FORMULACIÓN Y EVALUACIÓN DE UN ALGORITMO, BASADO EN LA META-HEURÍSTICA "BÚSQUEDA TABÚ" PARA LA OPTIMIZACIÓN DEL RUTEO DE VEHÍCULOS CON CAPACIDAD.

ALFREDO JOSÉ PERTUZ MONTENEGRO
KIMBERLY ROJAS SILVA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
BUCARAMANGA
2007

FORMULACIÓN Y EVALUACIÓN DE UN ALGORITMO, BASADO EN LA META-HEURÍSTICA "BÚSQUEDA TABÚ" PARA LA OPTIMIZACIÓN DEL RUTEO DE VEHÍCULOS CON CAPACIDAD

ALFREDO JOSÉ PERTUZ MONTENEGRO
KIMBERLY ROJAS SILVA

Proyecto de grado presentado para optar al título de
Ingeniero de Industrial

Director
EDWIN ALBERTO GARAVITO HERNÁNDEZ
Ingeniero Industrial

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍA FISICO MECÁNICAS
ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
BUCARAMANGA
2007

TABLA DE CONTENIDOS

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

Pág.

RESUMEN

TÍTULO

FORMULACIÓN Y EVALUACIÓN DE UN ALGORITMO, BASADO EN LA META-HEURÍSTICA "BÚSQUEDA TABÚ" PARA LA OPTIMIZACIÓN DEL RUTEO DE VEHÍCULOS CON CAPACIDAD.

AUTORES

ALFREDO JOSE PERTUZ MONTENEGRO
KIMBERLY ROJAS SILVA

PALABRAS CLAVES

Ruteo de Vehículos, Meta-Heurística, Búsqueda Tabú, Técnica Clarke and Wright, Técnica Lin-Kernighan.

DESCRIPCIÓN

El algoritmo desarrollado, es una herramienta que permite la solución del problema de ruteo de vehículos con capacidad. Este utiliza la metodología Búsqueda Tabú para la optimización de las rutas, apoyada en la técnica Clarke and Wright para la creación de la ruta inicial y en la técnica Lin-Kernighan para la construcción del vecindario. En la Búsqueda Tabú también se implementan las estrategias de Intensificación y Diversificación para explorar mejor el espacio de soluciones y hallar soluciones cercanas a las óptimas, evitando caer en óptimos locales.

El problema de CVRP o problema de Ruteo de Vehículos con capacidad es una variante del TSP (Traveling Salesman Problem) que se caracteriza por la existencia de una flota de vehículos que debe satisfacer la demanda de los clientes en una sola visita. Además cada vehículo tiene una capacidad limitada y debe regresar al depósito al término de la secuencia trazada para él. En este tipo de problema de ruteo se debe considerar la demanda de cada cliente y la capacidad de cada vehículo. Todos los itinerarios comienzan y terminan en el depósito común.

El algoritmo ideado fue implementado en un programa de computadora, desarrollado en la plataforma Java, a través del cual se calculan las soluciones dependiendo de ciertos parámetros iniciales como: Número de Iteraciones, Tamaño de la Lista Tabú, Tamaño de la Población y Número de Intercambios. Dicho programa arroja la mejor ruta encontrada y su debida descripción que incluye: Costo (Unidades de Distancia), Parámetros iniciales (Número de Iteraciones, Número de Intercambios, Capacidad de los Vehículos), la secuencia en la cual deben ser atendidos los clientes con sus respectivos datos (Coordenada en X, Coordenada en Y, Demanda).

Como es posible observar, el problema del Ruteo de Vehículos tiene su aplicación más directa en el área Logística de cualquier organización. Los estudios realizados acerca de éste datan de 1920, a pesar de la "simplicidad" de su enunciado, el reto de científicos y estudiantes en el área de la optimización consiste en encontrar mejores rutas en tiempos más cortos. Lo anterior implica un gran esfuerzo en el desarrollo de nuevas metodologías más efectivas, apoyadas en la tecnología para alcanzar el objetivo.

El 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.

Trabajo de Grado.
Facultad de Ingenierías Físico-Mecánicas, Escuela de Estudios Industriales y Empresariales, Edwin A. Garavito H.

SUMMARY

TITLE

OPTIMIZATION OF THE VEHICLE ROUTING PROBLEM WITH CAPACITY THROUGH AN ALGORITHM BASED ON THE META-HEURISTIC TABU SEARCH.

AUTHORS

ALFREDO JOSE PERTUZ MONTENEGRO
KIMBERLY ROJAS SILVA

KEY WORDS

Vehicle Routing Problem with Capacity, Meta-Heuristic, Tabu Search,

DESCRIPTION

The developed algorithm solves the Vehicle Routing Problem with capacity. The mentioned algorithm uses the "Tabu Search" methodology for the Routing Optimization, based on the Clarke and Wright technique for finding the initial solution and on the Lin-Kernighan technique for the Neighborhood Construction. The Tabu Search also implements the Intensification and Diversification strategies to explore more the solutions space and to find solutions close to the optimal ones.

The CVRP or the Capacity Vehicle Routing Problem is a variant of the TSP (Traveling Salesman Problem) known by the existence of a group of vehicles that have to satisfy the demand of certain clients in one single visit. Besides each vehicle has a restricted capacity and must return to the deposit at the end of the sequence traced for it. This kind of problem must consider the demand of each client and the capacity of each vehicle. Every route must begin and end in the deposit.

The algorithm created was implemented in software, developed in the Java platform, in order to calculate the solutions according to different parameters like: Number of Iterations, Size of Tabu List, Size of the Population and the Number of Exchanges. This program shows the best found solution and its description that includes: Cost (Distance Units), Initials Parameters (Number of Iterations, Size of Tabu List, Size of the Population and the Number of Exchanges) and the sequence of clients to be supplied.

The Vehicle Routing Problem has its most direct application in the Logistic area of any organization, and it's studied since 1920. Despite the simplicity of the definition, this is a NP Hard Problem, and the challenge of scientist and students is to design new methodologies to find better solutions in short processing times, using the technology as a support to reach the main goal.

Thesis.
Faculty of Physical-Mechanical Engineerings, School of Business and Industrial Studies, Edwin A. Garavito H

INTRODUCCIÓN

Este documento se propone cumplir la necesidad de documentar el desarrollo del algoritmo sugerido para aquellas personas que emprendan un estudio posterior en este tema, con el propósito de enriquecerlo anexándole nuevas funcionalidades e información que le permita adaptarse a las continuas investigaciones que se hagan del mismo.

El libro se encuentra dividido en siete capítulos que resumen el planteamiento del problema, la proposición de la solución, incluyendo su análisis, desarrollo e implementación, y de la manera como esta solución aplica todas las metodologías elegidas para dar respuesta al problema inicial.

Con el fin de ilustrar a las personas con respecto a la evolución del problema y las metodologías utilizadas para su solución, se presenta una breve historia de la Optimización Combinatoria, en el capítulo segundo, incluyendo las técnicas a aplicar en el presente proyecto. En el tercer capítulo se explica el algoritmo paso a paso y la forma en que resuelve el problema planteado, además se incluyen diagramas de flujo del mismo para facilitar su compresión. Esta sección se considera de gran importancia dado que presenta el funcionamiento del algoritmo diseñado por los autores, siendo este el objetivo principal del proyecto.

En el capítulo cuarto, se presentan las mejores soluciones obtenidas para cada uno de los ejercicios propuestos (EIL 51, EIL A76, EIL B76, EIL C76, EIL D76, EIL A101 y EIL B101), además de un análisis de estos resultados al compararlos con los mejores resultados de la literatura, incluyendo gráficas y tablas ilustrativas. En el capítulo quinto se estudian todos los datos recolectados para el diseño de experimentos, cuyos resultados indican la relevancia de los factores analizados y la combinación de factores para la obtención de soluciones de mejor calidad. Finalizando el estudio en el capítulo sexto donde se presentan las conclusiones obtenidas del presente proyecto, indicando los aspectos más importantes encontrados a lo largo del proceso de investigación.

Pensando en las necesidades y dudas generadas por los usuarios, se ha agregado como un anexo el manual para el usuario final de la herramienta software que le permitirá navegar a través de ella, buscando efectividad en tiempo de uso y así, disminuyendo las pérdidas del mismo, producto de un desconocimiento del programa de computadora por parte del usuario.

1 OBJETIVOS

1.1 OBJETIVO GENERAL

Formular y evaluar un algoritmo que utilice la metodología de la meta-heurística "Búsqueda Tabú" para solucionar el problema de ruteo CVRP (Capacitated Vehicle Routing Problem), e implementarlo en un programa de computadora para su uso.

1.2 OBJETIVOS ESPECIFICOS

Diseñar un algoritmo que implemente la metodología de la meta-heurística "Búsqueda Tabú" usando la técnica de Clarke and Wright para determinar las soluciones iniciales y Lin Kernighan para la construcción del vecindario de las soluciones.

Desarrollar una subrutina programada en Java® para a la solución del problema de ruteo CVRP que implemente el algoritmo que se elaborará.

Determinar los factores relevantes en la calidad de la respuesta ofrecida por la meta-heurística "Búsqueda Tabú", a través de un diseño de experimentos factorial 2^3 con puntos centrales.

Comparar los resultados obtenidos por el algoritmo de la meta-heurística "Búsqueda Tabú" y la mejor respuesta registrada en la literatura, para los problemas EIL51, EILA76, EILB76, EILC76, EILD76, EILA101 y EILB101 para evaluar la calidad de los mismos.

Elaborar el manual de usuario en el cual se explique cómo utilizar el programa de computadora desarrollado.

2 MARCO TEÓRICO

A Continuación se ilustra una síntesis informativa de los temas relacionados con la evolución histórica de la optimización combinatoria, científicos y metodologías relacionadas con ella, entre las cuales se mencionan los procedimientos y definiciones a utilizar a lo largo del desarrollo de este proyecto. Para obtener una información más amplia de estos argumentos, diríjase a la bibliografía citada.

2.1 OPTIMIZACIÓN COMBINATORIA

Los problemas de optimización combinatoria están relacionados con la asignación de recursos escasos para el logro de los objetivos deseados cuando los valores de algunas variables son restringidos a "ser enteros". Las restricciones en los recursos básicos reducen el número de alternativas consideradas como viables, una de las principales metas consiste en determinar cuál de las alternativas es la mejor.

La optimización combinatoria es el proceso mediante el cual se encuentra 1 o más "mejores" soluciones (óptimas) en un problema con un espacio discreto bien definido. La palabra combinatoria se refiere al hecho que solo existe un conjunto finito de alternativas de solución.

Este proceso involucra un número finito pero inmenso de opciones, y es ahí donde radica la complejidad al solucionarlos. Las aplicaciones de este tipo de problemas están creciendo rápidamente, entre ellas podemos mencionar: planeación de la producción, planeación y secuenciación de trabajos, diseños de lay-out, planeación y ruteo de vehículos, planeación de ventas estacionales, diseño de redes de telecomunicación, entre otras; estas actividades pertenecen a diversos sectores industriales como: la electrónica, manufactura, telecomunicación, defensa, ganadería, etc.

2.1.1 CLASIFICACIÓN DE LOS PROBLEMAS DE OPTIMIZACIÓN COMBINATORIA

1. P: Esta clase corresponde al tipo de problemas cuya complejidad permite solucionarlo en un tiempo polinomial, es decir puede ser resuelto en un tiempo razonable en los computadores que existen hoy.
2. NP ("Non-Deterministic Poyinomial-Time"): Este corresponde a un problema para el cual las soluciones son corroboradas por medio de un algoritmo en un tiempo de proceso polinomial, lo anterior no implica que las soluciones serán encontradas rápidamente, sólo que la solución hallada puede ser verificada o rechazada de forma rápida. Un problema de esta clase presenta un tiempo de proceso exponencial en un computador personal y generalmente no se soluciona en un tiempo razonable, pero si pueden ser resueltos en una máquina de Turing[1].

[1] "Una máquina de "Turing" es un dispositivo que transforma un "INPUT" en un "OUTPUT" después de algunos pasos. Tanto el "INPUT" como el "OUPUT" constan de números en código binario (ceros y unos). En su versión original la máquina de "Turing" consiste en una cinta infinitamente larga con unos y ceros que pasa a través de una caja. La caja es tan fina que solo el trozo de cinta que ocupa un "bit" (0 ó 1) está en su interior. La máquina tiene una serie de estados internos finitos que también se pueden numerar en binario. Para llevar a cabo algún algoritmo, la máquina se inicializa en algún estado interno arbitrario. A continuación, se pone en marcha y la máquina lee el "bit" que se encuentra en ese momento en su interior y ejecuta alguna operación con ese "bit" (lo cambia o no, dependiendo de su estado interno).

3. NP-Complete ("Non-Deterministic Polinomial-Time Complete"): En este tipo de problemas la relación entre el número de parámetros y la complejidad del problema es exponencial. Si alguna búsqueda iterativa es adoptada, entonces el incremento exponencial en la complejidad del problema trae como resultado un incremento exponencial en el tiempo de solución.

2.1.2 HISTORIA DE LA OPTIMIZACIÓN COMBINATORIA[2]

La diversidad de los orígenes de la optimización combinatoria se debe principalmente a que la mayoría de estos problemas provienen de la práctica y sus instancias fueron y aun son abordadas diariamente. Actividades como el transporte de productos, planeación de ventas o visitas, asignación de tareas, entre otros, son problemas que no son considerados solamente por matemáticos. Se comentará la evolución de las metodologías desarrolladas en seis áreas principales:

EL PROBLEMA DE ASIGNACIÓN

Este problema consiste en que dada una matriz de costos nxn C = ($c_{i,j}$) se debe hallar una permutación de π de 1,…,n donde $\sum_{i=1}^{n} c_{i,\tau(i)}$ sea lo mas pequeño posible.

MONGE (1784)

Este problema fue estudiado por Monge en 1784, motivado por el transporte de tierra, el cual fue considerado como un problema discontinuo combinatorio de transporte de moléculas. Monge desarrolló un modelo geométrico en el cual se dibuja una línea tangente entre las dos áreas, la molécula que es tocada se mueve a la segunda área por la línea trazada, repitiéndose hasta que la cantidad total sea transportada.

ROBINSON (1949)

En un intento por solucionar el problema de ruteo de vehículos, Robinson desarrolló el método de reducción de ciclo (cycle reduction method) donde se considera a la matriz ($a_{i,j}$) y se define una longitud $L_{i,j}$ para todo i,j y $L_{i,j}=a_{j,\pi(i)} - a_{j,\pi(i)}$ si $j \neq \pi(i)$ y $L_{i,\pi(i)}=\infty$.

Entonces si existe una longitud negativa en el circuito, hay posibilidades de mejorar π. Si no existe un circuito de este tipo quiere decir que π es una permutación óptima. Este es un método finito que puede aplicarse a 50 puntos si se cuenta con el equipo adecuado.

MÉTODO SIMPLEX

Dantzig demostró en 1951 que el problema de asignación puede ser formulado como un problema de programación lineal y por tanto tiene una solución entera óptima, esto lo desarrolló basado en el teorema de Birkhoff (1946).

Después se mueve hacia la derecha o hacia la izquierda, y vuelve a procesar el siguiente bit de la misma manera". ttp://etsiit.ugr.es/alumnos/mlii/Maquina%20de%20Turing.htm

[2]SCHRIJVER, Alexander. "On the history of combinatorial optimization". Pag. 1-34.

MÉTODO HÚNGARO

Esta metodología fue desarrollada por Kuhn (1955-1956) basado en el método propuesto por Egerváry (1931), la experiencia de Ford and Fulkerson al utilizar esta técnica al resolver un problema de 20x20 fue de poco más de una hora por medio del Simplex contra 30 minutos con el Húngaro.

EL PROBLEMA DEL TRANSPORTE

Este problema consta de una matriz de costos $C = (a_{i,j})$, un vector de demanda $d \in R_+^n$ y un vector de suministro $b \in R_+^m$.

$$\sum_{j=1}^{n} x_{i,j} = b_i \quad para \ i = 1,...,m$$

$$\sum_{i=1}^{m} x_{i,j} = d_j \quad para \quad j = 1,...,n$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j} x_{i,j} es \ lo \ más \ pequeño \ posible$$

Este problema es un caso especial de programación lineal.

TOLSTOI (1930)

Publicó un artículo llamado "Methods of finding the minimal total kilometrage in cargo-transportation planning space", en el cual se explican varios métodos y aproximaciones, donde no se demuestra que la "condición de ciclo" es necesaria para encontrar la optimalidad, pero se muestra la idea que en la solución óptima no hay ningún ciclo de costo negativo en el grafo residual. Tolstoi considera inicialmente solo dos fuentes (depósitos), observando que se pueden ordenar los destinos por la diferencia entre la distancia con respecto a las dos fuentes. Donde una provee a los clientes de la lista hasta que ya no tiene provisiones, entonces la segunda la reemplaza.

KOOPMANS (1942-1948)

Koopmans estudió la asignación de embarcaciones a convoys para hacer las entregas con el menor número de viajes vacíos. Este era un problema del tipo 3x2, donde Koopmans declara que si no es posible obtener una mejora reprogramando de forma cíclica el algoritmo, entonces la solución es óptima. La propuesta consiste principalmente en que existen potenciales $p_1,\ldots, p_n$ y $q_1,\ldots, q_n$ de tal forma que $c_{i,j} \geq p_i - q_j$ para todo i, j; donde $c_{i,j} = p_i - q_j$ para cualquier i, j en la solución óptima donde x tiene un $x_{i,j} > 0$.

EL MÉTODO SIMPLEX Y LA PROGRAMACIÓN LINEAL

En un artículo diferente Dantzig escribe sobre la implementación directa del método simplex aplicado al problema del transporte. Votaw y Orden (1952) afirmaron, sin prueba alguna, que el tiempo computacional del simplex es polinomial. En 1973 esta afirmación fue refutada por Zadeh.

EL TEOREMA DE MENGER Y EL FLUJO MÁXIMO (MAXIMUN FLOW)

Menger publicó su teorema en un artículo llamado: "Zur Allgemeiner Kurventheorie" en 1927. El resultado puede ser formulado en términos de grafos donde G= (V,E) y $P,Q \subseteq V$. Donde el máximo número de arcos P-Q es igual a la cardinalidad mínima de un conjunto de W de vértices de tal forma que cada arco P-Q intersecta a W.

Flujo Máximo: Dado un grafo con un vértice inicial "S" y un vértice final "T" específico, y dada una función de capacidad "C" definida en sus arcos, encontrar el flujo entre S y T sujeto a C, de tal forma que se alcance el valor máximo.

Ford y Fulkerson dieron una definición de este problema en 1954 y ofrecieron referencias más precisas en 1962 en su libro llamado "Flows and Networks". Para la solución se propuso una heurística llamada "flooding technique" la cual es descrita en 1955 por A. W. Bodyreff.

Dado que el problema inicialmente se refería a una red de rieles que conectan dos ciudades con otras intermedias, donde cada turno cuenta con una capacidad determinada. La heurística se explica como: involucrar la mayor cantidad de flujo por la red, si en algún vértice hay "embotellamiento" (si llegan más trenes que los que pueden ser despachados) los trenes de exceso son devueltos al origen. Esto no garantiza la optimalidad pero Bodyreff argumenta que al retirar los embotellamientos de la red "debería" conducir a un flujo máximo y al contrario de otras metodologías esta es más sencilla.

Ford y Fulkerson publicaron en un artículo de RAND su algoritmo, el cual garantiza la optimalidad para este problema. Con respecto al Simplex, los cálculos a realizar serían demasiados, y si pudieran obtenerse no se tendrían datos suficientemente exactos para justificar tal labor.

SHORTEST SPANNING TREE (EL ÁRBOL MÁS CORTO PARA ATRAVESAR)

BORUVKA 1926

El primero en considerar este problema es Boruvka en 1926, su interés nace en la construcción de una red eléctrica en la empresa "Electric Power Company Western Moravia" de tal forma que fuera lo más económica posible. El método que propuso se llama "parallel merging" donde conecta cada componente al componente vecino más cercano, y realiza iteraciones de lo anterior hasta obtener la ruta completa. Cada arco tiene un peso correspondiente, el árbol o ruta con menor peso corresponde a la mejor solución.

JARNÍK 1929

Jarník escribe un artículo en 1929 donde comenta una nueva solución para el problema tratado por el Sr. Boruvka, la cual consiste en hacer crecer el árbol poco a poco "tree-growing", agregando los arcos más cortos o de menor peso de forma iterativa.

PERIODO 1956-1959

Muchos métodos fueron presentados, muchos que no ofrecían mejoras significativas con respecto a los dos presentados anteriormente. Kruskal en 1956 escribió un artículo inspirado en el primero de Boruvka y su aplicación al problema del ruteo, este escribió tres algoritmos:

1. A: Escogencia iterativa de los arcos que pueden ser adicionados al árbol para crear el circuito.
2. B: Escogencia iterativa de los arcos más cortos de un conjunto U de vértices, dejando algún componente que intersecte U.
3. A': Eliminar iterativamente los arcos más largos que pueden ser desconectados sin necesidad de desconectar el algoritmo.

PRIM 1957

Prim publicó un artículo en 1957, en el cual propone escoger un vértice y luego conectarlo con los vértices más cercanos, además aclaró que los casos A y B de Kruskal están incluidos en esta metodología y puntualizó que sólo el orden de las longitudes determina si el árbol es el más corto o no.

EL CAMINO MÁS CORTO (SHORTEST PATH)

Comparado con otros problemas presentados anteriormente, la investigación de este empezó relativamente tarde. Este tipo de problemas se estudiaron a principios de los 50's como aplicación a las llamadas telefónicas, dado que en esa época las llamadas de larga distancia en USA eran automáticas. Cuando un cliente realiza una llamada a larga distancia, el principal problema radicaba en cómo hacer llegar la llamada al destino.

MÉTODOS UTILIZANDO MATRICES 1946-1953

Estos métodos se estudiaron por su aplicación a las redes de comunicación y a la sociología animal. Este método consiste en representar por medio de una matriz un grafo, luego se realizan productos de matrices hasta hallar el cierre transitivo (transitive closure).

FORD 1956

Para hallar el camino más corto entre P_0 y P_N, en una red de vértices "$P_0,\ldots P_N$", donde L_{ij} denota la longitud del arco que se extiende entre i y j. Inicialmente se asigna $x_0 = 0$ y $x_i = \infty$ para $i \neq 0$. Se revisa la red para un par de P_i y P_j con la propiedad: $x_i - x_j > l_{ij}$ y luego se reemplaza x_i por $x_i + L_{ij}$. Se continúa este proceso de forma iterativa, si eventualmente no pueden encontrarse dichos pares, se debe a que se tiene un x_N mínimo que corresponde a la distancia mínima entre P_0 y P_N. no se describe ninguna regla para la selección de arcos. Más tarde Jonson (1973-1977) demostró que esta regla toma tiempos exponenciales.

EL PROBLEMA DEL AGENTE VIAJERO (TRAVELLING SALESMAN PROBLEM - TSP) [3]

La primera definición conocida del TSP fue explicada por Karl Menger[4] a sus colegas de Viena en el año de 1920, pero fue hasta 1932 que hizo su publicación formal llamada "Das botenproblem" en "Ergebnisse eines Mathematischen Kolloquiums". Este volumen contiene una declaración acerca del problema planteado por él en Febrero 5 de 1920 ante el coloquio matemático de Viena.

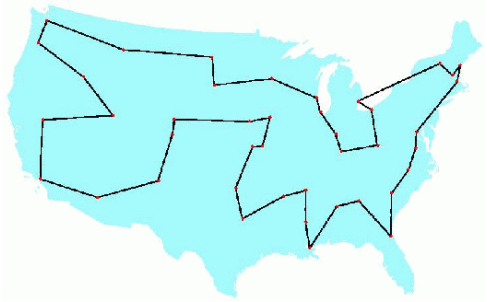
[3] CUMMINGS,Nigel. "A Brief History of TSP". OR Society. [en línea].2002.

[4] El matemático y economista Karl Menger es mejor conocido por sus estudios en geometría probabilística y algebra de funciones.

Menger llamó al TSP como el “Messenger Problem” o problema del mensajero y lo definió como la tarea de encontrar, por un número de puntos definidos (cuyas distancias entre ellos es conocida) el camino más corto que conecte dichos puntos, teniendo en cuenta que la regla que expresa que partiendo de un punto inicial es preciso ir al siguiente punto más cercano, no conduce generalmente a la mejor solución o camino más corto.

En los 40’s el TSP fue estudiado por los estadísticos, Mahalanobis (1940), Jessen (1942), Gosh (1948), y Marks (1948), en relación a la aplicación en la agricultura de Mahalanobis. En esta, se escogió una pequeña muestra del número de acres en Bengala y se discutieron las soluciones del TSP a través de la elección de las instancias de forma aleatoria en el plano Euclidiano, este trabajo estuvo bajo la influencia de uno ya realizado en 1938 en las granjas de Bengala, donde el mayor costo era representado por el transporte de equipos y hombres de un punto a otro.

Con el transcurrir del tiempo, el TSP ganó notoriedad como un problema de optimización combinatoria, donde examinar ruta por ruta acarreaba un trabajo difícil y engorroso debido al gran número de ellas. Sin embargo, en el año de 1954 un hito fue marcado en la historia con la aparición de la programación lineal como herramienta computacional, por que con ella era posible resolver el problema del TSP para un mayor número de instancias. En este año Dantzig, Fulkerson y Johnson publicaron la descripción del método aplicado a la solución del TSP, comprobando su capacidad al resolver el problema con 49 ciudades, lo cual era un gran avance para la época. Crearon las instancias escogiendo una ciudad de cada uno de los 48 estados de Estados Unidos[5] y Washington D.C. Los costos de transportes fueron calculados teniendo en cuenta las distancias por carretera entre las ciudades.



En 1955 G. Morton y A. H. Land publicaron un documento referido principalmente al uso de la programación lineal para la solución del problema, resaltando el uso de la técnica 3-Opt.[6], y el desarrollo de un sistema que según permitía investigar el cambio en el costo por cada movimiento sin destruir el circuito.

Más tarde, en el año de 1970 R. M. Karp y M. Held publicaron "The travelling-salesman problem and minimum spanning trees", artículo en el cual se introdujo el uso de la técnica “1-tree relaxation” para resolver el TSP y el uso de peso en las instancias para mejorar los movimientos aportados por la técnica 1-tree. Una segunda parte mejorada y más eficiente fue publicada en 1971, algoritmo que utilizaron para resolver varios problemas ya planteados y famosos como: las 49 ciudades de Dantzig, Fulkerson, y Johnson, las 57 ciudades de Karg y Thompson y 64 ciudades aleatorias euclidianas.

El reto de encontrar las soluciones óptimas en un tiempo más corto, ha conducido a científicos e ingenieros de varias áreas a unirse a la investigación que gira en torno al TSP para perfeccionar y crear algoritmos más

[5] Alaska y Hawai se convirtieron en estados hasta 1959

[6] La técnica “3-Opt.” es un algoritmo a través del cual una ruta crea una nueva al eliminar tres conexiones o aristas y unir los vértices libres de una forma diferente, tal que la nueva solución sea menos costosa que la inicial

eficientes y efectivos, contando con el apoyo del avance tecnológico el cual ha permitido que el tiempo de cálculo computacional se reduzca cada vez más.

Del Anexo 22 al Anexo 25 se presentan fotografías de varios científicos que han trabajado en este problema.

CLASIFICACIÓN DE LOS MÉTODOS UTILIZADOS PARA SOLUCIONAR EL TSP

Dos métodos comúnmente utilizados en la solución del TSP corresponden a los métodos de crecimiento y refinamiento.

1. Los métodos de crecimiento consisten en construir la ruta de forma desordenada pero garantizando que la longitud de la misma sea minimizada.
2. Los métodos de refinamiento son aquellos que toman una ruta ya creada y modifican su calidad, por medio de mejoras en diferentes sectores de la misma.

Entre los algoritmos de construcción de rutas encontramos:

1. Vecino Más Cercano: Esta metodología consiste en elegir un depósito de partida, luego, de acuerdo a las distancias de los clientes se elige el cliente más cercano, y así sucesivamente se va construyendo la ruta. Es necesario tener en cuenta que además de visitar a todos los clientes, se debe respetar la capacidad de cada vehículo. Este algoritmo tiene un tiempo de proceso de N^2
2. Algoritmo Greedy: Esta metodología consiste en el aumento de un ciclo Hamiltoniano[7] en el grafo, esto se lleva a cabo escogiendo el arco más corto para agregarlo a la ruta, hasta que todas las conexiones son incluidas. Un arco solo puede agregarse si no esta en la ruta, si tiene un vértice con grado mayor de dos o si su inclusión no crea un loop (lazo). Este problema tiene un tiempo de proceso equivalente a $N^2 \log N$.
3. Algoritmo Clarke-Wrigth: Esta metodología será explicada en la sección 4.2.6 por ser la técnica de crecimiento a utilizar en el presente proyecto.

También existen los algoritmos de Búsqueda Local cuya estrategia de solución consiste en tomar una solución inicial como el dato de entrada, luego la heurística examina el espacio de solución haciendo simples modificaciones en la ruta de la misma. Si la nueva ruta presenta una mejora, entonces se continúa el proceso tomando como dato de entrada la nueva solución. Si no hay mejora se repite el proceso a partir de la mejor solución actual. Generalmente las acciones consisten en intercambios o movimientos que permiten convertir una ruta en otra, las cuales se repiten iterativamente hasta que no se obtiene ninguna mejora o hasta cumplir con el criterio de parada. Entre estos algoritmos encontramos los siguientes:

1. 2-Opt.: Esta es la primera heurística probada para el TSP. Los primeros resultados fueron publicados en 1956. Consiste en eliminar dos arcos de la ruta y reconectarlos de una forma totalmente diferente.
2. 3-Opt.: Esta técnica es similar a la anterior pero en lugar de dos arcos se eliminan y reconectan tres arcos. En el momento de la reconexión el algoritmo valora la solución para decidir si esa es la mejor forma de reconectar la ruta en esa instancia, debido a esto el algoritmo toma más tiempo para el cálculo que la metodología anterior.

Backtracking-Jump: Este tipo de algoritmos se crearon con el fin de escapar de óptimos locales y de esta forma explorar mejor el espacio de soluciones. Por medio de esta técnica es posible realizar un conjunto de movimientos que conducirán a mejoras en la solución, aún si individualmente no tengan un impacto en la solución. Si la búsqueda no conduce a mejoras, es posible retroceder y comenzar de nuevo.

[7] Un ciclo Hamiltoniano en un grafo corresponde a un ciclo que recorre una sola vez todos los vértices del mismo.

VARIACIONES DEL TSP

Existen ciertas variaciones del TSP, que surgen al cambiar o agregar ciertas restricciones al problema original.

1. TSP (Traveling Salesman Problem – Problema del agente viajero): Corresponde a la forma general del ruteo de vehículo, en la cual se cuenta con una flota que debe atender la demanda de n clientes. En este caso se debe visitar a cada cliente una sola vez para satisfacer completamente sus necesidades y los vehículos no tienen límite de capacidad. Todos los itinerarios comienzan y terminan en el depósito común[8]. El TSP es un problema combinatorio clásico debido a que no existe un método exacto que pueda manejar con efectividad el arduo trabajo que implica. Se utilizan las heurísticas para darle solución porque este tipo de procedimientos arrojan soluciones de muy buena calidad en un período de tiempo relativamente corto con respecto a los demás métodos exactos.
2. CVRP (Capacitated Vehicle Routing Problem – Problema de ruteo con capacidad en los vehículos): Esta variante se caracteriza porque la flota de vehículos debe satisfacer la demanda de los clientes en una sola visita, teniendo en cuenta que tienen capacidad limitada. Cada vehículo debe regresar al depósito al término de la secuencia trazada para él. En este tipo de problema de ruteo se debe considerar la demanda de cada cliente y la capacidad de cada vehículo. Todos los itinerarios comienzan y terminan en el depósito común.
3. TCVRP (Time Constrained Vehicle Routing Problem – Problema de ruteo de vehículo con restricciones de tiempo): En este caso, se debe satisfacer la demanda de cada cliente con una sola visita y considerando además, la capacidad de los vehículos, se debe tener presente que la flota de transporte cuenta con restricciones de tiempo, es decir, que existe una brecha de tiempo para la utilización de los vehículos (la flota tiene un horario de atención bien definido).
4. CVRPTW (Capacitated Vehicle Routing Problem with Time Windows – Problema de ruteo con capacidad de vehículo y ventanas de tiempo): Este caso reúne las condiciones de las variantes anteriores, las cuales son: Capacidad limitada de los vehículos, Demanda conocida de los clientes, Horario de los vehículos definido para la distribución, Satisfacer al cliente en una sola visita, Retorno de los vehículos al depósito por cada secuencia de visita a los clientes, Pero además los clientes cuentan con un horario de recepción para los vehículos del proveedor.
5. PVRP (Periodic Vehicle Routing Problem – Problema de ruteo con periodicidad de visita): La principal variante en este caso, radica en que el cliente debe ser visitado por lo menos una vez dentro de un período determinado de tiempo, además de que los vehículos tienen una capacidad limitada, un horario de atención, y los clientes cuentan con un horario para la recepción de la mercancía.

En cualquiera de las variantes presentadas anteriormente, el objetivo principal consiste en la elaboración de la secuencia en que deben satisfacerse las necesidades de los clientes, minimizando el costo y la distancia recorrida.

El presente proyecto resuelve la variante del TSP con capacidad CVRP utilizando la metodología de Búsqueda Tabú, apoyada en las técnicas “Clarke and Wright” para la búsqueda de la solución inicial y la técnica de Lin-Kernighan para la construcción del vecindario.

[8] Los primeros problemas similares al TSP fueron tratados por el irlandés Sir William Rowan Hamilton y por el matemático británico Thomas Penyngton Kirkman, pero la forma general del TSP hace su aparición en el año de 1930 con los estudios de Karl Menger en Viena y Harvard. Más tarde el problema fue promovido por Hassler Whitney y Merrill Flood en Princeton.

2.2 BÚSQUEDA TABÚ (TS)

Como se ha mencionado previamente, el núcleo del trabajo de investigación será la solución del problema de ruteo a través de la TS. Esta última es un procedimiento meta-heurístico estudiado por Fred Glover a finales de los 80's y principios de los 90's, cuya característica distintiva es el uso de memoria adaptativa y de estrategias especiales en la resolución de problemas.

La "Búsqueda tabú" o TS por sus siglas en inglés tiene sus antecedentes en métodos diseñados para cruzar cotas de factibilidad u optimalidad local tratadas como barreras en procedimientos clásicos e imponer y eliminar cotas sistemáticamente para permitir la exploración de regiones no consideradas en otro caso. Una característica distintiva de este procedimiento es el uso de memoria adaptativa y estrategias especiales de resolución de problemas. TS es el origen del enfoque basado en memoria y estrategia intensiva en la literatura de las meta heurísticas, en contraposición con los métodos que no tienen memoria o que solo usan una memoria débil basada en la herencia. TS es también responsable de enfatizar el uso de los diseños estructurados para explotar los patrones históricos de la búsqueda, de forma opuesta a los procesos que confían casi exclusivamente en la aleatorización.

Los principios fundamentales de la Búsqueda Tabú fueron elaborados en una serie de artículos a finales de los años 80 y principio de los 90 y han sido unificados en el libro "Tabu Search"[9]. El destacable éxito de la TS para resolver problemas de optimización duros (especialmente en aquellos que surgen aplicaciones del mundo real) ha causado una explosión de nuevas aplicaciones TS durantes los últimos años

La filosofía de la búsqueda tabú es derivar y explotar una colección de estrategias inteligentes para la resolución de problemas, basadas en procedimientos implícitos y explícitos de aprendizaje. El marco de memoria adaptativa de TS no solo explota la historia del proceso de resolución del problema, sino que también exige la creación de estructuras para hacer posibles tal explotación. La historia de resolución del problema se extiende a la experiencia ganada tras resolver múltiples instancias de una clase de problema uniendo TS con un enfoque de aprendizaje asociado llamado análisis de objetivo (Target analysis).

Las estructuras de memoria de la TS funcionan mediante referencia a cuatro dimensiones principales consistentes en la propiedad de ser recientes en frecuencia en calidad y en influencia estas dimensiones se fijan contra unos antecedentes de conectividad y estructuras lógicas.[10]

La Búsqueda Tabú se centra en tres tipos de memoria principalmente:

1. Memoria de corto plazo. Guarda la lista de los últimos movimientos con el fin de evitar regresar a un óptimo local de iteraciones pasadas.
2. Memoria de mediano plazo: Contiene los atributos comunes de las mejores soluciones que se han dado, para encaminar la búsqueda hacia ellos.
3. Memoria de largo plazo: Encamina la búsqueda hacia regiones aún no exploradas, para evitar encerrarse en un valle profundo.

A la búsqueda se le llama Tabú debido a que con el apoyo de las memorias mencionadas, marca ciertos movimientos como Tabú o prohibidos con el fin de no regresar a ellos y así diversificar la búsqueda. Además, la TS sigue un proceso de aprendizaje al captar los atributos más relevantes en las mejores soluciones y evitar regresar a soluciones que vayan en detrimento de dichos atributos.

---

[9] GLOVER F. LAGUNA M. "Tabú Search" Kluwer Academic Publishers 1997.

[10] Tomado de GLOVER Fred. MELIAN Belén, "*Tabu Search"* Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.19 (2003),pp. 29-48 ISSN: 1137-3601. © AEPIA (http://www.aepia.org/revista).

Para la implementación de la TS, la literatura recomienda seguir los pasos que se muestran a continuación:

2.2.1 PASO I

Selección de la solución inicial, debe ser una solución factible. La cual puede hallarse por un proceso de multi-arranque aleatorio[11], pero para iniciar la búsqueda por una solución de una calidad aceptable se puede enriquecer con los subprocesos de "Heurística de Clarke and Wright" o "El vecino más cercano".

2.2.2 PASO II

Elección del entorno o vecindario y generación de una nueva solución. La búsqueda Tabú supone que pueden generarse soluciones adyacentes y un entorno de soluciones partiendo de la solución inicial. Generalmente se hacen intercambios por pares o se utilizan técnicas como: Lin Kernighan, S & C, Mutación, Inserción, entre otras.

2.2.3 PASO III

Evaluación de la función objetivo. Dada la nueva solución se verifica que esta sea mejor que la solución anterior, de lo contrario, se realiza la búsqueda de una nueva solución[12].

2.2.4 PASO IV

Actualización de la mejor solución. Si la nueva solución es mejor que la solución encontrada anteriormente, se toma como el nuevo punto de partida y se repiten los pasos anteriores nuevamente.

2.2.5 PASO V

Criterio de parada. El proceso se terminará luego de un número de iteraciones dado, este parámetro será hallado luego de correr el proceso con varios números de iteraciones para hallar un óptimo.

La búsqueda Tabú realizará tres tipos de procesos con el fin de mejorar las soluciones, apoyada en los tipos de memoria que tiene:

1. Intensificación: Consiste en identificar las n (parámetro que deciden los investigadores) mejores soluciones para realizar el proceso a partir de cada una de ellas y explorar mejor esas regiones del espacio.
2. Diversificación. Consiste en realizar el proceso nuevamente a partir de una solución aleatoria para explorar regiones no estudiadas.

[11] El proceso de Multiarranque Aleatorio consiste en iniciar el proceso con diferentes soluciones iniciales seleccionadas aleatoriamente, y ejecutarlo varias veces.
[12] Volver al Paso II.

3. Criterio de aspiración: Consiste en quitar el rótulo de Tabú a algún movimiento solo si este conduce a una mejor solución. Es un parámetro definido de acuerdo al número de iteraciones que se realizarán y la calidad de la solución a la que conducirán.

2.2.6 ALGORITMO CLARKE AND WRIGHT

El algoritmo de Clarke and Wright también es llamado "Clarke-Wright Savings", dado que se fundamenta en los ahorros entre las distancias entre los clientes y el depósito. A continuación se explican los pasos para su desarrollo.

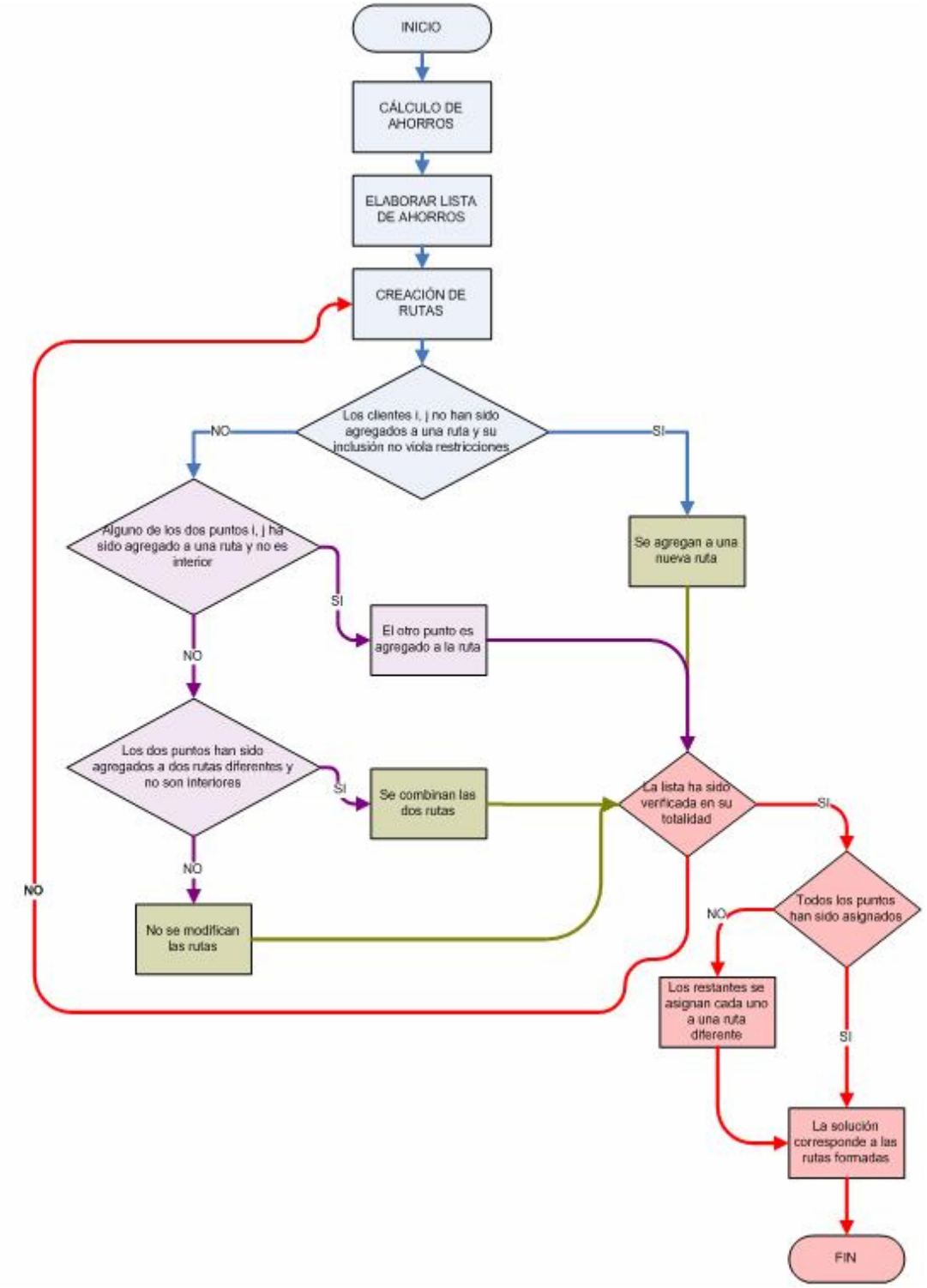
1. Se calculan los ahorros para cada par de puntos (clientes), este cálculo se hace de la siguiente forma: $s(i, j) = d(D, i) + d(D, j) - d(i, j)$ donde $d(D, i)$ corresponde a la distancia entre el depósito y el cliente i, $d(D, j)$ corresponde a la distancia entre el depósito y el cliente j, y finalmente $d(i, j)$ corresponde a la distancia entre el cliente i y el cliente j.
2. Se elabora una lista en orden descendente de todos los ahorros calculados en el paso anterior.
3. Para los ahorros en consideración, se incluye la pareja (i, j) en una ruta, si ninguna restricción será violada en esta inclusión y si:
    i. Ni i ni j han sido asignados a otra ruta previamente, en cuyo caso una nueva ruta será iniciada, donde se incluye a ambos i y j.
    ii. O si uno de los dos puntos (i o j) han sido incluidos en una ruta ya existente, y ese punto no es interior[13] a la misma, se agrega el "link" (i, j) a la ruta existente.
    iii. Si los dos puntos i y j han sido agregados a dos rutas diferentes, y ninguno es un punto interior, las dos rutas se unen en una sola[14].
4. Si las parejas de la lista de ahorros no han sido procesadas, se repite el paso 3 hasta que todas las parejas sean verificadas. De lo contrario se para el proceso, y la solución consiste en las diferentes rutas que han sido formadas.
5. Si todavía existen puntos que no han sido asignados en el paso 3, cada uno se asigna a una ruta diferente, donde el vehículo parte del depósito atiende al cliente y regresa al depósito.

[13] Un punto es INTERIOR a una ruta cuando NO es adyacente al depósito D. {1, 2,3,4,5,1} 1 representa al depósito, 2 y 5 son los puntos exteriores de la ruta y 3 y 4 son los puntos interiores de la ruta.

[14] Siempre y cuando no se viole ninguna restricción del problema original.

DIAGRAMA DE FLUJO DEL ALGORITMO CLARKE AND WRIGHT.

FIGURA 01. DIAGRAMA DE FLUJO DEL ALGORITMO CLARKE AND WRIGHT.



