Kao C., Hsu J.

Para abordar PDP en [13] se utilizó un enfoque variante del GA llamado *Family Competition GA* (FCGA), el que se usó previamente en problemas como TSP. Cabe

destacar que en el trabajo señalado se aborda específicamente PDP con ventanas de tiempo, de un sólo vehículo.

Para la estructura del cromosoma se usa una representación de permutación. En esta, cada gen representa una locación (número entero) que puede corresponder a una recogida o una entrega para alguna solicitud, y el orden que tengan en el cromosoma representa la ruta del vehículo. Por ejemplo, si se tiene la siguiente ruta solución: 0 → 3+ → 1+ → 1- → 2+ → 2- → 3-, el cromosoma correspondiente sería (0, 3+, 1+, 1-, 2+, 2-, 3-). Se puede ver que el largo del cromosoma es de $2^N + 1$, con N la cantidad de solicitudes.

Se define una función de calidad $c(S) = ft(S) + fp(S)$, donde ft es la función que representa el tiempo total de viaje del vehículo para completar la ruta S, y fp una penalización para la ruta S en relación a como se pasen a llevar las restricciones del problema. Estas restricciones se generan sólo en soluciones infactibles, para las soluciones factibles, fp es 0. En este sentido, este algoritmo usa ventanas de tiempo blandas.

Para la recombinación, se especifican 4 tipos de operadores: el cruzamiento de orden (*order crossover - OX*), el cruzamiento uniforme basado en orden (*uniform order-based crossover - UOX*), y dos variantes de *Merge Crossover* como las mencionadas en 4.1.2. Para la mutación se usaron algunos operadores específicos descritos en la publicación.

El FCGA en que se basa el trabajo se caracteriza principalmente por su población, en donde cada individuo pertenece o está asociado a una "familia" de cromosomas. Cada miembro de la familia es la recombinación de un padre de familia, cada individuo tiene la oportunidad de ser un padre de familia. Para mantener el tamaño constante de la población, sólo un miembro en la familia gana la competencia que se realice entre los individuos pertenecientes a ella. Una descripción detallada de cómo funciona el algoritmo se presenta en la publicación.

Respecto a los experimentos, al no existir sets de pruebas estándar para PDP, se desarrolló un algoritmo particular para generar datos de prueba. Se usaron 3 ratios de recombinación: 0.45, 0.6 y 0.7. Además los distintos operadores mencionados anteriormente se testearon

profundamente. Los resultados mostraron una mejora comparativamente conveniente respecto al GA tradicional.

4.3. Dial-a-Ride Problem

4.3.1. Bergvinsdottir K.

La aproximación del trabajo presentado en [1] y [14] es similar a la mostrada en 4.1.1, es decir, una estrategia *cluster-first route-second*. El modelo matemático que se utiliza en este trabajo para abordar DARPTW, es una generalización del mostrado en [16] que es el referenciado en el presente proyecto. Esta generalización se justifica en el hecho que Bergvinsdottir aborda el problema de DARPTW considerando ventanas de tiempo blandas, por lo que en la función objetivo del modelo se considera una serie de nuevos elementos referentes a la calidad de servicio. De cierto modo, entre el trabajo de Bergvinsdottir y [16] se pueden extraer una serie de propiedades de los modelos que serían aplicables al presente proyecto.

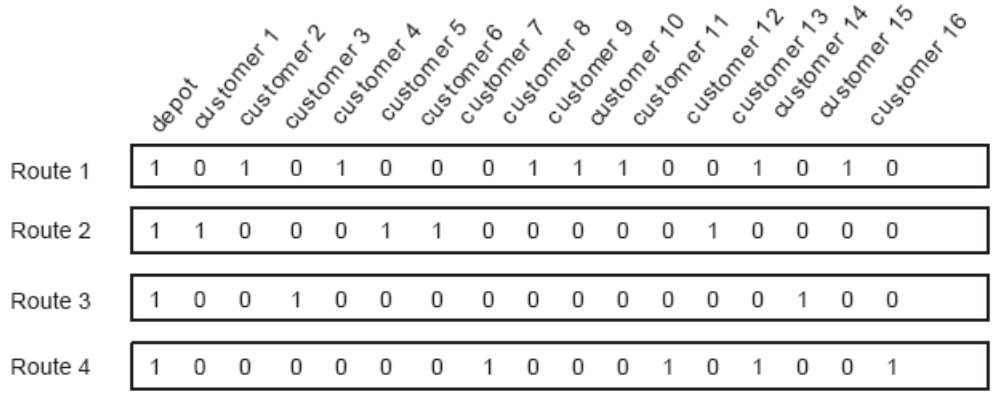
La representación está basada en los grupos de usuarios o clusters. Se utilizan tantos vehículos como clusters obtenidos. En la práctica se considera una matriz, donde hay tantas filas como vehículos disponibles y tantas columnas como clientes y depósitos. Si una coordenada en la matriz es 1, indica que el cliente/deposito correspondiente está asociado al vehículo correspondiente. Adicionalmente, una fila (vehículo) representa una ruta específica. Un ejemplo para 4 vehículos, un depósito y 16 clientes se ve en la Figura 4.1.

Se usan dos procedimientos de selección (uno para cada individuo en el proceso de recombinación), uno es usando el método de la ruleta y el otro es simplemente de forma aleatoria.

Para la recombinación, una fila es elegida aleatoriamente para ambos padres, generando una plantilla binaria. Esta plantilla es usada como un recipiente para una fila en el hijo, donde una entrada con valor 1 indica que el gene será tomado del padre 2. Finalmente, el hijo está formado por la nueva fila, más las demás que son copiadas del padre 1. Este proceso eventualmente puede generar soluciones no-válidas, al asociarse un pasajero a más

de un vehículo. Usando procedimientos particulares, estas soluciones se corrigen. La mutación consiste básicamente en mover un cliente desde un *cluster* a otro.

Figura 4.1 Representación del cromosoma (Bergvinsdottir K.) [4].



Ya realizada la asignación, se realiza el proceso de post-optimización. Para esto se ocupa una modificación de la heurística denominada *modified space-time nearest neighbor heuristic*. En ella, el costo de una ruta se considera como una suma balanceada entre el tiempo de viaje y la violación de las ventanas de tiempo. Este mecanismo trabaja muy ligado a lo que son las restricciones del problema, en particular a las ventanas de tiempo, como se detalla en la publicación.

Al igual que el caso de 4.2.1, para DARPTW no se tiene un set de datos de entrada estandarizados, lo que representó una dificultad en este estudio. Se recurrió a un conjunto de datos desarrollados por Cordeau y Laporte, se trata de 20 instancias aleatorias acorde a supuestos realistas. Estos datos contemplan desde 24 a 144 clientes. Los resultados obtenidos se comparan también a los del estudio de Cordeau y Laporte referenciado en ese trabajo, logrando resultados similares.

4.3.2. Cubillos C., Rodriguez N., Crawford B.

Este el trabajo detallado en [15] se presenta lo que puede considerarse la aproximación más cercana a los objetivos de este proyecto, que se puede encontrar en la literatura. Esto principalmente porque la estrategia desarrollada, aborda DARPTW para obtener soluciones

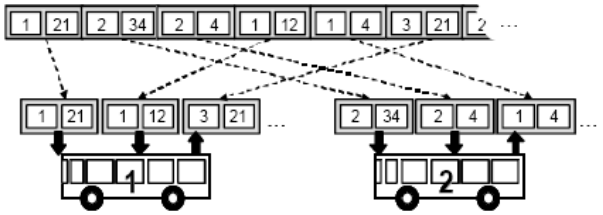
completas usando GAs, a diferencia de la mostrada en 4.3.1 por ejemplo, donde se usa GAs para resolver sólo una parte de todo el problema.

En la publicación se especifica como objetivo del trabajo, más que desarrollar un GA genérico a modo de optimizador universal, se pretende desarrollar un GA muy específico para abordar DARPTW y obtener resultados no engañosos. Para ello se habla de desarrollar un *framework* adecuado, que contemple todos los elementos más importantes de los GA. La instancia especificada de DARPTW a la vez es muy específica, a diferencia de otros estudios por ejemplo, se consideran sólo clientes de salida (especificados en 2.5).

Como parte del *framework*, se detallan los siguientes elementos:

• **Genotipo (Representación):** Se desarrolló una representación de la forma (bus, pasajero) para cada gen. La decodificación corresponde a una lectura ordenada, en donde la primera ocurrencia de un pasajero siempre es una recogida y la segunda siempre es una entrega, esto sugiere que el cromosoma sólo tenga dos elementos con el mismo pasajero. El bus asociado a la recogida es el considerado en la solución, eventualmente el que este especificado en el gen de la entrega puede ser distinto, pero no es considerado. La Figura 4.2 muestra la decodificación de un individuo.

Figura 4.2 Decodificación de un cromosoma (Cubillos C.) [15].



• **Evaluación:** Básicamente consiste en la evaluación de la función objetivo del problema.

• **Población inicial:** Se mencionan dos posibles mecanismos para generar la población inicial, una generación aleatoria a nivel de bits y la utilización de una heurística de inserción para asignar pasajeros a los vehículos disponibles.

• **Selección:** Se especificó una selección por torneo (Ver 3.1.4).

- **Recombinación:** Se consideraron el cruzamiento en un punto y PMX.

- **Mutación:** Se consideró la mutación clásica a nivel de bits y un operador llamado 2-opt.

Para los experimentos, en primera instancia se desarrolló un modelo basado en una representación binaria, el cual funcionó sin éxito al no lograr encontrar soluciones factibles. Posteriormente se usó una representación con valores enteros, considerando la generación inicial de individuos aleatoria, que funcionó bien con un tamaño de población bajo 30, sobre ese valor se generaron muchas soluciones infactibles. Una tercera implementación consideró el operador PMX, 2-opt y la heurística de inserción para la población inicial, lo que permitió generar soluciones factibles para poblaciones entre 50 y 100 individuos. Se realizaron 640 ejecuciones considerando distintos parámetros mostrados en la publicación. Los resultados finales apuntaron una mejora respecto a los valores de calidad, pero no de cantidad de vehículos, respecto a otros estudios anteriores. Por último, el trabajo señala la posibilidad de seguir realizando implementaciones con otro tipo de GAs que consideraran el problema del acoplamiento, como el LLGA, lo que eventualmente representa un punto de partida en el presente proyecto.

5. Modelos propuestos

En este capítulo se revisarán los detalles referentes a los modelos de GAs que se implementaron. Aunque los dos modelos a mostrar pueden considerarse independientes en ciertos aspectos, hay elementos en su estructura que funcionarán idénticamente en ambos, los que se señalarán adecuadamente. La definición y formalización de estos modelos es parte de una revisión amplia respecto a la estructura de distintos GAs que se pueden encontrar en la literatura. Ciertos criterios referentes a las características del problema DARPTW y la aplicabilidad de los elementos que se encuentran en las estructuras señaladas, son considerados para desarrollar los modelos propuestos.

En primer lugar, se hará una descripción general de la estructura de cada uno de los modelos, esto implica señalar los submodelos que se desarrollarán y se considerarán, a modo de marco de trabajo. Luego se describirá de forma general la estructura de ambos modelos.

5.1. Marco de trabajo

En el contexto de los GAs, en particular cuando se habla de definir un nuevo modelo de GA, ya sea que se conciba bajo una perspectiva muy general, esto es, que se adapte de la mejor forma posible a un conjunto de problemas de distinta naturaleza, o cuando se considera bajo una aplicación para un problema muy particular, son distintos los submodelos o elementos de su estructura que se pueden definir, y los mismos se asocian principalmente con cosas como los operadores, la representación, etc. En el caso de los GAs orientados a problemas muy particulares, es común que la mayoría (incluso todos) de estos elementos se definan detalladamente, los que en conjunto pueden considerarse como el marco de trabajo del modelo. En este proyecto se desarrolló un par de modelos orientados de forma muy particular a DARPTW, por lo que se considerará a su vez la definición del marco de trabajo respectivo.

A continuación se describirán los elementos que cada uno de los modelos considera, rescatando para parte de ellos la terminología ocupada en [15] respecto al marco de trabajo. Los mismos se pueden dividir en dos grupos:

• **Submodelos de soporte**, utilizados para lo que es el manejo de restricciones y factibilidad que impone el problema tratado. Se incorporan de forma transversal a distintos elementos del GA. Se consideran **esquemas de factibilidad preliminar, mecanismos de modificación de rutas**, y **generación de soluciones factibles**.

• **Submodelos de GA**, que corresponden a los elementos más comunes (siempre considerados) dentro de la formalización de un GA. Se consideran **genotipo, fenotipo, selección, recombinación, mutación** y **población inicial**.

5.1.1. Esquemas de factibilidad preliminar

Considerando que el contexto en que se trata el problema es estático (ver 2.4), los datos de entrada que se tienen entregarán información diversa respecto a las solicitudes de transporte de los clientes, que no cambiará en el tiempo. Esta información es valiosa no sólo por el hecho que plantea las condiciones en las que se evaluarán las soluciones del problema, adicionalmente ella se puede procesar para generar otro tipo de información que permita hacer la búsqueda, dentro del contexto del GA, más rápida y efectiva en términos de factibilidad. Como DARPTW es un problema de restricciones importantes, se puede indagar en los datos de entrada para establecer relaciones que permitan acotar el espacio de búsqueda y hacer el procesamiento más rápido.

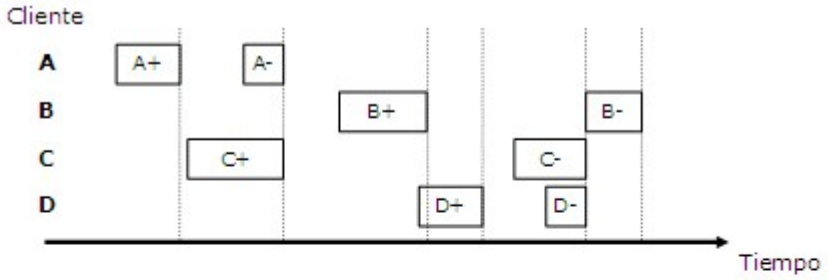
Hay dos aproximaciones consideradas en esta propuesta para lo anterior. La primera es la construcción de una **tabla de precedencia preliminar en ejecución de eventos**, y en base a ella se genera una **tabla de clientes incompatibles**.

5.1.1.1. Tabla de precedencia preliminar en ejecución de eventos

La disposición de las ventanas de tiempo que se entregan para la ejecución de las solicitudes de transporte de los clientes, puede ser analizada de forma que se pueda determinar, para cada una de estas solicitudes, que otras solicitudes deben siempre ser

ejecutadas previamente dentro de una ruta, de forma que la programación que se realice con ellas sea factible. En la Figura 5.1 se muestra un ejemplo de esto.

Figura 5.1 Disposición de ventanas de tiempo y tabla de precedencia preliminar.



A+ →	
A- →	A+
B+ →	A+, A-, C+
B- →	A+, A-, B+, C+, C-, D+, D-
C+ →	A+
C- →	A+, A-, B+, C+, D+
D+ →	A+, A-, B+, C+
D- →	A+, A-, B+, C+, D+

Si tenemos el caso que los clientes B y C deben ser colocados en una misma ruta, hay consideraciones lógicas en relación a sus respectivos eventos (B+, B-, C+, C-) que se pueden tener en cuenta. Por una parte, es evidente que los eventos de recogida de cada cliente deben siempre estar antes en una ruta que sus eventos de entrega, esta es una relación directa de precedencia. Sin embargo, es posible establecer relaciones similares entre eventos de clientes distintos. Por ejemplo, es imposible ejecutar el evento B- sin haber ejecutado antes C+, dado que la ventana de tiempo de B- es completamente posterior a la de C+, es decir, el extremo inferior de la ventana B- o *ET(B-)*, es mayor o igual que el extremo superior de la ventana de C+ o *LT(C+)*. Si se ejecutara antes B- dentro de su ventana de tiempo, sería imposible ejecutar después a C+ por lo que la secuencia se haría infactible. Por esto es que se asume que *C+ precede preliminarmente a B-*. En la tabla de la **¡Error! No se encuentra el origen de la referencia.**, esto se expresa con la presencia de C+ en la lista de eventos de B-.

Si se realizan otras suposiciones, se pueden definir más relaciones de precedencia entre los eventos. Por ejemplo, se puede suponer que la distancia mínima del recorrido realizado entre dos eventos es justamente la distancia entre ambos. Esto es práctico cuando por ejemplo, los datos de entrada a disposición muestran a los distintos lugares de ejecución de eventos como puntos dentro de un plano, donde la distancia entre dos lugares es la correspondiente euclidiana entre los puntos. De esta manera, viendo el ejemplo de la **¡Error! No se encuentra el origen de la referencia.**, si bien $ET(D+) < LT(B+)$, puede darse que B+ preceda preliminarmente a D+, pues al sumar a $ET(D+)$ el tiempo de viaje directo entre D+ y B+, es decir $DRT(D+,B+)$, y el tiempo de servicio de D+, es decir $ts(D+)$, se puede dar que $ET(D+) + ts(D+) + DRT(D+,B+) > LT(B+)$, esto implica que desde cualquier punto dentro de la ventana de D+, es imposible llegar a tiempo a la ventana de B+, por ende es necesario que B+ este antes que D+ en una misma ruta.

Si bien la forma en que se establece la precedencia dependerá de las suposiciones que hagamos, hay ciertas relaciones lógicas entre eventos que tienen un esquema de precedencia dado. Sea la expresión $X => Y$ equivalente a "*X precede preliminarmente a Y"*, y $X \not=> Y$ equivalente a "*X no precede preliminarmente a Y"*, entonces se puede considerar lo siguiente:

- Si $(X => Y) \wedge (Y => X)$, entonces X e Y tienen una relación de *dependencia cíclica.*
- Si $(X \not=> Y) \wedge (Y \not=> X)$, entonces X e Y son *preliminarmente paralelos*.
- Si $(X => Y) \wedge (Y => Z)$, entonces $X => Z$, si es que ambos eventos están en la misma ruta de Y.

La dependencia cíclica implica la incompatibilidad de la existencia de eventos en una misma ruta. Cuando los eventos son preliminarmente paralelos, es posible evaluar el cambio en el orden de los mismos dentro de una ruta. La tercera relación apunta a la *transitividad* de la precedencia. Estos aspectos serán abordados de forma práctica en secciones posteriores.

Por último cabe destacar que como su nombre lo indica, la precedencia es *preliminar*, dado que sólo se analiza desde el punto de vista de dos eventos particulares y fuera del

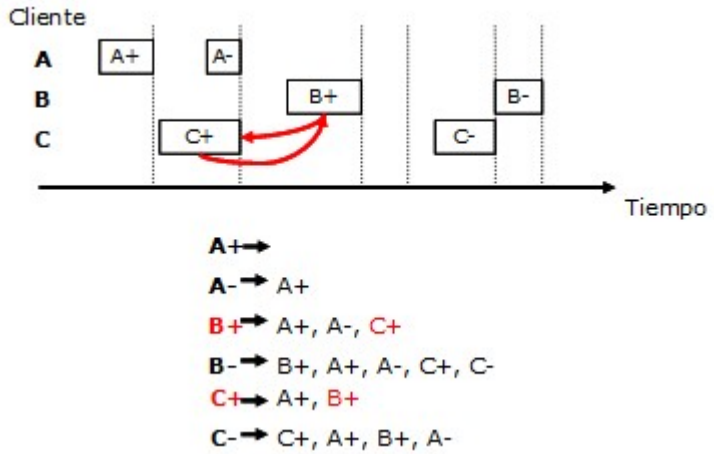
contexto de una ruta, salvo el caso de la transitividad. Realizar un análisis más detallado para un conjunto mayor de eventos, implicaría una revisión exhaustiva de las distintas posibilidades de combinar eventos en una ruta, lo que resultaría muy costoso en términos de recursos. La precedencia preliminar sirve para evitar los casos infactibles, no para determinar los casos factibles, que es tarea de otros elementos del marco de trabajo.

5.1.1.2. Incompatibilidad de clientes

La dependencia cíclica entre eventos provee una forma efectiva de identificar condiciones de infactibilidad, ya no sólo a nivel de rutas como la tabla de precedencia preliminar, sino también a nivel de asignación de clientes a vehículos o *clusters*.

En la Figura 5.2, se da que (*B*+=>*C*+) ^ (*C*+=>*B*+). Esta dependencia cíclica permite definir directamente la imposibilidad de colocar a dichos eventos en una misma ruta. Al mismo tiempo, esto implica que los clientes asociados a dichos eventos, B y C, nunca podrán asignarse al mismo vehículo de forma factible. De este modo se asume que *B* y *C son clientes incompatibles.*

Figura 5.2 Ejemplo de incompatibilidad entre clientes por dependencia cíclica.



La incompatibilidad de clientes provee un importante esquema de factibilidad por medio del cual se pueden generar más fácilmente asignaciones de clientes a vehículos. Esto es especialmente práctico en la generación de la población inicial del GA.

5.1.2. Mecanismos de modificación rutas

En esta parte del marco de trabajo, se evalúa la modificación de una ruta en base a la inserción, eliminación o el intercambio de un par de eventos en ella, considerando todas las restricciones importantes en este contexto, como son ventanas de tiempo y carga de los vehículos. Para la inserción y eliminación, se consideran los eventos asociados a un cliente, mientras que en el intercambio pueden ser eventos de clientes distintos. La factibilidad en las operaciones de inserción e intercambio se evalúa en parte usando la tabla de precedencia preliminar.

En los mecanismos propuestos, se consideró una modificación de la heurística de inserción factible descrita en el Framework MADARP propuesto por Cubillos [23]. Para entender adecuadamente los mecanismos, se hará una reseña de esta heurística, posteriormente se explicará la modificación, y por último se señalarán las diferencias en el funcionamiento de la inserción, eliminación y el intercambio.

