“MÉTODOS HEURÍSTICOS PARA LA SOLUCIÓN DE PROBLEMAS DE RUTEO DE VEHÍCULOS CON CAPACIDAD (CVRP)”

CLAUDIA MARCELA CONTRERAS PINTO
MARÍA FERNANDA DÍAZ DELGADO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
BUCARAMANGA
2010

“MÉTODOS HEURÍSTICOS PARA LA SOLUCIÓN DE PROBLEMAS DE RUTEO DE VEHÍCULOS CON CAPACIDAD (CVRP)”

CLAUDIA MARCELA CONTRERAS PINTO
MARÍA FERNANDA DÍAZ DELGADO

Trabajo de investigación en modalidad “Tesis de Grado” presentado como requisito parcial para optar al título de Ingeniera Industrial

Director
HENRY LAMOS
MATEMÁTICO
Ph.D. en Física-Matemática

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
BUCARAMANGA
2.010

AGRADECIMIENTOS

A todos aquellos que contribuyeron a hacer esto posible.

Henry Lamos Díaz, Javier Eduardo Arias Osorio, Miguel Jaimes y Carlos Ojeda.

DEDICATORIA

A mi hijo Hugo Santiago Martínez Díaz y a mi esposo Hugo Ernesto Martínez.

María Fernanda

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

TITULO:
"METODOS HEURÍSTICOS PARA LA SOLUCIÓN DE PROBLEMAS DE RUTEO DE VEHÍCULOS CON CAPACIDAD (CVRP)"*

AUTORES:
CONTRERAS PINTO, Claudia Marcela
DÍAZ DELGADO, María Fernanda**

PALABRAS CLAVES:
Ruteo de vehículos, Heurística, Método Clarke and Wright, Método del vecino más cercano

DESCRIPCIÓN:

El problema de ruteo de vehículos con capacidad ha sido de gran interés a través de los años debido a que representa un ahorro significante para las empresas en la entrega de sus pedidos, este tipo de problemas consiste en encontrar la ruta óptima de un conjunto de rutas con un costo mínimo de ruteo para una flota de vehículos idénticos con capacidad, que pueda visitar a todos los clientes requeridos y satisfacer sus demandas.

Desde un punto de vista matemático, el CVRP es un problema de optimización combinatoria que permite establecer diferentes rutas desde un mismo punto de origen hacia todos y cada uno de los nodos del problema, terminando su recorrido en el punto de partida, cumpliendo con una serie de restricciones sujetas a la capacidad del vehículo.

Los algoritmos estudiados en este proyecto son el algoritmo de ahorros de Clarke and Wright y el algoritmo del vecino más cercano. Se resuelven ciertas instancias por los dos métodos para comparar resultados y determinar que en la mayoría de los casos el algoritmo de ahorros de Clarke and Wright presenta menores costos que el del vecino.

Dentro del desarrollo del proyecto se hace un enfoque puntual en el algoritmo de ahorros para implementar una mejora donde se usa un parámetro llamado forma de la ruta para incrementar los buenos resultados del algoritmo y previene la formación de rutas circulares.

Una vez comparados todos los resultados, ahora con el algoritmo de ahorros mejorado, en el software H-CVRP, que fue creado para facilitar cálculos de este tipo para instancias con más de 50 nodos, se comprueba que el algoritmo de ahorros presenta mejores resultados con el parámetro forma de la ruta en la mayoría de las veces que el algoritmo de ahorros sin mejora y que el del vecino más cercano.

---

* Proyecto de grado.

** Facultad de Ingenierías Físico mecánicas. Escuela de Estudios Industriales y empresariales. Director Ph.D Henry Lamos Díaz.

ABSTRACT

TITLE:
"HEURISTICS METHODS FOR THE SOLUTION OF CAPACITATED VEHICLE ROUTING PROBLEM (CVRP)"*

AUTORS:
CONTRERAS PINTO, Claudia Marcela
DÍAZ DELGADO, María Fernanda**

KEY WORDS:
Vehicle Routing, Heuristic, Method Clarke and Wright, Method Nearest Neighbor.

DESCRIPTION:

The capacitated vehicle routing problem has been interesting thru the years due to it represent a significant save to the companies in the delivery of products, this particular problems consist in to find the optimal route from a set of routes with minimum routing costs for a fleet of identical vehicles with capacity that can visit all customers required and satisfy its demands.

Since a mathematical point of view, the CVRP problem is a combinatory optimization problem which allow to define different routes from the same point of inception to one and all nodes of the problem, ending the route in the same point where it begins, accomplishing with a quantity of requirements, like each customer can be visited just once, and just for only one vehicle and the rest of the capacity restrictions subject to each vehicle´s capacity.

The studied algorithms in this work were the saving algorithm of Clarke and Wright and the nearest neighbor algorithm. They both are used to solve the same instances to compare the results and decide that in the most of the times the saving algorithm of Clarke and Wright presents lower costs than the nearest neighbor.

During the development of the project, we punctually focus in the saving algorithm to implement an enhancement where it use a new parameter, called the route shape parameter, to increase the good results obtained by the algorithm and prevent the formation of circumferenced routes.

Once are compared all the results, now with the improved saving algorithm, in the H-CVRP software, which was created to facilitate the calculation of these kind of instances with more than 50 nodes, it proves that the saving algorithms presents better results with the route shape parameter mostly all the times, than the original saving algorithm and the nearest neighbor.

---

* Graduation Project.

** Fisical Mecanical Engineering Faculty. School of Industrial and Bussiness Studies. Director Ph.D Henry Lamos Díaz.

INTRODUCCIÓN

La administración y gerencia de las empresas a lo largo de la historia se han enfocado en disminuir sus costos para obtener mayores beneficios y así mismo brindar productos y/o servicios de la mejor calidad y en los menores tiempos de entrega. La logística de la distribución, el reparto y la entrega, desde el transporte y su capacidad, hasta las rutas y la cantidad de vehículos que se precisan para su cumplimiento son de vital importancia en las empresas. Se considera que en estas actividades se incurren en los mayores costos dentro de las compañías.
Con este proyecto se busca consolidar la línea métodos de solución para problemas de ruteo de vehículos que tiene actualmente el Grupo OPALO. El proyecto consistirá en programar algoritmos heurísticos para la solución del CVRP en Matlab. Naturalmente este tipo de investigación es de interés para aquellas empresas en las cuales dentro de sus procesos se encuentran el de distribución y comercialización de productos, así como empresas que usan intermediaros para la entrega del bien o servicio producido, o bien para aquellas en las que su fin es llevar desde uno o varios depósitos la mercancía directamente a sus clientes (nodos).

Con el software programado se podrá resolver el problema CVRP para instancias de tamaño moderadamente grande, esto es, con un número de clientes superior a 50 y menor a 300.

En el documento se hará una breve exposición sobre temas como la complejidad de un algoritmo, las clases P y NP, problemas NP complejos y problemas NP-difíciles. Además, se presentará a través de un ejemplo la diferencia entre algoritmos aproximados y heurísticos. Por último se escribe un manual básico para la programación de un toolbox en Matlab.

1. ANTECEDENTES DEL PROYECTO

El problema del CVRP (Capacitated Vehicle Routing Problem – Problema de Ruteo de Vehículos con Capacidad) definido hace más de 40 años, ha sido uno de los más desafiantes que presentan los problemas de optimización combinatoria.
Surge a partir del TSP (Travelling Salesman Problem – Problema del Agente Viajero), pero es Dantzing y Ramser quienes describen y formulan por primera vez el VRP (Vehicle Routing Problem – Problema de Ruteo de Vehículos) despertando un interés por su práctica utilidad, así como por su considerable dificultad.

En la literatura surgen muchas aplicaciones interesantes para el VRP y debido a esta gran cantidad de aplicaciones se despertó la curiosidad de muchos y de allí se crearon una serie de variaciones para un mejor planteamiento del problema según la necesidad. Una de las variaciones del VRP fue el CVRP, donde se tiene en cuenta la capacidad del vehículo.

Gran cantidad de industrias en numerosos países desarrollaron este problema para darle solución a la entrega de sus pedidos. Un ejemplo de esto fue la industria de la recolección de basuras, en la que a principios de 1970's a partir de la publicación de Beltrami y Bodin, quienes se enfocaron en la eficiencia de los vehículos de las rutas que recolectaban basuras en las zonas residenciales, basaron la asignación de sus recorridos no solo en las rutas, también en los horarios para programar las visitas que se debían realizar semanalmente. De allí se despertó el interés de muchos por evolucionarla, como lo fueron Russell e Igo, Christofides y Beasley, Gribbin, Golden, entre muchos otros.

La asignación de rutas para los vehículos vacíos y las demandas estimadas de las calles, desencadenaron un desafiante y enriquecedor problema de ruteo. Esta desafiante tarea llevó al planteamiento de diferentes heurísticas para brindar

solución al VRP, incurriendo en mejoras para el algoritmo de ahorros de Clarke and Wright.

Hasta el día de hoy siguen adelantando investigaciones sobre el tema para poder brindar soluciones más prácticas y eficientes al problema.

A continuación se presentan dos proyectos realizados en años anteriores por estudiantes de La Universidad Industrial de Santander, relacionados con la investigación y desarrollo del problema del CVRP.

- **TITULO:** Formulación y evaluación de un algoritmo, basado en la meta - heurística "Búsqueda Tabú" para la optimización del ruteo de vehículos con capacidad.

AUTORES: Alfredo José Pertuz Montenegro, Kimberly Rojas Silva.

ALCANCE: El algoritmo desarrollado es una herramienta que permite la solución del problema de ruteo de vehículos con capacidad, fue implementado en un programa de computadora desarrollado en la plataforma java a través del cual se calculan las soluciones dependiendo de ciertos parámetros iníciales como número de iteraciones, tamaño de la lista tabú, tamaño de la población y numero de intercambios. Este programa arroja la mejor ruta y su debida descripción que incluye costos parámetros iníciales y la secuencia en la cual deben ser atendidos los clientes con sus respectivos datos. Este proyecto está dirigido hacia el sector de la investigación a nivel universitario, específicamente en el área de logística e investigación de operaciones, con el propósito que éste pueda ser utilizado como un punto de partida para futuras generaciones interesadas en esta rama del conocimiento.

OBJETIVO: Formular y evaluar un algoritmo que utilice la metodología de la meta-heurística "búsqueda tabú" para solucionar el problema de ruteo CVRP e implementarlo en un programa de computadora para su uso. Año 2.007. Universidad Industrial de Santander.

- **TITULO:** Colonia de hormigas, fundamentación y aplicación en la optimización de problemas logísticos de ruteo con intervalos de recepción y tiempo de atención máximo. Año 2.005. Universidad Industrial de Santander.

AUTOR: José Luis Tolosa Barón.

ALCANCE: Este trabajo expone el uso de la metodología conocida como colonia de hormigas para el desarrollo de problemas de ruteo multiobjetivo. La metodología aplicada contempla una revisión extensiva bibliográfica de los problemas de ruteo, la creación de un modelo matemático, la creación de un algoritmo matemático, el estudio estadístico encaminado a determinar los parámetros que más influyen en la calidad de las respuestas obtenidas por el algoritmo y la determinación de los valores óptimos para los cuales el desempeño de los algoritmos es mayor.

OBJETIVOS: Desarrollar un modelo general para la optimización de sistemas logísticos de ruteo con intervalos de recepción y control del tiempo de atención máximo basado en la metodología heurística de colonia de hormigas como una muestra del potencial de esta metodología en la solución de problemas de trasporte y enrutamiento.

1.1. Justificación

Para el Grupo Ópalo (Grupo de Optimización y Organización de Sistemas Productivos, Administrativos y Logísticos) de la escuela de Estudios Industriales y

Empresariales de la UIS, es importante desarrollar conocimiento en diferentes líneas de investigación, que agreguen valor, no solo en la parte de extensión de los proyectos, además, en el desarrollo de las actividades académicas y formación de profesionales idóneos del programa de Ingeniería industrial de la UIS.

Este trabajo de investigación, se basó en una línea específica del grupo, la optimización de sistemas logísticos en su parte de ruteo, dada la importancia del tema para la sociedad en general, debido a los ahorros que se pueden obtener con el uso de diferentes métodos para la asignación y mejora de rutas de distribución. El producto final de este trabajo de investigación, es una herramienta de software que permite calcular las rutas de instancias desde 50 hasta 300 nodos, para el problema de ruteo de vehículos con capacidad (CVRP). Esta herramienta busca optimizar el tiempo de asignación de las rutas, obteniendo soluciones factibles al problema. Además, el estudio se convierte en ayuda para los estudiantes de la escuela, y la universidad en general, interesados en seguir profundizando en el aspecto logístico de distribución.

1.2. Planteamiento del problema

El CVRP es un problema de programación entera, que pertenece a la categoría de problemas NP-Hard, esto es, el tiempo computacional que se requiere para hallar una solución mediante un algoritmo crece de manera exponencial dependiendo de la magnitud del problema a resolver. Para problemas de magnitud pequeña los métodos exactos son una buena alternativa, sin embargo, para problemas de magnitud moderada y grande el costo computacional es muy elevado. El uso de técnicas heurísticas y meta-heurísticas han logrado resolver instancias de magnitud grande, de problemas de ruteo, en un tiempo computacional que se considera bajo respecto al tiempo en el que incurren los algoritmos exactos. En la literatura existen numerosas investigaciones dedicadas

al uso de las técnicas heurísticas para la solución del CVRP que parten de un mismo problema, que es el cómo definir las rutas de los vehículos, teniendo en cuenta su capacidad y la demanda de los clientes a visitar. Las heurísticas aunque arrojan soluciones factibles, no brindan certeza de que la solución encontrada, sea realmente la óptima.

Debido a los altos costos computacionales generados por la aplicación de técnicas heurísticas para problemas de magnitud grande, se tomaron en cuenta los conceptos de las heurísticas de ahorros de Clarke and Wright y del Vecino más cercano para desarrollar un toolbox, creado en Matlab, que pretende disminuir el tiempo computacional para instancias de gran magnitud y encontrar una solución más práctica y eficiente que permita el cumplimiento de la función objetivo del problema de ruteo, que es la de minimizar el costo total de los arcos recorridos.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar algunas técnicas heurísticas para la solución del problema de ruteo de vehículos con capacidad (CVRP) en Matlab.

1.3.2. Objetivos específicos

- Estudiar los métodos de solución de problemas de ruteo de vehículos con capacidad CVRP.

- Seleccionar y comparar los métodos Heurísticos que se usan para la solución del problema CVRP.

- Desarrollar el Toolbox en MATLAB correspondiente a los métodos heurísticos elegidos para la solución de problemas de ruteo de vehículos con capacidad CVRP.

- Elaborar el manual de usuario del Toolbox desarrollado.

1.4. Alcance

Al finalizar este trabajo de investigación, se habrá desarrollado un toolbox en Matlab con su respectivo manual, que brinda solución al problema del CVRP (Problema de Ruteo de Vehículos con Capacidad - Capacited Vehicle Routing Problem), por medio de los algoritmos del Vecino más Cercano y del algoritmo de ahorros de Clarke and Wright. Este producto final (Toolbox) servirá como herramienta de aprendizaje para futuras generaciones que requieran profundizar en temas de programación matemática, específicamente, para el problema de ruteo.

En el proyecto se llevaran a cabo todas las etapas que se requieren en una investigación formativa. Los temas de algoritmos heurísticos es un buen pretexto para conseguir esta formación en el mundo de la investigación.
Además de ser una herramienta de apoyo para la Escuela de estudios Industriales y Empresariales, este trabajo de investigación quiere abrir la posibilidad de incluir los métodos heurísticos en el desarrollo de la temáticas de optimización tratadas en las aulas.

2. MARCO TEÓRICO

2.1. Problemas combinatorios

Los problemas combinatorios son de tipo NP-Hard y existen diferentes tipos de problemas representativos, como el de la asignación de recursos, balanceo de líneas, de ruteo, programación de vehículos entre otros. El uso de problemas combinatorios se ha desarrollado en 6 áreas principales: El Problema de Asignación, el Problema de Transporte, el Teorema de Menger y el Flujo Máximo (Maximun Flow), el Árbol más Corto para Recorrer (Shortest Spanning Tree), el Camino más Corto (Shortest Path) y el Problema del Agente Viajero (TSP, Travelling salesman problem).

Los problemas combinatorios constan de los siguientes elementos:

1. Un conjunto de variables de decisión (Variables independientes).
2. Una función objetivo (F.O.) que mide la efectividad de cada sistema de decisiones.
3. Un conjunto de restricciones que representan las limitaciones bien sea de capital, recursos etc.

Los problemas combinatorios tratan de encontrar un objeto entre un conjunto finito de posibilidades. Este conjunto finito se define como la delimitación del problema de optimización. Los objetos pueden ser números naturales, estructuras de grafo o permutaciones, por lo tanto, debido a su naturaleza discreta llegan a ser muy complejos para encontrar la solución.

Dentro de la categoría de los problemas combinatorios, se encuentra el problema VRP. Este problema consiste en encontrar la ruta con el menor costo, usando la

cantidad mínima de vehículos para visitar a un conjunto de clientes distribuidos geográficamente.

Algunas variantes del VRP son:

- CVRP, Problema de Ruteo de Vehículos con Capacidad
- MDVRP, Problema de Ruteo de Vehículos con Múltiples Depósitos
- VRP periódico (PVRP)
- SDVRP, Problema de Ruteo de Vehículos de Entrega Dividida
- SVRP, Problema de Ruteo de Vehículos Estocástico
- VRPPD, Problema de Ruteo de Vehículos con Recogidas y Entregas
- VRPB, Problema de Ruteo de Vehículos con Backhauls
- VRPTW, Problema de Ruteo de Vehículos con ventanas de tiempo

Por lo tanto, se tiene que el problema de optimización combinatoria para el CVRP es:

- El conjunto de decisiones: Es el conjunto de rutas con costos mínimos de ruteo encontradas para solucionar el problema.
- La función objetivo: Encontrar la ruta con menor costo.
- Las restricciones sujetas a este problema son:

    1. Cada cliente es atendido exactamente una vez por exactamente un vehículo.
    2. La demanda de un solo cliente no debe exceder la capacidad del vehículo.
    3. La demanda total de cada ruta no debe exceder la capacidad del vehículo.
    4. Cada ruta empieza y termina en el depósito.

Los problemas combinatorios se clasifican en:

2.1.1. Problemas NP

Un problema de decisión se clasifica como NP si, y sólo si, puede resolverse mediante un algoritmo no determinístico que se ejecuta en tiempo polinomial. El nombre NP proviene de “nondeterministic polynomial - bounded” (Polinomialmente acotado no determinístico).

2.1.2. Problemas NP_Hard

Cuando se prueba que un problema de optimización combinatoria en su versión problema de decisión, pertenece a la clase NP completa, entonces la versión optimización es NP- *hard*. La complejidad computacional hace parte de uno de los siete problemas del milenio, que es determinar si todos los problemas no tratables (Los conocidos NP) eventualmente llegarán a ser tratables (Problemas tipo P[2]). Es decir, si la imposibilidad es tecnológica o lógica (P vs NP es un problema abierto), la pregunta a responder es: ¿Los problemas NP (intratables) se pueden reducir a problemas P (tratables)?
Los problemas NP-*hard* no presentan algoritmos polinómicos que corroboren posibles soluciones.

2.2. Instancia

Una instancia en el problema del CVRP, es una representación concreta y específica de su clase, comprende todos los elementos incorporados en el problema y generalmente están ligadas a los vehículos y a los clientes.

---

[2] Los problemas tipo P, son problemas de decisión, que brindan una respuesta por medio de algoritmos deterministicos en un tiempo polinomial.

Las instancias son aquellas características o datos del problema, que se usan en un planteamiento concreto.

Las variables involucradas en el planteamiento de una instancia del problema CVRP son[3]:

- El número de clientes
- La Localización de los clientes
- Los centros de distribución o depósitos
- La capacidad de los vehículos
- La demanda de los clientes
- Los tiempos de recorrido
- Los costos de transporte
- La descripción de las vías de transporte

2.3. Formulación matemática del CVRP

El problema de ruteo de vehículos con capacidad (CVRP) es un problema de optimización combinatoria, de tipo NP- Hard; consiste en hallar una serie de rutas de tal manera que se satisfaga la demanda de una cantidad determinada de *n* clientes distribuidos en una zona geográfica. La suma de las distancias recorridas por los vehículos debe ser la mínima. Cada vehículo parte de un depósito para visitar a los *n* clientes y debe regresar nuevamente a él, se considera una flota homogénea de vehículos, es decir, todos los vehículos tienen la misma capacidad.

[3] http://www.uaem.mx/posgrado/mcruz/cursos/optimizacion/vrpkoko.pdf

El problema de ruteo de vehículos con capacidad consiste en minimizar cierta función objetivo (*f*):

$$1.\quad \min f = \min = \sum_{i \in v} \sum_{j \in v} C_{ij} X_{ij}$$

Donde V = {0,….., n} representa el conjunto de vértices para el grafo G= {V, A}, y A es el conjunto de arcos que une los vértices del grafo.

La variable X_{ij}, es una variable binaria que toma el valor de 1 si en la ruta se incluyen los sitios *i* y *j*, y cero en caso contrario; los parámetros C_{ij} representan el costo de ir desde el sitio i hasta el sitio j.

Además se deben cumplir las siguientes restricciones:

$$2.\quad \sum_{i \in v} X_{ij} = 1 \quad \forall\, j \in V\{0\}$$

$$3.\quad \sum_{j \in v} X_{ij} = 1 \quad \forall\, i \in V\{0\}$$

Las ecuaciones 2 y 3 Imponen que exactamente un arco entra y sale de cada vértice asociado con un cliente, respectivamente. Junto a 2 y 3 se cumple además otra restricción que se tiene respecto al número de vehículos que salen y regresan al depósito:

$$4.\quad \sum_{i \in v} X_{i0} = K$$

$$5.\quad \sum_{j \in v} X_{0j} = K$$

Donde *K* es el número de vehículos que salen y entran del depósito.

Debido a que los vehículos tienen capacidad finita y cada ruta está constituida por un grupo de clientes que el vehículo visita solo una vez, entonces se necesitan el siguiente conjunto de restricciones:

$$6.\quad \sum_{i\notin S}\sum_{j\in S} X_{ij} \geq r(S) \quad \forall S \subseteq V\backslash\{0\}, S\neq 0$$

El conjunto de restricciones (6) se denomina restricciones de corte-capacidad (CCCs) El número r(S) es el número mínimo de vehículos que se requieren para atender el conjunto de clientes (S).

Así, el problema CVRP consiste en:

$$7.\quad \min = \sum_{i\in v}\sum_{j\in v} C_{ij} X_{ij}$$

Sujeto a:

$$8.\quad \sum_{i\in v} X_{ij} = 1 \quad \forall\, j \in V\{0\}$$

$$9.\quad \sum_{j\in v} X_{ij} = 1 \quad \forall\, i \in V\{0\}$$

$$10.\quad \sum_{i\in v} X_{i0} = K$$

$$11.\quad \sum_{j\in v} X_{0j} = K$$

$$12.\quad \sum_{i\notin S}\sum_{j\in S} X_{ij} \geq r(S) \quad \forall S \subseteq V\backslash\{0\}, S\neq 0$$

A través del tiempo se han planteado una gran variedad de métodos de solución para el CVRP.

Los métodos de solución se pueden clasificar en métodos exactos y métodos aproximados. Los métodos exactos son aquellos que brindan una solución óptima del problema mediante un algoritmo eficiente. Son de gran funcionalidad para problemas donde se tienen pocos clientes. Para problemas con una gran cantidad de clientes, pueden gastar largos tiempos computacionales en su solución, lo que hace que en estas ocasiones se prefiera buscar una solución aproximada y no la óptima.

2.4. Método de Branch and Bound para la solución del CVRP

El método de Branch and Bound se desarrolló por primera vez gracias a Lang y Doig. Este método consiste en dividir el problema inicial en sub-problemas para comparar las soluciones factibles y eliminar las menos favorables, de este modo, los tiempos de cómputo para el desarrollo exacto de problemas combinatorios se reducen notoriamente.

La técnica consiste en desarrollar un árbol con las posibles soluciones, de manera que en cada rama se plantea una solución. Cuando se encuentre una rama donde la solución no sea la óptima, se termina la ramificación por esta ruta. Se continúa la evaluación y la ramificación por las otras rutas factibles. El hecho de terminar ramificaciones infactibles representa un ahorro de recursos computacionales.

Para el desarrollo de problemas de ruteo de vehículos con capacidad, se debe iniciar con una solución factible, que tenga asociada una distancia total recorrida. Así se puede realizar el árbol eliminando las ramas que superen la distancia de la solución factible. Esta solución inicial puede ser obtenida con el uso de métodos heurísticos como Clarke and Wright o el Vecino más Cercano. Esta solución inicial será la base para las futuras comparaciones en el algoritmo de Branch and Bound.

El algoritmo de Branch and Bound para el problema del CVRP funciona de la siguiente manera:

Se resuelve el siguiente problema de programación lineal:

Función objetivo (F.O.)

$$13. \quad \min \sum_{i \in v} \sum_{j \in v} C_{ij} X_{ij}$$

Sujeto a:

$$14. \quad \sum_{i \in v} X_{ij} = 1 \quad \forall j \in V\{0\}$$

$$15. \quad \sum_{j \in v} X_{ij} = 1 \quad \forall i \in V\{0\}$$

$$16. \quad \sum_{i \in v} X_{i0} = K$$

$$17. \quad \sum_{j \in v} X_{0j} = K$$

$$18. \quad \sum_{i \notin S} \sum_{j \in S} X_{ij} \geq r(S) \quad \forall S \subseteq V \backslash \{0\}, S \neq 0$$

$$X_{ij} \geq 0$$

Siendo X_{ij} una variable real no negativa.

Si la solución de programación lineal satisface las condiciones del problema original, entonces esta es la solución del problema CVRP. En caso contrario se procede a realizar ramificaciones y acotamientos.

Acotamiento (Bounding): Inicialmente en el método de Branch and Bound se resuelve el problema continuo obtenido por relajaciones de las restricciones que impone el carácter entero de las variables. Estas relajaciones se llaman relajaciones continuas.

Existen varios tipos de relajaciones:

1. Agregar restricciones al problema.
2. Usar multiplicadores de LaGrange.
3. Usar relajaciones LP para los problemas de programación entera mixta (MIP). Se debe determinar que las variables sean reales no negativas.

Ramificación (Branching): Se hace ramificación cuando los valores de la solución óptima del problema no son enteros. Se define una variable entera $[x_i]$ (parte entera de x_i) tal que:

$$19. \quad [x_i] < x_i < [x_i] + 1$$

Se formulan entonces dos subproblemas cada uno de ellos con una restricción adicional al problema inicial. Para el primer subproblema se añade la restricción:

$$20. \quad x_i \leq [x_i]$$

Para el segundo subproblema se añade la restricción:

$$21. \quad x_i \geq [x_i] + 1$$

Este proceso de dividir el problema en subproblemas, se denomina ramificación. Este proceso consiste en dividir la región factible en 2 o más regiones factibles más pequeñas. Cada una de estas nuevas regiones factibles representa cada uno de los subproblemas y se resuelven como un problema continuo.

Para la ramificación de este método se deben tener en cuenta dos tipos de restricciones que lo conforman, unas de tipo excluyente y otras de tipo incluyentes

en cada solución. Aquellos problemas que se caracterizan por tener estos dos tipos de restricciones son conocidos como los problemas de asignación modificados (MAP).

Supóngase un grafo G = (V, A), para un conjunto de *k* nodos, los dos subconjuntos asociados a este grafo serían:

$E_k \subset A$ que contiene los arcos excluidos.

$I_k \subset A$ que contiene los arcos incluidos.

Se asume que la solución en el nodo *k*, para el MAP, presenta subrutas del árbol de decisión, y que hay una subruta con el menor número de arcos *m*. A partir del conjunto de vértices de la subruta con el menor número de arcos $V^1 = [r_1, r_2,\ldots, r_m]$ se construye un grafo $G^1 = (V^1, A^1)$ con un conjunto de arcos $A = [(r_1, r_2), (r_2, r_3),\ldots, (r_m, r_1)]$. Los nodos descendientes de *k* son *m*. Junto con el nodo *j* (*j* = 1, 2,…, *m*) se asocian a los subconjuntos:

$$22. \quad E_J = E_k \cup \{ (r_j, r_{j+1}) \}, j = 1, 2, ..., m$$

$$23. \quad I_J = I_k \cup \{ (r_u, r_{u+1}) : \quad u = 1, .., j-1 \}$$

Donde $r_{m+1} = r_1$. Los nuevos arcos incluidos $I_1, I_2, ..., I_m$ se encargan de generar la división del espacio solución con el nodo *k*, para las *m* subrutas de los *m* descendientes. Los nuevos arcos excluidos (r_j, r_{j+1}) en el conjunto E_J se encargan de eliminar las subrutas que se forman para todos los MAP en el grafo G^1 asociados al nodo *j*.

2.5. Método Branch and Cut para la solución del CVRP

Los métodos exactos dan la solución óptima a problemas de ruteo de vehículos con capacidad. Es recomendable el uso de estos métodos, para la solución del CVRP en instancias pequeñas, es decir menor a 50 clientes.

El método de Branch and Cut, hace parte de los métodos exactos. Desciende de las técnicas de Branch and Bound. Este método es usado para solucionar problemas combinatorios por medio del Simplex, debido a que se usa para resolver problemas de programación lineal. Branch and Cut usa algoritmos de planos de corte para confirmar que la solución al problema es de tipo entero, o muy aproximada a este si así lo necesitara.

Plano Cortante: Los planos cortantes son aquellos que determinan curvas o rectas en las superficies, para obtener soluciones enteras a un problema lineal. Inicialmente se resuelve un problema lineal no entero. En seguida, se examina que la solución encontrada sea entera, en caso de no ser así, se debe agregar una nueva restricción que corte la solución no entera encontrada, sin cortar ningún punto de la solución factible. Este procedimiento debe repetirse hasta encontrar la solución óptima entera.

El método de Branch and Cut ha sido muy exitoso en la solución de problemas combinatorios, sin embargo puede dar malos resultados si se presentan algunas de estas situaciones:

1. No se tiene un buen algoritmo para realizar la fase de plano de corte.
2. El número de iteraciones de la fase de plano de corte es demasiado alta.
3. El problema de programación lineal no tiene solución debido a su tamaño.
4. El árbol generado por el proceso de ramificación se vuelve muy grande y no tiene fin parece poco probable en un plazo de tiempo razonable.

Si se presentan las situaciones 1 y 2, no hay solución para ello. Si se presenta la situación 3, se puede pensar en hacer un esfuerzo extra y suprimir algunas restricciones inactivas.

La situación 4 es el problema central del método de Branch and Cut. La mayoría de los fracasos se deben a éste. El único arreglo posible para solucionar el inconveniente tipo 4 es fortalecer la relajación lineal, esto quiere decir que se deben agregar algunas desigualdades lineales que estén satisfechas por todas las soluciones. Encontrar estas desigualdades no es una tarea fácil, y hace parte del estudio del problema.

2.6. Algunos Métodos Metaheurísticos para el problema del CVRP

Debido a que los métodos heurísticos son ideas lógicas para hallar una buena solución de un problema, se ha llegado a construir numerosas heurísticas para cada realidad modelada. En los últimos años, se han propuesto métodos de solución general que dan una estructura como criterio estratégico para la elaboración de los métodos heurísticos específicos, dependiendo del problema y las variantes que requiera. A estas técnicas se les llama metaheurísticas.

2.6.1. Algoritmos Genéticos

Los Algoritmos Genéticos (GAs, por sus siglas en inglés Genetic Algorithms) fueron inventados por John Holland, a principios de la década de los 70s. Estos algoritmos se basan en las ideas de evolución según la selección natural y genética. La idea principal de estos algoritmos es la de usar el poder de la evolución, para darle solución a los problemas de optimización.

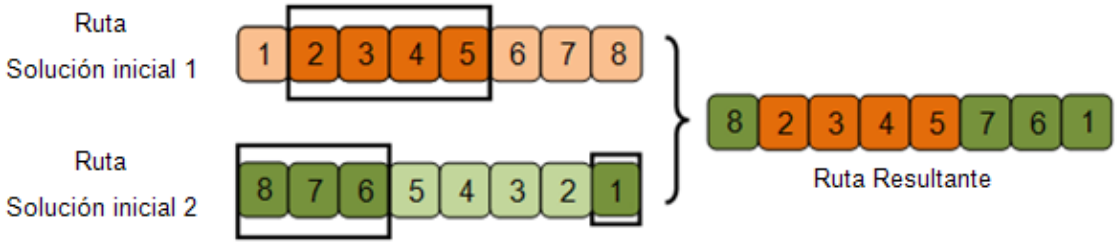
Para la solución del CVRP, los Algoritmos Genéticos realizan su búsqueda en el espacio de soluciones donde se encuentran todos y cada uno de los nodos,

basados en la información histórica, dirigen la búsqueda a aquellas áreas de mejor rendimiento. Es decir, se dirigen a las soluciones que ya saben con anterioridad que funcionan, para que el algoritmo pueda mejorarlas.

Luego de haber seleccionado el área o la población candidata de búsqueda, el algoritmo evalúa todos los nodos que la conforman, determina cuales nodos de la población son factibles y cuáles no, basándose en el aporte que cada uno de ellos puede brindar a la solución de la función objetivo. Los nodos que no sean prometedores, es decir que no contribuyan a la solución del problema, se eliminan. Por el contrario, aquellos nodos que se crea que son contribuyentes a la solución se conservan y “se reproducen”, para que al igual que ocurre con la genética, se perfeccionen sus genes de acuerdo a las necesidades del entorno. Este algoritmo consta de tres fases: Selección, Reproducción (o Cruce) y Mutación.

La reproducción consiste en un cruce entre individuos, es decir, se cruzan dos soluciones factibles, para que generen una nueva solución que hereda las características de sus “progenitores” (las dos rutas factibles iníciales). Esta nueva población “creada” va a ser aún mejor que la ya existente, por esto, la nueva población con mejores características, reemplaza a la inicial.

Figura 1. Cruce



Fuente: Autores

A medida que el proceso de reproducción avanza, alternamente se sigue haciendo el proceso de selección, para eliminar los nodos no factibles, y para escoger

aleatoriamente, aquellos nodos a los que se les deben hacer cambios (mutaciones) para mejorar sus características.

La etapa de mutación es importante dado que en algunas ocasiones durante la etapa 1 y 2 (Selección y reproducción) se pierde la potencia de la información genética afectando la solución.

Figura 2: Mutación

Fuente: Autores

Todo este proceso debe dirigirse al encuentro de la solución óptima, es decir, cada nueva población creada, debe llevar a una solución final, catalogada como la mejor de todas las posibles soluciones.

Entonces los pasos a seguir para el desarrollo de GAs para el problema del CVRP serian:

1. Escoger soluciones iníciales para el problema, aleatoriamente o por medio de una heurística.
2. Verificar que las soluciones iníciales no sean iguales.
3. Seleccionar las rutas iníciales que presenten menores costos de recorrido.
4. Escoger el tipo de operador de cruce para desarrollar el problema.
5. Escoger el operador de mutación.
6. Actualizar la solución inicial con la descendencia obtenida.

7. Terminar el algoritmo según criterio de parada[4].

2.6.2. La Búsqueda Tabú

La Búsqueda Tabú es un algoritmo metaheurístico que sirve para resolver problemas de optimización combinatoria, tales como el problema del agente viajero (TSP, del inglés *Travelling Salesman Problem*).

Este algoritmo también conocido por las siglas TS (Tabu Search - en ingles) usa un procedimiento por vecindades o de búsqueda local para moverse iterativamente desde una solución *x* hacia una solución *x*' en la vecindad de *x*, hasta satisfacer algún criterio de parada.

Para explorar las regiones del espacio de búsqueda, que serían dejadas de lado por el procedimiento de búsqueda local, la búsqueda tabú cambia la estructura de vecinos para cada solución a medida que la búsqueda avanza. Las soluciones admitidas para $N^*(x)$, el nuevo vecindario, son dadas por el uso de estructuras de memoria. La búsqueda entonces progresa moviéndose iterativamente de una solución *x* hacia una solución *x*' en $N^*(x)$.

Cuando en la lista Tabú hay una solución mejor que la actual, se elimina esta clasificación de esta lista, debido al criterio de aspiración.

La estructura de memoria más importante para determinar las soluciones permitidas a un $N^*(x)$, es la lista tabú. Una lista tabú es una memoria de corto plazo que tiene las soluciones que fueron visitadas en el pasado reciente (menos de *n* iteraciones atrás, donde *n* es el número de soluciones previas que van a ser

[4] Los criterios de parada de los GAs se establecen previamente al inicio del problema y pueden ser: Determinar números fijos de generaciones sin mejora, estipular tiempos límites o fijar un número determinado de iteraciones.

almacenadas (*n* también es llamado el tenor del tabú)). La búsqueda tabú excluye las soluciones en la lista tabú de $N^*(x)$.

Para dar solución al CVRP por medio de la búsqueda tabú, es necesario generar una solución inicial, mediante alguna heurística, y a partir de esta se crea la lista tabú. Para empezar a llenar la lista, primero se escoge un nodo inicial y se examina su vecindad, se elige la mejor unión dados los criterios de menor distancia entre los puntos. Esta unión se incluye en la lista tabú, pero de forma contraria, de modo que si se creó la ruta A→B, se incluye en la lista tabú la unión B→A para no ser tomada esta ruta. De no ser así, puede crear ciclos y no se cumpliría con las restricciones a las que está sujeto el CVRP. Este método iterativo termina, cuando se llegue al criterio de parada, que puede ser un cierto número de corridas definidas al inicio del problema.

3. MÉTODOS HEURÍSTICOS PARA EL CVRP

Los métodos heurísticos son aquellos que dan una solución factible al problema, sin asegurar que esa solución obtenida sea la óptima.

El término heurística proviene del griego “heuriskein”, que significa “encontrar” o “descubrir”. Un procedimiento se califica heurístico cuando se tiene un alto grado de confianza en que las soluciones “encontradas” son de alta calidad, es decir, con un tiempo de solución computacional razonable, aunque no garanticen una solución óptima. En los métodos heurísticos, la rapidez del proceso es tan importante como la calidad de la solución obtenida. Se usa el calificativo heurístico en contraposición a exacto[5].

Los métodos heurísticos son procedimientos que se usan para resolver los problemas de optimización mediante una aproximación intuitiva, en la que la estructura del problema se utiliza para obtener una buena solución.[6]

Los procedimientos heurísticos son usados para resolver problemas de optimización de decisión cuando[7]:

- No se conoce ningún método exacto que le brinde solución.
- Se conoce un método exacto que le brinde solución pero el tiempo computacional que requiere es demasiado largo.
- El método heurístico es más flexible que el método exacto, es decir, permite incorporar condiciones que son difíciles de modelar.

---

[5] Melián, Belén. Perez, Jose A. et al.”Metaheurísticas: una visión global”. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*. N.19 pp. 7-28 ISSN: 1137-3601. © AEPIA(2003). http://www.aepia.org/revista.

[6] Díaz, A., Glover, F., Ghaziri, H.M., et al, *Optimización Heurística y Redes Neuronales.* Madrid, Paraninfo, (1996).

[7] MARTÍ, RAFAEL. Procedimientos Metaheurísticos en Optimización Combinatoria. http://www.uv.es/~rmarti/

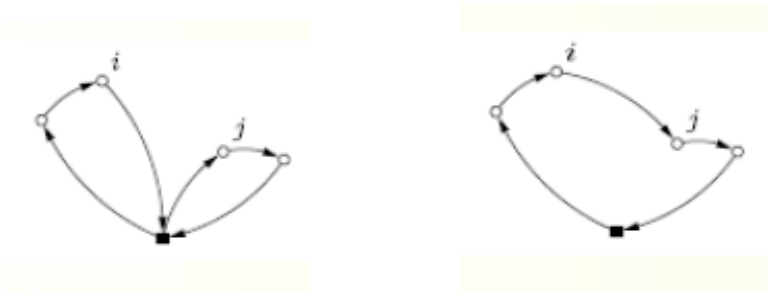
A continuación se introducen algunos de los métodos heurísticos más usados para la solución del CVRP.

3.1. Algoritmo de Ahorros de Clarke and Wright

El algoritmo de ahorros es uno de los más difundidos para la solución de problemas de ruteo de vehículos con capacidad. El algoritmo parte de la siguiente observación simple, existen dos rutas diferentes *(0,…i, 0)* y *(0, j,…,0)* (como se muestra en la Figura 3) estas se podrían combinar y formarse una nueva ruta *(0,…, i, j,…,0)* con un ahorro en los costos. El ahorro que se tiene para la nueva ruta es igual a:

$$s_{ij} = C_{i0} + C_{oj} - C_{ij} \quad 24$$

Figura 3. Algoritmo de Ahorros 2 rutas antes y después de ser unidas



Fuente: Fernando Sandoya Sánchez, Escuela superior Politécnica Del litoral. “Métodos Exactos y Heurísticos para resolver el problema del agente viajero y el problema de ruteo de vehículos” Guayaquil - Ecuador oct. 2007

La nueva solución de arcos incluye el arco *(i, j)* debido a que el arco *(i, 0)* y *(0, j)* son reemplazados por el primero. La idea es realizar uniones que den mayores ahorros pero que no violen las restricciones.

El algoritmo de ahorros se ha popularizado de manera rápida y creciente debido a su simplicidad para ser implementado. El algoritmo de ahorros de Clarke and Wright fue desarrollado para resolver el problema del agente viajero (TSP), sin embargo, mediante ciertas modificaciones, también sirve para hallar la solución del problema de ruteo de vehículos con capacidad (CVRP). Durante el transcurso del tiempo se han realizado mejoras para incrementar la efectividad para determinar la solución del CVRP.

En forma breve se describe el algoritmo. Supóngase que se tienen dos rutas diferentes, r_i = (0, *i,* 0) y r_j = (0, *j,* 0), donde 0 representa el depósito (D). El nodo *i* es el último nodo del recorrido de la ruta r_i, mientras que el nodo *j* es el inicio de la ruta r_j. El costo de transporte para las rutas r_i y r_j se denota como $C(0,i,0)\,\text{y}\,C(0,j,0)$ respectivamente. Al fusionar las rutas r_i y r_j en una sola, entonces, el ahorro que se obtiene por la unión será igual a:

$$25. \quad s_{ij} = C_{i0} + C_{oj} - C_{ij}$$

A partir de la observación anterior se construye el algoritmo de ahorros, para hallar la solución del CVRP simétrico.

Supóngase que se tiene un nodo depósito, identificado por *0* y *n* nodos que representan a los clientes. Se asume que son conocidos los costos de transporte de ir de un nodo *i* a un nodo *j* y sus respectivas demandas.

El algoritmo supone que hay un vehículo para cada uno de los clientes. Por lo tanto, hay *n* rutas separadas desde el depósito a cada uno de los clientes. El costo total asociado a ésta solución es:

$$26. \quad CT = 2\sum_{i=1}^{n} c_{0i}$$

A continuación se calculan todos los posibles ahorros asociados a las fusiones entre dos rutas. El número total de ahorros es igual a:

$$27. \quad \binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2}$$

Para las dos rutas (0, *i, 0*) y (0, *j, 0*) el ahorro por la unión de las rutas es igual a:

$$28. \quad s_{ij} = \left(C_{0i} + C_{i0} + C_{Oj} + C_{j0}\right) - \left(C_{oi} + C_{ij} + C_{j0}\right)$$

$$s_{ij} = C_{i0} + C_{Oj} - C_{ij}$$

Donde: c_{ij} es el costo de ir del nodo i al nodo j.

Suponga ahora que r*1*, sea la ruta (0, 3, 1, 0) y r*2* sea la ruta (0, 1, 2, 0), donde r*1* representa mayor ahorro que r*2*, es decir, de una lista de ahorros ordenada de mayor a menor ahorro, r*1* se encuentra primero que r*2*. Se debe determinar si son factibles o no, estas uniones.

Para que una unión entre dos rutas sea factible, se deben cumplir las siguientes restricciones:

- Las dos rutas que visitan a *i* y a *j* se fusionan, siempre y cuando el nodo *j* se recorre inmediatamente después del nodo *i* en la nueva ruta obtenida por la fusión.
- La fusión se realiza siempre y cuando no se borre una conexión directa establecida previamente entre los dos clientes.
- La suma de las demandas de los clientes de la ruta, no debe exceder la capacidad del vehículo.

Ahora, para r_1 los nodos *i-j* en consideración son 3 y 1 respectivamente; la fusión de estas dos rutas es factible, ya que el nodo 1 se recorre inmediatamente después del nodo 3, en la nueva ruta obtenida por la fusión, no están incluidos en otras conexiones previas, y se asume que cumple con las demandas.
Para r_2 los nodos *i-j* en consideración, en cambio, son 1 y 2 respectivamente. Esta fusión no es factible, debido a que el nodo 1 ya tiene una previa conexión directa establecida. Así todos los ahorros obtenidos se examinan para ver si las uniones de las rutas obtenidas son factibles o no. Este algoritmo termina una vez se hayan examinado todos los ahorros.

El algoritmo de ahorro tiene una versión paralela y una secuencial, las cuales se muestran en la siguiente figura:

Figura 4. Versión paralela y secuencial del Algoritmo de Ahorros

Algoritmo de Ahorros (Versión Paralela)

Paso 1. (Inicialización). Para cada cliente *i* construir la ruta (*0, i, 0*).

Paso 2. (Calculo de ahorros). Calcular S_{ij} para cada par de clientes *i* y *j*.

Paso 3. (Mejor unión). Sea $S_{i^*j^*}$ = máx. S_{ij}, donde el máximo se toma entre los ahorros que no han sido considerados aun. Sean r_{i^*} y r_{j^*} las rutas que contienen a los clientes *i** y *j** respectivamente. Si *i** es el ultimo cliente de r_{i^*} y *j** es el primer cliente de r_{j^*} y la combinación de r_{i^*} y r_{j^*} es factible, combinarlas. Eliminar $S_{i^*j^*}$ de futuras consideraciones. Si quedan ahorros por examinar ir a 3, sino terminar.

Algoritmo de Ahorros (Versión Secuencial)

Paso 1. (Inicialización). Para cada cliente *i* construir la ruta (*0, i, 0*).

Paso 2. (Calculo de ahorros). Calcular S_{ij} para cada par de clientes *i* y *j*.

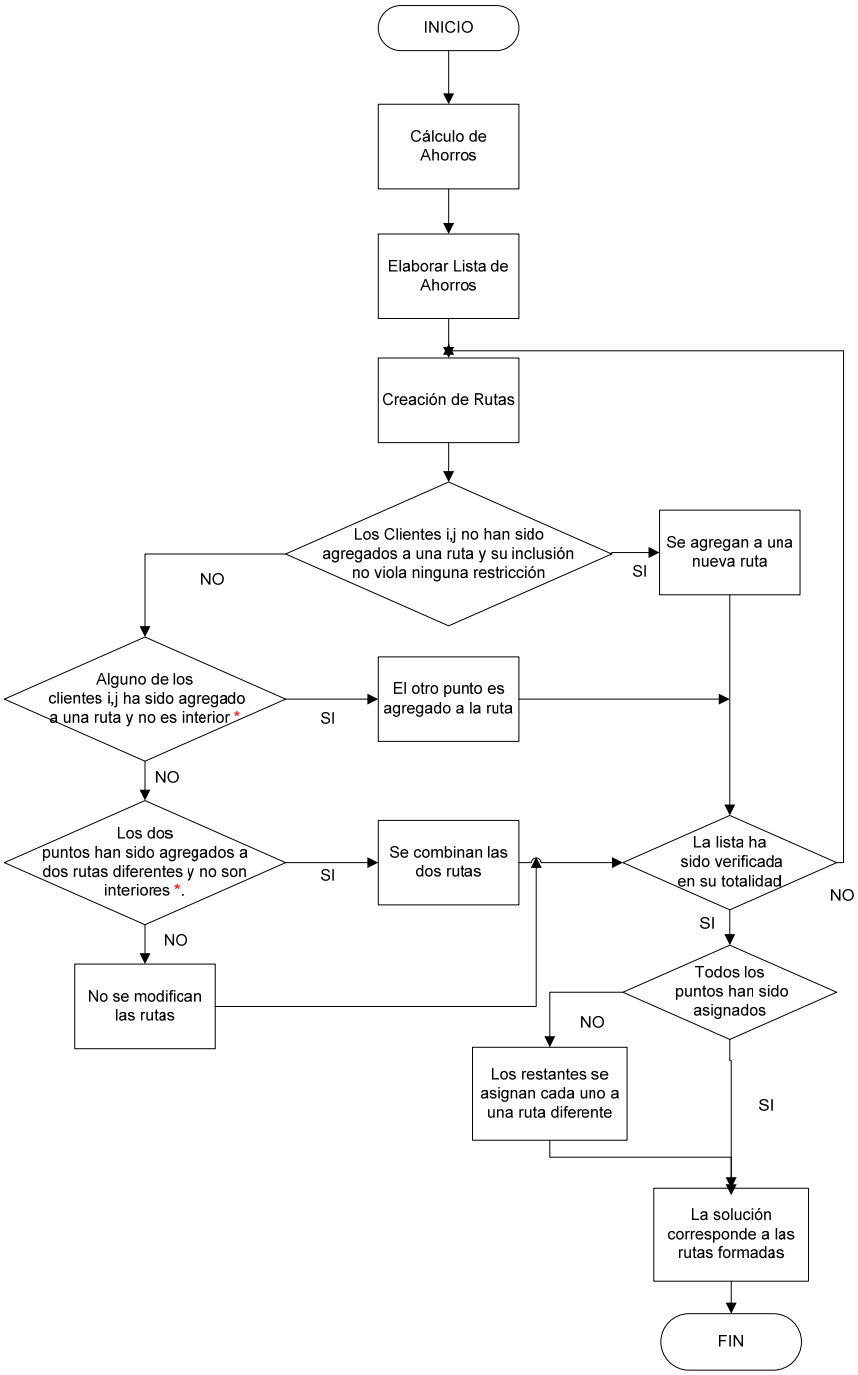
Paso 3. (Selección). Si todas las rutas fueron consideradas, terminar. Si no, seleccionar una ruta que aun no haya sido considerada.

