“SOLUCIÓN DEL PROBLEMA DE RUTEO DE VEHÍCULOS CON VENTANAS DE TIEMPO (VRPTW) MEDIANTE MÉTODOS HEURÍSTICOS”

ADRIANA LOZADA DÍAZ

RICARDO ANDRÉS CADENA GONZÁLEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
BUCARAMANGA
2012

“SOLUCIÓN DEL PROBLEMA DE RUTEO DE VEHÍCULOS CON VENTANAS DE TIEMPO (VRPTW) MEDIANTE MÉTODOS HEURÍSTICOS”

ADRIANA LOZADA DÍAZ

RICARDO ANDRÉS CADENA GONZÁLEZ

Trabajo de Grado como requisito para optar al título de Ingeniero Industrial

Director
HENRY LAMOS DÍAZ
Ph.D en Física-Matemática

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
BUCARAMANGA
2012

AGRADECIMIENTOS

Al Doctor Henry Lamos Díaz, por enseñarnos el maravilloso mundo de la investigación.

Al cuerpo docente del Grupo de Investigación OPALO y de la Escuela de Estudios Industriales y Empresariales de la UIS, por nuestra formación académica.

DEDICATORIA

A Dios por su Gracia.

A mis padres y mi hermana por su amor y abnegada ayuda.

A mi familia y amigos.

Ricardo Andrés cadena González

A Dios por ser mi motor e inspiración.

A mi madre por ser mi ejemplo, por sus sabios consejos.

A mi padre por infundirme el aliento para nunca fallar.

A mi tía Marlene por su cariño ilimitado, por ser mi fiel amiga.

A Dairo Álvarez por creer en mí.

Adriana Lozada Díaz

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

RESUMEN

TÍTULO:

"SOLUCIÓN DEL PROBLEMA DE RUTEO DE VEHÍCULOS CON VENTANAS DE TIEMPO (VRPTW) MEDIANTE MÉTODOS HEURÍSTICOS"[1]

AUTORES:

CADENA GONZÁLEZ, Ricardo Andrés
DÍAZ LOZADA, Adriana[2]

PALABRAS CLAVE:

VRP con ventanas de tiempo, heurísticas de construcción de rutas, Solomon.

DESCRIPCIÓN.

En el presente estudio, una de las variantes más importantes del Ruteo de Vehículos es tomada en consideración: El Problema de Ruteo de vehículos con ventanas de tiempo (VRPTW por sus siglas en inglés). El VRPTW aparece como un área importante de investigación en el área logística, puesto que describe de manera más adecuada la actividad de distribución física al tomar en consideración tanto restricciones espaciales como temporales. Básicamente, para esta variante del problema de ruteo, se busca servir la demanda de un conjunto de clientes esparcidos geográficamente, dentro de un intervalo de tiempo donde se permite el servicio.

El VRPTW es uno de los problemas de mayor complejidad en la optimización combinatoria. Para su solución, se han desarrollado diversas técnicas que brindan resultados factibles. Entre las herramientas de optimización estudiadas, se encuentran los métodos heurísticos, que exploran "buenas" soluciones en un tiempo razonable. Dentro del conjunto de algoritmos heurísticos están los algoritmos de construcción de rutas, que incorporan elementos a la solución parcial, a medida que el modelo es iterado.

Para la presente investigación, se estudiaron y sistematizaron cinco heurísticas de construcción de rutas que permiten solucionar las instancias más comunes del VRPTW de manera rápida y eficiente. La herramienta arroja mejores resultados en términos de costo de rutas con las heurísticas de inserción de Solomon (1987) para rutas de largo horizonte de programación.

[1] Proyecto de grado.
[2] Facultad de Ingenierías Físico mecánicas. Escuela de Estudios Industriales y Empresariales. Director: Ph.D. Henry Lamos Díaz.

ABSTRACT

TITLE:

"SOLUTION OF THE VEHICLE ROUTING PROBLEM WITH TIME WINDOWS THROUGH HEURISTIC METHODS"[3]

AUTHORS:

CADENA GONZÁLEZ, Ricardo Andrés
DÍAZ LOZADA, Adriana[4]

PALABRAS CLAVE:

VRP with time windows, route construction heuristics, Solomon.

DESCRIPTION

In this study, one of the most important variants of the Vehicle Routing is taken into consideration: The Vehicle Routing Problem with time windows (VRPTW). The VRPTW appears as an important area of research in logistics, as more adequately describes the physical distribution logistics activities by taking into account both spatial and temporal constraints. Basically, for this variant of the routing problem, we look for fulfilling the demand of a set of customers geographically dispersed within a time interval.

The VRPTW is a model of higher complexity on combinatorial optimization procedures. For its solution, various techniques have been developed. Among the studied optimization tools, we have heuristic methods, which explore "good" solutions in a reasonable computational time. Within the set of heuristic algorithms, we have route construction algorithms, which incorporate elements to the partial solution, as the model is iterated.

We studied and systematized five route construction heuristics that solve the most common VRPTW instances quickly and efficiently. The tool produces better results in terms of cost route to the Solomon insertion heuristic (1987) for long scheduling horizon routes.

[3] Proyecto de grado.
[4] Facultad de Ingenierías Físico mecánicas. Escuela de Estudios Industriales y Empresariales. Director: Ph.D. Henry Lamos Díaz.

INTRODUCCIÓN

En las últimas décadas se ha incrementado el uso de paquetes de optimización, basados en técnicas de investigación de operaciones y programación matemática, que facilitan el manejo y la gestión en las actividades de distribución de bienes y servicios.

El éxito de la utilización de las herramientas de optimización se debe principalmente al desarrollo de tecnología, tanto en hardware como en software, y a la integración acelerada de sistemas de información a los procesos productivos y comerciales. Sin embargo, los problemas relacionados con distribución y ruteo son de características complejas, puesto que son problemas de decisión de carácter combinatorio y exigen un reto computacional para su solución.

Entre las técnicas que se han estudiado y desarrollado para la solución del problema de ruteo y distribución se encuentran los métodos heurísticos, que proporcionan buenas soluciones a instancias seleccionadas en un tiempo de cómputo razonable.

El modelo VRPTW, Ruteo de Vehículos con Ventanas de Tiempo por sus siglas en inglés, es una variante del problema original VRP (Vehicle Routing Problem), donde clientes ubicados en una zona geográfica deben ser servidos en un tiempo determinado por una flota de camiones con una capacidad limitada, que parten de un depósito y hacen un recorrido específico, teniendo en cuenta que cada uno de los clientes impone una restricción temporal, también llamada ventana de tiempo, donde se permite el servicio de entrega o despacho.

En el presente trabajo, se propone dar solución al problema de ruteo de vehículos con ventanas de tiempo, a través de métodos heurísticos de construcción de ruta programados computacionalmente. La eficiencia de los algoritmos se mide con

instancias del tamaño de 100 nodos propuestas por Solomon (1987) La herramienta usada en la implementación de los algoritmos de solución del VRPTW es Matlab® versión R2010a.

Para el grupo de investigación OPALO y la Escuela de Estudios Industriales y Empresariales de la Universidad Industrial de Santander, el desarrollo de la herramienta aporta a la consolidación y fortalecimiento de la línea de investigación de ruteo, con miras a los retos industriales y comerciales que en la actualidad existen por la implementación de tratados de libre comercio y el desarrollo de infraestructura y tecnología logística de punta.

En el documento se exponen aspectos teóricos de la optimización combinatoria, las principales técnicas de optimización y el problema de ruteo de vehículos, y su variante VRPTW; igualmente se explican las heurísticas usadas para dar solución al problema y se realiza el análisis de los resultados obtenidos mediante la herramienta computacional. Por último se presenta el manual de usuario de la herramienta explicado paso a paso, el código del programa, y un ejercicio práctico en el que se exponen algunas heurísticas estudiadas.

TABLA DE CUMPLIMIENTO DE OBJETIVOS	
OBJETIVO	**CUMPLIMIENTO**
• Compilar la información de VRPTW de heurísticas utilizadas en la literatura para la solución del problema.	Subcapítulo 3.5. Capítulo 3. Capítulo 4.
• Buscar instancias aplicables y útiles para la solución al problema VRPTW.	Subcapítulo 3.3.
• Estudiar y sistematizar algunas de las heurísticas más representativas para el problema VRPTW.	Capítulo 4. Anexo 1. Anexo 2. Anexo 3.
• Seleccionar el conjunto de heurísticas de acuerdo a los criterios de eficiencia que sean definidos.	Capítulo 4. Página 74.
• Diseñar un programa en Matlab® que ejecute el modelo con las heurísticas seleccionadas.	Anexo 2. Anexo 3. Anexo: Software VRPTW.
• Diseñar y redactar un documento tipo manual de procedimientos de la herramienta programada.	Anexo 2.
• Evaluar y analizar los rendimientos de cada una de las heurísticas ejecutadas.	Capítulo 5.
• Documentar los análisis, resultados y conclusiones, en un artículo publicable.	Anexo 4.

1 OBJETIVOS

1.1 OBJETIVO GENERAL

Solucionar el Problema de ruteo de vehículos con ventanas de tiempo (VRPTW) mediante algunos métodos heurísticos en el Software Matlab ®

1.2 OBJETIVOS ESPECÍFICOS

- Compilar la información de VRPTW de heurísticas utilizadas en la literatura para la solución del problema.
- Buscar instancias aplicables y útiles para la solución al problema VRPTW.
- Estudiar y sistematizar algunas de las heurísticas más representativas para el problema VRPTW.
- Seleccionar el conjunto de heurísticas de acuerdo a los criterios de eficiencia que sean definidos.
- Diseñar un programa en Matlab® que ejecute el modelo con las heurísticas seleccionadas.
- Diseñar y redactar un documento tipo manual de procedimientos de la herramienta programada.
- Evaluar y analizar los rendimientos de cada una de las heurísticas ejecutadas.
- Documentar los análisis, resultados y conclusiones, en un artículo publicable.

2 GENERALIDADES

2.1 SISTEMAS LOGÍSTICOS

La Logística es "*un componente de la Cadena de Suministro que planifica, lleva a cabo y controla el flujo y almacenamiento eficiente y efectivo de bienes y servicios, así como de la información relacionada*".[5]

El Sistema Logístico es la integración de las actividades logísticas que permiten cumplir varios objetivos:[6]

- Hacer prioritarias las necesidades del cliente.
- Introducir la flexibilidad necesaria en la distribución para satisfacer las necesidades de un mercado cambiante.
- Reaccionar rápidamente ante los pedidos del cliente.

De tal manera que se puedan suministrar los productos o servicios necesarios, en el momento oportuno, en las cantidades requeridas, con la calidad apropiada y al mínimo costo.

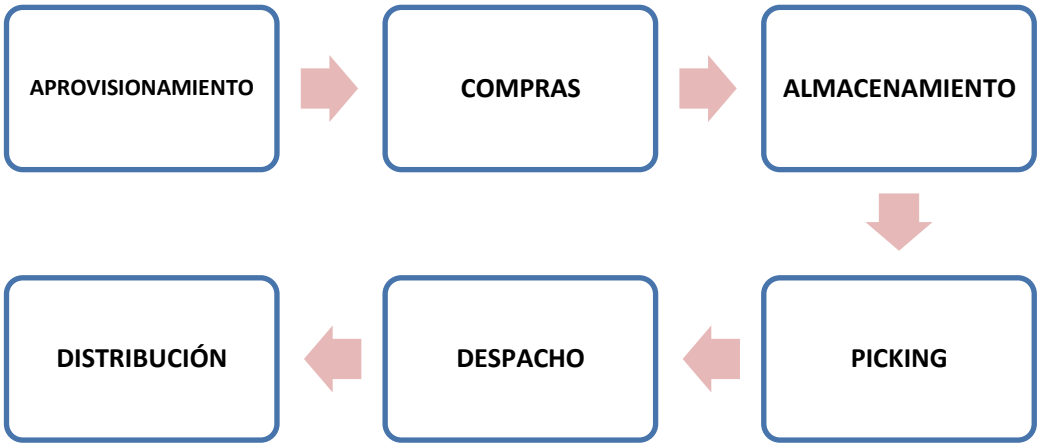
La Logística se afirma como una de las funciones clave de las empresas puesto que integra grandes actividades de administración de materiales, recursos e información, que permiten a las industrias cumplir con sus objetivos y razón de ser. La Logística en general, converge en una serie de actividades interrelacionadas entre sí que se representan en la Figura 1.

---

[5] SHAPHIRO, R.D. (1984) *Harvard Business Review,* mayo-junio, 119-126. "Apalancamiento a tráves de la logística".

[6] PAU COS, Jordi. Y IAVASCUES Y GASCA, Ricardo. Manual de Logística Integral [online]. Díaz de Santos [Madrid, España]: Google Books, 2011 [Citado el 24 de Noviembre de 2011; 17:00:00]. Disponible en:http://books.google.com.co/Books.

Figura 1. Flujo de las actividades logísticas



Fuente: Autores

De acuerdo a la Figura 1, las actividades principales del proceso logístico se organizan en forma de flujo. En el proceso de aprovisionamiento y compras, una empresa industrial obtiene de sus proveedores materiales, componentes e insumos que son transportados físicamente por sus instalaciones hasta la zona de almacenamiento; a estos artículos se les agrega valor mientras son transformados. Luego se preparan los pedidos en la zona de picking, según las requisiciones de los clientes y se despachan para ser transportados a través de una red de distribución física hasta los clientes o usuarios finales.

Cada empresa debe diseñar su proceso logístico de tal manera que pueda disminuir los costos de servicio y operación y redefinir sus procesos, conscientes de las tipologías de los clientes.

2.2 LOGÍSTICA DE DISTRIBUCIÓN

La logística de distribución es un término ampliamente utilizado y que suele abarcar muchas operaciones logísticas. Su objetivo es nivelar las variaciones físicas que existen entre la demanda de los clientes y la distribución.

Las actividades logísticas de distribución abarcan las operaciones comprendidas entre la carga del vehículo desde el depósito o Centro de Distribución hasta la descarga de mercancía en los distintos puntos de venta, y su objetivo es conseguir que los productos estén en el momento que el cliente lo requiera con un mínimo costo.

Se considera a la distribución como el último eslabón de las actividades logísticas de las empresas. Por ello es de suma importancia no escatimar recursos en la búsqueda de la efectividad en las operaciones de distribución, ya que de nada servirían los esfuerzos realizados en los procesos anteriores de diseño y producción en el caso que el cliente percibiera incumplimiento en las entregas o en los requerimientos.

Es importante que las herramientas logísticas caminen junto a la planeación estratégica, para identificar aquellos puntos o elementos diferenciales en relación a los competidores.

El sistema de distribución logística puede convertirse en un punto clave de la competitividad para la empresa, pues los únicos costos logísticos claramente identificados son los de transporte, que pueden intervenir en gran manera en la constitución del precio final del producto o servicio; por consiguiente, el presente trabajo se enfoca en el eslabón de la distribución, mediante la formulación y solución de cierto problema de transporte.

2.3 OPTIMIZACIÓN COMBINATORIA

Para estudiar el problema de distribución, se requiere modelar o construir una representación aproximada de la realidad de los procesos y actividades en un sistema logístico de distribución.

Un modelo, "*es una representación aproximada de la realidad. La construcción de un modelo implica la definición de un problema y la traducción del problema a relaciones matemáticas*".[7]

Existen dos tipos de problemas, los problemas de decisión, en los que se define un conjunto de alternativas o variables de decisión al problema, y los problemas de optimización, que hallan la mejor configuración existente en el conjunto de alternativas.

El primer tipo de problema toma decisiones representadas en variables de decisión $(x_1, x_2, x_3 \dots x_n)$, que pertenecen al conjunto de números reales $\mathbb{R}^{n+}$; Se utilizan para construir la función objetivo y las restricciones propias del modelo. La función objetivo se denota como $f(x_1, x_2, x_3 \dots x_n)$, mide el desempeño de un "mundo real" y describe el comportamiento del sistema a través de la relación que existe entre las variables y restricciones que lo definen.

Un problema de optimización consiste en encontrar la solución "óptima", o aquella que encuentre el mejor valor (máximo o mínimo) de la función objetivo. Evidentemente, es más fácil determinar si existe una configuración que cumpla ciertas restricciones, que encontrar la solución óptima, por esta razón, los problemas de decisión son más sencillos de resolver que los problemas de optimización.

[7] TAHA, Hamdy. Investigación de Operaciones. 7ma. Edición. Pearson Educación. México, 2004. P3.

Un problema de optimización se define como:[8]

> *Minimizar o Maximizar una determinada función,* $f(x_1, x_2, \dots, x_n)$ *sujeta a un conjunto de restricciones que acota el espacio posible de soluciones* $\underline{x} \in A$ *.*

$f(x_1, x_2, \dots, x_n)$, es la función objetivo o función de costo del problema $x = (x_1, x_2, \dots, x_n) \in A$ es el espacio de soluciones que contiene las combinaciones de las variables de decisión $(x_1, x_2, x_3 \dots x_n)$. Es decir, si se corre el modelo y x está en el conjunto A, se dice que x es factible.

2.3.1 Problemas Combinatorios

Cuando la función objetivo $f(x_0)$ y las restricciones del modelo son lineales, se presenta un problema de programación lineal. Sin embargo, si alguno de las funciones del modelo es no lineal, se presenta un problema de optimización no-lineal. Los modelos no-lineales pueden ser linealizados a través de artificios matemáticos.

Si las variables de decisión de los problemas lineales pertenecen a los enteros positivos, el problema de optimización se denomina problema de programación lineal entera

Existen problemas donde las variables de decisión son de naturaleza binaria, es decir, que admiten solo dos valores lógicos. Si a las variables se les incorpora cierta medida de incertidumbre, se tiene un problema de programación estocástica. Si las variables son de naturaleza discreta, binaria y continua, se presentan problemas de programación lineal entera mixta

[8] GALLEGO Olatz. Soluciones en Simulated Annealing para el VRPTW. Tesis Doctoral. Facultad de Informática. P.10. Universidad del País Vasco. 2002.

Cuando las variables se agrupan en varios conjuntos que representan objetos, casos o grafos, la técnica de optimización usada es de carácter combinatorio. La optimización combinatoria se encarga de ubicar dichas variables u objetos en ciertas posiciones. El conjunto de posiciones se llama configuración y la mejor configuración encontrada, es la optimización combinatoria del modelo. En una configuración, las variables del modelo están relacionadas de muchas maneras con los demás conjuntos de variables del problema. Un problema de optimización combinatoria se suele representar por medio de un problema de programación lineal (no lineal, entero).

Como se mencionó, el modelo logístico de distribución es un problema de optimización combinatoria, ya que las variables que caracterizan las rutas, los vehículos y los clientes son de naturaleza discreta y se relacionan de muchas maneras unas de otras: un vehículo puede recorrer una ruta específica para visitar un cliente específico, o el mismo vehículo puede ser usado para recorrer otra ruta y visitar el mismo cliente, etc. La idea es encontrar la mejor configuración de los objetos de tal manera que se optimice o mejore la función que los define.

2.3.2 Complejidad Computacional

El número de interrelaciones de las variables discretas en los problemas combinatorios puede llegar a ser tan grande, que la técnica usada para resolverlo tardaría una cantidad exorbitante de tiempo de cómputo explorando el universo de configuraciones; por eso, el tiempo computacional que usa una técnica para arrojar una respuesta es una condición muy importante a la hora de seleccionar una algoritmo de solución.

Los problemas de orden polinomial o de ejecución en tiempo polinómico, se definen como problemas ***P***, y son problemas tratables en términos de gasto computacional. De otro lado, los problemas de orden exponencial o ***NP***, son

problemas que se resuelven en un tiempo de cómputo no polinómico. Naturalmente, cualquier problema en la clase ***P***, también se encuentra en ***NP***.

El conjunto de problemas de orden ***NP*** son problemas que pueden ser resueltos en tiempo polinómico si se tiene un algoritmo que arroje una solución en tiempo polinómico. Si el algoritmo brinda una solución óptima del modelo en tiempo polinómico, entonces el conjunto de problemas puede ser reducido a orden ***P***, y se tendría un problema ***NP***-Completo.

Los problemas ***NP***-Completos, son los problemas ***NP*** más difíciles de resolver. Un problema es ***NP***-Completo, si cualquier otro problema ***NP*** puede ser trasformado en él. Los problemas considerados como los más difíciles en ***NP***-Completo, se definen como ***NP***-Hard.

Los problemas de distribución logística, de ruteo, o de encaminamiento, como el problema del agente viajero (TSP), el problema de ruteo de vehículos (VRP) o el problema de ruteo de vehículos con ventanas de tiempo (VRPTW), son problemas de orden ***NP***-Hard debido a la dificultad que se tiene de enumerar todas las configuraciones de las soluciones, debido a las múltiples interrelaciones de sus variables y el gran tamaño que tienen los datos de entrada del modelo.

2.3.3 Técnicas de Optimización

Debido a la importancia de los problemas de optimización combinatoria en aplicaciones prácticas, científicas e industriales, se han desarrollado múltiples métodos para resolverlos. Las técnicas desarrolladas se pueden clasificar en exactas o exhaustivas, y aproximadas.

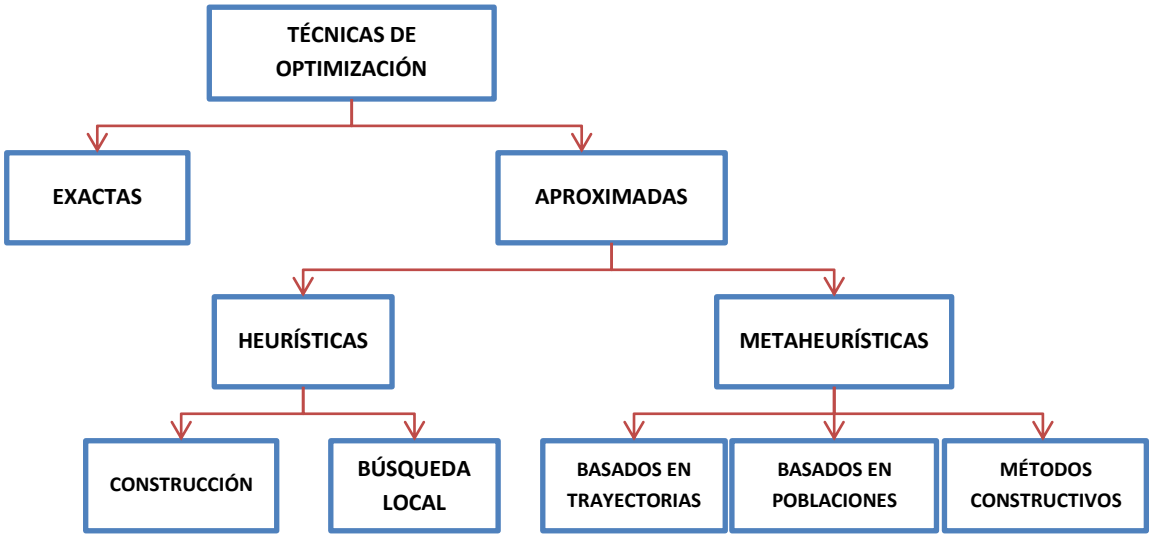
Las técnicas exactas encuentran la solución óptima para cualquier instancia de un problema en un tiempo determinado. Para problemas ***NP***-Hard, el inconveniente

de usar estos métodos, es el tiempo de cómputo necesario para determinar una solución óptima. Este tiempo crece exponencialmente con el tamaño del problema. Esto quiere decir, que a medida que el modelo es iterado, se requieren cantidades exponenciales de recurso computacional en la búsqueda de la solución.

En su lugar, se han desarrollado métodos aproximados que permiten encontrar una solución factible, que en algunos casos puede considerarse cercana a la solución óptima, y que es posible hallarla en un tiempo razonable y con moderado recurso computacional.

Dentro de los algoritmos aproximados se pueden encontrar dos tipos: las heurísticas y las metaheurísticas. En la Figura 2 se presenta una clasificación general de las principales técnicas de optimización.

Figura 2. Clasificación General de las Principales Técnicas de Optimización.



Fuente: Autores

Las heurísticas son técnicas de indagación y descubrimiento mediante métodos no rigurosos que buscan “buenas” soluciones a los modelos de optimización

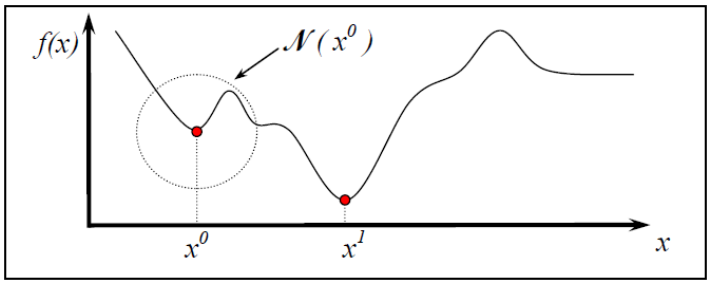
combinatoria. Se pueden clasificar en heurísticas de construcción y de búsqueda local.

Los métodos de búsqueda local parten de una solución inicial y, usando el concepto de vecindario, recorren parte del espacio de búsqueda hasta encontrar un óptimo. El vecindario de una solución factible es el conjunto de soluciones factibles, cercanas en cierto sentido a la actual, que se construyen a partir de un operador o una función específica denominada *movimiento.* El óptimo local es la mejor solución dentro de una vecindad. Los Heurísticos de búsqueda local, exploran el vecindario y se quedan con el mejor conjunto de soluciones locales.

Por otro lado, las heurísticas de construcción o constructivas, crean una solución desde cero e incorporan varios componentes en las iteraciones del modelo hasta satisfacer todas las restricciones del problema.

La Figura 3 ilustra un problema en el que se busca un valor mínimo para la función $f(x)$. $N(x)$ representa el vecindario de x, y $N(x^0)$ es la solución local del vecindario. A través de operaciones de movimiento, la técnica de optimización debe explorar todo el espacio de soluciones hasta hallar el vecindario del óptimo global $N(x^1)$.

Figura 3. Óptimo local y óptimo global



Fuente: VÉLEZ, Mario Cesar. Metaheurísticos: Una alternativa para la solución de problemas combinatorios en administración de operaciones. Escuela de Ingeniería de Antioquia, Medellín (Colombia) revista EIA, ISSN 1794-1237 Número 8, p. 99-115. Diciembre 2007.

Los heurísticos hacen que el método quede atrapado en el óptimo local x^0 y no generan movimientos para saltar a otro punto del espacio de búsqueda. En la actualidad, se han desarrollado técnicas de optimización más avanzadas que usan heurísticas de más alto nivel y que permiten la exploración de un mayor espacio de búsqueda de manera eficiente. Estas herramientas son denominadas metaheurísticas.

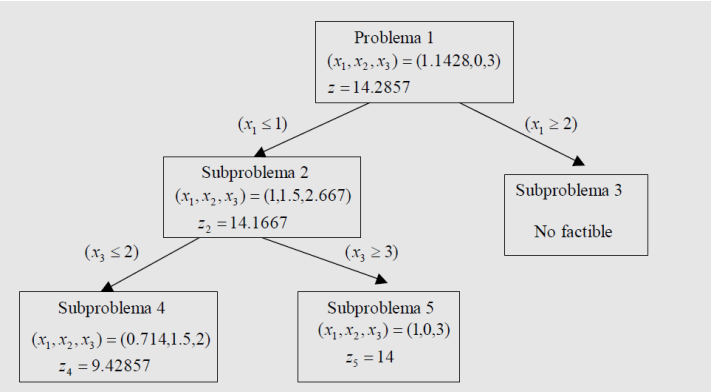
Las metaheurísticas son métodos aproximados diseñados para resolver problemas de optimización combinatoria, que proporcionan soluciones casi óptimas. Son procedimientos iterativos que guían una heurística dentro de un espacio de soluciones, combinando diferentes conceptos de la inteligencia artificial o la evolución biológica.

A continuación se describen algunas generalidades de las técnicas de optimización anteriormente descritas.

2.3.3.1 Métodos Exactos

Los métodos exactos son técnicas de búsqueda refinada; esto es, dividen el espacio de búsqueda en pequeños problemas más sencillos, que al solucionarlos, llevan a la solución final. La Figura 4 describe un método exacto que divide el problema general en pequeños sub-problemas, cada uno con restricciones y características definidas.

Figura 4. Procedimiento de un método exacto



Fuente: GARCÍA, José Pedro. Modelos y Métodos de Investigación de Operaciones. Grupo de Investigación ROGLE. Departamento de Administración de Empresas.

Para problemas complejos como el de Ruteo de Vehículos, enumerar y resolver todos los sub-problemas, resulta ineficiente o simplemente imposible. Generalmente se requiere alta memoria computacional que no poseen la mayoría de sistemas informáticos.

2.3.3.2 Métodos Heurísticos

En los problemas de carácter combinatorio, las soluciones que cumplen con las restricciones del modelo, se llaman *soluciones factibles del modelo.*

En el espacio de soluciones factibles, pueden existir uno o varios óptimos locales, que son definidos por el concepto de vecindad o respuestas vecinas de una solución. El óptimo local es aquella solución que es mejor en el vecindario de búsqueda. El conjunto de soluciones factibles puede variar por medio de una operación llamada movimiento, que depende de la naturaleza del problema y de la técnica a usar para la búsqueda de soluciones.

El óptimo global es aquella solución en el que la función objetivo no se puede mejorar a través de los movimientos del conjunto de soluciones en todo el espacio de búsqueda.

Existen problemas para los que no es indispensable hallar la solución óptima; con una buena solución basta para analizar el comportamiento del modelo que se optimiza. Los métodos heurísticos permiten hallar esas soluciones, que aunque no son óptimos globales, cumplen con las restricciones establecidas para el modelo y mejoran de alguna manera la función objetivo.

Las heurísticas exploran buenas soluciones en un tiempo razonable basados en el sentido común. Estas técnicas son usadas cuando se requiere tomar decisiones en los sistemas complejos o cuando se cuenta con poco tiempo para analizarlos y ponerlos en marcha; en ciertos sistemas logísticos, por ejemplo, es complejo conocer la combinación óptima de los recursos, debido a que se cuenta con poco tiempo o se encuentran múltiples variaciones o fluctuaciones en los procesos.

Las heurísticas permiten agregar condiciones y restricciones que son complicadas de modelar. Además son técnicas más flexibles para representar funciones y restricciones que los métodos exactos, es decir, “no todas las formulaciones heurísticas requieren de una especificación matemática detallada del sistema”.

Las técnicas heurísticas son la primera aproximación del proceso de optimización de un problema, ya que se usa como insumo para otros algoritmos más robustos.

Una descripción detallada de las técnicas más utilizadas se describe más adelante.

2.3.3.3 Métodos Metaheurísticos

Las metaheurísticas son usadas cuando no hay un método exacto de solución, requieren tiempos de cálculo grandes y uso de gran cantidad de recurso computacional.

Para obtener buenas soluciones, se debe establecer el balance adecuado entre dos características del proceso: La intensificación, que es la cantidad de esfuerzo de búsqueda empleado en el actual espacio de soluciones; y la diversificación, que es la cantidad de esfuerzo computacional usado en la exploración de otras regiones del espacio de soluciones.

Este equilibrio es necesario para identificar rápidamente regiones del espacio con soluciones de buena calidad, para no consumir recurso en regiones del espacio ya exploradas.

2.4 EL PROBLEMA DE RUTEO DE VEHÍCULOS

En los últimos años, se ha incrementado el desarrollo y uso de software de optimización basado en diversos modelos matemáticos, para un manejo más eficiente de la provisión de bienes y servicios en los procesos logísticos de distribución.

De la misma manera, el avance de las herramientas algorítmicas ha permitido modelar las actividades de distribución para analizar las restricciones complejas asociadas a procesos del mundo real y tener soluciones buenas en tiempos de cómputo aceptables, que permiten la toma de decisiones rápidas, precisas, y con un mejor manejo de los recursos logísticos.

La distribución de bienes y servicios entre depósitos o centros de distribución, a usuarios o clientes finales, es la actividad que se modela en los Problemas de Ruteo de Vehículos o VRP (Vehicle Routing Problems).

Las actividades de distribución en instancias reales, son tan variadas como el problema mismo: Transporte escolar, servicio de entrega de correspondencia, recolección de basura, transporte de ambulancias, de unidades de mantenimiento,

entre otras. El VRP pertenece a la clase de problemas **NP**-Hard y se usan técnicas de optimización combinatoria para solucionarlos.

2.4.1 Definición

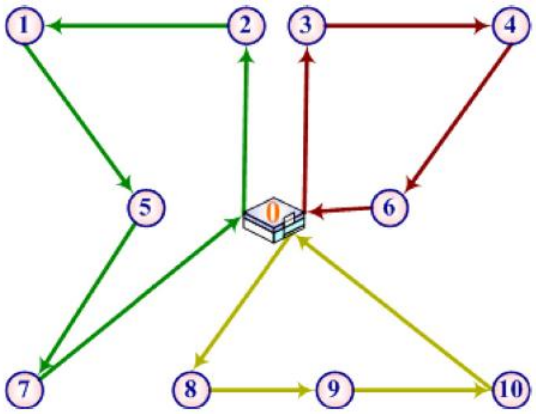
La distribución de bienes a través del ruteo, es el servicio prestado por una flota de vehículos que parten de uno o más depósitos, a unos clientes finales distribuidos en una zona geográfica. Para desplazarse hacia los clientes a través de esa zona, los vehículos usan una red de rutas que tienen asociadas variables de costo: costos de transporte, tiempos de recorrido, etc.

Establecer cuáles rutas debe seguir los vehículos para cumplir con los requerimientos del cliente, determina una solución al problema. La solución minimiza los recursos y costos que se deben invertir en el despliegue de la operación logística. Las rutas deben satisfacer gran cantidad de restricciones operacionales que dependen de los bienes que se transportan, el nivel de servicio que se ofrece o las características propias de los clientes y la flota.

La red de rutas en las que se desplazan los vehículos, generalmente se representa a través de un grafo, los arcos del grafo representan caminos o secciones de la ruta, y los nodos o vértices representan los clientes. Cada arco o camino, debe ser recorrido por un solo vehículo.

La Figura 5 representa un problema típico de Ruteo de Vehículos. Cada color de las flechas representa un grafo. Los grafos también son llamados Ciclos Hamiltonianos, que son caminos que inician y terminan en un mismo punto (para este caso el depósito o punto 0), y pasan por cada vértice una sola vez. Cada línea dirigida del grafo es un arco o camino entre dos clientes, vértices o nodos que se representan con un número.

Figura 5. Representación gráfica de un problema de ruteo de vehículos.



Fuente: B. Yu a, Z.Z. Yang a,*, B.Z. Yao. A hybrid algorithm for vehicle routing problem with time windows. journal homepage: www.elsevier.com/locate/eswa. Transportation Management College, Dalian Maritime University, Dalian 116026 PR China

Los clientes tienen asociada una demanda de bienes, que debe ser satisfecha por la flota de vehículos, ya sea a través de actividades de entrega de productos o de picking. Algunas veces, las demandas de los clientes no son un bien, sino un servicio, en este caso, el cliente solo es visitado una vez por el vehículo.

La flota de vehículos varía según las restricciones de problema. Algunos, parten de un centro de distribución, y terminan su recorrido en otro depósito; los vehículos pueden ser homogéneos o iguales, o pueden tener capacidades y requerimientos de carga distintos, basados en la configuración de productos que puedan transportar, volúmenes, o compartimentos.

Para calcular el costo total de la operación y el cumplimento de las distintas restricciones operacionales del modelo, se requiere conocer la distancia y el tiempo de viaje entre nodos, y de cada uno de los nodos hacia el depósito.

Cuando se halla una solución final, solo una pequeña parte de la red de rutas original queda activa e indica cuál es la mejor distribución de ellas. Las soluciones

del problema son factibles si se cumple el principio de desigualdad triangular, es decir, que el costo de transporte o el tiempo de viaje de dos clientes adyacentes, debe ser menor que el tiempo de viaje recorrido si se incluyera un cliente intermedio en la ruta parcialmente formada.

Muchas funciones de costo pueden ser definidas como criterio de decisión en un problema. El criterio puede consistir en minimizar el costo total de transporte, que depende del tiempo de recorrido total, o de la distancia global recorrida; o buscar minimizar el uso de vehículos de la flota, o la carga de cada uno de los vehículos, o minimizar las penalizaciones que los clientes imponen por no cumplir el servicio con los requisitos establecidos, o finalmente la función puede estar definida por un promedio ponderado de todas las anteriores características. Las características de cada problema determinan la naturaleza de la función que se minimiza y de las restricciones.

2.4.2 Variantes del Problema de distribución / Ruteo.

El problema básico de ruteo y el más estudiado es el Problema del Agente Viajero (*Traveling Salesman Problem, TSP)*; donde se busca la mejor configuración para visitar un conjunto de ciudades pasando sólo una vez por cada una y llegando a la misma ciudad de la que se partió, la configuración (ruta o tour) que halla el problema debe ser la de menor distancia, evitando subtours.

Si existe más de un vendedor, sería un problema de múltiples agentes, *m-TSP*, y si al mismo problema se le agregan restricciones de capacidad, se tendrá un problema *VRP*.

Si la demanda de los clientes es estocástica, el SVRP modela el proceso con variables de demanda aleatoria.

El Problema de ruteo de vehículos con devoluciones (*VRPB*), involucra las devoluciones que puedan hacer los clientes al depósito, para lo cual es necesario subdividir el conjunto de clientes en dos grupos: unos son los clientes a quienes se les debe entregar el producto y los otros los clientes a quienes se les debe recoger producto. A partir de esto surge una restricción de prioridad de las entregas sobre las devoluciones, en la cual cada vez que una ruta involucre a los dos tipos de clientes, las entregas deben ser atendidas primero que cualquier cliente que vaya a hacer una devolución.

En este caso se tiene un problema de ruteo de vehículos con entregas y recogidas (*VRPPD*), que cuenta con clientes que puedan tener al mismo tiempo cantidades de productos por ser entregados y por ser recogidos. La demanda puede ser atendida como la diferencia entre las entregas y las recogidas que deben hacerse, albergando la posibilidad de que la demanda se haga negativa.

Si el objetivo es construir un conjunto de rutas al costo mínimo, de tal manera que se puedan cumplir con los requerimientos de demanda y tiempo de servicio que cada cliente específico impone, se presenta el Problema de Ruteo de Vehículos con Ventanas de Tiempo o *VRPTW*.

3 PROBLEMA DEL RUTEO DE VEHÍCULOS CON VENTANAS DE TIEMPO

El VRPTW (*Vehicle Routing Problem* por sus siglas en inglés), aparece como un área importante de investigación en la generalización y la aproximación al problema real de los modelos de distribución física; incluye tanto la dimensión temporal, como la dimensión espacial, que permiten una descripción más adecuada de la actividad logística de distribución.

3.1 CARACTERÍSTICAS

El Problema de Ruteo de Vehículos con Ventanas de Tiempo, "V*ehicle Routing Problem with time Windows*" (VRPTW), es una extensión del Modelo Básico de Ruteo (VRP). El VRPTW es un problema de decisión que consiste en diseñar un conjunto posible de rutas para una flota de vehículos de igual capacidad, que salen y llegan a un depósito, de modo que se visiten todos los destinos una sola vez con el costo mínimo, satisfaciendo restricciones horarias y de capacidad.

Una ruta inicia cuando un vehículo sale del depósito y recorre cierta zona definida por las restricciones del problema y culmina en el momento que su capacidad es totalmente usada, o bien la capacidad restante es insuficiente para servir a otro cliente. Cada ruta esta asociada al recorrido de un solo vehículo. El servicio que se presta a un cliente, se supone en actividades de entrega de pedidos según demanda o en *picking* según capacidad de la flota.

Los clientes están dispersos en un área geográfica y cada uno tiene asociado un intervalo de tiempo, o Ventana de Tiempo, en el que permiten el servicio de recogida o despacho. Las ventanas de tiempo definidas para cada cliente, originan una espera si el vehículo llega antes del límite inferior de la ventana de tiempo de este cliente, e impiden el inicio del servicio si se supera su límite superior.

Las ventanas de tiempo del problema pueden ser flexibles o suaves, o no flexibles o duras. Para el problema de ruteo con ventanas de tiempo suaves, se permiten violar la restricción de inicio de servicio dentro de las ventanas de los clientes, sin embargo, este hecho implica una penalización económica por incumplimiento que afecta negativamente la función de costo. Por otro lado, las ventanas de tiempo duras no admiten el inicio del servicio después del tiempo más lejano en el que algún cliente puede ser atendido.

Las matrices de costo asociadas a las rutas y los tiempos de viaje, generalmente coinciden. Las matrices de Costo Simétricas son aquellas en las que el costo de ir de un cliente *i* a un cliente *j,* y volver por la misma ruta o grafo, es el mismo, es decir, son aristas no dirigidas; las matrices asimétricas, asignan costos distintos a la misma ruta.

La función de Costo varía según el alcance del modelo que se proponga; ya sea que se optimice el tiempo de viaje total, la cantidad de vehículos usados de la flota (para el caso VRPTW con flota heterogénea), la suma de los costos de transporte o una ponderación de todas.

Para la presente investigación, la función de costo está definida por variables de tiempo de recorrido, equivalentes a los costos de transporte de las rutas, (es decir que la unidad de costo es equivalente a la unidad de tiempo de recorrido definido para ella); la matriz de costo es simétrica, y las ventanas de tiempo asociadas a los clientes son duras o inflexibles. Además, cada ruta es recorrida por solo un vehículo, es decir que el número de rutas finales, es el número final de vehículos usados.

A continuación se presenta la formulación matemática del VRPTW. [9]

La red de transporte se representa por medio del grafo completo $G = (V, A)$ donde $V = \{0, \dots, n\}$ es el conjunto de vértices y $A \in (i, j)$ es el conjunto de arcos, con $i \neq j$. Un grafo completo posee aristas de todos a todos los nodos.

Los vértices $i = 1, \dots, n$ corresponden a los clientes y los vértices cero (0) o $n + 1$ corresponden al depósito. No aparece ningún arco que termine en 0 ni ninguno

[9] TOTH, Paolo, Daniele Vigo. “The vehicle routing problem”. SIAM monographs on discrete mathematics and applications. Philadelphia 2002. pp. 157.

que comience en $n+1$. Aunque corresponden al mismo punto, son distintos debido a las restricciones asociadas a cada uno.

La ventana de tiempo general asociada a los nodos de partida y de llegada se representa como $[a_0, b_0]$= $[a_{n+1}, b_{n+1}]$ = $[E, L]$, donde E y L representan la salida del depósito más temprana posible y la llegada al depósito más tardía posible respectivamente.

Toda ruta factible, corresponde a un camino $G = (V, A)$ que inicia en el nodo cero (0) y finaliza en el nodo $n+1$. Estos dos nodos no tienen demandas y tiempos de servicio definidos.

El sistema de distribución consta de un conjunto de vehículos igual a K con capacidad limitada igual a Q toneladas. El número de vehículos no es una restricción del problema. Se cuenta con una flota muy grande de vehículos para que, en caso de la peor solución, cada cliente sea servido por un vehículo en una ruta exclusiva.

- Para cada arco $A \in (i, j)$, Se definen:

    - c_{ij}. Costo no negativo correspondiente a cada arco $A \in (i, j)$, y asociado a la distancia, tiempo, flota, o una combinación ponderada de ellas. Representa el costo del camino más corto recorrido entre dos nodos i, j.

    - C_{ij}. Representa el costo de la función objetivo asociado al tiempo de viaje asignado a cada ruta.

- $t_{ij} =$ Tiempo de viaje para cada arco $A \in (i, j)$, que es equivalente a una unidad de costo. $C_{ij} = t_{ij}$.

- Se debe cumplir la desigualdad triangular $c_{ij} \leq c_{ih} + c_{hj}$ y $t_{ij} \leq t_{ih} + t_{hj}$.

- Variables de decisión de naturaleza binaria. $x_{ijk}, \ \forall i, j, \ 1 \leq i, j \leq n, \ i \neq j$.

  $x_{ijk} = 1$, si en la solución, el vehículo k va de i a j.
  $x_{ijk} = 0$, si en la solución, el vehículo k no va de i a j.

- Para cada nodo $V = \{0, \dots, n\}$ se definen:

  - s_i= Tiempo de duración del servicio para cada cliente.

  - $d_i =$ Demanda del nodo i, de tipo determinística.

  - Un intervalo de tiempo $[a_i, i], \ con \ i \in V$, llamado Ventana de Tiempo, dentro del cual un cliente puede ser servido.

  - La variable a_i, representa el tiempo más cercano de inicio de servicio en el cliente i.

  - La variable b_i, es el tiempo más lejano de inicio de servicio, o lo mismo, el fin de la ventana correspondiente al cliente i.

  - W_i, instante en el que el vehículo k, comienza a servir al cliente i, o cliente actual.

- W_j, instante en el que el vehículo k, comienza a servir al cliente j o cliente siguiente.

- Existe una solución factible si se cumplen las siguientes dos condiciones:

  - $a_0 = E \leq min_{i \epsilon V \setminus \{0\}} b_i - t_{0i}$, donde $b_i - t_{0i}$, es el tiempo disponible para servir al cliente i, saliendo del depósito. E representa el tiempo más tardío en el que se puede salir del depósito en función del cliente con ventanas de tiempo más prontas a cerrarse en el horizonte de programación.

  - $b_{n+1} = L \geq min_{i \epsilon V \setminus \{0\}} a_i + s_i + t_{i,n+1}$, donde $a_i + s_i + t_{i,n+1}$, es el tiempo que tarda un vehículo en llegar del último cliente servido hasta el depósito. L es el tiempo más temprano en el que se puede llegar al depósito, después de haber servido a todos los clientes. Si la llegada al depósito es mayor que L, no se garantiza el servicio a todos los clientes, debido a que al menos uno dejo de ser visitado.

- Un arco $A \in (i, j)$, es factible cuando cumple con las siguientes restricciones:

  Restricción de tiempo: Si la suma del tiempo de viaje de i a j, la duración del servicio en i y el intervalo correspondiente al inicio de la ventana de tiempo de i , superan el tiempo más tardío posible en el que el siguiente cliente j permite el servicio. $a_i + s_i + t_{ij} > b_j$.

  Restricciones de capacidad: Si la demanda de los clientes que van a ser servidos, es mayor que la capacidad de un vehículo. $d_i + d_j > Q$

El problema VRPTW es un problema de optimización combinatoria debido a la naturaleza discreta de sus variables. Existen miles de posibilidades de encontrar una solución que cumpla con las condiciones matemáticas y mejore la función objetivo que se establezca.

3.2 FORMULACIÓN MATEMÁTICA

El problema de Ruteo de Vehículos con Ventanas de Tiempo VRPTW se formula matemáticamente de la siguiente forma[10]:

Función objetivo: Expresa el Costo Total. Se pretende optimizar el tiempo total de recorrido $C_{ij} = \sum t_{ij}$, donde t_{ij} es el tiempo de recorrido entre dos clientes i e j El costo de la función objetivo es el tiempo de viaje asignado a cada ruta.

$$Minimizar \sum_{k \in K} \sum_{(i,j) \in A} C_{ij} t_{i,jk} \qquad (\mathbf{1})$$

Sujeto a:

$$\sum_{\boldsymbol{k \in K}} \sum_{\boldsymbol{j \in \Delta + (i)}} \boldsymbol{x_{ijk}} = \mathbf{1} \;\; ; \;\; \forall \;\; i \in N \quad (\mathbf{2})$$

Dado $N = V \backslash \{0, n+1\}$ que representa el conjunto de clientes, a excepción del depósito; la restricción (2) asigna a cada cliente exactamente una ruta de vehículo. $j \in \Delta + (i)$ Es el conjunto de grafos completos que parten de i y llegan a j.

$$\sum_{\boldsymbol{j \in \Delta + (0)}} \boldsymbol{x_{0jk}} = \mathbf{1} \;\; ; \;\; \forall \;\; k \in K \qquad (\mathbf{3})$$

[10] Ibid. p, 158.

$$\sum_{i \in \Delta-(n+1)} x_{i,n+1,k} = 1 \;;\; \forall\, k \in K \quad (4)$$

Las restricciones (3) y (4) limitan el número de rutas por vehículo a una, y caracterizan el flujo que debe seguir la flota. La primera hace referencia al número de veces que un vehículo sale del depósito y la segunda, el número de veces que llega. $j \in \Delta + (0)$, es el conjunto de grafos completos que parten del depósito a todos los clientes y $j \in \Delta - (n+1)$ es el conjunto de grafos que parten de todos los clientes hacia el depósito.

$$\sum_{i \in \Delta-(j)} x_{ijk} - \sum_{i \in \Delta+(j)} x_{jik} = 0 \;\; ; \;\; \forall\; k \in K,\; j \in N \quad (5)$$

La restricción (5) asegura que a cada cliente solo llegue un vehículo y el mismo salga de él. Esta restricción permite que en el modelo no se formen ciclos. Al activarse el nodo x_{ijk}, este se elimina del conjunto de clientes no visitados para evitar repeticiones en la asignación de rutas.

$$x_{ijk}(w_{ik} + s_i + t_{ij} - w_{jk}) \leq 0 \;\; ; \;\; \forall\, k \in K,\; (i,j) \in A \quad (6)$$

La restricción (6) asegura que el vehículo k no pueda comenzar el servicio, si la suma del tiempo de viaje de i a j, la duración del servicio en i y el tiempo total acumulado al inicio del servicio en i (w_{ik}) es mayor que la ventana de tiempo de j o del cliente siguiente.

$$a_i \sum_{j \in \Delta+(i)} x_{ijk} \leq w_{ik} \leq b_i \sum_{j \in \Delta+(i)} x_{ijk} \;;\; \forall\, k \in K,\; i \in N \quad (7)$$

Para un vehículo k dado, la restricción (7) hace que $w_{ik} = 0$ cada vez que el cliente i no es visitado por el vehículo k. Igualmente, garantiza que el inicio de

servicio en el cliente i se lleve a cabo dentro de la ventana de tiempo correspondiente.

$$\boldsymbol{E_i \leq w_{ik} \leq L} \;\; ; \;\; \forall \; k \in K, \;\; i \in \{0, n+1\} \qquad (\mathbf{8})$$

La restricción (8) hace que cada cliente sea servido dentro del intervalo de tiempo general $[a_0, b_0]$= $[a_{n+1}, b_{n+1}]$ = $[E, L]$, asociada a los nodos de llegada y de salida (depósito).

$$\sum_{\boldsymbol{i \in N}} \boldsymbol{d_i} \sum_{\boldsymbol{j \in \Delta + (i)}} \boldsymbol{x_{ijk} \leq Q} \;\; ; \forall \; k \; \in K \qquad (\mathbf{9})$$

La restricción (9) indica que la suma de las demandas de los clientes de una ruta no debe exceder la Capacidad del vehículo k.

$$\boldsymbol{x_{ijk} \geq 0} \;\; ; \;\; \forall \; k \; \in K, \;\; (i,j) \in A \quad (\mathbf{10})$$

La restricción (10) impone condiciones de no negatividad a las variables del modelo

$$\boldsymbol{x_{ijk}} \in \{\mathbf{0}, \mathbf{1}\} \;\; ; \;\; \forall \; k \; \in K, \;\; (i,j) \in A \quad (\mathbf{11})$$

La restricción (11) indica la naturaleza binaria de las variables.

En resumen, el VRPTW se formula formalmente así:

$$Minimizar \sum_{k \in K} \sum_{(i,j) \in A} C_{ij} t_{i,jk} \qquad (\mathbf{12})$$

Sujeto a:

$$\sum_{k \in K} \sum_{j \in \Delta+(i)} x_{ijk} = 1 \;\; ; \;\; \forall \;\; i \in N \quad (13)$$

$$\sum_{j \in \Delta+(0)} x_{0jk} = 1 \;\; ; \;\; \forall \;\; k \in K \quad (14)$$

$$\sum_{i \in \Delta-(n+1)} x_{i,n+1,k} = 1 \;\; ; \;\; \forall \; k \;\in K \quad (15)$$

$$\sum_{i \in \Delta-(j)} x_{ijk} - \sum_{i \in \Delta+(j)} x_{jik} = 0 \;\; ; \;\; \forall \;\; k \in K, \;\; j \in N \quad (16)$$

$$x_{ijk}\left(w_{ik} + s_i + t_{ij} - w_{jk}\right) \leq 0 \;\; ; \;\; \forall \; k \in K, \;\; (i,j) \in A \quad (17)$$

$$a_i \sum_{j \in \Delta+(i)} x_{ijk} \leq \; w_{ik} \; \leq b_i \sum_{j \in \Delta+(i)} x_{ijk} \;\; ; \forall \; k \in K, \;\; i \in N \quad (18)$$

$$E_i \leq w_{ik} \leq \; L \;\; ; \;\; \forall \;\; k \in K, \;\; i \in \{0, n+1\} \quad (19)$$

$$\sum_{i \in N} d_i \sum_{j \in \Delta+(i)} x_{ijk} \leq Q \;\; ; \forall \;\; k \;\in K \quad (20)$$

$$x_{ijk} \geq 0 \;\; ; \;\; \forall \;\; k \;\in K, \;\; (i,j) \in A \quad (21)$$

$$x_{ijk} \in \{0, 1\} \;\; ; \;\; \forall \;\; k \;\in K, \;\; (i,j) \in A \quad (22)$$

Cada ruta tiene asociado un tiempo de programación o tiempo acumulado T, que se define como la suma de los tiempos de recorrido, los tiempos de espera y los tiempos de servicio en cada cliente visitado. La solución final se expresa con los

tiempos totales de recorrido $\sum_{k\in K}\sum_{(i,j)\in A} C_{ij}t_{i,jk}$ y los tiempos totales de programación o tiempo acumulado de las rutas $\sum_{(i,j)\in A}\ t_{ij} + s_i + te_i$, donde te, es la suma de los tiempos de espera. Se incurre en tiempos de espera cuando el tiempo de programación de la ruta o tiempo acumulado parcial es menor al inicio de la ventana de tiempo del cliente que se visita $te = a_i - T\ \ y\ te > 0;\ \ te = 0\ \ si\ a_i - T \leq 0$

3.3 INSTANCIAS

Las instancias son conjuntos de datos que se usan para medir la bondad de los resultados de los métodos propuestos de optimización; son usadas con frecuencia en el campo científico para comparar resultados propios con algunos otros de la literatura. Están asociadas a características propias del problema y se usan en el planteamiento o corrida específica de un modelo.

El conjunto propuesto por Solomon en 1983[13] es un valioso recurso y punto de referencia en la validación de técnicas heurísticas del VRPTW. El banco de pruebas consta de 56 instancias que se representan mediante la siguiente notación: C1 (09), C2 (08), R1 (012), R2 (011), RC1 (08) y RC2 (08). El número entre paréntesis representa la cantidad de instancias que existe en cada conjunto.

Los grupos de instancias difieren entre sí respecto al ancho de sus ventanas de tiempo, medido como la diferencia entre el fin e inicio de ventana, la ubicación geográfica de los clientes según sus coordenadas y la densidad de ventanas, entendido como el porcentaje de clientes con ventanas de tiempo, algunas con 25%, 50%, 75% y 100% de clientes.

En una instancia pueden existir clientes cuyo inicio de ventana tiene valor de 0. El conjunto de clientes que tienen inicio de ventana de tiempo diferente de cero, son clientes que poseen corto horizonte de tiempo para programarlos en una ruta

parcial. Para una instancia de 100 clientes con densidad de ventanas del **25%**, existe un conjunto de **25** clientes con inicios de ventanas diferentes de cero. Para las instancias de largo horizonte, existen densidades de ventanas bajas de 25% y 50%. Para la mayoría de instancias de corto horizonte de programación, las densidades de ventanas son del orden del 75% y del 100%.

Los datos geográficos y las demandas de algunos de los problemas, son los mismos usados por Christofides, Mingozzi y Toth en su investigación (1979) [14]. A continuación, se explica el significado de cada grupo de instancias.

En el banco de instancias de tipo R1 y R2 las coordenadas geográficas son generadas aleatoriamente (Random). Los grupos C1 y C2 (Clúster), comprenden los problemas en el que los clientes están agrupados por zonas o por clústeres. Los grupos RC1 y RC2 (Random-Clúster) son problemas de semiclústeres, es decir, que son una mezcla de los dos anteriores.

El conjunto de problemas **1**: R**1**, C**1**, y RC**1**, son de corto horizonte de programación; Los problemas **2**: R**2**, C**2** y RC**2**, son de largo horizonte de programación, es decir que muchos clientes pueden ser servidos por un mismo vehículo (hasta 30 clientes), generando menos rutas que en el conjunto de problemas 1 (entre 5 y 10).

Se usa el nombre del grupo (C, R, RC), seguido del horizonte de programación respectivo (1 ó 2), el número de problema concreto (01, 02, 03, etc.), y el número de clientes de la instancia separado por un punto, para identificar el tipo de problema que se va a correr en el modelo: Ejemplo, RC101.100, corresponde a la primera instancia de los problemas de Semiclústeres con 100 clientes.

Solomon (1987) desarrolló instancias con 100 clientes y 50 clientes para cada uno de los grupos R, C y RC. En las últimas investigaciones se han desarrollado

problemas con menos clientes (10, 25 o 50) para cada uno de los grupos, al igual que problemas de mayor dimensión (200, 400, 800 y 1000 clientes) (Homberger, basado en Solomon 1987) con 25, 50, 100, 150 y 200 vehículos, para probar métodos más robustos. Para la presente investigación, se asume una flotilla de vehículos suficientemente grande, por consiguiente, el número de vehículos no es una restricción del problema.

En la Tabla 1 se muestran los veinte primeros datos de una instancia de tipo clúster, C106.100; $\boldsymbol{X}$ e $\boldsymbol{Y}$, son las coordenadas geográficas de los clientes; $\boldsymbol{d}$ es la demanda asociada a cada cliente; $\boldsymbol{a_i}$ y $\boldsymbol{b_i}$ son el inicio y fin de las ventanas de tiempo por tanto la diferencia $\boldsymbol{b_i - a_i}$, representa el ancho de ventana; $\boldsymbol{S_i}$ es el tiempo de servicio. En todas las instancias el depósito se conoce como el último cliente, quien no tiene demanda asociada ni tiempo de servicio.