2.2.7 ALGORITMO LIN-KERNIGHAN

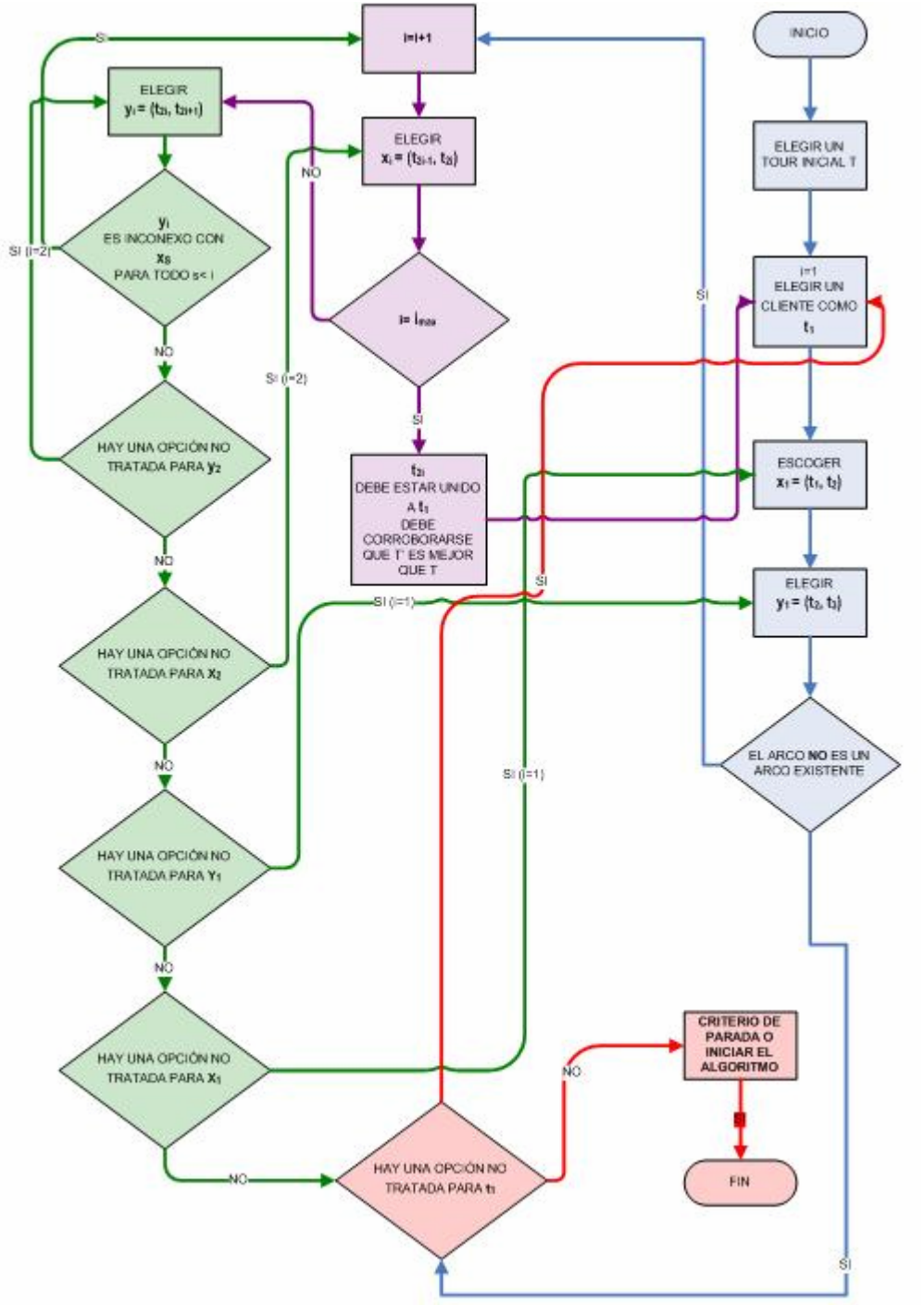
El algoritmo Lin Kernighan fue desarrollado hace treinta años, pero todavía es una de las mejores heurísticas para resolver el TSP. El algoritmo λ-opt. consiste en un proceso que a partir de una solución factible genera una solución más efectiva, reemplazando λ enlaces por λ enlaces nuevos, de tal forma que la nueva ruta sea más corta. Sin embargo, el algoritmo λ-opt. tiene la dificultad que para cada problema particular se debe elegir un λ adecuado que establezca un equilibrio entre la calidad de la solución y el tiempo de procesamiento.

Lin y Kernighan superaron esta dificultad, desarrollando un algoritmo que establece el valor de λ para cada etapa del proceso, esto lo logra examinando para valores ascendentes de λ si los intercambios posibles conducen a una mejor solución. Dada una solución factible el algoritmo realiza una cantidad de intercambios para hallar soluciones menos costosas, hasta llegar a un punto a partir de la cual no existan intercambios posibles que produzcan una mejor ruta. A continuación se enumeran los pasos para el desarrollo de este algoritmo.

1. Se elige una ruta o tour inicial T (en nuestro caso será la ruta resultante del algoritmo Clarke and Wright).
2. Se inicia el contador i (números de intercambios a realizar). $i = 1$. Se elige un cliente de forma aleatoria como t_1 (corresponde a uno de los vértices del primer arco a eliminar).
3. Se escoge el $x_1 = (t_1, t_2)$ (corresponde al primer arco que será reemplazado por uno nuevo, esta es una de las dos opciones a cada lado del t_1 escogido en el paso anterior).
4. Se elige el $y_1 = (t_2, t_3)$ (corresponde al arco que reemplazará al arco x_1 que se escogió en el paso anterior). y_i debe compartir un vértice con x_i y con x_{i+1}. y_1 debe cumplir con ciertas condiciones: el arco no puede ser un arco existente. Si no es posible realizar este paso se va al paso 12.
5. Se aumenta el contador $i = i+1$
6. Se elige $x_i = (t_{2i-1}, t_{2i})$. Este x_i debe cumplir con ciertas condiciones: debe ser inconexo con y_s para todo $s<i$. Para el i_{max} el t_{2i} debe estar unido a t_1 para que el T' sea un tour cerrado.
   Para el i_{max} debe corroborarse que T' es mejor que T, para luego hacer T = T' e ir al paso 2.
7. Si x_{i+1} existe, se escoge $y_i = (t_{2i}, t_{2i+1})$ de T, además y_i debe ser inconexo con x_s donde $s<i$.
   Si dicho y_i existe, ir al paso 5.
8. Si hay una alternativa no tratada para y_2 entonces se hace $i= 2$, e ir al paso 7.
9. Si hay una alternativa no tratada para x_2 entonces se hace $i= 2$, e ir al paso 6.
10. Si hay una alternativa no tratada para y_1 entonces se hace $i= 1$, e ir al paso 4.
11. Si hay una alternativa no tratada para x_1 entonces se hace $i= 1$, e ir al paso 3.
12. Si hay una alternativa no tratada para t_1, ir al paso 2.
13. Criterio de Parada (o ir al paso 1).

DIAGRAMA DE FLUJO ALGORITMO LIN KERNIGHAN

FIGURA 02. DIAGRAMA DE FLUJO DEL ALGORITMO LIN-KERNIGHAN.



3 DISEÑO DE LA SOLUCIÓN.

3.1 DESCRIPCIÓN DETALLADA DEL ALGORITMO BÚSQUEDA TABÚ

El algoritmo de Búsqueda Tabú resuelve el siguiente modelo de optimización.

S: Conjunto de clientes.
G: Grafo G:(V, E)
c_{ij} : Costo de ir de "i" a "j" $\forall i, j \in \{1,...n\} i < j$; costo en unidades de distancia.
Q: Capacidad de cada vehículo. Q>0.
q_i : Demanda de cada cliente. $0< q_i <Q$.
δ(S): Conjunto de caminos en G, con un solo vértice en S.
E(S): Conjunto de caminos en G con ambos vértices en S.
k(S): número mínimo de vehículos requeridos para servir a los clientes en S.
x_{ij} : Número mínimo de vehículos

Función Objetivo:

$$\min \sum_{i}^{N} \sum_{j}^{N} c_{ij} x_{ij}$$

Sujeto a:

$$x(\delta(\{i\})) = 2 \quad (i = 1,...,n) \qquad (1)$$

$$x(\delta(S)) \geq 2k(S) \quad |S| > 2 \qquad (2)$$

$$x_{ij} \in \{0,1\} \quad (1 \leq i < j \leq n) \qquad (3)$$

$$x_{ij} \in \{0,1,2\} \quad (i = 0,\ j = 1,...,n) \qquad (4)$$

(1) Asegura que los clientes son visitados una sola vez.

(2) Impone las restricciones de capacidad de los vehículos y asegura que las rutas están conectadas.

(3) Y (4) Condiciones de integralidad.

(4) Permite la existencia de rutas en las cuales un vehículo atienda a un solo cliente.

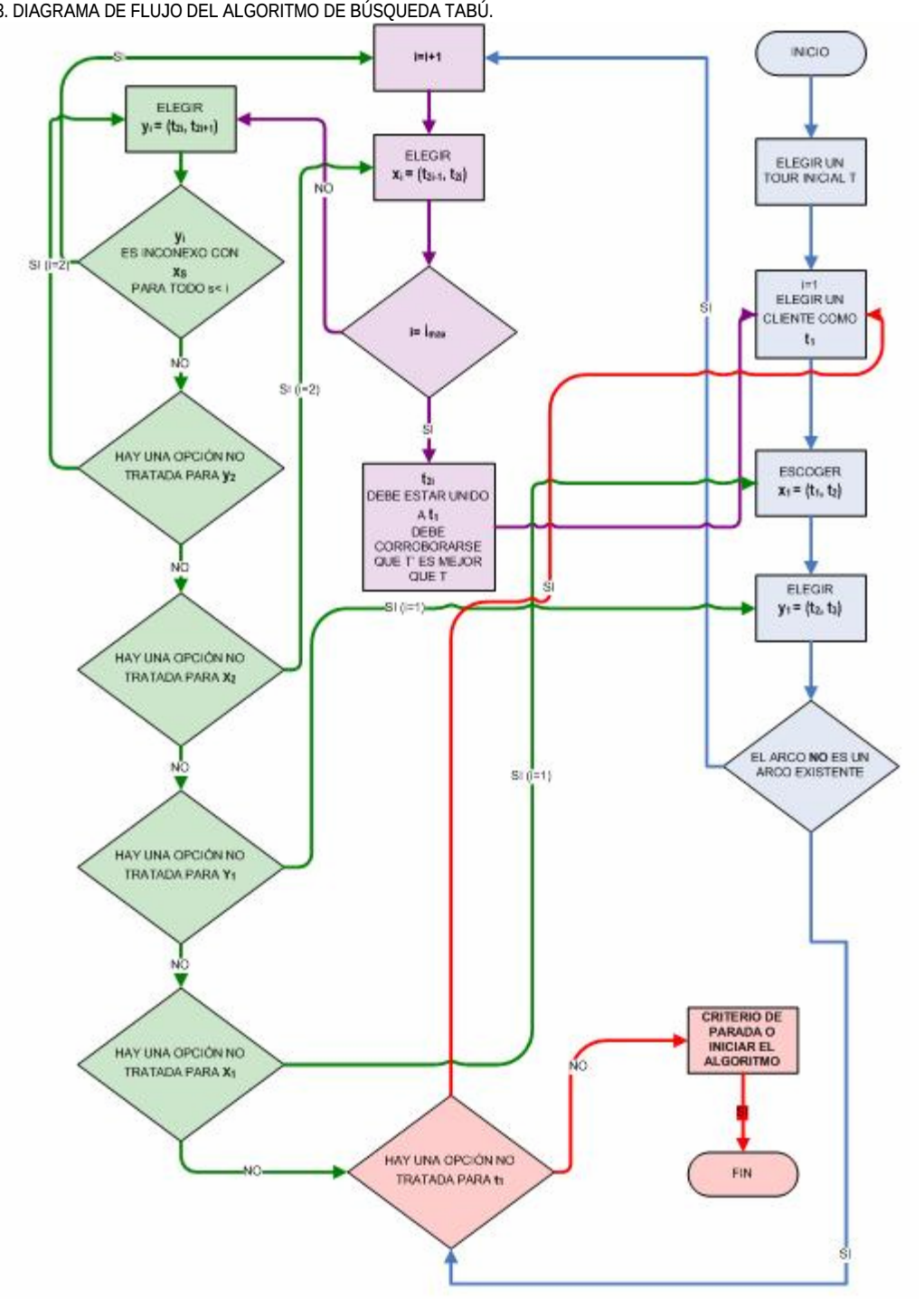
A continuación se presentan las diferentes etapas que comprenden el algoritmo de Búsqueda Tabú:

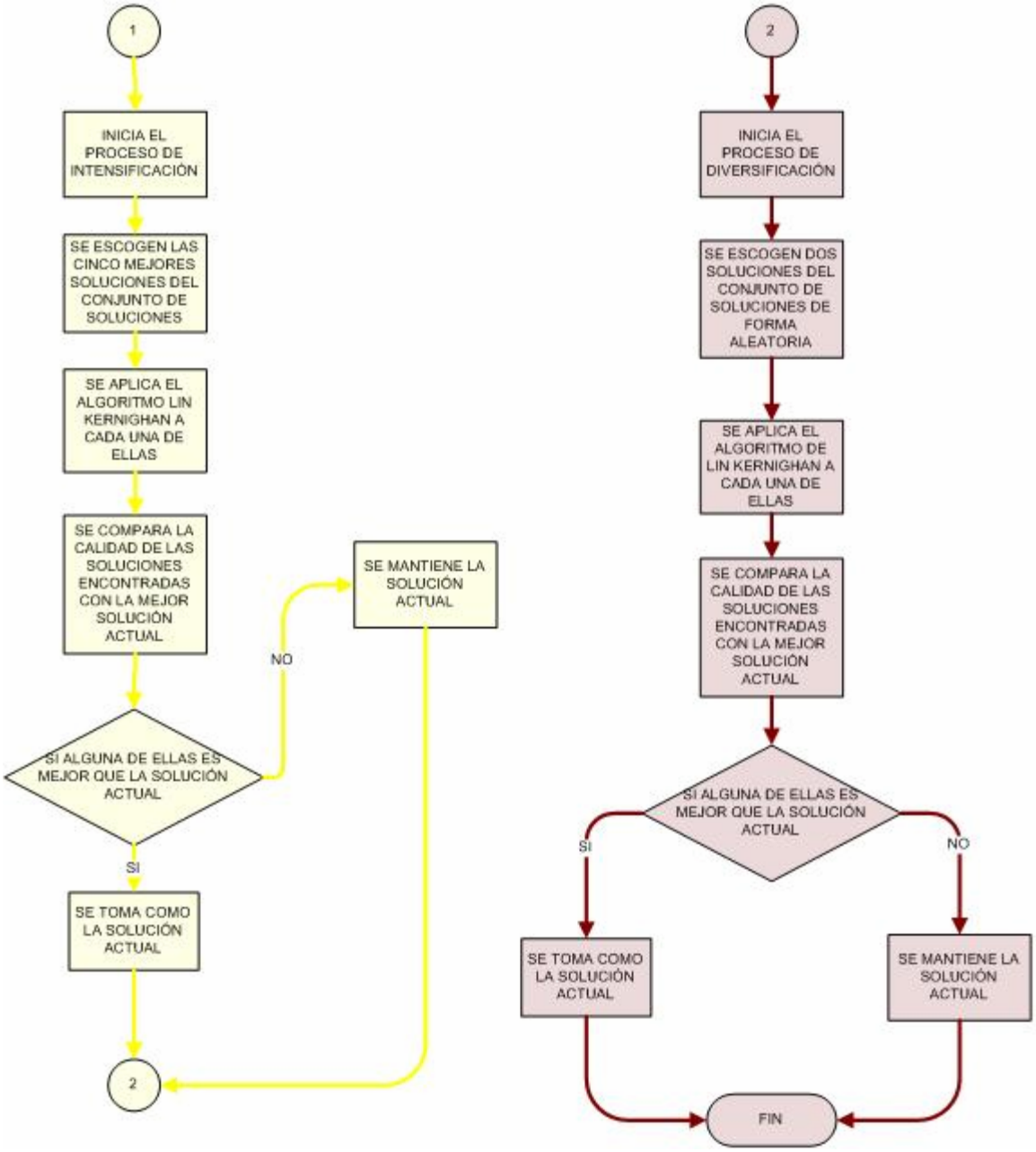
1. Se parte de la solución inicial obtenida gracias al algoritmo Clarke and Wright descrito en la sección 2.2.6.
2. A partir de la solución inicial se construye el vecindario utilizando el algoritmo Lin-Kernighan descrito en la sección 2.2.7.
3. Cuando se obtiene una solución por medio del algoritmo Lin-Kernighan, se verifica que esta no viole la restricción de capacidad, de ser así, se agregarán viajes a los depósitos donde los vehículos no den abasto.

4. Luego de corroborar que la nueva solución cumple con las condiciones de capacidad, se verifica el costo de la misma y se compara con la solución actual.
5. Si la nueva solución es mejor que la solución actual (presenta menores costos), entonces se proceda a corroborar si es considerada como tabú en la matriz.
6. Se elaborará una matriz del tipo m (i,j) donde cada elemento representará el arco (i,j) , este elemento será un contador que se activará cada vez que este arco desaparezca de la solución actual (tabú activo), y permanecerá activo tantas iteraciones como se hayan definido como parámetro al inicio del algoritmo, además debe tenerse en cuenta el tamaño de la lista tabú (el cual ha sido definido al inicio del algoritmo también). Esto garantiza que el arco no será incluido en un determinado número de iteraciones a la solución, con lo cual se evita realizar ciclos en el mismo espacio de solución. Esta matriz se actualizará cada vez que se obtenga una solución factible en el algoritmo Lin-Kernighan. Si la solución no es considerada como tabú, se considera como solución inicial en la próxima iteración del algoritmo Lin Kernighan, y se guarda en el conjunto de soluciones.
7. Si la solución es considerada como tabú pero es mejor que la solución actual, se aplica el criterio de aspiración. El cual permite admitirla como válida, dado que presenta mejoras en costos de la Función Objetivo.
8. El proceso termina con un número determinado de iteraciones.
9. Se toman las cinco mejores soluciones del conjunto de soluciones y a estas se les aplica el proceso de intensificación, el cual consiste en aplicar el algoritmo a partir de cada una de ellas para explorar de manera más detallada las regiones del espacio que presentan las soluciones de alta calidad.
10. Para evitar quedarse atrapado en una sola zona del espacio de soluciones, se escoge una solución inicial aleatoria y se aplica el proceso a ésta. Este proceso se conoce como el proceso de diversificación, que nos remite a otras zonas para complementar la búsqueda.
11. El criterio de parada se aplica luego de un número de iteraciones dado.

3.1.1 DIAGRAMA DE FLUJO DEL ALGORITMO DE LA BÚSQUEDA TABÚ

FIGURA 03. DIAGRAMA DE FLUJO DEL ALGORITMO DE BÚSQUEDA TABÚ.




1
INICIA EL PROCESO DE INTENSIFICACIÓN
SE ESCOGEN LAS CINCO MEJORES SOLUCIONES DEL CONJUNTO DE SOLUCIONES
SE APLICA EL ALGORITMO LIN KERNIGHAN A CADA UNA DE ELLAS
SE COMPARA LA CALIDAD DE LAS SOLUCIONES ENCONTRADAS CON LA MEJOR SOLUCIÓN ACTUAL
SI ALGUNA DE ELLAS ES MEJOR QUE LA SOLUCIÓN ACTUAL
NO
SE MANTIENE LA SOLUCIÓN ACTUAL
SI
SE TOMA COMO LA SOLUCIÓN ACTUAL
2
2
INICIA EL PROCESO DE DIVERSIFICACIÓN
SE ESCOGEN DOS SOLUCIONES DEL CONJUNTO DE SOLUCIONES DE FORMA ALEATORIA
SE APLICA EL ALGORITMO DE LIN KERNIGHAN A CADA UNA DE ELLAS
SE COMPARA LA CALIDAD DE LAS SOLUCIONES ENCONTRADAS CON LA MEJOR SOLUCIÓN ACTUAL
SI ALGUNA DE ELLAS ES MEJOR QUE LA SOLUCIÓN ACTUAL
SI
NO
SE TOMA COMO LA SOLUCIÓN ACTUAL
SE MANTIENE LA SOLUCIÓN ACTUAL
FIN

3.2 ESPECIFICACIÓN DE REQUERIMIENTOS FUNCIONALES

Este documento da a conocer las principales características, funcionalidades y servicios que prestará el programa de computadora en que se implementará el algoritmo diseñado. Se debe considerar como un manual de desarrollo, debido a que en él se encuentra explícita la información requerida para construir el programa que cumpla con los objetivos trazados en el plan de proyecto y que presente las características especificadas.

Esta especificación de requisitos, muestra lo que hará y lo que no hará la herramienta software que se desarrollará, por ello, es de vital importancia la lectura del mismo por parte del usuario y de la comunidad técnica. El manual de Usuario se encuentra documentado en los anexos, este materializa todas las condiciones y restricciones presentadas en esta sección.

3.2.1 DESCRIPCIÓN GENERAL

El programa que se creará para implementar la metodología de la meta-heurística "Búsqueda Tabú" para solucionar el problema de ruteo CVRP que es objetivo principal de este proyecto, se le denominará CVRP Solver, este programa permitirá determinar una ruta de entrega de buena calidad en corto tiempo para el problema del ruteo con capacidad, basado en los parámetros a saber: capacidad de los vehículos, número de clientes, coordenadas X, Y de ubicación de los clientes y del deposito, y demanda de cada cliente, Esta herramienta de software será de ayuda para aquellas personas interesadas en el tema (estudiantes, profesionales, industrias que involucren procesos logísticos de entrega, etc.), dado que les permitirá trazar planes y tomar acciones de una manera más rápida y confiable.

Alcance del programa

Como se mencionó anteriormente, el programa de computadora se encargará de implementar la metodología de la meta-heurística "Búsqueda Tabú" para solucionar el problema de ruteo CVRP, dados determinados parámetros. La construcción de la solución estará basada en una ruta inicial[15] que será obtenida a través del algoritmo crecimiento Clarke and Wright que fue presentado en la sección 4.2.6, el algoritmo de refinamiento será Lin – Kernighan, este utilizará la ruta inicial como parámetro de entrada para la construcción del vecindario y exploración del espacio de soluciones, la metodología de "Backtracking jump" Búsqueda tabú nos permitirá por medio de los procesos de intensificación y diversificación evitar óptimos locales para entregar una solución basada en una exploración exhaustiva del espacio.

Usuarios Finales

Básicamente, el programa de computadora está destinado a estudiantes, profesores e integrantes de la comunidad científica interesados en procesos logísticos de entrega, metodologías heurísticas, técnicas avanzadas de investigación de operaciones, problemas de optimización combinatoria, y en general, a todas aquellas personas que puedan utilizar este desarrollo como punto de partida para futuros proyectos.

Sin embargo, la utilización de esta herramienta requiere de conocimientos previos en el tema, debido a que la misma podría no ser lo suficientemente asistida para cualquier tipo de usuario. El objetivo de este proyecto es la creación del algoritmo de solución, más no el programa de computadora que lo implementará.

[15] La ruta inicial obtenida a través de Clarke and Wright se puede considerar de buena calidad respecto a una ruta inicial aleatoria

Dependencias y Supuestos

El programa de computadora a desarrollar utilizará todas y cada una de las técnicas mencionadas en el alcance del programa de forma invariante, los límites y topes del algoritmo como tamaño máximo de lista tabú y numero máximo de clientes que podrán ser ingresados al sistema, estos serán implementados como parámetros de diseño debido a que fueron definidos así dentro del alcance del presente proyecto.

3.2.2 REQUERIMIENTOS DE INTERFACES EXTERNAS

Este capítulo muestra las características y requerimientos de la herramienta software para con el entorno.

Interfaces de Usuario

La interfaz de usuario será implementada mediante el uso de ventanas gráficas usuales del sistema operativo que aloje a la máquina virtual Java JRE 1.5. El manejo de las utilidades del programa de computadora se podrá realizar a través de un Mouse o de teclas de acceso rápido utilizando un teclado convencional. Para navegar a través de los componentes de una ventana se utilizará la tecla de tabulación (TAB).

Básicamente, el programa de computadora desplegará una ventana inicial en pantalla completa con un menú en la parte superior, por el cual se podrá acceder a las diferentes funcionalidades a través de diversos ítems y una barra de herramientas que permitirá el acceso a las mismas.

Interfaces de Hardware

El programa de computadora estará adecuado para trabajar en equipos que cumplan como mínimo con los siguientes requerimientos: un teclado convencional, un Mouse, un monitor que permita una resolución de 800x600dpi, una CPU con procesador compatible x86, 512 MB de memoria física y 20MB de espacio de almacenamiento requeridos por el sistema operativo para las operaciones de intercambio y paginación.

Interfaces de Software

Para el funcionamiento de este programa de computadora, es necesario que el equipo donde se instale, tenga un sistema operativo que soporte la máquina virtual de Java®, la versión 1.5 del ambiente de ejecución de Java (JRE 1.5).

3.2.3 REQUERIMIENTOS

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

1. Especificación de requisitos del módulo de datos y de la interfaz de usuario.

RQ-1.1 El programa de computadora no necesitará de una base de datos para el manejo de la información de requerida para la ejecución.
RQ-1.2 El programa de computadora utilizará archivos para cargar información de los datos del problema.
RQ-1.3 El programa de computadora utilizará archivos para grabar información de los datos del problema.
RQ-1.4 El programa de computadora podrá utilizar archivos para grabar información de la solución del problema.

Interfaz gráfica

RQ-1.5 El programa de computadora debe estar basado en un sistema de ventanas gráficas o formularios.

RQ-1.6 Las ventanas que se desplegarán dentro del programa de computadora, deben ser del tipo Modal, esto es, no se pueden tener activas más de una ventana a la vez.
RQ-1.7 Cada ventana del programa de computadora, debe tener un breve descriptivo de las operaciones permitidas dentro de ella.
RQ-1.8 Las ventanas del programa de computadora deben tener las mismas características entre si, tales como colores, tamaño de fuentes, estilos y botones.
RQ-1.9 El programa de computadora debe permitir modificar la información de los datos del problema para la generación de una nueva solución.
RQ-1.10 La información que es requerida para el ingreso exitoso de un problema al aplicativo son mostrados en la tabla siguiente:

TABLA 01. VALORES REQUERIDOS PARA EL INGRESO DEL PROBLEMA EN EL APLICATIVO

PARÁMETRO	*DESCRIPCIÓN*	*REQUISITOS*
Número de clientes	Corresponde al número de destinos a atender.	De RQ 1.10.1 a RQ 1.10.2
Capacidad de vehículos	Cantidad máxima de producto que puede transportar un vehículo.	De RQ 1.10.3 a RQ 1.10.4
Ubicación depósito X	Coordenada X de la ubicación del depósito.	De RQ 1.10.5 a RQ 1.10.6
Ubicación depósito Y	Coordenada Y de la ubicación del depósito.	De RQ 1.10.7 a RQ 1.10.8
Tabla de clientes	Arreglo de datos con la información de los clientes	De RQ 1.10.9 a RQ 1.10.11
Coordenada X	Corresponde a la coordenada X de ubicación de cada cliente o destino a atender.	De RQ 1.10.12 a RQ 1.10.14
Coordenada Y	Corresponde a la coordenada Y de ubicación de cada cliente o destino a atender.	De RQ 1.10.12 a RQ 1.10.14
Demanda	Corresponde al número de productos a entregar a cada destino o cliente.	De RQ 1.10.12 a RQ 1.10.14

RQ-1.10.1 El Número de clientes debe ser un parámetro numérico no mayor a 5 caracteres.
RQ-1.10.2 La captura del Número de clientes no debe permitir el ingreso de caracteres que no generen como resultado un valor entero.
RQ-1.10.3 La capacidad de vehículos debe ser un parámetro numérico no mayor a 5 caracteres..
RQ-1.10.4 La capacidad de vehículos no debe permitir el ingreso de caracteres que no generen como resultado un valor entero.
RQ-1.10.5 La Ubicación de depósito X debe ser un parámetro numérico no mayor a 10 caracteres.
RQ-1.10.6 La Ubicación de depósito X no debe permitir el ingreso de caracteres que no generen como resultado un valor flotante.
RQ-1.10.7 La Ubicación depósito Y debe ser un parámetro numérico no mayor a 10 caracteres.

RQ-1.10.8 La Ubicación depósito Y no debe permitir el ingreso de caracteres que no generen como resultado un valor flotante.
RQ-1.10.9 Los clientes serán capturados a través de una tabla.
RQ-1.10.10. La tabla de captura de clientes contendrá tantas filas como clientes hayan sido ingresados en la variable "Número de clientes"
RQ-1.10.11 La tabla de captura de clientes tendrá 3 columnas, de las cuales la primera será la "Coordenada X", la segunda "Coordenada Y" y la tercera "Demanda".
RQ-1.10.12 Cada valor de las celdas de la tabla de captura de clientes será un valor flotante.
RQ-1.10.13 La nomenclatura que se utilizará a lo largo del desarrollo del software para referirse a un cliente i es la siguiente:

TABLA 02. NOMENCLATURA DE REFERENCIA DE UN CLIENTE.

PARÁMETRO	*DESCRIPCIÓN*
Clientes[i][0]	Coordenada X del cliente i, donde i es un valor numérico entre 1 y el número determinado en la variable "Número de clientes"
Clientes[i][1]	Coordenada Y del cliente i, donde i es un valor numérico entre 1 y el número determinado en la variable "Número de clientes".
Clientes[i][2]	Demanda del cliente i, donde i es un valor numérico entre 1 y el número determinado de en la variable "Número de clientes".

RQ-1.10.13.1 Los clientes están pertenecen al dominio i= 1,…, "Número de clientes". La ubicación i=0, será el espacio de memoria utilizado para la ubicación del depósito.
RQ-1.10.14 Los parámetros especificados en los requisitos RQ-1.10.1, RQ-1.10.3, RQ-1.10.5, RQ-1.10.7 y RQ-1.10.9 pueden ser modificados desde la interfaz de usuario para la obtención de una nueva solución.
RQ-1.11 El Menú del programa de computadora deberá contener los ítems que son enumerados en la siguiente tabla:

TABLA 03. ÍTEMS DEL MENÚ DEL PROGRAMA.

ÍTEMS	*DESCRIPCIÓN*	*REQUISITOS*
Archivo	Este ítem permitirá el manejo de la información de los datos del problema.	De RQ 1.11.1 a RQ 1.11.2
Solución	Este ítem permitirá el manejo de la información relativa a la ejecución del algoritmo de solución del programa, a los parámetros para la obtención de los mismos y a la información de esta solución.	RQ 1.11.3

RQ-1.11.1 La pestaña "Archivo" de la barra de menú podrá ser accedida a través del Mouse y con el teclado con las teclas de acceso rápido ALT+A.
RQ-1.11.2 Los ítems que estarán contenidos dentro de la pestaña de menú "Archivo" son:

TABLA 04. ÍTEMS DEL MENÚ ARCHIVO.

ÍTEMS	*DESCRIPCIÓN*	*REQUISITOS*
Nuevo	Este ítem permitirá ingresar un nuevo problema al programa a través del teclado.	De RQ 1.11.2.1 a RQ 1.11.2.3
Abrir	Este ítem permitirá ingresar un nuevo problema al programa a través de un archivo.	De RQ 1.11.2.4 a RQ 1.11.2. 6
Modificar	Este ítem permitirá modificar los datos actuales del problema ingresado previamente..	De RQ 1.11.2.7 a RQ 1.11.2.8
Guardar	Este ítem permitirá tomar la información que se encuentra ingresada como dato del problema y almacenarla en un archivo.	De RQ 1.11.2.9 a RQ 1.11.2.10
Guardar Como	Este ítem permitirá tomar la información que se encuentra ingresada como dato del problema y almacenarla en un archivo.	De RQ 1.11.2.11 a RQ 1.11.2.12
Salir	Este ítem permitirá salir de la aplicación.	De RQ 1.11.2.13 a RQ 1.11.2.14

RQ-1.11.2.1 La opción “Nuevo” se encarga de abrir el formulario de captura de datos, para ingresar un nuevo problema al programa.
RQ-1.11.2.2 La opción “Nuevo” será ejecutada a través del Mouse y del teclado con las teclas de acceso rápido CTRL+N.
RQ-1.11.2.3 Al ingresar a la opción “Nuevo”, si el sistema detecta que un problema ya ha sido ingresado y no han sido guardados los cambios recientes, desplegará un mensaje emergente de confirmación en donde se preguntará si se desea almacenar los datos actuales, con los botones de respuesta SI, NO y CANCELAR.
RQ-1.11.2.3.1 En caso de que el botón pulsado sea SI, el sistema automáticamente almacenará en Archivo los datos del problema ingresado, y luego mostrara el formulario de captura de datos para el nuevo problema.
RQ-1.11.2.3.2 En caso de que el botón pulsado sea NO, el sistema automáticamente eliminará los datos del anterior problema y desplegará el formulario de captura de datos para el ingreso del nuevo problema.
RQ-1.11.2.3.3 En caso de que el botón pulsado sea CANCELAR, el sistema automáticamente cancelará el mandato ingresado, manteniendo la información del problema anterior.
RQ-1.11.2.4 La opción “Abrir” se encarga de cargar la información de un nuevo problema, a través de un nuevo archivo de la extensión *.vrp
RQ-1.11.2.5 Al ingresar a la opción “Abrir”, si el sistema detecta que un problema ya ha sido ingresado y no han sido guardados los cambios recientes, desplegará un mensaje emergente de confirmación que se especifica en el RQ-1.11.2.3.
RQ-1.11.2.5.1 En caso de que el botón pulsado sea SI, el sistema automáticamente almacenará en Archivo los datos del problema ingresado, y luego abrirá la ventana de apertura de archivo con el estándar del sistema operativo residente, para que el usuario busque el archivo de extensión *.vrp, lo seleccione y sea cargado en el programa.
RQ-1.11.2.5.2 En caso de que el botón pulsado sea NO, el sistema automáticamente eliminará los datos del anterior problema, y luego abrirá la ventana de apertura de archivo con el estándar del sistema operativo residente, para que el usuario busque el archivo de extensión *.vrp, lo seleccione y sea cargado en el programa.
RQ-1.11.2.5.3 En caso de que el botón pulsado sea CANCELAR, el sistema automáticamente cancelará el mandato ingresado, manteniendo la información del problema anterior.

RQ-1.11.2.6 La opción “Abrir” será ejecutada a través del Mouse y del teclado por medio de las teclas de acceso rápido CTRL+A
RQ-1.11.2.7 La opción “Modificar” se encarga de Abrir el formulario de captura de datos con la información del problema que actualmente se encuentra cargado en el programa, para que el usuario pueda modificar la información que crea pertinente.
RQ-1.11.2.8 La opción “Modificar” será ejecutada a través del Mouse y del teclado por medio de las teclas de acceso rápido CTRL+M.
RQ-1.11.2.9 La opción “Guardar” se encarga de Guardar la información del problema que actualmente se encuentra cargada en el programa, para que el usuario pueda utilizarlo posteriormente.
RQ-1.11.2.10 La opción “Guardar” será ejecutada a través del Mouse y del teclado por medio de las teclas de acceso rápido CTRL+ G.
RQ-1.11.2.11 La opción “Guardar Como” se encargar de Guardar la información que actualmente se encuentra cargada en el programa, en un archivo.
RQ-1.11.2.12 La opción “Guardar Como” será ejecutada a través del Mouse.
RQ-1.11.2.13 La opción “Salir” se encarga de salir y cerrar el programa.
RQ-1.11.2.13.1 Al ingresar a la opción “Salir”, si el sistema detecta que un problema ya ha sido ingresado y no han sido guardados los cambios recientes, desplegará un mensaje emergente de confirmación en donde se preguntará si se desea almacenar los datos actuales, con los botones de respuesta SI, NO y CANCELAR.
RQ-1.11.2.13.2 En caso de que el botón pulsado sea SI, el sistema automáticamente almacenará en Archivo los datos del problema ingresado, y luego mostrara el formulario de captura de datos para el nuevo problema.
RQ-1.11.2.13.3 En caso de que el botón pulsado sea NO, el sistema automáticamente eliminará los datos del anterior problema y cerrará el programa.
RQ-1.11.2.14 La opción “Salir” será ejecutada a través del Mouse y del teclado por medio de las teclas de acceso rápido CTRL+ Alt. + S.
RQ-1.11.3 Los ítems que estarán contenidos dentro de la pestaña de menú “Solución” son:

TABLA 05. ÍTEMS DEL MENÚ SOLUCIÓN

ÍTEMS	*DESCRIPCIÓN*	*REQUISITOS*
Ejecutar	Este ítem ejecutará el algoritmo propuesto en este plan de proyecto para la solución del problema dado	De RQ 1.11.3.1 a RQ 1.11.3.4
Parámetros	Este ítem presentará un formulario en el que se podrán modificar los parámetros de Numero de iteraciones, Numero de intercambios, Tamaño de la lista tabú, tamaño de la población.	RQ 1.11.3.5
Ver Solución	Este ítem mostrará el costo de la solución y la secuencia de reparto para la solución propuesta.	RQ 1.11.3.6

RQ-1.11.3.1 La opción “ejecutar” deberá ser capaz de activar el hilo encargará de ejecutar el algoritmo propuesto dentro del presente plan para la solución ingresada.
RQ-1.11.3.2 La opción “ejecutar” deberá inactivar el menú del programa mientras la solución se encuentra en proceso de búsqueda.
RQ-1.11.3.3 La opción “ejecutar” debe activar las opciones del menú luego de encontrada la solución del problema planteado.
RQ-1.11.3.4 La opción “ejecutar” debe auto-activar la opción Ver solución del menú luego de encontrada la solución del problema planteado.

RQ-1.11.3.5 La opción “parámetros” debe mostrar un formulario en el cual se debe poder escoger los parámetros del sistema: Numero de iteraciones, Numero de intercambios, Tamaño de la lista tabú, tamaño de la población

RQ-1.11.3.6 La opción “Ver solución” debe presentar al usuario la información de la solución obtenida por el algoritmo propuesto como tiempo de procesamiento de dicha solución, costo de la solución y la secuencia de reparto.

2. Del Algoritmo de Solución

RQ-2.1 El programa de computadora debe validar que la información que sea registrada com o datos del problema sea consistente con los datos esperados a saber

RQ-2.1.1 El número de clientes debe ser mayor que cero.

RQ-2.1.2 Debe haber un único deposito.

RQ-2.1.3 La capacidad de cada vehiculo no puede ser excedida por la demanda de un cliente.

RQ-2.2 El sistema debe ser modular y compuesto por un módulo de datos o procesos y un modulo de interfaz.

RQ-2.3 El módulo de proceso debe estar compuesto de una clase para los datos de del problema, una clase para la solución, una clase para obtención de la respuesta inicial a través del algoritmo Clarke and Wright y una clase para la obtención de la respuesta final a través del algoritmo de Búsqueda Tabú

RQ-2.3.1 La clase de datos del problema debe tener las variables necesarias para almacenar las la información requerida por la interfaz descrita en el RQ- 1.10.

RQ-2.3.2 La clase solución debe contener información de Costo de la ruta obtenida en el proceso de solución. El costo de la ruta será la sumatoria de las distancias que componen cada una de los arcos del grafo en donde los vértices son los clientes y el depósito.

RQ-2.3.3 La clase solución debe tener la composición de la ruta respuesta almacenada como un vector de una dimensión en donde en cada uno de los ubicaciones puede ser un cliente o el depósito.

RQ-2.3.4 La clase solución debe tener una variable que corresponda al tiempo de proceso del último algoritmo ejecutado, este tiempo será medido en milisegundos y tomado desde el momento en que se inicia el algoritmo hasta cuando se termina.

RQ-2.3.5 La clase Clarke and Wright debe ser capaz de ejecutar el algoritmo descrito en el diagrama de flujo de la figura 01.

RQ-2.3.6 La clase Clarke and Wright debe recibir como parámetros un objeto de la clase datos del problema y un objeto de la clase Solución, del primero se toma la información para la obtención de la solución que será almacenada en el segundo.

RQ-2.3.7 La clase de Búsqueda Tabú, debe ser capaz de ejecutar el algoritmo descrito en el diagrama de flujo de la figura 03.

RQ-2.3.8 La clase Búsqueda Tabú debe recibir como parámetros un objeto de la clase datos del problema y un objeto de la clase Solución, del primero se toma la información básica del problema y del segundo la solución inicial para la obtención de una mejor respuesta que será almacenada en la misma variable

RQ-2.3.9 La clase Solución que recibe como parámetro la clase Búsqueda Tabú debe haber sido obtenida a través de la clase Clarke and Wright.

RQ-2.4 El proceso de obtención de respuesta del problema (ejecución del algoritmo Clarke and wright y ejecución de la Búsqueda Tabú) debe ser ejecutado como un hilo independiente al que procesa los eventos de la interfaz gráfica para evitar el congelamiento de la aplicación mientras el proceso genera la información.

RQ-2.5 El módulo de proceso debe ser capaz de controlas excepciones que sea despertadas durante la búsqueda de la solución para evitar la cancelación abrupta de la ejecución del programa de computadora que lo ejecute.

RQ-2.6 El módulo de proceso debe ser independiente de la interfaz que observa el usuario.

4 PRUEBAS DEL ALGORITMO

En esta sección se presentan los mejores resultados arrojados por el algoritmo y se contrastan con los resultados obtenidos de la literatura y que fueron documentados en el plan del proyecto.

4.1 PRESENTACIÓN DE LOS RESULTADOS

A continuación se presentan los resultados de la comparación de resultados obtenidos por el algoritmo propuesto dentro de este trabajo respecto a la mejor solución de la literatura y a la solución inicial Clarke and Wright. La salida de la herramienta software de las mejores soluciones para cada uno de los problemas se presentan en la sección de los anexos, desde el anexo 01 al 07.

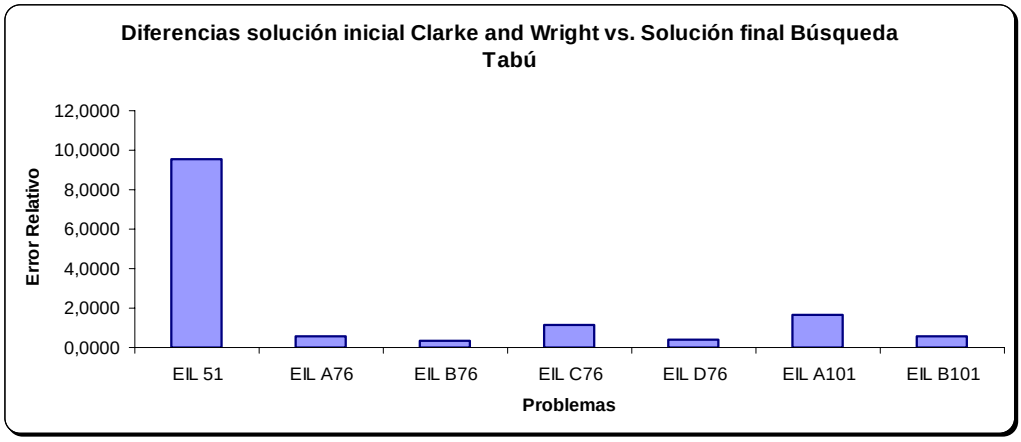
Inicialmente se comparan los resultados de la solución inicial Clarke and Wright con los datos de la solución final, con el fin de establecer en cual de los problemas se mejora en mayor medida la solución inicial. Dado los resultados anteriores es posible inferir que para el problema EIL 51 se obtiene la mayor mejora, presentando el mayor error relativo equivalente a [9.5294%]. En todos los problemas la solución final es menos costosa que la solución inicial.

TABLA 06. COMPARACIÓN RESULTADOS ALGORITMO CONTRA LA SOLUCIÓN INICIALCLARKE AND WRIGHT.

Problema	Mejor Resultado Algoritmo	Solución inicial Clarke and Wright	Solución inicial vs. final[16]
EIL 51	**528,9249**	**584,6372084**	**9,5294**
EIL A76	902,0872	907,392376	0,5847
EIL B76	1051,0956	1054,599368	0,3322
EIL C76	785,6270	794,740514	1,1467
EIL D76	735,0818	738,1324473	0,4133
EIL A101	876,8008	891,8278166	1,6850
EIL B101	1132,2389	1139,071309	0,5998

[16] Esta columna corresponde al Error Relativo de la solución con respecto a la solución inicial Clarke and Wright. Este se calcula de la siguiente manera E= 100*[abs(Solución Inicial)-(Solución final)]/(Solución Inicial).

FIGURA 04. ERROR RELATIVO DE LA SOLUCIÓN FINAL CON RESPECTO A LA SOLUCIÓN INICIAL CLARKE AND WRIGHT.



Luego de comparar los resultados finales con la solución inicial, se procede a realizar la comparación con los mejores resultados de la literatura. De esta comparación se puede precisar que la solución para cada problema es menos eficiente que las soluciones de la literatura, es decir, que presentan un costo (Unidades de Distancia) mayor. De acuerdo a los resultados, el problema que más se acerca a la solución de la literatura es el problema EIL 51, dado que presenta el menor error relativo correspondiente a [2.7989%]. Además el problema que se aleja más de la solución de la literatura es el EIL B76, con un error relativo de [47,3154%], este es un valor extremo dado que los errores de los demás problemas oscilan entre el 10 y el 15%.

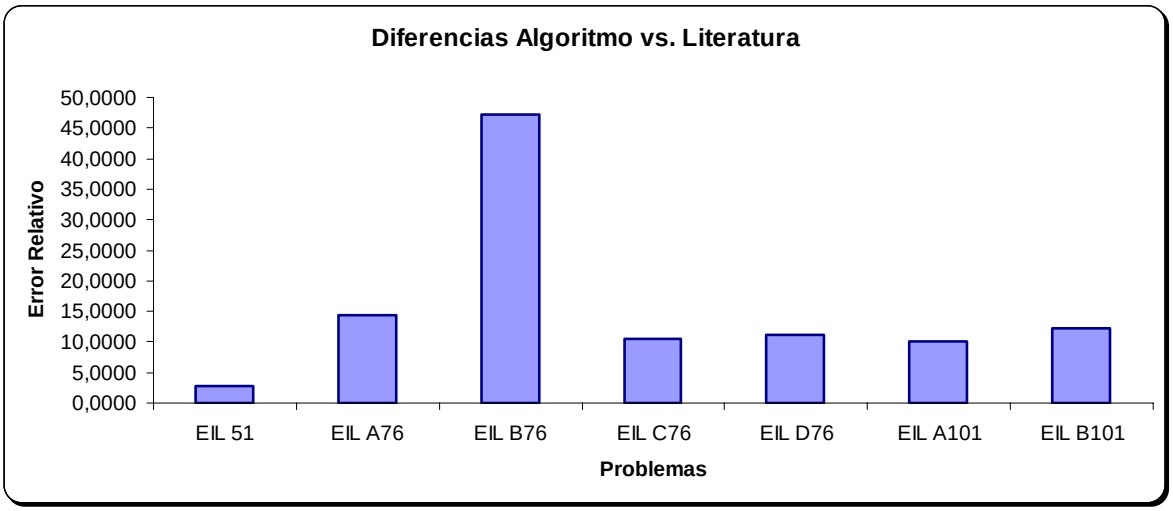
TABLA 07. COMPARACIÓN RESULTADOS ALGORITMO CONTRA RESULTADOS DE LA LITERATURA

Problema	Mejor Resultado Literatura	Mejor Resultado Algoritmo	Diferencia[17]	Error Relativo respecto a solución Literatura[18]
EIL 51	**514,5240**	**528.9249**	**14,4009**	**2,7989**
EIL A76	789,4160	902,0872	112,6712	14,2727
EIL B76	713,5000	1051,0956	337,5956	47,3154
EIL C76	711,1700	785,6270	74,4570	10,4696
EIL D76	661,2990	735,0818	73,7828	11,1572
EIL A101	796,3140	876,8008	80,4868	10,1074
EIL B101	1008,1460	1132,2389	124,0929	12,3090

[17] La Diferencia corresponde a D= [Mejor solución Literatura] – [Mejor solución Algoritmo].

[18] El Error relativo corresponde a E= [(Mejor solución Literatura) – (Mejor solución Algoritmo)] * 100/(Mejor solución Literatura)

FIGURA 05. ERROR RELATIVO DE LA MEJOR SOLUCIÓN DEL ALGORITMO CON RESPECTO A LA MEJOR DE LA LITERATURA.



Adicionalmente se corrobora la consistencia del algoritmo, dado que al evaluar las calidades de los diferentes resultados para problemas en los cuales solo varía la capacidad, se obtienen mejores resultados para los problemas con mayor capacidad. Estos problemas son EIL A76, EIL B76, EIL C76 y EIL D76; EIL A101 y EIL B101. A continuación se presentan dos figuras que muestran este comportamiento.

