PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

RESOLUCIÓN DEL VRP CON LOGÍSTICA REVERSA A TRAVÉS DE UN SISTEMA DE COLONIA DE HORMIGAS

FELIPE ANDRÉS PERALTA VALDÉS

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO CIVIL EN INFORMATICA

(DICIEMBRE 2008)

Pontificia Universidad Católica de Valparaíso
Facultad de Ingeniería
Escuela de Ingeniería Informática

RESOLUCIÓN DEL VRP CON LOGÍSTICA REVERSA A TRAVÉS DE UN SISTEMA DE COLONIA DE HORMIGAS

FELIPE ANDRÉS PERALTA VALDÉS

Profesor Guía: **Guillermo Cabrera Guerrero**

Profesor Co-referente: **Broderick Crawford Labrín**

Carrera: **Ingeniería Civil en Informática**

(Diciembre 2008)

I Dedicatorias

A Dios ya que sin Él ningún logro de mi vida hubiera sido posible, y a mis padres, el otro pilar fundamental de mi vida, por su amor, apoyo y comprensión a lo largo de todos estos años.

Felipe Andrés Peralta Valdés

II Agradecimientos

Agradezco a Dios y a mis padres por su amor incondicional y por darme las herramientas necesarias para lograr todos mis objetivos en la vida.

A la Pontificia Universidad Católica de Valparaíso por entregarme los conocimientos necesarios a lo largo de mi carrera y darme la oportunidad de ser una persona con formación profesional.

Felipe Andrés Peralta Valdés

Resumen

En el presente proyecto se ha resuelto una variante del conocido problema de ruteo de vehículos (VRP) el cual busca obtener las rutas que generen un menor gasto, ya sea de tiempo o distancia, para una flota de vehículos que deben visitar a una cierta cantidad de clientes. Esta variante utiliza logística reversa y se denomina VRP con entrega y recogida simultánea (VRPSPD) que significa que los clientes pueden requerir, al mismo tiempo, recibir una cierta cantidad de bienes como también devolver otros. Este problema se ha implementado mediante un Sistema de Colonia de Hormigas y como último paso se han utilizado instancias de la literatura para el VRP en un primer momento y, luego, para el VRPSPD, para ver la calidad de los resultados obtenidos y compararlos con los de otros autores.

Palabras claves: Vehicle Routing Problem, VRP, Logística Reversa, VRPSPD, Sistema de Colonia de Hormigas.

Abstract

In this project has been resolved a variant of the known Vehicle Routing Problem (VRP) which seeks routes that generate less expenditure either time or distance, for a fleet of vehicles that must visit a certain amount of customers. This variant uses reverse logistic and is called VRP with simultaneous pickup and delivery (VRPSPD) which means that customers may require, at the same time, to receive a certain quantity of goods as also return other. This problem has been implemented through an Ant Colony System and as a last step have been used instances from the literature for the VRP at first and then for VRPSPD to see the quality of the results obtained and compare them with from the others authors.

Keywords: Vehicle Routing Problem, VRP, Reverse Logistic, VRPSPD, Ant Colony System.

Índice

Índice de figuras

Lista de tablas

1 Introducción

En los últimos años un incremento en la conciencia ambiental ha creado la necesidad de proteger el medioambiente, por lo que se están creando leyes para que las empresas cumplan con esta conciencia, y éstas en su afán de reducción costos, como por ejemplo, en la disminución del gasto de energía, también ayudan a cumplir este objetivo. Debido a esto surge la necesidad de que las empresas empiecen a tratar de recuperar sus productos desde los usuarios finales, esto para reciclaje, remanufactura, reparación, reutilización o tomar el producto de vuelta cuando ha cumplido su ciclo de vida para su correcta eliminación. Este problema ha hecho que a la logística tradicional (llamada logística forward) de llevar los productos hacia los clientes (los ya mencionados usuarios finales) se le agregue la logística reversa que los trae de vuelta a las empresas. Los clientes tienen la necesidad que ambas tareas se realicen en forma simultánea si es posible, es decir, si éste desea recibir un producto y devolver otro deseará que la empresa satisfaga ambas necesidades simultáneamente, (Dethloff, 2001). Un ejemplo claro de esto es una empresa de productos electrónicos donde un cliente compra un equipo nuevo a través de Internet y a la vez quiere devolver otro ya sea para repararlo o eliminarlo, en este caso la empresa en una sola visita a éste debe entregarle un equipo y recibir otro.

Las empresas deben atender a las necesidades de muchos clientes respecto a la distribución de sus productos por lo que deben manejar de una manera eficiente las visitas que se realizan como también las cantidades de productos a repartir a éstos, ante lo cual la mejor forma de expresar este problema es a través del VRP (Vehicle Routing Problem). El funcionamiento básico es el siguiente: se debe resolver cada ruta de un grupo de vehículos que salen desde un mismo depósito central para entre ellos visitar a todos los clientes por lo que se le asigna a cada vehículo un grupo de éstos, los cuales deben ser visitados sólo una vez y finalmente los vehículos deben volver al depósito. Estos generalmente tienen la misma capacidad de transporte. La idea principal es reducir la distancia total o el tiempo total de viaje de los vehículos para realizar esta tarea, (Barajas, 2006).

Existen distintas variantes del VRP, en este caso, el enfoque ha sido orientado a utilizar logística reversa, para dar solución al problema planteado anteriormente, para lo

cual existe una variante llamada Vehicle Routing Problem with Simultaneous Pick-up and Delivery (VRPSPD) que es el VRP con Entrega y Recogida Simultánea, lo que quiere decir que los clientes pueden tanto entregar como recibir bienes en una única visita por parte del vehículo.

El VRP es uno de los problemas más conocidos en optimización combinatoria, la forma del problema no nos permite llegar a una solución óptima con el uso de una cantidad razonable de recursos, (Barajas, 2006). Los métodos exactos encuentran la solución óptima, pero a un costo de recursos y cantidad de tiempo enormes para problemas de gran tamaño. El costo de encontrar la solución óptima crece exponencialmente a medida que aumenta el tamaño del problema, (Díaz, 2006), ante lo cual se prefiere usar métodos heurísticos que encuentran soluciones cercanas al óptimo con un costo de recursos y cantidad de tiempo razonables.

Para este proyecto se ha utilizado la metaheurística Optimización basada en Colonia de hormigas (ACO por su nombre en inglés Ant Colony Optimization) para resolver tanto el VRP como la variante VRPSPD. Este método se basa en el comportamiento natural de las hormigas para buscar su alimento. Se sabe que las hormigas logran encontrar los caminos más cortos entre su hormiguero y la comida aunque en muchas de sus especies su visión sea casi nula, (Alonso *et al.*, 2004). Las hormigas se mueven en forma aleatoria para buscar el alimento dejando a su paso una cantidad de feromona que es captada por las demás hormigas las cuales en forma probabilística tienden a seguir los rastros donde hay mayor cantidad de esta sustancia. Las hormigas que encuentran el alimento vuelven inmediatamente al hormiguero dejando a su paso un rastro de feromona que poco a poco se hace más fuerte gracias a las hormigas que siguieron este rastro. Esto continúa hasta que todas ellas, por la gran cantidad de feromona existente, siguen este camino finalmente, (Barajas, 2006). Se puede encontrar información en Dorigo y Gambardella (1997) donde se presentó por primera vez el algoritmo Sistema de Colonia de Hormigas, el que se ha utilizado para resolver tanto el VRP como la variante VRPSPD. En Zabala y Torres (2005) se ha resuelto el VRP con restricciones de capacidad y ventanas de tiempo (que se explica en la sección 3.4) a través del Sistema de Colonia de Hormigas.

El trabajo de título está dividido de la siguiente manera: el capítulo 1 es la introducción, en el capítulo 2 se plantean los objetivos y la metodología de trabajo a

utilizar, después en el capítulo 3, llamado estado del arte, se presenta el VRP, una breve descripción de sus variantes e información relacionada a este problema, luego se explica en detalle la variante VRPSPD, y a continuación la metaheurística Optimización basada en Colonia de Hormigas como también el algoritmo Sistema de Colonia de Hormigas. El capítulo 4, llamado solución propuesta, explica la implementación hecha del VRP, luego se muestra pruebas realizadas a esta implementación con instancias de la literatura, a continuación se muestra el trabajo realizado para resolver la variante VRPSPD, a través de modificaciones a la implementación del VRP, esto con una instancia pequeña y finalmente este capítulo muestra los resultados del VRPSPD para instancias de la literatura. Luego en el capítulo 5 se muestran las conclusiones del trabajo realizado y finalmente en el 6 se muestran las referencias utilizadas. Además al final de este trabajo de título se agregan un apéndice y dos anexos, a los cuales se hará referencia más adelante.

2 Objetivos y metodología de trabajo

2.1 Objetivo general

Resolución, a través de un Sistema de Colonia de Hormigas, de una variante del VRP, la cual maneja logística reversa, llamada VRPSPD.

2.2 Objetivos específicos

- Obtención de una base de conocimiento sobre VRP, VRPSPD, Optimización basada en Colonia de Hormigas e instancias de la literatura.
- Elaboración de un modelo para resolver el VRP con un Sistema de Colonia de Hormigas, para ser adaptado a la resolución de VRPSPD.
- Implementación en lenguaje C de los modelos propuestos para VRP y VRPSPD.
- Obtención de resultados, de la minimización de la distancia total recorrida por una flota de vehículos, a partir de instancias conocidas y comparación de la calidad de estos resultados con los mejores que existen en la actualidad.

2.3 Metodología de trabajo

La metodología de trabajo es tomada de Perissé (2001), donde explican que, toda actividad debe estar basada en una metodología y en principio, cualquier metodología es mejor que ninguna; lo más práctico es seguir los métodos que ya han demostrado su validez y son de aplicación universal.

Todas las metodologías; MERISE, YOURDON Y SSADM (Structured Systems Analysis and Design Method) y tantas otras, consideran el hecho informático dividido en fases, cuyo conjunto forma el ciclo de vida de un sistema informático.

Todas tienen en común la idea de descomposición del hecho informático en cuatro grandes grupos:

- **Análisis**
    - Definición del problema
    - Estudio de la situación actual
    - Requisitos a considerar
    - Estudio de factibilidad
- **Diseño lógico**
    - Análisis funcional
    - Definición de datos y procesos
    - Modelización
- **Diseño físico**
    - Creación de ficheros y tablas
    - Elaboración de programas
- **Implementación y control**
    - Formación del usuario
    - Implantación del sistema
    - Explotación del sistema
    - Mantenimiento

Esta metodología es la que representa de mejor forma la elaboración de este proyecto informático, ya que refleja los pasos a seguir para concretar con éxito la elaboración de un sistema de software.

2.4 Plan de trabajo

Las siguientes son las tareas que se realizaron a lo largo de proyecto 1 y 2.

1. Investigación de información sobre VRP, VRPSPD, la metaheurística Optimización basada en Colonia de Hormigas y más específicamente el Sistema de Colonia de Hormigas, e instancias de prueba de la literatura sobre VRP y VRPSPD.

2. Obtención de un modelo del VRP con Sistema de Colonia de Hormigas para ser implementado.
3. Resolución de VRP con Sistema de Colonia de Hormigas.
4. Obtención de un prototipo de programa en lenguaje C que resuelva el VRP, luego de realizar pruebas y evaluar resultados con instancias de la literatura.
5. Adaptación del modelo de VRP a la variante VRPSPD.
6. Resolución del VRPSPD con un Sistema de Colonia de Hormigas.
7. Ejecución de pruebas para verificar y validar que el programa funciona correctamente y aplicar instancias pequeñas.
8. Utilización de instancias de la literatura para comparar resultados obtenidos con los de otros autores.
9. Obtención de un programa en lenguaje C que resuelva el VRPSPD a través de un Sistema de Colonia de Hormigas.

En la figura 2.1 se muestran los puntos antes mencionados de acuerdo a calendarización propuesta para el año 2008.

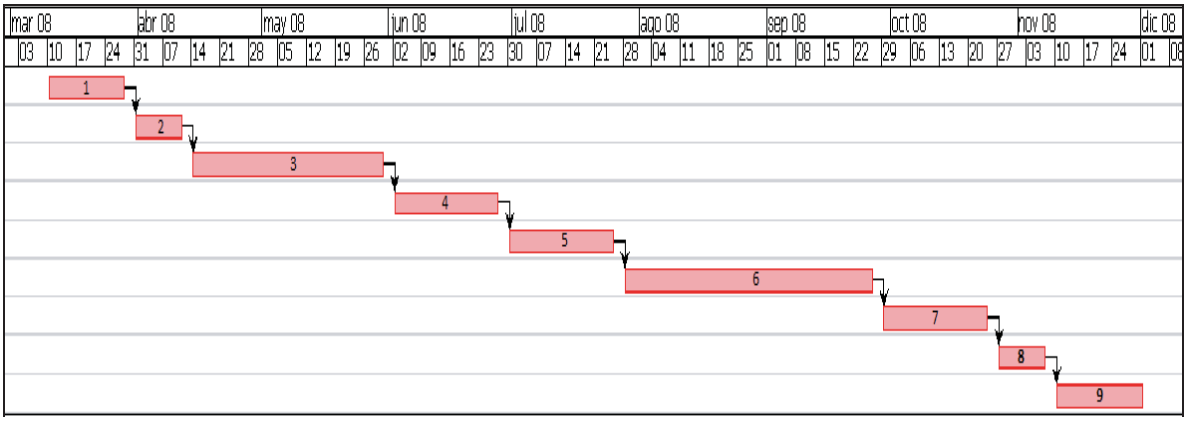

Figura 2.1 Calendarización de desarrollo del proyecto.

3 Estado del Arte

3.1 Vehicle Routing Problem

Hoy en día muchas empresas dedicadas al transporte necesitan trasladar bienes de un lugar a otro con el menor gasto de recursos posible; esto no es fácil de lograr lo cual se transforma en un problema, y la mejor forma de entenderlo es expresarlo como un VRP que es un conocido problema de optimización combinatoria el cual define de forma clara y precisa las rutas a seguir por una flota vehículos para satisfacer las necesidades de un grupo de clientes.

3.1.1 Definición

El problema trata de una flota de vehículos que parten de un depósito y deben viajar por rutas distintas para lograr entre ellos visitar a todos los clientes para lo cual a cada vehículo se le asigna un grupo de éstos. Los clientes deben ser visitados solo una vez y luego de esto los vehículos vuelven al depósito. Los vehículos tienen una cierta capacidad, generalmente la misma para todos, la cual no debe ser sobrepasada por la suma de cargas a entregar a los clientes asignados a cada vehículo. La idea principal es que se minimice ya sea la distancia total recorrida por los vehículos o el tiempo total utilizado por éstos. Gráficamente se representa en la figura 3.1, en este ejemplo existen diez clientes y una flota de tres vehículos.

El VRP nace del clásico Problema del Vendedor Viajero (TSP por su nombre en inglés Traveling Sales Problem). El TSP se refiere a que dado un conjunto de ciudades y cada trayecto entre un par de éstas tiene un costo, la idea es encontrar la forma más barata de visitar todas las ciudades exactamente una vez, y volver al punto de partida, (Espinoza, 2006). Si tomamos estas ciudades como clientes, se puede ver que el TSP es un VRP que solo tiene un vehículo y capacidad ilimitada para visitar a todos los clientes.