Tabla 1. Veinte primeros datos instancia C106.100

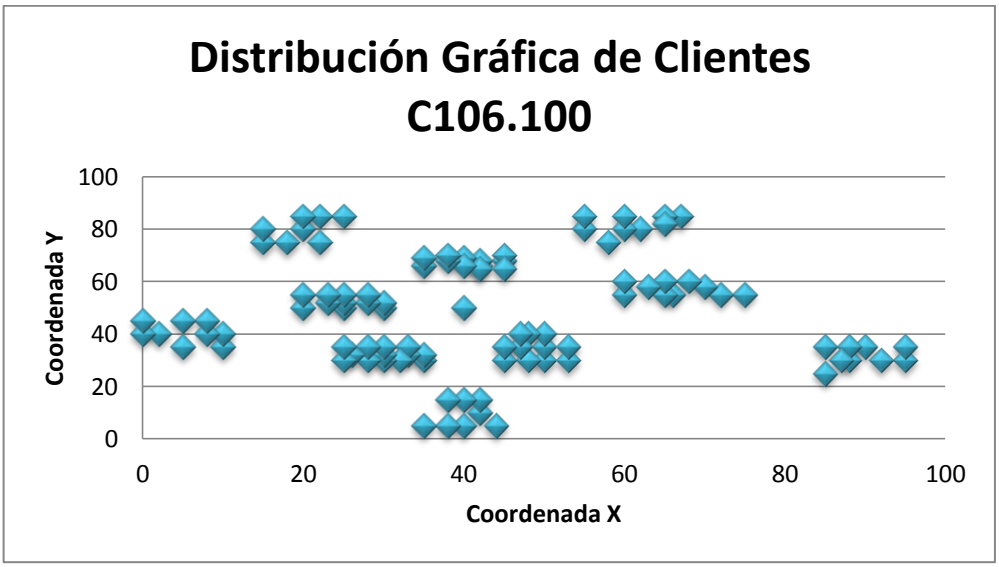
Cliente	X	Y	d	ai	Bi	si	bi-ai
1	45	68	10	890	989	90	99
2	45	70	30	816	879	90	63
3	**42**	66	10	55	156	90	101
4	**42**	68	10	703	806	90	103
5	**42**	65	10	15	60	90	45
6	40	69	20	559	764	90	205
7	40	66	20	172	223	90	51
8	38	68	20	250	329	90	79
9	38	70	10	489	650	90	161
10	35	66	10	361	406	90	45
11	35	69	10	450	503	90	53
12	25	**85**	20	647	726	90	79

13	22	75	30	30	95	90	65
14	22	**85**	10	571	616	90	45
15	20	80	40	392	421	90	29
16	20	**85**	40	478	525	90	47
17	18	75	20	105	142	90	37
18	15	75	20	172	261	90	89
19	15	80	10	274	349	90	75
20	30	50	10	10	77	90	67

Fuente: Autores

En la Figura 6 se observan las coordenadas X e Y para los 100 clientes de la instancia de clúster C106.100. Algunos clientes, como los clientes **3**, **4** y **5**, conservan la misma ubicación en la coordenada X, algunos otros en la coordenada Y (**12**, **14** y **16**). Para este tipo de instancias, La cercanía entre clientes hace que las rutas queden agrupadas en clústeres. Todas las instancias del grupo C, poseen esta misma característica.

Figura 6. Distribución gráfica de clientes C106.100

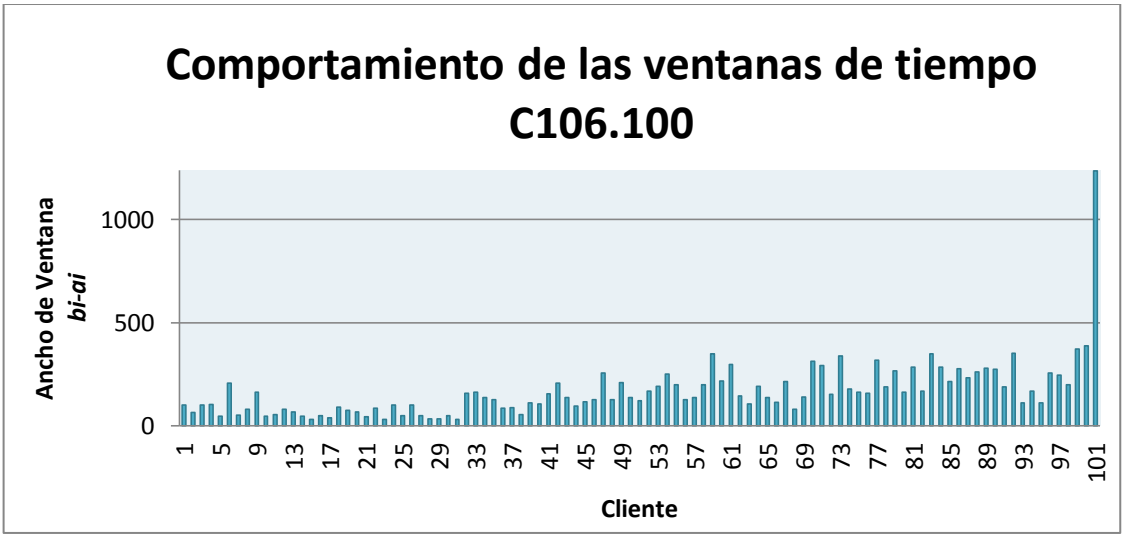


Fuente: Autores

Como se puede apreciar en la Figura 7 con respecto a las ventanas de tiempo, las instancias clusterizadas con corto horizonte de programación (C1), como la instancia C106.100, muestran intervalos de ventana variables. Algunos clientes tienen ventanas cortas del orden de las 30 unidades de tiempo, y otras ventanas de tiempo anchas de más de 100 unidades de tiempo.

La anterior distribución hace que las instancias de clústeres C1 tengan corto horizonte de programación. Los clientes con mayores intervalos de ventanas de tiempo, generalmente son servidos en rutas exclusivas o con pocos clientes en la misma ruta; por otro lado, los clientes con ventanas grandes generalmente acompañan rutas más largas con amplio horizonte de programación.

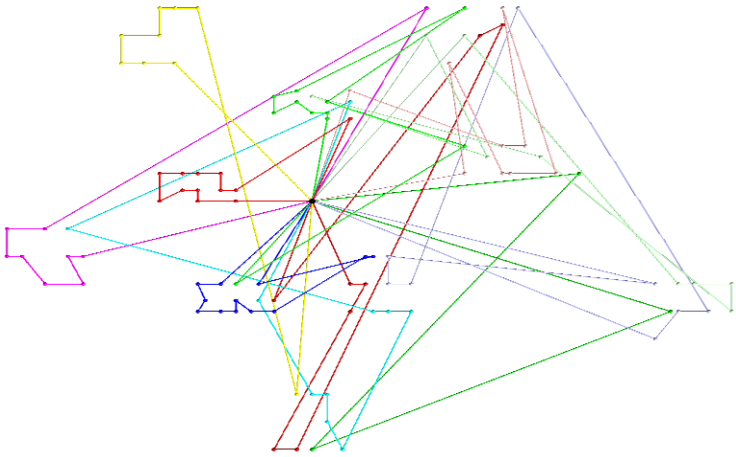
Figura 7. Comportamiento de las ventanas de tiempo C106.100



Fuente: Autores

La Figura 8 es la gráfica de una solución factible a la instancia C106.100, donde se evidencia la distribución clusterizada de los clientes en la zona geográfica y la uniformidad de las rutas. Cada vértice es un cliente y cada color de línea representa una ruta que inicia y termina en el punto central o depósito.

Figura 8. Solución factible Instancia C106.100 usando algoritmo del Vecino más cercano.



Fuente: Autores.

Las instancias de largo horizonte de programación clusterizada o C2, se caracterizan por tener intervalos de ventana mucho más grandes que las instancias C1. La Tabla 2 muestra los primeros veinte datos de la instancia C205.100.

Tabla 2. Veinte primeros datos instancia C205.100.

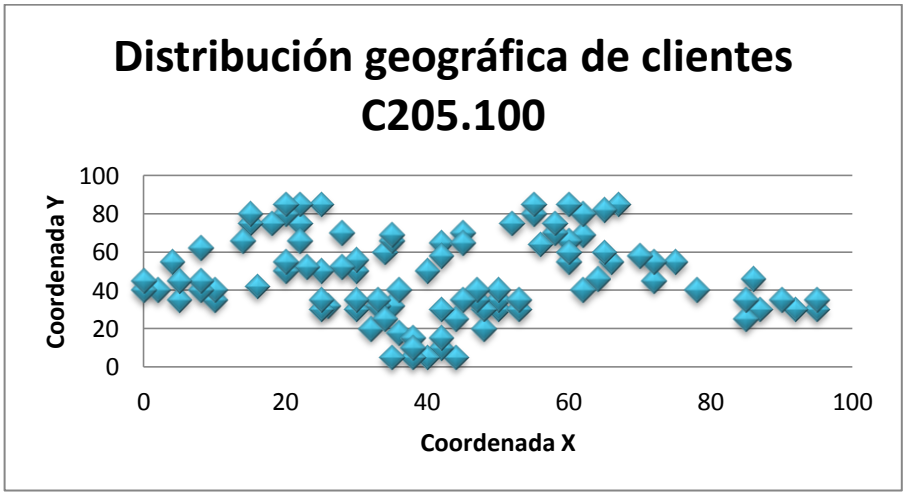
Cliente	X	Y	d	ai	Bi	si	bi-ai
1	52	75	10	231	551	90	**320**
2	45	70	30	133	453	90	**320**
3	62	69	10	1087	1407	90	**320**
4	60	66	10	1181	1501	90	**320**
5	42	65	10	15	335	90	**320**
6	16	42	20	417	737	90	**320**
7	58	70	20	993	1313	90	**320**
8	34	60	20	2807	3127	90	**320**

9	28	70	10	2521	2841	90	**320**
10	35	66	10	2711	3031	90	**320**
11	35	69	10	2618	2938	90	**320**
12	25	85	20	2039	2359	90	**320**
13	22	75	30	2325	2645	90	**320**
14	22	85	10	1946	2266	90	**320**
15	20	80	40	2136	2456	90	**320**
16	20	85	40	1854	2174	90	**320**
17	18	75	20	2231	2551	90	**320**
18	15	75	20	1662	1982	90	**320**
19	15	80	10	1757	2077	90	**320**
20	30	50	10	10	330	90	**320**

Fuente: Autores

En la Figura 9 se muestra la distribución clusterizada de los clientes de la instancia C205.100. Se aprecia la cercanía de los clientes formando varias agrupaciones que causan la programación de rutas propias a cada conjunto.

Figura 9. Distribución geográfica de clientes C205.100

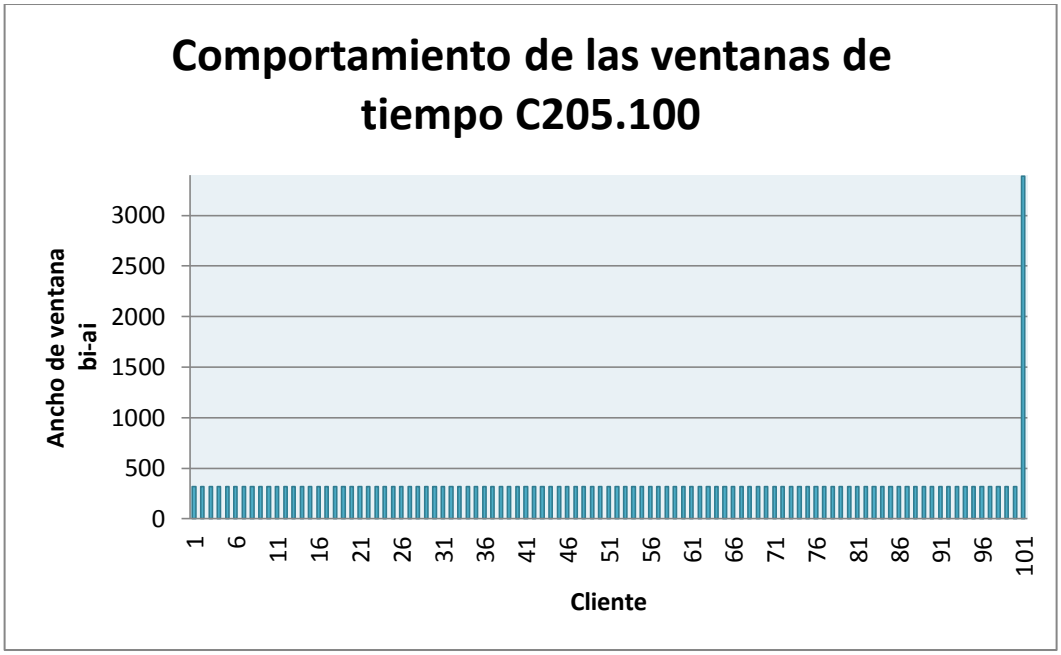


Fuente: Autores

Las instancias C2, como la C205.100, poseen grandes intervalos o anchos de ventana para flexibilizar la programación de las rutas. Muchos clientes son programados en la misma ruta puesto que hay holgura en las restricciones temporales, por ello las instancias C2 son de amplio horizonte de programación. Algunas instancias como la C207.100, tienen intervalos de ventana del orden de las 900 unidades.

Como se muestra en la Figura 10, la instancia C205.100 tiene intervalos de ventana del orden de las 320 unidades para todos los clientes.

Figura 10. Comportamiento de las ventanas de tiempo C205.100



Fuente: Autores

Las instancias con distribución geográfica aleatoria o del grupo R, se muestran en La Tabla 3 con la instancia R101.100. Se caracterizan porque sus coordenadas

fueron generadas de manera aleatoria para evitar la formación de patrones en la distribución espacial de los clientes.

Tabla 3. Veinte primeros datos instancia R101.100.

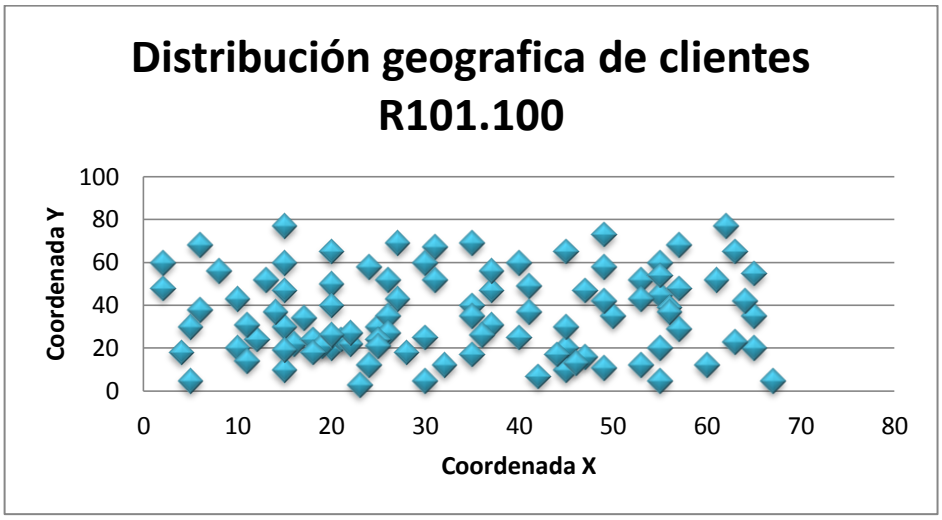
Cliente	X	Y	d	ai	Bi	si	bi-ai
1	41	49	10	161	171	10	**10**
2	35	17	7	50	60	10	**10**
3	55	45	13	116	126	10	**10**
4	55	20	19	149	159	10	**10**
5	15	30	26	34	44	10	**10**
6	25	30	3	99	109	10	**10**
7	20	50	5	81	91	10	**10**
8	10	43	9	95	105	10	**10**
9	55	60	16	97	107	10	**10**
10	30	60	16	124	134	10	**10**
11	20	65	12	67	77	10	**10**
12	50	35	19	63	73	10	**10**
13	30	25	23	159	169	10	**10**
14	15	10	20	32	42	10	**10**
15	30	5	8	61	71	10	**10**
16	10	20	19	75	85	10	**10**
17	5	30	2	157	167	10	**10**
18	20	40	12	87	97	10	**10**
19	15	60	17	76	86	10	**10**
20	45	65	9	126	136	10	**10**

Fuente: Autores

En la

Figura 11 se observa la distribución de los clientes de una instancia de tipo aleatorio; para el caso de la instancia R101.100 los clientes se encuentran dispersos entre 2 y 67 unidades en el eje de las abscisas y entre 3 y 77 en el eje de las ordenadas. Esta característica básica, hace que la programación de rutas tenga otro tipo de comportamiento, porque a diferencia de las instancias de tipo clúster, no hay una notoria definición de rutas asociadas a cada conjunto de clientes o clúster.

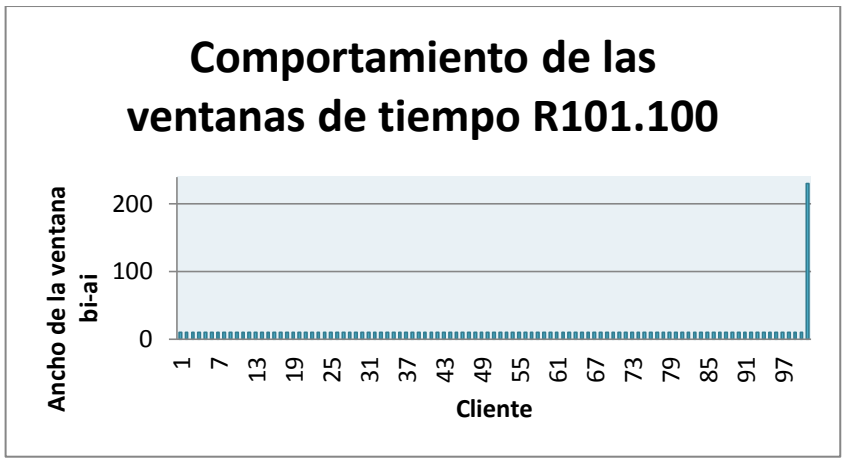
Figura 11. Distribución geográfica de clientes R101.100



Fuente: Autores

En el caso particular de la instancia R101.100 todos los clientes tienen el mismo valor del intervalo de ventana, en la Figura 12 se observan ventanas con 10 unidades de tiempo que resultan escasas, generan poca holgura en la programación de rutas y en algunos casos rutas exclusivas; dicho de otra manera, la distribución aleatoria de las coordenadas **X** e **Y** de los clientes y el corto intervalo de ventana hace necesario gran cantidad de rutas para servir la totalidad de clientes.

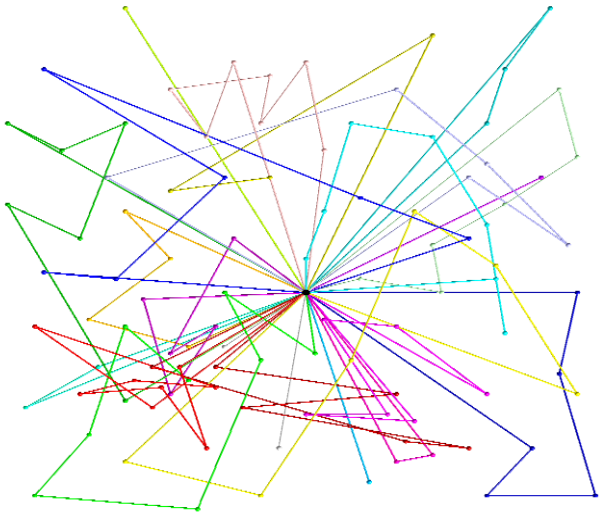
Figura 12. Comportamiento de las ventanas de tiempo R101.100



Fuente: Autores

En la Figura 13, se presenta la gráfica de una solución factible a la instancia R101.100. Allí se puede observar la distribución aleatoria de los clientes y la desproporción de las rutas. Cada vértice es un cliente y cada color de línea representa una ruta que inicia y termina en el punto central o depósito.

Figura 13. Solución Instancia R101.100 usando algoritmo del vecino más cercano.



Fuente: Autores.

Las instancias de orden aleatorio y de largo horizonte de programación R2, tienen intervalos de ventana que varían desde las 70 unidades hasta las 970 unidades y que hacen que se programen varios clientes en rutas con largo horizonte de programación. En la Tabla 4 se muestran los veinte primeros datos de la instancia R204.100.

Tabla 4. Veinte primeros datos instancia R204.100.

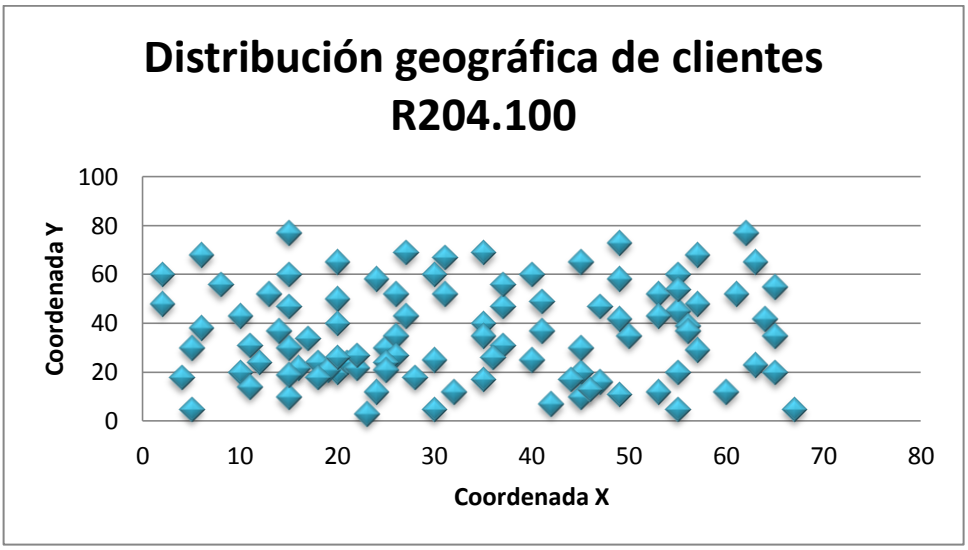
Cliente	X	Y	d	ai	Bi	si	bi-ai
1	41	49	10	0	974	10	**974**
2	35	17	7	0	972	10	**972**
3	55	45	13	0	967	10	**967**
4	55	20	19	678	801	10	**123**
5	15	30	26	0	969	10	**969**
6	25	30	3	0	978	10	**978**
7	20	50	5	0	968	10	**968**
8	10	43	9	404	481	10	**77**

9	55	60	16	400	497	10	**97**
10	30	60	16	0	964	10	**964**
11	20	65	12	206	325	10	**119**
12	50	35	19	0	975	10	**975**
13	30	25	23	690	827	10	**137**
14	15	10	20	0	957	10	**957**
15	30	5	8	175	300	10	**125**
16	10	20	19	0	960	10	**960**
17	5	30	2	0	959	10	**959**
18	20	40	12	0	974	10	**974**
19	15	60	17	0	957	10	**957**
20	45	65	9	0	958	10	**958**

Fuente: Autores

Como se observa en la Figura 14 las coordenadas geográficas de esta instancia son iguales a las coordenadas de todas las instancias de tipo R. Lo importante es notar que los clientes se encuentran dispersos de manera aleatoria, situación que puede llegar a hacer un poco más difícil la programación de las rutas.

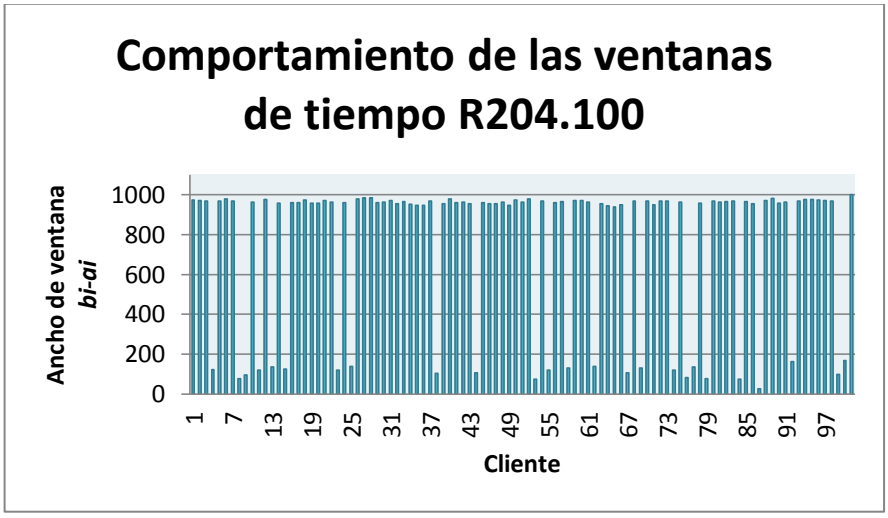
Figura 14. Distribución geográfica de clientes R204.100



Fuente: Autores

En la Figura 15 puede observarse el comportamiento de los intervalos, en este caso cada uno de los clientes difiere en su ventana de tiempo pero con una característica especial: se observan dos líneas de tendencia que destacan anchos de ventana cercanos al orden de los 150 y los 900 unidades de tiempo; aunque ambos valores son flexibles algunos clientes permiten mayor holgura haciendo más fácil la programación de las rutas.

Figura 15. Comportamiento de las ventanas de tiempo R204.100



Fuente: Autores

A continuación se estudiaran las instancias RC, o de semiclústeres, específicamente las del grupo de corto horizonte de programación. En la Tabla 5 se muestran los veinte primeros datos de la instancia RC101.100.

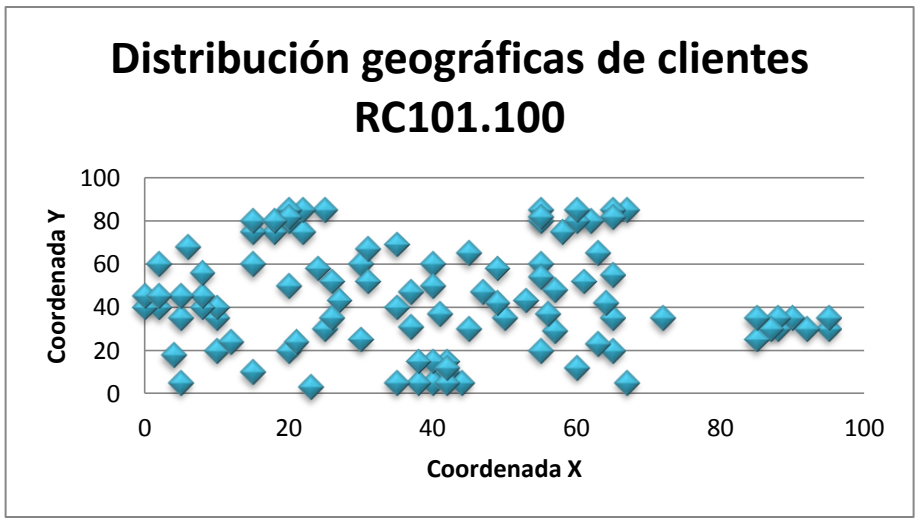
Tabla 5. Veinte primeros datos instancia RC101.100

Cliente	X	Y	d	ai	bi	si	bi-ai
1	25	85	20	145	175	10	**30**
2	22	75	30	50	80	10	**30**
3	22	85	10	109	139	10	**30**
4	20	80	40	141	171	10	**30**
5	20	85	20	41	71	10	**30**
6	18	75	20	95	125	10	**30**
7	15	75	20	79	109	10	**30**
8	15	80	10	91	121	10	**30**
9	10	35	20	91	121	10	**30**
10	10	40	30	119	149	10	**30**
11	8	40	40	59	89	10	**30**
12	8	45	20	64	94	10	**30**
13	5	35	10	142	172	10	**30**
14	5	45	10	35	65	10	**30**
15	2	**40**	20	58	88	10	**30**
16	0	**40**	20	72	102	10	**30**
17	0	45	20	149	179	10	**30**
18	44	5	20	87	117	10	**30**
19	42	10	40	72	102	10	**30**
20	42	15	10	122	152	10	**30**

Fuente: Autores

La distribución de clientes en el conjunto de instancias RC se encuentra definido mediante las mismas coordenadas geográficas. Se observa en la Figura 16, la tendencia de algunos puntos cercanos a formar clústeres y también algunos puntos que se encuentran dispersos en el espacio, característica fundamental de este tipo de instancias.

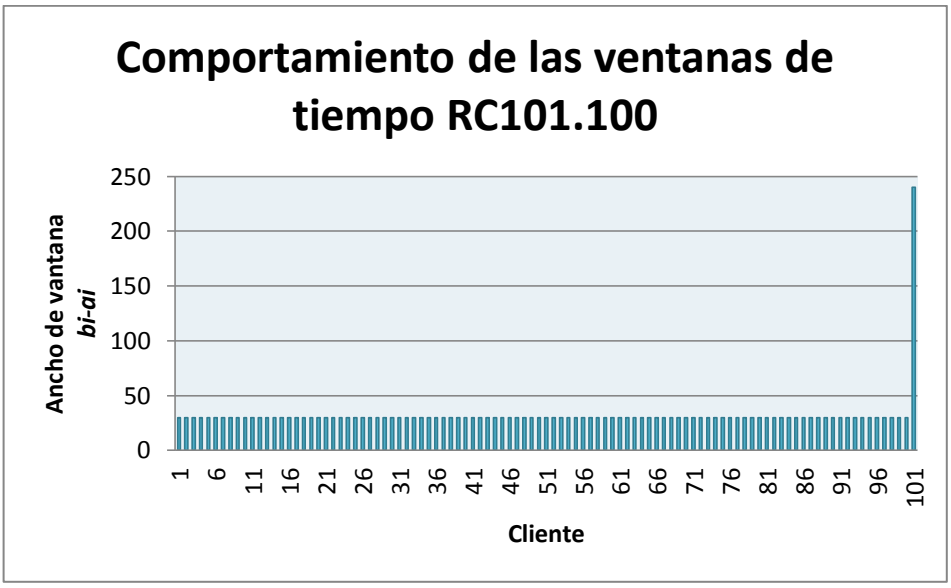
Figura 16. Distribución geográfica de clientes RC101.100



Fuente: Autores

Los intervalos de ventana para los clientes de la instancia RC101.100 son iguales con 30 unidades de tiempo, como se observa en la Figura 17, lo cual no es un periodo de tiempo cómodo para la programación de las rutas, esta restricción implica la formación de rutas más cortas y un poco mas personalizadas o individualistas.

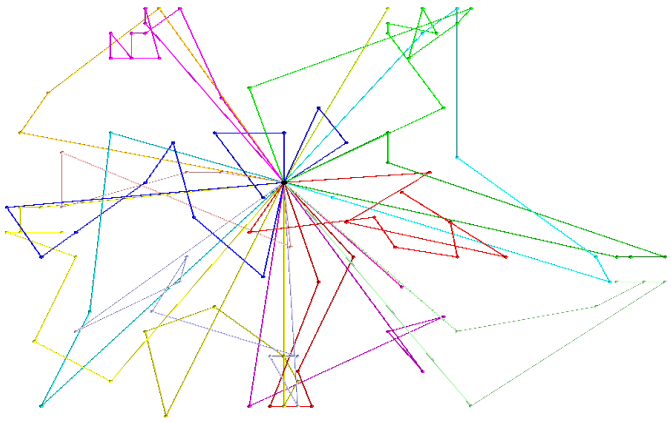
Figura 17. Comportamiento de las ventanas de tiempo RC101.100



Fuente: Autores

La Figura 18 es una solución para la instancia RC101.100, cada vértice es un cliente y cada color de línea representa una ruta que inicia y termina en el punto central o depósito.

Figura 18. Solución factible Instancia RC101.100, usando algoritmo del vecino más cercano.



Fuente: Autores.

Los intervalos de ventanas de tiempo de las instancias de semiclústeres con largo horizonte de programación RC2 son mucho más largos que los de tipo RC1 lo que influye en la programación aleatoria y clusterizada de las rutas. Tabla 6 muestra los veinte primeros datos de la instancia RC205.100.

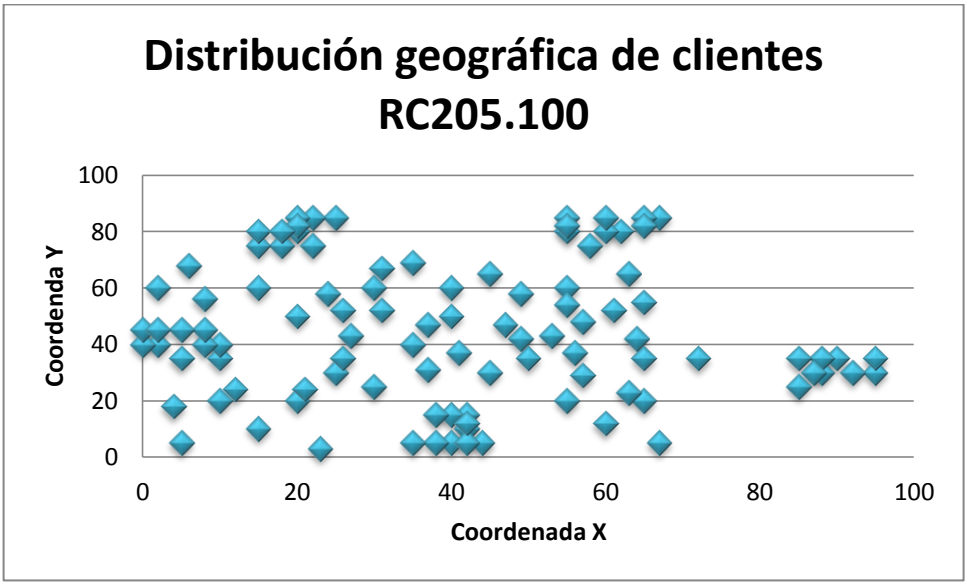
Tabla 6. Veinte primeros datos instancia RC205.100.

Cliente	X	Y	d	ai	bi	si	bi-ai
1	25	85	20	431	911	10	**480**
2	22	75	30	30	510	10	**480**
3	22	85	10	291	771	10	**480**
4	20	80	40	674	734	10	**60**
5	20	85	20	40	520	10	**480**
6	18	75	20	393	502	10	109
7	15	75	20	120	600	10	480
8	15	80	10	397	457	10	**60**
9	10	35	20	401	461	10	**60**
10	10	40	30	535	622	10	87
11	8	40	40	225	285	10	**60**
12	8	45	20	43	523	10	**480**
13	5	35	10	683	743	10	**60**
14	5	45	10	35	366	10	331
15	2	40	20	204	264	10	**60**
16	0	40	20	204	425	10	221
17	0	45	20	698	827	10	129
18	44	5	20	306	483	10	177
19	42	10	40	199	428	10	229
20	42	15	10	494	699	10	205

Fuente: Autores

En la Figura 19 puede verse la posición de cada uno de los clientes de este tipo de instancias. Es notoria la formación de algunos pocos clústeres y algunos otros clientes dispersos.

Figura 19. Distribución geográfica de clientes RC205.100

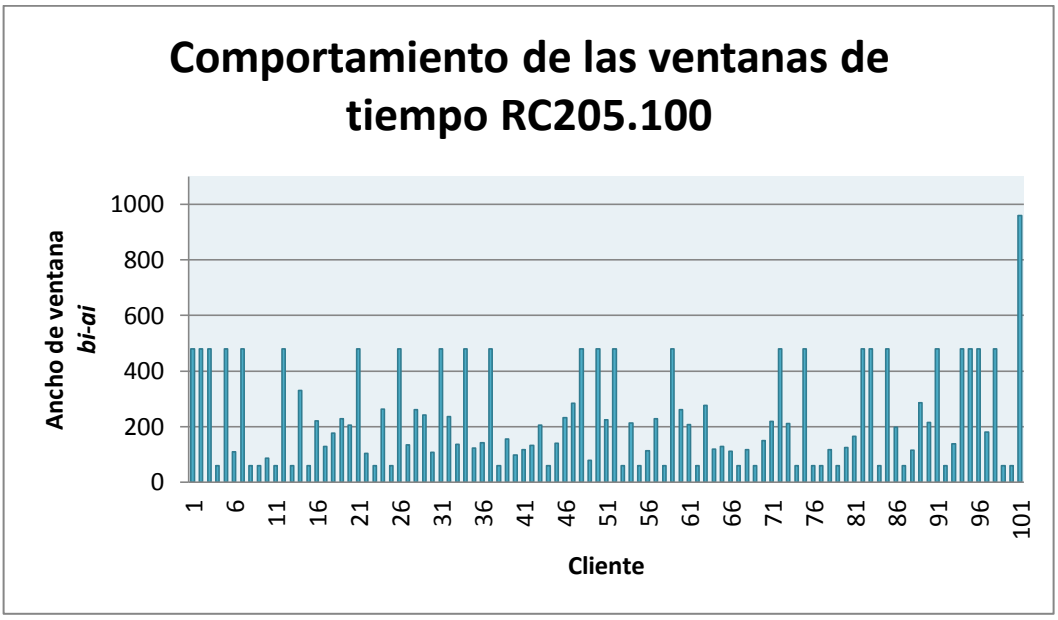


Fuente: Autores

Con respecto a las ventanas de tiempo, algunas de ellas varían desde las 60 unidades hasta las 480 unidades, en otras instancias de este mismo grupo (RC2) la variación es mucho mayor, como en la instancia RC207.100.

Algunas ventanas, como las de los clientes 1, 2, 3 ó 5, son del mismo tamaño y otras tienen valores muy cercanos generando líneas de tendencia en los datos, como se aprecia en la Figura 20.

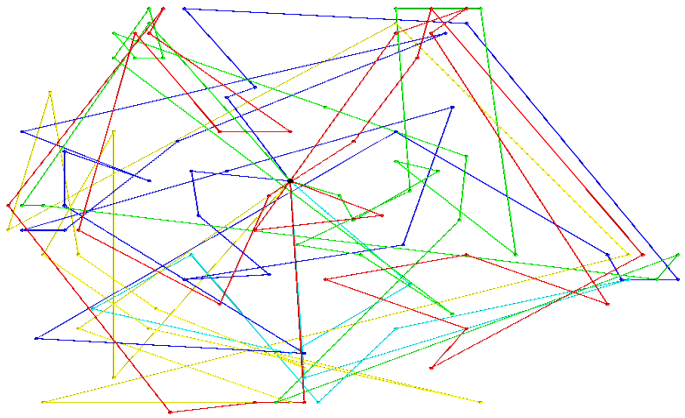
Figura 20. Comportamiento de las ventanas de tiempo RC205.100



Fuente: Autores

En la Figura 21 se muestra la solución gráfica a la instancia RC205.100, donde se observa la naturaleza “semi-clusterizada” de las rutas. Cada vértice es un cliente y cada color de línea representa una ruta que inicia y termina en el punto central o depósito.

Figura 21. Distribución en semiclústeres. Solución factible Instancia RC205.100, usando algoritmo del vecino más cercano.



Fuente: Autores.

3.4 VARIANTES DEL VRPTW

El problema de ruteo de vehículos ha sido estudiado por múltiples autores y existe un gran número de variantes del mismo. A continuación de manera breve se exponen algunos de ellos.

Si se eliminan las restricciones relacionadas con la capacidad de la flota de vehículo, se presenta el Problema de Múltiples Agentes con Ventanas de Tiempo (*m-TSPTW*). Si se cambian las ventanas de tiempo, por un tiempo fijo de servicio, el problema se convertiría en un Problema de Scheduling con Capacidad o (*FSPCC*).

En el caso de eliminar las ventanas de tiempo, se presenta el problema básico de Ruteo (VRP). Si existen múltiples depósitos o centros de distribución, se tiene el Problema de Ruteo con Ventanas de Tiempo y Múltiples Depósitos (*MDVRPTW*).

Los elementos del *VRPTW* pueden combinarse con otras restricciones para ampliar el horizonte del modelo. Si se tratan restricciones de demandas aleatorias, el problema es *SVRPTW* o con demandas estocásticas; si los vehículos tienen configuraciones heterogéneas, se tiene el Problema *MCVRPTW*, o con múltiples compartimentos internos y si se tienen múltiples ventanas temporales, se tiene el problema *VRPMTW*.

3.5 MÉTODOS DE SOLUCIÓN DEL VRPTW

El problema de ruteo con ventanas de tiempo, ha sido objeto de estudios exhaustivos en los últimos años debido a que las variables de tiempo, geográficas y de capacidad, modelan de manera cercana la realidad del proceso de distribución física, lo que lo convierte en un problema de carácter complejo en el conjunto de modelos de Ruteo.

Entre las herramientas estudiadas para la solución del VRPTW, se encuentran métodos exactos, y métodos aproximados con heurísticas y metaheurísticas.

3.5.1 Métodos Exactos para el VRPTW

Los métodos exactos usados para resolver el problema de ruteo de vehículos con ventanas de tiempo, han sido usados en problemas básicos de ruteo como el VRP [15] o el CVRP. Entre las investigaciones más destacadas y referenciadas de uso de métodos exactos para la solución del VRPTW se encuentran [16]:

Kolen (1987) [17] utiliza una estrategia Branch and Bound para resolver el VRPTW, basada en la unión de la programación dinámica y relajación de Lagrange con división de variables. El autor analiza varios métodos para la solución del problema, siendo la generación de columnas el método que aparece más robusto, numéricamente y la mejor elección en la práctica.

Fisher (1997) [18], presenta dos algoritmos de optimización para resolver el VRPTW basados en relajaciones de las restricciones. La primera es la relajación de Lagrange, que introduce división de variables para partir el problema original en dos sub-problemas, uno de asignación y otro de ruteo; la segunda es una técnica de k-árboles. El autor indica que se obtienen mejores resultados cuando se corren instancias agrupadas o clústeres.

Bramel (1997) [19] resuelve el VRPTW con una estrategia de relajación lineal de variables mediante la generación de columnas, para luego usar el método Branch and Bound y encontrar una solución entera del problema. El autor indica que la relajación lineal de las variables es muy eficiente a la hora de resolver los sub-problemas.

Rich (1999) [20] analiza el comportamiento de generación de columnas en el VRPTW, combinando métodos de programación dinámica. El autor presenta soluciones con varias estrategias para generar las columnas. Igualmente presenta resultados con aplicaciones programadas utilizando 4 u 8 computadores en línea.

Ribeiro (1994) [21], crea una aplicación para el VRPTW con múltiples depósitos usando el método de generación de columnas. Fisher (1994) [22], prueba que el problema VRPTW puede ser modelado como un problema de k-árboles de costos mínimo y sujeto a restricciones adicionales. Mingozzi (1997) [23], presenta un algoritmo exacto que se basa en la programación dinámica, para hallar soluciones al VRPTW empleando funciones de acotamiento para reducir el espacio de búsqueda.

Fischetti (1999) [24] resuelve el TSP simétrico en el caso en que los clientes aparecen divididos en clústeres, y el vendedor ha de visitar al menos un cliente de cada clúster. El autor usa métodos de branch and cut para dividir el problema, en sub-problemas más pequeños
En general, se puede destacar que las investigaciones previas donde se usan métodos exactos, arrojan resultados de buena calidad para problemas de tamaño reducido. Los autores consultados mencionan que la relación que existe en el tamaño del problema y el tiempo de ejecución del método es directamente proporcional a la calidad de la solución.

Los métodos exactos no son flexibles a la hora de adaptar restricciones asociadas a problemas del mundo real, y es muy poco probable que se llegue a una solución óptima debido a la complejidad del sistema.

3.5.2 Métodos Metaheurísticos para el VRPTW

Existen herramientas metaheurísticas para la solución del problema VRPTW. Algunas de ellas son: Recocido Simulado, Búsqueda Tabú, y Algoritmos Evolutivos tales como el Algoritmo Genético y Algoritmo de Hormigas. Todos los métodos exploran un gran espacio de soluciones con el fin de encontrar una solución factible posiblemente cercana al óptimo. [25]

Algunas Investigaciones recientes del VRPTW usando Metaheurísticas son:

Búsqueda Tabú de Taillard [26]. El autor describe una metaheurística que toma el problema VRPTW penalizando fuertemente cualquier demora en la distribución. La metaheurística usa el concepto de descomposición y reconstrucción propuesto para el VRP.

La memoria adaptativa, según Taillard, es un conjunto de rutas tomadas de buenas soluciones visitadas durante la búsqueda. Esta memoria se llena parcialmente con rutas producidas por inserción aleatoria basada en las heurísticas de Solomon (1987). Cada iteración de esta metaheurística, construye una ruta, y a través de un proceso aleatorio, se van mejorando las rutas hasta que nuevas soluciones factibles sean encontradas en el espacio de soluciones disponibles [27].

Keivan Ghoseiria y Ghannadpoura [28], presentan un nuevo modelo para un problema VRPTW multiobjetivo, usando programación de objetivos y algoritmo genético, en el cual el investigador mismo especifica el nivel de optimización que se quiere alcanzar y el nivel de cumplimiento de los objetivos del modelo.

Balseiro, Loiseau y J.Ramonet [29] desarrollaron un algoritmo de Colonia de Hormigas para el problema VRPTW basado en el comportamiento de agentes viajeros y la coordinación entre ellos.

Zhenggang y Linning [30] (2004), proponen un procedimiento mejorado de optimización por Colonia de Hormigas para resolver el problema VRPTW, en el cual el periodo de planeación es extendido para que cada cliente pueda ser servido en una ventana de tiempo definida. Una matriz multidimensión es usada para acumular la información de distintas corridas de las heurísticas. Dos operaciones de cruces se introducen para mejorar la calidad del algoritmo.

Deng y Mao [31] presentan un modelo matemático, en el que se consideran variables de tiempo y costos fijos del vehículo. La metaheurística Recocido Simulado es usada para encontrar soluciones.

G.B. Alvarengaa, G.R. Mateusb, G [32], proponen un Algoritmo Genético robusto para la solución de VRPTW usando la longitud de los recorridos como principal función objetivo. El problema lo resuelven a través de una formulación de partición de rutas. Se corrieron pruebas usando datos de casos reales y resultados previos publicados con heurísticas y métodos exactos. Este estudio muestra mejoras en el rendimiento computacional en términos de mínima distancia respecto a otras investigaciones hechas previamente.
Bouthilliera y Crainicb [33] desarrollan un método de búsqueda paralela para el VRPTW basado en una estrategia en la que se intercambia información de las distintas corridas hechas sobre las mejores soluciones identificadas.

Müller [34], formula el modelo como un problema de minimización bi-criterio usando el concepto de optimización de Pareto. El experimento muestra que la asignación de sanciones, da un ahorro significativo de los costos totales.

Robert A. Russell y Wen-Chyuan Chiang [35] estudian una metaheurística reactiva basada en La Búsqueda Tabú para mejorar la solución del problema VRPTW, comparando la calidad de los resultados de investigaciones similares.

En general, el comportamiento de las soluciones obtenidas con metaheurísticas se considera bueno. En la mayoría de investigaciones consultadas, se usan algoritmos de construcción de rutas para el conjunto de soluciones iniciales. Con métodos híbridos y combinaciones de algoritmos, se ha llegado a la solución óptima de la mayoría de instancias propuestas en la literatura.

3.5.3 Métodos Heurísticos para el VRPTW

Distintas familias de heurísticas se han propuesto para la solución del problema VRPTW. El objetivo de ellas, es brindar una solución factible en un tiempo computacional razonable.

Solomon (1986) [36] desarrolló una serie de heurísticas llamadas “Heurísticas de construcción de rutas”, donde se construyen nodos y arcos, según un criterio de minimización del costo, hasta que una solución factible haya sido creada. Allí, los clientes se programan en un “tour gigante”, y luego, se dividen en rutas más pequeñas.

Solomon describe varias heurísticas de este tipo para la solución del problema con ventanas de tiempo. Una de ellas es basada en el Algoritmo de ahorros de Clarke and Wright (1964). La heurística fue originalmente desarrollada para el problema clásico de ruteo de vehículos. Inicia con una solución en la cual cada cliente es atendido individualmente en una ruta separada. Combinando dos rutas de clientes se obtendrá un ahorro del costo, hasta que se llegue a una combinación de rutas factible a través de un proceso iterativo.

Otra de las heurísticas desarrolladas por Solomon (1987) [37] para el problema VRPTW es la orientada a buscar “el vecino más próximo”. Se inicia cada ruta buscando un cliente que no haya sido considerado en alguna ruta y que se encuentre cerca al depósito principal. En cada iteración subsecuente, la heurística

busca al cliente más cercano al último cliente añadido a la ruta, y lo adiciona al final de la ruta recién construida. Una nueva ruta se inicia cada vez que el proceso de búsqueda falla al encontrar un lugar de inserción factible para algún cliente.

La heurística más popular de Solomon es la llamada I1. La heurística inicia una ruta con un cliente “semilla”. Los demás clientes serán añadidos a la ruta, respecto al horizonte de programación o la restricción de capacidad del vehículo.

Dullaert (2000) [38] argumenta que el criterio de inserción de Solomon (1987) con su heurística I1, subestima el tiempo adicional requerido para insertar un nuevo cliente entre el depósito y el primer cliente agregado en la ruta construida. Según el autor, esto podría causar la selección de criterios de inserción sub-óptimas para los clientes que aun no han sido agregados a las rutas. Por esta razón el autor introduce nuevas restricciones de tiempo para resolver el problema y concluye que el nuevo criterio de inserción, ofrece un ahorro significativo de costos de más del 50%. Sin embargo, el ahorro de costos decrece en la medida que el número de clientes por ruta se incrementa.

Potvin y Rousseau (1993) [39], introducen una versión paralela a la heurística de inserción I1 de Solomon, donde todo el grupo de rutas se inician al mismo tiempo.

Foisy y Potvin (1993) [40] implementaron la misma versión de Potvin and Rousseau (1993), pero usando hardware paralelos para computar la misma instancia. Este paralelismo se despliega en el cálculo del costo al momento de insertar un cliente en la ruta. Los autores concluyen que la reducción total en el tiempo de cómputo es lineal al número de procesadores.

Ioannou (2001) [41] usa el desarrollo de inserción secuencial de Solomon para resolver instancias relacionadas con la industria de alimentos. La propuesta está basada en un nuevo criterio de selección de clientes donde se supone que la

inserción de un cliente nuevo debería reducir el impacto de inclusión de clientes en rutas ya construidas, en clientes no seleccionados y en las ventanas de tiempo correspondientes.

Balakrishnan (1993) [42] estudia tres heurísticas para El VRPTW. Se basan en los conceptos de “vecinos más cercanos” y “ahorro de rutas” de Clark-Wright, pero difiere solo por la manera en que selecciona el primer cliente en una ruta y en el criterio usado para agregar clientes subsecuentes en la misma ruta.

Bramel y Simchi-Levi (1996) [43] proponen una heurística de naturaleza asintótica, basada en la idea de resolver el problema de ventanas de tiempo con capacidad. El objetivo es seleccionar un subgrupo de posibles locaciones para ubicar un vehículo a cada sitio y asignar clientes a cada vehículo en la medida que se vayan localizando. El autor usa una técnica basada en “Relajación Lagrangeana” para asignar clientes. El autor concluye que esta heurística provee una mejor solución para 25 de los 56 problemas resueltos a través de las heurísticas de Solomon, usando un tiempo de cómputo razonable.

Por otro lado, existen las heurísticas de búsqueda local basadas en el concepto de mejoramiento iterativo, explorando soluciones vecinas.

Russell (1977) [44] desarrolló las primeras investigaciones usando Heurísticas de búsqueda local, resolviendo una instancia de 163 clientes en menos de 90 segundos, usando un equipo IBM 370/168.
Se han implementado herramientas eficientes para incrementar la velocidad de búsqueda de soluciones no factibles y evaluación de distintas funciones objetivo, reportados por Savlesberg (1986) [45], Desrosiers y Solomon (1988) y Baker and Schaffer (1988). La herramienta usada involucra técnicas de reproceso, mecanismos de actualización de soluciones y estrategias de búsqueda lexicográficas.

Baker y Schaffer (1986) [46] reportan un procedimiento de mejoramiento de rutas, en el cual se usan heurísticas para generar soluciones iniciales, y luego otras para evaluar la función objetivo teniendo en cuenta variables como la distancia, el incremento en el tiempo de llegada y el tiempo de espera.

Van Landeghem (1988) [47] presenta una heurística bi-criterio basada en la heurística de ahorros de Clarke-Wright (1964). El autor propone combinar la heurística de ahorros en términos del tiempo de transporte con “pérdida de flexibilidad”. Esta flexibilidad es definida como la diferencia entre la amplitud de la ventana de tiempo del cliente y la amplitud de la ventana de tiempo de la ruta después de combinarlas.

Koskosidis (1992) [48] reporta una variante del problema VRPTW con ventanas de tiempo flexibles. El problema es descompuesto en dos problemas más pequeños: Un problema donde se agrupan los clientes según la capacidad disponible y una serie de problemas de agente viajero con ventanas de tiempo. El objetivo es buscar clientes semilla con menores costos de agrupación y hacer cambios locales entre todos los pares de clientes pertenecientes a cada grupo.

Potvin y Rousseau (1995) [49] comparan diferentes criterios de cambios en las rutas. La idea básica es combinar dos rutas, de tal manera que el último cliente de una ruta dada es introducido después de los primeros clientes de otra ruta, con el fin de preservar la orientación de las rutas que se van formando.

Glover (1991 y 1992) [50] basa su estudio de heurísticas de búsqueda local en el concepto de generación de secuencias de movimientos compuestos, saltando de una solución a otra; los pasos que generan cambios en los clientes seleccionados, hacen que otros clientes sean “expulsados” de su actual posición. El objetivo es

“hacer espacio” para que un cliente sea removido de una ruta, removiendo otro cliente de la misma ruta, hasta que la función objetivo establecida mejore.

Thompson y Psaraftis (1993) [51] proponen un método basado en el concepto de transferencias cíclicas de demandas de una ruta a otra.

Antes y Derigs (1995) [52] proponen una heurística paralela donde se construyen y mejoran varias rutas simultáneamente. Esta aproximación paralela se basa en el concepto de “Negociación” entre clientes. Primero, Cada cliente no asignado a alguna ruta, pide un requerimiento de costo a cada tour y envía una propuesta de un tour que haya ofrecido el menor costo. Luego, cada tour selecciona la propuesta más eficiente o más conveniente.

Russell (1995) [53] desarrolla un procedimiento simultáneo a la construcción de rutas, basado en procedimientos de mejora de rutas globales. En el algoritmo, un punto semilla representa un cliente ficticio, que es seleccionado usando una ruta en la cual un vehículo “crea” un sector y decide la distancia de los puntos semilla y de las bodegas en cada sector creado.
Hamacher y Moll [54], describen una heurística aplicada a un caso real del problema VRPTW, en el contexto de distribución de víveres a restaurantes. El algoritmo consta de dos partes; en la primera parte, los clientes son divididos en regiones usando el criterio del Mínimo Árbol de Expansión (MST por sus siglas en inglés); en la segunda parte, los clientes que cumplan con este criterio, son insertados a las rutas con un sencillo algoritmo de mejoramiento local, que corta las rutas en piezas pequeñas y las inserta en una ubicación factible dentro de la misma ruta.

Shaw (1997) [55] describe un procedimiento de búsqueda en la vecindad basado en reprogramación de clientes visitados usando técnicas de Programación de Restricciones. Los clientes seleccionados son removidos de las rutas

programadas y luego son reinsertados en la ruta del costo óptimo. Este problema solo puede ser resuelto cuando los clientes están ubicados uno de otros, en un área geográfica pequeña.

Caseau y Laburthe (1999) [56] describe una heurística específicamente diseñada para grandes problemas de ruteo parecida a la usada por Shaw (1997) en su modelo de construcción de rutas basado en el criterio MST.

Cordone y Wolfer-Calvo (2001) [57] proponen una heurística determinística, basada en el clásico procedimiento de intercambio de rutas combinado con procedimientos de reducción de rutas. La característica especial del algoritmo es que alterna la minimización total de la distancia, con la duración total de la ruta, para escapar de soluciones que estén en los mínimos locales. Este algoritmo construye una serie de soluciones iniciales usando el criterio de inserción de Solomon. [58]

Recientemente se han implementado otras herramientas heurísticas para la solución del problema VRPTW en la construcción de rutas que usan procedimientos multi-nivel, como la desarrollada por Bräysy (2001) [59], en la que varias soluciones iniciales son creadas con heurísticas de construcción, haciendo combinaciones de parámetros diferentes. Algunos parámetros usados son entre otros: Reordenamiento de rutas, intercambio de clientes y expulsión de clientes a otras rutas.

4 MÉTODOS HEURÍSTICOS PARA LA SOLUCIÓN DEL VRPTW

Como se mencionó en la sección anterior, existen diversos métodos heurísticos para abordar el problema con ventanas de tiempo [60]. Se describirán apartes generales de las heurísticas de mejora de rutas y se hará especial énfasis en la

caracterización de las heurísticas de construcción de rutas, objeto de estudio del presente proyecto.

Para la selección de los métodos que se estudian en el proyecto se definió una lista de criterios comunes que califican al conjunto de heurísticas investigadas. Los criterios fueron planteados por los autores, basados en el conocimiento adquirido en la etapa de revisión bibliográfica, luego de revisar la literatura científica especializada en el VRPTW.

Teniendo en cuenta el planteamiento y la formulación matemática del problema de ruteo de vehículos con ventanas de tiempo, se considera el criterio de minimización de costo y por consiguiente las heurísticas seleccionadas tienen como característica primordial su disminución.

Los aspectos espaciales (Distancia) y temporales (Ventanas de tiempo) pueden ir en sentidos contrarios a la hora de seleccionar clientes en una ruta, es fundamental que el algoritmo tenga en cuenta un enfoque no sólo espacial, sino también temporal para evitar la formación de grandes tiempos ociosos por las esperas generadas.

Es muy importante que exista un criterio que evalúe la posibilidad de convertir a lenguaje de programación el procedimiento de la heurística, en ese sentido algunas pueden operarse bajo estructuras o programaciones informáticas más sencillas y de fácil entendimiento que otras. La consecución y el acceso de la información de cada heurística se consideran punto fundamental en la apropiación del conocimiento y por tanto de la implementación de técnicas heurísticas.

Los criterios descritos anteriormente se citan a continuación en la Tabla 7. Para medir los atributos descritos anteriormente se utiliza una escala entre cero (0) y cinco (5); definiendo cero (0) como la más baja calificación que se obtiene cuando

el criterio no se cumple y cinco (5) cuando se obtiene la mejor calificación y se cumple totalmente.

Todas las heurísticas estudiadas para el presente proyecto corresponden al conjunto de algoritmos de construcción de rutas. Con base en los resultados de los artículos referenciados en la respectiva sección del capítulo 3, se ponderan los algoritmos en la Tabla 7.

Tabla 7. Selección de heurísticas para la solución del VRPTW

CRITERIOS	Algoritmo del Vecino más Cercano	Algoritmo de Barrido	Algoritmos de Inserción de Solomon	Algoritmo de Ahorros de Clarke y Wright	Algoritmo de Pétalos	Algoritmo de Inserción con Relajación Lagrangeana
Acceso y consecución de la información	3	3	4	4	3	2
Disminución del costo	4	3	4	5	3	3
Presencia de enfoque temporal	4	2	5	2	0	1
Disminución tiempos de espera	4	2	5	3	0	1
Fácil implementación en el lenguaje de programación	4	3	2	2	2	2
Totales	**19**	**13**	**20**	**16**	**8**	**9**

Fuente: Autores

Luego de obtener la puntuación se destacan las heurísticas en el orden mostrado a continuación:

1. Algoritmos de Inserción de Solomon
2. Algoritmo del Vecino más Cercano
3. Algoritmo de Ahorros de Clarke y Wright
4. Algoritmo de Barrido
5. Algoritmo de Inserción con relajación Lagrangeana
6. Algoritmo de Pétalos

Dado que el primer algoritmo de la lista es el de Inserción de Solomon, se decide incluirlo en sus tres diferentes variantes: I1, I2 e I3 y de esta manera se aceptan dos algoritmos más, que en su orden son: algoritmo del Vecino más Cercano y el algoritmo de Ahorros de Clarke y Wright. Se descarta el algoritmo de barrido, el de relajación lagrangeana y el de pétalos, por sus bajas puntuaciones con respecto al criterio de *presencia del enfoque temporal*, aspecto fundamental en problemas de ruteo de vehículos con ventanas de tiempo. A continuación se rescriben las heurísticas seleccionadas.

