UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

INGENIERÍA DE SISTEMAS – INVESTIGACIÓN DE OPERACIONES

IMPLEMENTACIÓN DE UNA HEURÍSTICA INSPIRADA EN SISTEMAS DE HORMIGAS, CON SIMULACIÓN BASADA EN AGENTES PARA PROBLEMAS DE RUTEO VEHICULAR CAPACITADO

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERÍA

PRESENTA:

ING. SALVADOR AGUILAR GUADALAJARA

DRA. MAYRA ELIZONDO CORTÉS
Facultad de Ingeniería

MÉXICO, D. F. MAYO 2015

JURADO ASIGNADO

Presidente: Dra. Flores de la Mota Idalia

Secretario: Dra. Balderas Cañas Patricia

Vocal: Dra. Elizondo Cortés Mayra

1er Suplente: M. en I. Durán Rojas María Guadalupe

2do Suplente: Dr. Ordorica Mellado Manuel

Lugar donde se realizó la tesis:

Universidad Nacional Autónoma de México, Posgrado de la Facultad de Ingeniería

TUTOR DE TESIS:

Dra. Mayra Elizondo Cortés

FIRMA

ÍNDICE GENERAL

ÍNDICE DE FIGURAS.

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICAS

RESUMEN

El presente trabajo hace énfasis en una problemática de la vida real, que sobrellevan la mayoría de las empresas a nivel mundial y es, el transporte, no sólo de bienes sino también de servicios, que se requieren de manera óptima y a bajo costo. Nos enfocamos en el Problema de Ruteo Vehicular Capacitado (*CVRP*) por sus siglas en inglés *Capacitated Vehicle Routing Problem*, recordando que la naturaleza de este problema es de tipo NP-duro, lo cual nos hace seguir tratando de desarrollar herramientas que nos proporcionen buenas soluciones en tiempos razonables y de manera económica. En nuestro caso, decidimos trabajar con la Simulación Basada en Agentes (*ABS*) por sus siglas en inglés *Agent-Based Simulation*, en específico con un *software* libre llamado NetLogo, el cual permite programar con el enfoque de (*ABS*), y la base de nuestro trabajo es la implementación de un híbrido que utiliza el algoritmo de primero agrupar después rutear (*cluster-first, route-second methods*) y una heurística basada en Sistemas de Hormigas (*AS*) por sus siglas en inglés *Ant Systems*. La metodología aplicada en este trabajo, nos permitió obtener resultados con un GAP de menos del 20% con respecto al óptimo, lo cual cumple con el objetivo propuesto. Además, gracias al *tuning* de los parámetros, pudimos mejorar de manera considerable los resultados conforme avanzamos en el desarrollo de esta tesis.

INTRODUCCIÓN

Con la necesidad de transportarse surge un nuevo problema, hacerlo de manera eficiente. Una de las cuestiones más importantes para la industria a nivel mundial es el transporte eficiente de sus mercancías, ya sea materia prima, producto terminado o servicios, así que, es de vital importancia hacerlo en tiempo y forma y a menor costo.

Inspirados en lo dicho anteriormente, hemos plasmado en este trabajo el uso de las herramientas que desde hace algunas décadas se vienen implementando gracias al avance tecnológico de equipos de cómputo cada vez más poderosos en el procesamiento de datos. Lo que aquí utilizamos es la Simulación Basada en Agentes (*ABS*) por sus siglas en inglés *Agent-Based Simulation,* para resolver un problema de ruteo vehicular capacitado (*CVRP*) por sus siglas en inglés *Capacitated Vehicle Routing Problem,* y una heurística basada en Sistemas de Hormigas (*AS*) por sus siglas en inglés *Ant Systems*, recordando que la naturaleza del problema *CVRP* es de tipo NP-Duro, lo cual nos impulsa a seguir desarrollando herramientas que nos permitan solucionar este tipo de problemas en tiempos razonables y de manera económica ya que tienen gran impacto en las actividades de la vida real.