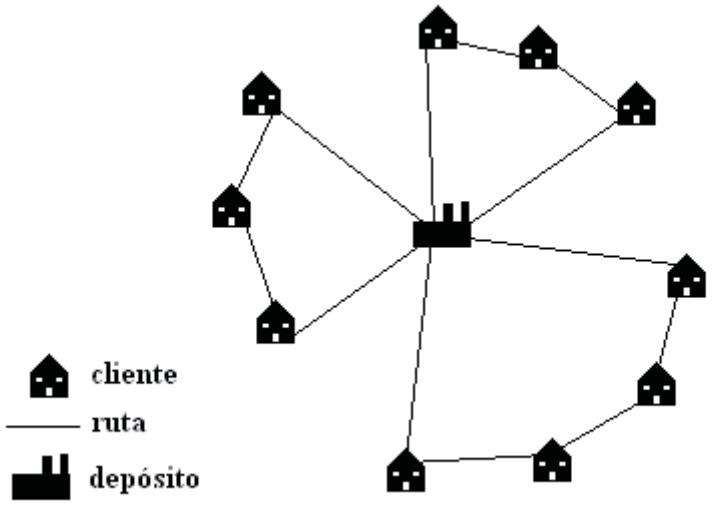

Figura 3.1 Ejemplo de VRP.

3.1.2 Complejidad

Según Díaz (2006), teoría de complejidad: "es parte de la teoría de la computación que estudia los recursos de cómputo requeridos durante el cálculo para resolver un problema". Los recursos más estudiados son:

- El tiempo, es decir, el número de pasos de ejecución de un algoritmo para resolver un problema.
- El espacio, es decir, la cantidad de memoria utilizada para resolver un problema.

Existen muchas clases de complejidad, pero acá se muestran algunas para comprender a cual pertenece el VRP:

- **Clase P:** Conjunto de todos los problemas de decisión que pueden ser resueltos por un algoritmo determinista de tiempo polinomial.
- **Clase NP:** Conjunto de todos los problemas de decisión que pueden ser acotados por algoritmos no determinísticos en tiempo polinomial.

- **Clase NP-Completo:** Conjunto de problemas NP más difíciles de resolver. Estos problemas tienen la propiedad que si uno de ellos puede resolverse en tiempo polinomial, entonces todos los problemas NP pueden hacerlo.
- **Clase NP-Duro:** Problemas que se encuentran dentro del conjunto de problemas NP-Completo, pero que son los más difíciles de resolver.

El VRP se encuentra dentro de la clase de problemas NP-Completo lo que quiere decir que los recursos necesarios para resolver el problema crecen en forma exponencial a medida que crece el problema. Esta sección ha sido tomada de Díaz (2006).

3.1.3 Formulación matemática

La siguiente es la formulación matemática según Hermosilla y Barán (2004) en donde se usa para un problema con ventanas de tiempo, pero en este caso fue adaptado para solucionar solo el VRP y con la idea de minimizar la distancia total recorrida:

El VRP puede verse como un grafo G(V,A) donde V son los nodos o clientes y A son los arcos o rutas.

$V=\{v_0, v_1, ..., v_n\}$ es el conjunto de nodos con v_0 el depósito y el resto los clientes.

$A=\{(v_i, v_j)/v_i, v_j \in V; i \neq j\}$ es el conjunto de arcos o rutas.

$C=\{c_{ij}\} \in \Re^{(n+1)x(n+1)}$ es una matriz de costos o distancias no negativas entre los nodos v_i y v_j, incluyendo el depósito.

$q=\{q_i\} \in \Re^n$ es el vector de demandas de los clientes y q_i es la demanda requerida por el cliente v_i.

m es el número de vehículos los cuales tienen la misma capacidad, que es Q.

R_i es la ruta del vehículo i.

Se considera a cada arco con el mismo costo para ambas direcciones, entonces el conjunto A se convierte en: $E=\{(v_i, v_j)/v_i, v_j \in V; i < j\}$.

$R_i = \{ v_0^i, v_1^i, ..., v_{k_i}^i, v_{k_i+1}^i \}$, $v_j^i \in V$ y $v_0^i = v_{k_i+1}^i = 0$ (el cero denota el depósito) y k_i es la cantidad de clientes que fueron visitados por el vehículo i.

Se debe cumplir la restricción de capacidad del vehiculo que es: $\sum_{j=1}^{k_i} q_j^i \leq Q$, con i = 1,….,m.

El costo de una ruta R_i es $T(R_i) = \sum_{j=0}^{k_i} c_{v_j^i, v_{j+1}^i}$.

La solución al problema es: $S = \{ R_1, ..., R_m \}$

El costo de la solución es $F(S) = \sum_{i=1}^{m} T(R_i)$.

3.1.4 Variantes

A continuación se presentarán las variantes más conocidas del VRP, las cuales realizan cambios al problema original como pueden ser tener los vehículos la misma capacidad, considerar más de un depósito, etc., que sirven para retratar exactamente un problema particular del mundo real por lo que eventualmente pueden crearse nuevas variantes o restricciones para un problema específico. Estas variantes fueron tomadas de Barajas (2006), las primeras cinco, y de The VRP Web (2008), el resto, excepto VRPSPD obtenida de Dethloff (2001).

- **VRP con capacidad (CVRP):** En realidad más que una variante es una restricción que ya casi es parte del VRP en la que los vehículos deben tener la misma capacidad, la que no se debe sobrepasar.
- **VRP con ventanas de tiempo (VRPTW):** Incluye ventanas de tiempo en cada uno de los clientes, es decir, los clientes tienen un horario mínimo y máximo dentro del cual el vehículo debe atenderlos lo que hace más complejo encontrar la solución al problema.
- **VRP con múltiples depósitos (MDVRP):** En este caso existen varios depósitos en que para cada uno existe una flota de vehículos y luego de visitar a un grupo de clientes deben volver al mismo depósito.

- **VRP periódico (PVRP):** Se tiene un plazo de varios días para cumplir la tarea de visitar a los clientes por la flota de vehículos, pero un cliente en particular solo debe ser atendido en un mismo día. Para el término de cada día los vehículos vuelven al depósito y salen nuevamente al siguiente.
- **VRP con entregas y recogidas (VRPPD):** Los clientes no necesariamente recibirán bienes, puede ser que ellos en cambio entreguen algunos por lo que este problema puede asociarse al traslado de personas que suben en un punto y se bajan en otro.
- **VRP con entrega y recogida simultánea (VRPSPD):** En esta variante, a diferencia de la anterior, los clientes tienen la necesidad de que se les entreguen ciertos productos y al mismo tiempo desea devolver otros para lo que el mismo vehículo debe responder a esta solicitud para que el cliente sólo reciba una visita. Se detallará esta variante en la sección 3.2.
- **VRP dividiendo las entregas (SDVRP):** Cada cliente puede ser atendido por varios vehículos, este caso sirve para clientes que necesitan recibir bienes con un tamaño total mayor que la capacidad de los vehículos por lo que la única forma de resolver sus necesidades es que su demanda sea dividida en otras más pequeñas.
- **VRP con backhauls (VRPB):** Esta variante es parecida al VRP con entregas y recogidas (VRPPD) con la diferencia de que los bienes a entregar deben llegar a los clientes antes de que comiencen la recogidas de productos, es decir, después de realizada la última entrega puede llevarse a cabo la labor de recuperación, lo que hace más fácil su resolución ya que solo debe asignarse a cada vehículo clientes cuyas demandas de entrega y recogida total (suma de cargas de todos los clientes asignados a un vehículo) no supere, cada una, la capacidad del vehículo. En este caso no se debe lidiar con verificar si al paso de cada cliente el vehículo aún tiene capacidad disponible si un cliente entrega productos y al vehículo todavía le quedan por entregar.

3.1.5 Aplicaciones

Como se ha explicado anteriormente el VRP resuelve el problema de empresas que necesitan del transporte de bienes, como pueden ser las dedicadas a este fin como también las que tienen a esta tarea como parte importante de su funcionamiento, como por ejemplo las que necesitan de ser abastecidas de materia prima, para la elaboración de productos, teniendo que traer ésta desde distintos proveedores. A continuación se presentan ejemplos de empresas que necesitan resolver el VRP, (Barajas, 2006):

- Empresas de correo.
- Empresas dedicadas a recolectar basura.
- Empresas de transporte de escolares.
- Empresas de reparto de productos (supermercados, tiendas de retail, etc.)

Existen empresas que sin tener la necesidad del transporte también pueden requerir resolver el VRP, como por ejemplo para determinar que movimientos deben realizar un grupo de brazos robóticos para llegar a todos los puntos y desplazarse lo menos posible, esto porque aunque este movimiento sea de unos cuantos centímetros, el hecho de realizar esta tarea una gran cantidad de veces implica una mantención de esta maquinaría que podría reducirse.

3.2 VRP con logística reversa

3.2.1 Introducción

En los últimos años un crecimiento en la conciencia ambiental ha hecho que se incrementen los esfuerzos por proteger el medioambiente por la industria privada y la legislación. La reducción de la cantidad de desperdicios producidos y la energía consumida son dos de los objetivos principales. Tomando en cuenta que esto ha sido buscado por las empresas privadas ya que en su afán de ahorrar costos buscan reducir el consumo de energía, esto cada vez ha cobrado más importancia ya sea por un sentido de responsabilidad o por efecto de las leyes, Dethloff (2001).

Con respecto a la reducción de desperdicios, cuando los productos cumplen su ciclo de vida, estos pueden ser en parte o totalmente reciclados, remanufacturados o reusados. En algunos países hay regulaciones que consisten en que las empresas deben recuperar estos productos ya sea para las actividades antes mencionadas o para disponer de su correcta eliminación. Todo esto genera que los productos viajen de vuelta desde los usuarios hacia las empresas.

Para el caso del VRP la relación entre las logísticas forward (funcionamiento básico del VRP) y reversa debe ser tomada en cuenta. Según como se tome es la variante del VRP que se usará. Para este problema en particular de VRP con logística reversa los clientes desean recibir una cantidad de bienes y al mismo tiempo tienen otros para devolver a las empresas. En el caso que ambas logísticas sean tomadas en forma independiente entonces se tendrá que resolver un VRP para cada una de ellas.

Al considerar estas logísticas en forma independiente puede resultar en un innecesario derroche de recursos. Esto puede ser evitado al combinar ambas en el trayecto de los mismos vehículos, lo que significa que estos mismos se encarguen tanto de la distribución de productos como de la recuperación de otros. Además los clientes querrán ser visitados solo una vez para satisfacer ambas necesidades para lo cual los vehículos deberán resolver en forma simultánea tanto las entregas (entregar productos a los clientes) como las recogidas (recibir productos de los clientes). Es aquí donde aparece la variante al VRP llamada en inglés Vehicle Routing Problem with Simultaneous Pick-up and Delivery (VRPSPD) que es la que se resolverá en el presente proyecto.

Un claro ejemplo en donde es necesaria la entrega y recogida de productos en forma simultánea es en el caso de vehículos de reparto de bebidas ya que deben entregar estos productos a los locales para su futura venta, pero a la vez deben retirar envases vacíos para ser devueltos a la embotelladora.

Como la legislación obliga a ciertas empresas a preocuparse por sus productos por toda su vida útil, éstas empiezan a querer tomar el control de todo el ciclo de vida de los productos para aumentar la eficiencia en la recuperación. Otra actividad de importancia como ejemplo de VRPSPD es el arrendamiento de bienes, los cuales pueden ser intercambiados para renovación de equipamiento de las empresas que los solicitan.

3.2.2 Formulación matemática

Se presenta la formulación matemática para el VRPSPD.

- **Notación**

  - **Conjuntos:**

    J: Conjunto de todas las localidades de clientes.

    J_0: Conjunto de todos los nodos, esto es, los clientes y el depósito, $J_0 = J \cup \{0\}$.

    V: Conjunto de todos los vehículos.

  - **Parámetros:**

    C: Capacidad del vehículo

    C_{ij}: Distancia entre los nodos $i \in J_0, j \in J_0, i \neq j; C_{ii} := M, i \in J, C_{00} := 0$.

    D_j: Cantidad de entrega del cliente $j \in J$.

    n: Número de nodos, esto es, $n = |J_0|$.

    P_j: Cantidad de recogida del cliente $j \in J$.

    M: Número grande, por ejemplo $M = \text{máx}\{\sum_{j \in J}(D_j + P_j), \sum_{i \in J_0} \sum_{j \in J_0, j \neq i} C_{ij}\}$

  - **Variables de decisión:**

    l'_v: Carga del vehículo $v \in V$ cuando deja el depósito; puede ser eliminado del modelo.

    l_j: Carga del vehículo después de haber servido al cliente $j \in J$.

    π_j: Variable usada para prohibir subtours; puede ser interpretado como la posición del nodo $j \in J$ en la ruta.

    x_{ijv}: Variable binaria que indica si el vehículo $v \in V$ viaja directamente desde el nodo $i \in J_0$ al nodo $j \in J_0 (x_{ijv} = 1)$ o no $(x_{ijv} = 0)$.

- **Modelo**

Minimizar $z = \sum_{i \in J_0} \sum_{j \in J_0} \sum_{v \in V} C_{ij} x_{ijv}$ (1)

(Minimizar la distancia total de viaje)

Sujeto a:

$$\sum_{i \in J_0} \sum_{v \in V} x_{ijv} = 1 \qquad (j \in J) \quad (2)$$

(Servir a los clientes exactamente una vez)

$$\sum_{i \in J_0} x_{isv} = \sum_{j \in J_0} x_{sjv} \qquad (s \in J, v \in V) \quad (3)$$

(Arribar y dejar a cada cliente con el mismo vehículo)

$$l'_v = \sum_{i \in J_0} \sum_{j \in J} D_j x_{ijv} \qquad (v \in V) \quad (4)$$

(Carga inicial del vehículo)

$$l_j \geq l'_v - D_j + P_j - M(1 - x_{0jv}) \qquad (j \in J, v \in V) \quad (5)$$

(Carga del vehículo después del primer cliente)

$$l_j \geq l_i - D_j + P_j - M(1 - \sum_{v \in V} x_{ijv}) \qquad (i \in J, j \in J, j \neq i) \quad (6)$$

(Carga del vehículo 'en ruta')

$$l'_v \leq C \qquad (v \in V) \quad (7)$$

$$l_j \leq C \qquad (j \in J) \quad (8)$$

(Capacidad del vehículo después del primer cliente y 'en ruta')

$$\pi_j \geq \pi_i + 1 - n(1 - \sum_{v \in V} x_{ijv}) \qquad (i \in J, j \in J, j \neq i) \quad (9)$$

(Restricción de violación de subtour)

$$\pi_j \geq 0 \qquad (j \in J) \quad (10)$$

$$x_{ijv} \in \{0,1\} \qquad (i \in J_0, j \in J_0, v \in V) \quad (11)$$

3.2.3 Relación entre VRPSPD y otros VRP

Una de las variantes del VRP parecidas a VRPSPD es el Vehicle Routing Problem with Backhauls (VRPB). En este caso se hacen todas las entregas antes de empezar a realizar las recogidas. Otra variante es el VRP with Mixed Pick-up and Delivery (VRPMPD), también llamado VRP with Backhauls Mixed (VRPBM), que puede realizar las recogidas antes de terminar de hacer las entregas, lo que significa que hay una "carga mixta" de bienes de recogida y entrega en el mismo vehículo, pero donde los clientes solo tienen una petición de servicios, siendo separados en los que "sólo requieren entregas" y los que "sólo requieren recogidas". El la figura 3.2 (Zachariadis *et al.*, 2007) se puede ver más claramente las diferencias entre las dos variantes del VRP antes mencionadas y el VRPSPD.

