



PONTIFICIA **UNIVERSIDAD CATÓLICA** DEL PERÚ

Esta obra ha sido publicada bajo la licencia Creative Commons
Reconocimiento-No comercial-Compartir bajo la misma licencia 2.5 Perú.

Para ver una copia de dicha licencia, visite
<http://creativecommons.org/licenses/by-nc-sa/2.5/pe/>



PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

**ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO
META HEURÍSTICO GRASP QUE PERMITA RESOLVER EL
PROBLEMA DE RUTAS DE VEHÍCULOS CON CAPACIDAD**

Tesis para optar por el Título de Ingeniero Informático, que presenta el bachiller:

Joseph Gallart Suárez

ASESOR: Magíster Manuel Tupia Anticona

Lima, Julio del 2009

Resumen

El problema de ruteo de vehículos consiste en hallar un conjunto de rutas óptimas de reparto que permitan satisfacer la demanda de clientes. Desde el punto de vista algorítmico, son problemas de optimización combinatoria de alta complejidad debido a la gran cantidad de posibles soluciones existentes que se podrían establecer por las conexiones entre el punto de origen (base de los vehículos de reparto) y los puntos destino (clientes).

En este proyecto se busca implementar dos algoritmos meta heurísticos GRASP que permitan resolver el problema de ruteo de vehículos con capacidad uniforme considerando la demanda compartida en caso el cliente tenga mayor demanda que la capacidad del vehículo de reparto; además se realiza una mejora de la solución utilizando un algoritmo de búsqueda local 2-Opt.

Dentro de la construcción de la solución del primer algoritmo, se optimiza el espacio de embalaje del vehículo de reparto, esto se logra implementando un segundo algoritmo GRASP que permita resolver el problema de embalaje de paquetes en tres dimensiones tomando en consideración el orden de reparto de los destinatarios, el tamaño y peso de los paquetes.

En este proyecto se compara la solución obtenida por los dos algoritmos GRASP con una solución voraz, una solución aleatoria y una solución híbrida (voraz y aleatoria). Los resultados obtenidos comprueban que el algoritmo GRASP obtiene mejores soluciones que un algoritmo voraz, un algoritmo aleatorio y un algoritmo híbrido (voraz y aleatorio).



Este trabajo está dedicado a aquellas personas que tienen como gusto afín la investigación.



Agradezco a mis padres y a mi hermana por su apoyo constante e incondicional hacia la realización del presente proyecto.

TABLA DE CONTENIDO

INTRODUCCIÓN	1
CAPÍTULO 1: GENERALIDADES	3
1.1 Identificación de Problema	3
1.2 Objetivo General	4
1.3 Objetivos Específicos.....	5
1.4 Resultados Esperados	5
1.5 Marco Conceptual.....	6
1.5.1 Complejidad de un problema computacional.....	6
1.5.2 Optimización Combinatoria.....	6
1.5.3 Heurísticos y Meta Heurísticos.....	8
1.5.4 Algoritmo GRASP.....	11
1.5.5 Algoritmo 2-Opt	14
1.6 Estado del arte	14
1.6.1 Definición del problema del Agente Viajero (Traveler Salesman Problem)	14
1.6.2 Variante objeto de estudio: Vehicle Routing Problem VRP	15
1.6.3 Problema de Ruteo de Vehículos con Capacidad (Capacitated VRP- CVRP)19	
1.6.4 Métodos exactos y aproximados ya existentes	20
1.6.5 Aplicaciones existentes	25
CAPÍTULO 2: METODOLOGÍA Y ARQUITECTURA	28
2.1 Definición de la metodología de la solución.....	28
2.1.1 Fases del XP	28
2.2 Arquitectura de la Información	30
2.3 Arquitectura del Sistema.....	32
CAPÍTULO 3: DISEÑO	34
3.1 Definición de interfaz gráfica	34
3.2 División modular	35
CAPÍTULO 4: CONSTRUCCIÓN DE LOS ALGORITMOS.....	44
4.1 Algoritmo generador de grafo temporal	44
4.1.1 Descripción.....	44
4.1.2 Datos de Entrada.....	44
4.1.3 Dato de Salida	45
4.1.4 Algoritmo	45
4.1.5 Ejemplo y Comentarios	46
4.2 Algoritmo generador de grafo virtual.....	48
4.2.1 Descripción.....	48
4.2.2 Datos de Entrada.....	48

4.2.3 Dato de Salida	49
4.2.4 Algoritmo	49
4.2.5 Ejemplo y Comentarios	49
4.3 Algoritmo Primero el Mejor (Best First Search)	51
4.3.1 Descripción.....	51
4.3.2 Datos de Entrada.....	51
4.3.3 Dato de Salida	51
4.3.4 Función Objetivo BFS.....	51
4.3.5 Algoritmo	51
4.3.6 Ejemplo y Comentarios	52
4.4 Algoritmo GRASP que resuelve el problema de ruteo de vehículos con capacidad uniforme considerando la demanda compartida	54
4.4.1 Descripción.....	54
4.4.2 Datos de Entrada.....	54
4.4.3 Dato de Salida	54
4.4.4 Función Objetivo CVRP.....	54
4.4.5 Función de Costo CVRP	55
4.4.6 Algoritmo	56
4.4.7 Ejemplo y Comentarios	57
4.5 Algoritmo GRASP que resuelve el problema de embalaje de paquetes en tres dimensiones.....	61
4.5.1 Descripción.....	61
4.5.2 Datos de Entrada.....	61
4.5.3 Dato de Salida	62
4.5.4 Función Objetivo BPP3D	62
4.5.5 Función de Costo BPP3D.....	63
4.5.6 Algoritmo	63
4.5.7 Ejemplo y Comentarios	65
4.6 Algoritmo de mejoría 2-OPTCVRP que optimiza la solución generada por el algoritmo GRASPCVRP	68
4.6.1 Descripción.....	68
4.6.2 Datos de Entrada.....	68
4.6.3 Dato de Salida	68
4.6.4 Algoritmo	68
4.6.5 Ejemplo y Comentarios	69
CAPÍTULO 5: RESULTADOS DE LA EXPERIMENTACIÓN NUMÉRICA.....	72
5.1 Objetivo de la experimentación numérica	72
5.2 Resultados.....	73
CAPÍTULO 6: OBSERVACIONES, CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS.	75
6.1 Observaciones	75
6.2 Conclusiones.....	76
6.3 Recomendaciones y trabajos futuros	76
BIBLIOGRAFIA	78

Índice de Figuras

Un ejemplo del método de descomposición.....	9
Universo del problema y lista de candidatos.....	10
Elección del elemento dentro del conjunto de candidatos.....	10
Grafo en donde el cuadrado representa el depósito y los demás nodos representan los clientes a visitar.....	16
Diagrama de las variantes del problema de ruteo de vehículos.....	18
Google Maps.....	25
Software Grafos.....	27
Grafo con 5 nodos y 6 aristas.....	30
Base de datos del núcleo del sistema (algoritmo).....	33
Vista principal del Software JaMi2o Soft.....	34
Módulo de carga del Software JaMi2o Soft.....	35
Ventana de carga de vehículos.....	35
Módulo del algoritmo.....	36
Parámetros del algoritmo.....	37
Módulo de resultados.....	37
Ventana de resultados.....	38
Ventana de embalaje del vehículo con los paquetes embalados.....	39
Módulo de experimentación numérica.....	39
Ventana de experimentación numérica (Grafo).....	40
Ventana de experimentación numérica (Vehículo).....	41
Ventana de experimentación numérica (Paquete y destinatario).....	42
Ventana de experimentación numérica (Algoritmos).....	43
Archivo de destinatarios.....	46
Grafo utilizado donde se muestra los nodos y aristas relacionados a un distrito, el nodo B va ser el punto de partida (o nodo inicio) de los vehículos.....	46
Ubicación de la avenida 8 de Octubre con número de dirección 121 y la ubicación de los nodos N1 y N2.....	47
Creación del nuevo nodo temporal entre N1 y N2 (Grafo Temporal).....	47
Grafo Temporal donde está incluido los nodos temporales (T1, T2, T3, T4, T5, T6 y T7) y el nodo base (B).....	48
Grafo Temporal donde está incluido los nodos temporales (T1, T2, T3, T4, T5, T6 y T7) y el nodo base (B).....	50
Grafo virtual formado por los nuevos nodos y el nodo base.....	50
Grafo Temporal donde está incluido los nodos temporales (T1, T2, T3, T4, T5, T6 y T7) y el nodo base (B).....	52
Ruta generada (Nodos 154-155-72-44-158-141-16-22-18-17) desde el nodo base (B) al nodo temporal T1.....	53

Grafo virtual formado por los nuevos nodos y el nodo base.....58
Grafo virtual formado por los nuevos nodos y el nodo base.....70



Índice de Tablas

Datos de los productos por enviar.....	58
Solución.....	61
División de paquetes.....	65
Solución de embalajes... ..	68
Rutas de vehículos... ..	70
Nuevas rutas de vehículos.....	71
Acrónimos de los algoritmos simulados.....	72
Resumen de resultados de los algoritmos.....	73



Índice de Algoritmos

Algoritmo GRASP	13
Algoritmo GRASP Construcción	13
Algoritmo GRASP Mejoría	14
Algoritmo generador de grafo temporal	45
Algoritmo generador de grafo virtual.....	49
Algoritmo Best First Search.....	51
Algoritmo GRASPCVRP... ..	56
Algoritmo GRASPCVRP Construcción	57
Algoritmo GRASPBPP3D... ..	63
Algoritmo GRASPBPP3D Construcción.....	64
Algoritmo 2-OPTCVRP.....	69



INTRODUCCIÓN

El problema de ruteo de vehículos, conocido en la literatura como Vehicle Routing Problem, consiste en hallar un conjunto de rutas óptimas de reparto para flotas de medios de transportes (autos, camiones, motos) que permitan satisfacer la demanda de reparto de un grupo de clientes.

Este tipo de problema de optimización combinatoria no tiene una solución exacta debido a la gran cantidad de posibles soluciones existentes por las conexiones entre el punto de origen (base de los vehículos de reparto) y los puntos destino (clientes). Forma parte de la clasificación algorítmica del tipo NP.

Una variante particular del escenario explicado es aquella en la que se agrega como restricción la capacidad del vehículo de reparto, tomando como consideración adicional la demanda compartida: en caso el cliente tenga mayor demanda que la capacidad del vehículo de reparto, un punto destino puede ser visitado por más de un vehículo de reparto.

En este proyecto se implementan dos algoritmos GRASP que permiten resolver este problema y el problema de embalaje de paquetes en tres dimensiones; además, se busca mejorar la solución optimizándola a través de un algoritmo de búsqueda local 2-Opt.

Dentro de la construcción de la solución del primer algoritmo, se optimiza el espacio de embalaje del vehículo de reparto; para ello se implementa un segundo algoritmo GRASP que permita resolver el problema de embalaje de paquetes en tres dimensiones tomando en consideración el orden de reparto de los destinatarios, el tamaño y peso de los paquetes de los destinatarios.

La metodología a seguir para la construcción del sistema de prueba de la idoneidad de los algoritmos será XP (eXtreme Programming).

Para efectos de la experimentación numérica y la medida de la calidad de los algoritmos diseñados, se compara la solución obtenida por los dos algoritmos GRASP con tres algoritmos distintos adicionales: una solución totalmente voraz, una solución totalmente aleatoria y una solución híbrida (voraz y aleatoria).



CAPÍTULO 1: GENERALIDADES

En este capítulo se explicará los conceptos necesarios para poder comprender el problema de ruteo de vehículos con capacidad uniforme, además se explicará los métodos y soluciones alternativas que existen en la actualidad.

1.1 Identificación de Problema

En la actualidad el problema de distribuir productos a partir de un depósito original (punto de origen) y una cantidad de clientes con una demanda por atender, juega un papel importante en empresas comercializadoras ya que planificar adecuadamente estos envíos puede significar considerables ahorros logísticos y sobretodo en costos como: el consumo de combustible, horas hombre, entre otros; que ayudarán a una mejor rentabilidad para los negocios hoy en día. Son por estas causas que surge el problema de ruteo de vehículos (de la literatura Vehicle Routing Problem o VRP).

Este problema consiste en generar rutas de reparto dado una cantidad de clientes por atender, un conjunto de vehículos de reparto y un punto de origen, permitiendo minimizar ciertos factores que ayuden a la empresa a obtener beneficios; estos pueden ser: minimizar el tiempo de reparto, maximizar el ahorro de combustible en los vehículos, minimizar la cantidad

de vehículos de reparto, lo cual llevaría a obtener menores costos y por lo tanto obtener beneficios y una mejor calidad de servicio e imagen. Asimismo, presenta una serie de variantes como es el caso de incluir: la capacidad de un vehículo, espacios de tiempo de entrega, incluir varios puntos de origen (que vendrían a ser los depósitos), entre otros.

Este proyecto se enfoca en la variante en la cual se agrega como condición la capacidad uniforme a cada vehículo de reparto, debiendo satisfacer la demanda de los clientes. Por lo tanto como objetivo principal será minimizar la cantidad de vehículos y el tiempo de viaje (reduciendo así el gasto de combustible, choferes, horas hombre), siempre y cuando se respete que cada reparto no pueda exceder la capacidad que tiene un vehículo de reparto.

Se tomará como consideración adicional la demanda compartida en caso el cliente tenga mayor demanda que la capacidad del vehículo de reparto, además se tratará en lo posible llevar la mayor cantidad de paquetes de los destinatarios en los vehículos, respetando el orden de reparto de los destinatarios en el vehículo.

Este tipo de problemas no tienen una solución exacta porque encontrar la ruta mínima entre dos puntos en un mapa que contiene miles de conexiones llevaría a realizar cálculos computacionales muy elevados. Por ello este tipo de problemas son clasificados como problemas de clase NP, que significa problemas intratables. Si bien se pueden resolver, sus soluciones no son exactas sino aproximadas.

Por lo tanto se propone utilizar dos algoritmos Meta Heurísticos GRASP, ya que este tipo de algoritmos buscan una posible solución y luego la optimizan, dando así una muy buena solución.

1.2 Objetivo General

El presente proyecto de tesis pretende desarrollar e implementar dos algoritmos GRASP que permitan hallar una óptima solución al problema de ruteo de vehículos con capacidad uniforme considerando la demanda

compartida en caso el cliente tenga mayor demanda que la capacidad del vehículo de reparto.

1.3 Objetivos Específicos

- A. Estudiar la aplicabilidad, ventajas y desventajas de los Algoritmos Meta Heurísticos a la solución del problema de ruteo de vehículos con capacidad uniforme considerando la demanda compartida.
- B. Implementar un algoritmo GRASP Construcción que resuelva el problema de ruteo de vehículos con capacidad uniforme considerando la demanda compartida.
- C. Implementar un algoritmo GRASP Mejoría 2-Opt que permita optimizar la solución inicial al problema de ruteo de vehículos con capacidad uniforme considerando la demanda compartida.
- D. Implementar un algoritmo GRASP Construcción que resuelva el problema de embalaje de paquetes en tres dimensiones.
- E. Comparar la solución obtenida por el algoritmo GRASP y la fase de mejora 2-Opt contra un algoritmo totalmente voraz, un algoritmo totalmente aleatorio y un algoritmo híbrido (voraz y aleatorio), estos algoritmos serán simulados en base al algoritmo GRASP.

1.4 Resultados Esperados

- A. Se logrará conocer las ventajas y desventajas de los Algoritmos Meta Heurísticos para la solución del problema de ruteo de vehículos con capacidad uniforme, estas ventajas y desventajas se pueden comprobar mediante pruebas probabilísticas y estadísticas.
- B. Implementar una solución factible para el problema de ruteo de vehículos con capacidad uniforme considerando la demanda compartida.
- C. Implementar una solución óptima (mejorando la solución factible) para el problema de ruteo de vehículos con capacidad uniforme considerando la demanda compartida.
- D. Implementar una solución factible para el problema de embalaje de paquetes en tres dimensiones.

- E. Probar que los dos algoritmos GRASP y la fase de mejora 2-Opt obtienen mejores soluciones que un algoritmo totalmente voraz, un algoritmo totalmente aleatorio y un algoritmo híbrido (voraz y aleatorio) para el problema de ruteo de vehículos con capacidad uniforme considerando la demanda compartida, esto se comprueba mediante la experimentación numérica.

1.5 Marco Conceptual

Es necesario desarrollar los conceptos de complejidad computacional de un problema, optimización combinatoria, algoritmos heurísticos, algoritmos meta heurísticos y en particular el algoritmo GRASP.

1.5.1 Complejidad de un problema computacional

La complejidad de un problema computacional viene a ser los recursos que intervienen en el cálculo para resolver un problema determinado [PAPADIMITRIOU, 1982].

Se clasifican en dos tipos:

- Complejidad P

Se denota con complejidad P a los problemas que pueden ser resueltos por algoritmos que toman un tiempo polinomial en resolverlos [PAPADIMITRIOU, 1982].

- Complejidad NP

Se denota con complejidad NP a los problemas que no pueden ser resueltos por algoritmos que toman un tiempo polinomial, es por ello que los resultados no son exactos sino aproximados [PAPADIMITRIOU, 1982].

1.5.2 Optimización Combinatoria

La Optimización Combinatoria es una rama de las Ciencias Matemáticas que plantea la construcción de métodos que permitan realizar combinaciones y ordenamientos a grupos de elementos, es decir alteraciones de las posiciones de los elementos dentro del universo [PAPADIMITRIOU, 1982]. Se utiliza este

nombre para unificar términos que se refieren a la programación entera, la teoría de grafos, programación dinámica, entre otros. Estas áreas son de gran importancia ya que existen problemas prácticos que se pueden formular y solucionar con la optimización combinatoria [CHRISTOFIDES, 1978].

El objetivo es conseguir patrones adecuados que permitan resolver determinados tipos de problema de complejidad NP o NP difícil [PAPADIMITRIOU, 1982].

Un problema de optimización combinatoria se puede expresar como un sistema de ecuaciones y expresiones matemáticas relacionadas que describen la esencia del problema: [HILLIERLIEB, 1997]

En su forma más general sería de la siguiente manera:

Optimizar $f(x)$

Sujeto a:

$$g_i(x) > 0 \text{ para } i:1..m$$

$$h_j(x) > 0 \text{ para } j:1..n$$

Donde:

- $f(x)$: es conocida como la función objetivo (o mérito) y representa un valor que debe ser optimizado, ya sea maximizándolo o minimizándolo.
- $g(x)$ y $h(x)$: son denominadas las restricciones del problema y especifican las condiciones que debe poseer toda solución viable para el mismo [PAPADIMITRIOU, 1982].

Estos tipos de problemas se pueden resolver en dos grandes ramas:

- Como problemas de programación lineal:
Un problema de programación lineal consta de una función objetivo lineal por maximizar o minimizar, sujeta a ciertas restricciones, en la forma de igualdades o desigualdades lineales [HILLIERLIEB, 1997]. Son problemas que cuentan con una solución exacta, para ello se usan algoritmos exactos.
- Como problemas de optimización:

Un problema de optimización son problemas similares a los de programación lineal, salvo que son problemas que no cuentan con una solución exacta debido a su complejidad no polinomial, para ello se usan algoritmos heurísticos y meta heurísticos que permitan hallar una buena solución.

1.5.3 Heurísticos y Meta Heurísticos

Las técnicas heurísticas y meta heurísticas tratan de métodos o algoritmos exploratorios durante la resolución de problemas en los cuales las soluciones se descubren por la evaluación del progreso logrado en la búsqueda de un resultado final. Estas técnicas son empleadas en una gran cantidad de disciplinas y áreas del conocimiento y su finalidad es la de entregar soluciones que satisfagan al máximo el problema al cual se pretenden encontrar salidas [U.REPUBLICA, 2007]. Ofrecen soluciones lo suficientemente buenas a problemas que, hasta hoy, no presentan solución exacta.