De lo anterior llegamos al planeamiento del siguiente objetivo:

Utilizar la simulación basada en agentes (*ABS*) mediante la implementación de un híbrido que consiste en la aplicación de un método de dos fases "primero agrupar luego rutear" y una heurística basada en sistemas de hormigas (*AS*), para resolver problemas de ruteo vehicular capacitado[1] logrando con esto, acercarnos al óptimo a menos del 20% en instancias medianas y grandes obtenidas de las librerías TSPLIB[2] y COIN-OR[3], además de permitir una fácil implementación apoyándonos en un programa de acceso libre.

El presente trabajo consta de 4 capítulos mismos que se estructuran de la siguiente manera:

- Capítulo I. Partimos de los problemas logísticos que existen en la vida real y como son interpretados desde el punto de vista de la Investigación de Operaciones hasta llegar a la formulación general de problemas *VRP* y en particular problemas tipo *CVRP.*
- Capítulo II. Tratamos lo referente al estado del arte, el marco teórico y la estrategia de investigación seguida para la realización de ésta tesis.
- Capítulo III. Se desarrolla la metodología de la simulación utilizada para la obtención de resultados producto del trabajo realizado y se explica de manera detallada cada uno de los puntos llevados a cabo.
- Capitulo IV. Consiste en el diseño, ejecución y análisis de experimentos, se presentan los resultados obtenidos al aplicar la metodología de la simulación utilizada, se detallan los efectos propios de aplicar variaciones en los parámetros de la simulación y como estos nos acercan o alejan de la solución óptima. Éste capítulo cierra con las conclusiones obtenidas a partir del desarrollo del presente trabajo.

[1] Los problemas a tratar pertenecen a las librerías TSPLIB y COIN-OR

[2] Disponible en: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

[3] COmputational INfrastructure for Operations Research. Disponible en: http://www.branchandcut.org/

Capítulo 1

I. EL PROBLEMA DE RUTEO VEHICULAR CAPACITADO

En el ámbito de competencia que caracteriza al siglo XXI, la logística industrial es usada por las compañías con el fin de generar ventajas competitivas. Dentro de este contexto son de vital importancia los procesos de aprovisionamiento y/o distribución, por lo que el establecimiento de las rutas para vehículos de una manera óptima ha generado un gran interés investigativo. Como resultado, se han propuesto un sin número de modelos que abarcan este problema con el fin de mejorar el desempeño logístico [Rocha, L. *et al.* 2011].

Para la gestión eficaz de los recursos, bienes y servicios en sistemas de distribución, a lo largo de los años se ha utilizado *software* de Optimización, Investigación de Operaciones y técnicas de Programación Matemática. Un gran número de aplicaciones en la vida real, nos muestran que el uso de procedimientos matemáticos en los procesos de planificación de la distribución, generan un ahorro substancial, de alrededor del 5% al 20% [Toth and Vigo. 2002] de los costos globales de transporte. Dicho lo anterior, es fácil observar que el ahorro en el sistema económico global es significativo. De hecho, los procesos de transporte involucran todas las etapas de los sistemas de producción y distribución y representan, generalmente, alrededor del 10% al 20% de los costos totales de los bienes [Toth and Vigo. 2002].

El problema de ruteo vehicular (*VRP*) por sus siglas en inglés *Vehicle Routing Problem*, es utilizado para la determinación del conjunto de rutas óptimas utilizadas por una flota de vehículos para atender a un conjunto de clientes, este es uno de los más importantes y estudiados, problemas de optimización combinatoria.

Podemos decir, que el problema de ruteo vehicular es un problema de gran interés en diversas áreas debido a la complejidad y alto costo computacional al tratar de resolver instancias grandes del problema (más de 50 clientes [Toth and Vigo. 2002]).

Día con día, los vehículos dentro de su jornada laboral se dirigen a cada uno de sus clientes, ubicados en diferentes puntos geográficos, para entregar los pedidos de sus productos o servicios, teniendo como punto de partida y retorno algún centro de distribución, depósito, planta, oficina o lugar en común [Toth and Vigo. 2002].

La Investigación de Operaciones aborda el problema con la finalidad de determinar el orden en que se deben visitar a los clientes y de esta manera proporcionar la ruta óptima,

es decir la de menor costo, para hacer las respectivas entregas de productos o servicios, tomando en cuenta que estos están dispersos geográficamente.