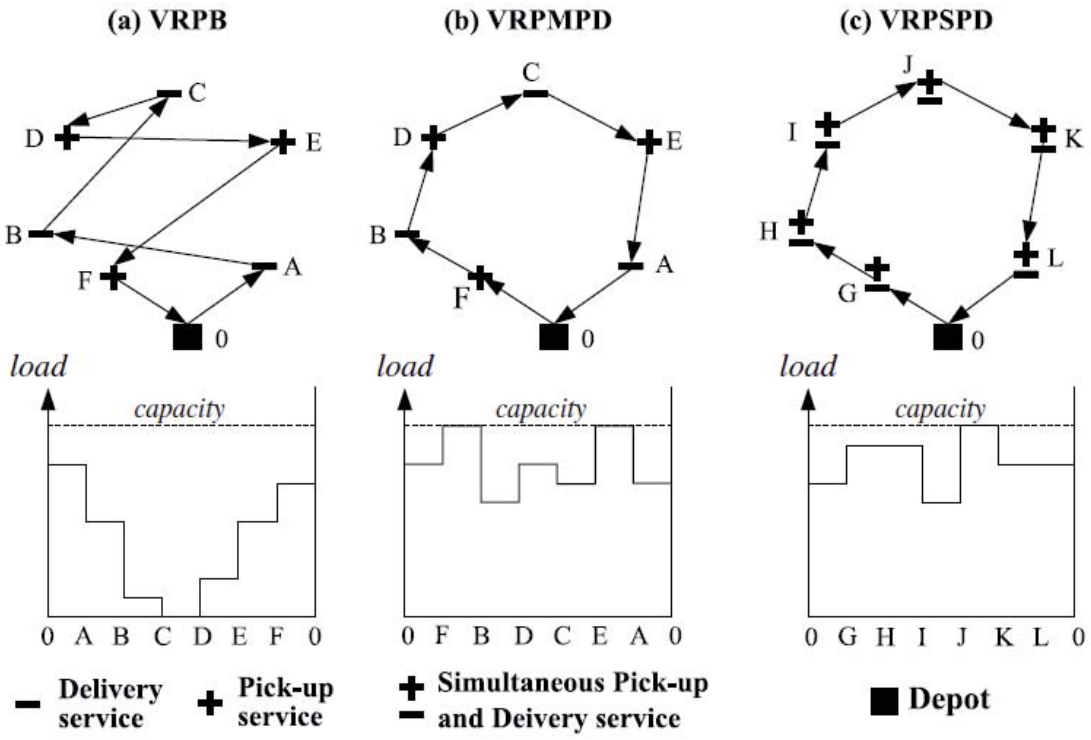

Figura 3.2 Comparación gráfica de distintas variantes del VRP, (Zachariadis et al., 2007).

La variante VRPB, comparada con VRPMPD, es más fácil de resolver ya que solo se debe considerar que las cargas totales de los clientes seleccionados para cada vehículo no sobrepasen la capacidad de éste, ya sea la cantidad total de entregas a repartir a los clientes

como la cantidad total de recogidas que se deben recuperar de los demás asignados. En este caso no se debe considerar la capacidad del vehículo después de atender a cada cliente como tampoco la ruta a seguir para que el vehículo no vaya a quedar sin capacidad en cierto momento, ya que en esta variante "se entrega todo" antes de "recoger todo".

Distinto es el caso de VRPMPD ya que aquí sí se debe ver cómo va la capacidad del vehículo después de pasar por cada cliente ya que las cargas se mezclan entre las que se entregan y las que se recogen. Esto quiere decir que en este caso la capacidad del vehículo depende no únicamente de los clientes a visitar sino también de la ruta a seguir. Por lo tanto es más difícil de resolver. Esto también se presenta en el VRPSPD el cual presenta la característica, que lo distingue del VRPMPD, que cada cliente tiene tanto recogidas como entregas. Otra variante es el VRPPD en donde existen peticiones de recogida y entrega, y que se usa por lo general para el transporte donde a menudo tanto el origen como el destino de ese transporte no es el depósito.

El VRPPD es una generalización del VRPSPD, ya que en este último, ya sea el origen o el destino de una carga es el depósito y las peticiones de cada cliente son de a dos, una del depósito hacia el cliente y la otra del cliente hacia el depósito. A su vez el VRPSPD es una generalización del VRPMPD donde ya sea la entrega o la recogida, una de las dos, de cada cliente es cero. Toda la información de la sección 3.2 ha sido basada en Dethloff (2001).

3.3 Optimización basada en Colonia de Hormigas

La metaheurística Optimización basada en Colonia de Hormigas fue la usada para resolver el VRP en el presente proyecto, pero para tener una visión general de otras técnicas de resolución del VRP, se explican las más importantes en el anexo 1 de este informe.

3.3.1 Definición

Esta metaheurística fue creada en los años noventa, por lo que tiene pocos años desde su creación. También puede ser llamada ACO por su nombre en inglés Ant Colony Optimization. Se inspira en el comportamiento natural de las hormigas para encontrar los caminos más cortos entre su hormiguero y la comida. Desde el trabajo de Dorigo, Maniezzo

y Colorni en el Sistema de Hormigas, esta metaheurística se ha ido convirtiendo en un importante campo de investigación y además se han desarrollado modelos cada vez más complejos en la resolución de problemas de optimización combinatoria. (Alonso *et al.*, 2004).

Las hormigas pueden encontrar los caminos más cortos hacia el alimento aunque carezcan casi por completo de la visión, ya que ellas se mueven en forma aleatoria y van dejando a su paso una sustancia llamada feromona que es percibida por las demás hormigas, de esta manera las que encuentran primero el alimento volverán inmediatamente al hormiguero dejando a su paso feromona que influirá en la decisión que tomen las otras, (Barajas, 2006). Esto es debido a que las hormigas se mueven en forma probabilística siendo influenciadas por los caminos donde hay más feromona, (Alonso *et al.*, 2004). Esta búsqueda continúa con hormigas que han tomado estos caminos ya que cada vez se hacen los más probables de ser escogidos y termina cuando todas han escogido solo uno entre el hormiguero y la comida.

Para explicar cómo las hormigas encuentran los caminos más cortos se puede ver la figura 3.3 de Dorigo y Gambardella (1997) donde en la figura A se encuentran caminando hormigas por un lado y por otro de dos alternativas de camino a seguir. Las primeras tomarán un camino en forma aleatoria, por lo que al considerar el promedio de estas decisiones una hormiga toma un camino y la siguiente toma el otro, como se puede ver en la figura B. Luego aparecen en la figura C donde las hormigas que tomaron el camino más corto llegan primero al otro extremo lo que hace que ese camino quede con una cantidad mayor de feromona, pero acá nuevamente cada una de las dos siguientes hormigas toma un camino. En la figura D ya se refleja claramente que aunque las hormigas hayan tomado aleatoriamente los caminos, la cantidad de feromona en el camino de abajo duplica la cantidad que hay en el de arriba lo que hará que el más corto sea más probable de ser utilizado por las siguientes hormigas hasta que finalmente todas irán por éste.

Cabe señalar también que se produce una evaporación de la feromona lo cual hace que los caminos que no se toman vayan perdiendo importancia en la elección de las hormigas. Para (Alonso *et al.*, 2004) estudios biológicos señalan que los rastros de feromona son muy persistentes lo que hace que las hormigas no “olviden” un camino con gran cantidad de feromona aunque hayan encontrado uno más corto y que se debe

considerar que si se traslada este comportamiento a un computador para diseñar un problema de búsqueda, los resultados podrían quedar atrapados en un óptimo local.

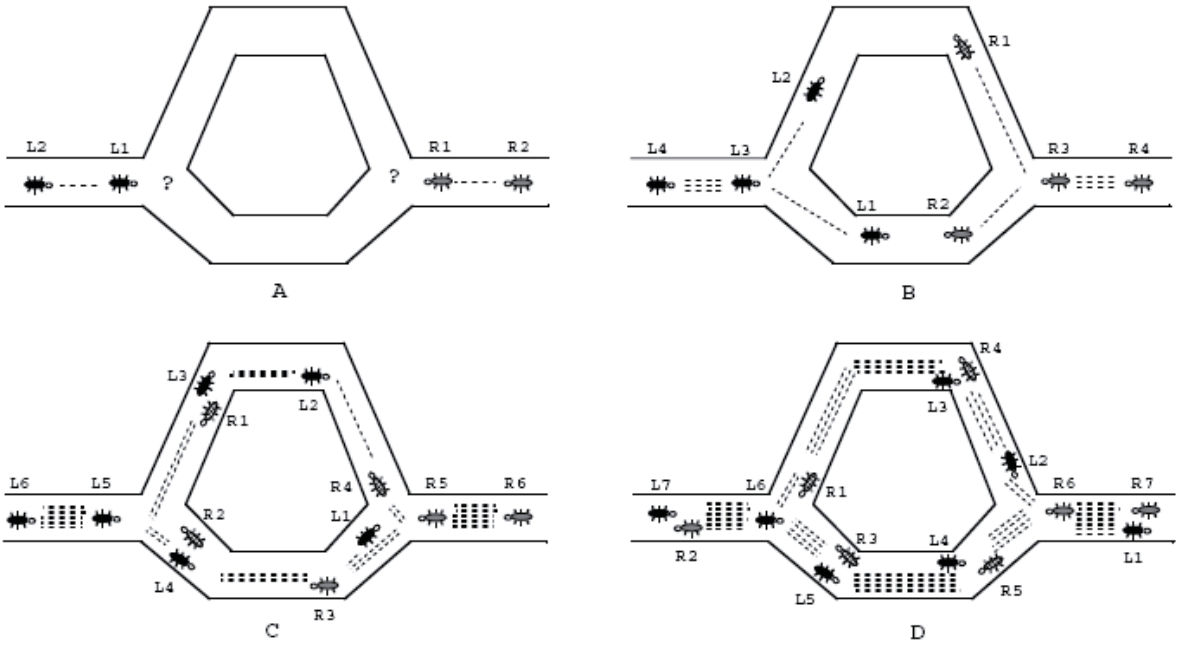

Figura 3.3 Ejemplo de cómo las hormigas terminan usando el camino más corto, (Dorigo y Gambardella, 1997).

3.3.2 Similitudes y diferencias entre las hormigas reales y las artificiales

Gran parte de las ideas de ACO vienen de las hormigas reales. Específicamente en el uso de (Dorigo *et al.*, 1999):

- **Una colonia de individuos cooperando** compuesta por las hormigas que son entidades concurrentes y asíncronas que cooperan globalmente para encontrar una solución al problema tratado. Además estos agentes pueden trabajar en paralelo lo que puede permitir desarrollar una solución distribuida.
- **Un rastro de feromona (artificial) para una comunicación local estimérgica** que significa que las hormigas reales cambian el estado de su ambiente que visitan usando esta sustancia para comunicarse entre ellas al igual que las artificiales que cambian datos numéricos en cada paso para influir en la decisión de ruta de las demás hormigas.

- **Una secuencia de movimientos locales para encontrar los caminos más cortos**, es decir, las hormigas reales y artificiales comparten la misma meta que es encontrar los caminos más cortos (mínimo costo) para resolver su problema, y esto lo hacen paso a paso.
- **Una política de decisión estocástica usando información local y no predictiva** que se refiere a que tanto las hormigas reales como las artificiales solo deciden en forma probabilística la ruta a seguir tomando como información solo la cantidad de feromona que se encuentra en las alternativas a escoger.

Las hormigas artificiales tienen características que no tienen las reales según Dorigo *et al.* (1999), de donde se obtienen las siguientes, que dicen que las hormigas artificiales:

- Viven en un mundo discreto.
- Tienen un estado interno que guarda el camino seguido por la hormiga.
- Depositan una cantidad de feromona que es en base a la calidad de la solución obtenida.
- Pueden ser equipadas con capacidades extra como pueden ser búsqueda local.

3.3.3 Sistemas de Hormigas

El Sistema de Hormigas fue el comienzo del estudio sobre las hormigas para resolver problemas de optimización y fue propuesto por Marco Dorigo en 1992. El funcionamiento de este método es el siguiente (Dorigo y Gambardella, 1997): cada arco (r,s) tiene una medida de deseabilidad $\tau(r,s)$ llamada feromona. Si se toma este método para resolver el TSP, y se ve como un grafo de ciudades y rutas, entonces cada hormiga genera una solución completa o ruta, y la manera de escoger las ciudades es de acuerdo a una probabilística **regla de transición de estado** en que ellas prefieren los caminos que sean más cortos, es decir, de menor costo, y que tengan una mayor cantidad de feromona. Luego que las hormigas completan sus soluciones o rutas una **regla de actualización global de feromona** es ejecutada en que una parte de la feromona que hay en cada arco es evaporada y luego cada hormiga deposita una cantidad de feromona en los arcos que pertenezcan a su

solución y mientras más pequeña sea la solución, más grande es la cantidad de feromona depositada. Este proceso se itera.

La regla de transición de estado, llamada **regla proporcional-aleatoria**, se muestra en la ecuación (12) que entrega la probabilidad de que la hormiga k en la ciudad r escoja ir a la ciudad s.

$$p_k(r,s) = \begin{cases} \dfrac{[\tau(r,s)] \cdot [\eta(r,s)]^{\beta}}{\sum_{u \in J_k(r)} [\tau(r,u)] \cdot [\eta(r,u)]^{\beta}} & \text{si } s \in J_k(r) \\ 0 & \text{en otro caso} \end{cases} \qquad (12)$$

La letra τ representa la feromona, $\eta = \frac{1}{\delta}$ el es inverso a la distancia $\delta(r,s)$ que es el costo del arco (r,s), $J_k(r)$ es el conjunto de ciudades que restan y pueden ser visitadas por la hormiga k desde su ubicación en r, siempre que la solución sea factible, y β es un parámetro que da la importancia relativa entre la feromona y la distancia ($\beta>0$).

Como se puede ver en la ecuación (12) la feromona se multiplica por el inverso a la distancia lo que significa que se busca elegir arcos cortos y con gran cantidad de feromona.