Una ventaja de su uso es el hecho de encontrar soluciones casi óptimas a un coste computacional razonable.

Las características de estas técnicas son:

- Son métodos o algoritmos exploratorios.
- Son aplicados a la resolución de problemas en los cuales las soluciones se descubren por la evaluación progresiva de un conjunto de elementos.
- Forma candidatos que forman parte de la solución final.
- Son búsquedas no exhaustivas [IEEE, 2003].
- Se pueden hacer estudios estadísticos sobre los resultados.
- El tiempo de los resultados es aceptable.

Según Moscato (1996) los factores que favorecen y justifican el uso de heurísticas son los siguientes:

- Cuando no se necesita una solución exacta.
- Cuando no existe un método exacto de la resolución o de existir, requiere mucho tiempo de procesamiento.
- Cuando los datos son poco fiables o poco variados.

- Cuando aparecen limitaciones de tiempo y de proceso computacional (hardware).
- Como paso intermedio en la aplicación de otro tipo de algoritmos.

Dentro de los tipos de heurísticas se tienen los siguientes:

- Métodos constructivos:

Se basan en el paulatino aumento de los componentes individuales a los conjuntos solución de los problemas. Un ejemplo de ello son los algoritmos voraces.

Dentro de las características de este método se tienen los siguientes:

- Son Evolutivos, es decir mediante una secuencia de pasos se va formando el conjunto solución.
- Usan varios criterios de selección para formar el conjunto solución.

- Métodos de descomposición:

Hacen uso de divisiones sucesivas del problema que pretenden resolver [BALL, 1981] (técnica divide y vencerás), tal como se muestra en la figura 1.1.

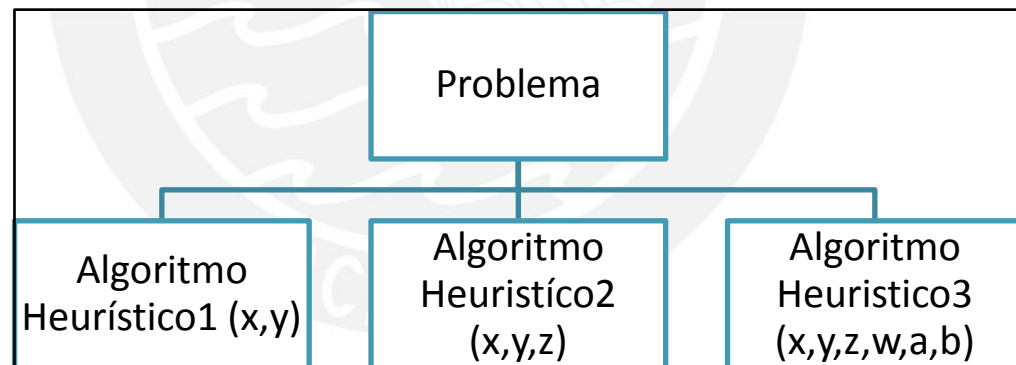


Figura 1.1: Un ejemplo del método de descomposición.

- Métodos de reducción:

Características:

- Forman una lista de candidatos para pertenecer al conjunto solución, tal como se muestra en la figura 1.2.

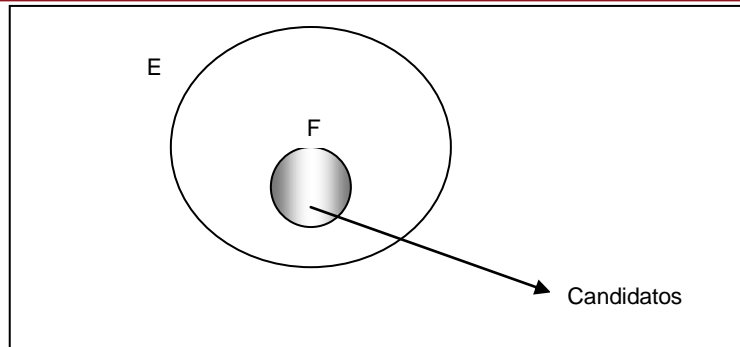


Figura 1.2: Universo del problema y lista de candidatos.

Donde:

- E es el universo del problema.
 - F es el conjunto de candidatos.
- Seleccionan un elemento de esa lista de candidatos, tal como se muestra en la figura 1.3.



Figura 1.3: Elección del elemento dentro del conjunto de candidatos.

Donde:

- F es el conjunto de candidatos.
- X es el elemento seleccionado de la lista de candidatos, en el caso de las heurísticas siempre se selecciona el mejor elemento.

Los algoritmos meta heurísticos surgen de la necesidad de superar algunas limitaciones de los algoritmos heurísticos, como son:

- Voracidad:

Un algoritmo heurístico siempre escoge el mejor candidato para formar parte de la solución, aquel que tenga mejor valor en la función de mérito.

- Miopía:

Un algoritmo heurístico siempre elige una única e inmodificable solución, por lo tanto no analiza más allá de los efectos de haber seleccionado un elemento como parte de la solución.

Por lo tanto los algoritmos meta heurísticos no manejan uno sino varios candidatos para formar parte de la solución final, además no se escoge necesariamente el mejor elemento para formar parte de la solución (ya que se verá más adelante que algunos algoritmos meta heurísticos eligen una solución al azar).

1.5.4 Algoritmo GRASP

El algoritmo GRASP viene de las siglas “Greedy Randomize Adaptive Search Procedure”, significa que son procedimientos de búsqueda voraces, adaptativos y aleatorios [FEO; RESENDE, 1995].

Son procedimientos de búsqueda que no necesariamente evalúan a todos los elementos del problema, son voraces ya que eligen a los mejores candidatos que cumplan ciertas propiedades, son adaptativos porque cada generación de candidatos puede variar de acuerdo a las condiciones necesarias del problema y son aleatorias ya que de la lista de candidatos se escoge un elemento al azar.

El algoritmo GRASP presenta dos fases:

1.5.4.1 Fase de Construcción: (Procedimiento Voraz)

El objetivo de esta etapa es construir una solución relajando el criterio goloso.

Para ello se siguen los siguientes pasos: [FEO; RESENDE, 1995]

- Se determina un rango de viabilidad de acuerdo a la función objetivo (mérito).
- Se construye iterativamente una solución factible, añadiendo un elemento en cada paso.
- En cada iteración, la elección del próximo elemento a ser añadida a un conjunto de candidatos es determinada por la función voraz, el rango de

viabilidad y los siguientes factores: la constante de relajación (α), el mejor valor de la función de mérito (β) y el peor valor de la función de mérito (τ).

- De este conjunto de candidatos se extrae de manera aleatoria un elemento, formando así una posible solución.

En formulación matemática la lista de candidatos restringida sería de la siguiente manera: (en caso se quiera minimizar la función de mérito)

$$\beta := \text{Mejor } \{c(S_i) : S_i \in E\}$$

$$\tau := \text{Peor } \{c(S_i) : S_i \in E\}$$

$$\text{RCL} = \{S_i \in E : \beta \leq c(S_i) \leq \beta + \alpha(\tau - \beta)\}$$

En donde:

- c : es la función de mérito.
- β : el mejor valor de la función de mérito.
- τ : el peor valor de la función de mérito.
- α : la constante de relajación, esta constante adquiere un valor entre 0 y 1, si esta constante es 1 quiere decir que la construcción va ser totalmente aleatoria, en cambio si es 0 quiere decir que la construcción va ser totalmente voraz.
- S : conjunto de elementos que están dentro del rango de viabilidad.
- RCL: lista de candidatos restringida.

1.5.4.2 Fase de Optimización: (Búsqueda local)

El objetivo de esta etapa es optimizar la solución generada en la fase de construcción, para ello lo que se realiza es una búsqueda iterativa reemplazando sucesivamente la solución actual por otra que este en el conjunto de soluciones vecinas, esta selección se puede hacer de dos tipos:

- Estrategia "Best improvement": se busca la mejor solución entre el conjunto de soluciones vecinas.
- Estrategia "First improvement": se toma la primera solución entre el conjunto de soluciones vecinas.

Esta fase termina cuando no encuentra una mejor solución en el conjunto de soluciones vecinas.

Existen diversos algoritmos que permiten optimizar la solución inicial como es el caso de: el operador λ -intercambio, el algoritmo de Lin-Kernighan, el algoritmo 2-Opt, Operadores de Van Breedam, GENI y GENIUS, entre otros [U.REPUBLICAB, 2007]. En este proyecto se realizará la optimización usando el algoritmo 2-Opt, dado que brinda soluciones aceptables a un bajo costo computacional.

A continuación se presenta los algoritmos básicos: procedimiento GRASP, procedimiento voraz, la etapa de mejoría y la búsqueda local.

1	Procedimiento GRASP (Datos del problema)
2	Mientras <no se cumpla el criterio de parada> hacer
3	Se construye una solución usando el procedimiento voraz
4	Se realiza una búsqueda local para mejorar la solución construida
5	Si el costo(solucionActualOptimizada) < costo(mejorSolucionEncontrada) Entonces
6	actualizar(mejorSolucionEncontrada, solucionActualOptimizada)
7	Fin Si
8	Fin Mientras
9	Retornar mejorSolucionEncontrada
10	Fin Procedimiento GRASP

Algoritmo 1.1: Algoritmo GRASP

1	Procedimiento Voraz()
2	Inicializar s (s es una solución parcial en este caso)
3	Mientras <no se cumpla el criterio de parada> hacer
4	RCL \leftarrow generarListaCandidatosRestringida()
5	X \leftarrow seleccionAleatoria(RCL)
6	s \leftarrow s U x
7	actualizarFuncionVoraz(s)
8	Fin Mientras
9	Retornar s
10	Fin Procedimiento Voraz

Algoritmo 1.2: Algoritmo GRASP Construcción

1	Procedimiento búsquedaLocal (MejorSoluciónEncontrada)
2	Repetir
3	nuevaMejorSolucion<-Mejorar(nuevaMejorSolucion, mejorSolucionEncontrada)
4	Hasta no Haya Mejora Posible
5	Retornar nuevaMejorSolucion
6	Fin búsquedaLocal

Algoritmo 1.3: Algoritmo GRASP Mejoría

1.5.5 Algoritmo 2-Opt

La idea es determinar dos conjuntos de aristas $\{x_1, \dots, x_k\}$ e $\{y_1, \dots, y_k\}$ tales que su intercambio disminuya el costo de la solución. Los arcos x deben ser parte de la ruta, ambos conjuntos deben ser disjuntos y, además, eliminar los arcos “ x ” y agregar los arcos “ y ” debe formar una ruta cerrada [U.REPUBLICAB, 2007].

Este procedimiento se repite hasta terminar en un óptimo local, es decir cuando no es posible encontrar intercambios que mejoren la solución.

1.6 Estado del arte

Es necesario conocer los conceptos de TSP, VRP, sus variantes, los métodos exactos y aproximados ya existentes además de las aplicaciones con las que se ha pretendido resolver estas variantes del problema.

1.6.1 Definición del problema del Agente Viajero (Traveler Salesman Problem)

El problema del agente viajero (TSP) consiste en dado un conjunto de ciudades y un viajero, visitar todas las ciudades pasando por ellas sólo una vez, regresando al punto de origen y haciendo el menor tiempo posible [PAPADIMITRIOU, 1982].

1.6.1.1 Formulación:

Sea un grafo $G (V,E)$ en donde: [U.MÁLAGA, 2009]

- $V = \{v_0, v_1, v_2 \dots v_n\}$ es un conjunto de vértices (nodos) del grafo, en donde v_0 representa el origen del viajero y los demás vértices representan las ciudades.
- $E =$ conjunto de aristas del grafo.

Además:

- d_{ij} : es la longitud de una arista (conformada por el nodo i y j).
- A : es la distancia entre ciudades, es decir va estar conformado por d_{ij} en donde i y $j \in N$, si el TSP es simétrico entonces la distancia de ir de i a j va ser igual a la distancia de ir de j a i $d_{ij} = d_{ji}$

El objetivo es minimizar: $\min \sum_{i=1}^{n-1} \sum_{j=i+1}^n (d_{ij} * X_{ij})$

Donde:

- $X_{ij} = 1$ si la ruta formada incluye a la arista formada por los nodos i y j o 0 si no la incluye.
- $\sum_{k=1}^{i-1} X_{ki} + \sum_{k=i+1}^n X_{ik} = 2$, para todo $i \in V$.
- $\sum_{ij \in Z} X_{ij} < |Z|$, donde $\emptyset \subset Z \subset V$

1.6.2 Variante objeto de estudio: Vehicle Routing Problem VRP

Luego de la aparición del problema del agente viajero aparece el problema de ruteo de vehículos (Vehicle Routing Problem "VRP"), a grandes rasgos consiste en, dado un conjunto de clientes (en el problema TSP eran ciudades) por atender, depósitos dispersos geográficamente y una flota de vehículos de reparto (en el problema de TSP era solo un viajero), determinar un conjunto de rutas de costo mínimo que comiencen y terminen en los depósitos, de modo que los vehículos de reparto visiten a los clientes por atender sólo una vez, tal como se muestra en la figura 1.4.

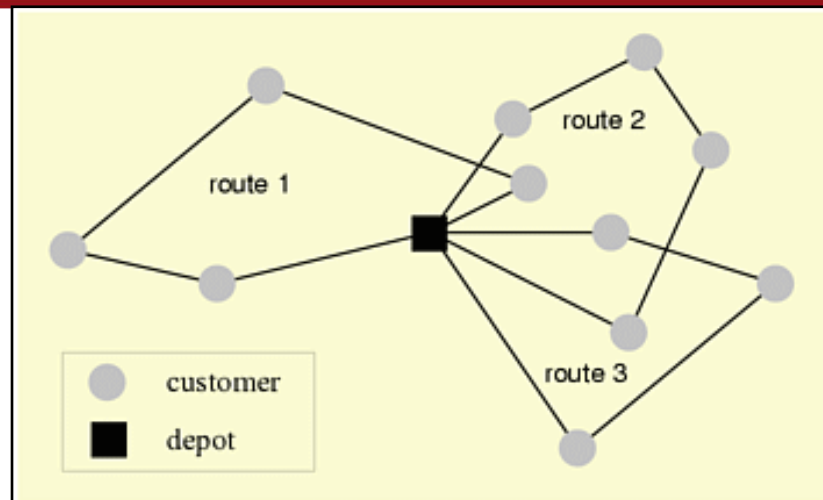


Figura 1.4 Grafo en donde el cuadrado representa el depósito y los demás nodos representan los clientes a visitar [SCHULZE, 2007].

En la figura citada sólo existe un depósito, sin embargo el problema de ruteo de vehículos puede aceptar más de uno.

Según Olivera (2004, p5) Los Problemas de Ruteo de Vehículos son Problemas de Optimización Combinatoria y pertenecen, en su mayoría, a la clase NP-Difícil. La motivación académica por resolverlos radica en que no es posible construir algoritmos que en tiempo polinomial resuelvan cualquier instancia del problema (a no ser que $P = NP$).

El problema de ruteo de vehículos crece naturalmente como un problema central en el campo del transporte, distribución y logística. En algunos sectores del mercado, el transporte significa un alto porcentaje de valor añadido a las mercancías. Por lo tanto, la utilización de los métodos automatizados para el transporte da lugar a menudo a los ahorros significativos que se extienden a partir de 5% hasta el 20% en los costes totales, según lo divulgado en [P. TOTH; D. VIGO, 2001].

El problema de ruteo de vehículos matemáticamente se puede definir de la siguiente manera: [U.MÁLAGA, 2009]

Sea un grafo $G(V,E)$ en donde:

- $V = \{v_0, v_1, v_2 \dots v_n\}$ es un conjunto de vértices (nodos) del grafo, en donde v_0 representa el depósito y los demás vértices representan a los clientes.
- $E =$ conjunto de aristas del grafo.

Además:

- $V' = V / \{v_0\}$ se va a usar como el sistema de n ciudades.
- $A = \{(v_i, v_j) / v_i, v_j \in V, i \neq j\}$, quiere decir que A va a significar el conjunto de vértices (arcos) que pertenezcan al grafo y que sean disjuntos.
- C va a ser la matriz no negativa de costos o distancias c_{ij} entre los clientes v_i y v_j .
- d es un vector de demanda de clientes.
- R_i es la ruta del vehículo i .
- m es el número de vehículos (todos idénticos). Una ruta es asignada a cada vehículo.
- Cuando $c_{ij} = c_{ji}$ para todo $(v_i, v_j) \in A$, se dice que el problema es simétrico y es común reemplazar A por el conjunto:

$$E = \{(v_i, v_j) / v_i, v_j \in V, i < j\}.$$
- Cada vértice (v_i) en V' es asociado a una cantidad q_i de alguna mercancía que se entregará por un vehículo.

Por lo tanto el problema de ruteo de vehículos consiste en determinar un conjunto de “ m ” rutas para “ m ” vehículos en donde el coste total sea mínimo, comenzando y terminando en un depósito, tal que cada vértice V' sea visitado exactamente una sola vez.

1.6.2.1 Formulación

Matemáticamente se plantearía su formulación de la siguiente manera:

$$b(V) = \lceil \sum_{v_i \in V'} d_i / C \rceil$$

En el caso de estudio del presente proyecto, no se va a considerar el tiempo de servicio que se necesita para descargar toda la mercancía, dado que se va

a considerar otros factores de mayor importancia como las dimensiones y peso de los paquetes a embalar en el vehículo, por lo tanto el costo de una ruta estaría definido directamente de la siguiente manera:

$$C(R_i) = \sum_{i=0}^m c_{i,i+1}$$

El costo del problema de ruteo de vehículos vendría a estar dado por:

$$F_{VRP}(S) = \sum_{i=1}^m C(R_i)$$

Las características de los clientes, depósitos y vehículos así como las diferentes restricciones operativas sobre las rutas, dan lugar a diferentes variantes del problema [U.REPÚBLICAB, 2007].

1.6.2.2 Variantes

Un esquema de las variantes del problema de ruteo de vehículos sería tal como se muestra en la figura 1.5 [U.MÁLAGA, 2009].

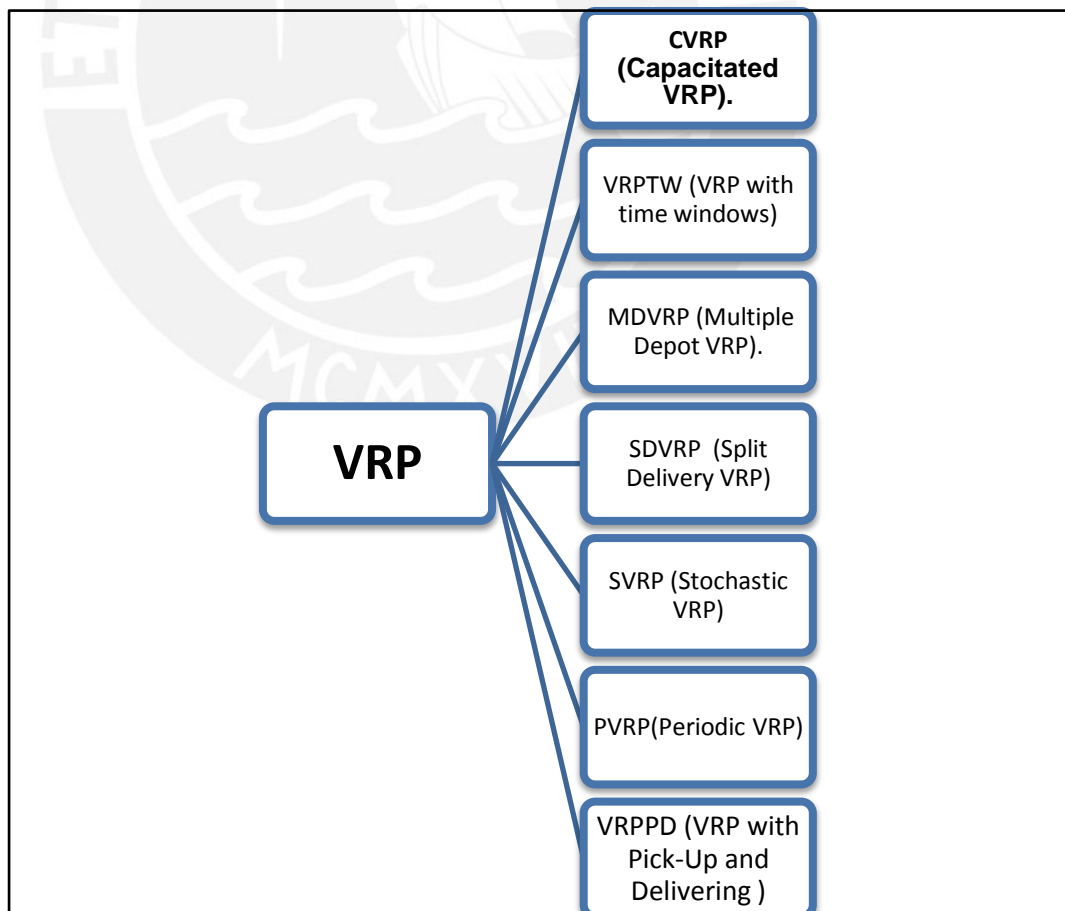


Figura 1.5 Diagrama de las variantes del problema de ruteo de vehículos.