FIGURA 06. COMPARACIÓN DEL CAMBIO DE LA VARIABLE RESPUESTA CON RESPECTO A LA CAPACIDAD. PROBLEMAS EIL A76, EIL B 76, EIL C76, EIL D76.

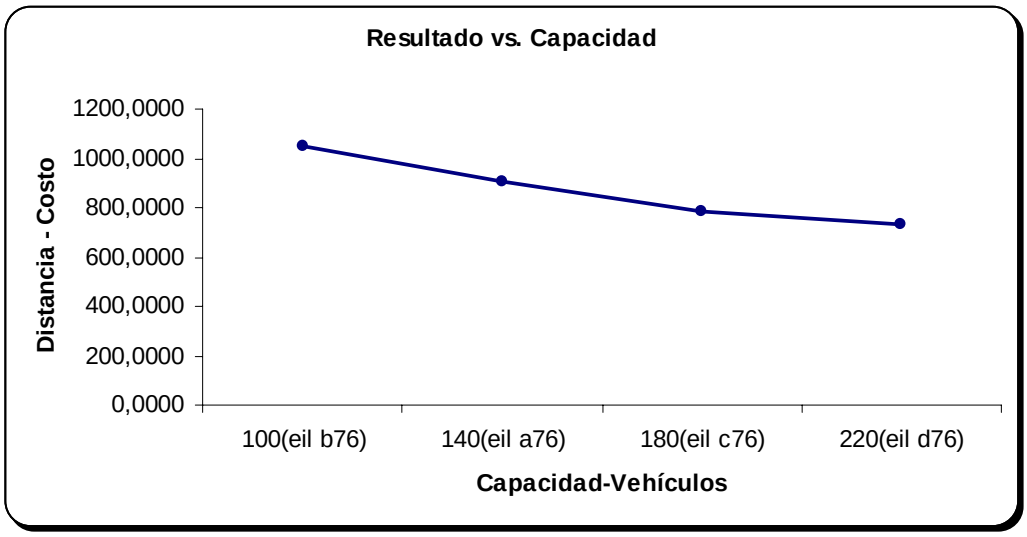
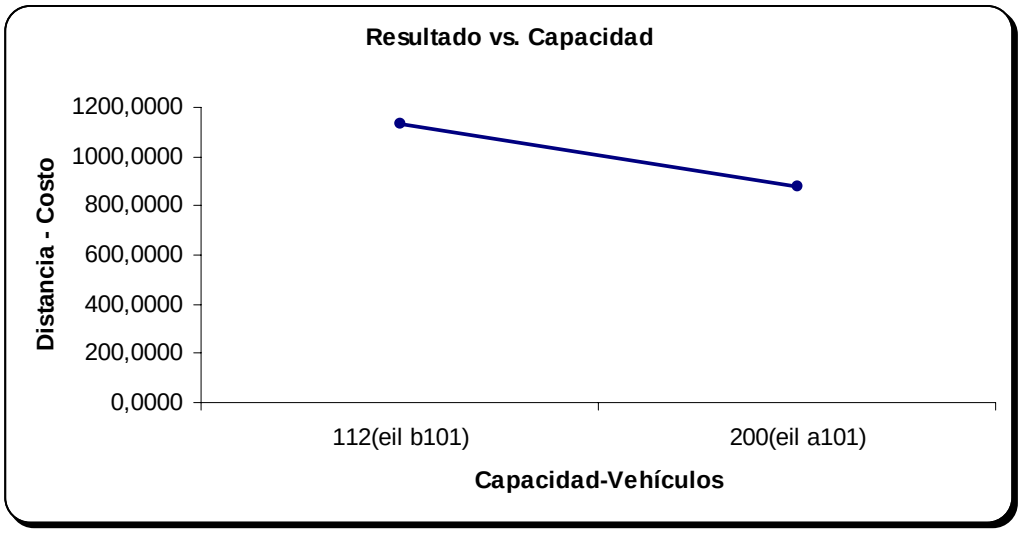


FIGURA 07. COMPARACIÓN DEL CAMBIO DE LA VARIABLE RESPUESTA CON RESPECTO A LA CAPACIDAD. PROBLEMAS EIL A101, EILB101.



5 DISEÑO DE EXPERIMENTOS PARA LA DETERMINACIÓN DE LOS FACTORES INFLUYENTES EN LA BÚSQUEDA TABÚ

En esta sección se identificarán los factores relevantes en el algoritmo desarrollado de acuerdo a la influencia en la variable respuesta.

5.1 IDENTIFICACIÓN DE LOS FACTORES PRINCIPALES.

Luego de realizar varias pruebas con los diferentes problemas propuestos en el presente proyecto, se llegó a la conclusión que los factores más relevantes en la variable respuesta son:

Tamaño de la Lista Tabú.
Tamaño de la Población (conjunto de concentración)
Número de Iteraciones.
Número de Intercambios.

Los tres primeros factores fueron propuestos en plan de proyecto como los únicos factores a estudiar en diseño de experimentos a desarrollar. Pero luego de varias pruebas del algoritmo se considera que el Número de Intercambios es una variable que afecta considerablemente la variable respuesta, razón por la cual este factor será incluido en un diseño de experimentos adicional (Ver sección 5.2.3).

5.2 DESARROLLO DEL DISEÑO DE EXPERIMENTOS

Se estudiarán dos problemas y se analizarán los factores relevantes por medio de la teoría de Diseño de Experimentos. Estos problemas a estudiar corresponden a EIL 51 y EIL 101B, los cuales representan los problemas con el número menor y número mayor (respectivamente) de clientes. Es preciso recordar que en todos los problemas tratados en el presente proyecto existe uno y sólo un depósito.

5.2.1 PROBLEMA EIL51

El diseño de experimentos escogido para analizar los datos arrojados por el algoritmo es el diseño 2^K con puntos centrales que presenta la siguiente combinación de tratamientos:

TABLA 08. COMBINACIONES DE FACTORES PARA EL DISEÑO FACTORIAL 2^3 CON PUNTOS CENTRALES.

	F1	F2	F3
T1	-1	-1	-1
T2	-1	-1	1
T3	-1	1	-1

T4	-1	1	1
T5	1	-1	-1
T6	1	-1	1
T7	1	1	-1
T8	1	1	1
T9	0	0	0
T9	0	0	0
T9	0	0	0
T9	0	0	0
T9	0	0	0

Los factores que se analizarán en el presente diseño de experimentos se muestran en la siguiente tabla:

TABLA 09. DESCRIPCIÓN DE LOS FACTORES A TRATAR EN EL DISEÑO FACTORIAL 2^3 CON PUNTOS CENTRALES.

FACTOR	NIVEL 1	NIVEL 2	Ptos. Centrales
Tamaño de la lista Tabú A	6	18	12
Población B	100	300	200
Número de iteraciones C	500	1000	750

El diseño factorial 2^3 con puntos centrales se describe por medio del siguiente modelo matemático:

$$y = \beta_0 + \sum_{j=1}^{k} \beta_1 x_j + \sum \sum_{i<j} \beta_{ij} x_i x_j + \sum \beta_{jj} x_j^2 + \varepsilon$$

Para aplicar un análisis de varianzas es preciso corroborar tres asunciones sobre los residuos del modelo:

Cada nivel del factor o tratamiento debe estar distribuido normalmente.

Cada nivel del factor o tratamiento tiene la misma varianza.

Las observaciones son muestras aleatorias independientes.

A continuación se muestra la descripción de cada tratamiento a tratar en el presente diseño e experimentos.

T1 Corresponde al primer tratamiento o combinación de factores, donde los valores para cada factor son (6, 1001 500) (-1,-1,-1) (a, b, c)

T2 Corresponde al segundo tratamiento o combinación de factores donde los valores para cada factor son (6, 100, 1000) (-1, -1, 1) (a, b, C).

T3 Corresponde al tercer tratamiento o combinación de factores donde los valores para cada factor son (6, 300, 500) (-1, 1, -1) (a, B, c).

T4 Corresponde al cuarto tratamiento o combinación de factores donde los valores para cada factor son (6, 300, 1000) (-1, 1, 1) (a, B, C).

T5 Corresponde al quinto tratamiento o combinación de factores donde los valores para cada factor son (18, 100, 500) (1, -1, -1) (A, b, c).

T6 Corresponde al sexto tratamiento o combinación de factores donde los valores para cada factor son (18, 100, 1000) (1, -1, 1) (A, b, C).

T7 Corresponde al séptimo tratamiento o combinación de factores donde los valores para cada factor son (18, 300, 500) (1, 1, -1) (A, B, c).

T8 Corresponde al octavo tratamiento o combinación de factores donde los valores para cada factor son (18, 300, 1000) (1, 1, 1) (A, B, C).
T9 Corresponde al noveno tratamiento o combinación de factores donde los valores para cada factor son (12, 200, 750) (0, 0, 0) Este tratamiento corresponde a las corridas en el punto central.

PRUEBA DE NORMALIDAD. DISEÑO 2^3 CON PUNTOS CENTRALES. PROBLEMA EIL 51.

Para determinar la normalidad de los datos se construye la gráfica de probabilidad normal de los residuos del modelo. Esta prueba se realiza en el programa Minitab 14. Las hipótesis nula y alterna son las siguientes:

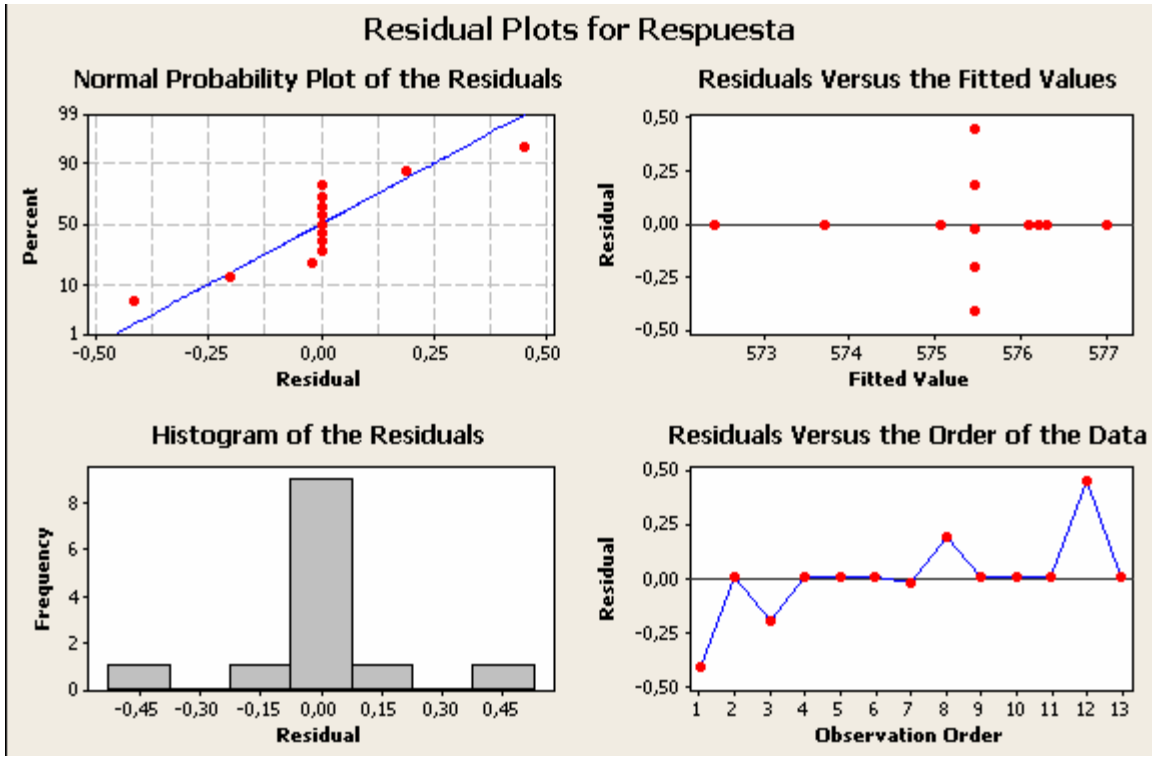
$$H_o = FDA\ y_i\ es\ N(\mu_x, \sigma_x^2)$$

$$H_1 = FDA\ y_i\ no\ es\ N(\mu_x, \sigma_x^2)$$

$$para\ i = 1,2...9$$

A continuación se presentan los resultados obtenidos.

FIGURA 08. GRÁFICOS PARA EL ANÁLISIS DE LOS RESIDUOS DEL MODELO. DISEÑO FACTORIAL 2^3 CON PUNTOS CENTRALES. PROBLEMA EIL 51.



De acuerdo a los gráficos anteriores, no es posible concluir la existencia de una normalidad contundente, dado que al trabajar con muestras pequeñas pueden aparecer fluctuaciones significativas, que no implican necesariamente la violación de los supuestos. Por esto se continúa con el proceso para determinar que tipo de resultados arroja la prueba ANOVA.

PRUEBA DE IGUALDAD DE VARIANZAS. PROBLEMA EIL 51.

Para demostrar el cumplimiento de esta asunción se realizó la prueba de Levene para los tratamientos. La hipótesis nula y alterna en este caso corresponde a:

$$H_o : \sigma_1^2 = \sigma_2^2 = \sigma_3^2 = ... = \sigma_9^2$$

$$H_1 : Al\ \ menos\ \ una\ \ es\ \ diferente$$

El primer paso en esta prueba consiste en la obtención de la mediana para cada tratamiento, la siguiente ecuación nos permite la construcción de la matriz característica.

$$d_{i,j} = \left| y_{i,j} - \tilde{y}_{i.} \right| \ \ donde$$

$$\tilde{y}_{i.}\ corresponde\ a\ la\ mediana\ del\ tratamiento$$

$$y_{i,j}\ corresponde\ a\ los\ datos\ del\ tratamiento$$

Esta matriz característica es sometida a una prueba Anova, y P-Value que se obtiene de esta nos permite identificar si las varianzas de los tratamientos son iguales o no. La matriz característica para este caso corresponde a

```
LEVENE =

    0.3925    0.3932    0.8654    0.6032    0.3925    0.3925    0.7660    1.0761    0.3925
    0.1818    0.1818    0.6546    0.2107    0.6553    0.1818    0.2931    0.2107    0.1818
         0         0         0         0    0.9767         0    0.1001         0         0
    0.2107    0.4729    0.5039    0.2621    0.1818    0.6553    0.0994    0.0824    0.2107
    0.4729    0.6553    1.0955    0.4446         0    0.6836         0    1.0359    0.4729
```

La prueba Anova que se realizó a la matriz característica arrojo los siguientes datos.

PValue	0.6489
F	0.7485

Dado el PValue alto, no es posible rechazar hipótesis nula que afirma que las varianzas entre los tratamientos son iguales, dado que no existe evidencia estadística para ello. Ver la salida completa del programa MatLab 6.5 en el Anexo 09.

PRUEBA DE ALEATORIEDAD. PROBLEMA EIL 51.

Para comprobar aleatoriedad de los datos, se realizó una prueba de corridas sobre los datos escogidos para el diseño 2^3 con puntos centrales, obteniéndose como resultado la siguiente tabla a través del programa "SPSS 7.5 for Windows Student Version".

TABLA 10. RESULTADOS DE LA PRUEBA DE ALEATORIEDAD DE DATOS.. DISEÑO FACTORIAL 2^3 CON PUNTOS CENTRALES.. PROBLEMA EIL 51.

NPar Tests

Runs Test

	VAR00001
Test Value[a]	575,6691
Cases < Test Value	6
Cases >= Test Value	7
Total Cases	13
Number of Runs	8
Z	,022
Asymp. Sig. (2-tailed)	,982

a. Median

En este caso el nivel crítico (Asymp. Sig. (2 - tailed) = 0.982) nos indica que no podemos rechazar la hipótesis de independencia entre las observaciones.

ANÁLISIS DE VARIANZAS PARA UN DISEÑO 2^3 CON PUNTOS CENTRALES

Luego de comprobar el cumplimiento de las tres asunciones, es posible aplicar la Prueba Anova a los datos recolectados. En la tabla siguiente se muestran los datos ordenados para la variable respuesta.

TABLA 11. DATOS SELECCIONADOS PARA EL DISEÑO FACTORIAL 2^3 CON PUNTOS CENTRALES.. PROBLEMA EIL 51.

DISEÑO FACTORIAL		PUNTOS CENTRALES	
TRATAMIENTO	VALOR	OBS	VALOR
(1)	575,065813	1	575,065813
c	577,026650	2	575,276547
b	577,026650	3	575,458320
bc	576,323693	4	575,669054
a	572,415245	5	575,931185
ac	576,113639		
ab	573,700040		
abc	576,224312		

A continuación se muestra la tabla de la Prueba Anova para este diseño factorial con puntos centrales.

FIGURA 09. ANALISIS DE VARIANZAS PARA EL DISEÑO 2^3 CON PUNTOS CENTRALES. RESULTADOS ARROJADOS POR MINITAB14. PROBLEMA EIL 51.

Analysis of Variance for Respuesta (coded units)

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	3	13,9816	13,9816	4,66054	41,18	0,002
2-Way Interactions	3	4,9247	4,9247	1,64157	14,50	0,013
3-Way Interactions	1	0,2774	0,2774	0,27739	2,45	0,193
Curvature	1	0,0001	0,0001	0,00014	0,00	0,973
Residual Error	4	0,4527	0,4527	0,11318		
Pure Error	4	0,4527	0,4527	0,11318		
Total	12	19,6366				

De los anteriores resultados se puede inferir que no existe evidencia estadística del efecto de una curvatura de segundo orden dado que el P-Value obtenido equivale a [0.973], un valor alto que indica que la curvatura no es relevante en el modelo.

FIGURA 10. EFECTOS ESTIMADOS Y COEFICIENTES DEL MODELO. DISEÑO FACTORIAL 2^3 CON PUNTOS CENTRALES. PROBLEMA EIL 51.

Estimated Effects and Coefficients for Respuesta (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		575,487	0,1189	4838,32	0,000
Lista	-1,747	-0,874	0,1189	-7,35	0,002
Población	0,663	0,332	0,1189	2,79	0,049
Iteraciones	1,870	0,935	0,1189	7,86	0,001
Lista*Población	0,034	0,017	0,1189	0,14	0,892
Lista*Iteraciones	1,241	0,621	0,1189	5,22	0,006
Población*Iteraciones	-0,959	-0,480	0,1189	-4,03	0,016
Lista*Población*Iteraciones	0,372	0,186	0,1189	1,57	0,193
Ct Pt		-0,007	0,1918	-0,04	0,973

S = 0,336424 R-Sq = 97,69% R-Sq(adj) = 93,08%

Analysis of Variance for Respuesta (coded units)

Los datos de la figura anterior permite la construcción del modelo siguiente, excluyendo los factores no relevantes:

$$y = 575.487 - 0.874x_1 + 0.332x_2 + 0.935x_3 + 0.621x_1x_3 - 0.480x_2x_3 + \varepsilon$$

Dado los antecedentes con la prueba de normalidad y teniendo en cuenta que el efecto de la curvatura no es significativo, no es posible asegurar que el modelo explica fielmente el comportamiento de los datos.

DISEÑO FACTORIAL 2^3. PROBLEMA EIL 51.

Dado que el diseño de experimentos anterior demuestra que el efecto de los términos cuadráticos no es significativo en el modelo se procede a la realización de un diseño factorial 2^3, cuyos datos se presentan a continuación.

TABLA 12. DESCRIPCIÓN DE LOS FACTORES PARA EL DISEÑO FACTORIAL 2^3. PROBLEMA EIL 51.

FACTOR	NIVEL 1	NIVEL 2
Tamaño de la lista Tabú A	6	18
Población B	100	300
Número de iteraciones C	500	1000

TABLA 13. COMBINACIONES DE FACTORES PARA EL DISEÑO FACTORIAL 2^3. PROBLEMA EIL 51.

	F1	F2	F3
T1	-1	-1	-1
T2	-1	-1	1
T3	-1	1	-1
T4	-1	1	1
T5	1	-1	-1
T6	1	-1	1
T7	1	1	-1
T8	1	1	1

TABLA 14. DATOS ESCOGIDOS PARA EL DISEÑO FACTORIAL 2^3. PROBLEMA EIL 51.

	OBSERVACIONES		
TRATAMIENTOS	OBS 1	OBS 2	OBS 3
T1	575,065813	575,45832	576,224312
T2	576,113639	575,276547	575,45832
T3	577,02665	577,394499	575,276547
T4	575,45832	576,141919	577,02665
T5	569,854891	569,200253	569,966226
T6	575,276547	576,141919	576,224312
T7	570,148019	573,317046	569,200253
T8	578,14813	576,224312	575,931185

El diseño factorial 2^3 se describe por medio del siguiente modelo matemático

$$y = \beta_0 + \sum_{j=1}^{k} \beta_1 x_j + \sum_{i<j} \sum \beta_{ij} x_i x_j + \varepsilon$$

A continuación se muestra la descripción de cada tratamiento a tratar en el presente diseño e experimentos.

T1 Corresponde al primer tratamiento o combinación de factores, donde los valores para cada factor son (6, 1001 500) (-1,-1,-1) (a, b, c)
T2 Corresponde al segundo tratamiento o combinación de factores donde los valores para cada factor son (6, 100, 1000) (-1, -1, 1) (a, b, C).
T3 Corresponde al tercer tratamiento o combinación de factores donde los valores para cada factor son (6, 300, 500) (-1, 1, -1) (a, B, c).
T4 Corresponde al cuarto tratamiento o combinación de factores donde los valores para cada factor son (6, 300, 1000) (-1, 1, 1) (a, B, C).
T5 Corresponde al quinto tratamiento o combinación de factores donde los valores para cada factor son (18, 100, 500) (1, -1, -1) (A, b, c).
T6 Corresponde al sexto tratamiento o combinación de factores donde los valores para cada factor son (18, 100, 1000) (1, -1, 1) (A, b, C).
T7 Corresponde al séptimo tratamiento o combinación de factores donde los valores para cada factor son (18, 300, 500) (1, 1, -1) (A, B, c).
T8 Corresponde al octavo tratamiento o combinación de factores donde los valores para cada factor son (18, 300, 1000) (1, 1, 1) (A, B, C).

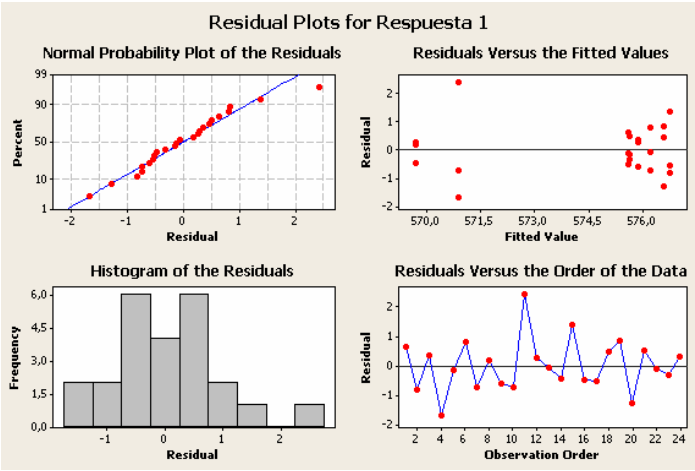
PRUEBA DE NORMALIDAD. DISEÑO 2^3. PROBLEMA EIL 51.

Para determinar la normalidad de los datos se construye la gráfica de probabilidad normal de los residuos del modelo. Esta prueba se realiza en el programa Minitab 14. Las hipótesis nula y alterna son las siguientes:

$$H_o = FDA\ y_i\ es\ N(\overline{\mu}_x, \sigma_x^2);$$
$$H_1 = FDA\ y_i\ no\ es\ N(\overline{\mu}_x, \sigma_x^2)$$
$$para\ i = 1,2...8$$

A continuación se presentan los resultados obtenidos.

FIGURA 11. GRÁFICOS PARA EL ANÁLISIS DE LOS RESIDUOS DEL MODELO. DISEÑO FACTORIAL 2^3. PROBLEMA EIL 51.



De acuerdo al gráfico anterior es posible identificar que los residuos siguen una distribución normal teniendo en cuenta las desviaciones pequeñas con respecto a la línea de ajuste. Además se ratifica la aleatoriedad de los mismos, debido al evidente patrón aleatorio que muestran en la figura de los residuos con respecto al orden de la observación.

PRUEBA DE IGUALDAD DE VARIANZAS. DISEÑO 2^3. PROBLEMA EIL 51.

La igualdad de varianzas entre los tratamientos es una de las asunciones que deben cumplirse para que sea posible la utilización del Análisis de Varianzas. En este caso se utilizó la Prueba de Igualdad de Varianzas de Levene para demostrar la igualdad de varianzas de los tratamientos del diseño factorial 2^3 descrito en la tabla 18.

Las hipótesis nula y alterna en este caso corresponden a:

$$H_o : \sigma_1^2 = \sigma_2^2 = \sigma_3^2 = ... = \sigma_8^2$$

$$H_1 : Al\ menos\ una\ es\ diferente$$

El primer paso en esta prueba consiste en la obtención de la mediana para cada tratamiento, la siguiente ecuación nos permite la construcción de la matriz característica.

$$d_{i,j} = \left| y_{i,j} - \tilde{y}_{i.} \right| \quad donde$$

$\tilde{y}_{i.}$ *corresponde a la mediana del tratamiento*

$y_{i,j}$ *corresponde a los datos del tratamiento*

Esta matriz característica es sometida a una prueba Anova, y P-Value que se obtiene de esta nos permite identificar si las varianzas de los tratamientos son iguales o no. La matriz característica en este caso es:

```
LEVENE =

    0.3925    0.6553         0    0.6836         0    0.8654         0    1.9238
         0    0.1818    0.3678         0    0.6546         0    3.1690         0
    0.7660         0    1.7501    0.8847    0.1113    0.0824    0.9478    0.2931
```

Los resultados obtenidos en esta prueba son:

PValue	0.7329
F	0.6191

De acuerdo a estos resultados, es posible inferir que no existe evidencia estadística para rechazar la hipótesis nula sobre la igualdad de las varianzas, teniendo en cuenta que el PValue es igual a [0.7329]. Para observar los resultados y salida completa del programa MatLab ir al Anexo 10.

PRUEBA DE ALEATORIEDAD. DISEÑO 2^3. PROBLEMA EIL 51.

Para comprobar aleatoriedad de los datos, se realizó una prueba de corridas sobre los datos escogidos para el diseño 2^3, obteniéndose como resultado la siguiente tabla a través del programa "SPSS 7.5 for Windows Student Version".

TABLA 15. RESULTADOS PARA LA PRUEBA DE ALEATORIEDAD DE DATOS. DISEÑO FACTORIAL 2^3. PROBLEMA EIL 51.

NPar Tests

Runs Test

	VAR00001
Test Value[a]	575,4583
Cases < Test Value	10
Cases >= Test Value	14
Total Cases	24
Number of Runs	10
Z	-,931
Asymp. Sig. (2-tailed)	,352

a. Median

En este caso el nivel crítico (Asymp. Sig. (2 - tailed) = 0.352) nos indica que no podemos rechazar la hipótesis de independencia entre las observaciones.

ANÁLISIS DE VARIANZAS PARA EL DISEÑO 2^3.PROBLEMA EIL 51.

Luego de comprobar el cumplimiento de las tres asunciones se realizó un análisis de varianzas en el programa MatLab 6.5, para el cual se utilizaron los datos de la tabla 18. Los resultados arrojados para este análisis se muestran en las siguientes secciones.

Análisis de Varianzas para todas las combinaciones de los factores.

Inicialmente se consideran en el Análisis de Varianzas a todas las posibles combinaciones de los tratamientos. Los resultados se muestran a continuación:

FIGURA 12. TABLA ANOVA PARA TODAS LAS COMBINACIONES DE LOS FACTORES. DISEÑO FACTORIAL 2^3. PROBLEMA EIL 51. PROGRAMA MINITAB 14.

Analysis of Variance for Respuesta 1 (coded units)

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	3	100,412	100,412	33,4707	29,76	0,000
2-Way Interactions	3	57,988	57,988	19,3295	17,19	0,000
3-Way Interactions	1	0,002	0,002	0,0018	0,00	0,969
Residual Error	16	17,995	17,995	1,1247		
Pure Error	16	17,995	17,995	1,1247		
Total	23	176,398				

FIGURA 13. EFECTOS ESTIMADOS Y COEFICIENTES DEL MODELO. DISEÑO FACTORIAL 2^3. PROBLEMA EIL 51. PROGRAMA MINITAB 14.

Estimated Effects and Coefficients for Respuesta 1 (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		574,647	0,2165	2654,54	0,000
Lista	-2,692	-1,346	0,2165	-6,22	0,000
Población	0,921	0,461	0,2165	2,13	0,049
Iteraciones	2,939	1,470	0,2165	6,79	0,000
Lista*Población	0,133	0,067	0,2165	0,31	0,762
Lista*Iteraciones	3,101	1,550	0,2165	7,16	0,000
Población*Iteraciones	-0,178	-0,089	0,2165	-0,41	0,687
Lista*Población*Iteraciones	0,017	0,009	0,2165	0,04	0,969

S = 1,06052 R-Sq = 89,80% R-Sq(adj) = 85,34%

De estos resultados es posible inferir que no existe efecto de las interacciones “Lista*Población”, “Población*Iteraciones” y “Lista*Población*Iteraciones”, dado que el valor del P-Value arrojado para cada una de ellas es considerablemente alto (0.762; 0.687; 0.969; respectivamente), lo que indica una falta de evidencia estadística para rechazar la hipótesis nula y por tanto sus efectos no tienen relevancia en la variable respuesta. De las figuras anteriores también es posible la construcción del modelo, excluyendo los efectos no relevantes.

$$y = 574.647 - 1.346x_1 + 0.461x_2 + 1.470x_3 + 1.550x_1x_3 + \varepsilon$$

Análisis de Varianzas para el efecto de los factores puros.

Luego de realizar el análisis de varianzas con todas las combinaciones, se practicó un análisis de varianzas para determinar la influencia de los factores puros en la variable respuesta. Los resultados se muestran en la figura siguiente.

FIGURA 14. EFECTOS ESTIMADOS Y COEFICIENTES DEL MODELO. PROGRAMA MINITAB 14, SOLO FACTORES PUROS. DISEÑO FACTORIAL 2^3. PROBLEMA EIL 51.

Estimated Effects and Coefficients for Respuesta 1 (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		574,647	0,3979	1444,30	0,000
Lista	-2,692	-1,346	0,3979	-3,38	0,003
Población	0,921	0,461	0,3979	1,16	0,261
Iteraciones	2,939	1,470	0,3979	3,69	0,001

S = 1,94917 R-Sq = 56,92% R-Sq(adj) = 50,46%

De acuerdo a la anterior figura, es posible inferir que el factor Población no es significativo para la variable respuesta dado que presenta un PValue de 0.261, lo cual indica que no tiene relevancia en el modelo. Excluyéndolo, el modelo que explica el comportamiento de los datos tiene la siguiente forma:

$$y = 574.647 - 1.346x_1 + 1.470x_3 + 1.550x_1x_3 + \varepsilon$$

La tabla ANOVA para los efectos puros se presenta a continuación.

FIGURA 15. TABLA ANOVA PARA LOS EFECTOS PUROS DE LOS FACTORES. PROGRAMA MINITAB 14. DISEÑO FACTORIAL 2^3. PROBLEMA EIL 51.

Analysis of Variance for Respuesta 1 (coded units)

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	3	100,41	100,41	33,471	8,81	0,001
Residual Error	20	75,99	75,99	3,799		
Lack of Fit	4	57,99	57,99	14,498	12,89	0,000
Pure Error	16	18,00	18,00	1,125		
Total	23	176,40				

PRUEBA DE COMPARACIÓN DE MEDIAS. PRUEBA DUNN-SIDAK

Como una prueba complementaria, se realizó la prueba de comparación de medias Dunn-Sidak a través del programa MatLab 6.5, tiene como datos de entrada los arrojados por la prueba ANOVA del anexo 11. Las pruebas de comparación de medias son utilizadas especialmente cuando el diseño de experimentos tiene interés en descubrir si existen diferencias significativas entre las medias de los diferentes tratamientos o la combinación de factores que conduce a un determinado resultado[19]. Luego de la prueba ANOVA, la cual nos arroja la información primaria en relación a la significancia de factores, el experimentador puede realizar otros procedimientos que arrojen información de la calidad de la solución con respecto a los tratamientos o diferentes niveles de los factores, esto se logra con las pruebas de comparación de medias. La hipótesis nula y alterna para esta prueba son:

[19] MASON, Robert L; GUNST, Richard F; HESS, James L. "Statistical Design and Analysis of Experiments with applications to Engineering and Science", Second Edition, Wiley Interscience. Pág. 196.

$$H_o : \mu_i = \mu_j$$

$$H_1 : \mu_i \neq \mu_j$$

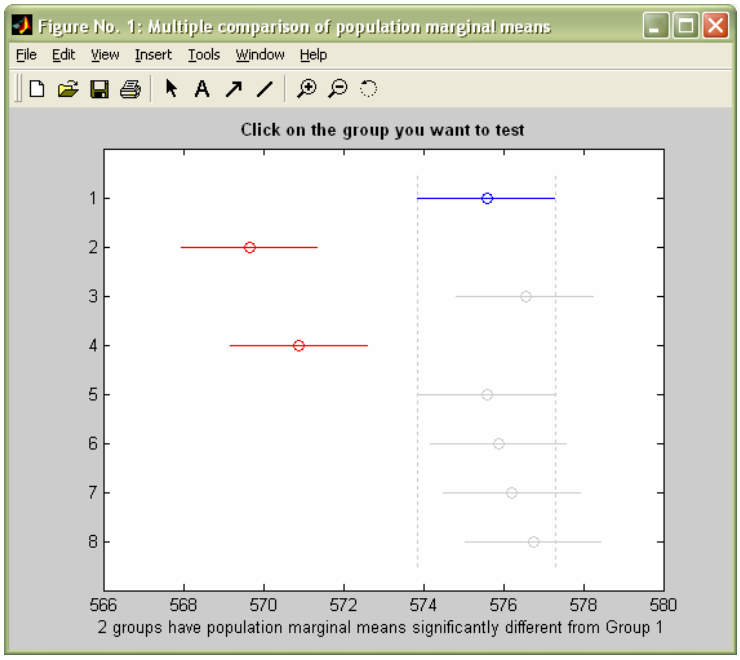
$$para\ \ toda\ \ i \neq j$$

Los resultados se muestran en las figuras siguientes.

FIGURA 16. RESULTADOS ARROJADOS POR EL PROGRAMA MATLAB 6.5 PARA LA PRUEBA DE COMPARACIÓN DE MEDIAS.. DISEÑO FACTORIAL 2^3. PROBLEMA EIL 51.

```
>> multcompare(STAT, 0.01,'on', 'dunn-sidak','on', [1 2 3 ])

ans =

    1.0000    2.0000    2.6965    5.9247    9.1529
    1.0000    3.0000   -4.1957   -0.9674    2.2608
    1.0000    4.0000    0.9667    4.6944    8.4220
    1.0000    5.0000   -3.2459   -0.0177    3.2105
    1.0000    6.0000   -4.0257   -0.2981    3.4295
    1.0000    7.0000   -4.3538   -0.6261    3.1015
    1.0000    8.0000   -4.3976   -1.1694    2.0588
    2.0000    3.0000  -10.6197   -6.8921   -3.1645
    2.0000    4.0000   -4.4585   -1.2303    1.9979
    2.0000    5.0000   -9.6700   -5.9424   -2.2148
    2.0000    6.0000   -9.4510   -6.2228   -2.9946
    2.0000    7.0000   -9.7790   -6.5508   -3.3226
    2.0000    8.0000  -10.8217   -7.0941   -3.3665
    3.0000    4.0000    2.4336    5.6618    8.8900
    3.0000    5.0000   -2.7779    0.9497    4.6774
    3.0000    6.0000   -2.5589    0.6693    3.8975
    3.0000    7.0000   -2.8869    0.3413    3.5695
    3.0000    8.0000   -3.9296   -0.2020    3.5257
    4.0000    5.0000   -7.9403   -4.7121   -1.4839
    4.0000    6.0000   -8.7201   -4.9925   -1.2649
    4.0000    7.0000   -9.0482   -5.3205   -1.5929
    4.0000    8.0000   -9.0920   -5.8638   -2.6356
    5.0000    6.0000   -3.5086   -0.2804    2.9478
    5.0000    7.0000   -3.8367   -0.6084    2.6198
    5.0000    8.0000   -4.8793   -1.1517    2.5759
    6.0000    7.0000   -4.0557   -0.3280    3.3996
    6.0000    8.0000   -4.0995   -0.8713    2.3569
    7.0000    8.0000   -3.7715   -0.5433    2.6850

>>
```

FIGURA 17. GRÁFICA DE LA PRUEBA DE COMPARACIÓN DE MEDIAS DUNN-SIDAK. DISEÑO FACTORIAL 2^3. PROBLEMA EIL 51



De acuerdo a los resultados se puede inferir que existen dos grupos que presentan diferencias significativas con respecto a los demás. Además el Tratamiento 2 presenta los mejores resultados (Tamaño de la Lista= 6; Tamaño de la Población= 100; Número de Iteraciones =1000). A continuación se presentan los intervalos de confianza para cada una de las medias muestrales.

$$P\{573.9 \le \bar{x}_1 \le 577.3\} = 99\% \;\; T1$$
$$P\{567.9 \le \bar{x}_2 \le 571.4\} = 99\% \;\; T2$$
$$P\{574.8 \le \bar{x}_3 \le 578.3\} = 99\% \;\; T3$$
$$P\{569.2 \le \bar{x}_4 \le 572.6\} = 99\% \;\; T4$$
$$P\{573.9 \le \bar{x}_5 \le 577.3\} = 99\% \;\; T5$$
$$P\{574.2 \le \bar{x}_6 \le 577.6\} = 99\% \;\; T6$$
$$P\{574.5 \le \bar{x}_7 \le 577.9\} = 99\% \;\; T7$$
$$P\{575 \le \bar{x}_8 \le 578.5\} = 99\% \;\; T8$$

5.2.2 PROBLEMA EIL101B

El diseño de experimentos escogido para analizar los datos arrojados por el algoritmo es el diseño 2^K con puntos centrales que presenta la siguiente combinación de tratamientos:

TABLA 16. COMBINACIONES DE FACTORES PARA EL DISEÑO FACTORIAL 2^3 CON PUNTOS CENTRALES. PROBLEMA EIL B101.

	F1	F2	F3
T1	-1	-1	-1
T2	-1	-1	1
T3	-1	1	-1
T4	-1	1	1
T5	1	-1	-1
T6	1	-1	1
T7	1	1	-1
T8	1	1	1
T9	0	0	0
T9	0	0	0
T9	0	0	0
T9	0	0	0
T9	0	0	0

Los factores que se analizarán en el presente diseño de experimentos se muestran en la siguiente tabla:

TABLA 17. DESCRIPCIÓN DE LOS FACTORES PARA EL DISEÑO FACTORIAL 2^3 CON PUNTOS CENTRALES. PROBLEMA EIL B101.

FACTOR	NIVEL 1	NIVEL 2	Ptos. Centrales
Tamaño de la lista Tabú A	6	18	12
Población B	100	300	200
Número de iteraciones C	500	1000	750

El diseño factorial 2^3 con puntos centrales se describe por medio del siguiente modelo matemático:

$$y = \beta_0 + \sum_{j=1}^{k} \beta_1 x_j + \sum \sum_{i<j} \beta_{ij} x_i x_j + \sum \beta_{jj} x_j^2 + \varepsilon$$

Para aplicar un análisis de varianzas es preciso corroborar tres asunciones:

Cada nivel del factor o tratamiento debe estar distribuido normalmente.

Cada nivel del factor o tratamiento tiene la misma varianza.

Las observaciones son muestras aleatorias independientes.

A continuación se muestra la descripción de los tratamientos a utilizar en el diseño factorial con puntos centrales:

T1 Corresponde al primer tratamiento o combinación de factores, donde los valores para cada factor son (6, 1001 500) (-1,-1,-1) (a, b, c)

T2 Corresponde al segundo tratamiento o combinación de factores donde los valores para cada factor son (6, 100, 1000) (-1, -1, 1) (a, b, C).

T3 Corresponde al tercer tratamiento o combinación de factores donde los valores para cada factor son (6, 300, 500) (-1, 1, -1) (a, B, c).

T4 Corresponde al cuarto tratamiento o combinación de factores donde los valores para cada factor son (6, 300, 1000) (-1, 1, 1) (a, B, C).
T5 Corresponde al quinto tratamiento o combinación de factores donde los valores para cada factor son (18, 100, 500) (1, -1, -1) (A, b, c).
T6 Corresponde al sexto tratamiento o combinación de factores donde los valores para cada factor son (18, 100, 1000) (1, -1, 1) (A, b, C).
T7 Corresponde al séptimo tratamiento o combinación de factores donde los valores para cada factor son (18, 300, 500) (1, 1, -1) (A, B, c).
T8 Corresponde al octavo tratamiento o combinación de factores donde los valores para cada factor son (18, 300, 1000) (1, 1, 1) (A, B, C).
T9 Corresponde al noveno tratamiento o combinación de factores donde los valores para cada factor son (12, 200, 750) (0, 0, 0) Este tratamiento corresponde a las corridas en el punto central

PRUEBA DE NORMALIDAD. DISEÑO 2^3 CON PUNTOS CENTRALES.

La construye una gráfica de probabilidad normal para los residuos del modelo, la cual tiene por objeto demostrar la normalidad de los datos. La hipótesis nula y alterna son las siguientes:

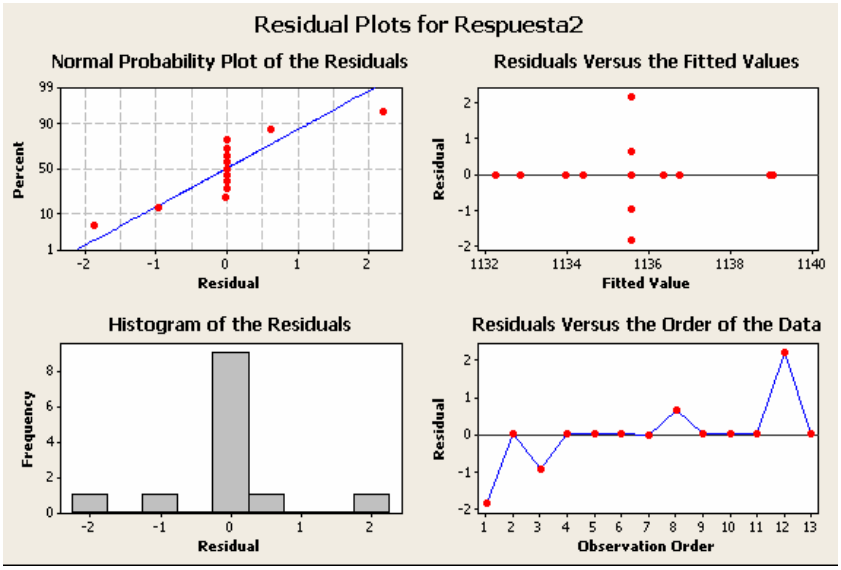
$$H_o = FDA\ y_i\ es\ N\left(\mu_x, \sigma_x^2\right)$$

$$H_1 = FDA\ y_i\ no\ es\ N\left(\mu_x, \sigma_x^2\right)$$

$$para\ i = 1,2...9$$

Las gráficas para el análisis del comportamiento de los residuos se obtuvieron por medio del programa Minitab 14. A continuación se presentan los resultados:

FIGURA 18. GRÁFICAS PARA EL ANÁLISIS DEL COMPORTAMIENTO DE LOS RESIDUOS DEL MODELO. DISEÑO FACTORIAL 2^3 CON PUNTOS CENTRALES. PROBLEMA EIL B101. PROGRAMA MINITAB 14.



De acuerdo a los gráficos anteriores, no es posible concluir la existencia de una normalidad contundente, dado que al trabajar con muestras pequeñas pueden aparecer fluctuaciones significativas, que no implican necesariamente la violación de los supuestos. Por esto se continúa con el proceso para determinar que tipo de resultados arroja la prueba ANOVA.

PRUEBA DE IGUALDAD DE VARIANZAS. DISEÑO 2^3 CON PUNTOS CENTRALES.

Para demostrar el cumplimiento de esta asunción se realizó la prueba de Levene para los tratamientos. La hipótesis nula y alterna en este caso corresponde a:

$$H_o : \sigma_1^2 = \sigma_2^2 = \sigma_3^2 = ... = \sigma_9^2$$

$$H_1 : Al\ menos\ una\ es\ diferente$$

El primer paso en esta prueba consiste en la obtención de la mediana para cada tratamiento, la siguiente ecuación nos permite la construcción de la matriz característica.

$$d_{i,j} = \left| y_{i,j} - \tilde{y}_{i.} \right| \quad donde$$

$$\tilde{y}_{i.}\ corresponde\ a\ la\ mediana\ del\ tratamiento$$

$$y_{i,j}\ corresponde\ a\ los\ datos\ del\ tratamiento$$

Esta matriz característica es sometida a una prueba Anova, y P-Value que se obtiene de esta nos permite identificar si las varianzas de los tratamientos son iguales o no. La matriz característica en este caso es:

```
LEVENE =

    0.2620    0.1496    0.8736    0.2918    0.2385    0.2385    0.2385    0.0659    0.2385
    0.0240    0.0963    0.5648    0.0533    0.1322    0.1322    0.1322    0.0121    0.1322
         0         0         0         0         0         0         0         0         0
    0.0300    0.0180    0.0254    0.0963    0.0231    0.0368    0.0368    0.1887    0.0368
    0.1260    0.1593    0.0659    0.1143    0.0368    0.1063    0.1063    0.4530    0.1677
```

Estos valores obtenidos en la prueba son:

PValue	0.5629
F	0.8540

Como se puede observar el valor de 0.5629 para el P-Value indica que no existe suficiente evidencia estadística para rechazar la hipótesis nula, por tanto se concluye que los datos tienen igual varianza. La salida completa de esta prueba se muestra en el Anexo 12.

PRUEBA DE ALEATORIEDAD. DISEÑO 2^3 CON PUNTOS CENTRALES.

Para comprobar aleatoriedad de los datos, se realizó una prueba de corridas sobre los datos escogidos para el diseño 2^3 con puntos centrales, obteniéndose como resultado la siguiente tabla a través del programa "SPSS 7.5 for Windows Student Version".

TABLA 18. RESULTADOS DE LA PRUEBA DE ALEATORIEDAD DE DATOS.. DISEÑO FACTORIAL 2^3 CON PUNTOS CENTRALES. PROBLEMA EIL B101.

NPar Tests

Runs Test

	VAR00001
Test Value[a]	1135,5516
Cases < Test Value	6
Cases >= Test Value	7
Total Cases	13
Number of Runs	6
Z	-,561
Asymp. Sig. (2-tailed)	,575

a. Median

En este caso el nivel crítico (Asymp. Sig. (2 - tailed) = 0.575) nos indica que no podemos rechazar la hipótesis de independencia entre las observaciones.

ANÁLISIS DE VARIANZAS. DISEÑO 2^3 CON PUNTOS CENTRALES.

Luego de comprobar el cumplimiento de las tres asunciones, es posible aplicar la Prueba Anova a los datos recolectados. En la tabla siguiente se muestran los datos ordenados.