En el caso de la regla de actualización global, luego que todas las hormigas completaron sus soluciones, la feromona es actualizada en todos los arcos de acuerdo a la ecuación (13).

$$\tau(r,s) \leftarrow (1-\alpha) \cdot \tau(r,s) + \sum_{k=1}^{m} \Delta\tau_k(r,s) \qquad (13)$$

donde $$\Delta\tau_k(r,s) = \begin{cases} 1/L_k & \text{si } (r,s) \in \text{ a la ruta hecha por la hormiga k} \\ 0 & \text{en otro caso} \end{cases}$$

El parámetro α indica el decremento de feromona y su valor está dentro del intervalo [0,1], L_k es el valor del largo total de la solución de la hormiga k y m es el número de hormigas.

La actualización de feromona busca dejar con la mayor cantidad de feromona las rutas más cortas, y además se encarga del cambio en la cantidad de feromona que se produce entre la evaporación y la cantidad que dejan las hormigas.

3.3.4 Sistema de Colonia de Hormigas

El método Sistema de Colonia de Hormigas que desde ahora llamaremos ACS presenta tres grandes cambios con respecto al Sistema de Hormigas los cuales son: modificación de la regla de transición de estado; la regla de actualización global se aplica solo a la ruta de la hormiga con la mejor solución; y aparece una regla de actualización local que modifica el rastro de feromona en los arcos a cada paso de las hormigas. A continuación se detallarán cada uno de estos cambios, tomados de Dorigo y Gambardella (1997) y además se puede ver en el apéndice 1 el algoritmo propuesto por ambos autores.

- **Regla de transición de estado en ACS**

La regla de transición de estado entrega un balance entre exploración de nuevos espacios de búsqueda y explotación del conocimiento que se tenga del problema. Una hormiga en r escoge ir a una ciudad s usando la regla que aparece en la ecuación (14).

$$s = \begin{cases} \arg\max_{u \in J_k(r)} \left\{ [\tau(r,u)] \cdot [\eta(r,u)]^{\beta} \right\} & \text{si } q \leq q_0 \text{ (explotación)} \\ S & \text{en otro caso (exploración parcial)} \end{cases} \qquad (14)$$

En este caso q es un número aleatorio uniformemente distribuido en [0..1], q_0 es un parámetro dado dentro del intervalo [0,1], y S significa que si $q > q_0$ se aplica la ecuación (12).

La regla de transición de estado ahora pasa a ser **regla proporcional-pseudo-aleatoria** ya que ahora q entrega un número en forma aleatoria que si es menor o igual a q_0 se favorece la explotación ya que se escoge el mejor camino en la relación entre feromona y

el costo de la ruta, en cambio si es mayor a q_0 se favorece la exploración ya que aplica probabilidad a través de la ecuación (12).

- **Regla de actualización global en ACS**

Esta actualización global de feromona se realiza solo en la ruta de la hormiga que pertenece a la mejor solución global. Esta actualización se realiza según la ecuación (15).

$$\tau(r,s) \leftarrow (1-\alpha) \cdot \tau(r,s)+\alpha \cdot \Delta\tau(r,s) \qquad (15)$$

donde $$\Delta\tau(r,s) = \begin{cases} \left(L_{gb}\right)^{-1} & \text{si (r,s)} \in \text{ mejor-ruta-global} \\ 0 & \text{en otro caso} \end{cases}$$

En esta ecuación, α nuevamente es el parámetro de decremento de feromona y L_{gb} es el valor del costo total de la mejor ruta de las hormigas, pero desde el comienzo de la ejecución no en una iteración solamente. Al igual que en el Sistema de Hormigas en este caso también la idea es entregar más feromona a los caminos más cortos. En Dorigo y Gambardella (1997) también se probó con una regla de actualización global llamada **mejor-iteración** que en vez de actualizar con feromona solo a la mejor ruta que se tenga hasta el momento luego de cada iteración como es el caso de **mejor-global**, en esta nueva regla se actualiza la mejor ruta que se entregue en cada iteración. Experimentos dieron una leve ventaja en los resultados a mejor-global.

- **Regla de actualización local en ACS**

A cada paso las hormigas actualizan la cantidad de feromona en la ruta a través de la ecuación (16).

$$\tau(r,s) \leftarrow (1-\rho) \cdot \tau(r,s)+\rho \cdot \Delta\tau(r,s) \qquad (16)$$

La letra ρ también es un parámetro de decremento de feromona y su valor está dentro del intervalo [0,1]. Se probaron tres valores para $\Delta\tau(r,s)$ en el TSP siendo $\Delta\tau(r,s) = \tau_0$ (τ_0 es un valor inicial de feromona) el escogido por ser uno de los de mejor resultados y por requerir menos computación, Dorigo y Gambardella (1997).

4 Solución Propuesta

4.1 Implementación del VRP con Sistema de Colonia de Hormigas

Como se ha dicho, primero se ha implementado el VRP en su versión clásica antes de resolver la variante VRPSPD, a partir de modificaciones a la implementación detallada en esta sección. La resolución del VRPSPD se puede ver en este capítulo, en el punto 4.3, donde además se pueden ver los resultados para una instancia pequeña; los resultados para instancias de la literatura se pueden ver en la sección 4.4.

El VRP, aquí resuelto, es en realidad el CVRP ya que se maneja la capacidad de los vehículos, igual para todos, que no se debe sobrepasar por las peticiones de los clientes. Para su implementación se tomó como idea básica los algoritmos 1 y 2 de Zabala y Torres (2005) y se tomaron las ecuaciones del Sistema de Colonia de Hormigas, y varios de los valores de los parámetros, de Dorigo y Gambardella (1997).

4.1.1 Funcionamiento básico

En la resolución del VRP se tomó a cada hormiga como una solución completa del problema, es decir, el camino recorrido por la hormiga es el conjunto de todos los caminos hechos por los vehículos, como se muestra en el ejemplo de la figura 4.1 donde una hormiga toma el papel de un vehículo volviendo al depósito y saliendo nuevamente de él para representar a un nuevo vehículo hasta visitar todos los clientes. Este ejemplo como ya se dijo en la sección 3.1.1 es de una flota de 3 vehículos que atienden a 10 clientes.

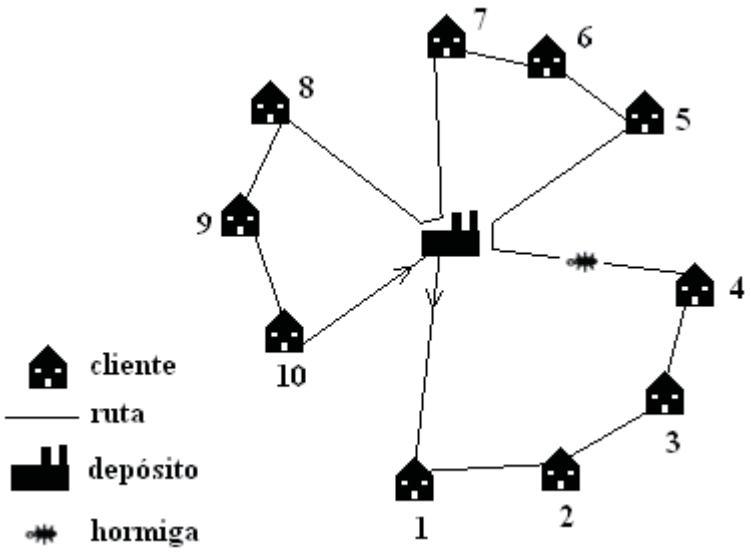

Figura 4.1 Ejemplo de solución del VRP por una hormiga.

Ahora se explica brevemente el funcionamiento básico del VRP con un Sistema de Colonia de Hormigas para en la siguiente sección profundizar en la implementación que se hizo de éste. Se tienen k hormigas en un depósito central, una de ellas se mueve hacia un nodo que no haya sido visitado y que cumpla con la restricción de capacidad del vehículo que representa en ese momento. Este movimiento se realiza aplicando las ecuaciones (12) y (14) de exploración y explotación mostradas en las secciones 3.3.3 y 3.3.4 respectivamente. En este movimiento se incluye al depósito como posible nodo a ser visitado, exceptuando cuando la hormiga está ubicada en él. A cada paso de un nodo a otro la hormiga realiza una actualización local de feromona a través de la ecuación (16) mostrada en la sección 3.3.4. Si la hormiga no tiene más posibilidades de clientes donde ir entonces vuelve al depósito lo que no quiere decir que no pueda hacerlo antes como ya se dijo. Cuando vuelve al depósito la hormiga representa a un nuevo vehículo por lo que su capacidad vuelve a cero. Al finalizar el recorrido de todos los vehículos la hormiga vuelve al depósito, se realiza una actualización local de feromona solo en los tramos recorridos por la hormiga, y una nueva sale del depósito para realizar la misma tarea. Este mecanismo se podría hacer en paralelo en un sistema distribuido, es decir, que todas las hormigas salgan del depósito al mismo tiempo buscando soluciones en vez de hacerlo en forma secuencial como acá se ha resuelto. Al término del recorrido de todas las hormigas se realiza una actualización global de feromona donde solo los tramos entre dos nodos que pertenezcan a la mejor solución global

son inyectados con feromona luego de realizar una evaporación de ésta en ellos, para los demás tramos o arcos solo se realiza una evaporación de feromona. Las soluciones obtenidas por las hormigas se comparan con la mejor solución global que se tenga desde el inicio de la ejecución del sistema, la cual en un comienzo es obtenida por la heurística del vecino más cercano. Para la hormiga que encontró una solución con la menor distancia total recorrida, si esta distancia es menor que la de la solución global, entonces el recorrido de esta hormiga pasa a ser la solución global. Todo este proceso se realiza un cierto número de veces para ir mejorando cada vez más la solución global.

4.1.2 Forma de implementación y estructuras de datos

La implementación del sistema se realizó en lenguaje C y se explicará a continuación su funcionamiento en detalle siempre en referencia a la figura 4.2 que es un diagrama que muestra a grandes rasgos el funcionamiento del sistema y tomando en cuenta la figura 4.3 que muestra las estructuras de datos usadas para resolver el VRP.

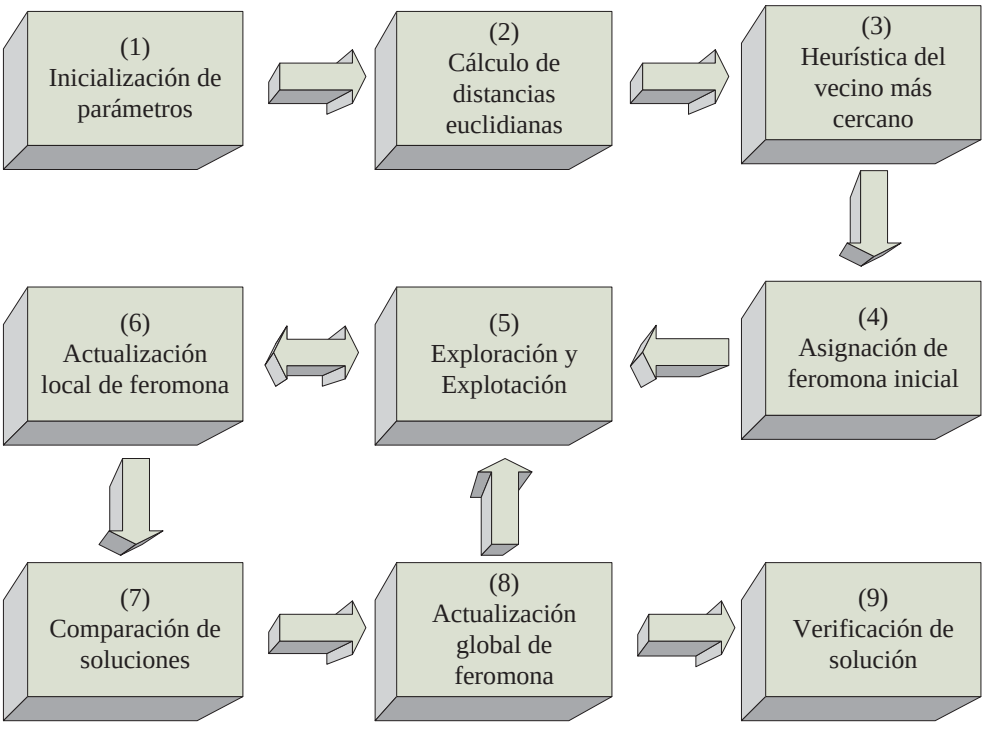

Figura 4.2 Diagrama que representa el funcionamiento general del sistema.

La **inicialización de parámetros** de la figura 4.2 se refiere a la inicialización de: las coordenadas x e y de las ubicaciones tanto del depósito como de los clientes, cantidad de productos a entregar a cada cliente, número de clientes y vehículos y capacidad de éstos últimos. También están los parámetros propios del Sistema de Colonia de Hormigas que son: cantidad de hormigas, importancia relativa entre feromona y distancia, y los decrementos de feromona tanto para actualización global de feromona como para actualización local de ésta.

Para usar instancias de la literatura el sistema tiene previamente ingresados a través del código los datos de las ubicaciones en coordenadas x e y del depósito y los clientes y a partir de estos datos se calculan las **distancias euclidianas** de todos los pares de nodos las cuales se guardan en una matriz cuadrada cuyo tamaño es el número de clientes más 1 ya que se toma en cuenta al depósito y que en la matriz es la primara fila y la primera columna. Esta matriz se encuentra en la figura 4.3. Estas distancias se aproximan un valor entero antes de ser guardadas. Esta matriz es simétrica, lo que quiere decir que las distancias entre los nodos son las mismas no importando la dirección.

Sean los nodos N_1 y N_2 con coordenadas (x,y), siendo: $N_1 = (x_1, y_1)$ y $N_2 = (x_2, y_2)$. La distancia euclidiana entre estos nodos se calcula con la siguiente ecuación (17).

$$z = \sqrt{|x_2 - x_1|^2 + |y_2 - y_1|^2} \tag{17}$$

Se realiza la **heurística del vecino más cercano** para obtener una solución global inicial y el largo de ésta, es decir, la distancia total recorrida por los vehículos de esta solución. Lo que realmente importa es el largo de la solución para aplicarlo a una ecuación que da como resultado cuanta feromona se le asigna inicialmente a cada nodo. Si la solución obtenida no es correcta, a este largo se le asigna un número más grande que el esperado.

Después de la heurística del vecino más cercano a cada arco (i,j) se le asigna una **cantidad inicial de feromona** llamada τ_0 que se calcula con la ecuación (18), obtenida de Dorigo y Gambardella, (1997).

$$\tau_0 = (n \cdot L_{nn})^{-1} \tag{18}$$

La letra *n* representa el número de clientes y L_{nn} es el largo de la solución global inicial. La cantidad de feromona a asignar en los arcos (*i*,*j*) está representada en una matriz simétrica de feromona, que también se muestra en la figura 4.3.