Nuestro interés por el problema de Ruteo Vehicular (*VRP*) y en específico su variante de (*CVRP*) por sus siglas en inglés *Capacitated Vehicle Routing Problem* está motivado tanto por su utilidad práctica como por su considerable dificultad computacional: los mayores casos de C*VRP* que se pueden resolver por los algoritmos exactos más efectivos propuestos hasta ahora contienen cerca de 50 clientes, mientras que instancias más grandes pueden ser resueltas de manera óptima sólo en casos particulares [Toth and Vigo. 2002].

El problema de VRP es uno de los más comunes en la optimización de operaciones logísticas y uno de los más estudiados; plantea la búsqueda de la solución óptima con diferentes restricciones tales como: número de vehículos, su capacidad, lugares de destino (clientes) y demanda de los clientes, entre otras. Una formulación de éste tipo puede incluir un amplio número de variables y diversos parámetros. Este tema presenta un interés práctico y académico por constituirse en un problema de optimización combinatoria y pertenecen en su mayoría a la clase *NP-Hard*, pues no es posible construir algoritmos que resuelvan cualquier instancia del problema (a no ser que P=NP) [Toth and Vigo. 2002] [M. Balinzki and R. Quandt. 1964] [W. M. Garvin. *et al.* 1957] [Rocha, L. *et al.* 2011].

El problema de ruteo vehicular o *VRP*, está implícito en el área de transporte, logística y distribución, y es uno de los problemas más conocidos y, por su naturaleza, desafiantes en la programación entera debido a que el tiempo y esfuerzo computacional necesario para resolverlo aumenta de manera exponencial conforme la instancia del problema crece.

Este problema fue planteado por primera vez por Dantzig y Ramser en 1959 [G. B. Dantzig and J. H. Ramser. 1959]. Debido al carácter combinatorio del problema y a las implicaciones económicas de las aplicaciones del *VRP*, ha recibido mucha atención y algunos algoritmos tanto exactos como heurísticos, han sido desarrollados desde que apareció por primera vez. Dichos autores describieron una aplicación del mundo real, relacionada con la entrega de combustible a la estación de servicio y proponen la primera formulación de programación matemática con un enfoque algorítmico [Toth and Vigo. 2002].

Para resolver el *VRP*, en 1964 Clarke y Wright [G. Clarke and J. Wright. 1964] proponen un eficaz heurístico glotón que mejora el enfoque de Dantzig-Ramser.

Continuando con estos dos principales artículos, cientos de modelos y algoritmos se propusieron para dar soluciones tanto exactas como aproximadas, a diferentes versiones del problema de ruteo (*VRP*).

Por otro lado, en lo relacionado a complejidad computacional, Lenstra y Rinnooy kan [J. K. Lenstra and A.H.G. Rinnooy Kan. 1981], en 1981 demostraron que este problema pertenece al tipo NP-Duro [Antonio Brandão. *et al.* 1997] algo de lo cual hablaremos más adelante.

Hoy en día, decenas de programas computacionales para la solución de problemas del mundo real relacionados con (*VRP*) ya están disponibles en el mercado.

Actualmente procesos conocidos como metaheurísticas son utilizados para resolver problemas complejos, algunos de estos procesos son; Recocido Simulado, Algoritmos Genéticos, Búsqueda Tabú, Colonia de hormigas, entre otras. Estos "nuevos" modelos y algoritmos deben su éxito principalmente a la evolución de los sistemas informáticos, el crecimiento en el poder de cómputo y la baja en sus costos ha permitido una disminución en los tiempos de ejecución de los algoritmos [Sandoya 2007].

Recordemos que básicamente los problemas de tipo *VRP* se derivan de la variación del *m-TSP* por sus siglas en inglés (*multi-Travel Salessman Problem*), en el cual se ha asociado una restricción al número de nodos que el vendedor puede visitar, a la distancia máxima o mínima que puede recorrer o cualquier otra restricción donde *m* es generalmente desconocida y se refiere al número de vendedores que se utilizaran dependiendo de la restricción que se esté utilizando y se determina como una solución del problema [Sandoya 2007]. Lo antes mencionado se tratará más a fondo en las secciones siguientes.