Paso 4. (Extensión). Sea (*0, i,…, j, 0*) la ruta actual. Si no existe ningún ahorro conteniendo a *i* o a *j*, ir a 3. Sea S_{k*i} (o S_{jl*}) el máximo ahorro conteniendo a i (o a *j*). Si *k** (o *l**) es el ultimo (o primer) cliente de su ruta y la combinación de dicha ruta con la actual es factible, realizar dicha combinación. Eliminar S_{k*i} (S_{jl*}) de futuras consideraciones. Ir a 4.

Fuente: Tomada del artículo Heurísticas para problemas de ruteo de Alfredo Oliveira, Instituto de computación, facultad de ingeniería, Universidad de la republica, Montevideo, Uruguay

En la figura 5 se muestra el diagrama de flujo del funcionamiento del algoritmo de ahorros de Clarke and Wright.

Figura 5. Diagrama de flujo del algoritmo CLARKE AND WRIGHT

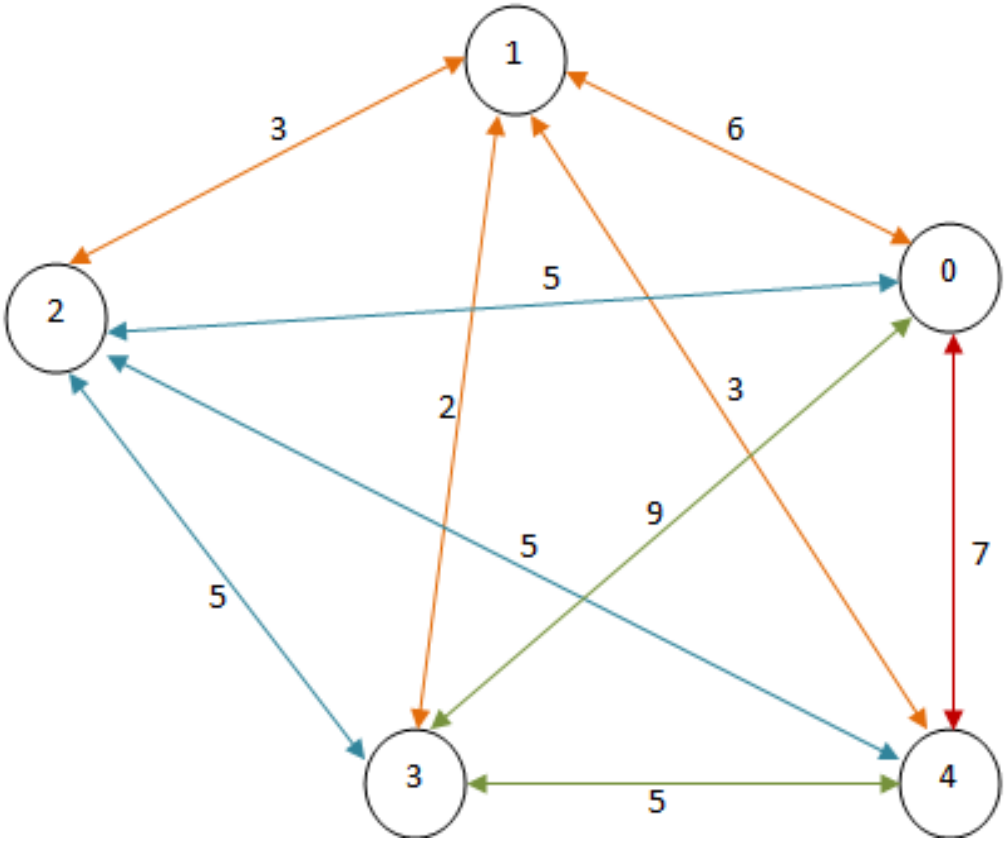


Fuente: Autores

A continuación desarrollaremos un ejemplo para el CVRP simétrico; con 4 clientes, un depósito y dos vehículos. Con el propósito de ilustrar el funcionamiento del algoritmo de ahorros.

Ejemplo 1. En la figura 6, se muestra el grafo de transporte, y en la tabla 1, Se muestran los costos de transporte.

Figura 6. Rutas con costos asociados para el ejemplo 1.



Fuente: Autores

La matriz de costo asociada a los clientes es de:

Tabla 1. Matriz de Costos C_{ij} del ejemplo 1.

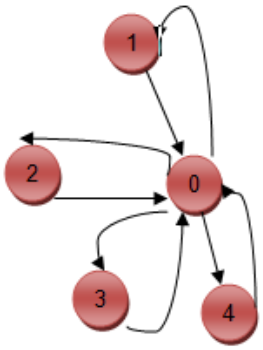
	0	1	2	3	4
0	0	6	5	9	7
1	6	0	3	2	3
2	5	3	0	5	5
3	9	2	5	0	5
4	7	3	5	5	0

Fuente: Autores

Las filas están etiquetadas desde *i=0, 1, 2, 3,4* y *j=0, 1, 2, 3,4,* 0 representa el depósito. El vector demanda es: D = [24, 25, 31, 28]. La capacidad de cada vehículo es de 55 unidades.

1. El primer paso en el algoritmo es calcular todos los ahorros s_{ij} para todas las parejas de clientes (i, j) involucradas en el ejercicio, con $i \neq j$, donde i = 1, 2, 3, 4 y j = 1, 2, 3, 4.

Figura 7. Rutas iníciales para el ejemplo 1.



Fuente: Autores

En este caso los ahorros por calcular son: S_{12}, S_{13}, S_{14}, S_{23}, S_{24}, S_{34}, por ser un problema simétrico. Como se mostró anteriormente en la figura 7 todas las rutas parten del depósito y regresan a él.

Tabla 2. Calculo de Ahorros

Pares de Rutas	Nueva Ruta	Ahorro Obtenido (Calculado)
(0,1,0) y (0, 2, 0)	(0, 1, 2, 0)	$S_{12} = 8$
(0,1,0) y (0, 3, 0)	(0, 1, 3, 0)	$S_{13}=13$
(0, 1, 0) y (0, 4, 0)	(0, 1, 4, 0)	$S_{14}=10$
(0, 2, 0) y (0, 3, 0)	(0, 2, 3, 0)	$S_{23}=9$
(0, 2, 0) y (0, 4, 0)	(0, 2, 4, 0)	$S_{24}=7$
(0, 3, 0) y (0, 4, 0)	(0, 3, 4, 0)	$S_{34}=11$

Fuente: Autores

2. Los ahorros obtenidos se ordenan de mayor a menor en una lista, que denominaremos lista de los ahorros S_{ij}. La lista para el ejemplo que se está desarrollando aparece a continuación.

Tabla 3. Lista de Ahorros Ejemplo 1.

Ruta	Ahorro
S_{13}	13
S_{14}	10
S_{34}	11
S_{23}	9
S_{12}	8
S_{24}	7

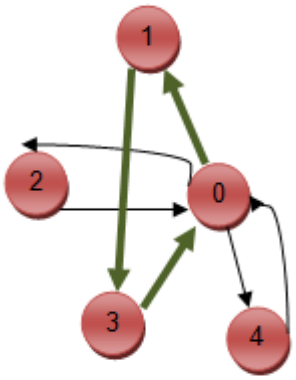
Fuente: Autores

3. De la lista de ahorros cada par de puntos se considera paso a paso. Para el par de puntos *i-j* que está en consideración, las dos rutas que visitan a *i* y a *j* se fusionan, siempre y cuando cumplan las condiciones enunciadas anteriormente, en la explicación del funcionamiento del algoritmo.

Para nuestro ejemplo se escoge el ahorro S_{13}, y se revisa si está ruta es factible y si la demanda de los clientes es menor a la capacidad del vehículo. En caso que la capacidad del vehículo se exceda, la unión no se realiza y se selecciona el siguiente ahorro. Cuando el primer vehículo complete su capacidad, se inicia otro recorrido con el segundo vehículo.

Por ejemplo, en la figura 8 aparece en verde la ruta que resulta de fusionar las rutas (0, 1,0) y (0, 3,0), cumplen con la capacidad y tienen el mejor ahorro, por lo que la hace una solución factible. Si la ruta resultante es factible se agrega a la solución, de lo contrario se rechaza la unión y no puede ser agregada a la solución. Para este caso la ruta (0, 1, 3, 0) es factible. En la figura 8 se muestra esta solución.

Figura 8. Solución inicial del ejemplo 1



Fuente: Autores

La cantidad de unidades que recoge el primer vehículo, es la suma de las demandas del cliente 1 y el cliente 3. Dado que la capacidad del vehículo, es de 55 unidades, se satisface los requerimientos para esta ruta. Se procede a hacer uso del segundo vehículo para continuar con el paso 4.

4. Eliminar de futuras consideraciones el ahorro escogido en el paso 3.

5. Escoger el siguiente mejor ahorro de la lista del paso 2, examinar si los clientes contenidos en la nueva ruta no hacen parte de conexiones directas previas, si es factible o no su unión y revisar si la suma de sus demandas cumplen con la capacidad del vehículo. Si es factible su unión, incluir la nueva ruta en la solución, en caso contrario continuar con el siguiente ahorro de la lista y volver al paso 3. Este paso se repite hasta que se hayan considerado todos los ahorros comprendidos en la lista de ahorros del paso 2. A continuación se presenta una lista con los tramos no factibles del ejemplo 1.

Tabla 4. Uniones Factibles del ejemplo 1.

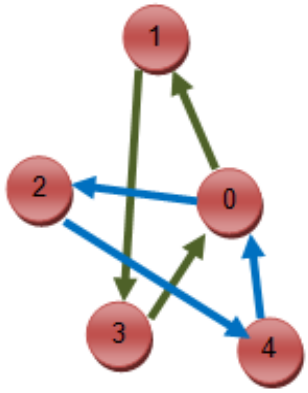
Ruta	¿Es Factible?	Explicación
S_{13}	Factible	La ruta (0, 1, 3, 0) hace parte de la solución ya que los clientes 1 y 3 no han hecho conexiones previas con otros clientes. Además la suma de las demandas de los clientes cumple con la capacidad del vehículo.
S_{14}	No Factible	Debido a que el cliente 1, ya tiene una previa conexión directa con el cliente 3, en la unión de la ruta inicial (0, 1, 3, 0) por lo que no se puede incluir al cliente 1 en otra ruta.
S_{34}	No Factible	La ruta (0, 3, 4, 0) no se puede unir porque el cliente 3 ya tiene una conexión previa en la ruta (0, 1, 3, 0) que no se puede borrar y además, la suma de sus demandas, excede la capacidad del vehículo
S_{23}	No Factible	La ruta (0, 2, 3, 0) no se puede unir debido a que el cliente 3, ya tiene una conexión directa previa. Además, la suma de sus demandas excede la capacidad del vehículo.
S_{12}	No Factible	La ruta (0, 1, 2, 0) no es factible porque el cliente 1 ya tiene una conexión directa previa con el cliente 3. Aunque la suma de sus demandas si cumpla con la capacidad del vehículo.

S_{24}	Factible	La ruta (0, 2, 4, 0) cumple con la capacidad del vehículo y puede hacer parte de la solución ya que ninguno de los dos clientes involucrados en la ruta ha hecho previamente conexiones directas

Fuente: Autores

La capacidad usada para el segundo vehículo en este caso es la suma de las demandas de los clientes 2 y 4. La capacidad del vehículo es de 55 unidades que satisface los requerimientos de esta ruta. A continuación se muestra en la figura 9 la solución final del ejemplo.

Figura 9. Solución final del ejemplo 1



Fuente: Autores

6. Unir todas las rutas que fueron factibles para obtener la solución final mostrada en la figura 9, donde las flechas de color verde representan la ruta que realiza el primer vehículo, y las flechas azules representan la ruta del segundo vehículo.

3.2. Mejora del Algoritmo de Ahorros de Clarke and Wright

Se han escrito muchos documentos acerca de cómo mejorar el algoritmo de ahorros de Clark and Wright, debido a que fue uno de los métodos iniciales propuestos para dar solución al CVRP. El algoritmo de ahorros es considerado

una de las heurísticas más sencillas y rápidas para solucionar el problema del CVRP. Es por esto que en su versión paralela (la cual se explica en el capítulo siguiente) se combinan distancias y demandas de clientes para encontrar una solución aun más rápida y efectiva que la que brinda la versión secuencial.

Se parte de la relación que a menor distancia entre clientes, mayor ahorro en costos, lo que se verá reflejado en la solución del ejercicio. Cuando los clientes se encuentran ubicados geográficamente en una zona, el algoritmo de ahorros tiende a construir rutas circulares. Para solucionar este inconveniente se ha propuesto introducir un parámetro λ llamado ***forma de la ruta*** (λ), el cual busca evitar la formación de rutas circulares mientras se va desarrollando el ejercicio, buscando así resultados más efectivos y con un incremento en los ahorros. La nueva fórmula para el cálculo del ahorro en costos de la nueva ruta quedaría:

$$29. \quad S_{ij} = C_{i0} + C_{0j} - \lambda C_{ij}$$

El parámetro forma de la ruta (λ) puede tomar valores positivos entre 0 y 1, lo que aumenta el ahorro conseguido por la versión secuencial del problema y permite mejores soluciones a lo largo de todo su desarrollo.

3.3. Algoritmo del Vecino más Cercano

El algoritmo del vecino más cercano fue introducido por primera vez por J. G. Skellam, en el cual se usan la mitad de las distancias observadas para determinar si los datos están agrupados. El mayor trabajo hecho para este algoritmo, fue hecho por P. J. Clark y F. C. Evans en 1954.

Este algoritmo determina una solución basada en la cercanía de ubicación, para unir un conjunto de clientes distribuidos en el espacio. El algoritmo consiste en ir

construyendo las rutas de forma secuencial, eligiendo como nodo siguiente, al nodo más cercano del nodo actual, iniciando desde el depósito. La inspección de la cercanía de los nodos, se hace de manera iterativa, y en cada paso, se examina la vecindad del nodo actual para la elección del nodo a insertar en la ruta.

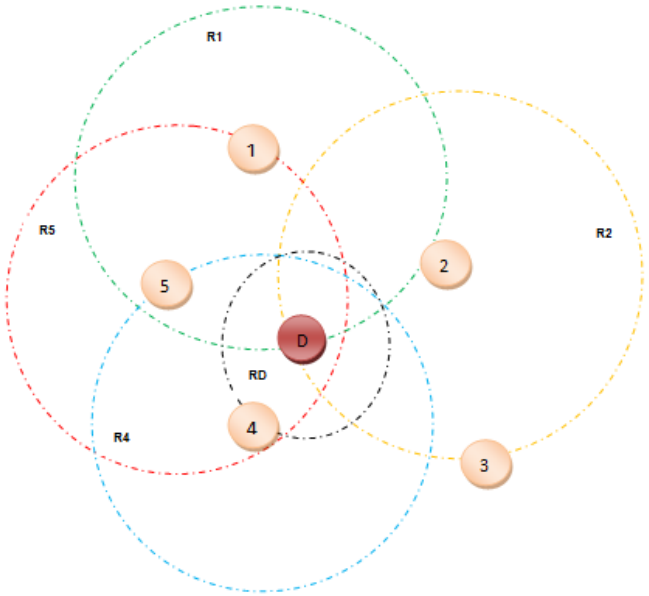
Se parte del nodo (cliente) más próximo al depósito. Una vez ubicado en el nodo más cercano al depósito, se inspecciona la vecindad de este nodo para encontrar el más cercano a él, y verificar las restricciones asociadas al CVRP. Si se satisfacen estas restricciones de capacidad, ahora la vecindad a inspeccionar será la vecindad del nodo actual o ultimo nodo insertado en la ruta, y se repite este procedimiento hasta insertar todos los nodos del problema. Cuando se inserten todos los nodos se termina la ruta regresando desde el último nodo insertado hasta el depósito o punto de partida.

A continuación se describe el algoritmo del Vecino más Cercano para el problema del CVRP:

1. Ubicarse en un nodo inicial (Depósito).
2. Tomar la distancia desde el nodo agregado hacia todos y cada uno de los nodos restantes.
3. Escoger el nodo que tenga menor distancia en relación con el último nodo agregado, siempre y cuando no haya sido considerado.
4. Repetir el paso 2 hasta incluir todos los nodos. De esta forma se construye la trayectoria.
5. El algoritmo termina cuando se han unido todos los nodos según su cercanía.
6. Después de haber construido toda la trayectoria, unir el último nodo encontrado con el depósito, para indicar que la ruta regresa al depósito de donde partió inicialmente.

El algoritmo del vecino más cercano hace parte de los algoritmos heurísticos, y es uno de los métodos más sencillos para brindar solución al problema de ruteo de vehículos con capacidad (CVRP).

Figura 10. Selección de una ruta por el Algoritmo del Vecino más Cercano

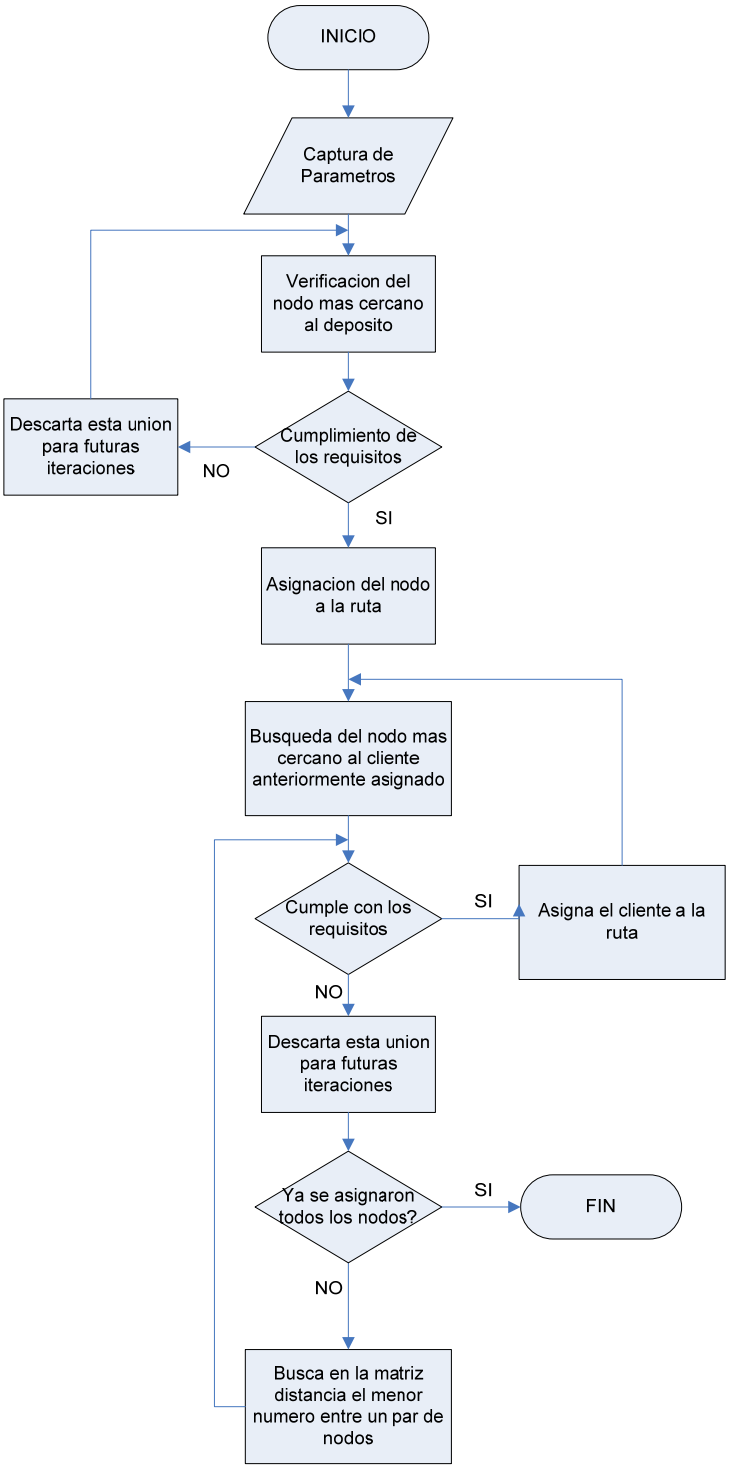


Fuente: Autores

El objetivo de este algoritmo, es ir encontrando clientes cercanos entre si, a medida que se avanza. Situado desde un nodo inicial, encontrar el nodo más cercano a éste, y después de ubicarlo, dirigirse hacia él para poder hacer el mismo procedimiento desde ese nuevo nodo. El problema se soluciona después de haber repetido este procedimiento las veces necesarias para visitar todos los nodos y poder regresar al depósito.

A continuación, se muestra en la figura 11, el diagrama de flujo del funcionamiento del algoritmo del Vecino más Cercano.

Figura 11. Diagrama de flujo del Vecino más Cercano



Fuente: Autores

3.4. Heurísticas de inserción para el CVRP

Las Heurísticas de inserción son métodos con los cuales se pueden hallar las soluciones parciales de las rutas. En cada iteración, hay una solución parcial donde las rutas visitan solo a un subconjunto de los clientes. Se elige a uno de los clientes que aun no haya sido visitado y se inserta en la solución.

A continuación se discuten algunas estrategias para la elección de los clientes a insertar.

3.4.1. Algoritmo 3-opt

El algoritmo 3-opt, es un método de muestreo utilizado para generar poblaciones de soluciones localmente óptimas. Este algoritmo, utiliza una heurística de búsqueda local, para hallar m óptimos locales y así crear la población P. Para llevar a cabo la creación de la población, este algoritmo guarda en una variable denominada r-opt, una ruta hallada aleatoriamente como se describe a continuación:

1. La ruta arranca en el depósito.
2. Se elige aleatoriamente un cliente de V, donde V es el conjunto de clientes que aun no han sido visitados. Este proceso se repite hasta que se hayan asignado todos los clientes a la ruta.
3. Cuando V = □ esto quiere decir que se hayan asignado todos los clientes a la ruta, se debe regresar al depósito.

Luego de haber credo la ruta inicial se calcula la población de rutas con el algoritmo 3-opt. El método 3-opt funciona de la siguiente manera:

1. Se define un número de iteraciones a realizar. Este será el criterio de parada.
2. A la ruta creada en 2 (r-opt) se le intercambian 3 arcos al azar. Este proceso da origen a una nueva ruta que se denomina (r-new).
3. Se comparan las longitudes de (r-new) y (r-opt).
4. Si (r-new) < (r-opt), entonces (r-opt) = (r-new). Si (r-new) > (r-opt), se repite el paso 2 hasta que se cumpla n número de iteraciones realizadas, definidas previamente.

3.4.2. Inserción secuencial de Mole & Jameson

La heurística de Mole & Jameson usa dos criterios para la elección de clientes a insertar. Por una parte, se examinan los clientes que aún no han sido visitados y se calcula la mejor posición para ubicarlos en la ruta actual. Se tienen en cuenta las distancias, demandas y no se vuelven a reordenar los clientes que actualmente están en las rutas.

Sea $(v_0, v_1, ..., v_t, v_{t+1})$ una ruta donde $(v_0 = v_{t+1} = 0)$ y 0 es el depósito. Suponiendo que *w* es un cliente que aun no ha sido visitado, entonces el costo de insertar *w* entre $v_i \quad y \quad v_{i+1}$ con (0<= i<=t) es:

$$30. \quad C_1 \ (v_i, w) = \begin{cases} C_{v_i,w} + C_{w,v_{i+1}} - \lambda C_{v_i,v_{i+1}} & si \quad (v_0, v_i, w, v_{i+1}, ..., v_{t+1}) \quad es\ factible; \\ \infty \quad si\ no\ es\ factible. \end{cases}$$

La mejor posición para insertar el cliente *w* en la ruta que se está construyendo actualmente es:

$$31. \quad i(w) = \operatorname{argmin} \; C_1(v_i, w) \qquad i = 0,...,t.$$

Usando esta metodología es evidente que los clientes que se van insertando en la ruta actual, son los más cercanos a los que pertenecen a esta. Para evitar que los clientes lejanos solo sean tenidos en cuenta en las últimas iteraciones se usa C_2 o también llamada medida de urgencia. Esta medida es el segundo criterio que la heurística de inserción de Mole and Jameson utiliza.

C_2 se define como:

$$32. \quad C_2(v_i, w) = \mu C_{0,w} - C_1(v_i, w) \quad \textit{para cada cliente } w.$$

En cada iteración se busca el cliente que maximice la medida de urgencia C_2 y se inserta en la posición que da el mínimo valor de C_1.

Para revisar la factibilidad de la inserción, se verifica la capacidad disponible del vehículo y la demanda del cliente a insertar. De no ser factible, y si ninguna de las demandas de los clientes w es menor o igual que la capacidad disponible del vehículo, se da por terminada esa ruta y se comienza con otra.

En la figura 12 se describe el funcionamiento paso a paso del algoritmo.

Figura 12. Secuencia del algoritmo de Mole & Jameson

Paso 1 (Creación de una ruta): Si todos los clientes pertenecen a alguna ruta, terminar. Si no, seleccionar un cliente no visitado w y crear la ruta *r* = (0, *w*, 0).

Paso 2 (Inserción): Sea r = $(v_0, v_1, ..., v_t, v_{t+1})$ donde $(v_0 = v_{t+1} = 0)$. Para cada cliente no visitado *w*, calcular $i(w) = \operatorname{argmin} \; C_1(v_i, w)$ Si no hay inserciones

factibles, ir al paso 1. Calcular $w^* = \text{argmax}_w C_2(v_{i(w)}, w)$ Insertar w* luego de $v_{i(w^*)} en\, r$

Paso 3 (Optimización): Aplicar el algoritmo 3-opt sobre r. Ir al paso 2.

Fuente: ALFREDO OLIVERA, "Heurísticas para el problema de ruteo de vehículos", Instituto de computación, Facultad de Ingeniería. Universidad de la República, Montevideo, Uruguay.

EJEMPLO:

Tabla 5. Demandas ejemplo Mole and Jameson

Clientes	1	2	3	4	5	6
Demandas	10	20	15	12	17	8
K Capacidad	50					
λ	1					
μ	1					

Fuente: Autores

Tabla 6. Primera iteración ejemplo Mole and Jameson

PRIMERA ITERACION								
Medida C_1	**$Cv_{i,w}$**	**$Cw_{,vi+1}$**	**$Cv_{i,vi+1}$**	**$C_1(V_i,W)$**	**Medida C_2**	**$C_{o,w}$**	**$C_1(V_i,W)$**	**$C_2(V_i,W)$**
$C_1(0,1)$	10	12	0	22	**$C_2(0,1)$**	10	22	-12
$C_1(0,2)$	7	10	0	17	**$C_2(0,2)$**	7	17	-10
$C_1(0,3)$	5	5	0	10	**$C_2(0,3)$**	5	10	-5
$C_1(0,4)$	11	11	0	22	**$C_2(0,4)$**	11	12	-1
$C_1(0,5)$	8	11	0	19	**$C_2(0,5)$**	8	19	-11
$C_1(0,6)$	8	2	0	10	**$C_2(0,6)$**	8	10	-2

Fuente: Autores

Tabla 7. 3 – OPT primera iteración ejemplo Mole and Jameson

3 - OPT				**CT**
Ruta Parcial	0	4	0	12
K Actual	38			

Fuente: Autores

Tabla 8. Segunda iteración ejemplo Mole and Jameson

SEGUNDA ITERACION								
Medida C_1	**$Cv_{i,w}$**	**$C_{w,vi+1}$**	**$Cv_{i,vi+1}$**	**$C_1(V_i,W)$**	**Medida C_2**	**$C_{o,w}$**	**$C_1(V_i,W)$**	**$C_2(V_i,W)$**
$C_1(0,1)$	10	7	11	6	**$C_2(0,1)$**	10	6	4
$C_1(4,1)$	4	12	1	15	**$C_2(4,1)$**	10	15	-5
$C_1(0,2)$	7	2	11	-2	**$C_2(0,2)$**	7	-2	9
$C_1(4,2)$	7	10	1	16	**$C_2(4,2)$**	7	16	-9
$C_1(0,3)$	5	4	11	-2	**$C_2(0,3)$**	5	-2	7
$C_1(4,3)$	10	5	1	14	**$C_2(4,3)$**	5	14	-9
$C_1(0,5)$	8	8	11	5	**$C_2(0,5)$**	8	5	3
$C_1(4,5)$	5	11	1	15	**$C_2(4,5)$**	8	15	-7
$C_1(0,6)$	8	7	11	4	**$C_2(0,6)$**	8	4	4
$C_1(4,6)$	9	2	1	10	**$C_2(4,6)$**	8	10	-2

Fuente: Autores

Tabla 9. 3 – OPT segunda iteración ejemplo Mole and Jameson

3 - OPT					CT
Ruta Parcial	0	2	4	0	10
Cambio de Aristas	0	4	2	0	28
K Actual	18				

Fuente: Autores

Tabla 10. Tercera iteración ejemplo Mole and Jameson

TERCERA ITERACION								
Medida C_1	**$Cv_{i,w}$**	**$C_{w,vi+1}$**	**$Cv_{i,vi+1}$**	**$C_1(V_i,W)$**	**Medida C_2**	**$C_{o,w}$**	**$C_1(V_i,W)$**	**$C_2(V_i,W)$**
$C_1(0,1)$	10	5	7	8	**$C_2(0,1)$**	10	8	2
$C_1(2,1)$	9	7	2	14	**$C_2(2,1)$**	10	14	-4
$C_1(4,1)$	4	12	1	15	**$C_2(4,1)$**	10	15	-5
$C_1(0,3)$	5	7	7	5	**$C_2(0,3)$**	5	5	0
$C_1(2,3)$	10	4	2	12	**$C_2(2,3)$**	5	12	-7
$C_1(4,3)$	10	5	1	14	**$C_2(4,3)$**	5	14	-9
$C_1(0,5)$	8	10	7	11	**$C_2(0,5)$**	8	11	-3
$C_1(2,5)$	6	8	2	12	**$C_2(4,5)$**	8	12	-4
$C_1(4,5)$	5	11	1	15	**$C_2(4,5)$**	8	15	-7
$C_1(0,6)$	8	9	7	10	**$C_2(0,6)$**	8	10	-2
$C_1(2,6)$	4	7	2	9	**$C_2(2,6)$**	8	9	-1
$C_1(4,6)$	9	2	1	10	**$C_2(4,6)$**	8	10	-2

Fuente: Autores

Tabla 11. 3 – OPT tercera iteración ejemplo Mole and Jameson

3 - OPT						CT
Ruta Parcial	0	1	2	4	0	18
Cambio de Aristas	0	2	1	4	0	24
	0	1	4	2	0	34
	0	4	1	2	0	30
K Actual	8					

Fuente: Autores

Se crea una nueva ruta debido a que la capacidad actual es 8 y ninguna de las demandas de los nodos que aun no han sido insertados es menor a este valor.

Tabla 12. Cuarta iteración ejemplo Mole and Jameson

CUARTA ITERACION								
Medida C_1	**$Cv_{i,w}$**	**$Cw_{,vi+1}$**	**$Cv_{i,vi+1}$**	**$C_1(V_i,W)$**	**Medida C_2**	**$C_{o,w}$**	**$C_1(V_i,W)$**	**$C_2(V_i,W)$**
$C_1(0,3)$	5	5	0	10	**$C_2(0,3)$**	5	10	-5
$C_1(0,5)$	8	11	0	19	**$C_2(0,5)$**	8	19	-11
$C_1(0,6)$	8	2	0	10	**$C_2(0,6)$**	8	10	-2

Fuente: Autores

Tabla 13. 3 – OPT cuarta iteración ejemplo Mole and Jameson

3 - OPT				CT
Ruta Parcial	0	6	0	10
K Actual	42			

Fuente: Autores

Tabla 14. Quinta iteración ejemplo Mole and Jameson

QUINTA ITERACION								
Medida C_1	**$Cv_{i,w}$**	**$Cw_{,vi+1}$**	**$Cv_{i,vi+1}$**	**$C_1(V_i,W)$**	**Medida C_2**	**$C_{o,w}$**	**$C_1(V_i,W)$**	**$C_2(V_i,W)$**
$C_1(0,3)$	5	2	8	-1	**$C_2(0,3)$**	5	-1	6
$C_1(6,3)$	3	5	2	6	**$C_2(6,3)$**	5	6	-1
$C_1(0,5)$	8	11	8	11	**$C_2(0,5)$**	8	11	-3
$C_1(6,5)$	4	11	2	13	**$C_2(6,5)$**	8	13	-5

Fuente: Autores

Tabla 15. 3 – OPT quinta iteración ejemplo Mole and Jameson

3 - OPT					CT
Ruta Parcial	0	3	6	0	9
Cambio de Aristas	0	6	3	0	16
K Actual	27				

Fuente: Autores

Tabla 16. Sexta iteración ejemplo Mole and Jameson

SEXTA ITERACION								
Medida C_1	$Cv_{i,w}$	Cw,v_{i+1}	$Cv_{i,vi+1}$	$C_1(V_i,W)$	Medida C_2	$C_{o,w}$	$C_1(V_i,W)$	$C_2(V_i,W)$
$C_1(0,5)$	8	15	5	18	$C_2(0,5)$	8	18	-10
$C_1(3,5)$	3	11	2	12	$C_2(3,5)$	8	12	-4
$C_1(6,5)$	4	11	2	13	$C_2(6,5)$	8	13	-5

Fuente: Autores

Tabla 17. 3 – OPT sexta iteración ejemplo Mole and Jameson

3 - OPT						CT
Ruta Parcial	0	5	3	6	0	27
Cambio de Aristas	0	3	5	6	0	21
	0	3	6	5	0	22
	0	6	3	5	0	25
K Actual	10					

Fuente: Autores

Tabla 18. Solución final ejemplo Mole and Jameson

	Clientes	Capacidad Usada	K	Costo	CT
RUTAS FINALES	0-3-5-6-0	40	50	21	39
	0-1-2-4-0	42		18	

Fuente: Autores

3.4.3. Inserción en paralelo de Christofides, Mingozzi y Toth

La inserción en paralelo de Christofides, Mingozzi y Toth opera en 2 fases. En la fase 1 se calculan la cantidad de rutas a asignar y el cliente que inicia en cada una de las rutas. En la fase 2, se crean las rutas y se insertan el resto de clientes.

Fase 1. Se aplica un algoritmo de inserción secuencial. No se da importancia a la ubicación de los clientes dentro de las rutas debido a que solo interesa encontrar el cliente que iniciara la ruta y el total de rutas a asignar. Para comenzar la ruta k-ésima, se busca un cliente V_k que no haya sido visitado aun. El costo de insertar el cliente *W* en la ruta que contiene V_k es:

$$33. \quad \delta_{w,v_k} = C_{0,w} + \lambda\, C_{w,v_k}$$

Si no es factible esta inserción, entonces $\delta_{w,v_k} = \infty$.

A la ruta se le insertan los clientes que tengan los menores valores de (δ) hasta que no hayan mas inserciones factibles. Cuando no se encuentren más inserciones factibles, se crea una nueva ruta o si no hay mas nodos w para insertar, se termina el algoritmo.

Figura 13. Algoritmo de Christofides, Mingozzi y Toth. Fase 1

Paso 1 (Nueva ruta). Hacer k = 1.
Paso 2 (Cliente inicial). Seleccionar un cliente no visitado v_k para insertar en la ruta. Para cada cliente no visitado *w*, calcular δ_{w,v_k} .

Paso 3 (Inserciones). Calcular $w^* = \arg\min_w \delta_{w,v_k}$ sobre los clientes no visitados w. Insertar w∗ en la ruta y aplicar el algoritmo 3-opt. Si quedan clientes no visitados que puedan insertarse en la ruta, ir a 3.
Paso 4 (Siguiente ruta). Si todos los clientes pertenecen a alguna ruta, terminar. Si no, hacer k = k + 1 e ir a 2.

Fuente: ALFREDO OLIVERA, “Heurísticas para el problema de ruteo de vehículos”, Instituto de computación, Facultad de Ingeniería. Universidad de la República, Montevideo, Uruguay.

Fase 2. En esta segunda fase de la inserción e paralelo de Christofides, Mingozzi y Toth, se crean las rutas y se inician con los clientes seleccionados en la fase 1

paso 2. Cada uno de los clientes w se asocia a la ruta a la cual se minimiza su costo de inserción. El orden de inserción de los clientes w se calcula obteniendo la diferencia entre el costo de su inserción y el costo de la segunda mejor opción para este cliente. Entre más grande sea la diferencia es mayor la urgencia de insertarlo.

Figura 14. Algoritmo de Christofides, Mingozzi y Toth. Fase 2

Paso 5 (Inicialización): Crear k rutas $r_t = (0, v_t, 0)$ para t = 1,. . ., k, siendo k la cantidad de rutas obtenidas en la fase 1. Sea $J = \{r_1, ..., r_k\}$ conjunto de rutas a asignar.

Paso 6 (Asociación): Para cada cliente w que no haya sido visitado calcular $t_w = \text{argmin}_{(t|r_t \in J)} \delta_{w,v_t}$.

Paso 7 (Urgencias): Seleccionar $r_t \in$ J y hacer $J = J \setminus \{r_t\}$. Para cada cliente w tal que $t_w = t$, calcular $t'_w = \text{argmin}_{(t|r_t \in J)} \delta_{w,v_t}$ y $\tau_w = t'_w - t_w$.

Paso 8 (Inserción): Calcular $w^* = \text{argmax}_{(w|t_w = t} \tau_w$ Insertar w^* en la ruta r_t y aplicar el algoritmo 3-opt. Si quedan clientes asociados a r_t que pueden ser insertados, ir a 8.

Paso 9 (Finalización): Si $J \neq \phi$, ir a 6. Si todos los clientes han sido visitados, terminar. Si no, aplicar el algoritmo nuevamente (incluyendo la fase 1) sobre los clientes no visitados.

Fuente: ALFREDO OLIVERA, “Heurísticas para el problema de ruteo de vehículos”, Instituto de computación, Facultad de Ingeniería. Universidad de la República, Montevideo, Uruguay.

EJEMPLO:

Tabla 19. Matriz costos ejemplo Christofides, Mingozzi y Toth.

MATRIZ COSTOS							
	0	**1**	**2**	**3**	**4**	**5**	**6**
0	**X**	10	7	5	11	8	8
1	12	**X**	5	4	7	9	1
2	10	9	**X**	10	2	6	4
3	5	8	7	**X**	4	3	2
4	1	4	7	10	**X**	5	9
5	11	12	10	15	8	**X**	11
6	2	7	9	3	7	4	**X**

Fuente: Autores

Tabla 20. Demandas ejemplo Christofides, Mingozzi y Toth

Clientes	1	2	3	4	5	6
Demandas	10	15	13	17	20	16
K Capacidad	54					
Λ	1					

Fuente: Autores

Tabla 21. Selección nodos iniciales ejemplo Christofides, Mingozzi y Toth

Seleccion	**k1**	**k2**	**Justificacion**
Nodos iniciales	3	5	Representan los menores cotos desde el deposito

Fuente: Autores

Tabla 22. Fase 1, ruta 1 ejemplo Christofides, Mingozzi y Toth

$\delta_{(w,v_k)}$	$C_{o,w}$	$C_{(w,v_k)}$	$\delta_{(w,v_k)}$	Orden de Asignacion	k1	Cap. Actual
$\delta_{(1,3)}$	10	4	14	2	x	15
$\delta_{(2,3)}$	7	10	17	3	x	0
$\delta_{(4,3)}$	11	10	21	4	-	Sin capacidad
$\delta_{(5,3)}$	8	15	23	5	-	Sin capacidad
$\delta_{(6,3)}$	8	3	11	1	x	25
Nodo Inicial					3	41

Fuente: Autores

Tabla 23. Fase 1, ruta 2 ejemplo Christofides, Mingozzi y Toth

$\delta_{(w,v_k)}$	$C_{o,w}$	$C_{(w,v_k)}$	$\delta_{(w,v_k)}$	Orden de Asignacion	k2	Cap. Actual
$\delta_{(4,5)}$	11	5	16	1	x	17
Nodo Inicial					5	34

Fuente: Autores

Tabla 24. Fase 2 ejemplo Christofides, Mingozzi y Toth

t_w	$C_{o,w}$	$C_{(w,v_k)}$	$\delta_{(w,v_k)}$	t'_w	$C_{o,w}$	$C_{(w,v_k)}$	$\delta_{(w,v_k)}$	τ_w		Urgencia de Insercion	k1	k2
t_1	10	4	14	t'_1	10	9	19	τ_1	5	1	x	
t_2	7	10	17	t'_2	7	6	13	τ_2	-4	3	x	
t_4	11	10	21	t'_4	11	5	16	τ_4	-5	4		x
t_6	8	3	11	t'_6	8	4	12	τ_6	1	2	x	

Fuente: Autores

Tabla 25. Iteraciones ejemplo Christofides, Mingozzi y Toth

PRIMERA INSERCION			3 - OPT	
Nodo	**Ruta**	**Costo**	**Ruta**	**Costo**
1	0-3-1-0	25	0-1-3-0	19
PRIMERA INSERCION			**3 - OPT**	
Nodo	**Ruta**	**Costo**	**Ruta**	**Costo**
6	0-1-3-6-0	18	0-1-6-3-0	19
			0-6-1-3-0	24
			0-3-1-6-0	16
PRIMERA INSERCION			**3 - OPT**	
Nodo	**Ruta**	**Costo**	**Ruta**	**Costo**
2	0-3-1-6-2-0	33	0-3-1-2-6-0	24
			0-3-2-1-6-0	24
			0-2-3-1-6-0	28
			0-1-3-6-2-0	35
			0-1-6-3-2-0	31
PRIMERA INSERCION			**3 - OPT**	
Nodo	**Ruta**	**Costo**	**Ruta**	**Costo**
4	0-5-4-0	17	0-4-5-0	27

Fuente: Autores

Tabla 26. Solución ejemplo Christofides, Mingozzi y Toth

	Clientes	Capacidad Usada	K	Costo	CT
RUTAS FINALES	0-3-1-2-6-0	54	54	24	41
	0-5-4-0	37		17	

Fuente: Autores

3.4.4. Algoritmo de pétalos

Se supone un grupo de rutas R donde cada una de las rutas r que pertenece a R es factible. Se supone que cada cliente es visitado por varias de las rutas. A este problema de seleccionar el subconjunto de R con un costo mínimo que visite

solamente una vez a cada uno de los clientes se denomina Set Partitioning Problem (SPP).

$$34. \quad \min \sum_{k \in R} c_k x_k$$

s.a.

$$35. \quad \sum_{k \in R} a_{ik} x_k = 1 \quad \forall i \in V \setminus \{0\}$$

$$x_k \in \{0,1\} \quad \forall k \in S$$

Donde a_{ik} vale 1 si el cliente *i* es visitado por la ruta r_k y 0 si no y donde c_k es el costo de la ruta r_k. La variable x_k indica si la ruta r_k es seleccionada o no en la solución final. Esta formulación se debe a Balinski y Quandt [8]. Dado el caso donde R contenga todas las rutas factibles posibles, solucionar el SPP es igual que resolverlo por un método exacto. Normalmente el numero de rutas factibles es exponencial al número de los clientes se selecciona un solo grupo que se denomina de buenas rutas.

EJEMPLO:

Tabla 27. Matriz costos ejemplo algoritmo Pétalos

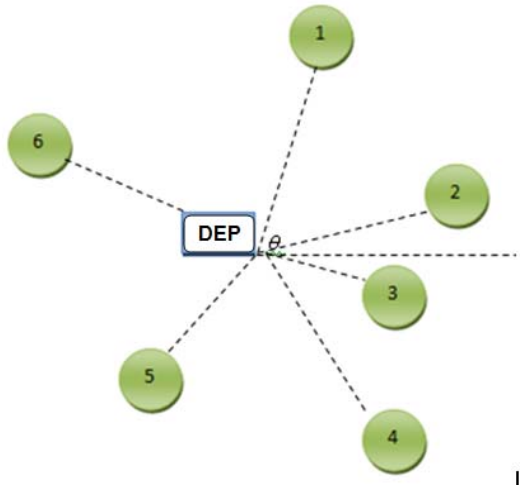
MATRIZ COSTOS							
	0	**1**	**2**	**3**	**4**	**5**	**6**
0	**0**	10	7	5	11	8	8
1	12	**0**	5	4	7	9	1
2	10	9	**0**	10	2	6	4
3	5	8	7	**0**	4	3	2

[8] Heurísticas Clásicas para el VRP

4	1	4	7	10	**0**	5	9
5	11	12	10	15	8	**0**	11
6	2	7	9	3	7	4	**0**

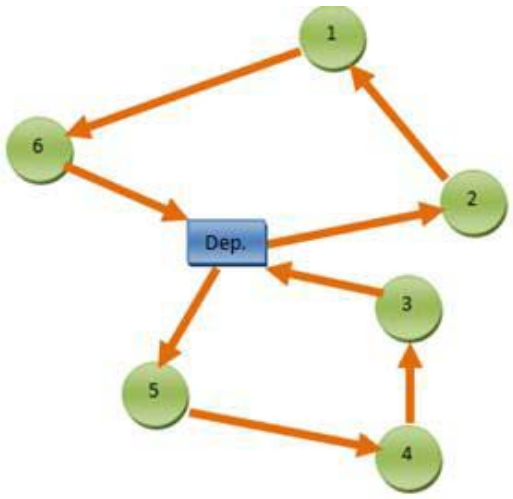
Fuente: Autores

Figura 15. Barrido inicial del algoritmo Pétalos



Fuente: Autores

Figura 16. Solución ejemplo algoritmo Pétalos



Fuente: Autores

Tabla 28. Solución final ejemplo algoritmo Pétalos

Ruta	Nodos					Capacidad	Costo
1	0	2	1	6	0	41	19
2	0	5	4	3	0	50	31
						Costo Total:	50

Fuente: Autores

3.4.5. Método rutear Primero Asignar Después

Para solucionar el CVRP con el método rutear primero asignar después es necesario primero calcular una ruta que visite a todos los clientes. Esta ruta asignada no respeta las restricciones del problema y es particionada en varias rutas donde estas son todas factibles.

Dado $r = (0, v_1, ..., v_n, 0)$ la solución que se obtiene para el CVRP en esta primera fase es determinada por la mejor partición de r que respeta la capacidad del vehículo.

Este tipo de problemas se pueden formular hallando el camino mínimo en el grafo dirigido y acíclico. G = (X, V, W) donde $X = \{0, v_1, ..., v_n\}$. Los arcos del G conectan todo par de clientes v_i y v_j con i < j y tales que la demanda total de los clientes $v_{i+1}, ..., v_j$ no supera la capacidad del vehículo:

$$36. \quad v = \{(v_1, v_j) i < j, \sum_{k=i+1}^{j} d_{vk} \leq Q\}$$

Cada arco (v_i, v_j) se pondera con el costo de la ruta $(0, v_{i+1}, \ldots, v_j, 0)$, es decir:

$$37. \quad w(v_i, v_j) = C_{0,v_{i+1}} + C_{v_j,0} + \sum_{k=i+1}^{j-1} C_{v_k, v_{k+1}}$$

Un arco (v_i, v_j) representa la ruta $(0, v_{i+1}, \ldots, v_j, 0)$. Cada camino de 0 a v_n en G representa una posible partición de la ruta r en rutas que respetan las restricciones de demanda. Por lo tanto, el camino de costo mínimo entre 0 y v_n representa la partición de costo mínimo de la ruta original en rutas que respetan la restricción de capacidad. Como el grafo es acíclico (solo hay arcos (v_i, v_j) con i < j, puede utilizarse el Algoritmo de Dijkstra para hallar dicho camino.