- Cada vehículo tiene un límite de capacidad (Capacitated VRP- CVRP).
- Cada cliente tiene que ser provisto dentro de cierto espacio de tiempo (VRP with time windows - VRPTW).
- El proveedor usa más de un depósito para proveer a los clientes (Multiple Depot VRP - MDVRP).
- Los clientes pueden ser servidos (entrega de mercancía) por más de un vehículo (Split Delivery VRP - SDVRP).
- Algunos valores como el número de clientes, la demanda, tiempo de servicio, tiempo de viaje puede ser aleatorio (Stochastic VRP - SVRP).
- Las entregas se pueden hacer en algunos días específicos (Periodic VRP - PVRP).
- Los clientes pueden retornar alguna mercancía al depósito (VRP with Pick-Up and Delivering - VRPPD).

Este proyecto se enfoca en la variante que agrega como condición la capacidad, además se tendrá en consideración un caso particular de la variante SDVRP al considerar que el cliente puede ser servido por varios vehículos en caso la demanda del cliente sea mayor a la capacidad del vehículo.

1.6.3 Problema de Ruteo de Vehículos con Capacidad (Capacitated VRP-CVRP)

El Problema de Ruteo de Vehículos tradicional consiste en agregar a todos los vehículos una capacidad uniforme, de tal manera que deban satisfacer la demanda de los clientes [U.MÁLAGA, 2009].

1.6.3.1 Objetivo

El objetivo es reducir al mínimo la cantidad de vehículos así como la suma del tiempo recorrido en realizar el reparto, además la demanda total de las materias para cada ruta no debe exceder la capacidad del vehículo de reparto.

1.6.3.2 Viabilidad

Una solución es factible si la cantidad total asignada a cada ruta no excede la capacidad del vehículo de reparto.

1.6.3.3 Formulación

Matemáticamente la formulación del problema es básicamente la misma vista en el punto 1.6.2.1, agregando la siguiente restricción:

Sea Q la capacidad de un vehículo de reparto, por lo tanto la demanda total de todos los clientes en una ruta (R_i) no debe exceder la capacidad del vehículo:

$$\sum_{i=1}^m d_i \leq Q$$

En caso exceda la capacidad, se tendrá que enviar otro vehículo para que pueda satisfacer la demanda del cliente (un caso particular de la variante SDVRP que se tomará en cuenta).

1.6.4 Métodos exactos y aproximados ya existentes

Para resolver estos tipos de problemas (VRP, VRPTW, PVRP, etc) se han desarrollado distintos algoritmos que se pueden dividir en algoritmos exactos y algoritmos aproximados.

1.6.4.1 Métodos exactos

Dentro de los métodos exactos que resuelven problemas de tipo VRP se tienen:

- Ramificación y acotación (Branch and bound) [RAMOS, 2007]
 - Es un método que usa la técnica de “divide y vencerás”.
 - Divide (ramifica) el conjunto de soluciones enteras en subconjuntos disjuntos cada vez menores.
 - Determina (acota) el valor de la mejor solución del subconjunto. De acuerdo a una cota (superior o inferior).
 - Poda (elimina) la rama del árbol si la cota indica que no puede contener la solución óptima.

Dentro de los algoritmos de tipo ramificación y acotación destacan los trabajos de Laporte, Mercure y Nobert (1986); Fischetti, Toth y Vigo (1994); Fisher (1994) [CONTARDO, 2005].

Cabe destacar que el método de Fisher puede llegar a resolver hasta 71 nodos destino (clientes), sin embargo esto requiere un costo computacional muy elevado [U.MÁLAGA, 2009].

- Ramificación y corte (Branch and cut) [RAMOS, 2007]
 - Es un método híbrido que usa los métodos de ramificación y acotación y también el método de planos de corte en los nodos.
 - Para ello se elige un nodo a evaluar (inicialmente el nodo raíz).
 - Luego se decide si se va a generar o no planos de corte.
 - Por último se aplican los criterios del método de ramificación y acotación.

Dentro de los algoritmos de tipo Ramificación y corte destacan los trabajos de Cornuéjols, Fonlupt y Naddef (1985); Naddef y Rinaldi (1993); Toth y Vigo (2002) [CONTARDO, 2005].

Lamentablemente este tipo de soluciones depende mucho de la estructura particular del VRP, ya que al aplicar cortes a una estructura que no favorece a este algoritmo, la solución puede llegar a complicarse. [CONTARDO, 2005]

En general estos métodos son aplicables a una pequeña cantidad de nodos (máximo 100), es decir sólo para fines académicos, puesto que en problemas reales de VRP se tiene una estructura de más de 500 nodos, por lo tanto no son recomendables, porque requieren mucho tiempo de ejecución y mucho recurso computacional.

Para problemas reales de VRP se utilizan métodos aproximados que se detalla a continuación.

1.6.4.2 Métodos aproximados

Dentro de los métodos aproximados se describen los métodos heurísticos y meta heurísticos

A) Métodos heurísticos

Dentro de los métodos heurísticos que solucionan problemas de tipo VRP se categorizan en dos grandes ramas:

- Métodos constructivos

Este método construye gradualmente una solución factible de acuerdo al coste de la solución, estos métodos no tienen una fase de mejora.

Dentro de los métodos constructivos más conocidos se tiene el Algoritmo de Ahorro de Clarke y Wright (1964), y el Algoritmo de Ahorro basado en matching, en donde cabe destacar los trabajos de Desrochers and Verhoog (1989) y Altinkemer and Gavish (1991) [U.MÁLAGA, 2009].

Este tipo de algoritmo resuelve los problemas de tipo VRP general.

- Métodos de inserción

Los métodos de inserción parten con rutas inicialmente vacías (o que contienen un único nodo) e iterativamente evalúan la mejor forma de insertar un nodo en alguna ruta, y se quedan con el par (nodo, ruta) que representa la mejor inserción. [CONTARDO, 2005]

Dentro de los métodos de inserción se tienen los trabajos de Mole y Jameson (1976); Christofides, Mingozzi y Toth (1979); Solomon (1987).

B) Métodos meta heurísticos

Dentro de los métodos meta heurísticos que solucionan problemas VRP se tienen los siguientes: [U. TRIER, 2009]

- Colonia de hormigas

Solucionan las siguientes variantes de VRP:

- Problema de Ruteo de Vehículos con Capacidad.
(Karl Doerner, Richard F. Hartl, Guenter Kiechle, Mária Lucká, Marc Reimann, 2004).
- Problema de Ruteo de Vehículos con Ventana de Tiempo.
(Ismail Ellabib, Otman A. Basir, Paul H. Calamai, 2002).
(Tao Zhang, Shanshan Wang, Wenxin Tian, Yuejie Zhang, 2006).
- Problema de Ruteo de Vehículos con Ventana de Tiempo y Conectividad.
(Marc Reimann, Karl Doerner, Richard F. Hartl, 2002).
- Problema de Ruteo de Vehículos Periódico.
(Ana Cristina Matos, Rui Carvalho Oliveira, 2004).
- Algoritmos genéticos

Solucionan las siguientes variantes de VRP:

- Problema de Ruteo de Vehículos.
(Francisco B. Pereira, Jorge Tavares, Penousal Machado, Ernesto Costa, 2002).
(Enrique Alba, Bernabé Dorronsoro, 2004).
- Problema de Ruteo de Vehículos con Capacidad.
(Jean Berger, Mohamed Barkaoui, 1999).
- Problema de Ruteo de Vehículos con Ventana de Tiempo.
(Jean Berger, Mourad Sassi, Martin Salois, 1999).
(Soonchul Jung, Byung Ro Moon, 2002).
(Kenny Qili Zhu, 2003).
(Guilherme Bastos Alvarenga, Geraldo Robson Mateus, 2004).
(Humberto Cesar Brandao de Oliveira, Jose Lima Alexandrino, Mariane Moreira de Souza, 2004).
(Tao-Shen Li, Jing-Li Wu, 2005).
- Problema de Ruteo de Vehículos con Ventana de Tiempo y Capacidad.
(Joe L. Blanton Jr., Roger L. Wainwright, 1993).

- Problema de Ruteo de Vehículos con Plazos.
(Sam R. Thangiah, Rajini Vinayagamooty, Ananda V. Gubbi, 1993).

- Búsqueda tabú

Solucionan las siguientes variantes de VRP:

- Problema de Ruteo de Vehículos.
(Etiene Pozzobom Lazzeris Simas, Arthur Tórgo Gómez, 2006).
(Gülay Barbarosoglu, Demet Ozgur, 1999).
(Dharmendra Kumar Gupta, 2002).
- Problema de Ruteo y Programación de Vehículos.
(Herbert Kopfer, Jörn Schönberger, 2002).
- Problema de Ruteo de Vehículos con Ventana de Tiempo.
(Panagiotis P. Repoussis, Dimitris C. Paraskevopoulos, Christos D. Tarantilis, George Ioannou, 2006).
(Wen-Chyuan Chiang, Robert A. Russell, 1997).
- Problema de Ruteo de Vehículos con Ventana de Tiempo y Entrega Partida.
(Sin C. Ho, Dag Haugland, 2004).
- Problema de Ruteo de Vehículos con Ventana de Tiempo y Apremios.
(Bruno-Laurent Garcia, Jean-Yves Potvin, Jean-Marc Rousseau, 1994).
- Problema de Ruteo de Vehículos con Entrega y Recogida de Productos Simultáneamente.
(Fermín Alfredo Tang Montané, Roberto Diéguez Galvão, 2006).
- Problema de Ruteo de Vehículos con Múltiples Depósitos.
(Jacques Renaud, Gilbert Laporte, Fayez F. Boctor, 1996).

- GRASP

Solucionan las siguientes variantes de VRP:

- Problema de Ruteo de Vehículos Periódico.
(Luciana B. Goncalves, Luiz S. Ochi, Simone L. Martins, 2005).
- Problema de Ruteo de Vehículos con Ventana de Tiempo.
(Wanpracha Chaovaitwongse, Dukwon Kim, Panos M. Pardalos, 2003).
(Kontoravdis, George, Bard, Jonathan F, 1995).

1.6.5 Aplicaciones existentes

En la actualidad existen aplicaciones que permiten hallar la ruta óptima entre dos o más puntos, a continuación se detallan dichas aplicaciones.

- Google Maps [GOOGLE MAPS, 2009]
 - Es una herramienta Web gratuita de Google que permite encontrar la ruta óptima entre dos a más puntos destino.
 - Considera el sentido de la calle (en caso se seleccione la opción a coche).
 - Está integrado con Google Earth dando de esta manera un mejor detalle de la ruta real a seguir.
 - No se precisa el algoritmo que utiliza para calcular la ruta óptima.
 - La interfaz del producto es sencilla y amigable, tal como se muestra en la figura 1.6.

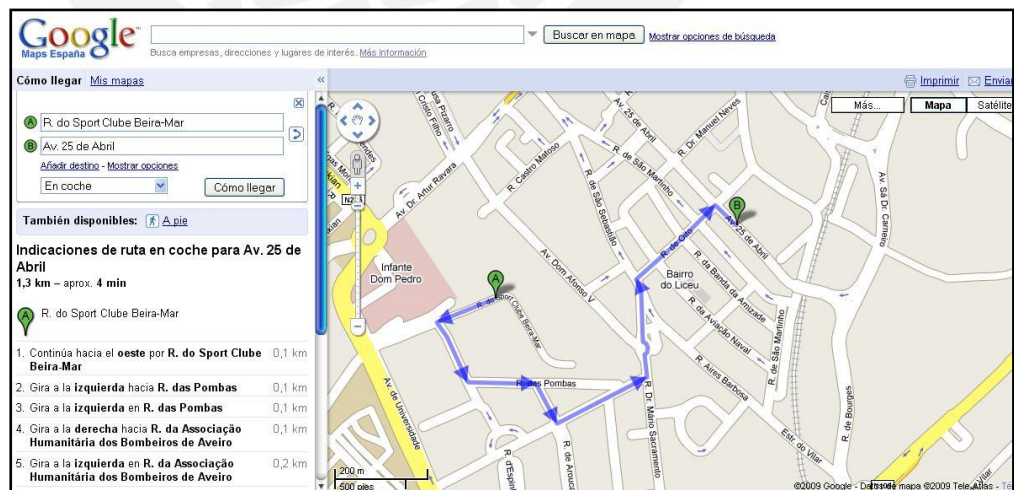


Figura 1.6 Google Maps.

- Guía de Calles - Software de Rutas [CALLES, 2009]

- Características:
 - o Calcula la ruta óptima para un conjunto de clientes a visitar mostrando los mapas y hojas de ruta.
 - o Incluye el detalle de la secuencia de visita, tiempos, distancia y costos involucrados.
 - o Trabaja en conjunto con PC Guía de Calles.

- Beneficios:
 - o Obtener la ruta y la tabla con la matriz de costos, distancias y tiempos de cada transporte.
 - o Generar y guardar los datos necesarios para las rutas diarias.
 - o Crear o modificar las zonas de venta.

- Áreas de Aplicación:
 - o Delivery, reparto postal, distribución (logística).
 - o Opcional el control de GPS.

Cabe resaltar que esta aplicación es de origen peruano, sin embargo no se precisa que algoritmo en especial utiliza para generar las rutas óptimas.

- Grafos [GRAFOS, 2009]

Es un software para la construcción, edición y análisis de grafos. Esta herramienta puede servir de gran ayuda en los campos de la ingeniería de organización industrial, logística, transporte, investigación operativa, diseño de redes, etc. Grafos se puede usar perfectamente para el modelado y resolución de problemas reales. Con esta herramienta es posible armar los nodos clientes y elegir el/los punto(s) de origen y con ello hallar la ruta óptima (tal como se muestra en la figura 1.7), esta herramienta además permite elegir el método (algoritmo) que ayuda a resolver el problema.

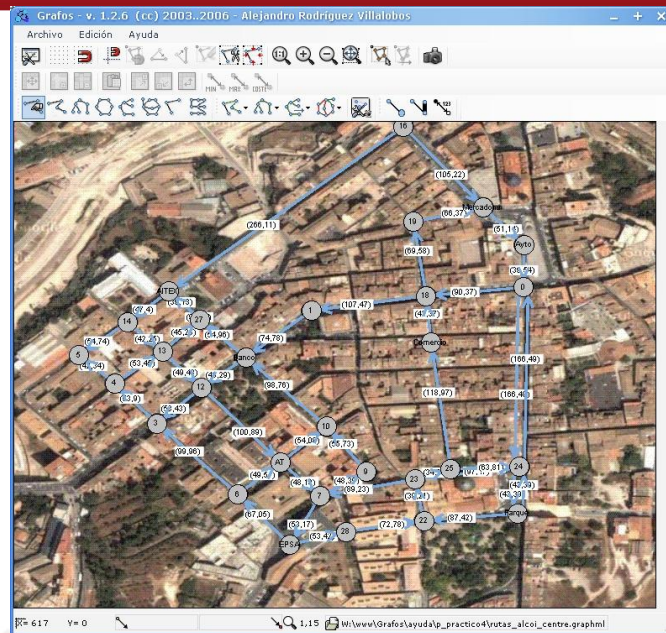


Figura 1.7: Software Grafos.

1.6.6 Actividades y Congresos

En la actualidad se celebran conferencias y congresos importantes sobre el tema de ruteo de vehículos, entre estos congresos cabe resaltar los siguientes:

- Advances in Artificial Intelligence.
- International Symposium on Artificial Intelligence and Mathematics.
- Conference on Artificial Intelligence and Cognitive Science.
- Ant Colony Optimization and Swarm Intelligence, International Workshop.
- Genetic and Evolutionary Computation Conference.
- Evolutionary Computation in Combinatorial Optimization.
- European Transport Conference.

CAPÍTULO 2: METODOLOGÍA Y ARQUITECTURA

En este capítulo se definirá la metodología, la arquitectura de la información y la arquitectura del sistema que se utilizará a lo largo del proyecto.

2.1 Definición de la metodología de la solución

Para este proyecto se va adaptar la metodología XP (eXtreme Programming), ya que es una metodología ligera de desarrollo de software que se caracteriza por su simplicidad, grado de comunicación y la realimentación o reutilización del código desarrollado [DSI, 2002].

Además el objetivo de este proyecto es la implementación del algoritmo, por lo tanto esta metodología se ajusta muy bien dado que se centra más en la implementación que en la documentación [PX, 2004].

2.1.1 Fases del XP

Las Fases del XP son:

- Planificación

En esta etapa se plantea los alcances del sistema a desarrollar.

Se presentará el siguiente documento:

- Historias de usuario: en este documento irán los requerimientos del sistema, según esta metodología los clientes son los que se encargan de llenar dicho documento, sin embargo en este proyecto el cliente va ser reemplazado por el encargado del proyecto y por el asesor.

- Diseño

En esta etapa se plantea el diseño que tendrá el sistema a desarrollar.

Se presentarán los siguientes documentos:

- Glosario de términos: en este documento se especifican los métodos y clases que tendrá el sistema.
- Prototipo de interfaz de usuario: en este documento se presenta el diseño del sistema.
- Prototipo de base de datos: en este documento se detalla la base de datos que tendrá el sistema.

- Desarrollo

Esta es la etapa más importante tanto en la metodología como en el proyecto ya que se busca implementar los dos algoritmos GRASP y la optimización 2-Opt.

Se presentará el siguiente documento:

- Estándar de programación: en este documento se detalla los estándares de programación que se utilizarán en el sistema.

- Pruebas

En esta etapa se desarrollan las pruebas al sistema, al ser un sistema inteligente las pruebas se van a enfocar prácticamente en la eficiencia del algoritmo y la calibración de las constantes de relajación.

Se presentará el siguiente documento:

- Pruebas de optimización: en este documento se detallará las pruebas que se realizaron para calibrar la constante de relajación y las pruebas de rendimiento y tiempo de ejecución del algoritmo.

Las historias de usuario, glosario de términos, estándar de programación y las pruebas de optimización irán en el anexo, mientras que el diseño de interfaz se detalla en la sección 2.3 y el prototipo de base de datos en el capítulo 3.

2.2 Arquitectura de la Información

La arquitectura de la información se va a trabajar a nivel de grafos.

Un grafo representa un modelo de una realidad empresarial en forma de red, un grafo está compuesto por nodos y aristas [GRAFOS, 2009], tal como se muestra en la figura 2.1.