5.1.2.1. Heurística de inserción de MADARP

La heurística se basa en un trabajo propuesto anteriormente por Jaw et al., en donde el autor define una serie de valores precalculados para los eventos que permiten evaluar qué tanto pueden ser anticipado o postergado el tiempo de llegada del vehículo al evento, con el fin de evaluar una posible inserción. Estos valores se describen del siguiente modo:

$$BUP(X_i) = Min(Min(AT(X_i) - ET(X_i)), SLK_0)$$
$$BDOWN(X_i) = Min(LT(X_i) - AT(X_i))$$
$$AUP(X_i) = Min(AT(X_i) - ET(X_i))$$
$$ADOWN(X_i) = Min(Min(LT(X_i) - AT(X_i)), SLK_{w+1})$$

En donde $0 < i < w+1$, SLK_0 y SLK_{w+1} son los periodos de *slack* posibles que preceden o siguen inmediatamente al bloque respectivo. $ET(X_i)$, $AT(X_i)$ y $LT(X_i)$ son el tiempo menor (extremo inferior de la ventana), actual (tiempo para el cual la llegada del vehículo fue programada) y mayor (extremo superior de la ventana) del evento X_i respectivamente. Cabe destacar que estos valores son acumulativos, es decir para X_i, los valores *BUP* y *BDOWN* se

calculan en base a todos los eventos anteriores a X_i, lo mismo con *AUP* y *ADOWN* para sus eventos posteriores.

En la propuesta de MADARP, a estos cálculos se suma el realizar operaciones de intersección de ventanas de tiempo para evaluar de manera más simple la inserción de los eventos asociados a un pasajero en una ruta. En términos generales, la intersección de un conjunto de ventanas consiste básicamente en comparar los extremos de las mismas. Para explicar mejor esto, consideremos los siguientes valores:

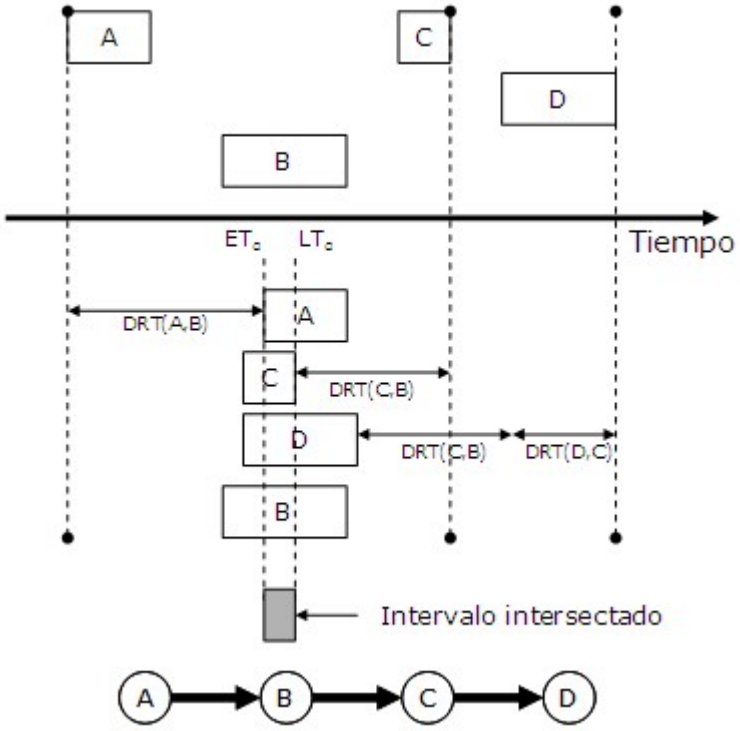
Sea $X = \{X_1, X_2, \ldots, X_n\}$ el conjunto de n eventos cuyas ventanas se intersectarán, entonces:

$$ET_\alpha = Max(ET(X_i)), \text{ con } 0 \leq i \leq n$$
$$LT_\alpha = Min(LT(X_i)), \text{ con } 0 \leq i \leq n$$

La intersección se basa en dejar fija la ventana de un evento X_i y desplazar todas las demás del conjunto en dirección a X_i. Este desplazamiento depende del contexto en que se aplique esta operación. Cuando se trata de evaluar eventos de forma independiente, corresponde a la distancia directa *DRT* entre el nodo de la ventana desplazada y el nodo de X_i más el tiempo de servicio *ts* del evento asociado a la ventana desplazada. Si la intersección se aplica en el contexto de una ruta, habrá que considerar las distancias de los nodos intermedios. Para simplificar las explicaciones, por ahora se asumirá que *ts* está contenido en *DRT*.

Por ejemplo, para desplazar un evento X_{i+3} hacia X_i, habría que restar a los extremos de X_{i+3} la distancia $d = DRT(X_{i+3}, X_{i+2}) + DRT(X_{i+2}, X_{i+1}) + DRT(X_{i+1}, X_i)$. Si se desplaza el evento X_{i-2} *a* X_i, entonces se suma a sus extremos $d = DRT(X_{i-2}, X_{i-1}) + DRT(X_{i-1}, X_i)$. En la Figura 5.3 se tiene un ejemplo gráfico de esto. Con las ventanas ya desplazadas, se verifica que $ET_\alpha < LT_\alpha$, lo que implica la existencia de intersección.

Figura 5.3 Intersección de ventanas a lo largo de una ruta.



Con esta idea de intersección, debe considerarse el bloque dentro de la ruta en el cual se realizará la inserción. Un bloque está delimitado en sus extremos por *slacks* o depósitos, y la inserción tanto el evento de recogida como de entrega para un cliente determinado debe realizarse dentro de un bloque. Si la inserción es válida del punto de vista del bloque a tratar, se deben considerar las ventanas de tiempo que se van a intersectar. En la Figura 5.4 se muestra un esquema gráfico de estas ventanas. Con el fin de identificarlas, el bloque en cuestión se divide en 3 sub bloques: el segmento anterior al evento de recogida del nuevo cliente, el segmento entre los eventos de recogida y entrega, y el segmento posterior al evento de entrega. De cierto modo, los eventos que unen estos bloques son puntos en los cuales deben considerarse las intersecciones. Para el evento A_M se utiliza la vntana de tiempo que considera sus eventos previos, esta se puede calcular del siguiente modo:

$$ET^A = AT(A_M) - BUP(A_M)$$
$$LT^A = AT(A_M) + BDOWN(A_M)$$

En C_1 el caso es análogo, pero utilizando los valores *AUP* y *ADOWN*. En B_1 se hace lo mismo que C_1, pero sólo considerando los eventos hasta B_N, esto requiere un nuevo cálculo

de los valores *AUP* y *ADOWN* para este segmento. Eventualmente, los eventos de recogida y entrega del cliente nuevo serán consecutivos, lo que implica que este cálculo no se considera. Además, las ventanas que se usarán con los eventos a insertar, naturalmente son las que define el cliente. Adicional a los intervalos mencionados, se debe tener en consideración las distancias que unen estos intervalos, las mismas también se muestran en la **¡Error! No se encuentra el origen de la referencia.**. Cabe destacar que la distancia D^B se puede obtener con $AT(B_N) - AT(B_1)$. Con las distancias y las ventanas, se puede realizar el proceso de intersección según lo explicado anteriormente. Si se toma a X^+ como el evento fijo, los cálculos que se realizan pueden seguir el siguiente esquema:

$$ET^1 = Max(ET^-, ET^C - D2^-)$$
$$LT^1 = Min(LT^-, LT^C - D2^-)$$

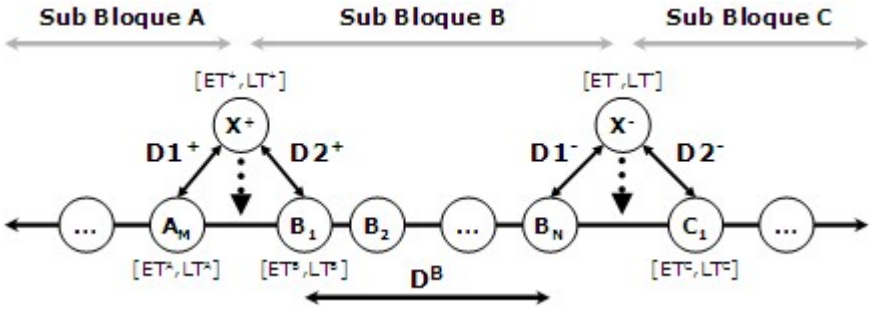
$$ET^2 = Max(ET^+, ET^A + D1^+)$$
$$LT^2 = Min(LT^+, ET^A + D1^+)$$

$$ET^3 = Max(ET^2, ET^B - D2^+)$$
$$LT^3 = Min(LT^2, ET^B - D2^+)$$

$$ET^F = Max(ET^3, ET^1 - D1^- - D^B - D2^+)$$
$$LT^F = Min(LT^3, ET^1 - D1^- - D^B - D2^+)$$

De esta manera, el intervalo $[ET^F, LT^F]$, si es válido, representa la intersección de las ventanas señaladas y el rango de factibilidad respecto al evento X^+, en el que un enrutamiento se puede programar (donde se puede insertar el *AT*). Cabe destacar que los cálculos anteriores se pueden hacer de manera conjunta, pero esto no ayuda a tener visibilidad de los eventos que, en caso de infactibilidad, no permitieron la inserción. Tener esta visibilidad puede ayudar a planear una nueva inserción de manera adecuada.

Figura 5.4 Elementos a considerar para la inserción de nuevos eventos.



5.1.2.2. Heurística propuesta

Una desventaja que tiene la heurística descrita anteriormente, es que no considera la eventual modificación de los bloques tras una inserción. Puede ocurrir por ejemplo, que en el punto donde se inserta el evento de recogida, se genere de forma factible un nuevo slack, dividiendo el bloque en otros dos. O en caso contrario, si el evento se inserta al principio del bloque, puede generar que el *slack* que exista en ese punto desaparezca, lo que implica la fusión del bloque anterior y el actual. Los únicos casos considerados en este contexto, son los que implican la inserción de cualquiera de los eventos en los extremos del bloque, sin la desaparición del *slack* asociado.

El considerar la modificación de la estructura de la ruta no es fácil, dado que cualquier variación en ella puede cambiar de muchas formas los eventos posteriores en la ruta y son muchos los casos nuevos a considerar, respecto a la situación original de no evaluar estos posibles cambios. Además, hay valores precalculados, en particular *AUP* y *ADOWN*, que dejan de ser útiles, dado que los mismos se generan en referencia al slack más próximo hacia delante, siendo que el mismo tras una modificación en la ruta puede desaparecer, por lo que la información que entrega no sería válida en ciertas situaciones. Considerando las dificultades mencionadas, la heurística se modificó en este trabajo de forma que sólo se consideran los valores precalculados en referencia a los intervalos previos en la ruta y que fuera aplicable no sólo a inserción, sino también a eliminación e intercambio de eventos.

La nueva heurística se basa en la idea de reconstruir la ruta modificada, desde el punto en que empiezan los cambios, hasta el último punto que puede ser afectado por ellos en la ruta, que no necesariamente corresponde al fin del bloque. La idea es empezar a realizar intersecciones de ventanas de tiempo a medida que se avanza en la ruta. En la Figura 5.5 se

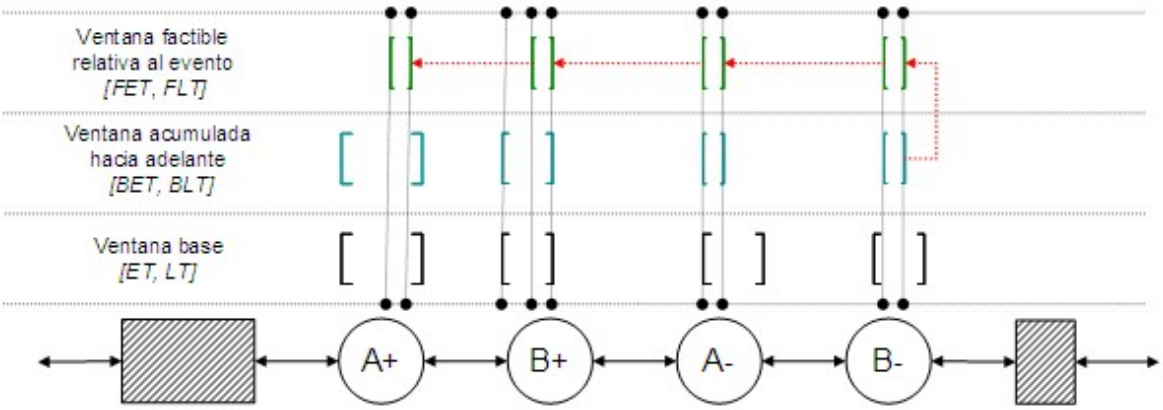
ejemplifican las ventanas que se obtienen al ir realizando las intersecciones. Las mismas se describen a continuación:

• **Ventana base (*[ET, LT]*)**, que corresponde a la ventana de tiempo que se entrega como dato de entrada en relación a la solicitud de transporte asociada al evento.

• **Ventana acumulada hacia adelante (*[BET, BLT]*)**, que corresponde al resultado de intersectar la ventana acumulada hacia adelante del evento anterior en la ruta con la ventana base del evento actual. Para el primer evento en la ruta, o el primero después de un slack, los extremos de esta ventana equivalen a los de la ventana base. Se trata de un dato precalculado en el evento, y respecto a la heurística de inserción de MADARP, es equivalente a *[AT-BUP, AT+BDOWN]*, es decir, hace referencia a las intersecciones previas en la ruta.

• **Ventana factible relativa al evento (*[FET, FLT]*)**, corresponde al resultado final de todas las intersecciones, cuando se llega al último evento de la ruta, o al final de un bloque, relativo al evento actual. Esta ventana representa el intervalo factible en el que *AT* puede colocarse. En este trabajo, se procurará siempre colocar *AT* al principio de este intervalo.

Figura 5.5 Ventanas de tiempo consideradas en el funcionamiento de la nueva heurística.



Cuando una ruta sufre modificaciones, estos datos deberían actualizarse de forma adecuada. Sin embargo, ya sea en la inserción, en la eliminación o en el intercambio de eventos, la ventana acumulada hacia adelante se va calculando a medida que se va

realizando la verificación, por ello, si estos valores se conservan, sólo bastaría con actualizar la ventana factible para cada evento.

En la Figura 5.6 se muestra un esquema de cómo se realiza este proceso de verificación, para el caso de la inserción. En los otros casos el proceso es análogo, salvo los eventos que cambian en la ruta. Los pasos a seguir son los siguientes:

• Se identifica el evento previo al primero que resultará modificado tras los cambios aplicados a la ruta (Figura 5.6 (a)). Se debe considerar la situación de que en este evento puede generarse un *slack* o desaparecer uno existente. La ventana acumulada hacia adelante del evento identificado se usará como punto de partida para empezar a realizar las intersecciones en los eventos que siguen en la ruta.

• Desde el primer evento a modificar en la ruta, se va generando un *segmento modificado*, que contendrá los eventos de la ruta original, modificados de acuerdo a la operación que se esté aplicando (Figura 5.6 (b)). Este segmento también contiene los nuevos eventos que tendrá la ruta, en el caso de la inserción. La idea es ir tomando cada uno de los eventos de la ruta original, copiarlo en el segmento y realizar la intersección de su ventana base con la ventana acumulada hacia delante del evento previo (desplazada de forma adecuada hacia el evento actual) y con ello realizar las verificaciones necesarias para definir si el segmento construido hasta ese momento es factible. Este segmento estará compuesto de una parte *base* y otra *adicional*. La parte base corresponde a los eventos en la ruta que es inevitable que cambien o que sean reevaluados en base a los cambios realizados en la ruta, para el caso de la inserción por ejemplo, este segmento base está compuesto por los nuevos eventos insertados más los eventos que quedan entre ellos. La parte adicional del segmento corresponde a los eventos externos al segmento base que también resultan modificados tras la operación respectiva en la ruta. Eventualmente, este segmento adicional podría no existir. El mismo se extenderá hasta el punto en la ruta en donde, al evaluar al siguiente evento que debería ser insertado en el segmento, se genera un *slack* siendo que en la ruta original ya existía uno en dicha posición. Como los *slacks* separan la ruta en segmentos independientes desde el punto de vista de cómo resultan las ventanas acumuladas posteriores, entonces no es necesario seguir actualizando eventos, por ello la construcción del segmento modificado se detiene y adicionalmente se identifica el evento posterior al último evento agregado al

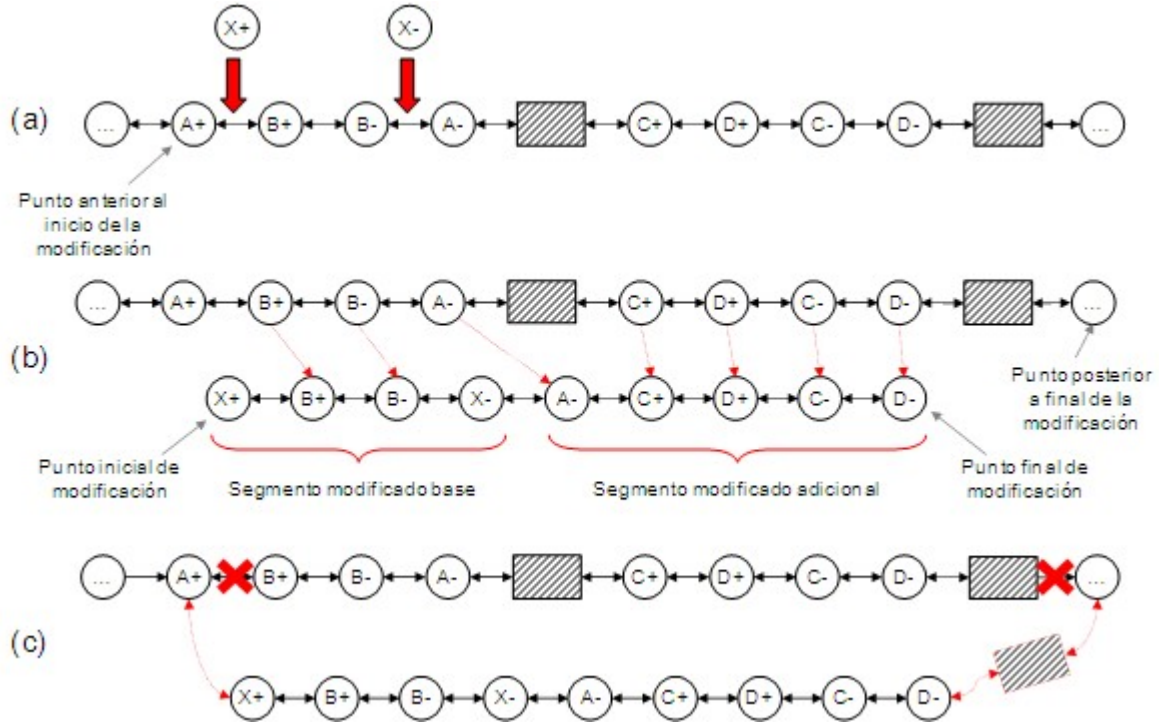
segmento. Cabe destacar que en la generación de este segmento, hay *slacks* de la ruta original que pueden desaparecer, u otros nuevos que pueden aparecer. En el caso de la Figura 5.6 (b), el *slack* que estaba después del evento A+ desaparece.

• Se reemplaza la parte de la ruta cambiada por el segmento modificado (Figura 5.6 (c)). En lo que es eliminación, eventualmente el segmento modificado puede estar vació, por ejemplo cuando se elimina un bloque que contiene un solo cliente. En este caso sólo se conectan el evento anterior al inicio de la modificación con el posterior.

Por último, cabe destacar que la ruta original no será modificada hasta que se haya comprobado que todos los cambios que se generarán en ella son factibles. De hecho, es posible separar el proceso de verificación y de actualización de la ruta, siempre y cuando se guarde la información del punto anterior al inicio de la modificación (considerando además si en él se genera un *slack* o no), el punto posterior al final de la modificación y naturalmente el segmento modificado. En la actualización final de la ruta, se deben considerar las siguientes situaciones:

• Como el segmento modificado contiene los eventos actualizados de la ruta en lo que respecta a la ventana acumulada hacia adelante, sólo bastaría actualizar las ventanas factibles en relación a cada evento del segmento modificado.

Figura 5.6 Evaluación de los cambios generados en una ruta tras un proceso de inserción.



• En caso de que en el punto anterior al inicio de la modificación se genere un *slack* siendo que ya existía uno en esa posición de la ruta, no será necesario realizar actualizaciones de la ventana factible salvo las del segmento modificado. En los otros casos se deberán actualizar los eventos previos al punto anterior del inicio de la modificación, hasta el *slack* anterior o depósito inicial que este en la ruta.

5.1.2.3. Inserción de pares

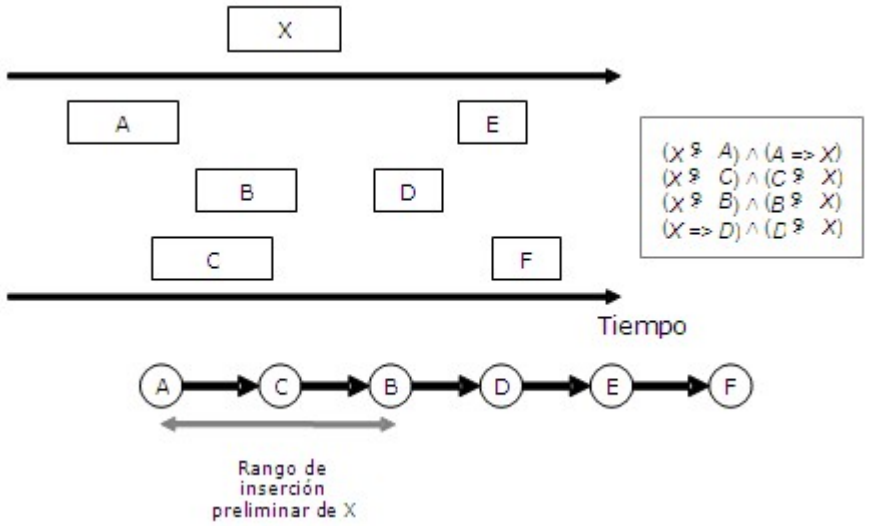
En la inserción de eventos, siempre se generará un segmento modificado, que puede contener como mínimo dos eventos (los que se insertarán) y como máximo todos los eventos en la ruta desde el punto de inserción hasta el final de la misma, incluyendo a los eventos que se insertarán.