TABLA 19. DATOS SELECCIONADOS PARA EL DISEÑO FACTORIAL 2^3 CON PUNTOS CENTRALES. PROBLEMA EIL B101.

DISEÑO FACTORIAL		PUNTOS CENTRALES	
TRATAMIENTO	VALOR	OBS	VALOR
(1)	1138.945526	1	1133.706157
c	1136.718243	2	1134.608909
b	1132.232131	3	1135.551581
bc	1139.042917	4	1136.199414
a	1132.836656	5	1137.762945
ac	1133.943406		

ab	1134.353015		
abc	1136.334154		

FIGURA 19. TABLA ANOVA DE TODAS LAS COMBINACIONES DE FACTORES. DISEÑO FACTORIAL 2^3 CON PUNTOS CENTRALES. PROBLEMA EIL B101. PROGRAMA MINITAB 14.

Analysis of Variance for Respuesta2 (coded units)

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	3	18,5991	18,5991	6,19971	2,58	0,191
2-Way Interactions	3	21,1643	21,1643	7,05477	2,94	0,162
3-Way Interactions	1	8,3307	8,3307	8,33071	3,47	0,136
Curvature	1	0,0007	0,0007	0,00070	0,00	0,987
Residual Error	4	9,6030	9,6030	2,40076		
Pure Error	4	9,6030	9,6030	2,40076		
Total	12	57,6979				

De acuerdo a la figura anterior, podemos inferir que no existe efecto de la curvatura de segundo orden dado que el PValue que presenta corresponde a [0.987].

FIGURA 20. EFECTOS ESTIMADOS Y COEFICIENTES DEL MODELO. DISEÑO FACTORIAL 2^3 CON PUNTOS CENTRALES. PROBLEMA EIL B101. PORGRAMA MINITAB 14.

Estimated Effects and Coefficients for Respuesta2 (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		1135,55	0,5478	2072,90	0,000
Lista	-2,37	-1,18	0,5478	-2,16	0,097
Población	-0,12	-0,06	0,5478	-0,11	0,918
Iteraciones	1,92	0,96	0,5478	1,75	0,155
Lista*Población	2,07	1,04	0,5478	1,89	0,131
Lista*Iteraciones	-0,37	-0,19	0,5478	-0,34	0,750
Población*Iteraciones	2,48	1,24	0,5478	2,26	0,087
Lista*Población*Iteraciones	-2,04	-1,02	0,5478	-1,86	0,136
Ct Pt		0,02	0,8833	0,02	0,987

S = 1,54944 R-Sq = 83,36% R-Sq(adj) = 50,07%

La figura anterior arroja los coeficientes para la construcción del siguiente modelo, excluyendo la curvatura.

$$y = 1135.55 - 1.18x_1 - 0.06x_2 + 0.96x_3 + 1.04x_1x_2 - 0.19x_1x_3 + 1.24x_2x_3 - 1.02x_1x_2x_3 + \varepsilon$$

Dado los antecedentes con la prueba de normalidad y teniendo en cuenta que el efecto de la curvatura no es significativo, no es posible asegurar que el modelo explica fielmente el comportamiento de los datos.

DISEÑO FACTORIAL 2^3

Dado que el diseño de experimentos anterior demuestra que el efecto de los términos cuadráticos no es significante en el modelo se procede a la realización de un diseño factorial 2^3, cuyos datos se presentan a continuación.

TABLA 20. DESCRIPCIÓN DE FACTORES PARA UN DISEÑO FACTORIAL 2^3. PROBLEMA EIL B101

FACTOR	NIVEL 1	NIVEL 2
Tamaño de la lista Tabú A	6	18
Población B	100	300
Número de iteraciones C	500	1000

TABLA 21. COMBINACIÓN DE FACTORES PARA EL DISEÑO FACTORIAL 2^3. PROBLEMA EIL B101.

	F1	F2	F3
T1	-1	-1	-1
T2	-1	-1	1
T3	-1	1	-1
T4	-1	1	1
T5	1	-1	-1
T6	1	-1	1
T7	1	1	-1
T8	1	1	1

TABLA 22. DATOS ESCOGIDOS PARA EL DISEÑO FACTORIAL 2^3. PROBLEMA EIL B101.

	OBSERVACIONES		
TRATAMIENTOS	OBS 1	OBS 2	OBS 3
T1	1139.017506	1138.293460	1138.143910
T2	1136.227805	1133.430888	1135.061904
T3	1131.068551	1130.478380	1131.193100
T4	1137.905407	1139.042917	1138.452746
T5	1132.695486	1132.395667	1132.581098
T6	1133.537237	1133.943406	1133.430888
T7	1133.699328	1134.739446	1134.080945
T8	1135.168253	1136.346273	1135.061904

El diseño factorial 2^3 se describe por medio del siguiente modelo matemático

$$y = \beta_0 + \sum_{j=1}^{k} \beta_1 x_j + \sum \sum_{i<j} \beta_{ij} x_i x_j + \varepsilon$$

A continuación se muestra la descripción de los tratamientos a utilizar en el diseño factorial con puntos centrales:

T1 Corresponde al primer tratamiento o combinación de factores, donde los valores para cada factor son (6, 1001 500) (-1,-1,-1) (a, b, c)
T2 Corresponde al segundo tratamiento o combinación de factores donde los valores para cada factor son (6, 100, 1000) (-1, -1, 1) (a, b, C).
T3 Corresponde al tercer tratamiento o combinación de factores donde los valores para cada factor son (6, 300, 500) (-1, 1, -1) (a, B, c).
T4 Corresponde al cuarto tratamiento o combinación de factores donde los valores para cada factor son (6, 300, 1000) (-1, 1, 1) (a, B, C).
T5 Corresponde al quinto tratamiento o combinación de factores donde los valores para cada factor son (18, 100, 500) (1, -1, -1) (A, b, c).
T6 Corresponde al sexto tratamiento o combinación de factores donde los valores para cada factor son (18, 100, 1000) (1, -1, 1) (A, b, C).
T7 Corresponde al séptimo tratamiento o combinación de factores donde los valores para cada factor son (18, 300, 500) (1, 1, -1) (A, B, c).
T8 Corresponde al octavo tratamiento o combinación de factores donde los valores para cada factor son (18, 300, 1000) (1, 1, 1) (A, B, C).

PRUEBA DE NORMALIDAD. DISEÑO 2^3

Para determinar la normalidad de los datos se construye la gráfica de probabilidad normal de los residuos del modelo. Esta prueba se realiza en el programa Minitab 14. Las hipótesis nula y alterna son las siguientes:

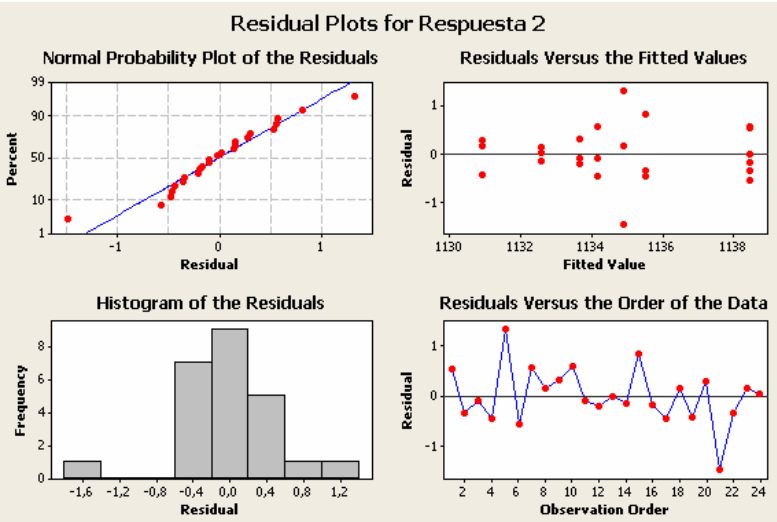
$$H_o = FDA\ y_i\ es\ N(\overline{\mu}_x, \sigma_x^2);$$

$$H_1 = FDA\ y_i\ no\ es\ N(\overline{\mu}_x, \sigma_x^2)$$

$$para\ i = 1,2...8$$

A continuación se presentan los resultados obtenidos.

FIGURA 21 GRÁFICOS PARA EL ANÁLISIS DE LOS RESIDUOS DEL MODELO. DISEÑO FACTORIAL 2^3. PROBLEMA EIL B101. PROGRAMA MINITAB 14.



De acuerdo al gráfico anterior es posible identificar que los residuos siguen una distribución normal teniendo en cuenta las desviaciones pequeñas con respecto a la línea de ajuste. Además se ratifica la aleatoriedad de los mismos, debido al evidente patrón aleatorio que muestran en la figura de los residuos con respecto al orden de la observación.

PRUEBA DE IGUALDAD DE VARIANZAS. DISEÑO 2^3.

Para demostrar el cumplimiento de esta asunción se realizó la prueba de Levene para los tratamientos. La hipótesis nula y alterna en este caso corresponde a:

$$H_o : \sigma_1^2 = \sigma_2^2 = \sigma_3^2 = ... = \sigma_8^2$$

$$H_1 : Al\ \ menos\ \ una\ \ es\ \ diferente$$

El primer paso en esta prueba consiste en la obtención de la mediana para cada tratamiento, la siguiente ecuación nos permite la construcción de la matriz característica.

$$d_{i,j} = \left| y_{i,j} - \tilde{y}_{i.} \right| \ \ donde$$

$$\tilde{y}_{i.}\ corresponde\ a\ la\ mediana\ del\ tratamiento$$

$$y_{i,j}\ corresponde\ a\ los\ datos\ del\ tratamiento$$

Esta matriz característica es sometida a una prueba Anova, y P-Value que se obtiene de esta nos permite identificar si las varianzas de los tratamientos son iguales o no. Estos valores se muestran a continuación:

PValue	0.4972
F	0.9502

La matriz característica utilizada en este caso es:

```
LEVENE =

    0.7240    1.1659         0    0.5473    0.1144         0    0.3816         0
         0    1.6310    0.5902    0.5902    0.1854    0.4062    0.6585    1.1780
    0.1496         0    0.1245         0         0    0.1063         0    0.1063
```

Como se puede observar el valor de 0.4972 para el P-Value indica que no existe suficiente evidencia estadística para rechazar la hipótesis nula, por tanto se concluye que los datos tienen igual varianza. Para ver la tabla completa ir al Anexo 13.

PRUEBA DE ALEATORIEDAD. DISEÑO 2^3.

Para comprobar aleatoriedad de los datos, se realizó una prueba de corridas sobre los datos escogidos para el diseño 2^3 con puntos centrales, obteniéndose como resultado la siguiente tabla a través del programa "SPSS 7.5 for Windows Student Version".

TABLA 23. RESULTADOS DE LA PRUEBA DE ALEATORIEDAD DE DATOS.. DISEÑO FACTORIAL 2^3. PROBLEMA EIL B101.

NPar Tests

Runs Test

	VAR00001
Test Value[a]	1134,4102
Cases < Test Value	12
Cases >= Test Value	12
Total Cases	24
Number of Runs	9
Z	-1,461
Asymp. Sig. (2-tailed)	,144

a. Median

En este caso el nivel crítico (Asymp. Sig. (2 - tailed) = 0.144) nos indica que no podemos rechazar la hipótesis de independencia entre las observaciones.

ANÁLISIS DE VARIANZAS. DISEÑO 2^3.

En esta sección se desarrolla el análisis de varianzas para el diseño factorial 2^3 del problema EIL B101.

Análisis de Varianzas para todas las combinaciones de los tratamientos.

Inicialmente se aplicó una prueba Anova para evaluar los efectos de todas las combinaciones posibles de factores. A continuación se muestran los resultados arrojados por el programa Minitab 14.

FIGURA 22. RESULTADOS ARROJADOS POR EL PROGRAMA MINITAB 14 PARA EL ANÁLISIS DE VARIANZAS DE TODAS LAS COMBINACIONES DE FACTORES. DISEÑO FACTORIAL 2^3. PROBLEMA EIL B101.

Analysis of Variance for Respuesta 2 (coded units)

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	3	33,238	33,238	11,0792	25,02	0,000
2-Way Interactions	3	70,847	70,847	23,6156	53,33	0,000
3-Way Interactions	1	44,222	44,222	44,2218	99,87	0,000
Residual Error	16	7,085	7,085	0,4428		
Pure Error	16	7,085	7,085	0,4428		
Total	23	155,391				

FIGURA 23. RESULTADOS ARROJADOS POR EL PROGRAMA MINITAB 14 EFECTOS ESTIMADOS Y LOS COEFICIENTES DEL MODELO.

Estimated Effects and Coefficients for Respuesta 2 (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		1134,83	0,1358	8354,69	0,000
Lista	-1,72	-0,86	0,1358	-6,33	0,000
Población	-0,13	-0,06	0,1358	-0,47	0,647
Iteraciones	1,60	0,80	0,1358	5,90	0,000
Lista*Población	1,88	0,94	0,1358	6,92	0,000
Lista*Iteraciones	-0,39	-0,19	0,1358	-1,42	0,175
Población*Iteraciones	2,85	1,43	0,1358	10,49	0,000
Lista*Población*Iteraciones	-2,71	-1,36	0,1358	-9,99	0,000

S = 0,665438 R-Sq = 95,44% R-Sq(adj) = 93,45%

De acuerdo con los resultados mostrados anteriormente, es posible inferir que no hay evidencia estadística del efecto del Factor puro "Tamaño de la Población" y tampoco del efecto de la interacción "Tamaño de Lista*Número de Iteraciones", dado que sus P-Values corresponden a 0.647 y 0.175 respectivamente, valores considerablemente altos que impiden rechazar la hipótesis nula, y por tanto se descarta su relevancia en la variable respuesta. El modelo que se obtiene excluyendo los elementos no significativos es:

$$y = 1134.83 - 0.86x_1 + 0.80x_3 + 0.94x_1x_2 + 1.43x_2x_3 - 1.36x_1x_2x_3 + \varepsilon$$

Análisis de Varianzas para el efecto de las interacciones de dos Factores.

Con el propósito de conocer el efecto de las interacciones de dos factores, se aplicó la prueba Anova obteniéndose los resultados que se muestran a continuación.

FIGURA 24. RESULTADOS ARROJADOS POR EL PROGRAMA MINITAB PARA LOS EFECTOS ESTIMADOS Y LOS COEFICIENTES DEL MODELO., SOLO LAS INTERACCIONES DE DOS FACTORES. DISEÑO FACTORIAL 2^3. PROBLEMA EIL B101.

Estimated Effects and Coefficients for Respuesta 2 (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		1134,83	0,3546	3200,18	0,000
Lista	-1,72	-0,86	0,3546	-2,42	0,027
Población	-0,13	-0,06	0,3546	-0,18	0,860
Iteraciones	1,60	0,80	0,3546	2,26	0,037
Lista*Población	1,88	0,94	0,3546	2,65	0,017
Lista*Iteraciones	-0,39	-0,19	0,3546	-0,54	0,593
Población*Iteraciones	2,85	1,43	0,3546	4,02	0,001

S = 1,73725 R-Sq = 66,98% R-Sq(adj) = 55,33%

FIGURA 25. RESULTADOS ARROJADOS POR EL PROGRAMA MINITAB PARA LA PRUEBA ANOVA CON LOS EFECTOS DE LAS INTERACCIONES DE DOS FACTORES.. DISEÑO FACTORIAL 2^3. PROBLEMA EIL B101.

Analysis of Variance for Respuesta 2 (coded units)

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	3	33,238	33,238	11,0792	3,67	0,033
2-Way Interactions	3	70,847	70,847	23,6156	7,82	0,002
Residual Error	17	51,307	51,307	3,0180		
Lack of Fit	1	44,222	44,222	44,2218	99,87	0,000
Pure Error	16	7,085	7,085	0,4428		
Total	23	155,391				

Con los resultados anteriores se corrobora que la interacción “Tamaño de Lista*Número de Iteraciones” y el factor puro “Tamaño de la Población” no tienen efecto significativo en la variable respuesta, dado que sus P-Values corresponden a 0.860 y 0.593, valores altos que demuestran que no tienen relevancia en el modelo.

PRUEBA DE COMPARACIÓN DE MEDIAS DUNN-SIDAK

Como una prueba complementaria, se realizó la prueba de comparación de medias Dunn-Sidak a través del programa MatLab 6.5. Esta prueba utiliza como datos de entrada los resultados arrojados por la prueba ANOVA del anexo 14. Las pruebas de comparación de medias son utilizadas especialmente cuando el diseño de experimentos tiene interés en descubrir si existen diferencias significativas entre las medias de los diferentes tratamientos o la combinación de factores que conduce a un determinado resultado. Luego de la prueba ANOVA, la cual nos arroja la información primaria en relación a la significancia de factores, el experimentador puede realizar otros procedimientos que arrojen información de la calidad de la solución con respecto a los tratamientos o diferentes niveles de los factores, esto se logra con las pruebas de comparación de medias. La hipótesis nula y alterna para esta prueba son:

$$H_o : \mu_i = \mu_j$$

$$H_1 : \mu_i \neq \mu_j$$

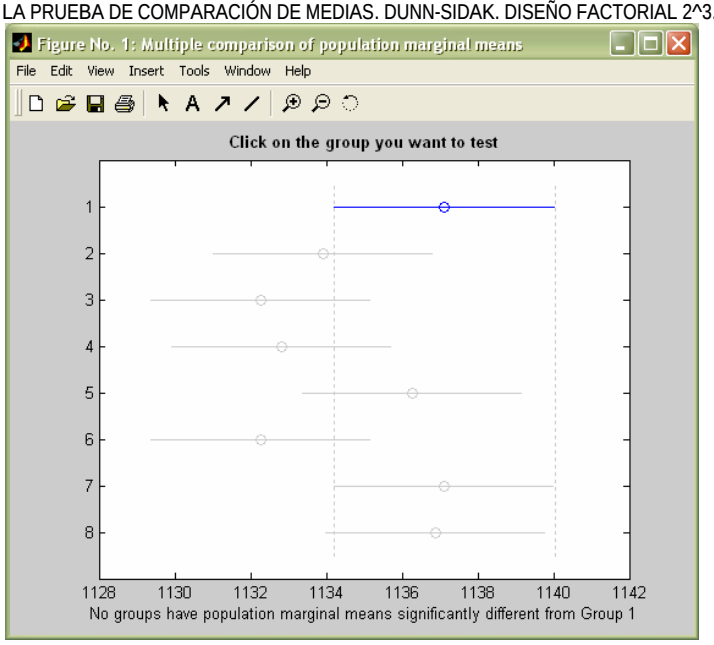
$$para\ toda\ i \neq j$$

Los resultados se muestran en las figuras siguientes.

FIGURA 26. RESULTADOS ARROJADOS POR EL PROGRAMA MATLAB 6.5 PARA LA PRUEBA DE COMPARACIÓN DE MEDIAS. DUNN-SIDAK. DISEÑO FACTORIAL 2^3. PROBLEMA EIL B101.

```
>> multcompare(STAT, 0.01,'on','dunn-sidak','on', [1 2 3])

ans =

    1.0000    2.0000   -2.2417    3.2127    8.6671
    1.0000    3.0000   -0.5976    4.8568   10.3112
    1.0000    4.0000   -1.9865    4.3117   10.6099
    1.0000    5.0000   -4.5912    0.8633    6.3177
    1.0000    6.0000   -1.4504    4.8478   11.1460
    1.0000    7.0000   -6.2803    0.0179    6.3162
    1.0000    8.0000   -5.2098    0.2447    5.6991
    2.0000    3.0000   -4.6542    1.6441    7.9423
    2.0000    4.0000   -4.3554    1.0990    6.5534
    2.0000    5.0000   -8.6477   -2.3494    3.9488
    2.0000    6.0000   -3.8194    1.6351    7.0895
    2.0000    7.0000   -8.6492   -3.1948    2.2596
    2.0000    8.0000   -9.2663   -2.9681    3.3302
    3.0000    4.0000   -5.9995   -0.5451    4.9094
    3.0000    5.0000  -10.2917   -3.9935    2.3047
    3.0000    6.0000   -5.4634   -0.0090    5.4454
    3.0000    7.0000  -10.2933   -4.8389    0.6156
    3.0000    8.0000  -10.9104   -4.6121    1.6861
    4.0000    5.0000   -8.9029   -3.4484    2.0060
    4.0000    6.0000   -5.7622    0.5361    6.8343
    4.0000    7.0000  -10.5920   -4.2938    2.0044
    4.0000    8.0000   -9.5215   -4.0671    1.3874
    5.0000    6.0000   -1.4699    3.9845    9.4389
    5.0000    7.0000   -6.2998   -0.8453    4.6091
    5.0000    8.0000   -6.9168   -0.6186    5.6796
    6.0000    7.0000  -11.1281   -4.8298    1.4684
    6.0000    8.0000  -10.0575   -4.6031    0.8513
    7.0000    8.0000   -5.2277    0.2267    5.6811

>>
```

FIGURA 27. GRÁFICA PARA LA PRUEBA DE COMPARACIÓN DE MEDIAS. DUNN-SIDAK. DISEÑO FACTORIAL 2^3. PROBLEMA EIL B101.



De acuerdo a los resultados arrojados por el procedimiento, no existen grupos con medias significativamente diferentes. Con base en los resultados es posible inferir que se obtienen mejores resultados con el tratamiento 6 (Tamaño de la Lista=100; Tamaño de la Población=300; Número de Iteraciones= 1000). Los intervalos de confianza para cada una de las medias muestrales se presentan a continuación.

$$P\{1134 \le \bar{x}_1 \le 1140\} = 99\% \;\; T1$$
$$P\{1131 \le \bar{x}_2 \le 1137\} = 99\% \;\; T2$$
$$P\{1129 \le \bar{x}_3 \le 1135\} = 99\% \;\; T3$$
$$P\{1130 \le \bar{x}_4 \le 1136\} = 99\% \;\; T4$$
$$P\{1133 \le \bar{x}_5 \le 1139\} = 99\% \;\; T5$$
$$P\{1129 \le \bar{x}_6 \le 1135\} = 99\% \;\; T6$$
$$P\{1134 \le \bar{x}_7 \le 1140\} = 99\% \;\; T7$$
$$P\{1134 \le \bar{x}_8 \le 1140\} = 99\% \;\; T8$$

5.2.3 Diseño de Experimentos para El Número de Intercambios

Luego de realizar varias pruebas con el algoritmo se percibe la influencia del número de intercambios en la calidad de la respuesta. Razón por la cual se decidió realizar un diseño de experimentos de un solo factor para determinar si el efecto de este factor es relevante o no.

Diseño de Experimentos de un solo factor para el Ejercicio Eil B101

Los datos escogidos para este experimento se muestran en la tabla siguiente.

TABLA 24. DATOS ESCOGIDOS PARA EL DISEÑO DE EXPERIMENTOS DE UN SOLO FACTOR. NÚMERO DE INTERCAMBIOS.

	N=2	N=3	N=4
1	1134,59679	1132,55092	1133,40142
2	1133,66939	1132,51506	1133,42683
3	1133,43089	1133,47024	1132,5811
4	1133,81894	1133,42683	1133,47024
5	1133,97823	1133,7875	1132,52782
6	1133,72267	1133,40142	1132,56459
7	1133,53724	1133,69933	1132,69549
8	1133,70616	1132,28932	1132,5811
9	1134,54299	1133,76471	1132,39567

A continuación se presenta la descripción de los niveles del factor a tratar en el presente diseño.

C1, Corresponde a los Valores arrojados por el algoritmo cuando el Número de Intercambios es igual a 2.
C2, Corresponde a los Valores arrojados por el algoritmo cuando el Número de Intercambios es igual a 3.
C3, Corresponde a los Valores arrojados por el algoritmo cuando el Número de Intercambios es igual a 4.

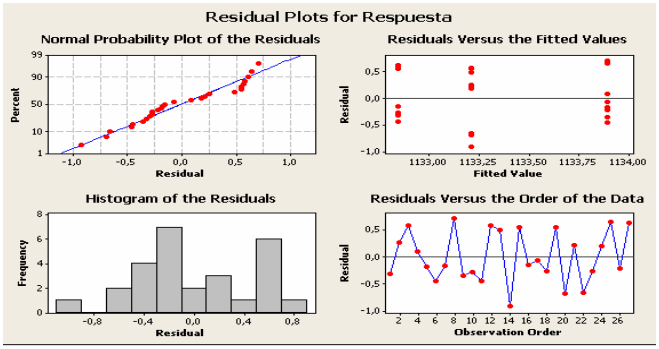
PRUEBA DE NORMALIDAD. EJERCICIO EIL B101.

A través de la gráfica de probabilidad normal se corrobora la asunción de normalidad de las poblaciones, esta se realizó en el programa Minitab 14. La hipótesis nula y alterna correspondiente son:

$$H_o = FDA\ y_i\ es\ N\left(\overline{\mu}_x, \sigma_x^2\right);$$
$$H_1 = FDA\ y_i\ no\ es\ N\left(\overline{\mu}_x, \sigma_x^2\right)$$
$$para\ i = 1,2,3.$$

Los resultados obtenidos se muestran en la figura 28. De acuerdo a la gráfica de probabilidad normal se presume que los residuos siguen una distribución, debido a las desviaciones pequeñas con respecto a la línea de ajuste. Además se corrobora la aleatoriedad de los mismos, gracias a la gráfica de los residuos con respecto al orden de la observación en la cual se percibe un patrón aleatorio entre las observaciones.

FIGURA 28. GRÁFICAS PARA EL ANÁLISIS DE LOS RESIDUOS DEL MODELO. PROGRAMA MINITAB 14. PROBLEMA EIL B101. NÚMERO DE INTERCAMBIOS.



PRUEBA DE IGUALDAD DE VARIANZAS. PROBLEMA EIL B101.

Se utilizó la Prueba de Igualdad de Varianzas de Levene para demostrar la igualdad de varianzas de los tratamientos tabla

Las hipótesis nula y alterna en este caso corresponden a:

$$H_o : \sigma_1^2 = \sigma_2^2 = \sigma_3^2$$
$$H_1 : Al\ menos\ una\ es\ diferente$$

El primer paso en esta prueba consiste en la obtención de la mediana para cada tratamiento, la siguiente ecuación nos permite la construcción de la matriz característica.

$$d_{i,j} = \left| y_{i,j} - \tilde{y}_{i.} \right| \quad donde$$

$\tilde{y}_{i.}$ *corresponde a la mediana*

$y_{i,j}$ *corresponde a los datos del tratamiento*

Esta matriz característica es sometida a una prueba Anova, y P-Value que se obtiene de esta nos permite identificar si las varianzas de los tratamientos son iguales o no. Estos valores se presentan a continuación: para ver la tabla completa ir al Anexo 15.

PValue	0.6905
F	0.3761

La matriz característica utilizada en este caso es:

LEVENE =

0.8741	0.8759	0.8203
0.0533	0.9118	0.8457
0.2918	0.0434	0
0.0963	0	0.8891
0.2556	0.3607	0.0533
0	0.0254	0.0165
0.1854	0.2725	0.1144
0.0165	1.1375	0
0.8203	0.3379	0.1854

Como se puede observar el valor de 0.6905 para el P-Value indica que no existe suficiente evidencia estadística para rechazar la hipótesis nula, por tanto se concluye que los datos tienen igual varianza.

PRUEBA DE ALEATORIEDAD DE DATOS. EJERCICIO EIL B101.

Para comprobar aleatoriedad de los datos, se realizó una prueba de corridas sobre los datos escogidos para el diseño 2^3 con puntos centrales, obteniéndose como resultado la siguiente tabla a través del programa "SPSS 7.5 for Windows Student Version".

TABLA 25. RESULTADOS DE LA PRUEBA DE ALEATORIEDAD. DISEÑO PARA EL FACTOR "NÚMERO DE INTERCAMBIOS".

NPar Tests

Runs Test

	VAR00001
Test Value[a]	1133,4309
Cases < Test Value	13
Cases >= Test Value	14
Total Cases	27
Number of Runs	12
Z	-,779
Asymp. Sig. (2-tailed)	,436

a. Median

En este caso el nivel crítico (Asymp. Sig. (2 - tailed) = 0.436) nos indica que no podemos rechazar la hipótesis de independencia entre las observaciones.

PRUEBA ANOVA DE UN SOLO FACTOR. EJERCICIO EIL B101.

Luego de corroborar el cumplimiento de las asunciones, se aplicó una Prueba Anova. La cual arrojó los resultados mostrados en las siguientes figuras.

FIGURA 29. RESULTADOS ARROJADOS POR EL PROGRAMA MATLAB 6.5 PARA LA PRUEBA ANOVA DE UN SOLO FACTOR. PROBLEMA EIL B101. EVAUACIÓN DEL COMPORTAMIENTO DEL FACTOR "NÚMERO DE INTERCAMBIOS".

```
TABLE =

    'Source'     'SS'          'df'    'MS'         'F'           'Prob>F'
    'Columns'    [ 5.0152]     [ 2]    [2.5076]     [10.4428]     [5.4609e-004]
    'Error'      [ 5.7631]     [24]    [0.2401]           []               []
    'Total'      [10.7784]     [26]          []           []               []
```

La prueba Anova para el factor "Número de Intercambios" (en este caso), arrojó un valor de 5.4609e-004 para el P-Value, lo cual indica que existen diferencias entre las medias de cada columna, además gracias a esto es posible inferir que el Factor "Número de Intercambios" es significante en la variable respuesta.

PRUEBA DE COMPARACIÓN DE MEDIAS. EJERCICIO EIL B101.

Como una prueba complementaria, se realizó la prueba de comparación de medias Dunn-Sidak a través del programa MatLab 6.5. Las pruebas de comparación de medias son utilizadas especialmente cuando el diseño de experimentos tiene interés en descubrir si existen diferencias significativas entre las medias de los diferentes tratamientos o la combinación de factores que conduce a un determinado resultado. Luego de la prueba ANOVA, la cual nos arroja la información primaria en relación a la significancia de factores, el experimentador puede realizar otros procedimientos que arrojen información de la calidad de la solución con respecto a los tratamientos o diferentes niveles de los factores, esto se logra con las pruebas de comparación de medias. La hipótesis nula y alterna para esta prueba son:

$$H_o : \mu_i = \mu_j$$

$$H_1 : \mu_i \neq \mu_j$$

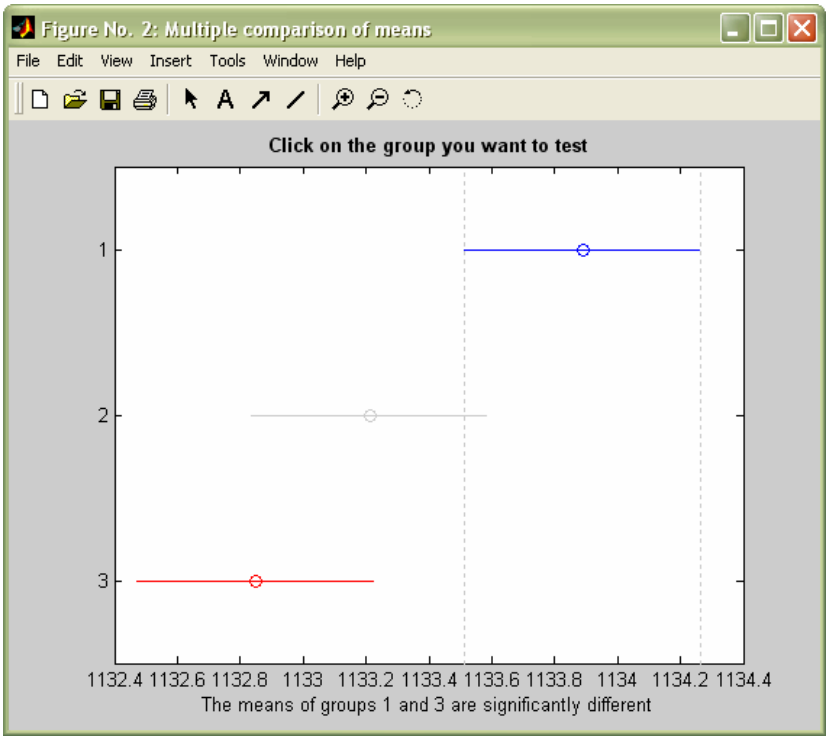
$$para\ \ toda\ \ i \neq j$$

Esta prueba utiliza como datos de entrada los resultados arrojados por la prueba ANOVA de la figura 29.Los resultados se muestran en las figuras siguientes.

FIGURA 30. RESULTADOS DE LA PRUEBA DE COMPARACIÓN DE MEDIAS. DUNN-SIDAK. EVALUACIÓN DEL FACTOR "NÚMERO DE INTERCAMBIOS"

```
STAT =

    gnames: [3x1 char]
         n: [9 9 9]
    source: 'anoval'
     means: [1.1339e+003 1.1332e+003 1.1328e+003]
        df: 24
         s: 0.4900

>> multcompare(STAT,0.01,'on', 'dunn-sidak', 'on',[1 2])

ans =

    1.0000    2.0000   -0.0748    0.6776    1.4299
    1.0000    3.0000    0.2875    1.0399    1.7923
    2.0000    3.0000   -0.3900    0.3623    1.1147

>>
```

FIGURA 31. GRÁFICA DE LA PRUEBA DE COMPARACIÓN DE MEDIAS. DUNN-SIDAK. EVALUACIÓN DEL FACTOR "NÚMERO DE INTERCAMBIOS"



Con base en los datos de las figuras anteriores es posible inferir que existen diferencias significativas entre las medias de la C1 y las medias de la C2, las cuales corresponden a los valores arrojados por el algoritmo para 2 y 4 intercambios respectivamente. De acuerdo a la figura, las mejores soluciones (menores distancias) se obtienen cuando el número de intercambios corresponden a 4. A continuación se presentan los intervalos de confianza para las medias.

$$P\{1133.55 \le \bar{x}_1 \le 1134.35\} = 99\% \quad N = 2$$
$$P\{1132.83 \le \bar{x}_2 \le 1133.65\} = 99\% \quad N = 3$$
$$P\{1132.45 \le \bar{x}_3 \le 1133.25\} = 99\% \quad N = 4$$

5.3 PRESENTACIÓN DE LOS RESULTADOS DEL DISEÑO DE EXPERIMENTOS

El diseño de experimentos se realizó con el fin de determinar si los factores utilizados tienen o no relevancia en la variable respuesta. Se realizaron las pruebas a dos ejercicios EIL 51 y EIL B101, los cuales corresponden a los problemas con el mayor y menor número de clientes respectivamente.

Luego de corroborar las asunciones principales para la aplicación de la prueba ANOVA en cada uno de los casos se procedió a realizarla, para el diseño factorial 2^3 con puntos centrales esta arrojó un F_0 de 0,00126506 para el problema EIL 51 y un F_0 0,000291113 para el problema EIL B101 para el efecto de la

curvatura de segundo orden. En ambos casos el F_0 es menor que el F (7.71 y 1.81 respectivamente) estos valores indican que no existe evidencia estadística para rechazar la hipótesis nula de la ANOVA, por tanto, el efecto de la curvatura de segundo orden de los factores no es significante.

Dada la no existencia para la curvatura de segundo orden, se practica un diseño factorial 2^3 para los dos ejercicios. En los dos ejercicios existe evidencia estadística para afirmar que el efecto del factor puro "Tamaño de la Población" no es significante y no debe tenerse en cuenta en el modelo matemático propuesto. Los valores obtenidos en una prueba ANOVA para el P-Value del factor antes mencionado es [0.2617] en el ejercicio EIL 51 (Anova factores puros), [0.6469] en el ejercicio EIL B101 (Anova para todas la combinaciones e los factores) y [0.8602] en el ejercicio EIL B101 (Anova para las interacciones de dos factores).

Existen diferencias con respecto a las interacciones de dos y tres factores en los dos ejercicios, lo cual se atribuye a la diferencia de Números de clientes y la capacidad de los vehículos en cada uno. En el ejercicio EIL 51 las interacciones no significativas son AB (Tamaño de la Lista*Tamaño de la Población), BC (Tamaño de la Población*Número de Iteraciones) y ABC (Tamaño de la Lista*Tamaño de la Población*Número de Iteraciones) dado que los P-Values obtenidos para cada una de ellas son grandes y corresponde a [0.7652 0.6837 0.9716] respectivamente. En el ejercicio EIL B101 la única interacción que resultó no significativa fue AC (Tamaño de la Lista Tabu* Número de iteraciones) con un P-Value igual a [0.5934].

Para enriquecer el estudio, se realizó la prueba "Dunn-Sidak", prueba de comparación múltiple de medias, útil para encontrar cual de los tratamientos presenta el mejor comportamiento. En esta se presentaron diferencias entre los dos ejercicios, para el ejercicio EIL 51 la mejor combinación de factores la ofrece el tratamiento 2 (Tamaño de la lista = 6; Tamaño de la Población =100; Número de Iteraciones = 1000) con el siguiente intervalo de confianza para la media muestral, de $P=\{567.9 \leq x2 \leq 577.3\}=99\%$. Para el ejercicio EIL B101 la mejor combinación de factores la ofrece el tratamiento 6 (Tamaño de la lista = 18; Tamaño de la Población =300; Número de Iteraciones = 1000) con el siguiente intervalo de confianza para la media muestral, de $P=\{1129 \leq x6 \leq 1135\}=99\%$.

Para determinar el efecto del número de intercambios se realizó un experimento adicional de un solo factor. En dicho experimento se realizaron corridas en el ejercicio EIL B101 para N= 2, 3 y 4. La Prueba Anova demuestra que en este caso existen diferencias significativas entre los tres niveles, dado que el P-Value arrojado en esta prueba fue de 0.0005, razón suficiente para rechazar la hipótesis nula. En este experimento también se realizó una prueba de comparación múltiple de medias para determinar con cuántos intercambios se obtienen mejores soluciones. Esta prueba apunta a N=4 como el nivel que ofrece los mejores resultados del experimento con el siguiente intervalo de confianza para la media muestral, de $P= \{1132.45 \leq x3 \leq 1133.25\}=99\%$.

6 CONCLUSIONES Y OBSERVACIONES

El algoritmo creado en el presente proyecto soluciona el problema de Ruteo de Vehículos con Capacidad, a través de la Meta-Heurística Búsqueda Tabú. Esta Metodología se apoya en las técnicas "Clarke and Wright" y "Lin Kernighan" para la construcción de la solución inicial y el vecindario respectivamente, además de las estrategias de Diversificación e Intensificación, las cuales permiten una exploración más amplia del espacio de soluciones. Las etapas y condiciones necesarias para el desarrollo del proyecto fueron planteadas y explicadas debidamente en las secciones precedentes de este mismo documento.

Se considera como el mayor logro de este proyecto y un gran aporte a la comunidad académica del área de Logística e Investigación de Operaciones la aplicación de las metodologías generales, comúnmente desarrolladas solo para el TSP (Versión genérica del problema tratado) al problema específico CVRP, identificando los aspectos a modificar y a adaptar a la situación específica asegurando el cumplimiento de todas las restricciones y garantizando la obtención de soluciones de calidad durante todo el proceso.

Al contrastar los datos del algoritmo con los de la Literatura se obtuvieron errores relativos del 2, 10, 11,12 y 14%, hallándose un solo valor extremo del 47% para el problema EIL B76. Siendo el problema EIL 51 el que presentó el menor error relativo. Es preciso mencionar también que en todos los casos estudiados la solución inicial Clarke and Wright fue mejorada por el procedimiento de la Búsqueda Tabú, presentando los mejores resultados el problema EIL 51, seguido por el problema EIL A101[20].

Cabe mencionar que las pruebas del diseño de experimentos fueron aplicadas a dos problemas diferentes. EIL 51 y EIL B101, dado que estos presentan el número menor y mayor de clientes. Con esto se pretendía analizar si el algoritmo responde de manera diferente cuando el problema a tratar es grande o pequeño. Inicialmente se utilizó un diseño factorial 2^3 con puntos centrales, pero al realizar la prueba ANOVA se demostró que el efecto de la curvatura de segundo orden (en los dos problemas) no es significante. Por ello se decidió aplicar un diseño factorial 2^3.

Gracias al diseño factorial 2^3 se descartó la influencia del factor "Tamaño de la Población" en la variable respuesta, pero en relación a las interacciones entre factores se obtuvieron resultados diferentes para los dos problemas. Para el problema EIL 51 las interacciones no significativas son "Tamaño de la Lista*Tamaño de la Población", "Tamaño de la Población*Número de Iteraciones" y "Tamaño de la Lista*Tamaño de la Población*Número de Iteraciones". En el ejercicio EIL B101 la única interacción que resultó no significativa fue Tamaño de la Lista Tabu* Número de iteraciones. Estas diferencias en los resultados se atribuyen a las diferencias en el número de clientes a atender y en la capacidad de los vehículos, dado que éstas son las únicas condiciones que difieren en los dos problemas (debido a la definición de los mismos).

Las pruebas de comparación de medias presentan diferentes resultados también, para el problema EIL 51 la mejor combinación de factores la ofrece el tratamiento 2 cuyos parámetros de entrada son: Tamaño de la lista = 6; Tamaño de la Población =100; Número de Iteraciones = 1000. Para el ejercicio EIL B101 la mejor

[20] Ver sección 5

combinación de factores la ofrece el tratamiento 6 es: Tamaño de la lista = 18; Tamaño de la Población =300; Número de Iteraciones = 1000. Dado que el segundo factor no es relevante, se puede afirmar que el problema EIL 51 (con 50 clientes) presenta mejores soluciones con un tamaño de lista pequeño (6) y que el problema EIL B101 (con 100 clientes) presenta mejores soluciones con un tamaño de lista grande (18)

Durante el proceso de pruebas del algoritmo se percibió que el Número de Intercambios (Metodología Lin Kernighan) ejerce cierta influencia sobre la variable respuesta, razón suficiente para realizar un diseño de experimentos adicional y corroborar esta teoría. El diseño que se practico fue un diseño de un solo factor con tres niveles para el Número de Intercambios, manteniendo los demás parámetros constantes en todas las corridas. De este diseño se obtuvo que los mejores resultados se presentan cuando el número de intercambios es igual a cuatro. Es claro que a pesar que se presenten mejores resultados, el tiempo computacional aumenta considerablemente dado el mayor esfuerzo para encontrar soluciones que cumplan con todas las condiciones y restricciones.

El presente proyecto tiene como usuario y cliente final a la comunidad académica en el área de Logística, Investigación de Operaciones, específicamente en el estudio de Metodología y Técnicas de Optimización. Además pretende consolidarse como un punto de partida para futuras investigaciones de este tipo, construyendo poco a poco una base de conocimientos más amplia y diversa. El código y el algoritmo son abiertos, lo que implica que pueden estar sujetos a modificaciones y comparaciones con otro tipo de metodología, y de esta forma contribuir al proceso de innovación y creación de nuevos Métodos más efectivos y eficientes, tanto en la calidad de la solución como en el tiempo de computacional.

BIBLIOGRAFÍA

1. GAMBOA, Dorabela; REGO, Cesar y GLOVER, Fred. “Implementation analysis of efficient heuristic algorithms for the traveling salesman problem”. 2006. Computers and Operations Research 33. Pages 1154-1172.

2. LYSGAARD, Jens; LETCHFORD, Adam y EGLESE, Richard. “A Branch and Cut algorithm for the Capacitated Open Vehicle Routing Problem”. Apr. 2006.

3. GAMBOA, Dorabela; REGO, Cesar y GLOVER, Fred. “Data structures and ejection chains for solving large-scale traveling salesman problem”. 2005. European Journal of Operational Research 160. Pages 154-171.

4. BARROS, Heydi J. “Optimización de Ruteo de Vehiculos empleando Busqueda Tabú”. Universidad de los Andes, Bogota- Colombia. 2005.

5. ATINEL, I.K.; ÖNCAN, T. “A new enhacement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem”. 2005. Journal of the Operational Research Society 56. Pages 954- 961.

6. VAN HEMERT, Jano I. “Property Analysis of Symmetric Traveling Salesman Problem Instances Acquired through Evolution”. 2005. Centre for Emergent Computing, Napier University, Edinburgh, UK. Springer – Verlag.

7. OLIVERA, Alfredo y VIERA, Omar. “Adaptive memory programming for the vehicle routing problem with multiple trips”. 20 Oct 2004. Elsevier Science.

8. CHIANG, W. C. y RUSSELL, R.A. “A Meta-heuristic for the Vehicle Routing Problem with soft time windows”. 2004. Journal of the Operational Research Society 55. Pages 1298 – 1310.

9. FISCHETTI, Matteo; TOTH, Paolo y FRANCESCHI, Roberto De. “A new heuristic algorithm for the vehicle routing problem”. Jan 2004.

10. KOCHENBERGER, Gary A; GLOVER, Fred; ALIDAEE, Bahram y REGO, Cesar. “A unified modeling and solution framework for combinatorial optimization problems”. 2004. OR Spectrum, Springer-Verlag.

11. BALDACCI, R.; HADJICONSTANTINOU, E y MINGOZZI, A. “An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow communication”. Sept-Oct 2004. Operations Research Vol. 52, N 5. Pages 723 – 738.

12. FUNKE, Birger; GRÜNERT, Tore y IRNICH, Stefan. “Local Search for Vehicle Routing and Scheduling Problems: Review and Conceptual Integration”. Apr 2004. Journal of Heuristics 11, Springer Science+ Business Media Inc. Pages 267 – 306.

13. MELIAN, Belén. “Búsqueda Tabú”. Feb 2004.

14. APPLEGATE, David; COOK, William y ROHE, André. “Chained Lin-Kernighan for Large Traveling Salesman Problems”. 2003. Journal on Computing 15, N 1. Page 82.

15. LYSGAARD, Jens; LETCHFORD, Adam y EGLESE, Richard. “A New Branch and Cut for the Capacitated Vehicle Routing Problem”. Nov 2003. Springer-Verlag.

16. GLOVER, Fred y MELIAN, Belen. “Tabu Search”. 2003. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 19. Pages 29 – 48.

17. FUNKE, Birger; GRÜNERT, Tore y IRNICH, Stefan. “A note on Single Alternating Cycle Neighborhoods for the TSP”. Sept 2003. Journal of Heuristics 11, Springer Science+ Business Media Inc. Pages 135 – 146.

18. HJERTENES, Oystein. “A multilevel scheme for the traveling salesman problem”. 2002.University Bergen.

19. JOHNSON, David y MCGEOCH, Lyle. “Experimental analysis of heuristics for the STSP”. 2002. Kluwer Academic Publishers. The Traveling Salesman Problem and its Variations.Pages 369-443.

20. CUMMINGS, Nigel. “A brief history or the TSP”. 2002. The OR Society.

21. LYSGAARD, Jens; LETCHFORD, Adam y EGLESE, Richard. “Multistars, partial Multistars and the capacitated vehicle routing problem”. Sept 2002. Springer- Verlag.

22. WALSHAW, Chris. “A Multilevel Lin-Kernighan Algorithm for the Traveling Salesman Problem”. Sept 2001. Computing and Mathematical Sciences. University of Greenwich.

23. LINDEROTH, Jeff; LEE, Eva y SAVELSBERGH, Martin. “A parallel, linear programming based heuristic for large scale set partitioning problems”. Aug. 2000.

24. ARAGON, A; DELGADO, C y PACHECO, J.A. “Diseño de Algoritmos para el problema del transporte escolar. Aplicación en la Provincia de Burgos”. 2000. QÜESTIIÓ vol. 24, 1. Pages. 55-82.