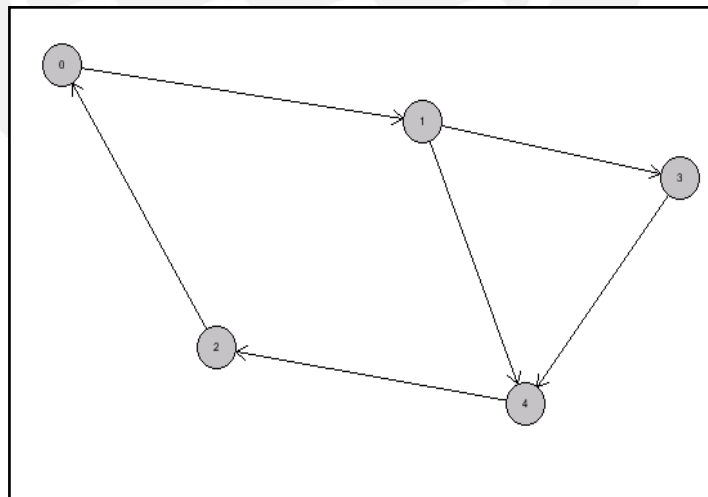


Figura 2.1: Grafo con 5 nodos y 6 aristas.

En este proyecto un grafo va a ser la representación de una parte de una ciudad.

Los nodos representan las intersecciones de las calles, se manejan dos tipos de nodos:

- Nodos temporales de destinatarios: son los nodos que el algoritmo va a tomar obligatoriamente como parte de su solución. Estos nodos se van a crear temporalmente por cada ejecución del algoritmo.
- Nodos comunes: son los nodos de paso del algoritmo.

Los nodos comunes contendrán la siguiente información:

- Estado: indica si el nodo está activo o inactivo en el grafo.

Los nodos temporales de destinatarios contendrán la siguiente información:

- Número de paquetes: indica la cantidad de paquetes que se deben repartir al destinatario, además por cada paquete se indica la dimensión (largo, ancho, alto) y peso que tiene.
- Estado: indica si el nodo está activo o inactivo en el grafo.

Las aristas representan una cuadra de una calle delimitada por dos nodos, y contendrán la siguiente información:

- Distancia: indica el tamaño de la calle.
- Constante de tráfico: indica el nivel de tráfico de la calle (del 1 al 5) donde 1 significa que no hay tráfico y 5 significa que es una calle muy transitada.
- Sentido de la calle: indica si es unidireccional o bidireccional.
- Estado de la cuadra: indica si está disponible o no dicha cuadra.

Además dependiendo del turno (mañana o tarde) contendrán la siguiente información:

- Tiempo de recorrido.
- Consumo de gasolina.
- Costo de gasolina.

Para ingresar la información de estos tres datos es necesario cargar la información con los siguientes parámetros:

- Consumo promedio de galón por hora.
- Costo promedio de gasolina por galón.

Para las direcciones de los clientes se crean nodos temporales, donde se agrupa en un solo nodo temporal aquellos que vivan en una misma cuadra.

El objetivo del problema al ser VRP va ser recorrer todos los nodos temporales de destinatarios, minimizando la distancia, el tiempo, el costo y consumo del combustible y el número de vehículos a emplear.

Se van a manejar tres tipos de grafos:

- Grafo Normal: grafo formado por nodos y aristas comunes que representan una parte de una ciudad.
- Grafo Temporal: grafo formado dinámicamente (al ejecutar el algoritmo) formado por nodos y aristas comunes y por los nodos temporales de destinatarios.
- Grafo Virtual: grafo formado únicamente por nodos y aristas virtuales de destinatarios.

2.3 Arquitectura del Sistema

La arquitectura del sistema va ser Cliente/Servidor de 2 capas, debido a que es una arquitectura que permite separar la lógica de negocio con la lógica de datos.

La base de datos va ser tal como se muestra en la figura 2.2.

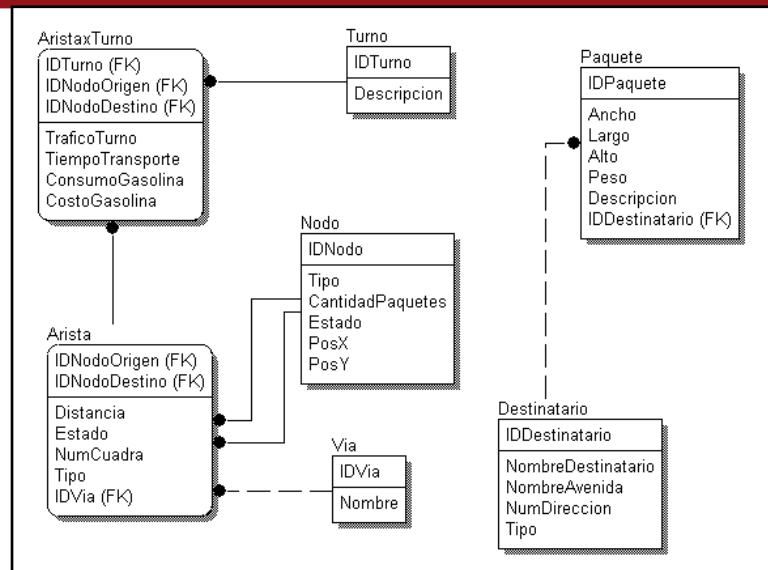


Figura 2.2: Base de datos del núcleo del sistema (algoritmo).

El lenguaje de programación a usar será JAVA, debido a los siguientes motivos:

- Es un lenguaje multiplataforma, esto quiere decir es un lenguaje que puede funcionar en Windows, Linux y en otros sistemas operativos.
- Es un lenguaje que posee una gran cantidad de librerías gráficas que permite al desarrollador hacer que este proyecto sea más agradable en el aspecto visual.

CAPÍTULO 3: DISEÑO

En este capítulo se definirá la interfaz gráfica y la división modular que tendrá el sistema.

3.1 Definición de interfaz gráfica

En este proyecto se va a seguir la siguiente interfaz gráfica, tal como se muestra en la figura 3.1.

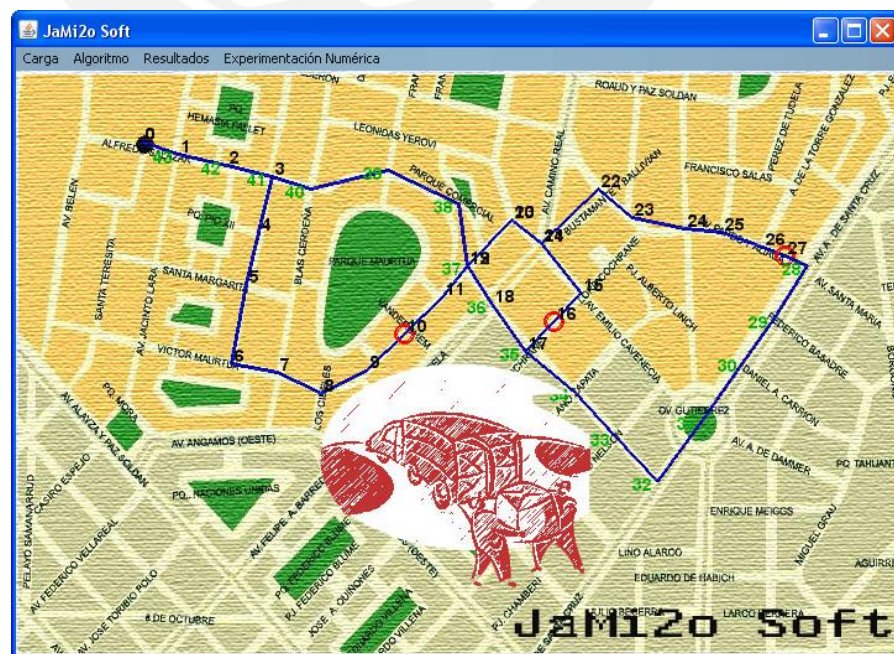


Figura 3.1: Vista principal del Software JaMi2o Soft.

- Se va a definir una barra de Menú con los módulos del sistema.
- La letra a seguir será Tahoma de tamaño 11 de color negro.
- La pantalla será de 714x491.

3.2 División modular

La división modular del sistema será de la siguiente manera:

- Módulo de carga, tal como se muestra en la figura 3.2.

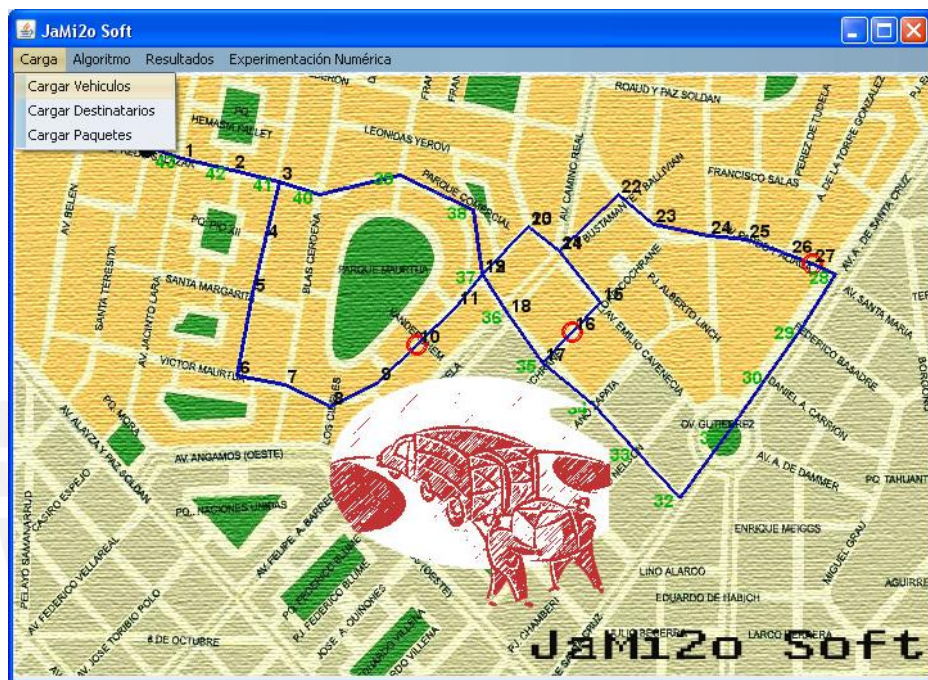


Figura 3.2: Módulo de carga del Software JaMi2o Soft.

En este módulo se realizará la carga de Destinatarios y Paquetes a través de archivos de texto.

La carga de vehículos se realizará tal como se muestra en la figura 3.3.

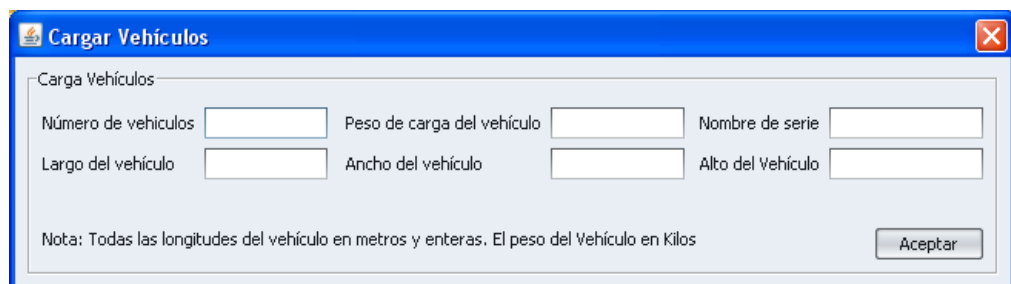


Figura 3.3: Ventana de carga de vehículos.

En la ventana de carga de vehículos se ingresa los siguientes datos:

- Número de vehículos: cantidad máxima de vehículos que se utilizarán para realizar el reparto.
 - Peso de carga del vehículo: en kilogramos.
 - Nombre de serie: nombre inicial de la serie de vehículos de reparto. Los siguientes nombres de vehículos serán autogenerados de manera correlativa.
 - Largo del vehículo: medida en metros, medida que corresponde al contenedor del vehículo.
 - Ancho del vehículo: medida en metros, medida que corresponde al contenedor del vehículo.
 - Alto del vehículo: medida en metros, medida que corresponde al contenedor del vehículo.
- Módulo del algoritmo, tal como se muestra en la figura 3.4.

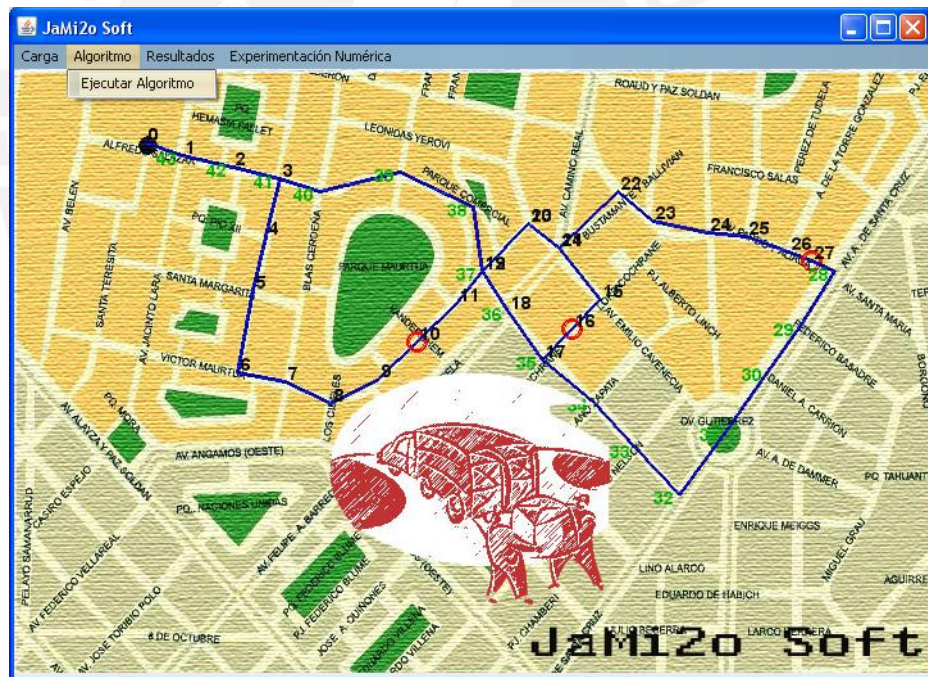


Figura 3.4: Módulo del algoritmo.

En este módulo se configurarán los parámetros de los dos algoritmos GRASP tal como se muestra en la figura 3.5.

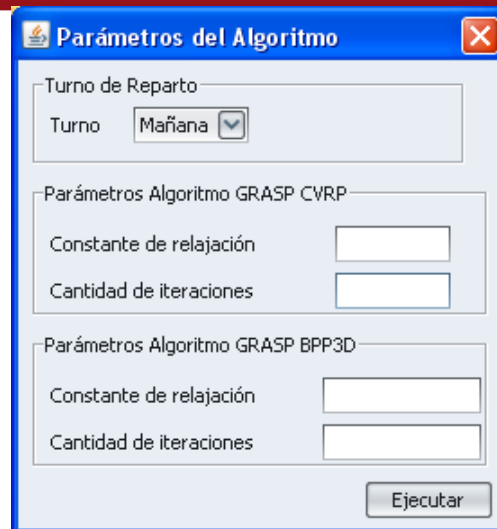


Figura 3.5: Ventana de parámetros del algoritmo.

En la ventana de parámetros del algoritmo se ingresan los siguientes datos:

- Turno de reparto: Mañana o Tarde, la constante de tráfico variará en cada calle de acuerdo al turno de reparto.
 - Constante de relajación del algoritmo GRASP CVRP.
 - Cantidad de iteraciones del algoritmo GRASP CVRP.
 - Constante de relajación del algoritmo GRASP BPP3D.
 - Cantidad de iteraciones del algoritmo GRASP BPP3D.
- Módulo de resultados, tal como se muestra en la figura 3.6.

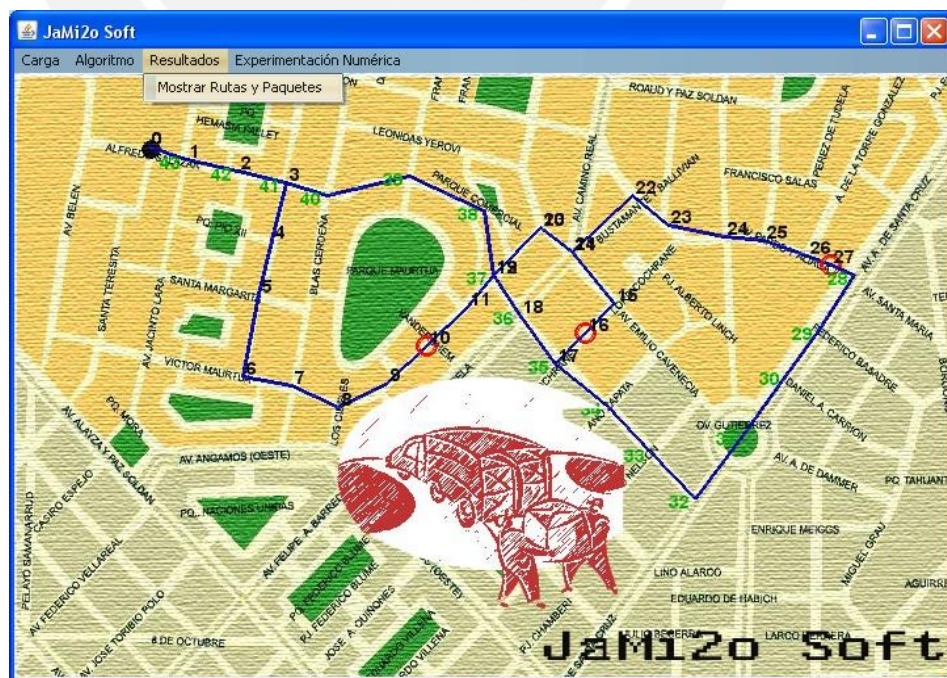


Figura 3.6: Módulo de resultados.

Uno de los módulos más importantes en donde se mostrará el resultado tal como se muestra en la figura 3.7.

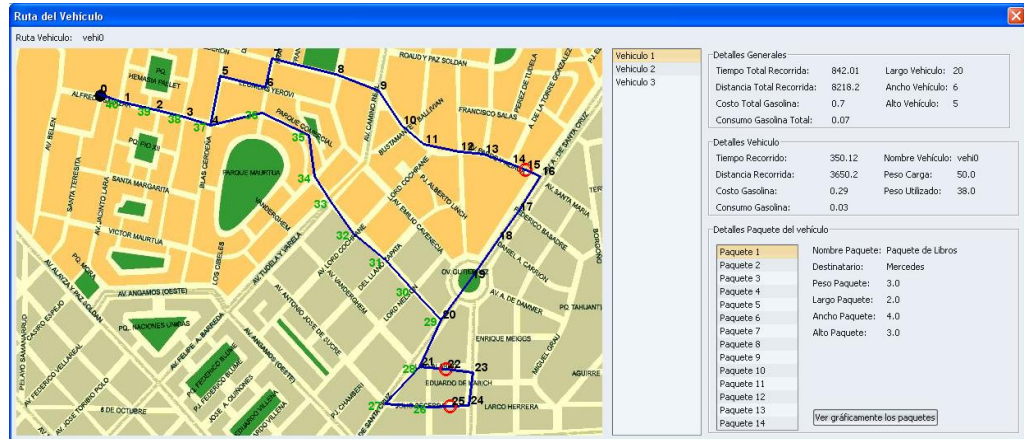


Figura 3.7: Ventana de resultados.

En esta ventana de resultados se detallan los siguientes datos:

- Lista de vehículos que se utilizarán para el reparto.
- Medidas del vehículo: largo, ancho y alto (en metros).
- Ruta a seguir por vehículo.
- Tiempo total recorrido de todos los vehículos y por vehículo (en minutos).
- Distancia total recorrida de todos los vehículos y por vehículo (en kilómetros).
- Costo total de gasolina de todos los vehículos y por vehículo (en nuevo soles).
- Consumo total de gasolina de todos los vehículos y por vehículo (en galones).
- Lista de paquetes por vehículo, mostrando el nombre del paquete, el destinatario, el peso, largo, ancho y alto del paquete. Además al momento de dar clic en "Ver gráficamente los paquetes" se muestra la ventana tal como se muestra en la figura 3.8.