EJEMPLO:

Tabla 29. Matriz costos ejemplo algoritmo Rutear primero asignar después

MATRIZ COSTOS											
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	**0**	10	7	5	11	8	8	5	4	10	8
1	12	**0**	5	4	7	9	1	3	3	7	4
2	10	9	**0**	10	2	6	4	2	5	8	8
3	5	8	7	**0**	4	3	2	9	7	4	6
4	1	4	7	10	**0**	5	9	5	6	6	2
5	11	12	10	15	8	**0**	11	2	4	10	12
6	2	7	9	3	7	4	**0**	9	13	10	11
7	4	5	9	7	2	8	3	**0**	5	7	11
8	10	9	7	5	6	8	9	9	**0**	2	5
9	4	6	7	9	8	12	11	9	5	**0**	3
10	4	9	11	10	4	7	8	5	7	2	**0**

Fuente: Autores

Tabla 30. Demandas ejemplo algoritmo Rutear primero asignar después

Clientes	1	2	3	4	5	6	7	8	9	10
Demandas	10	12	13	17	20	16	14	13	10	15
K Capacidad	75									

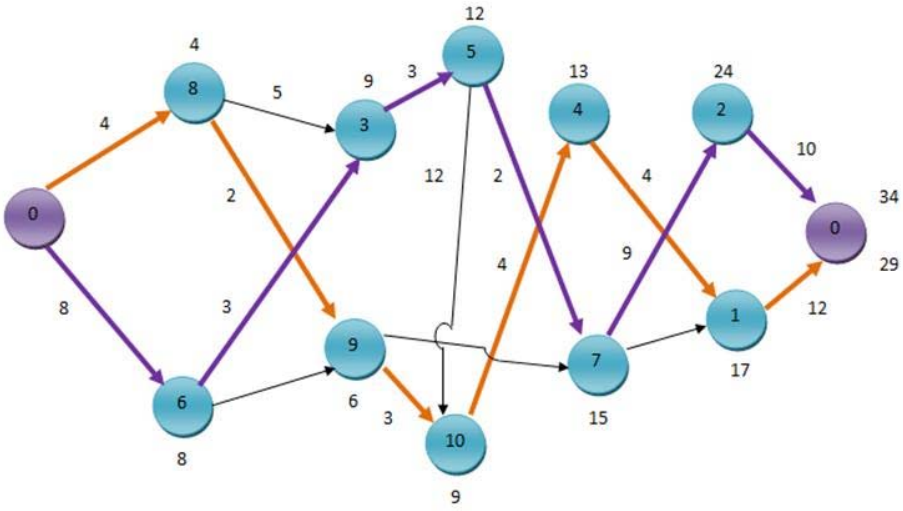
Fuente: Autores

Tabla 31. Solución inicial ejemplo algoritmo Rutear primero asignar después

Solucion inicial
0-8-9-10-4-1-6-3-5-7-2-0

Fuente: Autores

Figura 17. Dijkstra ejemplo algoritmo Rutear primero asignar después



Fuente: Autores

Tabla 32. Solución final ejemplo algoritmo Rutear primero asignar después

RUTAS FINALES	Clientes	Capacidad Usada	K	Costo	CT
	0-8-9-10-4-0	65	75	14	49
	0-6-3-5-7-2-0	75		35	

Fuente: Autores

3.5. Planteamiento de un problema para el CVRP

Ejemplo 2. Una empresa cuyo depósito está ubicado en Bucaramanga (0) debe distribuir sus productos a sus 6 clientes localizados en las ciudades de Barranquilla (1), Bogotá (2), Medellín (3), Leticia (4), Cali (5) e Inírida (6). La matriz de costo asociada a la red de distribución se presenta en la tabla 5:

Tabla 33. Matriz de Costos C_{ij} del ejemplo 2

	0	1	2	3	4	5	6
0	0	8	15	8	7	6	3
1	8	0	12	9	13	4	2
2	15	12	0	6	8	12	9
3	8	9	6	0	6	14	7
4	7	13	8	6	0	5	8
5	6	4	12	14	5	0	11
6	3	2	9	7	8	11	0

Fuente: Autores

Las demandas de los clientes se muestran en la tabla 6

Tabla 34. Demandas de los clientes ejemplo 2

Barranquilla (1)	Bogotá (2)	Medellín (3)	Leticia (4)	Cali (5)	Inírida (6)
32	48	31	39	40	35

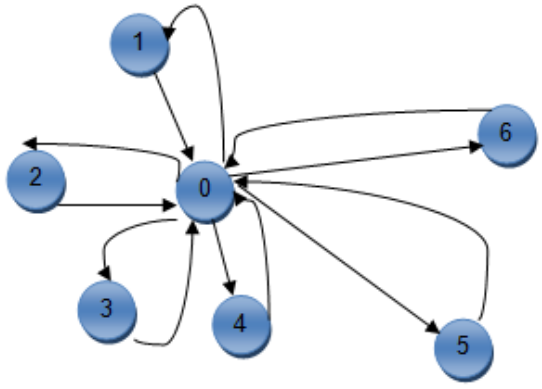
Fuente: Autores

Se tienen tres vehículos para realizar los recorridos y la capacidad de cada vehículo es de 80 unidades.

3.5.1. Desarrollo del ejemplo 2 por el algoritmo de ahorros de Clarke and Wright

En la figura 15 se representan las rutas iníciales:

Figura 18. Rutas Iníciales ejemplo 2



Fuente: Autores

Cada vehículo, sale del depósito 0 en la ciudad de Bucaramanga, a entregar sus pedidos en las 6 ciudades donde están ubicados sus clientes, y regresa al depósito. El costo asociado a esta solución inicial es de

$$38 \quad CT = 2\sum_{i=1}^{6} c_{0i} = 94$$

Los ahorros calculados para cada unión se muestran a continuación en la tabla 7.

Tabla 35. Calculo de Ahorros ejemplo 2

Pares de Rutas	Nueva Ruta	Ahorro Obtenido
(0,1,0) y (0, 2, 0)	(0, 1, 2, 0)	S_{12} = 11
(0,1,0) y (0, 3, 0)	(0, 1, 3, 0)	S_{13}= 7
(0, 1, 0) y (0, 4, 0)	(0, 1, 4, 0)	S_{14}= 2

(0, 1, 0) y (0, 5, 0)	(0, 1, 5, 0)	S_{15}= 18
(0, 1, 0) y (0, 6, 0)	(0, 1, 6, 0)	S_{16}= 10
(0, 2, 0) y (0, 3, 0)	(0, 2, 3, 0)	S_{23}= 17
(0,2,0) y (0, 4, 0)	(0, 2, 4, 0)	S_{24}= 14
(0,2,0) y (0, 5, 0)	(0, 2, 5, 0)	S_{25}= 9
(0,2,0) y (0, 6, 0)	(0, 2, 6, 0)	S_{26}= 9
(0,3,0) y (0, 4, 0)	(0, 3, 4, 0)	S_{34}= 9
(0,3,0) y (0, 5, 0)	(0, 3, 5, 0)	S_{35}= 0
(0,3,0) y (0, 6, 0)	(0, 3, 6, 0)	S_{36}= 4
(0,4,0) y (0, 5, 0)	(0, 4, 5, 0)	S_{45}= 8
(0,4,0) y (0, 6, 0)	(0, 4, 6, 0)	S_{46}= 2
(0,5,0) y (0, 6, 0)	(0, 5, 6, 0)	S_{56}= 0

Fuente: Autores

El número total de ahorros es de:

$$\binom{6}{2}=\frac{6!}{2!(6.-2)!}=\frac{6(6-1)}{2}=15$$

A continuación se muestra en la tabla 8 la lista ordenada de los ahorros calculados.

Tabla 36. Lista ordenada de ahorros ejemplo 2

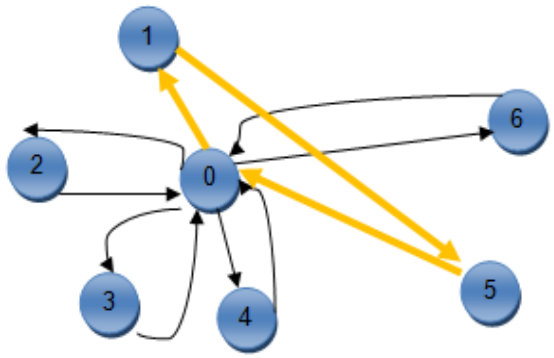
Ruta	Ahorro
S_{15}	18
S_{23}	17
S_{24}	14
S_{12}	11
S_{16}	10
S_{25}	9
S_{26}	9

S_{34}	9
S_{45}	8
S_{13}	7
S_{36}	4
S_{46}	2
S_{14}	2
S_{35}	0
S_{56}	0

Fuente: Autores

El mayor ahorro obtenido fue de 18, es decir, el de la unión de la ruta del cliente 1 con el 5, (0, 1, 5, 0). La capacidad del vehículo es de 80 y la suma de las demandas de estos es de 32+40 = 72. Por lo tanto, esta unión hace parte de la solución, ya que satisface todas las posibles restricciones. Esta ruta la realiza el vehículo 1, por lo tanto no se considera en futuras uniones.

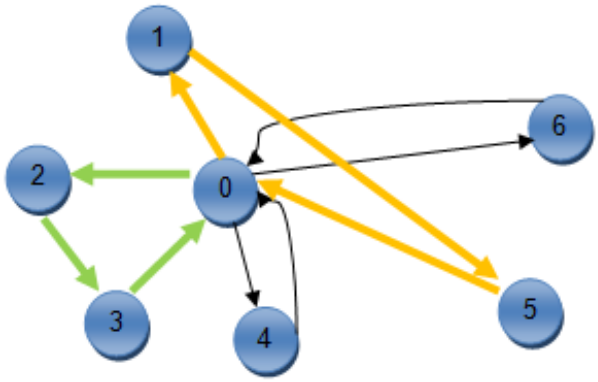
Figura 19. Ruta resultante de la primera iteración



Fuente: Autores

El siguiente mayor ahorro calculado es el de la unión de la ruta del cliente 2 con el cliente 3, (0, 2, 3, 0), que tiene un valor igual a 17. La demanda de 2 y 3 respectivamente es de 48 y 31 unidades, por lo tanto es factible su unión ya que no excede las 80 unidades de capacidad del vehículo 2. Esta segunda iteración hace parte de la solución.

Figura 20. Ruta resultante de la segunda iteración



Fuente: Autores

Cada color del circuito representa una ruta recorrida por un vehículo diferente, es decir, el color amarillo es para el vehículo 1, y el color verde para el vehículo 2.

El siguiente ahorro encontrado es el de la ruta (0, 2, 4, 0) que no es factible de realizar ya que el cliente 2 se encuentra unido al cliente 3, por medio de la ruta (0, 2, 3, 0) y la capacidad del vehículo que recorre dicha ruta ya se ha completado.

El siguiente ahorro corresponde a la ruta (0, 1, 2, 0) que no satisface la condición de no borrar condiciones directas previas, el cliente 1 ya ha sido unido con el cliente 5, y la capacidad de su vehículo está completa. Lo mismo sucede con el ahorro de la ruta (0, 1, 6, 0).

Los siguientes ahorros de las rutas (0, 2, 5, 0) y (0, 2, 6, 0) tampoco son factibles, debido a que el cliente 2, ya ha sido unido con el cliente 3 previamente, además la suma de sus demandas exceden la capacidad del vehículo.

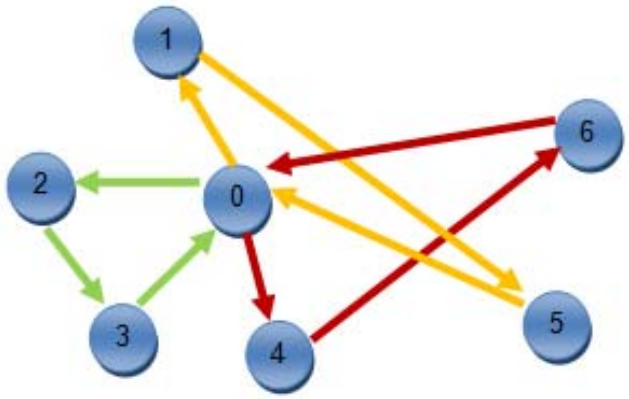
El ahorro de la ruta (0, 3, 4, 0) no es factible debido a que el vehículo que recorre la ruta (0, 2, 3, 0) ya ha completado su capacidad. Igualmente ocurre con el ahorro de la ruta (0, 4, 5, 0), el vehículo que recorre la ruta (0, 1, 5, 0) ha llenado su capacidad.

Los ahorros de la rutas (0, 1, 3, 0) y (0, 3, 6, 0) no son factibles por la condición que algunos de estos clientes se han considerado en uniones previas.

El siguiente ahorro de la ruta (0, 4, 6, 0) es factible, los dos clientes no han sido considerados en uniones anteriores, la suma de sus demandas es de 74 unidades, y no supera la capacidad del vehículo.

El último ahorro de la ruta (0, 1, 4, 0) no es factible ya que estos clientes ya han sido considerados previamente en otras uniones. Las rutas (0, 3, 5, 0) y (0, 5, 6, 0) no representan ningún ahorro por lo tanto no son consideradas.

Figura 21. Solución final del ejemplo 2



Fuente: Autores

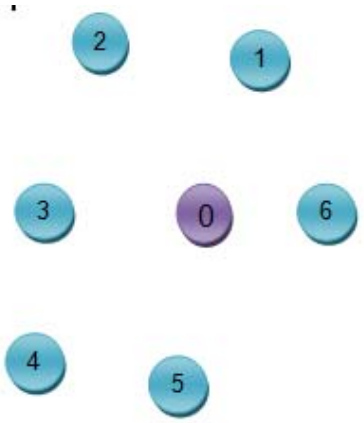
El costo total de la solución final es el costo de: (0,1) + (1,5) + (5, 0) + (0, 2) + (2, 3) + (3, 0) + (0, 4) + (4, 6) + (6, 0), es decir, 8 + 4 + 6 + 15 + 6 + 8 + 7 + 8 + 3 = 65 respectivamente.

Lo que demuestra que hubo un ahorro de 94 – 65 = 29.

3.5.2. Desarrollo del ejemplo 2 por el algoritmo del Vecino más Cercano

En la figura 19 se muestra la ubicación de los 6 clientes respecto al depósito.

Figura 22. Ubicación de los clientes



Fuente: Autores

Para iniciar el recorrido se debe escoger de la matriz distancia la menor distancia que hay entre el depósito y los clientes (Los empates se rompen de manera arbitraria)

Tabla 37. Elección del primer nodo de salida

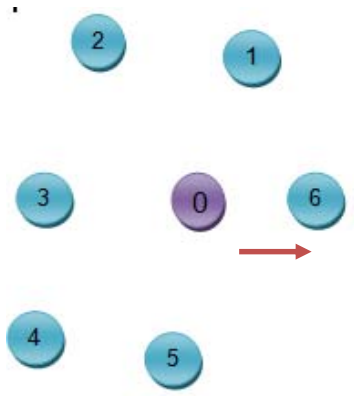
	0	**1**	**2**	**3**	**4**	**5**	**6**
0	0	8	15	8	7	6	3
1	8	0	12	9	13	4	2
2	15	12	0	6	8	12	9
3	8	9	6	0	6	14	7
4	7	13	8	6	0	5	8
5	6	4	12	14	5	0	11
6	3	2	9	7	8	11	0

Fuente: Autores

El recorrido de búsqueda inicia en el depósito (0), fila subrayada con celeste. Desde esta fila se comparan todas las distancias que hay de 0 a cada uno de los clientes, eligiendo la menor de estas. Para este ejemplo, la menor distancia está entre el depósito (0) y el nodo 6. Ahora se verifican las restricciones de capacidad, esto quiere decir que se mira que la demanda del nodo sea menor a la capacidad disponible del vehículo. La demanda del nodo 6 es de 35 unidades, y la capacidad del vehículo es de 80 unidades, de este modo el nodo 6 puede ser agregado a la ruta. La capacidad restante del vehículo es de 45 unidades.

El cliente 6 se elimina de futuras consideraciones, dado que cada cliente puede ser visitado una vez y por un único vehículo.

Figura 23. Primer nodo asignado a la ruta 1



Fuente: Autores

Tabla 38. Eliminación de nodos después de la primera búsqueda

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	0	8	15	8	7	6	3
1	8	0	12	9	13	4	2
2	15	12	0	6	8	12	9
3	8	9	6	0	6	14	7
4	7	13	8	6	0	5	8
5	6	4	12	14	5	0	11
6	3	2	9	7	8	11	0

Fuente: Autores

La asignación de los nuevos clientes a la ruta continúa desde el cliente 6 en su vecindad; en busca del vecino más cercano como se hizo en el paso anterior.

Tabla 39. Elección del segundo nodo a agregar en la ruta

	0	1	2	3	4	5	6
0	0	8	15	8	7	6	3
1	8	0	12	9	13	4	2
2	15	12	0	6	8	12	9
3	8	9	6	0	6	14	7
4	7	13	8	6	0	5	8
5	6	4	12	14	5	0	11
6	3	2	9	7	8	11	0

Fuente: Autores

El cliente más cercano al 6 es el cliente 1, con una distancia de 2 unidades, entre ellos; a continuación se verifica la capacidad disponible del vehículo 1 asignado a la ruta y la demanda del cliente. Para nuestro ejemplo es posible hacer la asignación debido a que la demanda del cliente 1 es de 32 unidades y la capacidad disponible en el vehículo es de 45 unidades. Luego de insertar este cliente a la ruta se actualiza la capacidad del vehículo. Se calcula la nueva capacidad es ahora de: 80 – (32 + 35) = 13 unidades y se compara con las demandas de los clientes que aún no han sido asignados a la ruta; como ninguna de las demandas es ≤ 13 unidades, se procede a hacer uso del vehículo 2 para continuar con el desarrollo del ejercicio.

Tabla 40. Eliminación de nodos después de la segunda búsqueda

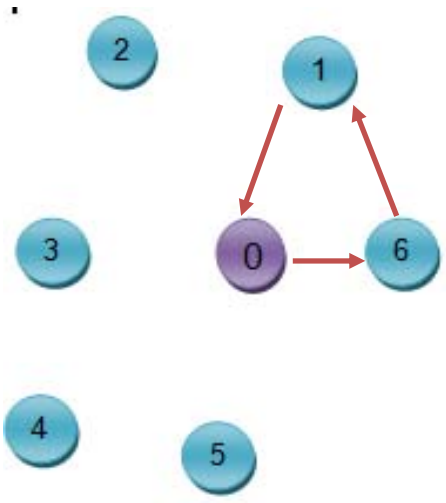
	0	1	2	3	4	5	6
0	0	8	15	8	7	6	3
1	8	0	12	9	13	4	2
2	15	12	0	6	8	12	9
3	8	9	6	0	6	14	7
4	7	13	8	6	0	5	8
5	6	4	12	14	5	0	11
6	3	2	9	7	8	11	0

Fuente: Autores

Como los clientes 1 y 6 son visitados por el vehículo 1 se eliminan de futuras consideraciones.

La matriz de costos con las eliminaciones de las filas y columnas asociadas a los clientes 1 y 6 se muestran en la tabla 12.

Figura 24. Ruta 1



Fuente: Autores

Se procede a elegir el siguiente vehículo que visitará a los clientes aún no asignados. Nuevamente se elige el cliente más cercano al depósito. De la tabla 13 se observa que corresponde al cliente 5.

Tabla 41. Elección del tercer nodo para ser agregado a la solución

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	0	8	15	8	7	6	3
~~**1**~~	~~8~~	~~0~~	~~12~~	~~9~~	~~13~~	~~4~~	~~2~~
2	15	12	0	6	8	12	9
3	8	9	6	0	6	14	7
4	7	13	8	6	0	5	8
5	6	4	12	14	5	0	11
~~**6**~~	~~3~~	~~2~~	~~9~~	~~7~~	~~8~~	~~11~~	~~0~~

Fuente: Autores

Se parte nuevamente del depósito con un vehículo. Se verifica que cumpla con el requisito de capacidad. Capacidad actual del vehículo = 80 unidades. Demanda del nodo 5 = 40 unidades. Nueva capacidad del vehículo = 80 – 40 = 40 unidades.

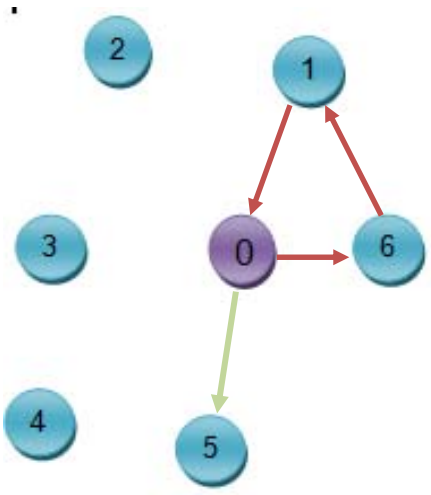
Tabla 42. Eliminación de nodos después de la tercera búsqueda

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	0	8	15	8	7	6	3
1	8	0	12	9	13	4	2
2	15	12	0	6	8	12	9
3	8	9	6	0	6	14	7
4	7	13	8	6	0	5	8
5	6	4	12	14	5	0	11
6	3	2	9	7	8	11	0

Fuente: Autores

Se elimina el cliente 5 de futuras consideraciones.

Figura 25. Primer nodo asignado a la ruta 2.



Fuente: Autores

Se construye la vecindad del nodo 5 para escoger el vecino más próximo. El vecino más cercano del cliente 5 es el cliente 4 con una distancia de 5 unidades. La capacidad restante del vehículo es de 40 unidades y la demanda del cliente 4 es de 39 unidades, por lo tanto el cliente 4 se incluye en la ruta del vehículo 2.

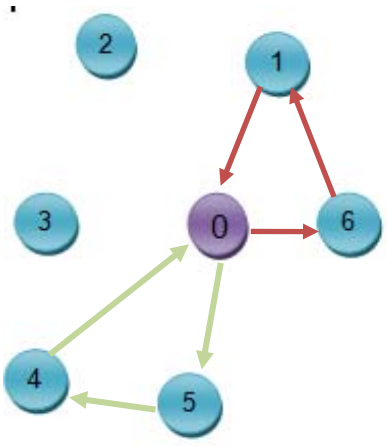
Tabla 43. Elección del cuarto nodo de solución de la ruta

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	0	8	15	8	7	6	3
1	8	0	12	9	13	4	2
2	15	12	0	6	8	12	9
3	8	9	6	0	6	14	7
4	7	13	8	6	0	5	8
5	6	4	12	14	5	0	11
6	3	2	9	7	8	11	0

Fuente: Autores

La capacidad restante del vehículo es de una unidad. Al examinar las demandas de los clientes que aun no se han visitado, se evidencia que ninguno es menor o igual a la capacidad disponible del vehículo, es decir, 1 unidad. Entonces la ruta asignada para el vehículo 2 es: 0 – 5 – 4 – 0.

Figura 26. Rutas 1 y 2



Fuente: Autores

Los clientes 5 y 4 no pueden volver a ser visitados por otro vehículo; por lo tanto, se restringe la llegada y salida de alguno de estos dos clientes para futuras consideraciones. En la tabla 16 se presenta la nueva matriz de costo que se considera para futuras asignaciones.

Tabla 44. Eliminación de nodos después de la cuarta búsqueda

	0	1	2	3	4	5	6
0	0	8	15	8	7	6	3
1	8	0	12	9	13	4	2
2	15	12	0	6	8	12	9
3	8	9	6	0	6	14	7
4	7	13	8	6	0	5	8
5	6	4	12	14	5	0	11
6	3	2	9	7	8	11	0

Fuente: Autores

Se construye la ruta 3 con un nuevo vehículo.

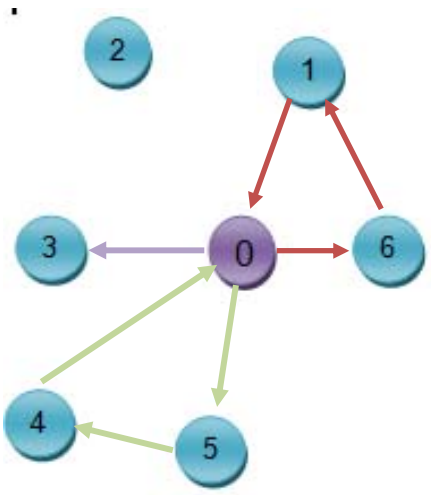
Tabla 45. Elección del quinto nodo de salida para la solución

	0	1	2	3	4	5	6
0	0	8	15	8	7	6	3
1	8	0	12	9	13	4	2
2	15	12	0	6	8	12	9
3	8	9	6	0	6	14	7
4	7	13	8	6	0	5	8
5	6	4	12	14	5	0	11
6	3	2	9	7	8	11	0

Fuente: Autores

El cliente más cercano al depósito es el cliente 3, con una demanda de 31 unidades. La capacidad del vehículo 3 es de 80 unidades, por lo tanto la capacidad disponible del vehículo es de: 80 – 31 = 49 unidades.

Figura 27. Primer nodo asignado a la ruta 3



Fuente: Autores

Tabla 46. Eliminación de nodos después de la quinta iteración

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	0	8	15	8	7	6	3
1	8	0	12	9	13	4	2
2	15	12	0	6	8	12	9
3	8	9	6	0	6	14	7
4	7	13	8	6	0	5	8
5	6	4	12	14	5	0	11
6	3	2	9	7	8	11	0

Fuente: Autores

El cliente 3 se elimina de futuras consideraciones. El vecino del cliente 3 corresponde al cliente 2.

Tabla 47. Elección del sexto y último nodo de salida de la solución

	0	1	2	3	4	5	6
0	0	8	15	8	7	6	3
1	8	0	12	9	13	4	2
2	15	12	0	6	8	12	9
3	8	9	6	0	6	14	7
4	7	13	8	6	0	5	8
5	6	4	12	14	5	0	11
6	3	2	9	7	8	11	0

Fuente: Autores

La demanda del cliente 2 es de 48 unidades, y la capacidad restante del vehículo es de 49 unidades, por lo que cumple con el requisito de capacidad y puede ser agregado a la ruta del vehículo 3.

En este punto ya no hay más clientes por visitar, la solución entonces es:

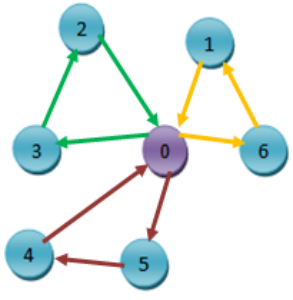
- Para el vehículo 1: 0 – 6 – 1 – 0.
- Para el vehículo 2: 0 – 5 – 4 – 0.
- Para el vehículo 3: 0 – 3 – 2 – 0.

El valor de la solución encontrada es:

$$C_{06}+C_{61}+C_{10}+C_{03}+C_{32}+C_{20}+C_{05}+C_{54}+C_{40}$$

El costo total es de: 3 + 2 + 8 + 8 + 6 + 15 + 6 + 5 + 7 = 60 unidades monetarias.

Figura 28. Solución final del ejemplo 2



Fuente: Autores

Al comparar los resultados obtenidos con el algoritmo de ahorros y el del vecino más cercano:

Para el algoritmo de ahorros, z = 65

Para el algoritmo del vecino más cercano, z = 60

Se concluye que para esta instancia el algoritmo del vecino más cercano es mejor.

4. DESARROLLO DEL SOFTWARE H-CVRP

4.1. Especificaciones de requerimientos funcionales

En este capítulo se describen las características principales, servicios y funcionalidades que presta la herramienta que se usó para el desarrollo del software, Matlab. Aquí se encuentra explícita la información requerida para elaborar el programa en concordancia con los objetivos que se trazaron en el plan de proyecto de grado.

Los requisitos especificados que se describen a continuación, dan una guía de lo que hará y no, la herramienta, por eso es de vital importancia para el usuario leerlos y entenderlos.

4.1.2. Descripción general

El programa que se desarrolló para implementar los métodos heurísticos Vecino más Cercano y Clarke And Wright (o algoritmo de ahorros) para la solución del CVRP se llama H-CVRP (Heuristics for the CVRP).

Los datos de entrada para el H-CVRP son: capacidad de los vehículos, vector demandas, matriz de distancias, matriz de costos y coordenadas de los nodos.

La herramienta permitirá tanto a estudiantes como profesionales en las ramas de la logística determinar rutas “buenas” en una red de transporte.

Los datos de salida del programa son: Grafo de rutas solución, rutas, costo de cada ruta, costo total y la ubicación de los parámetros de entrada.

4.1.2.1. Alcance del programa

El H-CVRP está diseñado tanto para resolver instancias del CVRP desde 50 hasta 300 clientes, sin embargo, funciona también para instancias menores de 50.

Además de ejecutar los dos algoritmos mencionados en la descripción general, incluye en su programación la opción de implementar o no, el parámetro forma de la ruta (λ), permitiendo comparar cuál método arroja mejores resultados.

4.1.2.2. Usuarios Finales

La herramienta de software H-CVRP está dirigida a estudiantes, profesores y la comunidad de la Universidad Industrial de Santander que estén interesados en procesos logísticos de entrega, métodos heurísticos, técnicas avanzadas de investigación de operaciones, problemas de optimización combinatoria. Además la herramienta sirve como base para futuros proyectos de grado en la modalidad de investigación.

Cabe señalar que el uso del H-CVRP requiere de conocimientos previos por parte del usuario ya que esta herramienta de software podría no ser lo suficientemente asistida para cualquier usuario. El objetivo de este trabajo de investigación es la programación de los algoritmos de solución y no el programa de computadora que lo implementó.

4.1.3. Requerimientos de interfaces externas

Esta sección, muestra las características y requerimientos de la herramienta software para con el entorno.

4.1.3.1. Interfaces de Usuario

El manejo de las utilidades del programa se podrá realizar a través de un mouse, o de teclas de acceso rápido del teclado convencional. Al ejecutar el programa se le presenta al usuario una ventana con las siguientes opciones que debe elegir:

1. Algoritmo del Vecino más Cercano
2. Algoritmo de Ahorros de Clarke and Wright
3. Parámetros de la red de transporte

La solución del problema se presenta en forma gráfica y en forma de texto. Las gráficas son las diferentes rutas para la distribución de bienes según el algoritmo escogido. En el texto se muestran los datos de salida junto con sus ubicaciones dentro del sistema.

4.1.3.2. Interfaces de Hardware

H-CVRP está adecuado para trabajar en equipos de cómputo que cumplan como mínimo con las siguientes especificaciones:

Tabla 48. Requerimientos de hardware

1	Teclado convencional
2	Mouse
3	Monitor con resolución mínima. 800x600dpi
4	CPU con procesador compatible x86

Fuente: Autores

4.1.3.3. Interfaces de Software

Para el buen funcionamiento del programa de computadora, se necesita que el equipo donde se instalará tenga un sistema operativo que soporte la herramienta de software Matlab en la versión R2010a.

4.1.4. Requerimientos

A continuación se presentan las características y requerimientos de la herramienta software, relacionadas con el modulo de datos, la interfaz del usuario y el algoritmo de solución.

Tabla 49. Requerimientos de interfaz del usuario

#	**Descripción de los requerimientos para la interfaz del usuario**
RQ-1.1	El programa H-CVRP utiliza archivos para grabar información de los datos del problema.
RQ-1.2	El programa H-CVRP puede utilizar archivos para grabar información de la solución del problema.
RQ-1.3	El programa H-CVRP no necesita de una base de datos para el manejo de la información requerida para el desarrollo del problema.
RQ-1.4	El programa H-CVRP utiliza archivos para cargar la información con los datos del problema.

Fuente: Autores

Tabla 50. Requerimientos de interfaz gráfica

#	**Descripción de los requerimientos para la interfaz Gráfica**
RQ-1.5	El programa H-CVRP está basado en un sistema de ventanas gráficas.
RQ-1.6	El programa H-CVRP permite la modificación de datos del problema para el cálculo de nuevas soluciones.
RQ-1.7	Las ventanas del programa H-CVRP tienen las mismas características entre sí como color, tamaño de las letras, botones etc.

RQ-1.8	Las ventanas que se despliegan en el programa H-CVRP son de tipo modal, esto quiere decir que no se pueden tener activas más de una a la vez.
RQ-1.9	El número de clientes debe ser un parámetro numérico no mayor a 5 caracteres.
RQ-1.10	La capacidad de los vehículos debe ser un parámetro numérico no mayor a 5 caracteres.
RQ-1.11	La capacidad de los vehículos debe ser un número entero.
RQ-1.12	La captura del número de clientes no permite el ingreso de número diferentes a enteros.
RQ-1.13	La captura de la capacidad del vehículo no permite el ingreso de número diferentes a enteros.
RQ-1.14	Las coordenadas de ubicación de los nodos permite el uso de enteros negativos.
RQ-1.15	Las coordenadas de ubicación de los nodos, no permite parámetros numéricos mayor a 10 caracteres.
RQ-1.16	La captura del vector demandas se hace por medio de un archivo en formato .xls
RQ-1.17	La captura de la matriz costos se hace por medio de un archivo en formato .xls
RQ-1.18	La captura de las coordenadas de los nodos se hace por medio de un archivo en formato .xls
RQ-1.19	La captura de la matriz distancia se hace por medio de un archivo en formato .xls
RQ-1.20	El ítem Vecino más cercano, permite calcular las rutas usando este algoritmo heurístico.
RQ-1.21	El ítem Clarke And Wright permite calcular las rutas usando esta metodología heurística.
RQ-1.22	Para el desarrollo del problema por medio de la heurística vecino se deben cargar los datos de demandas y coordenadas. (Matriz distancia, opcional).
RQ-1.23	Para el desarrollo del problema por medio de la heurística de Clarke and Wright se deben cargar los datos de demandas y coordenadas. (Matriz costos, opcional).
RQ-1.24	El ítem parámetro lambda concerniente a la mejora del algoritmo de Clarke and Wright debe tener valores entre 0 y 1.
RQ-1.25	La magnitud de la matriz de costos debe superar en 1 la magnitud del vector demandas.

RQ-1.26	La magnitud de la matriz distancias debe ser igual a la magnitud de la matriz costos.
RQ-1.27	La magnitud de la tabla con las coordenadas de los nodos debe ser igualа la magnitud de la matriz costos o distancias.
RQ-1.28	La gráfica de las rutas asignadas por medio de la metodología del vecino más cercano, requiere que los datos hayan sido ingresados en su totalidad y de manera adecuada.
RQ-1.29	La gráfica de las rutas asignadas por medio de la metodología de Clarke and Wright, requiere que los datos hayan sido ingresados en su totalidad y de manera adecuada.
RQ-1.30	El ítem calcular requiere que los datos hayan sido ingresados de la manera correcta.
RQ-1.31	El ítem reiniciar requiere que se haya corrido con anterioridad un ejercicio.
RQ-1.32	El ítem guardar, requiere que ya se hayan calculado las rutas.
RQ-1.33	El ítem ayuda se puede ejecutar durante y después del cálculo de las rutas.

Fuente: Autores

Tabla 51. Valores requeridos para el ingreso del problema en la aplicación

PARAMETROS	DESCRIPCION	REQUISITOS
Número de clientes	Número de nodos a atender	RQ-1.9 y RQ-1.12
Capacidad de los vehículos	Cantidad máxima de unidades que puede transportar un vehículo.	RQ-1.10 y RQ-1.11
Coordenadas	Ubicación de cada uno de los nodos en el plano cartesiano.	RQ-1.14 , RQ-1.15 y RQ-1.18
Demandas	Número de unidades a transportar a cada nodo	RQ-1.16
Distancias	Distancia entre los nodos	RQ-1.19
Costos	Unidades monetarias entre los nodos	RQ-1.17

Fuente: Autores

Tabla 52. Ítems de la ventana de ejecución

ITEM	DESCRIPCION	REQUISITOS
VECINO MAS CERCANO	Metodología heurística para la solución del CVRP	RQ-1.20 y RQ-1.22
CLARKE AND WRIGHT	Metodología heurística para la solución del CVRP	RQ-1.21 y RQ-1.23
CALCULAR	Permite correr el programa y encontrar las rutas según el o los algoritmos elegidos	RQ-1.30
REINICIAR	Permite introducir datos para el cálculo de un nuevo ejercicio.	RQ-1.31
AYUDA	Este ítem despliega una ventana con algunos tips para la ejecución del software	RQ-1.33
GUARDAR	Permite guardar el ejercicio y los gráficos correspondientes a la asignación de rutas que se calculo en el momento inmediatamente anterior.	RQ-1.32

Fuente: Autores

Tabla 53. Requisitos de los algoritmos desarrollados

#	Descripción de los requerimientos de los algoritmos de solución
RQ-1.34	EL programa H-CVRP valida la información registrada y cargada del problema con los datos esperados a saber.
RQ-1.35	El número de clientes debe ser mayor que cero.
RQ-1.36	**Debe haber un único depósito.**
RQ-1.37	La capacidad de cada vehículo no puede ser excedida por la demanda de un solo cliente.
RQ-1.38	EL sistema debe ser modular. Un modulo de procesos y un modulo de interfaces.
RQ-1.39	El modulo de procesos debe ser independiente de la interfaz que puede ver el usuario

Fuente: Autores

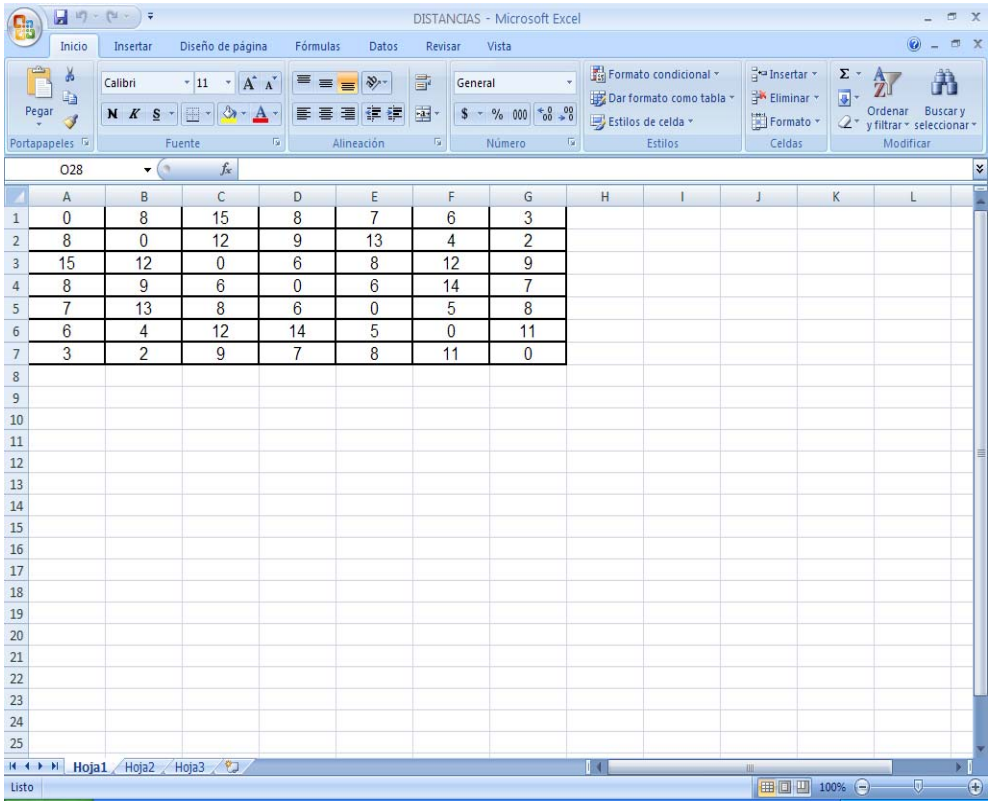
4.2. Desarrollo del ejemplo 2 con H-CVRP

Todos los datos de entrada y de salida del programa, asumen el depósito como 1.

4.2.1. Ingreso de los datos al programa

Los archivos en formato .xls con los parámetros del modelo se muestran en las figuras 26, 27 y 28.

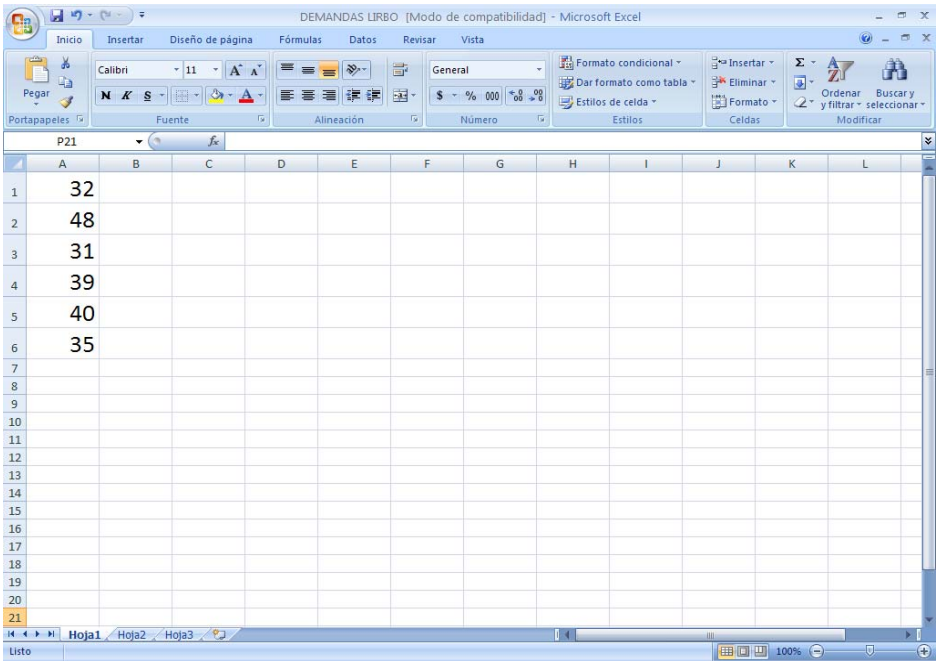
Figura 29. Matriz Costos



	A	B	C	D	E	F	G
1	0	8	15	8	7	6	3
2	8	0	12	9	13	4	2
3	15	12	0	6	8	12	9
4	8	9	6	0	6	14	7
5	7	13	8	6	0	5	8
6	6	4	12	14	5	0	11
7	3	2	9	7	8	11	0

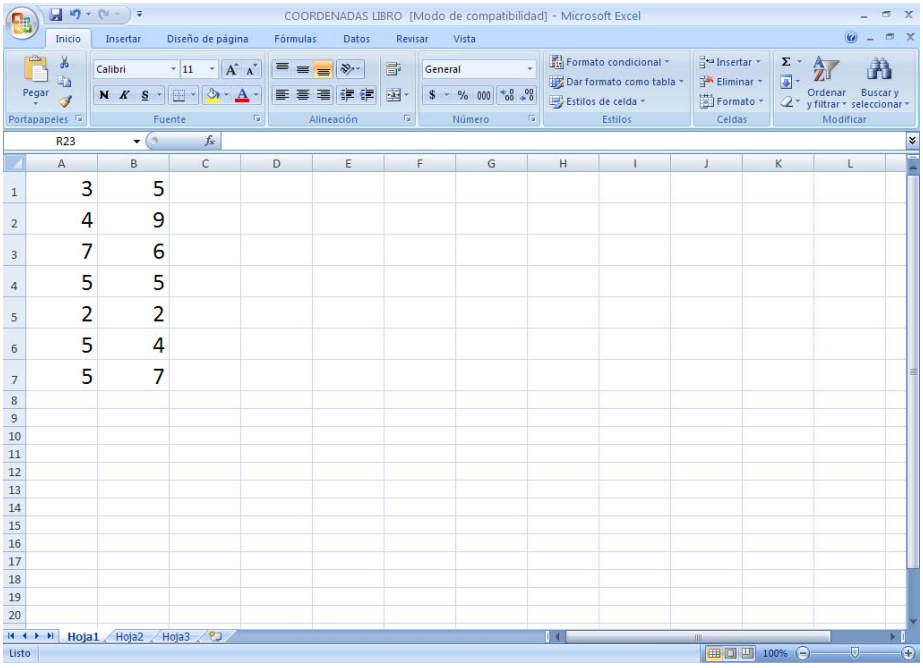
Fuente: Autores

Figura 30. Demandas de los clientes



Fuente: Autores

Figura 31. Coordenadas de los nodos

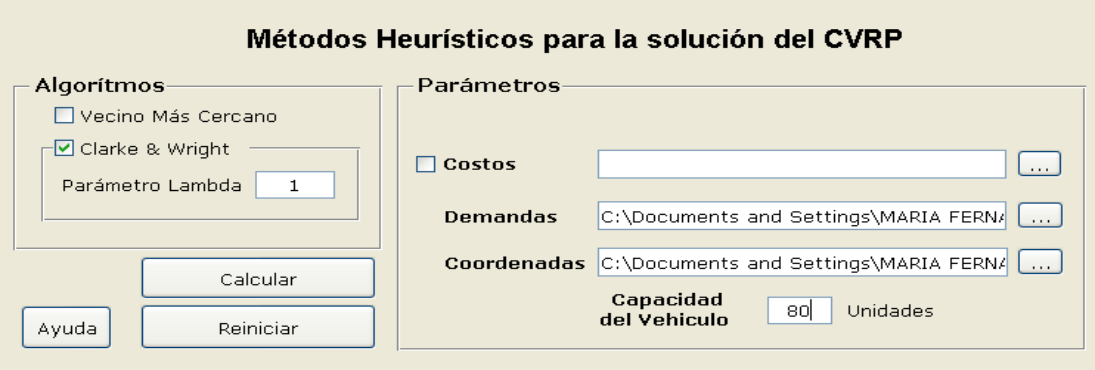


Fuente: Autores

4.2.2. Desarrollo del ejemplo 2 usando el método heurístico de Clarke and Wright

En la figura 29 se muestran los diferentes parámetros del ejemplo desarrollado. Se usa el valor de λ = 1.

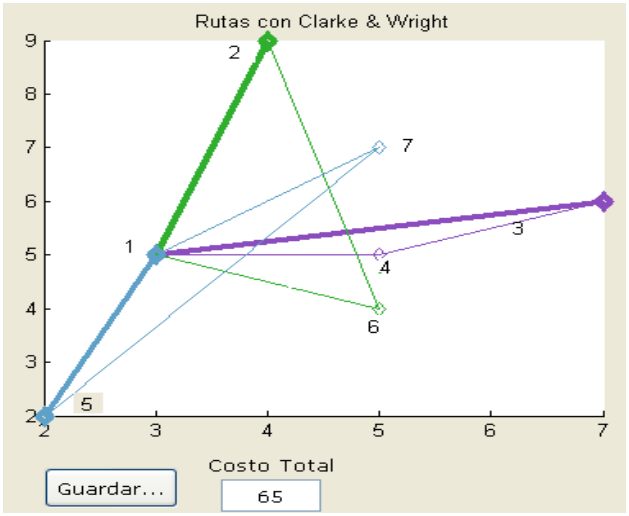
Figura 32. Parámetros de entrada para el algoritmo de ahorros Clarke and Wright



Fuente: Programa H-CVRP - Autores

La figura 30 muestra la salida del programa con el método de Clarke and Wright.

Figura 33. Resultado arrojado por H-CVRP por el método heurístico Clarke and Wright para el ejemplo 2.



Fuente: Programa H-CVRP – Autores

La solución arroja un costo total de 65 unidades monetarias. Las rutas son las siguientes:

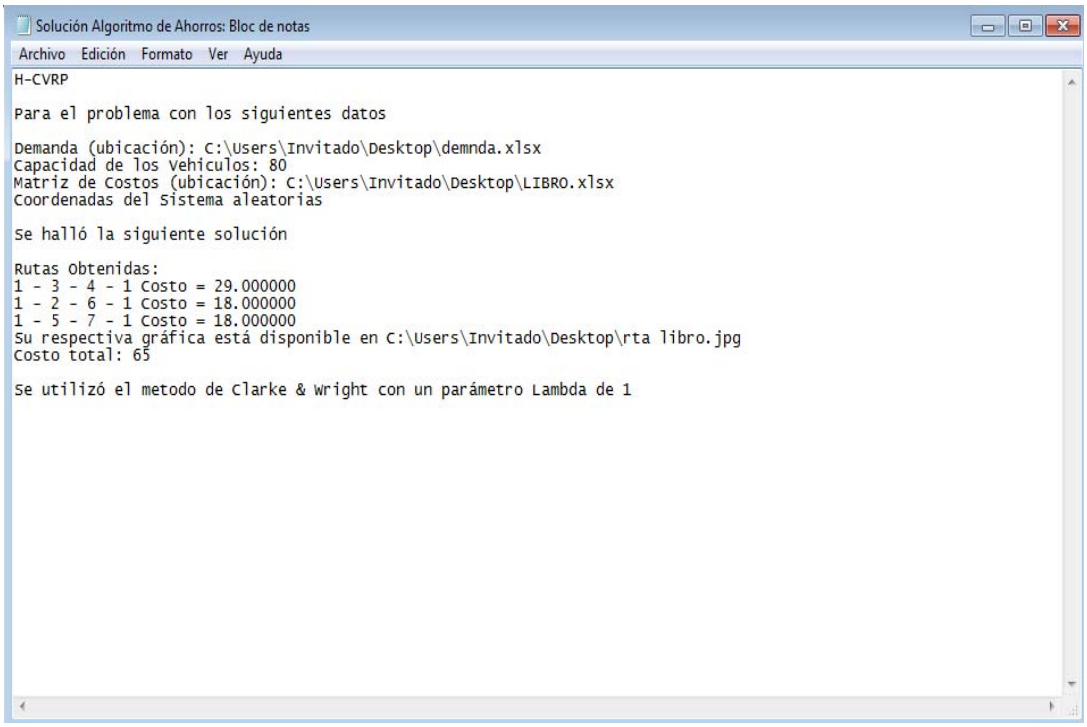
Ruta 1: 1 – 3 – 4 – 1.

Ruta 2: 1 – 2 – 6 – 1.

Ruta 3: 1 – 5 – 7 – 1.

La opción guardar, almacena todos los datos de entrada y salida del problema, con sus respectivas ubicaciones en el sistema.

Figura 34. Archivo de salida .txt del software H-CVRP usando el algoritmo de Clarke and Wright

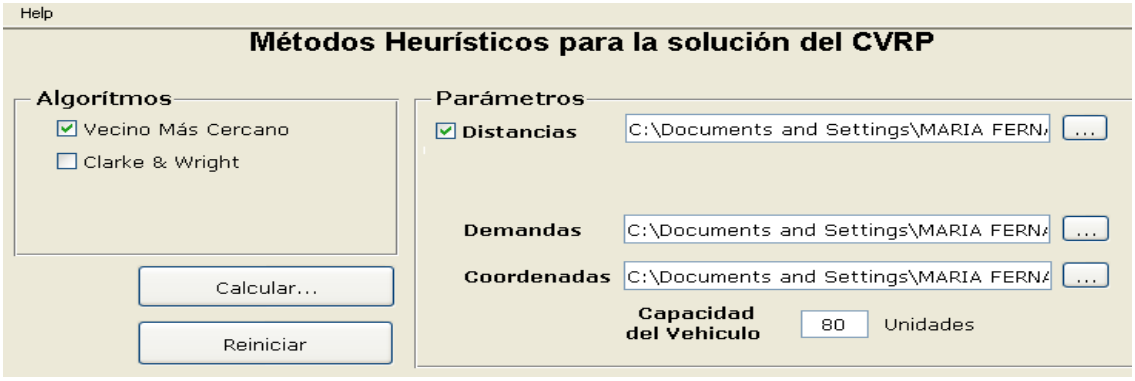


Fuente: Autores

4.2.3. Desarrollo del ejemplo 2 usando el método heurístico del vecino más cercano

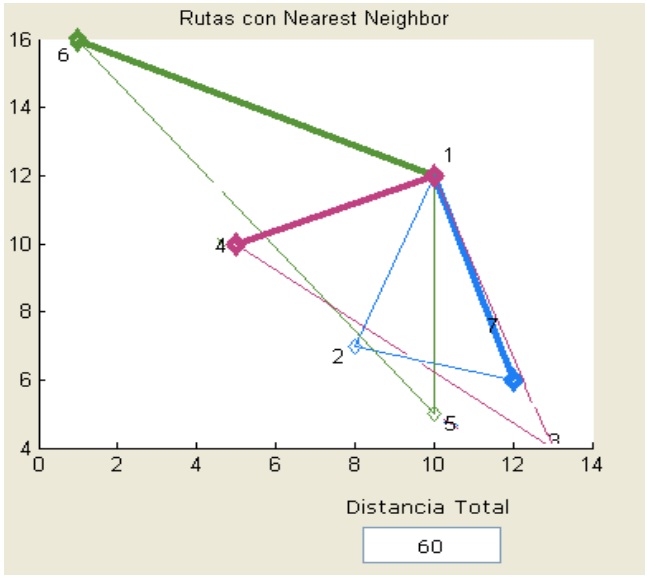
Nuevamente se corre el programa y se llenan los respectivos cálculos. En este caso seleccionamos el algoritmo del Vecino más Cercano. Las figuras 32 y 33 muestran las acciones que se realizaron y la salida.