4.1 HEURÍSTICAS DE CONSTRUCCIÓN DE RUTAS

Las heurísticas de construcción de rutas se dividen a su vez en secuenciales (una solo ruta a la vez) (Solomon, 1987) y en paralelas, donde se inician simultáneamente varias rutas (Russell, 1995).

4.1.1 Algoritmo De Ahorros De Clarke & Wright

4.1.1.1 Generalidades

El algoritmo de ahorros fue publicado en 1964 por G. Clarke and J.W. Wright [61] quienes extendieron el enfoque propuesto por Dantzig and Ramser [62], para la

solución del Problema de Ruteo de Vehículos teniendo en cuenta la distancia entre los nodos.

Clarke and Wright definen el concepto de *ahorro,* al calcular la reducción en distancia que se genera cuando se unen dos clientes en una ruta, en lugar de servir a cada uno de ellos en rutas diferentes.

4.1.1.2 Consideraciones teóricas para la definición de "Ahorro"

El método inicia con rutas factibles, asociadas a cada uno de los vehículos de una flota de transporte. Cada ruta vehicular parte del depósito, recorre uno o varios clientes y regresa nuevamente a él. El nodo depósito se representa mediante la etiqueta P_0 y los clientes por visitar se representan por P_y y P_z.

En la Fuente: Autores se muestran dos rutas factibles: $P_0, P_{y-1}, P_y, P_{y+1}, P_0$ y $P_0, P_{z-1}, P_z, P_{z+1}, P_0$. Se observa que el nodo etiquetado con P_y tiene como antecesor al nodo etiquetado con P_{y-1} y como sucesor al nodo P_{y+1} (De igual manera para el nodo P_z).

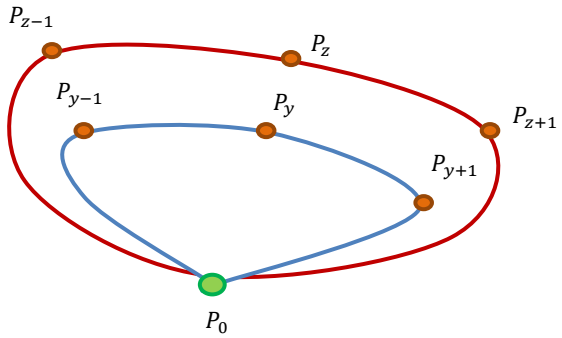
La siguiente etapa del procedimiento analiza el efecto de alterar las rutas originales, se seleccionan dos nodos P_y y P_z que pertenecen a diferentes rutas y se unen mediante una arista con el fin de formar una nueva ruta. Por consiguiente, el nuevo arco entre ellos hace necesario eliminar el enlace previo que existía con otro cliente asociado. Los clientes que estaban enlazados a P_y y a P_z conforman nuevas rutas que producen variaciones a la rutas originales formando diferentes escenarios.

En la

Figura 23 y Figura 24 se presentan dos posibles soluciones obtenidas a partir de la unión de los clientes P_y y P_z. Para el primer caso, la ruta que se muestra en rojo $P_0, P_{z-1}, P_z, P_y, P_{y+1}, P_0$ se conforma siguiendo un orden de precedencia entre nodos; el arco que existía entre P_y - P_{y-1} fue eliminado, así como el que existía entre P_z - P_{z+1}, ambos clientes separados de la ruta son servidos individualmente. Lo mismo ocurre en el escenario 2 (figura 24), con la ruta $P_0, P_{y-1}, P_y, P_z, P_{z+1}, P_0$, en este caso los arcos eliminados son P_y - P_{y+1} y P_z - P_{z-1}.

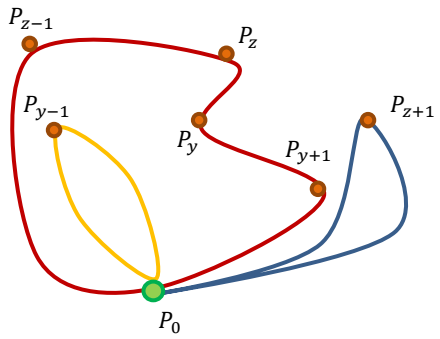
En la Figura 25 y Figura 26, el orden de precedencia no se cumple es su estricto sentido, sin embargo se cumple que tanto el nodo P_y como el nodo P_z se encuentran enlazados a algunos de sus nodos antecesores o sucesores, garantizándose una relación de continuidad en la ruta. En el escenario de la Figura 25, los arcos P_y- P_{y-1} y P_z - P_{z-1} que hacían parte de la ruta original, son eliminados, mientras que en el último escenario son eliminados los arcos P_{y+1} - P_y y P_z - P_{z+1}, haciendo necesario crear nuevas rutas para servir a los respectivos clientes. Dado el orden de procedencia y la relación de continuidad en las rutas, sólo existen cuatro posibles escenarios, cada uno tiene asociado un ahorro, los cálculos se presentan a continuación.

Figura 22. Rutas iniciales



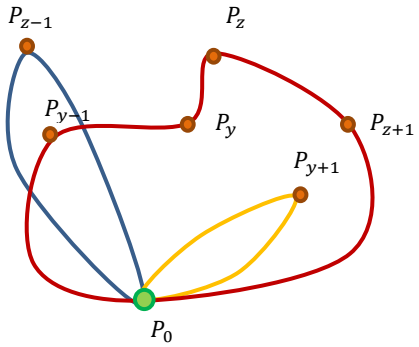
Fuente: Autores

Figura 23. Primer escenario



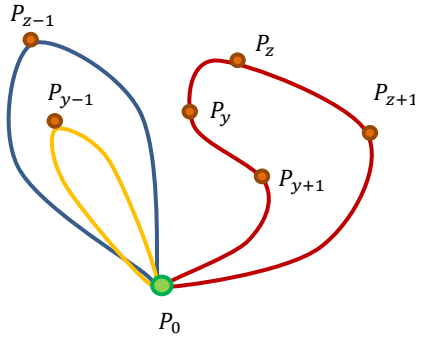
Fuente: Autores

Figura 24. Segundo escenario



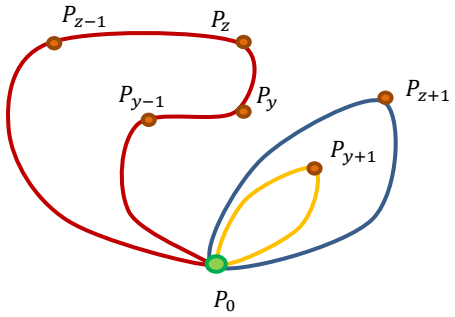
Fuente: Autores

Figura 25. Tercer escenario



Fuente: Autores

Figura 26. Cuarto escenario



Fuente: Autores

Escenario Original

$$\boldsymbol{Costo} = \boldsymbol{d_{0,y-1}} + \boldsymbol{d_{y-1,y}} + \boldsymbol{d_{y,y+1}} + + \boldsymbol{d_{0,z-1}} + \boldsymbol{d_{z-1,z}} + \boldsymbol{d_{z,z+1}} + \boldsymbol{d_{z+1,0}}$$

Escenario 1

$$\boldsymbol{Costo} = d_{0,z-1} + d_{z-1,z} + d_{y,z} + d_{y,y+1} + d_{y+1,0} + d_{0,y-1} + d_{y-1,0} + d_{0,z+1} + d_{z+1,0}$$

Ahorros Escenario 1:

$$\boldsymbol{AE1} = d_{0,y-1} + d_{y-1,y} + d_{y,y+1} + + d_{0,z-1} + d_{z-1,z} + d_{z,z+1} + d_{z+1,0} - (d_{0,z-1} + d_{z-1,z} + d_{y,z} + d_{y,y+1} + d_{y+1,0} + d_{0,y-1} + d_{y-1,0} + d_{0,z+1} + d_{z+1,0})$$

$$\boldsymbol{AE1} = d_{y-1,y} - d_{y-1,0} + d_{z,z+1} - d_{0,z+1} - d_{y,z}$$

Escenario 2

$$\boldsymbol{Costo} = d_{0,y-1} + d_{y-1,y} + d_{y,z} + d_{z,z+1} + d_{z+1,0} + d_{0,y+1} + d_{y+1,0} + d_{0,z-1} + d_{z-1,0}$$

Ahorros Escenario 2:

$$\boldsymbol{AE2} = d_{0,y-1} + d_{y-1,y} + d_{y,y+1} + d_{0,z-1} + d_{z-1,z} + d_{z,z+1} + d_{z+1,0} - (d_{0,y-1} + d_{y-1,y} + d_{y,z} + d_{z,z+1} + d_{z+1,0} + d_{0,y+1} + d_{y+1,0} + d_{0,z-1} + d_{z-1,0})$$

$$\boldsymbol{AE2} = d_{y,y+1} - d_{0,y+1} + d_{z,z-1} - d_{0,z-1} - d_{y,z}$$

Escenario 3

$$\boldsymbol{Costo} = d_{y+1,0} + d_{y,y+1} + d_{y,z} + d_{z,z+1} + d_{z+1,0} + d_{0,y-1} + d_{y-1,0} + d_{0,z-1} + d_{z-1,0}$$

Ahorros Escenario 3:

$$\boldsymbol{AE3} = d_{0,y-1} + d_{y-1,y} + d_{y,y+1} + d_{y+1,0} + d_{0,z-1} + d_{z-1,z} + d_{z,z+1} + d_{z+1,0} - (d_{y+1,0} + d_{y,y+1} + d_{y,z} + d_{z,z+1} + d_{z+1,0} + d_{0,y-1} + d_{y-1,0} + d_{0,z-1} + d_{z-1,0})$$

$$\boldsymbol{AE3} = d_{y-1,y} - d_{y-1,0} + d_{z-1,z} - d_{z-1,0} - d_{y,z}$$

Escenario 4

$$\boldsymbol{Costo} = d_{0,y-1} + d_{y-1,y} + d_{y,z} + d_{z-1,z} + d_{0,z-1} + d_{0,y+1} + d_{y+1,0} + d_{0,z+1} + d_{z+1,0}$$

Ahorros Escenario 4:

$$\boldsymbol{AE4} = d_{0,y-1} + d_{y-1,y} + d_{y,y+1} + d_{0,y+1} + d_{0,z-1} + d_{z-1,z} + d_{z,z+1} + d_{z+1,0} - (d_{0,y-1} + d_{y-1,y} + d_{y,z} + d_{z-1,z} + d_{0,z-1} + d_{0,y+1} + d_{y+1,0} + d_{0,z+1} + d_{z+1,0})$$

$$\boldsymbol{AE4} = d_{y,y+1} - d_{y+1,0} + d_{z,z+1} - d_{0,z+1} - d_{y,z}$$

Los ahorros se calculan para cada par de clientes y se selecciona el que genera un ahorro máximo que cumpla con las restricciones de capacidad y disponibilidad del vehículo. Puede considerarse la asignación de un precio sombra (Dada la similitud del concepto en Programación Lineal) al costo que representa servir a un cliente P_y . Un precio sombra permite tener en cuenta las variaciones que pueden originarse con la ruptura de un arco que está asociado a ese cliente. Para el cliente P_y los costos sombra: son $d_{y-1,y} - d_{0,y-1}$ y $d_{y,y+1} - d_{0,y+1}$.

Cuando un enlace sirve, el costo sombra apropiado se reduce por el valor de la celda que causó la ruptura. Dado que el valor del arco es el máximo, si se elimina ese enlace y el cliente se une a otros dos clientes (que no sea P_0), el valor del arco se convierte en negativo y el nodo asociado no vuelve a ser considerado para un nuevo enlace.

Siendo así, los únicos enlaces que pueden ser eliminados son las de aquellos nodos que se encuentren unidos al depósito P_0. Si tales nodos se encuentran enlazados al par y-z el ahorro sería: $d_{0y} + d_{0z} - d_{yz}$.

4.1.1.3 Mejora del Algoritmo de Ahorros

Una debilidad del algoritmo de ahorros es la tendencia a generar rutas de trayectorias circulares. La cercanía o la distribución uniforme de distancias al depósito de clientes, ocasiona asignaciones que construyen una ruta periférica. Gaskell y Yellow en 1967 mejoraron el algoritmo al introducir un parámetro $\boldsymbol{\lambda}$ en la fórmula del ahorro, que regula la forma que toma la ruta.

Los valores que toma $\boldsymbol{\lambda}$ deben ser mayores o iguales a cero, entre más alto sea su valor, más énfasis se pone a la distancia entre los nodos que van a ser conectados. El ahorro ahora se halla de la siguiente manera:

$$\boldsymbol{S_{ij} = C_{io} + C_{oj} - \lambda C_{ij}} \ , \boldsymbol{\lambda} \geq 0$$

4.1.1.4 Algoritmo de Ahorros para el VRPTW

El algoritmo de ahorros para la solución del problema de ruteo de vehículos con ventanas de tiempo, además de considerar el ahorro en distancia, tiene en cuenta las ventanas temporales, para la selección de los clientes factibles que asigna a la ruta parcial.

La existencia de ventanas de tiempo obliga a tener en cuenta la orientación de la ruta. Si dos rutas parciales tienen clientes finales i y j respectivamente existe compatibilidad, si i es el primero (último) y j es el último (primero) en la nueva ruta conformada [62].

Cuando se unen dos clientes cercanos en distancia pero lejanos temporalmente, puede suceder que se incurra en tiempos de espera significativamente costosos para la ruta. Cuando un vehículo está obligado a esperar para ser atendido, se genera un costo de oportunidad que debe ser asumido, al dejar de atender a otros clientes que pueden estar disponibles.

Marius M. Solomon, quien fue el primero en abordar el Algoritmo de Ahorros para VRPTW, propone un criterio para limitar los tiempos de espera: Se fija un parámetro W que determina el tiempo máximo que se está dispuesto a esperar cuando se selecciona un cliente para asignarlo a la ruta. Siendo W_f^{new} el parámetro resultante en el cliente f del tiempo de espera, al unir en un arco a los clientes l y f; sea W el parámetro fijado para garantizar la minimización de los tiempos de espera, se compara W_f^{new} con W. Si $W_f^{new} > W$ no se usa este par de clientes en la ruta.

El algoritmo permite formar rutas parciales para todos los vehículos de la flota, que se van formando simultáneamente hasta que se atiendan todos los clientes y ningún vehículo exceda su capacidad, en este sentido se utiliza la *Versión Paralela*.

Se utiliza la *Versión Secuencial* cuando se asignan clientes a una ruta dada hasta que el vehículo no tenga capacidad disponible y secuencialmente se inicia otra ruta para atender al resto de clientes.

4.1.1.5 Procedimiento

- Calcular los ahorros de todos los pares de clientes: $\boldsymbol{S_{ij} = C_{io} + C_{oj} - \lambda C_{ij}}$
- Organizar en una lista los pares de acuerdo al ahorro que generan en orden descendente.
- Se observa desde el comienzo de la lista para seleccionar el primer par de clientes. Se asignan a la ruta, si cumplen con las condiciones de factibilidad: si la suma de las demandas asociadas no excede la capacidad del vehículo y si se respetan las ventanas de tiempo de los dos clientes, es decir si el tiempo de llegada al cliente i, no excede el tiempo más tardío en el que el cliente j permite ser servido.
- El siguiente cliente se escoge del siguiente par con ahorro más alto según la lista, que esté relacionado con el último cliente de la ruta parcial, examinado la factibilidad de la inserción (*Versión Secuencial*)

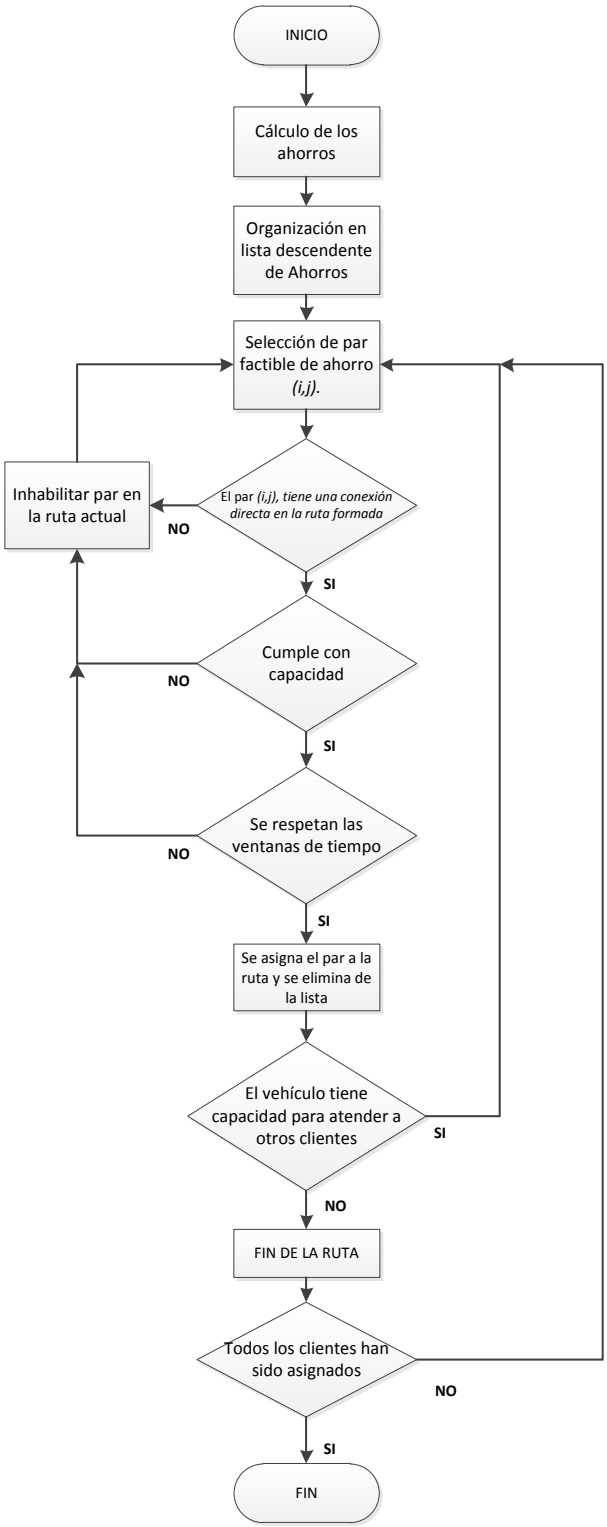
  Nota: Al seleccionar un par no se debe eliminar una conexión directa previamente establecida entre dos clientes.
- Se escoge el siguiente par con ahorro más alto de la lista y se asigna a una nueva ruta, verificando para estos nuevos clientes la factibilidad de ser asignados (*Versión Paralela*).

  Nota: Al seleccionar un par no se debe eliminar una conexión directa previamente establecida entre dos clientes.

- Los clientes asignados se van eliminando de la lista.
- Se repiten los pasos desde el número 4, hasta tanto todos los clientes estén servidos.
- Se calculan los costos de la ruta final.
-

El procedimiento del algoritmo de ahorros puede observarse gráficamente en la figura 27 mediante un diagrama de flujo.

Figura 27. Diagrama de Flujo para Algoritmo de Ahorros



Fuente: Autores

4.2 HEURÍSTICAS DE INSERCIÓN SECUENCIAL PARA EL VRPTW.

Se estudian cuatro heurísticas de inserción secuencial para resolver el VRPTW. La primera se basa en extensiones de las heurísticas del modelo clásico de ruteo de vehículos (Rosenkrantz, Stearns y Lewis (1977)), con la diferencia de que en este enfoque, se incorporan métricas que integran la dimensión espacial y la temporal.

A continuación, se define una heurística de inserción más cercana, o del vecino más cercano.
Las tres siguientes, son las heurísticas de inserción de Solomon (Solomon 1987) (I1, I2 e I3), que fueron inicialmente desarrolladas para el problema de Ruteo de Vehículos con Ventanas de Tiempo.

4.2.1 Heurística del Vecino más Cercano con enfoque temporal.

El algoritmo halla una solución basada en la cercanía de dos nodos o clientes adyacentes. En el enfoque temporal de la heurística del vecino más cercano, se usa una métrica o medida que hace un balance ponderado entre la cercanía geográfica de los clientes y el tiempo de recorrido respectivo de un nodo a otro.

En esta aproximación, un cliente cercano geográficamente no implica factibilidad en términos de tiempo, por esta razón el costo de la ruta al insertar un cliente hace un balance entre estos dos parámetros, y asigna los clientes a la ruta dando prioridad a aquellos cuyo “balance” sea menor.

Si se tiene una ruta $(0,\dots u_i\,,,0)$, se define el costo de insertar el cliente u_j a continuación de u_i en la ruta como:

$$C_{ij} = \delta_1 d_{ij} + \delta_2 T_{ij} + \delta_3 V_{ij} \ , \delta_1 + \delta_2 + \delta_3 = 0$$

$$T_{ij} = W_j - (W_i + s_i)$$

$$V_{ij} = b_j - (W_j + s_i + t_{ij})$$

Donde los parámetros δ_1, δ_2, y δ_3 son no negativos y suman 1.

- El término d_{ij}, es la distancia directa entre dos nodos y mide su cercanía geográfica (Se asume que cada unidad de distancia es equivalente a una unidad de tiempo de recorrido).

- El valor de T_{ij} indica la diferencia entre la hora de comienzo del servicio en j y la del fin del servicio en i, midiendo la cercanía de los clientes en términos temporales. Este parámetro minimiza el menor tiempo de recorrido y el menor tiempo de espera entre dos clientes.

- Por otro lado, V_{ij} mide la urgencia de realizar la inserción. La urgencia se define como la diferencia entre la hora de arribo a j (sin incluir la espera) y la última hora a la que se podría arribar a dicho cliente. Este parámetro prioriza los clientes a insertar teniendo en cuenta la diferencia de tiempo más tardía para servir al cliente j.

Diferentes valores de δ_1,δ_2, y δ_3, arrojan resultados distintos, según el enfoque que se quiera dar al modelo.

- Si se busca insertar los clientes en las rutas minimizando su cercanía geográfica, se dará menor valor a δ_1.

- Si se priorizan clientes por su cercanía temporal, menores valores de δ_2, guiarán la heurística a que forme rutas con valores ponderados de tiempo y distancia menores.

- Si existen clientes cuyo tiempo de servicio más tardío está alejado del horizonte de planeación de rutas, el coeficiente δ_3, debe ser menor para que se garantice el servicio a clientes críticos.

4.2.1.1 Procedimiento

- A cada uno de los coeficientes δ_1,δ_2, y δ_3, se asigna un valor teniendo en cuenta el enfoque que se quiera dar al modelo. La suma de los tres términos debe ser igual a 1.
- Se calculan los costos C_{ij} desde el depósito hasta cada uno de los clientes. Al hacer este cálculo es importante tener en cuenta que el término T_{ij} desde el depósito hasta el cliente j, es igual al tiempo de inicio de servicio en j,W_j, porque el depósito no tiene asociado un tiempo de inicio del servicio; por lo tanto $W_i = 0$. La Métrica V_{ij}, es equivalente a la diferencia entre el final de la ventana de tiempo en el cliente j y su tiempo de recorrido desde el depósito t_{0j}, así:

$$T_{ij} = W_j$$

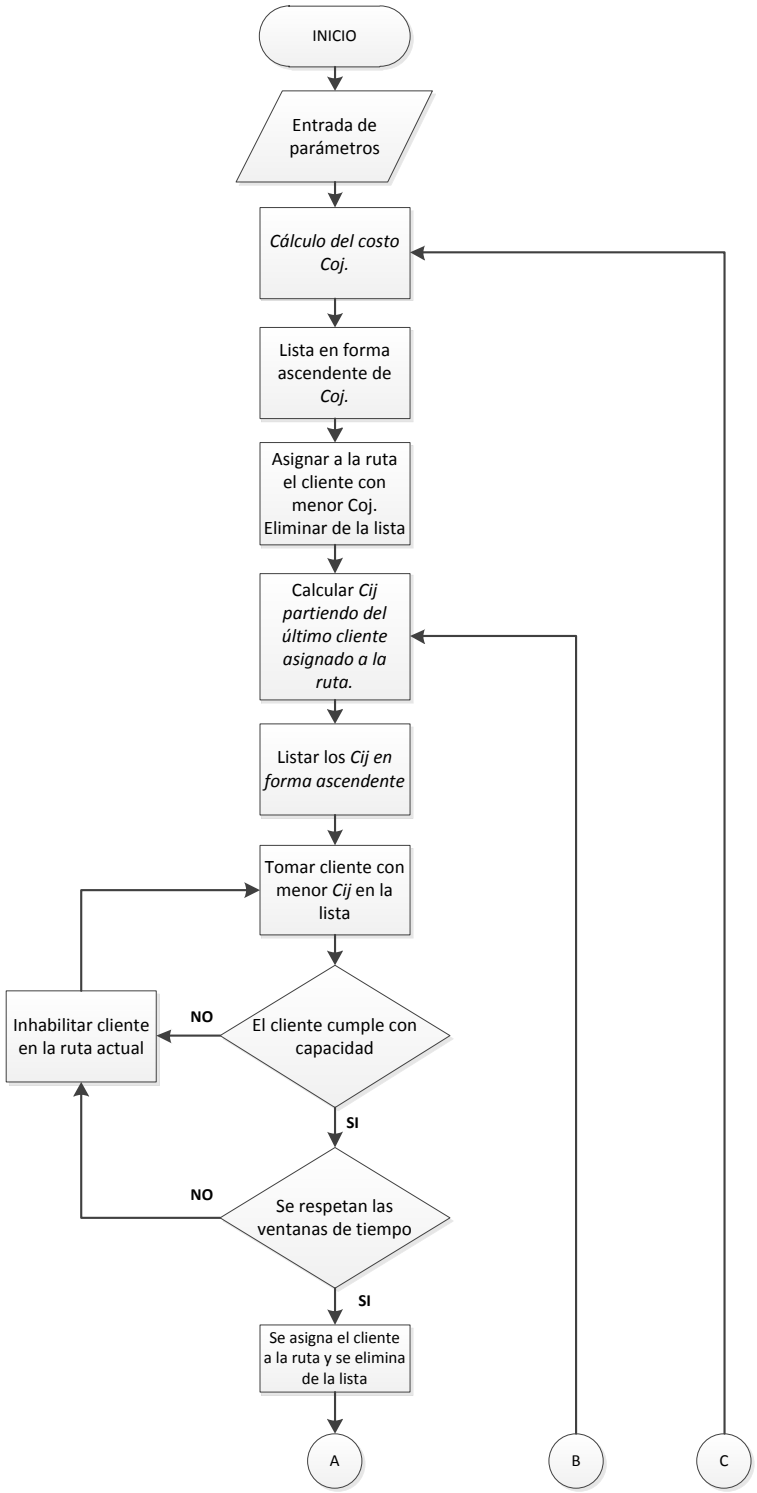
$$V_{ij} = b_j - t_{0j}$$

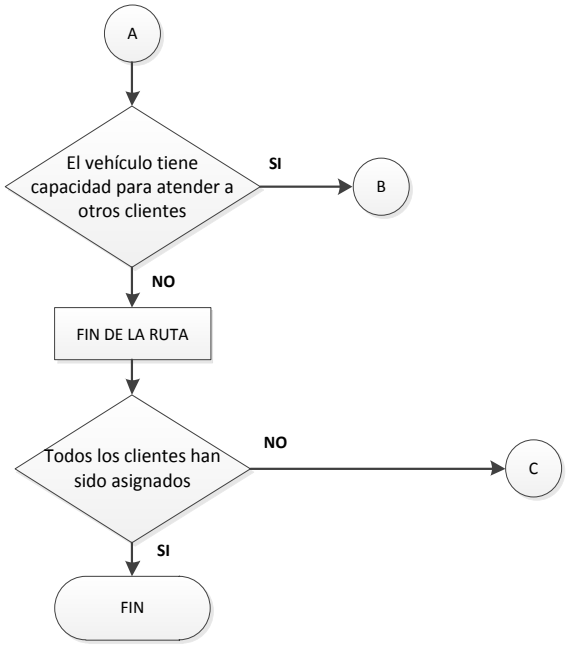
- Se listan los costos C_{0j} en orden ascendente. El algoritmo selecciona el cliente inicial de una ruta como el más cercano al depósito, según la medida C_{0j}, dentro de los clientes no visitados.

- Se calculan los costos C_{ij} partiendo del último cliente asignado a la ruta, desde y hacia cada uno de los clientes restantes y se listan de manera ascendente. En cada paso se selecciona al cliente no visitado que sea más cercano al último cliente de la ruta en términos del parámetro calculado, considerando solamente las inserciones factibles, y se agrega al final de la ruta parcialmente formada.
- Para que una inserción sea factible, la suma de las demandas asociadas con los clientes de la ruta parcialmente formada no debe exceder la capacidad del vehículo. Igualmente se debe observar que el tiempo de llegada al cliente j, no debe ser mayor que el tiempo más tardío en el que el cliente j permite el servicio.
- Si una inserción es no factible, se busca en la lista el segundo costo menor y se inserta en la ruta, si el segundo costo es no factible se buscan en la lista aquellos clientes que no violen las restricciones de capacidad y de ventanas de tiempo, siempre y cuando se respete el orden ascendente.
- Cuando no hayan más clientes para insertar en la ruta actual, se crea una nueva hasta que todos los clientes sean visitados.

El procedimiento del algoritmo de vecino más cercano puede observarse gráficamente en la Figura 28 mediante un diagrama de flujo.

Figura 28. Diagrama de flujo para el Algoritmo de Vecino más Cercano





Fuente: Autores

4.2.2 Heurísticas de Inserción de Solomon.

Por otro lado, se definen 3 heurísticas más que responden a criterios de inserción. Solomon (1987) desarrolló una heurística de construcción secuencial de rutas para el problema VRPTW que ha sido empleada en numerosas metaheurísticas (Potvin, 1996; Potvin y Bengio, 1996; Taillard, 1997; Badeau, 1997). Básicamente estos algoritmos eligen un criterio para comenzar un itinerario y a continuación unas reglas de inserción de clientes.

- **Criterios de inicio de una ruta:** La heurística determina como criterio de elección del primer cliente de una ruta, aquel que se encuentre más alejado del depósito o bien el que presente un límite horario de aceptación del servicio más temprano. Se pretende escoger en primer lugar aquellos nodos con dificultades para asegurar su inclusión temprana en una ruta.

  Criterio 1: El más lejano al depósito.

Criterio 2: El cliente cuya ventana de tiempo termine antes.

Criterio 3: El cliente que minimice la suma del tiempo y distancia al depósito

- **Reglas de Inserción de la Ruta:** Las Heurísticas de inserción de Solomon (1987) proponen varios Criterios de Inserción (Criterios de inserción I1, I2 e I3) que minimizan de forma ponderada el incremento de distancia y tiempo que supone la inclusión de un cliente. Estas métricas están relacionadas con los costos en los que se incurre al introducir un nuevo nodo.

Dado un conjunto de destinos que aún no han satisfecho su demanda y una ruta iniciada, debe elegirse algún criterio que permita intercalar al mejor nodo en el lugar adecuado del recorrido. Esta inclusión modifica los tiempos de llegada y de inicio del servicio de los clientes que se ven precedidos por el cliente insertado en último lugar. (Solomon 1987)

Solomon propone tres reglas o criterios de inserción, que seleccionan los clientes bajo un parámetro C_2 y los inserta en los nodos más baratos según un parámetro C_1.

4.2.2.1 Heurística de Inserción I1

El primer criterio de inserción (I1) considera la ventaja de visitar al cliente u, que es el cliente a insertar, dentro de una ruta parcial y no en una ruta específica para él, es decir, el criterio mide el beneficio de servir el cliente u en una misma ruta con los clientes i e j , en lugar de medirlo a través de un servicio directo o una ruta directa. En el criterio de inserción I1 se definen:

$$C_1(i,u,j) = \alpha_1 C_{11}(i,u,j) + \alpha_2 C_{12}(i,u,j)\,, \alpha_1 + \alpha_2 = 1$$

$$C_{11} = d_{iu} + d_{uj} - \mu\, d_{ij}\,, \mu \geq 0$$

$$C_{12}(i,u,j) = W_{jnew} - W_j = PF\ (Push\ Forward)$$

$$C_2(i,u,j) = \lambda d_{0u} - C_1(i,u,j)\,, \lambda\ \geq 0$$

- A cada cliente no servido u, le corresponde un valor C_1 que determina la posición más barata donde se puede insertar. Se representa como la suma ponderada de dos términos C_{11} y C_{12}.

- C_{11} mide el beneficio, en términos de distancia, si se insertara u entre i e j.

- C_{12} tiene en cuenta el retardo que provoca insertar un cliente nuevo en una ruta parcialmente formada. El tiempo adicional de inserción es llamado también *Push Forward.*

- C_2 selecciona el cliente a insertar tomando el valor C_1 y la distancia del cliente al depósito, de esta manera se privilegia a los clientes para los que sería demasiado crear una ruta individual. El mejor lugar de inserción dado por el criterio C_1 es aquel que maximice el beneficio derivado de servir al cliente en una ruta parcial, en lugar de en una ruta exclusiva.

- Los coeficientes α_1 y α_2, determinan el enfoque de la heurística. Si en la inserción pesan variables temporales, entonces se dará mayor ponderación a α_2, y si es más importante optimizar la distancia, se dará prioridad a α_1.

4.2.2.2 Heurística de Inserción I2.

El criterio de inserción I2 intenta seleccionar el cliente que, si fuera insertado en la ruta, minimiza una medida de la distancia y el tiempo total de la misma. Para la Heurística de Inserción I2 se definen:

$$C_1(i,u,j) = \alpha_1 C_{11}(i,u,j) + \alpha_2 C_{12}(i,u,j)\,, \alpha_1 + \alpha_2 = 1$$

$$C_{11} = d_{iu} + d_{uj} - \mu\, d_{ij}\,, \mu \geq 0$$

$$C_{12}(i,u,j) = W_{jnew} - W_j = PF\ (Push\ Forward)$$

$$C_2(i,u,j) = \beta_1 Rd(u) + \beta_2 Rt(u)\,,\ \beta_1 \geq 0;\ \ \beta_2 \geq 0\,;\ \ \beta_1 + \beta_2 = 1$$

- Aquí, C_1 es igual que en el criterio de inserción I1 y $C_2(i,u,j) = \beta_1 Rd(u) + \beta_2 Rt(u)$ donde $Rd(u)$ y $Rt(u)$ son el tiempo y la distancia total de la ruta si u es insertado entre i y j. En este caso, el cliente seleccionado es el que minimiza C_2.

- Los parámetros β_1 y β_2 determinan las características del modelo. Mayores valores de β_1 tienen en cuenta restricciones geográficas, por otro lado, β_2 guía a la heurística a priorizar las restricciones temporales en la solución.

4.2.2.3 Heurística de Inserción I3.

Para el último criterio de inserción I3, se intenta seleccionar a los clientes cuyos límites en las ventanas de tiempo son cortas para garantizar su servicio, es decir aquellos clientes que tienen urgencia de ser servidos. En este enfoque se definen:

$$C_1(i,u,j) = \alpha_1 C_{11}(i,u,j) + \alpha_2 C_{12}(i,u,j) + \alpha_3 C_{13}(i,u,j);\ \alpha_1 + \alpha_2 + \alpha_3 = 1$$

$$C_{11} = d_{iu} + d_{uj} - \mu\, d_{ij}\,, \mu \geq 0$$

$$C_{12}(i,u,j) = W_{jnew} - W_j = PF\ (Push\ Forward)$$

$$C_{13}(i,u,j) = b_j - W_j$$

$$C_2(i,u,j) = C_1(i,u,j)$$

- C_{11} y C_{12} son los mismos que en I1, solo que en este caso, se agrega un nuevo término a C_1; C_3; que mide qué tan cerca del final de su ventana de tiempo se arribaría al cliente u si este fuera insertado en la ruta.

- Los coeficientes α_1, α_2 y α_3 guían a la heurística en tres distintos enfoques: espacial, temporal y de urgencia para servir a los clientes, respectivamente.

- Finalmente, se utiliza C_1 para seleccionar el próximo cliente a insertar.

Procedimiento Heurísticas I1, I2 e I3.

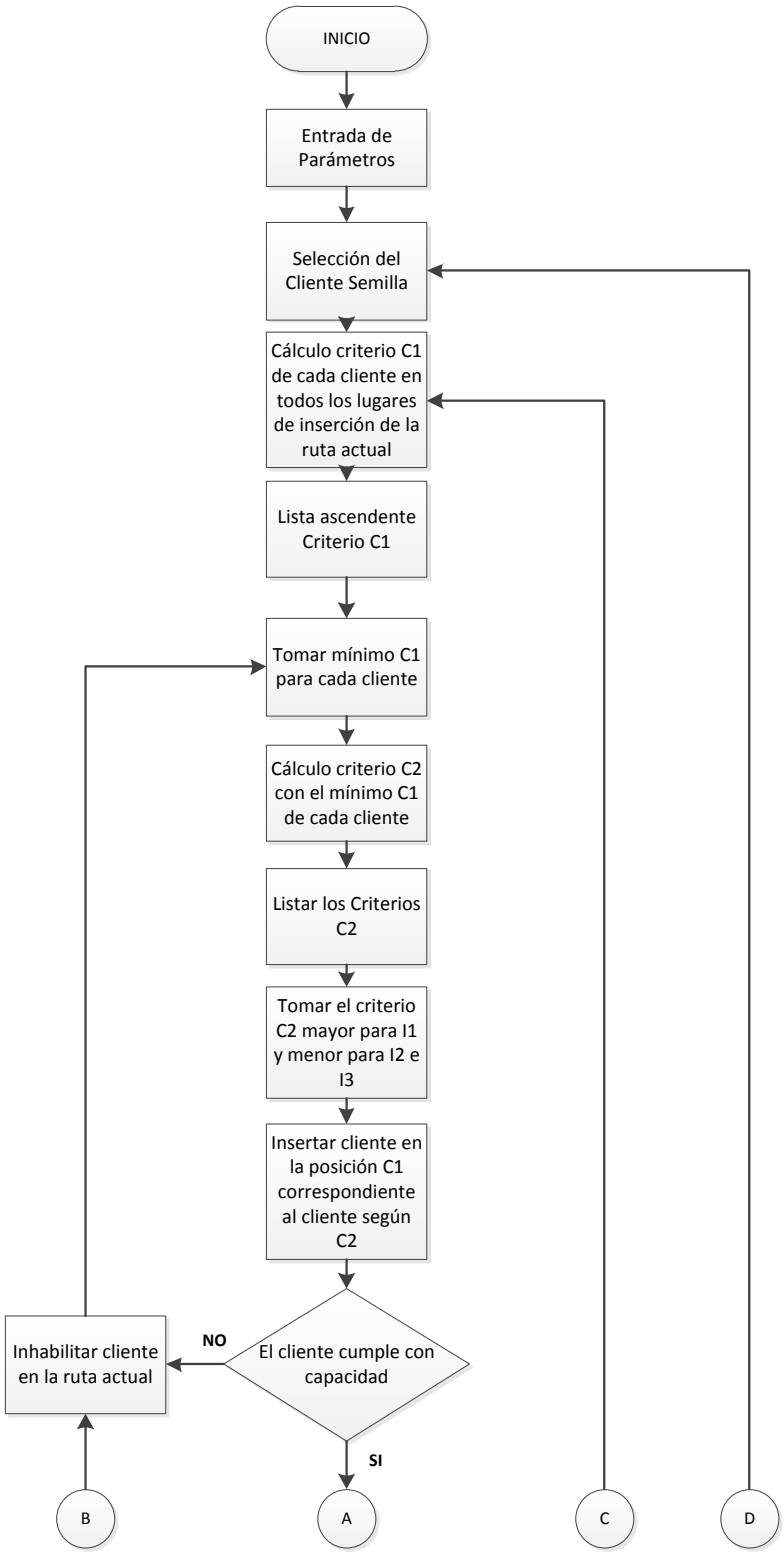
- Seleccionar un cliente semilla con alguno de los criterios establecidos.
- Calcular el criterio C_1 para cada uno de los clientes, exceptuando el cliente semilla.
- Listar los resultados más baratos para cada uno de los clientes en forma ascendente.
- Calcular el criterio C_2 con cada uno de los C_1 más baratos para cada cliente.
- Listar en orden descendente o ascendente (Para I1 es el mayor valor de C_2, para I2 e I3 se selecciona el menor valor de C_2) los resultados de C_2.

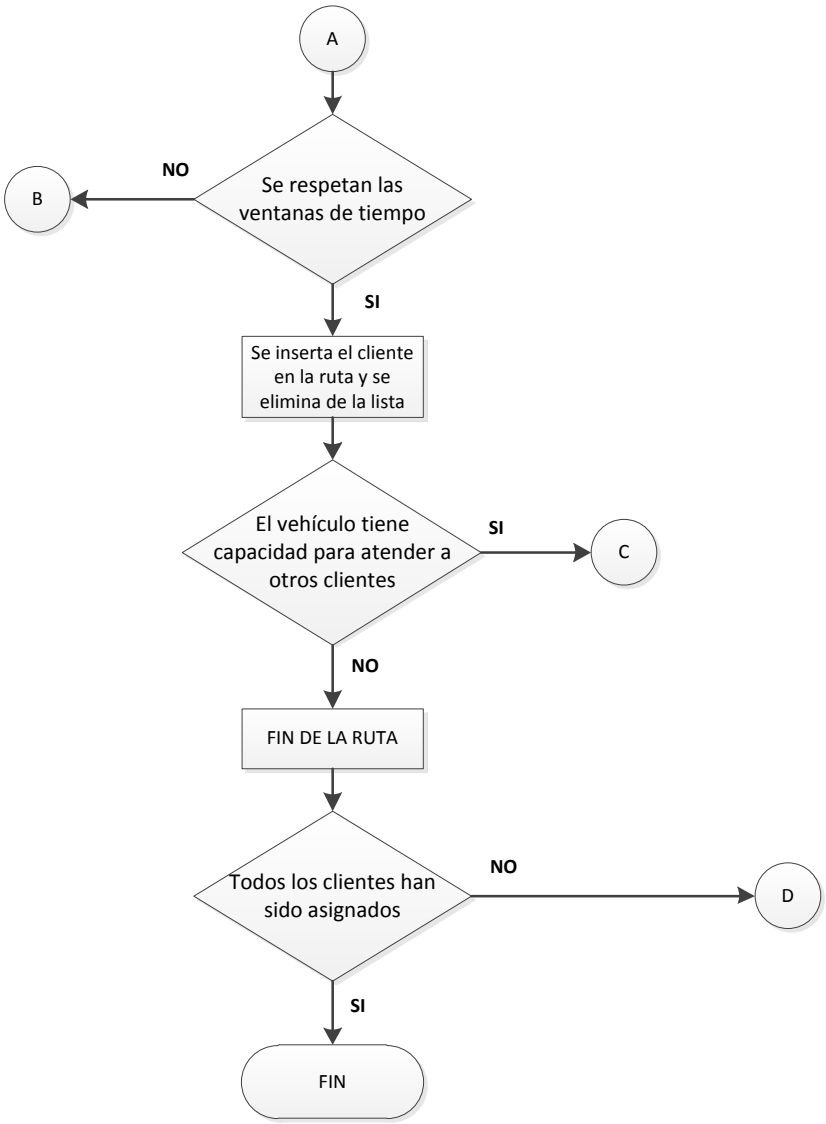
- Seleccionar el criterio C_2, mayor o menor, e insertar el cliente correspondiente en el lugar indicado por el criterio C_1.
- Revisar si la inserción cumple con las restricciones de capacidad y de ventanas de tiempo. Si no es factible, tomar el siguiente cliente en la lista y ubicarlo en la mejor posición correspondiente dada por $C_{1,}$ hasta evaluar todos los clientes.
- Si no hay más clientes para asignar, iniciar una nueva ruta hasta que todos los nodos sean visitados.

Se observa que las heurísticas de inserción de Solomon son una aproximación de la heurística del Vecino más cercano con enfoque temporal, con la diferencia de que en los criterios de inserción I1, I2 e I3, se permite la inclusión de un cliente u_i en cualquier lugar factible dentro de un par de clientes, y no solo al final de la ruta.

El procedimiento de los algoritmos de inserción I1, I2 e I3, puede observarse gráficamente en la Figura 29 mediante un diagrama de flujo.

Figura 29. Diagrama de flujo para los Algoritmos de Inserción de Solomon.





Fuente: Autores

4.3 HEURÍSTICAS DE MEJORA DE RUTAS

En este caso, empezando por una solución factible dada por algún procedimiento de construcción de rutas, se inicia una búsqueda local mediante modificaciones de la solución actual (Thompson y Psaraftis, 1993).

Para los problemas con ventanas de tiempo, es dispendioso en términos computacionales mejorar las soluciones iniciales debido a que cada vez que se cambia de configuración la ruta original, se debe verificar si el cambio cumple con las restricciones temporales y de capacidad de la instancia que se corre.

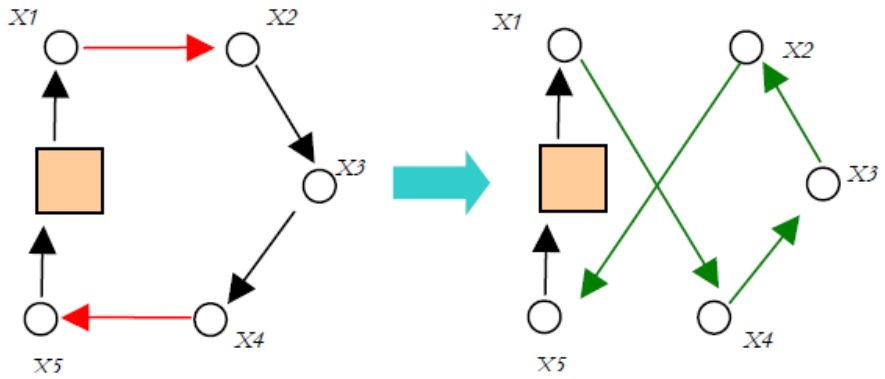
La base de las heurísticas de mejora, es el vecindario de búsqueda. Al hacer modificaciones de este vecindario, se pueden obtener soluciones que mejoren la función objetivo.

La técnica de vecindario más usada es la heurística *r-Opt,* que selecciona un conjunto de arcos y lo sustituye por otros que pueden mejorar la función objetivo inicial. Entre mayor sea el espacio de soluciones, mayor es el conjunto de grafos que se pueden intercambiar.

Para el caso de VRPTW, resulta muy costoso explorar aquellas configuraciones de grafos que no violen las ventanas de tiempo establecidas y que cumplan al mismo tiempo con restricciones de capacidad. Si no existe un conjunto de arcos sustituibles, la solución se llama *r-optimal*.

Cuando se intercambian dos arcos, por otros dos que cumplan con las restricciones temporales y de capacidad y se respeta la dirección que lleva la ruta, se denomina heurística 2-*opt.* Al sustituir los grafos, es necesario cambiar el sentido de algunos arcos, para que el resultado siga siendo una solución factible.

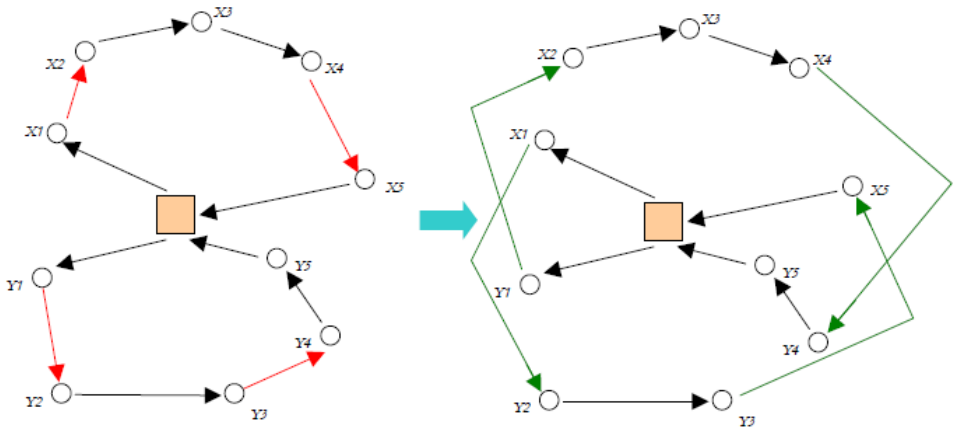
Figura 30. Algoritmo 2-Opt.



Fuente: GALLEGO Olatz. Soluciones en Simulated Annealing para el VRPTW. Tesis Doctoral. Facultad de Informática. P.55. Universidad del País Vasco. 2002.

La operación de *cruce*, consiste en el intercambio de un segmento de una ruta con el segmento de otra, respetando el sentido de la ruta original.

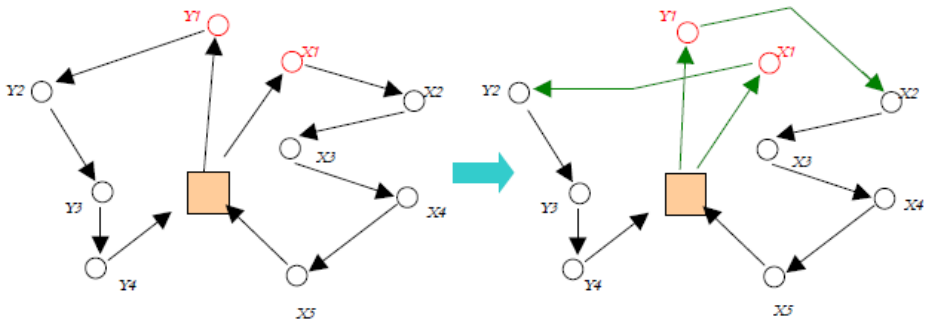
Figura 31. Operación de cruce.



Fuente: GALLEGO Olatz. Soluciones en Simulated Annealing para el VRPTW. Tesis Doctoral. Facultad de Informática. P.55. Universidad del País Vasco. 2002.

La *relocalización*, es la que consiste en seleccionar un cliente de una ruta y ubicarlo en otra. En la operación de intercambio, se intercalan clientes asignados a rutas distintas para genera el vecindario de búsqueda.

Figura 32. Operación de relocalización.



Fuente: GALLEGO Olatz. Soluciones en Simulated Annealing para el VRPTW. Tesis Doctoral. Facultad de Informática. P.55. Universidad del País Vasco. 2002.

5 RESULTADOS COMPUTACIONALES

5.1 ANÁLISIS DE RESULTADOS PARA INSTANCIAS DE CORTO HORIZONTE DE PROGRAMACIÓN

Para analizar el desempeño de las heurísticas, se implementaron cinco algoritmos de construcción de rutas, de todos los estudiados, con el Software Matlab versión R2010a, en un equipo estándar con procesador Intel Core i3 con 3 GB de memoria RAM instalada.

La calidad de la solución se mide en términos del mínimo costo de la ruta, que como se explicó en la sección 3.1, representa la distancia directa recorrida por un vehículo, partiendo y regresando al depósito. Para el presente estudio, una unidad de distancia es recorrida en una unidad de tiempo.

Los algoritmos programados son: Algoritmo de Ahorros de Clarke y Wright en versión secuencial, algoritmo del vecino más cercano con enfoque temporal, heurística de inserción I1, heurística de inserción I2, y la heurística de inserción I3 de Solomon.

Las tablas a continuación, muestran los mejores resultados promedio de cada uno de los grupos de instancias referenciados en la sección 3.3, usando algunos parámetros sugeridos en Solomon [36].

En la columna **Costo**, se muestra el resultado **promedio** del tiempo de recorrido del grupo de rutas. La columna **Tiempo** representa el tiempo **promedio** de programación de las rutas, y la columna **No. Rutas**, es la cantidad de rutas **promedio** finales programadas por cada heurística. El número de rutas es también la cantidad de vehículos usados en la programación, como se definió en la sección 3.1.

El tiempo de programación de las rutas es la suma de los tiempos de recorrido entre rutas, los tiempos de servicio en cada cliente y los tiempos de espera generados al inicio de las ventanas de tiempo. La parte derecha de la tabla contiene las desviaciones porcentuales de las corridas obtenidas respecto a los mejores resultados promedio observados con las mismas heurísticas en Solomon [36].

Para los grupos de instancias R1 y C1, se usan las cinco heurísticas programadas, a diferencia del conjunto de instancias, RC1, probado solo con cuatro: algoritmo del vecino más cercano, y las tres heurísticas de inserción I1, I2 e I3 [36] Todas las instancias son de 100 pedidos o 100 clientes.

Para obtener los resultados reportados en el presente análisis, se usa el siguiente grupo de parámetros:

- **Algoritmo de Ahorros (Versión Secuencial):** Los resultados para esta heurística son los mejores promedios obtenidos de tres corridas con los siguientes parámetros, $\lambda = 1$ $\lambda = 0.6$ y $\lambda = 0.2$, para los grupos de instancias R1 y C1. El Algoritmo de ahorros se programó sin utilizar el parámetro W_f^{new} referenciado en la sección 4.1.1.4, que determina el tiempo máximo que se está dispuesto a esperar cuando se selecciona un cliente para asignarlo a la ruta.

  El primer par de clientes que el algoritmo de ahorros asigna a la ruta, evaluando la factibilidad en términos de capacidad y tiempo de las dos combinaciones posibles de orden. Se selecciona el par con el menor tiempo de espera entre clientes y se asigna a la ruta. Los demás clientes se asignan al final de la ruta parcial, sin examinar el tiempo de espera que se puede generar.

- **Algoritmo del Vecino más cercano:** Los parámetros para esta heurística son: $(\delta_1, \delta_2, \delta_3)$ en cuatro corridas: (0.4, 0.4, 0.2), (0, 1, 0), (0.5, 0.5, 0) y (0.3, 0.3, 0.4).

- **Heurística de inserción I1:** Los resultados consignados en las tablas son los mejores promedios obtenidos de cuatro corridas con los siguientes parámetros $(\alpha_1, \alpha_2, \mu, \lambda)$: (1, 0, 1, 2), (1, 0, 1, 1), (0, 1, 1, 1) y (0, 1, 1, 2). Solo un criterio de inicio de ruta fue probado: El cliente más lejano al depósito.

- **Heurística de inserción I2:** Fueron ejecutadas tres corridas con los siguientes parámetros $(\alpha_1, \alpha_2, \mu, \beta_2, \beta_2)$: (0.5, 0.5, 1, 0.5, 0.5), (1, 0, 1, 0.5, 0.5) y (0, 1, 1, 1, 0). El criterio de inicio de ruta es el cliente más lejano al depósito.

- **Heurística de inserción I3:** Fueron ejecutadas cuatro corridas con los siguientes parámetros $(\alpha_1, \alpha_2, \alpha_3, \mu)$: (0.2, 0.2, 0.6, 0.1), (0.5, 0.5, 0, 1), (0.4, 0.4, 0.2, 1) y (0, 1, 0, 1). Igual que las heurísticas de inserción anteriores, el criterio de inicio de ruta es el cliente más lejano al depósito.

A continuación se presentan los mejores resultados promedio por grupos de instancias y los porcentajes de desviación respecto a los mejores resultados promedio en Solomon [36].

De igual manera, los siguientes gráficos de dispersión complementan la información relacionada en las tablas. Los puntos azules son los resultados obtenidos por la herramienta programada y los puntos rojos son los mejores resultados de Solomon (1987) respecto al mejor costo promedio. Se puede observar a través del gráfico, qué tan lejanos o cercanos están los resultados de la herramienta programada en relación a la investigación de referencia.

Tabla 8. Resultados promedio grupo de instancias R1.

Algoritmo	**Parámetros**		**Resultados (Promedio)**			**Porcentajes de desviación respecto a los mejores resultados promedio. (Solomon 1987)**		
			Costo	**Tiempo**	**No. Rutas**	**Costo**	**Tiempo**	**No. Rutas**
I1	α1	1	1434,60	2808.5	14.5	-0.13%	4.19%	6.62%
	α2	0						
	μ	1						
	λ	1						

I2	α1	1	1703.85	3002.24	15.17	3.98%	3.95%	4.60%
	α2	0						
	μ	1						
	β1	0.5						
	β2	0.5						
I3	α1	0.5	1708.29	3019.38	15.42	3.47%	5.75%	9.34%
	α2	0.5						
	α3	0						
	μ	1						
Ahorros	λ	1	2719.27	6803.13	48.33	81.42%	118.31%	191.16%
Vecino	δ1	0.4	1965.00	3836.09	21.92	22.81%	29.21%	51.17%
	δ2	0.4						
	δ2	0.2						

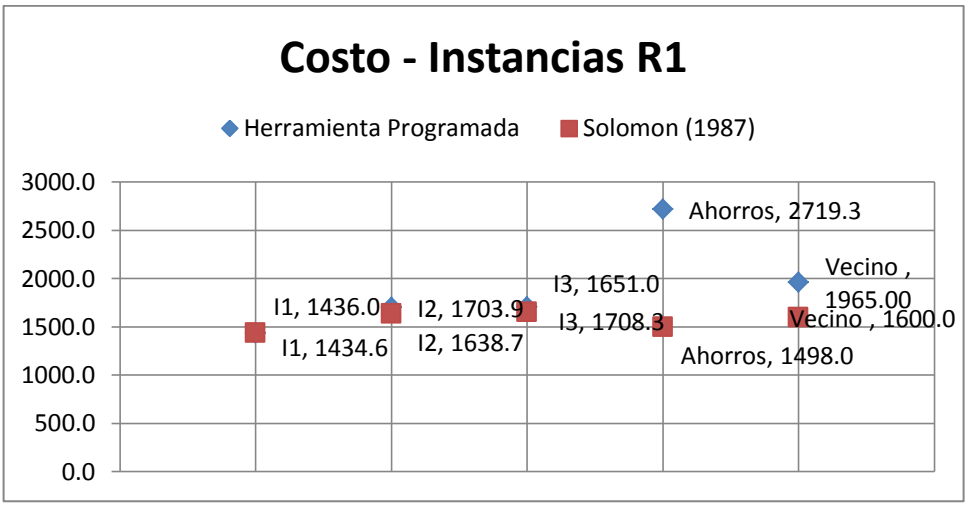
Fuente: Autores

Se evidencia en la Tabla 8 que la heurística de inserción I1 es la de mejor desempeño entre el grupo de heurísticas programadas e incluso entrega un mejor resultado comparado con los de la literatura, su desviación porcentual del 0.13% por debajo del de referencia, con un tiempo de programación de 4.19%, respecto al mejor resultado promedio obtenido por la misma heurística en Solomon 1987.

La Figura 33 es la comparación de los mejores costos promedios para el conjunto de instancias R1. Como se observa, el desempeño de la heurística de inserción I1 es mayor debido a la cercanía de los resultados encontrados en Solomon, seguido por la heurística I3, y luego por la heurística I2, para el conjunto de instancias R1. La heurística de peor desempeño es la heurística de Ahorros, con un porcentaje de desviación del 81.42% respecto a la mejor solución de Solomon 1987.

Figura 33. Comparación Gráfica de resultados por costo. Instancias R1.