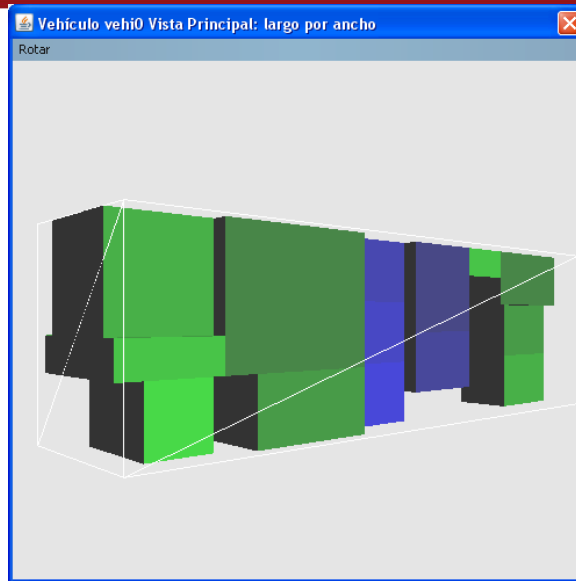


Figura 3.8: Ventana de embalaje del vehículo con los paquetes embalados.

En la figura 3.8 se muestra la posición de los paquetes embalados en el vehículo elegido, mostrando el largo y ancho del contenedor del vehículo como vista principal, además se puede rotar la imagen logrando ver los paquetes y la capacidad del vehículo en tres dimensiones.

- Módulo de experimentación numérica, tal como se muestra en la figura 3.9.

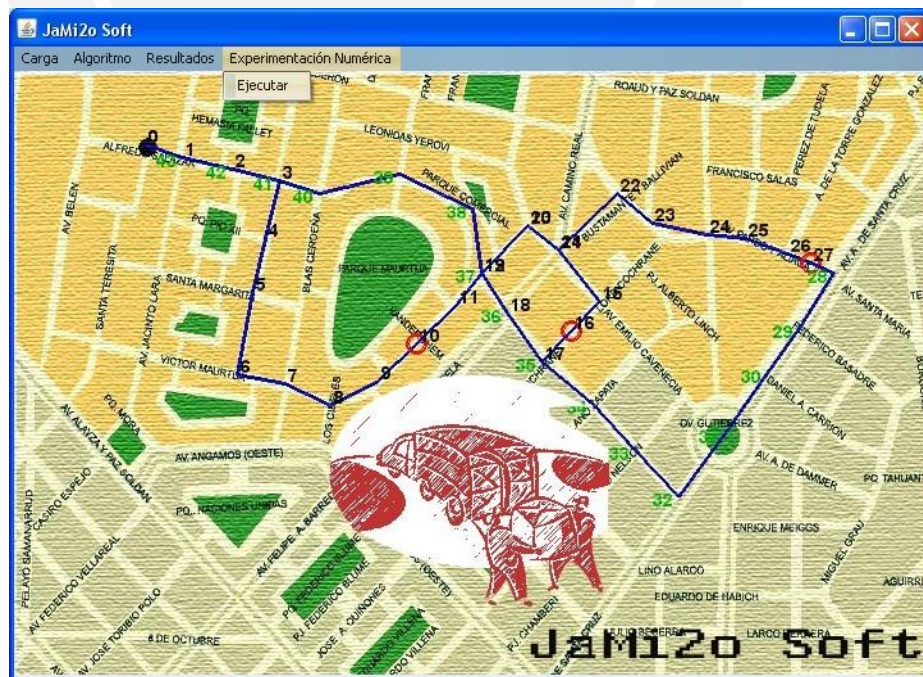


Figura 3.9: Módulo de experimentación numérica.

Módulo donde se realizará la experimentación numérica de los algoritmos implementados, creando grafos, vehículos, destinatarios y paquetes aleatorios según el rango que se indique, tal como se muestran en las figuras 3.10, 3.11, 3.12 y 3.13.

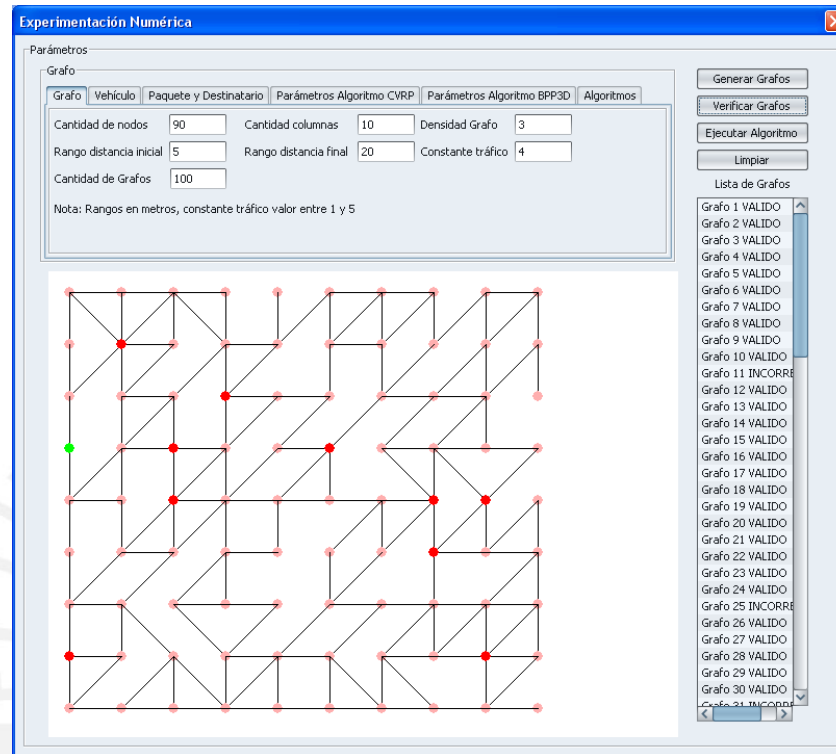


Figura 3.10: Ventana de experimentación numérica (Grafo).

Los puntos de color rosado indican los nodos comunes, los nodos de color rojo son los nodos destinatarios y el nodo de color verde es el nodo base o punto de partida de los vehículos.

En la ventana de experimentación numérica (Pestaña grafo) se ingresan los siguientes datos:

- Cantidad de nodos del grafo.
- Cantidad de columnas del grafo.
- Densidad del grafo (máxima cantidad de aristas por nodo).
- Rango distancia inicial (en metros).
- Rango distancia final (en metros).
- Constante de tráfico (valor del 1 al 5).
- Cantidad de grafos a generar.

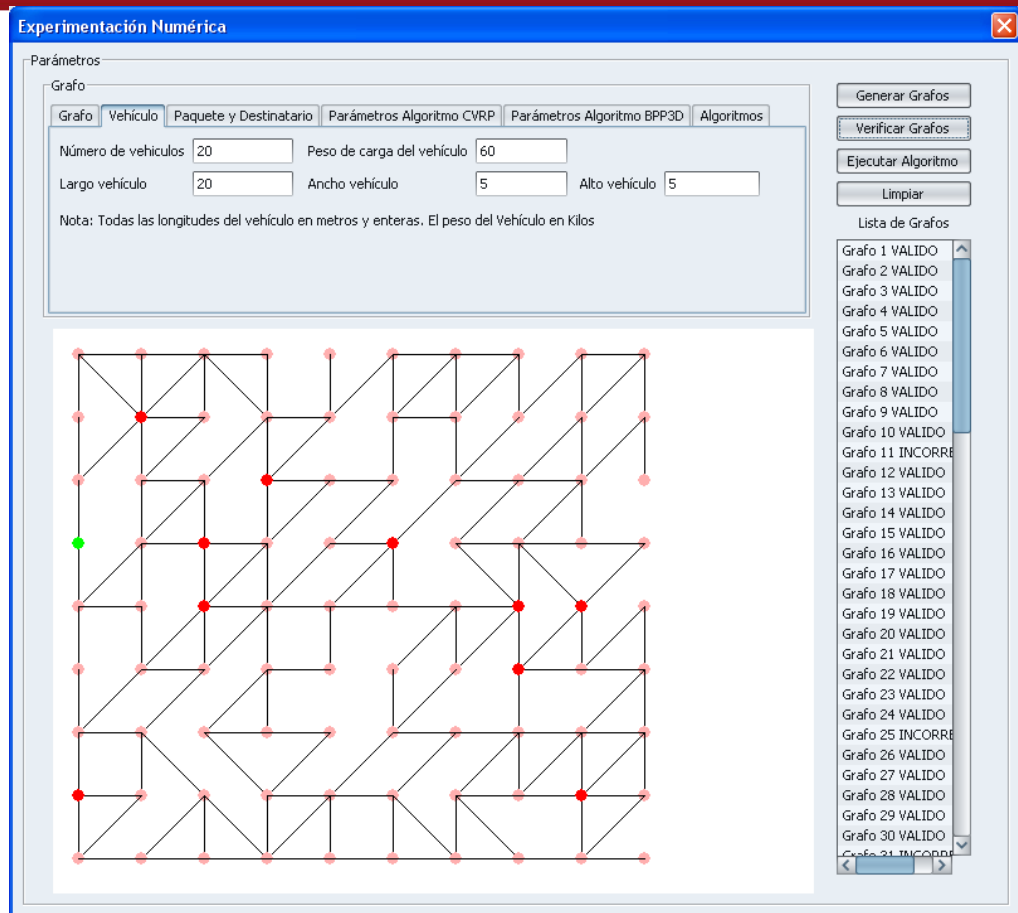


Figura 3.11: Ventana de experimentación numérica (Vehículo).

En la ventana de experimentación numérica (Pestaña vehículo) se ingresan los siguientes datos:

- Número de vehículos.
- Peso de carga de los vehículos (en kilogramos).
- Largo del contenedor del vehículo (en metros).
- Ancho del contenedor del vehículo (en metros).
- Alto del contenedor del vehículo (en metros).

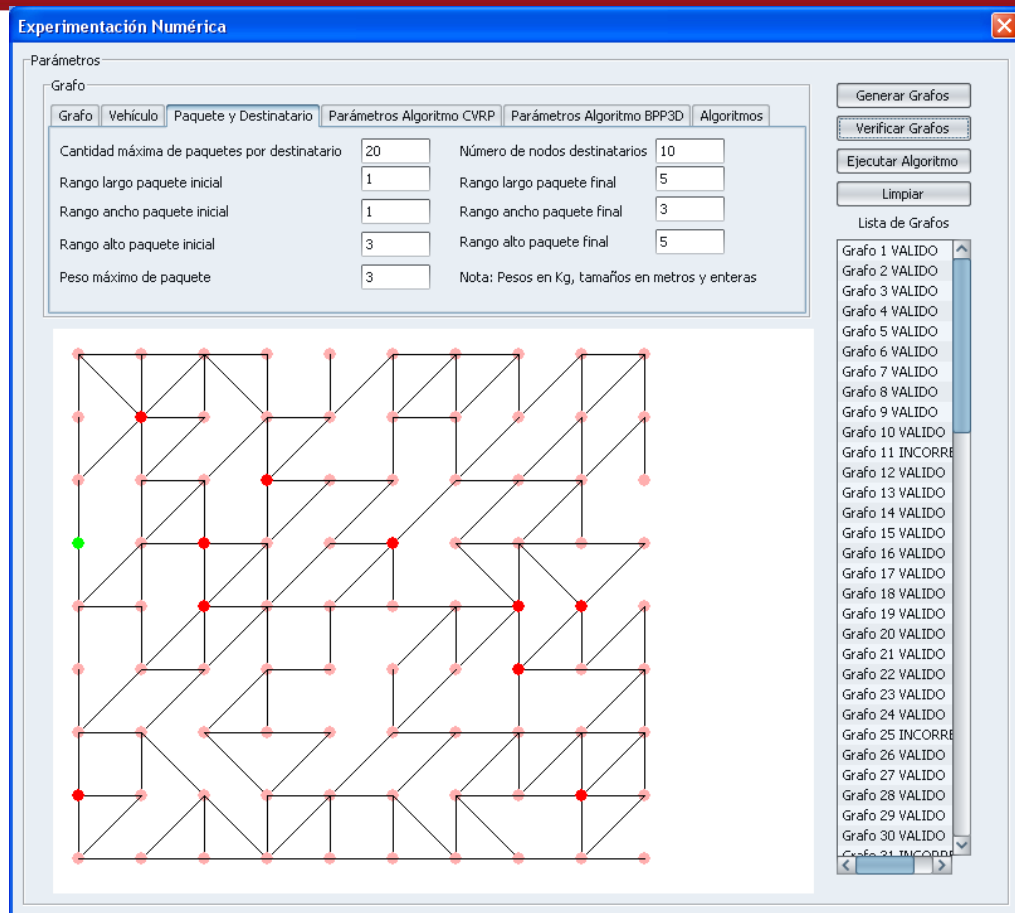


Figura 3.12: Ventana de experimentación numérica (Paquete y destinatario).

En la ventana de experimentación numérica (Pestaña paquete y destinatario) se ingresan los siguientes datos:

- Cantidad máxima de paquetes por destinatario.
- Número de nodos destinatario.
- Rango del largo inicial y final del ancho, largo y alto del paquete del vehículo (en metros).

Los parámetros del algoritmo CVRP y BPP3D son iguales a los parámetros del módulo del algoritmo.

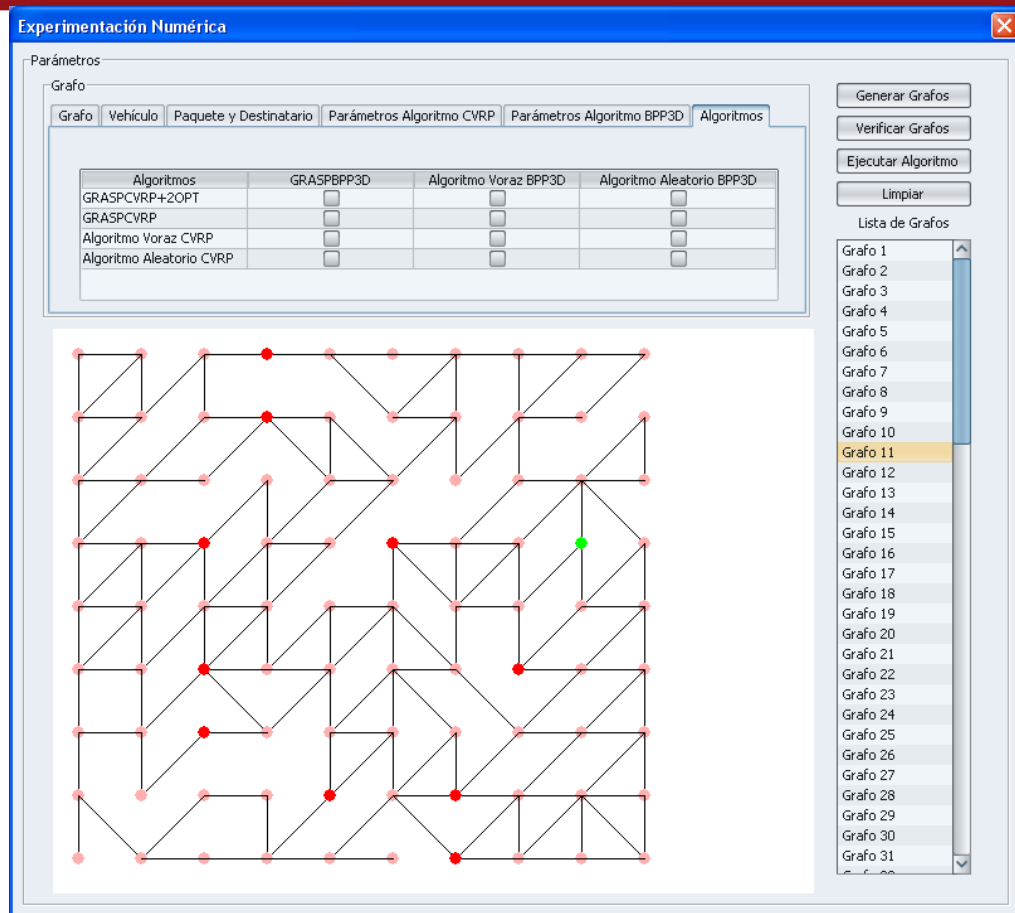


Figura 3.13: Ventana de experimentación numérica (Algoritmos).

El sistema permite seleccionar hasta 12 combinaciones de algoritmos:

Fase CVRP:

- GRASPCVRP + 2-Opt.
- GRASPCVRP.
- Algoritmo Voraz CVRP.
- Algoritmo Aleatorio CVRP.

Fase BPP3D:

- GRASP BPP3D.
- Algoritmo Voraz BPP3D.
- Algoritmo Aleatorio BPP3D.

CAPÍTULO 4: CONSTRUCCIÓN DE LOS ALGORITMOS

En este capítulo se construirá los algoritmos que solucionarán el problema de ruteo de vehículos con capacidad uniforme.

4.1 Algoritmo generador de grafo temporal

4.1.1 Descripción

Este algoritmo genera nodos temporales de acuerdo a las direcciones ingresadas como parámetro, agrupando aquellas que pertenezcan a la misma cuadra, devolviendo un nuevo grafo temporal.

4.1.2 Datos de Entrada

- **LDestinatario:** es la lista de destinatarios donde se va a realizar el reparto, en esta lista se detalla el nombre, dirección y número de casa del cliente.
- **Grafo:** es el conjunto de nodos y aristas que conforman una parte de un distrito.

4.1.3 Dato de Salida

El Algoritmo devolverá un nuevo grafo temporal, donde se incluirán los nuevos nodos formados por los destinatarios.

4.1.4 Algoritmo

1	Procedimiento GenerarGrafoTemporal(LDestinatario, Grafo)
2	Para cada i desde 0 hasta cantidad(LDestinatario) hacer
3	dirDestinatario ← obtenNumDireccion(LDestinatario[i])
4	nomAvenida ← obtenNombreAvenida(LDestinatario[i])
5	Si NofueCreadoNodoCuadra(nomAvenida, dirDestinatario) entonces
6	obtenNodos(N1,N2,dirDestinatario,nomAvenida,Grafo)
7	distanciaNodos ← obtenDistancia(N1, N2, Grafo)/2
8	posX ← (obtenPosX(N1) + obtenPosX(N2))/2
9	posY ← (obtenPosY(N1) + obtenPosY(N2))/2
10	nodoID ← crearNodo(MaxID(Grafo), posX, posY, Grafo)
11	Si haySentido(N1,N2) entonces
12	crearArista(N1, nodoID, distanciaNodo, obtenTrafico(N1, N2), Grafo)
13	crearArista(nodoID, N2, distanciaNodos, obtenTrafico(N2, N1), Grafo)
14	borrarArista(N1,N2, G)
15	Fin Si
16	Si haySentido(N2,N1) entonces
17	crearArista(N2, nodoID, distDest_N2, obtenTrafico(N2, N1), Grafo)
18	crearArista(nodoID, N1, distN1_Dest, obtenTrafico(N1, N2), Grafo)
19	borrarArista(N2,N1, G)
20	Fin Si
21	Caso contrario
22	Nodo ← buscarNodo(nomAvenida, dirDestinatario)
23	ActualizarNodo(Nodo, LDestinatario[i])
24	Fin Si
25	Fin Para
26	Retornar Grafo
27	Fin CargarGrafoTemporal

Algoritmo 4.1: Algoritmo generador de grafo temporal

4.1.5 Ejemplo y Comentarios

- LDestinatario:

En LDestinatario se carga la información de los destinatarios tal como se muestra en la figura 4.1.