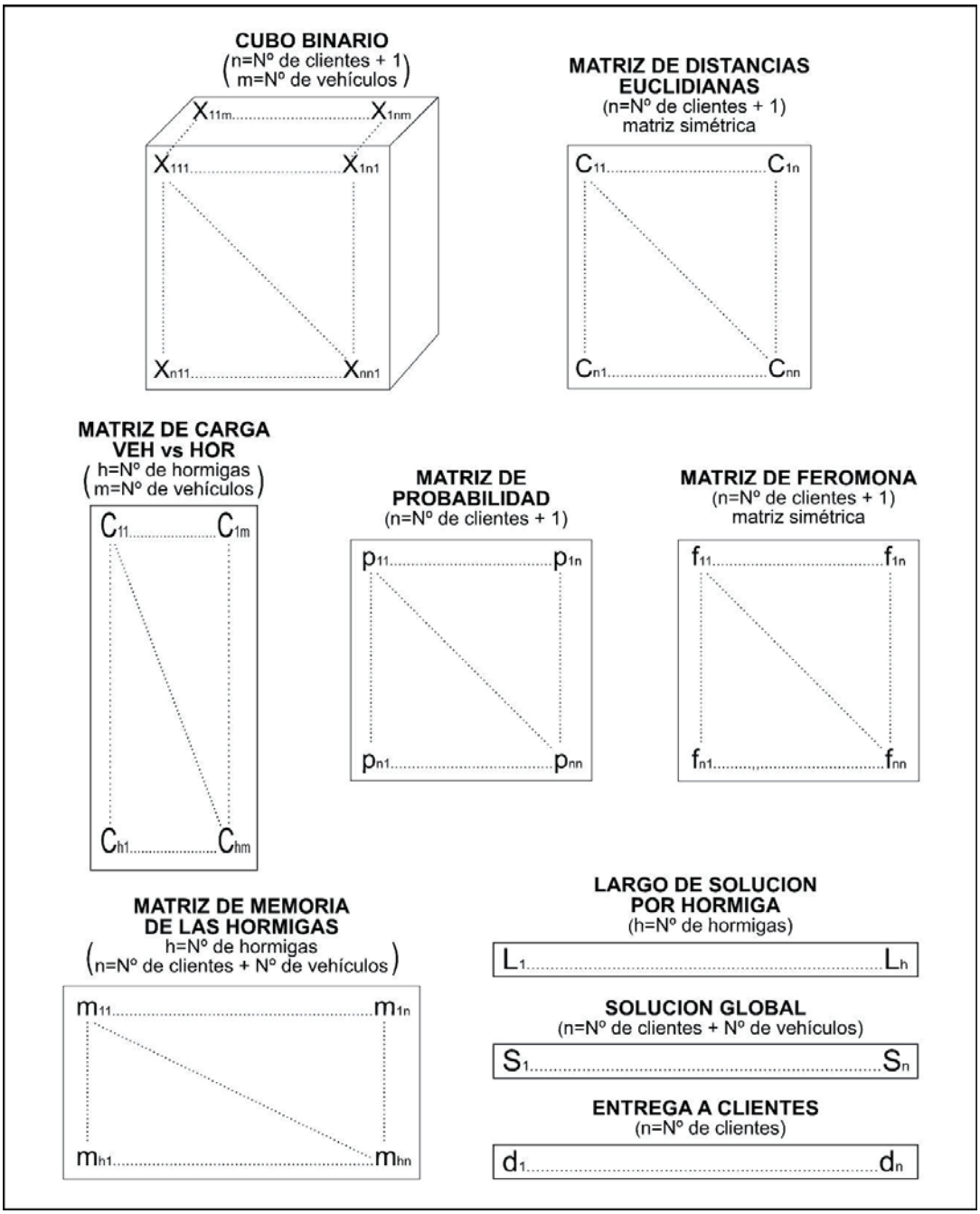

Figura 4.3 Diagrama que representa las estructuras de datos del sistema.

En el centro de la figura 4.2 vemos **exploración y explotación** que significa cual de los dos métodos usamos para elegir el próximo nodo a visitar. El funcionamiento puede verse en el pseudocódigo mostrado en la figura 4.4 que será explicado a continuación en detalle. El pseudocódigo muestra que para cada nodo j del sistema si este no ha sido visitado y la capacidad del vehículo no es sobrepasada con su petición de carga o este nodo es el depósito, entonces a la posición (*i,j*) de una matriz p, mostrada en la figura 4.3 con el nombre de probabilidad, se le asigna $[\tau(i,j)]\cdot[\eta(i,j)]^{\beta}$ y a sumatoria, inicializada previamente en cero, se le asigna lo que lleva esta variable más lo que se le acaba de asignar a p(i,j), siendo i un valor fijo en esta parte del sistema ya que es el nodo donde está la hormiga en el momento de tomar la decisión de donde ir por lo que se cambia solo al momento de tomar esta decisión por el nodo j escogido a visitar. Si un nodo j sobrepasa la capacidad del vehículo entre lo que éste lleva más la petición del nodo, o éste último ya fue visitado entonces al correspondiente *p(i,j)* se le asigna 0.

Si el valor aleatorio entre [0,1] de una variable *q* es menor o igual a q_0 y este último es distinto de 0, entonces se usa el método de explotación y para esto se toma el mayor valor entre los *p(i,j)* y el *j* a que corresponda es el próximo nodo a visitar. El parámetro q_0, con un valor asignado entre [0,1], establece que porcentaje de explotación se va a utilizar dejando el resto a exploración. Los siguientes pasos para explotación se explicarán más adelante ya que en su mayoría son los mismos que para exploración.

Luego de la explotación, que puede que se haya o no realizado dependiendo del valor de q, en la parte del pseudocódigo que dice "asignar a p(i,j) probabilidad según ecuación (12) de exploración" lo que se hace, que sirve si se llega a utilizar la exploración, es lo que se muestra en pseudocódigo de la figura 4.5, y es que a cada p(i,j), con i fijo, que no sea 0, se le asigna un rango entre [0,1], es decir, por ejemplo si se asigna al primer p(i,j) el valor 0,1 significa que tiene 10% de probabilidad de de ser escogido ese nodo para ser visitado desde el i actual, y si al segundo p(i,j) le asignamos 0,5 significa que tiene 40% de probabilidad de ser escogido. Esto se debe a que se toma un valor aleatorio y si está dentro del rango 0 y 0,1 se elige el j del primer p(i,j) para ser visitado y si el valor aleatorio está dentro del rango 0,1 y 0,5 se escoge el j del segundo p(i,j). La asignación de rango se hace para todos los nodos en que p(i,j) sea distinto de 0 y al último p(i,j) que no sea 0 se le

asigna el valor 1 lo que significa que el rango de ser escogido es desde el valor dado al p(i,j) anterior hasta 1.

Hacer

sumatoria ← 0

Para cada (nodo j)

Si (no ha sido visitado y cumple con la capacidad del vehiculo)

$p(i,j) \leftarrow [\tau(i,j)] \cdot [\eta(i,j)]^{\beta}$

sumatoria ← sumatoria + $[\tau(i,j)] \cdot [\eta(i,j)]^{\beta}$

Sino

p(i,j) ← 0

Fin Si

Fin para cada

Tomar un valor aleatorio entre [0,1] para q.

Si ((q $\leq q_0$) y (q_0 <> 0)) /*explotación*/

Se toma el mayor valor de los p(i,j) y el j respectivo es el nodo a visitar.

Se actualiza memoria de la hormiga.

Se suma al largo de la solución de la hormiga la distancia entre i y j.

Fin Si

Para cada (nodo j)

Asignar a p(i,j) probabilidad según ecuación (12) de exploración.

Fin Para cada

Si ((q > q_0) o (q_0 = 0)) /*exploración*/

Se toma un valor aleatorio y se busca a que p(i,j) pertenece.

Se actualiza memoria de la hormiga.

Se suma al largo de la solución de la hormiga la distancia entre i y j.

Fin Si

Mientras ((no se hayan visitado todos los clientes) y (no se hayan utilizado todos los vehículos))

Figura 4.4 Pseudocódigo que representa el uso de exploración y explotación en el sistema.

Siguiendo con el pseudocódigo de la figura 4.4 si q es mayor a q_0 o éste último es igual a cero se realiza la exploración escogiendo el nodo j a ser visitado, como ya se dijo, tomando un valor aleatorio entre [0,1] y se revisa a que rango pertenece.

```
Para cada (nodo j)
    Si ((p(i,j) <> 0) y (es el primer p(i,j)))
        p(i,j) ← p(i,j) / sumatoria
        sumaprobabilidad ← p(i,j)
    Sino
        Si (p(i,j) <> 0)
            p(i,j) ← (p(i,j) / sumatoria) + sumaprobabilidad
            sumaprobabilidad ← p(i,j)
        Fin Si
    Fin Si
Fin Para cada
```

Figura 4.5 Pseudocódigo que asigna probabilidad a cada p(i,j).

Los pasos a seguir dentro de explotación y exploración (que se encuentran en la figura 4.4), luego de haber escogido un nodo a ser visitado según las reglas para cada método, son básicamente actualizar una matriz que representa la memoria de las hormigas dando el valor de j, del primer nodo a visitar, al primer casillero de la matriz y, más adelante, cuando se escoja el valor de j del siguiente nodo a visitar, éste se guarda en el siguiente casillero hacia la derecha y así sucesivamente hasta completar la ruta de la hormiga, que si es correcta sería como la de la figura 4.6 que representa una solución de las rutas de tres vehículos para satisfacer las demandas de siete clientes donde el número 0 representa cuando cada vehículo vuelve al depósito. Las filas hacia abajo de la "matriz de memoria de las hormigas" son para guardar las rutas de las demás hormigas. La matriz es de tamaño número de clientes más número de vehículos hacia la derecha y número de hormigas hacia abajo, como se puede ver en la figura 4.3.

4	0	1	3	5	7	0	2	6	0

Figura 4.6 Solución correcta entregada por una hormiga para 3 vehículos y 7 clientes.

Otro paso a considerar dentro de explotación y explotación de la figura 4.4 es actualizar el largo de la solución de la hormiga para lo cual al largo que se tenga en un cierto momento se suma la distancia de ir del nodo i al j, del p(i,j) elegido. Aunque no se

encuentra en la figura 4.4, también se considera que si el nodo a ser visitado no es el depósito se guarda en un cubo de valores binarios, previamente inicializado en 0 en cada ubicación, un 1 en la ubicación (i,j,v) donde i es el nodo donde está la hormiga, j es el nodo a visitar y v es el vehículo que representa la hormiga. Este cubo luego es el que se usa para los siguientes movimientos para saber si un nodo ya fue visitado o no por la hormiga. Esta idea es tomada de la ecuación (2) de la sección 3.2.2 de la formulación matemática entregada por Dethloff (2001) y este cubo es mostrado en la figura 4.3. Además se utiliza una matriz de carga de los vehículos para las hormigas donde si el nodo escogido no es el depósito a la ubicación del vehículo respectivo de la hormiga que está creando la solución, es decir, a la ubicación (*k,v*), donde *k* es la hormiga y *v* el vehículo, se le suma a la carga que el vehículo lleve la carga que requiere el nodo *j*, carga obtenida de un casillero del vector "entrega a clientes". Por último si el nodo a visitar es el depósito a *v* se le suma 1. La matriz de carga antes mencionada también se encuentra en la figura 4.3.

Todo este proceso de la etapa de exploración y explotación de la figura 4.2 representado en el pseudocódigo de la figura 4.3 se realiza, como dice en la última línea de esta última figura, mientras se cumpla que todavía haya clientes por visitar y que queden vehículos por utilizar. La etapa (5) de explotación y exploración se realiza para cada hormiga de la colonia, que en este caso son 10 hormigas en total, valor tomado de Dorigo y Gambardella (1997).

Según la figura 4.2 después de exploración y explotación hay una **actualización local de feromona** que se hace según la ecuación (16) de la sección 3.3.4., donde se actualiza la matriz de feromonas, ya explicada, solo en los arcos por donde pasó la hormiga, es decir, en los arcos (i,j) que pertenezcan a la solución.

Las etapas (5) y (6) de la figura 4.2 tienen una conexión doble ya que si la solución de la hormiga no es correcta, ya que volvió al depósito con el último vehículo antes de visitar a todos los clientes, todo el proceso ya explicado en conjunto de las etapas (5) y (6) se realiza una vez más, pero con la condición de no volver al depósito, con cada vehículo que esté representando en ese momento, mientras pueda visitar a otro cliente, con lo que hay más posibilidades de obtener una solución correcta.

La siguiente etapa, **comparación de soluciones**, busca si el largo de la distancia total recorrida de la mejor solución entregada por las hormigas es menor que el largo de la

solución global que se tenga hasta ese momento, entonces, si es así, la mejor solución de las hormigas pasa a ser la solución global y en un vector que representa la solución global, llamado de la misma forma en la figura 4.3, se guarda la memoria de la hormiga respectiva, este proceso se puede ver en el pseudocódigo de la figura 4.7 donde además se puede ver que también se guarda el largo de la solución global, es decir, la distancia total recorrida por la hormiga o, dicho de otro modo, la distancia total recorrida por la flota de vehículos.

```
Para cada (hormiga)
    Escoger la hormiga que tenga la menor distancia total recorrida.
Fin Para cada
Si (el largo de la solución de la hormiga elegida es menor que el largo de la solución global)
    largo global ← largo de la hormiga escogida
    solución global ← solución de la hormiga escogida
Fin Si
```

Figura 4.7 Pseudocódigo de la etapa comparación de soluciones.

Después de la etapa comparación de soluciones sigue la de **actualización global de feromona** que usa la ecuación (15) de la sección 3.3.4 donde solo a los arcos (i,j) que pertenezcan a la solución global se les inyecta feromona luego de la evaporación de ésta, al resto de los arcos solo se les realiza la evaporación. Esta actualización se le hace a la matriz de feromonas.

La etapa (8) de la figura 4.2 tiene una conexión hacia la etapa (5) ya que el conjunto de las etapas (5), (6), (7) y (8) forman una iteración, es decir, que solo en una ocasión la colonia de hormigas sale en busca de soluciones y el largo de la mejor de éstas se compara con el de la solución global. Se pueden realizar la cantidad de iteraciones que se estime convenientes para encontrar los mejores resultados. Mientras más iteraciones mejor hasta llegar a un punto en que los resultados no puedan mejorar más o ya son los esperados.

La última etapa de la figura 4.2 es **verificación de resultados** donde se verifica que la hormiga volvió al depósito solo la cantidad veces igual a la cantidad de vehículos, que cada uno de los otros nodos solo fue visitado una vez, que todos los clientes fueron visitados, que cada vehículo no sobrepasó su capacidad y que la distancia total recorrida

entregada como respuesta es correcta con respecto a la ruta dada como solución. Si se cumplen todos estos requerimientos la solución es correcta.