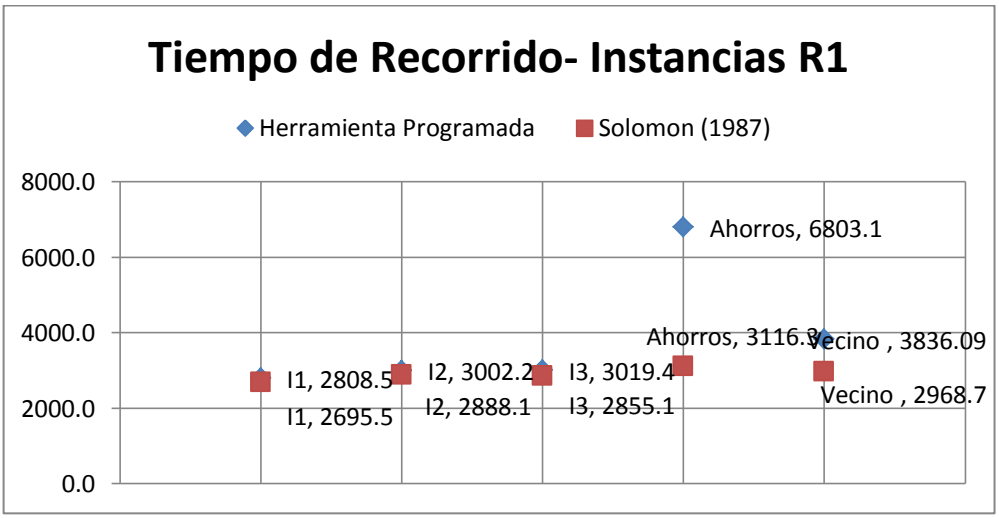
.



Fuente: Autores.

Puede notarse en la Figura 34, con respecto a los tiempos de recorrido de las instancias del grupo R1, que las heurísticas de inserción I1, I2 e I3 presentan un mejor desempeño sobre las demás, con errores porcentuales bajos; mientras que para la heurística de ahorros, la diferencia es significativa. De lo anterior puede inferirse que dado que el algoritmo de ahorros no tiene en cuenta en cada iteración el aspecto temporal, existe gran diferencia con los resultados.

Figura 34. Comparación Gráfica de resultados por Tiempo de recorrido. Instancias R1



Fuente: Autores

Tabla 9. Resultados promedio grupo de instancias C1

Algoritmo	Parámetros		Resultados (Promedio)			Porcentajes de desviación respecto a los mejores resultados promedio. (Solomon 1987)		
			Costo	Tiempo	No. Rutas	Costo	Tiempo	No. Rutas
I1	α1	1	1110.68	10528.18	10.67	16.68%	4.20%	6.67%
	α2	0						
	μ	1						
	λ	1						
I2	α1	0.5	1466.37	11022.44	11.33	39.68%	8.34%	12.21%
	α2	0.5						
	μ	1						
	β1	0.5						
	β2	0.5						
I3	α1	0.5	1454.43	11122.33	11.67	31.83%	9.31%	16.67%
	α2	0.5						
	α3	0						
	μ	1						
Ahorros	λ	0.6	2841.60	25674.11	47.11	191.09%	130.78%	302.66%
Vecino	δ1	0.4	1906.84	12855.78	16.67	62.84%	22.76%	63.40%
	δ2	0.4						
	δ2	0.2						

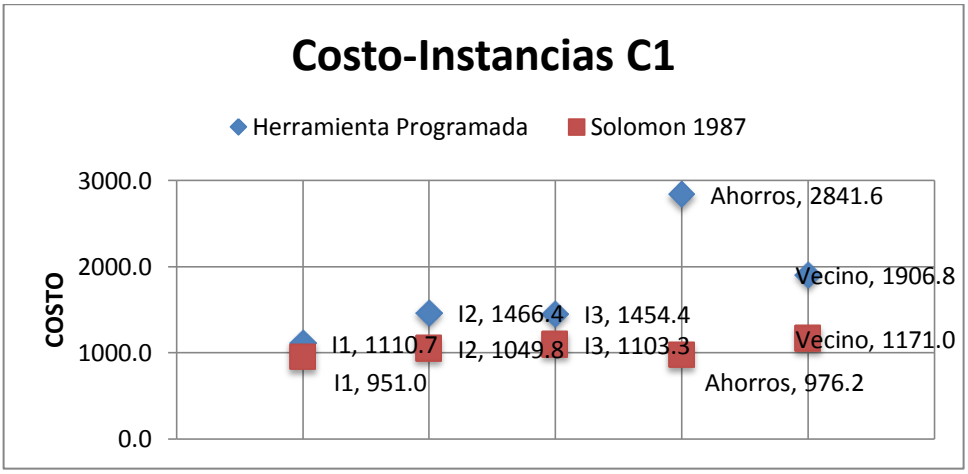
Fuente: Autores

Respecto a las instancias de distribución por clusters de la Tabla 9, se muestran los resultados de la heurística de inserción I1 con mayor desempeño aunque, no tan eficiente como en R1. La desviación fue del 16,68% respecto al mejor costo promedio con un tiempo de programación desviado en 4,20% a la de Solomon.

La Figura 35 es la comparación de los mejores costos promedios para el conjunto de instancias C1. Para las instancias distribuidas en clusters, la heurística de inserción I1 de Solomon, es la más eficiente en términos del mínimo costo promedio, con una diferencia del 16.68% respecto al mejor promedio en Solomon.

La heurística de ahorros arroja resultados muy lejanos al mejor promedio con una diferencia del 191%.

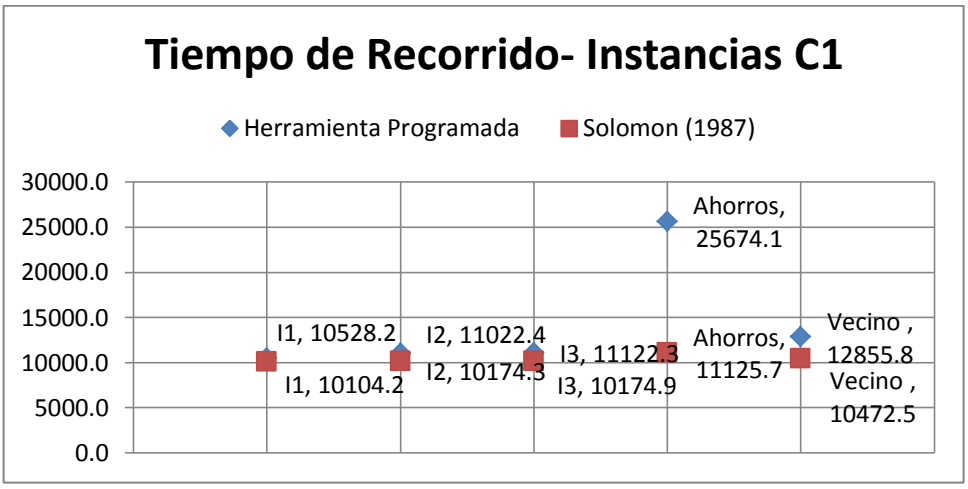
Figura 35. Comparación Gráfica de resultados por costo. Instancias C1.



Fuente: Autores.

Los tiempos de recorrido en las instancias C1, como se observa en la Figura 36, son cercanos a los de la literatura. El algoritmo de ahorros sigue siendo ineficiente para este conjunto de instancias por su gran desviación respecto al mejor promedio.

Figura 36. Comparación Gráfica de resultados por Tiempo de recorrido. Instancias C1



Fuente: Autores

Para las instancias RC de la Tabla 10, la heurística I1 arroja mejores resultados, con tan solo un 0.16% de desviación respecto al mejor costo promedio encontrado en la investigación de Solomon y con un tiempo de programación del 4,90%.

Tabla 10. Resultados promedio grupo de instancias RC1

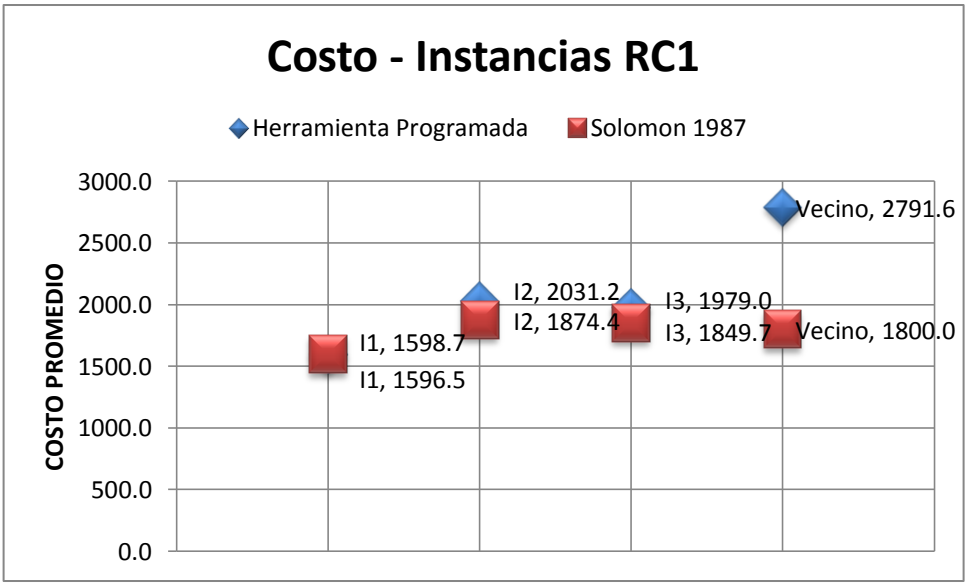
Algoritmo	Parámetros		Resultados (Promedio)			Porcentajes de desviación respecto a los mejores resultados promedio. (Solomon 1987)		
			Costo	Tiempo	No. Rutas	Costo	Tiempo	No. Rutas
I1	α1	1	1598.71	2911.04	14.25	0.14%	4.90%	5.56%
	α2	0						
	μ	1						
	λ	1						
I2	α1	1	2031.20	3269.09	15.75	8.37%	7.91%	10.92%
	α2	0						
	μ	1						
	β1	0.5						
	β2	0.5						
I3	α1	0.5	1978.99	3190.04	15.63	6.99%	5.84%	11.61%
	α2	0.5						
	α3	0						
	μ	1						

Vecino	δ1	0.4	2791.63	4581.04	27.38	55.09%	49.84%	92.78%
	δ2	0.4						
	δ2	0.2						

Fuente: Autores

La Figura 37 es la comparación de los mejores costos promedios para el conjunto de instancias RC1. La heurística de inserción I1 es cercana a la mejor solución promedio encontrada en Solomon para el grupo de instancias RC1 o Random-Clúster, igualmente se obtienen resultados menores en tiempos de programación y en el número de rutas en comparación a los demás grupos de instancias, como se observa en la tabla . La heurística con peor desempeño es la del vecino más cercano, con una desviación porcentual del 55.09% respecto al mejor resultado obtenido con la misma heurística en Solomon [36].

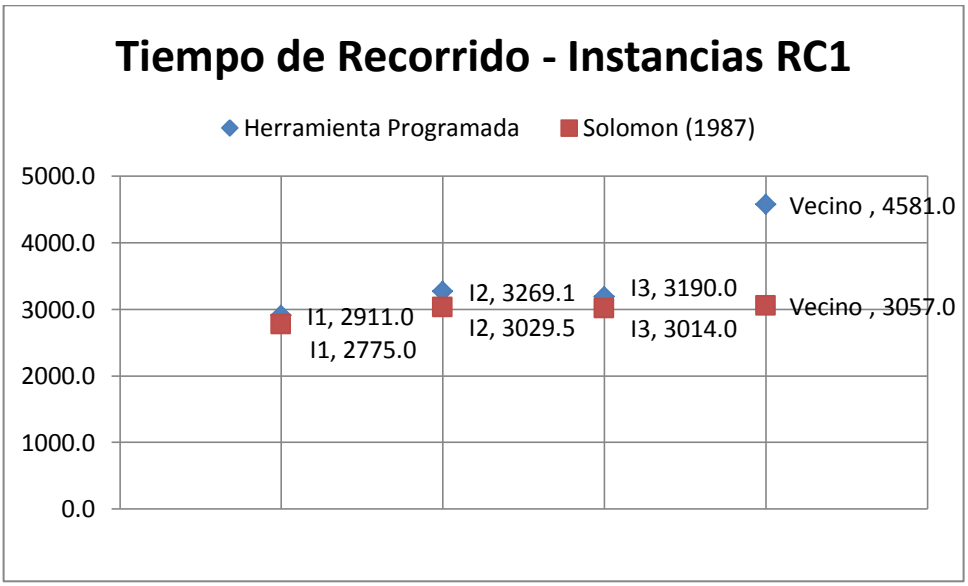
Figura 37. Comparación Gráfica de resultados por costo. Instancias RC1



Fuente: Autores.

Según la Figura 38, los tiempos de recorrido en las instancias RC1, se encuentran en un rango de valores cercanos a los de las instancias R1. Las heurísticas de inserción I1, I2 e I3, son las de mejor desempeño para este conjunto de instancias.

Figura 38. Comparación Gráfica de resultados por Tiempo de recorrido. Instancias RC1

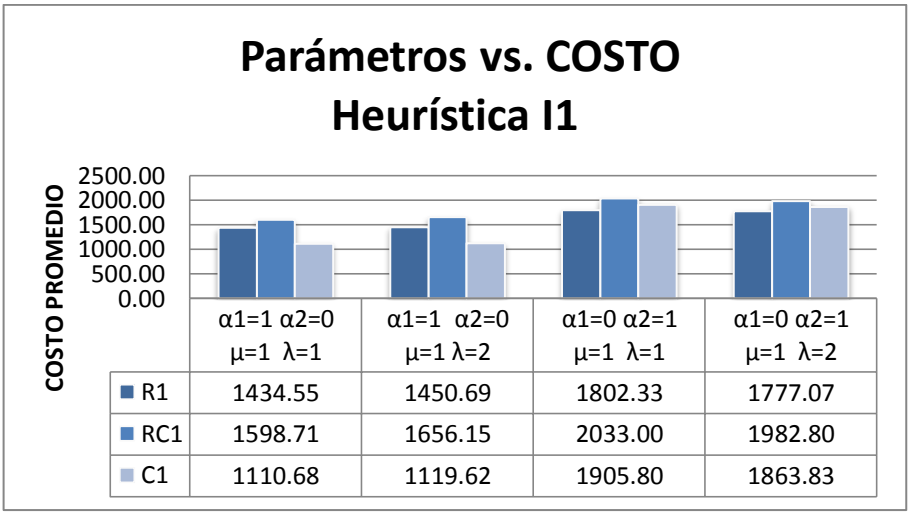


Fuente: Autores

Los siguientes gráficos de barras comparan grupos de parámetros contra el costo (tiempo de recorrido), para cada conjunto de instancias.

En la Figura 39 se muestra el rendimiento general de la heurística de inserción I1, respecto al costo promedio, con cuatro grupos de parámetros. La columna de la izquierda corresponde al grupo de instancias R1, la columna del medio corresponde al grupo de instancias RC1, de la misma manera, la columna de la derecha corresponde al grupo de instancias C1.

Figura 39. Comparación de resultados con cuatro grupos de parámetros para la heurística de inserción I1.



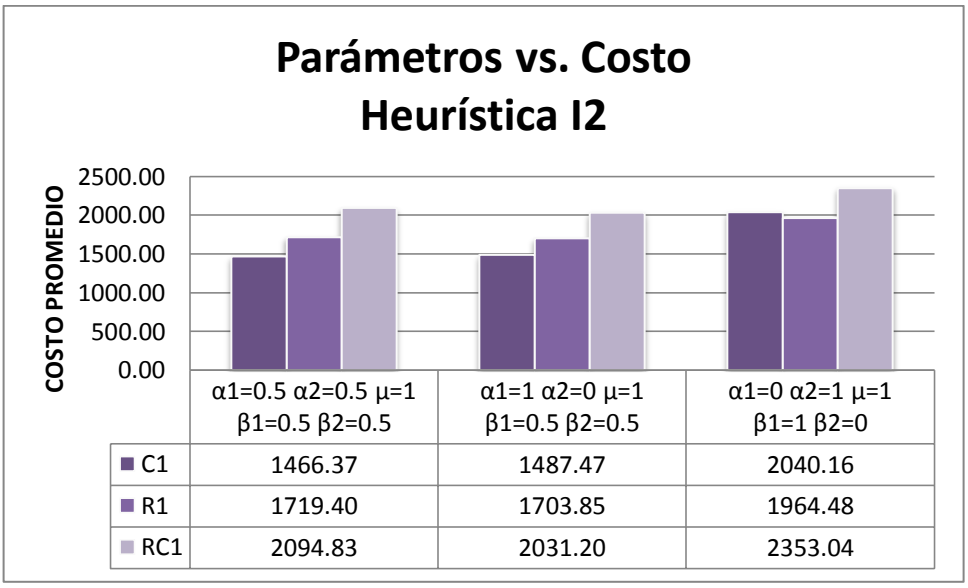
	α1=1 α2=0 μ=1 λ=1	α1=1 α2=0 μ=1 λ=2	α1=0 α2=1 μ=1 λ=1	α1=0 α2=1 μ=1 λ=2
R1	1434.55	1450.69	1802.33	1777.07
RC1	1598.71	1656.15	2033.00	1982.80
C1	1110.68	1119.62	1905.80	1863.83

Fuente: Autores.

Para la heurística de inserción I1 se muestran los mejores resultados con el siguiente grupo de parámetros: $(\alpha_1, \alpha_2, \mu, \lambda)$, (1, 0, 1, 1), que da prioridad a la distancia adicional generada al insertar un nuevo cliente en una ruta parcial, en lugar del tiempo adicional.

En la Figura 40 se grafica el rendimiento general de la heurística de inserción I2, que intenta seleccionar el cliente que, si fuera insertado en la ruta, minimiza una medida de la distancia y el tiempo total de la misma, con tres grupos de parámetros. La columna de la izquierda corresponde al grupo de instancias R1, la columna del medio corresponde al grupo de instancias RC1, de la misma manera, la columna de la derecha corresponde al grupo de instancias C1.

Figura 40. Comparación de resultados con tres grupos de parámetros para la heurística de inserción I2.

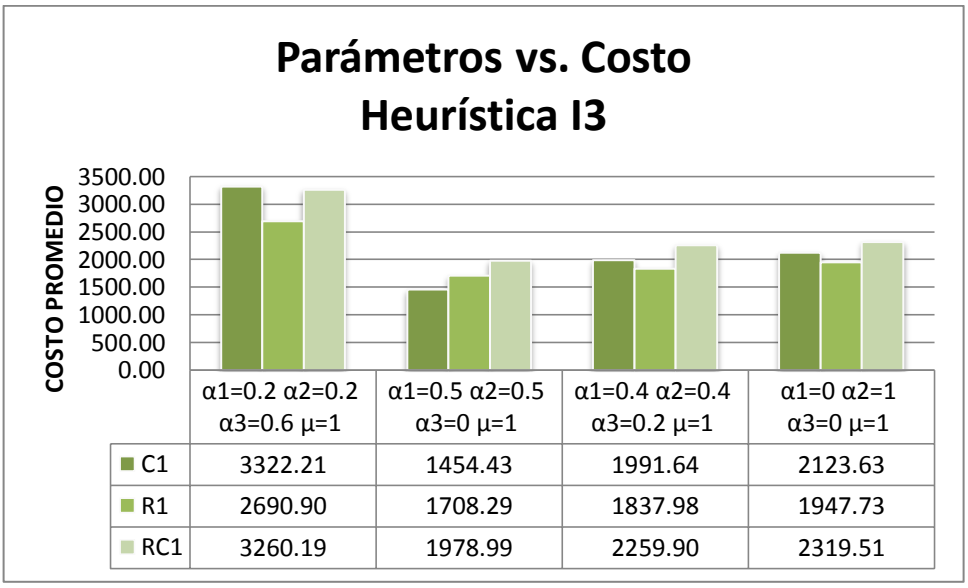


Fuente: Autores.

Se evidencian menores costos promedios al seleccionar el conjunto de parámetros $(\alpha_1,\ \alpha_2,\ \mu,\ \beta_2,\ \beta_2)$: (1, 0, 1, 0.5, 0.5). Este conjunto de parámetros prioriza la disminución de la distancia adicional ponderada, en lugar del tiempo adicional al momento de insertar un cliente en la ruta parcial.

En la Figura 41 se consigna el rendimiento general de la heurística de inserción I3, que selecciona a los clientes cuyos límites en las ventanas de tiempo son cortas para garantizar su servicio, es decir aquellos clientes que tienen urgencia de ser servidos. Se seleccionaron cuatro grupos de parámetros. La columna de la izquierda corresponde al grupo de instancias R1, la columna del medio corresponde al grupo de instancias RC1, de la misma manera, la columna de la derecha corresponde al grupo de instancias C1.

Figura 41. Comparación de resultados con cuatro grupos de parámetros para la heurística de inserción I3.



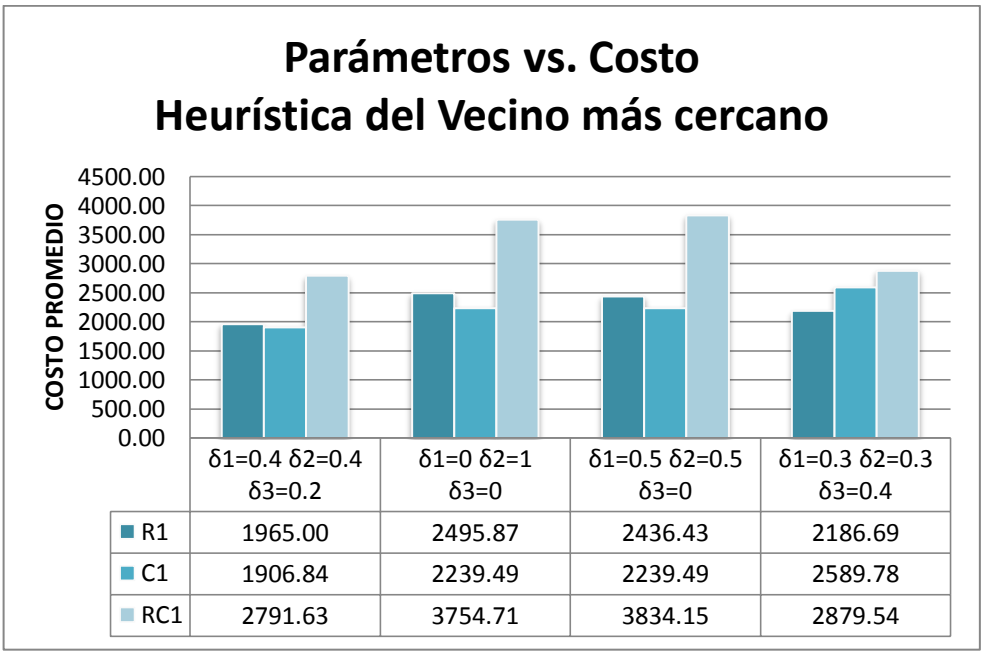
	α1=0.2 α2=0.2 α3=0.6 μ=1	α1=0.5 α2=0.5 α3=0 μ=1	α1=0.4 α2=0.4 α3=0.2 μ=1	α1=0 α2=1 α3=0 μ=1
C1	3322.21	1454.43	1991.64	2123.63
R1	2690.90	1708.29	1837.98	1947.73
RC1	3260.19	1978.99	2259.90	2319.51

Fuente: Autores.

El segundo grupo de parámetros: $(\alpha_1, \alpha_2, \alpha_3, \mu)$: (0.5, 0.5, 0, 1), es el mas eficiente para todo el conjunto de instancias. Para este grupo, el parámetro α_3 , es el que guía a la heurística a rutear clientes con ventanas de tiempo próximas a cerrarse. Sin embargo con un valor de 0 para α_3 , el rendimiento es mayor respecto a los demás valores.

En la Figura 42 se muestra el rendimiento general de la heurística del Vecino más Cercano. Se seleccionaron cuatro grupos de parámetros. La columna de la izquierda corresponde al grupo de instancias R1, la columna del medio corresponde al grupo de instancias RC1, de la misma manera, la columna de la derecha corresponde al grupo de instancias C1.

Figura 42. Comparación de resultados con cuatro grupos de parámetros para la heurística del Vecino más cercano.



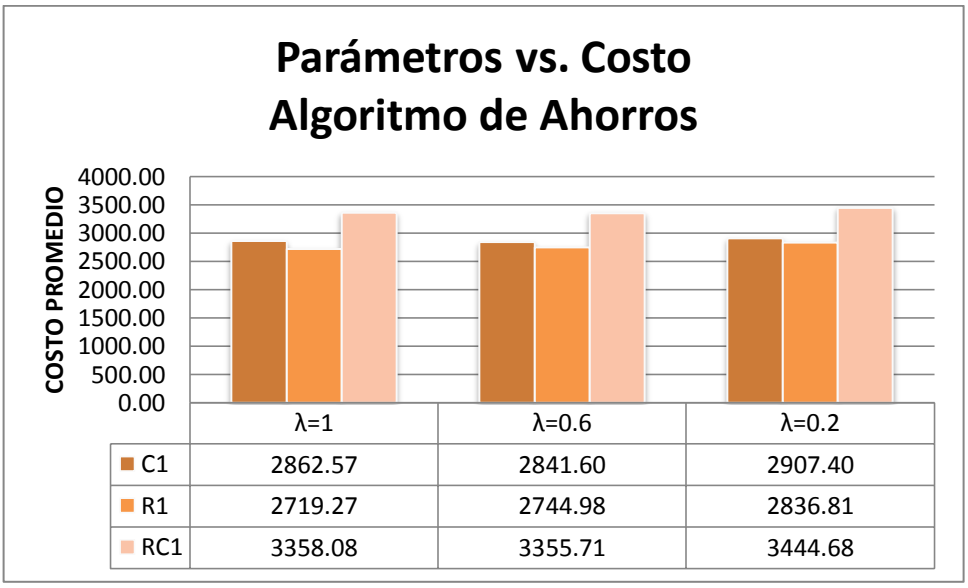
	δ1=0.4 δ2=0.4 δ3=0.2	δ1=0 δ2=1 δ3=0	δ1=0.5 δ2=0.5 δ3=0	δ1=0.3 δ2=0.3 δ3=0.4
R1	1965.00	2495.87	2436.43	2186.69
C1	1906.84	2239.49	2239.49	2589.78
RC1	2791.63	3754.71	3834.15	2879.54

Fuente: Autores.

Para la heurística del vecino más cercano, el primer conjunto de parámetros es el de mejores resultados: $(\delta_1, \delta_2, \delta_3)$ con valores de (0.4, 0.4, 0.2). El parámetro δ_3 con valores inferiores a 0,2 inserta a los clientes con ventanas de tiempo próximas a cerrarse de manera más eficiente.

En la Figura 43 se muestra el rendimiento general de la heurística de Ahorros. Se seleccionaron tres grupos de parámetros. La columna de la izquierda corresponde al grupo de instancias R1, la columna del medio corresponde al grupo de instancias RC1, de la misma manera, la columna de la derecha corresponde al grupo de instancias C1.

Figura 43. Comparación de resultados con tres grupos de parámetros para el algoritmo de ahorros.



	λ=1	λ=0.6	λ=0.2
C1	2862.57	2841.60	2907.40
R1	2719.27	2744.98	2836.81
RC1	3358.08	3355.71	3444.68

Fuente: Autores.

En general, el algoritmo de ahorros arroja costos promedios muy grandes a comparación de los demás grupos de heurísticas. Sin embargo, se evidencia mejores resultados para el para $\lambda = 0.2$.

5.2 ANÁLISIS DE RESULTADOS PARA INSTANCIAS DE LARGO HORIZONTE DE PROGRAMACIÓN

Las instancias de Solomon [35] para el VRPTW con largo horizonte de programación fueron objeto de prueba para las heurísticas de inserción I1, I2 e I3, y la del vecino más cercano. En el proceso de experimentación se utilizaron las Instancias de tipo Random (R), Clúster (C) y Random-clúster (RC) para 100 clientes.

Para cada una de las heurísticas se hacen corridas con varios grupos de parámetros sugeridos en Solomon [36].

Para la heurística I2 se emplean tres conjuntos de parámetros o corridas, para el algoritmo del vecino más cercano, I1 e I3 se emplean cuatro corridas; los parámetros utilizados se describen a continuación en la Tabla 11. Es importante recordar que el criterio de inicialización de ruta para las heurísticas de inserción es el del *cliente sin rutear más lejano al depósito*.

Tabla 11. Parámetros utilizados en la experimentación

Algoritmo	Parámetros	Corrida 1	Corrida 2	Corrida 3	Corrida 4
I1	α1	1	1	0	0
	α2	0	0	1	1
	μ	1	1	1	1
	λ	1	2	1	2
I2	α1	0.5	1	0	
	α2	0.5	0	1	
	μ	1	1	1	
	β1	0.5	0.5	1	
	β2	0.5	0.5	0	
I3	α1	0.5	0.4	0	0.2
	α2	0.5	0.4	1	0.2
	α3	0	0.2	0	0.6
	μ	1	1	1	1
Vecino más cercano	δ1	0.4	0	0.5	0.3
	δ2	0.4	1	0.5	0.3
	δ2	0.2	0	0	0.4

Fuente: Autores

Para seleccionar el mejor resultado se consideró un criterio de calidad, definido como el mínimo costo obtenido de todas las corridas, que es equivalente a la mínima distancia total.

Los resultados obtenidos se muestran a continuación. Los mejores resultados para las instancias R2 se encuentran en la Tabla 12; el costo, tiempo y Nº de rutas pertenecen al promedio de la mejor corrida. Los porcentajes del lado derecho de la tabla corresponden a los errores o la desviación respecto a los mejores resultados de Solomon [36].

Tabla 12. Comparación de resultados. Instancias R2

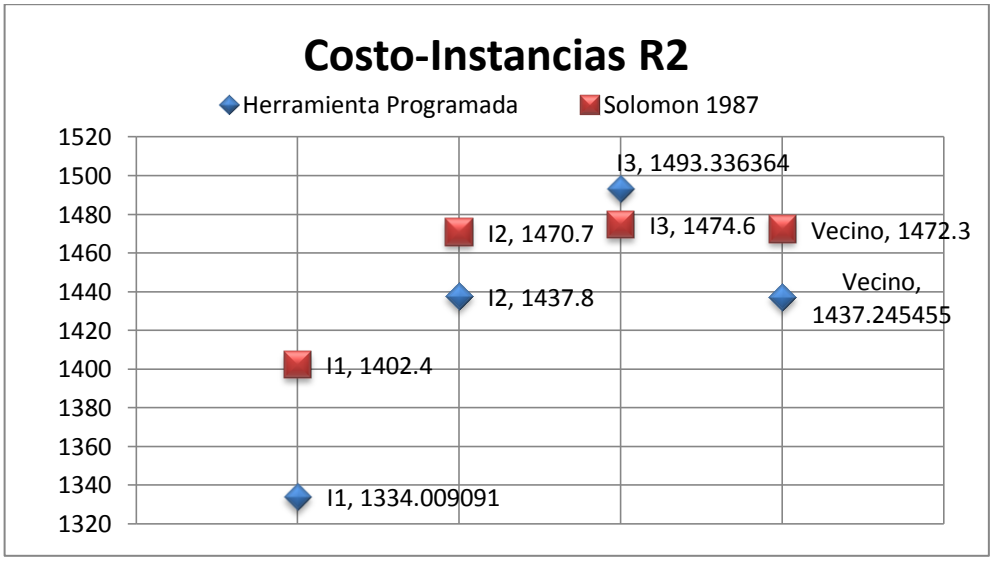
Algoritmo	**Parámetros**		**Resultados (Promedio)**			**Porcentajes de desviación respecto a los mejores resultados promedio. (Solomon 1987)**		
			Costo	**Tiempo**	**No. Rutas**	**Costo**	**Tiempo**	**No. Rutas**
I1	α1	1	1334,01	2857	3,64	-4,88%	10,82%	10,19%
	α2	0						
	μ	1						
	λ	2						
I2	α1	1	1437,80	2872,87	3,82	-2,24%	8,58%	-15,70%
	α2	0						
	μ	1						
	β1	0.5						
	β2	0.5						
I3	α1	0.5	1493,34	2805,82	3,64	1,27%	4,84%	6,95%
	α2	0.5						
	α3	0						
	μ	1						
Vecino más cercano	δ1	0.5	1437,24	3674,18	5,36	-2,4%	35,1%	57,8%
	δ2	0.5						
	δ2	0						

Fuente: Autores

Aunque los valores obtenidos divergen de los de Solomon y algunos son superiores (en términos de costo), es evidente que los errores no arrojan grandes cifras, que permitan afirmar que hay una diferencia significativa. Se mejoran los resultados con la heurística I1, I2 y la del vecino, siendo la heurística de inserción I1 la más eficiente con valores promedio de 4.88% por debajo de los valores de referencia.

En la Figura 44 se puede observar en detalle la comparación del costo entre los resultados obtenidos de la experimentación y los de la literatura. En las instancias R2 se observan buenas soluciones frente a los resultados de Solomon; Las heurísticas de inserción I1 e I2, así como la del vecino más cercano son ejemplos de ello.

Figura 44. Comparación Gráfica de resultados por costo. Instancias R2

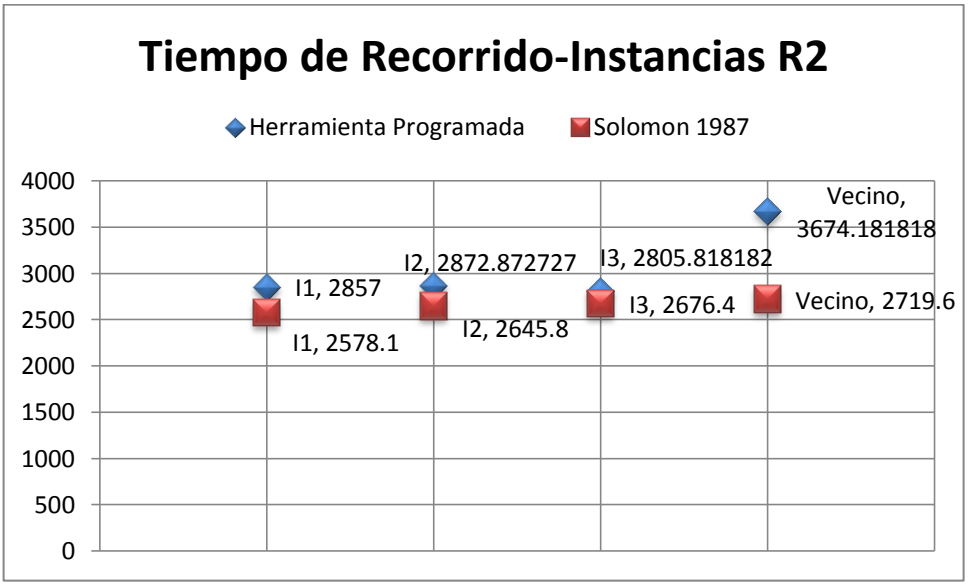


Fuente: Autores

En términos del tiempo de recorrido, los errores son un poco mayores que los del costo. Aunque los resultados obtenidos del tiempo son superiores a los encontrados en la literatura, son aceptables en las heurísticas de inserción I1, I2 e

I3 por ser cercanos, tal y como se evidencia en la Figura 45; caso diferente es la de vecino más cercano, donde el porcentaje de desviación es suficientemente grande como para ser aceptado como un buen resultado.

Figura 45. Comparación Gráfica de resultados por Tiempo de recorrido. Instancias R2



Fuente: Autores

Las instancias C2 por su característica de tener clientes ubicados espacialmente en clusters, genera rutas más cortas. Como puede observarse en la Tabla 13, si se comparan los costos de las instancias C2 con los de R2, hay una disminución de los resultados.

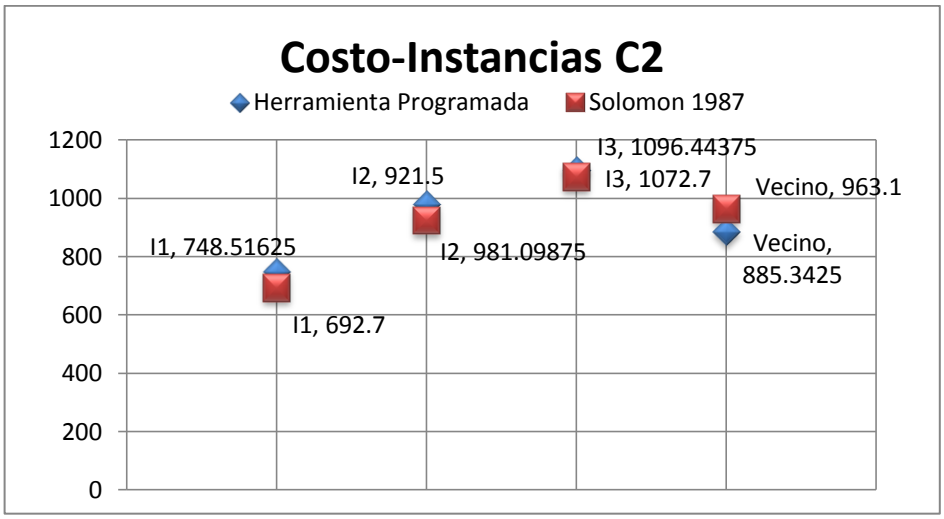
Tabla 13. Comparación de resultados. Instancias C2

Algoritmo	Parámetros		Resultados (Promedio)			Porcentajes de desviación respecto a los mejores resultados promedio. (Solomon 1987)		
			Costo	Tiempo	No. Rutas	Costo	Tiempo	No. Rutas
I1	α1	1	748,51	10006,87	3,62	8,06	0,86	16,94
	α2	0						
	μ	1						
	λ	1						
I2	α1	1	981,10	11667,62	3,87	6,47	14,94	13,97
	α2	0						
	μ	1						
	β1	0.5						
	β2	0.5						
I3	α1	0.5	1096,44	11220	3,87	2,21	10,89	10,71
	α2	0.5						
	α3	0						
	μ	1						
Vecino más cercano	δ1	0.5	885,34	13149,11	4,75	-8,1	21,9	35,7
	δ2	0.5						
	δ2	0						

Fuente: Autores

Los costos arrojados por la experimentación son cercanos a los costos de referencia para el conjunto de instancias C2. La heurística del vecino más cercano es la más eficiente, puesto que mejora el resultado en 8.1% respecto al mejor resultado encontrado.

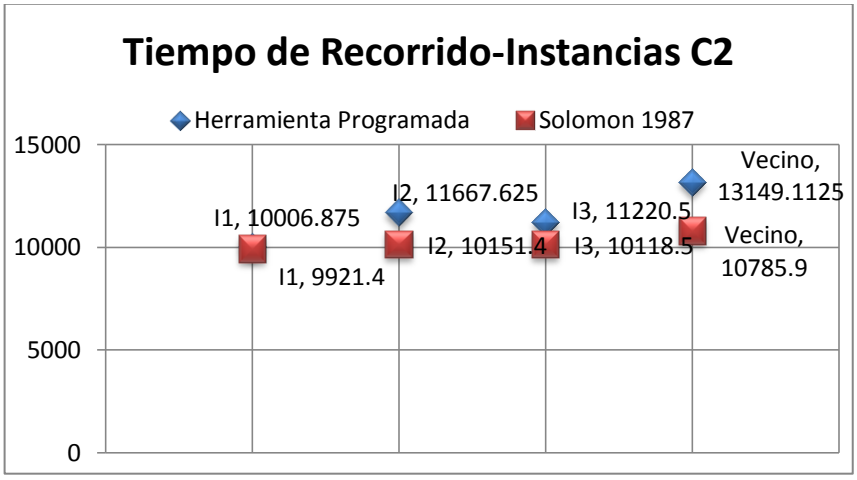
Figura 46. Comparación Gráfica de resultados por costo. Instancias R2



Fuente: Autores

Los tiempos de recorrido para cada heurística se encuentran por arriba de los encontrados en la literatura, la mayor desviación se encuentra en la heurística de vecino más cercano y la menor en la de inserción I1. En la Figura 47 puede apreciarse lo cercanos que se encuentran los resultados de tiempo de recorrido para la heurística de inserción I1 en las instancias de tipo C2.

Figura 47. Comparación Gráfica de resultados por Tiempo de recorrido. Instancias C2



Fuente: Autores

Las Instancias RC2 presentan valores de error bajos, de lo cual se puede inferir que los resultados de costo, tiempo de recorrido y número de rutas encontrados en la experimentación realizada son cercanos a los encontrados en la literatura; en la Tabla 14, puede apreciarse con detalle.

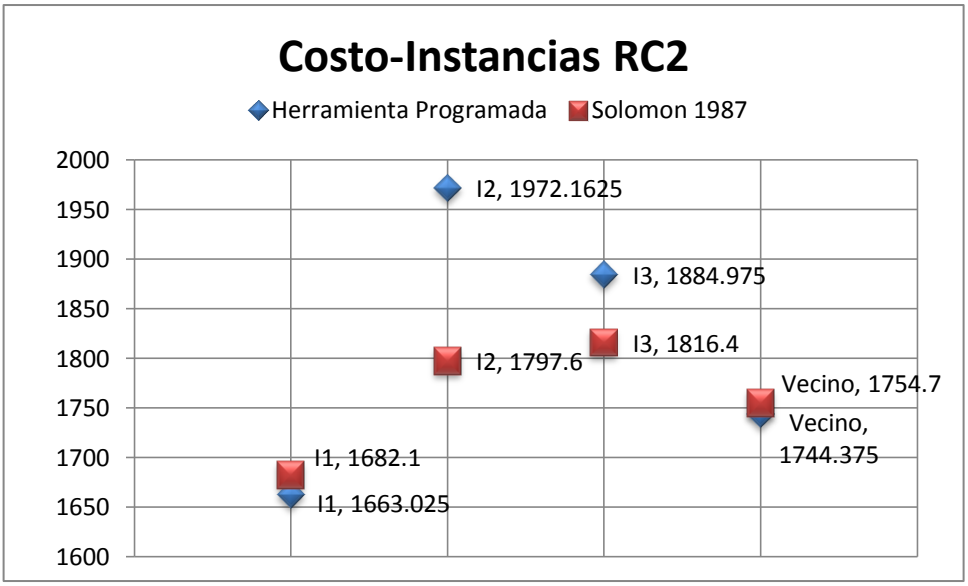
Tabla 14. Comparación de resultados. Instancias RC2

Algoritmo	**Parámetros**		**Resultados (Promedio)**			**Porcentajes de desviación respecto a los mejores resultados promedio. (Solomon 1987)**		
			Costo	**Tiempo**	**No. Rutas**	**Costo**	**Tiempo**	**No. Rutas**
I1	α1	1	1663.02	3174,87	4.12	-1,13	7,43	5,77
	α2	0						
	μ	1						
	λ	2						
I2	α1	1	1972.16	3283	4.12	9,71	4,94	0,61
	α2	0						
	μ	1						
	β1	0.5						
	β2	0.5						
I3	α1	0.5	1884,97	3289	4.25	3,78	4,44	6,25
	α2	0.5						
	α3	0						
	μ	1						
Vecino más cercano	δ1	0.5	1744.37	4014.52	6.37	-0,6	28,7	63,5
	δ2	0.5						
	δ2	0						

Fuente: Autores

En la Figura 48 puede notarse la cercanía de los datos de la herramienta programada y los mejores encontrado de la literatura. La Heurística de inserción I2 es la más alejada del mejor resultado, mientras que la de inserción I1 es cercana al mejor resultado promedio encontrado; por su parte el costo de la heurística del vecino más cercano es menor que el encontrado en la literatura.

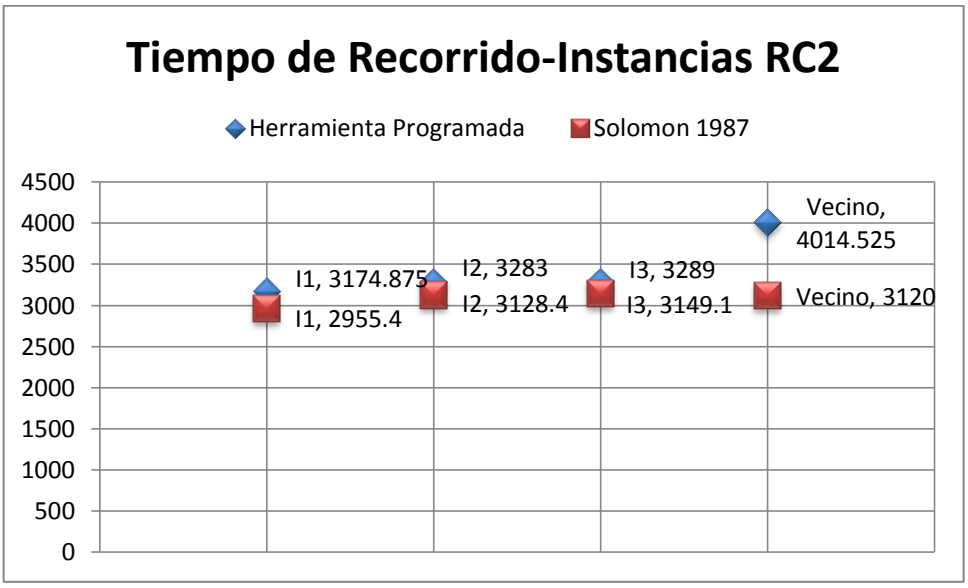
Figura 48. Comparación Gráfica de resultados por costo. Instancias RC2



Fuente: Autores

Cuando se examinan los tiempos de recorrido en las instancias RC2, todas las heurísticas demuestran un buen desempeño. Los valores obtenidos durante la experimentación y los de Solomon se encuentran bastante cercanos, como puede observarse en la Figura 49.

Figura 49. Comparación Gráfica de resultados por Tiempo de recorrido. Instancias RC2



Fuente: Autores

A continuación se hace un análisis por separado de las heurísticas usadas en la experimentación frente a las diferentes corridas, cambiando los valores de los parámetros.

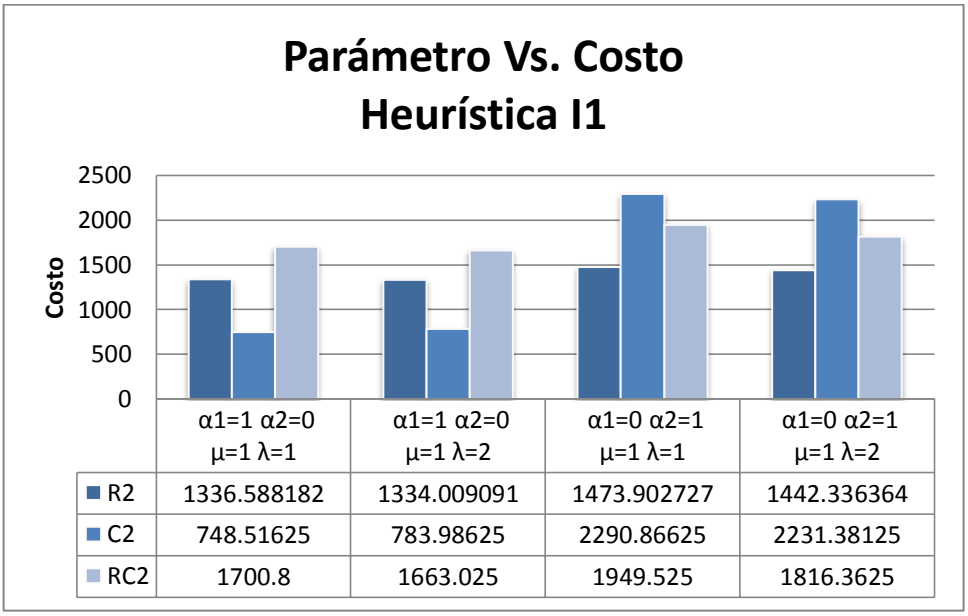
Con respecto a la heurística I1, pueden apreciarse un comportamiento guiado por los parámetros α_1 y α_2, el primero tiene un efecto en la distancia adicional que se agrega en la ruta al insertar al cliente u y el segundo, en el tiempo adicional por visitarlo. Cuando se asignan valores a los parámetros, el algoritmo genera rutas diferentes según la ponderación que se de.

En la Figura 50, se grafica el desempeño de la heurística de inserción I1 para cada conjunto de instancias y con cada conjunto de parámetros. Las dos primeras corridas se realizan con el conjunto de parámetros ($\alpha_1 = 1, \alpha_2 = 0$) y se observa que cuando se prioriza la distancia sobre el tiempo se obtienen costos más bajos.

Lo anterior debido al largo horizonte de programación de las instancias de tipo R2, C2 y RC2; las ventanas de tiempo asociadas a los clientes son mucho más

anchas y permiten mayor holgura en la asignación de clientes a la ruta, logrando que la ruta se enfoque en buscar clientes más cercanos y sin tener mucho impacto en los tiempos de espera.

Figura 50. Comparación de resultados con cuatro grupos de parámetros para la heurística de inserción I1

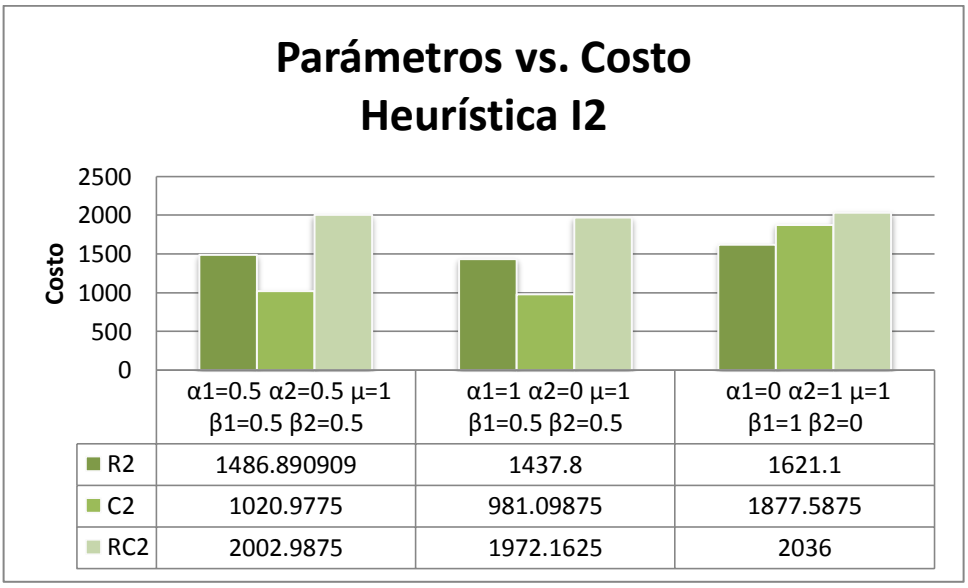


	α1=1 α2=0 μ=1 λ=1	α1=1 α2=0 μ=1 λ=2	α1=0 α2=1 μ=1 λ=1	α1=0 α2=1 μ=1 λ=2
R2	1336.588182	1334.009091	1473.902727	1442.336364
C2	748.51625	783.98625	2290.86625	2231.38125
RC2	1700.8	1663.025	1949.525	1816.3625

Fuente: Autores

El segundo conjunto de parámetros fue el que mejor desempeño tuvo para la heurística de inserción I2, como puede observarse en la Figura 51; al igual que en la heurística de inserción I1, se da una mayor ponderación al factor de distancia de inserción con $\alpha_1 = 1, \alpha_2 = 0$. Los parámetros α_1 y α_2 son claves para definir en qué proporción, la distancia y/o el tiempo contribuyen para seleccionar el mejor lugar de inserción del cliente.

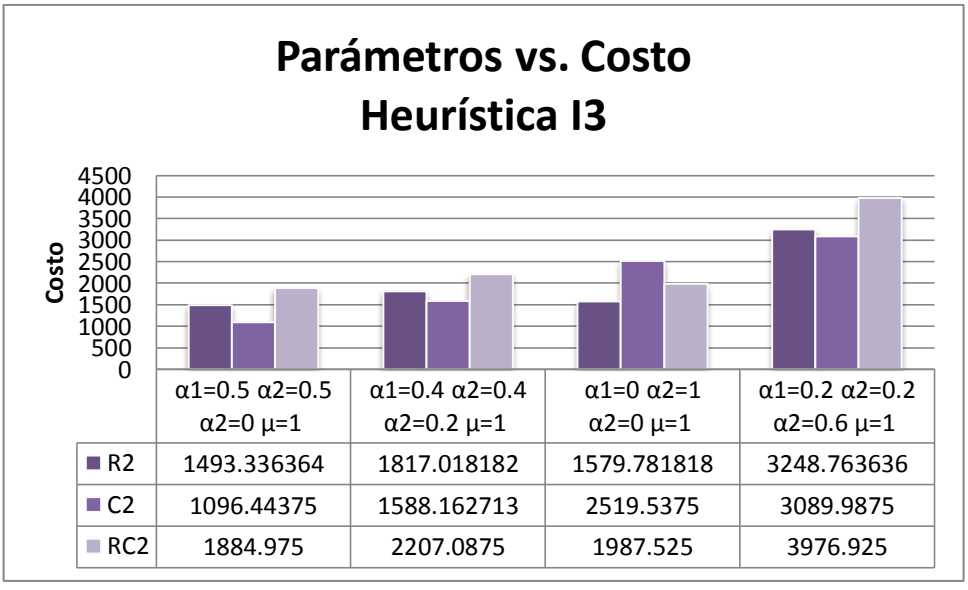
Figura 51. Comparación de resultados con cuatro grupos de parámetros para la heurística de inserción I2



Fuente: Autores

Para el caso de la heurística de inserción I3, aparece un nuevo parámetro α_3, su que no disminuye significativamente el costo de las rutas.

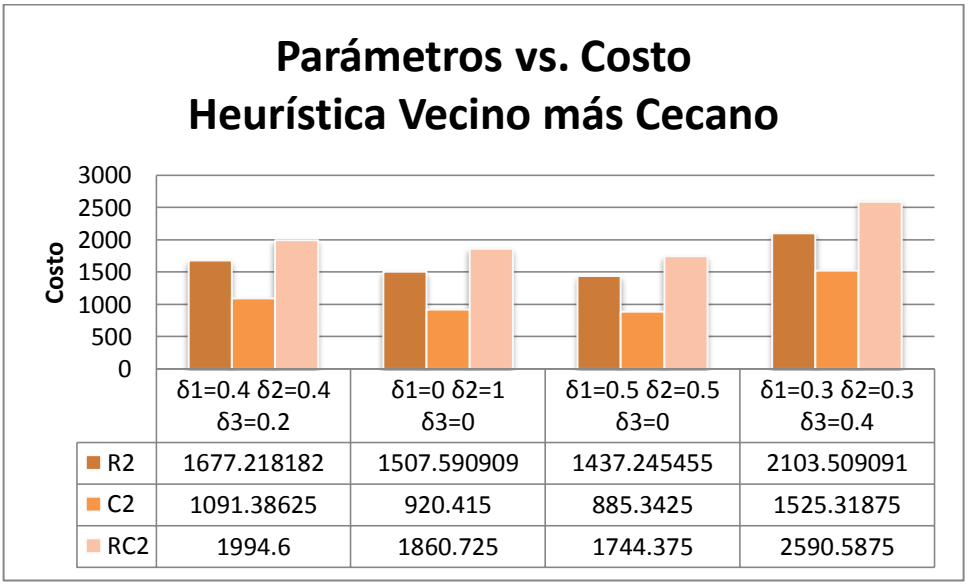
Figura 52. Comparación de resultados con cuatro grupos de parámetros para la heurística de inserción I3



Fuente: Autores

En la heurística del vecino más cercano se encuentran relacionados tres criterios importantes: el criterio espacial, el temporal y el de urgencia de atención; cada uno tiene asociado una ponderación definida por $\delta_1, \delta_2\ y\ \delta_3$ respectivamente. La heurística obtiene un buen desempeño cuando se da mayor ponderación al criterio espacial y temporal por encima del criterio de urgencia como puede apreciarse en la Figura 53.

Figura 53. Comparación de resultados con cuatro grupos de parámetros para la heurística de Vecino más cercano



	δ1=0.4 δ2=0.4 δ3=0.2	δ1=0 δ2=1 δ3=0	δ1=0.5 δ2=0.5 δ3=0	δ1=0.3 δ2=0.3 δ3=0.4
R2	1677.218182	1507.590909	1437.245455	2103.509091
C2	1091.38625	920.415	885.3425	1525.31875
RC2	1994.6	1860.725	1744.375	2590.5875

Fuente: Autores

6 CONCLUSIONES

Los objetivos propuestos para la investigación "Solución del problema de ruteo de vehículos con ventanas de tiempo mediante métodos heurísticos" se cumplieron en su totalidad; la herramienta desarrollada arroja resultados factibles que dan solución al problema. Los resultados son cercanos a los valores de referencia y fueron obtenidos en tiempos de cómputo cortos.

Para el desarrollo de la herramienta computacional fue necesario seleccionar algunas heurísticas que dieran solución al problema de ruteo de vehículos. El proceso de selección se llevó a cabo mediante un estudio exhaustivo en la literatura científica especializada en VRPTW, identificando las heurísticas implementadas que han presentado mejor desempeño en diferentes investigaciones.

Basados en el conocimiento adquirido por la revisión bibliográfica, se definieron algunos criterios de evaluación para calificar y elegir las heurísticas; los criterios tuvieron en cuenta: el acceso a la información y consecución de la misma, la fácil implementación en el lenguaje de programación, la presencia de aspectos temporales como espaciales en las iteraciones del algoritmo y por supuesto la disminución del costo y tiempos de espera.

Los algoritmos seleccionados para la programación son: Algoritmo de Ahorros de Clarke y Wright en versión secuencial, algoritmo del vecino más cercano con enfoque temporal, heurística de inserción I1, heurística de inserción I2, y la heurística de inserción I3 de Solomon.

Las instancias de los grupos Random (R), Clúster (C) y Random-Clúster (RC) de Solomon (1987) fueron objeto de pruebas en la experimentación de las heurísticas con la herramienta computacional. Se ejecutaron varias corridas con parámetros

diferentes midiendo la calidad de la solución en términos del mínimo costo de la ruta.

Al cambiar el criterio de calidad para medir la eficiencia de las heurísticas, por otro como el tiempo de programación de rutas, los parámetros que arrojaban la mejor corrida promedio fueron los mismos, por lo cual no vale la pena realizar un análisis con otro criterio. Se puede concluir, que en general, la mejor corrida promedio seleccionada minimiza todos los criterios de eficiencia.

Al comparar las heurísticas del conjunto seleccionado con las de la literatura, la heurística de inserción I1 se destaca como la más eficiente para las instancias de corto horizonte de programación; en las instancias de tipo R1 presenta mejor desempeño , entrega resultados con desviación porcentual de 0.13% por debajo de los tomados como referencia. En las instancias C1 y RC1, arrojan valores cercanos a los resultados promedio de la investigación que se tomó como referencia.

La razón del desempeño de la heurística de inserción I1 para instancias de corto horizonte es que el algoritmo busca el mejor lugar de inserción para un cliente, balanceando las restricciones temporales y espaciales en cada una de sus iteraciones, de manera que se puedan visitar clientes cercanos en distancia y en ventanas de tiempo para no incurrir en esperas que pueden aumentar el costo de la ruta.