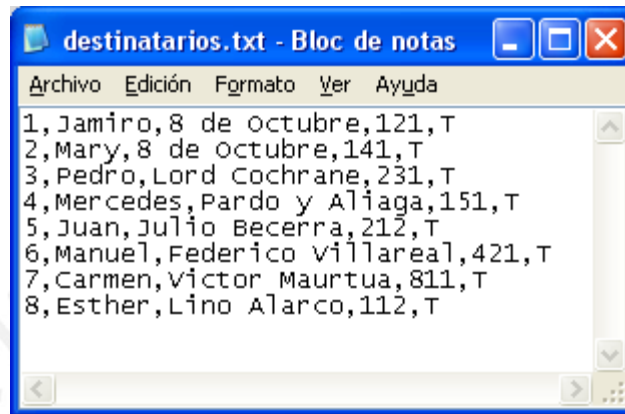


Figura 4.1: Archivo de destinatarios.

- Grafo:

Grafo utilizado tal como se muestra en la figura 4.2.

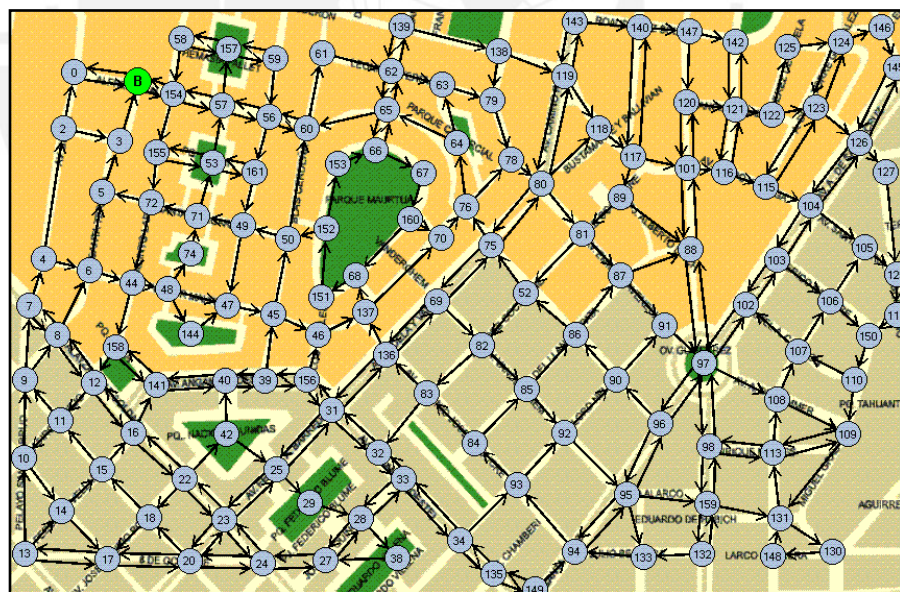


Figura 4.2: Grafo utilizado donde se muestra los nodos y aristas relacionados a un distrito, el nodo B va ser el punto de partida (o nodo inicio) de los vehículos.

- Comentarios del Algoritmo generador de grafo temporal:
 - En las líneas 3 y 4 se obtiene el nombre de la avenida y el número de dirección del destinatario, para el ejemplo en la primera iteración el

algoritmo devolverá la avenida “8 de Octubre” y como número de dirección “121”.

- En la línea 5 se hace la verificación de previa creación de nodo temporal según el nombre de la avenida y el número de dirección.
- En caso no se haya creado, en la línea 6 se obtienen los nodos N1 y N2 del Grafo, para la primera iteración el algoritmo devolverá los nodos con ID 13 y 17, tal como se muestra en la figura 4.3.

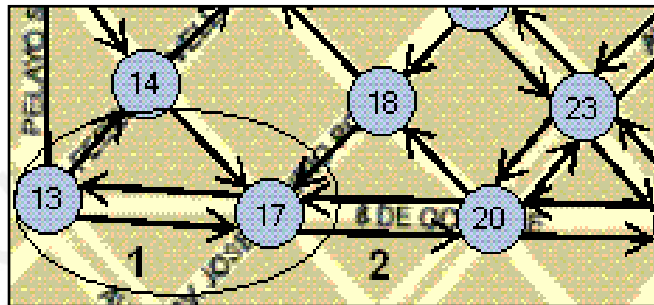


Figura 4.3: Ubicación de la avenida 8 de Octubre con número de dirección 121 y la ubicación de los nodos N1 y N2.

- En la línea 7 se obtiene la distancia entre N1 y N2, que va ser justo el centro de la cuadra.
- En las líneas 8 y 9 se obtiene la posición de los nodos N1 y N2, y con ello se calcula la nueva posición para el nuevo nodo.
- En la línea 10 se crea el nodo, y entre las líneas 11 al 20 se crean las nuevas aristas de acuerdo al sentido de tránsito y se eliminan las aristas entre N1 y N2 quedando la avenida 8 de Octubre tal como se muestra en la figura 4.4.

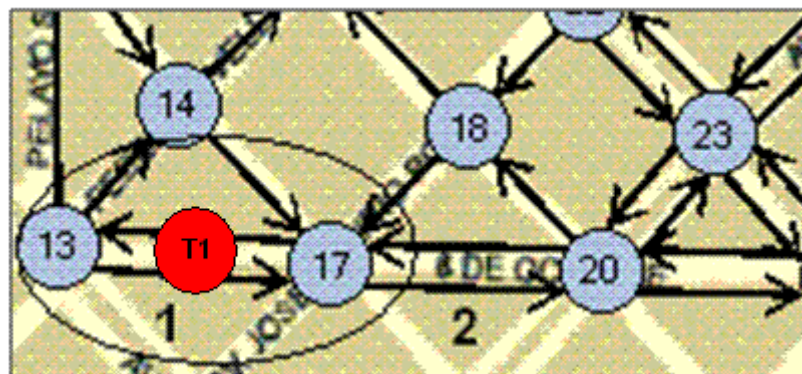


Figura 4.4: Creación del nuevo nodo temporal entre N1 y N2 (Grafo Temporal).

- En caso el nodo haya sido creado previamente, el algoritmo ejecutará la línea 22 y 23 que consiste en buscar el nodo temporal y actualizar con la información de los nuevos paquetes del destinatario.
- Este proceso se repite para los siguientes destinatarios, devolviendo el grafo temporal tal como se muestra en la figura 4.5.



Figura 4.5: Grafo Temporal donde está incluido los nodos temporales (T1, T2, T3, T4, T5, T6 y T7) y el nodo base (B).

4.2 Algoritmo generador de grafo virtual

4.2.1 Descripción

Es el algoritmo que genera un grafo virtual (con aristas virtuales) donde los nodos formados son: el nodo origen y los nodos temporales de destinatarios. Para crear las aristas virtuales se usará el algoritmo Primero el mejor (Best first search).

4.2.2 Datos de Entrada

- LNodosTemp: es la lista de nodos temporales nuevos, creados en el paso anterior, además se incluye el nodo base (origen) por donde partirán los vehículos de reparto.

- **GrafoTemporal:** es el conjunto de nodos y aristas que conforman una parte de un distrito, en ella se detalla el nombre y dirección de las avenidas, así como la distancia y tráfico de una avenida. Además en este GrafoTemporal están incluidos los nodos temporales de destinatarios.

4.2.3 Dato de Salida

El algoritmo devolverá un nuevo grafo llamado *GrafoVirtual* donde los nodos formados son exclusivamente nodos temporales de destinatarios, el nodo origen y las aristas virtuales.

4.2.4 Algoritmo

1	Procedimiento	CargarGrafoVirtual(LNodosTemp,GrafoTemporal)
2		Inicializar GrafoVirtual
3		$N \leftarrow \text{Cantidad_Nodos}(\text{LNodosTemp})$
4		Para cada i desde 0 hasta N hacer
5		Para cada j desde 0 hasta N hacer
6		Si $i \neq j$ entonces
7		$\text{GrafoVirtual}[i,j] \leftarrow \text{BestFirstSearch}(\text{LNodosTemp}[i], \text{LNodosNuevos}[j], \text{GrafoTemporal})$
8		Fin Si
9		Fin Para
10		Fin Para
11		Retornar GrafoVirtual
12		Fin CargarGrafoVirtual

Algoritmo 4.2: Algoritmo generador de grafo virtual

4.2.5 Ejemplo y Comentarios

Siguiendo el ejemplo anterior se tienen como datos de entrada los siguientes elementos:

- **LNodosTemp:**

Lista de nodos formada por el nodo base y por el algoritmo generador de grafo temporal, contiene los nodos B, T1, T2, T3, T4, T5, T6 y T7.

- **GrafoTemporal:**

Grafo temporal generado tal como se muestra en la figura 4.6.



Figura 4.6: Grafo Temporal donde está incluido los nodos temporales (T1, T2, T3, T4, T5, T6 y T7) y el nodo base (B).

- Comentarios del algoritmo generador de grafo virtual:
 - En la línea 3 se calcula la cantidad de nodos nuevos, en este caso son 7 nodos nuevos.
 - En la línea 7 se generan las aristas virtuales mediante el algoritmo primero el mejor, que se explicará más adelante.
 - En la línea 11 el algoritmo retorna el grafo virtual. Siguiendo el ejemplo el grafo virtual generado sería tal como se muestra en la figura 4.7.

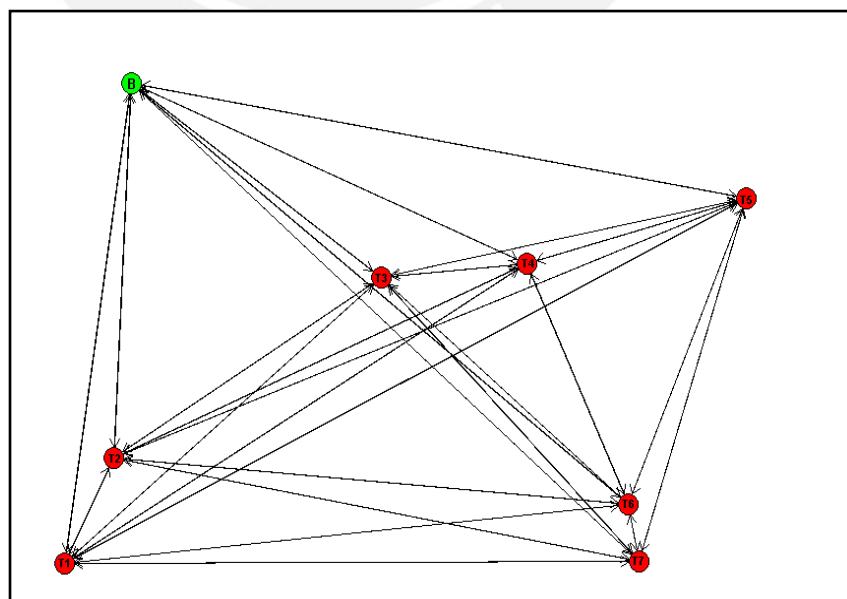


Figura 4.7: Grafo virtual formado por los nuevos nodos y el nodo base.

4.3 Algoritmo Primero el Mejor (Best First Search)

4.3.1 Descripción

Algoritmo generador de aristas virtuales para el grafo virtual.

4.3.2 Datos de Entrada

- NodoInicio: es el nodo inicial o punto de partida del algoritmo.
- NodoFin: es el nodo final o punto de llegada del algoritmo.
- GrafoTemporal: es el conjunto de nodos y aristas que conforman una parte de un distrito, en este GrafoTemporal están incluidos los Nodos Temporales.

4.3.3 Dato de Salida

El algoritmo devolverá una arista virtual conformada por un conjunto de nodos del grafo temporal.

4.3.4 Función Objetivo BFS

$$F_{(\text{Objetivo BFS})} = \text{Distancia} * \text{Constante Tráfico}$$

Donde:

- Distancia: es la distancia de la arista formada por el nodo actual y el nodo que se pretende recorrer. La medida es en metros.
- Constante Tráfico: es la constante de tráfico de la cuadra, varía del 1 al 5, si esta es cercana a 1 significa que no hay mucho tráfico, en caso contrario significa que es una calle con mucho tráfico.

4.3.5 Algoritmo

- | | |
|---|---|
| 1 | Procedimiento BestFirstSearch(NodoInicio, NodoFin, GrafoTemporal) |
| 2 | Inicializar LVisitados, Lestados |

```

3   Insertar_Inicio (Lestados, Nodoinicio)
4   Mientras (Lista_Vacia(LEstados) = Falso) y (NodoP ≠ NodoFin) hacer
5     MejorEstado ← Mejor_Elemento(LEstados, CriterioVoraz)
6     NodoP ← Ultimo_Nodo(MejorEstado)
7     Insertar_Final(LVisitados, NodoP)
8     LHijos_P ← Adyacente(GrafoTemporal, NodoP)
9     Si Lista_Vacia(LHijos_P) = Falso entonces
10      Para cada ( $\mu \in$  LHijos_P) y ( $\mu$  no  $\in$  LVisitados) hacer
11        Insertar_Fin(LEstados,  $\mu$ )
12      Fin Para
13    Fin Si
14  Fin Mientras
15  Retornar MejorEstado
16  Fin BestFirstSearch
  
```

Algoritmo 4.3: Algoritmo Best First Search

4.3.6 Ejemplo y Comentarios

Siguiendo el ejemplo anterior, la primera vez que sea llamado el algoritmo, recibirá como parámetros los siguientes elementos:

- Nodo Inicio: B.
- Nodo Fin: T1.
- Grafo Temporal:

Grafo temporal generado tal como se muestra en la figura 4.8.

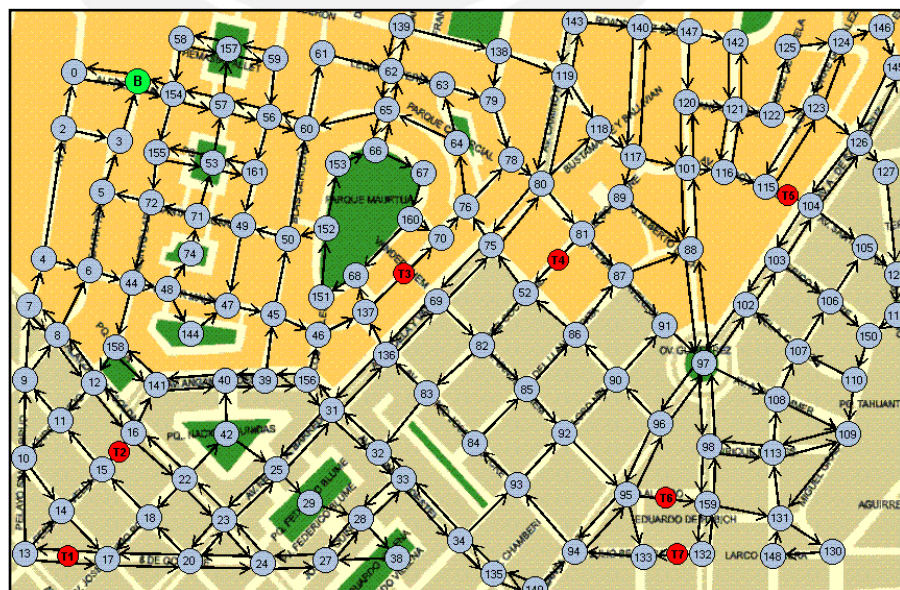


Figura 4.8: Grafo Temporal donde está incluido los nodos temporales (T1, T2, T3, T4, T5, T6 y T7) y el nodo base (B).

- Comentarios del Algoritmo Best First Search:
 - En la línea 5 se elige el mejor elemento utilizando la función objetivo BFS detallada en el punto 4.3.4.
 - De la línea 6 a la línea 11 se comienza a construir la ruta óptima para el vehículo, agregando siempre el nodo que tenga la mejor función objetivo BFS.
 - En la línea 15 el algoritmo retorna la ruta que seguirá el vehículo para ir desde B a T1, tal como se muestra en la figura 4.9.

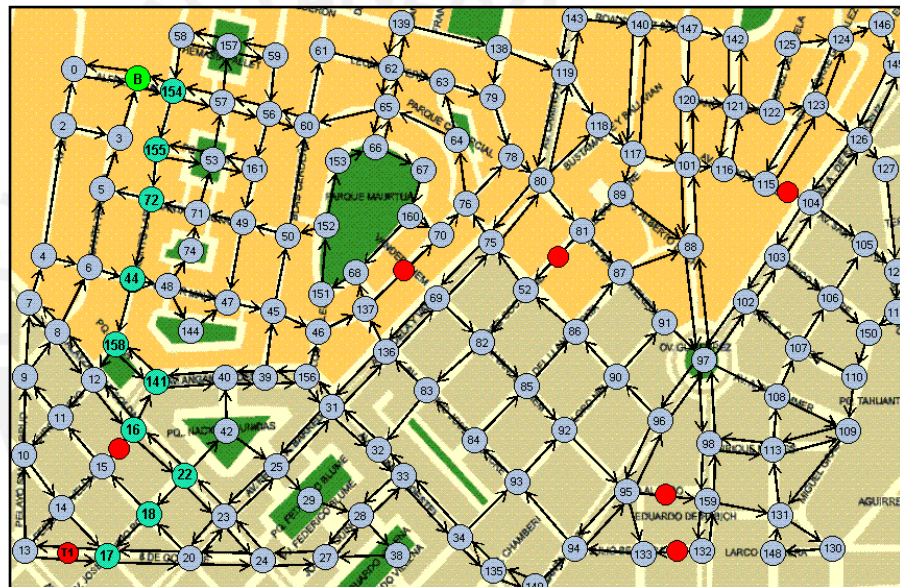


Figura 4.9: Ruta generada (Nodos 154-155-72-44-158-141-16-22-18-17) desde el nodo base (B) al nodo temporal T1.

- En este ejemplo pasar por la arista virtual conformada por los nodos B y T1 del grafo virtual significará en el grafo temporal pasar por los nodos, B, 154, 155, 72, 44, 158, 141, 16, 22, 18, 17 y T1.
- Este proceso se repite con los nodos restantes de la lista de nodos temporales.

4.4 Algoritmo GRASP que resuelve el problema de ruteo de vehículos con capacidad uniforme considerando la demanda compartida

4.4.1 Descripción

Algoritmo que proporciona una solución inicial al problema de ruteo de vehículos con capacidad uniforme, teniendo en consideración la demanda compartida en caso el cliente tenga una mayor demanda que el vehículo de reparto.

Dentro de la construcción del algoritmo GRASP se aplicará un segundo algoritmo GRASP que resolverá el problema de embalaje de paquetes en tres dimensiones (3D Bin Packing Problem) para optimizar el embalaje del vehículo, considerando el orden de reparto de los destinatarios.

4.4.2 Datos de Entrada

- GrafoVirtual: grafo conformado por el nodo inicio y los nodos temporales.
- L Vehiculos: es la lista de vehículos de reparto.
- L Destinatarios: es la lista de destinatarios.
- NumIter: es la cantidad de iteraciones que va realizar el algoritmo GRASPCVRP.
- cstRelaj: es la constante de relajación que determina que tan aleatorio o voraz es el algoritmo, si la constante de relajación es cercana a 0 entonces el algoritmo GRASP va ser más voraz que aleatorio, en caso contrario será más aleatorio.

4.4.3 Dato de Salida

El algoritmo devolverá la lista de vehículos actualizada, detallando por cada vehículo los destinatarios asignados y los paquetes embalados.