La inserción de un par de eventos (que conforman un cliente) puede hacerse en distintas posiciones según las restricciones lo permitan. En un bloque con *d* paradas (2 por cada cliente), existen hasta *(d+1)(d+2)/2* posibles inserciones, considerando que la recogida del cliente siempre debe estar antes de la entrega. Sin embargo, en la práctica, tanto el evento de recogida como el de entrega tienen un rango de posiciones de inserción que se puede

determinar gracias a la información de precedencia preliminar. Para un evento *X*, ese rango en primera instancia empieza con el último evento en la ruta que precede preliminarmente a *X* y termina con el primer evento al que *X* precede preliminarmente en la ruta (ver 5.1.1.1). En la Figura 5.7 se ejemplifica esto. La propiedad de transitividad de la precedencia preliminar asegura que se preservará la factibilidad de la ruta respecto a los eventos fuera del rango considerado, si es que el evento se insertara en ese rango. Si *B* y *C* son preliminarmente paralelos a *X*, *A* => *X* y *X* => *D*, entonces la inserción de *X* podrá evaluarse entre las posiciones 0 a la 2, considerando que *X* se insertará delante del evento asociado a la posición. En otro caso, si *A* => *X* y *X* => *C* (es decir, no hay eventos preliminarmente paralelos a *X*), la inserción de *X* sólo podría evaluarse en el punto 0. También puede darse que *C* sea preliminarmente paralelo a *X*, pero *B* no lo sea, en ese caso el inicio del rango de inserción seria la posición de *B*.

De esta forma, el evento de recogida podrá insertarse en el rango $[s_p, e_p]$ y el evento de entrega podrá insertarse en el rango $[s_d, e_d]$, con $s_d \geq s_p$. Las posiciones pueden ser escogidas de distintas formas. Una alternativa es una selección aleatoria para los dos eventos, otra es elegir ordenadamente una posición y aleatoriamente la otra, siempre procurando recordar las que anteriormente se han escogido. La aleatoriedad en este contexto sirve principalmente para probar nuevas opciones de rutas cuando se realizan estas evaluaciones. Sin embargo, hay que procurar que la selección evite que el punto para el evento de recogida sea mayor que el punto para el evento de entrega.

Figura 5.7 Rango de inserción preliminar de un evento (X).



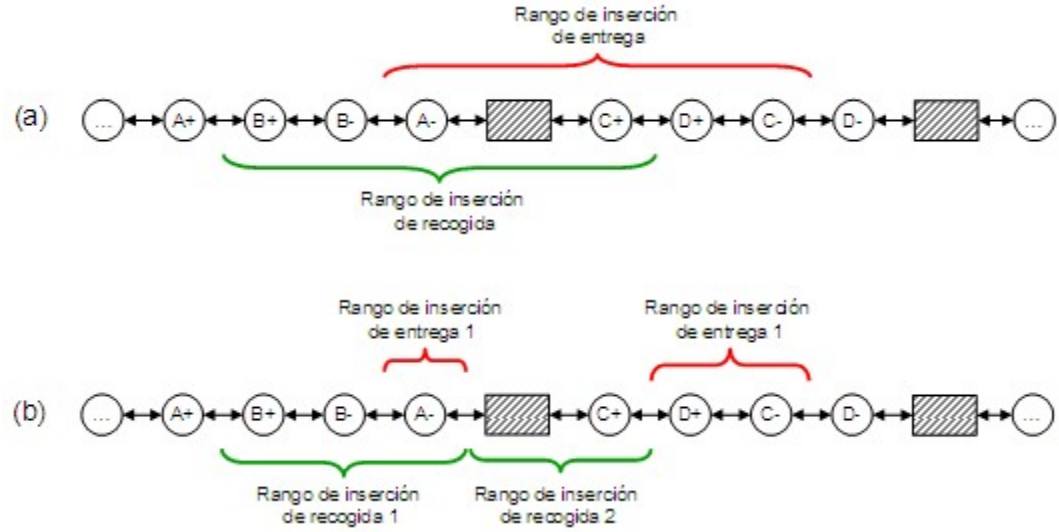
Con el mecanismo de inserción presentado, es factible encontrar rangos de inserción que incluyan *slacks* en ellos, lo que entrega casos adicionales a MADARP, pues al considerar que la inserción de los eventos puede modificar la estructura de la ruta, no se puede asumir que la inserción de eventos sólo debe darse dentro de un bloque. Sin embargo, los rangos se pueden segmentar si ellos incluyen un *slack absoluto*. Estos últimos corresponden a *slacks* que existirán independientemente de los cambios generados en la ruta, siempre y cuando los eventos que lo generan no cambien. Un esquema de cómo se puede dividir los rangos se presenta en la **¡Error! No se encuentra el origen de la referencia.**.

En (a) el *slack* que está contenido en el rango no es absoluto, pues si se intersectan los eventos *A*- y *C*+ usando sus ventanas de tiempo base, eventualmente se genera una intersección. Esto implica que ante un cambio en la ruta, la ventana acumulada hacia delante de *A*- puede modificarse de forma que la misma, al intersectarse con la ventana base de *C*+, genere una intersección y por ello elimine el *slack*. El caso de (b) es distinto, al intersectar las ventanas base de *A*- y *C*+, se genera igualmente un *slack*. Esto implica que si ocurre cualquier modificación en la ruta, mientras no se inserte ningún otro evento entre *A*- y *C*+, ese *slack* siempre existirá, por ende el mismo divide la ruta en ese punto en dos partes independientes. Esto permite dividir el rango de inserción.

Al considerar la inserción de dos eventos de un cliente, puede ocurrir que la división de los rangos a raíz de uno o más *slacks* absolutos, genere que para ciertas partes de la ruta sólo se

tenga un rango de inserción, ya sea el de la entrega como el de la recogida. En estos casos, la inserción en esa parte de la ruta se vuelve infactible, por ende sólo se deben considerar las partes de la ruta que tienen rangos de inserción tanto para la recogida como para la entrega. Cabe destacar también que cuando un *slack* absoluto genera la división de un par de rangos de inserción, es necesario que se pueda evaluar todas las inserciones que posiblemente generen la eliminación del mismo. Esto se logra incluyendo la posición del primer evento que genera el *slack* en los dos rangos de inserción del evento de recogida que se generan en la división. En la **¡Error! No se encuentra el origen de la referencia.** (a) por ejemplo, los dos rangos de inserción del evento de recogida contienen a la posición de *A-*.

Figura 5.8 Rangos de inserción según los slacks que contienen.



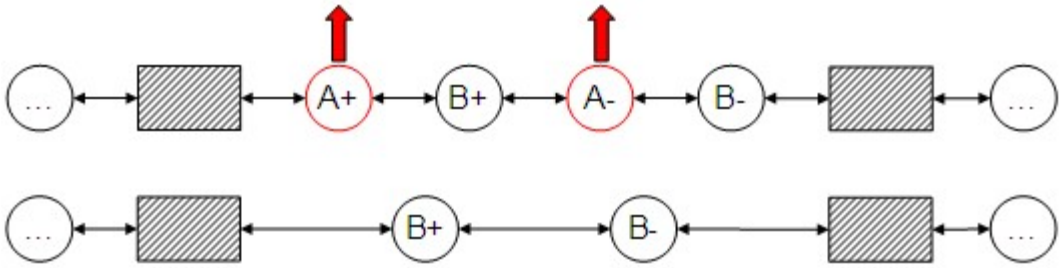
5.1.2.4. Eliminación de pares

En la eliminación, el segmento modificado puede no existir. Esto ocurre cuando por ejemplo, se elimina un par de eventos que son los únicos dentro de un bloque, dado que ello genera la eliminación del bloque y los que eventualmente estén antes o después se mantienen intactos. También el segmento no existe cuando se eliminan dos eventos contiguos que están al final de la ruta. En este sentido, el segmento modificado puede como mínimo no contener eventos y como máximo contener todos los eventos en la ruta desde el

evento posterior al evento de recogida hasta el final, sin incluir los eventos eliminados claro.

En comparación a la inserción, la eliminación es más simple, considerando que no se deben encontrar intervalos. Basta con recorrer la ruta y encontrar los eventos que se desean eliminar. En la Figura 5.9 se muestra un esquema de la eliminación de a pares.

Figura 5.9 Eliminación de pares.

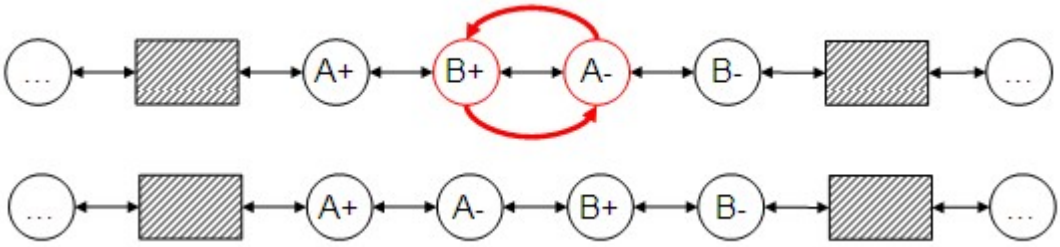


5.1.2.5. Intercambio de pares

En el intercambio, se cambia el orden de un par de eventos contiguos en la ruta. El segmento modificado puede contener como mínimo un par de eventos, los que se intercambian, y como máximo todos los eventos de la ruta desde el primero que se intercambia hasta el final.

Para evaluar la factibilidad del intercambio de un par de eventos, se evalúa si el primero precede preliminarmente al segundo. Si no es así, los eventos se asumen como intercambiables, dado que al principio el segundo ya esta después que el primero, por ende no lo precede preliminarmente. En la práctica acá se comprueba que los eventos sean paralelos preliminarmente. Un esquema del intercambio de eventos se muestra en la Figura 5.10.

Figura 5.10 Intercambio de pares.



5.1.3. Generación de soluciones factibles

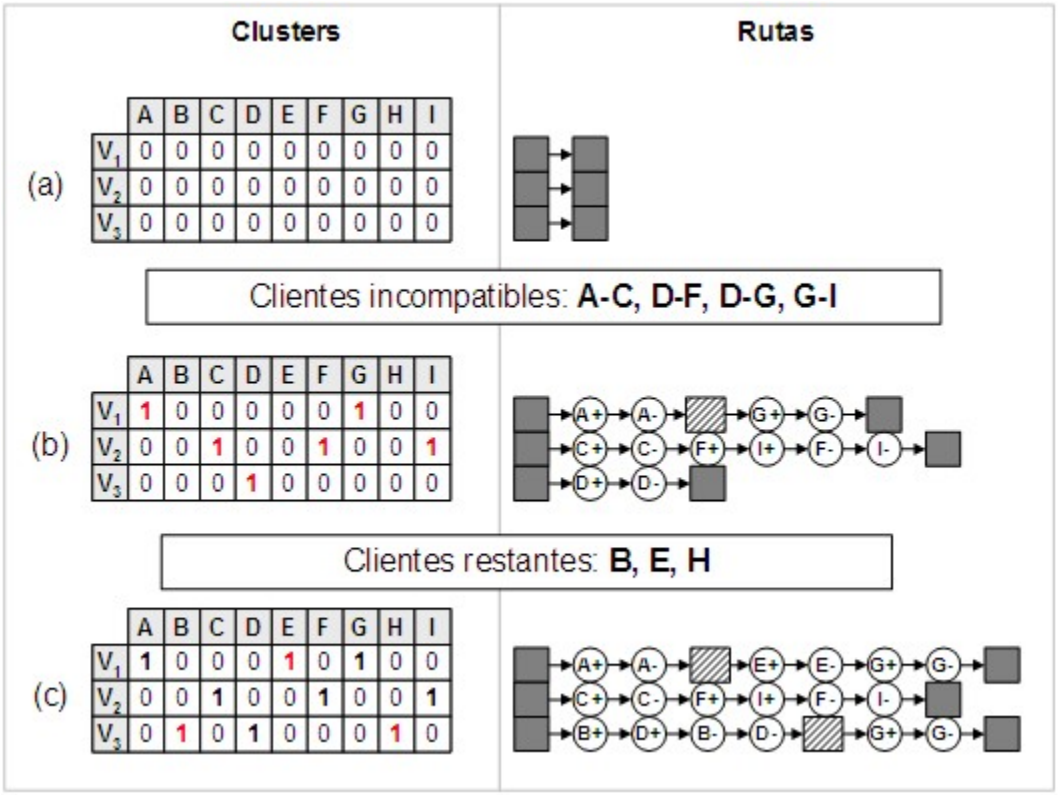
Con la identificación de dependencias cíclicas en los eventos y con la herramienta de inserción de eventos en una ruta, es posible construir un mecanismo efectivo para generar de forma aleatoria, soluciones que estén dentro del marco de factibilidad exigido por el problema. La idea de esta parte del marco de trabajo es entregar soluciones genéricas para que las mismas puedan ser aplicadas a los genotipos de los modelos a usar en el algoritmo.

El proceso consta de dos etapas que se ilustran en la Figura 5.11:

• **Generación de solución factible base** (Figura 5.11 (a) y (b)). En esta etapa, lo que se hace básicamente es tratar las dependencias cíclicas, evitando que los clientes incompatibles se asignen a las mismas rutas. Se parte con un clúster o asignación de pasajeros a vehículos vacía. También cada clúster tiene asociado una ruta que parte vacía. Se van tomando uno a uno los clientes que presentan incompatibilidad con otros clientes y se elige una ruta en donde no hayan clientes que puedan entrar en conflicto. Se evalúa y realiza la inserción en la ruta respectiva usando el mecanismo descrito en 5.1.2.3. Esto se hace hasta completar todos los clientes que presentan algún tipo de incompatibilidad.

• **Generación de solución factible final** (Figura 5.11 (c)). Con la solución factible base generada, se tiene cierta cantidad de clientes que aun no se han insertado en las rutas. Los mismos se van tomando y se van insertando en las rutas de la solución de forma aleatoria. Si al tratar de insertar en alguna ruta se produce algún error de factibilidad, se prueba con otra ruta distinta en la solución.

Figura 5.11 Generación de una solución factible.



5.1.4. Representación o Genotipo

Este elemento está vinculado a cómo en el cromosoma, una solución de DARPTW se codifica y también cómo ella se decodifica para que, en términos del problema, se obtenga un conjunto de pasajeros asignados a un vehículo y dependiendo del caso, las rutas consideradas en la solución. En el contexto del modelo en general, el genotipo juega un rol fundamental, pues es uno de los elementos que tiene mayor interacción o genera mayor dependencia respecto a otros elementos, muy en particular con lo que es la recombinación. Adicionalmente, en él es que se pueden ver afectadas una serie de restricciones del problema. En el transcurso de esta investigación hubo una gran preocupación por pensar un mecanismo de codificación que permitiera de forma directa abordar algunas restricciones que en DARPTW se podrían considerar como "secundarias" y que no eran abordadas por otros elementos del marco de trabajo. Cabe destacar que los genotipos mencionados en este punto están profundamente ligados a los operadores de recombinación, por lo que para entender el funcionamiento del modelo en general, no se debe aislar ambos elementos.

Para los modelos propuestos en esta investigación, son dos las representaciones que se consideraron. Una está basada en un GA genérico que se describió en 3.3.3, el LLGA. La otra es una representación experimental basada en rutas, propuesta en este trabajo.

5.1.4.1. Representación tipo LLGA

En este genotipo, la consideración principal va por el asunto de que pasajeros se asignan a que vehículos, lo que se puede definir como un tipo de *clustering*. En este sentido, aparece una restricción respecto a que un pasajero puede ser asignado a un sólo vehículo, y a la vez este vehículo debe ser el que ejecute los servicios de recogida y entrega de dicho pasajero. En representaciones como las de **¡Error! No se encuentra el origen de la referencia.** y **¡Error! No se encuentra el origen de la referencia.**, después de una recombinación, eventualmente se generan soluciones no válidas, entre cuyos motivos más visibles esta precisamente la asignación de pasajeros adecuada. En estos casos era necesario aplicar ciertas correcciones posteriores o verificaciones previas a los cromosomas resultantes.

Considerando esto, para el genotipo se consideró un modelo al estilo del LLGA descrito en **¡Error! No se encuentra el origen de la referencia.**. En este esquema, los elementos del cromosoma no tienen un ordenamiento rígido, la incorporación del *locus* al gen, permite ordenarlos de distintas maneras, obteniendo una solución idéntica. Un gen estaría compuesto por el *locus*, el pasajero a asignar, y el vehículo al que se le asigna. El problema de esto último es que habría que realizar validaciones respecto a que en un cromosoma estén presente no sólo todos los *locus*, si no también todos los pasajeros. Por ello es que, considerando que el genotipo se enfocará a abordar sólo la asignación de pasajeros a vehículos, el *locus* se considerará equivalente al identificador del pasajero. Lo descrito anteriormente corresponde a la *capa exterior* del cromosoma, que es donde se contienen los genes y opera la recombinación, sin embargo, este cromosoma también tiene una *capa interna*, que corresponde a las rutas que están asociadas a cada vehículo representado en los genes. Cuando se modifican los *clusters*, también se modifican las rutas, en base a todos los mecanismos presentados anteriormente. En la **¡Error! No se encuentra el origen de la referencia.** se muestra un ejemplo de este genotipo, donde los pasajeros son denotados con las letras A, B, C y D, y los vehículos con V1, V2 y V3.

Figura 5.12 Genotipo basado en el LLGA y su decodificación.

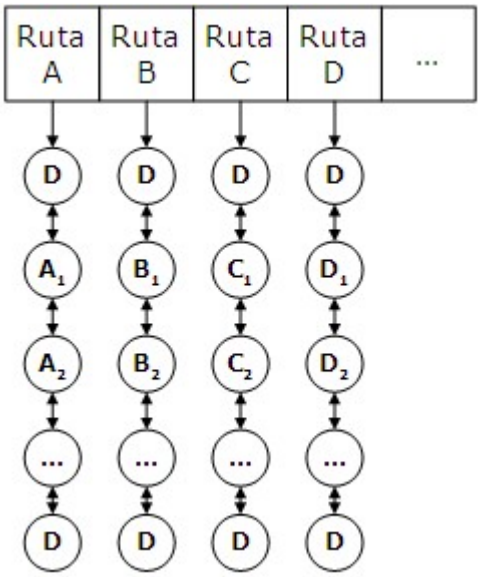
5.1.4.2. Representación basada en rutas

La representación tipo LLGA se enfoca en exponer el *clustering* en sus genes, como su capa externa. A diferencia de ello, la representación basada en rutas se enfoca a exponer explícitamente que rutas tiene considerada la solución, abordando tanto ese aspecto como el de asignación al mismo tiempo.

La forma que esta codificado un cromosoma en este genotipo es bastante directa y simple, en la Figura 5.13 se muestra. Corresponde a una lista de secuencias de eventos, en donde cada secuencia corresponde a una lista de eventos doblemente enlazada y hace referencia a una ruta que se ejecuta por un vehículo, definiendo explícitamente el enrutamiento asociado. Cada ruta definida en el individuo es independiente, desde el punto de vista que para cada una se pueden calcular tiempos que pueden ser utilizados de forma general en la solución para la evaluación de la misma. Naturalmente, la cantidad de elementos en la lista indica la cantidad de vehículos considerados en la solución.

En este contexto, no hay una funcionalidad mayor en este genotipo, la explotación de su funcionalidad se hace por el operador de recombinación apropiado, que será explicado en lo que sigue.

Figura 5.13 Genotipo basado en rutas.



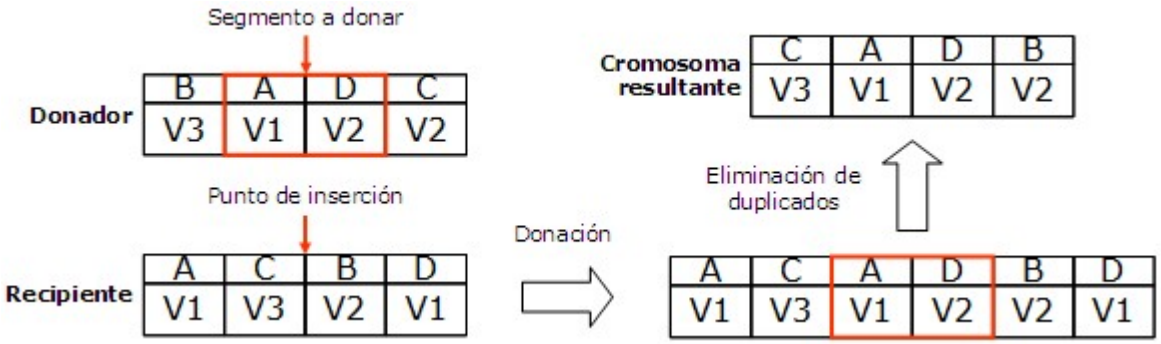
5.1.5. Recombinación

Para los modelos propuestos, se consideraron dos operadores de recombinación que tienen relación directa con las representaciones, una es la recombinación tipo LLGA y la otra es una recombinación de rutas.

5.1.5.1. Recombinación tipo LLGA

En concordancia con la estructura presentada en el genotipo LLGA, para la recombinación se pretende utilizar el operador que ese GA considera. Este, aparte de ofrecer los mecanismos que el algoritmo en si entrega, permitirá mantener una consistencia respecto a las restricciones de cómo se asignan los pasajeros a los vehículos. Como se explicó en 3.3.3, este operador inserta un segmento de uno de los padres en un punto específico del otro, lo importante es que tras esta operación, los elementos repetidos fuera del segmento donado se eliminan. Cuando se considera a los clientes como *locus*, esto permite que en todo el cromosoma exista solo un gen por cliente, lo que permite mantener controladas las restricciones correspondientes. En la Figura 5.14 se muestra un esquema de cómo funcionaría este operador, para el genotipo correspondiente.

Figura 5.14 Operación de recombinación para el cromosoma tipo LLGA.