En algunas de las corridas efectuadas con la heurística I1, los parámetros α_1 y α_2 toman valores de 1 y 0 respectivamente; el primer parámetro afecta la distancia adicional que se agrega en la ruta al insertar al cliente u y el segundo, afecta el tiempo adicional por visitarlo. Se observa que cuando se prioriza la distancia, asignando el valor de uno (1), sobre el tiempo, asignando el valor de cero (0), se obtiene un mejor desempeño con costos más bajos.

Respecto a las instancias con largo horizonte de programación, la heurística del vecino más cercano entrega mejores resultados respecto a los de la literatura de referencia en todos los grupos de instancias (R, C y RC), con desviaciones que varían entre 0.6% y 8.1%, por debajo de los mejores promedios, obteniendo mejores resultados para las instancias de configuración por clústeres. La heurística de inserción I1 por su parte, entrega mejores resultados respecto a los la literatura de referencia, en instancias de tipo Random (R) y Random-clúster (RC), con promedios de 4,88% y 1,13% respectivamente, por debajo de los mejores promedios.

La razón de tal desempeño, es que las instancias de largo horizonte como las de tipo R2, C2 y RC2, poseen ventanas de tiempo más anchas que permiten mayor holgura de tiempo en la asignación de clientes, permitiendo al algoritmo buscar clientes más cercanos sin incurrir en grandes tiempos de espera.

Las soluciones arrojadas por la heurística de ahorros no son consideradas en la mayoría de instancias debido a su bajo desempeño respecto a los resultados encontrados en la literatura académica. La razón de tal discrepancia se encuentra en que el algoritmo prioriza el beneficio del ahorro en distancia que se causa cuando se visitan un par de clientes, sin tomar en cuenta la diferencia de tiempo que puedan tener en sus ventanas, generando grandes tiempos de espera.

Se puede afirmar que, en general, la herramienta programada brinda buenos resultados en tiempos de cómputo pequeños, mejorando cuatro de las doce mejores corridas para el conjunto de instancias de largo horizonte de programación.

7 RECOMENDACIONES

Para trabajos futuros, se recomienda incluir más criterios de inicio de ruta para las heurísticas de inserción, como el de la urgencia de servir a clientes con ventanas de tiempo próximas a cerrarse, o la selección el cliente que minimice en mayor medida la función objetivo que se defina para el problema.

Con respecto al algoritmo de ahorros, se recomienda agregar el parámetro W_{new} sugerido en Solomon (1987) que minimiza el tiempo de espera cuando se agregan clientes a la ruta parcial. De igual manera se aconseja incluir algún parámetro o criterio que haga un balance temporal y espacial en los pares de clientes para mejorar el desempeño de tal algoritmo en el conjunto de instancias de estudio.

Dado que el problema de ruteo de vehículos con ventanas de tiempo es de carácter combinatorio, se sugiere implementar heurísticas de mejora para los resultados obtenidos en el presente estudio. De igual manera, las soluciones de la herramienta programada pueden servir como soluciones iniciales para la implementación de técnicas más avanzadas, como metaheurísticas o técnicas híbridas.

La herramienta computacional desarrollada puede servir para resolver problemas del mundo real, por consiguiente es una herramienta que fácilmente sirve en el diseño y desarrollo de un nuevo software comercial, con lenguajes de programación y estructuras informáticas más robustas.

Se deja evidencia teórica y práctica del uso de restricciones temporales en problemas de ruteo, como punto de partida para futuras investigaciones a nivel de posgrado y pregrado para la línea de investigación de ruteo del grupo OPALO.

BIBLIOGRAFÍA

[1] FISHER M.L. An application Oriented Guide to Lagrangean Relaxation. Interfaces, Vol 15, pp. 10-21. 1985.

[2] DANTZIG G.B, EAVES B.C. Studies in Optimization. The Mathematical Association of America (incorporated) Library of Congress Catalog. No. 74-21481.

[3] LARSEN, J. Parallelization of the Vehicle Routing Problem with Time Windows. Tesis Doctoral. IMM Ph.D. 2001.

[4] LUNA, Francisco. Metaheurísticas Avanzadas para problemas reales en redes de telecomunicaciones. Departamento de Lenguajes y Ciencias de Computación. Tesis Doctoral. Doctorado en Ciencias de la Computación. Universidad de Málaga, España., 2008. P 37.

[5] N. METRÓPOLIS, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. Journal of Chemical Physics, 21:1087 – 1092, 1953.

[6] F. GLOVER. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research, 13:533 – 549, 1986.

[7] H. R. Lourenço, O. Martin, and T. Stützle. *Handbook of Metaheuristics*, Kluwer Academic Publishers, chapter Iterated local search, pag. 321 – 353. 2002.

[8] N. MLADENOVIC and P. Hansen. Variable neighborhood search. Com. Oper. Res, 24:1097 – 1100, 1997.

[9] LUNA, Francisco. Metaheurísticas Avanzadas para problemas reales en redes de telecomunicaciones. Tesis Doctoral. Doctorado en Ciencias de la Computación Universidad de Málaga, España. Departamento de Lenguajes y Ciencias de Computación, 2008. P 35.

[10] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109 – 133, 1999.

[11] GLOVER, F.. A template for Scatter Search and Path Relinking. In J-K. Hao et al., editor, *Artificial Evolution*, number 1363 in LNCS, pages 13–54. Springer, 1998.

[12] MONTOYA, José Alejandro. Metaheurísticos: Una alternativa para la Solución de Problemas Combinatorios en Administración de Operaciones. Escuela de

Ingeniería de Antioquia. Revista EIA, ISSN 1794-1237 Número 8, p. 99-115. Diciembre 2007.

[13] MARIUS M**.** Solomon, Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Tomado de: Operations Research, Vol. 35, No. 2 (Mar. - Apr., 1987), pp. 254-265 Publicado por: INFORMS. Disponible en URL: http://www.jstor.org/stable/170697 Consultado: 28/11//2010 10:41

[14] CHRISTOFIDES, S. An Algorithm for the Vehicle Dispatching Problem. Operational Research Quartely 20:309-318. Disponible en Disponible en URL: http://www.jstor.org/stable/170697 Consultado: 12/01//2011 10:41

[15] BODIN L. Routing and Scheduling of Vehicles and Crews. The State of the art. Computers and Operarions Research, 1990, Vol 38. No4.

[16] LAPORTE, G. The Vehicle Routing Problem: An Overview of Exacts and Approximate Algorithms. European Journal of Operational Research. 59, 1992. Pp. 345-358.

[17] KOLEN. A.W.J., Vehicle Routing with Time Windows. Operation Research, Vol 7. No. 2.

[18] FISHER. M.L. Optimal Solution of Vehicle Routing Problems, using minimum K-trees. Operation Research. Vol 42. No 4. Pp. 626-642.

[19] BRAMEL, J. A Location Based Heuristic for General Routing Problems. Operation Research. Vol. 44, No. 3. Pp. 501-509,

[20] RICH, J.L. A Computational Study of Vehicle Routing Applications. Tesis Doctoral. Rice University, Houston Texas. 1999.

[21] RIBEIRO C.C. Column Generation Approach to the Multi-Depot Vehicle Scheduling Problem. Operation Research. 1994, Vol 42, No. 1.

[22] FISHER. M.L. Optimal Solution of Vehicle Routing Problems, using minimum K-trees. Operation Research. Vol 42. No 4. Pp. 626-642.

[23] MINGOZZI A. Dynamic Programming Strategies for the Traveling Salesman Problem with Time Windows and Precedence Constraints. Operation Research. 1997, Vol 45. No. 3.

[24] FISCHETTI M,. A Branch and Cut Algorithm for the Symmetric Generalized Traveling Salesman Problem. Operation Research. 1997. Vol 15, pp. 10-21.

[25] TOTH Paolo, Daniele Vigo. “The vehicle routing problem”. SIAM monographs on discrete mathematics and applications. Philadelphia 2002.

[26] TAILLARD, E.D.. “Parallel iterative search methods for vehicle routing problems”. *Networks,* 23:661-673, 1993. PP 5.

[27] E.D. Taillard, L.M. Gambardella, M. Gendreau, and J.-Y. Potvin. “Adaptive memory programming: A unified view of metaheuristics”. Research Report IDSIA/19-98, IDSIA, Lugano, Switzerland, 1998. PP 7.

[28] GHOSEIRIA, Keivan. Seyed Farid Ghannadpoura. “Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm”. School of Railway Engineering, Iran University of Science and Technology, 16846-13114, Iran. PP 5.

[29] BALSEIRO I.Loiseau, J.Ramonet. “An Ant Colony algorithm hybridized with insertion heuristics for the Time Dependent Vehicle Routing Problem with Time Windows” S.R., Graduate School of Business, Columbia University ,USA Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina, Facultad de Ingenieria, Universidad de Buenos Aires, Argentina.

[30] ZHENGGANG Y Linning. “Improved Multi-Agent System for the Vehicle Routing Problem with Time Windows”. Hunan University, Changsha 410082, China.

[31] MAO Chao, ZHOU Yan-ting. Optimizing Research of an Improved Simulated Annealing Algorithm to Soft Time Windows Vehicle Routing Problem with Pick-up and Delivery DENG Ai-min, School of Business Administration/Institute of Transportation and Logistics, Hunan University, Changsha 410082, China.

[32] ALVARENGAA, G.B., G.R. Mateusb, G. de Tomic. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. .*Department of Computer Science, Federal University of Lavras, UFLA, Lavras Brazil.*

[33] BOUTHILLIERA, Alexandre Le. Teodor Gabriel Crainicb Departement d’informatique et de recherche operationnelle and Centre de recherche sur les transports, Universite de Montreal, C.P. 6128, Succursale Centre-ville, Montreal, QC, Canada H3C 3J7.

[34] MÜLLER Juliane. Approximative solutions to the bicriterion Vehicle Routing Problem with Time Windows., Tampere University of Technology, Department of Mathematics, Korkeakoulunkatu 1, 33720 Tampere, Finland.

[35] RUSSELL , Robert A., Wen-Chyuan Chiang. Scatter search for the vehicle routing problem with time windows. Finance and Operations Management Department, The University of Tulsa, 600 South College Avenue, Tulsa, OK 74104, USA

[36] SOLOMON Marius M. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Operations Research, Vol. 35, No. 2 (1987), pp. 254-265. INFORMS

[37] CLARKE, G. y J. W. Wright (1964). "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". *Operations Research 12, 568-561.*

[38] DULLAERT, W. 2000. "Impact of relative route length on the choice of time insertion criteria for insertion heuristics for the vehicle routing problem with time windows". B. Maurizio, ed. Proc. Rome Jubilee 2000.

[39] POTVIN, J.-Y., J.-M. Rousseau. 1993. "A parallel route building algorithm for the vehicle routing and scheduling problem with time windows". Eur.J.Oper.Res. 66 331–340.

[40] FOISY, C., J.-Y. Potvin. 1993. "Implementing an insertion heuristic for vehicle routing on parallel hardware". Comput.Oper.Res. 20 737–745.

[41] IOANNOU, G., M. Kritikos, G. Prastacos. 2001. "A greedy look-ahead heuristic for the vehicle routing problem with time windows". J.Oper.Res.Soc. 52 523–537.

[42] BALAKRISHNAN, N. 1993. "Simple heuristics for the vehicle routeing problem with soft time windows". J.Oper.Res.Soc. 44. 279–287.

[43] BRAMEL, J., D. Simchi-Levi. 1996. "Probabilistic analyses and practical algorithms for the vehicle routing problem with time windows". Oper. Res. 44 501–509.

[44] RUSSELL, R. A. 1995. "Hybrid heuristics for the vehicle routing problem with time windows. Transportation" Sci. 29 156–166.

[45] SAVELSBERGH, M. W. P. 1986. "Local search in routing problems with time windows". Ann.Oper.Res. 4 285–305.

[46] BAKER, E. K., J. R. Schaffer. 1986. “Solution improvement heuristics for the vehicle routing and scheduling problem with time window constraints”. Amer.J.Math.Management Sci. 6 261–300.

[47] VAN LANDEGHEM, H. R. G. 1988. “A bi-criteria heuristic for the vehicle routing problem with time windows”. Eur.J.Oper.Res. 36 217–226.

[48] KOSKOSIDIS, Y. A., W. B. Powell, M. M. Solomon. 1992. “An optimization-based heuristic for vehicle routing and scheduling with soft time window constraints”. Transportation Sci. 26 69–85.

[49] POTVIN, J.-Y., J.-M. Rousseau. 1995. “An exchange heuristic for routeing problems with time windows”. J.Oper.Res.Soc. 46 1433–1446.

[50] GLOVER, F. 1992. “New ejection chain and alternating path methods for traveling salesman problems”. O. Balci, R. Sharda, S. Zenios, eds. Computer Science and Operations Research: New Developments in Their Interfaces. Pergamon Press, Oxford, U.K., 449–509. Golden, B. L., A. A. Assad. 1986. Perspectives on vehicle routing: Exciting new developments. Oper.Res. 34 803–809.

[51] THOMPSON, P. M., H. N. Psaraftis. 1993. “Cyclic transfer algorithm for multivehicle routing and scheduling problems”. Oper.Res. 41 935–946.

[52] ANTES, J., U. Derigs. 1995. “A new parallel tour construction algorithm for the vehicle routing problem with time window”s. Working paper, Department of Economics and Computer Science, University of Köln, Germany.

[53] RUSSELL, R. A. 1995. “Hybrid heuristics for the vehicle routing problem with time windows”. Transportation Sci. 29 156–166.

[54] HAMACHER, A., C. Moll. 1996. “A new heuristic for vehicle routing with narrow time windows”. U. Derigs, W. Gaul, R. H. Möhring, K.-P. Schuster, eds. Oper.Res.Proc.1996, Selected Papers Sympos. (SOR ’96), Braunschweig, Germany (September 3–6). Springer Verlag, New York, 301–306.

[55] SHAW, P. 1998. “Using constraint programming and local search methods to solve vehicle routing problems”. M. Maher, J.-F. Puget, eds. Principles and Practice of Constraint Programming. CP98, Lecture Notes in Computer Science. Springer-Verlag, New York, 417–431.

[56] CASEAU, Y., F. Laburthe. 1999. “Heuristics for large constrained vehicle routing problems”. J.Heuristics 5 281–303.

[57] CORDONE, R., R. Wolfler-Calvo. 2001. “A heuristic for the vehicle routing problem with time windows”. Journal of Heuristics 7 107–129.

[58] TOTH, Paolo; VIGO, Daniele. THE VEHICLE ROUTING PROBLEM. SIAM, Monógrafos en matemática discreta y aplicaciones. ISBN 0-89871-579-2. Copyright© 2002 por la sociedad para matemáticas industriales y aplicadas

[59] BRÄYSY, O. 2001. “Local search and variable neighborhood search algorithms for the vehicle routing problem with time windows”. Doctoral dissertation, University of Vaasa, Finland.

[60] SOLOMON, Marius M. Algorithms for the vehicle routing problems with time Windows Constraints. Operations Research, Vol. 35, Nº 2 (Mar.-Apr., 1987). P.254-265.

[61] CLARKE, G.; WRIGHT, J.W. Scheduling of vehicles from a central depot to a number of delivery points. Operations research, Vol. 12, Nº 4 (Jul.-Aug., 1964), p.568-581.

[62] DANTZIG, G.B.; RAMSER, J.H. The Truck Dispatching Problem. Management Sci. 6 (1959), p. 80-91.

ANEXOS

ANEXO A. PLANTEAMIENTO DE UN PROBLEMA PARA EL VRPTW USANDO HEURÍSTICAS DE CONSTRUCCIÓN DE RUTAS

En este capítulo se propone el desarrollo de un problema de ruteo de vehículos con ventanas de tiempo usando tres algoritmos de construcción de rutas: algoritmo de ahorros de Clarke y Wright, heurística del vecino más cercano y heurística de inserción I1 de Solomon.

Los datos iniciales para el problema contienen valores aleatorios de las instancias R101.10, R101.25, R101.100 propuestas por Solomon.

Se considera un conjunto de clientes que deben ser servidos por una flota de vehículos de igual tamaño al número de nodos del problema. Cada uno cuenta con una demanda determinista conocida, que debe ser atendida sólo por un vehículo dentro de un intervalo de tiempo que define el cliente según sus necesidades.

Los costos asociados a las rutas son simétricos, es decir, cada grafo dirigido tiene el mismo costo sin importar la dirección en que se recorra. El costo es la distancia total recorrida por un vehículo en una ruta.

Para el desarrollo del problema se dispone de un conjunto de siete clientes que se encuentran dispersos geográficamente según las coordenadas cartesianas *x* e *y*. En la Tabla A1 - 1 se listan las demandas, el inicio y fin de ventana de cada cliente.

Tabla A1 - 1. Datos de entrada

Cliente	X	Y	Demanda	ai	bi	si
0	35	35	0	0	230	0
1	41	49	10	34	44	10
2	35	17	7	32	42	10
3	55	45	13	50	60	10
4	55	20	19	97	107	10
5	15	30	26	87	98	10
6	25	30	3	99	111	10
7	20	50	5	81	91	10
			83			

Fuente: Autores

La Figura A1 - 1 muestra la ubicación geográfica de cada uno de los clientes con sus coordenadas cartesianas. De igual manera, se relacionan las ventanas de tiempo de cada cliente con intervalo gráfico. El valor de la izquierda es el inicio de la ventana, el valor de la derecha es el fin de la ventana.

Figura A1 - 1. Ubicación geográfica de los clientes.

Fuente: Autores

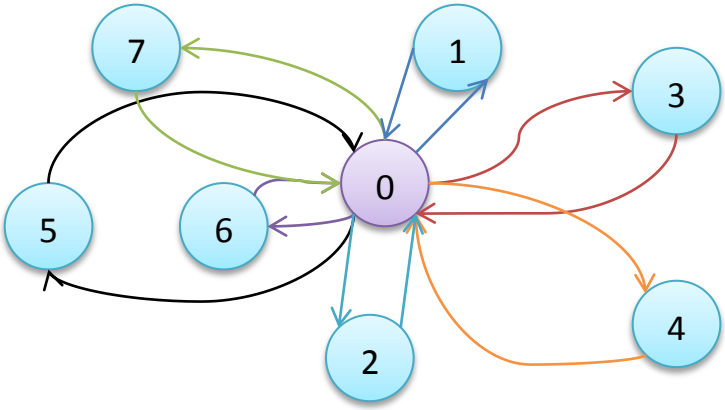
Los límites inferior y superior de los intervalos de tiempo se denominan a_i *y* b_i respectivamente:

- a_i*:* conocido como el momento más temprano en el que el cliente puede ser visitado
- b_i: el momento más lejano o tardío en el que el cliente puede recibir visitas.

El tiempo que tarda el servicio en el cliente *i,* se denomina S_i*.* El valor es constante para todos los clientes. La capacidad de un vehículo *k* de la flota *K* es de 30 unidades.

En la Figura A1 - 2 se representan los grafos iniciales que unen de manera individual a los clientes con el depósito o punto central.

Figura A1 - 2. Rutas Iniciales.



Fuente: Autores

Algoritmo de ahorros de Clarke & Wright.

El ejercicio se desarrolla usando la versión paralela del algoritmo de ahorros de Clarke y Wright.

Se considera el parámetro μ, con μ=1, usado para mejorar la trayectoria que toman las rutas.

La matriz de distancias euclidianas se calcula usando las coordenadas geográficas *x* e *y* correspondientes a cada cliente. La columna de clientes de la izquierda son los nodos de partida y los nodos de llegada se encuentran en la fila superior.

En la Tabla A1 - 2 se muestran las distancias que existen entre el cliente i y el cliente j.

Tabla A1 - 2. Matriz de Distancia

	1	2	3	4	5	6	7	0
1	0,000	32,558	14,560	32,202	32,202	24,839	21,024	15,232

2	32,558	0,000	34,409	20,224	23,854	16,401	36,249	18,000
3	14,560	34,409	0,000	25,000	42,720	33,541	35,355	22,361
4	32,202	20,224	25,000	0,000	41,231	31,623	46,098	25,000
5	32,202	23,854	42,720	41,231	0,000	10,000	20,616	20,616
6	24,839	16,401	33,541	31,623	10,000	0,000	20,616	11,180
7	21,024	36,249	35,355	46,098	20,616	20,616	0,000	21,213
0	15,232	18,000	22,361	25,000	20,616	11,180	21,213	0,000

Fuente: Autores

Matriz de Ahorros

Los ahorros para los de pares de clientes, se calculan con la fórmula de ahorros:

$$\boldsymbol{S_{ij} = C_{io} + C_{oj} - \mu C_{ij}}$$

En la Tabla A1 - 3 se observan los ahorros que se obtienen al fusionar las rutas que contienen a los clientes i y j respectivamente.

Tabla A1 - 3. Matriz de Ahorros

	1	**2**	**3**	**4**	**5**	**6**	**7**
1		0,674	23,032	8,029	3,645	1,573	15,421
2			5,9515	22,777	14,762	12,779	2,964
3				22,361	0,256	$7{,}105*10^{-15}$	8,219
4					4,385	4,558	0,116
5						21,796	21,213
6							11,778
7							

Fuente: Autores

A partir de la matriz de ahorros se obtiene un vector que contiene los ahorros ordenados de manera descendente, en la Tabla A1 - 4 se muestra tal vector que es de gran utilidad para tener en cuenta la prioridad de las visitas en la programación de las rutas.

Tabla A1 - 4. Lista descendente de pares de ahorro

S_{ij}																			
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**
S_{13}	S_{24}	S_{34}	S_{56}	S_{57}	S_{17}	S_{25}	S_{26}	S_{67}	S_{37}	S_{14}	S_{23}	S_{46}	S_{45}	S_{15}	S_{27}	S_{16}	S_{12}	S_{35}	S_{47}	S_{36}

Fuente: Autores

Primera Ruta

Se toma el primer par de la lista que corresponde al ahorro S_{13}. Para iniciar la ruta parcial deben verificarse el cumplimiento de las restricciones del par, la Tabla A1 - 5 muestra el comportamiento de las restricciones para el primer par seleccionado:

Tabla A1 - 5. Restricciones primer par

Restricción Capacidad	La suma de las demandas del par de clientes es 23 unidades. Dado que la capacidad del vehículo es de 30 unidades, el par se considera factible para insertarlo a la ruta.
Restricción Temporal	Por el concepto de desigualdad triangular, el primer cliente del par se considera factible por ventanas de tiempo y distancia desde el depósito. Como $d_{01} < a_1$ se incurre en un tiempo de espera, lo que

	hace que el inicio de servicio en el cliente 2 sea igual al límite inferior de su ventana: $w_1 = a_1 = 34$. El vehículo presta el servicio S_1 y se desplaza hacia 3 con d_{13}. Para determinar la factibilidad del cliente 3 en la ruta se considera: $t_{13} = w_1 + S_1 + d_{13}$. Como $t_{13} \leq b_3$ el cliente 3 es factible por ventanas de tiempo y puede ser agregado a la ruta.

Fuente: Autores

Para el caso de problemas de ruteo de vehículos básico como el CVRP, el sentido de la ruta no tiene importancia, pero para el VRPTW el cambio de sentido puede hacer que la construcción de rutas no sea posible debido a que el orden en que se asignan los clientes modifica el tiempo total de la ruta y se puede incumplir con las ventanas de tiempo.

Para el presente ejercicio, se calculan los ahorros combinando todos los clientes sin tener en cuenta el orden en que se tomen, es decir el par 1-3 es el mismo que 3-1; pero para la inserción en las rutas, se evalúa el orden 1-3 y 3-1 y se asigna el par que cumpla con las restricciones. El análisis de las combinaciones solo se hace cuando se inserta el primer par en cada ruta y los clientes siguientes se unen sólo a los últimos clientes de la ruta parcialmente formada.

En la Tabla A1 - 6 se analizan los pares que tengan relación con los clientes 3 y 1; de esta manera se verifica que al incluirse en la ruta actual cumplan con las restricciones garantizando su factibilidad.

Tabla A1 - 6. Verificación de restricciones para los pares de la lista

Par	Factible Restricciones de Capacidad		Factible Restricción Temporal	
	Si	No	Si	No
3-4		x	x	
1-7	x			x
3-7	x			x
2-3	x			x
1-5		x		x
1-6	x			x
1-2	x			x
3-6	x		x	

Fuente: Autores

- Se evalúa el par 3-4, pero la demanda del cliente 4 sobrepasa la capacidad del vehículo, así que se considera no factible y se elimina de la lista.
- El siguiente par considerado es el par 1-7, que cumple las restricciones de capacidad al agregar sólo cinco unidades al vehículo, pero no se cumple las restricciones temporales.
- El par 3-7 es factible por capacidad ya que al insertarlo en la ruta no sobrepasa la capacidad del vehículo, pero la inserción no cumple con la restricción temporal.
- El par 2-3 se considera factible en términos de capacidad, pero no factible en ventanas de tiempo.
- El par 1-5 no cumple con la restricción de capacidad ni la restricción temporal, así que se elimina cualquier par relacionado con 5 para esta ruta.
- Se examina el par 1-6 que es factible por la demanda asociada pero no factible por ventanas de tiempo, la unión del cliente 6 al cliente 1 que debe

conservarse, imposibilita el recorrido de la ruta, dados las grandes diferencias de sus intervalos.

- El par 1-2 no cumple con las restricciones temporales y el par 3-5 no se considera por que el cliente 5 sobrepasa la capacidad del vehículo si se agregan las demandas de los clientes de la ruta parcial.

De las dos posibles combinaciones, el par 1-3 cumple con las restricciones y se asigna a la ruta. Se selecciona el par que sigue en la lista que tenga relación con el cliente 3, que es el último cliente de la ruta parcial.

Según la Tabla A1 - 6, el par 3-6 cumple con capacidad y se respetan las ventanas de tiempo correspondientes. Se asigna el cliente 6 a la ruta.

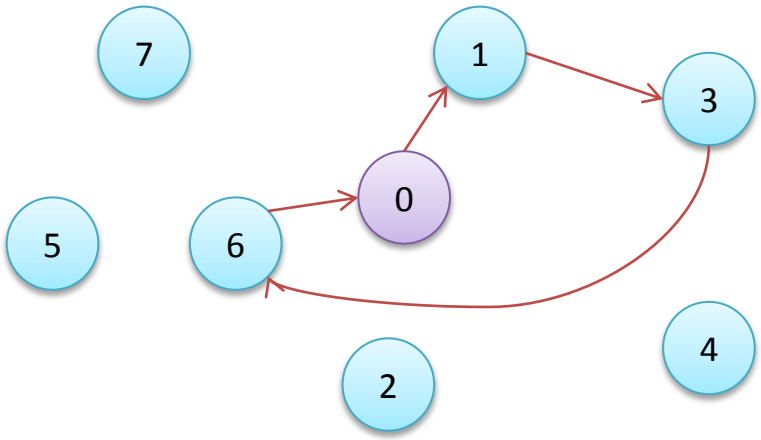
Ruta parcial Nº1: 0-1-3-6

En este punto, la capacidad restante del vehículo (4 unidades) es menor a la demanda de cualquiera de los clientes disponibles, así que se termina la ruta en el depósito.

Ruta Nº1: 0-1-3-6-0

A continuación en la Figura A1 - 3 se muestra la ruta parcial que resulta luego de las primeras iteraciones del algoritmo.

Figura A1 - 3. Ruta Nº1. Algoritmo de Clarke y Wright.



Fuente: Autores

Cada vez que se termina una ruta y se comienza otra, se verifica nuevamente la lista de ahorros en orden descendente (Versión Paralela). Aquellos pares que hayan sido asignados a las rutas o que tengan relación con algún cliente insertado en rutas parciales, deben eliminarse de la lista de ahorros; en la Tabla A1 - 7 se actualiza la el vector de ahorros organizados descendentemente.

Tabla A1 - 7. Lista actualizada de pares de ahorro.

S_{ij}					
1	**2**	**3**	**4**	**5**	**6**
S_{24}	S_{57}	S_{25}	S_{45}	S_{27}	S_{47}

Fuente: Autores

Segunda Ruta

La segunda ruta comienza con el segundo par de la lista, 2-4, que no se consideró por no tener un cliente relacionado con la ruta parcial Nº1.

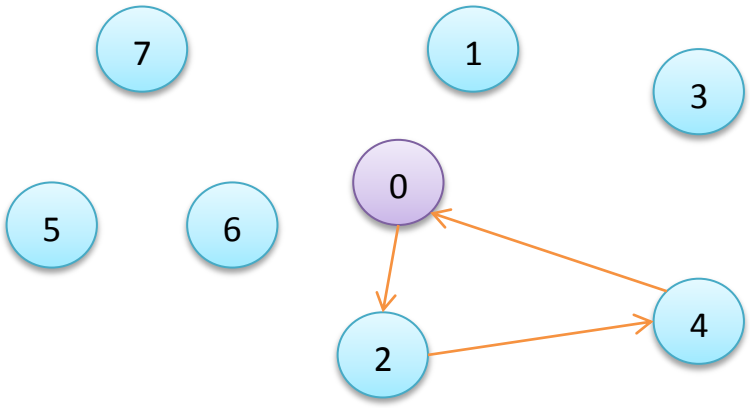
Verificando las restricciones de capacidad, la suma de las demandas del par 2-4 es de 26 unidades, menor que la capacidad del vehículo y los intervalos de tiempo son respetados, así que se consideran clientes factibles y se agregan a la ruta parcial.

La ruta 2 está compuesta solamente por el par 2-4 ya que la capacidad restante del vehículo (4 unidades) no puede ser completada por ninguna de las demandas de los clientes que aún no han sido asignados, así que la segunda ruta es:

Ruta Nº2: 0-2-4-0

En la Figura A1 - 4 se muestra la segunda ruta que se obtiene luego de las iteraciones del algoritmo.

Figura A1 - 4. Ruta Nº2. Algoritmo de Clarke y Wright.



Fuente: Autores

En la Tabla A1 - 8 se muestra la lista actualizada de ahorros donde se eliminan los pares y clientes que ya han sido asignados a las rutas.

Tabla A1 - 8. Lista actualizada de pares de ahorro.

S_{ij}
1
S_{57}

Fuente: Autores

Tercera y Cuarta Ruta

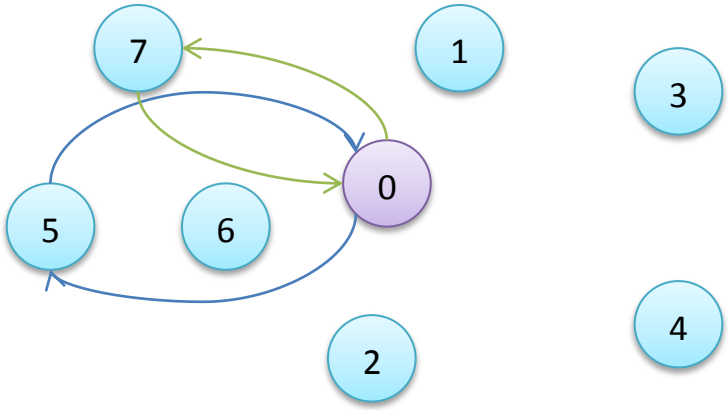
La tercera ruta toma el único par de clientes disponibles en la lista: el par 5-7.

Dado que las demandas de los dos clientes juntos exceden la capacidad del vehículo (31 unidades) y que los demás clientes ya han sido servidos, se asignan en rutas separadas, lo anterior se puede observar en la Figura A1 - 5.

Ruta Nº3: 0-5-0

Ruta Nº4: 0-7-0

Figura A1 - 5. Ruta Nº3 y Ruta Nº4. Algoritmo de Clarke y Wright.



Fuente: Autores

Para calcular el costo de la ruta, se asume que una unidad de distancia es igual a una unidad monetaria: $C_{ij} = \$1/_{und - distancia}$

El **COSTO TOTAL** de la ruta final es:

C=221,3 Unidades Monetarias

Los **TIEMPOS DE TOTALES DE LA RUTAS** o tiempos de programación acumulados de las rutas se muestran a continuación en la Tabla A1 - 9:

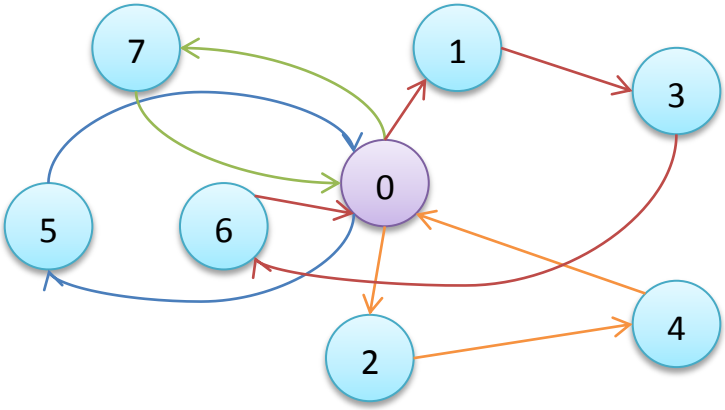
Tabla A1 - 9. Tiempos totales de las rutas.

Nº	Ruta	Tiempo acumulado (Unidades de tiempo)
1	0-1-3-6-0	123,3
2	0-2-4-0	132
3	0-5-0	117.6
4	0-7-0	112,2
Total		**485.1**

Fuente: Autores

La Figura A1 - 6 representa la configuración final de las rutas usando el algoritmo de Clarke y Wright.

Figura A1 - 6. Solución gráfica del ejercicio propuesto. Algoritmo de Clarke y Wright.



Fuente: Autores

ALGORITMO DE INSERCIÓN (I1)

La restricción espacial del algoritmo de inserción es evaluada a través del criterio C_{11} y la restricción temporal a través del criterio C_{12}. La suma ponderada de estos criterios conforma el parámetro C_1 que determina el mejor lugar para la inserción del cliente u.

El criterio C_2 determina el cliente u que represente el mayor beneficio en términos de costo al ser insertado en el lugar que determine el parámetro C_1.

Los datos del ejercicio se muestran a continuación en la Tabla A1 - 10:

Tabla A1 - 10. Datos de entrada

Cliente	X	Y	Demanda	ai	bi	si
0	35	35	0	0	230	0
1	41	49	10	34	44	10
2	35	17	7	32	42	10
3	55	45	13	50	60	10

4	55	20	19	97	107	10
5	15	30	26	87	98	10
6	25	30	3	99	111	10
7	20	50	5	81	91	10
			83			

Fuente: Autores

Los parámetros que guían la heurística son:

- **μ:** Pondera las distancias entre clientes (*i, j*). $\mu \geq 0$.
- $\boldsymbol{\lambda}$**:** Evalúa la distancia que existe desde el depósito al cliente de inserción *u*. $\lambda \geq 0$
- $\boldsymbol{\alpha_1 y\, \alpha_1}$: Se utiliza para dar un peso o ponderación a los lugares de inserción de la ruta parcial. $0 \leq \boldsymbol{\alpha_1} + \boldsymbol{\alpha_1} \leq 1$

Los valores de los parámetros se muestran a continuación en la Tabla A1 - 11:

Tabla A1 - 11. Parámetros I1

Parámetro	Valor
μ	1
λ	1
α1	1
α2	0

Fuente: Autores

Primero se escoge el cliente semilla, que es el primer cliente que se asigna a la ruta. Para el ejemplo, el criterio de selección del cliente semilla es la lejanía al depósito.

Para calcular el cliente inicial se requiere construir la matriz de distancias entre clientes; la Tabla A1 - 12 muestra la distancia entre clientes, la columna de clientes de la izquierda son los nodos de partida y los nodos de llegada se encuentran en la fila superior.

Tabla A1 - 12. Matriz Distancia

	1	2	3	4	5	6	7	0
1	0,000	32,558	14,560	32,202	32,202	24,839	21,024	15,232
2	32,558	0,000	34,409	20,224	23,854	16,401	36,249	18,000
3	14,560	34,409	0,000	25,000	42,720	33,541	35,355	22,361
4	32,202	20,224	25,000	0,000	41,231	31,623	46,098	25,000
5	32,202	23,854	42,720	41,231	0,000	10,000	20,616	20,616
6	24,839	16,401	33,541	31,623	10,000	0,000	20,616	11,180
7	21,024	36,249	35,355	46,098	20,616	20,616	0,000	21,213
0	15,232	18,000	22,361	25,000	20,616	11,180	21,213	0,000

Fuente: Autores

Primera Ruta.

El criterio de selección del cliente semilla encuentra que el cliente más lejano al depósito es 4, en la Tabla A1 - 13 se identifica al cliente que mejor cumple con el criterio:

Tabla A1 - 13. Selección del cliente semilla

	0=Depósito
1	15,232
2	18,000
3	22,361
4	25,000
5	20,616
6	11,180
7	21,213
0	0,000

Fuente: Autores

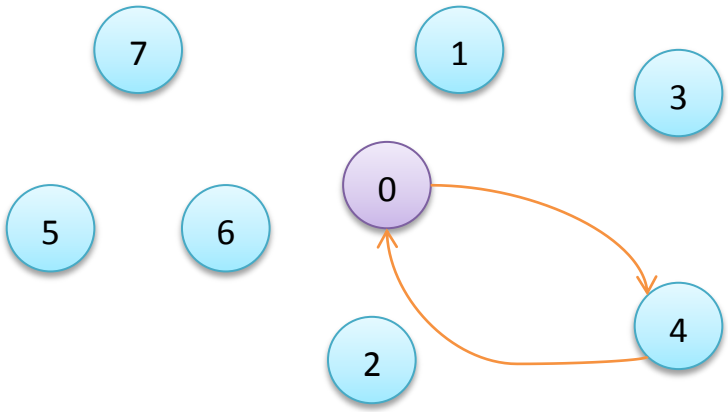
Ruta parcial Nº 1: 0-4-0

La inserción de clientes debe evaluarse en la ruta parcial que se ha formado hasta el momento, los clientes 1, 2, 3, 5, 6 y 7 se encuentran disponibles para ser insertados.

Para determinar los mejores lugares de inserción del cliente u, se evalúan los criterios C_{11} y C_{12} en todos los posibles lugares dela ruta parcial. Existen dos lugares de inserción para el cliente u, 0-4 y 4-0.

La ruta que se origina al unir al depósito con el cliente semilla seleccionado se muestra a continuación en la Figura A1 - 7.

Figura A1 - 7. Ruta Parcial Nº1. Heurística de Inserción I1.



Fuente: Autores

Cálculo del criterio C11.

Antes de realizar los cálculos, se examina la factibilidad de los clientes disponibles, en la Tabla A1 - 14 se observa el análisis de las restricciones para los clientes:

Tabla A1 - 14. Análisis de restricciones.

Cliente	Factible Restricciones de Capacidad		Factible Restricción de Tiempo (0-4)		Factible Restricción de Tiempo (4-0)	
	Si	No	Si	No	Si	No
1	x		x			x
2	x		x			x
3		x	x			x
5		x		X		x
6	x			X		x
7	x			X		X

Fuente: Autores

Los únicos clientes factibles por demanda y ventanas de tiempo son el cliente 1 y el cliente 2. Se procede a realizar el cálculo de los criterios, teniendo en cuenta el lugar de inserción para el que resultaron siendo factibles. En este caso 1 y 2 fueron factibles en el lugar de inserción 0-4.

Para el cálculo de los criterios se definen:

$$C_{11} = d_{iu} + d_{ij} - \mu\, d_{ij};\ 0 \leq\ \mu \geq 1$$
$$C_{12}(i,u,j) = W_{jnew} - W_j = PF\ (Push\ Forward)$$

Para C_{11} se consideran los valores requeridos de la matriz distancia, específicamente d_{01}, d_{14}, d_{40}.

Para C_{12} se considera:

- El término Wju es el inicio del servicio en el cliente j cuando se inserta el cliente u. Para obtener este valor debe tenerse en cuenta el tiempo de recorrido en la ruta, iniciando con la distancia del depósito al cliente u d_{01}.
- Si d_{01}es menor al límite inferior de la ventana de tiempo del cliente 1 entonces el tiempo acumulado se actualiza a este último valor (de no ser así el T sigue siendo el mismo), así: si $d_{01} \leq a_1$, $T = a_1 = 34$. Cuando se llega al cliente 1 se presta el servicio con un tiempo S_1 que debe adicionarse al tiempo acumulado, $T = 34 + 10$=44. Eventualmente el vehículo debe desplazarse al siguiente cliente que es 4, el tiempo de desplazamiento d_{14} se adiciona al T acumulado, si el resultado es menor o igual a la ventana de tiempo del cliente 4, (como sucede en este caso) el tiempo T debe actualizarse de nuevo: $T + d_{14} = 76{,}2 \leq a_4$, $T = a_4 = 97$. El inicio de servicio cuando se inserta el cliente 1 entre 4 y 0 es 97 unidades de tiempo.

- El término W_j se calcula de la misma manera exceptuando la inserción del cliente u.
- El mismo procedimiento se realiza para todos los clientes factibles, en este caso para el cliente 2.

La sustracción de los términos W_{ju} y W_j da lugar al criterio C_{12}; en consecuencia, la suma de los criterios C_{11} y C_{12}, dan lugar a C_1, el valor obtenido para cada uno de los criterios se observa en la Tabla A1 - 15:

$$C_1(i,u,j) = \alpha_1 C_{11}(i,u,j) + \alpha_2 C_{12}(i,u,j);\ \alpha_1 + \alpha_2 = 1$$

Tabla A1 - 15. Cálculo de los criterios C_{11} y C_{12}

U	C11 0-4	Wju	Wj	C12 0-4
1	22,43403059	97	97	0
2	13,22374842	97	97	0

Fuente: Autores

Se halla el mínimo valor de C_1 del cliente u en todos los lugares de inserción. En el ejercicio, los clientes 1 y 2 sólo son factibles entre el depósito y el cliente 4, los valores se observan en la Tabla A1 - 16:

Tabla A1 - 16. Cálculo criterio C_1

U	C1 0-4
1	22,43403059
2	13,22374842

Fuente: Autores

Se calcula el criterio C_2 para cada cliente factible tomando en cuenta el lugar correspondiente en C_1 . El mayor valor C_2 elige el cliente que será insertado en la ruta parcial en la posición dada por C_1. El parámetro λ, como se definió al comienzo del ejercicio, toma un valor de 1, para ponderar la distancia desde el depósito hasta el cliente u .

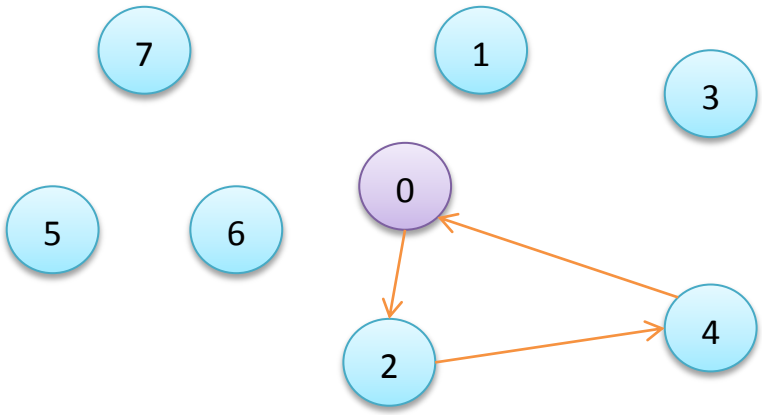
$$C_2(i,u,j) = \lambda d_{0u} - C_1(i,u,j);\ 0 \leq\ \lambda\ \geq 1$$

El mayor valor del criterio C_2 es 4,776 y pertenece al cliente 2 en la posición 0-4. La ruta queda formada así:

Ruta Nº 1: 0-2-4-0

Para la ruta parcial Nº 1, el único cliente que cumple con restricciones de capacidad es el cliente 6, pero no es factible por ventanas de tiempo, por lo tanto no puede ser evaluado para insertarlo en la ruta parcial. Se finaliza la ruta Nº 1 que puede detallarse en la Figura A1 - 8 y se inicia otra que atienda los clientes no servidos. La capacidad usada del vehículo son 26 unidades de las 30 disponibles.

Figura A1 - 8. Ruta Nº1. Heurística de Inserción I1



Fuente: Autores

Segunda Ruta.

En la Tabla A1 - 17 se identifica al cliente semilla de la nueva ruta parcial Nº 2, es el cliente 3:

Tabla A1 - 17. Selección del cliente semilla

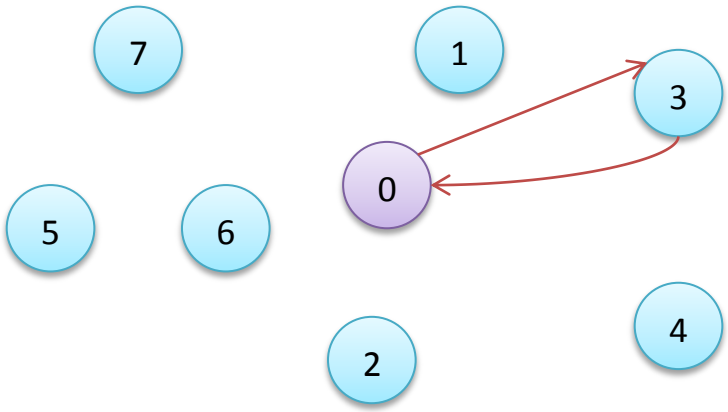
	0=Depósito
1	15,232
3	22,361
5	20,616
6	11,180
7	21,213
0	0,000

Fuente: Autores

Ruta parcial Nº 2: 0-3-0

La ruta que se origina al unir al depósito con el cliente semilla recién seleccionado se muestra a continuación en la Figura A1 - 9.

Figura A1 - 9. Ruta Parcial Nº2. Heurística de Inserción I1.



Fuente: Autores

Los clientes disponibles son: 1, 3, 5, 6 y 7, para ellos se evalúan las restricciones de capacidad y de tiempo, la Tabla A1 - 18 muestra el análisis de las restricciones.

Tabla A1 - 18. Análisis de restricciones

Cliente	Factible Restricciones de Capacidad		Factible Restricción de Tiempo (0-3)		Factible Restricción de Tiempo (3-0)	
	Si	No	Si	No	Si	No
1	*x*		*x*			*x*
5		*x*		*x*		*x*
6	*x*			*x*	*x*	
7	*x*			*x*		*x*

Fuente: Autores

Los clientes factibles son: el cliente 1 en la posición 0-3 y el cliente 6 en la posición 3-0, para ellos se muestra el cálculo de los criterios en la Tabla A1 - 19.

Tabla A1 - 19. Cálculo de los criterios C_{11} y C_{12}

u	C11 0-3	Wju	Wj	C12 0-3	C11 3-0	Wju	Wj	C12 3-0
1	7,43	58,56	50,00	8,56				
6					67,08	120,18	60,00	60,18

Fuente: Autores

Se halla el mínimo valor de C_1 del cliente u en todos los lugares de inserción, a continuación en la Tabla A1 - 20 se muestran los valores.

Tabla A1 - 20. Criterio de Inserción C_1 en la primera iteración

u	C1 0-3	C1 3-0
1	7,43	
6		67,08

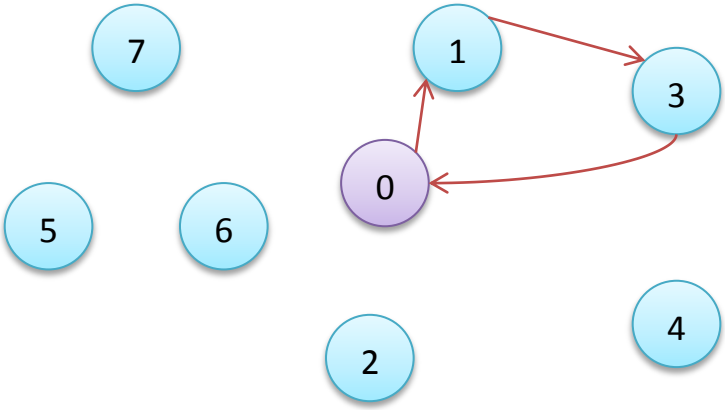
Fuente: Autores

El cliente con menor valor de C_1 es 1 en la posición 0-3. La ruta Parcial Nº2 se actualiza:

Ruta parcial Nº2: 0-1-3-0

La ruta parcial Nº2 se muestra gráficamente en la Figura A1 - 10, mediante la unión de los grafos involucrados.

Figura A1 - 10. Ruta Parcial Nº2. Heurística de Inserción I1.



Fuente: Autores

Los clientes restantes que cumplen restricciones de capacidad con la ruta actual son 6 y 7. Para ellos se examinan las restricciones de tiempo en la Tabla A1 - 21:

Tabla A1 - 21. Análisis de restricciones

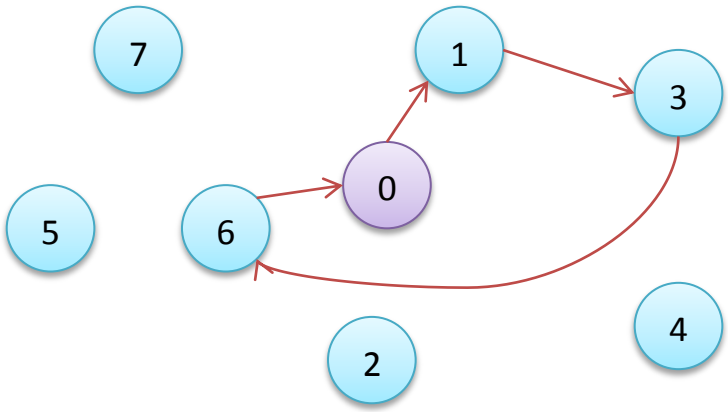
Cliente	Factible Restricciones de Capacidad		Factible Restricción de Tiempo (0-1)		Factible Restricción de Tiempo (1-3)		Factible Restricción de Tiempo (3-0)	
	Si	No	Si	No	Si	No	Si	No
6	*x*			*x*		*x*	*x*	
7	*x*			*x*		*x*		*X*

Fuente: Autores

El cliente 6 es el único que puede ser insertado debido a que cumple con las restricciones de capacidad y de tiempo. La ruta parcial se actualiza y se muestra en la Figura A1 - 11:

Ruta Nº2: 0-1-3-6-0

Figura A1 - 11. Ruta Nº2. Heurística de Inserción I1.



Fuente: Autores

La suma de las demandas de la ruta Nº2 es de 26 unidades. No es posible surtir ninguno de los clientes restantes (5 ó 7). La ruta finaliza y se da inicio a la siguiente.

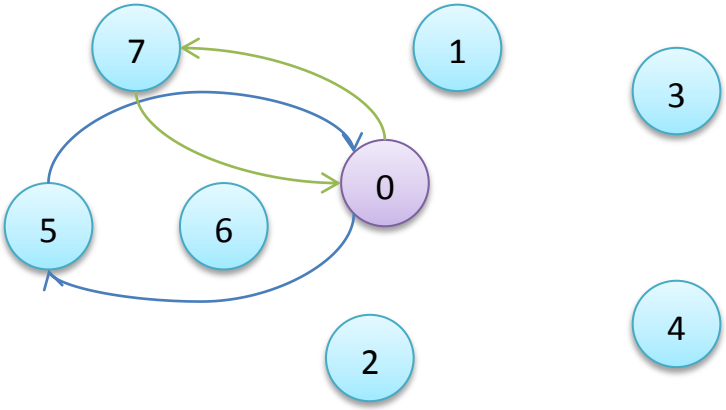
Tercera y Cuarta Rutas.

Las demandas de los clientes 5 y 7 sobrepasan la capacidad del vehículo. Por lo tanto cada uno se asigna a una ruta independiente y se muestra la unión de los respectivos grafos en la Figura A1 - 12.

Ruta Nº3: 0-5-0

Ruta Nº4: 0-7-0

Figura A1 - 12**.** Ruta Nº3 y Ruta Nº4. Heurística de Inserción I1.



Fuente: Autores

Para calcular el costo de la ruta, se asume que una unidad de distancia es igual a una unidad monetaria: $C_{ij} = \$1/_{und - distancia}$

El **COSTO TOTAL** de la ruta final es:

C=221,3 Unidades Monetarias

Los **TIEMPOS DE TOTALES DE LA RUTAS** o tiempos de programación acumulados de las rutas se observan en la Tabla A1 - 22:

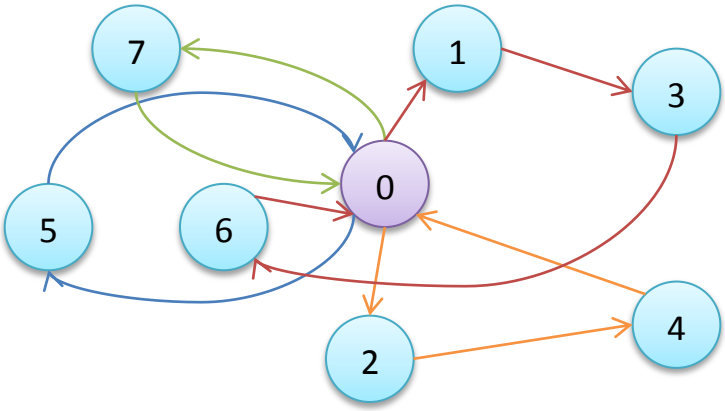
Tabla A1 - 22. Tiempos totales de las rutas.

Nº	Ruta	Tiempo acumulado (Unidades de tiempo)
1	0-2-4-0	132
2	0-1-3-6-0	123,3
3	0-5-0	117.6
4	0-7-0	112,2
Total		**485.1**

Fuente: Autores

En la Figura A1 - 13 se encuentra dibujada la solución del algoritmo de Inserción, cada ruta está representada con un color.

Figura A1 - 13. Solución gráfica del ejercicio propuesto. Heurística de inserción I1.



Fuente: Autores

ALGORITMO DE VECINO MÁS CERCANO

Un conjunto de siete clientes se encuentra disperso geográficamente según las coordenadas cartesianas *x* e *y*. En la Tabla A1 - 23 se aprecian las demandas, los inicios y fin de ventana de cada cliente.

Tabla A1 - 23. Datos iniciales

Cliente	X	Y	Demanda	ai	bi	si
0	35	35	0	0	230	0
1	41	49	10	34	44	10
2	35	17	7	32	42	10
3	55	45	13	50	60	10

4	55	20	19	97	107	10
5	15	30	26	87	98	10
6	25	30	3	99	111	10
7	20	50	5	81	91	10
			83			

Fuente: Autores

La heurística inicia tomando al cliente más cercano al depósito y continúa en cada iteración buscando el más cercano al último cliente de la ruta parcial.

Para determinar qué tan cercanos están dos clientes, se hace uso de una métrica que pondera tanto la cercanía geográfica como la temporal de los clientes:

Métrica de la Heurística del Vecino más Cercano: $\boldsymbol{\delta_1 d_{ij} + \delta_2 T_{ij} + \delta_3 V_{ij}}$

Donde:

- d_{ij}: Es la distancia euclidiana que existe entre los cliente i, j

- T_{ij}: Es la diferencia de tiempo que se genera al terminar el servicio en el cliente i y comenzar el servicio en el cliente j, expresado como:

$$T_{ij} = W_j - (W_i + S_i)$$

- V_{ij}: Es la urgencia de servir al cliente j, dicho de otra forma, el tiempo disponible para servir al cliente j, expresado como:

$$V_{ij} = b_j - (W_i + S_i + t_{ij})$$

δ_1, δ_2 y δ_3 son parámetros que pesan cada término de la métrica y definen la relación de prioridades entre ellos.

La matriz de distancias para el ejercicio propuesto se muestra a continuación en la Tabla A1 - 24. La columna de clientes de la izquierda son los nodos de partida y los nodos de llegada se encuentran en la fila superior:

Tabla A1 - 24. Matriz de distancias

	1	**2**	**3**	**4**	**5**	**6**	**7**	**0**
1	0,000	32,558	14,560	32,202	32,202	24,839	21,024	15,232
2	32,558	0,000	34,409	20,224	23,854	16,401	36,249	18,000
3	14,560	34,409	0,000	25,000	42,720	33,541	35,355	22,361
4	32,202	20,224	25,000	0,000	41,231	31,623	46,098	25,000
5	32,202	23,854	42,720	41,231	0,000	10,000	20,616	20,616
6	24,839	16,401	33,541	31,623	10,000	0,000	20,616	11,180
7	21,024	36,249	35,355	46,098	20,616	20,616	0,000	21,213
0	15,232	18,000	22,361	25,000	20,616	11,180	21,213	0,000

Fuente: Autores

Para el desarrollo del ejercicio, Los parámetros δ_1, δ_2 y δ_3 se definen en la Tabla A1 - 25:

Tabla A1 - 25. Parámetros vecino más cercano

$\boldsymbol{\delta_1}$	**0,3**
$\boldsymbol{\delta_2}$	0,3
$\boldsymbol{\delta_3}$	0,4

Fuente: Autores

Primera Ruta

Se calcula la métrica del vecino más cercano para todos los clientes con respecto al depósito:

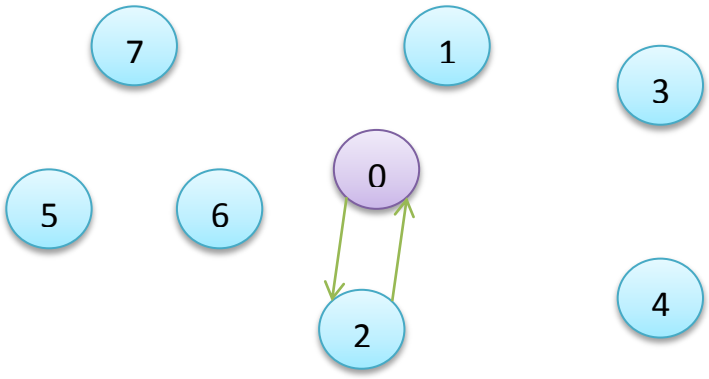
$$C_{01} = (0{,}3)(15{,}2) + (0{,}3)(15{,}2) + (0{,}4)(44 - 15{,}2) = \mathbf{20{,}64}$$
$$C_{02} = (0{,}3)(18) + (0{,}3)(18) + (0{,}4)(42 - 18) = \mathbf{20{,}4}$$
$$C_{03} = (0{,}3)(22{,}4) + (0{,}3)(22{,}4) + (0{,}4)(60 - 22{,}4) = \mathbf{28{,}48}$$
$$C_{14} = (0{,}3)(25) + (0{,}3)(25) + (0{,}4)(107 - 25) = \mathbf{47{,}8}$$
$$C_{05} = (0{,}3)(20{,}6) + (0{,}3)(20{,}6) + (0{,}4)(98 - 20{,}6) = \mathbf{43{,}32}$$
$$C_{06} = (0{,}3)(11{,}2) + (0{,}3)(11{,}2) + (0{,}4)(111 - 11{,}2) = \mathbf{46{,}64}$$
$$C_{07} = (0{,}3)(21{,}2) + (0{,}3)(21{,}2) + (0{,}4)(91 - 21{,}2) = \mathbf{97{,}88}$$

El costo menor de la métrica C corresponde al cliente 2. Se inserta el cliente 2 en la ruta parcial:

Ruta parcial Nº1: 0-2-0

En la Figura A1 - 14 se muestra la ruta parcial que surge con el enlace del cliente 2 quien inicia la ruta Nº1.

Figura A1 - 14. Ruta Parcial Nº1. Algoritmo del vecino más cercano.