4.4.4 Función Objetivo CVRP

La siguiente función es utilizada para la construcción del algoritmo GRASPCVRP.

$$F_{(\text{objetivo CVRP})} = \frac{\text{Distancia} * \text{Constante Tráfico}}{F_{(\text{costo BPP3D})}}$$

Donde:

- Distancia: es la distancia de la arista virtual formada por el nodo actual y el nodo que se pretende recorrer. La medida es en metros.
- Constante Tráfico: es la constante de tráfico de la arista virtual, varía del 1 al 5, si esta es cercana a 1 significa que no hay mucho tráfico, en caso contrario significa que es una calle con mucho tráfico.
- $F_{(\text{costo BPP3D})}$: es la función de costo del algoritmo GRASP BPP3D, la cual busca la optimización del vehículo, (tanto volumen como peso) la fórmula se detalla en el punto 4.5.5.
- El objetivo es minimizar la función objetivo.

4.4.5 Función de Costo CVRP

La siguiente función se va aplicar para conocer la calidad de la solución del algoritmo

$$F_{(\text{Costo CVRP})} = \frac{\sum \text{Distancia} * \overline{\text{Constante tráfico}} * \text{Número de vehículos}}{F_{(\text{costo BPP3D})}}$$

Donde:

- $\sum \text{Distancia}$: es la sumatoria de la distancia recorrida en la ruta por todos los vehículos. La medida es en metros.
- $\overline{\text{Constante tráfico}}$: es el promedio de la constante de tráfico de todas las avenidas que conforman la ruta de los vehículos de reparto.

- Número de vehículos utilizados: es la cantidad de vehículos utilizados para realizar el reparto.
- $\overline{F}_{(costo_{BPP3D})}$: es el promedio de la función de costo del algoritmo GRASBP3D, la cual busca la optimización del vehículo, (tanto volumen como peso) la fórmula se detalla en el punto 4.5.5.
- El objetivo es minimizar la función de costo.

4.4.6 Algoritmo

A continuación se detalle el algoritmo GRASPCVRP en su etapa general y su etapa de construcción.

1	Procedimiento GRASPCVRP(GrafoVirtual, L Vehículos, L Destinatarios, NumIter, cstRelaj)
2	Para cada i desde 0 hasta NumIter hacer
3	NuevaL Vehículos ← GRASPCVRP Construcción (GrafoVirtual, L Vehículos, L Destinatarios, cstRelaj)
4	NuevaL Vehículos2OPT ← 2-OPTCVRP (GrafoVirtual, NuevaL Vehículos)
5	Si i = 0 entonces
6	actualizar(L Vehículos, NuevaL Vehículos2OPT)
7	Caso contrario
8	Si el costoCVRP(NuevaL Vehículos2OPT) < costoCVRP(L Vehículos) entonces
9	actualizar(L Vehículos, NuevaL Vehículos2OPT)
10	Fin Si
11	Fin Si
12	Fin Para
13	Retornar L Vehículos
14	Fin Procedimiento GRASPCVRP

Algoritmo 4.4: Algoritmo GRASPCVRP

1	Procedimiento GRASPCVRPConstruccion (GrafoVirtual, L Vehiculos, LDestinatarios, cstRelaj)
2	indiceVehiculo ← 0
3	Vehículo ← obtenerVehiculo(LVehiculos, indiceVehiculo)
4	Mientras noEntregadoTodosLosPaquetes(GrafoVirtual,LDestinarios) hacer
5	Mientras(noEsteCapacidadLlena(Vehículo) o entrePaquetes(Vehículo, LDestinatarios)) Hacer
6	RCL←generarListaCandidatosRestringida(β , τ , cstRelaj)
7	nodoElegido ← seleccionAleatoria(RCL)
8	agregarNodo(Vehículo, nodoElegido)
9	actualizarCapacidadVehiculo(Vehículo, nodoElegido)
10	Si todosPaquetesEntregados(nodoElegido) entonces
11	actualizarNodoEntregado(LDestinatarios, nodoElegido)
12	Si EntregadoTodosLosPaquetes(GrafoVirtual, LDestinatarios) Entonces
13	Actualizar(LVehiculos, Vehículo)
14	Retornar LVehiculos
15	Fin Si
16	Fin Si
17	Fin Mientras
18	Actualizar(LVehiculos, Vehículo)
19	indiceVehiculo ← indiceVehiculo+1
20	Vehículo ← obtenerVehiculo (LVehiculos, indiceVehiculo)
21	Fin Mientras
22	Retornar LVehiculos
23	Fin Procedimiento GRASPCVRPConstruccion

Algoritmo 4.5: Algoritmo GRASPCVRP Construcción

4.4.7 Ejemplo y Comentarios

Siguiendo el ejemplo anterior se tienen los siguientes datos de entrada:

- Grafo Virtual:

Grafo virtual tal como se muestra en la figura 4.10.

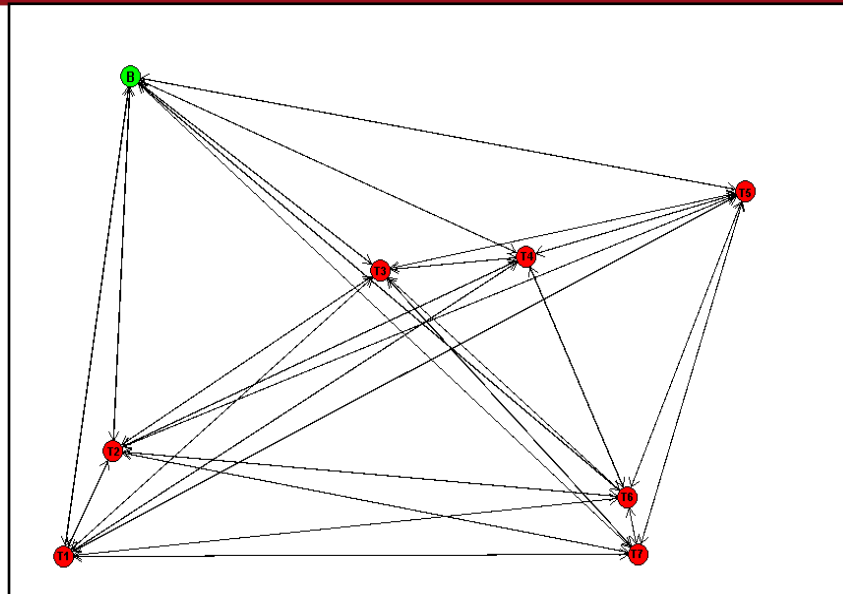


Figura 4.10: Grafo virtual formado por los nuevos nodos y el nodo base.

- L Vehículos:

Lista formada por 5 Vehículos (V1, V2, V3, V4, V5), sus medidas son 10 metros de largo, 5 metros de ancho, 5 metros de alto y un peso de carga de 25 Kg.

- L Destinatarios:

Lista de destinatarios tal como se muestra en la tabla 4.1.

Nombre del destinatario	Descripción del paquete	Ancho del paquete(m)	Largo del paquete(m)	Alto del paquete(m)	Peso del paquete(kg)
Jamiro	Paquete de comics	2	2	2	3.0
Jamiro	Paquete de CD's	2	3	2	2.0
Mary	Wii	2	2	2	2.0
Mary	Ps3	3	3	2	2.0
Pedro	Xbox360	1	2	3	2.0
Pedro	Paquete de celulares	3	2	3	2.0
Jamiro	Paquete de teclados	3	2	2	2.0
Mercedes	Paquete DP1	2	3	2	1.0
Mercedes	Paquete de cuadernos	3	2	2	2.0
Mercedes	Libro de algoritmos	2	1	3	3.0
Mercedes	Paquete de libros	4	2	3	3.0
Juan	Paquete CD's de	3	3	3	4.0

Juan	Jamiroquai				
Juan	Caja con fuegos artificiales	2	4	4	5.0
Juan	Caja de posters de Kiss	4	5	4	1.0
Juan	Guitarra	2	2	3	3.0
Juan	Bateria	1	3	4	2.0
Manuel	Caja de gorras	3	2	2	2.0
Manuel	Caja de chullos	4	3	3	3.0
Manuel	Caja de discos	2	4	2	4.0
Carmen	Manual de BI	2	3	1	2.0
Carmen	Manual de IA	2	1	2	3.0
Esther	Espada	3	3	3	4.0
Esther	Paquete de flautas	2	3	2	2.0
Esther	Paquete de arcos	3	2	3	2.0
Esther	Paquete de flechas	3	2	2	3.0
Esther	Paquete de lentes	3	2	2	3.0

Tabla 4.1: Datos de los productos por enviar.

- NumIter:

Se van a realizar 200 iteraciones.

- cstRelaj:

La constante de relajación va ser 0.8, quiere decir que la construcción del GRASP va ser más aleatoria que voraz.

- Comentarios del Algoritmo GRASPCVRP:

- En la línea 3 se invoca al algoritmo GRASPCVRPConstrucción, obteniendo la lista de vehículos con sus respectivos destinatarios y paquetes asociados.
- En la línea 4 se invoca al algoritmo 2-OPTCVRP el cual se explicará más adelante.

- Desde la línea 5 hasta la línea 10 se compara la solución optimizada por el algoritmo 2-OPTCVRP con la mejor solución almacenada. Para determinar si una solución es mejor que otra se compara la calidad de respuesta del algoritmo utilizando la función de costo CVRP descrita en el punto 4.4.5.
- El algoritmo en la línea 13 retorna la mejor lista de vehículos actualizada con los nodos temporales de destinatarios que toca repartir.
- Comentarios del Algoritmo GRASPCVRP Construcción:
 - En la línea 4 se detalla el criterio de parada del algoritmo, se va detener cuando todos los paquetes de los destinatarios hayan sido entregados.
 - En la línea 5 se muestra el criterio de parada por cada vehículo, se va a cambiar de vehículo cuando el vehículo esté totalmente lleno o no entren más paquetes.
 - En la línea 6 se genera la lista de candidatos que cumplan con el siguiente rango establecido:

$$\beta \leq F_{(\text{Objetivo CVRP})} \leq \beta + \alpha(\tau - \beta)$$

Donde:

- β : es el mejor valor de la función objetivo detallada en la sección 4.4.4.
- α : es la constante de relajación.
- τ : es el peor valor de la función objetivo detallada en la sección 4.4.4.
- En las siguientes líneas se procede a agregar los nodos temporales de destinatarios a los vehículos de reparto.
- Al finalizar, el algoritmo en la línea 22 retorna la lista de vehículos actualizada con los nodos temporales de destinatarios que toca repartir.
- Una posible solución a este ejemplo sería tal como se muestra en la tabla 4.2.

Vehículo	Destinatario	Paquetes embalados
V1	Mercedes	Paquete de libros
V1	Mercedes	Paquete de cuadernos
V1	Mercedes	Libro de algoritmos
V1	Mercedes	Paquete DP1
V1	Mary	Ps3
V1	Jamiro	Paquete de CD's
V2	Pedro	Paquete de celulares
V2	Pedro	Xbox360
V2	Juan	Caja de posters de Kiss
V2	Juan	Bateria
V2	Juan	Guitarra
V3	Jamiro	Paquete de teclados
V3	Jamiro	Paquete de comics
V3	Mary	Wii
V3	Esther	Espada
V3	Esther	Paquete de arcos
V3	Esther	Paquete de flechas
V4	Carmen	Manual de BI
V4	Carmen	Manual de IA
V4	Juan	Caja con fuegos artificiales
V4	Juan	Paquete CD's de Jamiroquai
V4	Esther	Paquete de lentes
V4	Esther	Paquete de flautas
V5	Manuel	Caja de chullos
V5	Manuel	Caja de discos
V5	Manuel	Caja de gorras

Tabla 4.2: Solución.

4.5 Algoritmo GRASP que resuelve el problema de embalaje de paquetes en tres dimensiones

4.5.1 Descripción

Algoritmo que proporciona una solución al problema de embalaje de paquetes en tres dimensiones (3D Bin Packing Problem).

4.5.2 Datos de Entrada

- Vehículo: es el vehículo donde se va embalar los paquetes.

- LPaquetes: es la lista de paquetes que se van a embalar.
- NumIter: es la cantidad de iteraciones que va realizar el GRASP.
- cstRelaj: es la constante de relajación que determinará que tan aleatorio o voraz es el algoritmo, si la constante de relajación es cercana a 1 entonces el algoritmo GRASP va ser más voraz que aleatorio, caso contrario el algoritmo será más aleatorio, ya que se trata de maximizar la función objetivo detallada en la sección 4.5.4.

4.5.3 Dato de Salida

El Algoritmo devolverá el vehículo con los paquetes ya embalados.

4.5.4 Función Objetivo BPP3D

$$F_{(Objetivo\ BPP3D)} = \frac{AnchoPaquete * Largo\ Paquete * Alto\ Paquete}{ACVehiculo * LCVehiculo * AltoVehiculo}$$

Donde:

- AnchoPaquete: es el ancho del paquete a embalar, en metros.
- LargoPaquete: es el largo del paquete a embalar, en metros.
- AltoPaquete: es el alto del paquete a embalar, en metros.
- ACVehiculo: es el ancho de la región de corte del vehículo a examinar, en metros.
- LCVehiculo: es el largo de la región de corte del vehículo a examinar, en metros.
- AltoVehiculo: es el alto del vehículo, en metros.
- El objetivo es maximizar la función objetivo.

4.5.5 Función de Costo BPP3D

$$F_{(Costo_{BPP3D})} = \left(\frac{(Vol_{Tp}/Vol_{veh}) * 2 + Peso_{Tp}/Peso_{veh}}{3} \right) * 100\%$$

Donde:

- Vol Tp: es el volumen utilizado al embalar los paquetes al vehículo.
- Vol veh: es el volumen total de espacio de almacenamiento del vehículo.
- Peso Tp: es la suma del peso de todos los paquetes embalados al vehículo.
- Peso veh: es el peso total de carga que posee el vehículo.
- Se divide entre 3 siendo 2 factores, duplicando el factor volumen con el fin de darle mayor prioridad en la función objetivo.
- Al final se multiplica (*100%) dando de esta manera el porcentaje de optimización del vehículo
- Esta fórmula se aplica en el denominador tanto de la función objetivo como en la función de costo del algoritmo GRASPCVRP.
- El objetivo es maximizar la función de costo.

4.5.6 Algoritmo

A continuación se detalla el algoritmo GRASPBPP3D en su etapa general y su etapa de construcción.

1	Procedimiento GRASPBPP3D(Vehículo, LPaquetes, Numlter, cstRelaj)
2	Copiar(NuevoVehiculo,Vehículo)
3	Para cada i desde 0 hasta Numlter hacer
4	NuevoVehiculo←GRASPBPP3DConstruccion(NuevoVehiculo, LPaquetes, cstRelaj)

5	Si $i = 0$ entonces
6	ActualizarEmbalaje(Vehículo, NuevoVehiculo)
7	Caso contrario
8	Si el costoBPP3D (Vehículo) < costoBPP3D(NuevoVehiculo) entonces
9	ActualizarEmbalaje (Vehículo, NuevoVehiculo)
10	Fin Si
11	Fin Si
12	Fin Para
13	Retornar Vehículo
14	Fin Procedimiento GRASBP3D

Algoritmo 4.6: Algoritmo GRASBP3D

1	Procedimiento GRASBP3DConstruccion (Vehículo, LPaquetes, cstRelaj)
2	corteLargoInicial ← obtenerCorteLargoInicial(Vehículo)
3	anchoIndice ← 0
4	corteLargoFinal ← obtenerMayorLargo(Lpaquetes)
5	Mientras (hayaPaquetes(LPaquetes) y hayaEspacio(Vehículo, LPaquetes) y entrePaqueteporPeso(Lpaquetes, Vehiculo))
6	Mientras (AlcanceAnchoVehiculo(Vehículo, anchoIndice) y hayaPaquetes(Lpaquetes) y entrePaqueteporPeso(Lpaquetes, Vehículo))
7	RCL ← generarListaCandidatosRestringida(mejorV, peorV, cstRelaj)
8	paqueteElegido ← seleccionAleatoria(RCL)
9	embalarPaquete(Vehículo, paqueteElegido)
10	Si huboRotacion(paqueteElegido) entonces
11	anchoÍndice ← anchoÍndice + obtenerLargo(paqueteElegido)
12	Sino
13	anchoÍndice ← anchoÍndice + obtenerAncho(paqueteElegido)
14	EliminarPaquete (LPaquetes, paqueteElegido)
15	Fin Mientras
16	anchoÍndice ← 0
17	corteLargoInicial ← actualizarcorteLargoInicial(Vehículo)
18	corteLargoFinal ← obtenerMayorLargo(Lpaquetes)
19	Fin Mientras
20	Retornar Vehículo
21	Fin Procedimiento GRASBP3DConstruccion

Algoritmo 4.7: Algoritmo GRASBP3D Construcción

4.5.7 Ejemplo y Comentarios

Siguiendo el ejemplo anterior se tienen los siguientes datos de entrada:

- Vehículo:

Vehículo inicial V1 con medidas de 10 metros de largo, 5 metros de ancho, 5 metros de alto y un peso de carga de 25 Kg.

- LPaquetes:

Lista de paquetes tal como se muestra en la tabla 4.3.

Descripción del paquete	Ancho del paquete	Largo del paquete	Alto del paquete	Peso del paquete
Paquete de comics	2	2	2	3.0
Paquete de CD's	2	3	2	2.0
Wii	2	2	2	2.0
Ps3	3	3	2	2.0
Xbox360	1	2	3	2.0
Paquete de celulares	3	2	3	2.0
Paquete de teclados	3	2	2	2.0
Paquete DP1	2	3	2	1.0
Paquete de cuadernos	3	2	2	2.0
Libro de algoritmos	2	1	3	3.0
Paquete de libros	4	2	3	3.0
Paquete CD's de Jamiroquai	3	3	3	4.0
Caja con fuegos artificiales	2	4	4	5.0
Caja de posters de Kiss	4	5	4	1.0

Tabla 4.3: División de paquetes.

- NumIter:

Se van a realizar 400 iteraciones.

- cstRelaj:

La constante de relajación va ser 0.7, quiere decir que la construcción del GRASP va ser más voraz que aleatoria.

- Comentarios del Algoritmo GRASPBPP3D:
 - En la línea 2 Se realiza una copia exacta de las características del vehículo al nuevo vehículo.
 - En la línea 4 se invoca al algoritmo GRASPBPP3DConstruccion, obteniendo el vehículo con los paquetes embalados.
 - Desde la línea 5 al 11 se compara con la mejor solución almacenada. Para determinar si una solución es mejor que otra se compara la calidad de respuesta del algoritmo utilizando la función de costo BPP3D descrita en el punto 4.5.5.
 - El algoritmo en la línea 13 retorna el vehículo con la mejor forma encontrada de embalar los productos.
 - Cabe resaltar que para este algoritmo la cantidad de iteraciones va ser mayor que las iteraciones del algoritmo GRASPCVRP debido a que en este algoritmo GRASP no va ver fase de mejoría.
- Comentarios del Algoritmo GRASPBPP3DConstruccion:
 - El algoritmo GRASPBPP3DConstrucción trabaja de la siguiente manera:
 - Se realizan cortes al largo del vehículo.
 - En cada corte se maximiza el espacio.
 - Para ello se trata de rotar los paquetes horizontalmente (ancho por largo del paquete y viceversa) para que se utilice el mayor largo y el menor ancho del vehículo posible (Los vehículos por lo general tienen mayor medida de largo que de ancho).
 - Siempre se respeta el orden de los destinatarios, esto quiere decir que en las primeras líneas deben estar los paquetes de los nodos temporales que primero se van a visitar.