1.1. FORMALIZACIÓN DEL PROBLEMA DE RUTEO VEHICULAR CAPACITADO (*VRP*)

En general, los problemas de tipo *VRP*, son una variación del problema *TSP*, es decir, el *m-TSP* es ya, básicamente un VRP, y las variaciones del *VRP* se dan gracias a los diferentes tipos de restricciones que podemos asignar al *VRP*, tales como, la capacidad de los vehículos, las ventanas de tiempo, el retorno de mercancías, etc.

Explicando lo anterior de manera más detallada tenemos que, el *m-TSP* se define como un conjunto dado de nodos, donde permitimos que existan m vendedores localizados en el nodo de depósito. Los nodos restantes (ciudades) que vayan a ser visitados, son nodos intermedios. Entonces, el *m-TSP* consiste en encontrar rutas para todos los m vendedores, quienes comienzan y terminan su recorrido en el depósito, de manera que cada nodo intermedio es visitado exactamente una vez y el costo total por visitar todos los nodos en minimizado. La medida del costo puede ser definida en términos de la distancia, tiempo, etc. Las restricciones pueden ser el número de nodos que cada vendedor puede visitar, la máxima o mínima distancia de viaje de los vendedores o cualquier otra restricción de este tipo [Davendra 2013].

Al respecto tenemos que el modelo de programación lineal entera según [Davendra 2013] es:

Minimizar

$$\sum_{i=1}^{n}\sum_{j=i}^{n} c_{ij}x_{ij} \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots (1.1)$$

Sujeto a

$$\sum_{j=2}^{n} x_{1j} = m \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots . (1.2)$$

$$\sum_{j=2}^{n} x_{j1} = m \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots (1.3)$$

$$\sum_{i=1}^{n} x_{ij} = 1, j = 2, \dots, n \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots (1.4)$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 2, \dots, n \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots (1.5)$$

+ las restricciones de eliminación de sub-*tours*……………………….(1.6)

$$x_{ij} \in \{0,1\}, \forall\ (i,j) \in A, \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots (1.7)$$

Donde 1.4, 1.5, 1.7 son restricciones de asignación habitual, 1.2 y 1.3 aseguran que exactamente m vendedores salgan y regresen al nodo 1 (depósito). Aunque la restricción 1.3 ya está implícita en 1.2, 1.4 y 1.5.

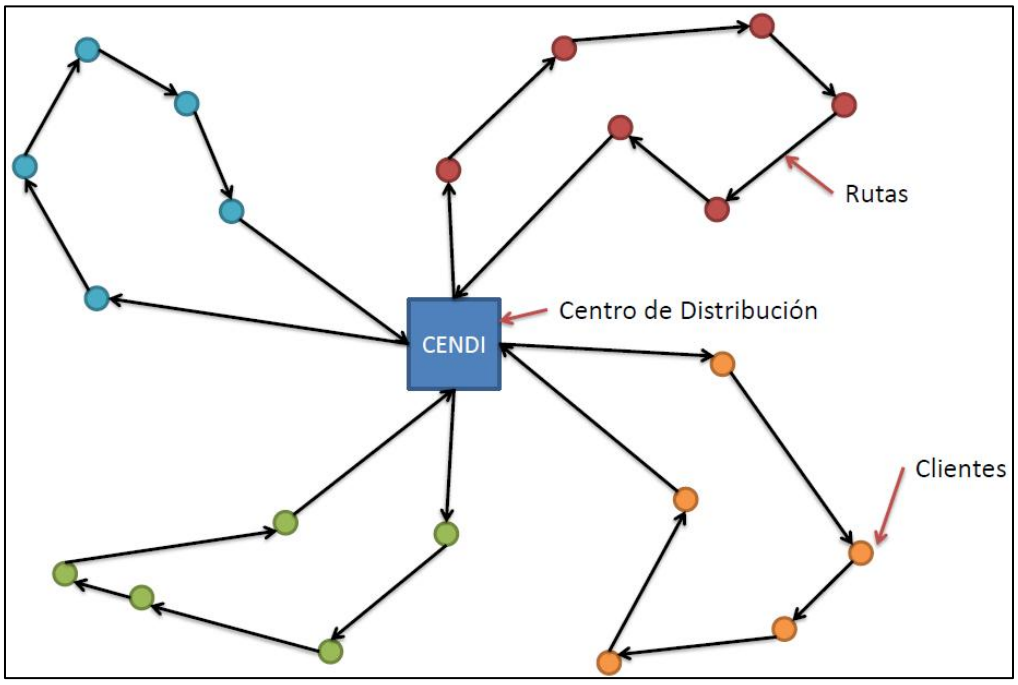
Por otro lado, para cada variación del *VRP*, se han propuesto modificaciones menores y se han analizado en literatura, a menudo diferentes problemas reciben el mismo nombre. Aunque en muchos casos los métodos de solución, particularmente algunas heurísticas, hayan sido adaptadas para agregar características adicionales. Esta indeterminación en la definición general del problema causa muchas confusiones. Por lo tanto, para este problema, lo primero que haremos será describir la versión básica de un *CVRP*.