Uno de los problemas respecto a esta recombinación, es que se pueden pasar a llevar restricciones relativas al enrutamiento, es decir, las asociadas a la capa interna, al recombinar dos individuos. Si bien algunas restricciones son aseguradas en la capa externa, no son las más críticas. En este sentido, se adoptará una estrategia de **verificación previa**. Se trata de evaluar el segmento de cromosoma que planea donarse, definiendo cual será la nueva asignación de pasajeros, evaluar cómo se modificarán las rutas usando las herramientas de soporte del marco de trabajo y permitir o no la recombinación final según al resultado, todo esto en base a la estructura interna de rutas y **clusters** del cromosoma. Para cada uno de los genes dentro del segmento a donar, se realiza lo siguiente:

- Se identifica el cliente a cambiar dentro de la asignación, el cual se define en el gen.

- Se identifican los vehículos (y las rutas asociadas) de origen y destino del cliente. El de origen corresponderá al que está asignado actualmente el cliente en el cromosoma recipiente. El vehículo de destino corresponderá al que está definido en el gen del segmento a donar.

- Se evalúa la eliminación del cliente desde la ruta origen.

- Se evalúa la inserción del cliente en la ruta destino, en puntos aleatorios para la recogida y la entrega.

Si para alguno de los genes, esta secuencia de operaciones es infactible en algún punto, se asume que el intento de recombinación es infactible también, procurando iniciarlo nuevamente en base a la selección de un nuevo segmento desde el cromosoma donante.

5.1.5.2. Recombinación basada en rutas

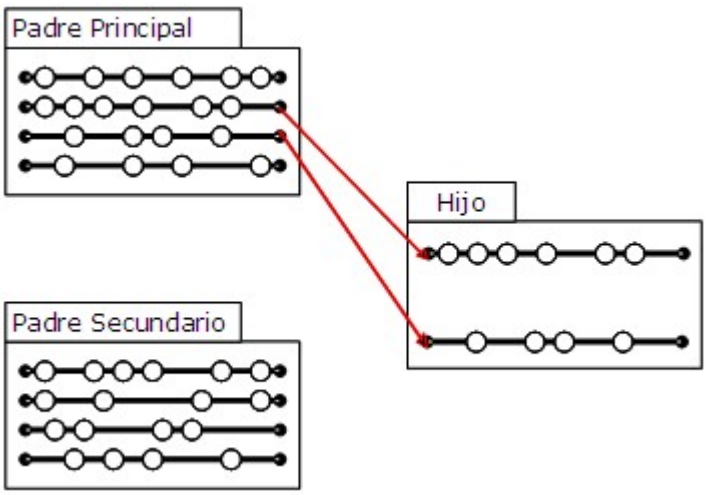
Este operador trabaja en conjunto a la representación basada en rutas. En la práctica, es este operador el que le da mayor funcionalidad al genotipo. La idea principal de la recombinación basada en rutas, es la reutilización en cierto grado de las mismas, de forma que se puedan rescatar las que presentan una estructura de mejor calidad a medida que avanzan las generaciones en el algoritmo.

En esta recombinación se tienen dos padres para generar un hijo. Ambos padres asumen roles distintos: uno es el padre "principal" y otro es el "secundario". Esta diferencia se explicará en las etapas del proceso de recombinación, pero tiene relación más que nada con la forma que ambos entregan rutas al hijo. Como se genera un solo hijo, al igual que en la recombinación del modelo LLGA, los padres intercambian roles para generar otro hijo.

La ejecución de la recombinación tiene 3 etapas generales que se detallan a continuación.

- **Donación directa de rutas** (Figura 5.15). En esta etapa, el padre principal dona una parte menor de sus rutas al hijo, de forma directa. Esto implica que las rutas no sufren ninguna modificación, simplemente se copian en el hijo. La idea es que la cantidad de rutas donadas de este modo no sea muy grande, para que en la etapa siguiente, las donaciones del padre secundario no se modifiquen demasiado.

Figura 5.15 Primera etapa en recombinación para el cromosoma basado en rutas.



• **Donación filtrada de rutas** (Figura 5.16). A estas alturas, el cromosoma hijo ya tiene algunos clientes considerados en su solución. La idea en esta etapa es donar rutas desde el padre secundario de forma que las mismas no entren en conflicto con la asignación de pasajeros definida en el hijo actualmente. Para eso, se toman todas las rutas del padre secundario y se filtran, a través de la eliminación de los clientes que están actualmente en el hijo y que las rutas del padre secundario contienen. De las rutas filtradas, se escogen en orden las que tienen mayor cantidad de clientes para ser traspasadas al hijo. La cantidad de rutas que se donan dependerá del límite de vehículos del problema. El escoger las rutas procesadas con más clientes sirve para hacer más fácil el proceso de la reparación en la siguiente etapa. Se avanza a esta última sólo si faltaron clientes por insertar en la solución. De otro modo, se asume que el hijo está completo.

• **Reparación por inserción** (Figura 5.17). Con las etapas anteriores, es posible que no todos los clientes necesarios para completar la solución se hayan donado al hijo en las rutas. Para arreglar esto, se construye una lista de los clientes que faltan y los mismos se van insertando de forma aleatoria en las rutas que ya tiene la solución, utilizando las herramientas de soporte del modelo.

Figura 5.16 Segunda etapa en recombinación para el cromosoma basado en rutas.

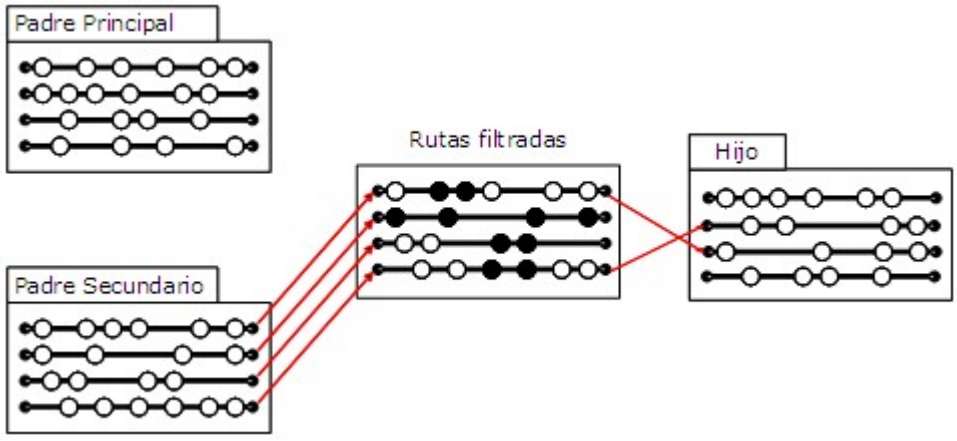
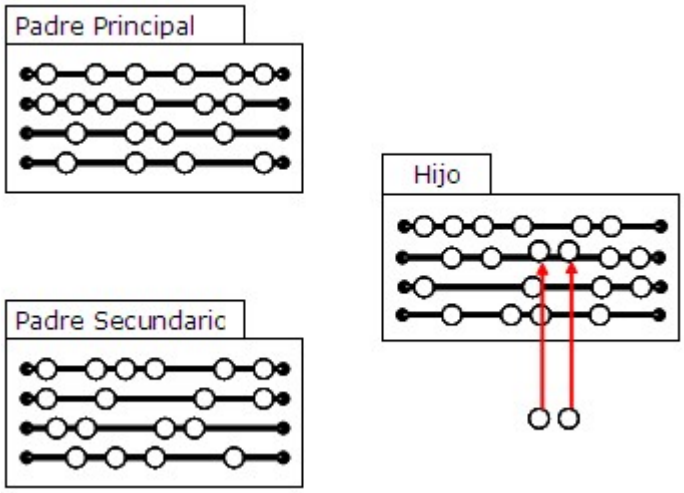


Figura 5.17 Tercera etapa y final en recombinación para el cromosoma basado en rutas.



Cabe destacar que esta recombinación no es completamente factible. En la segunda etapa por ejemplo, puede que a ninguna ruta del padre secundario se le puedan eliminar todos los clientes necesarios para traspasarlas al hijo. A la vez, puede que en la tercera etapa no se pueda insertar todos los clientes que faltan en la solución. En estos casos, se vuelve a ejecutar el proceso, considerando las condiciones de aleatoriedad que presenta.

5.1.6. Evaluación o Fenotipo

En este elemento se describe el mecanismo por el cual se evalúa un cromosoma dado, para lo que es posteriormente el proceso de selección. En el caso de DARPTW la función de evaluación a utilizar es simplemente la función objetivo del modelo en 2.9.

Para esta función, hay que tener presente algunas consideraciones importantes. Por un lado, los factores de las constantes de peso pueden tener un gran impacto en el funcionamiento de los modelos. En general se evalúan 5 elementos en la función objetivo:

- Tiempo total de viaje de los vehículos.
- Tiempo de duración de los slacks.
- Cantidad de vehículos.

• Exceso en el tiempo de viaje para los clientes, equivalente a la diferencia entre tiempo de viaje directo desde punto de recogida a punto de entrega y el tiempo de viaje realizado en la ruta para llegar desde un punto al otro.

• Tiempo de espera de los clientes, esto es la diferencia entre el tiempo actual de llegada a ejecutar un servicio y el extremo inferior de la ventana de tiempo establecida para ese servicio.

Para cada uno de estos factores se asigna un peso respectivo. En el caso de la cantidad de vehículos, depende mucho de los datos de entrada usados, dado que algunos definen como limitante la cantidad de vehículos a utilizar, por ende este factor no sería considerado.

5.1.7. Selección

Este elemento permite la aplicación del resultado de una evaluación en definir las oportunidades de selección de un cromosoma. Para los modelos presentados, se utilizó la selección por torneo que se describió en 3.1.4. Las ventajas de este tipo de selección son por una parte, la posibilidad de manejar la presión de la selección a través del tamaño del torneo y la otra es que el mapeo entre la prioridad de selección de los integrantes del torneo es directa: los que tienen una función de evaluación con resultado menor, son los que tienen mayores posibilidades de ganar el torneo.

5.1.8. Mutación

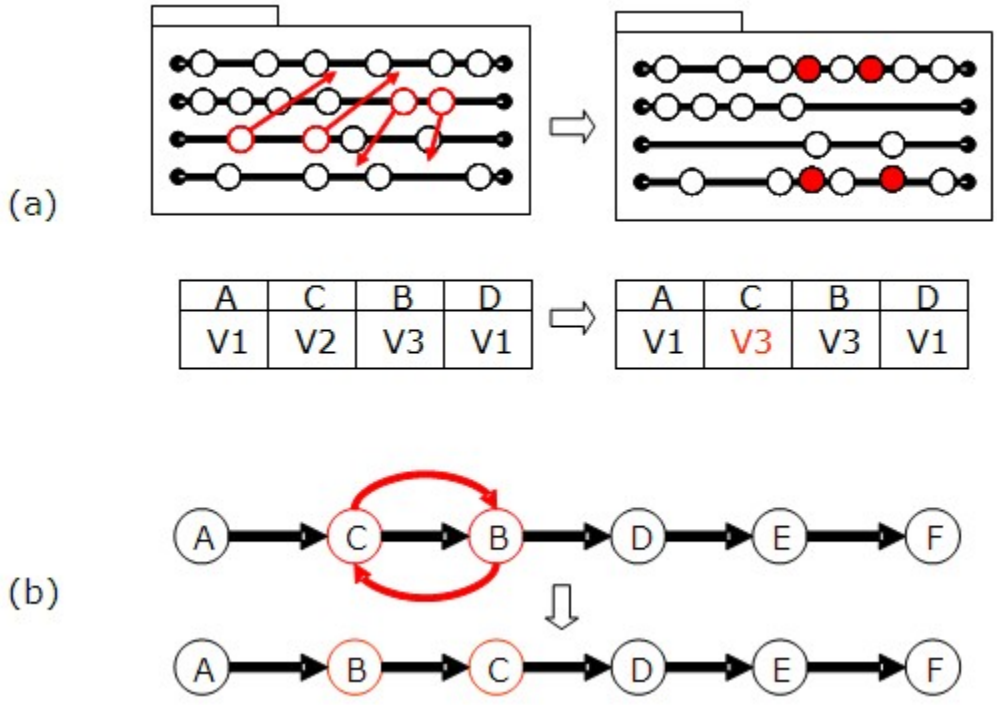
En la mutación, se harán pequeñas modificaciones a la codificación de un cromosoma para obtener variaciones en los individuos. Este operador, al igual que la recombinación, está sujeto a problemas de factibilidad, los que se pretenden abordar en parte usando la información de incompatibilidad de clientes. En la Figura 5.18 se muestran los dos tipos de mutación que son considerados. Cabe destacar que ambos tipos de mutación se tratan de forma independiente, cada una con su propia probabilidad de ocurrencia, dada las diferencias en su naturaleza.

• **Mutación de clusters** (Figura 5.18 (a)). En esta mutación se cambia el vehículo de uno de los pasajeros en el cromosoma. Cada vez que se cambia uno de ellos, se realiza una

verificación correspondiente, en base a la eliminación del cliente desde el vehículo de origen y la inserción del mismo en el vehículo destino, además de considerar las incompatibilidades entre clientes. Si el resultado es negativo, se intenta con otro vehículo. En caso de que no se pueda cambiar a ningún otro vehículo, la mutación se anula. Cabe destacar que la probabilidad aplicada a esta mutación es para cada cliente en la solución, es decir, por cada cliente se evalúa la probabilidad de realizar el cambio de vehículo. Por esto es que dicha probabilidad no puede ser muy grande, y si la mutación es infactible, el no realizarla no será problema considerando que otro cliente en la solución puede mutar también.

• **Mutación de rutas** (Figura 5.18 (b)): la idea de este operador es realizar un intercambio entre eventos que tienen posibilidades de cambiar de orden. Para esta mutación se usa directamente la herramienta explicada en 5.1.2.5. En este caso, la probabilidad se evalúa para cada ruta en el problema, de modo que si se efectúa la mutación, se selecciona un punto "intercambiable" de la ruta de forma aleatoria y se evalúa el intercambio. Si este es infactible, se intenta con otro punto. En caso de no poder intercambiar ninguno de los eventos en la ruta, se anula la mutación.

Figura 5.18 Los dos tipos de mutación considerados en los modelos.



5.1.9. Población inicial

Este elemento del modelo está vinculado directamente a los mecanismos que se utilizarán para tener un conjunto de soluciones para DARPTW con las cuales el modelo pueda empezar a operar. El impacto de la población inicial en el rendimiento de un GA es importante, y en problemas como el tratado en esta investigación la factibilidad de las soluciones puede afectar mucho la rapidez con que las soluciones se encuentren.

Para la población inicial se utiliza la herramienta de generación de soluciones factibles explicada en 5.1.3. La forma en que estas soluciones se incorporan a los cromosomas, dependerá de cada genotipo.

5.2. Estructura de los modelos

Como se mencionó anteriormente, los modelos propuestos combinarán distintos elementos descritos en la sección anterior. Los mismos se muestran en la Figura 5.19 y la Figura 5.20. La estructura y diferencias principales en ellos estarán contempladas en el genotipo y el operador de recombinación. Otros elementos, como la población inicial, generación de rutas y la mutación serán aplicados de forma equivalente para cada modelo.

Figura 5.19 Esquema del modelo LLGA.

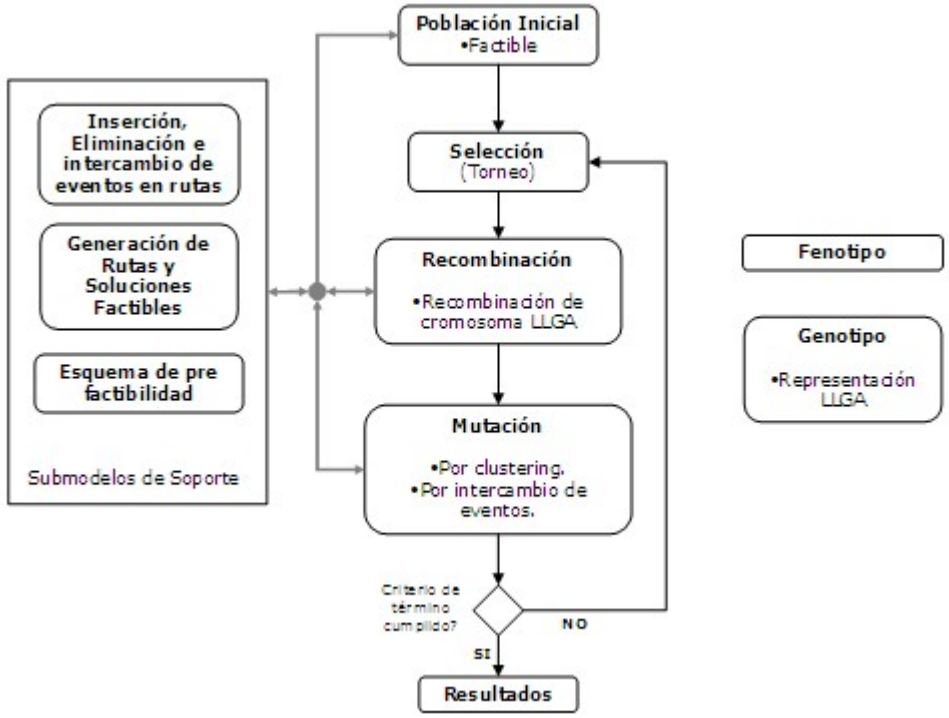
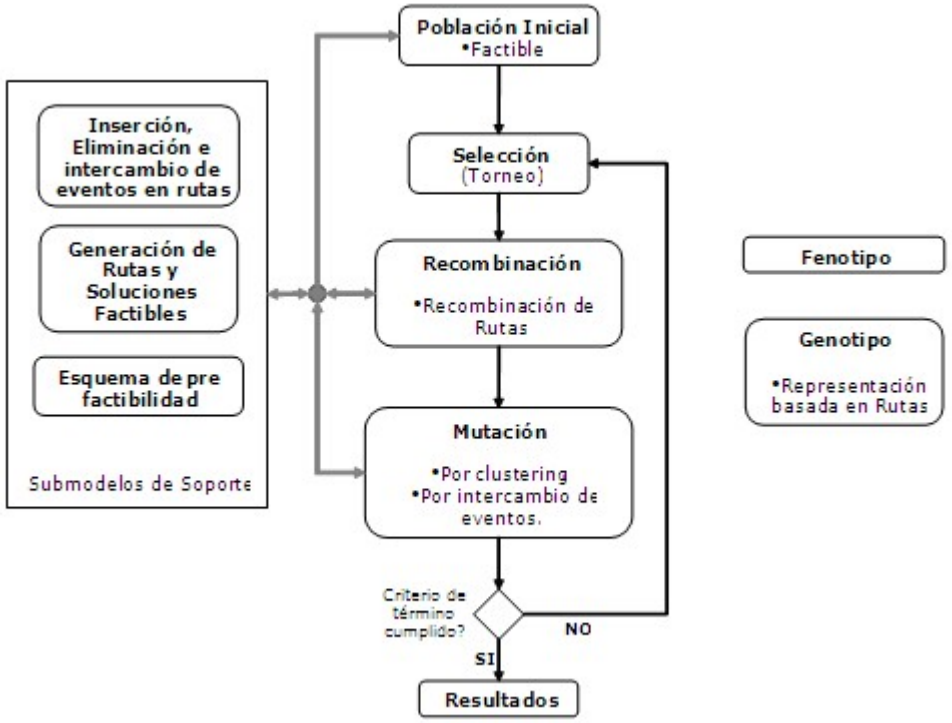


Figura 5.20 Esquema del modelo basado en rutas.



El primer modelo está basado en el cromosoma LLGA y su operador de recombinación respectivo. El segundo modelo está basado en la representación de rutas, en el que se usa también el operador compatible.

Hay un conjunto de elementos que serán comunes para los dos modelos y que no tendrán variantes importantes, estos son:

- Todo los submodelos de soporte.
- Operador de selección (torneo).
- El Fenotipo.
- Los operadores de mutación
- La generación de población inicial.

6. Resultados experimentales

En este capítulo se revisarán los resultados obtenidos con los modelos propuestos en este trabajo. Adicionalmente se señalan una serie de detalles importantes que fueron considerados de forma previa a sus respectivas implementaciones. La importancia de estos detalles radica en su impacto directo en la funcionalidad y la operabilidad del algoritmo. Más que características que describen cierta instancia de DARPTW, se trata de observaciones que por obligación hay que considerar al realizar las comparaciones pertinentes con otros trabajos que se encuentren en la literatura.

Primero se describirán los datos de entrada utilizados en la evaluación de los modelos, tanto en las etapas de calibración como en las etapas finales donde se realizarán las pruebas definitivas con él. Posteriormente se hará un resumen respecto a las variables más influyentes en la especificación del problema, señalando como podrían afectar al rendimiento del algoritmo. Finalmente se mostrarán resultados de ejecución y algunas comparaciones importantes, especialmente entre los modelos desarrollados.

6.1. Especificación de los datos de entrada

Una problemática común que aparece al estudiar y desarrollar heurísticas para el problema DARPTW, sean cual sean las características de implementación que se definan, son los datos de entrada que se van a utilizar para evaluar la propuesta que se presente. Esto se debe a la carencia de algún conjunto de datos que tengan carácter estándar, o que en su defecto, hayan sido ampliamente utilizados en estudios relacionados, para poder realizar comparaciones consistentes y adecuadas.

En estudios como [15] se han utilizado datos basados en escenarios reales y restringidos en cierto grado, tanto en el aspecto de las locaciones y sus distancias, como en lo que se refiere a las solicitudes de transporte, en base a sistemas funcionales. Naturalmente, tener

acceso a este tipo de datos de entrada es complejo, si no se tiene la posibilidad de interactuar con ellos de forma directa, o tener un acceso adecuado a su entorno funcional.

Una de las alternativas manejadas en esta investigación, es la de construir un conjunto de datos de entrada, basándose en otros problemas relacionados a DARPTW en donde si se tengan entradas de carácter más estándar. En particular, se pensó respecto a las instancias de M. M. Solomon que se utilizan ampliamente en estudios de VRP. Si bien este problema es más genérico que DARPTW, con la inserción de las restricciones que correspondan, el conjunto de datos podría ser adaptado para ser tratado correctamente en un estudio de DARPTW.