Fuente: Autores

En la Tabla A1 - 26 se analizan las restricciones de tiempo y capacidad de los clientes factibles que pueden insertarse partiendo del cliente 2:

Tabla A1 - 26. Análisis de restricciones

Cliente	Factible Restricciones de Capacidad		Factible Restricción Temporal	
	Si	No	Si	No
1	x			x
3	x			x
4	x		x	
5		x	x	
6	x		x	
7	x		x	

Fuente: Autores

Se calcula la métrica de selección para los clientes factibles:

$$C_{24} = (0{,}3)(20{,}2) + (0{,}3)(97 - 32 - 10) + (0{,}4)(107 - 32 - 10 - 20{,}2) = \mathbf{40{,}48}$$

$$C_{26} = (0{,}3)(16{,}4) + (0{,}3)(99 - 32 - 10) + (0{,}4)(111 - 32 - 10 - 16{,}4) = \mathbf{43{,}6}$$

$$C_{27} = (0{,}3)(36{,}2) + (0{,}3)(81 - 32 - 10) + (0{,}4)(91 - 32 - 10 - 36{,}2) = \mathbf{27{,}68}$$

El cliente 7 posee el menor valor de C_{2j}. En la Tabla A1 - 27 se analiza las restricciones de capacidad y de ventanas de tiempo para el cliente 7 respecto al cliente 2:

Tabla A1 - 27. Análisis de restricciones grafo 2-7.

Restricción Capacidad	Las demandas del cliente 2 y 7 suman 12 unidades, menor que la capacidad del vehículo de 30 unidades. Por esta razón, el cliente 7 se considera factible para insertarse en la ruta.
Restricción Temporal	Como $T + d_{27} < a_7$ se incurre en un tiempo de espera, lo que hace que el inicio de servicio en el cliente 7 sea igual al limite inferior de su ventana: $w_7 = a_7 = 81$. El vehículo presta el servicio S_7 y el nuevo tiempo actual de recorrido se actualiza: $T = w_7 + S_2 = 32 + 10 = 91$

Fuente: Autores

Se observa que al insertar el cliente 7 después del cliente 2 en la ruta parcial, se respetan las restricciones de capacidad y de ventanas de tiempo. Se actualiza la ruta parcial:

Ruta parcial Nº1:0-2-7-0

En la siguiente iteración, se examina la factibilidad de los clientes disponibles; en la Tabla A1 - 28 se realiza el análisis de las restricciones:

Tabla A1 - 28**.** Análisis de restricciones

Cliente	Factible Restricciones de Capacidad		Factible Restricción Temporal	
	Si	No	Si	No
1	x			x
3	x			x
4		x		x
5		x		x
6	x			x

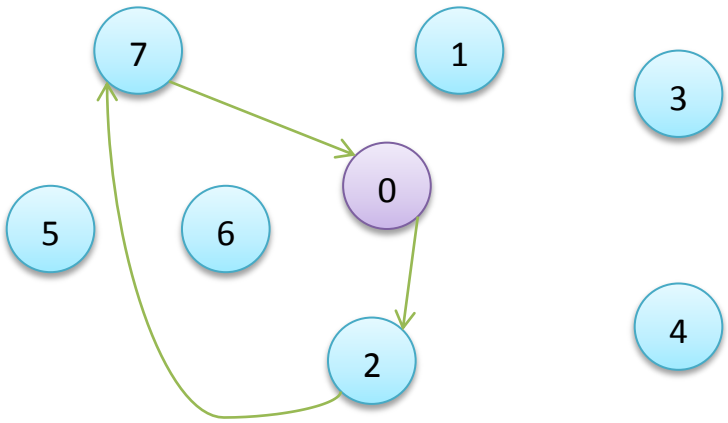
Fuente: Autores

Ninguno de los clientes disponibles se considera factible. Se finaliza la ruta Nº 1:

Ruta Nº1: 0-2-7-0

En la Figura A1 - 15 se muestra la ruta Nº1 que surge con las primeras iteraciones del algoritmo de vecino más cercano.

Figura A1 - 15. Ruta Nº1. Algoritmo del vecino más cercano.



Fuente: Autores

El tiempo de recorrido de la ruta Nº1 es de 112,2 unidades de tiempo considerando la distancia del cliente 7 al depósito.

Segunda Ruta.

Se calcula la métrica C desde el depósito hasta cada uno de los clientes disponibles.

$$C_{01} = (0{,}3)(15{,}2) + (0{,}3)(15{,}2) + (0{,}4)(44 - 15{,}2) = \mathbf{20{,}64}$$
$$C_{03} = (0{,}3)(22{,}4) + (0{,}3)(22{,}4) + (0{,}4)(60 - 22{,}4) = \mathbf{28{,}48}$$
$$C_{14} = (0{,}3)(25) + (0{,}3)(25) + (0{,}4)(107 - 25) = \mathbf{47{,}8}$$
$$C_{05} = (0{,}3)(20{,}6) + (0{,}3)(20{,}6) + (0{,}4)(98 - 20{,}6) = \mathbf{43{,}32}$$
$$C_{06} = (0{,}3)(11{,}2) + (0{,}3)(11{,}2) + (0{,}4)(111 - 11{,}2) = \mathbf{46{,}64}$$
$$C_{07} = (0{,}3)(21{,}2) + (0{,}3)(21{,}2) + (0{,}4)(91 - 21{,}2) = \mathbf{97{,}88}$$

Se considera al cliente 1 como el mejor para iniciar la ruta Nº 2 porque arroja el menor valor de la métrica C. En la Tabla A1 - 29 se analizan las restricciones para el cliente 1.

Tabla A1 - 29. Análisis de restricciones cliente 1

Restricción Capacidad	La demanda del cliente 1 es de 10 unidades, menor que la capacidad del vehículo de 30 unidades, por lo tanto, El cliente 1 se considera factible para insertarse en la ruta.
Restricción Temporal	Por el concepto de desigualdad triangular, el cliente 1 se considera factible por ventanas de tiempo y distancia desde el depósito. Como $d_{01} < a_1$, se incurre en un tiempo de espera, lo que hace que el inicio de servicio en el cliente 1 sea igual al limite inferior de su ventana: $w_1 = a_1 = 34$. El vehículo presta el servicio s y el tiempo actual de recorrido se obtiene de: $T = w_1 + s_1 = 34 + 10 = 44$

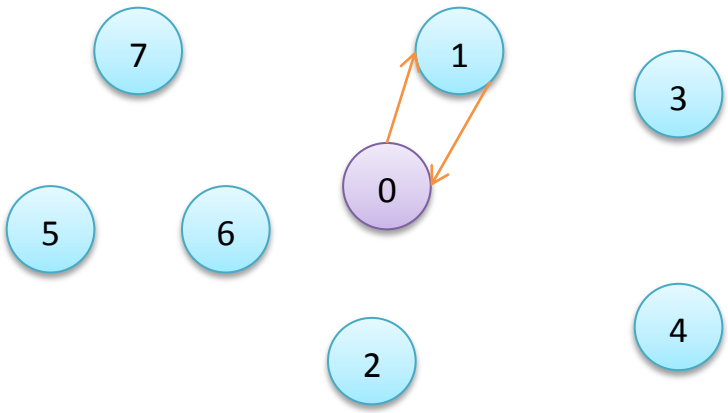
Fuente: Autores

Dado que se respetan las restricciones de capacidad y de tiempo, se selecciona al cliente 1 para iniciar la segunda ruta parcial:

Ruta parcial Nº 2: 0-1-0

El cliente 1 quien inicia la ruta Nº2 conforma la ruta parcial que se muestra en la Figura A1 - 16.

Figura A1 - 16. Ruta Parcial Nº2. Algoritmo del vecino más cercano.



Fuente: Autores

Se verifica el cumplimiento de las restricciones de los clientes que aún no han sido visitados en la Tabla A1 - 30.

Tabla A1 - 30. Análisis de restricciones.

Cliente	Factible Restricciones de Capacidad		Factible Restricción Temporal	
	Si	No	Si	No
3	*x*		*x*	
4	*x*		*x*	
5		*x*	*x*	
6	*x*		*x*	

Fuente: Autores

En la siguiente iteración, se calcula la métrica para los clientes factibles que se encuentran disponibles:

$$C_{13} = (0{,}3)(14{,}6) + (0{,}3)(58{,}6 - 34 - 10) + (0{,}4)(60 - 34 - 10 - 14{,}6) = \mathbf{9{,}32}$$

$$C_{14} = (0{,}3)(32{,}2) + (0{,}3)(97 - 34 - 10) + (0{,}4)(107 - 34 - 10 - 32{,}2) = \mathbf{37{,}88}$$

$$C_{15} = (0{,}3)(32{,}2) + (0{,}3)(87 - 34 - 10) + (0{,}4)(98 - 34 - 10 - 32{,}2) = \mathbf{31{,}28}$$

$$C_{16} = (0{,}3)(24{,}8) + (0{,}3)(99 - 34 - 10) + (0{,}4)(111 - 34 - 10 - 24{,}8) = \mathbf{40{,}82}$$

El menor valor de la métrica C corresponde al cliente 3. Se procede a analizar las restricciones del cliente 3 respecto al cliente 1 en la Tabla A1 - 31.

Tabla A1 - 31. Análisis de restricciones grafo 1-3.

Restricción Capacidad	Las demandas del cliente 1 y 3 suman 23 unidades, menor que la capacidad del vehículo de 30 unidades. Por esta razón el cliente 3 se considera factible para insertarse en la ruta.

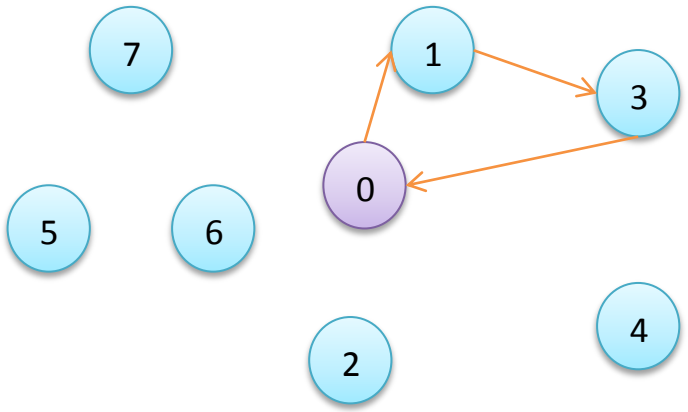
Restricción Temporal	Como $a_7 < T + d_{13} < b_7$ el inicio de servicio en el cliente 3 es igual al tiempo de recorrido acumulado hasta ese momento: $w_3 = 58{,}6$. El vehículo presta el servicio S_3 y el nuevo tiempo actual de recorrido se actualiza: $T = w_1 + S_2 = 58{,}6 + 10 = 68{,}6$

Fuente: Autores

El cliente tres cumple con las restricciones de capacidad y de tiempo al ser insertado junto al cliente 1. La ruta parcial queda actualizada y se muestra en la Figura A1 - 17:

Ruta parcial Nº 2: 0-1-3-0

Figura A1 - 17. Ruta Parcial Nº2. Algoritmo del vecino más cercano.



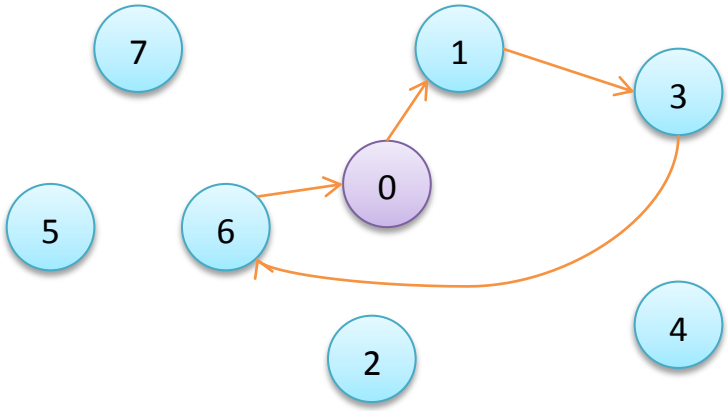
Fuente: Autores

La suma de las demandas de los clientes de la ruta parcial es de 23 unidades. De los clientes disponibles, el único que podría ingresar a la ruta sin sobrepasar la capacidad del vehículo es el cliente 6 (3 unidades). Además, al ser insertado en la ruta, el cliente 6 cumple con la restricción de tiempo, (Tiempo acumulado= 68,6;

Inicio de ventana cliente 6 = 99) por lo tanto se selecciona para ser servido en la ruta Nº 2, y se muestra en la Figura A1 - 18:

Ruta Nº 2: 0-1-3-6-0

Figura A1 - 18. Ruta Nº2. Algoritmo del vecino más cercano.



Fuente: Autores

De esta manera se finaliza la ruta Nº 2, El tiempo de recorrido de la ruta es de 120,2 unidades de tiempo, considerando el regreso hacia el depósito.

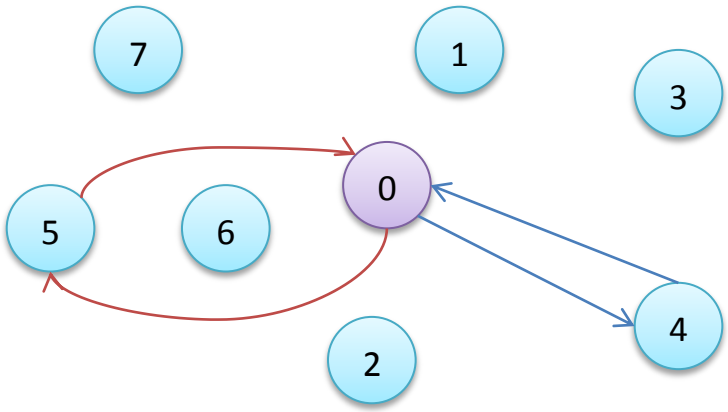
Tercera y Cuarta Ruta

Los clientes restantes son 4 y 5. Como la suma de sus demandas sobrepasa la capacidad de un vehículo, deben servirse en rutas separadas como se muestra en la Figura A1 - 19:

Ruta Nº 3: 0-4-0

Ruta Nº 2: 0-5-0

Figura A1 - 19. Ruta Nº3 y Ruta Nº4. Algoritmo del vecino más cercano.



Fuente: Autores

Para calcular el costo de la ruta, se asume que una unidad de distancia es igual a una unidad monetaria: $C_{ij} = \$1/_{und - distancia}$

El **COSTO TOTAL** de la ruta final es:

C=241,1 Unidades monetarias

Los **TIEMPOS DE TOTALES DE LA RUTAS** o tiempos de programación acumulados de las rutas se observa en la Tabla A1 - 32:

Tabla A1 - 32. Tiempos totales de las rutas.

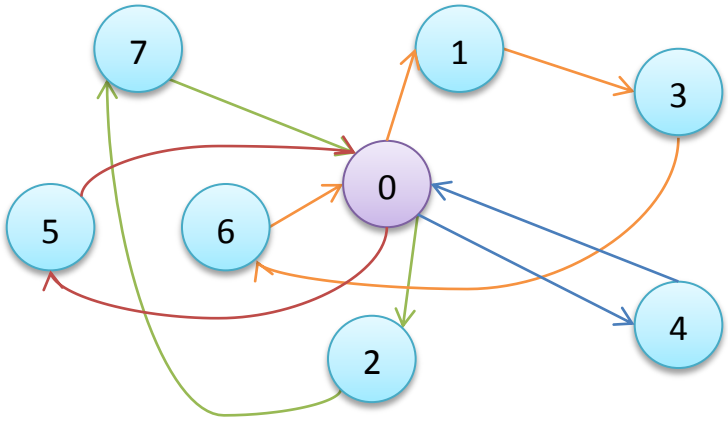
Nº	Ruta	Tiempo acumulado (Unidades de tiempo)
1	0-2-7-0	112.2
2	0-1-3-6-0	120.2

3		0-4-0	132
4		0-5-0	117.6
Total			**482**

Fuente: Autores

Luego de realizar el procedimiento del algoritmo de vecino más cercano se dibujan las rutas resultantes en la Figura A1 - 20. Solución gráfica del ejercicio propuesto. Algoritmo del vecino más cercano..

Figura A1 - 20. Solución gráfica del ejercicio propuesto. Algoritmo del vecino más cercano.



Fuente: Autores

Análisis de Resultados

En el ejemplo desarrollado, se observa que las heurísticas con mejor desempeño como se identifica en la Tabla A1 - 33. Resultados heurísticas de construcción de rutas. son la de Clarke-Wright y la de Inserción I1, debido al menor valor de costo, que coincide para este caso en 221.3 unidades. El tiempo de programación de las rutas es de 485.1 unidades de tiempo para las dos heurísticas.

A pesar de que la heurística del vecino más cercano programa las rutas en menor tiempo (482 unidades de tiempo), el desempeño de la función de costo es mejor en comparación con los otros dos algoritmos respecto a la distancia recorrida en las rutas. (241,1 unidades).

La función de costo representada por la suma de los tiempos de recorrido totales de las rutas, es el factor principal para medir el desempeño de la heurística. El tiempo de programación de rutas y el número total de rutas, son los otros dos factores de interés.

Tabla A1 - 33. Resultados heurísticas de construcción de rutas.

HEURÍSTICA	COSTO (unidades monetarias)	TIEMPO TOTAL (unidades de tiempo)	Número Vehículos/Rutas
Clarke y Wright	221.3	485.1	4
Inserción I1	221.3	485.1	4
Vecino más Cercano	241.1	482	4

Fuente: Autores

ANEXO B. MANUAL DE USUARIO - HERRAMIENTA PARA LA SOLUCIÓN DEL VRPTW.

A continuación se describe de manera breve el uso de la herramienta desarrollada para resolver el problema de ruteo con ventanas de tiempo (VRPTW) a través de heurísticas de construcción de rutas.

Instalación.

El software VRPTW se diseñó con el propósito de resolver el problema de ruteo con ventanas de tiempo usando heurísticas de construcción de rutas aplicadas a las instancias propuestas en Solomon (1987) con 100 y 50 clientes. Igualmente, el toolbox permite cargar datos propios de heurísticas pequeñas y de instancias de más de 100 clientes.

Es importante aclarar que el usuario que desee ingresar datos propios, debe tener claro los conceptos del problema, desde su definición y formulación matemática. Siguiendo la lógica y estructura del modelo VRPTW y de las instancias de la literatura de Solomon, se garantiza no generar datos que alteren los principios básicos del problema, como cuando se ingresan datos al azar.

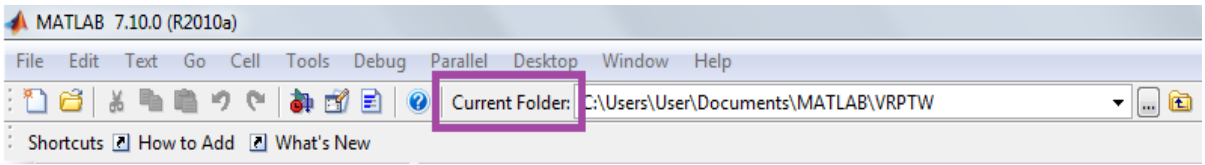
Para usar la herramienta, es necesaria la instalación del software Matlab® versión 2009 o superior en un equipo de cómputo estándar que soporte las operaciones del programa.

La carpeta VRPTW contiene los códigos libres de las heurísticas de construcción de rutas (Ahorros, vecino más cercano, e inserción 1, 2 y 3), así como la programación que enlaza y estructura el entorno gráfico del software (Se recomienda no editar los códigos de inicialización del programa). Igualmente

incluye los datos en formato .xls de las instancias de Solomon (1987) de 100 y 50 clientes, que se cargan automáticamente al programa cuando se ejecuta. Las funciones contenidas en la carpeta son: “distancias”, “vecinos” y “ventanas”, que se explican detalladamente el anexo “Códigos de programación”.

La carpeta debe copiarse en la ubicación que Matlab® genera por defecto, o donde el usuario lo desee, cuidando de actualizar su dirección en el campo “*Current Folder”* cuando se abra el software.

Figura A2 - 1. Ubicación de la carpeta.

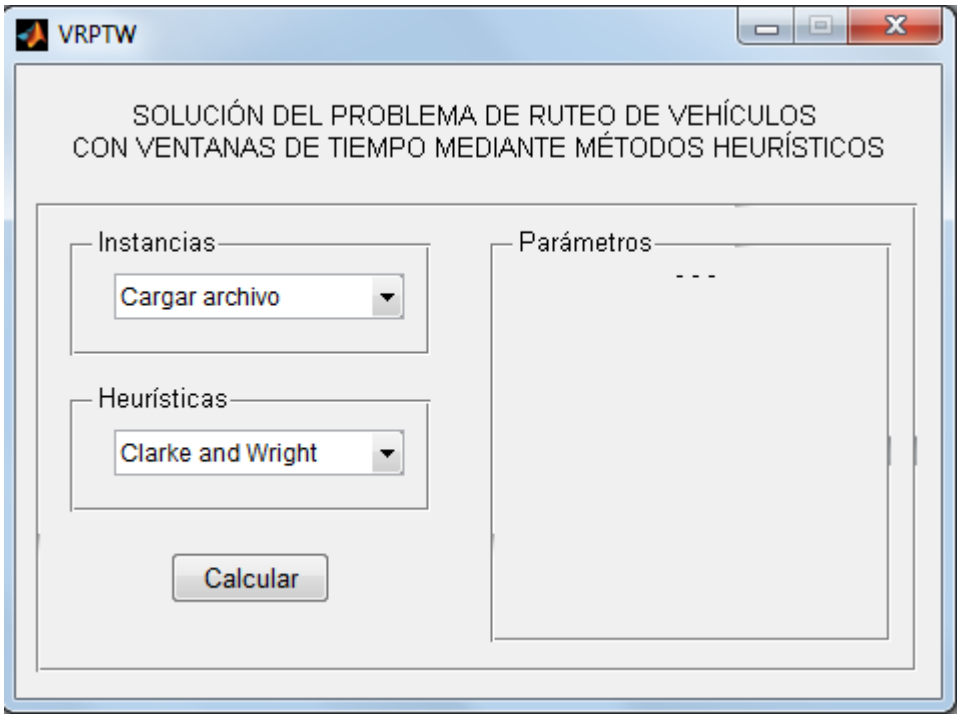


Fuente: Autores.

El programa se ejecuta escribiendo en la ventana “*Command Window*” la sigla HCR (Heurísticas de Construcción de Rutas) y oprimiendo la tecla “*Enter*”. De otra manera se puede ejecutar haciendo doble clic en el archivo HCR tipo MATLAB M-file con formato .m 

. Una vez abierto Matlab®, se hace clic en el botón “*Run*”  . Se despliega el menú de inicio del toolbox, que se muestra a continuación.

Figura A2 - 2. Ventana de Inicio.

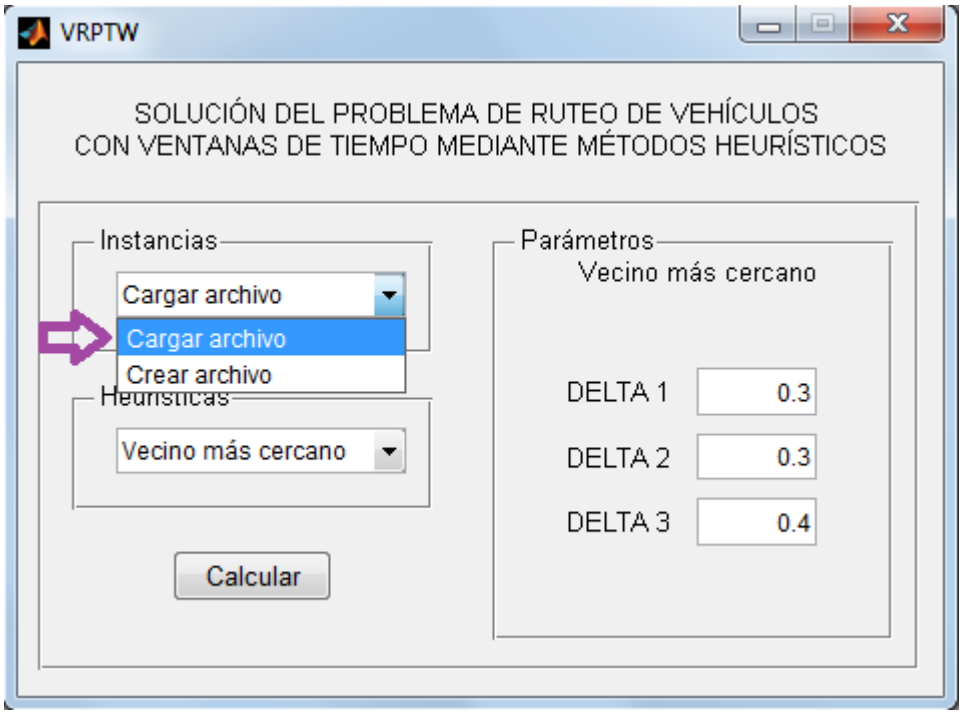


Fuente: Autores.

Cargar Instancias.

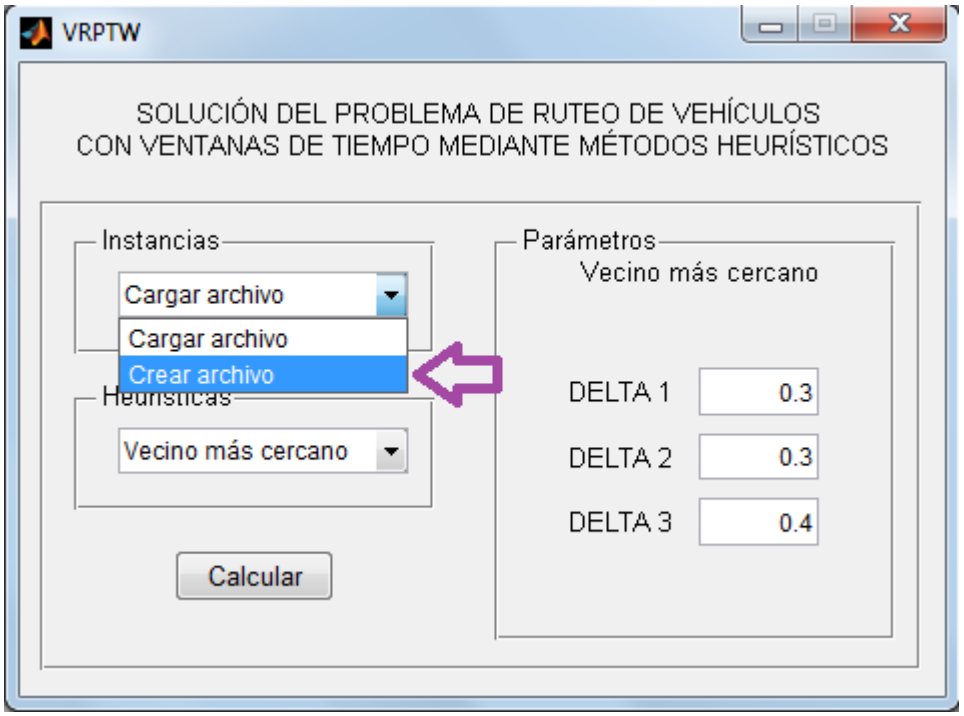
El primer paso para el uso del toolbox es cargar la matriz de datos. Si se desean correr las instancias predeterminadas, se hace clic en la opción “Cargar archivo” en el menú desplegable “Instancias”. Por el contrario, si se desea cargar una matriz de datos propia, se hace clic en la opción “Crear archivo”.

Figura A2 - 3. Opción Cargar Archivo.



Fuente: Autores.

Figura A2 - 4. Opción Crear Archivo.

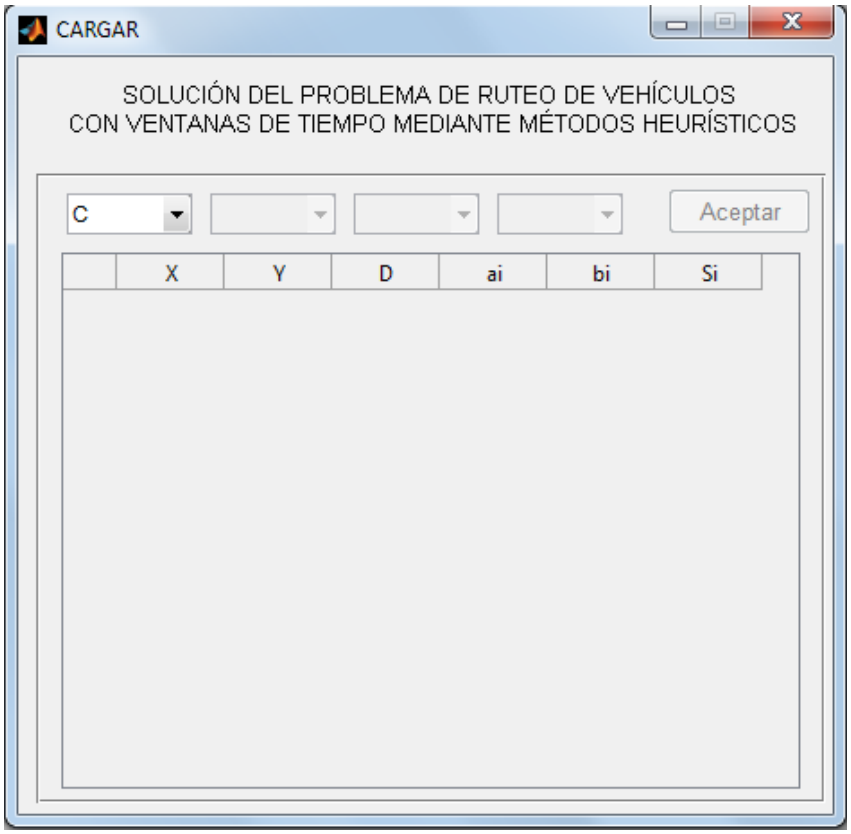


Fuente: Autores.

Opción “Cargar archivo”.

La opción “Cargar archivo”, despliega la ventana “Cargar”, que permite configurar la matriz de datos de la instancia de Solomon que se desee usar.

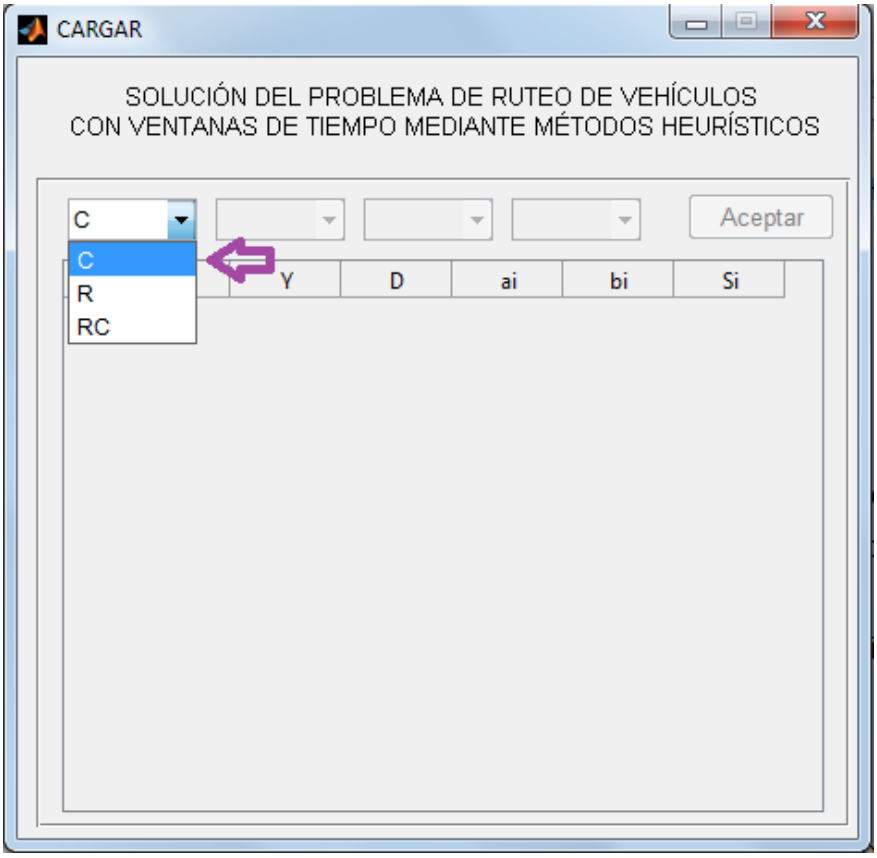
Figura A2 - 5. Ventana “Cargar”.



Fuente: Autores.

Inicialmente, la ventana “Cargar” activa un menú que indica al usuario la naturaleza de la instancia que se va a correr: tipo C (Clúster), R (Random) o RC (Random-Clúster). Para el ejemplo, se correrá la instancia C103.50 de Solomon.

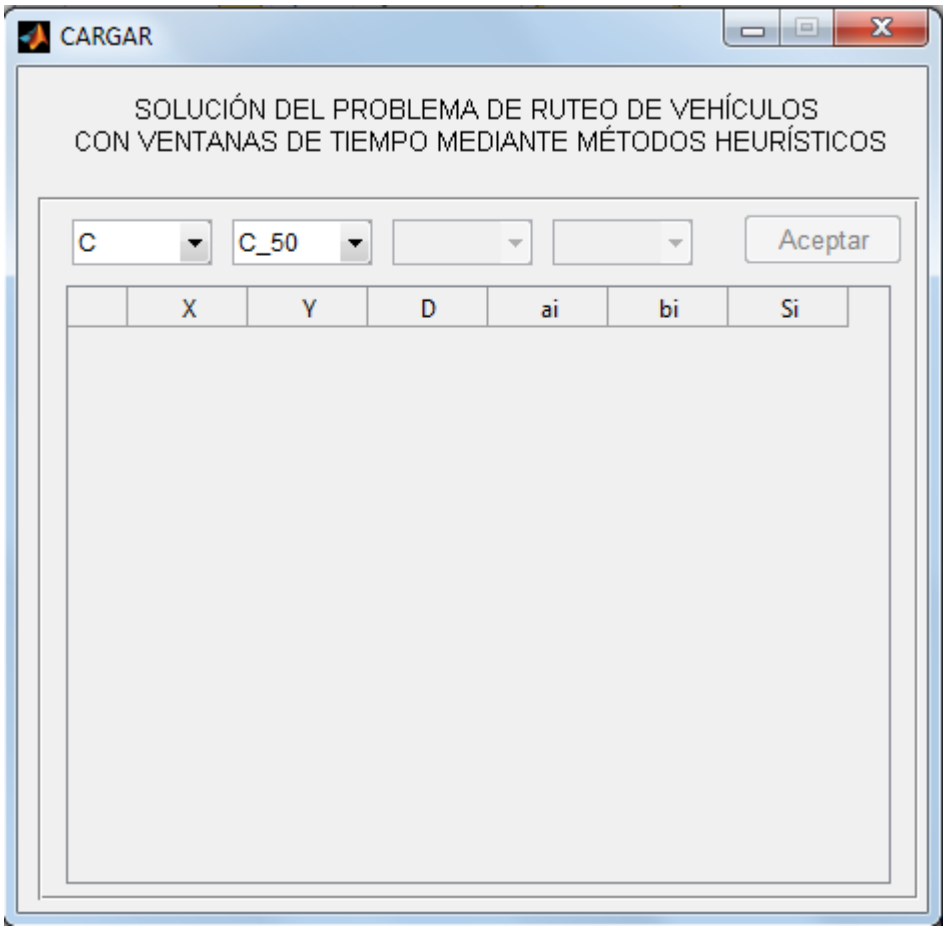
Figura A2 - 6. Opción: Tipo de instancia.



Fuente: Autores.

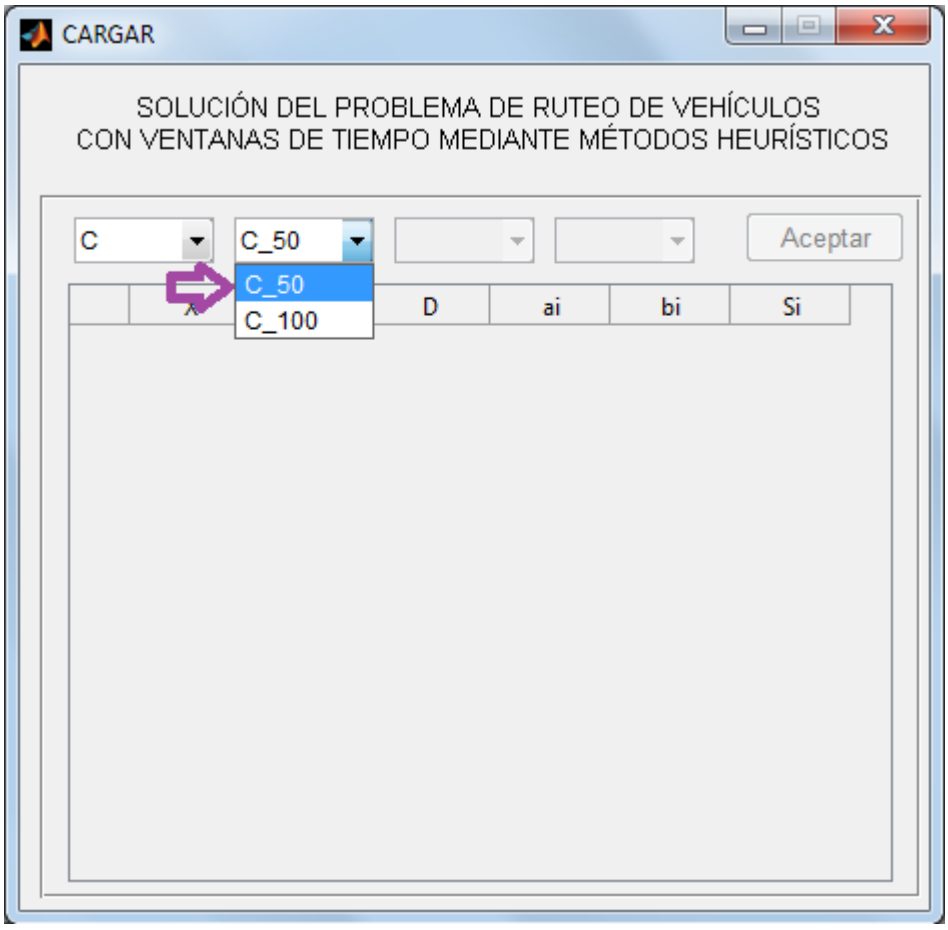
Al hacer clic en la opción C del primer menú desplegable, se activa el siguiente menú que permite seleccionar la cantidad de clientes del conjunto, 100 o 50 en instancias predeterminadas. Para este caso, se selecciona la opción C_50 con un clic.

Figura A2 - 7. Opción: Cantidad de clientes.



Fuente: Autores.

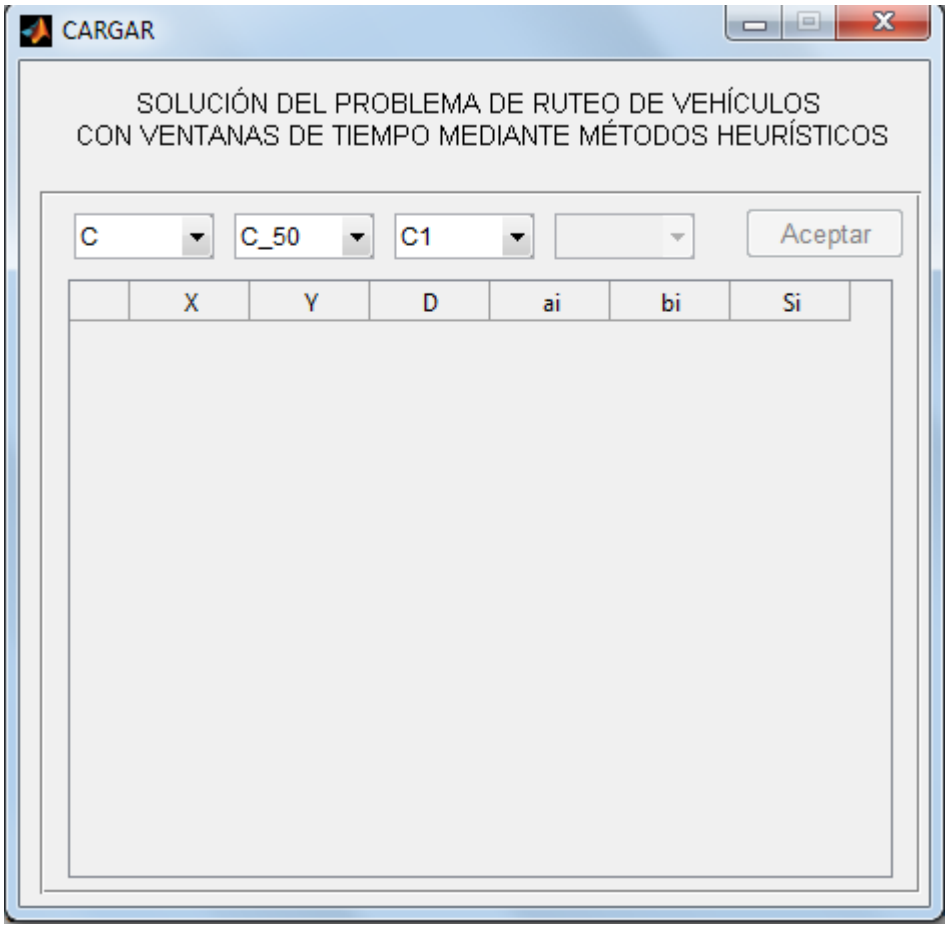
Figura A2 - 8. Selección instancias C_50.



Fuente: Autores.

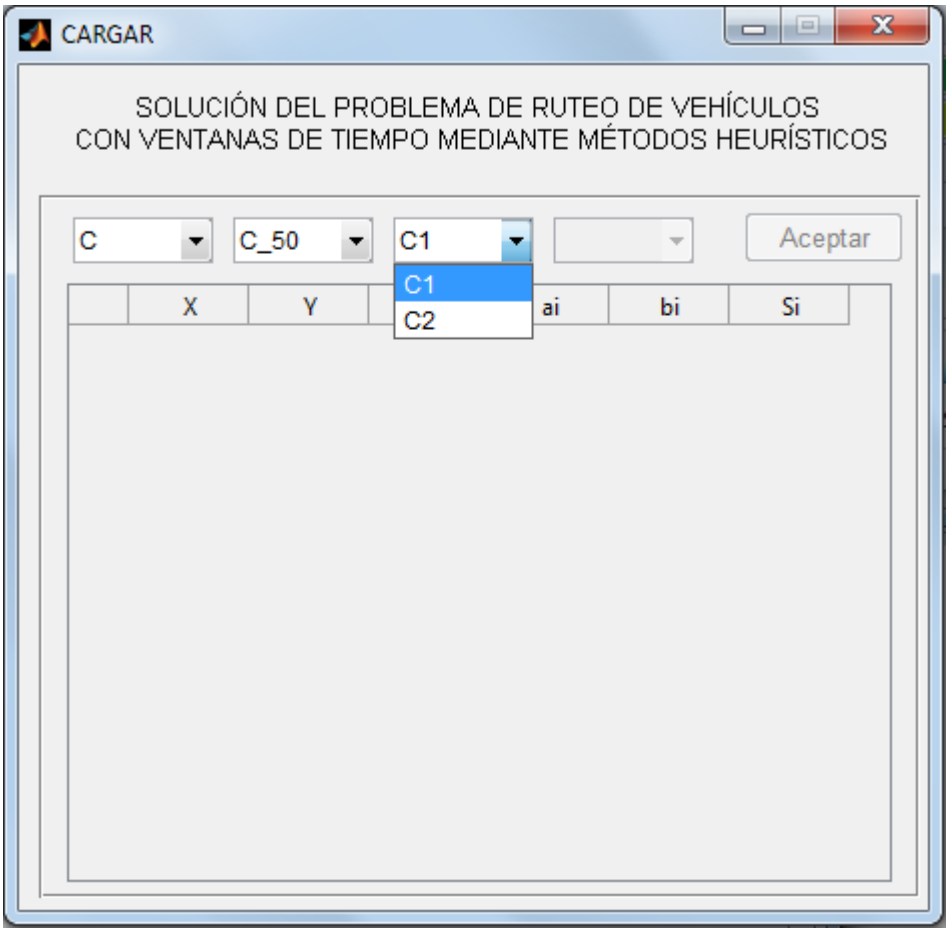
Cuando se selecciona la cantidad de clientes (C_50), se activa el menú que permite elegir el horizonte de programación de las rutas: C1, con horizonte de programación corto, o C2 con horizonte de programación largo. Se hace clic en la opción C1.

Figura A2 - 9. Selección Horizonte de Programación.



Fuente: Autores.

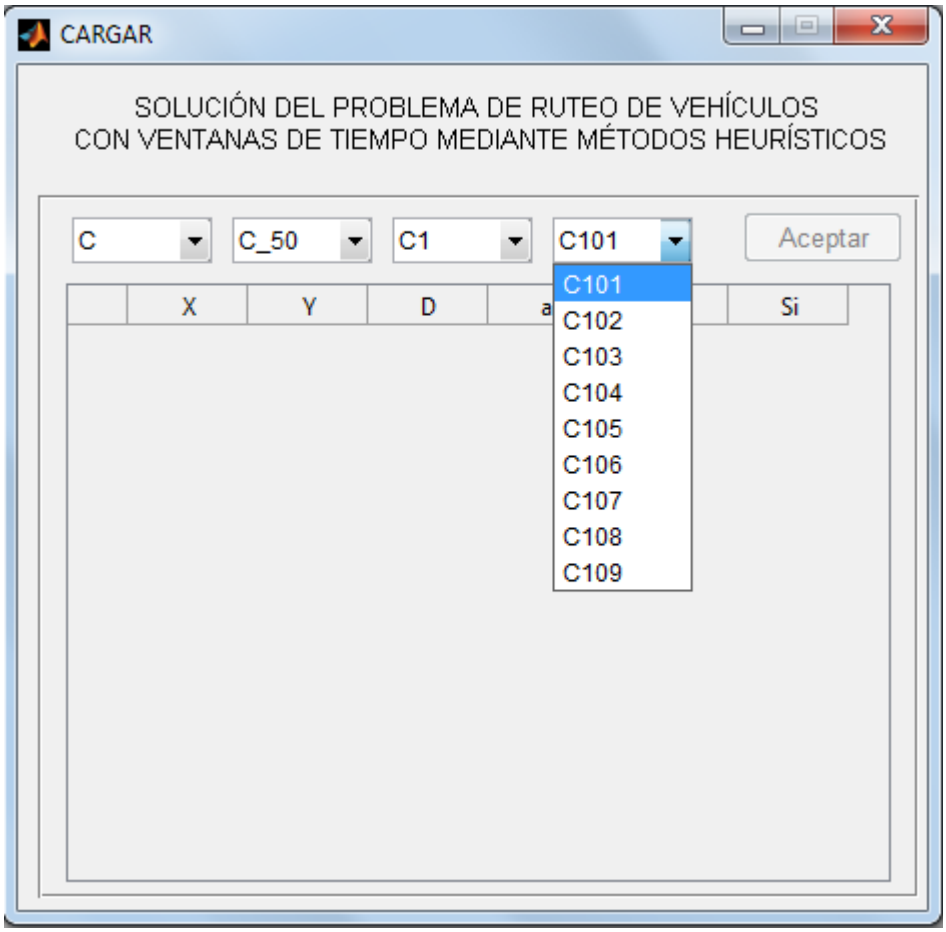
Figura A2 - 10. Selección Horizonte de Programación C1.



Fuente: Autores.

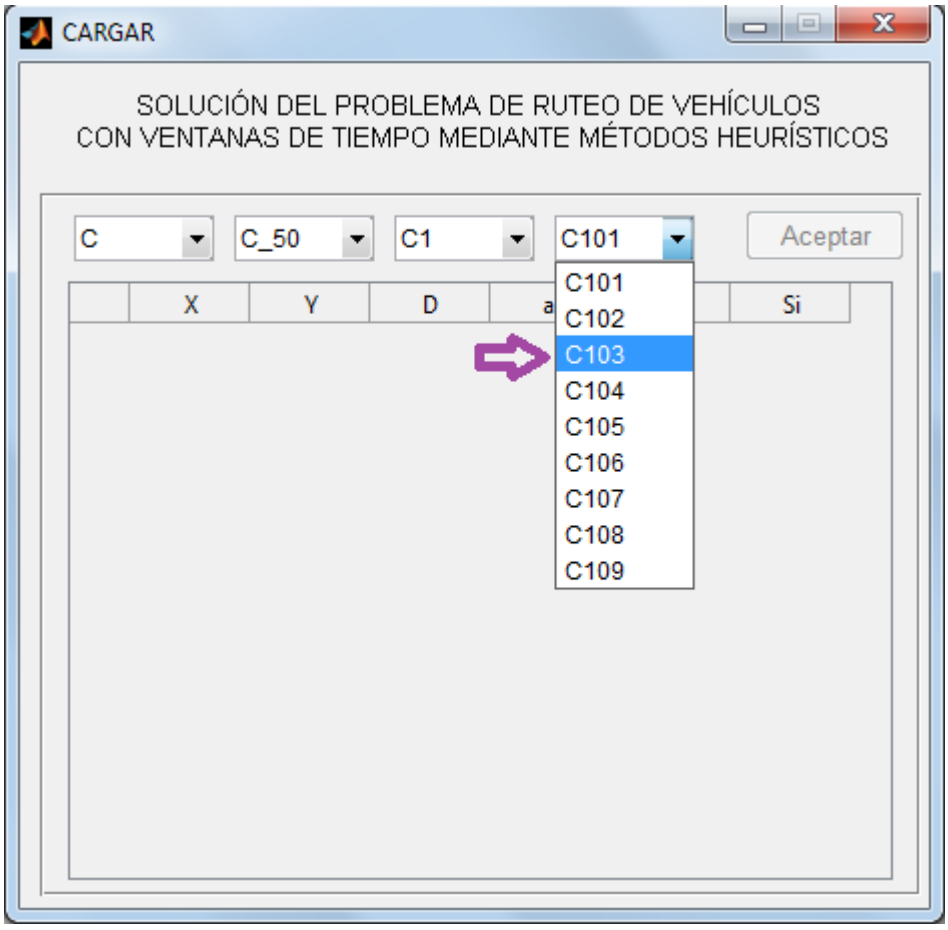
Una vez seleccionado el horizonte de programación de las rutas (C1 en este caso), se activa el último menú que contiene el conjunto de problemas pertenecientes a la configuración de instancias elegida. Para la configuración C1.50, existen 9 instancias. Se selecciona la tercera opción C103.

Figura A2 - 11. Selección del grupo de problemas.



Fuente: Autores.

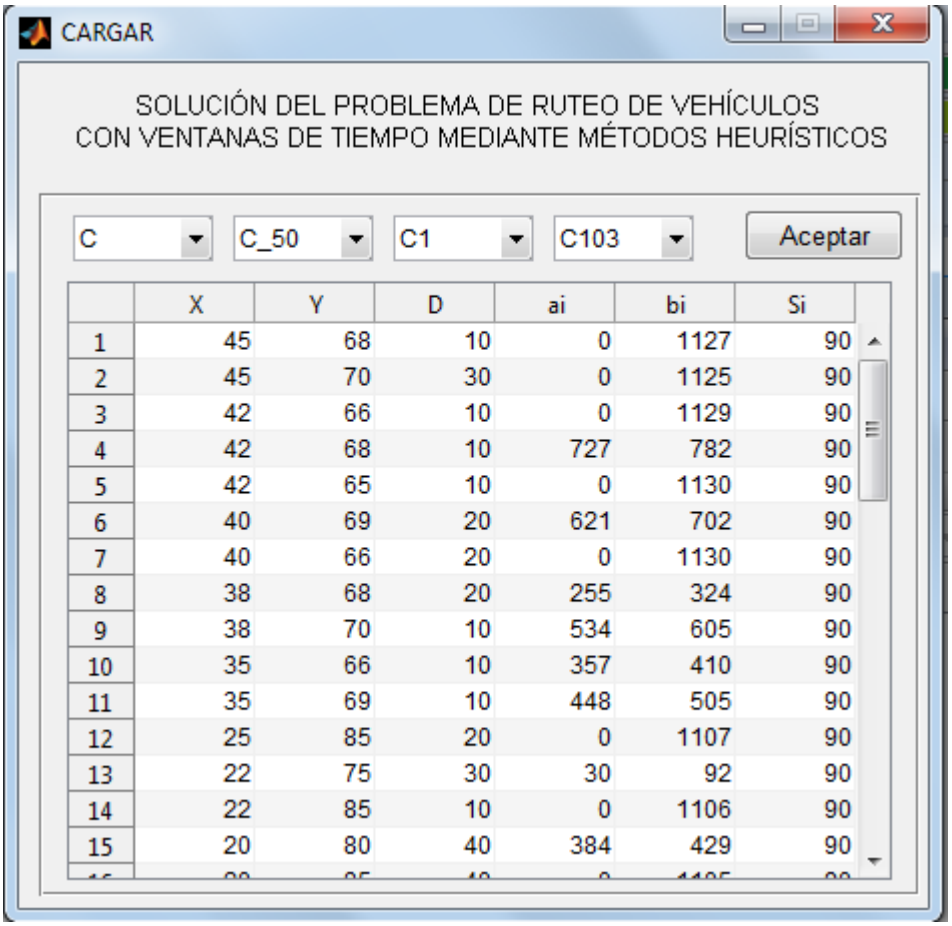
Figura A2 - 12. Selección instancia C103.



Fuente: Autores.

Cuando se hayan activado todas las opciones, se despliega la matriz de datos correspondiente a la instancia seleccionada. Las filas de la matriz indican el número de clientes con sus datos respectivos en este orden: **X**, **Y**, **D**, **ai**, **bi**, **Si**. Cada dato es una columna de la matriz. **X** e **Y**, son las coordenadas geográficas de los clientes, **D** es la demanda, **ai** es el inicio de la ventana de tiempo, **bi** es el final de la ventana de tiempo y **Si** es el tiempo de servicio en cada cliente.

Figura A2 - 13. Matriz de datos.



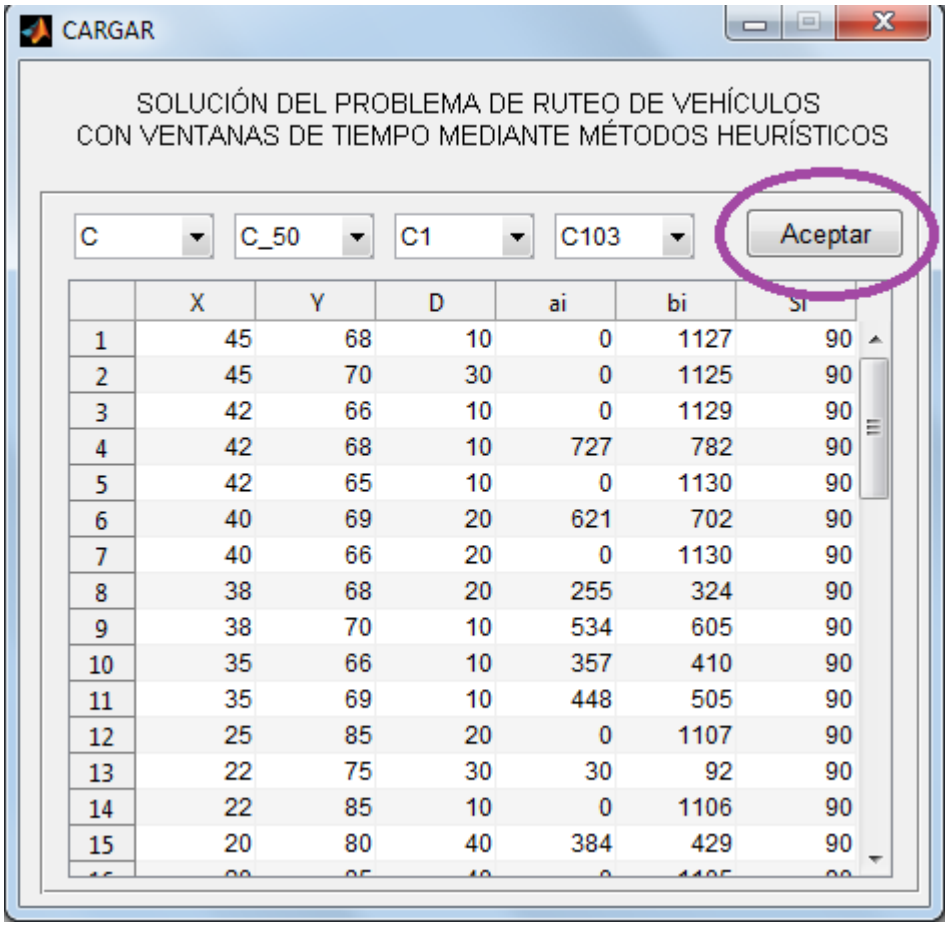
	X	Y	D	ai	bi	Si
1	45	68	10	0	1127	90
2	45	70	30	0	1125	90
3	42	66	10	0	1129	90
4	42	68	10	727	782	90
5	42	65	10	0	1130	90
6	40	69	20	621	702	90
7	40	66	20	0	1130	90
8	38	68	20	255	324	90
9	38	70	10	534	605	90
10	35	66	10	357	410	90
11	35	69	10	448	505	90
12	25	85	20	0	1107	90
13	22	75	30	30	92	90
14	22	85	10	0	1106	90
15	20	80	40	384	429	90

Fuente: Autores.

Los datos se pueden verificar con la barra de desplazamiento que aparece a la izquierda de la ventana. Los datos de capacidad se incluyen automáticamente en el momento en que se configura toda la instancia.

Una vez verificados los datos, se activa el botón “Aceptar” que carga finalmente la matriz de datos seleccionada.

Figura A2 - 14. Botón Aceptar.