25. CAMPOS, V y MOTA, E. “Heuristics Procedures for the Capacitated Vehicle Routing Problem” 2000. Computational Optimization and Applications 16. Pages 265-277.

26. BAUER, P y LINDEROTH, J.T. “A Branch and Cut Approach to the Cardinality Constrained Circuit Problem”. Sept 1999.

27. ACHUTHAN, N.R.; CACETTA, L. y HILL, S.P. “Capacitated vehicle routing problem: some new cutting planes”. May 1998.Asia- Pacific Journal of Operational Research 15, N 1. Page 109.

28. REGO, Cesar. “Relaxed tours and path ejections for the traveling salesman problem”. 1998. European Journal of Operational Research 106. Pages 522-538.

29. DAHL, Geir. “An introduction to convexity, polyhedral theory and combinatorial optimization”. 19 Sept 1997.

30. ZHANG, Weixiong y KORF, Richard. “Performance of linear-space search algorithms”. 1995. Artificial Intelligence 79. Pages 241-292. University of California, Los Angeles.

31. JOHNSON, David y MCGEOCH, Lyle. “The traveling salesman problem: A case of Study in local optimization”. Nov 1995.

32. GLOVER, Fred. “Tabu Search Fundamentals and Uses”. Jun. 1995.Universidad de Colorado.

33. BATTITI, Roberto y TECCHIOLLI, Giampietro. “The reactive Tabu Search”. 1994. Journal on Computing 6, N 2. Pages 126 – 140.

34. FISCHETTI, Matteo; TOTH, Paolo y VIGO, Daniele. “A branch and bound algorithm for the capacitated vehicle routing problem on directed graphs”. Sept-Oct 1994. Operations Research. Page 846.

35. ARAQUE, J.R.; HALL, L.A. y MAGNANTI. “Capacitated trees, Capacitated routing and Associated Polyhedral”. Oct 1990.

36. LAI, Ten-Hwang y SAHNI, Sartaj. “Anomalies in parallel branch and bound algorithms”. Computer Science Department, the Ohio State University, Columbus, Ohio.

37. GARDEL, Pedro Esteban; GOMEZ, Oswaldo y BARAN, Benjamín. “Análisis del Omicrón ACO con optimización local”.Universidad de Asunción, Centro Nacional de Computación, San Lorenzo, Paraguay.

38. SCHRIJVER, Alexander. “On the history of combinatorial optimization”.

39. REGO, Cesar; GAMBOA, Dorabela y OSTERMAN, C. “On the performance of data structures for traveling salesman problem”.

40. PACHECO, Joaquin y DELGADO, Cristina. “Uso de conjunto de concentración en Búsqueda Tabú para problemas de rutas”. Departamento de Economía Aplicada, Universidad de Burgos, España.

41. ACHUTHAN, N.R.; CACETTA, L. y HILL, S.P. “An improved branch and cut algorithm for the capacitated vehicle routing problem”. Transportation Science 37, N 2, page 153.

42. MACIEJEWSKI, Michal y WALERJANCZYK Waldemar. “Solving Vehicle Routing Problem with Multiple Objective Metaheuristic Algorithms”.

43. HELSGAUN, Keld. “An effective implementation of the Lin Kernighan Traveling Salesman Heuristic”. Roskilde University, Denmark.

44. GREENE, William. “A Kernighan-Lin Local Improvement Heuristics that Softens Several Hard Problems in Genetic Algorithms”. University of New Orleans.

45. GLOVER, Fred. “Tabu Search and Adaptive Memory Programming, Advances, Applications and Challenges”. University of Colorado.

46. MARTINEZ, Amadís A. “Algoritmo Basado en Tabu Search para el cálculo del Índice de Transmisión de un grafo”. Universidad de Carabobo, Valencia, Venezuela.

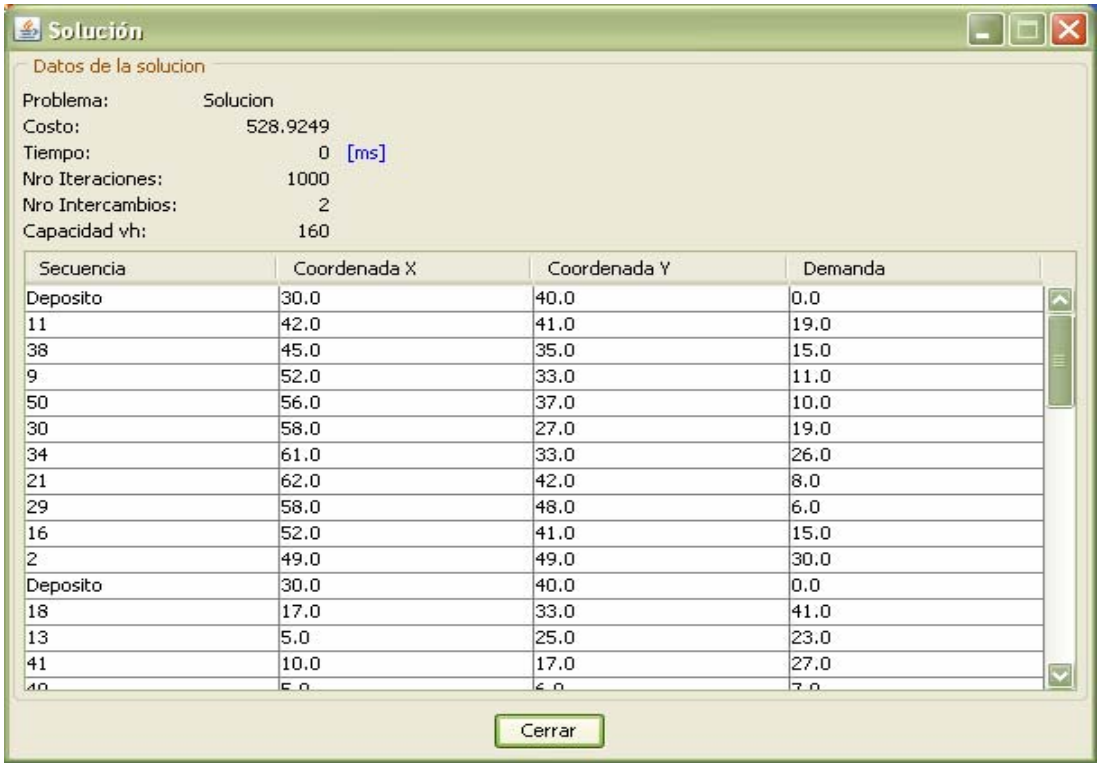
47. NILSSON, Christian. “Heuristics for the Traveling Salesman Problem”. Linköping University.

48. GENDREAU, Michel; LORI, Manuel y LAPORTE, Gilbert ; MARTELLO, Silvano. “A Tabu Search Heuristic for the Vehicle Routing Problem with two dimensional loading constraints”.

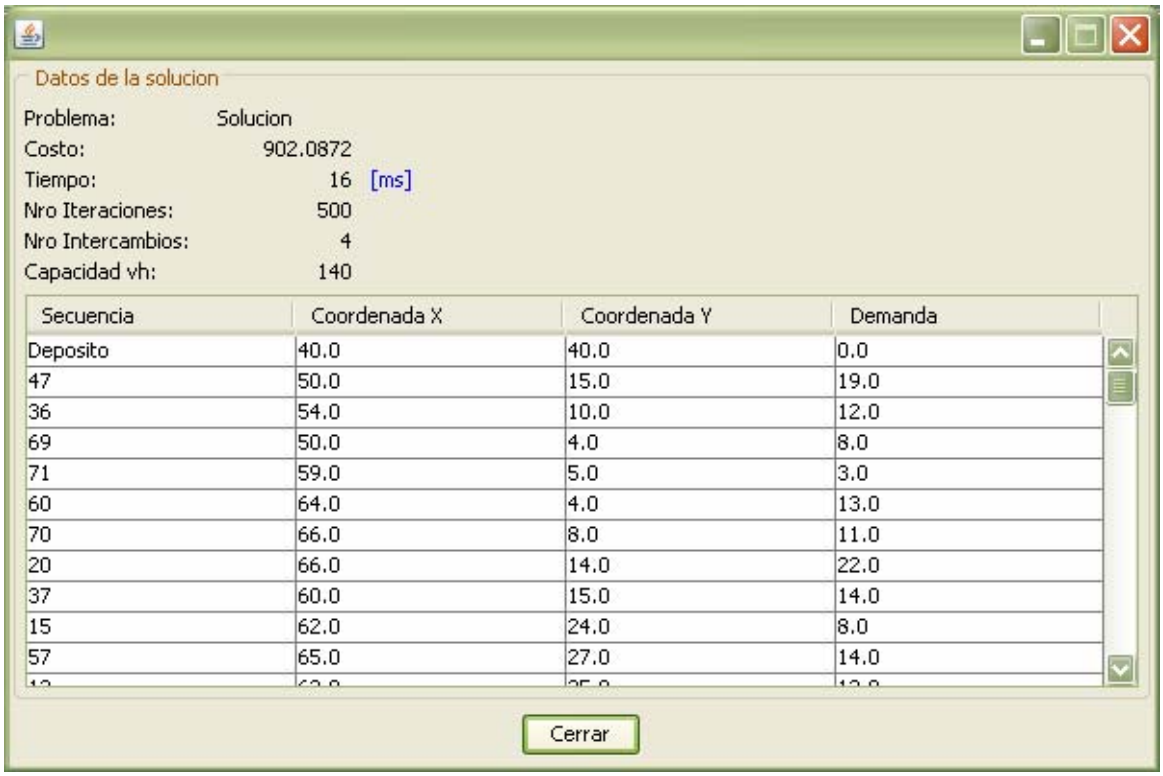
49. LAGUNA, Manuel y GLOVER, Fred. “Tabu Search”.

ANEXOS

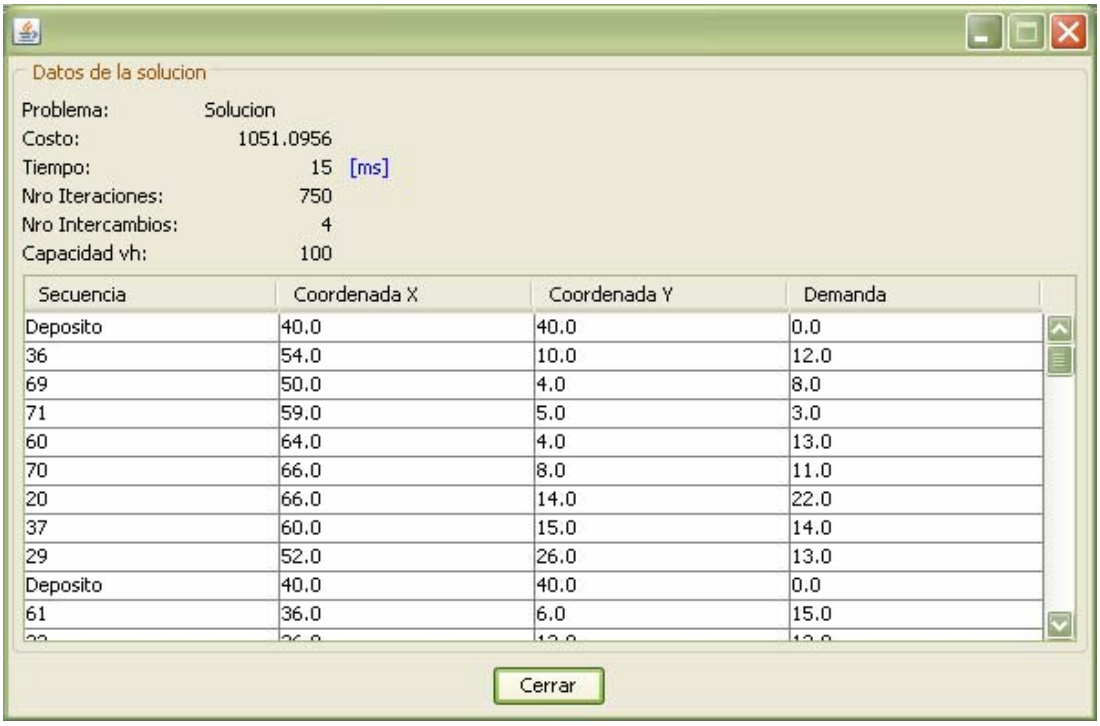
ANEXO 01. SALIDA DEL ALGORITMO. MEJOR RESPUESTA PROBLEMA EIL 51.



ANEXO 02. SALIDA DEL ALGORITMO. MEJOR RESPUESTA PROBLEMA EIL A76.



ANEXO 03. SALIDA DEL ALGORITMO. MEJOR RESPUESTA PROBLEMA EIL B76.



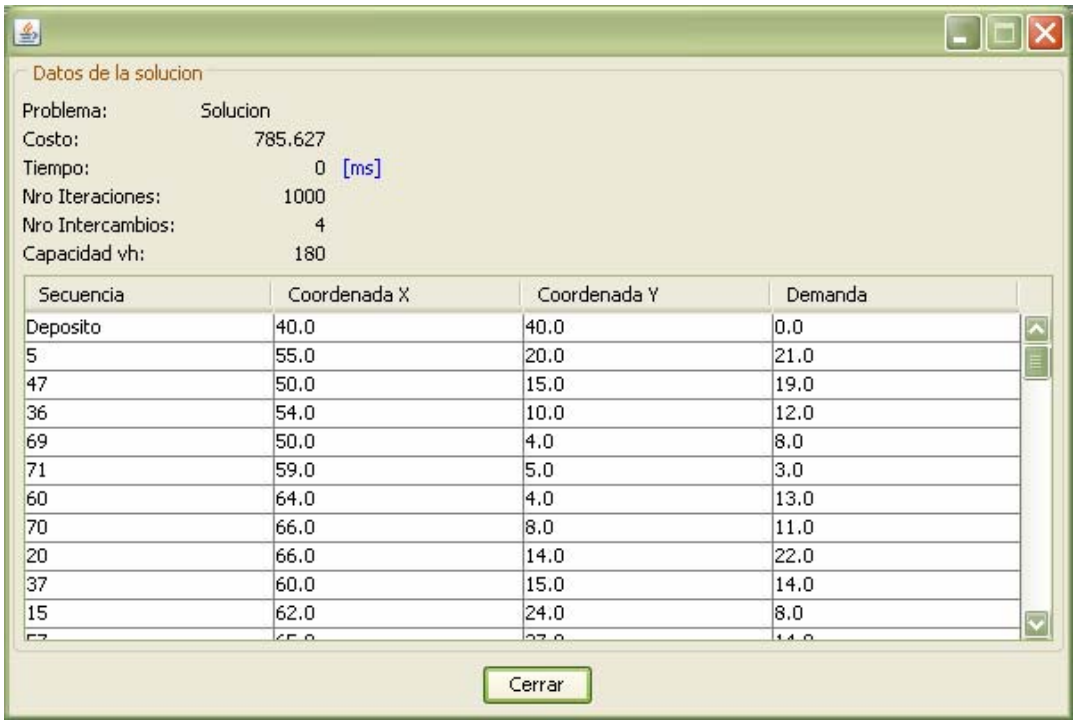
Datos de la solucion

Problema: Solucion
Costo: 1051.0956
Tiempo: 15 [ms]
Nro Iteraciones: 750
Nro Intercambios: 4
Capacidad vh: 100

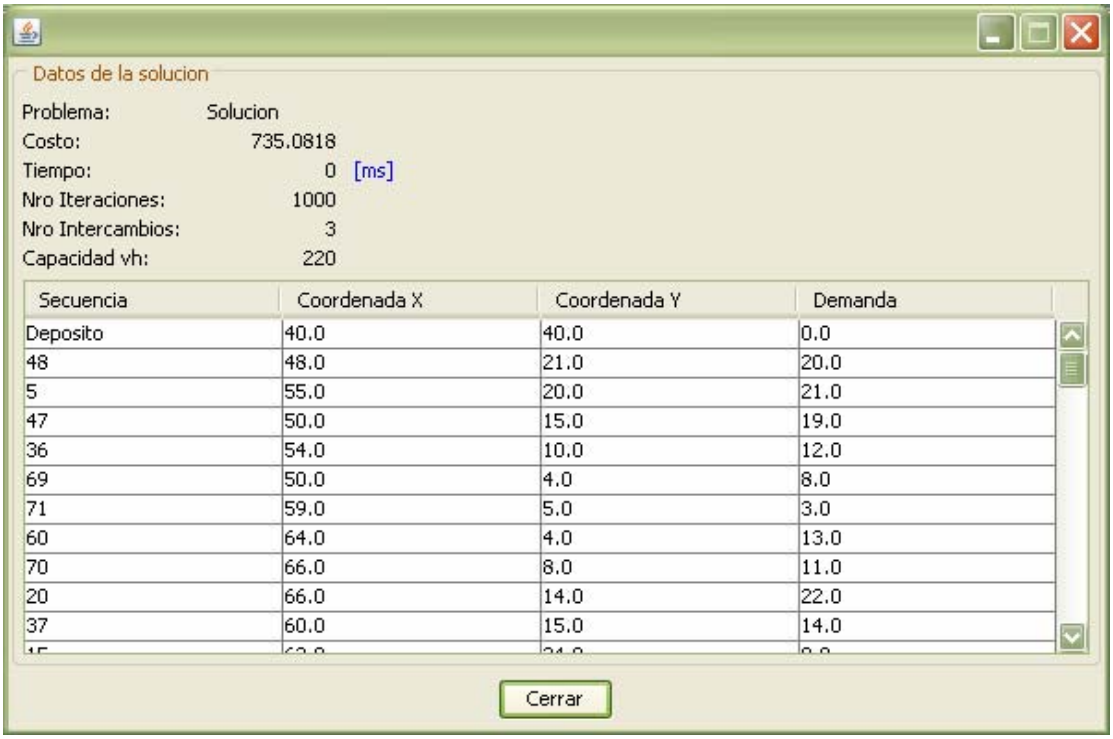
Secuencia	Coordenada X	Coordenada Y	Demanda
Deposito	40.0	40.0	0.0
36	54.0	10.0	12.0
69	50.0	4.0	8.0
71	59.0	5.0	3.0
60	64.0	4.0	13.0
70	66.0	8.0	11.0
20	66.0	14.0	22.0
37	60.0	15.0	14.0
29	52.0	26.0	13.0
Deposito	40.0	40.0	0.0
61	36.0	6.0	15.0

Cerrar

ANEXO 04. SALIDA DEL ALGORITMO. MEJOR RESPUESTA PROBLEMA EIL C76.



ANEXO 05. SALIDA DEL ALGORITMO. MEJOR RESPUESTA PROBLEMA EIL D76.



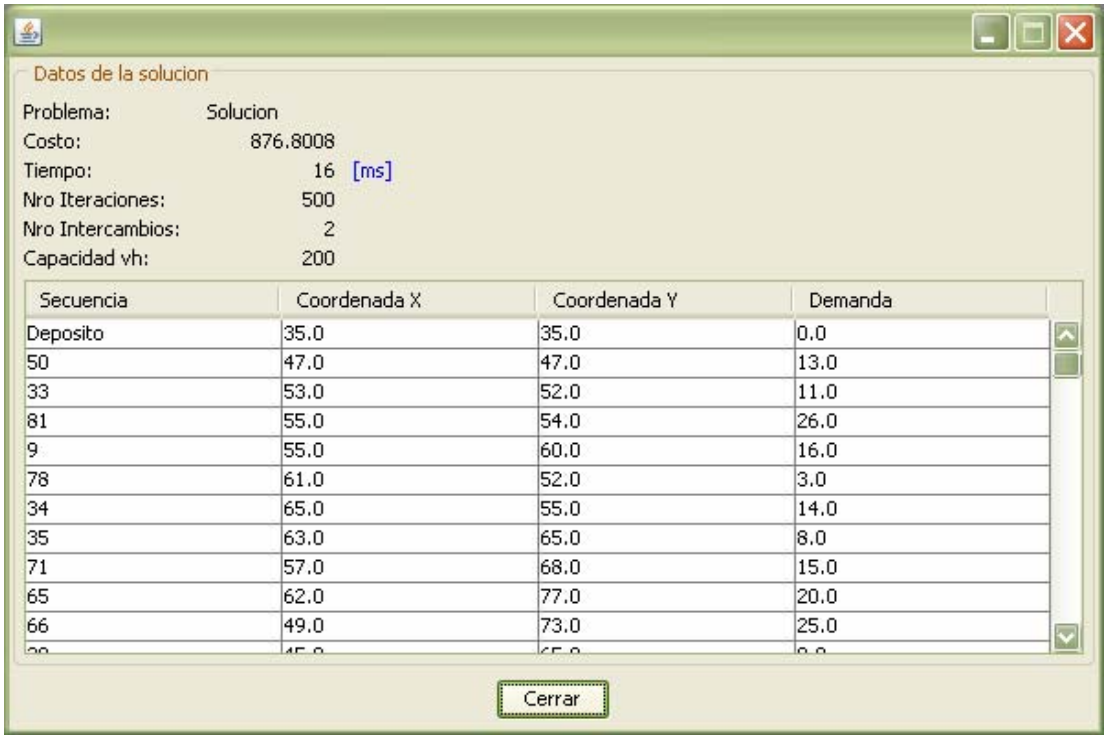
Datos de la solucion

Problema: Solucion
Costo: 735.0818
Tiempo: 0 [ms]
Nro Iteraciones: 1000
Nro Intercambios: 3
Capacidad vh: 220

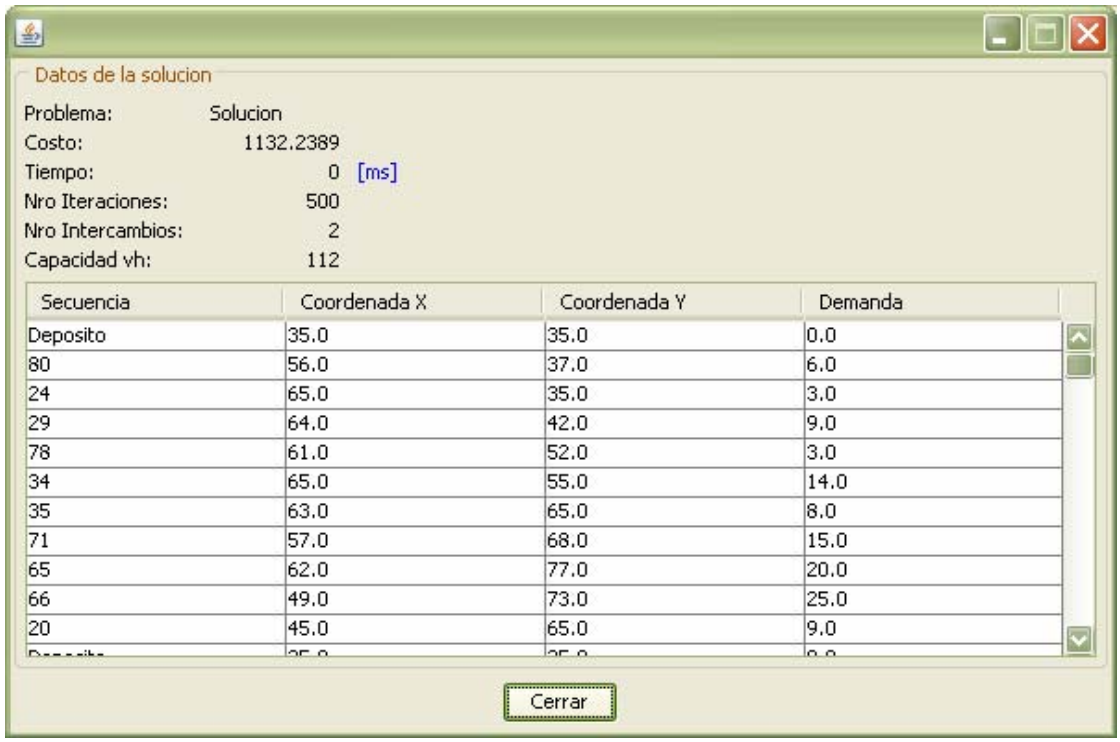
Secuencia	Coordenada X	Coordenada Y	Demanda
Deposito	40.0	40.0	0.0
48	48.0	21.0	20.0
5	55.0	20.0	21.0
47	50.0	15.0	19.0
36	54.0	10.0	12.0
69	50.0	4.0	8.0
71	59.0	5.0	3.0
60	64.0	4.0	13.0
70	66.0	8.0	11.0
20	66.0	14.0	22.0
37	60.0	15.0	14.0

Cerrar

ANEXO 06. SALIDA DEL ALGORITMO. MEJOR RESPUESTA PROBLEMA EIL A101.



ANEXO 07. SALIDA DEL ALGORITMO. MEJOR RESPUESTA PROBLEMA EIL B101.



ANEXO 08. SECUENCIA DE LA MEJOR RUTA OBTENIDA PARA CADA PROBLEMA.

	EIL 51	EIL A76	EIL B76	EIL C76	EIL D76	EIL A101	EIL B101
1	Deposito	Deposito	Deposito	Deposito	Deposito	Deposito	Deposito
2	11	47	36	5	48	50	80
3	38	36	69	47	5	33	24
4	9	69	71	36	47	81	29
5	50	71	60	69	36	9	78
6	30	60	70	71	69	78	34
7	34	70	20	60	71	34	35
8	21	20	37	70	60	35	71
9	29	37	29	20	70	71	65
10	16	15	Deposito	37	20	65	66
11	2	57	61	15	37	66	20
12	Deposito	13	22	57	15	20	Deposito
13	18	Deposito	64	13	57	51	92
14	13	73	42	54	13	1	37
15	41	1	41	Deposito	54	69	100
16	40	62	43	63	19	Deposito	85
17	19	28	Deposito	23	35	100	61
18	42	21	6	56	Deposito	91	5
19	17	5	75	41	24	85	Deposito
20	4	48	4	43	49	61	40
21	47	Deposito	67	42	23	16	4
22	Deposito	14	Deposito	64	56	86	54
23	46	59	34	22	41	38	68
24	5	66	46	61	43	44	77
25	49	65	7	21	42	14	76
26	10	11	26	74	64	43	Deposito
27	39	Deposito	17	Deposito	22	15	8
28	33	49	Deposito	17	61	42	46
29	45	24	39	40	21	Deposito	47
30	15	18	31	3	28	52	36
31	44	50	25	44	Deposito	7	49
32	37	25	55	32	58	88	64
33	12	55	18	9	10	31	11
34	Deposito	31	50	39	31	10	19
35	32	10	Deposito	72	25	62	Deposito
36	1	72	74	12	55	19	58
37	22	Deposito	21	Deposito	18	48	2
38	20	61	47	19	50	82	87
39	35	22	5	14	32	8	97

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

83		53	68	75	Deposito	68	Deposito
84		38	Deposito	Deposito		77	57
85		58	51			76	42
86		Deposito	3			12	43
87			44			28	15
88			32			Deposito	41
89			9			27	22
90			Deposito			89	75
91						96	74
92						99	72
93						59	73
94						93	Deposito
95						98	6
96						37	16
97						92	86
98						97	38
99						87	14
100						57	44
101						2	91
102						26	Deposito
103						Deposito	88
104						94	62
105						95	10
106						13	63
107						58	90
108						53	32
109						Deposito	30
110							70
111							69
112							Deposito
113							27
114							28
115							Deposito

ANEXO 09. RESULTADOS ARROJADOS POR EL PROGRAMA MATLAB 6.5 PARA LA PRUEBA DE IGUALDAD DE VARIANZAS LEVENE.

```
>> LEVENE=[L1,L2,L3,L4,L5,L6,L7,L8,L9]

LEVENE =

    0.3925    0.3932    0.8654    0.6032    0.3925    0.3925    0.7660    1.0761    0.3925
    0.1818    0.1818    0.6546    0.2107    0.6553    0.1818    0.2931    0.2107    0.1818
         0         0         0         0    0.9767         0    0.1001         0         0
    0.2107    0.4729    0.5039    0.2621    0.1818    0.6553    0.0994    0.0824    0.2107
    0.4729    0.6553    1.0955    0.4446         0    0.6836         0    1.0359    0.4729

>> [P,TABLE,STAT]=ANOVA1(LEVENE,[],'OFF')

P =

    0.6489


TABLE =

    'Source'     'SS'        'df'    'MS'        'F'         'Prob>F'
    'Columns'    [0.6463]    [ 8]    [0.0808]    [0.7485]    [0.6489]
    'Error'      [3.8852]    [36]    [0.1079]          []          []
    'Total'      [4.5315]    [44]          []          []          []


STAT =

    gnames: [9x1 char]
         n: [5 5 5 5 5 5 5 5 5]
    source: 'anoval'
     means: [0.2516 0.3406 0.6239 0.3041 0.4413 0.3826 0.2517 0.4810 0.2516]
        df: 36
         s: 0.3285

>>
```

ANEXO 10. RESULTADOS ARROJADOS POR EL PROGRAMA MATLAB 6.5 PARA LA PRUEBA DE IGUALDAD DE VARIANZAS LEVENE. DISEÑO FACTORIAL 2^3. PROBLEMA EIL 51.

```
LEVENE =

    0.3925    0.6553         0    0.6836         0    0.8654         0    1.9238
         0    0.1818    0.3678         0    0.6546         0    3.1690         0
    0.7660         0    1.7501    0.8847    0.1113    0.0824    0.9478    0.2931

>> [P, TABLE, STAT]=ANOVA1(LEVENE,[],'OFF')

P =

    0.7329


TABLE =

    'Source'     'SS'          'df'    'MS'         'F'          'Prob>F'
    'Columns'    [ 2.9247]     [ 7]    [0.4178]     [0.6191]     [0.7329]
    'Error'      [10.7974]     [16]    [0.6748]           []           []
    'Total'      [13.7222]     [23]          []           []           []


STAT =

    gnames: [8x1 char]
         n: [3 3 3 3 3 3 3 3]
    source: 'anova1'
     means: [0.3862 0.2790 0.7060 0.5228 0.2553 0.3159 1.3723 0.7390]
        df: 16
         s: 0.8215
```

ANEXO 11. RESULTADOS ARROJADOS POR EL PROGRAMA MATLAB 6.5 PARA EL ANÁLISIS DE VARIANZAS PARA TODAS LAS COMBINACIONES DE FACTORES. DISEÑO FACTORIAL 2^3. PROBLEMA EIL 51.

```
>> [P, TABLE, STAT]=ANOVAN(X,{F1 F2 F3}, 'full')

P =

    0.0000
    0.0495
    0.0000
    0.7652
    0.0000
    0.6837
    0.9716


TABLE =

    'Source'      'Sum Sq.'     'd.f.'    'Singular?'    'Mean Sq.'    'F'          'Prob>F'
    'X1'          [ 43.4393]    [   1]    [        0]    [ 43.4393]    [38.6757]    [1.2285e-005]
    'X2'          [  5.0714]    [   1]    [        0]    [  5.0714]    [ 4.5153]    [     0.0495]
    'X3'          [ 51.8880]    [   1]    [        0]    [ 51.8880]    [46.1978]    [4.2895e-006]
    'X1*X2'       [  0.1036]    [   1]    [        0]    [  0.1036]    [ 0.0923]    [     0.7652]
    'X1*X3'       [ 57.7545]    [   1]    [        0]    [ 57.7545]    [51.4210]    [2.2248e-006]
    'X2*X3'       [  0.1933]    [   1]    [        0]    [  0.1933]    [ 0.1721]    [     0.6837]
    'X1*X2*X3'    [  0.0015]    [   1]    [        0]    [  0.0015]    [ 0.0013]    [     0.9716]
    'Error'       [ 17.9707]    [  16]    [        0]    [  1.1232]          []             []
    'Total'       [176.4223]    [  23]    [        0]           []           []             []
```

ANEXO 12. RESULTADOS ARROJADOS POR EL PROGRAMA MATLAB PARA LA PRUEBA DE IGUALDAD DE VARIANZAS LEVENE.

```
LEVENE =

    0.2620    0.1496    0.8736    0.2918    0.2385    0.2385    0.2385    0.0659    0.2385
    0.0240    0.0963    0.5648    0.0533    0.1322    0.1322    0.1322    0.0121    0.1322
         0         0         0         0         0         0         0         0         0
    0.0300    0.0180    0.0254    0.0963    0.0231    0.0368    0.0368    0.1887    0.0368
    0.1260    0.1593    0.0659    0.1143    0.0368    0.1063    0.1063    0.4530    0.1677

>> [P, TABLE, STAT]=ANOVA1(LEVENE,[],'OFF')

P =

    0.5629


TABLE =

    'Source'     'SS'        'df'    'MS'        'F'         'Prob>F'
    'Columns'    [0.1942]    [ 8]    [0.0243]    [0.8540]    [0.5629]
    'Error'      [1.0231]    [36]    [0.0284]          []          []
    'Total'      [1.2173]    [44]          []          []          []


STAT =

    gnames: [9x1 char]
         n: [5 5 5 5 5 5 5 5 5]
    source: 'anoval'
     means: [0.0884 0.0846 0.3059 0.1111 0.0861 0.1028 0.1028 0.1440 0.1150]
        df: 36
         s: 0.1686

>>
```

ANEXO 13. RESULTADOS ARROJADOS POR EL PROGRAMA MATLAB 6.5 PARA LA PRUEBA DE IGUALDAD DE VARIANZAS LEVENE. PROBLEMA EIL B101. DISEÑO FACTORIAL 2^3.

```
LEVENE =

    0.7240    1.1659         0    0.5473    0.1144         0    0.3816         0
         0    1.6310    0.5902    0.5902    0.1854    0.4062    0.6585    1.1780
    0.1496         0    0.1245         0         0    0.1063         0    0.1063

>> [P, TABLE, STAT]=ANOVA1(LEVENE, [],'OFF')

P =

    0.4972


TABLE =

    'Source'     'SS'        'df'    'MS'        'F'         'Prob>F'
    'Columns'    [1.3671]    [ 7]    [0.1953]    [0.9502]    [0.4972]
    'Error'      [3.2883]    [16]    [0.2055]          []          []
    'Total'      [4.6554]    [23]          []          []          []


STAT =

    gnames: [8x1 char]
         n: [3 3 3 3 3 3 3 3]
    source: 'anoval'
     means: [0.2912 0.9323 0.2382 0.3792 0.0999 0.1708 0.3467 0.4281]
        df: 16
         s: 0.4533

>>
```

ANEXO 14. RESULTADOS ARROJADOS POR EL PROGRAMA MATLAB 6.5 PARA EL ANÁLISIS DE VARIANZAS PARA TODAS LAS COMBINACIONES DE FACTORES. DISEÑO FACTORIAL 2^3. PROBLEMA EIL B101.

```
>> [P, TABLE, STAT]=ANOVAN(X,{F1 F2 F3}, 'full')

P =

    0.0000
    0.0495
    0.0000
    0.7652
    0.0000
    0.6837
    0.9716


TABLE =

    'Source'        'Sum Sq.'     'd.f.'    'Singular?'    'Mean Sq.'    'F'           'Prob>F'
    'X1'            [ 43.4393]    [   1]    [        0]    [ 43.4393]    [38.6757]     [1.2285e-005]
    'X2'            [  5.0714]    [   1]    [        0]    [  5.0714]    [ 4.5153]     [     0.0495]
    'X3'            [ 51.8880]    [   1]    [        0]    [ 51.8880]    [46.1978]     [4.2895e-006]
    'X1*X2'         [  0.1036]    [   1]    [        0]    [  0.1036]    [ 0.0923]     [     0.7652]
    'X1*X3'         [ 57.7545]    [   1]    [        0]    [ 57.7545]    [51.4210]     [2.2248e-006]
    'X2*X3'         [  0.1933]    [   1]    [        0]    [  0.1933]    [ 0.1721]     [     0.6837]
    'X1*X2*X3'      [  0.0015]    [   1]    [        0]    [  0.0015]    [ 0.0013]     [     0.9716]
    'Error'         [ 17.9707]    [  16]    [        0]    [  1.1232]          []              []
    'Total'         [176.4223]    [  23]    [        0]            []          []              []
```

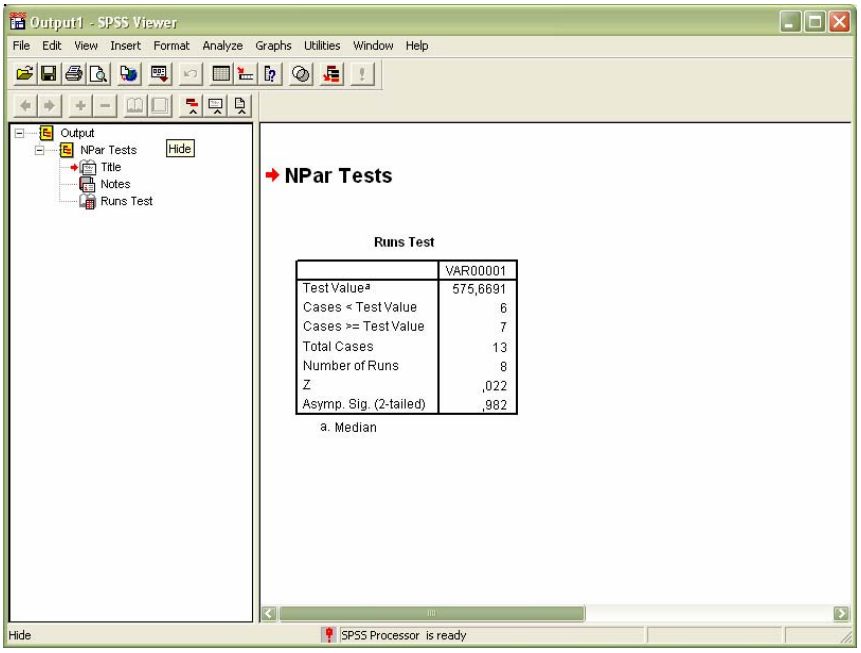
ANEXO 15. RESULTADOS ARROJADOS POR EL PROGRAMA MATLAB 6.5 PARA LA PRUEBA DE IGUALDAD DE VARIANZAS LEVENE. DISEÑO PARA EL NÚMERO DE INTERCAMBIOS.

```
>> LEVENE=[L1,L2,L3]

LEVENE =

    0.8741    0.8759    0.8203
    0.0533    0.9118    0.8457
    0.2918    0.0434         0
    0.0963         0    0.8891
    0.2556    0.3607    0.0533
         0    0.0254    0.0165
    0.1854    0.2725    0.1144
    0.0165    1.1375         0
    0.8203    0.3379    0.1854

>> [P, TABLE, STAT]=ANOVA1(LEVENE,[],'OFF')

P =

    0.6905


TABLE =

    'Source'     'SS'        'df'    'MS'        'F'         'Prob>F'
    'Columns'    [0.1138]    [ 2]    [0.0569]    [0.3761]    [0.6905]
    'Error'      [3.6325]    [24]    [0.1514]          []          []
    'Total'      [3.7463]    [26]          []          []          []
```

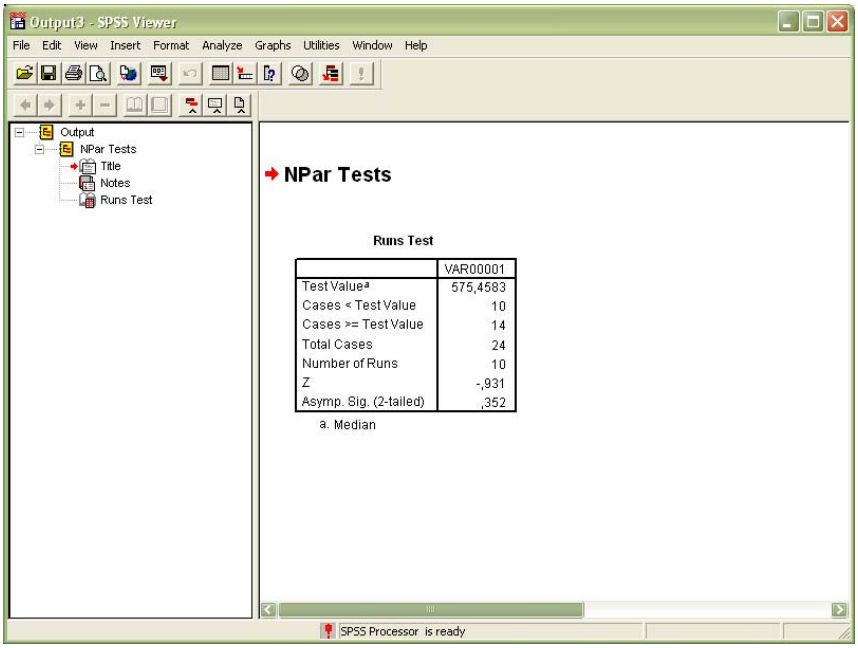
ANEXO 16. RESULTADOS ARROJADOS POR EL PROGRAMA MATLAB 6.5 PARA EL ANÁLISIS DE VARIANZAS DEL DISEÑO DE UN SOLO FACTOR.

```
MR =

  1.0e+003 *

    1.1346    1.1326    1.1334
    1.1337    1.1325    1.1334
    1.1334    1.1335    1.1326
    1.1338    1.1334    1.1335
    1.1340    1.1338    1.1325
    1.1337    1.1334    1.1326
    1.1335    1.1337    1.1327
    1.1337    1.1323    1.1326
    1.1345    1.1338    1.1324

>> [P, TABLE, STAT]=ANOVA1(MR,[])

P =

  5.4609e-004


TABLE =

    'Source'     'SS'          'df'    'MS'         'F'           'Prob>F'
    'Columns'    [ 5.0152]     [ 2]    [2.5076]     [10.4428]     [5.4609e-004]
    'Error'      [ 5.7631]     [24]    [0.2401]           []               []
    'Total'      [10.7784]     [26]          []           []               []
```

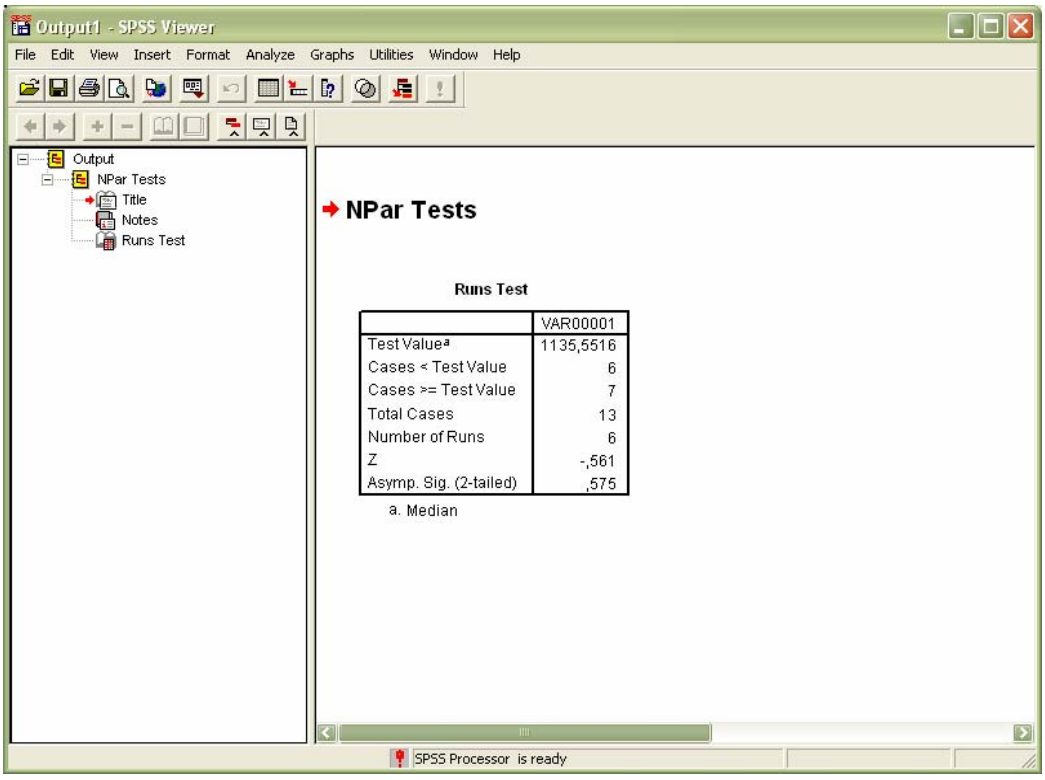
ANEXO 17. RESULTADOS ARROJADOS POR EL PROGRAMA SPSS PARA LA PRUEBA DE ALEATORIEDAD DEL DISEÑO FACTORIAL CON PUNTOS CENTRALES. PROBLEMA EIL 51.



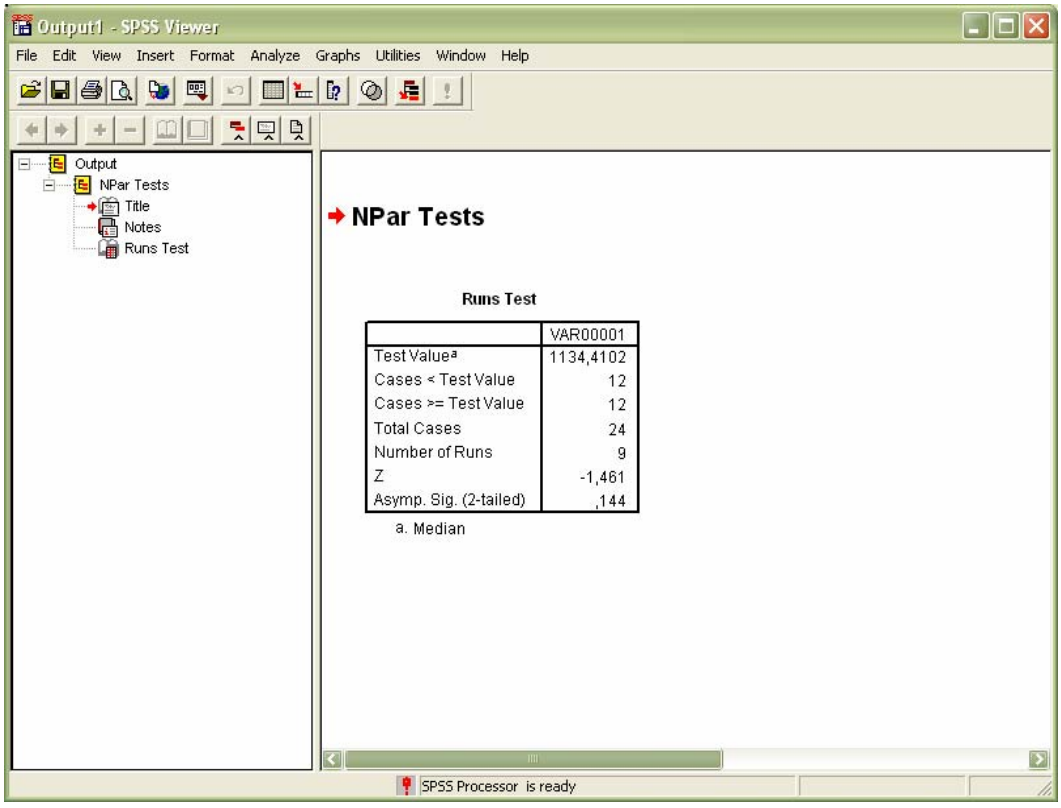
ANEXO 18. RESULTADOS ARROJADOS POR EL PROGRAMA SPSS PARA LA PRUEBA DE ALEATORIEDAD DEL DISEÑO FACTORIAL 2^3. PROBLEMA EIL 51.