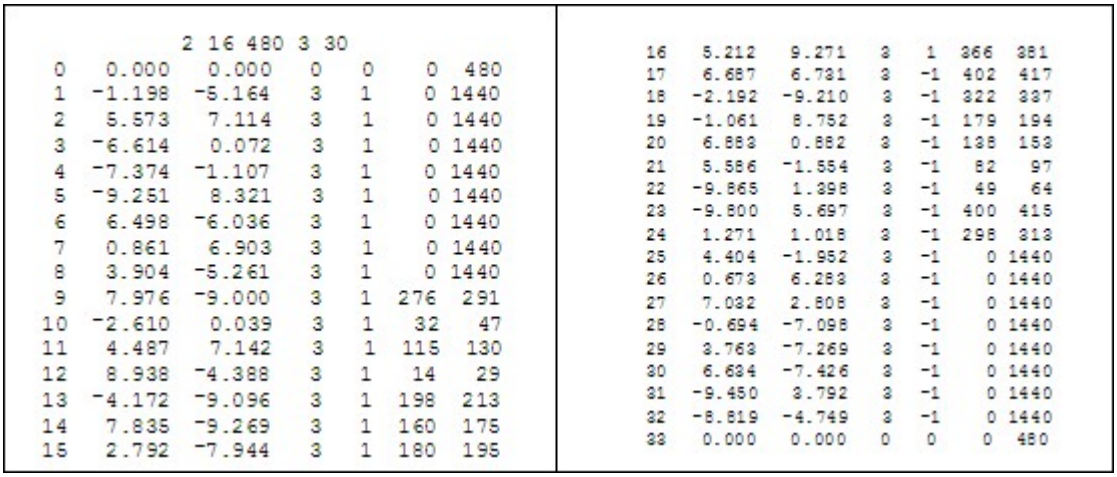
Sin embargo, en la literatura se puede encontrar un conjunto de datos de entrada que en los últimos años se ha utilizado en ciertos estudios de heurísticas enfocadas a resolver DARPTW. Estas instancias han sido desarrolladas por Cordeau y Laporte y se pueden encontrar en http://neumann.hec.ca/chairedistributique/data/darp/. Estos datos están divididos en dos subconjuntos. Uno es utilizado en [15] para un algoritmo de Branch-and-Cut. El otro es usado en [25] para un algoritmo de búsqueda Tabú y en [1] para un Algoritmo Genético y será el considerado en este proyecto, tanto en lo que se refiere a la calibración de los parámetros asociados al GA, como a las pruebas finales del mismo. Los datos contienen todas las características deseadas para la instancia de DARPTW evaluada en este trabajo. A continuación se explicará la estructura y las propiedades de estos datos de entrada.

6.1.1. Conjunto Nº1 Branch-and-Cut

Estos datos corresponden a dos conjuntos de instancias generadas de forma aleatoria de 16 a 96 clientes. La **¡Error! No se encuentra el origen de la referencia.** ejemplifica una instancia de 16 clientes para este grupo y la **¡Error! No se encuentra el origen de la referencia.** muestra un resumen del mismo. En una instancia de *n* clientes, para las entradas de *1* a *n/2*, sus respectivos clientes se consideran como “de salida”, y para las entradas de *n/2 +1* a *n*, sus respectivos clientes se consideran como “de entrada” (revisar esta clasificación en **¡Error! No se encuentra el origen de la referencia.**). Las coordenadas de las locaciones de recogida y entrega para todas las instancias son escogidas

aleatoria e independientemente en un cuadrado de dimensiones *[-10,10] x [-10,10]* de acuerdo a una distribución uniforme, además el depósito de vehículos se encuentra en el centro de este cuadro. Para cualquier par de locaciones en el cuadro, el tiempo de viaje y el costo del viaje entre ellas equivalen a la distancia euclidiana entre ambos puntos.

Figura 6.1 Ejemplo de instancia Branch-and-Cut para 16 clientes.



```
          2 16 480 3 30
 0   0.000   0.000   0   0     0  480
 1  -1.198  -5.164   3   1     0 1440
 2   5.573   7.114   3   1     0 1440
 3  -6.614   0.072   3   1     0 1440
 4  -7.374  -1.107   3   1     0 1440
 5  -9.251   8.321   3   1     0 1440
 6   6.498  -6.036   3   1     0 1440
 7   0.861   6.903   3   1     0 1440
 8   3.904  -5.261   3   1     0 1440
 9   7.976  -9.000   3   1   276  291
10  -2.610   0.039   3   1    32   47
11   4.487   7.142   3   1   115  130
12   8.938  -4.388   3   1    14   29
13  -4.172  -9.096   3   1   198  213
14   7.835  -9.269   3   1   160  175
15   2.792  -7.944   3   1   180  195
16   5.212   9.271   3   1   366  381
17   6.687   6.731   3  -1   402  417
18  -2.192  -9.210   3  -1   322  337
19  -1.061   8.752   3  -1   179  194
20   6.883   0.882   3  -1   138  153
21   5.586  -1.554   3  -1    82   97
22  -9.865   1.398   3  -1    49   64
23  -9.800   5.697   3  -1   400  415
24   1.271   1.018   3  -1   298  313
25   4.404  -1.952   3  -1     0 1440
26   0.673   6.283   3  -1     0 1440
27   7.032   2.808   3  -1     0 1440
28  -0.694  -7.098   3  -1     0 1440
29   3.763  -7.269   3  -1     0 1440
30   6.684  -7.426   3  -1     0 1440
31  -9.450   3.792   3  -1     0 1440
32  -8.819  -4.749   3  -1     0 1440
33   0.000   0.000   0   0     0  480
```

A cada nodo/entrada se asocia una ventana de tiempo $[e_i, l_i]$. Para un cliente "de salida" i, una ventana de tiempo se genera escogiendo un número l_{n+i} en el intervalo $[60, T]$ y luego asignando $e_i = l_{n+i} - 15$, con T el horizonte de planificación. En el caso de un cliente "de entrada", e_i se escoge dentro del intervalo de tiempo $[0, T - 60]$ y $l_i = e_i + 15$. Para la generación de la ventana de tiempo en el nodo origen para los clientes "de salida", y la del nodo destino para los clientes "de entrada", se utilizan los mecanismos expuestos en **¡Error! No se encuentra el origen de la referencia.**. En el primer conjunto de instancias, la carga máxima de un vehículo es 3, la diferencia de carga en recogida o entrega es 1 y el tiempo de servicio es 3 para cada cliente. Adicionalmente, *MRT* equivale a 30 min. En el segundo conjunto de instancias, la carga máxima de un vehículo es 6, la diferencia de carga es escogida aleatoriamente bajo una distribución uniforme en {1,..,6}, siendo este valor equivalente al tiempo de servicio. Además, *MRT* equivale a 45 min. El primer conjunto simula la utilización de pequeños automóviles para el transporte de pasajeros, mientras que el segundo conjunto simula la utilización de mini-buses.

Tabla 6.1 Resumen de instancias Branch-and-Cut:

Primer conjunto						**Segundo conjunto**					
Instancia	$\|K\|$	n	T	Q	MRT	**Instancia**	$\|K\|$	n	T	Q	MRT
a2-16	2	16	480	3	30	b2-16	2	16	480	6	45
a2-20	2	20	600	3	30	b2-20	2	20	600	6	45
a2-24	2	24	720	3	30	b2-24	2	24	720	6	45
a3-18	3	18	360	3	30	b2-18	3	18	360	6	45
a3-24	3	24	480	3	30	b3-24	3	24	480	6	45
a3-30	3	30	600	3	30	b3-30	3	30	600	6	45
a3-36	3	36	720	3	30	b3-36	3	36	720	6	45
a4-16	4	16	240	3	30	b4-16	4	16	240	6	45
a4-24	4	24	360	3	30	b4-24	4	24	360	6	45
a4-32	4	32	480	3	30	b4-32	4	32	480	6	45
a4-40	4	40	600	3	30	b4-40	4	40	600	6	45
a4-48	4	48	720	3	30	b4-48	4	48	720	6	45
a5-40	5	40	480	3	30	b5-40	5	40	480	6	45
a5-50	5	50	600	3	30	b5-50	5	50	600	6	45
a5-60	5	60	720	3	30	b5-60	5	60	720	6	45
a6-48	6	48	480	3	30	b6-48	6	48	480	6	45

a6-60	6	60	600	3	30	b6-60	6	60	600	6	45
a6-72	6	72	720	3	30	b6-72	6	72	720	6	45
a7-56	7	56	480	3	30	b7-56	7	56	480	6	45
a7-70	7	70	600	3	30	b7-70	7	70	600	6	45
a7-84	7	84	720	3	30	b7-84	7	84	720	6	45
a8-64	8	64	480	3	30	b8-64	8	64	480	6	45
a8-80	8	80	600	3	30	b8-80	8	80	600	6	45
a8-96	8	96	720	3	30	b8-96	8	96	720	6	45

6.1.2. Conjunto Nº2 Tabu Search

Este conjunto corresponde a 20 instancias generadas en base a asunciones reales. La información relacionada con el ancho de las ventanas de tiempo, la capacidad de los vehículos, la duración de la ruta y el *MRT* fue provista por la Comisión de Transito de Montreal (MTC), según se señala en [25]. En la Figura 6.2 se muestra un ejemplo de 24 clientes para el primer grupo de este conjunto de instancias. En la Tabla 6.2 se presenta un resumen con las características de las instancias, considerando ambos grupos.

Figura 6.2 Ejemplo de instancia Tabu Search para 24 clientes.

	3 48 480 6 90					
0	-1.044	2.000	0	0	0	1440
1	-2.973	6.414	10	1	0	1440
2	-3.066	0.546	10	1	0	1440
3	5.164	0.547	10	1	0	1440
4	-1.317	6.934	10	1	0	1440
5	-6.741	6.832	10	1	0	1440
6	4.891	0.627	10	1	0	1440
7	0.524	2.226	10	1	0	1440
8	-6.500	7.723	10	1	0	1440
9	-0.417	-0.157	10	1	0	1440
10	2.303	1.164	10	1	0	1440
11	2.548	0.629	10	1	0	1440
12	-4.261	-2.639	10	1	0	1440
13	-7.667	9.934	10	1	325	358
14	-2.067	5.789	10	1	111	152
15	-5.204	0.657	10	1	395	421
16	-4.138	5.082	10	1	386	401
17	-9.194	2.759	10	1	86	114
18	-6.512	3.021	10	1	409	426
19	1.860	9.672	10	1	454	470
20	-4.094	8.321	10	1	175	202
21	-3.776	-3.333	10	1	416	453
22	2.377	2.908	10	1	147	177
23	-4.303	2.045	10	1	471	499
24	-3.530	-2.490	10	1	321	346
25	-5.476	1.437	10	-1	258	287
26	-4.933	3.337	10	-1	329	361
27	5.740	2.382	10	-1	209	252
28	-2.275	5.541	10	-1	416	460
29	-5.662	7.334	10	-1	305	349
30	-3.856	-0.370	10	-1	432	458
31	-1.678	1.954	10	-1	202	236
32	-1.156	1.161	10	-1	225	252
33	-4.655	9.797	10	-1	102	123
34	1.623	0.932	10	-1	260	276
35	0.129	0.735	10	-1	178	215
36	-2.640	2.953	10	-1	381	397
37	0.435	1.469	10	-1	0	1440
38	-5.066	-2.313	10	-1	0	1440
39	-2.283	-0.981	10	-1	0	1440
40	-7.110	-1.862	10	-1	0	1440
41	-0.785	3.207	10	-1	0	1440
42	1.188	-2.493	10	-1	0	1440
43	-1.893	-2.373	10	-1	0	1440
44	-1.192	1.175	10	-1	0	1440
45	2.984	1.163	10	-1	0	1440
46	1.227	-5.581	10	-1	0	1440
47	-3.793	-2.161	10	-1	0	1440
48	4.288	-0.297	10	-1	0	1440

Se trata de instancias generadas aleatoriamente que consideran de 21 a 144 clientes. En una instancia de *n* clientes, para las entradas de *1* a *n/2*, sus respectivos clientes se consideran como "de salida", y para las entradas de *n/2 +1* a *n*, sus respectivos clientes se consideran como "de entrada". Las coordenadas de las locaciones se generan en base a un procedimiento especial, en el que se crean conjuntos de vértices en base a un determinado número de puntos base o "semillas". Se generan *2n* vértices dentro de un cuadrado de dimensiones *[-10,10] x [-10,10]*. Para cada entrada, el tiempo de servicio es 10 y la variación de carga, sea recogida o entrega, es de 1. La coordenada del depósito de vehículos corresponde al promedio de los puntos semilla usados para generar todos los vértices y se indica en la entrada 0. Para cualquier par de locaciones en el cuadro, el tiempo de viaje y el costo del viaje entre ellas equivalen a la distancia Euclidiana entre ambos puntos.

A cada nodo/entrada se asocia una ventana de tiempo $[e_i, l_i]$. Según como se generan estas ventanas, el conjunto de datos se divide en dos grupos: en el primero, la ventana definida (la de origen para clientes "de entrada" y la de destino para los "de salida") se establece asignando a e_i un número aleatorio uniforme en el rango *[60, 480]* y a l_i otro número aleatorio en el rango $[e_i + 15, e_i + 45]$; En el segundo grupo, las ventanas son más anchas, pues se asigna a e_i un número aleatorio uniforme en el rango *[60, 480]* y a l_i otro número en

el rango $[e_i + 30, e_i + 90]$. En todas las instancias, *MRT* equivale a 90, la duración máxima de la ruta es 480 y la capacidad del vehículo es 6.

Tabla 6.2 Resumen de instancias Tabu Search:

Primer grupo						**Segundo grupo**					
Instancia	**|K|**	***n***	***T***	***Q***	***MRT***	**Instancia**	**|K|**	***n***	***T***	***Q***	***MRT***
pr01	3	24	480	6	90	pr11	3	24	480	6	90
pr02	5	48	480	6	90	pr12	5	48	480	6	90
pr03	7	72	480	6	90	pr13	7	72	480	6	90
pr04	9	96	480	6	90	pr14	9	96	480	6	90
pr05	11	120	480	6	90	pr15	11	120	480	6	90
pr06	13	144	480	6	90	pr16	13	144	480	6	90
pr07	4	36	480	6	90	pr17	4	36	480	6	90
pr08	6	72	480	6	90	pr18	6	72	480	6	90
pr09	8	108	480	6	90	pr19	8	108	480	6	90
pr10	10	144	480	6	90	pr20	10	144	480	6	90

6.2. Variables influyentes del problema

Existen numerosas posibilidades para definir de qué forma se tratará DARPTW cuando se pretende desarrollar un algoritmo o una heurística particular. El problema es bastante amplio y posee muchas restricciones importantes, de hecho, muchos detalles pueden ser

considerados o dejados de lado. Sin embargo, es vital definir bien esto último cuando los detalles en cuestión afectan directamente el funcionamiento del algoritmo, en aspectos como por ejemplo, la facilidad de otorgar factibilidad a las soluciones.

Así mismo, si lo que se pretende es realizar un estudio comparativo, se debe evaluar hasta qué punto se aplicará la estructura del problema usada en otros trabajos que se encuentran en la literatura. Esto permite acotar adecuadamente las comparaciones, establecer márgenes de diferenciación con otros estudios y justificar según corresponda el funcionamiento del algoritmo o heurística desarrollada respecto a las otras revisadas.

En esta sección se pretende revisar aspectos de la formulación del problema, que están relacionados en cierto grado y que será importante tener en cuenta al momento de presentar resultados comparativos, por las razones señaladas anteriormente.

6.2.1. ¿Single depot o multi depot?

Para problemas de transporte que involucren el enrutamiento de vehículos desde y hasta puntos específicos, es decir, los depósitos, es importante definir si el escenario contemplará uno o más de ellos.

En general, cuando se tiene más de un depósito, es decir, cuando estamos ante un escenario *multi-depot*, la factibilidad de las soluciones puede verse restringida considerablemente. En primer lugar, si lo que se aplica es *clustering* (como se hace en parte de este trabajo), no hay facilidad de definir directamente desde que depósito partirá un vehículo y en cual terminará su ruta. Se pueden realizar asunciones en relación a los lugares recorridos en la ruta, por ejemplo se puede decir que el depósito inicial será el más cercano en distancia al primer nodo visitado, mientras que el depósito final será el más cercano en distancia al último nodo visitado. Sin embargo, este método puede fallar cuando se consideran otras restricciones importantes. Por ejemplo, es común que los depósitos tengan una capacidad limitada, por lo que es natural que al final de una planificación, sea infactible que todos o gran parte de los vehículos terminen sus rutas en un depósito en particular. Así mismo, puede que los tiempos de viaje desde el depósito inicial asumido al primer nodo de la ruta y/o el tiempo de viaje desde el último nodo de la ruta al depósito final asumido, se

salgan de los límites de tiempo de una ruta, o del horizonte de planeamiento establecido. Si se realiza una planificación directa de la ruta, se dejarían de lado las asunciones, pero las restricciones anteriormente mencionadas seguirían influyendo en la factibilidad de las soluciones.

En este trabajo se considerará el otro caso, es decir, sólo existirá un depósito, lo que implica un escenario *single-depot*. La razón de esto es más que nada práctica: los datos de entrada que se utilizarán, están pensados para escenarios single-depot. Si bien este estudio pretende seguir ciertos parámetros del que se presentan en [15], en donde lo que se considera es un escenario *multi depot*, los datos de entrada necesitarían ser modificados de forma consistente, lo que podría representar un esfuerzo que no vale la pena asumir, considerando que otros estudios como [4], [15] y [25] si trabajan con escenarios *single depot*, por lo tanto se pueden establecer puntos en común con ellos.

6.2.2. ¿*Slacks* con o sin pasajeros?

Como se describe en 2.7, hay dos formas distintas de representar la estructura de una ruta. En la primera (Figura 2.3), la ruta puede ser dividida en bloques de planificación, donde un bloque se considera como el conjunto de eventos entre dos periodos de inactividad (o *slacks*) del vehículo, considerando que las solicitudes de recogida y entrega de un cliente dado, deben estar contenidas en un sólo bloque. Esto implica que en los tiempos de inactividad, el vehículo no puede tener pasajeros a bordo. En la segunda forma pasa lo contrario: el vehículo si puede tener pasajeros a bordo al estar detenido esperando la ejecución de algún otro evento.

El considerar uno u otro caso para tratar DARPTW puede influir de dos formas importantes al comportamiento de una heurística:

• Al utilizar bloques de planificación, la factibilidad de las soluciones se verá restringida importantemente. En general, por asuntos de precedencia en el tiempo, los eventos podrán insertarse en partes específicas de una ruta, siempre dependiendo de los clientes que ella considere. En este sentido, el tener la restricción de que los eventos de recogida y entrega

para cualquier cliente siempre estén en el mismo bloque, puede dificultar la inserción de un cliente en una ruta cuando hay *slacks* que potencialmente separarán ambos eventos.

• Si se considera la segunda forma, es más fácil generar rutas factibles, sin embargo, es obligación considerar el costo en términos de calidad de servicio por tener pasajeros esperando dentro de un vehículo detenido. Esto se traduce en la modificación de la función objetivo del problema, en donde se tienen que agregar los parámetros necesarios que midan la insatisfacción del cliente en este sentido. Adicionalmente, hay que considerar que la cantidad de alternativas de combinación del problema aumenta al no establecer la restricción en los *slacks*. Esto puede afectar directamente el rendimiento de las heurísticas utilizadas.

Como se ve, cada situación tiene sus ventajas y sus desventajas en términos de cómo afectan la definición del problema. En el caso de este estudio, se optó por abordar la primera situación, en que se utilizan bloques de planificación. Hay dos razones para esta decisión: una se basa en la idea de seguir un esquema similar al del estudio presentado en [15] en donde también se considera esta estructura de rutas. Por otro lado, esta forma de establecer las condiciones representa un punto de partida para la implementación de los distintos mecanismos que se presentan en este trabajo, lo que posibilita su futura modificación en estudios futuros, con tal de establecer puntos de comparación en torno al funcionamiento de los algoritmos en otros escenarios asociados a esta restricción.

Al momento de realizar comparaciones con [4] y [25], que no consideran bloques de planificación, habrá que tener en cuenta las ventajas y desventajas de cada situación.

6.2.3. ¿Ventanas de tiempo blandas o rígidas?

La forma en que las ventanas de tiempo sean consideradas en un problema como DARPTW, constituye uno de los aspectos más importantes a tener en cuenta. Cuando las ventanas son blandas, se permite la violación de tiempo en la ejecución de sus eventos asociados. Al contrario, si son rígidas, el programar una ruta fuera de una ventana de tiempo correspondiente a algún evento asociado, implica automáticamente la infactibilidad de la ruta pertinente.

Los factores que están en juego cuando se escoge uno de los dos tipos de ventana, son exactamente los mismos que en la definición de la estructura de la ruta, señalados en 6.2.2. Por un lado, se puede restringir considerablemente el número de soluciones factibles. Por otro lado, se requiere considerar el costo que tiene la violación de las ventanas en la función objetivo, junto con el hecho que la cantidad de combinaciones y el espacio de búsqueda aumenta su extensión. En este sentido, la misma lógica se aplicó para escoger qué situación se abordará en este trabajo: se utilizarán ventanas de tiempo rígidas siguiendo el esquema presentado en [15]. Además, se considerarán las ventajas y desventajas al realizar comparaciones con los estudios [4] y [25], que se basan en ventanas de tiempo blandas.

6.3. Calibración de los modelos

Para una correcta medición de la eficiencia con que un Algoritmo Genético trabaja, es necesario considerar los numerosos parámetros que una ejecución del mismo involucra. Distintas combinaciones de estos parámetros podrán arrojar resultados en distintos niveles de calidad, por ello es importante realizar un análisis completo de estos parámetros y su comportamiento, de forma previa a la realización de las pruebas finales que se usen como referencia para mostrar el funcionamiento del algoritmo. Naturalmente, el objetivo en esta etapa de calibración, es buscar aquellos valores de los parámetros (o rangos de ellos) con los que se obtengan los resultados de mejor calidad. Es importante considerar que estas ejecuciones siempre están en el contexto de la utilización de recursos que deben ser racionalizados, por ejemplo, lo que es el tiempo.

Los parámetros considerados en la etapa de calibración de los algoritmos, son los siguientes:

- Tamaño de la población.
- Número de generaciones máximas de ejecución.
- Probabilidad de recombinación.
- Probabilidad de Mutación en clústers.