Figura 35. Parámetros de entrada del vecino más cercano



Fuente: Autores

Figura 36. Resultado arrojado por H-CVRP por el método heurístico del vecino más cercano para el ejemplo 2.



Fuente: Autores

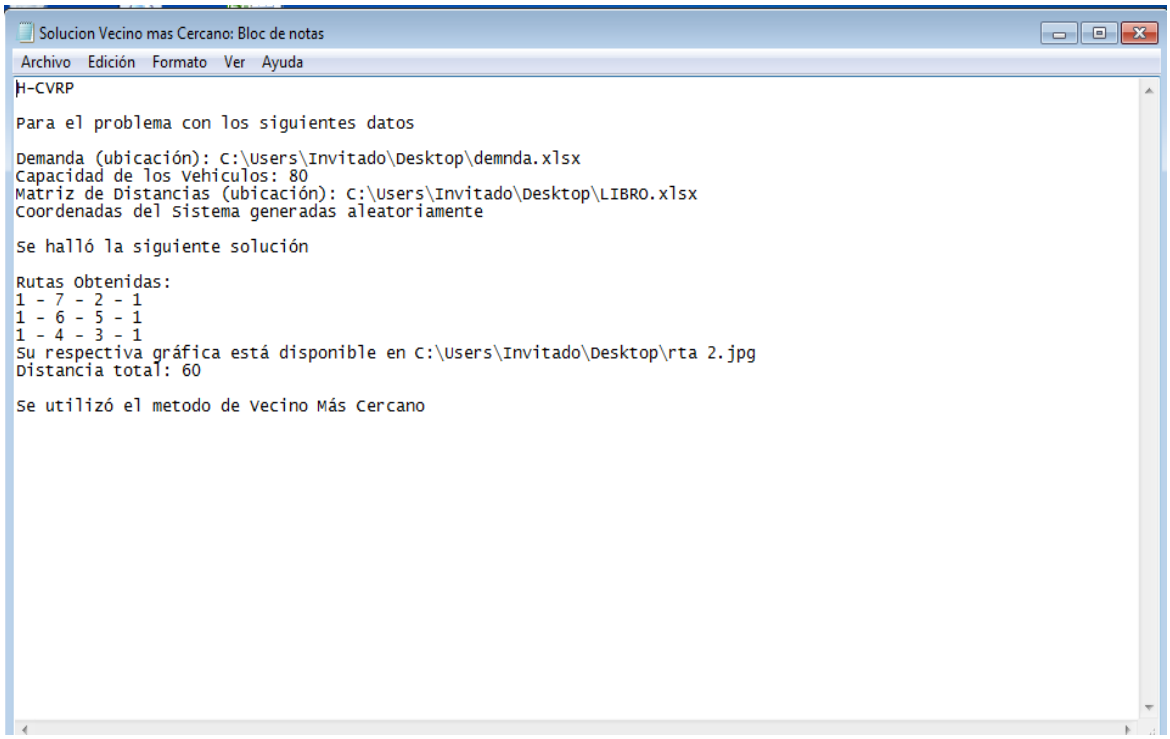
La solución obtenida para este algoritmo tiene un costo total de 60 unidades. Las rutas creadas fueron:

Ruta 1: 1 – 7 – 2 – 1.

Ruta 2: 1 – 6 – 5 – 1.

Ruta 3: 1 – 4 – 3 – 1.

Figura 37. Archivo de salida .txt del software H-CVRP usando el algoritmo del vecino mas cercano



Fuente: Autores

4.3. Comparación de resultados obtenidos con H-CVRP vs. Instancias de la literatura para el CVRP

4.3.1. Comparación de los resultados obtenidos con H-CVRP para el algoritmo de Clarke and Wright Vs. Instancias de la literatura

A continuación se muestra el error entre los resultados para distintas instancias del CVRP encontradas en la literatura. Estas librerías comparadas corresponden a los cálculos realizados por Christofides y Eilon.

La salida de H-CVRP para cada una de las instancias que se corrieron, se muestran en la sección de anexos.

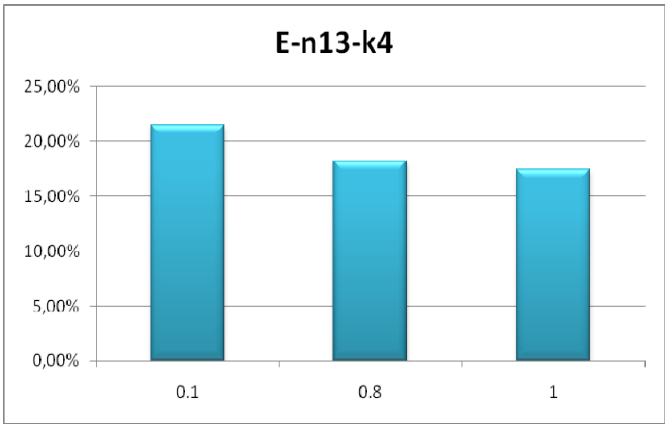
Tabla 54. Comparación del resultado obtenido por H-CVRP para la instancia E-n13-k4

Instancia	# Nodos	λ	Sln. H-CVRP	Optimo	Error
E-n13-k4	13	0.1	300	247	21,46%
		0.8	292		18,22%
		1	290		17,41%

Fuente: Autores

La mejor solución obtenida para la instancia E-n13-k4 fue usando un valor del parámetro lambda igual a 1.

Figura 38. Análisis del error para los diferentes valores de lambda en la instancia E-n13-k4



Fuente: Autores

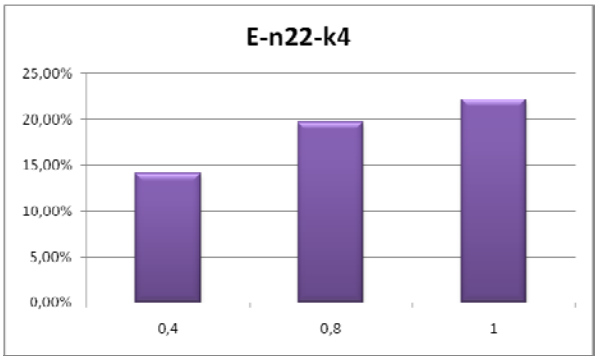
Tabla 55. Comparación del resultado obtenido por H-CVRP para la instancia E-n22-k4

Instancia	# Nodos	λ	Sln. H-CVRP	Optimo	Error
E-n22-k4	22	0.4	428	375	14.13%
		0.8	449		19.73%
		1	458		22.13%

Fuente: Autores

La mejor solución obtenida para la instancia E-n22-k4 fue usando un valor del parámetro lambda igual a 0.4.

Figura 39. Análisis del error para los diferentes valores de lambda en la instancia E-n22-k4



Fuente: Autores

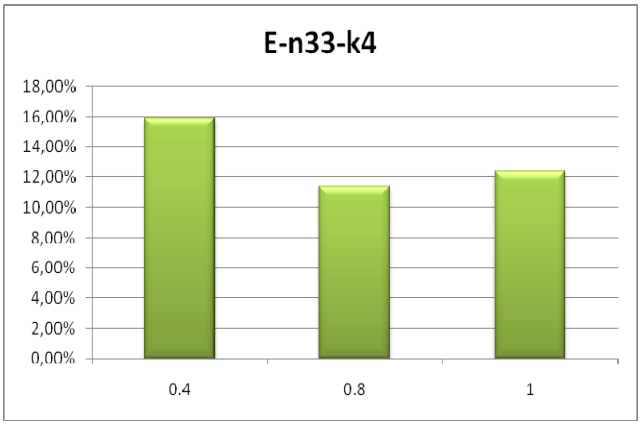
Tabla 56. Comparación del resultado obtenido por H-CVRP para la instancia E-n33-k4

Instancia	# Nodos	λ	Sln. H-CVRP	Optimo	Error
E-n33-k4	33	0.4	968	835	15.9%
		0.8	930		11.37%
		1	938		12.33%

Fuente: Autores

La mejor solución obtenida para la instancia E-n33-k4 fue usando un valor del parámetro lambda igual a 0.8.

Figura 40. Análisis del error para los diferentes valores de lambda en la instancia E-n33-k4



Fuente: Autores

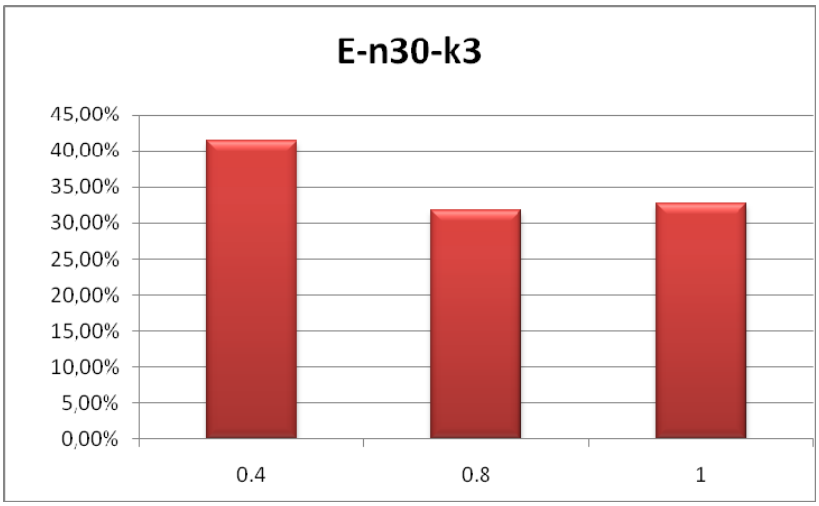
Tabla 57. Comparación del resultado obtenido por H-CVRP para la instancia E-n30-k3

Instancia	# Nodos	λ	Sln. H-CVRP	Optimo	Error
E-n30-k3	30	0.4	755	534	41,39%
		0.8	703		31.64%
		1	708		32.58%

Fuente: Autores

La mejor solución obtenida para la instancia E-n30-k3 fue usando un valor del parámetro lambda igual a 0.8.

Figura 41. Análisis del error para los diferentes valores de lambda en la instancia E-n30-k3



Fuente: Autores

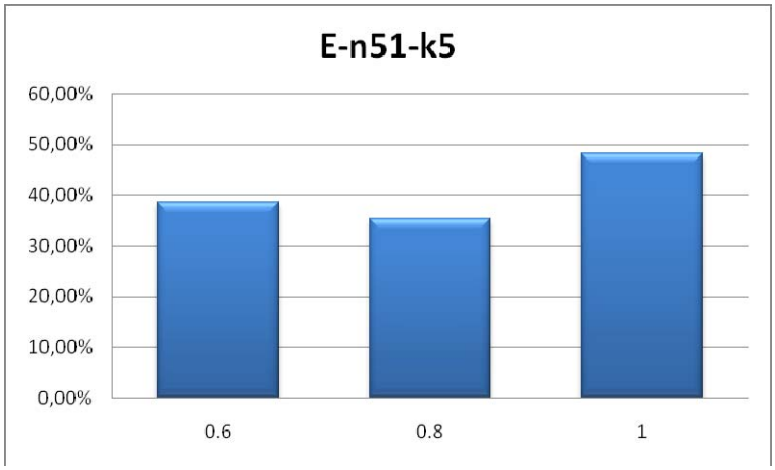
Tabla 58. Comparación del resultado obtenido por H-CVRP para la instancia E-n51-k5

Instancia	# Nodos	λ	Sln. H-CVRP	Optimo	Error
E-n51-k5	51	0.6	722	521	38,58%
		0.8	705		35,32%
		1	772		48,18%

Fuente: Autores

La mejor solución obtenida para la instancia E-n51-k5 fue usando un valor del parámetro lambda igual a 0.8.

Figura 42. Análisis del error para los diferentes valores de lambda en la instancia E-n51-k5



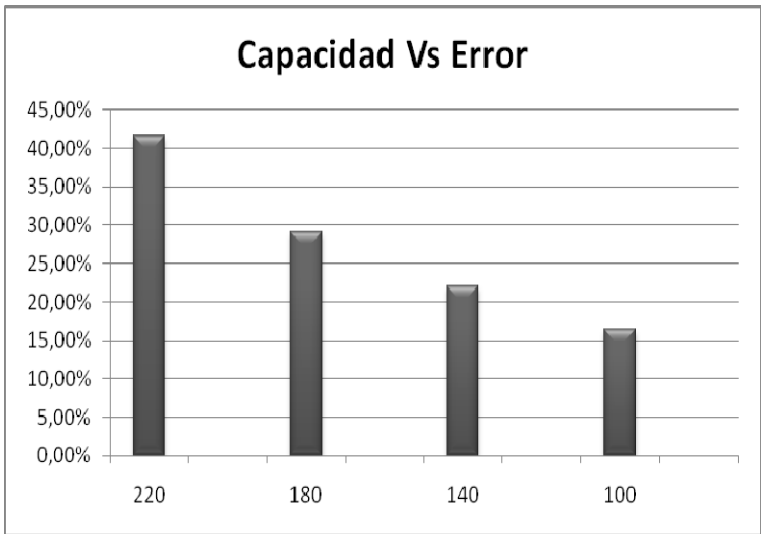
Fuente: Autores

Tabla 59. Resultados obtenidos por H-CVRP para la instancia E-n76

Instancia	# Nodos	λ	Sln. H-CVRP	Optimo	Error	Error Prom.	Capacidad
E-n76-k7	76	0,9	987	682	44,72%	41,57%	220
		1	944		38,42%		
E-n76-k8		0,9	931	735	26,67%	29,12%	180
		1	967		31,56%		
E-n76-k10		0,9	1023	830	23,25%	21,99%	140
		1	1002		20,72%		
E-n76-k14		0.8	1204	1021	17,92%	16,45%	100
		1	1174		14,99%		

Fuente: Autores

Figura 43. Análisis del error vs la capacidad en la instancia E-n76



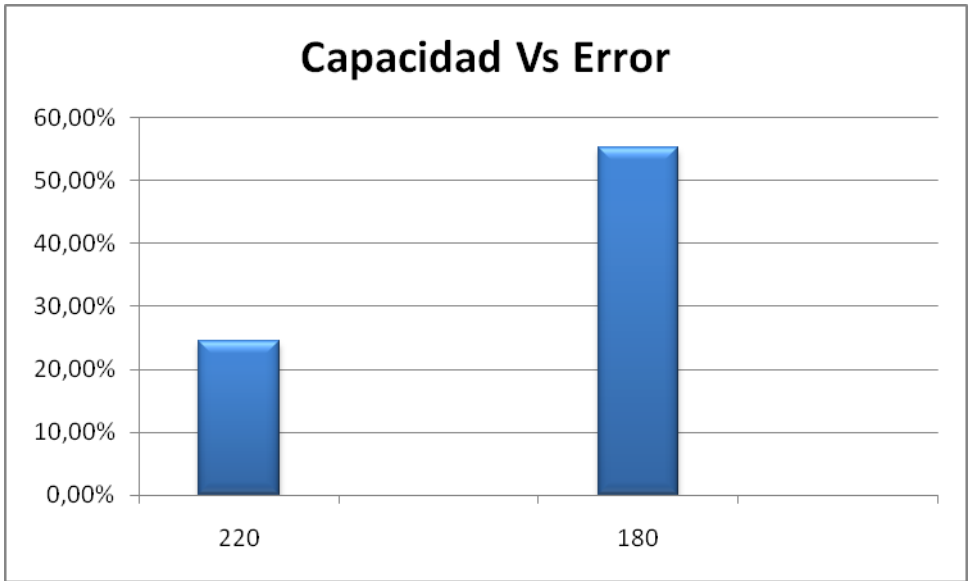
Fuente: Autores

Tabla 60 Resultados obtenidos por H-CVRP para la instancia E-n101

Instancia	# Nodos	λ	Sln. H-CVRP	Optimo	Error	Error Prom.	Capacidad
E-n101-k14	101	0,8	1322	1071	23,44%	24,51%	220
		1	1345		25,58%		
E-n101-k8		0,9	1264	817	54,71%	55,32%	180
		1	1274		55,94%		

Fuente: Autores

Figura 44. Análisis del error vs la capacidad en la instancia E-n101



Fuente: Autores

4.3.2. Comparación de los resultados obtenidos con H-CVRP para el algoritmo del vecino más cercano Vs. Instancias de la literatura

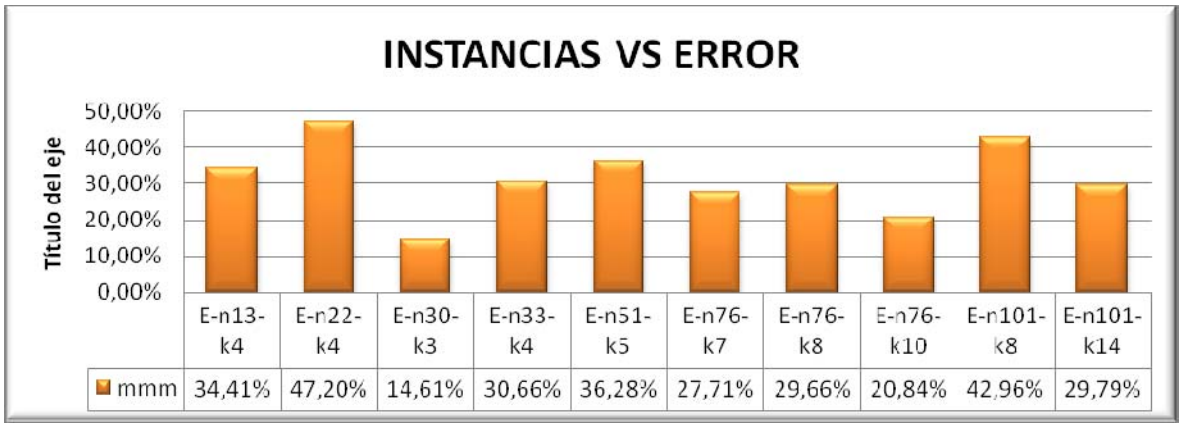
Tabla 61. Comparación del resultado obtenido por H-CVRP usando el método heurístico del vecino más cercano para las distintas instancias

Instancia	# Nodos	Sln. H-CVRP	Optimo	Error
E-n13-k4	13	332	247	34,41%
E-n22-k4	22	552	375	47,20%
E-n30-k3	33	612	534	14,61%
E-n33-k4	30	1091	835	30,66%
E-n51-k5	51	710	521	36,28%
E-n76-k7	76	871	682	27,71%
E-n76-k8	76	953	735	29,66%
E-n76-k10	76	1003	830	20,84%
E-n101-k8	101	1168	817	42,96%
E-n101-k14	101	1390	1071	29,79%

Fuente: Autores

El menor error generado por la solución obtenida usando el programa H-CVRP fue en la corrida de la instancia E-n30-k3.

Figura 45. Comparación de los resultados obtenidos por H-CVRP usando el método heurístico del vecino más cercano vs las instancias de la literatura.



Fuente: Autores

5. CONCLUSIONES Y OBSERVACIONES

Los resultados obtenidos en este trabajo de investigación son la solución al problema de ruteo de vehículos con capacidad a través de una herramienta de software llamada H-CVRP que utiliza dos métodos heurísticos para hallar las rutas a asignar a la flota de vehículos.

El primer método implementado en el programa fue el vecino más cercano con el cual se corrieron todas las instancias planteadas por Christofides. Los errores obtenidos al comparar el valor optimo del problema versus la solución arrojada por H-CVRP, oscilan entre el 14.61% y el 47.2%.

El segundo método implementado en el programa de computadora fue el algoritmo de ahorros de Clarke And Wright con la mejora, la cual fue propuesta por Gaskell y Yellow y que consiste en adicionar a la fórmula del cálculo de los ahorros, un parámetro lambda llamado parámetro de forma de la ruta que evita la formación de rutas circulares. Los resultados a las diferentes instancias que se obtuvieron mejoraron en la mayoría de los casos cuando lambda toma valores de 0.4 y 0.8, sin embargo en algunos casos, el mejor resultado se obtiene al ingresar un lambda igual a 1 que equivale a ejecutar el programa con el algoritmo de Clarke and Wright sin la mejora.

Los errores que se obtuvieron de la comparación de los valores óptimos de las instancias de Christofides versus los valores arrojados por H-CVRP para el algoritmo de ahorros oscilan entre el 11.37% y 55.94%.

Comparando los resultados obtenidos por los dos algoritmos para las mismas instancias se puede concluir que en la mayoría de los casos el algoritmo de ahorros llega a mejores soluciones que el algoritmo del vecino más cercano.

Teniendo en cuenta que la forma de búsqueda del vecino más cercano consiste en inspeccionar en la vecindad de cada nodo y que las heurísticas constructivas buscan en cada iteración una mejor solución, puede ocurrir que las soluciones encontradas sean solo óptimos locales. Por esta razón los errores encontrados al realizar comparaciones con los valores óptimos de las instancias son significativos. Sin embargo, se debe aclarar que las heurísticas debido a que son procedimientos con un alto grado de confianza, en algunas ocasiones encuentran el valor óptimo de la función objetivo, aunque este no pueda llegar a corroborarse.

Estas heurísticas pueden ser usadas para calcular las soluciones iniciales de métodos metaheurísticos.

BIBLIOGRAFIA

- İ K ALTINEL AND T ÖNCAN, “A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem”, *Journal of the Operational Research Society* 56, 954-961 (August 2005).

- CHRISTOFIDES N AND EILON S (June, 1969).” An algorithm for the vehicle dispatching problem”. Operational Research Quartely 20:309-318.

- GASKELL TJ (1967). “Bases for vehicle fleet scheduling”. Operational Research Quartely 18: 281-295.

- G. CLARKE AND J. WRIGHT "Scheduling of vehicles from a central depot to a number of delivery points", Operations Research, 12 #4, 568-581, 1964.

- DIAZ, A., GLOVER, F., GHAZIRI, H.M., et al, “Optimización Heurística y Redes Neuronales”. Madrid, Paraninfo, (1996).

- VIGO, DANIELE, TOTH, PAOLO, “The Vehicle Routing Problem”, Siam (2002).

- GLOVER, Fred. “Tabu Search – Part I”, ORSA Journal on computing, v. 1, n. 3, pp. 190-206. Summer 1989.

- LAPORTE G. GENDREAU M., POTVIN J-Y., SEMET F., “Classical and Modern Heuristics for the vehicle routing problem”, International Transaction in Operational Research, vol. 7, pp. 285-300, 2000.

- HOLLY MOORE, “Matlab para ingenieros”, Pearson Prentice Hall, Mexico, 1ª edición, 2007.

- G. LAPORTE, Y. NOBERT AND M. DESROCHERS (1985): “Optimal Routing with Capacity and Distance Restrictions”. Operations Research 33, 1050–1073.

- BRUCE GOLDEN, S. RAGHAVAN, EDWARD WASIL “The Vehicle Routing Problem: Latest Advances and New Challenges" Springer, 2008.

- ERNESTO LEVEL, ALBIN PALHAZI, Vehicle Routing Problem, “Implementación de Meta-Heurísticas”. Diseño de algoritmos II, Departamento de Computación y Tecnología de la Información. Universidad Simón Bolívar. 2009.

- ALFREDO OLIVERA, “Heurísticas para el problema de ruteo de vehículos”, Instituto de computación, Facultad de Ingeniería. Universidad de la República, Montevideo, Uruguay.

- HOSPITALER A. (P), JAÉN P., SANTAMARINA C., MONTALVÁ J.M. “Algoritmo Híbrido Basado En Colonias de Hormigas para la Solución del Problema de la Distribución en Planta en el Marco del S.L.P”.

- MELIÁN, BELÉN. PEREZ, JOSE A. et al.”Metaheurísticas: una visión global”. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. N.19 pp. 7-28 ISSN: 1137-3601. © AEPIA (2003). http://www.aepia.org/re

- FISCHETTI, MATTEO; TOTH, PAOLO Y FRANCESCHI, ROBERTO DE. “A new heuristic algorithm for the vehicle routing problem” Enero 2004.

- BENJAMÍN BARÁN Y MARTA ALMIRÓN. “Colonia de Hormigas en un Ambiente Paralelo Asíncrono”. Universidad Nacional de Asunción Centro Nacional de Computación San Lorenzo, Paraguay.

- MOCHOLI ARCE, MANUEL, Mocholi, R Sala Garrido Colaborador R Sala Garrido. Programación lineal: Metodología y problemas. Editorial Tebar, 1993. Pág. 175.

- JAMES R. EVANS, JAMES ROBERT EVANS, EDWARD MINIEKA. “Optimization Algorithms for Networks and Graphs”. Second edition, rev. and expanded.

- MIKHAIL J. ATALLAH, MARINA BLANTON. “Algorithms and theory of computation handbook”. Second edition. Ed. Taylor and Francis Group.

- ADEYINKA J. IDOWU, “The application of Clarke-Wright savings algorithm in vehicle routing”. Published 1987**.** Edition Notes

- FERNANDO SANDOYA SÁNCHEZ, Escuela superior Politécnica Del litoral. “Métodos Exactos y Heurísticos para resolver el problema del agente viajero y el problema de ruteo de vehículos” Guayaquil - Ecuador oct. 2007.

- MARTÍ, RAFAEL. Procedimientos Metaheurísticos en Optimización Combinatoria.

- M. GENDREAU, F. GUERTIN, J.-Y. POTVIN, AND R. SEGUIN. “Neighborhood search heuristic for a dynamic vehicle dispatching problem

with pick-ups and deliveries". Technical Report CRT-98-10, Centre de recherche sur les transports, Universite de Montreal, Canada, 1998.

ANEXOS

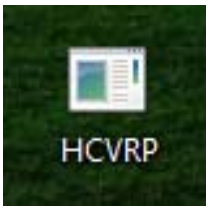
ANEXO 1. MANUAL DEL USUARIO H-CVRP

Este anexo tiene como finalidad mostrar al lector cómo resolver en H-CVRP las instancias planteadas en el trabajo de grado. Se explica en detalle el manejo del software.

PASOS PARA LA IMPLEMENTACIÓN DEL H-CVRP

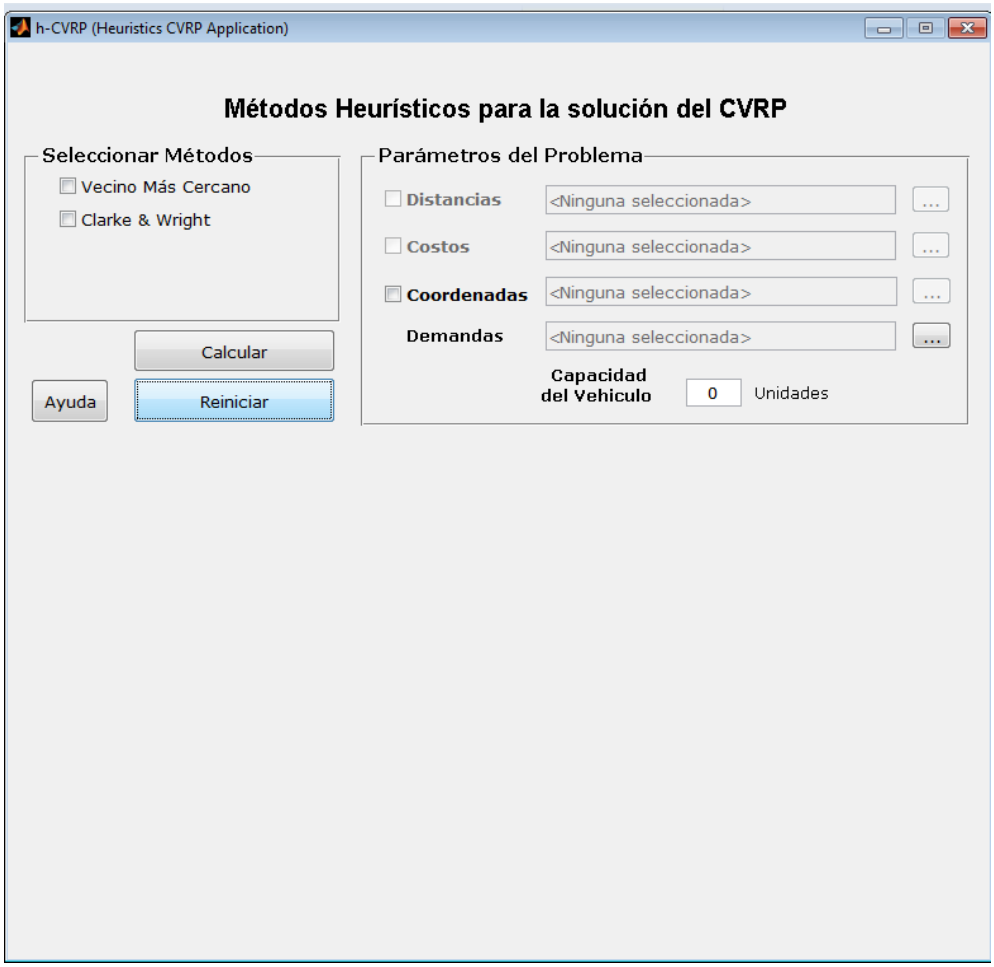
Primero: Para empezar, se debe instalar en el equipo el software de MATLAB 2010, el cual es un programa de alta calidad que integra programación y visualización.

Luego de la correcta instalación se da clic en el icono ubicado en el escritorio:

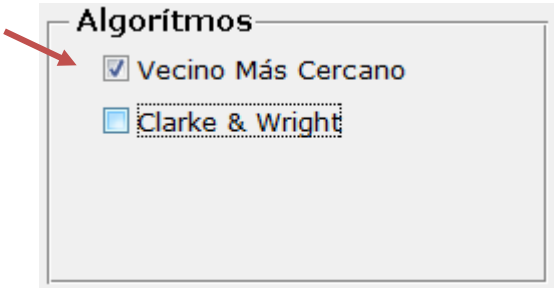


Segundo: Al ejecutar el programa H-CVRP se despliega una única ventana principal, donde se muestran los campos que deben ser cumplidos.

A continuación se pasan a describir las opciones que se muestran en la siguiente figura.

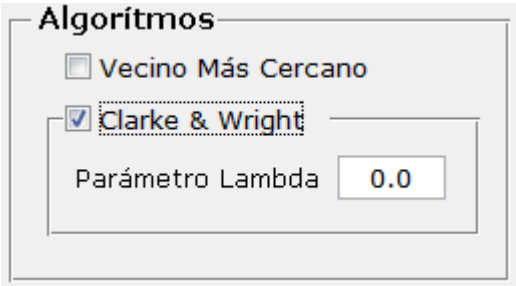


Tercero: Para ingresar un problema en el H-CVRP, se debe escoger el algoritmo a usar para la solución del problema, ya sea uno ó los dos. Se señala el algoritmo dando un clic sobre el recuadro ubicado al lado del nombre de cada uno de ellos.

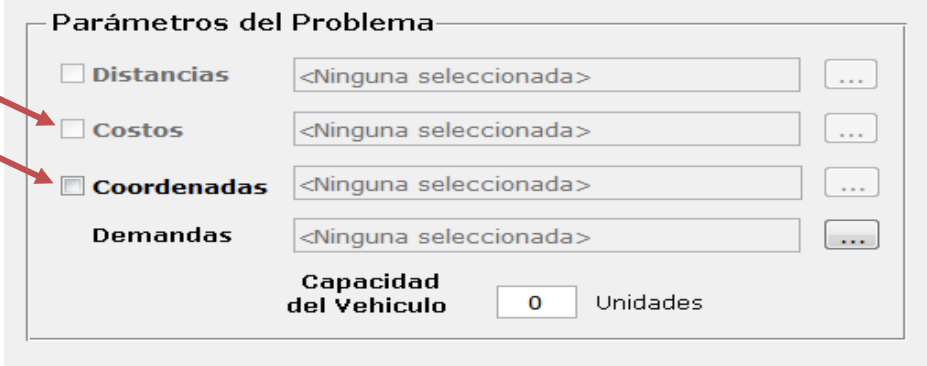


En caso de escoger el algoritmo de ahorros CLARKE AND WRIGHT, inmediatamente aparece un recuadro donde se pide el parámetro lambda, o forma

de la ruta, el cual es una mejora implementada para este tipo de algoritmo que evita la formación de rutas circulares en el desarrollo del problema. Este parámetro lo digita el usuario, se debe tener en cuenta que según Golden, Magnanti y Nguyen los mejores valores de lambda para la solución de cualquier ejercicio son 0.4 y 1.[9]



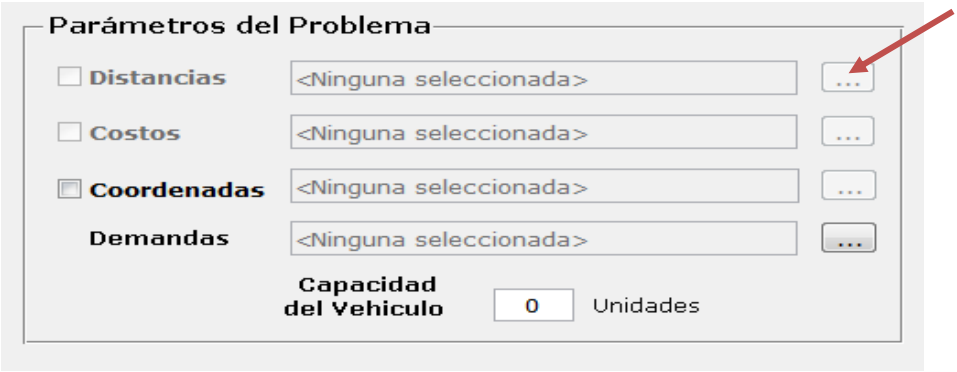
Cuarto: En los datos de entrada además de escoger el tipo de algoritmo, especificar la capacidad y las demandas de los clientes, también se puede escoger entre ingresar la matriz de costos ó ingresar las coordenadas de los clientes para solucionar el problema.



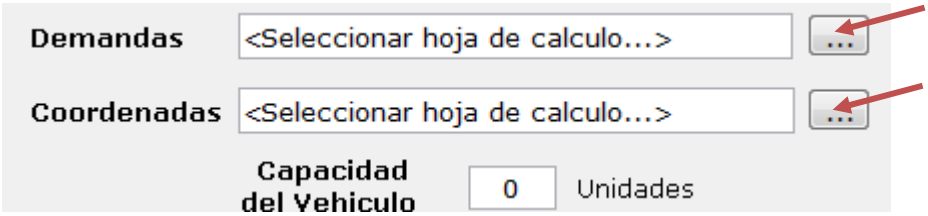
Seleccionado el algoritmo se define si se quiere trabajar con las coordenadas del problema, o con la matriz de costos para calcular la solución. Estas matrices se introducen señalando el recuadro, ya sea de costos ó de coordenadas ubicado al

[9] M. Gendreau, F. Guertin, J.-Y. Potvin, and R. Seguin. Neighborhood search heuristic for a dynamic vehicle dispatching problem with pick-ups and deliveries. Technical Report CRT-98-10, Centre de recherche sur les transports, Universite de Montreal, Canada, 1998.

lado de cada término. Se ubica el archivo, donde se tiene almacenada la matriz con el botón de los tres puntos ubicado a la izquierda del término. El archivo como se explicó anteriormente debe ser un archivo .xls.

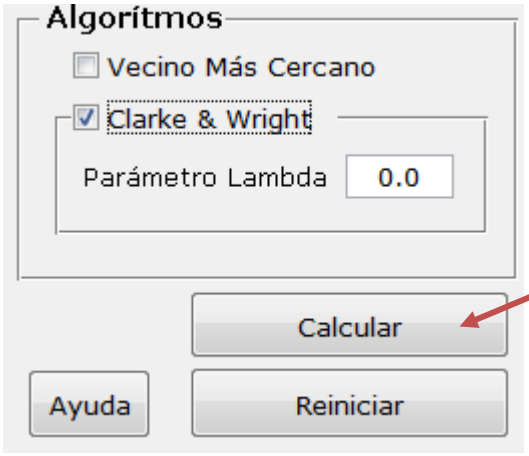


Quinto: Finalizada esta secuencia, se procede a ingresar los datos necesarios para que el programa empiece a ejecutarse, los cuales son:

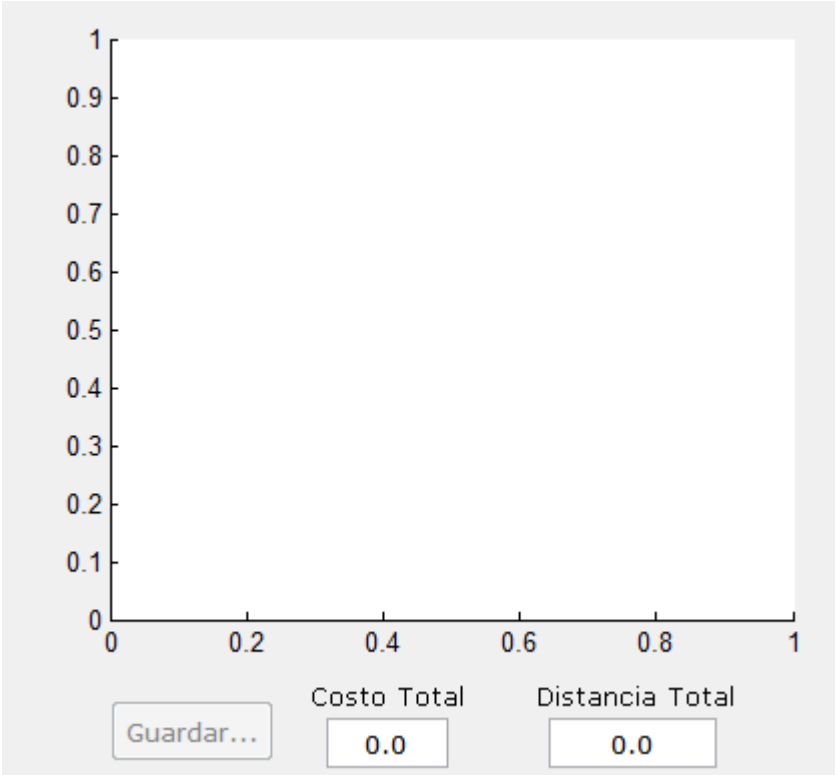


Las demandas, junto con las coordenadas deben estar guardadas cada una en un archivo diferente, de formato .xls (Excel) para que el programa pueda identificarlos y ejecutarlos. La capacidad del vehículo se digita directamente en la ventana principal.

Sexto: En el momento en el que ya se han registrado todos los datos de entrada se hace clic en el botón calcular:

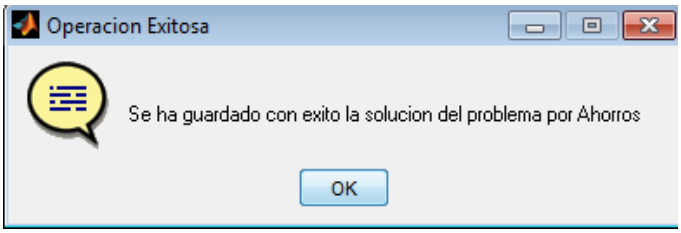


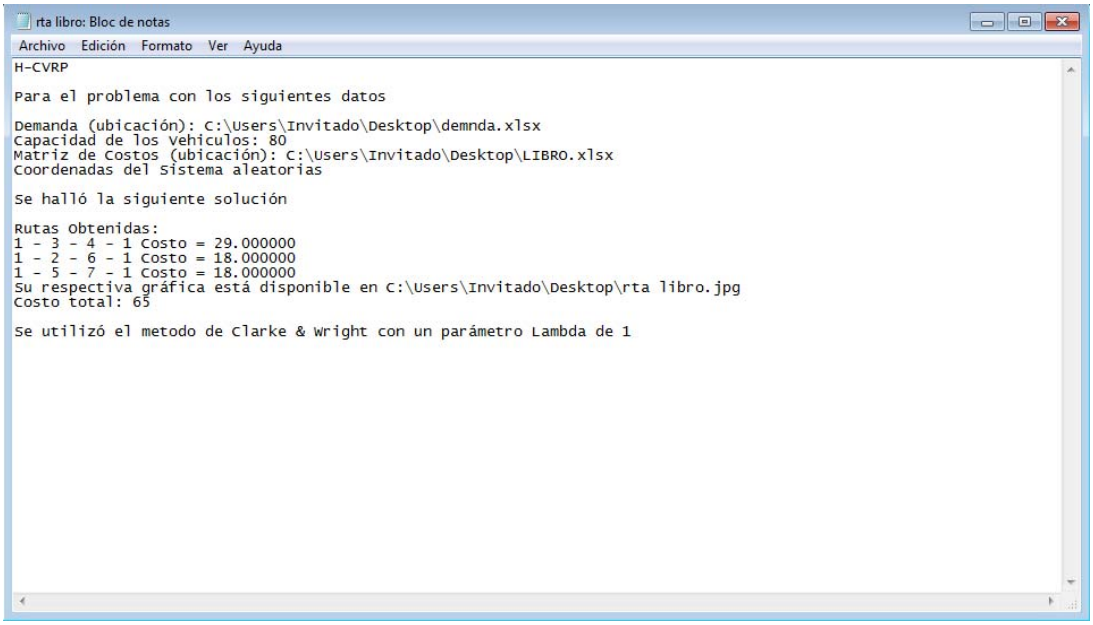
Inmediatamente el programa arroja el gráfico de solución junto con el costo asociado a este ejercicio:



Séptimo: Para terminar, se hace clic en el botón guardar, donde quedan almacenadas las rutas, la matriz respuesta, y el costo final asociado al problema.

El gráfico queda guardado con formato .jpg (de imagen). Este botón guardar, además de almacenar los cálculos incluye también la ubicación de los archivos de entrada y salida correspondientes al ejercicio.

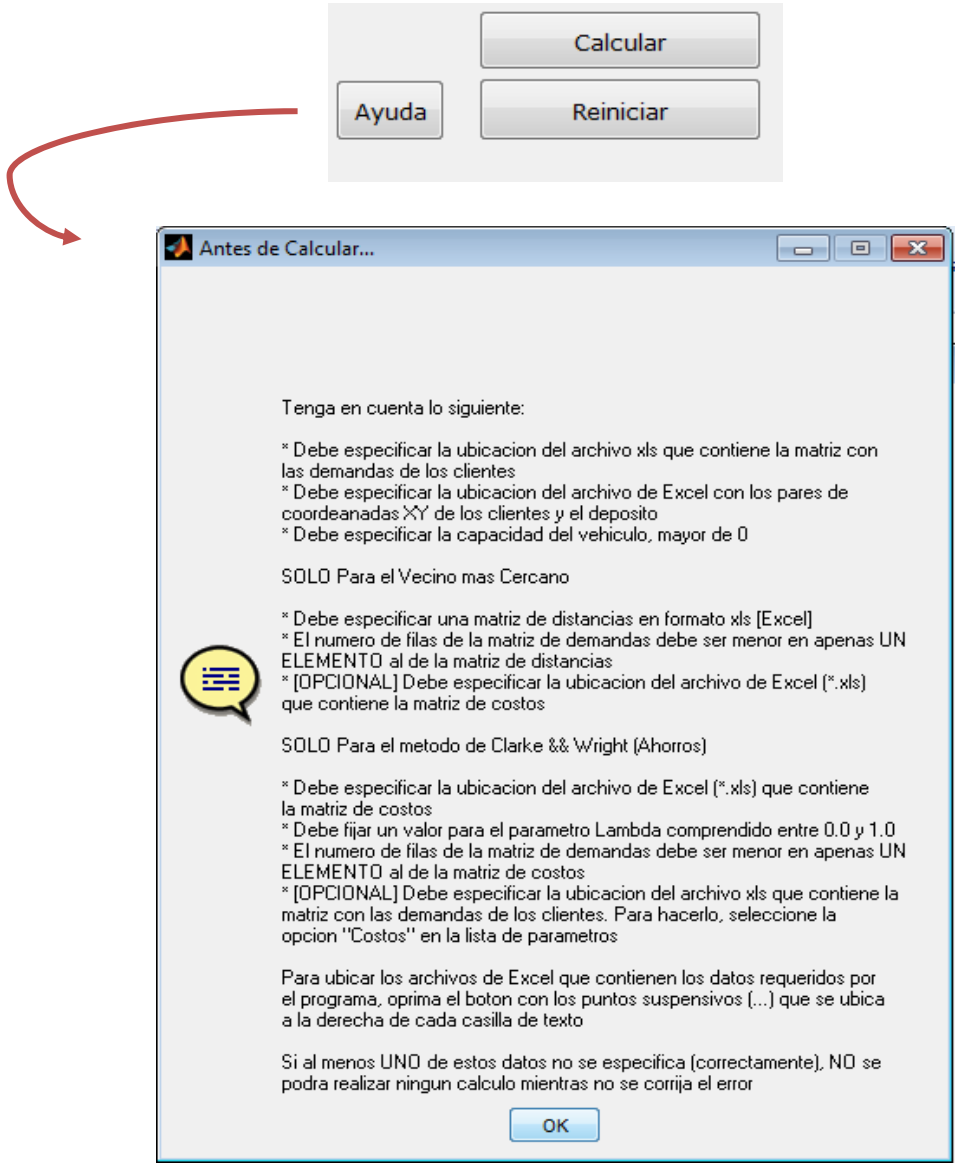




Octavo: El gráfico respuesta comienza su recorrido por la línea de unión que presenta mayor grosor de esta manera el usuario identifica fácilmente el sentido que lleva la ruta. Los colores arrojados en la gráfica respuesta son aleatorios, por lo que se aconseja que cuando no sea muy clara la gráfica dada por el programa, se oprima nuevamente el botón calcular para que el software arroje otra gráfica con diferentes colores y el usuario pueda diferenciar las rutas solución que se han elaborado.

Noveno: En caso de tener otro problema para solucionar se escoge la opción reiniciar, para volver a ingresar los nuevos datos de entrada y hallar la nueva solución.

Décimo: En el momento en que el usuario, requiera una guía del manejo del programa durante su ejecución, en cualquier momento puede dar clic en el botón de ayuda, ubicado junto al botón de calcular, para que lea las instrucciones de uso y manejo del H-CVRP, donde encontrara paso a paso las indicaciones a seguir para su uso.



NOTA: Los archivos guardados en Excel, parten de un depósito que se guarda con el numero 1, y los clientes se inician en 2, se tendría entonces como clientes totales = n-1. Donde n, es el número total de nodos que se tienen en el archivo.

RECOMENDACIÓN: Se aconseja que todos los archivos sean guardados en la misma carpeta para su fácil ubicación.