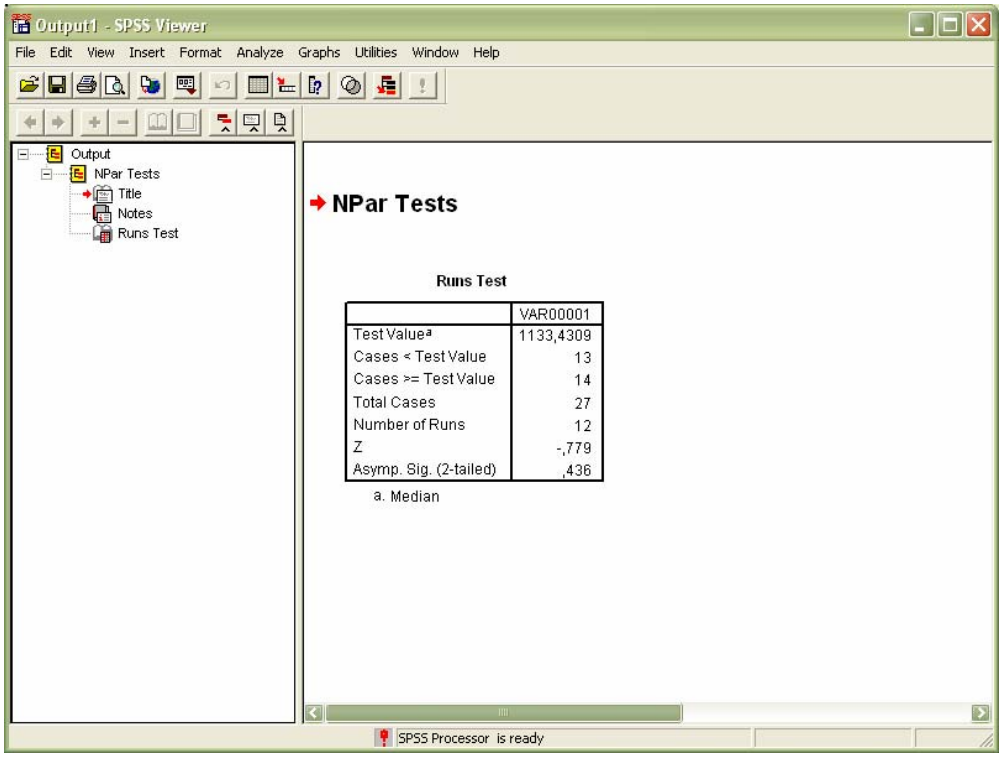
ANEXO 19. RESULTADOS ARROJADOS POR EL PROGRAMA SPSS PARA LA PRUEBA DE ALEATORIEDAD DEL DISEÑO FACTORIAL CON PUNTOS CENTRALES. PROBLEMA EIL B 101.



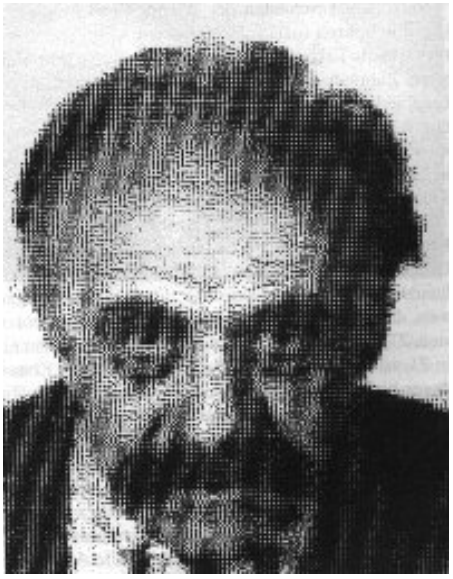
ANEXO 20. RESULTADOS ARROJADOS POR EL PROGRAMA SPSS PARA LA PRUEBA DE ALEATORIEDAD DEL DISEÑO FACTORIAL 2^3. PROBLEMA EIL B 101.



ANEXO 21. RESULTADOS ARROJADOS POR EL PROGRAMA SPSS PARA LA PRUEBA DE ALEATORIEDAD DEL DISEÑO DE UN SOLO FACTOR “NÚMERO DE INTERCAMBIOS”. PROBLEMA EIL B 101.



ANEXO 22. KARL MENGER



ANEXO 23. P.C. MAHALANOBIS

ANEXO 24. JULIA ROBINSON

ANEXO 25. GEORGE DANTZIG

ANEXO 26. MANUAL DE USO DEL PROGRAMA DE COMPUTADORA

MANUAL DE USO DEL PROGRAMA DE COMPUTADORA

Este apartado presentará una descripción del uso del CVRP Solver en cada una de sus funcionalidades. A pesar que el uso del mismo puede darse de manera intuitiva, este manual tiene por objeto explicar detalladamente las funciones y las razones por las cuales fueron creadas.

RESEÑA

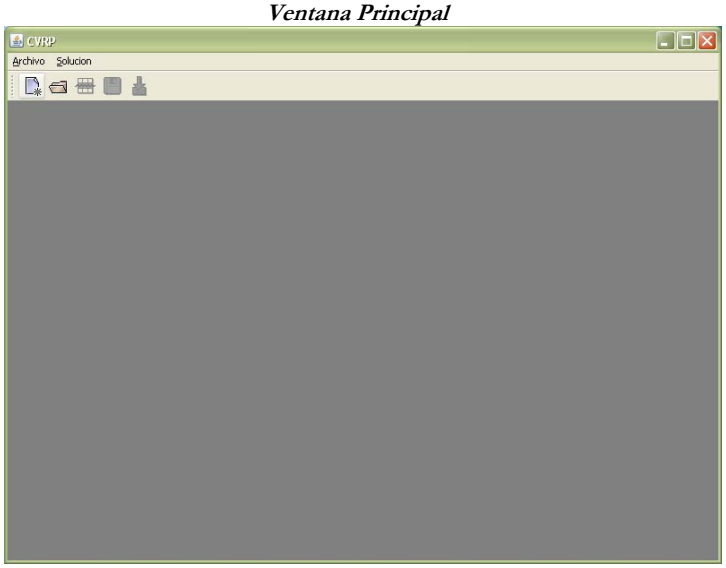
Luego de la correcta instalación del programa de computadora, debe darse clic en el icono de acceso directo CVRP Solver.exe para dar inicio e ingresar al mismo. Inmediatamente se muestra la imagen de la *Figura 47,* la cual indica que el programa se está cargando en memoria.

Imagen de CVRP Solver.

VENTANA PRINCIPAL

La ventana principal es la única ventana que permanece vigente durante la ejecución del programa, dado que esta contiene todas las actividades que éste puede realizar. Además, cabe resaltar que cada una de estas actividades puede ser ejecutada de manera más rápida gracias a la utilización de diversas combinaciones en el teclado. La siguiente figura ilustra la Ventana Principal.

Ventana Principal

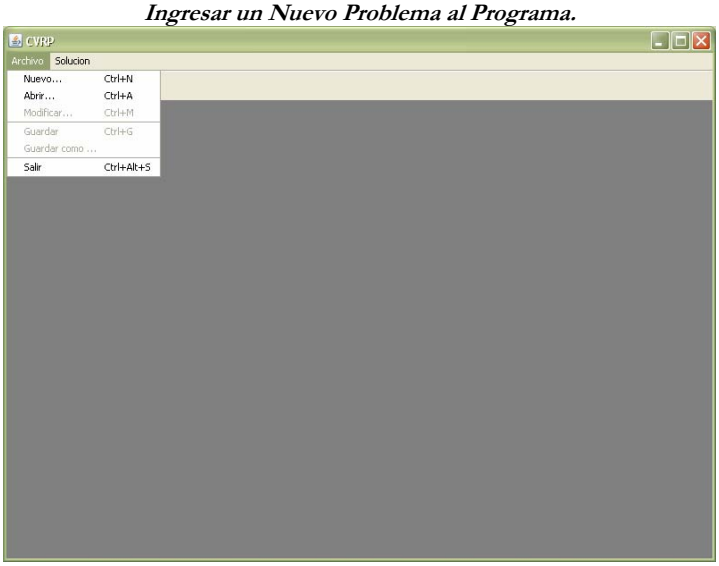


.

INGRESAR UN NUEVO PROBLEMA AL PROGRAMA.

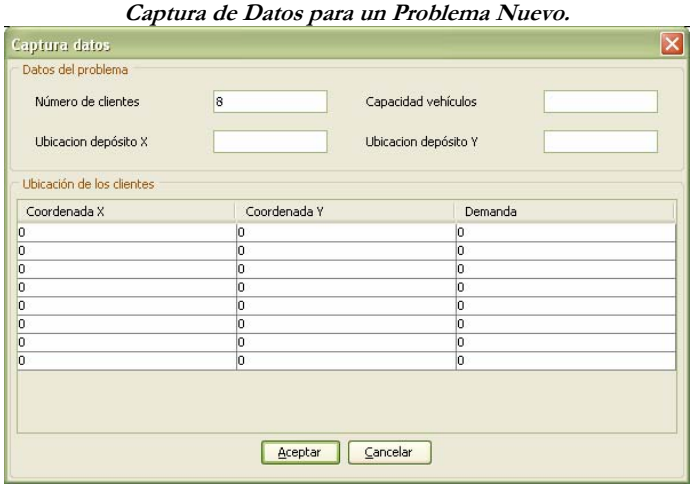
Para ingresar un nuevo problema al programa de computadora debe hacerse clic en el *Archivo* y luego en *Nuevo,* o de la forma rápida a través del teclado con las teclas ctrl.+ N o por medio del icono  .

Ingresar un Nuevo Problema al Programa.



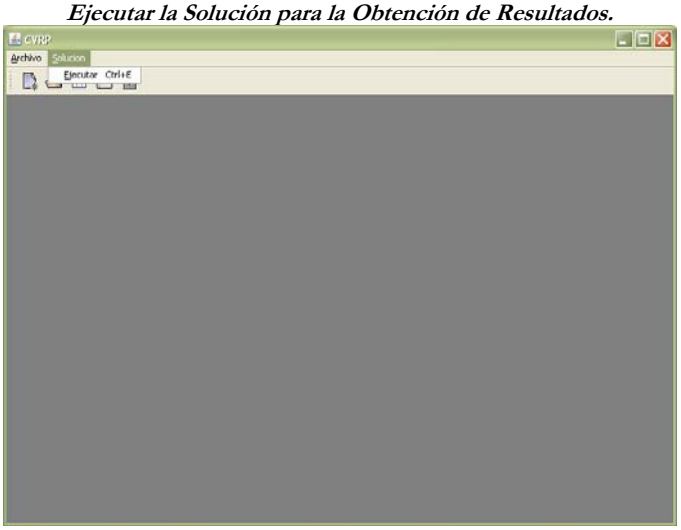
Al finalizar esta secuencia de pasos se muestra el cuadro de dialogo de la figura 49. En el cual deben ingresarse los datos siguientes: Número de Clientes, Capacidad de Vehículos, Ubicación depósito X (Coordenada X de la ubicación del depósito), Ubicación depósito Y (Coordenada Y de la ubicación del depósito). Dependiendo del Número de clientes registrado, se desplegaran celdas en igual número para ingresar los datos de los clientes correspondientes a la Coordenada en X, La Coordenada en Y y la demanda. Luego se da clic en *Aceptar*.

Captura de Datos para un Problema Nuevo.



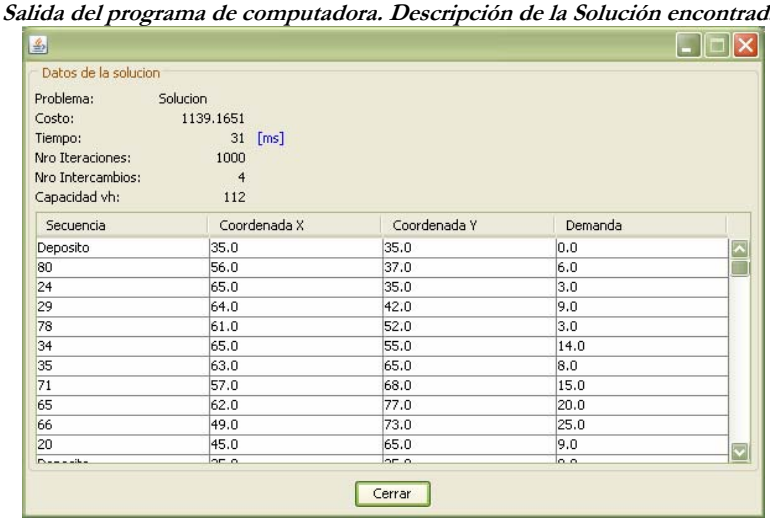
Para obtener los resultados se dará clic en *Solución* y luego en *Ejecutar*, como se muestra en la figura 50, o por medio del acceso rápido a través del teclado con las teclas ctrl. + E o por medio del icono  .

Ejecutar la Solución para la Obtención de Resultados.



Luego de la Ejecución del programa se muestra la solución, ver figura 51. En este cuadro podemos encontrar todos los datos relacionados a la solución, como el costo de la solución, los parámetros de entrada (Número de Iteraciones, Número de Intercambios, Capacidad de los Vehículos), además se presentan en una tabla la secuencia de atención a los clientes con sus respectivos datos (Coordenada en X, Coordenada en Y, Demanda).

Salida del programa de computadora. Descripción de la Solución encontrada.

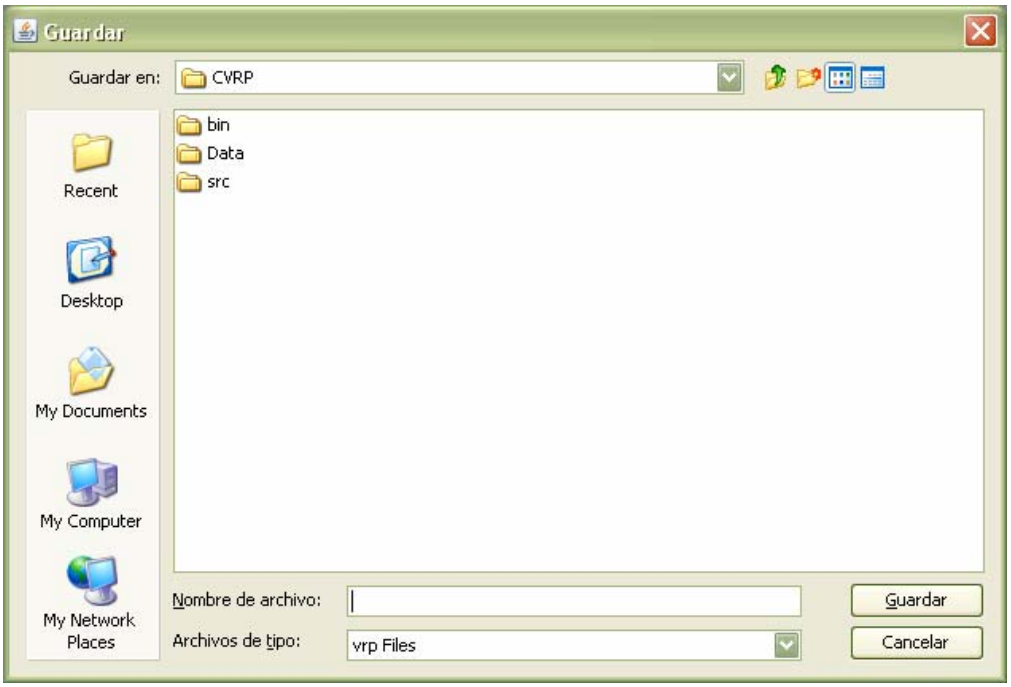


Durante el proceso descrito anteriormente puede utilizarse la opción *Modificar,* en el menú *Archivo,* la cual nos mostrará inmediatamente la información ingresada en el cuadro de la figura 07, la cual esta lista para ser modificada. A esta función puede accederse de manera rápida por medio del teclado con las teclas ctrl. + M o por medio del icono .

GUARDAR UN PROBLEMA

Para guardar un problema nuevo debe hacerse clic en *Guardar Como* del menú *Archivo*, luego de esta secuencia se presenta un cuadro de diálogo que nos permite ubicar el problema en las diferentes carpetas de nuestro computador.

Guardar un Problema Nuevo.



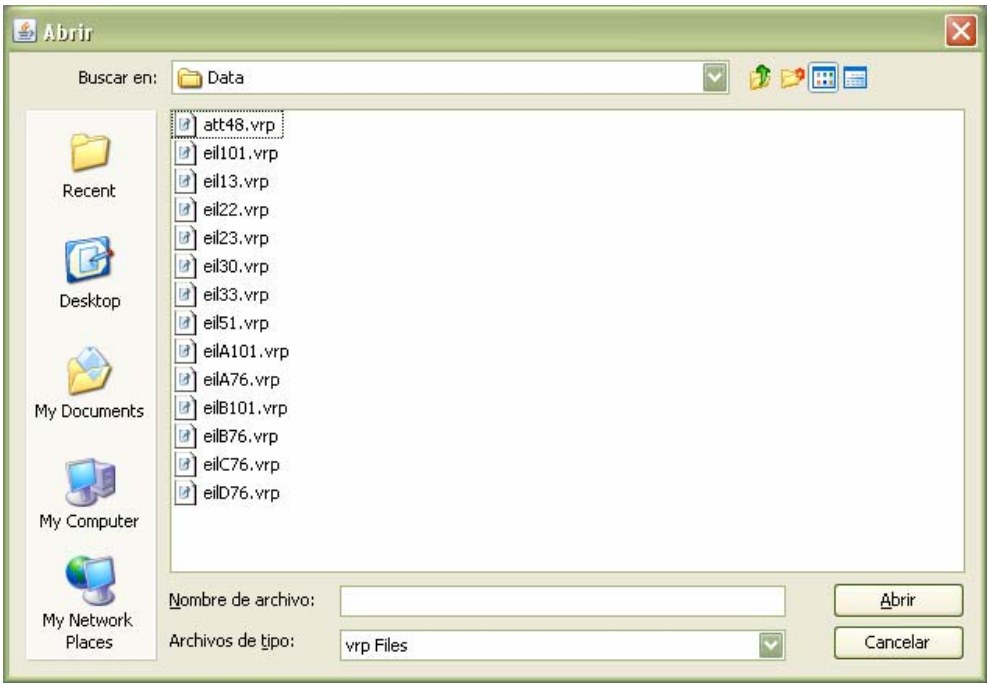
Cuando el problema ya ha sido previamente guardado por medio del procedimiento descrito anteriormente, solo debe darse *Guardar* en el Menú *Archivo*, para guardar los cambios que se le han realizado últimamente. Puede ingresarse a esta opción de manera rápida por medio de las teclas ctrl. + G o por medio del icono .

ABRIR UN PROBLEMA DESDE UN ARCHIVO.

El programa es capaz de abrir un archivo del tipo *.vrp. Para realizar esta operación debe darse *Abrir* en el Menú *Archivo*, o por medio de las teclas ctrl. + A o por medio del icono , luego de esta secuencia se abrirá

un cuadro de diálogo como se muestra en la figura 53, en el cual se podrá indicará la ubicación del problema a resolver.

Abrir un Problema desde Archivo.



Luego de abrir el archivo deseado, se procede a Ejecutar el programa, y se obtendrá la respuesta de forma idéntica a la descrita en la sección 9.3.

CARACTERÍSTICAS DEL LA CLASE DE ARCHIVO QUE LEE EL PROGRAMA.

El programa de computadora es capaz de abrir un archivo del tipo *.vrp, que presenta las siguientes características:

La primera fila llamada “NAME”, esta contiene el nombre del problema a tratar. Ver figura 11.

La segunda fila llamada “COMMENT”, contiene información relacionada con el número de vehículos mínimo permitido y el mejor valor encontrado y publicado en la librería TSPLIB. Ver figura 11.

La tercera fila llamada “TYPE”, muestra el tipo de problema a tratar. (en nuestro caso siempre será CVRP). Ver figura 11.

La cuarta fila llamada “DIMENSION”, muestra el número de instancias en el problema (número de clientes y el depósito). Ver figura 11.

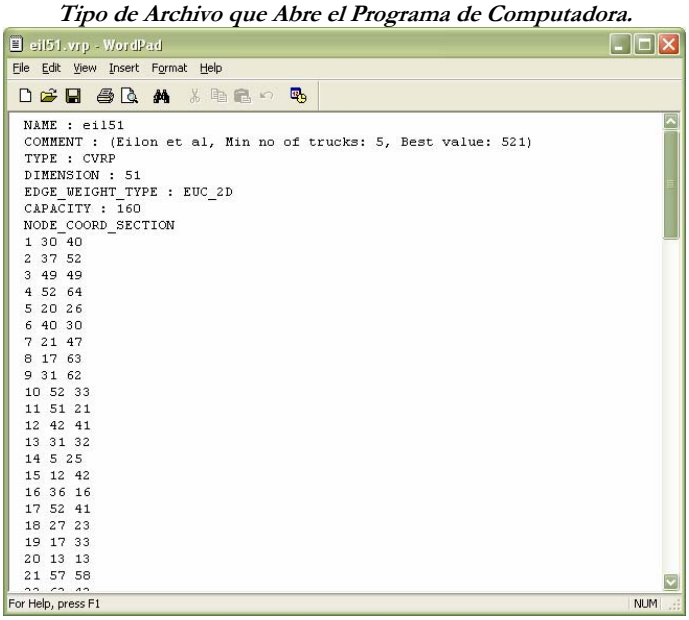
La quinta fila llamada “EDGE WEIGHT TYPE” indica el tipo de distancias entre las distintas instancias, en todos los casos tratados serán distancias euclidianas de dos dimensiones (EUC_2D)

La sexta fila llamada “CAPACITY” muestra la capacidad de cada vehiculo.

La séptima fila llamada “NODE COORD SECTION”, es el título para la siguiente sección, que muestra las coordenadas de cada instancia. Luego de ésta seguirán tantas filas como hayan sido declaradas en la fila llamada “DIMENSION”. En cada una de estas filas se declara la ubicación de las

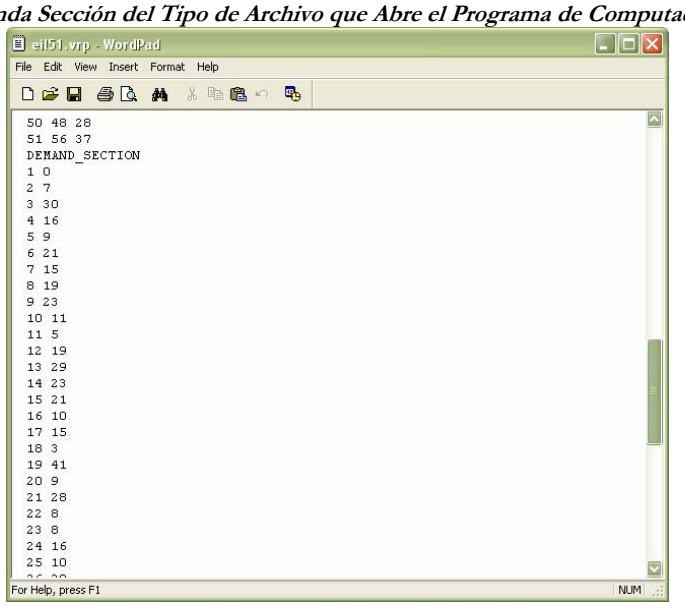
instancias del problema, donde el primer número corresponde al número de la instancia, el siguiente corresponde a la Coordenada X de ubicación y el tercer número corresponde a la Coordenada Y de ubicación. Ver figura 54.

Tipo de Archivo que Abre el Programa de Computadora.



La última sección corresponde a “DEMAND SECTION”, la cual muestra la demanda de cada instancia, identificándose el depósito por tener una demanda igual a cero. Ver figura 55.

Segunda Sección del Tipo de Archivo que Abre el Programa de Computadora.



ANEXO 27. DESCRIPCIÓN PROBLEMA EIL 51.

#	X	Y	Demanda
1	30	40	0
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

#	X	Y	Demanda
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

ANEXO 28. DESCRIPCIÓN PROBLEMA EIL A76.

#	X	Y	Demanda
1	0	40	0
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
26	17	64	14
27	41	46	18
28	55	34	17
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
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

ANEXO 29. DESCRIPCIÓN PROBLEMA EIL B76.

#	X	Y	Demanda
1	0	40	0
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

#	X	Y	Demanda
26	17	64	14
27	41	46	18
28	55	34	17
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
51	15	56	22

#	X	Y	Demanda
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

ANEXO 30. DESCRIPCIÓN PROBLEMA EIL C76.

#	X	Y	Demanda
1	0	40	0
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

#	X	Y	Demanda
26	17	64	14
27	41	46	18
28	55	34	17
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
51	15	56	22

#	X	Y	Demanda
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

ANEXO 30. DESCRIPCIÓN PROBLEMA EIL D76.

#	X	Y	Demanda
1	0	40	0
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

#	X	Y	Demanda
26	17	64	14
27	41	46	18
28	55	34	17
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
51	15	56	22

#	X	Y	Demanda
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

ANEXO 31. DESCRIPCIÓN PROBLEMA EIL A101.

#	X	Y	Demanda
1	35	35	0
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

#	X	Y	Demanda
25	65	35	3
26	65	20	6
27	45	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

#	X	Y	Demanda
51	47	47	13
52	49	58	10
53	27	43	9
54	37	31	14
55	57	29	18
56	63	23	2
57	53	12	6
58	32	12	7
59	36	26	18
60	21	24	28
61	17	34	3
62	12	24	13
63	24	58	19
64	27	69	10
65	15	77	9
66	62	77	20
67	49	73	25
68	67	5	25
69	56	39	36
70	37	47	6
71	37	56	5
72	57	68	15
73	47	16	25
74	44	17	9
75	46	13	8
76	49	11	18

#	X	Y	Demanda
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	31	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

ANEXO 32. DESCRIPCIÓN PROBLEMA EIL B101.

#	X	Y	Demanda
1	35	35	0
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

#	X	Y	Demanda
25	65	35	3
26	65	20	6
27	45	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

#	X	Y	Demanda
51	47	47	13
52	49	58	10
53	27	43	9
54	37	31	14
55	57	29	18
56	63	23	2
57	53	12	6
58	32	12	7
59	36	26	18
60	21	24	28
61	17	34	3
62	12	24	13
63	24	58	19
64	27	69	10
65	15	77	9
66	62	77	20
67	49	73	25
68	67	5	25
69	56	39	36
70	37	47	6
71	37	56	5
72	57	68	15
73	47	16	25
74	44	17	9
75	46	13	8
76	49	11	18

#	X	Y	Demanda
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	31	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

ANEXO 36. DOCUMENTACIÓN DEL CÓDIGO FUENTE DEL PROGRAMA DE COMPUTADORA. PAQUETE KERNEL.

clsDatosProblema.java

En esta sección del programa se toman los datos del problema y se almacenan en una estructura de datos que es utilizada por las otras clases del paquete.

```
package Kernel;

public class clsDatosProblema {
	static public int MAXCLIENTES=110;
	private int CapacidadVehiculos;
	private int NumeroClientes;
	private boolean Guardado;
	private double Clientes[][];
	public clsDatosProblema(){
		Clientes = new double[MAXCLIENTES][3];
		reset();
	};
	public void reset(){
		NumeroClientes=0;
		CapacidadVehiculos=0;
		Guardado=true;
		for(int i=0; i<MAXCLIENTES;i++){
			Clientes[i][0]=0.0;
			Clientes[i][1]=0.0;
			Clientes[i][2]=0;
		}
	}
	public void addCliente(double CoordX, double CoordY, int Demanda){
		Guardado=false;
		NumeroClientes++;
		//En este punto se requiere validacion de los datos que
		//estoy ingresando, como que el cliente no esté duplicado
		//Si la validacion no es exitosa, dsepertar Exception 100
		Clientes[NumeroClientes][0]=CoordX;
		Clientes[NumeroClientes][1]=CoordY;
		Clientes[NumeroClientes][2]=Demanda;
	}
	public void setDeposito(double CoordX, double CoordY){
		Guardado=false;
		//En este punto se requiere validacion de los datos
		Clientes[0][0]=CoordX;
		Clientes[0][1]=CoordY;
```

```
                Clientes[0][2]=0;
        }
        public void setX(int Cliente, double value){
                Guardado=false;
                Clientes[Cliente][0] = value;
        }
        public void setX(int Cliente, Double value){
                Guardado=false;
                Clientes[Cliente][0] = value.doubleValue();
        }
        public void setY(int Cliente, double value){
                Guardado=false;
                Clientes[Cliente][1] = value;
        }
        public void setY(int Cliente, Double value){
                Guardado=false;
                Clientes[Cliente][1] = value.doubleValue();
        }
        public void setD(int Cliente, double value){
                Guardado=false;
                Clientes[Cliente][2] = value;
        }
        public void setD(int Cliente, Double value){
                Guardado=false;
                Clientes[Cliente][2] = value.doubleValue();
        }
        public double X(int Position){//Coordenada Y del cliente position
                if (Position > this.NumeroClientes){
                        return 0.0;
                }else
                        return Clientes[Position][0];
        }
        public double Y(int Position){//Coordenada Y del cliente position
                if (Position > this.NumeroClientes){
                        return 0.0;
                }else
                        return Clientes[Position][1];
        }
        public double D(int Position){//Demanda del cliente position
                if (Position > this.NumeroClientes){
                        return 0.0;
                }else
                        return Clientes[Position][2];
        }
```

```
//Propiedad numero de clientes
public int NumeroClientes(){
		return this.NumeroClientes;
}
//Propiedad tieneDatos indica si hay informacion almacenada en el objeto
public boolean tieneDatos(){
		return NumeroClientes > 0;
}
public void setNumeroClientes(int NumeroClientes){
		Guardado=false;
		this.NumeroClientes=NumeroClientes;
}

//propiedad capacidad de vehiculos
public int CapacidadVehiculos(){
		return this.CapacidadVehiculos;
}
public void setCapacidadVehiculos(int CapacidadVehiculos){
		Guardado=false;
		this.CapacidadVehiculos = CapacidadVehiculos;
}
//Propiedad Guardado
public boolean Guardado(){
		return Guardado;
}
//Propiedad Guardado
public void setGuardado(){
		Guardado=true;
}
//Toma la informacion del objeto y la asigna a otro
public void Assign(clsDatosProblema DT){
		this.reset();
		Guardado=false;
		this.CapacidadVehiculos=DT.CapacidadVehiculos;
		this.NumeroClientes=DT.NumeroClientes;
		for(int i =0; i<=this.NumeroClientes; i++){
				this.Clientes[i][0]=DT.X(i);
				this.Clientes[i][1]=DT.Y(i);
				this.Clientes[i][2]=DT.D(i);
		}
}

public void dumpData(){
```

```
                        System.out.println("// *** Datos del problema *** //");
                        System.out.println("Capacidad de los vehiculos: "+CapacidadVehiculos );
                        System.out.println("Ubicacion deposito: ["+Clientes[0][0]+","+Clientes[0][1]+"]");
                        System.out.println("Total clientes: "+NumeroClientes );
                        for(int i=1; i<=NumeroClientes;i++){
                                System.out.println("Cliente "+i+": ["+Clientes[i][0]+","+Clientes[i][1]+"] Demanda:"+Clientes[i][2]);
                        }
                        System.out.println("Guardado :" + Guardado);
        }
}
```

clsClarkAndWright.java

Esta seccion del programa es la encargada de implementar el algoritmo de Clarke and wright para un problema dado que debe estar almacenado en la clase clsDatosProblema.

```
package Kernel;

import java.util.Date;
public class clsClarkAndWright{
        /**
         * Datos básicos de una ruta, esto es, puntos no interiores y demanda de la misma
         */
        class clsRuta extends Object{
                public clsRuta(double Demanda,int Ci,int Cd){
                        this.Demanda=Demanda;
                        this.Ci=Ci;
                        this.Cd=Cd;
                }
                public void Inicializar(){
                        this.Demanda=0.0;
                        this.Ci=0;
                        this.Cd=0;
                }
                double Demanda;
                int Ci, Cd;
        }
        /**
         * Constructor de la clase
         */
        public clsClarkAndWright(clsDatosProblema listaClientes, clsSolucion Sol) {
                super();
                this.C=listaClientes;
                this.S=Sol;
                this.M=new double[clsDatosProblema.MAXCLIENTES][clsDatosProblema.MAXCLIENTES];
```

```
                this.Ahorros=new double[(clsDatosProblema.MAXCLIENTES*clsDatosProblema.MAXCLIENTES-
clsDatosProblema.MAXCLIENTES)/2][3];
                this.CW=new int[(clsDatosProblema.MAXCLIENTES*2+1)];
                this.totCW=0;
                // TODO Auto-generated constructor stub
        }
        /**
         * QuickSort
         */
        private void sortV(double a[][], int lo, int hi){
    int h, l;
    double p;
    double t[];
    if (lo < hi) {
      l = lo;
      h = hi;
      p = a[hi][2];

      do {
        while ((l < h) && (a[l][2] >= p)) l++;
        while ((h > l) && (a[h][2] <= p)) h--;
        if (l < h) {
          t = a[l];
          a[l] = a[h];
          a[h] = t;
        }
      } while (l < h);

      t = a[l];
      a[l] = a[hi];
      a[hi] = t;

      sortV(a, lo, l-1);
      sortV(a, l+1, hi);
    }
        }
        /**
         * Adiciona un cliente al centro de una ruta nueva
         * @param Ci Cliente no interior izquierdo
         * @param Cd Cliente no interior derecho
         */
        private void AdicionarClientesCentrales(int Ci, int Cd){
                if(totCW==0){//En caso que sea el primero se debe de inicializar el vector
                        CW[totCW]=0;
```

```
                        totCW++;
                }
                CW[totCW]=Ci;//Esto debido a que cuando se van a agregar dos valores siempre se hace en una nueva ruta
                CW[totCW+1]=Cd;
                CW[totCW+2]=0;
                totCW+=3;
        }
        /**
         * Adiciona un cliente al extremo izquierdo de una ruta dada
         * @param Ci Cliente a ingresar
         * @param Ruta indice de la ruta donde se ingresará el cliente
         */
        private void AdicionarClienteIzquierdo(int Ci, int Ruta){
                int i,j;
                int rutasRecorridas;
                for(i=0,rutasRecorridas=0; i < totCW && rutasRecorridas<=Ruta;i++)       if(CW[i]==0) rutasRecorridas++;
                for(j=totCW-1; j>=i;j--)CW[j+1]=CW[j];
                CW[i]=Ci; totCW++;
        }
        /**
         * Adiciona un cliente al extremo derecho de una ruta dada
         * @param Cd Cliente a ingresar
         * @param Ruta indice de la ruta donde se ingresará el cliente
         */
        private void AdicionarClienteDerecho(int Cd, int Ruta){
                int i,j;
                int rutasRecorridas;
                for(i=0,rutasRecorridas=0; i < totCW && rutasRecorridas-1<=Ruta;i++)     if(CW[i]==0) rutasRecorridas++;
                for(j=totCW-1; j>=i-1;j--)CW[j+1]=CW[j];
                CW[i-1]=Cd; totCW++;
        }
        /**
         * Indica las variables donde inicia y termina una ruta
         * @param rutaMenor Indice de la primera ruta
         * @param rutaMayor Indice de la segunda ruta
         * @param RpC Indice (R)uta (p)rimera (C)omienzo
         * @param RpF Indice (R)uta (p)rimera (F)inal
         * @param RsC Indice (R)uta (s)egunda (C)omienzo
         * @param RsF Indice (R)uta (s)egunda (F)inal
         */
        private void posicionarPunteros(int rutaMenor, int rutaMayor, int [] posiciones){
                int i;
                int RutasRecorridas=0;
                boolean setRpC,setRpF,setRsC,setRsF;//Indican si el valor de las variables de posicionamiento ha sido asignado
```

```
                setRpC=false;
                setRpF=false;
                setRsC=false;
                setRsF=false;
                for(i=0,RutasRecorridas=0;i<totCW && !setRsF;i++){
                        if (CW[i]==0)RutasRecorridas++;
                        if(!setRpC&&(RutasRecorridas-1==rutaMenor)){
                                posiciones[0]=i;
                                setRpC=true;
                        }
                        if(!setRpF&&(RutasRecorridas-2==rutaMenor)){
                                posiciones[1]=i;
                                setRpF=true;
                        }
                        if(!setRsC&&(RutasRecorridas-1==rutaMayor)){
                                posiciones[2]=i;
                                setRsC=true;
                        }
                        if(!setRsF&&(RutasRecorridas-2==rutaMayor)){
                                posiciones[3]=i;
                                setRsF=true;
                        }
                }
        }
        /**
         * Fusiona dos rutas por los puntos izquierdos de la misma
         * @param rutaMenor indice de la ruta primera
         * @param rutaMayor indice de la ruta segunda y que desaparecerá
         * @param Clientes vector con las rutas a las que pertence cada cliente, el numero del cliente es la posición.
         */
        private void FusionarRutas(int TipoFusion, int rutaMenor, int rutaMayor, int[] Clientes){
                int Rpc, Rpf, Rsc, Rsf;
                int longitudSegundaRuta;
                int[] posiciones=new int[4];
                posicionarPunteros(rutaMenor,rutaMayor,posiciones);
                Rpc=posiciones[0];
                Rpf=posiciones[1];
                Rsc=posiciones[2];
                Rsf=posiciones[3];
                longitudSegundaRuta=Rsf-Rsc-1;
                int i,j;
                switch (TipoFusion){
                        case 1://"ZZ"
                                for(i=totCW-1;i>(Rpc);i--)  CW[i+longitudSegundaRuta]=CW[i];//Se mueven las posiciones para dar
```

```
cabida a la nueva ruta
                                        for(i=Rpc+1,j=Rsf+longitudSegundaRuta-1;j>Rsc+longitudSegundaRuta;i++,j--){
                                                CW[i]=CW[j];
                                                Clientes[CW[j]]=rutaMenor;
                                        }
                                        break;
                                case 2://"ZD"
                                        for(i=totCW-1;i>(Rpc);i--) CW[i+longitudSegundaRuta]=CW[i];//Se mueven las posiciones para dar
cabida a la nueva ruta
                                        for(i=Rpc+1,j=Rsc+longitudSegundaRuta+1;j<Rsf+longitudSegundaRuta;i++,j++){
                                                CW[i]=CW[j];
                                                Clientes[CW[j]]=rutaMenor;
                                        }
                                        break;
                                case 3://"DZ"
                                        for(i=totCW-1;i>=(Rpf);i--) CW[i+longitudSegundaRuta]=CW[i];//Se mueven las posiciones para dar
cabida a la nueva ruta
                                        for(i=Rpf,j=Rsc+longitudSegundaRuta+1;j<Rsf+longitudSegundaRuta;i++,j++){
                                                CW[i]=CW[j];
                                                Clientes[CW[j]]=rutaMenor;
                                        }
                                        break;
                                case 4://"ZD"
                                        for(i=totCW-1;i>=(Rpf);i--) CW[i+longitudSegundaRuta]=CW[i];//Se mueven las posiciones para dar
cabida a la nueva ruta
                                        for(i=Rpf,j=Rsf+longitudSegundaRuta-1;j>Rsc+longitudSegundaRuta;i++,j--){
                                                CW[i]=CW[j];
                                                Clientes[CW[j]]=rutaMenor;
                                        }
                                        break;
                        }

                        //Se ocupa el espacio de la ruta suprimida
                        for(i=Rsc+longitudSegundaRuta+1,j=Rsf+longitudSegundaRuta+1;j<totCW+longitudSegundaRuta;i++,j++){
                                CW[i]=CW[j];
                                Clientes[CW[j]]--;
                        }
                        totCW--;
                }
                /**
                 * Llena la informacion del vector solucion
                 */
                public void saveSolucion(){
                        /*for(i=0; i<TRutas;i++){
```

```
                System.out.println(" D1-C"+(Rutas[i].Ci+1)+"-C"+(Rutas[i].Cd+1)+"-D1");
                }*/
                S.reset();
                for(int i=0; i<totCW;i++) {
                        S.Solucion.put(CW[i]);
                        if (CW[i]>0)S.NDSolucion.put(CW[i]);
                }
                S.Costo=Costo;
                S.TiempoProceso = TiempoProceso;
        }
        /**
         * Algoritmo de Clark and Wright
         */
        public void Calcular(){
                        Date Data = new Date();
                        int i,j,totAhorros;
                        long Ti, Tf;
                        Ti = Data.getTime();
                        /**
                         * Cálculo de la matriz de distancias. diagonal inferior
                         * Calculo de la matriz de ahorro que estará en la diagonal superior y en el vector V
                         */
                        totAhorros=0;
                        for(i=0;i<=C.NumeroClientes();i++){
                                for(j=i;j<=C.NumeroClientes();j++){
                                        if(i==j){
                                                M[i][j]=0.0;
                                        }else{
                                                M[i][j]=Math.sqrt(Math.pow(C.X(i)-C.X(j),2)+Math.pow(C.Y(i)-C.Y(j),2));
                                                if (i==0){
                                                        M[j][i]=0;
                                                }else{
                                                        M[j][i]=M[0][i]+M[0][j]-M[i][j];
                                                        Ahorros[totAhorros][0]=i;
                                                        Ahorros[totAhorros][1]=j;
                                                        Ahorros[totAhorros][2]=M[j][i];
                                                        totAhorros++;
                                                }
                                        }
                                }
                        }
                        /**
                         * Sort the entire vector
                         */
```

```
                                sortV(Ahorros, 0, totAhorros - 1);
                                int Clientes[];
                                clsRuta Rutas[];
                                int TRutas=0;
                                Clientes = new int[C.NumeroClientes()+1];//Indica en que ruta está un cliente
                                Rutas = new clsRuta[C.NumeroClientes()];
                                for(i=0;i<C.NumeroClientes()+1;i++)Clientes[i]=-1;
                                for(i=0;i<totAhorros;i++){
                                        /**
                                         * Si la pareja de clientes no está en alguna ruta y la suma de sus demandas es menor que la
capacidad
                                         * de los vehiculos, una nueva ruta es creada
                                         */
                                        if((Clientes[(int)Ahorros[i][0]]==-1 && Clientes[(int)Ahorros[i][1]]==-
1)&&C.D((int)Ahorros[i][0])+C.D((int)Ahorros[i][1])<=C.CapacidadVehiculos()){
                                                Rutas[TRutas]=new
clsRuta(C.D((int)Ahorros[i][0])+C.D((int)Ahorros[i][1]),(int)Ahorros[i][0],(int)Ahorros[i][1]);
                                                Clientes[(int)Ahorros[i][0]]=TRutas;
                                                Clientes[(int)Ahorros[i][1]]=TRutas;
                                                AdicionarClientesCentrales((int)Ahorros[i][0],(int)Ahorros[i][1]);
                                                TRutas++;
                                                continue;
                                        }
                                        /**
                                         * Si el cliente izquierdo del vector de ahorros yá esta en alguna ruta y el cliente de derecho no,
                                         * y el cliente izquierdo es un punto no interior de la ruta a la que pertence
                                         * y la demanda de esa ruta más la demanda del cliente derecho no viola ninguna restriccion,
                                         * el cliente derecho es agregado a la ruta como punto no interior de ella y adyacente
                                         * al cliente izquierdo de la pareja
                                         */
                                        if((Clientes[(int)Ahorros[i][0]]!=-1 && Clientes[(int)Ahorros[i][1]]==-1)&&//Cliente izquierdo pertence a
una ruta y el derecho no

                (Rutas[Clientes[(int)Ahorros[i][0]]].Ci==(int)Ahorros[i][0]||Rutas[Clientes[(int)Ahorros[i][0]]].Cd==(int)Ahorros[i][0])&& //El punto izquierdo es
no interior

                (Rutas[Clientes[(int)Ahorros[i][0]]].Demanda+C.D((int)Ahorros[i][1])<=C.CapacidadVehiculos())){//Y la demanda de la ruta donde está el
cliente izquierdo más la demanda del cliente derecho no viola las restricciones
                                                Clientes[(int)Ahorros[i][1]]=Clientes[(int)Ahorros[i][0]];//Se le asigna a la ruta del cliente
derecho la ruta del cliente izquierdo, es decir, comparten la misma ruta
                                                Rutas[Clientes[(int)Ahorros[i][0]]].Demanda +=C.D((int)Ahorros[i][1]);//Esa ruta ahora
cuenta con la demanda adicional del cliente derecho
                                                if(Rutas[Clientes[(int)Ahorros[i][0]]].Ci==(int)Ahorros[i][0]){//Si el cliente izquierdo es
exterior izquierdo haga
```

```
                                                    Rutas[Clientes[(int)Ahorros[i][0]]].Ci = (int)Ahorros[i][1];//El nuevo punto exterior izquierdo de la ruta es el cliente derecho
                                                    AdicionarClienteIzquierdo((int)Ahorros[i][1],Clientes[(int)Ahorros[i][0]]); //Adiciono el cliente derecho a la punta izquierda de la ruta
                                                    continue;
                                                }else{//Si el cliente izquierdo es exterior derecho haga
                                                    Rutas[Clientes[(int)Ahorros[i][0]]].Cd = (int)Ahorros[i][1];//El nuevo punto exterior derecho de la ruta es el cliente derecho

            AdicionarClienteDerecho((int)Ahorros[i][1],Clientes[(int)Ahorros[i][0]]);//Adiciono el cliente derecho a la punta derecha de la ruta
                                                    continue;
                                                }
                                            }
                                            /**
                                             * Si el cliente izquierdo del vector de ahorros no está en ninguna ruta y el cliente de derecho si lo está,
                                             * y el cliente derecho es un punto no interior de la ruta a la que pertence
                                             * y la demanda de esa ruta más la demanda del cliente izquierdo no viola ninguna restriccion,
                                             * el cliente izquierdo es agregado a la ruta como punto no interior de ella y adyacente
                                             * al cliente derecho de la pareja
                                             */
                                            if((Clientes[(int)Ahorros[i][0]]==-1 && Clientes[(int)Ahorros[i][1]]!=-1)&&//Cliente izquierdo no tiene ruta y el derecho si

            (Rutas[Clientes[(int)Ahorros[i][1]]].Ci==(int)Ahorros[i][1]||Rutas[Clientes[(int)Ahorros[i][1]]].Cd==(int)Ahorros[i][1])&& //El punto derecho es no interior

            (Rutas[Clientes[(int)Ahorros[i][1]]].Demanda+C.D((int)Ahorros[i][0])<=C.CapacidadVehiculos())){//Y la demanda de la ruta donde está el cliente derecho más la demanda del cliente izquierdo no viola las restricciones
                                                Clientes[(int)Ahorros[i][0]]=Clientes[(int)Ahorros[i][1]];//Se le asigna a la ruta del cliente izquierdo la ruta del cliente derecho, es decir, comparten la misma ruta
                                                Rutas[Clientes[(int)Ahorros[i][1]]].Demanda +=C.D((int)Ahorros[i][0]);//Esa ruta ahora cuenta con la demanda adicional del cliente izquierdo
                                                if(Rutas[Clientes[(int)Ahorros[i][1]]].Ci==(int)Ahorros[i][1]){//Si el cliente derecho es exterior izquierdo haga
                                                    Rutas[Clientes[(int)Ahorros[i][1]]].Ci = (int)Ahorros[i][0];//El nuevo punto exterior izquierdo de la ruta es el cliente izquierdo
                                                    AdicionarClienteIzquierdo((int)Ahorros[i][0],Clientes[(int)Ahorros[i][1]]); //Adiciono el cliente izquierdo a la punta izquierda de la ruta
                                                    continue;
                                                }else{//Si el cliente derecho es exterior izquierdo haga
                                                    Rutas[Clientes[(int)Ahorros[i][1]]].Cd = (int)Ahorros[i][0];//El nuevo punto exterior derecho de la ruta es el cliente izquierdo
```

```
AdicionarClienteDerecho((int)Ahorros[i][0],Clientes[(int)Ahorros[i][1]]);//Adiciono el cliente izquierdo a la punta derecha de la ruta
                                                continue;
                                        }
                                }
                                /**
                                 * Si los clientes ya pertenecen a sendas ruta y son puntos no interiores de las mismas y la suma
                                 * de esas rutas no violan ninguna restrición, las dos rutas son fusionadas en una.
                                 */
                                if((Clientes[(int)Ahorros[i][0]]!=-1 && Clientes[(int)Ahorros[i][1]]!=-1)&&//Ambos clientes estan en alguna ruta
                                                (Clientes[(int)Ahorros[i][0]]!= Clientes[(int)Ahorros[i][1]])&& //La ruta a la que pertenecen los clientes no es la misma

        (Rutas[Clientes[(int)Ahorros[i][0]]].Ci==(int)Ahorros[i][0]||Rutas[Clientes[(int)Ahorros[i][0]]].Cd==(int)Ahorros[i][0])&& //El punto izquierdo es no interior

        (Rutas[Clientes[(int)Ahorros[i][1]]].Ci==(int)Ahorros[i][1]||Rutas[Clientes[(int)Ahorros[i][1]]].Cd==(int)Ahorros[i][1])&& //El punto derecho es no interior

        (Rutas[Clientes[(int)Ahorros[i][0]]].Demanda+Rutas[Clientes[(int)Ahorros[i][1]]].Demanda<=C.CapacidadVehiculos())){
                                                int rutaMenor=0;//Almacenará la ruta menor en indice
                                                int rutaMayor=0;//Almacenará la ruta mayor en indice
                                                if(Clientes[(int)Ahorros[i][0]] < Clientes[(int)Ahorros[i][1]]){
                                                        rutaMenor=Clientes[(int)Ahorros[i][0]];
                                                        rutaMayor=Clientes[(int)Ahorros[i][1]];
                                                }else{
                                                        rutaMenor=Clientes[(int)Ahorros[i][1]];
                                                        rutaMayor=Clientes[(int)Ahorros[i][0]];
                                                }
                                                Rutas[rutaMenor].Demanda+=Rutas[rutaMayor].Demanda;//La demanda de la primera ruta se incrementa con el valor de la demanda de la ruta con la que se fusionará
                                                //ZZ
                                                if((Rutas[rutaMenor].Ci==(int)Ahorros[i][0] && Rutas[rutaMayor].Ci==(int)Ahorros[i][1])||(Rutas[rutaMenor].Ci==(int)Ahorros[i][1] && Rutas[rutaMayor].Ci==(int)Ahorros[i][0])){//Ambos clientes son izquierdos
                                                        Rutas[rutaMenor].Ci =Rutas[rutaMayor].Cd;
                                                        FusionarRutas(1,rutaMenor,rutaMayor,Clientes);
                                                }else{//ZD
                                                        if((Rutas[rutaMenor].Ci==(int)Ahorros[i][0] && Rutas[rutaMayor].Cd==(int)Ahorros[i][1])||(Rutas[rutaMenor].Ci==(int)Ahorros[i][1] && Rutas[rutaMayor].Cd==(int)Ahorros[i][0])){//Un cliente es izquierdo en la primera ruta y el otro es derecho en la segunda ruta
                                                                Rutas[rutaMenor].Ci =Rutas[rutaMayor].Ci;
                                                                FusionarRutas(2,rutaMenor,rutaMayor,Clientes);
                                                        }else{//DZ
```

```
                                                    if((Rutas[rutaMenor].Cd==(int)Ahorros[i][0] &&
Rutas[rutaMayor].Ci==(int)Ahorros[i][1])||(Rutas[rutaMenor].Cd==(int)Ahorros[i][1] && Rutas[rutaMayor].Ci==(int)Ahorros[i][0])){//Un cliente es derecho
en la primera ruta y el otro es izquierdo en la segunda ruta
                                                            Rutas[rutaMenor].Cd =Rutas[rutaMayor].Cd;
                                                            FusionarRutas(3,rutaMenor,rutaMayor,Clientes);
                                                    }else{//DD
                                                            if((Rutas[rutaMenor].Cd==(int)Ahorros[i][0] &&
Rutas[rutaMayor].Cd==(int)Ahorros[i][1])||(Rutas[rutaMenor].Cd==(int)Ahorros[i][1] && Rutas[rutaMayor].Cd==(int)Ahorros[i][0])){//Ambos clientes son
derechos
                                                                    Rutas[rutaMenor].Cd
=Rutas[rutaMayor].Ci;

            FusionarRutas(4,rutaMenor,rutaMayor,Clientes);
                                                            }
                                                    }
                                            }
                                    }
                                    for(int ind=rutaMayor; ind<TRutas-1;ind++){
                                            Rutas[ind].Ci =Rutas[ind+1].Ci;
                                            Rutas[ind].Cd =Rutas[ind+1].Cd;
                                            Rutas[ind].Demanda =Rutas[ind+1].Demanda;
                                    }
                                    Rutas[TRutas-1]=null;
                                    TRutas--;
                            }
                    }
                    /**
                     * Todos los clientes que por alguna razon no tienen ruta son incluidos en la solución con una ruta propia
                     */
                    for(i=1;i<=C.NumeroClientes();i++){
                            if(Clientes[i]==-1){
                                    CW[totCW++]=i;
                                    CW[totCW++]=0;
                            }
                    }
                    /**
                     * Busco el valor de la solucion
                     */
                    Costo = 0;
                    for(i=0;i<totCW;i++){
                            if(CW[i]<CW[i+1]){
                                    Costo+=M[CW[i]][CW[i+1]];
                            }else{
                                    Costo+=M[CW[i+1]][CW[i]];
```

```
                                        }
                                }
                                Data = new Date();
                                Tf = Data.getTime();
                                TiempoProceso = Tf-Ti;
                                this.saveSolucion();
        }
        double M[][];
        double Ahorros[][];//Ahorros. ir del cliente que esta en V[0][i] hasta el cliente que esta en V[1][i] tiene un ahorro de V[2][i]
        int CW[];
        int totCW;
        double Costo;
        long TiempoProceso;
        clsDatosProblema C;
        clsSolucion S;

}
```