El problema básico de un *CVRP* consiste en encontrar un conjunto de exactamente *k* circuitos simples (todos correspondes a la ruta de cada vehículo) al mínimo costo, se define como la suma del costo de los arcos pertenecientes a los circuitos definiendo la suma de los costos de los arcos, de tal manera que:

(i) Cada circuito visita el vértice de depósito;
(ii) Cada vértice cliente es visitado por exactamente un circuito; y
(iii) La suma de las demandas de los vértices visitados por el circuito no deben exceder la capacidad del vehículo, *C*.

Lo que acabamos de mencionar lo podemos observar de manera un poco más clara en la figura 1:

Figura 1. Esquema básico de un CVRP.



Fuente: elaboración propia.

1.2. CLASIFICACIÓN DEL PROBLEMA DE RUTEO VEHICULAR *(VRP)*

En los problemas de ruteo vehicular (*VRP*) del mundo real, existen muchas restricciones específicas de cada problema. Esto da como resultado el surgimiento de variaciones del problema original. Entre las principales variaciones tenemos:

- El problema del agente viajero (*Traveling Salesman Problem - TSP*);
- Vehículos con capacidad limitada (*Capacitated VRP - CVRP*);
- Clientes que tienen que ser atendidos dentro de una cierta ventana de tiempo (*VRP with time windows - VRPTW*);
- El vendedor utiliza diversos depósitos para abastecer a los clientes (*Multiple Depot VRP - MDVRP*);
- Los clientes tienen la opción de devolver algunos artículos al depósito (*VRP with Pick-Up and Delivering - VRPPD*);
- Los clientes pueden ser abastecidos por diferentes vehículos (*Split Delivery VRP - SDVRP*);
- Algunas características (como número de clientes, sus demandas, tiempo de servicio o tiempo de viaje) son aleatorios (*Stochastic VRP - SVRP*);
- Los pedidos son abastecidos sólo en ciertos días (*Periodic VRP - PVRP*).

Para mayores referencias acerca de los diferentes tipos de *VRP* podemos dirigirnos al escrito [Toth and Vigo. 2002] ya que se tratan a detalle cada una de las diferentes variantes.

1.3. FORMULACIÓN DEL PROBLEMA DE RUTEO VEHICULAR CAPACITADO (*CVRP*)

El modelo de programación lineal entera del *VRP* según Toth y Vigo [Toth and Vigo. 2002] se define de la siguiente manera:

Sea $G = (V, A)$ un grafo completo, donde $V = \{0, \ldots, n\}$ es el conjunto de vértices y A es el conjunto de arcos. Los vértices $i = 1, \ldots, n$ corresponden a los clientes, donde el vértice 0 corresponde al depósito.