ANEXO 2. DATOS DE ENTRADA DE H-CVRP PARA LA SOLUCIÓN DE LA INSTANCIA E-n101

Tabla .1 Datos de entrada de H-CVRP para la solución de la instancia E-n7101

NODOS	COORDENADAS		DEMANDAS
1	35	35	-
2	41	49	10
3	35	17	7
4	55	45	13
5	55	20	19
6	15	30	26
7	25	30	3
8	20	50	5
9	10	43	9
10	55	60	16
11	30	60	16
12	20	65	12
13	50	35	19
14	30	25	23
15	15	10	20
16	30	5	8
17	10	20	19
18	5	30	2
19	20	40	12
20	15	60	17
21	45	65	9
22	45	20	11
23	45	10	18
24	55	5	29
25	65	35	3

NODOS	COORDENADAS		DEMANDAS
26	65	20	6
27	46	30	17
28	35	40	16
29	41	37	16
30	64	42	9
31	40	60	21
32	31	52	27
33	35	69	23
34	53	52	11
35	65	55	14
36	63	65	8
37	2	60	5
38	20	20	8
39	5	5	16
40	60	12	31
41	40	25	9
42	42	7	5
43	24	12	5
44	23	3	7
45	11	14	18
46	6	38	16
47	2	48	1
48	8	56	27
49	13	52	36
50	6	68	30

NODOS	COORDENADAS		DEMANDAS
51	47	47	13
52	49	58	10
53	27	43	19
54	37	31	22
55	57	29	16
56	63	23	7
57	53	12	26
58	32	12	14
59	36	26	21
60	21	24	24
61	17	34	13
62	12	24	15
63	24	58	18
64	27	69	11
65	15	77	28
66	62	77	9
67	49	73	37
68	67	5	30
69	56	39	10
70	37	47	8
71	37	56	11
72	57	68	3
73	47	16	1
74	44	17	6
75	46	13	10

NODOS	COORDENADAS		DEMANDAS
76	49	11	18
77	49	42	13
78	53	43	14
79	61	52	3
80	57	48	23
81	56	37	6
82	55	54	26
83	15	47	16
84	14	37	11
85	11	1	7
86	16	22	41
87	4	18	35
88	28	18	26
89	26	52	9
90	26	35	15
91	31	67	3
92	15	19	1
93	22	22	2
94	18	24	22
95	26	27	27
96	25	24	20
97	22	27	11
98	25	21	12
99	19	21	10
100	20	26	9
101	18	18	17

Fuente: Autores

ANEXO 3. DATOS DE ENTRADA DE H-CVRP PARA LA SOLUCIÓN DE LA INSTANCIA E-n13-K4

Tabla 1. Datos de entrada de H-CVRP para la solución de la instancia E-n13-k4

NODOS	COORDENADAS		DEMANDAS
1	145	215	-
2	151	264	1200
3	159	261	1700
4	130	254	1500
5	128	252	1400
6	163	247	1700
7	146	246	1400
8	161	242	1200
9	142	239	1900
10	163	236	1800
11	148	232	1600
12	128	231	1700
13	156	217	1100

Fuente. Autores

Tabla 2. Matriz Distancia de entrada a H-CVRP para la solución de la instancia E-n13-k4

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	9	14	21	23	22	25	32	36	38	42	50	52
2	9	0	5	12	22	21	24	31	35	37	41	49	51
3	14	5	0	7	17	16	23	26	30	36	36	44	46
4	21	12	7	0	10	21	30	27	37	43	31	37	39
5	23	22	17	10	0	19	28	25	35	41	29	31	29
6	22	21	16	21	19	0	9	10	16	22	20	28	30
7	25	24	23	30	28	9	0	7	11	13	17	25	27
8	32	31	26	27	25	10	7	0	10	16	10	18	20
9	36	35	30	37	35	16	11	10	0	6	6	14	16
10	38	37	36	43	41	22	13	16	6	0	12	12	20
11	42	41	36	31	29	20	17	10	6	12	0	8	10
12	50	49	44	37	31	28	25	18	14	12	8	0	10
13	52	51	46	39	29	30	27	20	16	20	10	10	0

Fuente. Autores

ANEXO 4. DATOS DE ENTRADA DE H-CVRP PARA LA SOLUCIÓN DE LA INSTANCIA E-n22-K5

Tabla 1. Datos de entrada de H-CVRP para la solución de la instancia E-n22-k5

NODOS	COORDENADAS		DEMANDAS
1	145	215	-
2	151	264	1100
3	159	261	700
4	130	254	800
5	128	252	1400
6	163	247	2100
7	146	246	400
8	161	242	800
9	142	239	100
10	163	236	500
11	148	232	600
12	128	231	1200
13	156	217	1300
14	129	214	1300
15	146	208	300
16	164	208	900
17	141	206	2100
18	147	193	1000
19	164	193	900
20	129	189	2500
21	155	185	1800
22	139	182	700

Fuente: Autores

ANEXO 5. DATOS DE ENTRADA DE H-CVRP PARA LA SOLUCIÓN DE LA INSTANCIA E-n33-k4

Tabla 1. Datos de entrada de H-CVRP para la solución de la instancia E-n33-k4

NODOS	COORDENADAS		DEMANDAS
1	292	495	-
2	298	427	700
3	309	445	400
4	307	464	400
5	336	475	1200
6	320	439	40
7	321	437	80
8	322	437	2000
9	323	433	900
10	324	433	600
11	323	429	750
12	314	435	1500
13	311	442	150
14	304	427	250
15	293	421	1600
16	296	418	450
17	261	384	700
18	297	410	550
19	315	407	650
20	314	406	200
21	321	391	400
22	321	398	300
23	314	394	1300
24	313	378	700
25	304	382	750
26	295	402	1400
27	283	406	4000
28	279	399	600
29	271	401	1000
30	264	414	500
31	277	439	2500
32	290	434	1700
33	319	433	1100

Fuente: Autores

ANEXO 6. DATOS DE ENTRADA DE H-CVRP PARA LA SOLUCIÓN DE LA INSTANCIA E-n30-k3

Tabla 1. Datos de entrada de H-CVRP para la solución de la instancia E-n30-k3

NODOS	COORDENADAS		DEMANDAS
1	162	354	-
2	218	382	300
3	218	358	3100
4	201	370	125
5	214	371	100
6	224	370	200
7	210	382	150
8	104	354	150
9	126	338	450
10	119	340	300
11	129	349	100
12	126	347	950
13	125	346	125
14	116	355	150
15	126	335	150
16	125	355	550
17	119	357	150
18	115	341	100
19	153	351	150
20	175	363	400
21	180	360	300
22	159	331	1500
23	188	357	100
24	152	349	300
25	215	389	500
26	212	394	800
27	188	393	300
28	207	406	100
29	184	410	150
30	207	392	1000

Fuente: Autores

ANEXO 7. DATOS DE ENTRADA DE H-CVRP PARA LA SOLUCIÓN DE LA INSTANCIA E-n51-k5

Tabla 1. Datos de entrada de H-CVRP para la solución de la instancia E-n51-k5

NODOS	COORDENADAS		DEMANDAS
1	30	40	-
2	37	52	7
3	49	49	30
4	52	64	16
5	20	26	9
6	40	30	21
7	21	47	15
8	17	63	19
9	31	62	23
10	52	33	11
11	51	21	5
12	42	41	19
13	31	32	29
14	5	25	23
15	12	42	21
16	36	16	10
17	52	41	15
18	27	23	3
19	17	33	41
20	13	13	9
21	57	58	28
22	62	42	8
23	42	57	8
24	16	57	16
25	8	52	10
26	7	38	28
27	27	68	7
28	30	48	15
29	43	67	14
30	58	48	6
31	58	27	19
32	37	69	11
33	38	46	12
34	46	10	23
35	61	33	26
36	62	63	17
37	63	69	6
38	32	22	9
39	45	35	15
40	59	15	14
41	5	6	7
42	10	17	27
43	21	10	13
44	5	64	11
45	30	15	16
46	39	10	10
47	32	39	5
48	25	32	25
49	25	55	17
50	48	28	18
51	56	37	10

Fuente: Autores

ANEXO 8. Datos de entrada de H-CVRP para la solución de la instancia E-n76

Tabla 1. Datos de entrada de H-CVRP para la solución de la instancia E-n76

NODOS	COORDENADAS		DEMANDAS
1	40	40	-
2	22	22	18
3	36	26	26
4	21	45	11
5	45	35	30
6	55	20	21
7	33	34	19
8	50	50	15
9	55	45	16
10	26	59	29
11	40	66	26
12	55	65	37
13	35	51	16
14	62	35	12
15	62	57	31
16	62	24	8
17	21	36	19
18	33	44	20
19	9	56	13
20	62	48	15
21	66	14	22
22	44	13	28
23	26	13	12
24	11	28	6
25	7	43	27

NODOS	COORDENADAS		DEMANDAS
26	17	64	14
27	41	46	18
28	55	34	1
29	35	16	29
30	52	26	13
31	43	26	22
32	31	76	25
33	22	53	28
34	26	29	27
35	50	40	19
36	55	50	10
37	54	10	12
38	60	15	14
39	47	66	24
40	30	60	16
41	30	50	33
42	12	17	15
43	15	14	11
44	16	19	18
45	21	48	17
46	50	30	21
47	51	42	27
48	50	15	19
49	48	21	20
50	12	38	5

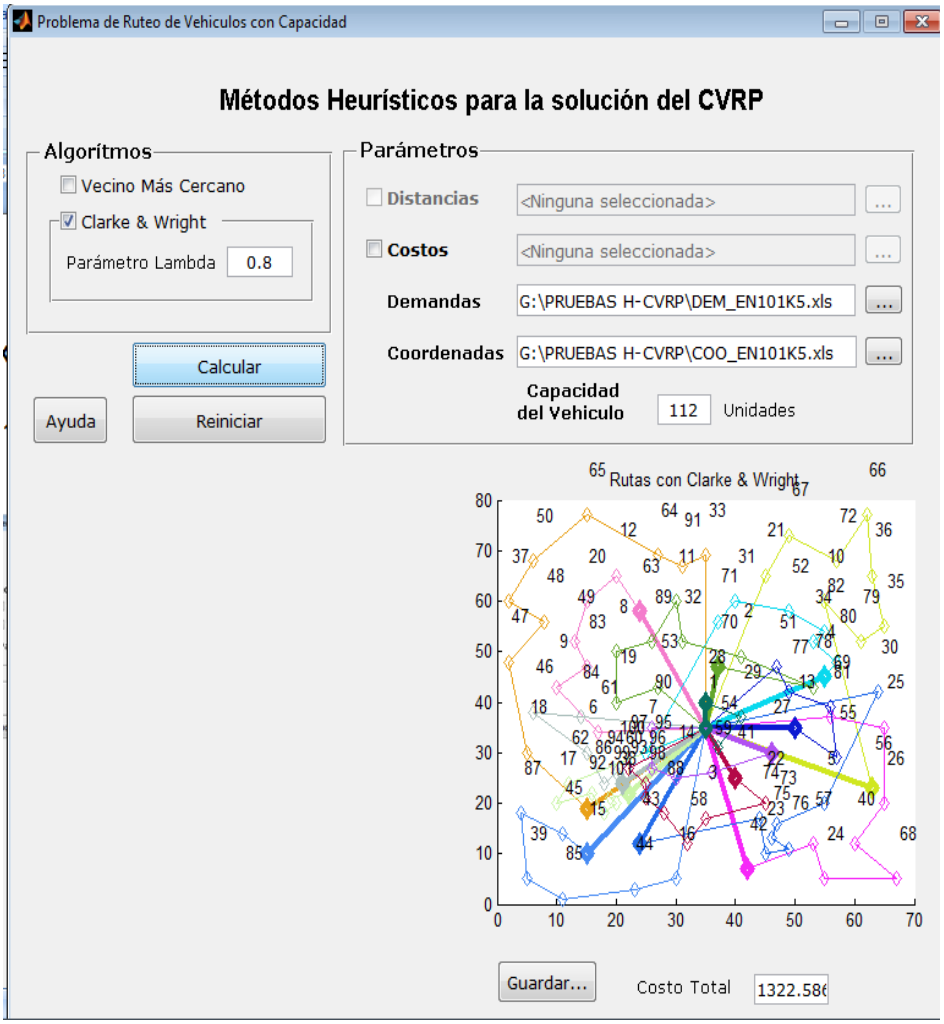
Fuente: Autores

NODOS	COORDENADAS		DEMANDAS
51	15	56	22
52	29	39	12
53	54	38	19
54	55	57	22
55	67	41	16
56	10	70	7
57	6	25	26
58	65	27	14
59	40	60	21
60	70	64	24
61	64	4	13
62	36	6	15
63	30	20	18
64	20	30	11
65	15	5	28
66	50	70	9
67	57	72	37
68	45	42	30
69	38	33	10
70	50	4	8
71	66	8	11
72	59	5	3
73	35	60	1
74	27	24	6
75	40	20	10
76	40	37	20

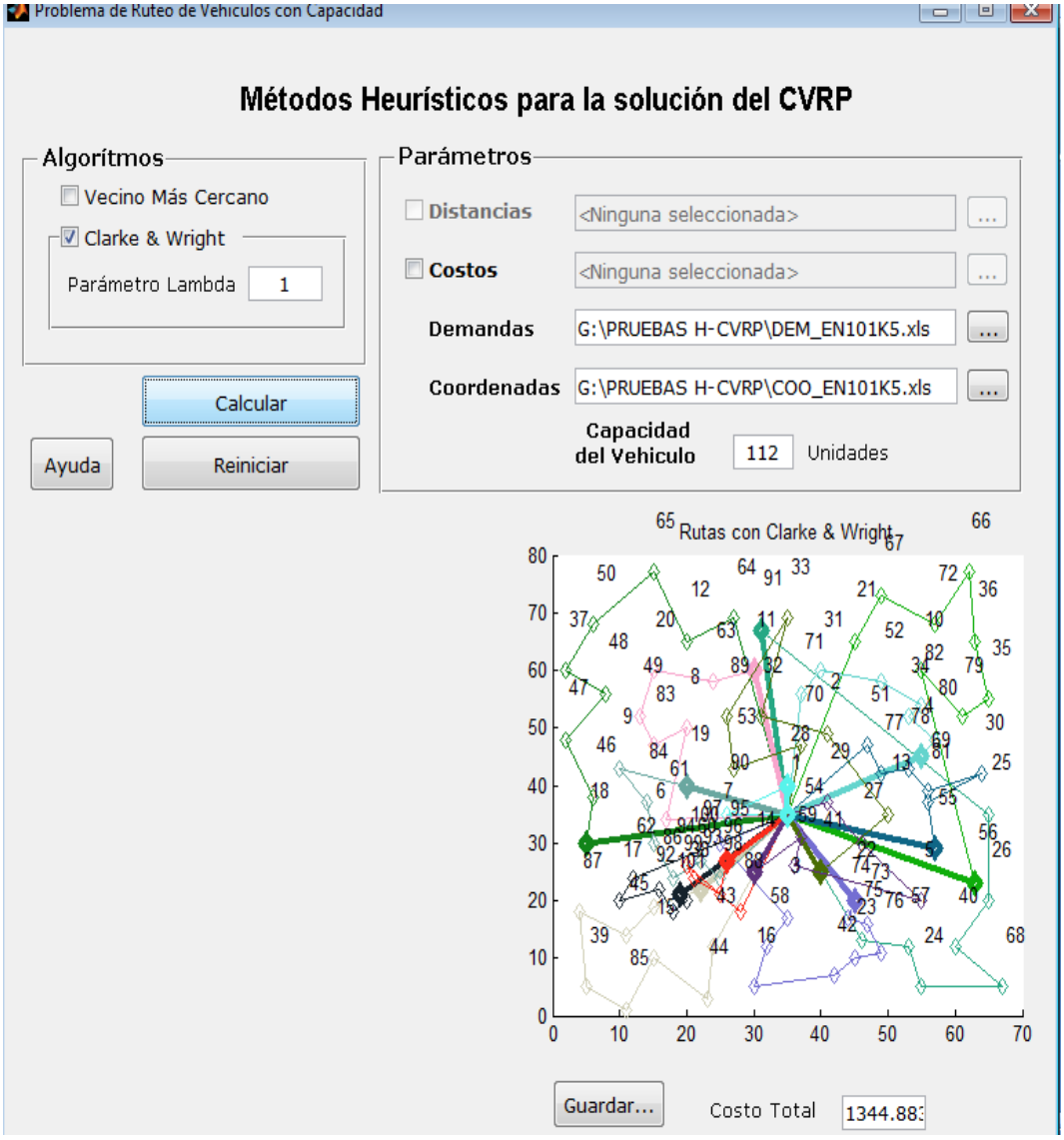
Fuente: Autores

ANEXO 9. SALIDA DE H-CVRP PARA LA PRUEBA DE LA INSTANCIA E-n101-K14

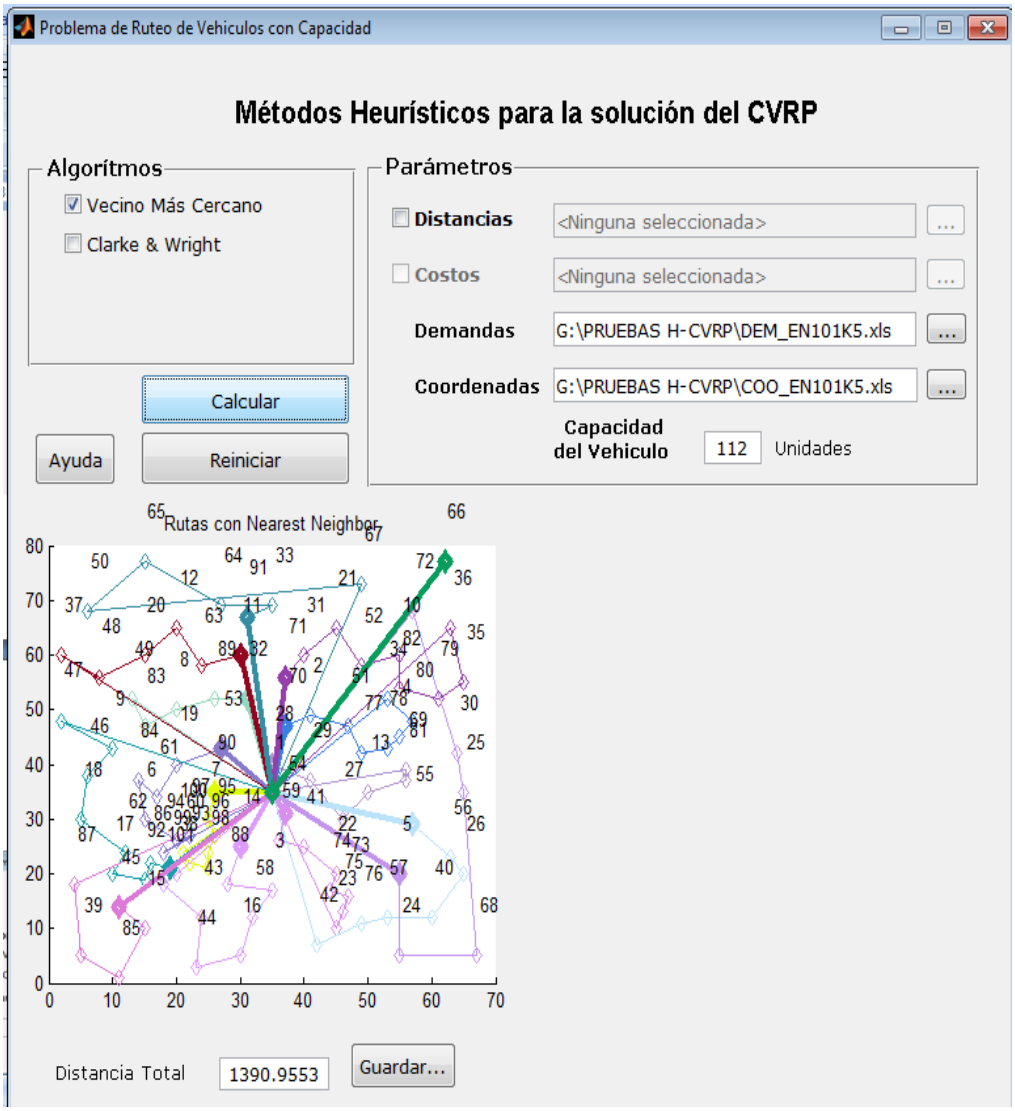
Clarke and Wright λ = 0.8



Clarke and Wright λ = 1



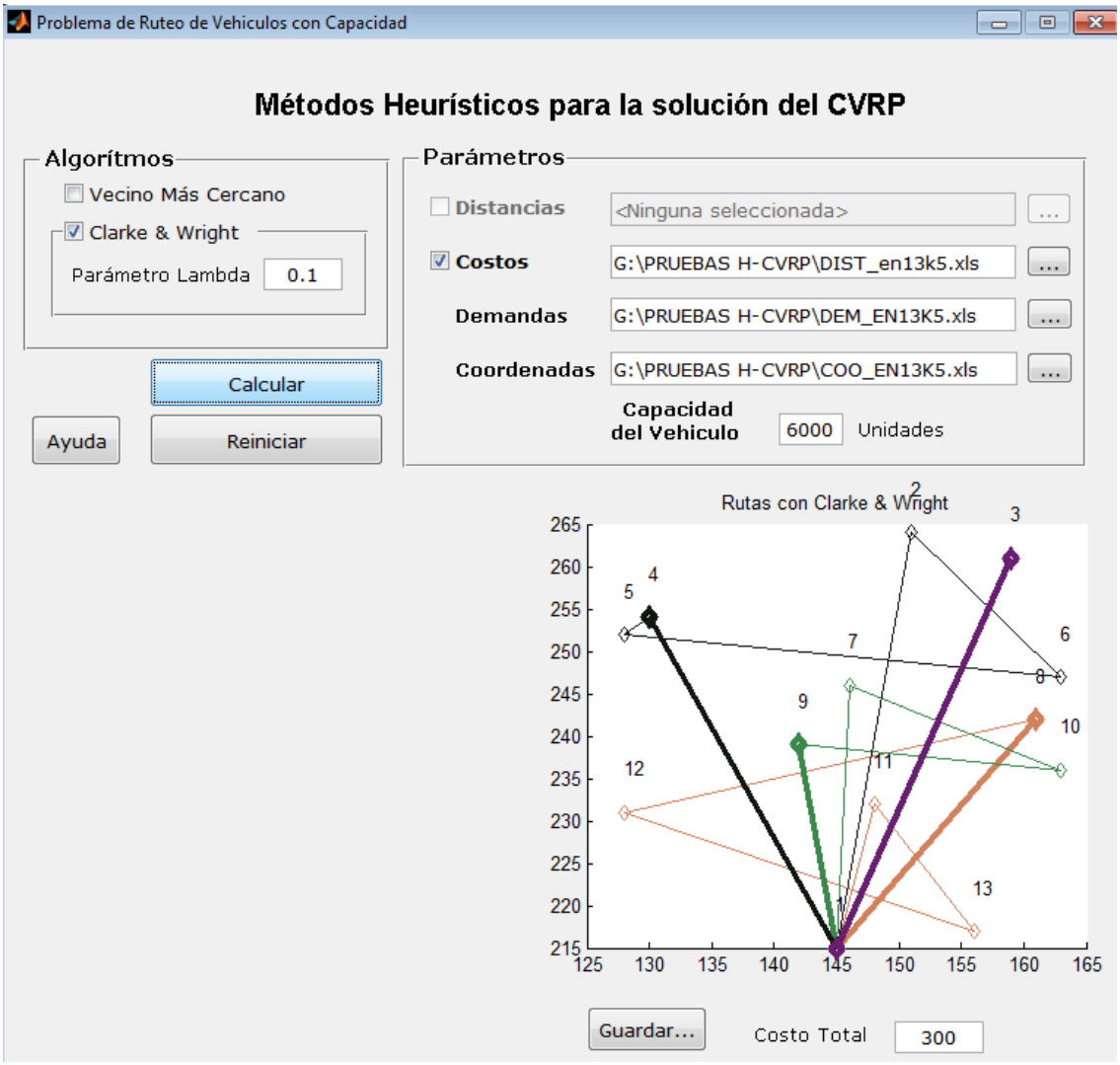
Vecino más cercano



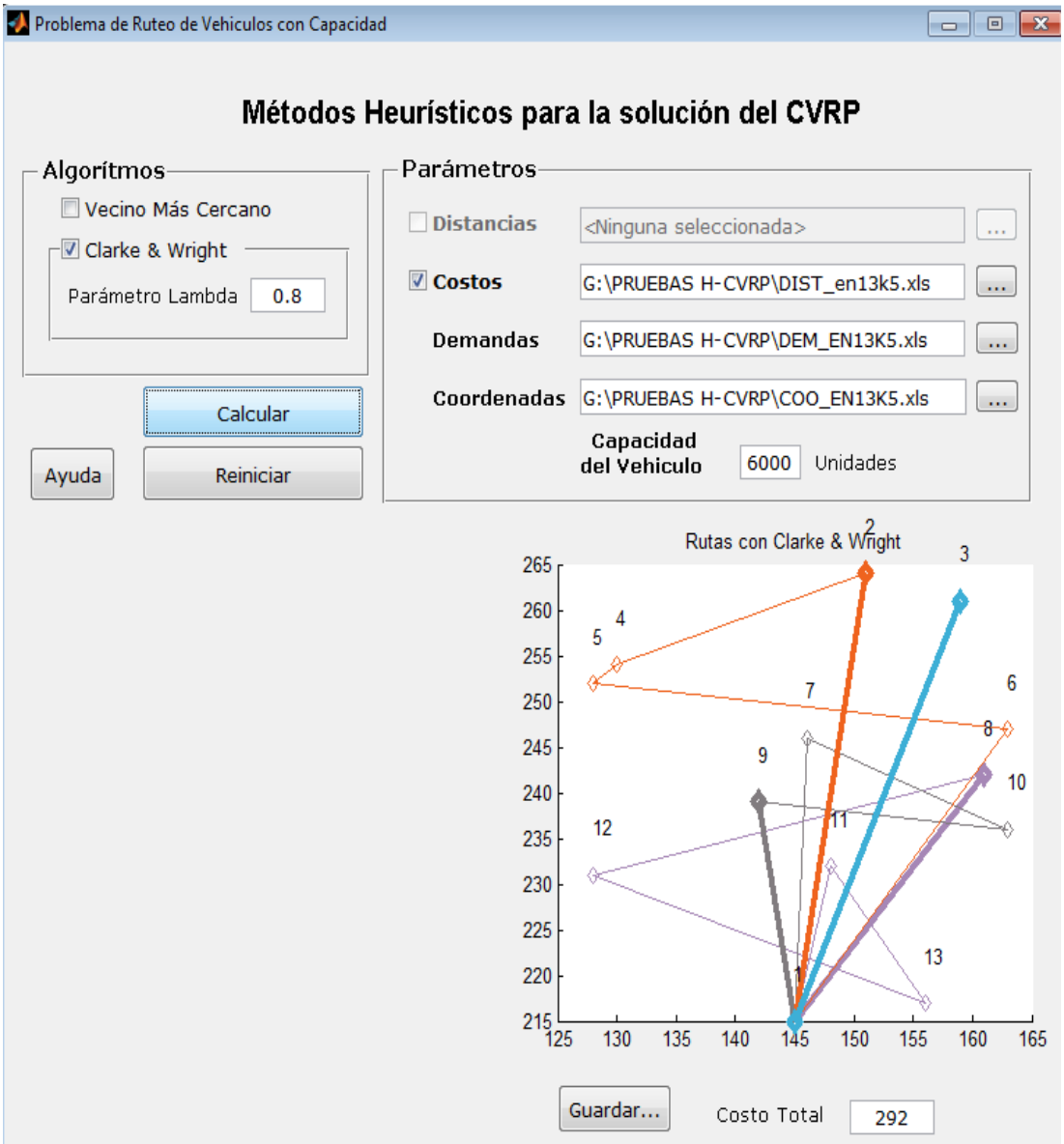
ANEXO 10. SALIDA DE H-CVRP PARA LA PRUEBA DE LA INSTANCIA