clsSolucion.java

En esta clase es almacenada la información de los resultados obtenido por los algoritmos de solución.

```
package Kernel;

public class clsSolucion {
        public clsCVRPLista NDSolucion;//Solucion sin depositos, es con la que se trabaja
        public clsCVRPLista Solucion;//Solucion con depositos, es la que se muestra
        public double Costo;//Costo de la solucion
        public long TiempoProceso;//Tiempo de esta solucion
        public boolean blnTieneSolucion(){
                return (Solucion.totLista > 0);
        }
        public clsSolucion(){
                this.reset();
        }
        public void construirDesdeNDSolucion(clsDatosProblema DatosProblema, double MatrizDistancias[][]){//Construye todo el objeto apartir de
la solucion sin depositos
//              Con el vector de solucion sin depositos, cargo el vector con depósitos a apartir de él.
//              La solución con depositos requiere iniciar con uno y terminar con otro
                double DemandaAcumulada=0.0;
                this.Solucion.Lista[this.Solucion.totLista++]=0;
                for(int i=0;i<this.NDSolucion.totLista;i++){
                        if(DatosProblema.D(this.NDSolucion.Lista[i])+DemandaAcumulada <= DatosProblema.CapacidadVehiculos()){
                                DemandaAcumulada += DatosProblema.D(this.NDSolucion.Lista[i]);
```

```
                                        this.Solucion.Lista[this.Solucion.totLista++]=this.NDSolucion.Lista[i];
                                }else{
                                        DemandaAcumulada = 0;
                                        this.Solucion.Lista[this.Solucion.totLista++]=0;
                                        DemandaAcumulada += DatosProblema.D(this.NDSolucion.Lista[i]);
                                        this.Solucion.Lista[this.Solucion.totLista++]=this.NDSolucion.Lista[i];
                                }
                        }
                        this.Solucion.Lista[this.Solucion.totLista++]=0;
//                      Con el vector solucion con depositos, calculos los costos de la operacion
                        for(int i=1;i<this.Solucion.totLista;i++){
                                this.Costo +=MatrizDistancias[this.Solucion.Lista[i-1]][this.Solucion.Lista[i]];
                        }
                }
                public void print(boolean imprimirDepositos){//Imprime la solucion por defecto
                        System.out.println("Composicion de la ruta: ");
                        if(imprimirDepositos)Solucion.print();else NDSolucion.print();
                        System.out.println("Solucion: "+Costo);
                        System.out.println("Tiempo de proceso: "+TiempoProceso+"ms");
                }
                public void reset(){
                        this.NDSolucion = new clsCVRPLista((clsDatosProblema.MAXCLIENTES*2+1));
                        this.Solucion = new clsCVRPLista((clsDatosProblema.MAXCLIENTES*2+1));
                        Costo = 0;
                        TiempoProceso = 0;
                }
                public void Assing(clsSolucion value){
                        NDSolucion.Assign(value.NDSolucion);
                        Solucion.Assign(value.Solucion);
                        Costo=value.Costo;
                        TiempoProceso=value.TiempoProceso;
                }

                public class clsCVRPLista{
                        public int Lista[];
                        public int totLista;
                        private int MAXLISTA;
                        public clsCVRPLista(int value){
                                this.MAXLISTA = value;
                                this.reset();
                        }
                        public void reset(){//Iniciliza la lista
                                this.Lista=new int[this.MAXLISTA];
                                this.totLista=0;
```

```
                }
                public void put(int value){
                        this.Lista[this.totLista++] = value;
                }
                public int get(int position){
                        return this.Lista[position];
                }
                public void print(){
                        for(int i=0;i<this.totLista;i++) System.out.println(this.Lista[i]+1);
                }
                public void Assign(clsCVRPLista value){
                        Lista = (int[])value.Lista.clone();
                        totLista = value.totLista;
                        MAXLISTA = value.MAXLISTA;
                }
        }
}
```

clsTabuSearch.java

Esta clase implementa la busqueda tabú a partir de la información que se encuentre en los datos del problema y con la solucion inicial dada.

```
package Kernel;

import java.util.Date;

public class clsTabuSearch {
        public clsTabuSearch(clsDatosProblema Datos, clsSolucion SolucionInicial, int nroIteraciones, int nroCambiosMaximo, int tamanoLista, int tamanoPoblacion, int nroIntensificacion, int nroDiversificacion){
                this.Datos = new clsDatosProblema();
                this.Datos.Assign(Datos);
                this.SolucionInicial = SolucionInicial;
                this.nroIteraciones = nroIteraciones;
                this.nroCambioMaximo = nroCambiosMaximo;
                this.tamanoLista = tamanoLista;
                this.tamanoPoblacion = tamanoPoblacion;
                this.nroIntensificacion = nroIntensificacion;
                this.nroDiversificacion = nroDiversificacion;
                this.M=new double[clsDatosProblema.MAXCLIENTES][clsDatosProblema.MAXCLIENTES];
        }

        /**
         * Cálculo de la matriz de distancias. diagonal inferior
```

```
 */
private void BuscarMatrizDistancias(){
        int i,j;
        for(i=0;i<=Datos.NumeroClientes();i++){
                for(j=i;j<=Datos.NumeroClientes();j++){
                        if(i==j){
                                M[i][j]=0.0;
                        }else{
                                M[i][j]=Math.sqrt(Math.pow(Datos.X(i)-Datos.X(j),2)+Math.pow(Datos.Y(i)-Datos.Y(j),2));
                                M[j][i]=M[i][j];
                        }
                }
        }
}
/**
 * Toma los arcos eliminados de una solucion y los marca como tabú para que se les impida
 * ser adicionados hasta que no haya transcurrido "tamanoLista" iteraciones
 * @param Vx :Arcos a ser eliminados
 * @param totVx : Cantidad de arcos a eliminar
 */
private void actualizaTabu(int Vx[][], int totVx){
        for(int i=0;i<Datos.NumeroClientes();i++)
                for(int j=0;j<Datos.NumeroClientes();j++)

MovimientosTabu[i][j]=(MovimientosTabu[i][j]>tamanoLista||MovimientosTabu[i][j]==0)?0:MovimientosTabu[i][j]+1;
        for(int i=0;i<totVx;i++){
                MovimientosTabu[Vx[i][0]][Vx[i][1]]=1;
                MovimientosTabu[Vx[i][1]][Vx[i][0]]=1;
        }
}
/**
 * Valida que los arcos a adicionar no esten marcados como tabu en tal caso
 * se retorno un false
 * @param Vy : Arcos a adicionar
 * @param totVy: Total arcos adicionar
 */
private boolean validaTabu(int Vy[][], int totVy){
        for(int i=0;i<totVy;i++)
                if(MovimientosTabu[Vy[i][0]][Vy[i][1]]>0)
                        return false;
        return true;
}
/**
 * Obtinene un objeto del tipo solucion apartir de los arcos que serán removidos, y de la solucion anterior
```

```
     * si la ruta que puede obtener no es cerrada, devuelve null
     * @param SolucionAnterior Solucion anterior
     * @param Vx Arcos a ser eliminados con la nueva solucion
     * @return Objeto con la nueva solucion propuesta o null si la ruta no es cerrada
     */
    private clsSolucion obtenerSolucion(clsSolucion SolucionAnterior, int Vx[][], int nroArcosTratados, int nroCambiosMaximo){
            clsSolucion Retorno = new clsSolucion();
            double DemandaAcumulada;
            boolean Encontrado,validacionTabu;
            int Vy[][] = new int[nroCambiosMaximo][3];
            int VArcos[][] = new int[Datos.NumeroClientes()][3];//Vector de arcos
            //Se construye el vector Vy con los arcos a adicionar, a partir de los Arcos a eliminar Vx
            for(int i=0;i<nroCambiosMaximo;i++){//Construyo los vectores Vy
                    Vy[i][0] = Vx[i][1];
                    Vy[i][1] = (i+1<nroCambiosMaximo)?Vx[i+1][0]:Vx[0][0];
                    Vy[i][2] = 0;
            }
            //Se valida si los arcos a adicionar ya estan marcados como tabu
            validacionTabu=validaTabu(Vy,nroArcosTratados);
            //si no pasa la validacion tabu la ruta obtenida no sirve
            if(!validacionTabu) return null;
            //Se construye un vector con todos los arcos iniciales del problema, si el arco esta marca
            //para eliminacion en el vector Vx, se marca posicion 2 del vector para hacerla invisible
            //para el proceso
            for(int i=0;i<Datos.NumeroClientes();i++){
                    VArcos[i][0]=i;
                    VArcos[i][1]=(i+1<Datos.NumeroClientes())?i+1:0;
                    VArcos[i][2]=0;
                    Encontrado=false;
                    //Se marca el arco como eliminado si esta en el vector Vx
                    for(int j=0;j<nroCambiosMaximo&&!Encontrado;j++){
                            if((VArcos[i][0]==Vx[j][0])&&(VArcos[i][1]==Vx[j][1])||(VArcos[i][0]==Vx[j][1])&&(VArcos[i][1]==Vx[j][0])){
                                    VArcos[i][2]=1;
                                    Encontrado=true;
                            }
                    }
            }
//          Se construye la nueva ruta sin depositos con base en el vector de arcos y los arcos a adicionar Vy
//          pare ello busco cada una de las posiciones en el vector de arcos indiferente de la posicion
//          Si no lo encuentro en el vector de arcos lo busco en el vector Vy. Mi nueva posicion siguiente
//          a buscar será la pareja del dato que busqué. Se repite el proceso tantas veces como
//          clientes tenga.
//          Se debe recordar que la ruta que se tiene corresponde a posiciones dentro de la solucion
            int Valor = 0;
```

```
                int Contador = 0;
                for(int i=0;i<Datos.NumeroClientes();i++){//Proceso se repite NClientes veces hasta incluir todos los clientes en la ruta
                        Encontrado=false;
                        for(int j=0;j<Datos.NumeroClientes()&&!Encontrado;j++){//Busco primero en el vector de arcos
                                if(VArcos[j][2]!=1){//Si el arco no ha sido marcado como usado busque esa posicion
                                        //Si encuentra el valor, se incluye en la solucion, el nuevo numero a buscar será la pareja
del actual, marco el arco como encontrado
                                        if(VArcos[j][0]==Valor){

        Retorno.NDSolucion.Lista[Retorno.NDSolucion.totLista++]=SolucionAnterior.NDSolucion.Lista[Valor];
                                                Valor = VArcos[j][1];
                                                Encontrado=true;
                                                VArcos[j][2]=1;
                                                Contador ++;
                                        }else{
                                                if(VArcos[j][1]==Valor){

        Retorno.NDSolucion.Lista[Retorno.NDSolucion.totLista++]=SolucionAnterior.NDSolucion.Lista[Valor];
                                                        Valor = VArcos[j][0];
                                                        Encontrado=true;
                                                        VArcos[j][2]=1;
                                                        Contador ++;
                                                }
                                        }

                                }
                        }
//                      Si no hallé el valor en el vector de arcos lo busco en el vector de arcos a adicionar Vy
                        for(int j=0;j<=nroArcosTratados&&!Encontrado;j++){
                                if(Vy[j][2]!=1){//Si la posicion del vector Vx no ha sido marcada como usada busque esa posicion
//                                      Si encuentra el valor, se incluye en la solucion, el nuevo numero a buscar será la pareja
del actual, marco el arco como encontrado
                                        if(Vy[j][0]==Valor){

        Retorno.NDSolucion.Lista[Retorno.NDSolucion.totLista++]=SolucionAnterior.NDSolucion.Lista[Valor];
                                                Valor = Vy[j][1];
                                                Encontrado=true;
                                                Vy[j][2]=1;
                                                Contador ++;
                                        }else{
                                                if(Vy[j][1]==Valor){

        Retorno.NDSolucion.Lista[Retorno.NDSolucion.totLista++]=SolucionAnterior.NDSolucion.Lista[Valor];
                                                        Valor = Vy[j][0];
```

```
                                                                        Encontrado=true;
                                                                        Vy[j][2]=1;
                                                                        Contador ++;
                                                                }
                                                        }

                                                }
                                        }
                                }
                                //Si el contador de puntos encontrados es menor que el numero de clientes ingresados es por que
                                //la ruta no es cerrada
                                if (Contador<Datos.NumeroClientes()){
                                        return null;
                                }
//                              Con el vector de solucion sin depositos, cargo el vector con depósitos a apartir de él.
//                              La solución con depositos requiere iniciar con uno y terminar con otro
                                Retorno.Solucion.Lista[Retorno.Solucion.totLista++]=0;
                                DemandaAcumulada=0.0;
                                for(int i=0;i<Retorno.NDSolucion.totLista;i++){
                                        if(Datos.D(Retorno.NDSolucion.Lista[i])+DemandaAcumulada <= Datos.CapacidadVehiculos()){
                                                DemandaAcumulada += Datos.D(Retorno.NDSolucion.Lista[i]);
                                                Retorno.Solucion.Lista[Retorno.Solucion.totLista++]=Retorno.NDSolucion.Lista[i];
                                        }else{
                                                DemandaAcumulada = 0;
                                                Retorno.Solucion.Lista[Retorno.Solucion.totLista++]=0;
                                                DemandaAcumulada += Datos.D(Retorno.NDSolucion.Lista[i]);
                                                Retorno.Solucion.Lista[Retorno.Solucion.totLista++]=Retorno.NDSolucion.Lista[i];
                                        }
                                }
                                Retorno.Solucion.Lista[Retorno.Solucion.totLista++]=0;
//                              Con el vector solucion con depositos, calculos los costos de la operacion
                                for(int i=1;i<Retorno.Solucion.totLista;i++){
                                        Retorno.Costo +=M[Retorno.Solucion.Lista[i-1]][Retorno.Solucion.Lista[i]];
                                }
                                //si no pasa la validacion tabu la ruta obtenida no sirve
                                if(!validacionTabu) return null;
                                return Retorno;
                        }
                        private boolean ValidarRuta(clsSolucion Solucion, int Vx[][],int PosicionActual,int nroCambiosMaximo){
                                //Si solo se ha eliminado un arco el arco a adicionar solo podria se su inverso
                                //por tanto la solucion es trivial
                                if (PosicionActual==0) return true;
//                              Contruyo apartir de los arcos ha ser eliminados una nueva solucion con la que validaré
//                              el costo contra el de la solucion anterior y además si la ruta obtenida es cerrada
```

```
                clsSolucion SolucionPreliminar = obtenerSolucion(Solucion,Vx,PosicionActual,nroCambiosMaximo);
                if (SolucionPreliminar==null){
                        return false;//La ruta no es cerrada es tabu
                }
                VectorConcetracion.Adicionar(SolucionPreliminar);
//              Se agrega la solucion en el vector de soluciones
                return (SolucionPreliminar.Costo < Solucion.Costo);
        }
        /**
         * Esta funcion se encarga de Buscar un valor dado dentro del vector de posiciones enconrtado
         * @param Vx
         * @param Valor; valor a buscar en el vector Vx
         * @param Posicion; Tamaño actual de la lista
         * @return true si lo encuentra false si no lo encuentra
         */
        private boolean YaUsado(int Vx[][], int Valor, int PosicionActual){
                boolean Encontrado = false;
                for(int j=0;j<PosicionActual;j++){
                        if(Valor==Vx[j][0]||Valor==Vx[j][1])Encontrado=true;
                }
                return Encontrado;
        }

        private void LinKernighan(clsSolucion Solucion,int nroCambiosMaximoIniciales){
                int l,i;
                int Vx[][] = new int[nroCambiosMaximoIniciales][2];
                int MemoriaCPlazo[][]= new int[nroCambiosMaximoIniciales*2][Datos.NumeroClientes()+1];
                boolean Encontrado, Error;
                int nroCambiosMaximo = nroCambiosMaximoIniciales;
                l=0;
                while(l<nroIteraciones&&nroCambiosMaximo>1){
                        i=0;
                        while(i<nroCambiosMaximo*2&&nroCambiosMaximo>1){
                                //Seleccionar t1 como un punto al azar en la solucion dada que no haya sido usado previamente.
                                Encontrado = false;
                                if(i % 2 == 0){
                                        Error = false;
                                        while(!Encontrado&&!Error){
                                                Vx[i/2][0]=(int)(Math.random()*Datos.NumeroClientes());
                                                Encontrado=true;
                                                for(int j=1;j<=MemoriaCPlazo[i][0]&&Encontrado;j++){//Si el t encontrado ya
esta en la memoria de corto plazo para la posicion dada entonce no he encontrado el el t
                                                        if(Vx[i/2][0]==MemoriaCPlazo[i][j])Encontrado=false;
                                                }
```

```
                                        if (Encontrado){
                                                //Si el punto no esta en la memoria de corto plazo,
                                                //lo marca como encontrado de tal tipo que si ya ha
                                                //sido usado la siguiente vez lo verá como invalido
                                                MemoriaCPlazo[i][0]++;//Adicionar el punto hallado a la memoria
de corto plazo
                                                MemoriaCPlazo[i][MemoriaCPlazo[i][0]]=Vx[i/2][0];
                                                Encontrado = !YaUsado(Vx,Vx[i/2][0],i/2)&&Encontrado;//El
nuevo t no debe ser un valor elegido anteriormente
                                                if(Encontrado){
                                                        i++;
                                                }
                                        }
                                        //Si la iteracion no fue capaz de encontrar un t2i-1 debe buscarse
                                        //un t2(i-i)-1 diferente, por tanto se inicializan las memorias t2-1 y t2(i-i)
                                        if(!Encontrado&&(MemoriaCPlazo[i][0]>=Datos.NumeroClientes())){
                                                if(i==0){
                                                        System.out.println("Finalizacion Lin Kernighan");
                                                }
                                                MemoriaCPlazo[i][0]=0;
                                                if(i>1)         MemoriaCPlazo[i-1][0]=0;
                                                i=i-2;
                                                if(i<0){
                                                        i=0;
                                                        nroCambiosMaximo--;
                                                }
                                                Error=true;
                                        }
                                }
                        }else{
                                //Busco t2 como un punto adyacente a t1 inicion con el derecho
                                Vx[i/2][1]=Vx[i/2][0]+1;
                                if(Vx[i/2][1]>Datos.NumeroClientes()-1) Vx[i/2][1]=0;
                                MemoriaCPlazo[i][0]=0;//Cuando encuentra un t2i-1 se inicializa la memora t2i
                                MemoriaCPlazo[i][0]++;//Adicionar el punto hallado a la memoria de corto plazo
                                MemoriaCPlazo[i][MemoriaCPlazo[i][0]]=Vx[i/2][1];
                                if(YaUsado(Vx,Vx[i/2][1],i/2)||!ValidarRuta(Solucion,Vx,i/2,nroCambiosMaximo)){//Si la ruta
hallada no es valida o si el punto seleccionado ya ha sido usado, debe intentarse el otro punto, si no contua el proceso
                                        Vx[i/2][1]=Vx[i/2][0]-1;
                                        if(Vx[i/2][1]<0) Vx[i/2][1]=Datos.NumeroClientes()-1;
                                        MemoriaCPlazo[i][0]++;//Adicionar el punto hallado a la memoria de corto
plazo
                                        MemoriaCPlazo[i][MemoriaCPlazo[i][0]]=Vx[i/2][1];
```

```
            if(YaUsado(Vx,Vx[i/2][1],i/2)||!ValidarRuta(Solucion,Vx,i/2,nroCambiosMaximo)){//Si al intentar con el otro punto la respuesta no es valida, se disminuye un paso para que se escoja otro punto inicial
                                                                    MemoriaCPlazo[i][0]=0;//Se reinicia esta posicion por que si ninguno de los extremos sirve debe encontrarse otro t2i-1
                                                                    i--;
                                                            }else{
                                                                    i++;
                                                            }
                                                    }else{
                                                            i++;
                                                    }
                                            }
                                    }
                                    //nroCambios llego a ser 1 es por que no se pudo hallar una solucion mejor del TSP
                                    //en este punto la funcion no no debe continuar y el resultado será la solucion inicial
                                    if(nroCambiosMaximo==1){
                                            break;
                                    }
                                    //Se instancia un nuevo objeto solucion.
                                    clsSolucion SolucionEncontrada = obtenerSolucion(Solucion,Vx,nroCambiosMaximo,nroCambiosMaximo);
                                    if(Solucion.Costo > SolucionEncontrada.Costo){
                                            //System.out.println("Solucion Encontrada en: " + l + " Costo: " + SolucionEncontrada.Costo);
                                            actualizaTabu(Vx,nroCambiosMaximo);
                                            Solucion.Assing(SolucionEncontrada);
                                            //SolucionEncontrada.print(true);
                                            SolucionEncontrada = new clsSolucion();
                                    }
                                    //Inicializo variables de LinKernighan
                                    nroCambiosMaximo=nroCambiosMaximoIniciales;
                                    MemoriaCPlazo= new int[nroCambiosMaximoIniciales*2][Datos.NumeroClientes()+1];
                                    l++;
                            }
                    }
                    /**
                     * Crea una nueva solucion aleatoria o secuencial dependiedo del grado de verdad
                     * del parametro Aleatorio
                     * @param Aleatorio: true Solucion se construye con informacion aleatoria, false se contruye solucion secuencial
                     * @return
                     */
                    private clsSolucion GenerarSolucion(boolean Aleatorio){
                            clsSolucion retorno;
                            retorno = new clsSolucion();
                            if(Aleatorio){
                                    for(int i=1;i<=Datos.NumeroClientes();i++){
```

```
                                        boolean Encontrado=true;
                                        int Valor=0;
                                        while(Encontrado){
                                                Valor=(int)(Math.random()*Datos.NumeroClientes()+1);
                                                Encontrado=false;
                                                for(int j=0;j<retorno.NDSolucion.totLista&&!Encontrado;j++){
                                                        if(Valor==retorno.NDSolucion.Lista[j]) Encontrado = true;
                                                }
                                        }
                                        retorno.NDSolucion.Lista[retorno.NDSolucion.totLista++]=Valor;
                                }
                        }else{
                                for(int i=1;i<=Datos.NumeroClientes();i++){
                                        retorno.NDSolucion.Lista[retorno.NDSolucion.totLista++]=i;
                                }
                        }
                        retorno.construirDesdeNDSolucion(Datos,M);
                        return retorno;
                }
                /**
                 * Ejecuta una busqueda tabu para obtener una buena solucion del problema.
                 */
                public void Calcular(){
                        Date Data = new Date();
                        long Ti, Tf;
//                      Declaracion de variables
                        Ti = Data.getTime();
                        VectorConcetracion = new clsVectorConcentracion(tamanoPoblacion);
                        MovimientosTabu = new int[Datos.NumeroClientes()][Datos.NumeroClientes()];
                        if (this.nroCambioMaximo>Datos.NumeroClientes()/2){
                                this.nroCambioMaximo = (Datos.NumeroClientes()/2)-1;
                        }
                        clsSolucion Solucion = new clsSolucion();
                        Solucion.Assing(this.SolucionInicial);
                        this.BuscarMatrizDistancias();
                        Data = new Date();
                        Tf = Data.getTime();
//                      Apartir de la solucion ClarkAndWright
                        LinKernighan(Solucion,this.nroCambioMaximo);
//                      Insencifico las mejores "nroIntensificacion" soluciones
                        for(int i=0;i<=this.nroIntensificacion&&i<VectorConcetracion.totSoluciones;i++){
                                LinKernighan(VectorConcetracion.Registro[i],this.nroCambioMaximo);
                        }
//                      Diversificacion aleatoria "nroDiversificacion" veces
```

```
                        for(int i=0;i<this.nroDiversificacion;i++)
                                LinKernighan(GenerarSolucion(true),this.nroCambioMaximo);
//                      diversificacion secuencial
                        LinKernighan(GenerarSolucion(false),this.nroCambioMaximo);
//                      Insencifico las mejores "nroIntensificacion" soluciones nuevas
                        for(int  i=0;i<=this.nroIntensificacion&&i<VectorConcetracion.totSoluciones;i++){//Insencifico las mejores "nroIntensificacion"
soluciones
                                LinKernighan(VectorConcetracion.Registro[i],this.nroCambioMaximo);
                        }
//                      Cargo la mejor solucion a la salida
                        this.SolucionInicial.Assing(VectorConcetracion.Registro[0]);//Cargo la mejor solucion a la salida
//                      Imprimo el vector de las mejores soluciones
                        VectorConcetracion.print();
                        this.SolucionInicial.TiempoProceso = Tf-Ti;
        }

        private clsDatosProblema Datos;
        private clsSolucion SolucionInicial;
        private int nroIteraciones;
        private int nroCambioMaximo;
        private int tamanoLista;
        private int tamanoPoblacion;
        private int nroIntensificacion;
        private int nroDiversificacion;
        private double M[][];
        private int MovimientosTabu[][];
        private clsVectorConcentracion VectorConcetracion;
};
```

clsVectorConcentracion.java

Esta clase es la estructura utilizada como vector de concentración por la busqueda tabú.

```
package Kernel;

public class clsVectorConcentracion {
        public clsSolucion Registro[];
        int totSoluciones;
        int tamanoPoblacion;
        public clsVectorConcentracion(int tamanoPoblacion){
                totSoluciones=0;
                this.tamanoPoblacion = tamanoPoblacion;
                Registro = new clsSolucion[this.tamanoPoblacion];
                for(int i=0;i<this.tamanoPoblacion;i++){
```

```
                                        Registro[i]=new clsSolucion();
                        }
        }
        public void print(){
                        System.out.println("//---------->");
                        for(int i=0;i<totSoluciones;i++){
                                        System.out.println(i + " -- " + Registro[i].Costo);
                        }
        };
        private boolean BuscarXCosto(double Costo){
                        boolean retorno=false;
                        //long C1, C2;
                        //C1 = (long)(Costo*1000000000000.0);
                        for(int i=0;i<totSoluciones-1&&!retorno;i++){
                                        //C2 = (long)(Registro[i].Costo*1000000000000.0);
                                        //if((int)(Registro[i].Costo*1000000000) == (int)(Costo*1000000000)){
                                        if(Registro[i].Costo== Costo){
                                        //if(C1==C2){
                                                        retorno=true;
                                        }
                        }
                        return retorno;
        };
        private void Ordenar(int lo, int hi){
int h, l;
double p;
clsSolucion t;
if (lo < hi) {
   l = lo;
   h = hi;
   p = Registro[hi].Costo;

   do {
      while ((l < h) && (Registro[l].Costo <= p)) l++;
      while ((h > l) && (Registro[h].Costo >= p)) h--;
      if (l < h) {
         t = Registro[l];
         Registro[l] = Registro[h];
         Registro[h] = t;
      }
   } while (l < h);
   t = Registro[l];
   Registro[l] = Registro[hi];
   Registro[hi] = t;
```

```
        Ordenar(lo, l-1);
        Ordenar(l+1, hi);
    }
        }
//                  Validar si ya esta completa la lista en cuyo caso se debe
//                  validar si el costo es mejor que el de la peor solucion
//                  Validar si ya existe una solucion con el mismo costo
//                  Adicionar el registro
//                  Ordenar la lista nuevamente
        public void Adicionar(clsSolucion Registro){
                //Registro se adiciona si y solo si no hay otro con el mismo costo
                if (!BuscarXCosto(Registro.Costo)){
                        if(totSoluciones < tamanoPoblacion){//En este caso se adiciona la solucion luego se ordena nuevamente el vector
                                this.Registro[totSoluciones++].Assing(Registro);
                                Ordenar(0,totSoluciones-1);
                        }else{//adiciona la solucion si es mejor que la peor solucion luego se ordena nuevamente el vector
                                if(Registro.Costo < this.Registro[totSoluciones-1].Costo){
                                        int i;
                                        for(i=totSoluciones-1;i>0&&this.Registro[i-1].Costo > Registro.Costo;i--){
                                                if(this.Registro[i-1].Costo > Registro.Costo){
                                                        this.Registro[i].Assing(this.Registro[i-1]);
                                                }
                                        }
                                        this.Registro[i].Assing(Registro);
                                }
                        }
                };

        }
};
```

Paquete Interfaz.

CVRP.java

Esta es la clase main del programa de computadora

```
package Interfaz;

import javax.swing.JFrame;
import javax.swing.UIManager;
import java.io.File;
import Kernel.clsDatosProblema;
```

```
import Kernel.clsSolucion;



public class CVRP {
	/**
	 * @param args
	 */
	public static void main (String args[]){
		System.out.println("Application starting ...");
		frmSplash Splash = new frmSplash();
		Splash.setVisible(true);
		try{
			UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
		}catch(Exception E){
			JFrame.setDefaultLookAndFeelDecorated(true);
		}
		Datos = new clsDatosProblema();
		Solucion = new clsSolucion();
		frmDClientes = new frmClientes();
		frmSolucion = new frmSolucion();
		frmParametros = new frmParametros();
		Archivo=null;
		Iteraciones =1000;
		Intercambios = 2;
		TamanoLista = 18;
		TamanoPoblacion = 100;
		nroIntensificacion = 100;
		nroDiversificacion = 100;
		new mainForm().setVisible(true);
		Splash.setVisible(false);
	}
	static int WIDTH = 800;
	static int HEIGHT = 600;
	public static int Iteraciones;//Iteraciones de Busqueda Tabu
	public static int Intercambios;//Intercambios de Lin-Kernighan
	public static int TamanoLista; //Persistencia en iteraciones de los tabues
	public static int TamanoPoblacion;//Tamano del vector de mejores soluciones
	public static int nroIntensificacion;//Numero de primeras soluciones con las que insensifico
	public static int nroDiversificacion;//Numero de diversificaciones aleatorias
	public static frmClientes frmDClientes;
	public static frmSolucion frmSolucion;
	public static frmParametros frmParametros;
	public static clsDatosProblema Datos;
	public static clsSolucion Solucion;
```

```
		public static File Archivo;
}
```