4.2 Experimentos para el VRP con instancias de la literatura y sus respectivos análisis

A continuación se entregarán los resultados aplicados a instancias de Augerat *et. al.* obtenidas de The VRP Web (2008) y se harán algunos análisis de éstos. Primero se tomaron dos instancias una del grupo A, que significa que tanto las ubicaciones de los clientes como sus demandas son valores aleatorias, y una del grupo B que significa que los valores de estas instancias están agrupados (The VRP Web, 2008). La instancia del grupo A se llama A-n32-k5 que según el nombre: 32 significa que el número de clientes es 31 y a éstos se le suma el depósito, y 5 es el número de vehículos. Un dato que no aparece en el nombre es la capacidad de los vehículos que es 100. La instancia del grupo B se llama B-n31-k5 que tiene 30 clientes, un depósito, 5 vehículos y la capacidad de éstos es 100. Ambas instancias fueron resueltas para diferentes valores de exploración vs explotación llamado q_0 partiendo de 0 que significa solo exploración, variando en intervalos de 0,1 hasta llegar a 1 que es solo explotación. Además se resolvieron para 100, 300 y 500 iteraciones. Por último para cada valor q_0 de exploración vs explotación con cada grupo de iteraciones se obtuvieron 10 resultados entregados por el sistema, mostrando el promedio y el menor valor encontrado para cada grupo de resultados como se muestra en las tablas 4.1 y 4.2. Luego se graficó tanto el promedio de los 10 resultados como el mejor encontrado de éstos para cada instancia como se muestra en las figuras 4.8 y 4.9. Para la instancia A-n32-k5 el valor óptimo es 784 y para B-n31-k5 es 672.

Tabla 4.1 Resultados de 10 pruebas para la instancia A-n32-k5 (valor óptimo = **784**).

100 iteraciones				
q_0	**Menor valor**	**% de variación**	**Promedio**	**% de variación**
0	802	2,3%	848	4,34%
0,1	797	1,66%	815	3,95%
0,2	803	2,42%	817,4	4,26%
0,3	792	1,02%	815,6	4,03%
0,4	790	0,76%	806	2,81%
0,5	795	1,4%	817,2	4,23%
0,6	799	1,91%	815,7	4,04%
0,7	797	1,66%	814	3,83%
0,8	801	2,17%	832,9	6,24%
0,9	825	5,23%	875,4	11,66%
1	946	20,66%	946	20,66%
300 iteraciones				
q_0	**Menor valor**	**% de variación**	**Promedio**	**% de variación**
0	787	0,38%	801,9	2,28%
0,1	785	0,13%	795,3	1,44%
0,2	789	0,64%	801,2	2,19%
0,3	790	0,77%	794,8	1,38%
0,4	786	0,26%	794,2	1,3%
0,5	790	0,77%	798,6	1,86%
0,6	795	1,4%	804	2,55%
0,7	796	1,53%	804,5	2,61%
0,8	791	0,89%	808,1	3,07%
0,9	801	2,17%	821	4,.72%
1	946	20,66%	946	20,66%
500 iteraciones				
q_0	**Menor valor**	**% de variación**	**Promedio**	**% de variación**
0	784	0%	790,8	0,87%
0,1	786	0,26%	791,1	0,91%
0,2	789	0,64%	791,8	0,99%
0,3	784	0%	796,1	1,54%
0,4	786	0,26%	792,8	1,12%
0,5	788	0,51%	794	1,28%
0,6	790	0,77%	796,8	1,63%
0,7	790	0,77%	805,1	2,69%
0,8	791	0,89%	810,8	3,42%
0,9	813	3,7%	856,3	9,22%
1	946	20,66%	946	20,66%

Tabla 4.2 Resultados de 10 pruebas para la instancia B-n31-k5 (valor óptimo = **672**).

100 iteraciones				
q_0	**Menor valor**	**% de variación**	**Promedio**	**% de variación**
0	672	0%	675,4	0,51%
0,1	672	0%	674,5	0,37%
0,2	672	0%	675,1	0,46%
0,3	672	0%	676,2	0,63%
0,4	672	0%	673,9	0,28%
0,5	672	0%	673,9	0,28%
0,6	672	0%	676,2	0,63%
0,7	672	0%	677,3	0,79%
0,8	672	0%	681	1,34%
0,9	672	0%	679,9	1,18%
1	715	6,4%	715	6,4%
300 iteraciones				
q_0	**Menor valor**	**% de variación**	**Promedio**	**% de variación**
0	672	0%	672,9	0,13%
0,1	672	0%	673,5	0,22%
0,2	672	0%	673,9	0,28%
0,3	672	0%	672,1	0,01%
0,4	672	0%	672,9	0,13%
0,5	672	0%	672,6	0,09%
0,6	672	0%	673	0,15%
0,7	672	0%	673,6	0,24%
0,8	672	0%	675,2	0,48%
0,9	674	0,3%	681,3	1,38%
1	715	6,4%	715	6,4%
500 iteraciones				
q_0	**Menor valor**	**% de variación**	**Promedio**	**% de variación**
0	672	0%	674	0,3%
0,1	672	0%	672,7	0,1%
0,2	672	0%	672,7	0,1%
0,3	672	0%	672,2	0,03%
0,4	672	0%	672,5	0,07%
0,5	672	0%	672,8	0,12%
0,6	672	0%	675,1	0,46%
0,7	672	0%	674,3	0,34%
0,8	672	0%	677,4	0,8%
0,9	672	0%	680,8	1,31%
1	715	6,4%	715	6,4%

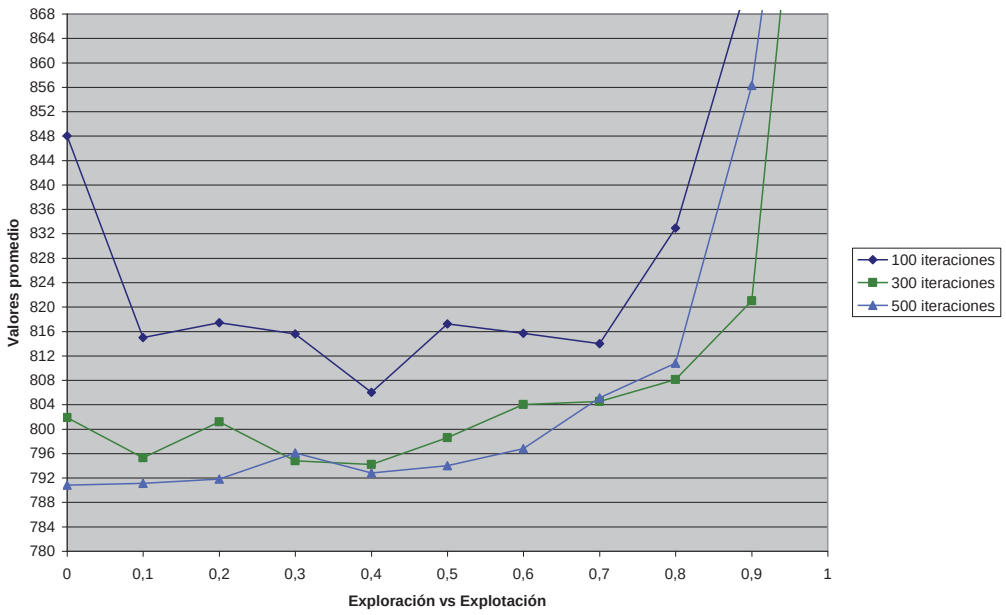

Gráfico (a)

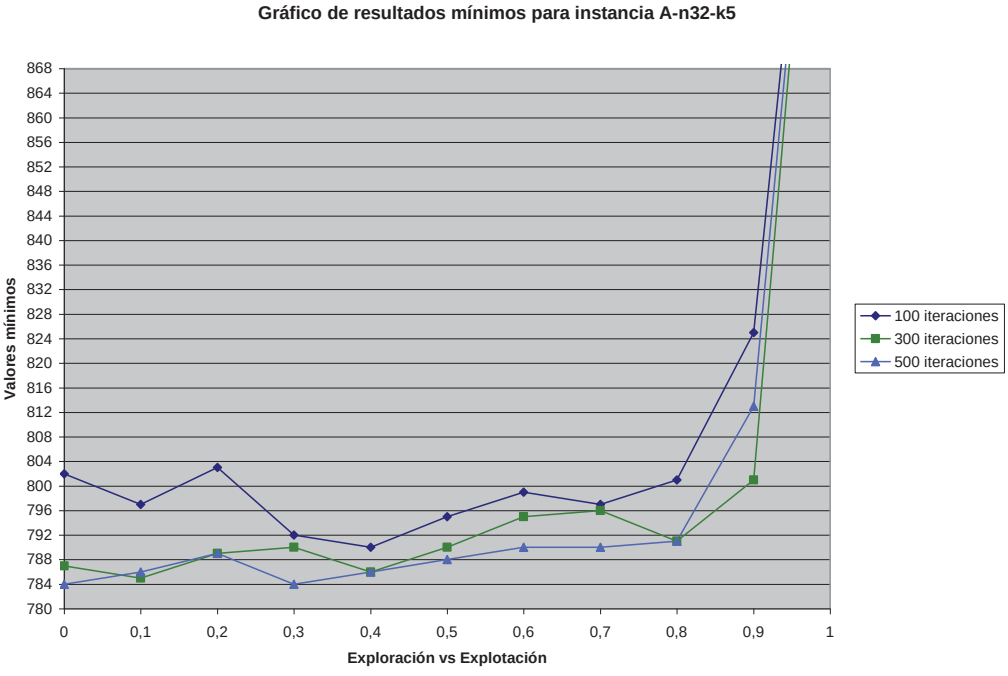

Gráfico (b)

Figura 4.8 Gráficos de valores promedio y mínimos para la instancia A-n32-k5.

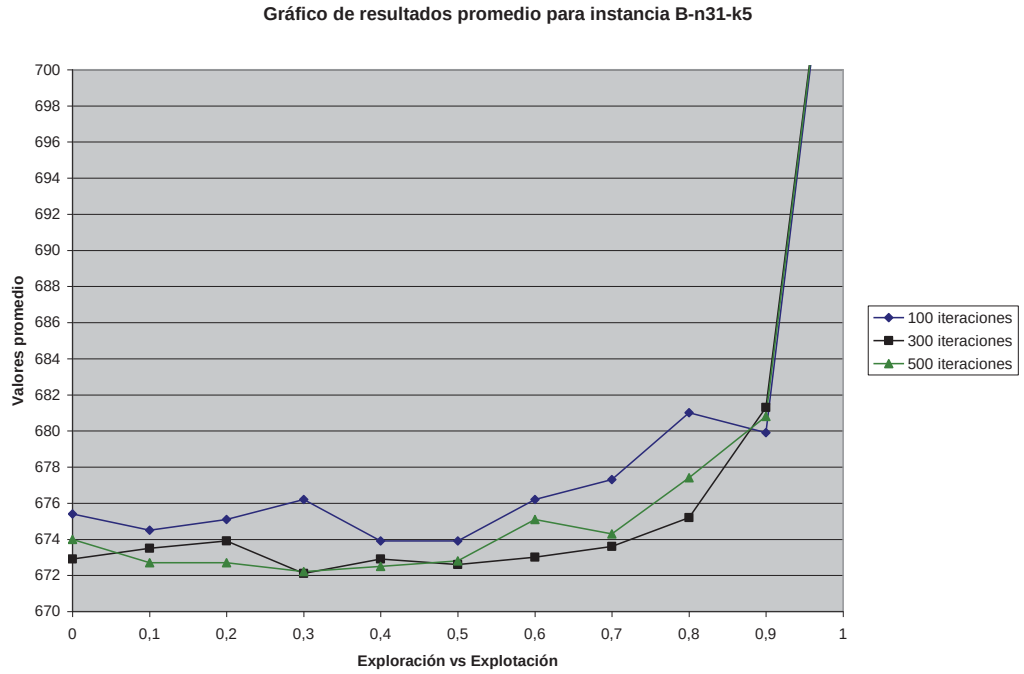

Gráfico (a)

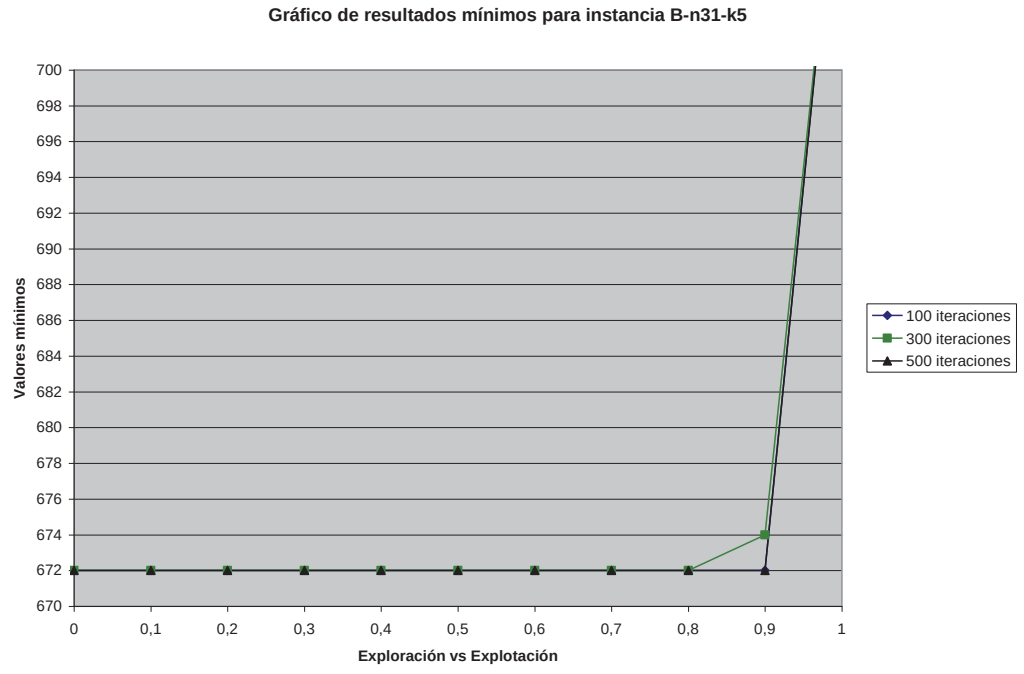

Gráfico (b)

Figura 4.9 Gráficos de valores promedio y mínimos para la instancia B-n31-k5.

En el gráfico (a) de la figura 4.8 se puede ver que, en promedio, con 300 iteraciones se logran mejores resultados que con 100, pero los que se lograron con 500 no mejoraron en la misma proporción, esto se puede deber a que ya están muy cercas del óptimo que es 784. Además se puede ver que para 300 y 500 hay una tendencia a dar mejores resultados con más exploración que explotación. Esto también puede verse en el gráfico (b) donde los mejores resultados, que justamente son el óptimo, se dieron con 500 iteraciones con $q_0 = 0$ y $q_0 = 0{,}3$.

En el gráfico (a) de la figura 4.9 se puede ver que el valor promedio de los 10 resultados está muy cerca del óptimo para 100, 300 y 500 iteraciones con q_0 entre 0 y 0,7 y que con 300 y 500 iteraciones el promedio era casi el valor óptimo con $q_0 = 0{,}3$. Esto se puede ver en la tabla 4.2 donde con 300 iteraciones el porcentaje de variación con respecto al promedio es 0,01% y con 500 el porcentaje es 0,03%, ambas con $q_0 = 0{,}3$. En el gráfico (b) de la figura 4.9 puede verse claramente que de 10 resultados con q_0 entre 0 y 0,8 siempre dio por lo menos una vez el valor óptimo ya sea para 100, 300 y 500 iteraciones.

A la vista de estos resultados se realizaron pruebas para estas y otras instancias con 500 iteraciones y $q_0 = 0{,}3$ que se estima es una de las combinaciones que dan mejores resultados. Por lo tanto manteniendo estos dos valores fijos se realizaron para las instancias A-n32-k5, A-n45-k7, B-31-k5, B-n50-k7 y P-n40-k5, 20 pruebas para cada una de ellas.