- Probabilidad de Mutación en rutas.

Esto sin incluir los dos modelos propuestos y los 20 set de datos de entradas que se pueden utilizar (mencionados en 6.1.2). En una primera instancia, las combinaciones de parámetros que se podían evaluar comprenden los siguientes valores:

- 2 modelos.
- 20 datos de entrada.
- Tamaño de la población: 50, 100, 200.
- Número de generaciones máximas de ejecución: 5000, 10000, 15000, 20000.
- Probabilidad de recombinación: 0.35, 0.45, 0.55, 0.75.
- Probabilidad de Mutación en clústers: 0.0025, 0.005, 0,0075.
- Probabilidad de Mutación en rutas: 0.025, 0.05, 0.075.
- Tamaño del torneo: 2, 3.

Esta disposición nos da una gama de 34560 pruebas distintas, sin considerar la posibilidad de realizar repeticiones para cada una de ellas. En términos de tiempo dentro del contexto de la investigación, se trataba de un plan de calibración inviable, por lo que se optó por la simplificación de algunos parámetros, que son explicados a continuación:

- Los datos de entrada para realizar la calibración de redujeron considerablemente en número. En vez de considerarlos todos, se optó por seleccionar 3 sets que fueran representativos del conjunto y que se pudieran clasificar en tres tipos: pequeño, mediano y grande, esto en relación a la cantidad de clientes que las instancias consideraban. De este modo, los sets considerados para la calibración son: pr02, pr03 y pr16 respectivamente.

- Sólo se consideró el valor de 100 individuos para el tamaño de la población, en base a ejecuciones intermedias que se realizaron en las etapas de desarrollo del algoritmo, donde

este valor mostraba resultados mucho mejores que los otros dos valores, además de representar un promedio en el rango en términos de tiempo de ejecución/calidad obtenida.

• Para la etapa de calibración se optó por utilizar el valor de 10000 generaciones máximas. Lo normal es que a un número mayor de generaciones, son mejores los resultados obtenidos, pero esto va asociado a un costo en tiempo de ejecución. Por esto es que se consideró este valor intermedio. Para las pruebas finales, se optó por utilizar el valor de 15000, considerando que serían más específicas las ejecuciones realizadas.

• Sólo se utilizó una cantidad de 2 individuos en el tamaño del torneo, considerando que es el valor más común para este tipo de selección y analizado como método en algunos estudios de la literatura [18].

De este modo, los valores finales a utilizados en las pruebas de calibración son los siguientes:

• 2 modelos.

• 3 datos de entrada.

• Tamaño de la población: 100.

• Número de generaciones máximas de ejecución: 10000.

• Probabilidad de recombinación: 0.35, 0.45, 0.55, 0.75.

• Probabilidad de Mutación en clústers: 0.0025, 0.005, 0,0075.

• Probabilidad de Mutación en rutas: 0.025, 0.05, 0.075.

• Tamaño del torneo: 2.

Esto entrega una posibilidad de 216 posibles pruebas y con 3 repeticiones cada una, se tiene un total de 648 ejecuciones para la etapa de calibración.

Por otro lado, los parámetros de la función DARP para realizar la evaluación, que se utilizaron en esta etapa, son los siguientes:

- Factor del tiempo de duración de las rutas: 8.
- Factor del tiempo de slacks: 1.
- Factor de la cantidad de vehículos: 0.
- Factor del exceso en el tiempo de transporte de clientes: 2.
- Factor del tiempo de espera de clientes: 0.
- Factor del tiempo de viaje de clientes: 4.

Se dio una mayor importancia al factor de la duración de las rutas, y por ende, del aspecto del sistema de transporte sobre la calidad del cliente, porque con las restricciones impuestas en el presente trabajo en torno a la estructura de las rutas, hay aspectos de calidad de servicio que se asumen como solventadas, por ejemplo lo que es la violación de las ventanas de tiempo, y la espera de los clientes en vehículos detenidos. El factor de la cantidad de vehículos fue anulado considerando que los datos de entrada utilizados consideran un máximo de vehículos que acota de manera adecuada su número en relación a la cantidad de clientes de las instancias específicas. En razón de esto último, no se considero el factor de tiempo de espera de los clientes, para darle prioridad a los otros dos factores vinculados a la calidad de servicio.

Para la evaluación de los parámetros, se compararon de a pares y de forma individual los resultados en términos de calidad, basándose en los parámetros del Algoritmo Genético que fueron utilizados en ellos. Se generaron gráficos comparativos de estos resultados, tanto para pares de parámetros como de forma individual. Desde la Figura 6.3 a la Figura 6.5 se muestran los resultados para el modelo LLGA y 3 conjuntos de datos utilizados, así mismo desde la Figura 6.6 a la Figura 6.8 para el modelo basado en rutas, correspondientes a las comparaciones individuales.

Para el set de datos pequeño y el modelo LLGA, los ratios de recombinación menores dieron mejores resultados, prevaleciendo el de 0.45 levemente sobre el 0.35. Para la mayor parte de los casos, el ratio de mutación en clusters de 0.005 fue más efectivo, así mismo con el valor de 0.075 para el ratio de mutación en rutas. En el set de datos mediano, resulto

efectivo un ratio de recombinación más bajo (0.35) y un ratio de mutación en clusters mayor (0.0075), siendo el mismo ratio de mutación en rutas del caso anterior. Por último, estos valores cambian totalmente para el caso de los set de datos grandes, donde un ratio alto de recombinación (0.75), y ratios bajos en ambos tipos de mutación (0.0025 y 0.05 respectivamente) arrojan mejores resultados.

Las mediciones obtenidas para el modelo basado en rutas difieren del LLGA en algunos valores. En el caso del set de datos pequeño, un ratio de mutación en clusters mayor resulta más efectivo (de 0.0075), en el set mediano un ratio de recombinación algo mayor y un ratio de mutación en rutas menor arrojan mejores resultados (0.45 y 0.05 respectivamente). Por último, los valores obtenidos para el set de datos mayor respecto al modelo LLGA, varían únicamente en la recombinación, donde es más efectivo un ratio menor para este parámetro (0.45).

A modo de síntesis, los mejores valores obtenidos en esta etapa de calibración para los parámetros variantes en los modelos, son los siguientes:

- Modelo LLGA
  - Set pequeño

Probabilidad de recombinación: 0.45. Probabilidad de mutación en clusters: 0.005. Probabilidad de mutación en rutas: 0.075.

  - Set mediano

Probabilidad de recombinación: 0.35. Probabilidad de mutación en clusters: 0.075. Probabilidad de mutación en rutas: 0.075.

  - Set grande

Probabilidad de recombinación: 0.75.

Probabilidad de mutación en clusters: 0.0025.

Probabilidad de mutación en rutas: 0.025.

- Modelo basado en rutas
  - Set pequeño

Probabilidad de recombinación: 0.45.

Probabilidad de mutación en clusters: 0.0075.

Probabilidad de mutación en rutas: 0.075.

  - Set mediano

Probabilidad de recombinación: 0.45.

Probabilidad de mutación en clusters: 0.075.

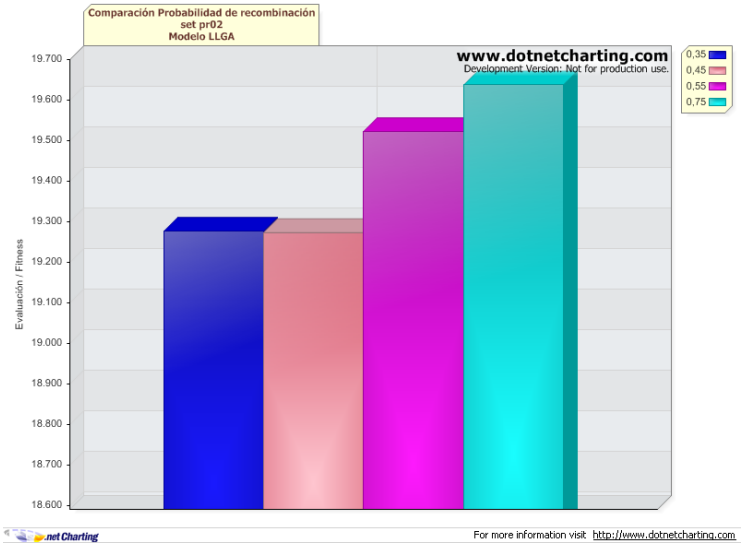
Probabilidad de mutación en rutas: 0.05.

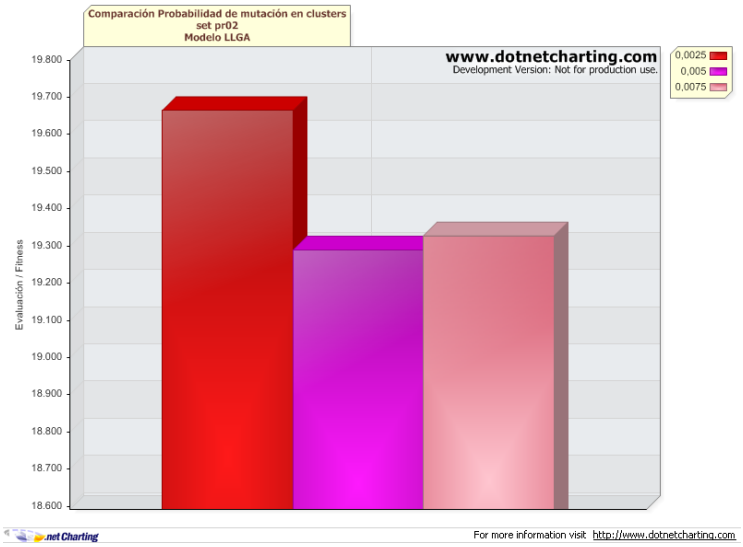
  - Set grande

Probabilidad de recombinación: 0.45.

Probabilidad de mutación en clusters: 0.0025.

Probabilidad de mutación en rutas: 0.025.


Comparación Probabilidad de recombinación
set pr02
Modelo LLGA
www.dotnetcharting.com
Development Version: Not for production use.
0,35
0,45
0,55
0,75
Evaluación / Fitness
19.700
19.600
19.500
19.400
19.300
19.200
19.100
19.000
18.900
18.800
18.700
18.600
.net Charting
For more information visit http://www.dotnetcharting.com


Comparación Probabilidad de mutación en clusters
set pr02
Modelo LLGA
www.dotnetcharting.com
Development Version: Not for production use.
0,0025
0,005
0,0075
Evaluación / Fitness
19.800
19.700
19.600
19.500
19.400
19.300
19.200
19.100
19.000
18.900
18.800
18.700
18.600
.net Charting
For more information visit http://www.dotnetcharting.com

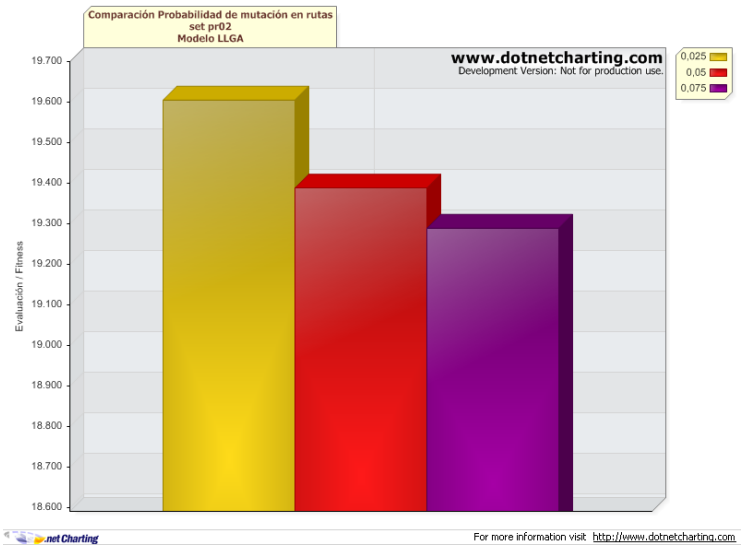
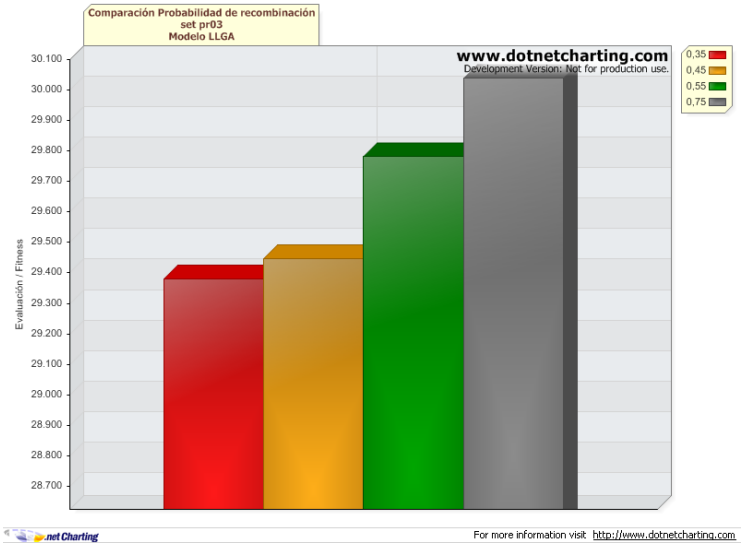
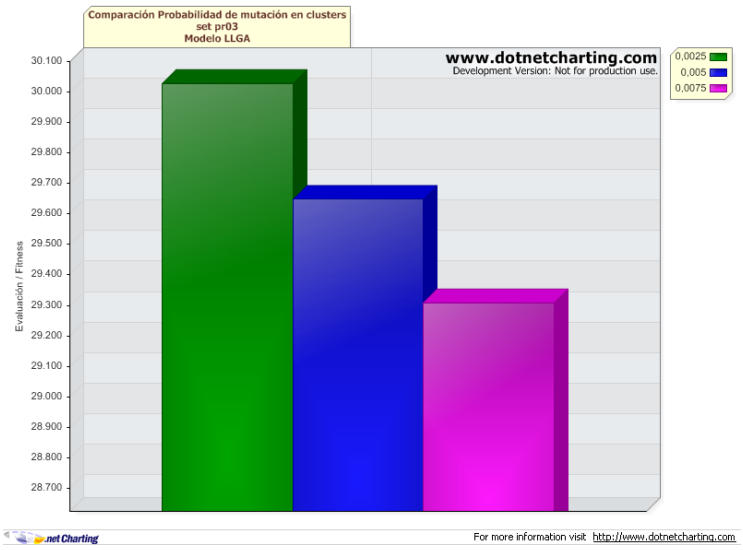


Figura 6.3 Gráficos comparativos Modelo LLGA, set de datos pr02.





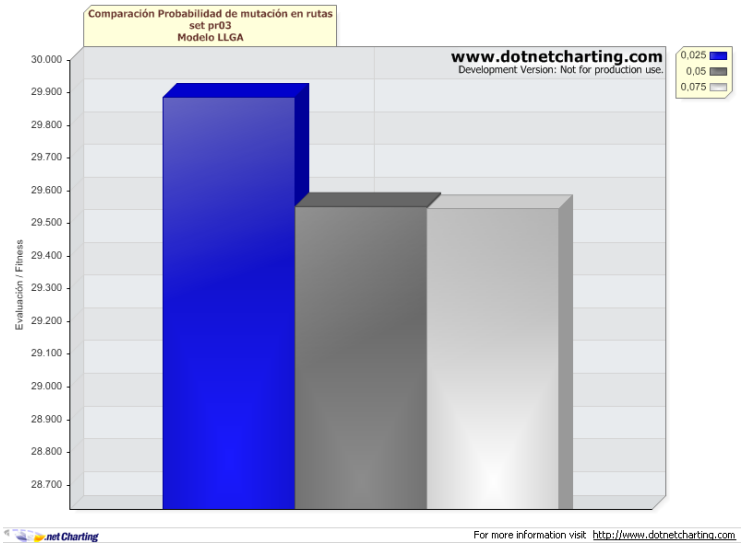
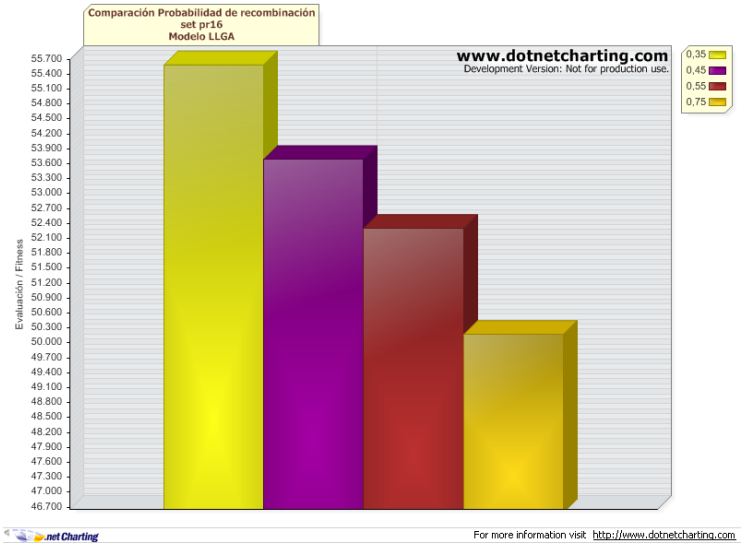
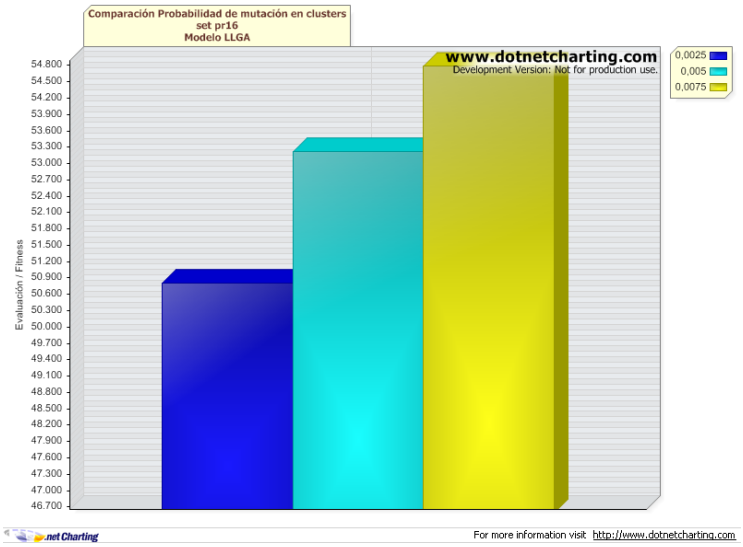


Figura 6.4 Gráficos comparativos Modelo LLGA, set de datos pr03.


Comparación Probabilidad de recombinación
set pr16
Modelo LLGA
www.dotnetcharting.com
Development Version: Not for production use.
0,35
0,45
0,55
0,75
Evaluación / Fitness
55.700
55.400
55.100
54.800
54.500
54.200
53.900
53.600
53.300
53.000
52.700
52.400
52.100
51.800
51.500
51.200
50.900
50.600
50.300
50.000
49.700
49.400
49.100
48.800
48.500
48.200
47.900
47.600
47.300
47.000
46.700
.net Charting
For more information visit http://www.dotnetcharting.com


Comparación Probabilidad de mutación en clusters
set pr16
Modelo LLGA
www.dotnetcharting.com
Development Version: Not for production use.
0,0025
0,005
0,0075
Evaluación / Fitness
54.800
54.500
54.200
53.900
53.600
53.300
53.000
52.700
52.400
52.100
51.800
51.500
51.200
50.900
50.600
50.300
50.000
49.700
49.400
49.100
48.800
48.500
48.200
47.900
47.600
47.300
47.000
46.700
.net Charting
For more information visit http://www.dotnetcharting.com

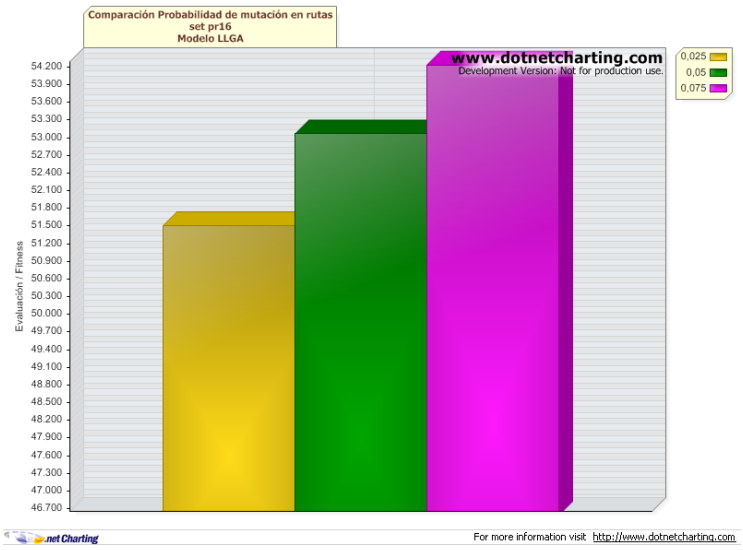
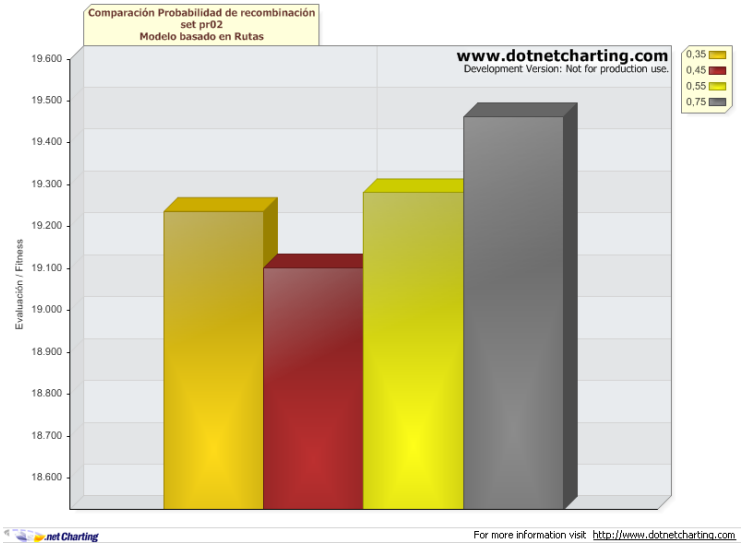
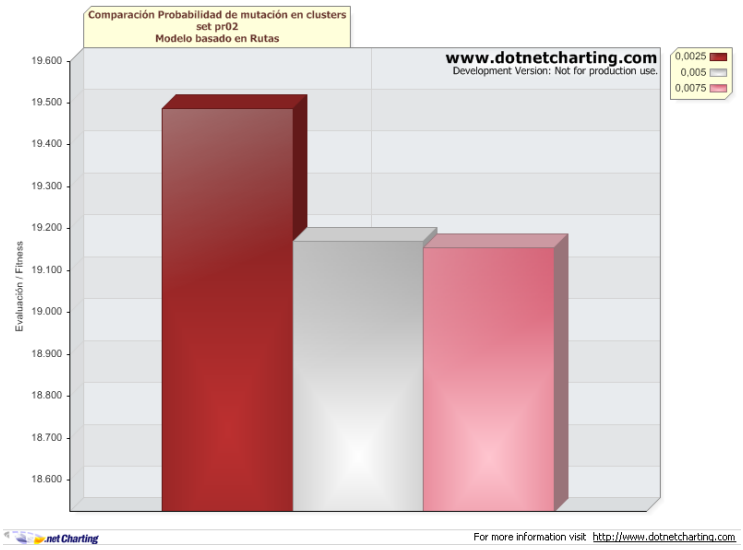


Figura 6.5 Gráficos comparativos Modelo LLGA, set de datos pr16.