Los costos, no negativos, c_{ij}, están asociados con el arco $(i, j) \in A$ y representa el costo de ir del vértice i al vértice j. Generalmente, el uso de bucles en los arcos, (i, j) no está permitido y esto se impone definiendo $c_{ii} = +\infty$ para todo $i \in V$. Si G es un grafo dirigido, la matriz de costos C es asimétrica, y el correspondiente problema es llamado *CVRP* asimétrico (*ACVRP*). Por otro lado si $c_{ij} = c_{ji}$ para todo $(i, j) \in A$, este problema es llamado *CVRP* simétrico (*SCVRP*), y el conjunto de arcos A es reemplazado por el conjunto no dirigido de aristas, E. Dada una arista $e \in E$, donde $\alpha(e)$ y $\beta(e)$ denotan los vértices en los puntos finales. Por otro lado *r(S)* es el número mínimo de vehículos necesarios para servir (cubrir) al conjunto *S.*

Modelo básico de programación lineal entera formulado por Toth y Vigo [Toth and Vigo. 2002] es:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \quad \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (2.1)$$

Sujeto a:

$$\sum_{i \in V} x_{ij} = 1 \; \forall \, j \in V \setminus \{0\}, \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . \ldots \ldots \ldots \ldots (2.2)$$

$$\sum_{j \in V} x_{ij} = 1 \; \forall \, i \in V \setminus \{0\}, \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . \ldots \ldots \ldots \ldots (2.3)$$

$$\sum_{i \in V} x_{i0} = K, \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . \ldots . \ldots \ldots \ldots \ldots . (2.4)$$

$$\sum_{j \in V} x_{0j} = K, \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . . \ldots . \ldots \ldots \ldots . . (2.5)$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S) \quad \forall \, S \subseteq V \setminus \{0\}, S \neq \emptyset, \ldots \ldots . \ldots . . \ldots \ldots . (2.6)$$

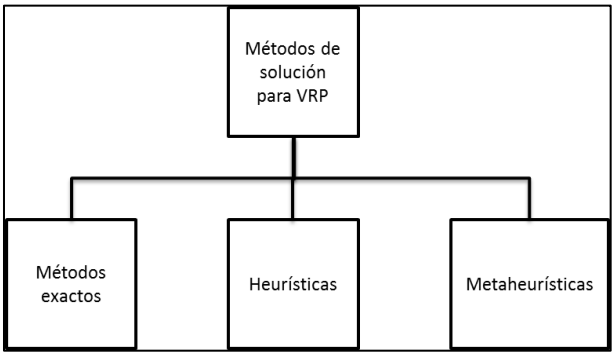
$$x_{ij} \in \{0,1\} \; \forall \, i, j \in V \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . . \ldots \ldots \ldots \ldots \ldots . . (2.7)$$

Dónde: 2.1 Corresponde a la definición de la función objetivo, 2.2 y 2.3 imponen que exactamente un arco entre y salga de cada vértice asociado con un cliente, respectivamente, 2.4 y 2.5 análogamente imponen los requisitos de grado del vértice de depósito y 2.6 también llamadas, limitaciones de capacidad de corte, imponen tanto la conectividad de la solución, como los requisitos de capacidad del vehículo.

1.4. MÉTODOS DE SOLUCIÓN DEL PROBLEMA DE RUTEO VEHICULAR

En lo que respecta a los métodos de solución, se han abordado tres grandes categorías, las cuales pueden ser agrupadas de la siguiente manera: métodos exactos, heurísticas y metaheurísticas. La figura 2 muestra dicha clasificación, a partir de la cual se desprenden las demás clasificaciones que se explicarán a continuación [Rocha, L. *et al.* 2011].

Figura 2. Clasificación según métodos de solución.

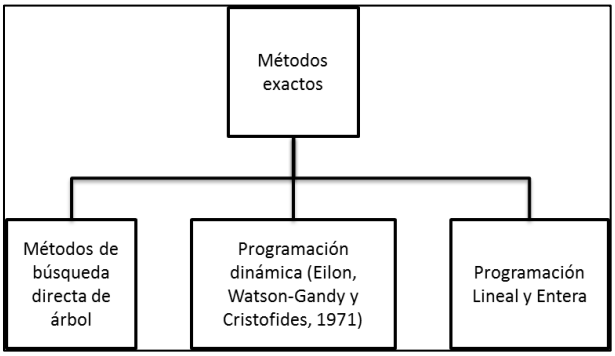


Fuente: elaboración propia, basado en: Rocha, L. et al. 2011

Métodos Exactos

Los métodos exactos son eficientes en problemas hasta 50 depósitos debido a restricciones de tiempo computacional. Los métodos exactos se pueden clasificar en tres grupos: búsqueda directa de árbol, programación dinámica, programación lineal y entera. En la figura 3 se muestra esta clasificación [Rocha, L. *et al.* 2011].