E-n13-k4

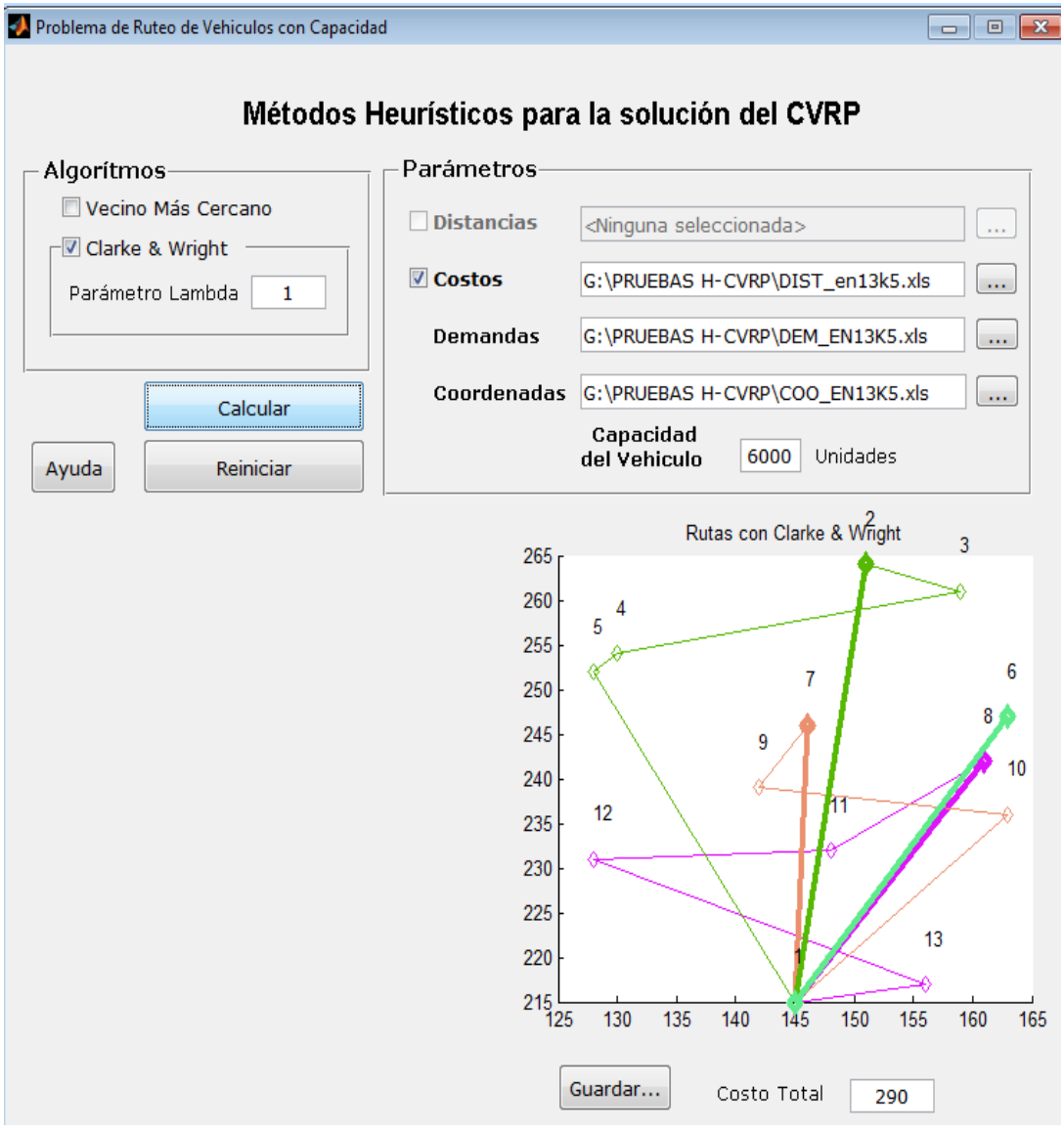
Clarke and Wright, con λ = 0.1



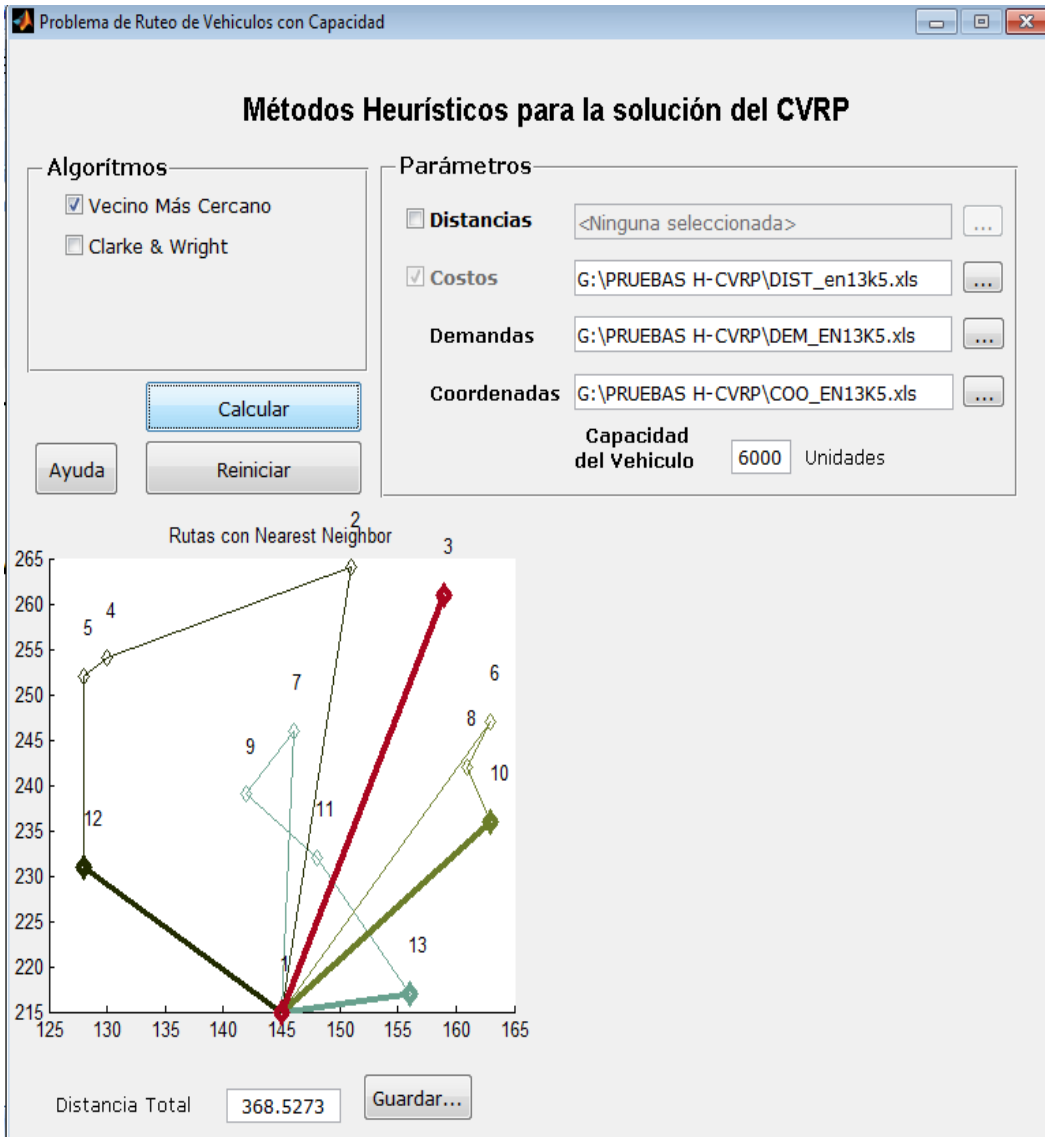
Clarke and Wright λ = 0.8



Clarke and Wright λ = 1

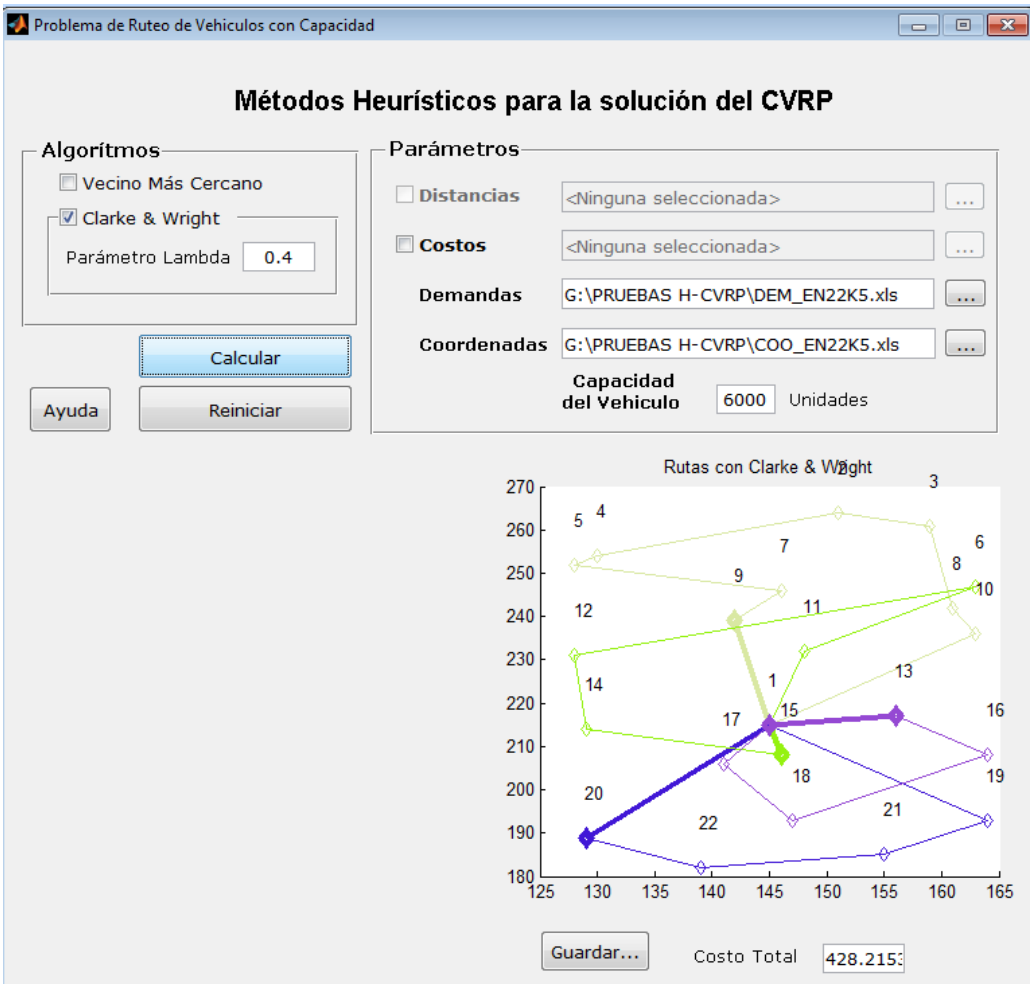


Vecino más cercano

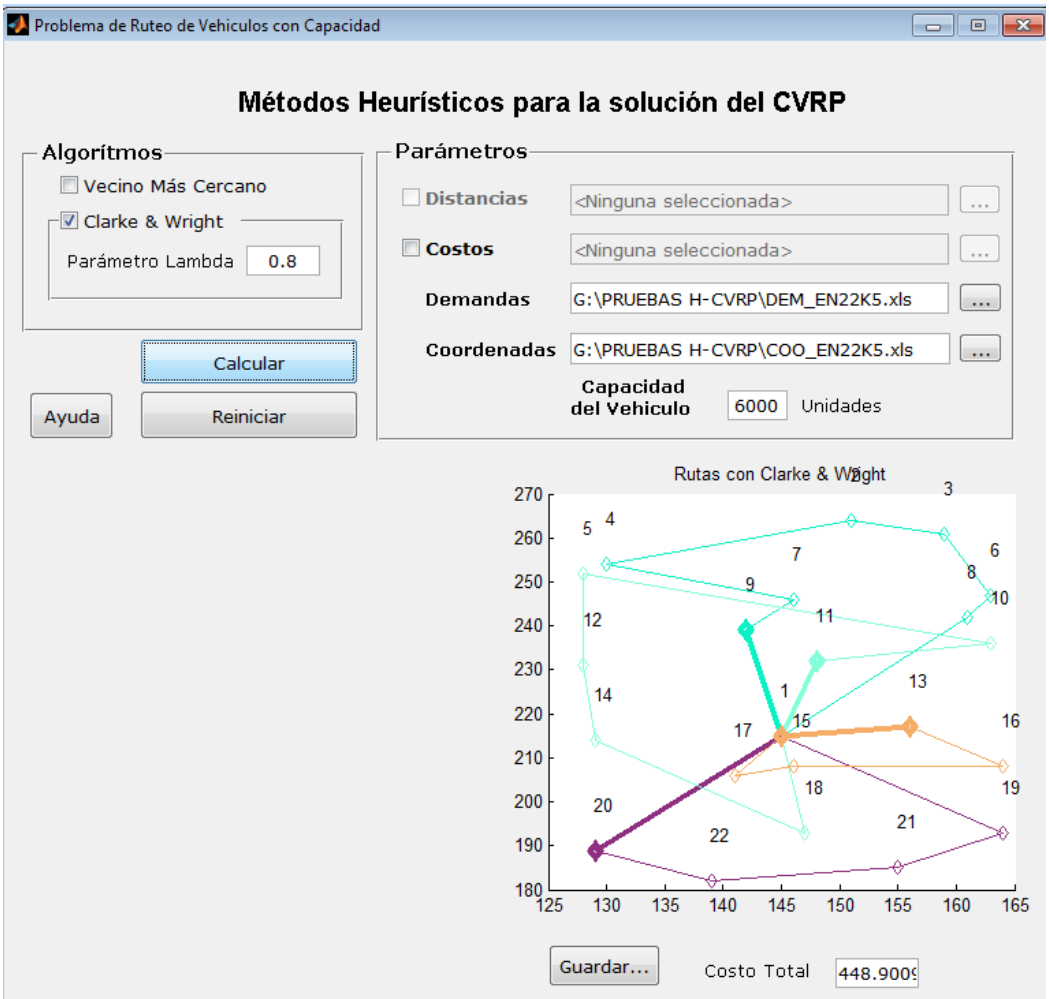


ANEXO 11. SALIDA DE H-CVRP PARA LA PRUEBA DE LA INSTANCIA E-n22-K4

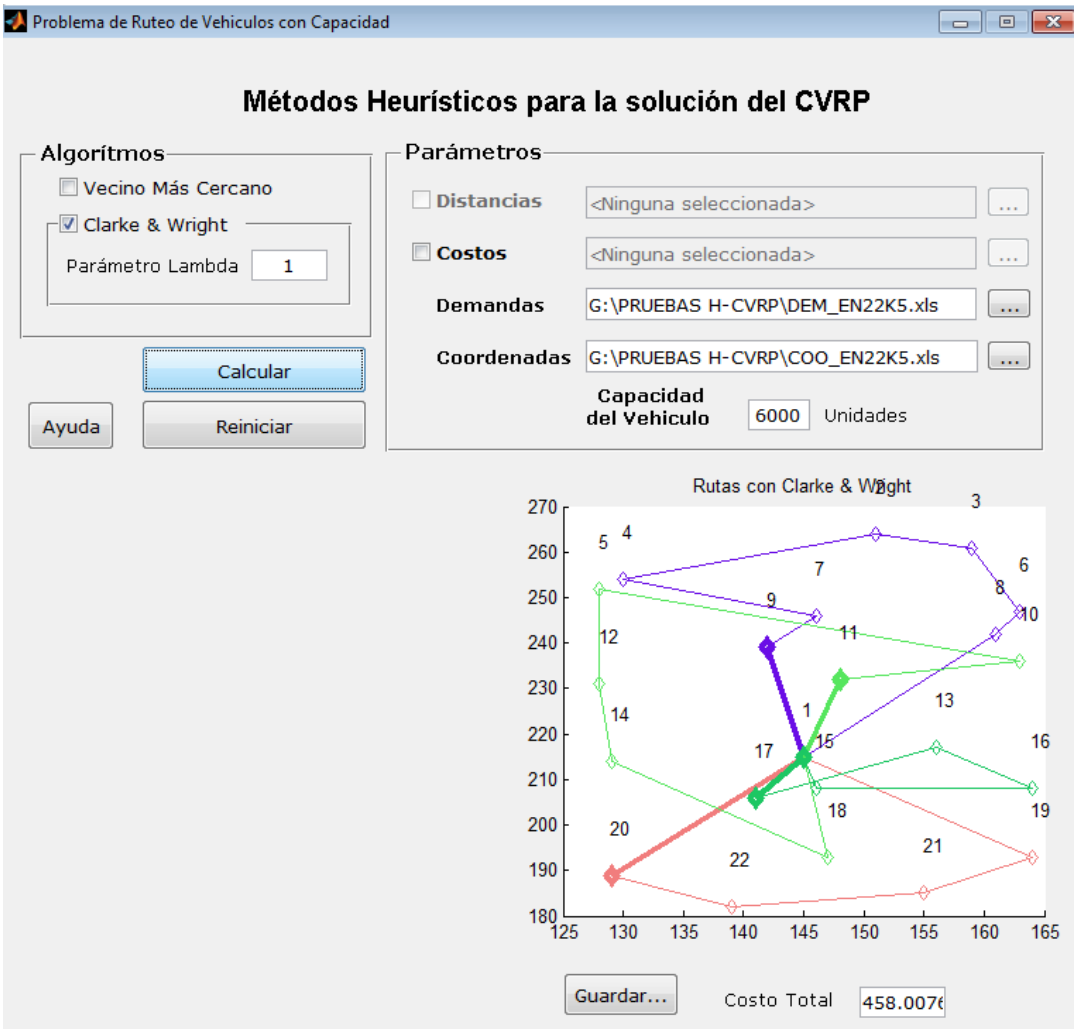
Clarke and Wright λ = 0.4



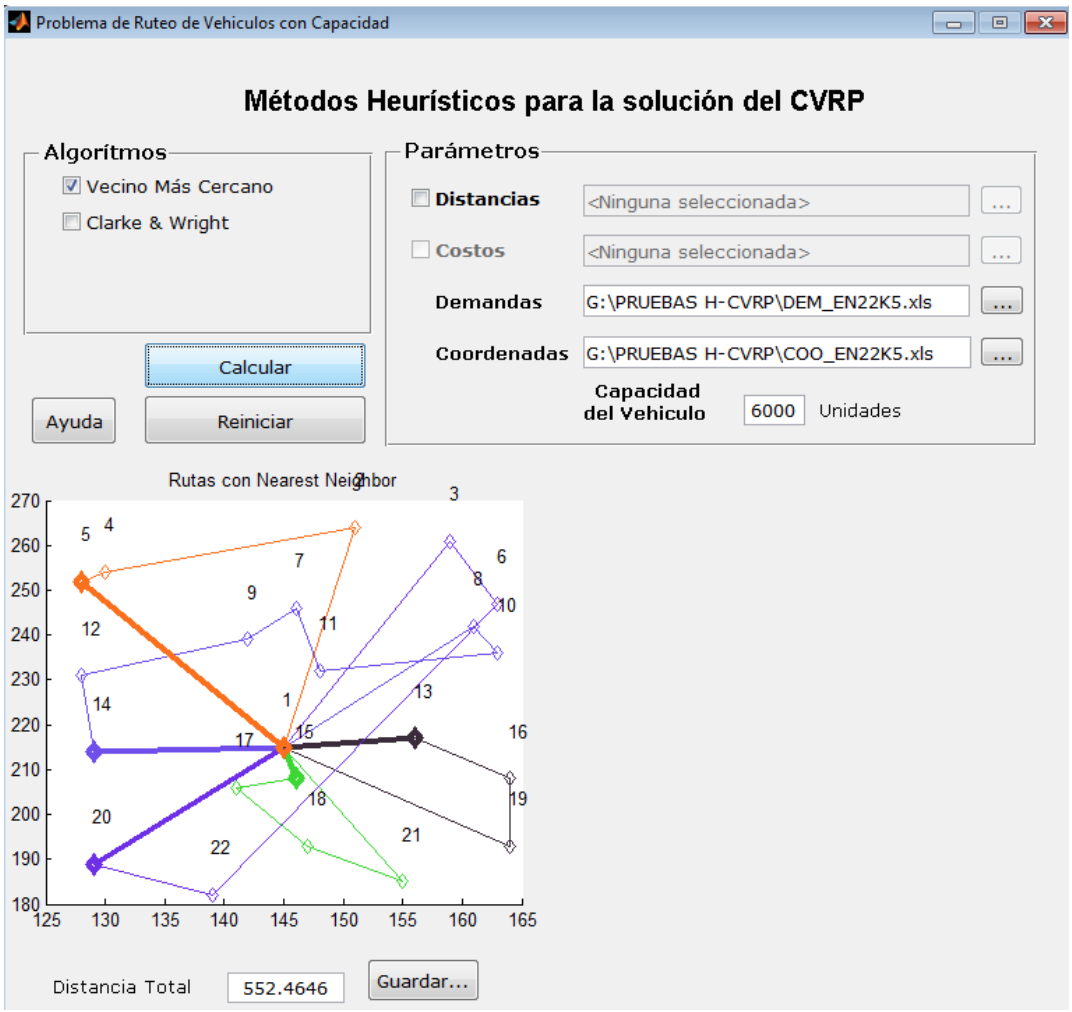
Clarke and Wright λ = 0.8



Clarke and Wright λ = 1

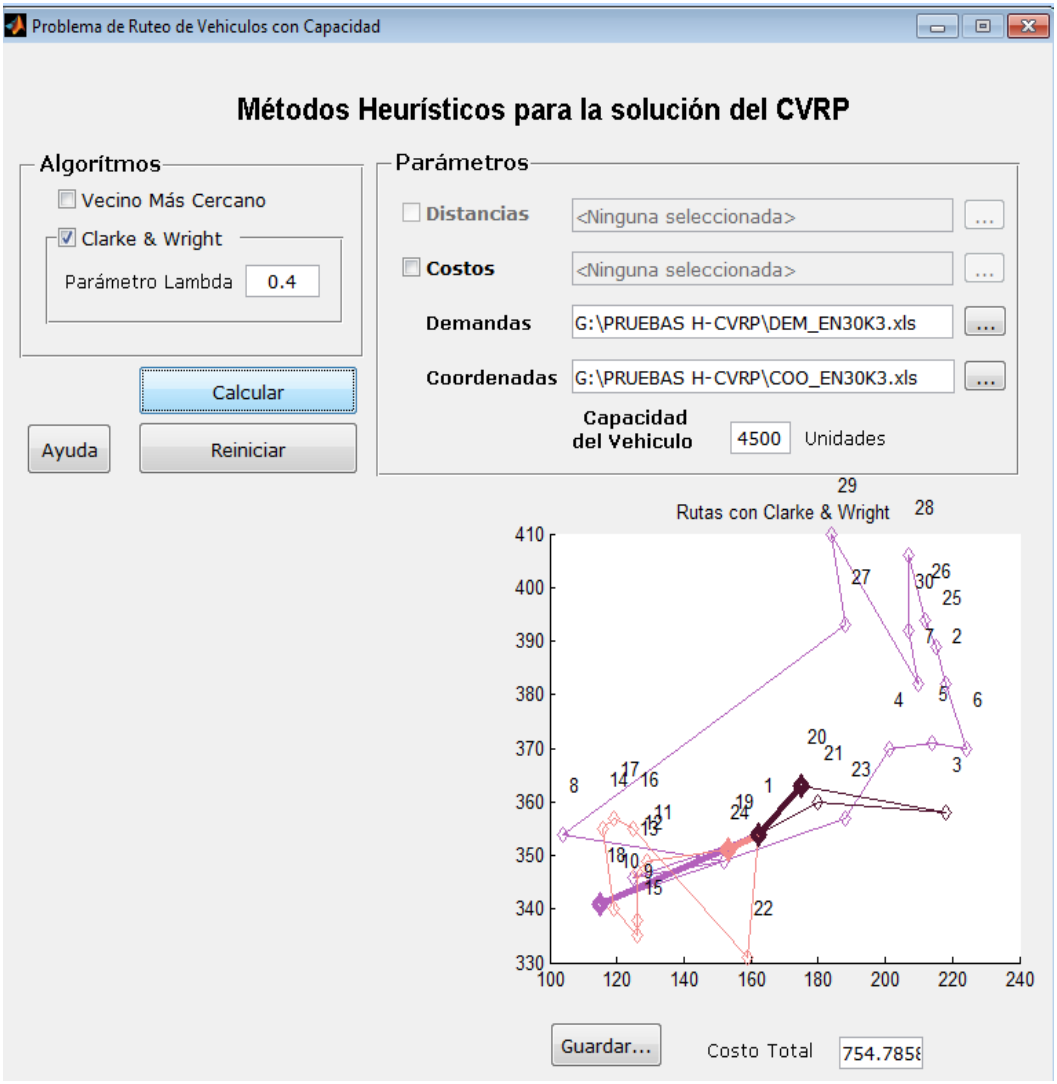


Vecino más cercano

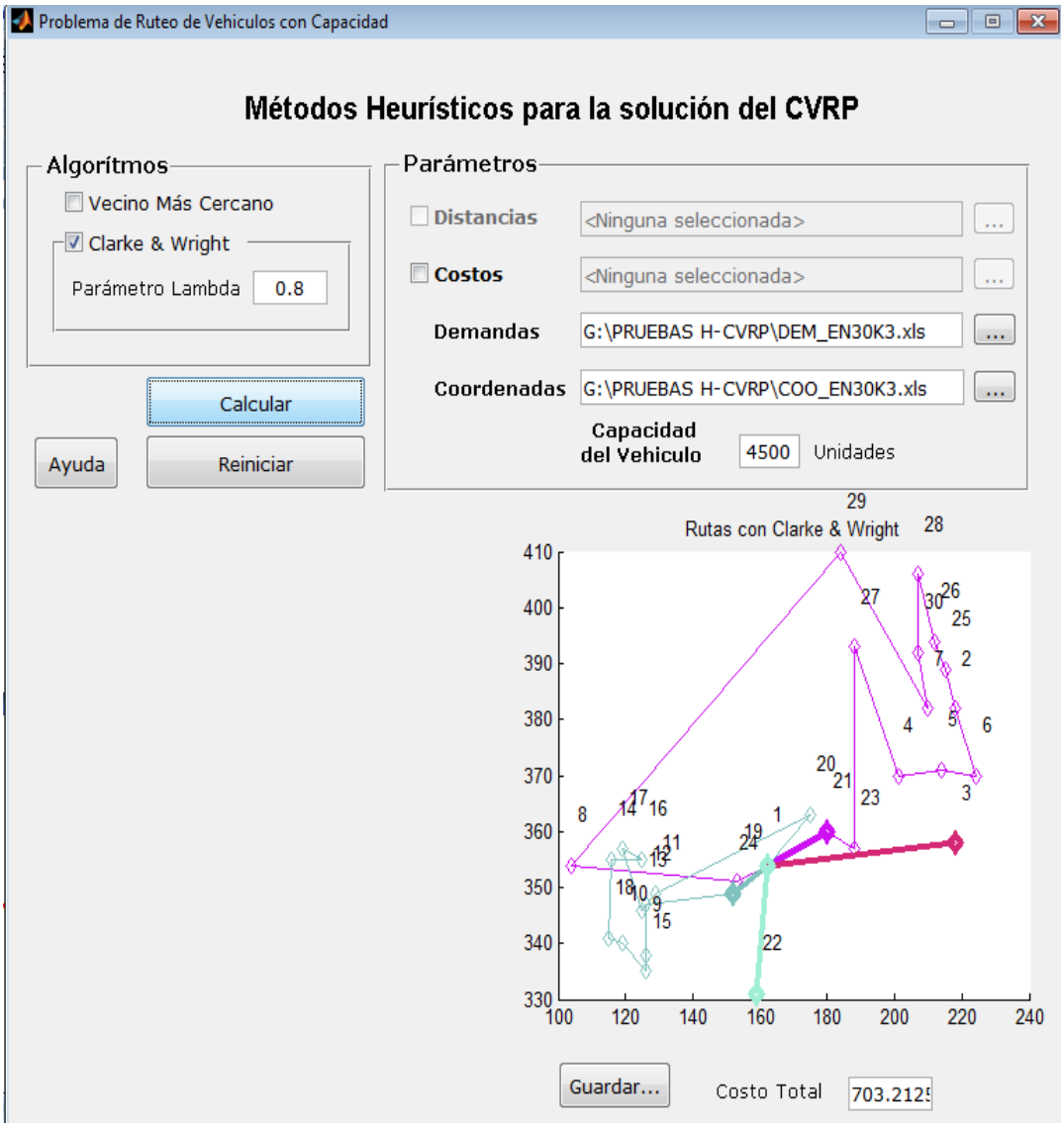


ANEXO 12. SALIDA DE H-CVRP PARA LA PRUEBA DE LA INSTANCIA E-n30-K3

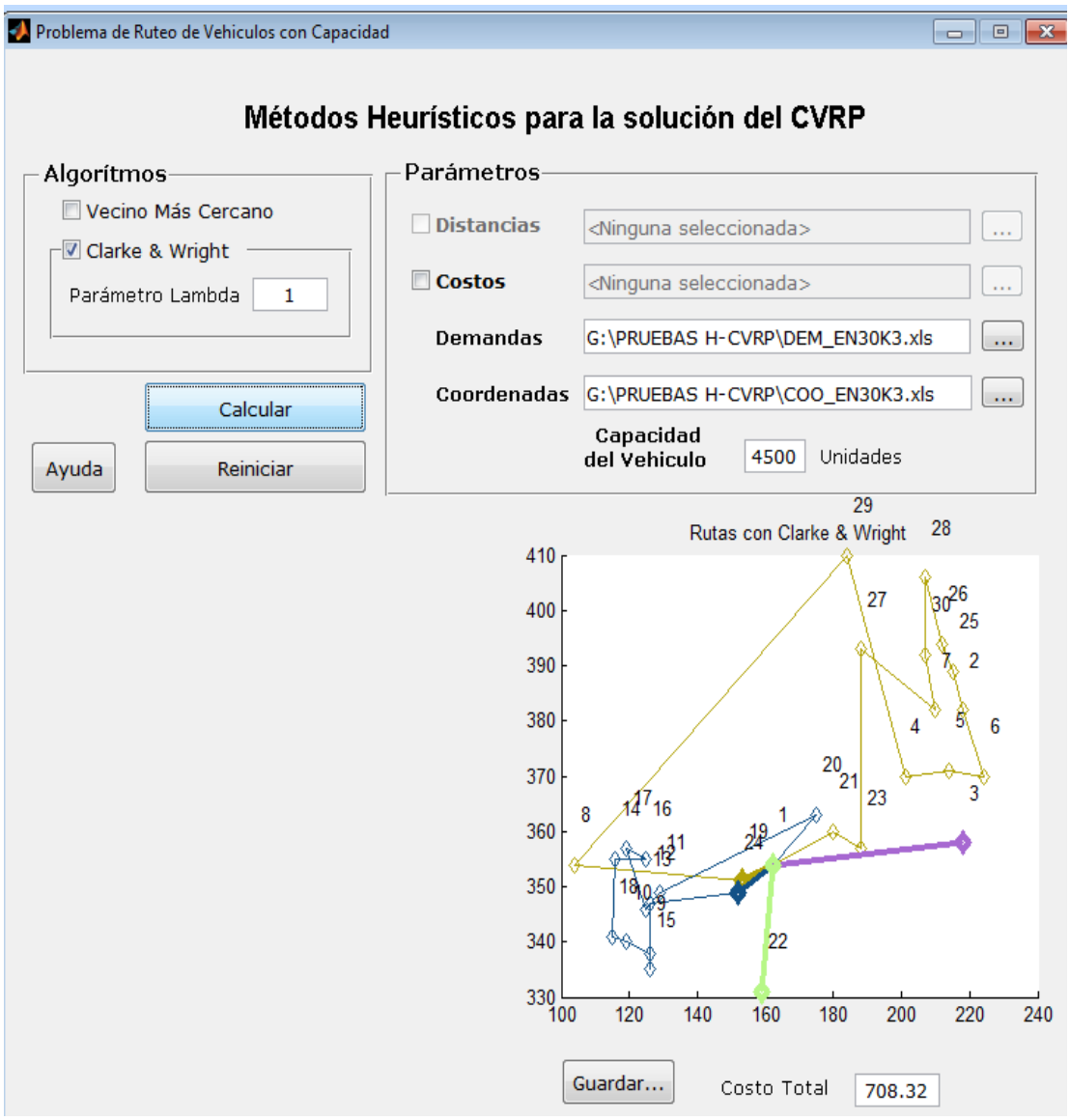
Clarke and Wright λ = 0.4



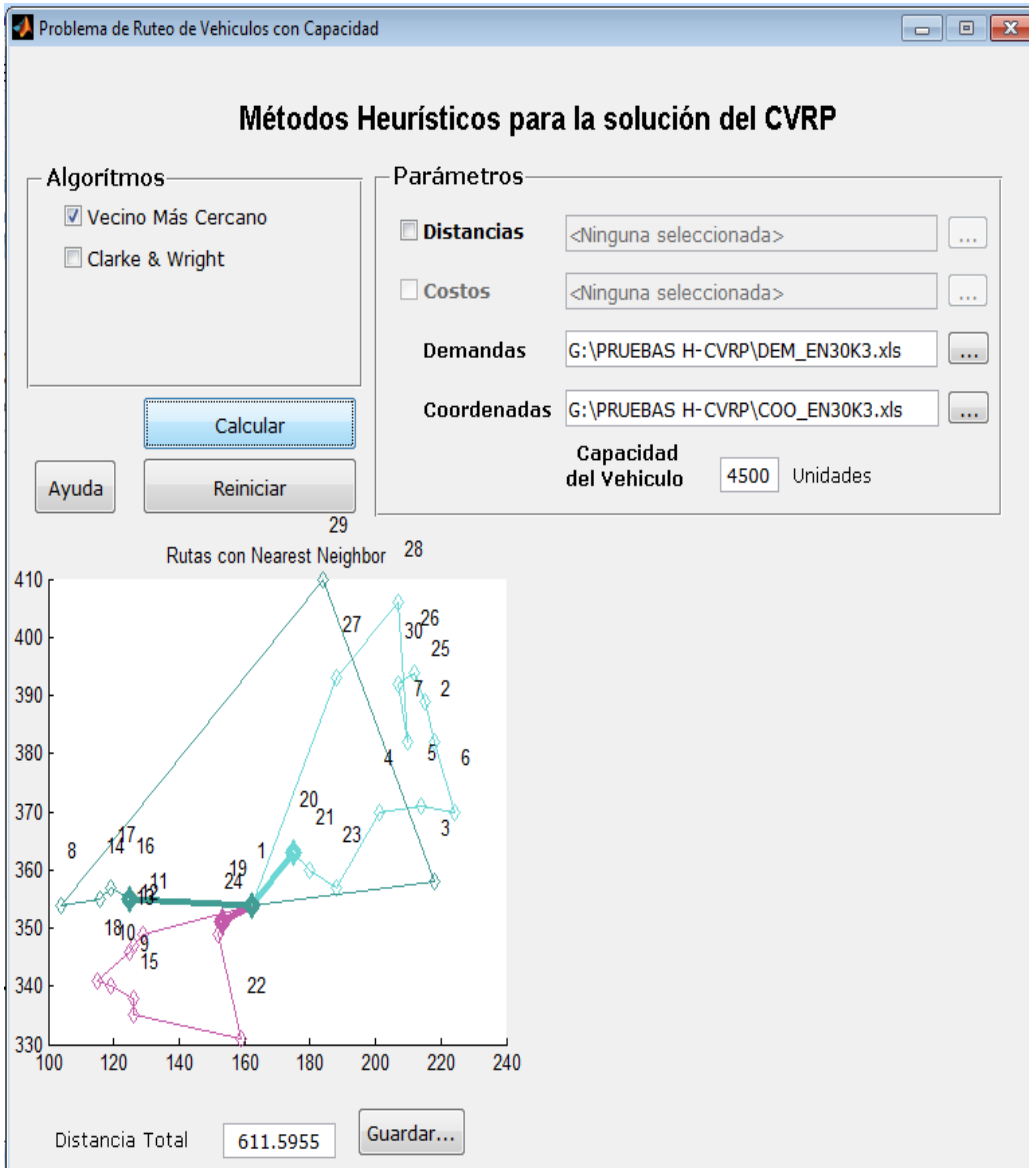
Clarke and Wright λ = 0.8



Clarke and Wright λ = 1

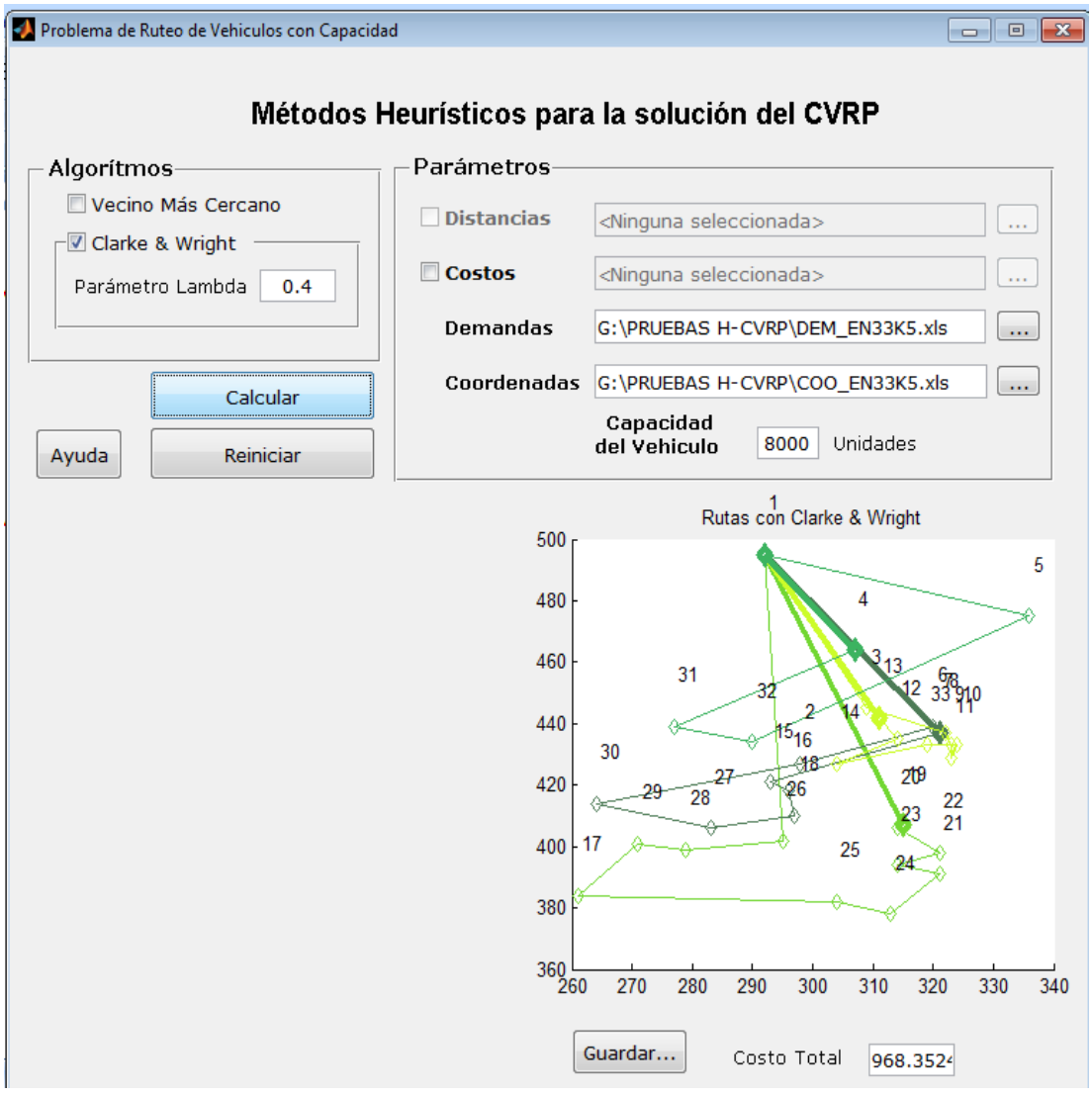


Vecino más cercano

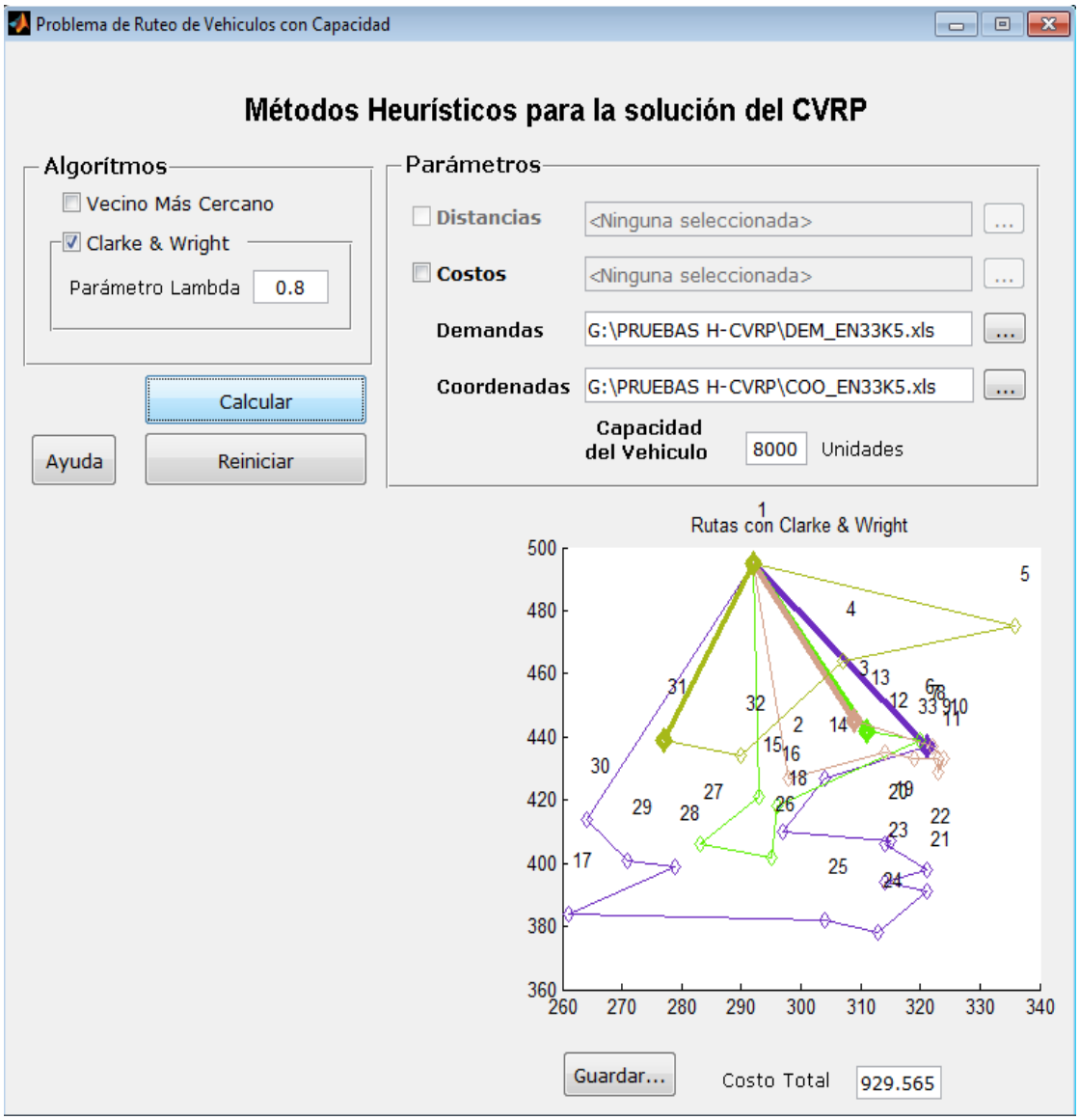


ANEXO 13. SALIDA DE H-CVRP PARA LA PRUEBA DE LA INSTANCIA E-n33-K4

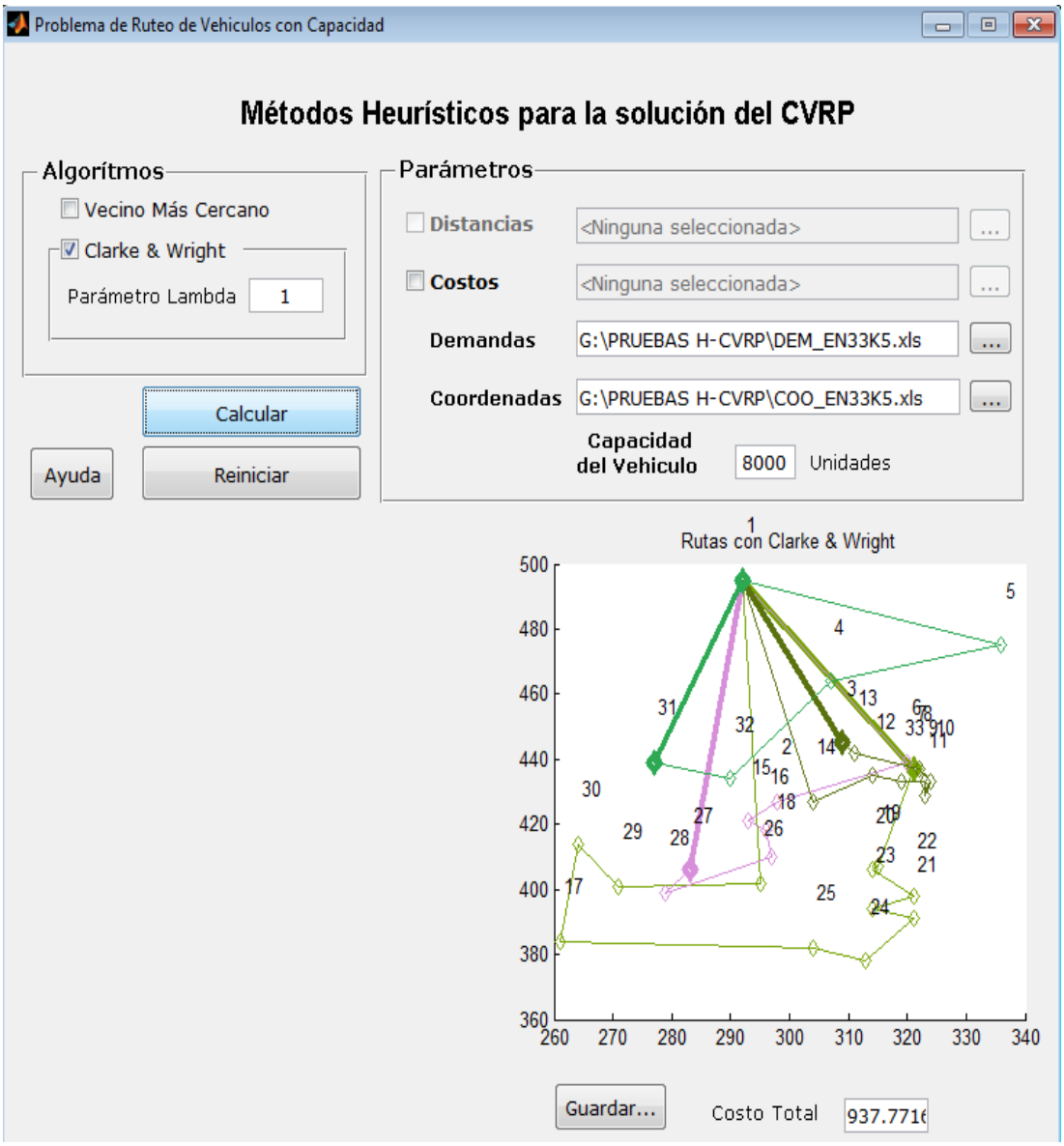
Clarke and Wright λ = 0.4



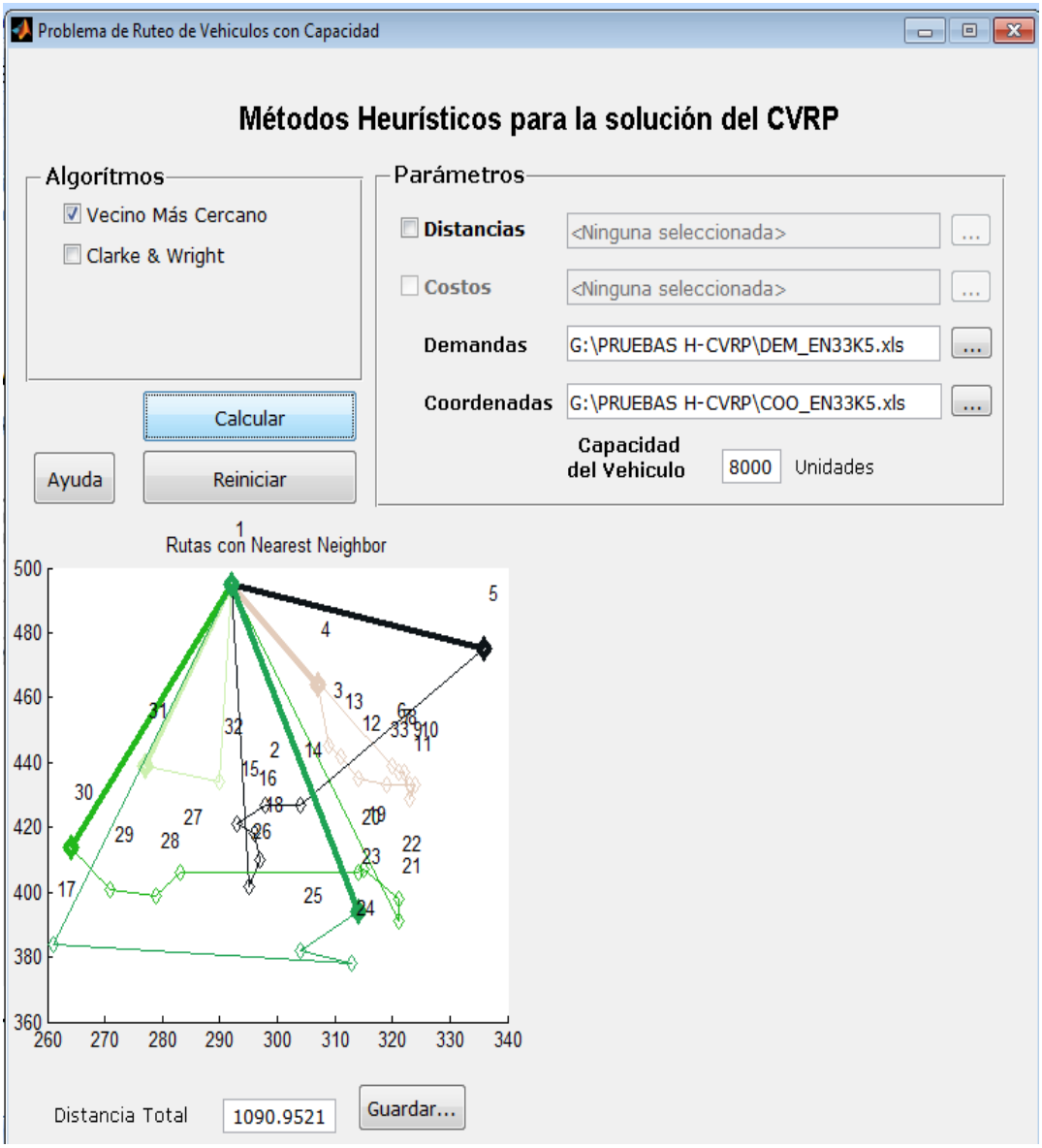
Clarke and Wright λ = 0.8



Clarke and Wright λ = 1

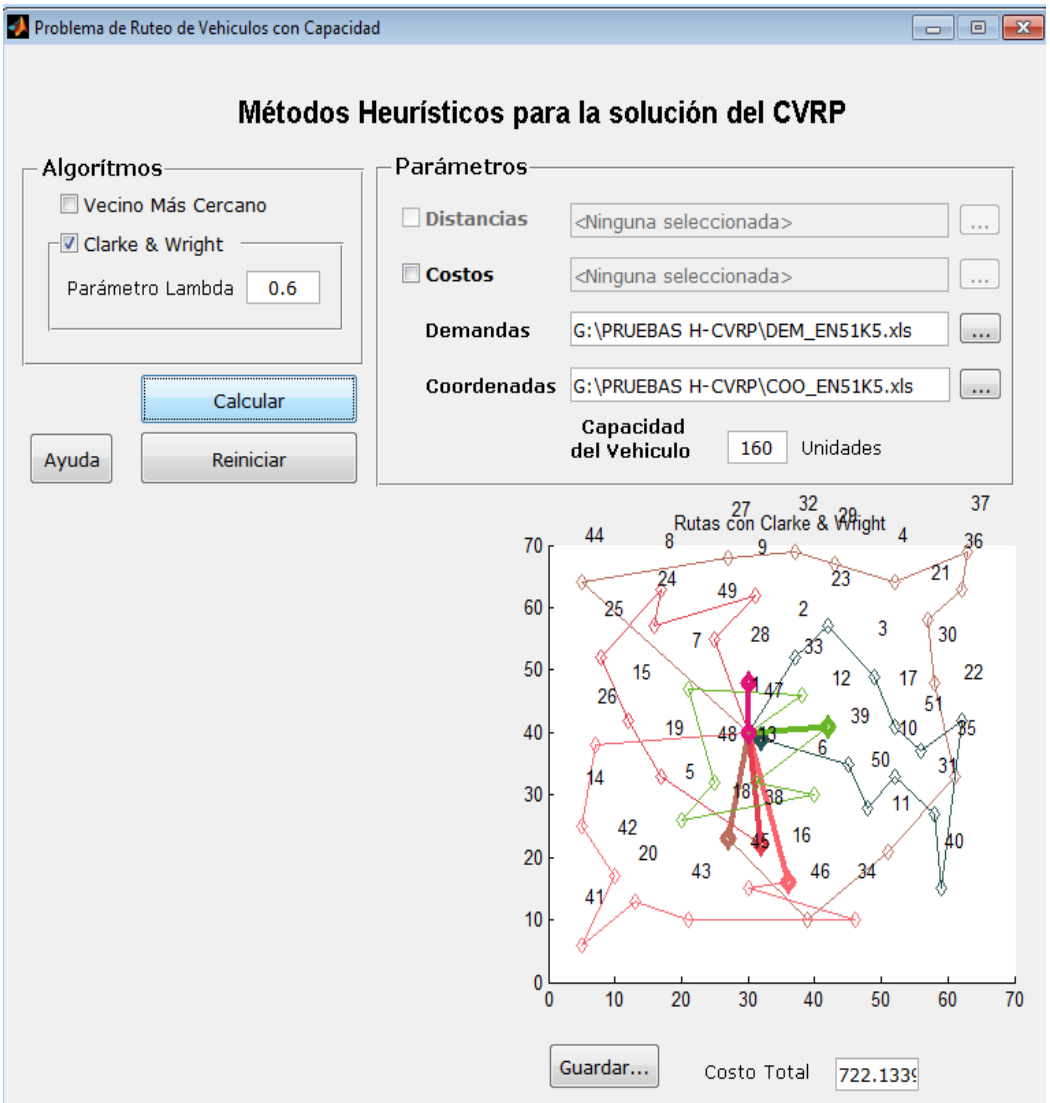


Vecino más cercano

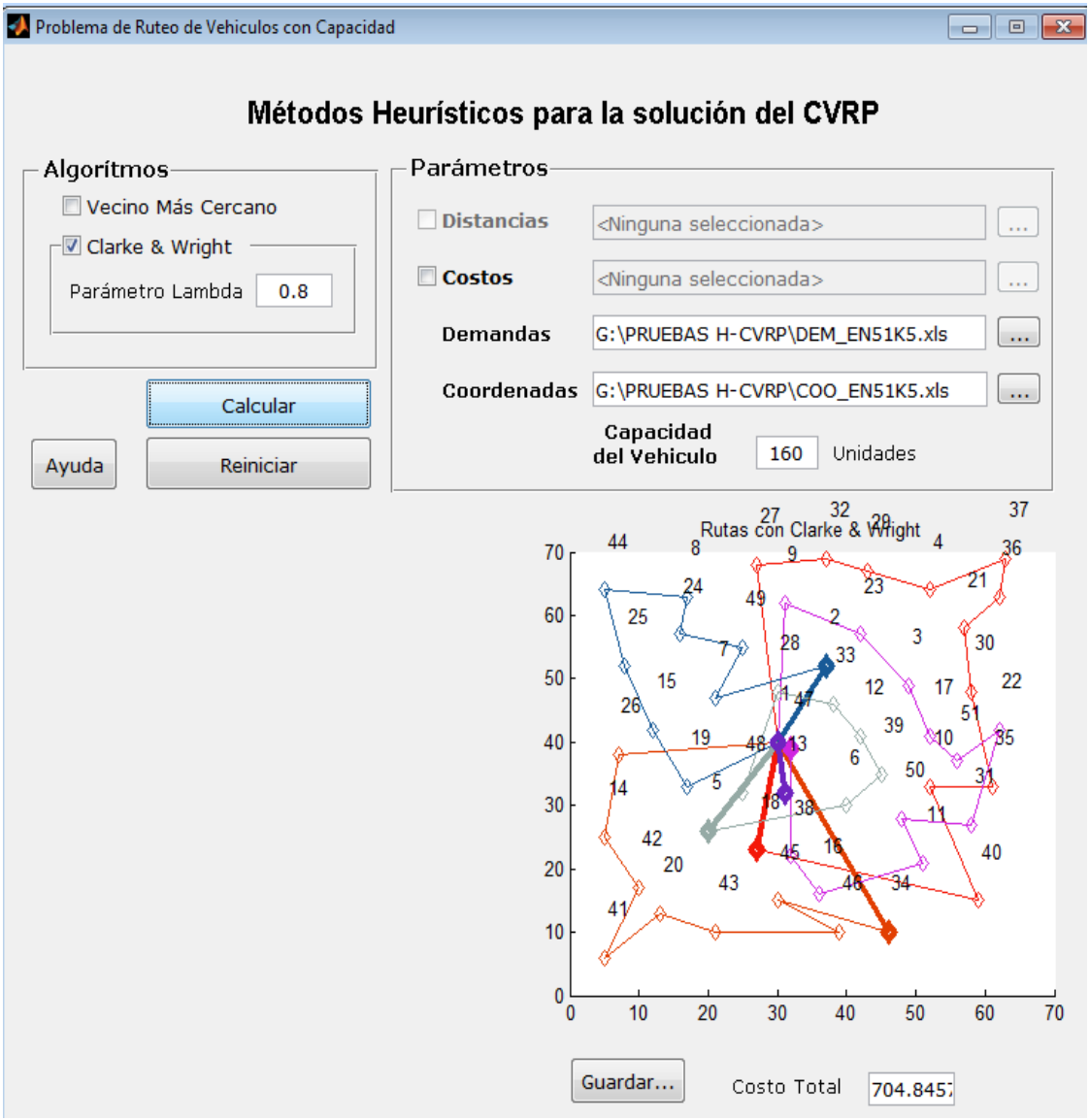


ANEXO 14. SALIDA DE H-CVRP PARA LA PRUEBA DE LA INSTANCIA E-n51-K5

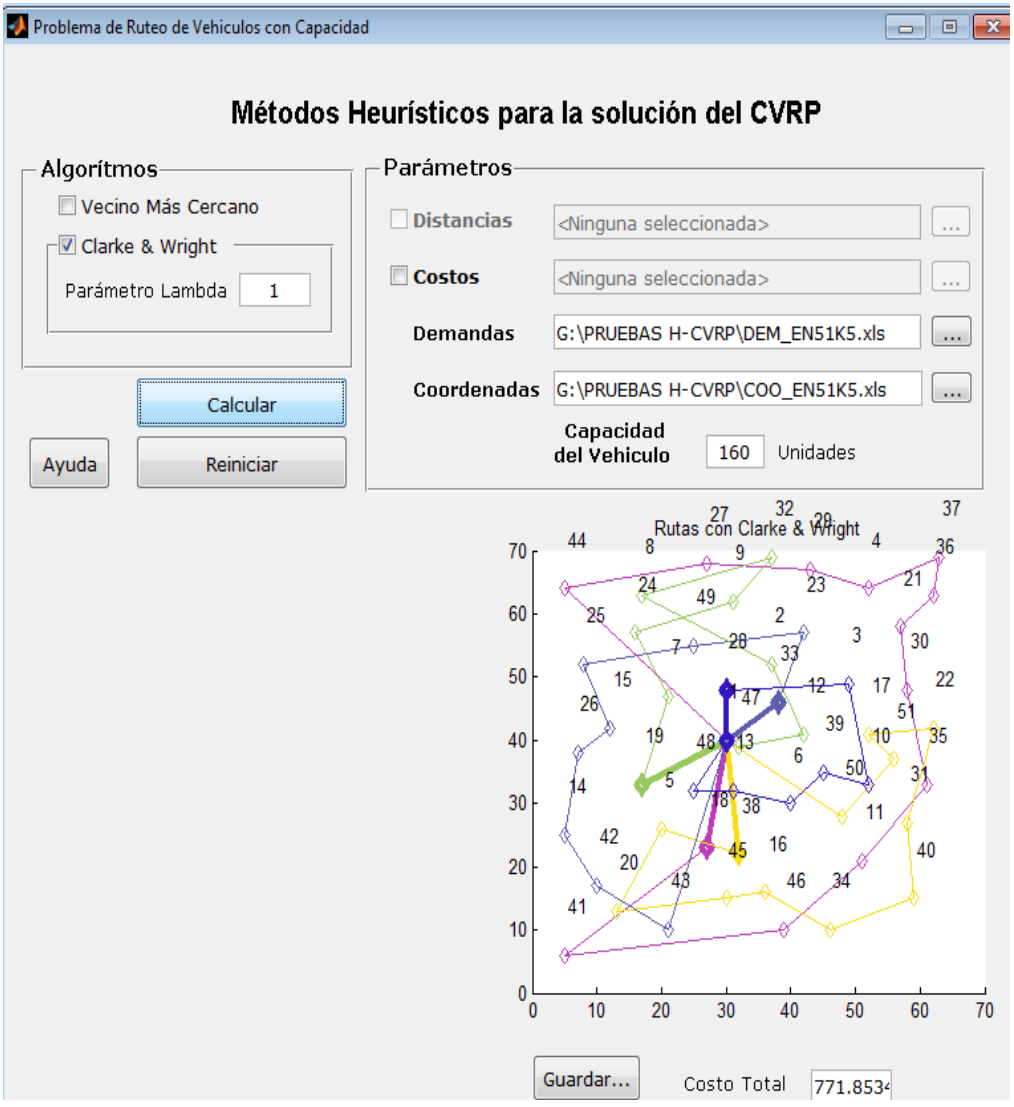
Clarke and Wright λ = 0.6



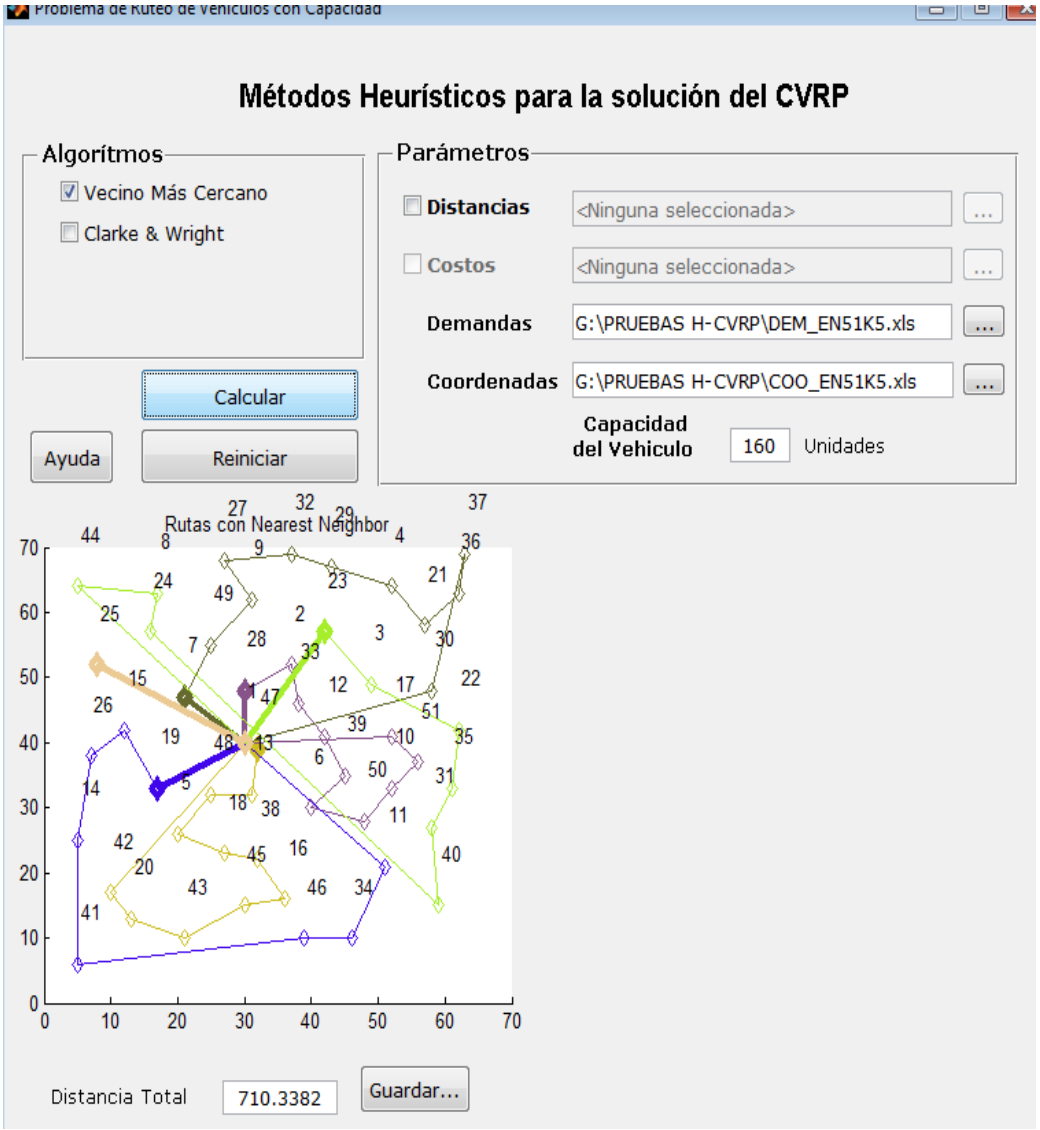
Clarke and Wright λ = 0.8



Clarke and Wright λ = 1

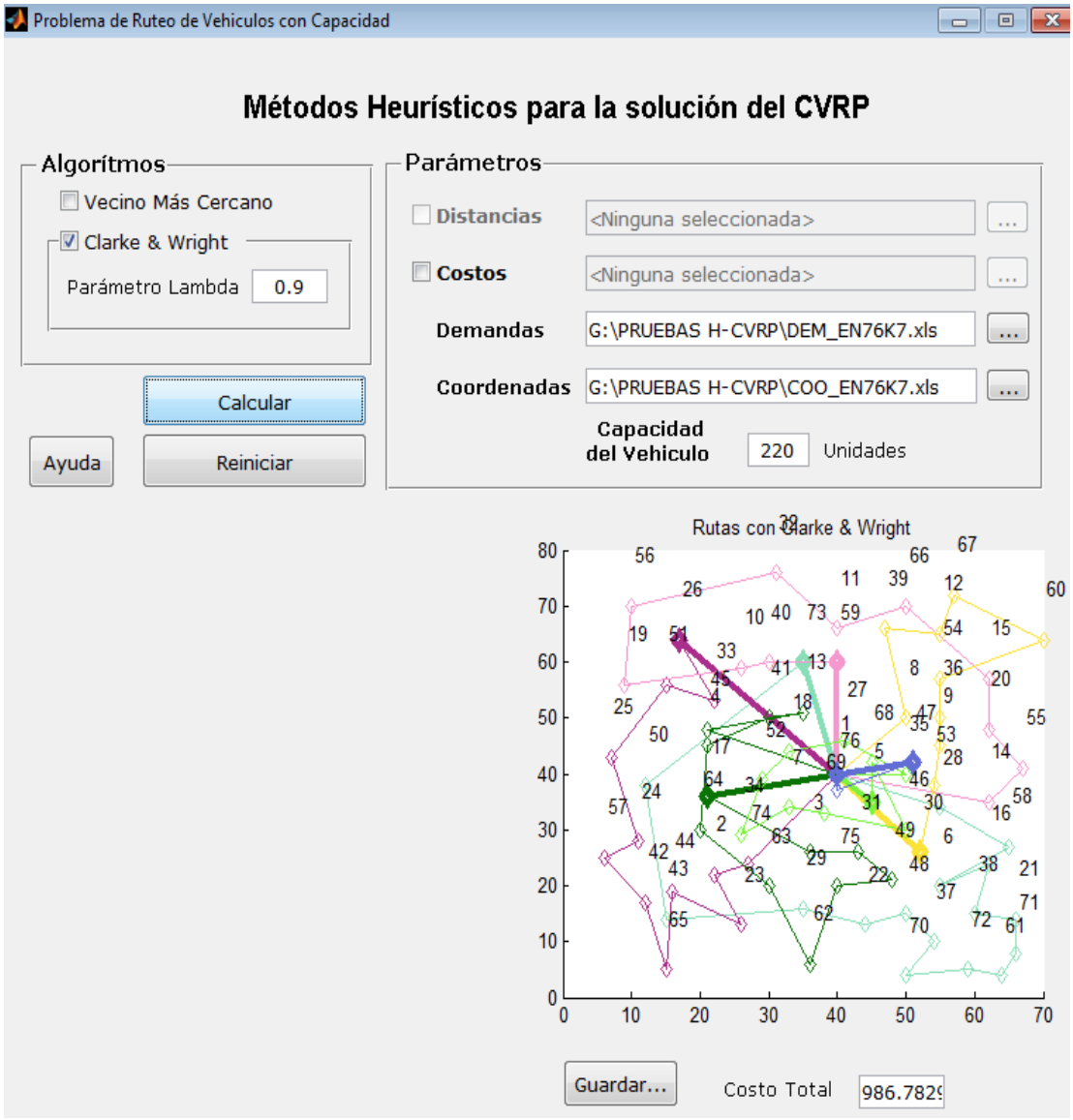


Vecino más cercano

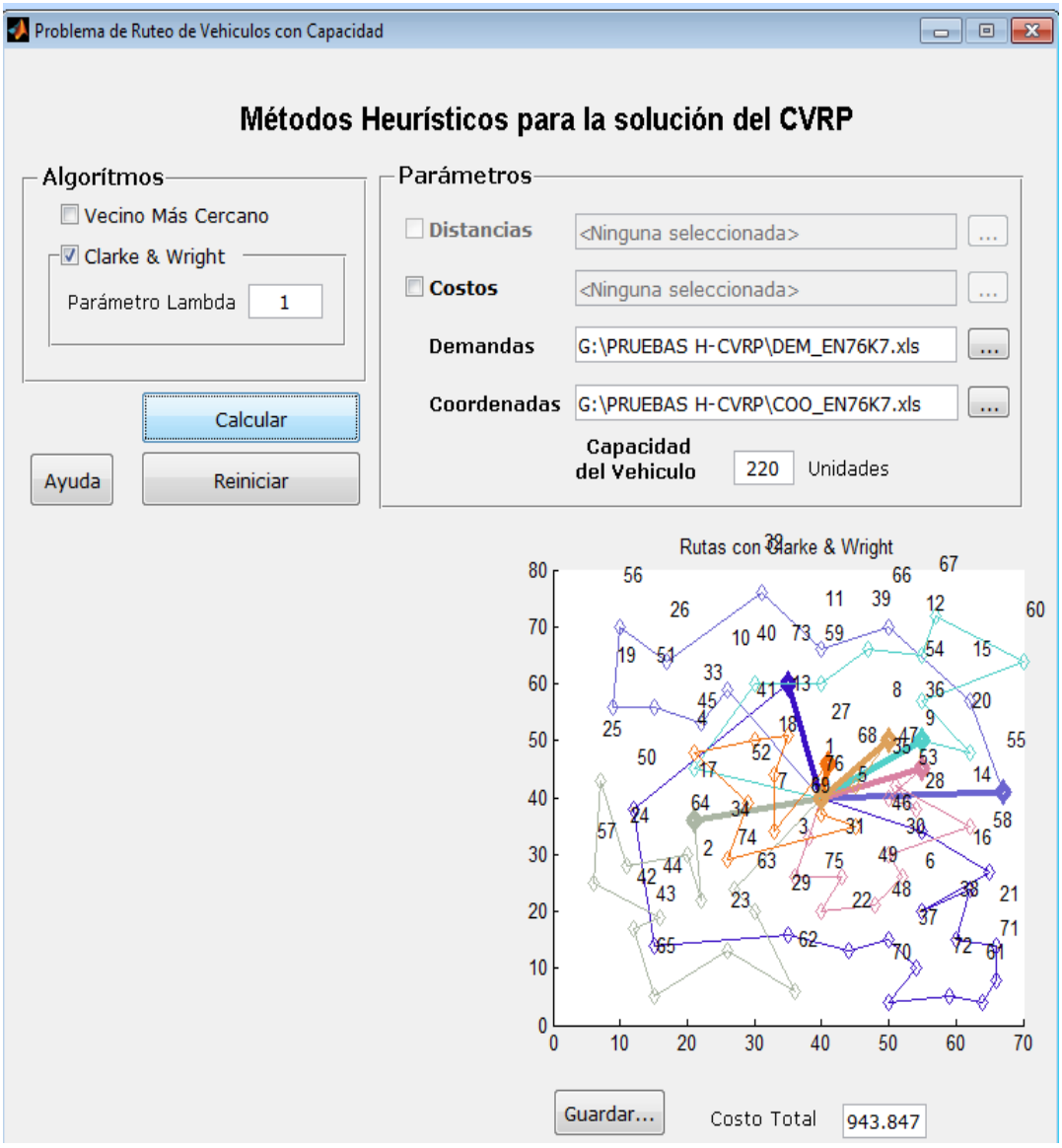


ANEXO 15. SALIDA DE H-CVRP PARA LA PRUEBA DE LA INSTANCIA E-n76-K7

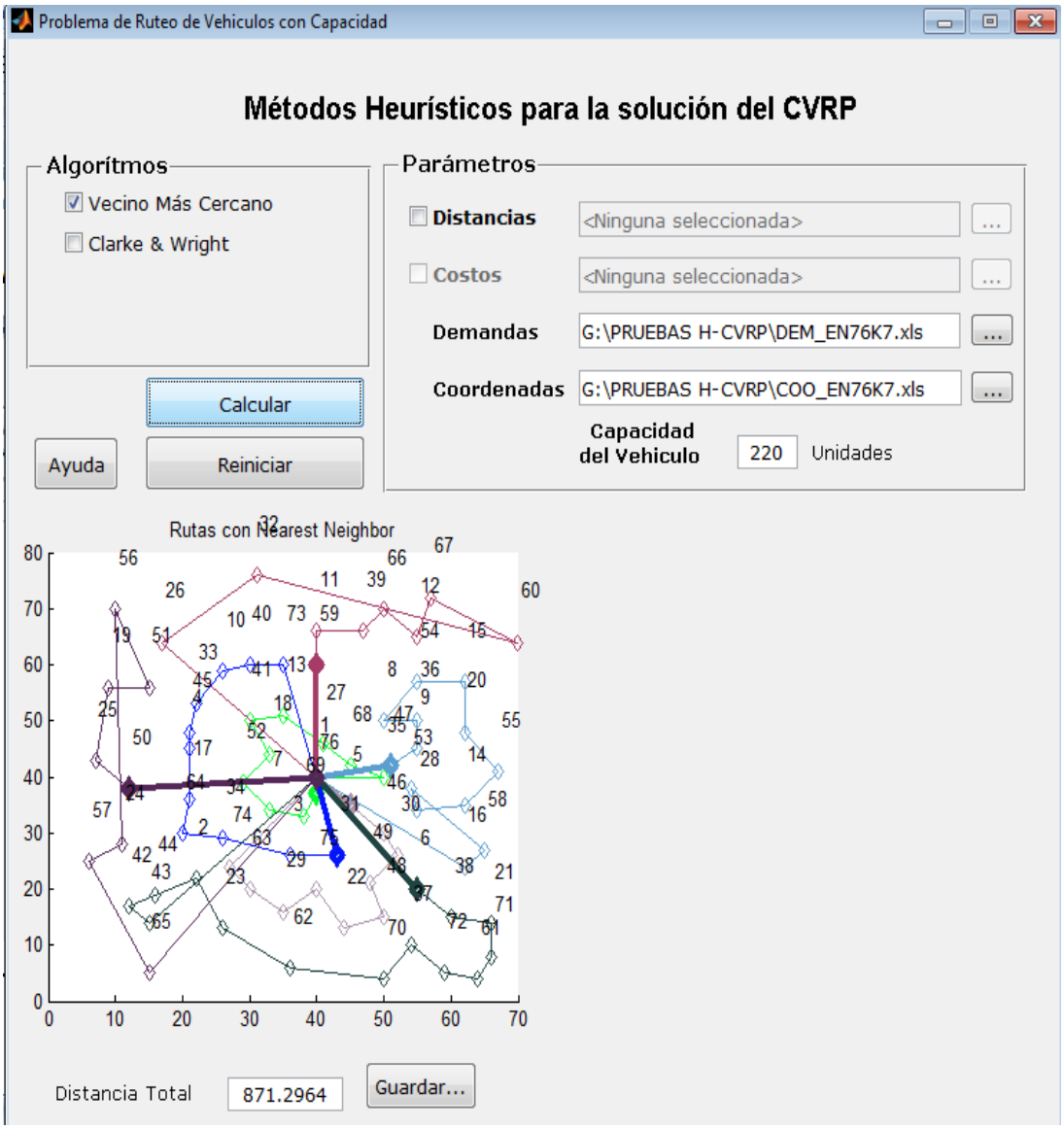
Clarke and Wright λ = 0.9



Clarke and Wright λ = 1

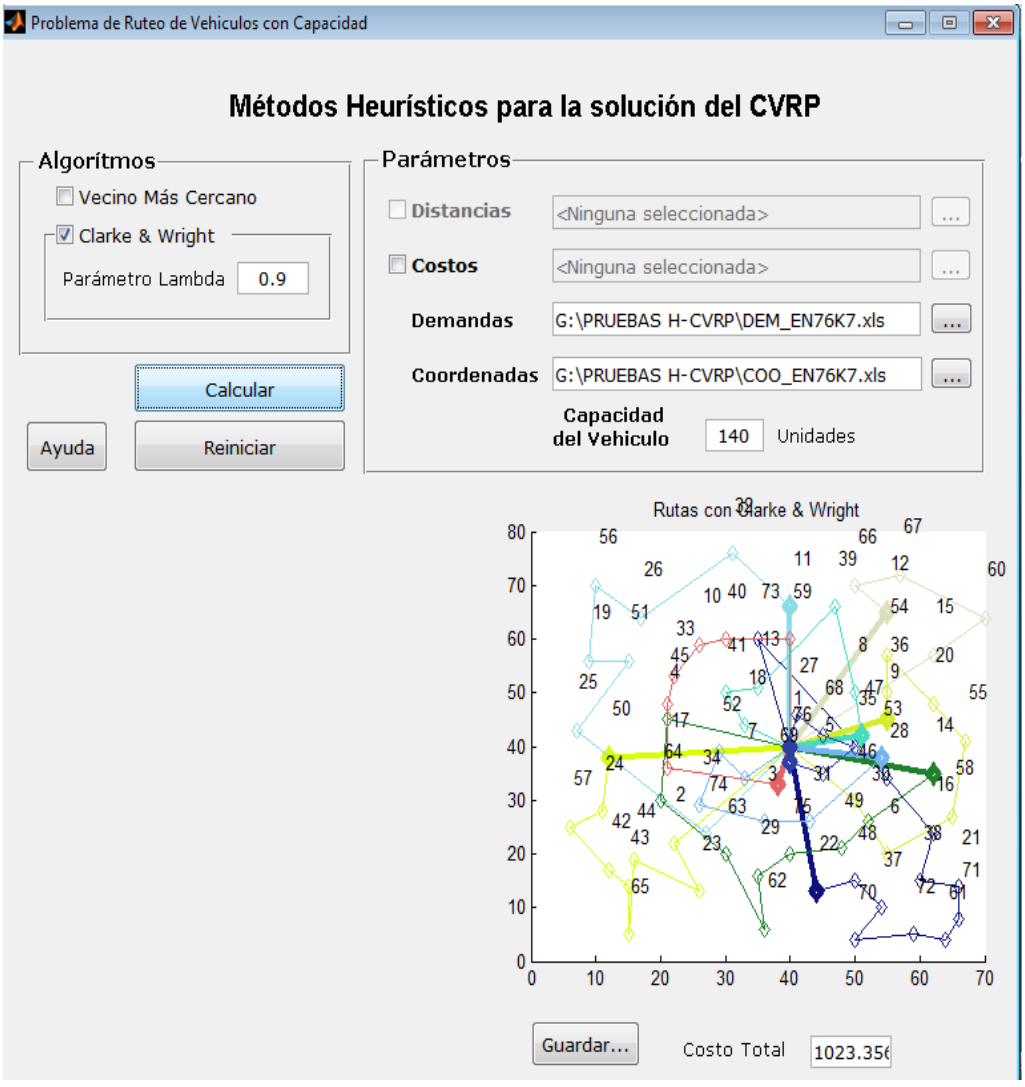


Vecino más cercano

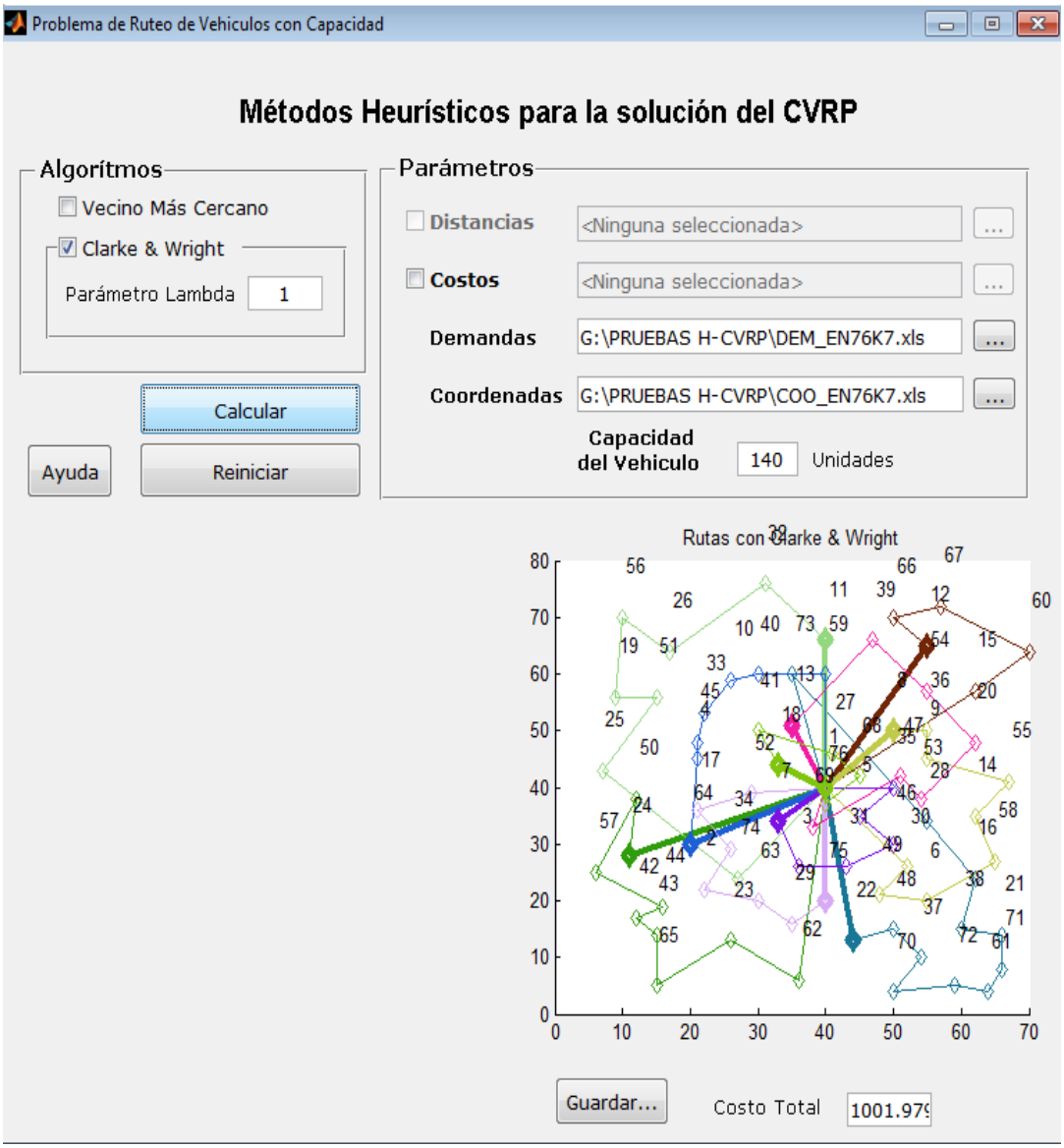


ANEXO 16. SALIDA DE H-CVRP PARA LA PRUEBA DE LA INSTANCIA E-n76-K10

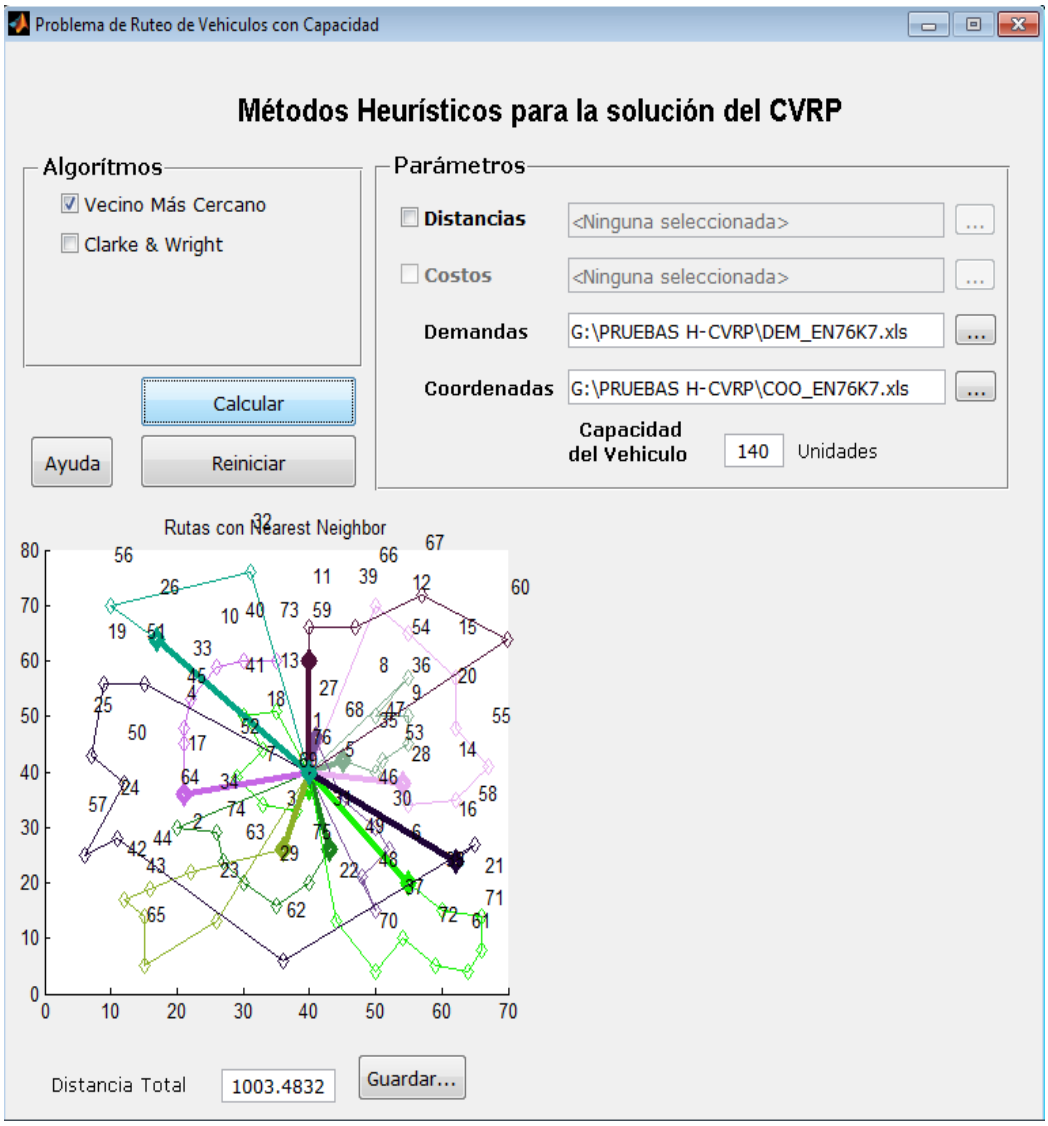
Clarke and Wright λ = 0.9



Clarke and Wright λ = 1



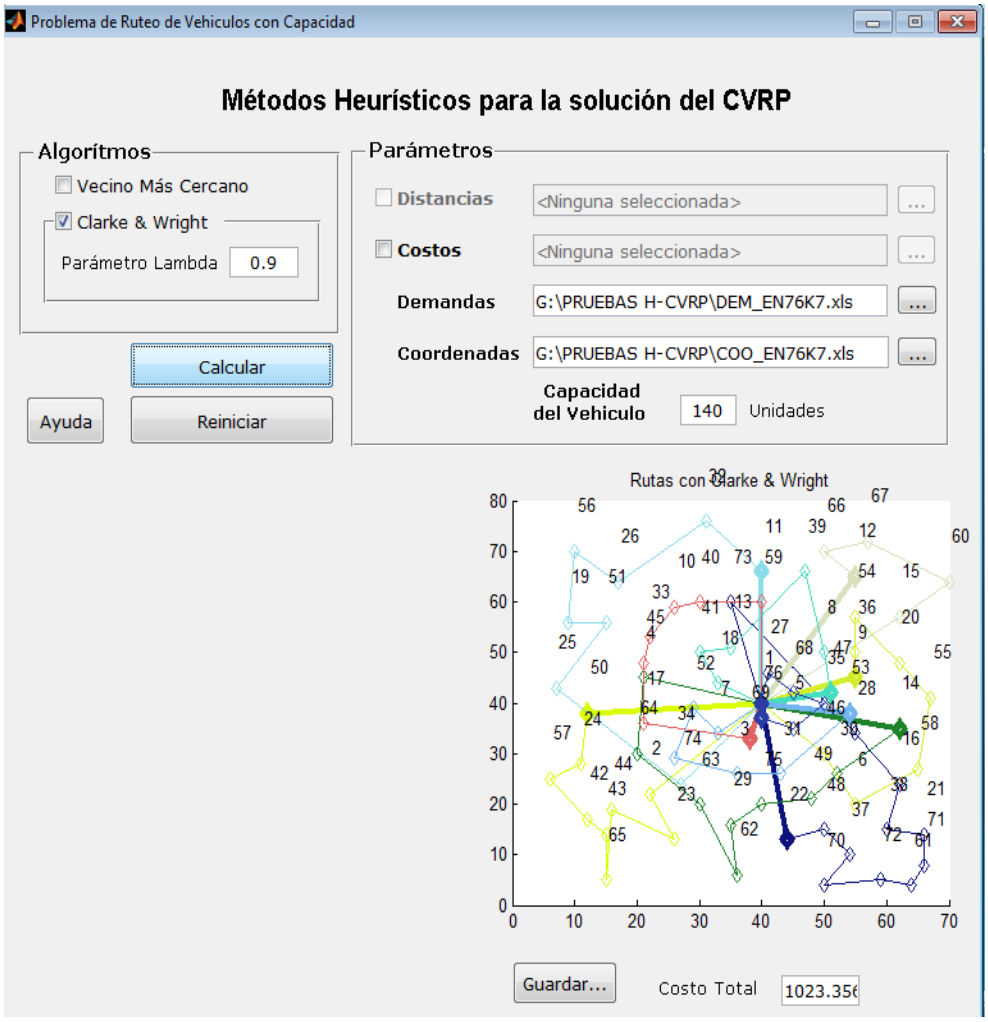
Vecino más cercano



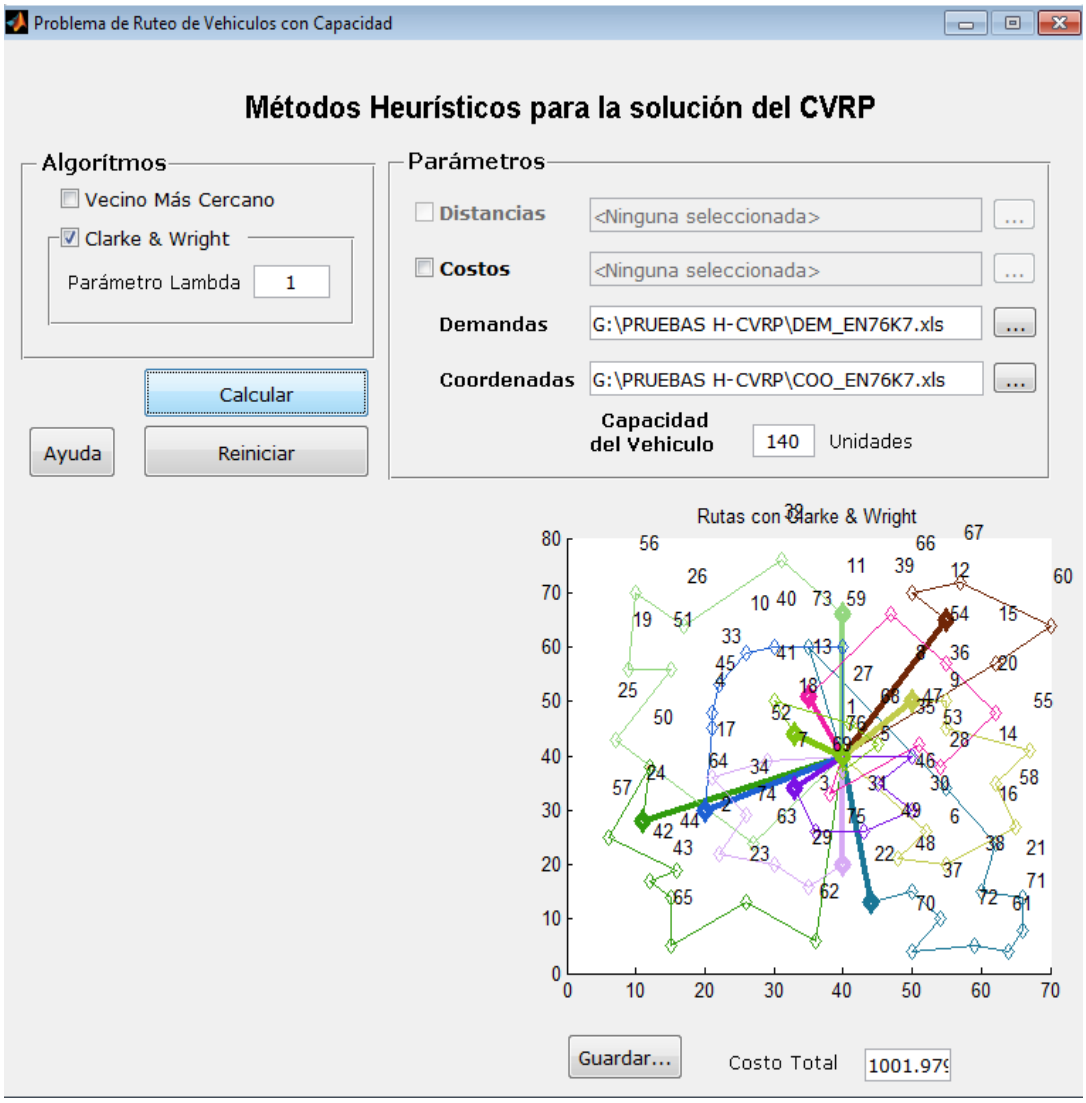
ANEXO 17. SALIDA DE H-CVRP PARA LA PRUEBA DE LA INSTANCIA

E-n76-K10

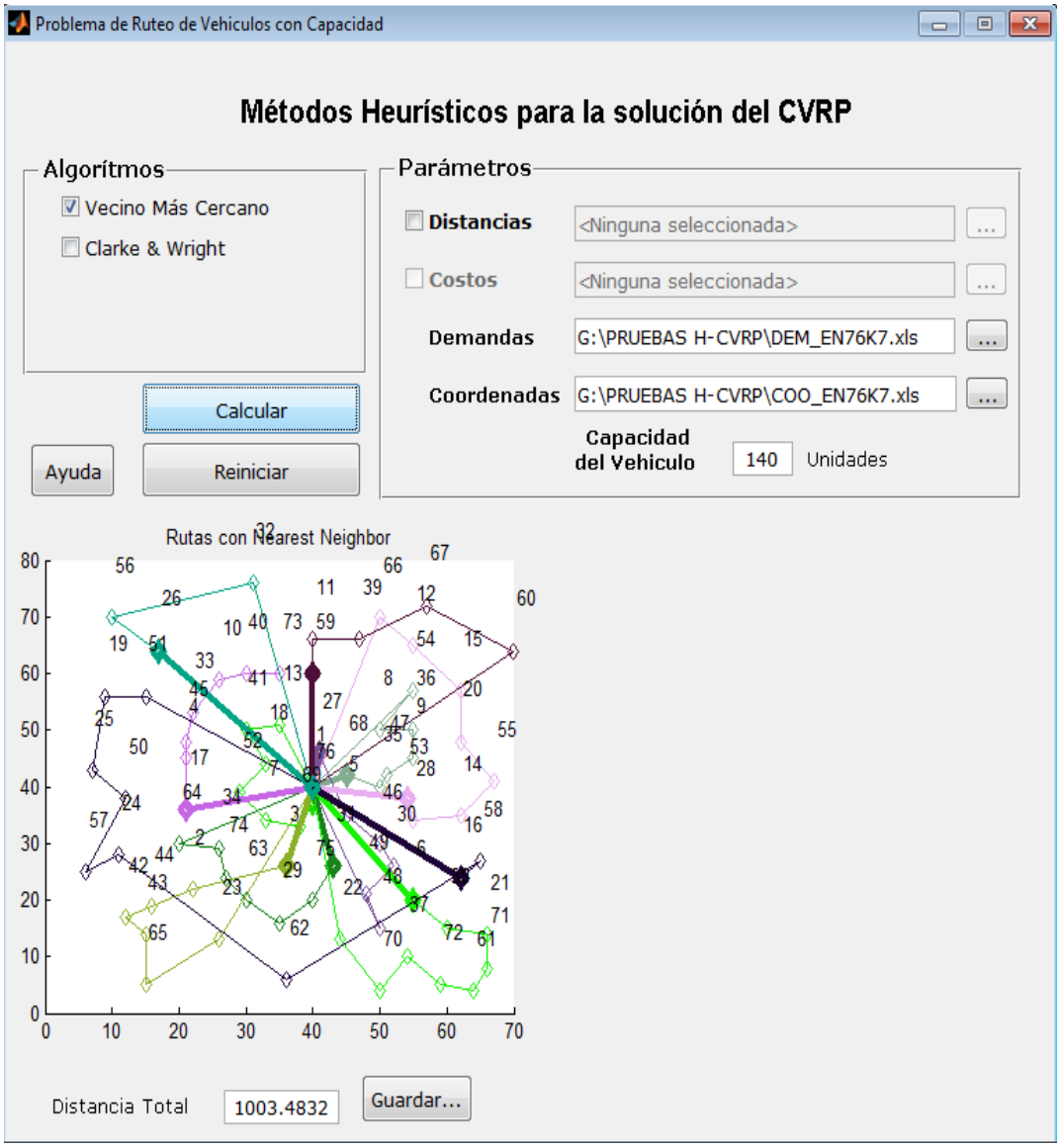
Clarke and Wright λ = 0.9



Clarke and Wright λ = 1

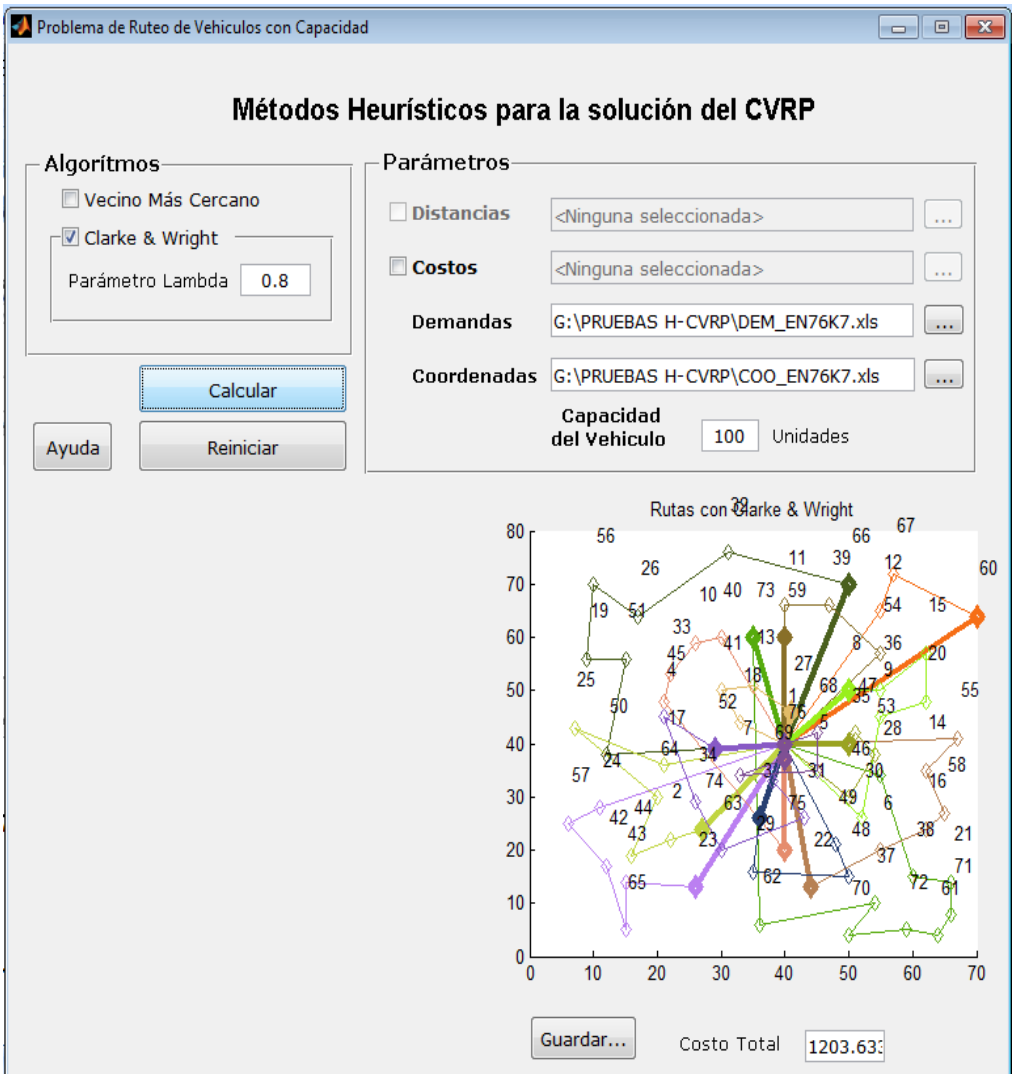


Vecino más cercano

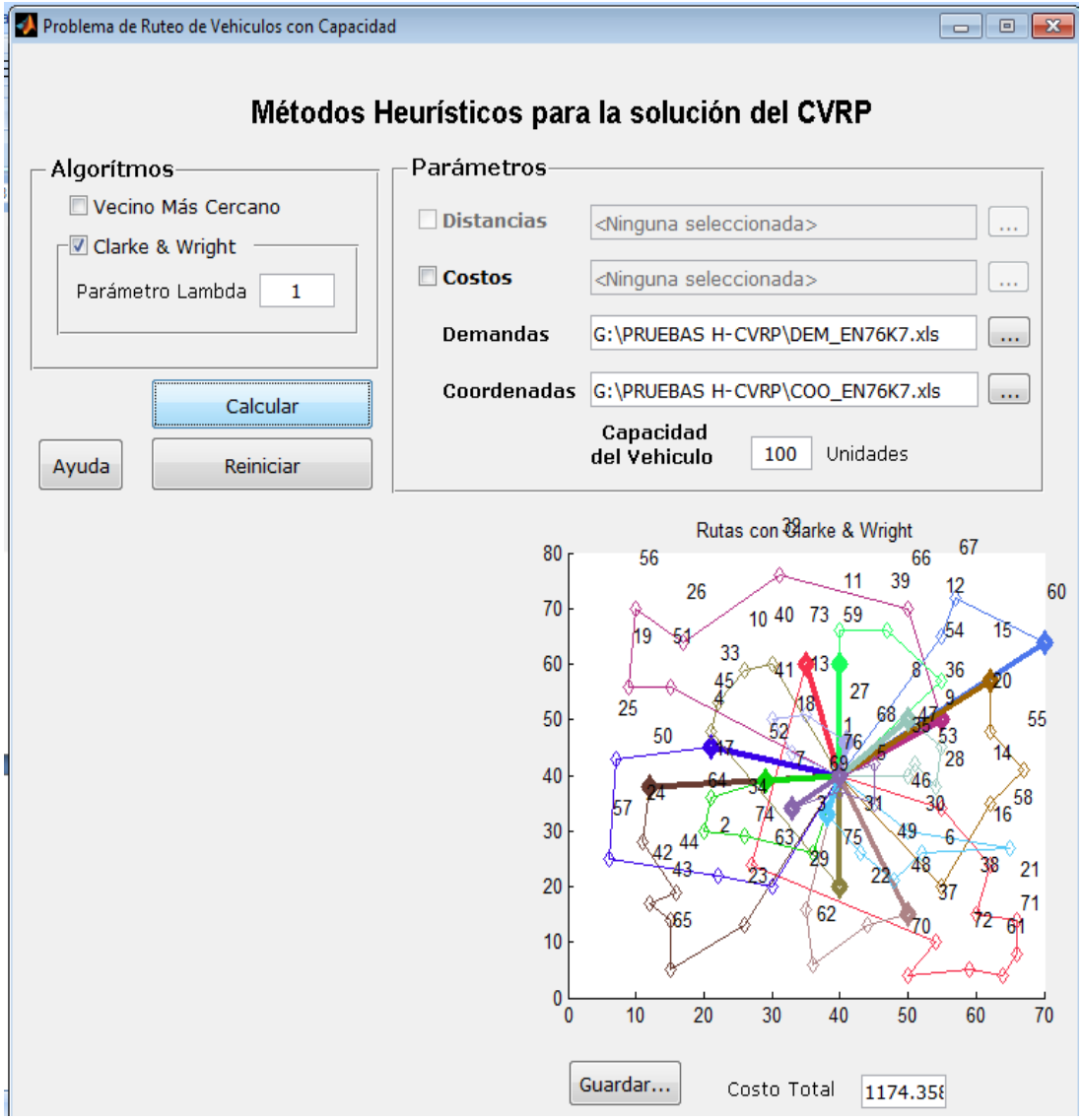


ANEXO 18. SALIDA DE H-CVRP PARA LA PRUEBA DE LA INSTANCIA E-n76-K14

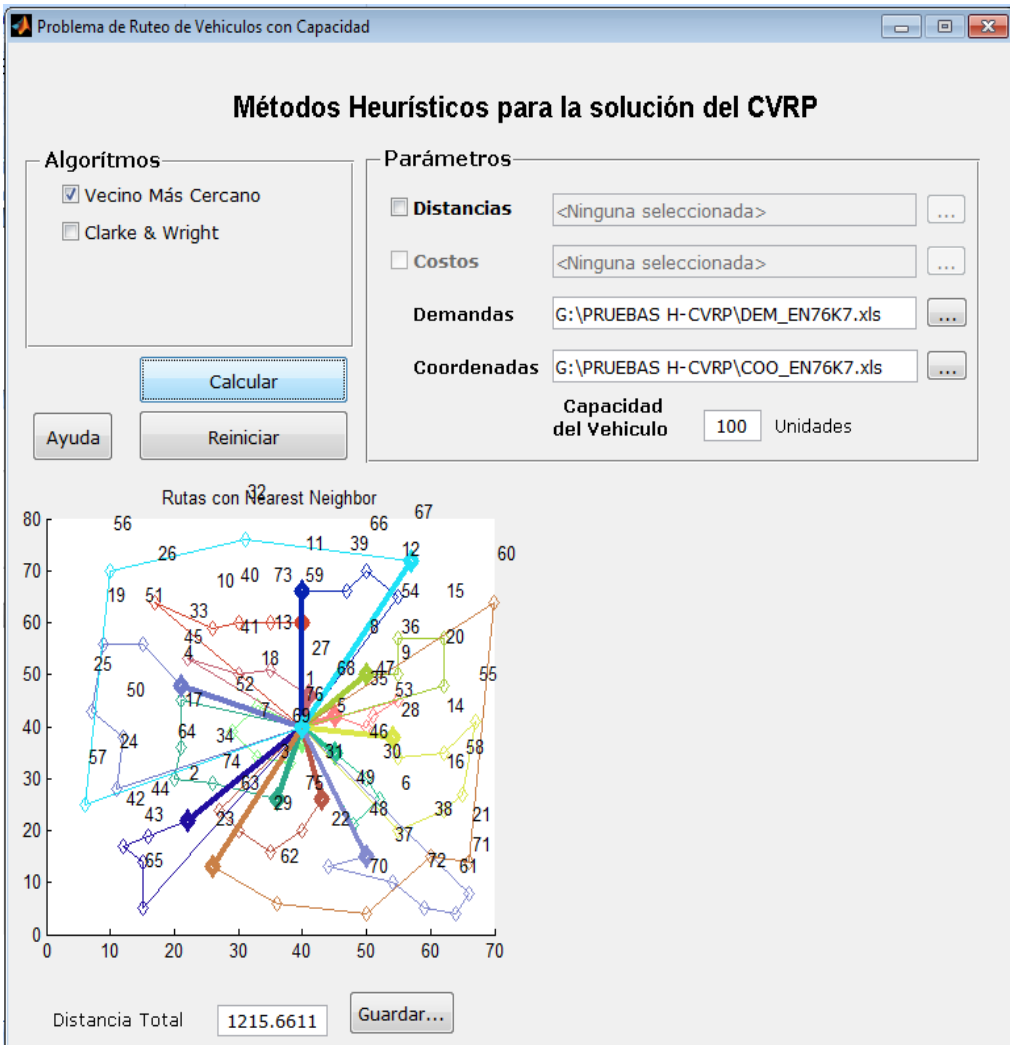
Clarke and Wright λ = 0.8



Clarke and Wright λ = 1

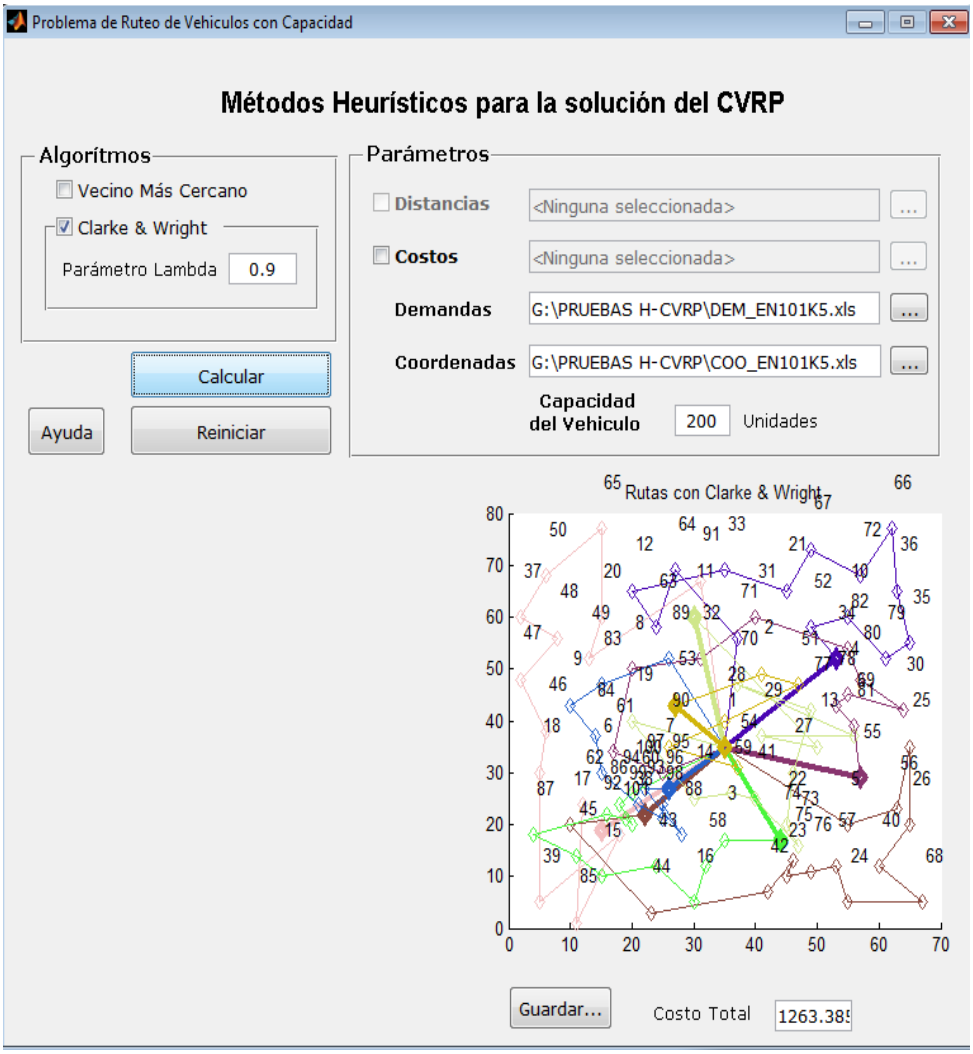


Vecino más cercano

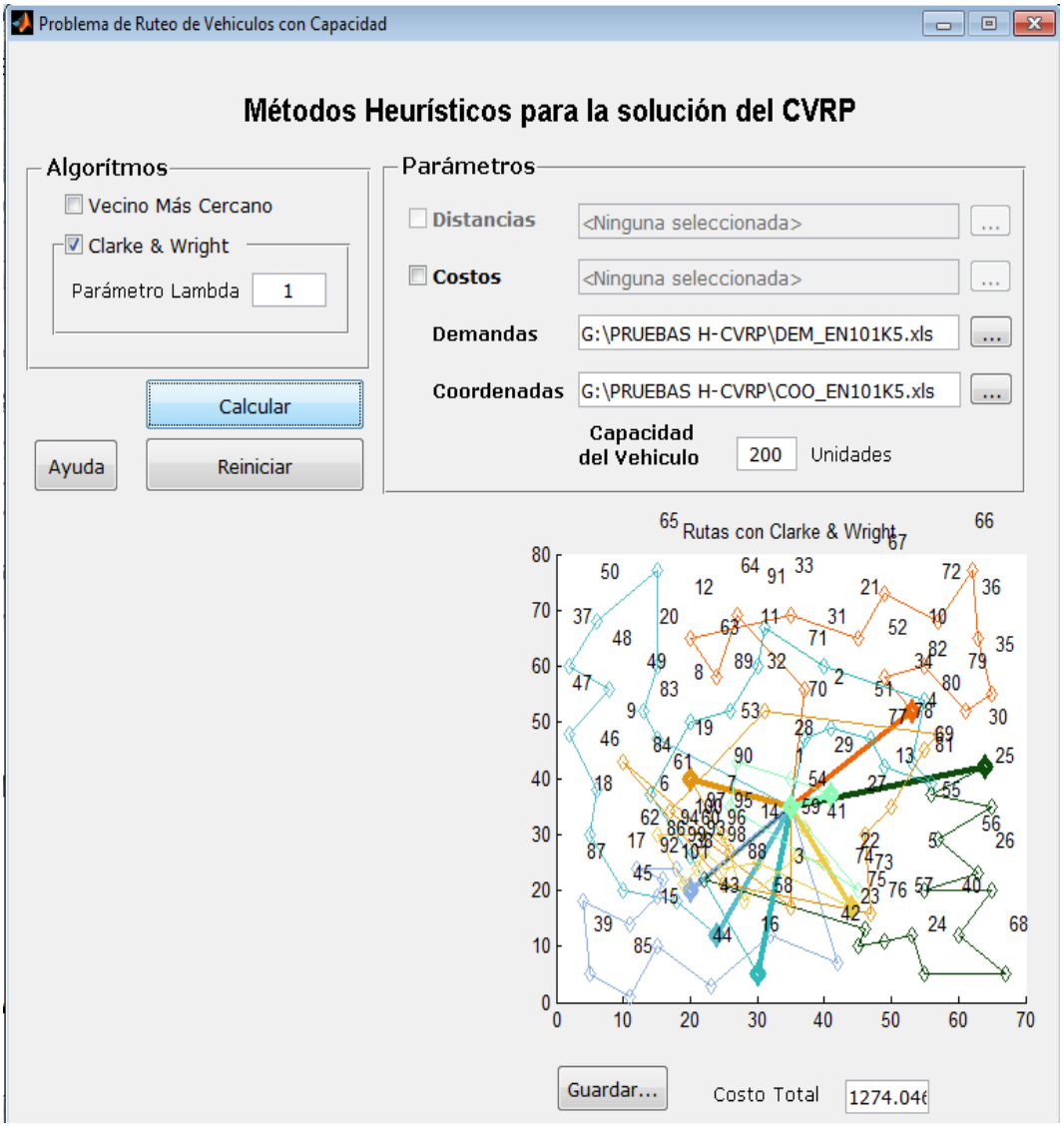


ANEXO 19. SALIDA DE H-CVRP PARA LA PRUEBA DE LA INSTANCIA E-n101-K8

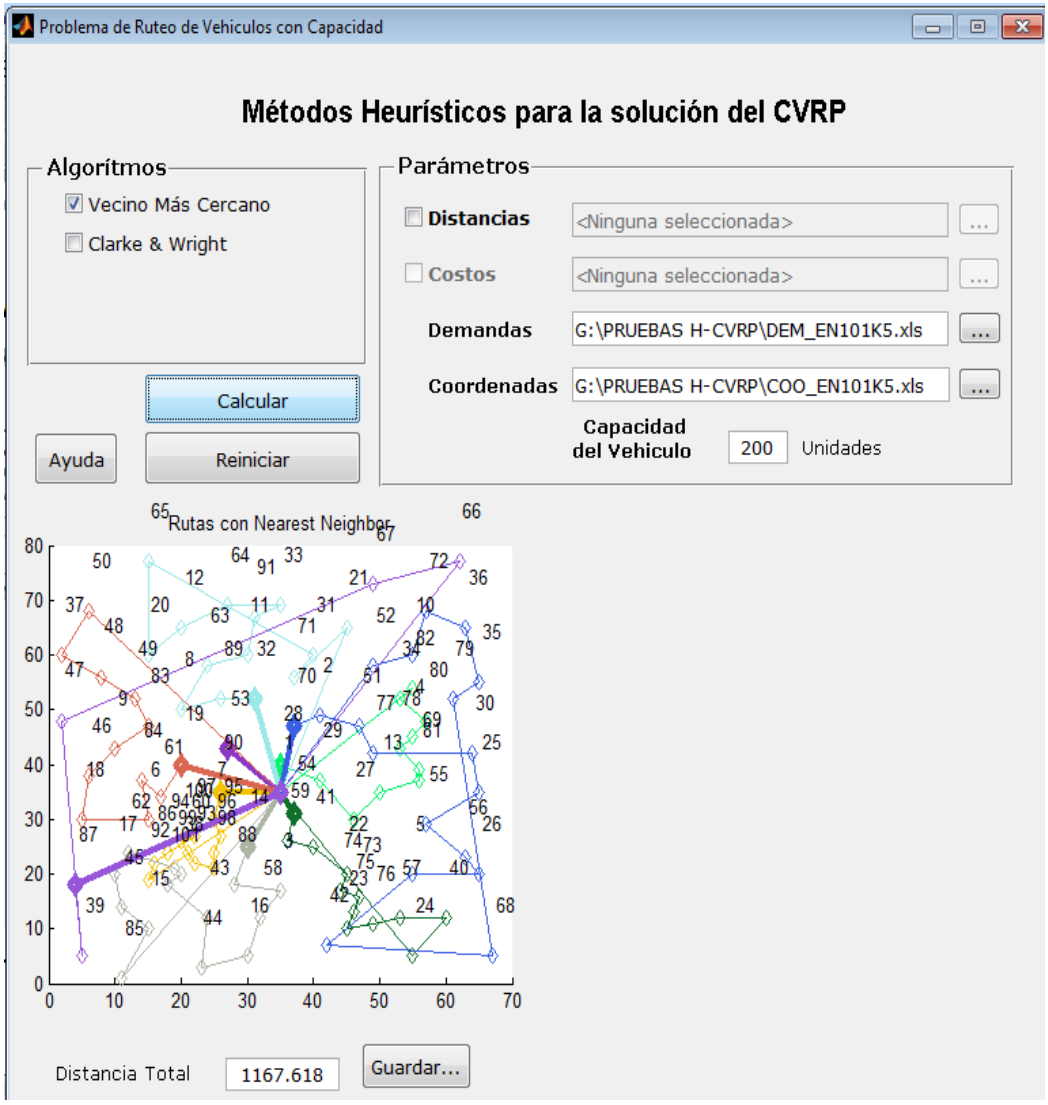
Clarke and Wright λ = 0.9



Clarke and Wright λ = 1



Vecino más cercano



ANEXO 20.CODIGO DEL PROGRAMA

- **CODIGO VECINO MÁS CERCANO**

```
function [ah,CT,L] = vecino(F,D,K,Pos)
%F es la demanda
%D es la matriz de distancia/costo/tiempo
%K es la capacidad del vehiculo
%Pos son las coordenadas
addpath([docroot '/techdoc/creating_plots/examples'])

clc
CT = 0;
Di = D; %duplique la matriz distancia para sacar las distancias.

%DEFINICIÒN DE VARIABLES
N = length(F); %Numero de clientes
V = N;
L=zeros([V,N+2]);%MATRIZ RESPUESTA, FILAS = VEHICULO, COLUMNAS =CLIENTE
i=1;%ARRANCA EN UNO ES DECIR EN LA PRIMERA FILA QUE ES EL DEPOSITO PARA
LA ELECCION DE LA MENOR DISTANCIA
a=0;% CONTADOR QUE SE COMPARA CON LAS DISTANCIAS PARA ELEGIR LA MENOR
j=2;
x=1;% PERMITE QUE EL SUBINDICE i PERMANEZCA EN UN VALOR ESTATICO PARA
LLENAR DE CEROS LA FILA QUE SE ANULA AL SER VISITADO EL CLIENTE Y PARA
INDICAR LA FILA EN LA CUAL DEBE BUSCAR EL PROXIMO CLIENTE
g=1;%GUARDA LA POSICION O EL SUBINDICE DEL CLIENTE QUE SE VA A VISITAR
s=0;% CONTADOR QUE PARA LAS ITERACIONES EN LA MATRIZ DISTANCIA, PORQUE
CADA VEZ QUE SE VISITA UNA DIAGONAL, AUMENTA EN UNO SU VALOR Y CUANDO
TIENE VALOR IGUAL A N+1 QUIERE DECIR QUE NO DEBE BUSCAR MAS CLIENTES
DENTRO DE LA MATRIZ
q=1;%SUBINDICE QUE CONTABILIZA EL VEHICULO AL QUE SE LE ESTA ASIGNANDO LA
RUTA

L(:,1) = 1;

for q=1:1:V%ELIGE EL PRIMER VEHICULO ASIGNARLE LA RUTA
    h=K;%H TOMA EL VALOR DE LA CAPACIDAD DEL VEHICULO AL QUE SE LE VA A
ASIGNAR LA RUTA