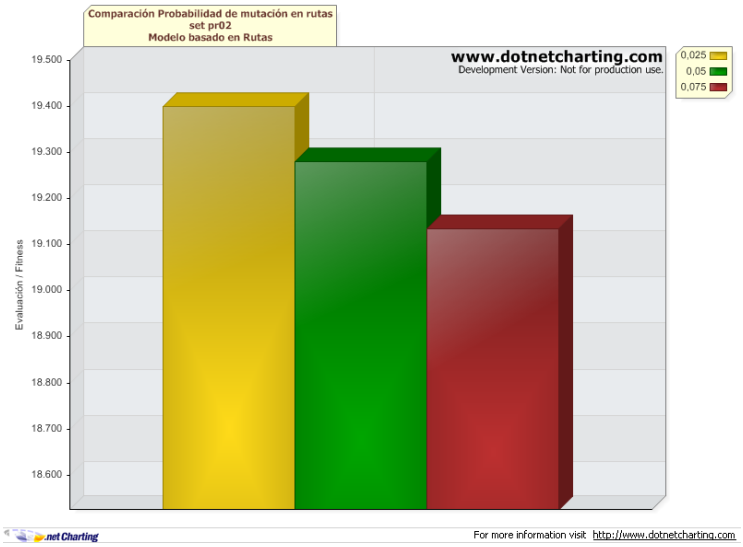
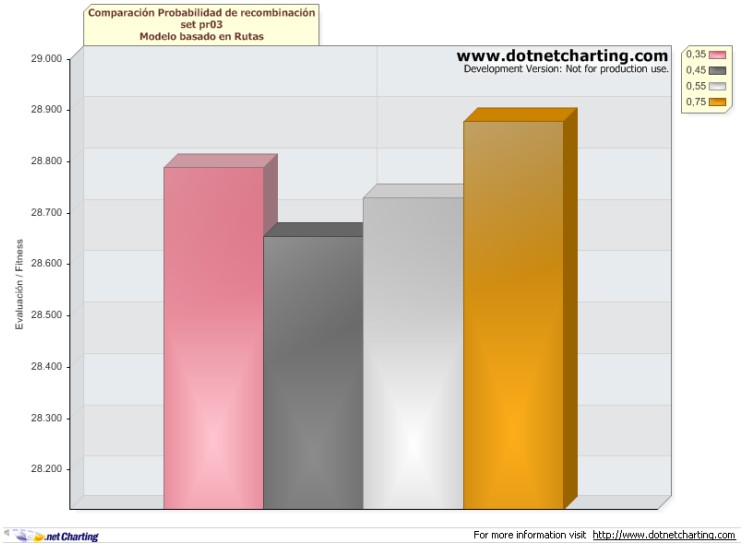
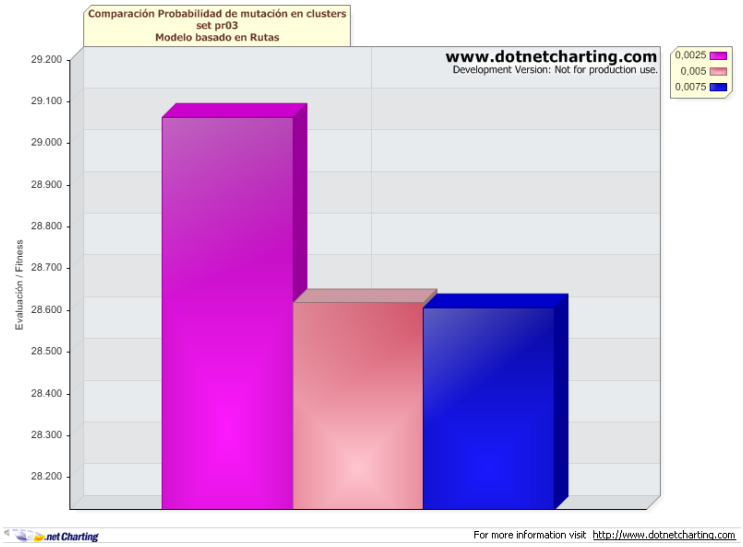


Figura 6.6 Gráficos comparativos Modelo basado en rutas, set de datos pr02.


Comparación Probabilidad de recombinación
set pr03
Modelo basado en Rutas
www.dotnetcharting.com
Development Version: Not for production use.
0,35
0,45
0,55
0,75
Evaluación / Fitness
29.000
28.900
28.800
28.700
28.600
28.500
28.400
28.300
28.200
.net Charting
For more information visit http://www.dotnetcharting.com


Comparación Probabilidad de mutación en clusters
set pr03
Modelo basado en Rutas
www.dotnetcharting.com
Development Version: Not for production use.
0,0025
0,005
0,0075
Evaluación / Fitness
29.200
29.100
29.000
28.900
28.800
28.700
28.600
28.500
28.400
28.300
28.200
.net Charting
For more information visit http://www.dotnetcharting.com

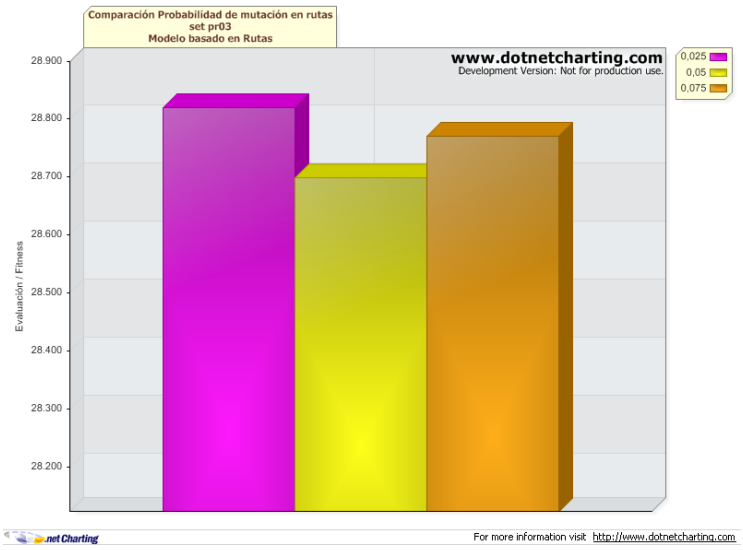
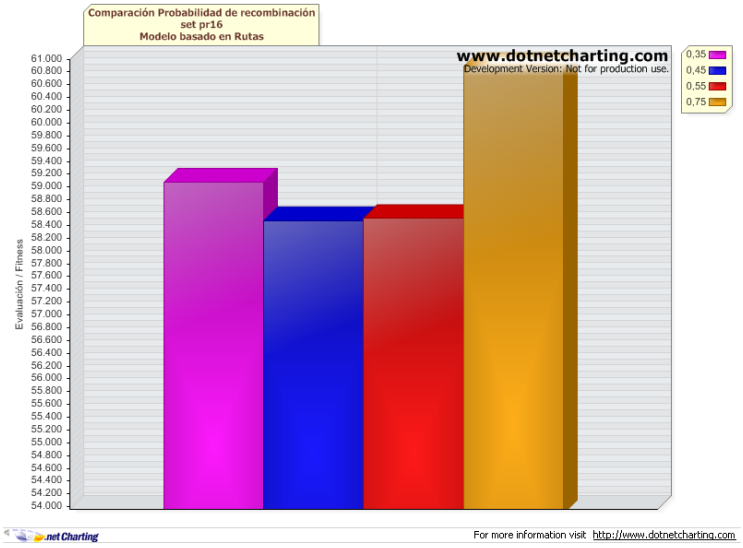
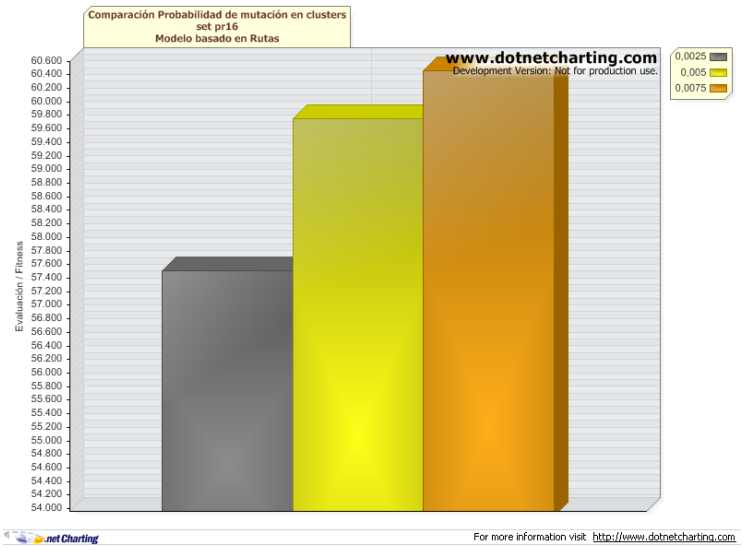


Figura 6.7 Gráficos comparativos Modelo basado en rutas, set de datos pr03.





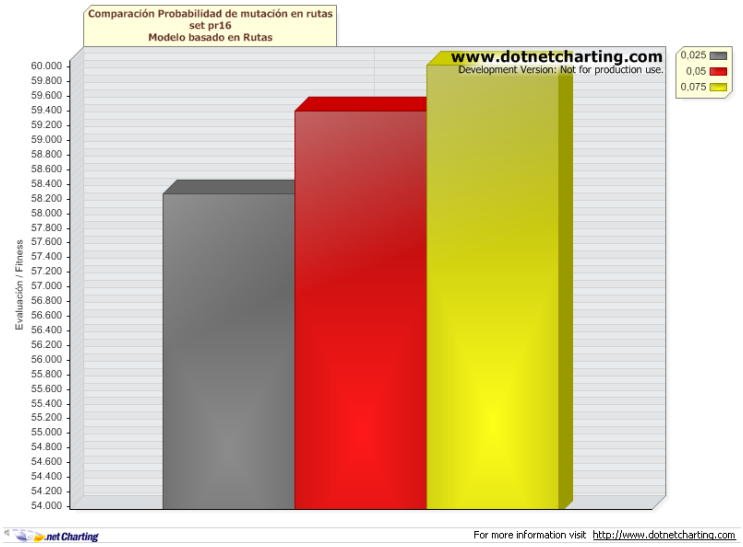


Figura 6.8 Gráficos comparativos Modelo basado en rutas, set de datos pr16.

6.4. Pruebas finales y resultados

Con los resultados obtenidos en la etapa anterior, fue posible realizar una serie de pruebas con los modelos para medir su comportamiento de forma más precisa, en términos de calidad de resultados y tiempo de ejecución. Estas pruebas se basaron, a diferencia de lo que fue la etapa de calibración, en la utilización de un número mayor de set de datos de entrada, un número mayor de repeticiones por prueba realizada y un número mayor de

generaciones por ejecución. En la Tabla 6.3 se puede observar la clasificación considerada para los conjuntos de datos. A cada uno de ellos se aplicaron los parámetros de ejecución obtenidos en 6.3, para cada ejecución se realizaron 20 repeticiones y se utilizaron 15000 generaciones. En total se realizaron 200 ejecuciones por modelo con las características mencionadas.

Tabla 6.3 Clasificación de instancias para la realización de las pruebas finales:

Tamaño conjunto	**Sets del Conjunto**
Pequeño	pr01 pr02 pr11 pr12 pr17
Mediano	pr03 pr05 pr15 pr19
Grande	pr16

Los resultados se compararon con el estudio de Bergvinsdottir, Larsen and Jorgensen [4] (Algoritmo genético *cluster-first route-second*) y con el de Cordeau y Laporte [25] (Tabu Search). En la Tabla 6.4 y Tabla 6.5 se muestran los resultados obtenidos por los modelos desarrollados en el presente trabajo y en la Tabla 6.6 y Tabla 6.7 se muestran los resultados de los estudios señalados anteriormente.

Como todos los modelos comparados no necesariamente se basan en mediciones de calidad iguales, la comparación se basó en resultados directos en unidad de tiempo de dos factores críticos del problema: por un lado la duración de la ruta que se vincula a la optimización de recursos del sistema de transporte, y por otro lado al tiempo de viaje de los clientes, vinculado al factor de calidad de servicio ofrecido por el sistema.

Desde la Figura 6.9 a la Figura 6.11 se muestran ejemplos de gráficos de convergencia de la ejecución con los mejores resultados del modelo LLGA, acorde a la clasificación de set de datos señalada anteriormente. Del mismo modo, desde la Figura 6.12 a la Figura 6.14 se muestran ejemplos de gráficos de convergencia de la ejecución con los mejores resultados del modelo basado en Rutas.

Es importante realizar ciertas consideraciones respecto a los resultados que se comparan y la naturaleza de los estudios involucrados:

Tabla 6.4 Resultados obtenidos por el modelo LLGA:

Modelo LLGA					
Set	**Route duration**		**Ride time**		**CPU Time (Min.)**
	Best	**Avg.**	**Best**	**Avg.**	
pr01	955,25	1012,32	524,59	546,56	1,36
pr02	1839,06	1836,05	838,41	889,05	4,08
pr03	2787,18	2810,59	1597,95	1634,1	7,96
pr05	4068,05	4118,03	2935,48	3007,97	18,43
pr11	902,18	908,09	449,91	454,4	1,58
pr12	1503,34	1503,71	744,93	766,21	4,49
pr15	4057,08	4118,34	3152,67	3160,79	22,09

pr16	4658,35	4665,77	2348,48	2377,45	17,48
pr17	1223,68	1227,46	612,4	612,4	3,13
pr19	3427,06	3441,71	2515,53	2597,1	25,42

Tabla 6.5 Resultados obtenidos por el modelo basado en Rutas:

Modelo basado en Rutas					
Set	**Route duration**		**Ride time**		**CPU Time (Min.)**
	Best	**Avg.**	**Best**	**Avg.**	
pr01	917,25	968,42	542,1	535,22	2,38
pr02	1851,87	1838,16	812,59	886,09	8,49
pr03	2769,06	2793,16	1439,36	1460,87	14,23
pr05	4156,88	4159,89	2950,89	3044,24	92,22
pr11	902,16	906,7	449,91	459,8	3,52
pr12	1514,97	1505,52	773,76	817,29	14,65
pr15	4117,89	4163,93	3112,07	3280,53	83,99
pr16	4670,28	4771,99	3320,08	3426,6	61,34
pr17	1225,21	1226,78	612,4	612,4	7,46
pr19	3473,02	3469,64	2376,49	2665,6	156,55

• Tanto en las investigación de Bergvinsdottir, Larsen and Jorgensen, como en la de Cordeau y Laporte, se relajaron restricciones importantes que en el presente estudio se consideraron en su pleno cumplimiento para la factibilidad de las soluciones. Entre estas restricciones, destacan principalmente la rigidez de las ventanas y la imposibilidad de que los vehículos se detengan en medio de la ejecución de *slacks*.

• Considerando el escenario más restrictivo en la presente investigación, no fue posible generar soluciones factibles para algunos de los conjuntos de datos que están señalados en 6.1.2, en especial aquellos casos en donde la relación de la cantidad de clientes versus la cantidad máxima de vehículos que se podían utilizar, era muy grande. Por ello las comparaciones sólo se enfocaron en los datos mostrados en las Tablas 6.4 a la 6.7.

• En la presente investigación, a diferencia de los otros estudios comparados, la restricción vinculada a la duración máxima de la ruta no fue considerada.

Tabla 6.6 Resultados obtenidos por Bergvinsdottir, Larsen y Jorgensen (GA):

Bergvinsdottir, Larsen and Jorgensen					
Set	**Route duration**		**Ride time**		**CPU Time (Min.)**
	Best	**Avg.**	**Best**	**Avg.**	
pr01	1039	1041	310	447	5,57
pr02	1994	1969	1330	1367	11,43
pr03	2781	2779	2894	3081	21,58
pr05	4274	4250	4837	5099	58,23
pr11	928	907	549	630	5,46
pr12	1710	1719	1300	1214	11,72
pr15	4336	4296	4720	4615	58,93

pr16	5227	5309	6397	6134	81,23
pr17	1316	1299	784	990	8,29
pr19	3676	3679	5358	5362	44,66

Tabla 6.7 Resultados obtenidos por Cordeau y Laporte (Tabu Search):

Cordeau and Laporte			
Set	**Route duration**	**Ride time**	**CPU Time (Min.)**
pr01	881	1095	1,9
pr02	1985	1977	8,06
pr03	2579	3587	17,18
pr05	3870	6154	46,24
pr11	965	1042	1,93
pr12	1565	2393	8,29
pr15	3596	6105	54,33
pr16	4072	7347	73,7
pr17	1097	1762	4,23
pr19	3249	5581	51,28

En términos de calidad, los modelos desarrollados en esta investigación mostraron resultados similares a los otros dos comparados en lo que se refiere a la duración de las rutas de los vehículos. Ninguno de los dos destaca sobre el otro en este sentido, para algunos casos el modelo LLGA muestra resultados mejores que el de rutas y viceversa, sin embargo se trata de diferencias menores en los tiempos. En este factor, los resultados de Cordeau y Laporte muestran las duraciones más bajas. Sin embargo, en lo que se refiere al tiempo de viaje de los clientes, tanto el modelo LLGA como el de rutas mostraron mejores resultados que los otros dos estudios en la mayoría de los casos. Esto se debe en gran parte a como se manejaron las restricciones. Como el escenario planteado para los modelos desarrollados en este trabajo es restrictivo, la tendencia a mejorar los tiempos de viaje de los clientes es mayor, dado que las ventanas de tiempo son respetadas en su totalidad y que los vehículos no pueden detenerse con pasajeros a bordo. En general, los vehículos realizan viajes más directos para recoger y entregar a los clientes. Esta tendencia se destacó especialmente en los casos donde la cantidad de clientes es grande.

Figura 6.9 Mejor ejecución del modelo LLGA con el set de datos pr02.

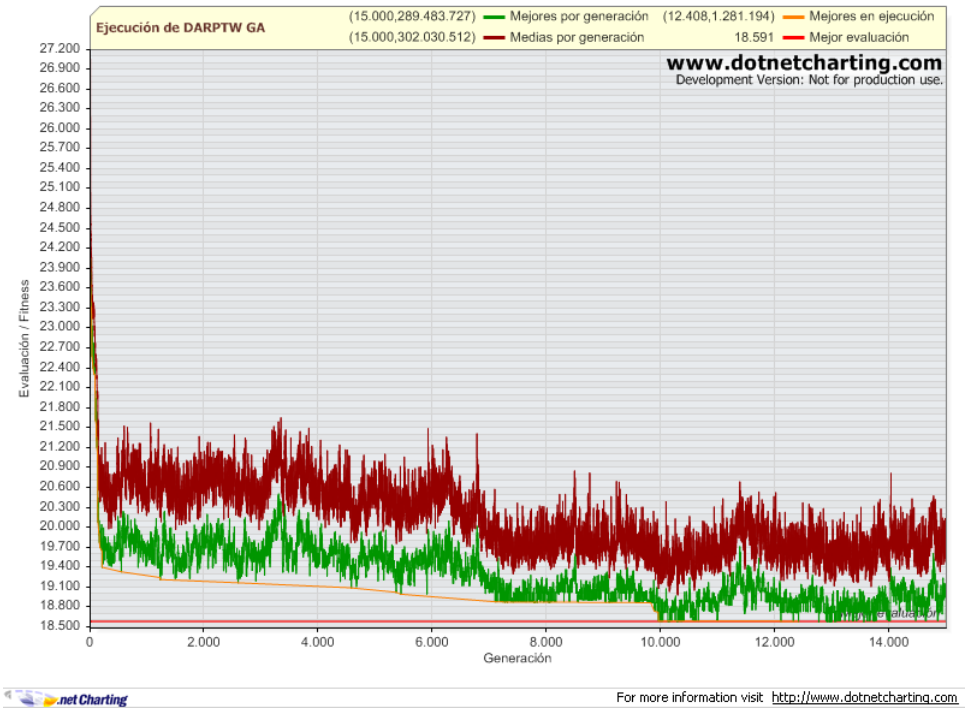


Figura 6.10 Mejor ejecución del modelo LLGA con el set de datos pr03.