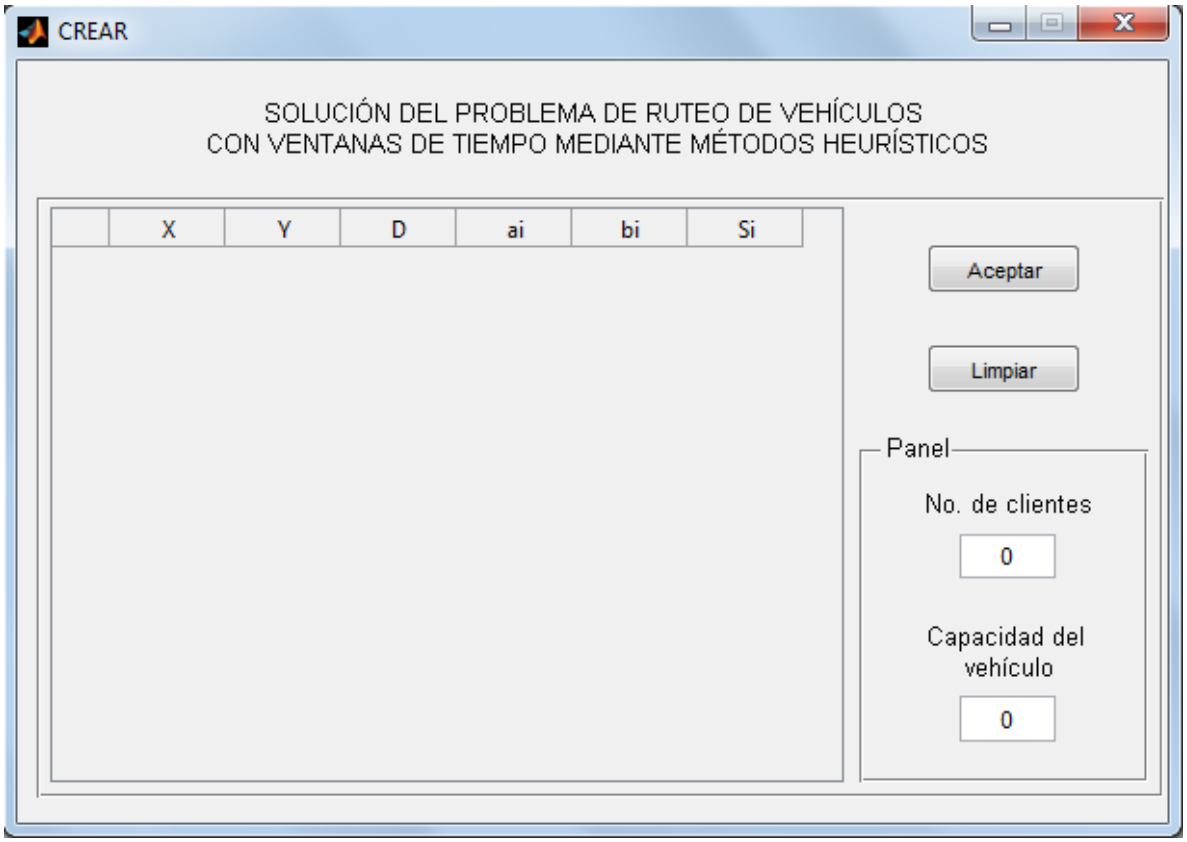
	X	Y	D	ai	bi	SI
1	45	68	10	0	1127	90
2	45	70	30	0	1125	90
3	42	66	10	0	1129	90
4	42	68	10	727	782	90
5	42	65	10	0	1130	90
6	40	69	20	621	702	90
7	40	66	20	0	1130	90
8	38	68	20	255	324	90
9	38	70	10	534	605	90
10	35	66	10	357	410	90
11	35	69	10	448	505	90
12	25	85	20	0	1107	90
13	22	75	30	30	92	90
14	22	85	10	0	1106	90
15	20	80	40	384	429	90

Fuente: Autores.

Opción “Crear archivo”.

La opción “Crear archivo”, despliega la ventana “Crear”, que permite configurar una matriz de datos propia. El propósito de esta ventana es crear matrices de pocos clientes que no estén en la literatura o que se puedan configurar con instancias del mundo real. El software permite correr matrices propias de hasta 1000 clientes.

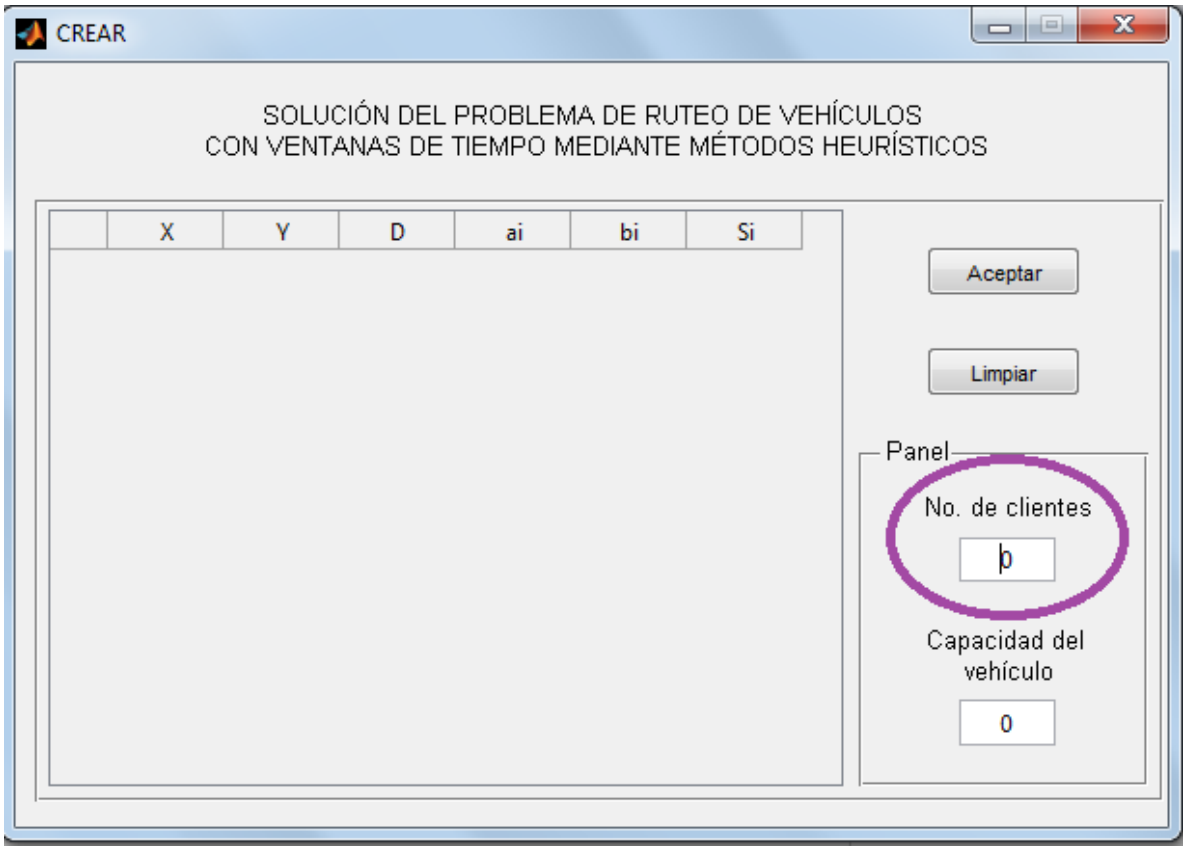
Figura A2 - 15. Opción “Crear”.



Fuente: Autores.

El primer paso para crear una instancia propia es ingresar el número de clientes en el cuadro de texto que aparece en la sección “panel” en el costado derecho de la ventana.

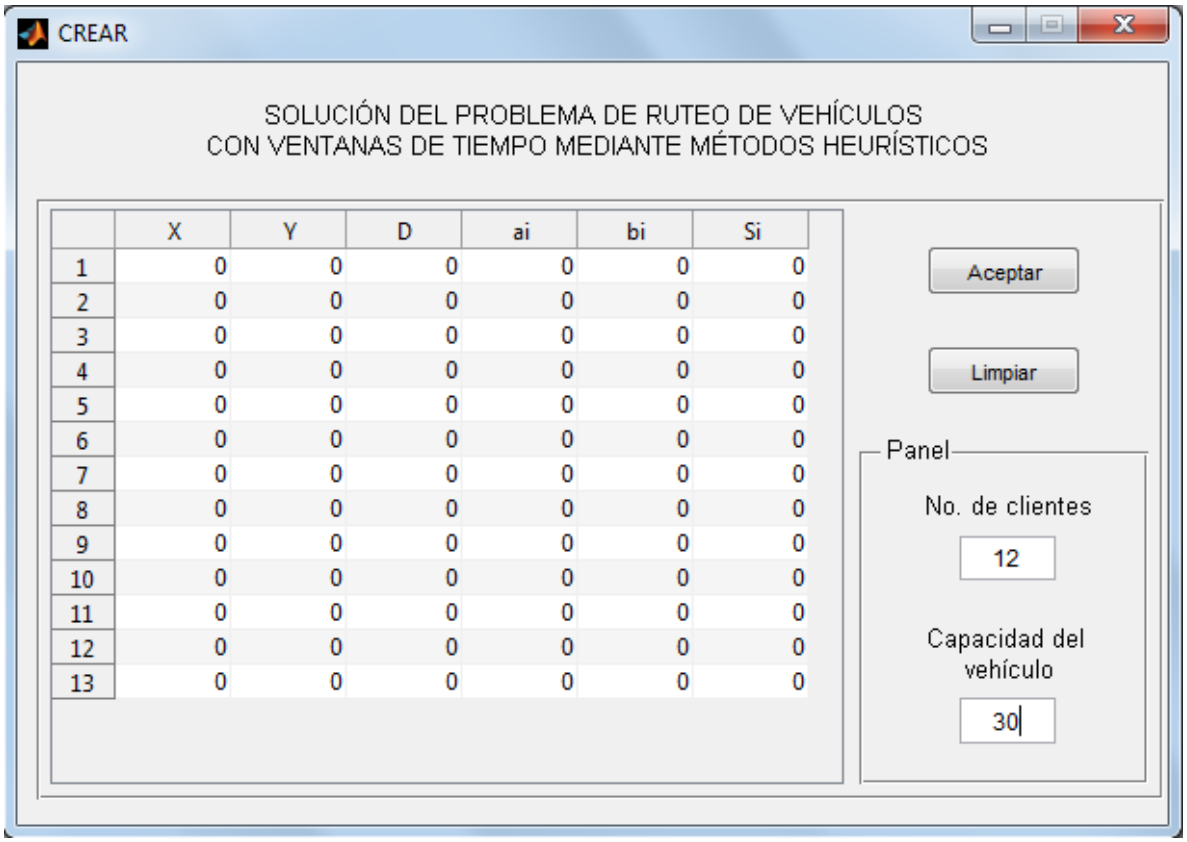
Figura A2 - 16. Número de clientes.



Fuente: Autores.

Cuando se ingresa cierto número de clientes, se activa una hoja de cálculo que inicialmente es una matriz de ceros. Las filas de la matriz corresponden al número de clientes que se indicó en el cuadro de texto. Las columnas de la matriz son los datos de los clientes. Estos aparecen en el mismo orden que en la opción "Cargar": **X**, **Y**, **D**, **ai**, **bi**, **Si**.

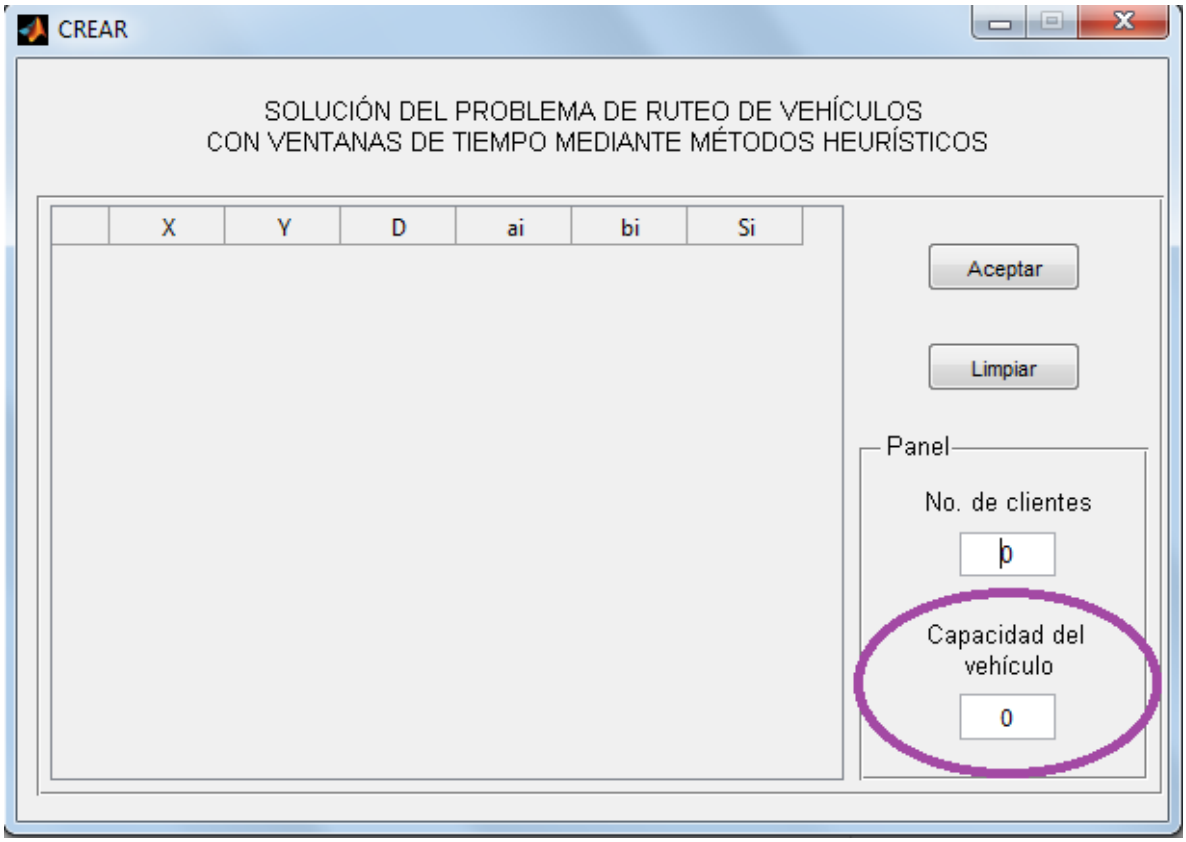
Figura A2 - 17. Matriz de datos.



Fuente: Autores.

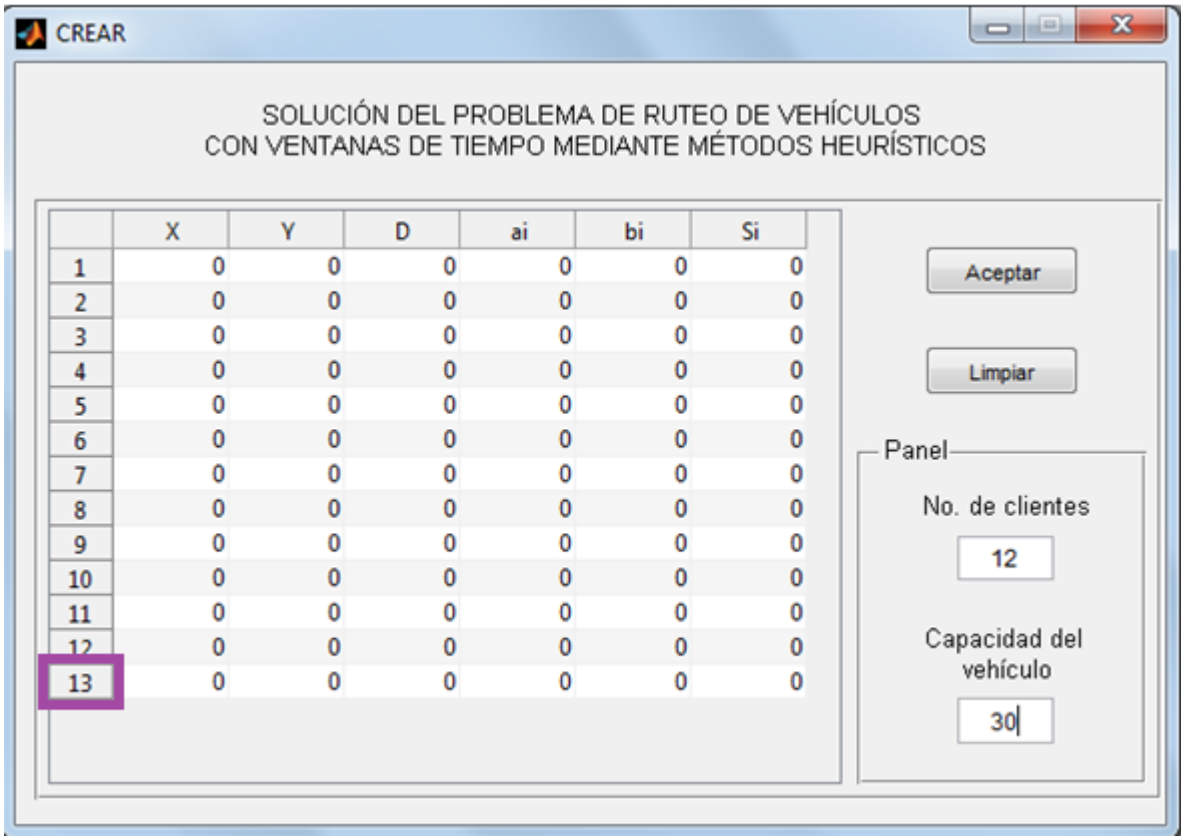
El siguiente paso, es llenar el cuadro de texto inferior que indica la carga o capacidad de los vehículos. Para el desarrollo de la herramienta, se supone una flota de vehículos muy grande y homogénea. La cantidad de vehículos de la flota no es una restricción del problema. Si *n* es la cantidad de clientes de un problema, *n+1* será el depósito.

Figura A2 - 18. Capacidad del vehículo.



Fuente: Autores.

Figura A2 - 19. Ubicación del depósito en la matriz de datos.

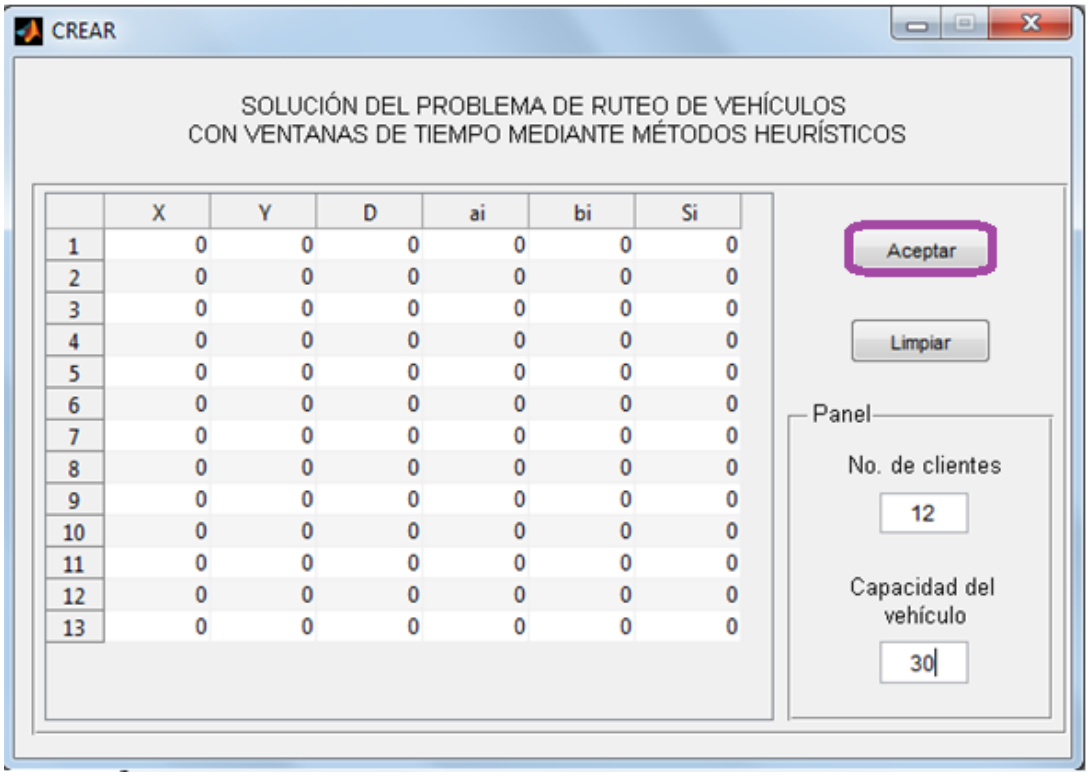


Fuente: Autores.

Para llenar la matriz de datos, se hace clic en cada una de las celdas y se ingresa la información deseada con el teclado numérico. Para desplazarse entre filas y columnas, se usan las teclas de dirección. Es importante aclarar que la matriz de datos no debe llenarse con información al azar, sino siguiendo la lógica del modelo que se pretende correr para evitar soluciones incongruentes. En el "peor" de los casos, cada cliente de la instancia creada, será asignado en una ruta exclusiva.

Una vez llena, la matriz de datos se carga al sistema con un clic en el botón "Aceptar" ubicado en el costado superior derecho de la ventana.

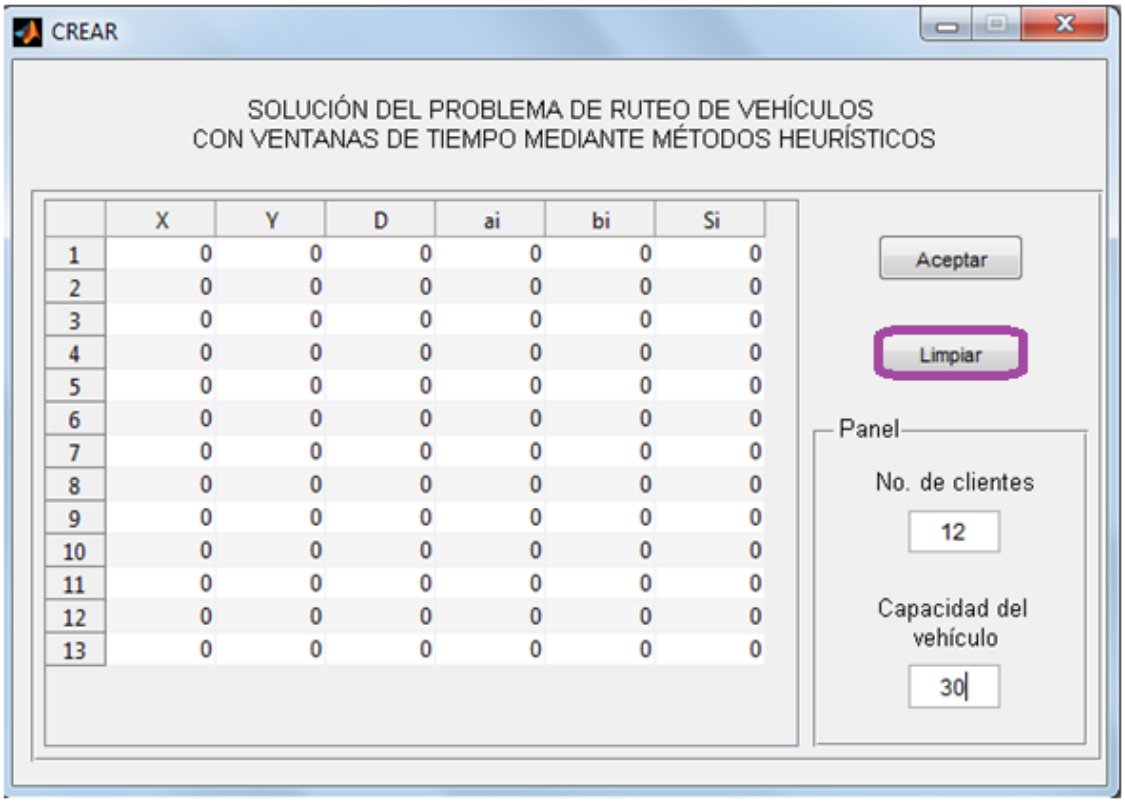
Figura A2 - 20. Botón “Aceptar” para cargar la matriz de datos al sistema.



Fuente: Autores.

Se puede editar la matriz seleccionando la celda y sobrescribiendo los datos como en una hoja de cálculo estándar. Para crear una nueva matriz de datos y repetir el proceso, se hace clic en el botón “Limpiar” en el costado superior derecho de la ventana.

Figura A2 - 21. Botón “Limpiar”.

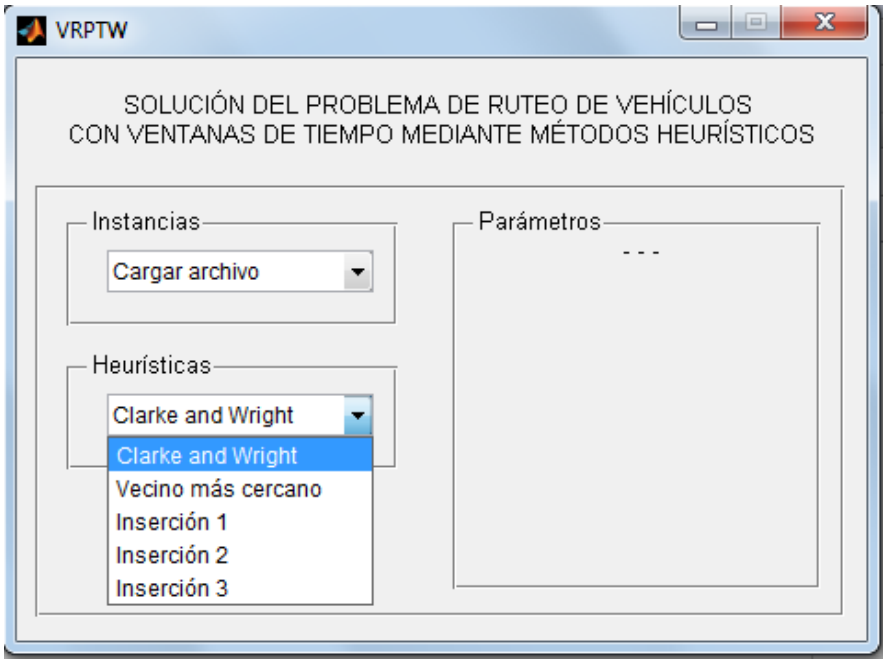


Fuente: Autores.

Selección de Heurísticas

Al cargar la matriz, el sistema muestra la ventana VRPTW. El siguiente paso es seleccionar la heurística que se va a usar para correr el modelo. Para el desarrollo de la presente investigación, se seleccionaron cinco heurísticas de construcción de rutas: algoritmo de ahorros de Clarke y Wright, heurística del vecino más cercano con enfoque temporal, y las heurísticas de inserción de Solomon Inserción 1 (I1), Inserción 2 (I2) e Inserción 3 (I3).

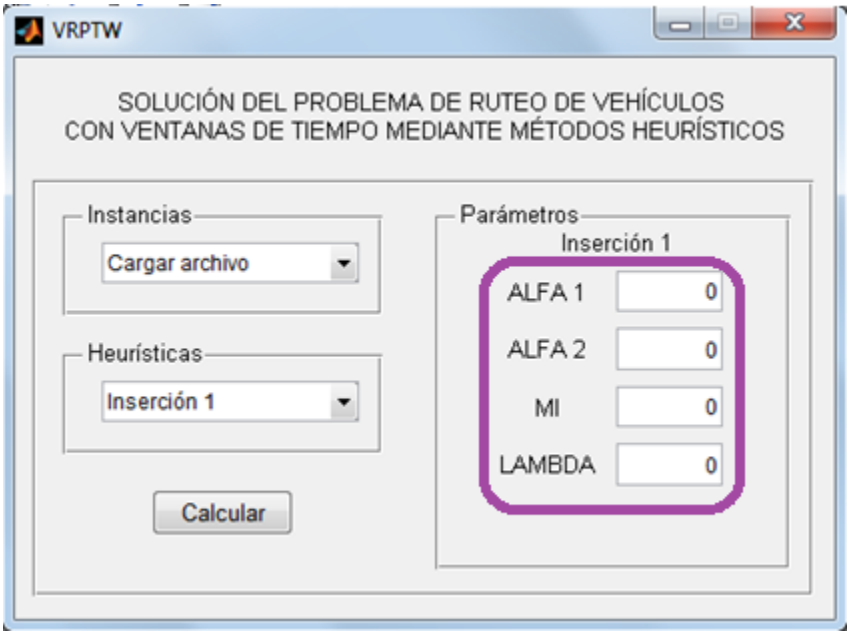
Figura A2 - 22. Menú desplegable heurísticas.



Fuente: Autores.

Cuando se selecciona una heurística, se activan los parámetros correspondientes en la sección “Parámetros”. Para el presente ejemplo se usará la heurística de Inserción 1. Los parámetros se editan escribiendo los datos en los cuadros de texto correspondientes, teniendo en cuenta los intervalos permitidos para cada uno.

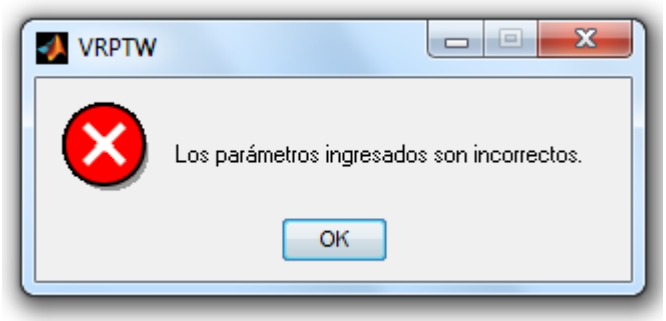
Figura A2 - 23. Parámetros heurísticas de Inserción 1.



Fuente: Autores.

En caso de que alguno de los parámetros se encuentre por fuera de los intervalos permitidos por la solución de la heurística, el sistema activa una ventana de verificación.

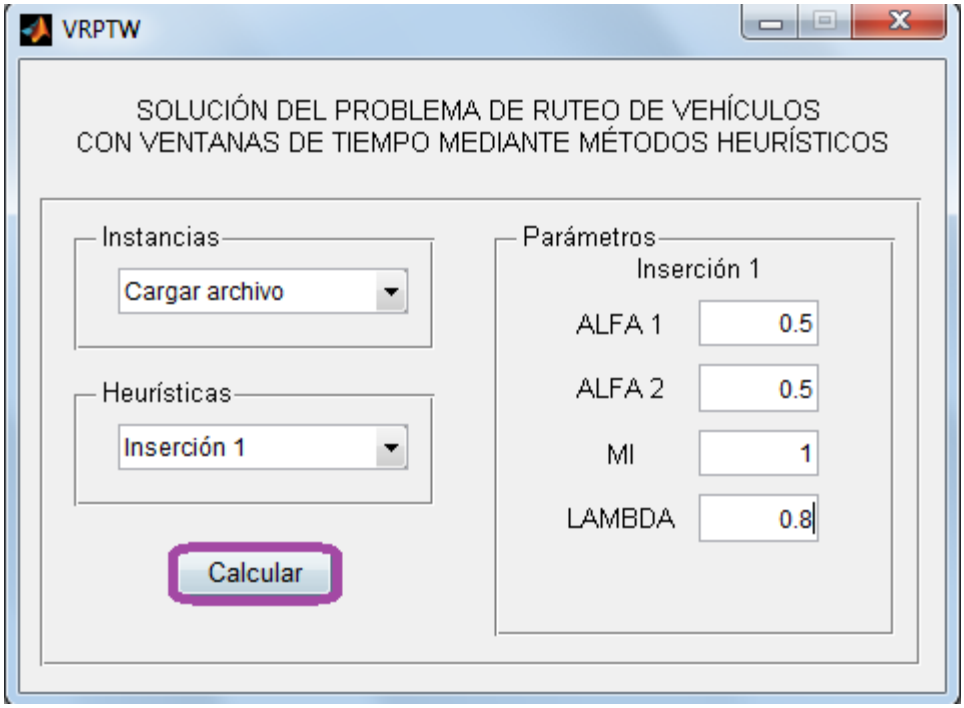
Figura A2 - 24. Ventana de verificación de parámetros



Fuente: Autores

El botón “Calcular” activa la ventana “Solución” una vez el sistema verifique la matriz de datos y el usuario seleccione la heurística.

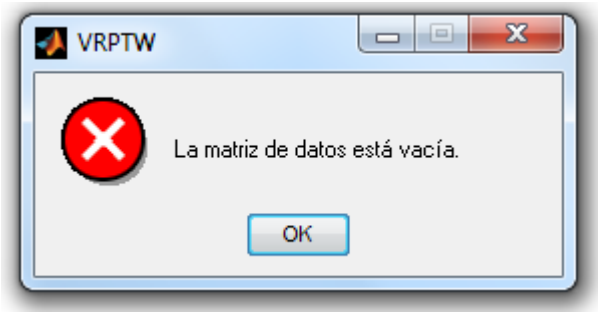
Figura A2 - 25. Botón “Calcular”.



Fuente: Autores.

En caso de “olvidar” seleccionar la heurística o llenar la matriz de datos, el sistema activa una ventana de verificación:

Figura A2 - 26. Ventana de Verificación.

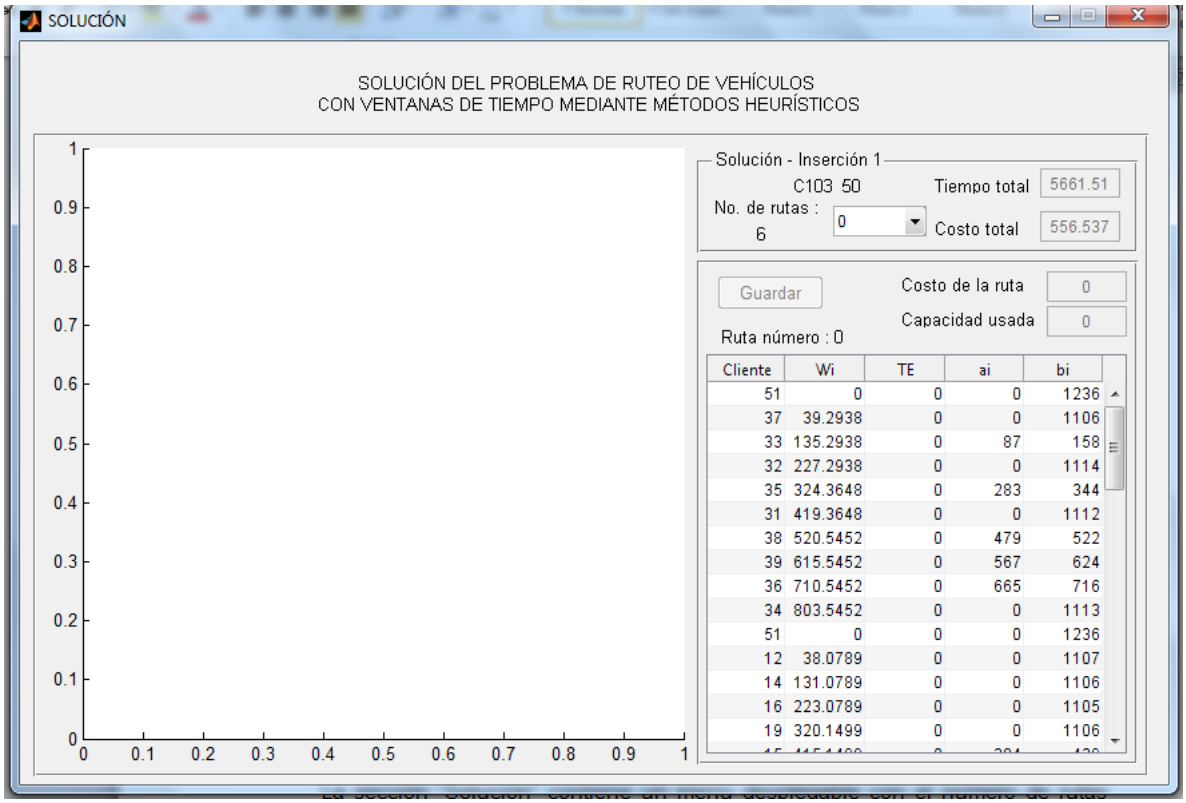


Fuente: Autores.

Resultados

La ventana “Solución” contiene los resultados de la instancia con la heurística seleccionada. Para el ejemplo, se usó la instancia C103.50, con la heurística de Inserción 1 (I1) de Solomon.

Figura A2 - 27. Ventana “Solución”.

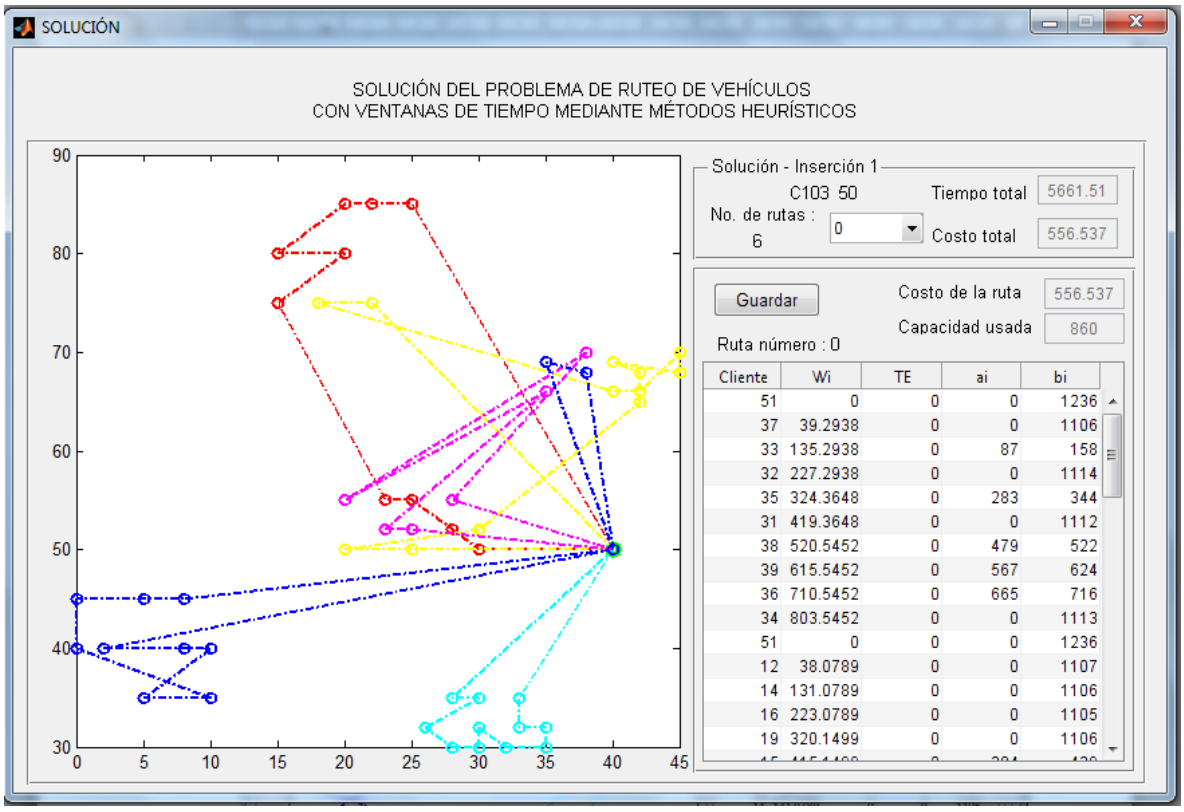


Fuente: Autores.

La sección “Solución” contiene el nombre de la heurística y de la instancia que se usaron para generar la solución, un menú desplegable con el número de rutas de despacho, un cuadro de texto no editable, “Costo Total” con el costo global de la programación de las rutas y un cuadro “Tiempo total”, con el tiempo total de programación de todas las rutas.

En la parte izquierda del menú, aparece un sistema de coordenadas que muestra la solución gráfica al hace clic sobre la opción “0” del menú desplegable “No de rutas”. Las rutas aparecen identificadas con un color diferente; el depósito, o cliente *n+1*, aparecerá siempre de color verde. En la sección derecha aparecen los resultados numéricos del problema dispuestos en una matriz.

Figura A2 - 28. Menú desplegable rutas. Solución gráfica instancia C103_50.

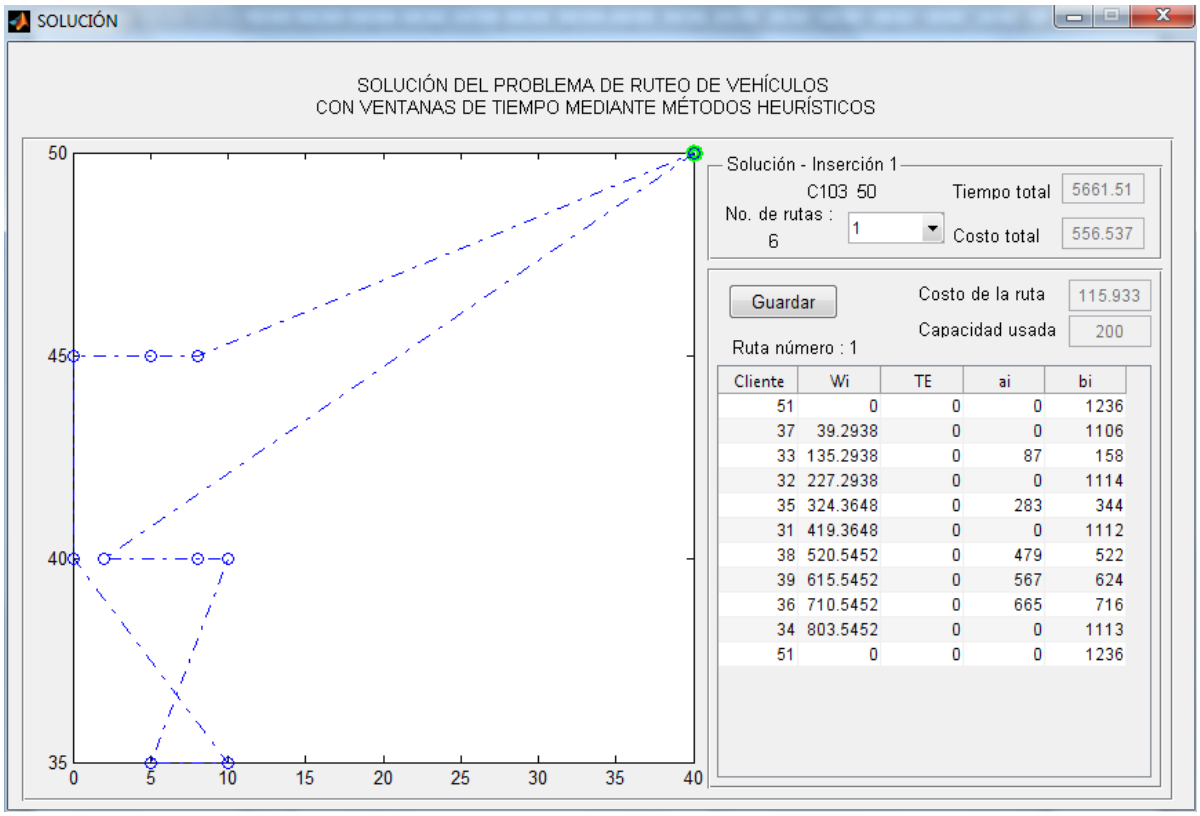


Fuente: Autores.

La columna **Clientes** muestra el orden en que fueron ruteados los clientes, la columna **Wi**, corresponde al instante en el horizonte de programación, en que comienza el servicio en el cliente especificado; la columna **TE**, es el tiempo de

espera en caso de que el vehículo llegue antes del inicio de la ventana de tiempo, que se muestra en la columna **ai**. La columna **bi** es el fin de la ventana de tiempo. Si se desean ver los resultados por rutas, se despliega el menú “No de rutas” y se hace clic en cualquiera de las opciones dispuestas. Automáticamente el software muestra la matriz de resultados, el costo y la capacidad usada de la ruta que se selecciona, así como la solución gráfica de la ruta. Es importante aclarar que si es necesario, el sistema de coordenadas se ajusta con cada ruta que se selecciona. El siguiente gráfico muestra la solución de la ruta 1 del problema.

Figura A2 - 29. Solución Ruta 1 Instancia C103_50.

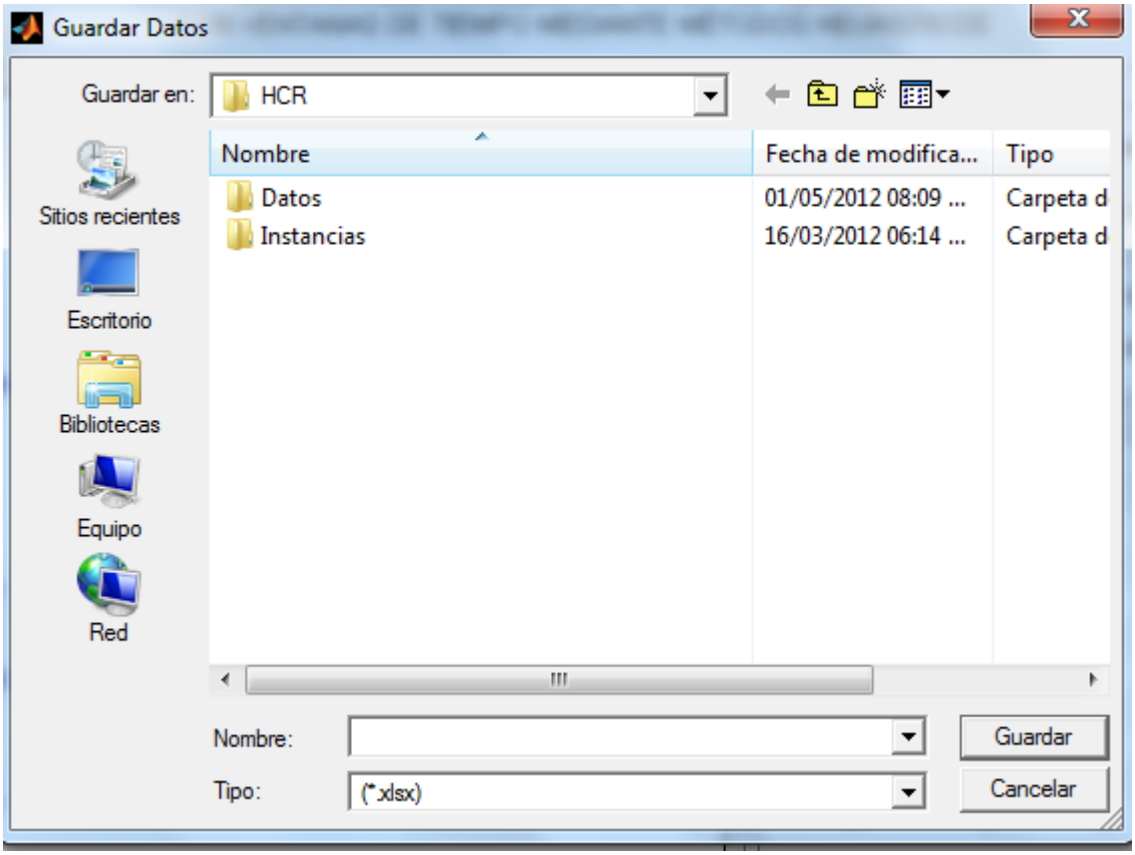


Fuente: Autores.

Por ultimo y teniendo en consideración la necesidad de tomar la solución que ofrece el programa, para imprimir o sencillamente guardarla, existe un botón

“Guardar” que da la posibilidad de salvar la matriz de resultados en un archivo con formato .xls; al dar clic en el botón, se despliega un cuadro de dialogo que permite asignar un nombre al archivo y escoger la ubicación donde será guardado.

Figura A2 - 30. Cuadro de dialogo para guardar resultados



Fuente: Autores

ANEXO C. CÓDIGOS DE PROGRAMACIÓN HERRAMIENTA VRPTW

El presente anexo describe las funciones creadas y utilizadas para la programación de los algoritmos seleccionados.

Función DISTANCIAS.

Crea la matriz simétrica de distancias euclidianas entre todos los clientes, incluido el deposito.

Datos de entrada:

- mat_Datos. Matriz de datos: Coordenadas geográficas, demandas, inicio y fin de ventanas de tiempo y duración del servicio.
- n_clientes: Número de clientes de la instancia.

```
function mat_Distancias = DISTANCIAS(mat_Datos,nClientes)%Cargar los
datos de la función DISTANCIAS.
%
mat_Distancias = zeros(nClientes + 1);%Crear matriz de distancias
for i = 1 : nClientes + 1%Recorrer filas de la matriz.
    for j = 1 : nClientes + 1%Recorrer columnas de la matriz.
        mat_Distancias(i,j) = sqrt((mat_Datos(j,1) - mat_Datos(i,1))^2 +
(mat_Datos(j,2) - mat_Datos(i,2))^2);%Cálculo de las distancias
euclidianas.
    end
end
```

Función INICIOS.

Facilita el cálculo de los tiempos de programación acumulados, para verificar si existen tiempos de espera o no. Calcula los tiempos de inicio del servicio en cada cliente.

Datos de entrada:

- mat_Datos: Matriz de datos: Coordenadas geográficas, demandas, inicio y fin de ventanas de tiempo y duración del servicio.
- mat_distancias: Matriz de distancias euclidianas.
- Vct_R: Vector ruta definido en cada uno de los algoritmos.
- pos: posición del cliente donde se cálcula y se actualiza los tiempos de programación.

```
function W = INICIOS(mat_Datos,mat_Distancias,vct_R,pos)%Carga los datos
de la función de INICIOS.
%
W = 0;
for i = 2 : pos %Contador
    W = W + mat_Datos(vct_R(i-1),6);%Actualizar variable W.
    if(W + mat_Distancias(vct_R(i-1),vct_R(i)) <= mat_Datos(vct_R(i),4))%
Verificar si el tiempo de programación acumulado es superior al inicio de
la ventana de tiempo del cliente actual.
        W = mat_Datos(vct_R(i),4);%Se genera tiempos de espera.
Actualizar el tiempo de programación con el valor del inicio de la
ventana de tiempo del cliente actual.
    else
        W = W + mat_Distancias(vct_R(i-1),vct_R(i));%No se genera tiempos
de espera. Se actualiza el tiempo de programación acumulado, sumando los
tiempos de recorrido correspondientes.
    end
end
```

Función VENTANAS.

Permite verificar si el tiempo de programación de la ruta parcial no excede el fin de la ventana de tiempo del cliente actual.

```
function Verif = VENTANAS(mat_Datos,mat_Distancias,vct_R)% Ingresar datos
y variables para la función verificar Ventanas de tiempo.
%
Verif = 1;%Si el valor de la función Verif es 1, se cumple la restricción
de tiempo del cliente que se evalúa. Si es cero, el cliente no cumple con
restricciones de tiempo.
nPos = length(vct_R);%Crear vector que recorre el número de clientes.
i = 2;
```

```
while(i <= nPos && vct_R(i) ~= 0)%Si el vector ruta en la posición actual
no ha sido ruteado.
    tInicio = INICIOS(mat_Datos,mat_Distancias,vct_R,i);%Cargar la
función de inicios.
    if(tInicio > mat_Datos(vct_R(i),5));%Si el tiempo de inicio del
servicio es mayor que el fin de la ventana de tiempo del cliente actual.
        Verif = 0;%No validar la ventana de tiempo
        break;
    end
    i = i + 1;%Contador
end
```

A continuación se describen los códigos de programación de cada una de las heurísticas de construcción de rutas seleccionadas.

Código Vecino más cercano con enfoque temporal.

Función Vecino: Carga los datos y permite ingresar el valor de los parámetros de la heurística del vecino más cercano con enfoque temporal.

- nClientes: Indica el número de clientes de la ruta.
- mat_Datos: Carga la matriz de datos,
- Capacidad: variable de capacidad o carga.
- d1, d2, d3: Valor de los parámetros de la heurística $\delta_1, \delta_2, \delta_3$.
- Vct_Coj: Vector de costos iniciales, desde el depósito a cada uno de los clientes sin rutear.
- Vct_Ruta: Vector donde se rutean los clientes.
- Wi: Tiempo de inicio de servicio en el cliente i.
- Wj: Tiempo de inicio del servicio en el siguiente cliente o cliente j.
- rPosición: Posición actual del vector Vct_Ruta.
- posV: Posición del cliente actual.
- Vct_Cij: Vector de costos, desde y hacia cada uno de los clientes sin rutear.

```
function vct_Ruta = VECINO(mat_Datos,Capacidad,d1,d2,d3)%Cargar datos,
ingresar parámetros.
```

```
%
nClientes = length(mat_Datos(:,1)) - 1; %Cálculo del número de clientes.
mat_Distancias = DISTANCIAS(mat_Datos,nClientes);% Crear matriz de
distancias
vct_Coj = zeros(1,nClientes);% Crear vector de costos partiendo del
depósito.
for i = 1 : nClientes% Cálculo del vector de costos partiendo del
depósito.
    vct_Coj(i) = mat_Distancias(nClientes+1,i)*d1 +
mat_Distancias(nClientes+1,i)*d2 + (mat_Datos(i,5)-
mat_Distancias(nClientes+1,i))*d3;
end
vct_Clientes = zeros(1,nClientes);%Crear vector clientes ruteados
vct_Ruta = zeros(1,(nClientes*2)+1);%Crear vector ruta
Carga = 0;
rPosicion = 1;%Primera posición vector ruta
while(sum(vct_Clientes) < nClientes)%Si el número de clientes es mayor
que el número de clientes, todos los clientes han sido ruteados.
    [x,posV] = min(vct_Coj);%Seleccionar el menor costo del vector de
costos.
    vct_Ruta(rPosicion) = nClientes + 1;%Aumentar en una posición el
vector ruta
    vct_Ruta(rPosicion + 1) = posV;%Rutear el cliente correspondientes al
menor costo.
    vct_Clientes(posV) = 1;%Actualiza el vector clientes ruteados.
    if(mat_Distancias(nClientes+1,posV) <= mat_Datos(posV,4))%Comprobar
ventana de tiempo primer cliente.
        Wi = mat_Datos(posV,4);%Cálculo del tiempo de inicio de servicio
en el cliente ruteado si existe tiempo de espera.
    else
        Wi = mat_Distancias(nClientes+1,posV);%Cálculo del tiempo de
inicio de servicio en el cliente ruteado si no existe tiempo de espera.
    end
    rPosicion = rPosicion + 2;%Actualizar vector ruta.
    vct_Coj(posV) = inf;%Eliminar el costo correspondiente al cliente
ruteado, agregando al vector un número muy grande.
    Carga = Carga + mat_Datos(posV,3);%Cálculo de sumatoria de las cargas
de los clientes de la ruta parcial.
    while(Carga <= Capacidad) %Comprobar si la sumatoria de las cargas de
la ruta parcial es menor a la carga del vehículo.
        Wj = zeros(1,nClientes);%Crear vector que guarda los tiempo
acumulados de programación en cada cliente.
        vct_Cij = zeros(1,nClientes);%Crear vector costo, partiendo del
último cliente ruteado.
        for i = 1 : nClientes%Recorre el vector de clientes para el
cálculo del costo desde y hacia cada uno de los clientes sin rutear.
            if(vct_Clientes(i) == 0)%Comprobar si el cliente actual ya ha
sido ruteado.
                if(Carga + mat_Datos(i,3) <= Capacidad)%Comprobar
capacidad clientes ruta parcial.
                    if((Wi + mat_Datos(vct_Ruta(rPosicion - 1),6) +
mat_Distancias(vct_Ruta(rPosicion - 1),i)) < mat_Datos(i,5))%Comprobar
que el tiempo de programación de la ruta no exceda el fin de la ventana
de tiempo del cliente actual.
```

```
                        if((Wi + mat_Datos(vct_Ruta(rPosicion - 1),6) +
mat_Distancias(vct_Ruta(rPosicion - 1),i)) <= mat_Datos(i,4))%cálculo de
tiempo de inicio de servicio en el cliente actual.
                            Wj(i)  = mat_Datos(i,4);%Inicio de servicio
cuando existe tiempo de espera.
                        else
                            Wj(i) = Wi + mat_Datos(vct_Ruta(rPosicion -
1),6) +  mat_Distancias(vct_Ruta(rPosicion - 1),i);%Inicio de servicio
cuando no existe tiempo de espera.
                        end
                        dij = mat_Distancias(vct_Ruta(rPosicion -
1),i);%Cálculo primer término de la función de costo, correspondiente a
la distancia.
                        Tij = Wj(i) - (Wi + mat_Datos(vct_Ruta(rPosicion
- 1),6));%Cálculo segundo término de la función de costo correspondiente
al tiempo de programación.
                        Vij = mat_Datos(i,5) - (Wi +
mat_Datos(vct_Ruta(rPosicion - 1),6) + mat_Distancias(vct_Ruta(rPosicion
- 1),i));%Cálculo del tercer término de la función de costo
correspondiente a la urgencia de servir clientes con ventanas de tienmpo
próximas a cerrarse.
                        vct_Cij(i) = (dij * d1) + (Tij * d2) + (Vij *
d3);%Cálculo de la función completa de costo
                    else
                        vct_Cij(i) = inf;%Si no se cumple la restricción
de tiempo, eliminar la posición actual en el vector de costo, agregando
un valor muy grande.
                    end
                else
                    vct_Cij(i) = inf;%Si no se cumple la restricción de
capacidad, eliminar la posición actual en el vector de costo, agregando
un valor muy grande.
                end
            else
                vct_Cij(i) = inf;%Si el cliente ya ha sido ruteado,
elminar la posición actual en el vector de costos, agregando un valor muy
grande.
            end
        end
        if(vct_Cij == Inf)%Si existen clientes que no cumplen con ninguna
de las restricciones, terminar la comprobación e iniciar una nueva ruta.
            break;
        else
            [x,posV] = min(vct_Cij);%Cálcular la posición del cliente
correspondiente al menor costo del vector de costo.
            vct_Ruta(rPosicion) = posV;%Agregar el cliente actual a la
ruta parcial.
            rPosicion = rPosicion + 1;%Actualizar la posición del vector
en la ruta parcial.
            vct_Clientes(posV) = 1;
            vct_Coj(posV) = inf;%Eliminar la posición actual en el vector
de costos desde el depósito.
            Wi = Wj(posV); %Actualizar el tiempo de programación de las
rutas.
```

```
        end
    end% Si la sumatoria de las cargas de la ruta parcial es mayor a la
capacidad del vehículo, finalizar ruta parcial.
end%Si todos lo clientes han sido ruteados, terminar el algoritmo.
vct_Ruta(rPosicion) = nClientes + 1;% Crear ruta final .
vct_Ruta = vct_Ruta(1:rPosicion);%Eliminar las posiciones que sobran en
el vector ruta.
```

Código Ahorros (Clarke y Wright).

Función AHORROS: Función que calcula los ahorros que se generan al servir pares de clientes, y los utiliza como criterio de selección para formar rutas cumpliendo con las restricciones de capacidad del vehículo y ventanas de tiempo.