mainForm.java

Esta clase es el formulario principal del programa de computadora, en ella se implementa el menú de la aplicación, y se invocan las diversas opciones programadas.

```
package Interfaz;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Cursor;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.InputEvent;
import java.awt.event.KeyEvent;
import java.lang.Object;
import java.io.*;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JSeparator;
import javax.swing.JToolBar;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.filechooser.FileFilter;
import javax.swing.KeyStroke;
import javax.swing.ProgressMonitorInputStream;

import Kernel.clsClarkAndWright;
import Kernel.clsTabuSearch;
//import javax.swing.SwingConstants;

public final class mainForm extends JFrame {
		/**
		 *
		 */
		private static final long serialVersionUID = 1L;
```

```
public mainForm(){
        WIDTH = CVRP.WIDTH;
        HEIGHT = CVRP.HEIGHT;
        setSize(WIDTH,HEIGHT);
        setMinimumSize(new Dimension(WIDTH,HEIGHT));
        getContentPane().setBackground(Color.gray);
        initComponents();
        publicarEstado();
}
//<editor-fold defaultstate="collapsed" desc=" Generated Code ">//GEN-BEGIN:initComponents
private void initComponents(){
        //Capturador de eventos Windows
        addWindowListener(new java.awt.event.WindowAdapter(){
                public void windowClosing(java.awt.event.WindowEvent evt){
                        onExit();
                }
        });

        JmnuListener = new ActionListener(){
                public void actionPerformed(ActionEvent evt) {
                        Object item = evt.getSource();
                        if (item ==jmitemNuevo||item ==jbtbarNuevo)
                                jmitemNuevo_onClick(evt);
                        else if (item ==jmitemAbrir||item ==jbtbarAbrir)
                                jmitemAbrir_onClick(evt);
                        else if (item ==jmitemModificar||item ==jbtbarModificar)
                                jmitemModificar_onClick(evt);
                        else if (item ==jmitemGuardar||item ==jbtbarGuardar)
                                jmitemGuardar_onClick(evt);
                        else if (item ==jmitemGuardarComo)
                                jmitemGuardarComo_onClick(evt);
                        else if (item ==jmitemSalir)
                                jmitemSalir_onClick(evt);
                        else if (item ==jmitemEjecutar||item==jbtbarEjecutar)
                                jmitemEjecutar_onClick(evt);
                        else if (item ==jmitemParametros||item==jbtbarParametros)
                                jmitemParametros_onClick(evt);
                        else if (item ==jmitemSolucion||item==jbtbarSolucion)
                                jmitemSolucion_onClick(evt);
                }
        };
        //Menu Aplicacion
        jmbarMenuBar = new JMenuBar();
        //Menu Archivo
```

```
                        jmnuArchivo = new JMenu();
                        jmnuArchivo.setText("Archivo");
                        jmnuArchivo.setMnemonic('A');
                        jmnuArchivo.addMenuListener(new javax.swing.event.MenuListener() {
    public void menuCanceled(javax.swing.event.MenuEvent evt) {
    }
    public void menuDeselected(javax.swing.event.MenuEvent evt) {
    }
    public void menuSelected(javax.swing.event.MenuEvent evt) {
          jmnuArchivo_onSelected(evt);
    }
});
                        //Item Nuevo
                        jmitemNuevo = new JMenuItem();
                        jmitemNuevo.setText("Nuevo...");
                        jmitemNuevo.setMnemonic('N');
                        jmitemNuevo.addActionListener(JmnuListener);
                        jmitemNuevo.setAccelerator(javax.swing.KeyStroke.getKeyStroke(
                                                KeyEvent.VK_N,
                                                InputEvent.CTRL_MASK)
                                );
                        jmnuArchivo.add(jmitemNuevo);
                        //Item Abrir
                        jmitemAbrir = new JMenuItem();
                        jmitemAbrir.setText("Abrir...");
                        jmitemAbrir.setMnemonic('A');
                        jmitemAbrir.addActionListener(JmnuListener);
                        jmitemAbrir.setAccelerator(javax.swing.KeyStroke.getKeyStroke(
                                        KeyEvent.VK_A,
                                        InputEvent.CTRL_MASK)
                                );
                        jmnuArchivo.add(jmitemAbrir);
                        //Item Modificar
                        jmitemModificar = new JMenuItem();
                        jmitemModificar.setText("Modificar...");
                        jmitemModificar.setMnemonic('M');
                        jmitemModificar.addActionListener(JmnuListener);
                        jmitemModificar.setAccelerator(javax.swing.KeyStroke.getKeyStroke(
                                        KeyEvent.VK_M,
                                        InputEvent.CTRL_MASK)
                                );
                        jmnuArchivo.add(jmitemModificar);
                        //Separator
                        jmnuArchivo.add(new JSeparator());
```

```
                    //Item Guardar
                    jmitemGuardar = new JMenuItem();
                    jmitemGuardar.setText("Guardar");
                    jmitemGuardar.setMnemonic('G');
                    jmitemGuardar.addActionListener(JmnuListener);
                    jmitemGuardar.setAccelerator(javax.swing.KeyStroke.getKeyStroke(
                                                KeyEvent.VK_G,
                                                InputEvent.CTRL_MASK)
                            );
                    jmnuArchivo.add(jmitemGuardar);
                    //Item Guardar Como
                    jmitemGuardarComo = new JMenuItem();
                    jmitemGuardarComo.setText("Guardar como ...");
                    jmitemGuardarComo.setMnemonic('C');
                    jmitemGuardarComo.addActionListener(JmnuListener);
                    jmnuArchivo.add(jmitemGuardarComo);
                    //Separator
                    jmnuArchivo.add(new JSeparator());
                    //Item Salir
                    jmitemSalir = new JMenuItem();
                    jmitemSalir.setText("Salir");
                    jmitemSalir.setMnemonic ('S');
                    jmitemSalir.addActionListener(JmnuListener);
                    jmitemSalir.setAccelerator(KeyStroke.getKeyStroke(
                                                KeyEvent.VK_S,
                                                InputEvent.ALT_MASK|
                                                InputEvent.CTRL_MASK)
                            );
                    jmnuArchivo.add(jmitemSalir);
                    jmbarMenuBar.add(jmnuArchivo);
                    //Menu Solucion
                    jmnuSolucion = new JMenu();
                    jmnuSolucion.setText("Solucion");
                    jmnuSolucion.setMnemonic('S');
                    jmnuSolucion.addMenuListener(new javax.swing.event.MenuListener() {
    public void menuCanceled(javax.swing.event.MenuEvent evt) {
    }
    public void menuDeselected(javax.swing.event.MenuEvent evt) {
    }
    public void menuSelected(javax.swing.event.MenuEvent evt) {
        jmnuArchivo_onSelected(evt);
    }
});
                    //Item Ejecutar
```

```
jmitemEjecutar = new JMenuItem();
jmitemEjecutar.setText("Ejecutar");
jmitemEjecutar.setMnemonic('E');
jmitemEjecutar.addActionListener(JmnuListener);
jmitemEjecutar.setAccelerator(javax.swing.KeyStroke.getKeyStroke(
                                        KeyEvent.VK_E,
                                        InputEvent.CTRL_MASK)
        );
jmnuSolucion.add(jmitemEjecutar);
//Item Parametros
jmitemParametros = new JMenuItem();
jmitemParametros.setText("Parametros");
jmitemParametros.setMnemonic('P');
jmitemParametros.addActionListener(JmnuListener);
jmitemParametros.setAccelerator(javax.swing.KeyStroke.getKeyStroke(
                                        KeyEvent.VK_P,
                                        InputEvent.CTRL_MASK)
        );
jmnuSolucion.add(jmitemParametros);
//Item Solucion
jmitemSolucion = new JMenuItem();
jmitemSolucion.setText("Ver Solucion");
jmitemSolucion.setMnemonic('V');
jmitemSolucion.addActionListener(JmnuListener);
jmitemSolucion.setAccelerator(javax.swing.KeyStroke.getKeyStroke(
                                        KeyEvent.VK_V,
                                        InputEvent.CTRL_MASK)
        );
jmnuSolucion.add(jmitemSolucion);

jmbarMenuBar.add(jmnuSolucion);
setJMenuBar(jmbarMenuBar);
//ToolBar
jtbarToolBar = new JToolBar("Tool Bar");
jbtbarNuevo = new JButton();
jbtbarNuevo.setIcon(new ImageIcon( getClass().getResource("/Interfaz/toolbarButtonGraphics/general/New24.gif")));
jbtbarNuevo.setToolTipText("Nuevo");
jbtbarNuevo.setFocusPainted(false);
jbtbarNuevo.addActionListener(JmnuListener);
jbtbarAbrir = new JButton();
jbtbarAbrir.setIcon(new ImageIcon( getClass().getResource("/Interfaz/toolbarButtonGraphics/general/Open24.gif")));
jbtbarAbrir.setToolTipText("Abrir");
jbtbarAbrir.setFocusPainted(false);
jbtbarAbrir.addActionListener(JmnuListener);
```

```
                        jbtbarModificar = new JButton();
                        jbtbarModificar.setIcon(new ImageIcon( getClass().getResource("/Interfaz/toolbarButtonGraphics/table/RowDelete24.gif")));
                        jbtbarModificar.setToolTipText("Modificar Datos");
                        jbtbarModificar.setFocusPainted(false);
                        jbtbarModificar.addActionListener(JmnuListener);
                        jbtbarGuardar = new JButton();
                        jbtbarGuardar.setIcon(new ImageIcon( getClass().getResource("/Interfaz/toolbarButtonGraphics/general/Save24.gif")));
                        jbtbarGuardar.setToolTipText("Guardar");
                        jbtbarGuardar.setFocusPainted(false);
                        jbtbarGuardar.addActionListener(JmnuListener);
                        jbtbarEjecutar = new JButton();
                        jbtbarEjecutar.setIcon(new ImageIcon( getClass().getResource("/Interfaz/toolbarButtonGraphics/general/Import24.gif")));
                        jbtbarEjecutar.setToolTipText("Buscar solucion");
                        jbtbarEjecutar.setFocusPainted(false);
                        jbtbarEjecutar.addActionListener(JmnuListener);
                        add(jtbarToolBar,BorderLayout.PAGE_START);
                        jtbarToolBar.add(jbtbarNuevo);
                        jtbarToolBar.add(jbtbarAbrir);
                        jtbarToolBar.add(jbtbarModificar);
                        //jtbarToolBar.add(new JSeparator(SwingConstants.VERTICAL));
                        jtbarToolBar.add(jbtbarGuardar);
                        jtbarToolBar.add(jbtbarEjecutar);
                        //jtbarToolBar.add(new JSeparator(SwingConstants.VERTICAL));
                        jtbarToolBar.setBounds(0,0,492,36);
        };
        //</editor-fold>//GEN-END:initComponents
//  Define que Items del menu estaran activos en que momento
        private void jmnuArchivo_onSelected(javax.swing.event.MenuEvent evt){
                publicarEstado();
        }
        private void onExit(){
                System.out.println("trying to close ...");
                System.exit(0);
        };
        //Abre un formulario para crear un nuevo problema
        private void jmitemNuevo_onClick(ActionEvent evt){
                if (!CVRP.Datos.Guardado()){
                        switch (JOptionPane.showConfirmDialog(this,"Desea almacenar los datos actuales","Application
Information",JOptionPane.YES_NO_CANCEL_OPTION)){
                        case(JOptionPane.YES_OPTION):
                                jmitemGuardar_onClick(evt);
                                CVRP.Datos.reset();
                                CVRP.Solucion.reset();
                                CVRP.Archivo=null;
```

```
                                        CVRP.frmDClientes.setVisible(true);
                                        break;
                                case(JOptionPane.NO_OPTION):
                                        CVRP.Datos.reset();
                                        CVRP.Solucion.reset();
                                        CVRP.Archivo=null;
                                        CVRP.frmDClientes.setVisible(true);
                                        break;
                                case(JOptionPane.CANCEL_OPTION):
                                        break;
                                }
                        }else{
                                CVRP.Datos.reset();
                                CVRP.Solucion.reset();
                                CVRP.Archivo=null;
                                CVRP.frmDClientes.setVisible(true);
                        }
                        publicarEstado();
                }
//              Guarda los datos del problema actual en la ubicacion de indicada por el usuario
                private void jmitemGuardarComo_onClick(ActionEvent evt){
                        try{
                                if (getFileName()){
                                        saveFile();
                                }
                        }catch(IOException e){

                        }
                        publicarEstado();
                }
                //  Abre un archivo del tipo vrp y lo carga como datos del problema
                private void jmitemAbrir_onClick(ActionEvent evt){
                        try{
                                if (!CVRP.Datos.Guardado()){
                                        switch (JOptionPane.showConfirmDialog(this,"Desea almacenar los datos actuales","Application
Information",JOptionPane.YES_NO_CANCEL_OPTION)){
                                        case(JOptionPane.YES_OPTION):
                                                jmitemGuardar_onClick(evt);
                                                CVRP.Datos.reset();
                                                CVRP.Solucion.reset();
                                                CVRP.Archivo=null;
                                                openFile();
                                                break;
                                        case(JOptionPane.NO_OPTION):
```

```
                                                CVRP.Datos.reset();
                                                CVRP.Solucion.reset();
                                                CVRP.Archivo=null;
                                                openFile();
                                                break;
                                        case(JOptionPane.CANCEL_OPTION):
                                                break;
                                        }
                                }else{
                                        CVRP.Datos.reset();
                                        CVRP.Solucion.reset();
                                        CVRP.Archivo=null;
                                        openFile();
                                }
                        }catch(IOException e){}
                        publicarEstado();
                }
//              Modifica los datos del problema actual
                private void jmitemModificar_onClick(ActionEvent evt){
                        CVRP.frmDClientes.setVisible(true);
                        publicarEstado();
                }
//              Guarda los datos del problema actual en la ubicacion de CVRP
                private void jmitemGuardar_onClick(ActionEvent evt){
                        try{
                                if (CVRP.Archivo==null){
                                        if (getFileName()){
                                                saveFile();
                                        }
                                }else{
                                        saveFile();
                                }
                        }catch(IOException e){

                        }
                        publicarEstado();
                }
//              Salida de la aplicacion
                private void jmitemSalir_onClick(ActionEvent evt){
                        onExit();
                }
//              Ejecuta el algoritmo de busqueda de la solucion
                private void jmitemEjecutar_onClick(ActionEvent evt){
                        estadoAntesCalcular(false);
```

```
                final clsClarkAndWright CW = new clsClarkAndWright(CVRP.Datos,CVRP.Solucion);
                final clsTabuSearch TS = new clsTabuSearch(CVRP.Datos,CVRP.Solucion,CVRP.Iteraciones,CVRP.Intercambios,CVRP.TamanoLista,CVRP.TamanoPoblacion,CVRP.nroIntensificacion,CVRP.nroDiversificacion);
                Thread thrCW = new Thread(){
                        public void run(){
                                CW.Calcular();
                                TS.Calcular();
                                estadoAntesCalcular(true);
                                jmitemSolucion_onClick(null);
                        }
                };
                thrCW.start();
        }
//      Muestra los parametros de busqueda de la solucion
        private void jmitemParametros_onClick(ActionEvent evt){
                CVRP.frmParametros.setVisible(true);
        }
//      Muestra el formulario de solucion
        private void jmitemSolucion_onClick(ActionEvent evt){
                setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
                CVRP.frmSolucion.setVisible(true);
                //CVRP.Solucion.print(true);
        }
//      Busca el nombre en el que se almacenara la informacion del archivo
        private boolean getFileName(){
                JFileChooser chooser = new JFileChooser();
                chooser.setCurrentDirectory(new File("."));
                chooser.setFileFilter(
                        new FileFilter(){
                                public  boolean accept(File f){
                                        String fname = f.getName().toLowerCase();
                                        return fname.endsWith(".vrp")||f.isDirectory();
                                }
                                public String getDescription(){
                                        return "vrp Files";
                                }
                        }
                );
                int r = chooser.showSaveDialog(this);
                if(r != JFileChooser.APPROVE_OPTION) return false;
                CVRP.Archivo=chooser.getSelectedFile();
                return true;
        }
```

```
//          Abre un archivo vrp para cargar un nuevo problema
        public void openFile() throws IOException{
                JFileChooser chooser = new JFileChooser();
                chooser.setCurrentDirectory(new File("."));
                chooser.setFileFilter(
                        new FileFilter(){
                                public  boolean accept(File f){
                                        String fname = f.getName().toLowerCase();
                                        return fname.endsWith(".vrp")||f.isDirectory();
                                }
                                public String getDescription(){
                                        return "vrp Files";
                                }
                        }
                );
                int r = chooser.showOpenDialog(this);
                if(r != JFileChooser.APPROVE_OPTION) return;
                final File f = chooser.getSelectedFile();
                CVRP.Archivo = f;
                FileInputStream  fileIn = new FileInputStream(f);
                ProgressMonitorInputStream progressIn = new ProgressMonitorInputStream(this,"Reading "+f.getName(),fileIn);
                InputStreamReader inReader = new InputStreamReader(progressIn);
                final BufferedReader in = new BufferedReader(inReader);
                Thread readThread = new Thread(){
                        public void run(){
                                try{
                                        String line;
                                        int i = 0;
                                        CVRP.Datos.reset();
                                        CVRP.Solucion.reset();
                                        while((line=in.readLine())!=null){
                                                if(line.length() > 0){
                                                        i++;
                                                        String[] Partes = line.split(":");
                                                        if (i==4 && Partes.length > 0){
                                                                try{

        CVRP.Datos.setNumeroClientes(Integer.parseInt(Partes[1].trim())-1);
                                                                }catch(Exception e){
                                                                        e.printStackTrace();
                                                                }
                                                        }
                                                        if (i==6 && Partes.length > 0){
                                                                try{
```

```
CVRP.Datos.setCapacidadVehiculos(Integer.parseInt(Partes[1].trim()));
                                                                        }catch(Exception e){
                                                                                e.printStackTrace();
                                                                        }
                                                                }
                                                                String[] Coord = line.split(" ");
                                                                //La ubicación del depósito esta en la posición 8 del archivo
                                                                if (i == 8 && i <= CVRP.Datos.NumeroClientes() + 8 &&
Coord.length > 0){
                                                                        try{

CVRP.Datos.setDeposito(Float.parseFloat(Coord[1].trim()),Float.parseFloat(Coord[2].trim()));
                                                                        }catch(Exception e){
                                                                                e.printStackTrace();
                                                                        }
                                                                }
                                                                //La ubicación de los clientes empiezan en la posición 9
                                                                if (i > 8 && i <= CVRP.Datos.NumeroClientes() + 8 &&
Coord.length > 0){
                                                                        try{
                                                                                CVRP.Datos.setX(i-
8,Float.parseFloat(Coord[1].trim()));
                                                                                CVRP.Datos.setY(i-
8,Float.parseFloat(Coord[2].trim()));
                                                                        }catch(Exception e){
                                                                                e.printStackTrace();
                                                                        }
                                                                }
                                                                //La ubicación de las demandas de los clientes 9
                                                                if (i > CVRP.Datos.NumeroClientes() + 10 && i <= 2 *
CVRP.Datos.NumeroClientes() + 10 && Coord.length > 0){
                                                                        try{
                                                                                CVRP.Datos.setD(i-
(CVRP.Datos.NumeroClientes() + 10),Float.parseFloat(Coord[1].trim()));
                                                                        }catch(Exception e){
                                                                                e.printStackTrace();
                                                                        }
                                                                }
                                                        }
                                                }
                                                CVRP.Datos.setGuardado();
                                                CVRP.Datos.dumpData();
                                                publicarEstado();
```

```
                                        in.close();
                                    }catch (IOException exception){
                                        exception.printStackTrace();
                                    }
                                }
                            };
                            readThread.start();
                    }
//                  Guarda en el archivo descrito en CVRP strNArchivo la informacion del problema
                    public void saveFile() throws IOException{
                            final PrintWriter out = new PrintWriter(new FileOutputStream(CVRP.Archivo));
                            Thread writeThread = new Thread(){
                                    public void run(){
                                        try{
                                            setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
                                            out.println("NAME : " + CVRP.Archivo.getCanonicalPath());
                                            out.println("COMMENT : (Creado con el programa CVRP)");
                                            out.println("TYPE : CVRP");
                                            out.println("DIMENSION : " + (CVRP.Datos.NumeroClientes() + 1));
                                            out.println("EDGE_WEIGHT_TYPE : EUC_2D");
                                            out.println("CAPACITY : " + CVRP.Datos.CapacidadVehiculos());
                                            out.println("NODE_COORD_SECTION");
                                            for(int i=0;i<=CVRP.Datos.NumeroClientes();i++){
                                                out.println(i + 1 + " " + CVRP.Datos.X(i) + " " + CVRP.Datos.Y(i));
                                            }
                                            out.println("DEMAND_SECTION");
                                            for(int i=0;i<=CVRP.Datos.NumeroClientes();i++){
                                                out.println(i + 1 + " " + CVRP.Datos.D(i));
                                            }
                                            out.println("DEPOT_SECTION");
                                            out.println(" 1 ");
                                            out.println(" -1");
                                            out.println("EOF");
                                            out.close();
                                            setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
                                        }catch (Exception exception){
                                            exception.printStackTrace();
                                            setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
                                        }
                                    }
                            };
                            writeThread.start();
                    }
//                  Define el estado del programa cuando se activa el calculo
```

```
	private void estadoAntesCalcular(boolean value){
		setCursor(new Cursor(Cursor.WAIT_CURSOR));
		jmitemNuevo.setEnabled(value);
		jbtbarNuevo.setEnabled(value);
		jmitemAbrir.setEnabled(value);
		jbtbarAbrir.setEnabled(value);
		jmitemModificar.setEnabled(value);
		jbtbarModificar.setEnabled(value);
		jmitemGuardar.setEnabled(value);
		jbtbarGuardar.setEnabled(value);
		jmitemGuardarComo.setEnabled(value);
		jmitemEjecutar.setEnabled(value);
		jbtbarEjecutar.setEnabled(value);
		jmitemSolucion.setEnabled(value);
		jmnuArchivo.setEnabled(value);
		jmnuSolucion.setEnabled(value);
	}
//	Define que Items del menu estaran activos en que momento
	private void publicarEstado(){
		setTitle(Titulo);
		if (CVRP.Archivo!=null)  setTitle(Titulo + " - " + CVRP.Archivo.getAbsolutePath());
		if (CVRP.Datos!=null){
			jmitemModificar.setEnabled(CVRP.Datos.tieneDatos());
			jbtbarModificar.setEnabled(CVRP.Datos.tieneDatos());
			jmitemEjecutar.setEnabled(CVRP.Datos.tieneDatos());
			jbtbarEjecutar.setEnabled(CVRP.Datos.tieneDatos());
			jmitemSolucion.setEnabled(CVRP.Solucion.blnTieneSolucion());
			jmitemGuardar.setEnabled(CVRP.Datos.tieneDatos() && !CVRP.Datos.Guardado());
			jbtbarGuardar.setEnabled(CVRP.Datos.tieneDatos());
			jmitemGuardarComo.setEnabled(CVRP.Datos.tieneDatos() && !CVRP.Datos.Guardado());
		}
	}
	//Components
	String Titulo = "CVRP";
	int WIDTH = 800;
	int HEIGHT = 600;
	JMenuBar jmbarMenuBar;
	JMenu jmnuArchivo;
	JMenuItem jmitemNuevo;
	JMenuItem jmitemAbrir;
	JMenuItem jmitemModificar;
	JMenuItem jmitemGuardar;
	JMenuItem jmitemGuardarComo;
	JMenuItem jmitemSalir;
```

```
		JMenu jmnuSolucion;
		JMenuItem jmitemEjecutar;
		JMenuItem jmitemParametros;
		JMenuItem jmitemSolucion;
		JToolBar jtbarToolBar;
		JButton jbtbarNuevo;
		JButton jbtbarAbrir;
		JButton jbtbarModificar;
		JButton jbtbarGuardar;
		JButton jbtbarEjecutar;
		JButton jbtbarParametros;
		JButton jbtbarSolucion;
		ActionListener JmnuListener;
}
```

frmClientes.java

Esta clase es la encargada de tomar la información de un problema a través del teclado y almacenarla en la respectiva estructura de datos para posterior uso.

```
package Interfaz;

import java.awt.Dimension;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.ActionEvent;

import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.text.JTextComponent;

import Kernel.clsDatosProblema;
public class frmClientes extends JDialog {
		private static final long serialVersionUID = 1L;
   public frmClientes() {
      setPreferredSize(new Dimension(WIDTH,HEIGHT));
      setTitle("Captura datos");
      setLocationRelativeTo(this);
      setModal(true);
      initComponents();
   }
   /** This method is called from within the constructor to
```

```
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     **/
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">//GEN-BEGIN:initComponents
    private void initComponents() {
        setDefaultCloseOperation(javax.swing.WindowConstants.HIDE_ON_CLOSE);
        setBounds(new java.awt.Rectangle(100, 100, WIDTH,HEIGHT));
        setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        setLocation(CVRP.WIDTH/2 - (WIDTH/2),CVRP.HEIGHT/2 - (HEIGHT/2));
        setResizable(false);
                Datos=null;
                /**
                 * Interfaz del formulario
                 */
        jPanel1 = new javax.swing.JPanel();
        jPanel5 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jtxtNClientes = new JTextFieldInteger();
        jPanel6 = new javax.swing.JPanel();
        jLabel2 = new javax.swing.JLabel();
        jtxtCVehiculos = new JTextFieldInteger();
        jPanel7 = new javax.swing.JPanel();
        jLabel3 = new javax.swing.JLabel();
        jtxtUXDeposito = new JTextFieldDouble();
        jPanel8 = new javax.swing.JPanel();
        jLabel4 = new javax.swing.JLabel();
        jtxtUYDeposito = new JTextFieldDouble();
        jPanel2 = new javax.swing.JPanel();
        jPanel3 = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jtblDClientes = new javax.swing.JTable();
        jPanel4 = new javax.swing.JPanel();
        jbtnAceptar = new javax.swing.JButton();
        jbtnCancelar = new javax.swing.JButton();
        jPanel1.setLayout(new java.awt.GridLayout(2, 2));
        //Se diseña el panel de Numero de clientes
        jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(" Datos del problema "));
        jPanel1.setPreferredSize(new java.awt.Dimension(100, 100));
        jLabel1.setText("N\u00famero de clientes");
        jLabel1.setPreferredSize(new java.awt.Dimension(150, 20));
        jPanel5.add(jLabel1);
        jtxtNClientes.setText("jTextField1");
        jtxtNClientes.setPreferredSize(new java.awt.Dimension(100, 20));
```

```
jPanel5.add(jtxtNClientes);
jPanel1.add(jPanel5);
//Se diseña el panel capacidad de vehiculos
jLabel2.setText("Capacidad veh\u00edculos");
jLabel2.setPreferredSize(new java.awt.Dimension(150, 20));
jPanel6.add(jLabel2);
jtxtCVehiculos.setText("jTextField2");
jtxtCVehiculos.setPreferredSize(new java.awt.Dimension(100, 20));
jPanel6.add(jtxtCVehiculos);
jPanel1.add(jPanel6);
//Se diseña el panel ubicacion de deposito X
jLabel3.setText("Ubicacion dep\u00f3sito X");
jLabel3.setPreferredSize(new java.awt.Dimension(150, 20));
jPanel7.add(jLabel3);
jtxtUXDeposito.setText("jTextField1");
jtxtUXDeposito.setPreferredSize(new java.awt.Dimension(100, 20));
jPanel7.add(jtxtUXDeposito);
jPanel1.add(jPanel7);
//Se diseña el panel ubicacion de deposito Y
jLabel4.setText("Ubicacion dep\u00f3sito Y");
jLabel4.setPreferredSize(new java.awt.Dimension(150, 20));
jPanel8.add(jLabel4);
jtxtUYDeposito.setText("jTextField2");
jtxtUYDeposito.setPreferredSize(new java.awt.Dimension(100, 20));
jPanel8.add(jtxtUYDeposito);
jPanel1.add(jPanel8);
//Se asocia el panel contenedor de los paneles independientes de datos a la parte Norte del formulario
getContentPane().add(jPanel1, java.awt.BorderLayout.NORTH);
//Se Crea el panel contenedor de los datos de los clientes
jPanel2.setLayout(new java.awt.BorderLayout());
jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder(" Ubicaci\u00f3n de los clientes "));
jPanel3.setLayout(new java.awt.GridLayout(1, 0));
jPanel2.add(jPanel3, java.awt.BorderLayout.NORTH);
jScrollPane1.setViewportView(jtblDClientes);
jPanel2.add(jScrollPane1, java.awt.BorderLayout.CENTER);
jbtnAceptar.setMnemonic('A');
jbtnAceptar.setText("Aceptar");
jbtnAceptar.setToolTipText("Aceptar");
jPanel4.add(jbtnAceptar);
jbtnCancelar.setMnemonic('C');
jbtnCancelar.setText("Cancelar");
jbtnCancelar.setToolTipText("Cancelar operacion");
jPanel4.add(jbtnCancelar);
jPanel2.add(jPanel4, java.awt.BorderLayout.SOUTH);
```

```
//Se asocia el panel contenedor de los paneles independientes de datos a la parte centro del formulario
getContentPane().add(jPanel2, java.awt.BorderLayout.CENTER);
pack();
/**
 * Capturadores de eventos
 */
                        //Capturadores de eventos Windows de la forma principal
                        addWindowListener(new java.awt.event.WindowAdapter(){
                                public void windowClosing(java.awt.event.WindowEvent evt){//Se ejecuta siempre que se muestra la forma
                                        onClose();
                                }

                                public void windowActivated(java.awt.event.WindowEvent evt){//Se ejecuta siempre que se muestra la forma
                                        onActivated();
                                }
                        });
                        //Capturadores del tipo onChanged
                        jtxtKeyAdapter = new KeyAdapter(){
        public void keyReleased(KeyEvent evt){
                                        Object item = evt.getSource();
                                        if (item ==jtxtNClientes)
                                                jtxtNClientes_onChanged(jtxtNClientes);
                                        else if (item ==jtxtCVehiculos)
                                                jtxtCVehiculos_onChanged(jtxtCVehiculos);
                                        else if (item ==jtxtUXDeposito||item ==jtxtUYDeposito)
                                                jtxtUDeposito_onChanged(evt);
        }
};
jtxtNClientes.addKeyListener(jtxtKeyAdapter);
jtxtCVehiculos.addKeyListener(jtxtKeyAdapter);
jtxtUXDeposito.addKeyListener(jtxtKeyAdapter);
jtxtUYDeposito.addKeyListener(jtxtKeyAdapter);
//Capturadores de eventos de los botones
jtxtAListener = new ActionListener(){
        public void actionPerformed(ActionEvent evt){
                        Object item =evt.getSource();
                        if(item==jbtnAceptar){
                                jbtnAceptar_onClick(jbtnAceptar);
                        }else if(item==jbtnCancelar){
                                jbtnCancelar_onClick(jbtnCancelar);
                        }
        }
};
getRootPane().setDefaultButton(jbtnAceptar);
```

```
        jbtnAceptar.addActionListener(jtxtAListener);
        jbtnCancelar.addActionListener(jtxtAListener);
    }// </editor-fold>//GEN-END:initComponents
    /**
     * Al momento del cerrar el formulario el modelo local no es mas válido
     */
    public void onClose(){
                jtxtNClientes.grabFocus();
                Datos=null;
    }
    /**
     * Siempre que se muestra el formulario, es necesario cargar la informacion que se encuentra
     * en modelo para evitar desincronizacion, cuando es el Show del formulario es necesario
     * tomar la informacion desde los datos del problema
     */
    public void onActivated(){
                if (Datos==null){
                        Datos = new clsDatosProblema(); //Se crea un objeto del tipo DatosProblema para trabajar con copias locales hasta que se
acepte el formulario
                        Datos.Assign(CVRP.Datos);//Cargo la informacion inicial desde los datos del programa
                }
                this.jtxtNClientes.setText((Datos.NumeroClientes()==0)?"":String.valueOf(Datos.NumeroClientes()));
                this.jtxtCVehiculos.setText((Datos.CapacidadVehiculos()==0)?"":String.valueOf(Datos.CapacidadVehiculos()));
                this.jtxtUXDeposito.setText((Datos.X(0)==0)?"":String.valueOf(Datos.X(0)));
                this.jtxtUYDeposito.setText((Datos.Y(0)==0)?"":String.valueOf(Datos.Y(0)));
        jtblDClientes.setModel(new ClientesTableModel(Datos));
    }
    /**
     * Métodos que son invocados siempre que hay cambio en los TextComponentes
     * @param sender Variable que desperto el evento
     */
    //Cambios en el número de clientes
    public void jtxtNClientes_onChanged(JTextComponent sender){
                try{
                        if(sender.getText().compareTo("")==0){
                                Datos.setNumeroClientes(0);
                        }else{
                                Datos.setNumeroClientes(Integer.parseInt(sender.getText()));
                        }
                jtblDClientes.setModel(new ClientesTableModel(Datos));
                }catch(Exception e){
                }
    }
    //Cambios en la capacidad de los vehiculos
```

```
public void jtxtCVehiculos_onChanged(JTextComponent sender){
	try{
		if(sender.getText().compareTo("")==0){
			Datos.setCapacidadVehiculos(0);
		}else{
			Datos.setCapacidadVehiculos(Integer.parseInt(sender.getText()));
		}
	}catch(Exception e){
	}
}
//Cambios en la ubicacion del depósito
public void jtxtUDeposito_onChanged(KeyEvent evt){
	double x,y;
	try{
		if(jtxtUXDeposito.getText().compareTo("")==0){//Validacion de la informacion de la coordenada X
			x=0;
		}else{
			x=Double.parseDouble(jtxtUXDeposito.getText());
		}
		if(jtxtUYDeposito.getText().compareTo("")==0){//Validacion de la informacion de la coordenada Y
			y=0;
		}else{
			y=Double.parseDouble(jtxtUYDeposito.getText());
		}
		Datos.setDeposito(x,y);
	}catch(Exception e){
	}
}
/**
 * accion de los botones
 */
private void jbtnAceptar_onClick(JButton sender){
	if (validarDatos()){
	CVRP.Datos.Assign(Datos);
	CVRP.Datos.dumpData();
	this.onClose();
	this.setVisible(false);
	}
};

private void jbtnCancelar_onClick(JButton sender){
	this.onClose();
	this.setVisible(false);
};
```

```
        private boolean validarDatos(){
                int intTemporal;
                //Validacion: El número de clientes sea mayor que cero
                intTemporal = (jtxtNClientes.getText().compareTo("")==0)?0:Integer.parseInt(jtxtNClientes.getText().trim());
                if (intTemporal <= 0){
                        JOptionPane.showMessageDialog(this,"El número de clientes debe ser superior a 0","Application Error",JOptionPane.ERROR_MESSAGE);
                        jtxtNClientes.grabFocus();
                        jtxtNClientes.selectAll();
                        return false;
                }
                //Validacion: La Capacidad de los vehiculos debe ser mayor que ceros
                intTemporal = (jtxtCVehiculos.getText().compareTo("")==0)?0:Integer.parseInt(jtxtCVehiculos.getText().trim());
                if (intTemporal <= 0){
                        JOptionPane.showMessageDialog(this,"La Capacidad de los vehiculos debe ser superior a 0","Application Error",JOptionPane.ERROR_MESSAGE);
                        jtxtCVehiculos.grabFocus();
                        jtxtCVehiculos.selectAll();
                        return false;
                }
                return true;
        }

    // Variables declaration - do not modify//GEN-BEGIN:variables
        int WIDTH = 600;
        int HEIGHT = 400;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JPanel jPanel6;
    private javax.swing.JPanel jPanel7;
    private javax.swing.JPanel jPanel8;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JButton jbtnAceptar;
    private javax.swing.JButton jbtnCancelar;
    private javax.swing.JTable jtblDClientes;
    private javax.swing.JTextField jtxtCVehiculos;
```

```
    private javax.swing.JTextField jtxtNClientes;
    private javax.swing.JTextField jtxtUXDeposito;
    private javax.swing.JTextField jtxtUYDeposito;
    private KeyAdapter jtxtKeyAdapter;
    private ActionListener jtxtAListener;
    private clsDatosProblema Datos;
    // End of variables declaration//GEN-END:variables
}
```

ClientesTableModel.java

Modelo de la vista “Datos de los clientes” a través de este modelo la información que es capturada por el teclado, es validada y mostrada nuevamente al usuario.

```
package Interfaz;

import javax.swing.table.AbstractTableModel;
import Kernel.clsDatosProblema;

class ClientesTableModel extends AbstractTableModel{
        /**
         * Esta clase es el modelo que permite la informacion de los
         * clientes sea mostrada en una tabla de Swing
         */
        private static final long serialVersionUID = 1L;
        public ClientesTableModel(clsDatosProblema C){
                this.C = C;
        }

        public int getRowCount(){
                return C.NumeroClientes();
        }


        public int getColumnCount(){
                return 3;
        }

        public Object getValueAt(int row, int col){
                String Cadena;
                switch (col){
                case 0:
                        Cadena = new Double(C.X(row+1)).toString();
                        return C.X(row+1)==0.0?"0":Cadena;
```

```
                case 1:
                        Cadena = new Double(C.Y(row+1)).toString();
                        return C.Y(row+1)==0.0?"0":Cadena;
                case 2:
                        Cadena = new Double(C.D(row+1)).toString();
                        return C.D(row+1)==0.0?"0":Cadena;
                default:
                        return "Nada";
                }
        }

        public void setValueAt(Object aValue, int rowIndex, int columnIndex){
                Double dblValue;
                try{
                        dblValue = new Double(aValue.toString());
                }
                catch(Exception ex){
                        dblValue = new Double(0);
                }
                switch (columnIndex){
                case 0:
                        C.setX(rowIndex+1, dblValue);
                        break;
                case 1:
                        C.setY(rowIndex+1, dblValue);
                        break;
                case 2:
                        C.setD(rowIndex+1, dblValue);
                        break;
                }
        }

        public String getColumnName(int c){
                switch (c){
                case 0:
                        return "Coordenada X";
                case 1:
                        return "Coordenada Y";
                case 2:
                        return "Demanda";
                default:
                        return "UNKNOW";
                }
        }
```

```
	public boolean isCellEditable(int rowIndex, int columnIndex){
		return true;
	}
	clsDatosProblema C;
}
```

JTextFieldDouble.java

Se creó esta clase para implementar un Text Field que no permitiera el ingreso de información diferente a datos flotantes.

```
package Interfaz;

public class JTextFieldDouble extends javax.swing.JTextField{
	private static final long serialVersionUID = 1L;

	protected javax.swing.text.Document createDefaultModel(){
	   return new PlainDocumentDouble();
	}
	protected class PlainDocumentDouble extends javax.swing.text.PlainDocument{
		private static final long serialVersionUID = 1L;
		public void insertString(int offs, String str, javax.swing.text.AttributeSet a)
	throws javax.swing.text.BadLocationException{
			char[] fuente = str.toCharArray();// fuente: almacena el contenido de la caja de texto
			char[] resultado = new char[fuente.length];// resultado: almacena el contenido de la caja de texto validado
			String Cadena = this.getText(0,offs) + str;//Este es el valor de la cadena como quedaria al agregar el dato
			if (Cadena.compareTo("-")==0) Cadena+="0";
			//Determinamos si el valor resultante de la insercion seria una flotante válido válida
			try{
				Double.parseDouble(Cadena);
				int j = 0;
				// Almacenar en resultado los caracteres válidos de fuente
				for (int i = 0; i < fuente.length; i++){
					if ((fuente[i] >= '0' && fuente[i] <= '9' || fuente[i] == '.' || (fuente[i] == '-' && offs==0)||
(fuente[i] == '+' && offs==0))&&offs <10)
						resultado[j++] = fuente[i];
				}
				super.insertString(offs, new String(resultado, 0, j), a);
			}catch(Exception e){}
		}
	}
}
```

JTextFieldInteger.java

Se creó esta clase para implementar un Text Field que no permitiera el ingreso de información diferente a datos enteros.

```
package Interfaz;
import javax.swing.JTextField;
public class JTextFieldInteger extends JTextField {
        private static final long serialVersionUID = 1L;

        protected javax.swing.text.Document createDefaultModel(){
           return new PlainDocumentDouble();
        }
        protected class PlainDocumentDouble extends javax.swing.text.PlainDocument{
                private static final long serialVersionUID = 1L;
                public void insertString(int offs, String str, javax.swing.text.AttributeSet a)
      throws javax.swing.text.BadLocationException{
                        // fuente: almacena el contenido de la caja de texto
                        char[] fuente = str.toCharArray();
                        // resultado: almacena el contenido de la caja de texto validado
                        char[] resultado = new char[fuente.length];
                        int j = 0;
                        // Almacenar en resultado los caracteres válidos de fuente
                        for (int i = 0; i < fuente.length; i++){
                                if (fuente[i] >= '0' && fuente[i] <= '9' && offs < 5)
                                        resultado[j++] = fuente[i];
                        }
                        super.insertString(offs, new String(resultado, 0, j), a);
                }
        }
}
```

frmParametros.java

Este formulario me permite ingresar la información de los parámetros configurables del algoritmo creado.

```
package Interfaz;

import java.awt.Cursor;
import java.awt.Dimension;
import java.awt.event.WindowAdapter;
```

```
public class frmParametros extends javax.swing.JFrame {

    /**
                 *
                 */
                private static final long serialVersionUID = 1L;
                /** Creates new form frmParametros */
    public frmParametros() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    private void initComponents() {
        jpnlDatos = new javax.swing.JPanel();
        jpnlNroIteraciones = new javax.swing.JPanel();
        jlblNroIteraciones = new javax.swing.JLabel();
        jcbxNroIteraciones = new javax.swing.JComboBox();
        jpnlNroIntercambios = new javax.swing.JPanel();
        jlblNroIntercambios = new javax.swing.JLabel();
        jcbxNroIntercambios = new javax.swing.JComboBox();
        jpnlTamanoLista = new javax.swing.JPanel();
        jlblTamanoLista = new javax.swing.JLabel();
        jcbxTamanoLista = new javax.swing.JComboBox();
        jpnlTamanoPoblacion = new javax.swing.JPanel();
        jlblTamanoPoblacion = new javax.swing.JLabel();
        jcbxTamanoPoblacion = new javax.swing.JComboBox();
        jpnlBotones = new javax.swing.JPanel();
        jbtnAceptar = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.HIDE_ON_CLOSE);
        setTitle("Parametros");
        setBounds(new java.awt.Rectangle(100, 100, WIDTH,HEIGHT));
        setLocation(CVRP.WIDTH/2 - (WIDTH/2),CVRP.HEIGHT/2 - (HEIGHT/2));
        setResizable(false);
        setMinimumSize(new java.awt.Dimension(WIDTH, HEIGHT));
        setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));

        jpnlDatos.setLayout(new java.awt.GridLayout(2, 2));
        jpnlDatos.setBorder(javax.swing.BorderFactory.createTitledBorder(" Parametros de la soluc\u00f3n "));
```

```
jpnlDatos.setPreferredSize(new java.awt.Dimension(WIDTH, HEIGHT));
jpnlDatos.setRequestFocusEnabled(false);
//Nro de iteraciones
jpnlNroIteraciones.setLayout(null);
jpnlNroIteraciones.setPreferredSize(new Dimension(100, 100));
jlblNroIteraciones.setText("Nro de iteraciones");
jlblNroIteraciones.setBounds(0,0,100,20);
jpnlNroIteraciones.add(jlblNroIteraciones);
jcbxNroIteraciones.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "500", "750", "1000", "10000" }));
jcbxNroIteraciones.setPreferredSize(new java.awt.Dimension(100, 20));
jcbxNroIteraciones.setBounds(100,0,100,20);
jpnlNroIteraciones.add(jcbxNroIteraciones);
//Nro Intercambios
jpnlNroIntercambios.setLayout(null);
jpnlNroIntercambios.setPreferredSize(new Dimension(100, 100));
jlblNroIntercambios.setText("Nro Intercambios");
jlblNroIntercambios.setBounds(0,0,100,20);
jpnlNroIntercambios.add(jlblNroIntercambios);
jcbxNroIntercambios.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "2", "3", "4" }));
jcbxNroIntercambios.setPreferredSize(new java.awt.Dimension(100, 20));
jcbxNroIntercambios.setBounds(100,0,100,20);
jpnlNroIntercambios.add(jcbxNroIntercambios);
//Tamano Lista
jpnlTamanoLista.setLayout(null);
jpnlTamanoLista.setPreferredSize(new Dimension(100,100));
jlblTamanoLista.setText("Tama\u00f1o lista");
jlblTamanoLista.setBounds(0,0,100,20);
jpnlTamanoLista.add(jlblTamanoLista);
jcbxTamanoLista.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "6", "12", "18" }));
jcbxTamanoLista.setPreferredSize(new java.awt.Dimension(100, 20));
jcbxTamanoLista.setBounds(100,0,100,20);
jpnlTamanoLista.add(jcbxTamanoLista);
//Tamano Poblacion
jpnlTamanoPoblacion.setLayout(null);
jpnlTamanoPoblacion.setPreferredSize(new Dimension(100,100));
jlblTamanoPoblacion.setText("Tama\u00f1o poblacion");
jlblTamanoPoblacion.setBounds(0,0,100,20);
jpnlTamanoPoblacion.add(jlblTamanoPoblacion);
jcbxTamanoPoblacion.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "100", "200", "300" }));
jcbxTamanoPoblacion.setPreferredSize(new java.awt.Dimension(100, 20));
jcbxTamanoPoblacion.setBounds(100,0,100,20);
jpnlTamanoPoblacion.add(jcbxTamanoPoblacion);
//Adiciono los paneles
jpnlDatos.add(jpnlNroIteraciones);
```

```
        jpnlDatos.add(jpnlTamanoLista);
        jpnlDatos.add(jpnlNroIntercambios);
        jpnlDatos.add(jpnlTamanoPoblacion);

        getContentPane().add(jpnlDatos, java.awt.BorderLayout.CENTER);

        jbtnAceptar.setMnemonic('A');
        jbtnAceptar.setText("Aceptar");
        jbtnAceptar.setCursor(new Cursor(Cursor.HAND_CURSOR));
        jbtnAceptar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jbtnAceptar_OnClick(evt);
            }
        });
        jpnlBotones.add(jbtnAceptar);

        getContentPane().add(jpnlBotones, java.awt.BorderLayout.SOUTH);
        pack();
        addWindowListener(
                        new WindowAdapter(){
                                public void windowActivated(java.awt.event.WindowEvent evt){//Se ejecuta siempre que se muestra la
forma
                                        onActivated();
                                }
                        }
        );

    }// </editor-fold>

    private void jbtnAceptar_OnClick(java.awt.event.ActionEvent evt) {
                CVRP.Iteraciones = Integer.parseInt((String) jcbxNroIteraciones.getSelectedItem());
                CVRP.Intercambios = Integer.parseInt((String) jcbxNroIntercambios.getSelectedItem());
                CVRP.TamanoLista = Integer.parseInt((String) jcbxTamanoLista.getSelectedItem());
                CVRP.TamanoPoblacion = Integer.parseInt((String) jcbxTamanoPoblacion.getSelectedItem());
        setVisible(false);
    }

    private void onActivated(){
                jcbxNroIteraciones.setSelectedItem(new String(Integer.toString(CVRP.Iteraciones)));
                jcbxNroIntercambios.setSelectedItem(new String(Integer.toString(CVRP.Intercambios)));
                jcbxTamanoLista.setSelectedItem(new String(Integer.toString(CVRP.TamanoLista)));
                jcbxTamanoPoblacion.setSelectedItem(new String(Integer.toString(CVRP.TamanoPoblacion)));
    }
    /**
```

```
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new frmParametros().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
                int WIDTH = 500;
                int HEIGHT = 100;
    private javax.swing.JPanel jpnlDatos;
    private javax.swing.JPanel jpnlBotones;
    private javax.swing.JPanel jpnlNroIteraciones;
    private javax.swing.JPanel jpnlNroIntercambios;
    private javax.swing.JPanel jpnlTamanoLista;
    private javax.swing.JPanel jpnlTamanoPoblacion;
    private javax.swing.JComboBox jcbxNroIteraciones;
    private javax.swing.JComboBox jcbxNroIntercambios;
    private javax.swing.JComboBox jcbxTamanoLista;
    private javax.swing.JComboBox jcbxTamanoPoblacion;
    private javax.swing.JLabel jlblNroIteraciones;
    private javax.swing.JLabel jlblNroIntercambios;
    private javax.swing.JLabel jlbllTamanoLista;
    private javax.swing.JLabel jlblTamanoPoblacion;
    private javax.swing.JButton jbtnAceptar;
    // End of variables declaration

}
```

frmSolucion.java

A través de este formulario se presenta la información resultado del algoritmo diseñado.

```
package Interfaz;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class frmSolucion extends javax.swing.JFrame {

    /**
```

```
             *
             */
            private static final long serialVersionUID = 1L;
            /** Creates new form frmSolucion */
public frmSolucion() {
    initComponents();
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
// <editor-fold defaultstate="collapsed" desc=" Generated Code ">
private void initComponents() {
    jPanel1 = new javax.swing.JPanel();
    jPanel3 = new javax.swing.JPanel();
    jLabel2 = new javax.swing.JLabel();
    jlblNProblema = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jlblCosto = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jlblTiempo = new javax.swing.JLabel();
    jLabel7 = new javax.swing.JLabel();
    jlblNroIteraciones = new javax.swing.JLabel();
    jLabel9 = new javax.swing.JLabel();
    jlblNroIntercambios = new javax.swing.JLabel();
    jLabel11 = new javax.swing.JLabel();
    jlblCapacidad = new javax.swing.JLabel();
    jLabel13 = new javax.swing.JLabel();
    jPanel4 = new javax.swing.JPanel();
    jScrollPane1 = new javax.swing.JScrollPane();
    jtblSolucion = new javax.swing.JTable();
    jPanel2 = new javax.swing.JPanel();
    jbtnCerrar = new javax.swing.JButton("Cerrar");

    setDefaultCloseOperation(javax.swing.WindowConstants.HIDE_ON_CLOSE);
    setTitle("Solución");
    setBounds(new java.awt.Rectangle(100, 100, WIDTH,HEIGHT));
    setLocation(CVRP.WIDTH/2 - (WIDTH/2),CVRP.HEIGHT/2 - (HEIGHT/2));
    setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
    setAlwaysOnTop(true);
    setResizable(false);
    setMinimumSize(new java.awt.Dimension(WIDTH, HEIGHT));
```

```
jPanel1.setLayout(new java.awt.BorderLayout());

jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(" Datos de la solucion "));
jPanel1.setPreferredSize(new java.awt.Dimension(WIDTH, HEIGHT));
jPanel1.setRequestFocusEnabled(false);
jPanel3.setLayout(null);

jPanel3.setPreferredSize(new java.awt.Dimension(100, 100));
jPanel3.setRequestFocusEnabled(false);
jLabel2.setText("Problema:");
jPanel3.add(jLabel2);
jLabel2.setBounds(0, 0, 100, 16);

jlblNProblema.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
jlblNProblema.setText("Nombre Problema");
jPanel3.add(jlblNProblema);
jlblNProblema.setBounds(100, 0, 100, 16);

jLabel3.setText("Costo:");
jPanel3.add(jLabel3);
jLabel3.setBounds(0, 16, 100, 16);

jlblCosto.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jlblCosto.setText("1234567890");
jPanel3.add(jlblCosto);
jlblCosto.setBounds(100, 16, 70, 16);

jLabel5.setText("Tiempo:");
jPanel3.add(jLabel5);
jLabel5.setBounds(0, 32, 100, 16);

jlblTiempo.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jlblTiempo.setText("1234567890");
jPanel3.add(jlblTiempo);
jlblTiempo.setBounds(100, 32, 70, 16);

jLabel7.setText("Nro Iteraciones:");
jPanel3.add(jLabel7);
jLabel7.setBounds(0, 48, 100, 16);

jlblNroIteraciones.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jlblNroIteraciones.setText("Iteraciones");
jPanel3.add(jlblNroIteraciones);
jlblNroIteraciones.setBounds(100, 48, 70, 16);
```

```
jLabel9.setText("Nro Intercambios:");
jPanel3.add(jLabel9);
jLabel9.setBounds(0, 64, 100, 16);

jlblNroIntercambios.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jlblNroIntercambios.setText("NroIntercambios");
jPanel3.add(jlblNroIntercambios);
jlblNroIntercambios.setBounds(100, 64, 70, 16);

jLabel11.setText("Capacidad vh:");
jPanel3.add(jLabel11);
jLabel11.setBounds(0, 80, 100, 16);

jlblCapacidad.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jlblCapacidad.setText("Capacidad");
jPanel3.add(jlblCapacidad);
jlblCapacidad.setBounds(100, 80, 70, 16);

jLabel13.setForeground(java.awt.Color.blue);
jLabel13.setText("[ms]");
jPanel3.add(jLabel13);
jLabel13.setBounds(180, 32, 70, 16);

jPanel1.add(jPanel3, java.awt.BorderLayout.NORTH);

jPanel4.setLayout(new java.awt.BorderLayout());

jScrollPane1.setViewportView(jtblSolucion);

jPanel4.add(jScrollPane1, java.awt.BorderLayout.CENTER);

jPanel1.add(jPanel4, java.awt.BorderLayout.CENTER);

getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);

jPanel2.add(jbtnCerrar);
getRootPane().setDefaultButton(jbtnCerrar);
getContentPane().add(jPanel2, java.awt.BorderLayout.SOUTH);
pack();
//Capturadores de eventos de los botones
jbtnCerrar.addActionListener(new ActionListener(){
                        public void actionPerformed(ActionEvent evt){
                                setVisible(false);
```

```
                        }
                }
    );
                        addWindowListener(new java.awt.event.WindowAdapter(){
                                public void windowActivated(java.awt.event.WindowEvent evt){//Se ejecuta siempre que se muestra la forma
                                        onActivated();
                                }
                        });

}// </editor-fold>
/**
 * Siempre que se muestra el formulario, es necesario cargar la informacion que se encuentra
 * en el modelo para evitar desincronizacion, cuando ocurre el Show del formulario es necesario
 * tomar la informacion desde la solucion del problema
 */
public void onActivated(){
        this.jlblNProblema.setText("Solucion");
        this.jlblCosto.setText(String.valueOf(Math.round(CVRP.Solucion.Costo*10000)/10000.0));
        this.jlblTiempo.setText(String.valueOf(CVRP.Solucion.TiempoProceso));
        this.jlblNroIteraciones.setText(String.valueOf(CVRP.Iteraciones));
        this.jlblNroIntercambios.setText(String.valueOf(CVRP.Intercambios));
        this.jlblCapacidad.setText(String.valueOf(CVRP.Datos.CapacidadVehiculos()));
        jtblSolucion.setModel(new SolucionTableModel(CVRP.Datos,CVRP.Solucion ));
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new frmSolucion().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
        int WIDTH = 600;
        int HEIGHT = 400;
private javax.swing.JButton jbtnCerrar;
private javax.swing.JLabel jlblNProblema;
private javax.swing.JLabel jlblCosto;
private javax.swing.JLabel jlblTiempo;
private javax.swing.JLabel jlblNroIteraciones;
```

```
    private javax.swing.JLabel jlblNroIntercambios;
    private javax.swing.JLabel jlblCapacidad;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable jtblSolucion;
    // End of variables declaration

}
```

SolucionTableModel.java

Modelo de la vista información de la secuencia, a través de este modelo se despliega una tabla con la información de la secuencia de reparto.

```
package Interfaz;

import javax.swing.table.AbstractTableModel;
import Kernel.clsDatosProblema;
import Kernel.clsSolucion;

class SolucionTableModel extends AbstractTableModel{
        /**
         * Esta clase es el modelo que permite la informacion de los
         * clientes sea mostrada en una tabla de Swing
         */
        private static final long serialVersionUID = 1L;
        public SolucionTableModel(clsDatosProblema C, clsSolucion S){
                this.C = C;
                this.S = S;
        }

        public int getRowCount(){
                return S.Solucion.totLista;
        }
```

```
public int getColumnCount(){
        return 4;
}

public Object getValueAt(int row, int col){
        String Cadena;
        switch (col){
        case 0:
                Cadena = (S.Solucion.Lista[row]==0)?"Deposito":String.valueOf(S.Solucion.Lista[row]);
                return Cadena;
        case 1:
                Cadena = new Double(C.X(S.Solucion.Lista[row])).toString();
                return C.X(S.Solucion.Lista[row])==0.0?"0.0":Cadena;
        case 2:
                Cadena = new Double(C.Y(S.Solucion.Lista[row])).toString();
                return C.Y(S.Solucion.Lista[row])==0.0?"0.0":Cadena;
        case 3:
                Cadena = new Double(C.D(S.Solucion.Lista[row])).toString();
                return C.D(S.Solucion.Lista[row])==0.0?"0.0":Cadena;
        default:
                return "Nada";
        }
}

public void setValueAt(Object aValue, int rowIndex, int columnIndex){
}

public String getColumnName(int c){
        switch (c){
        case 0:
                return "Secuencia";
        case 1:
                return "Coordenada X";
        case 2:
                return "Coordenada Y";
        case 3:
                return "Demanda";
        default:
                return "UNKNOW";
        }
}

public boolean isCellEditable(int rowIndex, int columnIndex){
```

```
                        return true;
                }
                clsDatosProblema C;
                clsSolucion S;
}
```

frmSplash.java

Formulario de presentacion del programa de computadora diseñado.

```
package Interfaz;
import javax.swing.JLabel;
import java.awt.Dimension;
import java.awt.BorderLayout;
import javax.swing.ImageIcon;
import javax.swing.JWindow;
public class frmSplash extends JWindow
 {
                private static final long serialVersionUID = 1L;

                /** Creates new form frmSplash */
   public frmSplash() {
      JLabel l = new JLabel();
      l.setIcon(createImageIcon("/Interfaz/Imagenes/Splash.jpg"));
      getContentPane().add(l, BorderLayout.CENTER);
      pack();
      //Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
      Dimension labelSize = l.getPreferredSize();
      WIDTH=labelSize.width;
      HEIGHT=labelSize.height;
      setLocation(CVRP.WIDTH/2 - (WIDTH/2),CVRP.HEIGHT/2 - (HEIGHT/2));
      setVisible(true);
      labelSize = null;
   }
   /** Returns an ImageIcon, or null if the path was invalid. */
   protected static ImageIcon createImageIcon(String path) {
      java.net.URL imgURL = frmSplash.class.getResource(path);
      if (imgURL != null) {
         return new ImageIcon(imgURL);
      } else {
         System.err.println("Couldn't find file: " + path);
         return null;
      }
   }
```

```
    int WIDTH;
    int HEIGHT;
}
```