- Además los paquetes de un mismo nodo temporal deben estar juntos para que sea más fácil y rápido el despacho de los paquetes al destinatario.
- En la línea 2 se obtiene el corte inicial del vehículo, si el vehículo es nuevo el corte inicial es 0, en caso contrario puede ser desde 1 hasta el tamaño del largo del vehículo -1 (siempre el corte de valor entero).
- En la línea 3 se inicia el ancho índice en 0, este índice va indicar el ancho utilizado en el corte actual del vehículo.
- En la línea 4 se obtiene la mayor medida de largo de los paquetes que aún faltan embalar, en caso sea necesario se rotarán los paquetes horizontalmente (ancho por largo) para obtener la mayor medida.
- En la línea 5 se detalla el criterio de parada del algoritmo, se va detener cuando:
 - Ya no existan paquetes por embalar.
 - Ya no exista espacio disponible en el vehículo.
 - Ya no entren más paquetes por sobrecarga.
- En la línea 6, se detalla el criterio de parada para el corte actual del vehículo, se va detener cuando:
 - Ya no existan paquetes por embalar.
 - Ya no alcance más paquetes en ese corte.
 - Sobrecarga de paquetes en el vehículo.
- En la línea 7 se genera la lista de candidatos, que cumplan con el siguiente rango establecido:

$$\tau + \alpha(\beta - \tau) \leq F_{(Objetivo CVRP)} \leq \beta$$

Donde:

- β : es el mejor valor de la función objetivo detallada en la sección 4.5.4.
- α : es la constante de relajación.
- τ : es el peor valor de la función objetivo detallada en la sección 4.5.4.

- En las siguientes líneas se procede a embalar los paquetes dentro del vehículo, actualizando el anchoÍndice, verificando si el paqueteElegido ha sido rotado y además actualizando el corteLargoInicial y corteLargoFinal con los paquetes embalados y los paquetes por embalar.
- El algoritmo en la línea 20 devuelve el vehículo con los paquetes ya embalados.
- Una posible solución al ejemplo sería tal como se muestra en la tabla 4.4.

Vehículo	Paquetes embalados	Peso del paquete
V1	Paquete de libros	3
V1	Paquete de cuadernos	2
V1	Libro de algoritmos	3
V1	Paquete DP1	1

Tabla 4.4: Solución de embalajes.

4.6 Algoritmo de mejoría 2-OPTCVRP que optimiza la solución generada por el algoritmo GRASPCVRP

4.6.1 Descripción

Algoritmo que mejora la solución obtenida por el algoritmo GRASPCVRP, intercambiando el orden de reparto de los nodos temporales de destinatarios por vehículo.

4.6.2 Datos de Entrada

- GrafoVirtual: grafo conformado por el nodo inicio y los nodos temporales.
- L Vehiculos: es la lista de vehículos de reparto.

4.6.3 Dato de Salida

El algoritmo devolverá la lista de vehículos optimizando el orden de reparto a los nodos temporales.

4.6.4 Algoritmo

1	Procedimiento 2-OPTCVRP (GrafoVirtual, L Vehiculos)
2	$N \leftarrow \text{Cantidad_Vehiculos}(\text{LVehiculos})$
3	Para $i \leftarrow 0$ hasta N hacer
4	$\text{LNodos} \leftarrow \text{Obtener_Nodos}(\text{LVehiculos}, i)$
5	$\text{LNodosMejor} \leftarrow \text{LNodos}$
6	$M \leftarrow \text{Cantidad_Nodos}(\text{LNodos})$
7	$\text{MejorCosto} \leftarrow \text{Obtener_Costo}(\text{LNodos}, \text{GrafoVirtual})$
8	Para $j \leftarrow 0$ hasta M hacer
9	Para $k \leftarrow j+1$ hasta M hacer
10	$\text{Nodok} \leftarrow \text{LNodos}(k)$
11	$\text{Borrar_Nodo}(\text{LNodos}, \text{posk})$
12	$\text{Insertar_Nodo}(\text{LNodos}, \text{posj}, \text{Nodok})$
13	$\text{Costo} \leftarrow \text{Obtener_Costo}(\text{LNodos}, \text{GrafoVirtual})$
14	Si $(\text{Costo} < \text{MejorCosto})$ entonces
15	$\text{LNodosMejor} \leftarrow \text{LNodos}$
16	$\text{MejorCosto} \leftarrow \text{Costo}$
17	Fin Si
18	Fin Para
19	$\text{LNodos} \leftarrow \text{Obtener_Nodos}(\text{LVehiculos}, i)$
20	Fin Para
21	$\text{Asignar_NuevaLista}(\text{LVehiculos } i, \text{LNodosMejor})$
22	Fin Para
23	Retornar LVehiculos
24	Fin Procedimiento 2-OPTCVRP

Algoritmo 4.8: Algoritmo 2-OPTCVRP

4.6.5 Ejemplo y Comentarios

Siguiendo el ejemplo anterior, se tienen los siguientes datos de entrada.

- Grafo Virtual

Grafo virtual tal como se muestra en la figura 4.11.

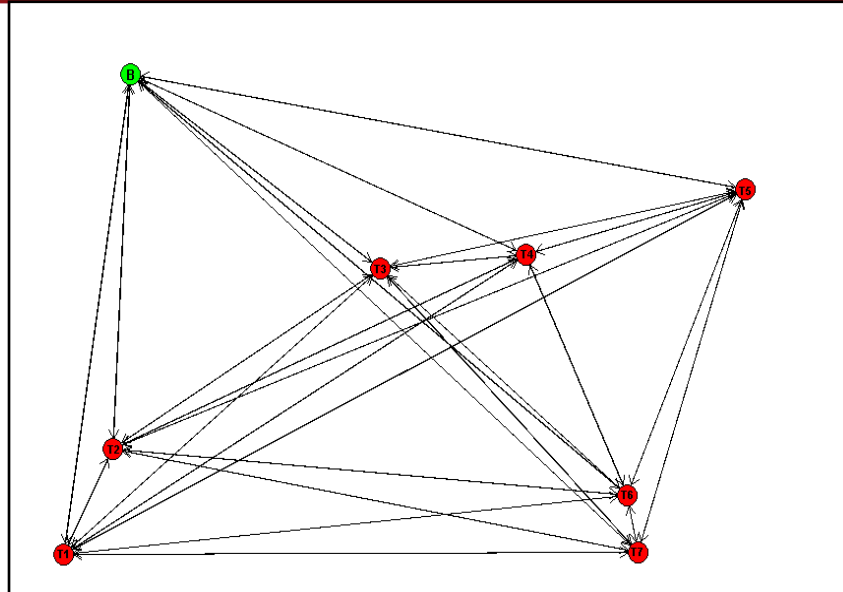


Figura 4.11: Grafo virtual formado por los nuevos nodos y el nodo base.

- L Vehículos:

Lista de vehículos tal como se muestra en la tabla 4.5.

Vehículo	Ruta a Seguir
V1	B → T2 → T3 → B
V2	B → T7 → B
V3	B → T1 → T5 → T4 → B
V4	B → T2 → T6 → T7 → B
V5	B → T5 → T7 → B

Tabla 4.5: Rutas de vehículos.

- Comentarios del Algoritmo 2-OPTCVRP:
 - En la línea 2 se obtiene la cantidad de vehículos que se utilizó en el algoritmo GRASPCVRP.
 - En la línea 4 se obtienen los nodos temporales que recorrerá el vehículo y en la línea 5 se hace una copia de dichos nodos.
 - En la línea 6 se obtiene el costo utilizando la función de costo CVRP detallada en la sección 4.4.5.

- Desde la línea 8 al 20 se hace un doble recorrido por los nodos temporales del vehículo de índice i , intercambiando todas las posiciones de los nodos y formando nuevas aristas en el cambio, por cada cambio se obtiene el costo y se compara con el costo almacenado, si este es mejor, entonces se actualiza el mejor costo y la lista de mejores nodos.
- En la línea 21 se actualiza la lista del nodo del vehículo de índice i con la mejor lista obtenida, además se actualiza el orden de los paquetes dentro del vehículo.
- Luego de recorrer todos los vehículos, el algoritmo en la línea 23 retorna la lista de vehículos con las rutas actualizadas.
- Una posible solución al ejemplo sería tal como se muestra en la tabla 4.6.

Vehículo	Ruta a Seguir
V1	B → T3 → T2 → B
V2	B → T7 → B
V3	B → T4 → T1 → T5 → B
V4	B → T2 → T7 → T6 → B
V5	B → T5 → T7 → B

Tabla 4.6: Nuevas rutas de vehículos.

CAPÍTULO 5: RESULTADOS DE LA EXPERIMENTACIÓN NUMÉRICA

En este capítulo se analizará los resultados obtenidos por la experimentación numérica.

5.1 Objetivo de la experimentación numérica

En este proyecto se utilizó dos algoritmos GRASP, uno en la fase de construcción de rutas (CVRP) y otro en la fase de optimización de la capacidad del vehículo (BPP3D).

El objetivo del experimento es comparar los dos algoritmos GRASP y la optimización 2-Opt (Acrónimo G) contra los siguientes algoritmos simulados que se muestra en la tabla 5.1.

Fase CVRP	Fase BPP3D	Acrónimo
Algoritmo Voraz	Algoritmo Voraz	VV
Algoritmo Aleatorio	Algoritmo Aleatorio	AA
Algoritmo Voraz	Algoritmo Aleatorio	VA
Algoritmo Aleatorio	Algoritmo Voraz	AV

Tabla 5.1: Acrónimos de los algoritmos simulados.

Los algoritmos se simularán de acuerdo a la constante de relajación del algoritmo GRASP, logrando que el algoritmo simulado se comporte de

manera voraz, de manera aleatoria y de manera híbrida, considerando solo una iteración en su solución. Por lo tanto los algoritmos simulados tendrán un menor tiempo de ejecución que el algoritmo GRASP, ya que el algoritmo GRASP va depender de la cantidad de iteraciones de la fase CVRP, de la fase BPP3D y de la optimización 2-Opt. La experimentación numérica se encuentra en los anexos de este proyecto.

5.2 Resultados

En todos los casos se realizaron 50 iteraciones del algoritmo GRASPCVRP y 100 iteraciones del algoritmo GRASPBPP3D, los resultados obtenidos se muestra en la tabla 5.2.

Algoritmo	#Grafos	#Nodos	#Destinatarios	Constantes GRASP		Media
				CVRP(α)	BPP3D(α)	
G	100	50	5	0.8	0.6	43.1289
VV	100	50	5	0.8	0.6	61.2622
VA	100	50	5	0.8	0.6	78.292
AV	100	50	5	0.8	0.6	68.2727
AA	100	50	5	0.8	0.6	88.8891
G	100	100	10	0.2	0.8	168.571
VV	100	100	10	0.2	0.8	217.097
VA	100	100	10	0.2	0.8	318.184
AV	100	100	10	0.2	0.8	267.48
AA	100	100	10	0.2	0.8	368.334
G	100	150	15	0.2	0.6	422.663
VV	100	150	15	0.2	0.6	547.359
VA	100	150	15	0.2	0.6	752.957
AV	100	150	15	0.2	0.6	688.805
AA	100	150	20	0.2	0.6	961.998
G	100	200	20	0.2	0.8	873.382
VV	100	200	20	0.2	0.8	1,066.65
VA	100	200	20	0.2	0.8	1,521.05
AV	100	200	20	0.2	0.8	1,447.77
AA	100	200	20	0.2	0.8	2,063.14

Tabla 5.2: Resumen de resultados de los algoritmos.

Con un 95% de nivel de confianza se puede concluir lo siguiente:

- Los dos algoritmos GRASP con fase de optimización 2-Opt brindan mejores soluciones que un algoritmo voraz, un algoritmo aleatorio y un

algoritmo híbrido para el problema de ruteo de vehículo con capacidad uniforme considerando la demanda compartida.

- En la mayoría de casos el algoritmo voraz brinda una buena solución al problema de ruteo de vehículos con capacidad uniforme considerando la demanda compartida.
- El algoritmo aleatorio siempre obtiene las peores soluciones al problema de ruteo de vehículos con capacidad uniforme considerando la demanda compartida.



CAPÍTULO 6: OBSERVACIONES, CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS.

En este capítulo se describirá las observaciones, conclusiones, recomendaciones y trabajos futuros del proyecto.

6.1 Observaciones

- Al implementar el algoritmo GRASPCVRP, se consideró como caso especial la demanda compartida (SDVRP), este caso se utiliza cuando la demanda del cliente sobrepasa la capacidad del vehículo, además considerando este caso se logra utilizar de una manera más eficiente la capacidad de los vehículos.
- Al implementar el algoritmo GRASPBPP3D, se realizó una mayor cantidad de iteraciones que el algoritmo GRASPCVRP, esto se debe a que el algoritmo GRASPBPP3D no cuenta con fase de mejoría por ello es necesario realizar más iteraciones, lo cual demanda una mayor cantidad de tiempo.
- Al implementar el algoritmo GRASPBPP3D se consideró el orden de los destinatarios, esto quiere decir que en las primeras filas del vehículo

deben estar los paquetes de los nodos temporales de destinatarios que primero se van a visitar, esta consideración hace que el problema sea más real en su aplicación.

- Al implementar el algoritmo 2-OPTCVRP, se realizan intercambios en la ruta de los vehículos (en el orden de visita de los nodos temporales), esto conlleva a actualizar el orden de los paquetes dentro del vehículo, respetando de esta manera el orden de reparto.

6.2 Conclusiones

- Un algoritmo Meta Heurístico produce mejores soluciones que un algoritmo Heurístico debido a que este explora un mayor espacio de soluciones, esto se debe a que un algoritmo Meta Heurístico no se encierra en su óptimo local, sino que gracias a la aleatoriedad hace que se logren explorar más regiones del problema y así encontrar mayores y mejores soluciones.
- Un algoritmo GRASP Construcción (GRASPCVRP) permite obtener una solución factible al problema de ruteo de vehículos con capacidad uniforme considerando la demanda compartida.
- Un algoritmo GRASP Mejoría 2-Opt permite optimizar la solución inicial del problema de ruteo de vehículos con capacidad uniforme considerando la demanda compartida.
- Un algoritmo GRASP Construcción (GRASPBPP3D) permite obtener una solución factible al problema de embalaje de paquetes en tres dimensiones.
- Los dos algoritmos GRASP con la fase de mejora 2-Opt obtienen mejores soluciones que un algoritmo totalmente aleatorio, un algoritmo totalmente voraz y un algoritmo híbrido (voraz y aleatorio).

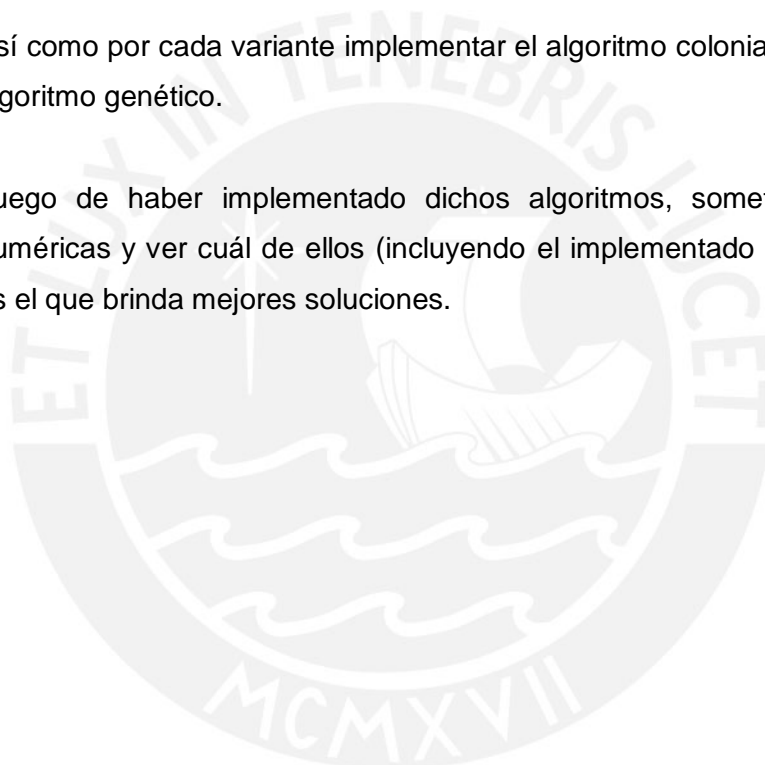
6.3 Recomendaciones y trabajos futuros

Luego de investigar el tema de ruteo de vehículos, es recomendable investigar la aplicación que este tiene en la realidad. A lo largo de esta tesis se ha explicado la aplicabilidad y los beneficios que otorgaría implementar sistemas inteligentes que hallen rutas óptimas considerando la capacidad del vehículo. Sería recomendable tomar este sistema de prueba implementado como núcleo para sistemas más complejos (a nivel funcional).

Como trabajo futuro se investigará más variantes del problema de ruteo de vehículos, en particular las variantes de: ventanas de tiempo, múltiples depósitos y diferentes capacidades del vehículo.

Así como por cada variante implementar el algoritmo colonia de hormigas y el algoritmo genético.

Luego de haber implementado dichos algoritmos, someterlos a pruebas numéricas y ver cuál de ellos (incluyendo el implementado en este proyecto) es el que brinda mejores soluciones.



BIBLIOGRAFIA

1. [PAPADIMITRIOU, 1982]
Papadimitriou, Christos H.
Combinatorial Optimization.
Prentice Hall.
1982
2. [CHRISTOFIDES, 1978]
Christofides, Louis.
Combinatorial Optimization.
1978.
3. [HILLIERLIEB, 1997]
Hillier F, Lieberman
Introducción a la investigación de operaciones.
McGraw-Hill.
México.
1997.
4. [MOSCATO, 1996]
Moscato, Pazos.
Optimización de heurísticas y redes neuronales.
Paraninfo
1996
España
5. [BALL, 1981]
Ball, M.
The design and analisis of heuristic
Networks
1981
6. [FEO; RESENDE, 1995]
Greedy Randomized Adaptive Search Procedure
Journal of Global Optimization
1995

7. [IEEE, 2003]
Institute of Electric and Electronic Engineering
Standar Collection: Software Engineering.
2003
8. [P. TOTH; D. VIGO, 2001]
"The Vehicle Routing Problem".
Monographs on Discrete Mathematics and Applications.
SIAM,
Philadelphia. 2001
9. [SCHULZE, 2007] Dr. Jürgen Schulze
<<http://wwwcs.uni-paderborn.de/cs/ag-monien/PERSONAL/SCHLUNZ/research.html>>
Universidad de Paderborn
10. [U.REPUBLICA, 2007] UNIVERSIDAD REPÚBLICA DE URUGUAY
<<http://www.fing.edu.uy/inco/grupos/invop/dtr/material/claseMetaheurísticas.pdf>>
11. [U.REPUBLICAB, 2007] UNIVERSIDAD REPÚBLICA DE URUGUAY-
BIBLIOTECA
Olivera, Alfredo
<<http://www.fing.edu.uy/inco/pedeciba/bibliote/reptec/TR0408.pdf> >
12. [U.MÁLAGA, 2009] UNIVERSIDAD DE MÁLAGA
<<http://neo.lcc.uma.es/radi-aeb/WebVRP/>>
13. [CONTARDO, 2005] FORMULACIÓN Y SOLUCIÓN DE UN
PROBLEMA DE RUTEO DE VEHÍCULOS CON DEMANDA
VARIABLE EN TIEMPO REAL, TRASBORDOS Y VENTANAS DE
TIEMPO.
Contardo, Claudio
<http://www.crt.umontreal.ca/~ccontard/claudio_contardo_thesis.pdf>
14. [RAMOS, 2007] OPTIMIZACION LINEAL ENTERA MIXTA

Ramos, Andrés

<http://www.doi.icaei.upcomillas.es/simio/transpa/t_mip_ar.pdf>

15. [CALLES, 2009] Guía de calles

<<http://www.guiacalles.com/pcruta.shtml>>

16. [GRAFOS, 2009] Sistema de grafos

<<http://personales.upv.es/arodrigu/grafos/>>

17. [U. TRIER, 2009] Computer Science Bibliography

<<http://www.informatik.uni-trier.de/~ley/db/index.html>>

18. [DSI, 2002] Departamento de sistemas informáticos

<<http://www.infoab.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf>>

19. [PX, 2004] Programación Extrema

<<http://www.programacionextrema.org/>>

20. [GOOGLE MAPS, 2009] Google Maps

<<http://maps.google.es/>>