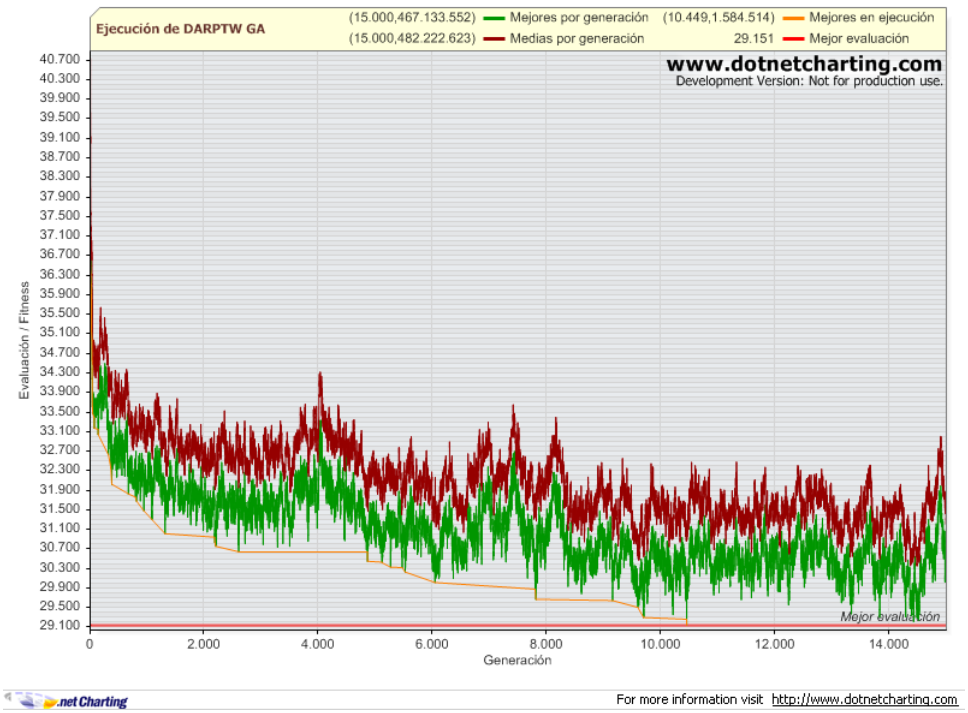


Figura 6.11 Mejor ejecución del modelo LLGA con el set de datos pr16.

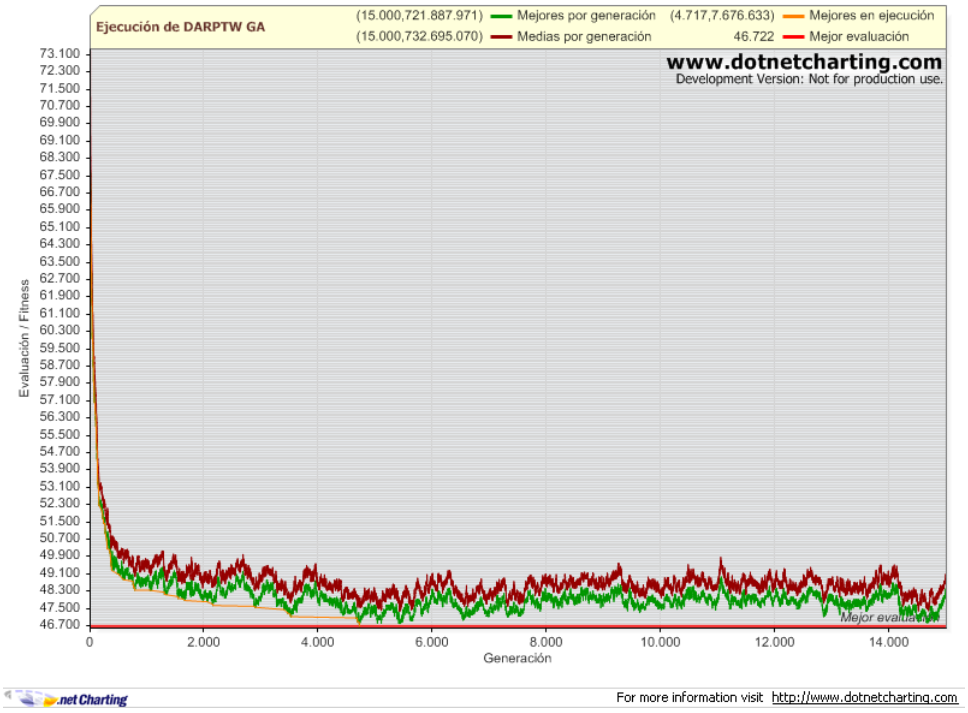


Figura 6.12 Mejor ejecución del Modelo basado en Rutas con el set de datos pr02.

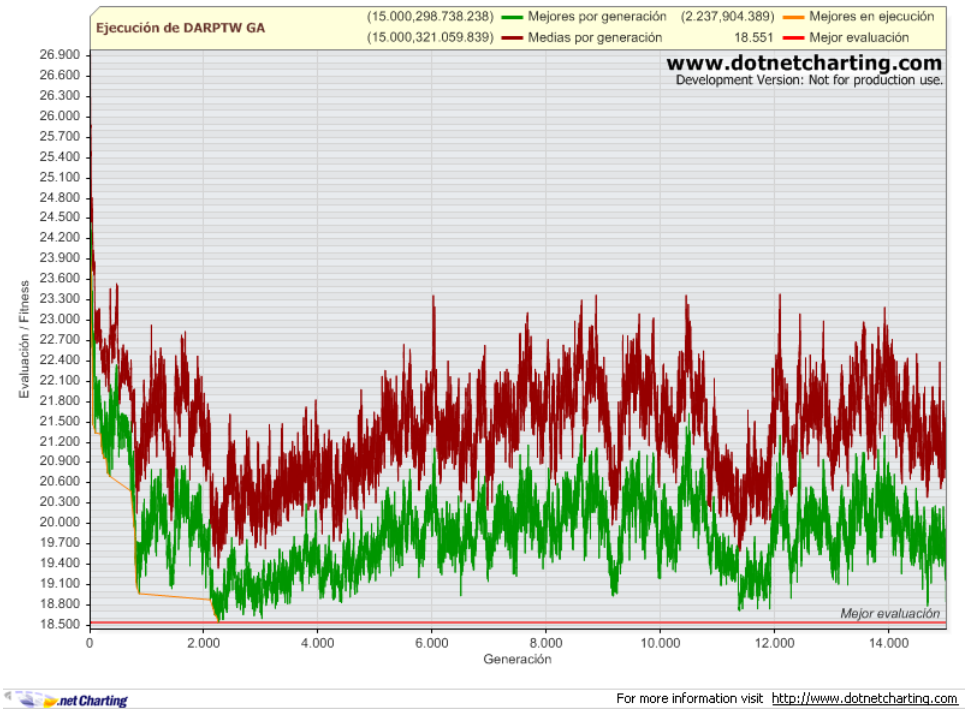


Figura 6.13 Mejor ejecución del Modelo basado en Rutas con el set de datos pr03.

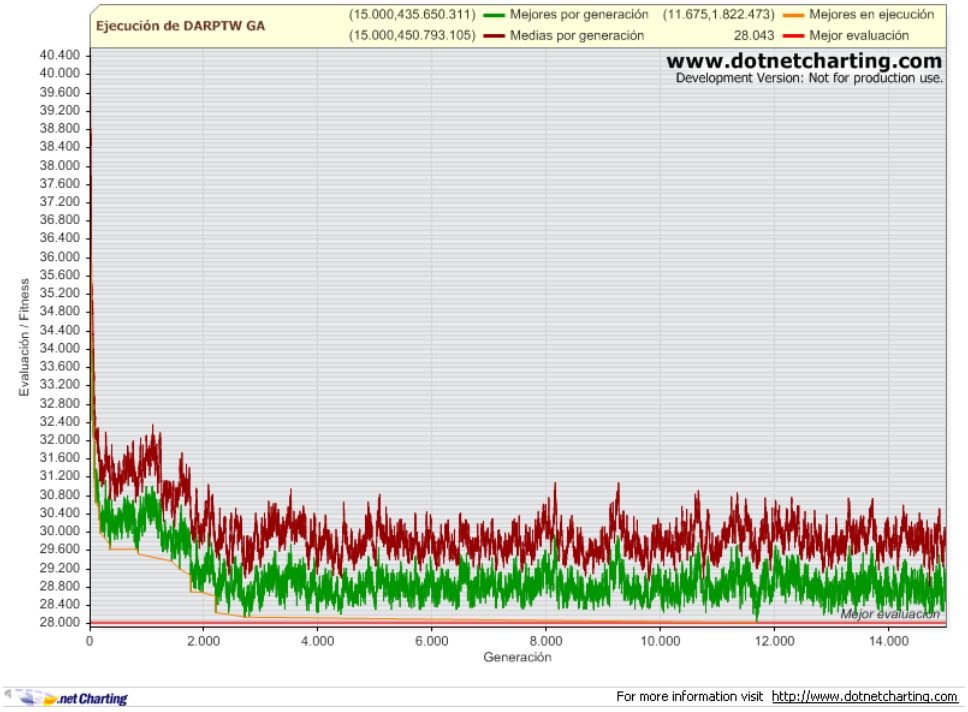
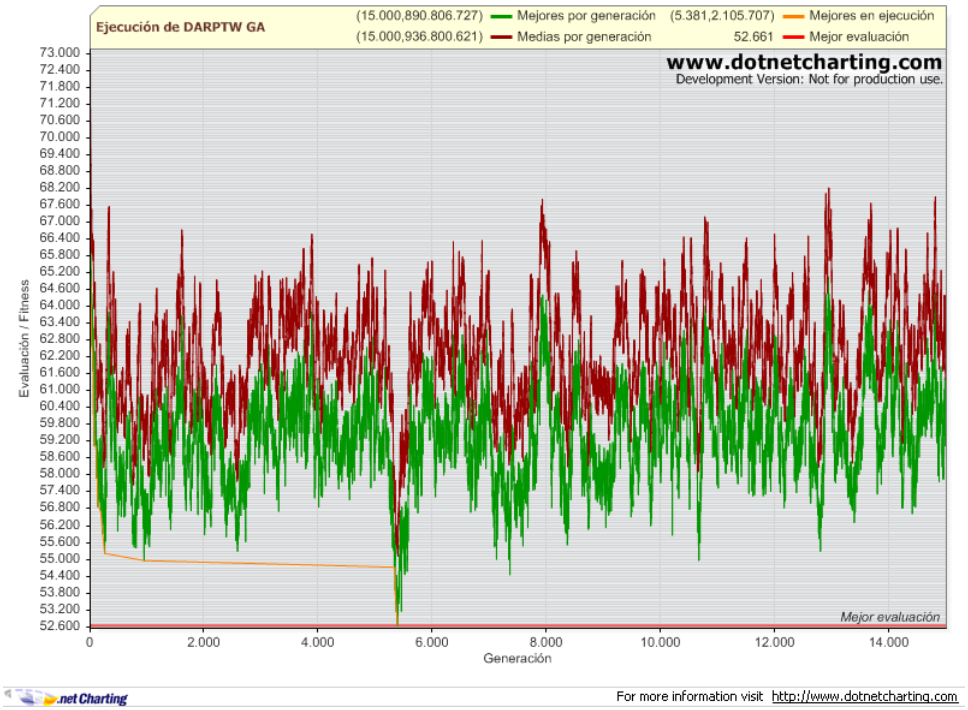


Figura 6.14 Mejor ejecución del Modelo basado en Rutas con el set de datos pr16.



Por otro lado, en lo que se refiere a los tiempos de ejecución, el modelo LLGA mostró un rendimiento interesante respecto a los otros. Los tiempos en todos los casos eran mejores en este modelo, incluyendo la comparación con el de rutas. Sin duda la aplicación de los esquemas de factibilidad preliminar y el procesamiento previo de los datos de entrada, ayudó a que este factor fuera optimizado. La ventaja que tiene el modelo LLGA sobre el de rutas, es que su operador de recombinación trabaja sólo en una etapa, en lo que se refiere a la modificación de la estructura del cromosoma, en cambio en el modelo de rutas se realizan tres etapas que tienen un costo de tiempo importante reflejado en los resultados, pese a que las modificaciones se realicen directamente en la estructura de las rutas, que en el caso del LLGA corresponderían a la capa interna. Cabe destacar que las pruebas para los modelos de esta investigación fueron realizadas en un computador con un procesador Intel Pentium 4 de 2.66 gHz, mientras que las de Cordeau y Laporte fueron realizadas con un Intel Pentium 4 de 2.0 gHz y las de Bergvinsdottir con un Intel Celeron de 2.0 gHz.

7. Conclusiones y trabajo futuro

Los modelos presentados en este trabajo han mostrado resultados interesantes según las comparaciones realizadas. Lo más destacable fue la reducción de tiempo obtenida por el modelo LLGA, sin duda uno de los factores más débiles de los GA. Sin embargo, no se debe perder de vista el hecho que se está trabajando con modelos de características distintas. Lamentablemente son pocos los estudios con los cuales se pueden realizar comparaciones, por ello las diferencias que presentan deben ser abordadas adecuadamente al momento de analizarlas. El presente trabajo permitirá a otros investigadores el realizar comparaciones con modelos altamente restrictivos, a diferencia de la mayoría de los presentados en la literatura, que tienen una mayor cantidad de restricciones relajadas.

Son dos los elementos importantes que deben destacarse en los GAs desarrollados en este trabajo. Por un lado, la utilización de esquemas de pre-factibilidad ha demostrado ser una alternativa interesante de apoyo al funcionamiento del GA. Aunque depende estrictamente del problema, los GA destacan por funcionar adecuadamente cuando el espacio de búsqueda es amplio y complejo, como en el caso de DARPTW. Sin duda en estos problemas es donde las restricciones son el detalle más complicado de abordar, por ende el trabajar dichas restricciones ayudará a mejorar el rendimiento del GA, centrándolo más que nada en la búsqueda en sí, y rebajando su carga de trabajo en lo que se refiere a validaciones de factibilidad. Por otro lado, el presente trabajo muestra la posibilidad de utilizar un modelo de GA pensado como un solucionador genérico, y con una adaptación adecuada puede ser aplicado eficientemente en un problema específico: es el caso de la utilización del modelo LLGA de Harik. Dicho modelo aun puede ser mejorado en este trabajo, tomando en cuenta que sólo se utilizó la versión básica del mismo. Como se detalló en 3.3.3, hay una serie de modificaciones al modelo original que pueden aportar a mejorar la eficiencia con que el algoritmo maneja el tema del acoplamiento.

Además de lo anterior, son varios los elementos con los que se puede seguir experimentando en el tema de GAs aplicados en DARPTW. Entre ellos destacan:

- Considerar el factor de duración de las rutas a las restricciones pertinentes y medir el comportamiento del modelo bajo este escenario, que naturalmente debería tener mayores dificultades para encontrar resultados factibles. El reducir la duración de las rutas implicará que se deban distribuir de manera más uniforme los clientes en los vehículos. Aunque la cantidad de los mismos no es un factor crítico considerando que los datos de entrada especifican un número máximo de ellos, el considerar la duración de la ruta exigirá al GA buscar soluciones donde la utilización de los vehículos sea más eficiente en un contexto factible, esto puede implicar la necesidad de buscar mecanismos en donde esta factibilidad se pueda asegurar de antemano.

- Para realizar una comparación más cercana a otros estudios de la literatura, se puede experimentar con la relajación de restricciones del problema. Una alternativa es permitir que un vehículo tenga pasajeros en tiempos de *slack*. Esto implica el considerar una gran cantidad de casos actualmente que son infactibles, lo que puede darle más libertad al GA en términos de la búsqueda que realiza, sin embargo habrá que considerar un nuevo factor de calidad de servicio en la función objetivo, tomando en cuenta que la espera en un vehículo detenido afecta negativamente el viaje del pasajero. Otra alternativa es permitir una violación a cierto nivel de las ventanas de tiempo. Esto implica el considerar una diferencia en que los rangos de las ventanas se puedan ampliar. Sin embargo, se debe tener siempre clara cuál era la ventana original, por que cuando un vehículo recae en los rangos extendidos de las ventanas, implica que no está respetando el escenario original, por ende ello también requiere de de una penalización en la función objetivo como costo adicional, tanto en la calidad de servicio del cliente como en el funcionamiento del sistema de transporte en sí. Este escenario nos puede conducir a soluciones menos efectivas, pero puede considerarse también más realista.

- También sería interesante el desarrollo de un operador de mutación que mezcle las ventajas de los dos tipos usados acá. Si bien tener un par de operadores, tanto a nivel de clusters como a nivel de rutas, permite mayor personalización en el comportamiento del

GA, el costo de por medio es la dificultad de la evaluación del mismo, especialmente en la etapa de calibración, donde se requiere medir un parámetro adicional. De los dos operadores de mutación presentados en este trabajo, el más cercano a esta visión es el de clusters, por que pese a que se enfoca en modificar la asignación de pasajeros a los vehículos, también requiere de una modificación factible a las rutas para que esto se cumpla.

- Se pueden mejorar los mecanismos de modificación de rutas con el fin de potenciar la eficiencia de los modelos desarrollados, en base a como se manejan los bloques de construcción. Actualmente las inserciones de clientes, cuando se consideran como grupo, se realizan individual y sucesivamente para cada uno de ellos, es decir, se inserta un cliente y se realiza la validación, después se inserta el siguiente y se valida, así sucesivamente hasta completar la lista de clientes. En el transcurso de la implementación, destacó el hecho que no es lo mismo insertar los clientes de este modo que considerar una inserción de varios clientes a la vez. Lo mismo pasa con la eliminación e intercambio. De este modo, se genera la posibilidad de crear nuevos mecanismos que permitan evaluar la inserción de un grupo de clientes en "un tiempo", no de forma sucesiva. Si se consideran grupos de clientes pequeños en este contexto, se puede generar un mayor acercamiento al concepto señalado en **¡Error! No se encuentra el origen de la referencia.**, que destaca la importancia de que el GA propague con el pasar de las generaciones, bloques pequeños y de buena calidad.

Por último, dependiendo de la evolución en el tiempo de los datos de entrada, es posible considerar un escenario multi-depot, es decir, con más de un depósito para los vehículos. Cuando esto se suma a la posibilidad de tener un largo de ruta acotado, el escenario se complica en términos de la factibilidad de soluciones, pero al igual que el asunto de las ventanas de tiempo, se acerca a escenarios más realistas.

8. Referencias

[1] Cubillos, C. et al: On user requirements and operator purposes in Dial-a-Ride services. Proceedings of the 9th Meeting of the EURO Working Group on Transportation, pp. 677-684, 2002.

[2] Quadrifoglio, L.; Dessouky, M.; Ordóñez, F.: A Simulation Study of Demand Responsive Transit System Design. Transportation Research, Part A: Policy and Practice, Vol. 42, pp. 718–737, 2008.

[3] Bergvinsdottir, K.: The Genetic Algorithm for solving the Dial-a-Ride Problem. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004.

[4] Savelsbergh, M.W.P.: The General Pickup and Delivery Problem. Transportation Science, Vol. 29, pp 17-29, 1995.

[5] Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to biology, control and artificial intelligence. University of Michigan Press, 1975.

[6] Goldberg, D.E.: Messy genetic algorithms: Motivation, analysis, and first results. Complex Systems, Vol. 3, pp. 493-530, 1989.

[7] Whitley, L. D.: Fundamental Principles of Deception in Genetic Search. Foundations of genetic algorithms, Rawlins, G. J. E., pp. 221-241, 1991.

[8] Harik, G. R.: Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. Ph.D. Thesis, University of Michigan, Ann Arbor, 1997.

[9] Harik, G.; Goldberg, D. E.: Learning linkage. Foundations of Genetic Algorithms 4, pp. 247-262, 1997.

[10] Goldberg, D.E.; Chen, Y.: Convergence time for the linkage learning genetic algorithm. Evolutionary Computation, Vol. 13, pp. 279-302, 2005.

[11] Thangiah, S.: Vehicle Routing with Time Windows using Genetic Algorithms. Application Handbook of Genetic Algorithms: New Frontiers, Volume II, Lance Chambers, CRC Press, pp. 253-277, 1995.

[12] Zhu, K. Q.: A New Genetic Algorithm for VRPTW. International Conference on Artificial Intelligence, Las Vegas, USA, 2000.

[13] Jih, W.; Kao, C.; Hsu, J.: Using Family Competition Genetic Algorithm in Pickup and Delivery Problem with Time Window Constraints. Proceedings of the 2002 IEEE International Symposium on Intelligent Control, pp. 496-501, 2002.

[14] Bergvinsdottir, K. B.; Larsen, J.; Jørgensen, R. M.: Solving the Dial-a-Ride Problem using Genetic algorithms. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004.

[15] Cubillos, C.; Rodriguez, N.; Crawford, B.: A Study on Genetic Algorithms for the DARP Problem. International Work-Conference on the Interplay Between Natural and Artificial Computation, pp. 498-507, 2007.

[16] Jørgensen, R. M.: Dial-a-Ride. Doctor's thesis, Technical University of Denmark, 2002.

[17] Whitley, D.: A genetic algorithm tutorial. Statistics and Computing, Vol. 4, pp. 65-85, 1994.

[18] Mitchell, M.: An Introduction to Genetic Algorithms. University of Michigan Press, 1996.

[19] Munetomo, M.; Goldberg, D.: Linkage identification by non-monotonicity detection for overlapping functions. Evolutionary Computation, Vol. 7, pp. 377-398, 1999.

[20] Lobo, F.; Deb, K.; Goldberg, D.; Harik, G.; Wang, L.: Compressed introns in a linkage learning genetic algorithms. Genetic Programming: Proceedings of the Third Annual Conference, pp. 551-558, 1998.

[21] Harik, G. R.; Lobo, F.G.; Goldberg, D.E.: The compact genetic algorithm. IEEE Conference on Evolutionary Computation, Vol. 3, pp. 523-528, 1999.

[22] Sastry, K.; Goldberg, D.: On Extended Compact Genetic Algorithm. Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference, Darrell Whitley, pp. 352-359, 2000.

[23] Cubillos, C.: MADARP: Multi-Agent Framework for Transportation Systems. Tesis para optar al grado de Dottore di Ricerca in Ingenneria Informatica e dei Sistemi (ciclo XVII), Politecnico di Torino, 2005.

[24] Cordeau, J.: A Branch-and-Cut Algorithm for the Dial-a-Ride Problem. Technical Report CRT-2004-23, Centre for Research on Transportation, Montreal, 2004.

[25] Cordeau, J.; Laporte, G.: A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological, Vol. 37, pp. 579-594, 2003.