    x=1;% INDICA QUE SE DEBE EMPEZAR POR LA FILA UNO QUE ES EL DEPOSITO

    for p=2:1:N+1%RECORRE LA MATRIZ RESPUESTA POR COLUMNAS
        i=x;%i TOMA LA POSICION QUE X LE ASIGNE
        a=Inf;% SE LE DA UN VALOR DE INFINITO PARA QUE EN LA PRIMERA
ITERACION SIEMPRE TOME EL VALOR QUE Dij LE ASIGNE
        for j=2:1:(N+1)% RECORRE POR COLUMNAS LA MATRIZ DISTANCIA
```

```
            if D(i,j)==0% PREGUNTA SI ES CERO PORQUE QUIERE DCIR QUE ES
LA DIAGONAL PRINCIPAL DE LA MATRIZ O UN CLIENTE YA VISITADO QUE SE ANULO
LLENANDO DE CEROS LA COLUMNA CORRESPONDIENTE AL MISMO
                s=s+1;% CONTADOR QUE PERMITE VER CUANTAS ITERACIONES
RELACIONADAS CON LAS CASILLAS DE CERO SE HAN VISITADO
            else
                 if a>D(i,j)%PREGUNTA SI LA DISTANCIA QUE a TRAE GUARDADA
ES MAYOR QUE LA NUEVA SELECCIONADA
                   a=D(i,j);% SI a SI ES MAYOR QUE Dij ENTONCES TOMA ESTE
VALOR ES DECIR QUE SE ESTA BUSCANDO QUE a DECREZCA
                   g=j;% g GUARDA LA POSICION DE LA COLUMNA DEL CLIENTE
QUE TENIA ESTA DISTANCIA MENOR.
                 else
                     continue
                 end
            end
        end

        %si a esta altura no se da con un g distinto de 0, quiere decir
que
        %ya el problema esta resuelto y el ciclo debe morir

        if(g == 1), break; end

        if  s~=N+1% SI S ES DIFERENTE DE N ES DECIR QEU SI NO SE HA
LLENADO DE CEROS YA TODA LA MATRIZ ENTOPNCES PERMITE SEGUIR GENERANDO
NUEVAS ITERACIONES
            if h>=F(g-1)% PREGUNTA SI LA CAPACIDAD DEL VEHICULO ES MAYOR
QUE LA DEMANDA DEL CLIENTE EN EL VECTOR Fg-1 DEBIDO A QUE g TRAE
POSICIONES DE LA MATRIZ DISTANCIA QUE ESTAN CORRIDAS EN 1 A LA MATRIZ
                x=g;% X GUARDA EL VALOR DEL G QUE ES EL CLIENTE CON MENOR
DISTANCIA EN LA MATRIZ Dij PARA DARSELO MAS ADELANTE A i CON EL FIN DE
DEJAR ESTA FILA O CLIENTE DESDE DONDE SE ARRANCA LA SIGUIENTE PARTE DE LA
RUTA, PARA BUSCAR EL MAS CERCANO A ESTE
                h=h-F(g-1);%H ACTUALIZA SU CAPACIDAD DISPONIBLE.
                L(q,p)=(g);%LA MATRIZ SE LLENA CON EL NUMERO DEL CLIENTE
QUE SE VISITA

                D(:,g) = 0;% LLENA DE CEROS LA COLUMNA CON EL CLIENTE QUE
YA SE METIO EN LA MATRIS RESPUESTA

                %for  i=1:1:(N+1)% SE  MUEVEN  LAS  FILAS  PERO  NO  LAS
COLUMNAS PARA LENAR DE CEROS LA COLUMNA DEL CLIENTE SELECCIONADO PARA QUE
NO S EPUEDA VOLVER A ELEGIR
                    %j=g;%TOMA EL VALOR DE g QUE FUE EL CLIENTE QUE SE
ELIGIO CON MENOR DISTANCIA, PARA QEU SEA ESTA COLUMNA LA QUE QUEDE QUIETA
Y SE MUEVAN POR LAS FILAS LLENANDOLA ASI DE CEROS
                    %D(i,j)=0;
                %end
            else
                L(q,p)=1;%CIERRA LA RUTA ASIGANAD AL VEHICULO CON UN CERO
QUERIENDO DECIR QUE DEL ULTIMO CLEINTE SE DEVUELVE AL DEPOSITO Y SE QUED
ALLI CAMBIANDO DE VEHICULO
```

```
                s=0;  % SE REINICIA EL CONTADOR DE LOS CEROS
                g = 1;  %se reinicia el contador de vertices (g)
                break;
            end
        else
            L(q,p)=1;
             s=0;
             g = 1;
             break;
        end

    end
end

title('Rutas con Vecino Más Cercano');
axis normal

%construccion de las rutas
for r = 1:1:V

    col = [rand(),rand(),rand()];

    if(L(r,2) == 0), break; end     %para indicar que deje de armar rutas

    for cl = 1:1:N+2

        if(cl > 1 && L(r,cl) == 1)
            break;
        end

        h    =    line([Pos(L(r,cl),1)    Pos(L(r,cl+1),1)],[Pos(L(r,cl),2)
Pos(L(r,cl+1),2)],'Marker','d','Color',col);

        %di      =      sqrt(abs(Pos(L(r,cl),1)-Pos(L(r,cl+1),1))^2      +
abs(Pos(L(r,cl),2)-Pos(L(r,cl+1),2))^2);

        di = Di(L(r,cl),L(r,cl+1));

        CT = CT + di;

        if (cl == 1)
            set(h,'LineWidth',3.7);
        end
    end

end

ah = zeros(1,N+1);

for sedes = 1:1:N+1
```

```
    [figx     figy]     =     dsxy2figxy([Pos(sedes,1)     Pos(sedes,2)-
0.4],[Pos(sedes,1)+1 Pos(sedes,2)+1.6]);

    if (figx(1) > figy(1)) && (figx(2) > figy(2))
        temp = figy;
        figy = figx;
        figx = temp;
    end

    ah(sedes)     =     annotation('textbox',[abs(figx(1))     abs(figx(2))
abs(figy(1)-figx(1))                                          abs(figy(2)-
figx(2))],'String',num2str(sedes),'LineStyle','none','FontSize',8.0);

end
```