En la tabla 4.3 se pueden ver los porcentajes de variación, con respecto al valor óptimo, del promedio y del menor valor de los resultados de las 20 pruebas para cada una de las instancias mencionadas. Esta tabla está hecha basada en una llamada “tabla resumen de soluciones reportadas”, de Zabala y Torres (2005). Para las pruebas con 31 y 32 nodos se puede ver que los resultados son excelentes, pero para las otras tres que justamente usan de 40 nodos hacia arriba, se observa que los resultados ya no son tan buenos, por lo que se presume que el sistema creado aún necesita ajustes ya que baja la calidad de los resultados a medida que crece el tamaño de los problemas, en este caso, el número de clientes.

Tabla 4.3 Variaciones de los resultados con respecto al óptimo.

Instancia	Valor óptimo	Menor valor	% de variación	Promedio	% de variación	Tiempo en Segundos
A-n32-k5	784	784	0%	792,2	1,05%	33
A-n45-k7	1146	1191	3,93%	1230,95	7,41%	128
B-n31-k5	672	672	0%	672,85	0,13%	27
B-n50-k7	741	763	2,97%	782,35	5,58%	157
P-n40-k5	458	508	10,92%	536,85	17,22%	66

4.3 Implementación del VRPSPD con Sistema de Colonia de Hormigas

Para el desarrollo de la variante VRPSPD con Sistema de Colonia de Hormigas se modificó el código creado para el VRP el cual desde un principio, como ya se ha dicho, estaba destinado para ser usado con la variante propuesta. Las modificaciones para resolver el VRPSPD se detallarán en esta sección.

La implementación del VRPSPD que se muestra acá se probó solo para una pequeña instancia de siete clientes y una flota de tres vehículos. En la sección 4.4 se muestran los resultados del VRPSPD para instancias de la literatura. Precisamente el código que se muestra en el anexo 1 está adaptado para resolver el VRPSPD, en este caso para resolver las instancias propuestas por Dethloff para lo cual solo se debe cambiar el nombre del archivo de la instancia que se desea leer y cambiar el nombre del archivo que se creará para guardar los resultados.

4.3.1 Implementación y estructuras de datos

En esta sección no se explicará toda la implementación ni todas las estructuras de datos creadas ya que gran parte de esto es tomado del VRP y éste se explicó en la sección 4.1. Acá solo se explicarán las estructuras de datos creadas y modificaciones realizadas a la implementación del VRP para resolver el VRPSPD.

Para satisfacer tanto la entrega de productos a los clientes como de la recolección de otros productos desde éstos, se creó un nuevo vector que tiene guardadas las cantidades de productos a recolectar desde cada cliente; se asemeja al vector “entrega a clientes” de la figura 4.3.

También se creó un vector que simula la carga que tendría un vehículo después de visitar a cada cliente si quisiera visitar a uno nuevo esto ya que, para resolver el VRPSPD, no solo se debe considerar si la entrega total o la recogida total que hará el vehículo en su recorrido no sobrepasen la capacidad de éste sino que además, luego de visitar a cada cliente, se debe revisar la carga que lleva ya que existen casos donde ésta puede sobrepasar la capacidad del vehículo aunque las cargas totales, ya sea “carga total de entrega” cuando parte del depósito y “carga total recogida” cuando llega a éste, no lo hagan.

En el pseudocódigo de la figura 4.10 se puede ver como se simula la carga de un vehículo después de visitar a cada cliente que pertenece a su ruta, si se tomara un nuevo nodo a ser visitado. En esta figura, que j sea mayor o igual a 1 significa que se tomará a cualquier nodo como posible de ser visitado a excepción del depósito.

No aparece en el pseudocódigo, pero luego de finalizar éste (mejor dicho, luego de finalizar el código representado en esta figura), se verifica si la “carga total de entrega” sobrepasa o no la capacidad del vehículo, ya que en este caso solo se ha sumado la carga del posible nodo a visitar, como se muestra en la segunda línea de la figura.

El vector carga simulada, en la posición j, como lo dice la figura 4.10 en la tercera línea, representa la carga total a recoger si se considera la del posible nodo a visitar y acá sí se comprueba inmediatamente si sobrepasa la capacidad del vehículo. El vector carga simulada, en la posición d, representa la carga que llevaría un vehículo después de visitar a cada cliente de su ruta si se agregara la visita al nodo j en ésta. La primera vez se toma la carga total de entrega representado por cargaveh, se resta lo que tiene que entregar al cliente d y se suma lo que tiene que recibir de éste, para ver si después de atenderlo se sobrepasa o no la capacidad. Luego cargaveh toma este valor para representar la carga antes de atender al siguiente cliente y así sucesivamente se comprueba toda la ruta desde el primer cliente a visitar.

Como se puede ver el pseudocódigo (a) de la figura 4.11, cuando se elige el siguiente nodo a visitar que no sea el depósito, a la variable clientes visitados se le suma 1 y a total

recogida se le asigna el valor de carga simulada en la posición del cliente elegido. Como lo muestra el pseudocódigo (b) de la misma figura, en caso de que el nodo escogido sea el depósito, a la variable "nuevo vehículo" se le asigna clientes visitados más 1, a clientes visitados se le suma 1 y a total recogida se le asigna 0.

```
Si ( j >= 1)
    carga total de entrega ← carga total de entrega + entrega a clientes [j]
    carga simulada [j] ← total recogida + recogida de clientes [j]
    Si(carga simulada [j] > capacidad )          /*si la carga total a recoger supera la capacidad del vehículo*/
        no se puede visitar a ese cliente
    Fin si
    Si (nuevo vehículo <> clientes visitados)
        cargaveh ← carga total de entrega
        Para cada (c = nuevo vehículo; mientras c < clientes visitados)
            d ← memoria de la hormiga [c]
            carga simulada [d] ← cargaveh – entrega a clientes [d] + recogida de clientes [d]
            Si(carga simulada [d] > capacidad )
                no se puede visitar a ese cliente
            Fin si
            cargaveh ← carga simulada [d]
        Fin Para cada
    Fin Si
Fin Si
```

Figura 4.10 Pseudocódigo que representa la verificación de que la capacidad del vehículo no se sobrepase.

4.3.2 Aplicación a una instancia pequeña y resultados mostrados

Se usó el código modificado para resolver el VRPSPD en una instancia pequeña de siete clientes y una flota de tres vehículos. Se guardaron las distancias entre los propios clientes y entre éstos y el depósito en una matriz cuadrada simétrica de tamaño 8 donde los valores de la diagonal no son usados ya que existe la restricción de que un vehículo no puede tomar como nodo a ser visitado el mismo donde se encuentra. Además se guardaron los valores correspondientes a las cantidades de productos a entregar a cada cliente en un vector de tamaño siete y también se guardaron las cantidades de productos a recibir por

parte de los clientes, esto también en un vector de tamaño 7. Los valores tanto de la matriz como de los vectores se pueden ver en la figura 4.12.

```
clientes visitados ← clientes visitados + 1
total recogida ← carga simulada [j]          /*acá j representa el nodo escogido a ser visitado*/
```

Pseudocódigo (a)

```
nuevo vehiculo ← clientes visitados + 1
clientes visitados ← clientes visitados + 1
total recogida ← 0
```

Pseudocódigo (b)

Figura 4.11 Pseudocódigos que representan que se hace: si el nodo escogido a visitar no es el depósito, en el caso de (a), o si lo es, en el caso de (b).

Matriz de distancias

0	5	7	3	2	5	7	5
5	1000	8	2	4	1	10	7
7	8	1000	3	8	5	7	3
3	2	3	1000	7	10	3	2
2	4	8	7	1000	7	10	7
5	1	5	10	7	1000	1	7
7	10	7	3	10	1	1000	5
5	7	3	2	7	7	5	1000

Entrega a clientes

7	3	5	7	8	9	3

Recogida de clientes

5	7	7	8	3	2	8

Figura 4.12 Valores de prueba ingresados para una instancia de siete clientes.

En la tabla 4.4 se pueden ver los parámetros utilizados tanto de la prueba específica como de los propios del Sistema de Colonia de Hormigas. En esta tabla se puede ver que el parámetro q_0 se utilizó con distintos valores al igual que la cantidad de iteraciones. La tabla 4.5 muestra los resultados obtenidos para 10 pruebas con estas combinaciones de los parámetros q_0 e iteraciones.

Tabla 4.4 Valores de parámetros para la instancia utilizada.

Parámetros	**Valores**
Número de clientes	7
Número de vehículos	3
Capacidad de los vehículos	20
Cantidad de iteraciones	20, 50 y 100
Explotación vs Exploración (q_0)	0; 0,3; 0,5; 0,7 y 1
Cantidad de hormigas	10
Importancia relativa (β)	2
Decremento de feromona (ρ)	0,1
Decremento de feromona (α)	0,1

Se puede ver en la tabla 4.5 que ya con solo 50 iteraciones se llega al mejor valor encontrado que es 37 y además que para esta instancia de tamaño pequeño se requiere prácticamente solo realizar exploración para llegar a los mejores resultados. Aunque se realicen 100 iteraciones, si se deja un porcentaje muy alto a explotación no se logra llegar a los mejores resultados en la mayoría de las pruebas.

Lo importante, más que analizar los resultados, es que la utilización de esta instancia propia sirvió para ver que las modificaciones al VRP para resolver el VRPSPD dieron resultados satisfactorios con lo que se está en condiciones de llevar al sistema a la utilización de instancias de la literatura y lograr resultados de buena calidad que es la meta de este proyecto, lo que se mostrará en la sección 4.4.

Tabla 4.5 Resultados para el VRPSPD de una instancia pequeña.

20 iteraciones		
q_0	**Menor valor**	**Promedio**
0	37	37,5
0,3	37	37,6
0,5	37	38,2
0,7	37	38,6
1	41	41
50 iteraciones		
q_0	**Menor valor**	**Promedio**
0	37	37
0,3	37	37,3
0,5	38	38,6
0,7	37	38,4
1	41	41
100 iteraciones		
q_0	**Menor valor**	**Promedio**
0	37	37
0,3	37	37
0,5	37	37,3
0,7	37	38,7
1	41	41

4.4 Experimentos para el VRPSPD con instancias de la literatura y sus respectivos análisis

En esta sección se mostrarán los resultados aplicados a 20 instancias de 50 clientes, propias del VRPSPD, tomadas de las propuestas por Dethloff (2001), las cuales se dividen en SCA que son instancias en que las coordenadas de los clientes están uniformemente distribuidas entre [0,100], y el resto, llamadas CON, en que la mitad de las coordenadas de los clientes están uniformemente distribuidas como en SCA, y las coordenadas de la otra mitad de éstos están uniformemente distribuidas entre [100/3,200/3], este último caso simula una distribución urbana, en donde, la mayor cantidad de los clientes están en "un noveno" del área, específicamente en la parte central. Las cantidades de entrega a los

clientes están uniformemente distribuidas sobre el intervalo [0,100]. En cambio la cantidad a recoger desde éstos, que llamaremos P_j, que se corresponde con la cantidad de entrega D_j, es resuelta a través de una fórmula que usa un valor aleatorio r_j uniformemente distribuido sobre el intervalo [0,1], la cual es $P_j = (0{,}5 + r_j) \cdot D_j$, Dethloff (2001).

Se modificó el código en como tomar las distancias entre cada par de nodos ya que para resolver las instancias de Dethloff, sus archivos entregan directamente estas distancias en vez de entregar las coordenadas de todos los nodos para calcular las distancias euclidianas.

Primero se hizo un análisis de sensibilidad para la instancia SCA3-0, para decidir los valores se utilizarían en los parámetros involucrados en estas pruebas para obtener mejores resultados. Se hicieron 10 pruebas para cada combinación de valores de q_0 (0,2;0,3;0,4), β (1,5;2;2,5;3;3,5) y un parámetro creado llamado importancia del depósito en que el valor 1 indica que el depósito puede ser escogido para ser visitado como cualquier nodo, pero si por ejemplo el valor es 0,1 significa que la probabilidad de ser escogido se reduce al 10% de lo que iba a ser originalmente. Este valor de importancia del depósito, que ahora se llamará i.d., se tomó para valores 0,1; 0,2 y 0,3. De este estudio se apreció que los mejores resultados se obtenían con i.d.= 0,3 y con β entre 3 y 3,5.

Con los datos obtenidos se verificó nuevamente la sensibilidad de los parámetros, pero con valores cercanos a los que dieron mejores resultados, estos son i.d. (0,3;0,4;0,5), β (3;3,5;4) y q_0 (0,2;0,3;0,4;0,5), nuevamente obteniendo 10 resultados para cada combinación. Los resultados demostraron que nuevamente el mejor valor de i.d. es 0,3 quedando la duda sobre los valores de β y q_0. La tabla 4.6 muestra los resultados para i.d. 0,3 y distintos valores de β y q_0.

Finalmente se decidió, viendo los resultados de los dos análisis, que solo faltaba decidir entre q_0 = 0,3 o q_0 = 0,4 para 500 iteraciones, i.d.= 0,3 y β = 3,5; esta vez realizando 20 pruebas para cada valor de q_0. Luego de analizar los resultados se decidió que q_0 = 0,3 sería adecuado para junto con el resto de valores escogidos para los parámetros ya señalados realizar pruebas al resto de las instancias. Tanto los valores de parámetros escogidos como las instancias utilizadas se muestran en las tablas 4.7 y 4.8 respectivamente.

Tabla 4.6 Resultados de la instancia SCA3-0 para distintos valores de β y q_0.

500 iteraciones, β = 3 e i.d. = 0,3		
q_0	**Menor valor**	**Promedio**
0,2	671	703,6
0,3	667	701,5
0,4	676	704,4
0,5	673	686,2
500 iteraciones, β = 3,5 e i.d. = 0,3		
q_0	**Menor valor**	**Promedio**
0,2	655	685,5
0,3	659	693
0,4	666	682,5
0,5	668	697,3
500 iteraciones, β = 4 e i.d. = 0,3		
q_0	**Menor valor**	**Promedio**
0,2	656	690,6
0,3	667	691
0,4	669	690,8
0,5	669	699,1

Tabla 4.7 Valores de los parámetros utilizados para resolver instancias de Dethloff con ACS.

Parámetros	**Valores**
Número de clientes	50
Número de vehículos	4
Cantidad de iteraciones	500
Importancia del depósito	0,3
Explotación vs Exploración (q_0)	0,3
Cantidad de hormigas	10
Importancia relativa (β)	3,5
Decremento de feromona (ρ)	0,1
Decremento de feromona (α)	0,1

En la tabla 4.8 se muestran los mejores valores encontrados de 20 pruebas para cada una de las instancias escogidas. Estos valores obtenidos se encuentran en la columna ACS, y el resto de los valores corresponden a pruebas realizadas por Dethloff (2001) y Tang y Galvão (2006). Se puede ver en ésta tabla que los resultados obtenidos son buenos en comparación a los de los otros autores, pero además puede observarse que este valor es mejor al de ambos en la instancia CON3-0, Cabrera y Peralta (2009).