- Lambda: Parámetro usado para “suavizar” las rutas que crea el algoritmo de ahorros.

```
function vct_Ruta = AHORROS(mat_Datos,Capacidad,lambda)
nClientes = length(mat_Datos(:,1)) - 1; %Número de clientes
mat_Distancias = DISTANCIAS(mat_Datos,nClientes); %Obtiene la matriz
distancia de la función DISTANCIAS
mat_Ahorros = zeros(nClientes); %Crea la matriz donde se almacenan los
ahorros
for i = 1 : nClientes
    for j = 1 : nClientes
        if(i ~= j)
            mat_Ahorros(i,j) = mat_Distancias(i,nClientes + 1) +
mat_Distancias(nClientes + 1,j) - (lambda * mat_Distancias(i,j));
        end
    end
end
vct_Ruta = zeros(1,(nClientes*2)+1); %Crea el vector de la ruta Final
rPosicion = 1; %Contador de los clientes asignados a la ruta
while(sum(sum(mat_Ahorros)) ~= 0) %mientras existan ahorros disponibles:
    [v_Mayores,v_PosCol] = max(mat_Ahorros); %Inicia la búsqueda del
mayor ahorro
    [mayorAhorro,y] = max(v_Mayores); %Muestra al cliente "y" del par
    x = v_PosCol(y); %Muestra al cliente "x" del par
    TiempoAcumulado = 0; %Inicializa la variable del tiempo acumulado
    Carga = mat_Datos(x,3) + mat_Datos(y,3); define %la carga parcial del
vehiculo con el par seleccionado
    if(Carga <= Capacidad) %si el par no excede la capacidad del vehículo
        AceptaRuta = 0; %Inicialización de la variable binaria que acepta
(1) o rechaza (0)la factibilidad del par
        TEsperai = mat_Datos(x,4) - mat_Distancias(nClientes + 1,x);
%Calcula el tiempo de espera cuando el orden de visita es "x"-"y"
```

```
        TEsperaj = mat_Datos(y,4) - mat_Distancias(nClientes + 1,y);
%Calcula el tiempo de espera cuando el orden de visita es "y"-"x"
        if(TEsperai < 0) %Restringe los Tiempos de espera negativos
            TEsperai = 0;
        end
        if(TEsperaj < 0) %Restringe los Tiempos de espera negativos
            TEsperaj = 0;
        end
        TEspera2 = 0; %Variable del tiempo de espera que se genera al
visitar al segundo cliente
        if(TEsperai <= TEsperaj) %Prioriza la atencion del cliente con
menor tiempo de espera
            Tiempo = mat_Distancias(nClientes + 1,x) + TEsperai +
mat_Datos(x,6) + mat_Distancias(x,y); %tiempo cuando se visita al cliente
"x" en primer lugar
            if(Tiempo <= mat_Datos(y,5)) %Examina si es factible visitar
al cliente "y" como segundo cliente
                if(Tiempo < mat_Datos(y,4)) %Determina si se incurre en
tiempo de espera al visitar a "y"
                    TEspera2 =  mat_Datos(y,4) - Tiempo; %Se actualiza la
variable
                end
                TiempoAcumulado = Tiempo + TEspera2 + mat_Datos(y,6);
%Actualiza el tiempo de recorrido hasta servir a "y"
                vct_Ruta(rPosicion) = nClientes + 1; %Asigna clientes a
la ruta en el orden establecido
                vct_Ruta(rPosicion + 1) = x;
                vct_Ruta(rPosicion + 2) = y;
                rPosicion = rPosicion + 3; %Incrementa el contador en el
mismo numero de asignaciones
                mat_Ahorros(x,:) = 0; %Elimina de la matriz de ahorros el
primer cliente asignado a la ruta
                mat_Ahorros(:,x) = 0;
                AceptaRuta = 1; %Confirma que ya se inicio la ruta con el
mayor par de ahorro
            else %si no es factible visitar al cliente "y" en segundo
lugar
                Tiempo = mat_Distancias(nClientes + 1,y) + TEsperaj +
mat_Datos(y,6) + mat_Distancias(x,y); %tiempo cuando se visita al cliente
"y" en primer lugar
                if(Tiempo <= mat_Datos(x,5)) %Examina si es factible
visitar al cliente "x" como segundo cliente
                    if(Tiempo < mat_Datos(x,4)) %Determina si se incurre
en tiempo de espera al visitar a "x"
                        TEspera2 =  mat_Datos(x,4) - Tiempo; %Se
actualiza la variable
                    end
                    TiempoAcumulado = Tiempo + TEspera2 + mat_Datos(x,6);
%Actualiza el tiempo de recorrido hasta servir a "x"
                    vct_Ruta(rPosicion) = nClientes + 1; %Asigna clientes
a la ruta en el orden establecido
                    vct_Ruta(rPosicion + 1) = y;
                    vct_Ruta(rPosicion + 2) = x;
```

```
                    rPosicion = rPosicion + 3; %Incrementa el contador en
el mismo numero de asignaciones
                    mat_Ahorros(y,:) = 0; %Elimina de la matriz de
ahorros al primer cliente asignado a la ruta
                    mat_Ahorros(:,y) = 0;
                    AceptaRuta = 1; %Confirma que ya se inicio la ruta
con el mayor par de ahorro
                else
                    mat_Ahorros(x,y) = 0; %El par seleccionado("x" y "y")
no pueden ser servidos en la misma ruta (debido a las ventanas de tiempo)
                    mat_Ahorros(y,x) = 0;
                end
            end
        else
            Tiempo = mat_Distancias(nClientes + 1,y) + TEsperaj +
mat_Datos(y,6) + mat_Distancias(x,y); %tiempo cuando se visita al cliente
"y" en primer lugar
            if(Tiempo <= mat_Datos(x,5)) %Examina si es factible visitar
al cliente "x" como segundo cliente
                if(Tiempo < mat_Datos(x,4)) %Determina si se incurre en
tiempo de espera al visitar a "x"
                    TEspera2 =  mat_Datos(x,4) - Tiempo; %Se actualiza la
variable
                end
                TiempoAcumulado = Tiempo + TEspera2 + mat_Datos(x,6);
%Actualiza el tiempo de recorrido hasta servir a "x"
                vct_Ruta(rPosicion) = nClientes + 1; %Asigna clientes a
la ruta en el orden establecido
                vct_Ruta(rPosicion + 1) = y;
                vct_Ruta(rPosicion +2) = x;
                rPosicion = rPosicion + 3; %Incrementa el contador en el
mismo numero de asignaciones
                mat_Ahorros(y,:) = 0; %Elimina de la matriz de ahorros al
primer cliente asignado a la ruta
                mat_Ahorros(:,y) = 0;
                AceptaRuta = 1; %Confirma que ya se inicio la ruta con el
mayor par de ahorro
            else %si no es factible visitar al cliente "x" en segundo
lugar
                Tiempo = mat_Distancias(nClientes + 1,x) + TEsperai +
mat_Datos(x,6) + mat_Distancias(x,y); %tiempo cuando se visita al cliente
"x" en primer lugar
                if(Tiempo <= mat_Datos(y,5)) %Examina si es factible
visitar al cliente "y" como segundo cliente
                    if(Tiempo < mat_Datos(y,4)) %Determina si se incurre
en tiempo de espera al visitar a "y"
                        TEspera2 =  mat_Datos(y,4) - Tiempo; %Se
actualiza la variable
                    end
                    TiempoAcumulado = Tiempo + TEspera2 + mat_Datos(y,6);
%Actualiza el tiempo de recorrido hasta servir a "y"
                    vct_Ruta(rPosicion) = nClientes + 1; %Asigna clientes
a la ruta en el orden establecido
                    vct_Ruta(rPosicion + 1) = x;
```

```
                    vct_Ruta(rPosicion + 2) = y;
                    rPosicion = rPosicion + 3; %Incrementa el contador en
el mismo numero de asignaciones
                    mat_Ahorros(x,:) = 0; %Elimina de la matriz de
ahorros al primer cliente asignado a la ruta
                    mat_Ahorros(:,x) = 0;
                    AceptaRuta = 1; %Confirma que ya se inicio la ruta
con el mayor par de ahorro
                else
                    mat_Ahorros(x,y) = 0; %El par seleccionado("x" y "y")
no pueden ser servidos en la misma ruta(debido a las ventanas de tiempo)
                    mat_Ahorros(y,x) = 0;
                end
            end
        end
        while(AceptaRuta == 1 && Carga <= Capacidad) %Selecciona el
cliente relacionado a la ruta con mayor ahorro
            xP = vct_Ruta(rPosicion - 1); %Define el lugar que tiene el
ultimo cliente de la ruta
            [mayorAhorroParcial,yP] = max(mat_Ahorros(xP,:)); %Selecciona
el cliente relacionado a "xP" de mayor ahorro
            CargaParcial = Carga + mat_Datos(yP,3); %Actualiza la carga
parcial del vehículo con el cliente "yP"
            if(mayorAhorroParcial == 0) %Si el ahorro del cliente
seleccionado es cero, busca otro cliente
                break;
            end
            if(CargaParcial <= Capacidad) %Examina si la demanda del
cliente "yP" no sobrepasa la capacidad del vehículo
                vct_RutaT = vct_Ruta; %Crea vector de ruta temporal para
la función INICIOS
                vct_RutaT(rPosicion) = yP; %asigna al cliente "yP" al
vector de ruta temporal
                Tiempo =
INICIOS(mat_Datos,mat_Distancias,vct_RutaT,rPosicion); %Actualiza el
tiempo de la ruta
                if(Tiempo <= mat_Datos(yP,5)) %Examina si es factible
visitar al cliente "yP"
                    vct_Ruta(rPosicion) = yP; %Asigna el cliente "yP" a
la ruta
                    rPosicion = rPosicion + 1; %Incrementa el contador en
el mismo numero de asignaciones
                    Carga = CargaParcial; %Actualiza la carga del
vehículo
                    mat_Ahorros(xP,:) = 0; %Elimina de la matriz de
ahorros el correspondiente al cliente "xP"
                    mat_Ahorros(:,xP) = 0;
                else %Si no es factible visitar al cliente "yP"
                    mat_Ahorros(xP,yP) = 0; %El cliente "yP" no puede ser
servido en la misma ruta(debido a las ventanas de tiempo)
                    mat_Ahorros(yP,xP) = 0;
                end
            else
```

```
                mat_Ahorros(xP,yP) = 0; %El cliente "yP" no puede ser
servido en la misma ruta(debido su demanda)
                mat_Ahorros(yP,xP) = 0;
            end
        end
    else
        mat_Ahorros(x,y) = 0; %El par seleccionado("x" y "y") no pueden
ser servidos en la misma ruta(debido a su demandas)
        mat_Ahorros(y,x) = 0;
    end
end
vct_AuxClientes = zeros(1,nClientes + 1);%Crea un vector auxiliar de la
ruta
for i = 1 : rPosicion - 1
    vct_AuxClientes(vct_Ruta(i)) = 1; %Asigna uno (1) a los clientes que
han sido asignados a la ruta y cero (0) a los que no han sido visitados
end
for i = 1 : nClientes
    if(vct_AuxClientes(i) == 0) %Si algún lugar del vector auxiliar es
cero
        vct_Ruta(rPosicion) = nClientes + 1; %El cliente se asigna a una
ruta individual
        vct_Ruta(rPosicion + 1) = i;
        rPosicion = rPosicion + 2; %Incrementa el contador en el mismo
numero de asignaciones
        vct_AuxClientes(i) = 1;
    end
end
vct_Ruta(rPosicion) = nClientes + 1; %Termina el vector ruta visitando al
depósito
vct_Ruta = vct_Ruta(1:rPosicion); %Define el vector ruta
```

Código Inserción (Solomon). I1, I2 e I3.

Función INSERCIÓN: Función que encuentra la ruta para visitar un conjunto de clientes mediante los algoritmos de inserción I1, I2 e I3.

```
function vct_Ruta = INSERCION(mat_Datos,Capacidad,I,vct_Parametros)
if(I == 1) %Si se utiliza el algoritmo de Inserción I1
    alfa1 = vct_Parametros(1); %Definición de parámetros
    alfa2 = vct_Parametros(2);
    mi = vct_Parametros(3);
    lambda = vct_Parametros(4);
elseif(I == 2) %Si se utiliza el algoritmo de Inserción I2
    alfa1 = vct_Parametros(1); %Definición de parámetros
    alfa2 = vct_Parametros(2);
    mi = vct_Parametros(3);
    beta1 = vct_Parametros(4);
    beta2 = vct_Parametros(5);
elseif(I == 3) %Si se utiliza el algoritmo de Inserción I3
```

```
    alfa1 = vct_Parametros(1); %Definición de parámetros
    alfa2 = vct_Parametros(2);
    alfa3 = vct_Parametros(3);
    mi = vct_Parametros(4);
end
nClientes = length(mat_Datos(:,1)) - 1; %Número de clientes
mat_Distancias = DISTANCIAS(mat_Datos,nClientes); %Obtiene la matriz
distancia de la función DISTANCIAS
vct_Distancias = mat_Distancias(nClientes+1,1:nClientes); %Define el
vector de distancias desde el depósito (para el CRITERIO DE INSERCIÓN)
vct_Ruta = zeros(1,(nClientes*2)+1); %Crea el vector de la ruta Final
vct_Visitados = zeros(1,nClientes); %Crea el vector de clientes visitados
rPosicion = 1; %Contador de los clientes asignados a la ruta
while(rPosicion < (nClientes*2) + 1) %Mientras no se hayan visitado todos
los clientes
    vct_RutaParcial = zeros(1,nClientes + 2); %Crea un vector de pruebas
para las rutas Parciales
    [mayorDistancia,y] = max(vct_Distancias); %Selecciona el cliente más
lejano al depósito (CRITERIO DE INSERCIÓN)
    if(mayorDistancia == 0) %Si el cliente no está disponible
        break;
    end
    vct_RutaParcial(1) = nClientes + 1; %Asigna el cliente selccionado a
la ruta parcial
    vct_RutaParcial(2) = y;
    vct_RutaParcial(3) = nClientes + 1; %Asigna el depósito para cerrar
la ruta
    Carga = mat_Datos(vct_RutaParcial(2),3); %Recarga la demanda del
cliente recien asignado a la carga del vehículo
    vct_Visitados(vct_RutaParcial(2)) = 1; %En el vector de visitados se
asigna uno (1) en la posición del cliente asignado
    vct_Distancias(y) = 0; %El cliente asignado se elimina del vector de
distancias
    rpPosicion = 3; %Incrementa el contador en el mismo numero de
asignaciones
    while(Carga <= Capacidad) %Mientras exista capacidad en el vehículo:
        vct_Posibles = zeros(1,nClientes); %Crea vector de clientes
posibles (Factibles)
        for i = 1 : nClientes %Se activan los clientes factibles por
capacidad en el vector de clientes posibles (Factibles)
            if(((Carga + mat_Datos(i,3)) <= Capacidad) &&
(vct_Visitados(i) == 0))
                vct_Posibles(i) = 1;
            end
        end
        if(sum(vct_Posibles) == 0) %Si ningún cliente es factible
            break;
        end
        mat_C1 = zeros(nClientes,rpPosicion - 1); %Crea la matriz del
criterio C1
        for i = 1 : nClientes
            for j = 1 : rpPosicion - 1
                if(vct_Posibles(i) == 1) %Para todos los clientes
factibles
```

```
                    vct_Rutau =
[vct_RutaParcial(1:j),i,vct_RutaParcial(j+1:nClientes + 1)]; %Crea un
vector de la ruta con el cliente u insertado
                    if(VENTANAS(mat_Datos,mat_Distancias,vct_Rutau) == 1)
%Si la función VENTANAS define que la inserción es factible por ventanas
de tiempo
                        C11 = mat_Distancias(vct_RutaParcial(j),i) +
mat_Distancias(i,vct_RutaParcial(j+1)) - (mi *
mat_Distancias(vct_RutaParcial(j),vct_RutaParcial(j+1))); %Calcula el
criterio C11
                        Wju =
INICIOS(mat_Datos,mat_Distancias,vct_Rutau,j+2); %Toma de la funcion
INICIOS la variable Wju
                        Wj =
INICIOS(mat_Datos,mat_Distancias,vct_RutaParcial,j+1); %Toma de la
funcion INICIOS la variable Wj
                        C12 = Wju - Wj; % Cálcula el criterio C12
                        if(I == 3) %Si se utiliza el algoritmo de
Inserción I3
                            Wu =
INICIOS(mat_Datos,mat_Distancias,vct_Rutau,j+1); %Toma de la funcion
INICIOS la variable Wu
                            C13 = mat_Datos(i,5) - Wu; % Cálcula el
criterio C13
                            mat_C1(i,j) = alfa1 * C11 + alfa2 * C12 +
alfa3 * C13; %Cálcula el criterio C1 mediante los parámetros
                        else %Si no se utiliza el algoritmo de Inserción
I3
                            mat_C1(i,j) = alfa1 * C11 + alfa2 * C12;
%Cálcula el criterio C1 mediante los parámetros

                        end
                    end
                end
            end
        end
        for i = 1 : nClientes
            if(sum(mat_C1(i,:)) ~= 0) %Si para el cliente i existen
valores en la matriz del criterio C1
                for j = 1 : rpPosicion - 1
                    if(mat_C1(i,j) == 0) %Los valores del criterio C1 que
sean cero se convierten a "infinito"
                        mat_C1(i,j) = inf;
                    end
                end
            else %Si para el cliente i no existen valores en la matriz
del criterio C1
                vct_Posibles(i) = 0; %El cliente i se penaliza con cero
(0) en el vector de cliente posibles
            end
        end
        if(sum(vct_Posibles) == 0) %Si no existen clientes factibles
            break;
        end
```

```
        vct_C2 = zeros(nClientes,2); %Crea el vector para almacenar el
criterio C2
        if(I == 1) %Si se utiliza el algoritmo de Inserción I1
            for i = 1 : nClientes
                if(vct_Posibles(i) == 1) %De los clientes factibles:
                    [minC1,vct_C2(i,1)] = min(mat_C1(i,:)); %Encuentra el
minimo criterio C1 para cada cliente
                    vct_C2(i,2) = (lambda * mat_Distancias(i,nClientes +
1)) - minC1; %Halla el criterio C2
                else % si el clientes no es factible
                    vct_C2(i,2) = -Inf; %el criterio C2 es "infinito"
                end
            end
        elseif(I == 2) %Si se utiliza el algoritmo de Inserción I1
            for i = 1 : nClientes
                if(vct_Posibles(i) == 1) %De los clientes factibles:
                    [minC1,u] = min(mat_C1(i,:)); %Encuentra el minimo
criterio C1 para cada cliente
                    vct_Rutau =
[vct_RutaParcial(1:u),i,vct_RutaParcial(u+1:nClientes + 1)]; %Re-define
el vector
                    DistanciaU = 0; %Crea una variable
                    j = 2;
                    while((vct_Rutau(j) ~= 0) && (j <=
length(vct_Rutau))) %Obtiene la distancia total cuando se inserta el
cliente u
                        DistanciaU = DistanciaU +
mat_Distancias(vct_Rutau(j-1),vct_Rutau(j));
                        j = j + 1;
                    end
                    TiempoU =
INICIOS(mat_Datos,mat_Distancias,vct_Rutau,j-1); %Obtiene el tiempo total
cuando se inserta el cliente u
                    vct_C2(i,2) = (beta1 * DistanciaU) + (beta2 *
TiempoU); %Halla el criterio C2
                    vct_C2(i,1) = u;
                else %Si el clientes no es factible
                    vct_C2(i,2) = -Inf; %El criterio C2 es "infinito"
                end
            end
        elseif(I == 3) %Si se utiliza el algoritmo de Inserción I3
            for i = 1 : nClientes
                if(vct_Posibles(i) == 1) %De los clientes factibles:
                    [vct_C2(i,2),vct_C2(i,1)] = min(mat_C1(i,:)); %Halla
el criterio C2
                else %Si el clientes no es factible
                    vct_C2(i,2) = -Inf; %El criterio C2 es "infinito"
                end
            end
        end
        if(I == 1) %Si se utiliza el algoritmo de Inserción I1
            [maxC2,C2] = max(vct_C2(:,2)); %Encuentra el máximo C2
            while(maxC2 ~= -Inf) %Mientras el máximo C2 no sea "infinito"
```

```
                vct_RutaParcial =
[vct_RutaParcial(1:vct_C2(C2,1)),C2,vct_RutaParcial(vct_C2(C2,1)+1:nClien
tes + 1)]; %Se actualiza el vector ruta parcial
                Carga = Carga + mat_Datos(C2,3); %Se actualiza la carga
                vct_Visitados(C2) = 1; %Se activa el cliente en el vector
de visitados
                vct_Distancias(C2) = 0; %El cliente asignado se elimina
del vector de distancias
                rpPosicion = rpPosicion + 1; %Incrementa el contador en
el mismo numero de asignaciones
                break;
            end
        else
            for i = 1 : nClientes
                if(vct_C2(i,2) == -Inf) %Se penalizan los valores del
vector C2
                    vct_C2(i,2) = Inf;
                end
            end
            [minC2,C2] = min(vct_C2(:,2)); %Encuentra el minimo C2
            while(minC2 ~= Inf)
                vct_RutaParcial =
[vct_RutaParcial(1:vct_C2(C2,1)),C2,vct_RutaParcial(vct_C2(C2,1)+1:nClien
tes + 1)]; %Se actualiza el vector ruta parcial
                Carga = Carga + mat_Datos(C2,3); %Se actualiza la carga
                vct_Visitados(C2) = 1; %Se activa el cliente en el vector
de visitados
                vct_Distancias(C2) = 0; %El cliente asignado se elimina
del vector de distancias
                rpPosicion = rpPosicion + 1; %Incrementa el contador en
el mismo numero de asignaciones
                break;
            end
        end
    end
    vct_Ruta(rPosicion:(rPosicion+rpPosicion-1)) =
vct_RutaParcial(1:rpPosicion);
    rPosicion = rPosicion + rpPosicion - 1;
end
vct_Ruta = vct_Ruta(1:rPosicion); %Define el vector Ruta
```

ANEXO D. ARTÍCULO "SOLUCIÓN DEL PROBLEMA DE RUTEO DE VEHÍCULOS CON VENTANAS DE TIEMPO (VRPTW) MEDIANTE MÉTODOS HEURÍSTICOS"

ADRIANA LOZADA DIAZ
Estudiante Ingeniería Industrial
Universidad Industrial de Santander
adrianalozada4789@hotmail.com

RICARDO ANDRÉS CADENA GONZÁLEZ
Estudiante Ingeniería Industrial
Universidad Industrial de Santander
rianca7@yahoo.es

RESUMEN

Para la presente investigación, se estudiaron y sistematizaron cinco heurísticas de construcción de rutas que permiten solucionar las instancias más comunes del VRPTW de manera rápida y eficiente. La herramienta fue programada en el software Matlab y arroja mejores resultados en términos de costo de rutas con las heurísticas de inserción de Solomon (1987) para rutas de largo horizonte de programación.

PALABRAS CLAVE: VRP con ventanas de tiempo, heurísticas de construcción de rutas, Solomon.

ABSTRACT

In this study, we studied and systematized five route construction heuristics that solve the most common VRPTW instances quickly and efficiently. The tool produces better results in terms of cost route to the Solomon insertion heuristic (1987) for long scheduling horizon routes.

KEYWORDS: VRP with time windows, route construction heuristics, Solomon.

1. INTRODUCCIÓN

El problema del ruteo de vehículos consiste en un conjunto de rutas de costo mínimo, originadas a partir del servicio de una flota de vehículos, que parten desde uno o más depósitos, hasta clientes finales distribuidos en una zona geográfica. Para desplazarse hacia los clientes, los vehículos usan una red de rutas que tienen asociadas variables de costo: costos de transporte, tiempos de recorrido, etc.

La red de rutas en las que se desplazan los vehículos, se representa mediante un grafo, donde los arcos del grafo representan caminos de un cliente *i* a un cliente *j*, y los nodos o vértices representan los clientes. Cada arco debe ser recorrido por un solo vehículo.

Cuando se determinan las rutas que deben seguir los vehículos, se halla una solución al problema. La solución cumple con los requerimientos y condiciones del modelo. [8]

Existen muchos tipos de problemas de ruteo de vehículos. Cada uno contiene características y restricciones propias de los procesos de distribución del mundo real. Las restricciones más comunes son: restricciones de capacidad, restricciones de tiempo de servicio, restricciones en el tamaño de la flota disponible, demanda estocástica, ventanas de tiempo, entre otros [1].

El Problema de Ruteo de Vehículos con Ventanas de tiempo, VRPTW (*Vehicle Routing Problem* por sus siglas en inglés), aparece como un área importante de investigación en la generalización y aproximación al problema real de los modelos de distribución física; incluye una dimensión espacial, y una dimensión de tiempo, bajo las cuales se obtiene una descripción más adecuada de la actividad logística de distribución; de esta manera la programación de las rutas también considera tiempos de espera en los que se incurre cuando un cliente es visitado muy temprano.

El problema de ruteo de vehículos es uno de los temas extensamente estudiados en el campo de la optimización combinatoria. Pertenece a la clase de problemas **NP**-Hard y se usan técnicas aproximadas, como las heurísticas o metaheurísticas, para hallar soluciones factibles en un tiempo de cómputo razonable.[13]

El presente artículo describe un conjunto de heurísticas de construcción de ruta para la solución del problema de ruteo de vehículos con ventanas de tiempo. Los algoritmos fueron implementados en Matlab ®, y para evaluar el desempeño, se usan las instancias de Solomon [9].

2. PROBLEMA DEL RUTEO DE VEHÍCULOS CON VENTANAS DE TIEMPO.

El Problema de Ruteo de Vehículos con Ventanas de Tiempo, "V*ehicle Routing Problem with time Windows*" (VRPTW), es una extensión del Modelo Básico de Ruteo (VRP). El VRPTW es un problema de decisión que consiste en diseñar un conjunto de rutas para una flota de vehículos de igual capacidad, que salen y llegan a un depósito, de modo que se visiten todos los destinos una sola vez con el costo mínimo, satisfaciendo restricciones horarias y de capacidad. [7]

Una ruta inicia cuando un vehículo sale del depósito y recorre cierta zona definida por las restricciones del problema y culmina en el momento que su capacidad es totalmente usada, o bien, la capacidad restante es insuficiente para servir a otro cliente. Cada ruta esta asociada al recorrido de un solo vehículo.

Los clientes están dispersos en un área geográfica y cada cliente tiene asociado un intervalo de tiempo, o Ventana de Tiempo, en el que permiten el servicio de recogida o despacho. Las ventanas de tiempo definidas para cada cliente, originan una espera si el vehículo llega antes del límite inferior de la ventana de tiempo de este cliente, e impiden el inicio del servicio si se supera su límite superior.

Las ventanas de tiempo del problema de ruteo pueden ser flexibles o suaves, o no flexibles o duras. Las Ventanas de Tiempo Suaves permiten violar la restricción de inicio de servicio dentro de las ventanas de los clientes, sin embargo, este hecho implica una penalización económica por incumplimiento que afecta negativamente la función de costo. Por otro lado, las Ventanas de Tiempo Duras no admiten el inicio del servicio después del tiempo más lejano en el que algún cliente puede ser atendido [5]

Para el presente estudio, la función de costo está definida por variables de tiempo de recorrido, equivalentes a los costos de transporte de las rutas, (es decir que la unidad de costo es equivalente a la unidad de tiempo de recorrido definido para ella); la matriz de costo es simétrica, y las ventanas de tiempo asociadas a los clientes son duras o inflexibles. Además, cada ruta es recorrida por solo un vehículo, es decir que el número de rutas finales, es el número final de vehículos usados.

La formulación matemática del VRPTW, según [14] se presenta a continuación:

$$Minimizar \sum_{k\in K}\sum_{(i,j)\in A} C_{ij}t_{i,jk} \qquad (\mathbf{1})$$

Sujeto a:

$$\sum_{k\in K}\sum_{j\in \Delta+(i)} \boldsymbol{x_{ijk}} = \mathbf{1} \;;\; \forall\; i \in N \quad (\mathbf{2})$$

$$\sum_{j\in \Delta+(\mathbf{0})} \boldsymbol{x_{0jk}} = \mathbf{1} \;;\; \forall\; k \in K \quad (\mathbf{3})$$

$$\sum_{i\in \Delta-(n+1)} \boldsymbol{x_{i,n+1,k}} = \mathbf{1} \;;\; \forall\, k \in K \quad (\mathbf{4})$$

$$\sum_{i\in \Delta-(j)} \boldsymbol{x_{ijk}} - \sum_{i\in \Delta+(j)} \boldsymbol{x_{jik}} = \mathbf{0} \;;\; \forall\; k \in K,\; j \in N \quad (\mathbf{5})$$

$$\boldsymbol{x_{ijk}}\left(\boldsymbol{w_{ik}} + \boldsymbol{s_i} + \boldsymbol{t_{ij}} - \boldsymbol{w_{jk}}\right) \leq \mathbf{0} \;;\; \forall\, k \in K,\; (i,j) \in A \quad (\mathbf{6})$$

$$a_i \sum_{j \in \Delta+(i)} x_{ijk} \leq w_{ik} \leq b_i \sum_{j \in \Delta+(i)} x_{ijk} \; ; \forall k \in K, \; i \in N \quad (7)$$

$$E_i \leq w_{ik} \leq L \; ; \; \forall \; k \in K, \; i \in \{0, n+1\} \quad (8)$$

$$\sum_{i \in N} d_i \sum_{j \in \Delta+(i)} x_{ijk} \leq Q \; ; \forall \; k \in K \quad (9)$$

$$x_{ijk} \geq 0 \; ; \; \forall \; k \in K, \; (i,j) \in A \quad (10)$$

$$x_{ijk} \in \{0, 1\} \; ; \; \forall \; k \in K, \; (i,j) \in A \quad (11)$$

La red de transporte se representa por medio del grafo completo $G = (V, A)$ donde $V = \{0, \dots, n\}$ es el conjunto de vértices y $A \in (i,j)$ es el conjunto de arcos, con $i \neq j$. Los vértices $i = 1, \dots, n$ corresponden a los clientes y los vértices cero (0) o $n+1$ corresponden al depósito.

La ventana de tiempo general asociada a los nodos de partida y de llegada se representa como $[a_0, b_0] = [a_{n+1}, b_{n+1}] = [E, L]$, donde E y L representan la salida del depósito más temprana posible y la llegada al depósito más tardía posible respectivamente.

El sistema de distribución consta de un conjunto de vehículos igual a K con capacidad limitada igual a Q toneladas. El tiempo de duración del servicio para cada cliente se conoce como s_i. La variable a_i, representa el tiempo más cercano de inicio de servicio en el cliente i. La variable b_i, es el tiempo más lejano de inicio de servicio, o fin de la ventana correspondiente al cliente i. El instante en el que el vehículo k, comienza a servir al cliente i, o cliente actual se conoce como W_i.

La función objetivo expresa el costo total; optimiza el tiempo total de recorrido $C_{ij} = \sum t_{ij}$, donde t_{ij} es el tiempo de recorrido entre dos clientes i e j El costo de la función objetivo es el tiempo de viaje asignado a cada ruta. Las variables binarias x_{ijk} indican si los clientes del arco $i - j$ son atendidos por el vehículo k. [10]

La restricción (2) asigna a cada cliente exactamente una ruta de vehículo, las restricciones (3) y (4) limitan el número de rutas por vehículo a una, y caracterizan el flujo que debe seguir la flota. La restricción (6) asegura que el vehículo k no pueda comenzar el servicio, si la suma del tiempo de viaje de i a j, la duración del servicio en i y el tiempo total acumulado al inicio del servicio en i (w_{ik}) es mayor que la ventana de tiempo de j o del cliente siguiente.

- Para cada arco $A \in (i,j)$, Se definen:
    - c_{ij}. Costo no negativo correspondiente a cada arco $A \in (i,j)$, y asociado a la distancia, tiempo, flota, o una combinación ponderada de ellas. Representa el costo del camino más corto recorrido entre dos nodos i, j.
    - C_{ij}. Representa el costo de la función objetivo asociado al tiempo de viaje asignado a cada ruta.
    - $t_{ij} =$ Tiempo de viaje para cada arco $A \in (i,j)$, que es equivalente a una unidad de costo. $C_{ij} = t_{ij}$.
    - Se debe cumplir la desigualdad triangular $c_{ij} \leq c_{ih} + c_{hj}$ y $t_{ij} \leq t_{ih} + t_{hj}$.
    - Variables de decisión de naturaleza binaria. $x_{ijk}, \; \forall \, i, j \, , \;\; 1 \leq i, j \leq n, \;\; i \neq j$.

$x_{ijk} = 1$, si en la solución, el vehículo k va de i a j.
$x_{ijk} = 0$, si en la solución, el vehículo k no va de i a j.

- Para cada nodo $V = \{0, \dots, n\}$ se definen:
    - s_i= Tiempo de duración del servicio para cada cliente.
    - $d_i =$ Demanda del nodo i, de tipo determinística.
    - Un intervalo de tiempo $[a_i, i], \; con \; i \in V$, llamado Ventana de Tiempo, dentro del cual un cliente puede ser servido.
    - La variable a_i, representa el tiempo más cercano de inicio de servicio en el cliente i.
    - La variable b_i, es el tiempo más lejano de inicio de servicio, o lo mismo, el fin de la ventana correspondiente al cliente i.
    - W_i, instante en el que el vehículo k, comienza a servir al cliente i, o cliente actual.
    - W_j, instante en el que el vehículo k, comienza a servir al cliente j o cliente siguiente.

3. INSTANCIAS.

Las instancias son un conjunto de datos que se usan para medir la bondad de los resultados de los métodos propuestos de optimización. Están asociadas a características propias del problema y se usan en el planteamiento o corrida específica de un modelo.

El conjunto propuesto por Solomon en 1983 es un recurso importante para la validación de técnicas de optimización para el VRPTW. El banco de pruebas consta de 56 instancias que se representan mediante la siguiente notación: C1 (09), C2 (08), R1 (012), R2 (011), RC1 (08) y RC2 (08). El número entre paréntesis representa la cantidad de instancias que existe en cada conjunto.

Los grupos de instancias difieren entre sí respecto al ancho de sus ventanas de tiempo, medido como la diferencia entre el fin e inicio de ventana, la ubicación geográfica de los clientes según sus coordenadas y la densidad de ventanas, entendido

como el porcentaje de clientes con ventanas de tiempo, algunas con 25%, 50%, 75% y 100% de clientes.

El conjunto de problemas **1**: R**1**, C**1**, y RC**1**, son de corto horizonte de programación; Los problemas **2**: R**2**, C**2** y RC**2**, son de largo horizonte de programación, es decir que muchos clientes pueden ser servidos por un mismo vehículo (hasta 30 clientes), generando menos rutas que en el conjunto de problemas 1 (entre 5 y 10).

Solomon (1987) desarrolló instancias con 100 clientes y 50 clientes para cada uno de los grupos R, C y RC. En las últimas investigaciones se han desarrollado problemas con menos clientes (10, 25 o 50) para cada uno de los grupos, al igual que problemas de mayor dimensión (200, 400, 800 y 1000 clientes). Para la presente investigación, su usó el conjunto de instancias propuesto por Solomon (1987) con 100 clientes.

4. HEURÍSTICAS DE CONSTRUCCIÓN DE RUTAS.

Debido a la importancia de los problemas de optimización combinatoria en aplicaciones prácticas, científicas e industriales, se han desarrollado múltiples métodos para resolverlos. Las técnicas desarrolladas se pueden clasificar en exactas o exhaustivas, y aproximadas.

Las técnicas exactas encuentran la solución óptima para cualquier instancia de un problema en un tiempo determinado. Para problemas ***NP***-Hard, el inconveniente de usar estos métodos, es el tiempo de cómputo necesario para determinar una solución óptima. Este tiempo crece exponencialmente con el tamaño del problema.

En su lugar, para instancias de tamaño grande, se han desarrollado métodos aproximados que permiten encontrar una solución factible, que en algunos casos puede considerarse cercana a la solución óptima, y que es posible hallarla en un tiempo razonable y con moderado recurso computacional. Dentro de los algoritmos aproximados se pueden encontrar dos tipos: las heurísticas y las metaheurísticas [2].

Las heurísticas son técnicas de indagación y descubrimiento mediante métodos no rigurosos que buscan "buenas" soluciones a los modelos de optimización combinatoria. A su vez, las heurísticas se pueden clasificar en heurísticas de construcción y de búsqueda local. Los métodos de búsqueda local parten de una solución inicial y, usando el concepto de vecindario, recorren parte del espacio de búsqueda hasta encontrar un óptimo. La base de las heurísticas de mejora, es el vecindario de búsqueda. Al hacer modificaciones de este vecindario, se pueden obtener soluciones que mejoren la función objetivo [10].

Por otro lado, las heurísticas de construcción o constructivas, crean una solución desde cero e incorporan varios componentes en las iteraciones del modelo hasta satisfacer todas las restricciones del problema. [6]

Para resolver el problema de ruteo de vehículos con ventanas de tiempo con las instancias propuestas, se estudiaron y programaron cinco de las heurísticas de construcción de rutas más estudiadas en la literatura: algoritmo de ahorros, algoritmo del vecino más cercano con enfoque temporal, heurística de inserción I1, heurística de inserción I2 y la heurística de inserción I3, que se describen a continuación:

4.1 Heurística de Ahorros.

Dantzig y Ramser [4], quienes fueron los primeros en estudiar el Problema de Ruteo de Vehículos, tomaron en cuenta la reducción en distancia que se podría generar al unir dos clientes en una ruta, en lugar de servir a cada uno de ellos en rutas diferentes. De tal reducción nace el concepto de "Ahorro" y el nombre del método heurístico.

Dos clientes i y j son servidos en rutas separadas, las rutas inician y finalizan en el depósito, denominado con valor cero (0), así: $(0{-}i-0)$ y $(0-j-0)$.

Las distancias del depósito al cliente y viceversa son: d_{io}, d_{oi}, d_{jo}, d_{oj} y los costos implicados : $C_{io}, C_{oi}, C_{jo}, C_{oj}$.

La distancia entre clientes es: d_{ij} y el costo asociado: C_{ij}.

El costo total para servir a los clientes i y j en rutas separadas es:

$$C_a = C_{oi} + C_{io} + C_{oj} + C_{jo}$$

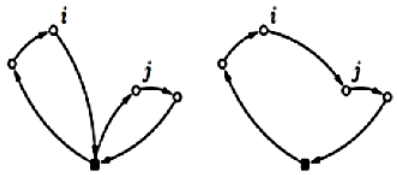
Si se considera unir a los dos clientes y servirlos en la misma ruta, causando la ruptura de los enlaces de cada uno con el depósito, el costo sería:

$$C_b = C_{oi} + C_{ij} + C_{jo}$$

La expresión anterior se busca generar un ahorro, que se obtiene al sustraer las dos expresiones anteriores:

$$\boldsymbol{Ahorro}: \boldsymbol{S_{ij}} = C_{oi} + C_{io} + C_{oj} + C_{jo} - (C_{oi} + C_{ij} + C_{jo})$$
$$\boldsymbol{S_{ij}} = \boldsymbol{C_{io}} + \boldsymbol{C_{oj}} - \boldsymbol{C_{ij}}$$

Figura 1. Definición del Algoritmo de Ahorros



Fuente: [3]

La heurística toma los pares de clientes cuya unión represente mayor ahorro de la función objetivo, y los asigna a la ruta conforme se respete la unión entre pares y se cumplan con las restricciones de capacidad y de tiempo del problema que se pretende solucionar [3].

4.2 Heurística del Vecino más cercano con enfoque temporal.

El algoritmo halla una solución basada en la cercanía de dos nodos o clientes adyacentes. En el enfoque temporal de la heurística del Vecino más Cercano, se usa una métrica o medida que hace un balance ponderado entre la cercanía geográfica de los clientes y el tiempo de recorrido respectivo de un nodo a otro.

En esta aproximación, un cliente cercano geográficamente no implica factibilidad en términos de tiempo, por esta razón el Costo de la ruta al insertar un cliente hace un balance entre estos dos parámetros, y asigna los clientes a la ruta dando prioridad a aquellos cuyo "balance" sea menor.

Si se tiene una ruta $(0, \dots u_i, , 0)$, se define el costo de insertar el cliente u_j a continuación de u_i en la ruta como:

$$C_{ij} = \delta_1 d_{ij} + \delta_2 T_{ij} + \delta_3 V_{ij}$$

$$T_{ij} = W_j - (W_i + s_i)$$

$$V_{ij} = b_j - (W_j + s_i + t_{ij})$$

Donde los parámetros δ_1, δ_2, y δ_3 son no negativos y suman 1.

- El término d_{ij}, es la distancia directa entre dos nodos y mide su cercanía geográfica (Se asume que cada unidad de distancia es equivalente a una unidad de tiempo).

- El valor de T_{ij} indica la diferencia entre la hora de comienzo del servicio en j y la del fin del servicio en i, midiendo la cercanía de los clientes en términos temporales. Este parámetro minimiza el menor tiempo de recorrido y el menor tiempo de espera entre dos clientes.

- Por otro lado, V_{ij} mide la urgencia de realizar la inserción. La urgencia se define como la diferencia entre la hora de arribo a j (sin incluir la espera) y la última hora a la que se podría arribar a dicho cliente. Este parámetro prioriza los clientes a insertar teniendo en cuenta la diferencia de tiempo más tardía para servir al cliente j.

El algoritmo inserta los clientes en las rutas calculando los costos C_{ij} partiendo del último cliente asignado a la ruta, desde y hacia cada uno de los clientes restantes y se listan de manera ascendente. En cada paso se selecciona al cliente no visitado que sea más cercano al último cliente de la ruta en términos del parámetro calculado, considerando solamente las inserciones factibles, y se agrega al final de la ruta parcialmente formada, hasta que todos los clientes sean ruteados.

4.3 Heurísticas de Inserción.

La Heurísticas de inserción de Solomon [9] proponen varios Criterios de Inserción que minimiza de forma ponderada el incremento de distancia y tiempo que supone la inclusión de un cliente. Estas métricas están relacionadas con los costos en los que se incurre al introducir un nuevo nodo.

Dado un conjunto de destinos que aún no han satisfecho su demanda y una ruta iniciada, debe elegirse algún criterio que permita intercalar al mejor nodo en el lugar adecuado del recorrido. Esta inclusión modifica los tiempos de llegada y de inicio del servicio de los clientes que se ven precedidos por el cliente insertado en último lugar.

Solomon propone tres reglas de inserción, que seleccionan los clientes bajo un parámetro C_2 y los inserta en los nodos más baratos según un parámetro C_1.

4.3.1 Heurística de inserción I1.

En esta heurística se considera la ventaja de visitar al cliente dentro de la ruta parcial y no en una ruta específica para él. Para este enfoque se definen:

$$C_1(i,u,j) = \alpha_1 C_{11}(i,u,j) + \alpha_2 C_{12}(i,u,j);\ \alpha_1 + \alpha_2 = 1$$

$$C_{11} = d_{iu} + d_{ij} - \mu\, d_{ij};\ 0 \leq \mu \geq 1$$

$$C_{12}(i,u,j) = W_{jnew} - W_j = PF\ (Push\ Forward)$$

$$C_2(i,u,j) = \lambda d_{0u} - C_1(i,u,j);\ 0 \leq \lambda \geq 1$$

- C_1 determina la posición más barata donde se puede insertar un nuevo cliente.

- C_{11} mide el ahorro en la distancia si se insertara u entre i e j.

- C_{12} tiene en cuenta el retardo que provoca insertar un cliente nuevo en una ruta parcialmente formada. El tiempo adicional de inserción es llamado también *Push Forward.*

- C_2 selecciona el cliente a insertar tomando el valor C_1 y la distancia del cliente al depósito, de esta manera se privilegia a los clientes para los que sería demasiado crear una ruta individual. El mejor lugar de inserción es aquel que minimice una combinación ponderada entre la extra-distancia y extra-tiempo de inserción.

- Los coeficientes α_1 y α_2, determinan el enfoque de la heurística. Si en la inserción pesan variables temporales, entonces se dará mayor ponderación a

α_2, y si es más importante optimizar la distancia, se dará prioridad a α_1.

4.3.2 Heurística de inserción I2.

En este caso se intenta seleccionar el cliente que, si fuera insertado en la ruta, minimiza una medida de la distancia y el tiempo total de la misma. Para la Heurística de Inserción I2 se tiene que:

$$C_1(i,u,j) = \alpha_1 C_{11}(i,u,j) + \alpha_2 C_{12}(i,u,j);\ \alpha_1 + \alpha_2 = 1$$

$$C_{11} = d_{iu} + d_{ij} - \mu\, d_{ij};\ 0 \leq \mu \geq 1$$

$$C_{12}(i,u,j) = W_{jnew} - W_j = PF\ (Push\ Forward)$$

$$C_2(i,u,j) = \beta_1 Rd(u) + \beta_2 Rt(u);\ 0 \leq \beta_1 \geq 1;\ 0 \leq \beta_2 \geq 1$$

- Aquí, C_1 es igual que en I1 y $C_2(i,u,j) = \beta_1 Rd(u) + \beta_2 Rt(u)$ donde $Rd(u)$ y $Rt(u)$ son el tiempo y la distancia total de la ruta si u es insertado entre i y j. En este caso, el cliente seleccionado es el que minimiza C_2, contrariamente a los demás casos en que se busca maximizar la medida C_2.

- Los parámetros β_1 y β_2; determinan las características del modelo. Mayores valores de β_1 tienen en cuenta restricciones geográficas, por otro lado, β_2 guía a la heurística a priorizar las restricciones temporales en la solución.

4.3.3 Heurística de Inserción I3.

En la última Heurística de inserción de Solomon, se intenta seleccionar a los clientes cuyos límites en las ventanas de tiempo son cortas para garantizar su servicio, es decir aquellos clientes que tienen urgencia de ser servidos. En este enfoque se define:

$$C_1(i,u,j) = \alpha_1 C_{11}(i,u,j) + \alpha_2 C_{12}(i,u,j) + \alpha_3 C_{13}(i,u,j);\ \alpha_1 + \alpha_2 + \alpha_3 = 1$$

$$C_{11} = d_{iu} + d_{ij} - \mu\, d_{ij};\ 0 \leq \mu \geq 1$$

$$C_{12}(i,u,j) = W_{jnew} - W_j = PF\ (Push\ Forward)$$

$$C_{13}(i,u,j) = b_j - W_j$$

$$C_2(i,u,j) = C_1(i,u,j)$$

- C_{11} y C_{12} son los mismos que en I1, solo que en este caso, se agrega un nuevo término a C_1; C_3; que mide qué tan cerca del final de su ventana de tiempo se arribaría al cliente u si este fuera insertado en la ruta.

- Los coeficientes α_1, α_2 y α_3 guían a la heurística en tres distintos enfoques: espacial, temporal y de urgencia para servir a los clientes, respectivamente.

- Finalmente, se utiliza C_1 para seleccionar el próximo cliente a insertar.

5. RESULTADOS COMPUTACIONALES

Para analizar los rendimientos de las heurísticas, se diseñó un toolbox en el Software Matlab versión R2010a, en un equipo estándar con procesador Intel Core i3 con 3 GB de memoria RAM instalada.

La calidad de la solución es medida en términos del mínimo costo de la ruta, que representa la distancia directa recorrida entre el cliente i y el cliente j. Para el presente estudio, una unidad de distancia es recorrida en una unidad de tiempo.

Las **tablas 1 a 6**, muestran los mejores resultados promedio de cada uno de los grupos de instancias usando algunos parámetros sugeridos en Solomon [9].

Cada una de las tablas contiene la mejor corrida promedio y el conjunto de parámetros usados en esa corrida. En la sección derecha de las tablas se relacionan las desviaciones porcentuales respecto a los mejores promedios encontrados en la literatura.

Para el conjunto de instancias R1 y C1, se usaron cinco heurísticas estudiadas anteriormente. El conjunto de instancias RC1, R2, C2, y RC2, fueron corridos con solo con cuatro de las cinco heurísticas: Vecino más cercano, I1, I2 e I3.

Tabla 1. Resultados promedio grupo de instancias R1.

Algoritmo	Parámetros		Resultados (Promedio)			Porcentajes de desviación respecto a los mejores resultados promedio. (Solomon 1987)		
			Costo	Tiempo	No. Rutas	Costo	Tiempo	No. Rutas
I1	α1	1	1434.6	2808.5	14.5	-0.13%	4.19%	6.62%
	α2	0						
	μ	1						
	λ	2						
I2	α1	1	1703.85	3002.24	15.17	3.98%	3.95%	4.60%
	α2	0						
	μ	1						
	β1	0.5						
	β2	0.5						
I3	α1	0.5	1708.29	3019.38	15.42	3.47%	5.75%	9.34%
	α2	0.5						
	α3	0						
	μ	1						
Ahorros	λ	1	2719.27	6803.13	48.33	81.42 %	118.31 %	191.16 %
Vecino	δ1	0.4	1965	3836.09	21.92	22.81 %	29.21%	51.17%
	δ2	0.4						
	δ2	0.2						

Fuente: Autores

Se evidencia en la Tabla 1 que la heurística de inserción I1 es la de mejor desempeño entre el grupo de heurísticas programadas e incluso entrega un mejor resultado comparado con los de la literatura, su desviación porcentual del 0.13% por debajo del de referencia, con un tiempo de programación de 4.19%, respecto al mejor resultado promedio obtenido por la misma heurística en Solomon 1987.

Tabla 2. Resultados promedio grupo de instancias C1.

Algoritmo	Parámetros		Resultados (Promedio)			Porcentajes de desviación respecto a los mejores resultados promedio. (Solomon 1987)		
			Costo	Tiempo	No. Rutas	Costo	Tiempo	No. Rutas
I1	α1	1	1110.68	10528.18	10.67	16.68 %	4.20%	6.67%
	α2	0						
	μ	1						
	λ	1						
I2	α1	0.5	1466.37	11022.44	11.33	39.68 %	8.34%	12.21 %
	α2	0.5						
	μ	1						
	β1	0.5						
	β2	0.5						
I3	α1	0.5	1454.43	11122.33	11.67	31.83 %	9.31%	16.67 %
	α2	0.5						
	α3	0						
	μ	1						
Ahorros	λ	0.6	2841.60	25674.11	47.11	191.09 %	130.78 %	302.66 %
Vecino	δ1	0.4	1906.84	12855.78	16.67	62.84 %	22.76 %	63.40 %
	δ2	0.4						
	δ2	0.2						

Fuente: Autores

Respecto a las instancias de distribución por clusters de la Tabla 2, se muestran los resultados de la heurística de inserción I1 con mayor desempeño aunque, no tan eficiente como en R1. La desviación fue del 16,68% respecto al mejor costo promedio con un tiempo de programación desviado en 4,20% a la de Solomon.

Tabla 3. Resultados promedio grupo de instancias RC1.

Algoritmo	Parámetros		Resultados (Promedio)			Porcentajes de desviación respecto a los mejores resultados promedio. (Solomon 1987)		
			Costo	Tiempo	No. Rutas	Costo	Tiempo	No. Rutas
I1	α1	1	1598.71	2911.04	14.25	0.14%	4.90%	5.56%
	α2	0						
	μ	1						
	λ	1						
I2	α1	1	2031.20	3269.09	15.75	8.37%	7.91%	10.92 %
	α2	0						
	μ	1						
	β1	0.5						
	β2	0.5						
I3	α1	0.5	1978.99	3190.04	15.63	6.99%	5.84%	11.61 %
	α2	0.5						
	α3	0						
	μ	1						
Vecino	δ1	0.4	2791.63	4581.04	27.38	55.09 %	49.84 %	92.78 %
	δ2	0.4						
	δ2	0.2						

Fuente: Autores

Tabla 4. Comparación de resultados. Instancias R2

Algoritmo	Parámetros		Resultados (Promedio)			Porcentajes de desviación respecto a los mejores resultados promedio. (Solomon 1987)		
			Costo	Tiempo	No. Rutas	Costo	Tiempo	No. Rutas
I1	α1	1	1334,01	2857	3,64	-4,88 %	10,82 %	10,19 %
	α2	0						
	μ	1						
	λ	2						
I2	α1	1	1437,80	2872,87	3,82	-2,24 %	8,58 %	-15,70 %
	α2	0						
	μ	1						
	β1	0.5						
	β2	0.5						
I3	α1	0.5	1493,34	2805,82	3,64	1,27 %	4,84 %	6,95%
	α2	0.5						
	α3	0						
	μ	1						
Vecino más cercano	δ1	0.5	1437,24	3674,18	5,36	-2,4%	35,1 %	57,8%
	δ2	0.5						
	δ2	0						

Fuente: Autores

Tabla 5. Comparación de resultados. Instancias C2

Algoritmo	Parámetros		Resultados (Promedio)			Porcentajes de desviación respecto a los mejores resultados promedio. (Solomon 1987)		
			Costo	Tiempo	No. Rutas	Costo	Tiempo	No. Rutas
I1	α1	1	748,51	10006,87	3,62	8,06	0,86	16,94
	α2	0						
	μ	1						
	λ	1						
I2	α1	1	981,10	11667,62	3,87	6,47	14,94	13,97
	α2	0						
	μ	1						
	β1	0.5						
	β2	0.5						
I3	α1	0.5	1096,44	11220	3,87	2,21	10,89	10,71
	α2	0.5						
	α3	0						
	μ	1						
Vecino más cercano	δ1	0.5	885,34	13149,11	4,75	-8,1	21,9	35,7
	δ2	0.5						
	δ2	0						

Fuente: Autores

Los costos arrojados por la experimentación son cercanos a los costos de referencia para el conjunto de instancias C2. La heurística del vecino más cercano es la más eficiente, puesto

que mejora el resultado en 8.1% respecto al mejor resultado encontrado

Tabla 6. Comparación de resultados. Instancias RC2

Algoritmo	Parámetros		Resultados (Promedio)			Porcentajes de desviación respecto a los mejores resultados promedio. (Solomon 1987)		
			Costo	Tiempo	No. Rutas	Costo	Tiempo	No. Rutas
I1	α1	1	1663.02	3174,87	4.12	-1,13	7,43	5,77
	α2	0						
	μ	1						
	λ	2						
I2	α1	1	1972.16	3283	4.12	9,71	4,94	0,61
	α2	0						
	μ	1						
	β1	0.5						
	β2	0.5						
I3	α1	0.5	1884,97	3289	4.25	3,78	4,44	6,25
	α2	0.5						
	α3	0						
	μ	1						
Vecino más cercano	δ1	0.5	1744.37	4014.52	6.37	-0,6	28,7	63,5
	δ2	0.5						
	δ2	0						

Fuente: Autores

Las Instancias RC2 en general presentan valores de error bajos en todas las heurísticas estudiadas, especialmente las heurísticas I1 y vecino más cercano se destacan presentando mejores resultados que los de la literatura de comparación. En resumen, se puede inferir que el costo, tiempo de recorrido y número de rutas encontrados en la experimentación realizada son considerablemente cercanos a los encontrados en la literatura; en la Tabla 6, puede apreciarse con detalle.

6. CONCLUSIONES

Las instancias de corto horizonte de programación generan rutas casi exclusivas o de pocos clientes, las ventanas de tiempo son estrictas porque sus intervalos de duración son pequeños y en ocasiones se encuentran superpuestas entre clientes. De acuerdo a lo anterior, la experimentación arrojó soluciones notablemente favorables, sobretodo en el caso de la heurística I1 la cual mejora los resultados frente a los de la literatura de referencia; aunque otras heurísticas presentan valores mayores en costo con respecto a los de comparación, son aceptables y provechosos porque se encuentran bastante cercanos con porcentajes de dispersión significativamente pequeños.

El ventajoso desempeño que tiene la heurística I1 en todos los grupos de instancia de corto horizonte (R1, C1 y RC1) puede presentarse porque tiene en cuenta en cada una de sus iteraciones un aspecto espacial y uno temporal, que ponderados puede significar disminuciones en distancia recorrida y en tiempos de espera simultáneamente.

La heurística que peor desempeño tiene es la de ahorros dado que presenta grandes porcentajes de desviación. Esta situación se debe al diseño del algoritmo de ahorros que considera la reducción del impacto de los tiempos de espera cuando se inicia cada una de las rutas y continúa en sus iteraciones examinando la factibilidad temporal de los clientes a insertar, respetando la prioridad de ahorro sin dar algún tipo de prioridad al aspecto temporal en la selección de sus clientes a rutear.

Una recomendación favorable para obtener mejores resultados en la heurística de ahorros, es considerar en cada una de las iteraciones un criterio limitante del tiempo de espera. El ahorro que se causa unido a la prioridad de la disminución de los tiempos de espera en la selección de los clientes que se asignan a la ruta, puede ser un procedimiento significativo en la reducción del costo total.

En lo concerniente a instancias de largo horizonte de programación, puede observarse que los errores no arrojan grandes cifras, de manera que permitan afirmar que hay diferencias significativas entre los valores obtenidos y los mejores. Aunque algunos resultados obtenidos durante la experimentación, son superiores (en términos de costo), algunos otros mejoraron en comparación con los de Solomon (1987). Las heurísticas I1, I2 y la de vecino más cercano, resaltan como las más eficientes con muy buenos valores promedio, ubicándose por debajo de los valores referencia.

Con respecto a la heurística I1, pueden distinguirse un comportamiento guiado por los parámetros α_1 y α_2, el primero tiene un efecto en la distancia adicional que se agrega en la ruta al insertar al cliente u y el segundo, en el tiempo adicional por visitarlo. Cuando los parámetros se ponderan, el algoritmo prioriza los efectos generando rutas diferentes en cada ponderación. Se observa que cuando se prioriza la distancia sobre el tiempo se obtienen costos más bajos.

En la heurística del vecino más cercano se encuentran relacionados tres criterios importantes: el criterio espacial, el temporal y el de urgencia de atención; la heurística obtiene un buen desempeño cuando se da mayor ponderación al criterio espacial y temporal por encima del criterio de urgencia

Lo anterior debido al largo horizonte de programación de las instancias de tipo R2, C2 y RC2; las ventanas de tiempo asociadas a los clientes son mucho más anchas y permiten mayor holgura en la asignación de clientes a la ruta, logrando que la ruta se enfoque en buscar clientes más cercanos y sin tener mucho impacto en los tiempos de espera.

7. REFERENCIAS

[1] BRAMEL, J. A Location Based Heuristic for General Routing Problems. Operation Research. Vol. 44, No. 3. Pp. 501-509.

[2] CHRISTOFIDES, S. An Algorithm for the Vehicle Dispatching Problem. Operational Research Quartely 20:309-

318. Disponible en Disponible en URL: http://www.jstor.org/stable/170697 Consultado: 12/01//2011 10:41

[3] CLARKE, G. y J. W. Wright (1964). "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". *Operations Research 12, 568-561.*

[4] DANTZIG, G.B.; RAMSER, J.H. The Truck Dispatching Problem. Management Sci. 6 (1959), p. 80-91.

[5] DULLAERT, W. 2000. "Impact of relative route length on the choice of time insertion criteria for insertion heuristics for the vehicle routing problem with time windows". B. Maurizio, ed. Proc. Rome Jubilee 2000.

[6] FOISY, C., J.-Y. Potvin. 1993. "Implementing an insertion heuristic for vehicle routing on parallel hardware". Comput.Oper.Res. 20 737–745.

[7] KOLEN. A.W.J., Vehicle Routing with Time Windows. Operation Research, Vol 7. No. 2.

[8] LAPORTE, G. The Vehicle Routing Problem: An Overview of Exacts and Approximate Algorithms. European Journal of Operational Research. 59, 1992. Pp. 345-358.

[9] MARIUS M**.** Solomon, Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Tomado de: Operations Research, Vol. 35, No. 2 (Mar. - Apr., 1987), pp. 254-265 Publicado por: INFORMS. Disponible en URL: http://www.jstor.org/stable/170697 Consultado: 28/11//2010 10:41

[10] MINGOZZI A. Dynamic Programming Strategies for the Traveling Salesman Problem with Time Windows and Precedence Constraints. Operation Research. 1997, Vol 45. No. 3.

[11] POTVIN, J.-Y., J.-M. Rousseau. 1995. "An exchange heuristic for routeing problems with time windows". J.Oper.Res.Soc. 46 1433–1446.

[12] RUSSELL, R. A. 1995. "Hybrid heuristics for the vehicle routing problem with time windows". Transportation Sci. 29 156–166.

[13] SAVELSBERGH, M. W. P. 1986. "Local search in routing problems with time windows". Ann.Oper.Res. 4 285–305.

[14} TOTH Paolo, Daniele Vigo. "The vehicle routing problem". SIAM monographs on discrete mathematics and applications. Philadelphia 2002.