- **CODIGO ALGORITMO DE AHORROS**

```
function [a,CT,M] = algoaho(D,C,lambda,K,Pos)
%OUTPUTS:
%a son los handles para las etiquetas de los nodos
%CT es el resultado del problema (supuesto valor otimo)
%M es la matriz con las rutas y el costo de cada ruta

%INPUTS:
%D: demandas
%C: matriz costos-distancias
%lambda: valor del parametro Lambda entre [0.0,1.0]
%K: capacidad vehiculo
%Pos: coordenadas XY

addpath([docroot '/techdoc/creating_plots/examples'])

%DEFINICIÒN DE VARIABLES
N=length(C)-1;% Saca la magnitud de la matriz costos y le resta uno
porque en esa matriz estaba incluido el deposito
L=zeros([3,((factorial(N))/(factorial(N-2)))]);%matriz de resultados
A=zeros(N+1,N+1);%vector que guarda los ahorros
G=D(1);%variable que empieza con el valor de la demanda 1
V=N;%numero de vehiculos

H=K;% GUARDA EL VALOR DE LA CAPACIDAD
T=0;%VARIABLE QUE TOMA EL VALOR DEL CLIENTE PARA MIRAR SU DEMANDA
U=0;%IGUAL QUE T
X=0;
Y=0;
P=0;
Q=0;
H=0;

%CALCULO DE LOS AHORROS con base en DISTANCIAS
```

```
for i=2:1:N+1% comienza a recorrer las filas desde la numero 2 ya que la numero 1 son los costos del deposito
    for j=2:1:N+1% comienza a recorrer las columnas desde la numero 2 ya que la numero 1 son los costos del deposito
        if i~=j% si i es diferente de j si se calcula el ahorro
            A(i,j)=C(i,1)+C(1,j)-(lambda*C(i,j));%CALCULO DEL AHORRO
        end
    end
end

for  S=1:1:((factorial(N))/(factorial(N-2)))  %CONTADOR QUE DETERMINA CUANTAS CASILLAS DE LA MATRIZ RESULTADO SE HAN LENADO
    B=0;        %vuelve cero la variable B
    for i=2:1:N+1
        for j=2:1:N+1
            %si no es cero hace el procedimiento, si es cero esporque es la fila 1, la columna uno o la diagonal principal, o alguno que ya se haya anulado
            if A(i,j)~=0 && B<A(i,j)
                % compara la variable B con el ahorros calculado y verifica que sea mayor
                B=A(i,j);    %B TOMA EL VALOR DEL AHORRO MAS GRANDE QUE HAYA ENCONTRADO
                P=i;         %guarda la posicion del nodo de salida de la union con el ahorro mas grande hasta el momento
                Q=j;        %guarda la posicion del nodo de entrada de la union con el ahorro mas grande hasta el momento
            end
        end
    end
    L(1,S)=P;   %LA FILA 1 DEL VECTOR RESULTADO INDICA EL NODO DE SALUDA DE LA UNION
    L(2,S)=Q;   %LA FILA DOS INDICA EL VECTOR AL QUE LLEGA LA UNION
    A(P,Q)=0;   %vuelve cero el ahorro que ya asigno a la matriz L
end

for g=1:1:V% SE EMPIEZA POR EL VEHICULO 1
    H=K;% H toma el valor de K
    numaris=0;% variable que determina el numero de aristas empieza en cero
    for i=1:1:((factorial(N))/(factorial(N-2)))
        T=L(1,i);%T guarda el numero del cliente de salida de la union
        U=L(2,i);%U guarda el cliente de llegada en la unionT=L(1,i);
        if L(3,i)==0% si en la fila 3 la columna correspondiente a la union que se esta analizando es cero puede continuar
            if numaris==0% si el numero de aristas es cero quiere decir que no se han hecho nuevas uniones
                if D(T-1)+D(U-1)>H% como no se han hecho uniones se suman las demandas del nodo de salida y llegada de la union inicial y se verifica si son menores que la capacidad del vehiculo
                    L(3,i)=V+1;% si es mayor que la capacidad del vehiculo se pone en la fila 3 y en la columna de la union analizada el numero que corresponde a V+1 significa restriccion de capacidad
                    %AL CONTRARIO LE ASIGNA V+2
```

```
                    for  j=1:1:((factorial(N))/(factorial(N-2)))% recorre
la matriz L
                        if L(1,j)==U && L(2,j)==T%busca en la matriz L el
nodo contrario al que no cumplio con la capacidad
                            L(3,j)=V+2;    %pone    en    la    casilla
correspondiente al nodo contrario el numero V+1 que corresponde a la
restriccion de capacidad
                            break;        %cuando encuentra la contrario
como solo hay uno entonces se sale del for
                            %j=((factorial(N))/(factorial(N-2)));
                        end
                    end
                else %cuando no se han hecho uniones y la suma de las
demandas de los nodos de la union es menor o igual que la capacidad del
vehiculo entonces se le asigna un un vehiculo a dicha union
                    L(3,i)=g;% se pone el numero del vehiculo en la
casilla que corresponda a la union que se valido
                    numaris=numaris+1;% numeris incrementa en uno
                    H=H-D(T-1)-D(U-1);%   se   actualiza   la   capacidad
disponible del vehiculoelse
                    for j=1:1:((factorial(N))/(factorial(N-2)))
                        if L(1,j)==U && L(2,j)==T% busca el contrario del
que se asigno
                            L(3,j)=V+2;% le pone V+2 como restriccion de
contrario al asignado
                            break % cuando ha encontrado el contrario
deja de buscar saliendose del for
                            %j=((factorial(N))/(factorial(N-2)));
                        end
                    end
                    for j=1:1:((factorial(N))/(factorial(N-2)))
                        if L(1,j)==T && L(3,j)==0%busca todas las uniones
que salgan del nodo que se asigno para anularlas ya que no pueden salir
dos rutas de un mismo nodo
                           L(3,j)=V+3;%  pone  la  restriccion  V+3  que
significa que es restriccion de nodo ya visitado
                        end

                        if L(2,j)==U && L(3,j)==0% busca las uniones que
lleguen a nodo que ya se asigno para anularlas debido a que solo puede
llegar una ruta a cada cliente
                            L(3,j)=V+3;%  pone  la  restriccion  V+3  que
significa que es restriccion de nodo ya visitado
                        end
                    end
                end
            else
            %ignora la ruta si forma ciclo
                ao=0;
                bo=0;
                for az=1:1:((factorial(N))/(factorial(N-2)))
                    if  L(2,az)==T  &&  L(3,az)==g%busca  en  L  uniones  q
hayan sido asignadas al vehiculo que puedan formar ciclo
```

```
                        ao=ao+1;% si encuentra posibles ciclos ao aumenta
en uno
                    end
                    if L(1,az)==U && L(3,az)==g%busca en la fila 1 el
nodo de llegada de la ruta que ya se asigno y que además tenga en la fila
3  el mismo vehiculo asigando y lo anula porque forma ciclo
                        bo=bo+1;% cuando encuentra posible ciclo aumenta
en uno bo
                    end
                end
                if ao>0 && bo>0% si ambos son mayores que cero fue porque
se encontraron ciclos
                    L(3,i)=V+3;% anula con V+3 la fila 3 que corresponde
a la union prohibida
                else
                 % la arista no forma ciclo y puede ser asignada a una
ruta ya comenzada
                    for k=1:1:(i-1)
                        if L(1,k)==U && L(3,k)==g% busca en L una union
que empiece por el nodo de salida de la casilla que se esta revisando
(T,U) y que además ya tenga asignado el mismo vehiculo
                            if (H-D(T-1))>=0% verifica que se pueda hacer
la union
                                numaris=numaris+1;% si se hizo la union
numaris aumenta
                                L(3,i)=g;% se le aigna a la union el
vehiculo de la ruta
                                H=H-D(T-1);% se actualiza la capacidad
del vehiculo
                                for    j=1:1:((factorial(N))/(factorial(N-
2)))
                                    if L(1,j)==U && L(2,j)==T% busca el
contrario
                                        L(3,j)=V+2; % anula el contrario
                                        break;      % cuando ya encuentra
el contraro termina la busqueda saliendose del for
                                        %j=((factorial(N))/(factorial(N-
2)));
                                    end
                                end
                                for    j=1:1:((factorial(N))/(factorial(N-
2)))
                                    if L(1,j)==T && L(3,j)==0%busca las
uniones en L que lleguen
                                       L(3,j)=V+3;
                                    end

                                    if L(2,j)==U && L(3,j)==0
                                       L(3,j)=V+3;
                                    end
                                end
                                break;
                                %k=i-1;
                            else
```

```
                                L(3,i)=V+1;
                            end
                        elseif L(2,k)==T && L(3,k)==g
                            if (H-D(U-1))>=0
                                numaris=numaris+1;
                                L(3,i)=g;
                                H=H-D(U-1);
                                for    j=1:1:((factorial(N))/(factorial(N-
2)))
                                    if L(1,j)==U && L(2,j)==T
                                        L(3,j)=V+2;
                                        break;
                                    end
                                end
                                for    j=1:1:((factorial(N))/(factorial(N-
2)))
                                    if L(1,j)==T && L(3,j)==0
                                       L(3,j)=V+3;
                                    end

                                    if L(2,j)==U && L(3,j)==0
                                       L(3,j)=V+3;
                                    end
                                end
                                break;
                                %k=i-1;
                            else
                                L(3,i)=V+1;
                            end
                        end
                    end
                end
            end
        end
    end
    for i=1:1:((factorial(N))/(factorial(N-2)))
        if L(3,i)==g
            Ta=L(1,i);
            Ua=L(2,i);
            for j=1:1:((factorial(N))/(factorial(N-2)))
                if i~=j && L(3,j)==0
                    if   L(1,j)==Ta   ||   L(1,j)==Ua   ||   L(2,j)==Ta   ||
L(2,j)==Ua
                        L(3,j)=V+4;
                    end
                end
            end
        end
    end
end

%por aca debiera reducirse el numero de V dependiendo de lo observado en
%L(3,:), que puede tomar el valor de algunos vehiculos y luego los
```

```
%triggers V+1, V+2, V+3, V+4. Habria que buscar el mayor de los valores
que
%sean menores al V inicial y dimensionar M segun el valor que se pille

v = 0;

for fv = 1:1:length(L)
    if L(3,fv) > V
        continue
    end

    if v < L(3,fv)
       v = L(3,fv);
    end

end

%V = v;

M=zeros([V,(N+2)]);%matriz  fila=vehiculo,  columnas=nodo  a  visitar,  el
primero y el ultimo simpre es 1
M(:,1)=1;   %llena con el numero uno la primera posicion de la matriz en
cada fila

for j=1:1:V
    z=1;
    for k=1:1:(factorial(N))/(factorial(N-2))
        ao=0;
        if L(3,k)==j
            for  i=1:1:(factorial(N))/(factorial(N-2))% de los que están
en una ruta de más de un cliente, cual no está conectado con otro
               if L(3,i)==j && L(1,k)==L(2,i)
                    ao=ao+1;
               end
            end
            if ao==0 && L(3,k)==j %es la conexión con el origen
                  z=z+1;
                  M(j,z)=L(1,k);
                  z=z+1;
                  M(j,z)=L(2,k);
                  break;
            end
        end
    end
    %???
       for k=z:1:N+1%Coloca el resto de nodos de la ruta donde hay más de
un cliente
         for i=1:1:(factorial(N))/(factorial(N-2))
             if L(3,i)==j && L(1,i)==M(j,z)
                z=z+1;
                M(j,z)=L(2,i);
                break;
             end
```

```
        end
      end
    %end
    if z==1
        %Entonces es una ruta de un solo cliente
        for co=2:1:N+1
            do=0;
            for pp=1:1:V
               for i=1:1:N+2
                   if M(pp,i)==co
                       do=1;
                       %pp=V;
                       break;
                   end
               end

               if(do == 1)
                   break;
               end
            end

            if do==0
                z=z+1;
                M(j,z)=co;
                z=z+1;
                M(j,z)=1;
                j=j+1; %continue??
                z=1;
                break;
            end
        end
    else
        z=z+1;
        M(j,z)=1;
    end
 end

 % CALCULO DEL COSTO TOTAL DE LAS RUTAS
CT=0;
CR=0;
for k=1:1:V
    for i=1:1:N+1
        if  M(k,i+1)==0% si la casilla es cero, deja de sumar costos
saliendose del for que recorre esa fila y pasa a la siguiente a sumar la
proxima ruta
            break;% salida del for
        else
            r=M(k,i);%guarda el numero del cliente i
            s=M(k,i+1);%guarda el numero del cliente j
            CR=CR+C(r,s);%actualiza la variable CR(costo de la ruta)
sumandole el costo de la ruta i,j que se encuentra en la matriz C (matriz
de costos inicial)
        end
    end
```

```
    for j=2:1:N+2
        if M(k,j)==1
            M(k,j+1)=CR;%guarda al finalizar la ruta el costo total de la
misma
        end
    end
    COS(k)=CR;%guarda en un vector aparte los costos totales de cada una
de las rutas
    CT=CT+CR;%costo total
    CR=0;%costo de la ruta
end

%elaboracion del grafo

%leg = legend('show');

%set(leg,'String',rutas);

title('Rutas con Clarke & Wright');
for nM = 1:1:V

    col = [rand(),rand(),rand()];

    for nC = 1:1:N+3

        if (nC > 1 && M(nM,nC) <= 1) || M(nM,nC+1) == 0
            break;  %con esto se termina el for
        end

        %graficar la linea recta que une los clientes

        h   =   line([Pos(M(nM,nC),1)   Pos(M(nM,nC+1),1)],[Pos(M(nM,nC),2)
Pos(M(nM,nC+1),2)],'Marker','d','Color',col);
        %rutas =
        %di     =     sqrt(abs(Pos(M(nM,nC),1)-Pos(M(nM,nC+1),1))^2     +
abs(Pos(M(nM,nC),2)-Pos(M(nM,nC+1),2))^2);

        if(nC == 1)
            set(h,'LineWidth',3.7);
        end

    end

end

a = zeros(1,N+1);

for sedes = 1:1:N+1

    [figx    figy]    =    dsxy2figxy([Pos(sedes,1)-0.5    Pos(sedes,2)-
1.7],[Pos(sedes,1) Pos(sedes,2)-1]);
```

```
    a(sedes)     =     annotation('textbox',[figx(1)     figy(2)     0.04
0.07],'String',num2str(sedes),'LineStyle','none','FontSize',8.0);

end
```

- **CÒDIGO CVRP**

```
function varargout = CVRP(varargin)
% CVRP M-file for CVRP.fig
%      CVRP, by itself, creates a new CVRP or raises the existing
%      singleton*.
%
%      H = CVRP returns the handle to a new CVRP or the handle to
%      the existing singleton*.
%
%      CVRP('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CVRP.M with the given input arguments.
%
%      CVRP('Property','Value',...) creates a new CVRP or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before CVRP_OpeningFcn gets called.  An
%          unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to CVRP_OpeningFcn via varargin.
%
%       *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help CVRP

% Last Modified by GUIDE v2.5 11-Jul-2010 20:26:01

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @CVRP_OpeningFcn, ...
                   'gui_OutputFcn',  @CVRP_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before CVRP is made visible.
function CVRP_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to CVRP (see VARARGIN)

% Choose default command line output for CVRP
handles.output = hObject;

handles.ahCW = [];
handles.ahNN = [];

handles.rutasCW = [];
handles.rutasNN = [];

% Update handles structure
guidata(hObject, handles);

pushReset_Callback(hObject,eventdata,handles);

% UIWAIT makes CVRP wait for user response (see UIRESUME)
% uiwait(handles.costosCW);


% --- Outputs from this function are returned to the command line.
function varargout = CVRP_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushCalcule.
function pushCalcule_Callback(hObject, eventdata, handles)
% hObject    handle to pushCalcule (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if(get(handles.chkAhorros,'Value')           ==           0           &&
get(handles.chkVecinito,'Value') == 0)
    errordlg('Seleccione al menos uno de los algoritmos a evaluar');
```

```
    return
end

K = str2double(get(handles.K,'String'));

if(K <= 0 || isnan(K))
    errordlg('La capacidad del Vehiculo debe ser un numero mayor de 0');
    return
end

if(strcmp(get(handles.rutaDemandas,'Enable'),'off'))
    errordlg('No se ha seleccionado el archivo de Demandas');
    return
end

D = xlsread(get(handles.rutaDemandas,'String'));

if(get(handles.chkAhorros,'Value') == 1)

    if(get(handles.chkCostos,'Value')==1.0)

        if(strcmp(get(handles.rutaCostos,'Enable'),'off'))
            errordlg('No se ha seleccionado el archivo con la matriz de
costos');
            return
        end

        C = xlsread(get(handles.rutaCostos,'String'));
        %si se especifica usar una matriz de costos, se reemplaza la nula
por
        %la que este almacenada en el archivo especificado

        if(length(C) ~= (length(D)+1))
            errordlg('Ejercicio incorrecto: no coincide el numero de
clientes que reportan las matrices de demanda y datos de entrada');
            return
        end

        if get(handles.chkCoordenadas,'Value') == 0
            P = 70*rand(length(C),2);
        else
            P = xlsread(get(handles.rutaCoordenadas,'String'));
        end

    else
        %arme matriz de Costo/Distancia tomando en cuenta los valores de
las
        %coordenadas
        if get(handles.chkCoordenadas,'Value') == 0
            errordlg('No se ha seleccionado el archivo de excel con las
coordenadas de los nodos y del deposito','Error en el metodo de Clarke &
Wright');
```

```
            return;
        end

        P = xlsread(get(handles.rutaCoordenadas,'String'));
        C = zeros(length(D)+1);      %al inicio se asume una matriz de
distancia/costo llena de ceros

        for i = 1:1:length(C)
            for j = i:1:length(C)

                if i == j
                    continue;
                end

                C(i,j)   =   sqrt(abs(P(i,1)-P(j,1))^2   +   abs(P(i,2)-
P(j,2))^2);
                C(j,i) = C(i,j);

            end
        end

    end

     lambda = str2double(get(handles.txtLambda,'String'));

     if(isnan(lambda) || (lambda < 0.0))
        errordlg('El parametro Lambda debe ser un numero real mayor de
0','Error en el metodo de Clarke & Wright');
        return
     end

    for na = 1:1:length(handles.ahCW)
        delete(handles.ahCW(na));
        handles.ahCW(na) = 0;
    end

    cla(handles.graphCW);
    set(gcf,'CurrentAxes',handles.graphCW);  %selecciona  el  plot  para
Ahorros a fin de trazar las rutas
    [handles.ahCW,ctCW,handles.rutasCW] = algoaho(D,C,lambda,K,P);
    set(handles.costosCW,'String',num2str(ctCW));
    set(handles.pushSaveCW,'Enable','on');
end

if(get(handles.chkVecinito,'Value') == 1)

    if(get(handles.chkDistancias,'Value')==1)

        if(strcmp(get(handles.rutaDistancias,'Enable'),'off'))
            errordlg('No se ha seleccionado el archivo con la matriz de
distancias');
            return
```

```
        end

        Di = xlsread(get(handles.rutaDistancias,'String'));
        %si se especifica usar una matriz de costos, se reemplaza la nula
por
        %la que este almacenada en el archivo especificado

        if(length(Di) ~= (length(D)+1))
            errordlg('Ejercicio incorrecto: no coincide el numero de
clientes que reportan las matrices de demanda y datos de entrada');
            return
        end

        if get(handles.chkCoordenadas,'Value') == 0
            P = 70*rand(length(Di),2);
        else
            P = xlsread(get(handles.rutaCoordenadas,'String'));
        end

    else

        if get(handles.chkCoordenadas,'Value') == 0
            errormsg('No se ha seleccionado el archivo de excel con las
coordenadas de los nodos y del deposito','Error en el metodo del Vecino
mas Cercano');
            return;
        end

        P = xlsread(get(handles.rutaCoordenadas,'String'));

        Di = zeros(length(D)+1);

        for i = 1:1:length(Di)
            for j = i:1:length(Di)

                if i == j
                    continue;
                end

                Di(i,j)   =   sqrt(abs(P(i,1)-P(j,1))^2   +   abs(P(i,2)-
P(j,2))^2);
                Di(j,i) = Di(i,j);

            end
        end

    end

    for na = 1:1:length(handles.ahNN)
        delete(handles.ahNN(na));
        handles.ahNN(na) = 0;
    end
```

```
    cla(handles.graphNN);
    set(gcf,'CurrentAxes',handles.graphNN);  %selecciona  el  plot  para
Nearest Neighbor a fin de trazar las rutas
    [handles.ahNN,diNN,handles.rutasNN] = vecino(D,Di,K,P);
    set(handles.distNN,'String',num2str(diNN));
    set(handles.pushSaveNN,'Enable','on');
end

guidata(hObject,handles);

if(length(P) > (length(D)+1))
    helpdlg(['La  matriz  de  coordenadas  tiene  mas  pares  que  nodos.  Se
utilizaron los primeros ' num2str(length(D)+1) ' pares'],'Calculo exitoso
con el siguiente aviso');
end

% --- Executes on button press in chkVecinito.
function chkVecinito_Callback(hObject, eventdata, handles)
% hObject    handle to chkVecinito (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chkVecinito

if(get(hObject,'Value') == 1)
    set(handles.graphNN,'Visible','on');
    set(handles.pushSaveNN,'Visible','on');
    set(handles.distNN,'Visible','on');
    set(handles.textDistNN,'Visible','on');
    set(handles.chkDistancias,'Enable','on');
    set(handles.chkCoordenadas,'Value',1);
    set(handles.chkCoordenadas,'Enable','inactive');

    chkCoordenadas_Callback(handles.chkCoordenadas,eventdata,handles);
else
    cla(handles.graphNN,'reset');
    set(handles.chkDistancias,'Enable','off');
    set(handles.chkDistancias,'Value',0);
    set(handles.graphNN,'Visible','off');
    set(handles.pushSaveNN,'Visible','off');
    set(handles.pushSaveNN,'Enable','off');
    set(handles.distNN,'Visible','off');
    set(handles.textDistNN,'Visible','off');

    for cl = 1:1:length(handles.ahNN)
        delete(handles.ahNN(cl));
    end

    handles.ahNN = [];
    handles.rutasNN = [];
```

```
    guidata(hObject,handles);

    chkDistancias_Callback(hObject,eventdata,handles);
    set(handles.chkCoordenadas,'Enable','on');
end



% --- Executes on button press in chkAhorros.
function chkAhorros_Callback(hObject, eventdata, handles)
% hObject    handle to chkAhorros (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chkAhorros

if(get(hObject,'Value') == 1)
    set(handles.panelLambda,'Visible','on');
    set(handles.graphCW,'Visible','on');
    set(handles.costosCW,'Visible','on');
    set(handles.textCostoCW,'Visible','on');
    set(handles.pushSaveCW,'Visible','on');
    set(handles.chkCostos,'Enable','on');
    set(handles.chkCoordenadas,'Value',1);
    set(handles.chkCoordenadas,'Enable','inactive');

    chkCoordenadas_Callback(handles.chkCoordenadas,eventdata,handles);

else
    cla(handles.graphCW,'reset');
    set(handles.chkCostos,'Enable','off');
    set(handles.chkCostos,'Value',0);
    set(handles.graphCW,'Visible','off');
    set(handles.panelLambda,'Visible','off');
    set(handles.pushSaveCW,'Visible','off');
    set(handles.pushSaveCW,'Enable','off');
    set(handles.costosCW,'String','0.0');
    set(handles.costosCW,'Visible','off');
    set(handles.textCostoCW,'Visible','off');

    for cl = 1:1:length(handles.ahCW)
        delete(handles.ahCW(cl));
    end

    handles.ahCW = [];
    handles.rutasCW = [];

    guidata(hObject,handles);

    chkCostos_Callback(handles.chkCostos,eventdata,handles);
    set(handles.chkCoordenadas,'Enable','on');
end
```

```
function txtLambda_Callback(hObject, eventdata, handles)
% hObject    handle to txtLambda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of txtLambda as text
%        str2double(get(hObject,'String')) returns contents of txtLambda
as a double


% --- Executes during object creation, after setting all properties.
function txtLambda_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtLambda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if        ispc        &&        isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function rutaCostos_Callback(hObject, eventdata, handles)
% hObject    handle to rutaCostos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of rutaCostos as text
%        str2double(get(hObject,'String')) returns contents of rutaCostos
as a double



% --- Executes during object creation, after setting all properties.
function rutaCostos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rutaCostos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if        ispc        &&        isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in FindCostos.
function FindCostos_Callback(hObject, eventdata, handles)
% hObject    handle to FindCostos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[fileCostos,rutaCostos]   =   uigetfile('*.xls','Seleccione   la   hoja   de
calculo con la matriz de costos');

if isequal(fileCostos,0)
    return;
end

set(handles.rutaCostos,'String',[rutaCostos fileCostos]);
set(handles.rutaCostos,'Enable','on');


function rutaDemandas_Callback(hObject, eventdata, handles)
% hObject    handle to rutaDemandas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of rutaDemandas as text
%                 str2double(get(hObject,'String'))  returns  contents  of
rutaDemandas as a double


% --- Executes during object creation, after setting all properties.
function rutaDemandas_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rutaDemandas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty  -  handles  not  created  until  after  all  CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function rutaCoordenadas_Callback(hObject, eventdata, handles)
% hObject    handle to rutaCoordenadas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints:  get(hObject,'String')  returns  contents  of  rutaCoordenadas  as
text
```

```
%                str2double(get(hObject,'String'))  returns  contents  of
rutaCoordenadas as a double



% --- Executes during object creation, after setting all properties.
function rutaCoordenadas_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rutaCoordenadas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty  -  handles  not  created  until  after  all  CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function K_Callback(hObject, eventdata, handles)
% hObject    handle to K (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of K as text
%          str2double(get(hObject,'String')) returns contents of K as a
double


% --- Executes during object creation, after setting all properties.
function K_CreateFcn(hObject, eventdata, handles)
% hObject    handle to K (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty  -  handles  not  created  until  after  all  CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushFindD.
function pushFindD_Callback(hObject, eventdata, handles)
% hObject    handle to pushFindD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
[xlsDemanda,rutaDemanda]   =   uigetfile('.xls','Seleccione   la   hoja   de
calculo con la demanda de los clientes');

if isequal(xlsDemanda,0)
    return;
end

set(handles.rutaDemandas,'String',[rutaDemanda xlsDemanda]);
set(handles.rutaDemandas,'Enable','on');

% --- Executes on button press in pushFindXY.
function pushFindXY_Callback(hObject, eventdata, handles)
% hObject    handle to pushFindXY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[fileXY,rutaXY] = uigetfile('*.xls','Seleccione la hoja de calculo con
las coordenadas');

if isequal(fileXY,0)
    return;
end

set(handles.rutaCoordenadas,'String',[rutaXY fileXY]);
set(handles.rutaCoordenadas,'Enable','on');


% --- Executes on button press in pushReset.
function pushReset_Callback(hObject, eventdata, handles)
% hObject    handle to pushReset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

nullPath = '<Ninguna seleccionada>';

set(handles.rutaCostos,'String',nullPath);
set(handles.rutaCostos,'Enable','off');
set(handles.rutaDemandas,'String',nullPath);
set(handles.rutaDemandas,'Enable','off');
set(handles.rutaCoordenadas,'String',nullPath);
set(handles.rutaCoordenadas,'Enable','off');
set(handles.rutaDistancias,'String',nullPath);
set(handles.rutaDistancias,'Enable','off');
set(handles.K,'String','0');
set(handles.txtLambda,'String','0.0');
set(handles.FindCostos,'Enable','off');
set(handles.FindDistancias,'Enable','off');
set(handles.pushFindXY,'Enable','off');
set(handles.chkVecinito,'Value',0);
set(handles.chkAhorros,'Value',0);

set(handles.chkDistancias,'Value',0);
```

```
set(handles.chkCostos,'Value',0);
set(handles.chkCoordenadas,'Value',0);
set(handles.chkDistancias,'Enable','off');
set(handles.chkCostos,'Enable','off');
set(handles.chkCoordenadas,'Enable','on');

set(handles.textCostoCW,'Visible','off');
set(handles.textDistNN,'Visible','off');

set(handles.costosCW,'Visible','off');
set(handles.costosCW,'String','0.0');
set(handles.distNN,'Visible','off');
set(handles.distNN,'String','0.0');

set(handles.pushSaveNN,'Visible','off');
set(handles.pushSaveNN,'Enable','off');
set(handles.pushSaveCW,'Visible','off');
set(handles.pushSaveCW,'Enable','off');

for cl = 1:1:length(handles.ahNN)
    delete(handles.ahNN(cl));
end

for cl = 1:1:length(handles.ahCW)
    delete(handles.ahCW(cl));
end

handles.ahNN = [];
handles.ahCW = [];
handles.rutasNN = [];
handles.rutasCW = [];

cla(handles.graphNN,'reset');
cla(handles.graphCW,'reset');
set(handles.graphNN,'Visible','off');
set(handles.graphCW,'Visible','off');
set(handles.panelLambda,'Visible','off');

guidata(hObject,handles);


function costoNN_Callback(hObject, eventdata, handles)
% hObject    handle to costoNN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of costoNN as text
%        str2double(get(hObject,'String')) returns contents of costoNN as
a double


% --- Executes during object creation, after setting all properties.
```

```
function costoNN_CreateFcn(hObject, eventdata, handles)
% hObject    handle to costoNN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function costosCW_Callback(hObject, eventdata, handles)
% hObject    handle to costosCW (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of costosCW as text
%          str2double(get(hObject,'String')) returns contents of costosCW
as a double


% --- Executes during object creation, after setting all properties.
function costosCW_CreateFcn(hObject, eventdata, handles)
% hObject    handle to costosCW (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function V_Callback(hObject, eventdata, handles)
% hObject    handle to V (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of V as text
%          str2double(get(hObject,'String')) returns contents of V as a
double


% --- Executes during object creation, after setting all properties.
```

```
function V_CreateFcn(hObject, eventdata, handles)
% hObject    handle to V (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a
double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in radioCostos.
function radioCostos_Callback(hObject, eventdata, handles)
% hObject    handle to radioCostos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radioCostos


% --- Executes on button press in radioDistancias.
function radioDistancias_Callback(hObject, eventdata, handles)
% hObject    handle to radioDistancias (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radioDistancias


% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton3



function rutaDistancias_Callback(hObject, eventdata, handles)
% hObject    handle to rutaDistancias (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of rutaDistancias as text
%                str2double(get(hObject,'String'))  returns  contents  of
rutaDistancias as a double


% --- Executes during object creation, after setting all properties.
function rutaDistancias_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rutaDistancias (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty  -  handles  not  created  until  after  all  CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in FindDistancias.
function FindDistancias_Callback(hObject, eventdata, handles)
% hObject    handle to FindDistancias (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[fileDi,rutaDi]  =  uigetfile('*.xls','Especifique  el  archivo  con  las
distancias entre clientes');

if isequal(fileDi,0)
```

```
    return;
end

set(handles.rutaDistancias,'Enable','on');
set(handles.rutaDistancias,'String',[rutaDi fileDi]);




function distCW_Callback(hObject, eventdata, handles)
% hObject    handle to distCW (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of distCW as text
%        str2double(get(hObject,'String')) returns contents of distCW as
a double


% --- Executes during object creation, after setting all properties.
function distCW_CreateFcn(hObject, eventdata, handles)
% hObject    handle to distCW (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function distNN_Callback(hObject, eventdata, handles)
% hObject    handle to distNN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of distNN as text
%        str2double(get(hObject,'String')) returns contents of distNN as
a double


% --- Executes during object creation, after setting all properties.
function distNN_CreateFcn(hObject, eventdata, handles)
% hObject    handle to distNN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in chkDistancias.
function chkDistancias_Callback(hObject, eventdata, handles)
% hObject    handle to chkDistancias (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chkDistancias

if (get(hObject,'Value') == 1)
    set(handles.FindDistancias,'Enable','on');
    set(handles.chkCoordenadas,'Enable','on');

    if(strcmp(get(handles.rutaDistancias,'String'),'<Ninguna
seleccionada>') == 0)
        set(handles.rutaDistancias,'Enable','on');
    end
else
    set(handles.rutaDistancias,'Enable','off');
    set(handles.FindDistancias,'Enable','off');

    set(handles.chkCoordenadas,'Value',1);
    set(handles.chkCoordenadas,'Enable','inactive');

    chkCoordenadas_Callback(handles.chkCoordenadas,eventdata,handles);
end




% --- Executes on button press in chkCostos.
function chkCostos_Callback(hObject, eventdata, handles)
% hObject    handle to chkCostos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chkCostos

if (get(hObject,'Value') == 1)
    set(handles.FindCostos,'Enable','on');
    set(handles.chkCoordenadas,'Enable','on');
```

```
    if(strcmp(get(handles.rutaCostos,'String'),'<Ninguna  seleccionada>')
== 0)
        set(handles.rutaCostos,'Enable','on');
    end
else
    set(handles.rutaCostos,'Enable','off');
    set(handles.FindCostos,'Enable','off');

    set(handles.chkCoordenadas,'Value',1);
    set(handles.chkCoordenadas,'Enable','inactive');

    chkCoordenadas_Callback(handles.chkCoordenadas,eventdata,handles);
end


% --- Executes on button press in pushSaveCW.
function pushSaveCW_Callback(hObject, eventdata, handles)
% hObject    handle to pushSaveCW (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[file,path] = uiputfile({'*.jpg','Imagen JPEG (*.jpg)'},'Guardar Grafo
respuesta por Ahorros como...');

if(isequal(file,0))
    return;
end

filetxt = strrep(file,'jpg','txt');

fh = fopen([path filetxt],'w');

if fh == -1
   errordlg('No se puede guardar un informe con los resultados del
problema');
   return
end

fprintf(fh,'H-CVRP\r\n\r\nPara el problema con los siguientes
datos\r\n\r\nDemanda (ubicación): %s\r\nCapacidad de los Vehiculos:
%s',get(handles.rutaDemandas,'String'),get(handles.K,'String'));

if get(handles.chkCostos,'Value') == 1
    fprintf(fh,'\r\nMatriz de Costos (ubicación):
%s',get(handles.rutaCostos,'String'));
else
    fprintf(fh,'\r\nMatriz de Costos deducida apartir de las Coordenadas
del Sistema');
end

if get(handles.chkCoordenadas,'Value') == 1
```

```
    fprintf(fh,'\r\nCoordenadas del Sistema (ubicación):
%s',get(handles.rutaCoordenadas,'String'));
else
    fprintf(fh,'\r\nCoordenadas del Sistema aleatorias');
end

fprintf(fh,'\r\n\r\nSe halló la siguiente solución\r\n\r\nRutas
Obtenidas:\r\n');

for i=1:1:size(handles.rutasCW,1)

    if handles.rutasCW(i,2) == 0
        continue;
    end

    for j=1:1:size(handles.rutasCW,2)

        if handles.rutasCW(i,j) == 0
            break;
        end

        fprintf(fh,'%d - ',handles.rutasCW(i,j));

        if j > 1 && handles.rutasCW(i,j+1) == 1
            fprintf(fh,'%d Costo = %d
',handles.rutasCW(i,j+1),handles.rutasCW(i,j+2));
            break;
        end
    end
    fprintf(fh,'\r\n');
end

fprintf(fh,'Su respectiva gráfica está disponible en %s\r\nCosto total:
%s\r\n\r\nSe utilizó el metodo de Clarke & Wright con un parámetro Lambda
de %s',[path
file],get(handles.costosCW,'String'),get(handles.txtLambda,'String'));

fclose(fh);

sv = getframe(handles.graphCW);
imwrite(sv.cdata,[path file]);

helpdlg('Se ha guardado con exito la solucion del problema por
Ahorros','Operacion Exitosa');


% --- Executes on button press in push4help.
function push4help_Callback(hObject, eventdata, handles)
% hObject    handle to push4help (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
szInfo = 'Tenga en cuenta lo siguiente:\n';

if(get(handles.chkVecinito,'value')==0.0                                    &&
get(handles.chkAhorros,'value')==0.0)
    szInfo = strcat(szInfo, '\nSeleccione AL MENOS UNO de los dos metodos
heuristicos que desee trabajar: Vecino mas Cercano y/o Clarke & Wright
(Ahorros)\n\nEnseguida:\n\n* Debe atender las indicaciones propias de
cada metodo seleccionado');
end

szInfo = strcat(szInfo,'\n* Debe especificar la ubicacion del archivo xls
que contiene la matriz con las demandas de los clientes\n* Debe
especificar la capacidad del vehiculo, mayor de 0');

if get(handles.chkCoordenadas,'Value') == 1
    szInfo = strcat(szInfo,'\n* Debe especificar la ubicacion del archivo
de Excel con los pares de coordeanadas XY de los clientes y el
deposito');
else
    szInfo = strcat(szInfo,'\n\nLas coordenadas XY donde se localizan el
deposito y los clientes serán aleatorias y no se tendrán en cuenta para
la solución del problema');
end

if(get(handles.chkVecinito,'Value')==1.0)
    szInfo = strcat(szInfo, '\n\nSOLO Para el metodo de Vecino mas
Cercano\n\n* [OPCIONAL] Debe especificar la ubicacion del archivo de
Excel (*.xls) que contiene la matriz de distancias. El numero de filas de
la matriz de demandas debe ser menor en apenas UNA al de dicha matriz. Si
especifica esta matriz, la matriz de coordenadas es OPCIONAL');
end

if(get(handles.chkAhorros,'Value')==1.0)
    szInfo = strcat(szInfo, '\n\nSOLO Para el metodo de Clarke & Wright
(Ahorros)\n\n* Debe fijar un valor para el parametro Lambda que sea mayor
de 0.0 ');
    szInfo = strcat(szInfo, '\n* [OPCIONAL] Debe especificar la ubicacion
del archivo de Excel (*.xls) que contiene la matriz de costos. El numero
de filas de la matriz de demandas debe ser menor en apenas UNA al de
dicha matriz. Si omite este dato, se deduciran los costos de la matriz de
coordenadas. De lo contrario, la matriz de coordenadas es OPCIONAL');
end

szInfo = strcat(szInfo, '\n\nPara ubicar los archivos de Excel que
contienen los datos requeridos por el programa, oprima el boton con los
puntos suspensivos (...) que se ubica a la derecha de cada casilla de
texto. Para habilitarlo (si es el caso), debe antes seleccionar la
variable opcional que piense usar');
szInfo = strcat(szInfo, '\n\nSi al menos UNO de estos datos no se
especifica (correctamente), NO se podra realizar ningun calculo mientras
no se corrija el error');

msgbox(sprintf(szInfo),'Antes de Calcular...','help','modal');
```

```
% --- Executes on button press in pushSaveNN.
function pushSaveNN_Callback(hObject, eventdata, handles)
% hObject    handle to pushSaveNN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[file,path] = uiputfile('*.jpg','Guardar Grafo respuesta por Vecino Mas
Cercano como...');

if(isequal(file,0))
    return;
end

filetxt = strrep(file,'jpg','txt');

fh = fopen([path filetxt],'w');

if fh == -1
   errordlg('No se puede guardar un informe con los resultados del
problema');
   return
end

fprintf(fh,'H-CVRP\r\n\r\nPara el problema con los siguientes
datos\r\n\r\nDemanda (ubicación): %s\r\nCapacidad de los Vehiculos:
%s',get(handles.rutaDemandas,'String'),get(handles.K,'String'));

if get(handles.chkDistancias,'Value') == 1
    fprintf(fh,'\r\nMatriz de Distancias (ubicación):
%s',get(handles.rutaDistancias,'String'));
else
    fprintf(fh,'\r\nMatriz de Distancias [simétrica] obtenida con base en
las Coordenadas XY del Sistema');
end

if get(handles.chkCoordenadas,'Value') == 1
    fprintf(fh,'\r\nCoordenadas del Sistema (ubicación):
%s',get(handles.rutaCoordenadas,'String'));
else
    fprintf(fh,'\r\nCoordenadas del Sistema generadas aleatoriamente');
end

fprintf(fh,'\r\n\r\nSe halló la siguiente solución\r\n\r\nRutas
Obtenidas:\r\n');

for i=1:1:size(handles.rutasNN,1)

    if handles.rutasNN(i,2) == 0
        continue;
    end
```

```
    for j=1:1:size(handles.rutasNN,2)

        if handles.rutasNN(i,j) == 0
            break;
        end

        fprintf(fh,'%d - ',handles.rutasNN(i,j));

        if j > 1 && handles.rutasNN(i,j+1) == 1
            fprintf(fh,'%d ',handles.rutasNN(i,j+1));
            break;
        end
    end
    fprintf(fh,'\r\n');
end

fprintf(fh,'Su respectiva gráfica está disponible en %s\r\nDistancia
total: %s\r\n\r\nSe utilizó el metodo de Vecino Más Cercano',[path
file],get(handles.distNN,'String'));

fclose(fh);

sv = getframe(handles.graphNN);
imwrite(sv.cdata,[path file]);

helpdlg('Se ha guardado con exito la solucion del problema por Vecino mas
Cercano','Operacion Exitosa');


% --- Executes on button press in distSimetrica.
function distSimetrica_Callback(hObject, eventdata, handles)
% hObject    handle to distSimetrica (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of distSimetrica


% --- Executes on button press in distAsimetrica.
function distAsimetrica_Callback(hObject, eventdata, handles)
% hObject    handle to distAsimetrica (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of distAsimetrica


% --- Executes when selected object is changed in panelDistancias.
function panelDistancias_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in panelDistancias
% eventdata  structure with the following fields (see UIBUTTONGROUP)
```

```
%   EventName: string 'SelectionChanged' (read only)
%    OldValue: handle of the previously selected object or empty if none
was selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)



% --- Executes on button press in chkDistancias.
function checkbox6_Callback(hObject, eventdata, handles)
% hObject    handle to chkDistancias (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chkDistancias


% --- Executes on button press in chkCoordenadas.
function chkCoordenadas_Callback(hObject, eventdata, handles)
% hObject    handle to chkCoordenadas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chkCoordenadas

if get(hObject,'Value') == 1
    set(handles.pushFindXY,'Enable','on');

    if              strcmp(get(handles.rutaCoordenadas,'String'),'<Ninguna
seleccionada>') == 0
        set(handles.rutaCoordenadas,'Enable','on');
    end

else
    set(handles.pushFindXY,'Enable','off');
    set(handles.rutaCoordenadas,'Enable','off');
end
```