Tabla 4.8 Comparación de los mejores valores encontrados.

Instancia	Dethloff		Tang y Galvão		ACS	
	Mejor	Vehículos	Mejor	Vehículos	Mejor	Vehículos
SCA3-0	689.0	-	640.55	4	656	4
SCA3-1	765.6	-	697.84	4	705	4
SCA3-2	742.8	-	659.34	4	662	4
SCA3-3	737.2	-	680.04	4	695	4
SCA3-4	747.1	-	690.50	4	712	4
SCA3-5	784.4	-	659.90	4	669	4
SCA3-6	720.4	-	653.81	4	669	4
SCA3-7	707.9	-	659.17	4	682	4
SCA3-8	807.2	-	719.47	4	733	4
SCA3-9	764.1	-	681.00	4	694	4
CON3-0	672.4	-	631.39	4	622	4
CON3-1	570.6	-	554.47	4	570	4
CON3-2	534.8	-	522.86	4	525	4
CON3-3	656.9	-	591.19	4	596	4
CON3-4	640.2	-	591.12	4	602	4
CON3-5	604.7	-	563.70	4	583	4
CON3-6	521.3	-	506.19	4	520	4
CON3-7	602.8	-	577.68	4	588	4
CON3-8	556.2	-	523.05	4	541	4
CON3-9	612.8	-	580.05	4	600	4

5 Conclusiones

En la primera parte de este trabajo se mostró cual era la meta de este proyecto para lo cual se dio información básica para entender de que se trata el VRP, cuáles son sus variantes, su forma de resolverlo y sus aplicaciones. Luego se mostró información sobre la variante VRPSPD, la cual utiliza logística reversa, que era la meta de este proyecto de título. Después se detalló la forma elegida para resolverlo la cual era Sistema de Colonia de hormigas. Más adelante se explicó cómo fue resuelto el VRP en su variante CVRP y se dieron resultados de esta implementación para luego pasar a la parte más importante del proyecto que fue la adaptación del sistema para que resuelva el VRPSPD, que en un primer momento fue probado para una instancia pequeña, pero luego se mostraron los resultados al aplicarlo a instancias de la literatura.

Se realizaron distintos análisis de sensibilidad para obtener la mejor combinación de parámetros y finalmente se compararon los resultados obtenidos con los de otros autores, culminando este proyecto con la satisfacción de haber conseguido, no sólo terminar el desarrollo del VRPSPD con un Sistema de Colonia de Hormigas, sino también por la calidad de los resultados.

Además todavía se podría mejorar aún más el método acá presentado, ya sea por la adición de otra heurística que complemente a la ya realizada, como puede ser una búsqueda local, como también seguir haciendo análisis de sensibilidad para afinar todavía más los valores de los parámetros.

También se debe considerar que el Sistema de Colonia de Hormigas fue tratado de forma local, es decir, cada hormiga comenzaba su recorrido sólo después de haber terminado la anterior, lo que limita la simulación de una colonia verdadera en que todas buscan el alimento en forma simultánea, por lo que a futuro este sistema puede ser planteado en forma distribuida, lo que supone mejorar aún más los resultados.

6 Bibliografía

Alonso, S., Cordón, O., Fernández de Viana, I., Herrera, F., 2004. La Metaheurística de Optimización Basada en Colonias de Hormigas: Modelos y Nuevos Enfoques. G. Joya, M.A. Atencia, A. Ochoa, S. Allende (Eds.), Optimizacion Inteligente: Técnicas de Inteligencia Computacional para Optimización, Servicio de Publicaciones de la Universidad de Málaga, 261-313.

Barajas, N., 2006. Estado del Arte del Problema de Ruteo de Vehículos (VRP). Universidad Nacional de Colombia. Facultad de Ingeniería. Maestría en Ingeniería de Sistemas y Computación.

Cabrera, G., Peralta, F., 2009. Ant Colony System Algorithm for the Vehicle Routing Problem with Simultaneous Delivery and Pick-Up. iccit, pp.1575-1580, 2009 Fourth International Conference on Computer Sciences and Convergence Information Technology

Coello, C., 1997. Búsqueda Tabú: Evitando lo Prohibido. Soluciones Avanzadas. Tecnologías de Información y Estrategias de Negocios, Año 5, Número 49, pp. 72-80.

Dethloff, J., 2001. Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. OR Spektrum 23: 79–96.

Díaz, O., 2006. Problema del Transporte. VRP (Vehicle Routing Problem). Propuesta de investigación doctoral ante el cuerpo académico de optimización y software. Centro de Investigación en Ingeniería y Ciencias Aplicadas (CIICAp). Universidad Autónoma del Estado de Morelos.

Dorigo, M., Di Caro, G., Gambardella, L. M., 1999. Ant Algorithms for Discrete Optimization. Artificial Life, MIT Press.

Dorigo, M., Gambardella, L. M., 1997. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. Accepted for publication in the IEEE Transactions on Evolutionary Computation, Vol.1, No.1. In press.

Espinoza, D., 2006. EL Problema del Vendedor Viajero (TSP) y Programación Entera (IP). Universidad de Chile. Facultad de Ciencias Físicas y Matemáticas. Departamento de Ingeniería Industrial.

Hermosilla, A., Barán, B., 2004. Comparación de un sistema de colonias de hormigas y una estrategia evolutiva para un Problema Multiobjetivo de Ruteo de Vehículos con Ventanas de Tiempo. Proceedings of CLEI'2004. Latin-American Conference on Informatics (CLEI), Arequipa, Perú.

Melián, B., Moreno, J. A., Moreno, M., 2003. Metaheurísticas: una visión global. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.19, pp. 7-28.

Modo, 2008. ¿Qué es Soft Computing? http://modo.ugr.es/es/soft_computing (accedido abril 2008).

Perissé, M. C., 2001. Proyecto Informático Una Metodologías Simplificada. http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/index.htm (accedido febrero 2010).

Tang Montané, F. A., Galvão, R. D., 2006. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. Computers and Operations Research, 33(3), 595–619.

The VRP Web, 2008. VRP Variants. http://neo.lcc.uma.es/radi-aeb/WebVRP/ (accedido abril 2008).

Zabala, C., Torres, J., 2005. Implementación del Sistema de Colonia de Hormigas con Búsqueda Local al Problema de Ruteo de Vehículos con Capacidad y Ventanas de Tiempo (CVRPTW). Departamento de Ingeniería Industrial, Universidad de los Andes, Bogotá, Colombia.

Zachariadis, E., Tarantilis, C., Kiranoudis, C., 2007. A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. Expert Systems with Applications 36 (2009) 1070-1081.

Apéndice 1: Algoritmo para Sistema de Colonia de Hormigas

A continuación se presenta el algoritmo para ACS propuesto por Dorigo y Gambardella (1997).

```
1./* Initialization phase */
  For each pair (r,s) τ(r,s):= τ0  End-for
   For k:=1 to m do
      Let rk1 be the starting city for ant k
      Jk(rk1):= {1, ..., n} - rk1
      /* Jk(rk1) is the set of yet to be visited cities for
         ant k in city rk1 */
      rk:= rk1                        /* rk is the city where ant k is located */
   End-for
2. /* This is the phase in which ants build their tours. The tour of ant k
      is stored in Tourk. */
   For i:=1 to n do
     If i<n
      Then
       For k:=1 to m do
         Choose the next city sk according to Eq. (3) and Eq. (1)
         Jk(sk):= Jk(rk) - sk
         Tourk(i):=(rk ,sk)
       End-for
      Else
       For k:=1 to m do
       /* In this cycle all the ants go back to the initial city rk1 */
         sk := rk1
         Tourk(i):=(rk ,sk)
       End-for
     End-if
    /* In this phase local updating occurs and pheromone is
       updated using Eq. (5)*/
     For k:=1 to m do
       τ(rk ,sk):=(1-ρ)τ(rk ,sk)+ ρτ0
       rk := sk /* New city for ant k */
     End-for
   End-for
3. /* In this phase global updating occurs and pheromone is updated */
  For k:=1 to m do
    Compute Lk   /* Lk is the length of the tour done by ant k*/
  End-for
  Compute Lbest
  /*Update edges belonging to Lbest using Eq. (4) */
  For each edge (r,s)
      τ(rk ,sk):=(1-α)τ( rk ,sk)+ α(Lbest)-1
  End-for
4. If (End_condition = True)
    then Print shortest of Lk
    else goto Phase 2
```

Anexo 1: Técnicas de solución del VRP

Existen distintas técnicas o métodos para solucionar el VRP entre ellos están los exactos y las heurísticas y metaheurísticas. La resolución del VRP se hace ineficiente en la búsqueda de la solución óptima para problemas de gran tamaño por lo que los métodos exactos, que encuentran esta solución óptima, usarían demasiados recursos que los computadores actuales no estarían en condiciones de soportar. Por esta razón se prefiere usar métodos heurísticos, más específicamente las metaheurísticas, para resolver el VRP y otros problemas de optimización combinatoria ya que entregan soluciones cercanas al óptimo, con un uso razonable de recursos. Solo se verá en forma breve algunos métodos exactos y heurísticas clásicas para hacer énfasis en las metaheurísticas que son los métodos más usados hoy en día.

- **Métodos exactos y heurísticas clásicas**

Branch and Bound es un método exacto que encuentra la solución óptima a un problema construyendo un árbol con las posibles soluciones excepto las ramificaciones donde se supere el tamaño de la distancia o tiempo total de una solución inicial dada. Aunque se ahorren recursos computacionales, con este acotamiento, sigue siendo ineficiente para problemas grandes.

Branch and Cut es método exacto utilizado, pero también existen antiguos métodos heurísticos clásicos para resolver el VRP como son: **Algoritmo de ahorro de Clark and Wright** y algoritmos de dos fases como el método de **Barrido** y el de **Pétalo**. La definición de Branch and Bound está basada en Barajas (2006) donde se puede encontrar información de cómo funcionan los demás métodos mencionados.

- **Metaheurísticas**

En Modo (2008) se dice que suele existir mucha controversia por la diferencia entre heurística y metaheurística. Aquí se menciona que el término heurística proviene de la palabra griega "heuriskein" que tiene un significado que tiene que ver con "encontrar" y que con ese origen se han creado muchos procedimientos heurísticos para problemas de optimización que al sacar lo mejor de ellos y usarlos en otros problemas o ámbitos ayuda al desarrollo de la investigación en esta área y por esta razón aparecen las metaheurísticas cuyo origen de la palabra viene de heurística junto con el sufijo meta que quiere decir "más allá" o "de nivel superior", por lo que, según este texto, "el concepto de metaheurística tiene un carácter más generalista que el de heurística".

Según Melián *et al.* (2003), las metaheurísticas debieran tener un conjunto de "propiedades deseables" para favorecer su interés práctico y teórico, entre las cuales están:

- **Simplicidad**, que se refiere a que la metaheurística debe estar basada en un principio sencillo y claro, para su fácil comprensión.
- **Precisión,** esto es, que los pasos y fases de la metaheurística deben estar declarados de forma concreta.
- **Coherencia,** que significa que los elementos deben deducirse naturalmente de sus principios.
- **Efectividad,** quiere decir que los algoritmos que se aplican deben dar soluciones cercanas al óptimo.
- **Eficacia,** es que al resolver un problema real con la metaheurística, la probabilidad de obtener soluciones cercanas al óptimo debe ser grande.
- **Eficiencia,** es usar de buena forma los recursos computacionales, tales como tiempo y espacio.
- **Generalidad,** quiere decir que sirva para diversos tipos de problema.
- **Adaptabilidad,** como la palabra lo dice, debe ser adaptable para distintos contextos de aplicación o cambios que se produzcan en el modelo.
- **Robustez,** la metaheurística debiera comportarse sin problemas ante pequeños cambios en el contexto de aplicación o modelo.

- **Interactividad,** significa que el usuario pueda aplicar lo que ha aprendido anteriormente para que la metaheurística funcione mejor.
- **Multiplicidad,** es que se deben dar diferentes soluciones al usuario para que elija, y éstas deben ser muy cercanas al óptimo.
- **Autonomía,** como la palabra lo dice, la metaheurística debe dejar que se produzca un funcionamiento autónomo, sin parámetros, o que estos puedan aparecer automáticamente.

A continuación se describirán brevemente dos de las metaheurísticas más importantes que existen.

- **Algoritmos genéticos:** Se basan en la evolución de las especies, es decir, sobreviven los mejores individuos, los más aptos, para lo cual se toman las mejores soluciones, llamadas padres y se combinan para obtener una solución con lo mejor de cada una, es decir, un hijo. Estas iteraciones se van realizando hasta que el individuo de la última generación es una solución adecuada para el problema. Estas soluciones se llaman cromosomas, que son cadenas en forma de vector a las que se le realizan operaciones como cruce, que obtiene una solución a partir de otras dos, mutación, que consiste en intercambiar un par de genes (o casilleros del vector) de una misma solución, e inserción, que cambia el orden de una parte de la solución. (Barajas, 2006). En la figura anexo.1 se puede ver el mecanismo básico del método en el cual en "operadores genéticos" es donde se realizan las operaciones antes mencionadas.
- **Búsqueda Tabú:** Este método mientras busca nuevas soluciones mantiene una lista tabú, que son movimientos prohibidos de utilizar, para evitar caer en óptimos locales o ciclos. Estos movimientos se prohíben por corta duración para más adelante poder ser utilizados nuevamente. Esta técnica deja empeorar la solución que se tenga en ese momento, pero esto se permite para ampliar el espacio de búsqueda y por este camino, tal vez, encontrar una mejor solución. Glover fue quien presentó esta técnica y también dice que puede ser llamada búsqueda de "inhibición débil" al prohibir una parte muy pequeña de

movimientos con respecto a las posibilidades que hay y se mantienen así por corto tiempo. Esto contrasta con la técnica Branch and Bound que también evita ciertos movimientos, pero en forma permanente, lo que hace ser considerada como búsqueda con "inhibición fuerte", (Coello, 1997). En la figura anexo.2 se puede ver el funcionamiento básico de esta técnica.

Figura anexo.1 Algoritmo genético básico, (Barajas, 2006).

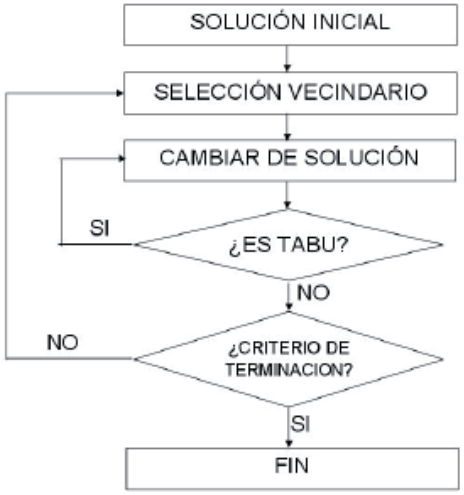

Figura anexo.2 Búsqueda tabú, (Barajas, 2006).

Anexo 2: Código fuente del VRPSPD con Sistema de Colonia de Hormigas para resolver instancias de Dethloff