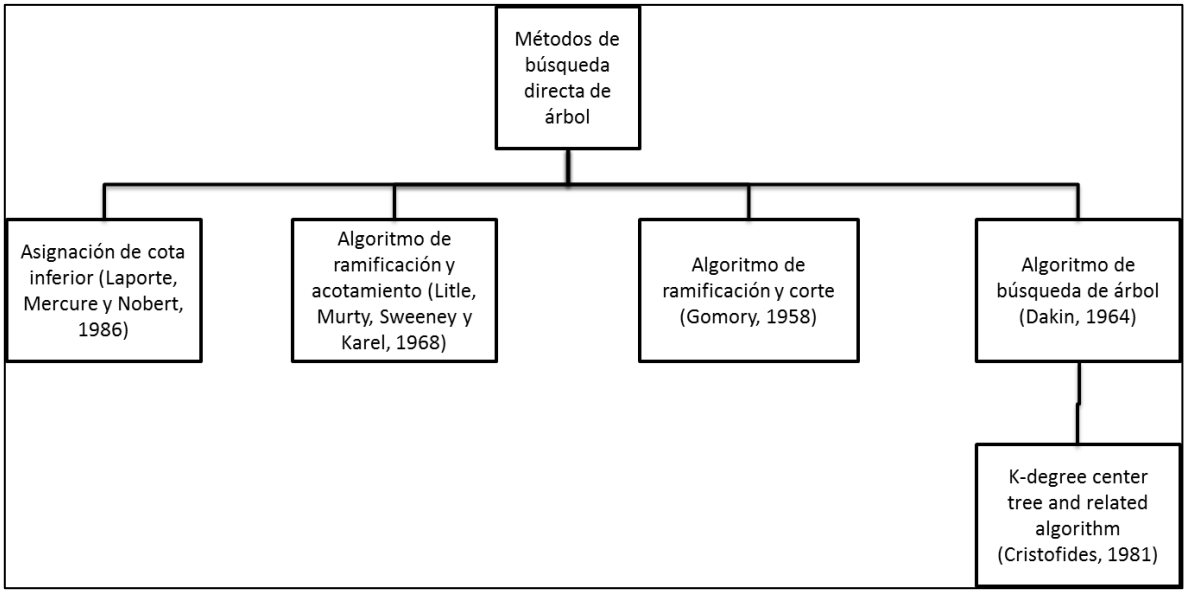
Figura 3. Clasificación según métodos de solución.



Fuente: elaboración propia, basado en: Rocha, L. et al. 2011

Métodos de Búsqueda Directa de Árbol: La búsqueda se realiza sobre todos los nodos de un árbol de acuerdo a criterios específicos propios de cada método. La figura 4 muestra la clasificación establecida a partir de Gilbert Laporte [Laporte 1991]. Una descripción breve se presenta a continuación [Rocha, L. *et al.* 2011].

Figura 4. Clasificación para los métodos de solución mediante búsquedas en árbol.



Fuente: elaboración propia, basado en: Rocha, L. et al. 2011

El algoritmo de asignación de cota inferior asigna una cota inferior que permite disminuir el número de vehículos requeridos para visitar todos los vértices. Esto se realiza por medio del *m-TSP*, como relajación del *VRP*, proporcionando una cota superior para el número de vehículos y transformándolo en un *TSP* [Rocha, L. *et al.* 2011].

Por su parte, el algoritmo de ramificación y acotamiento consiste en recorrer cada nodo del árbol desde el nivel superior hacia la base del árbol y los nodos terminales resolviendo en cada nodo un programa lineal y determina que nodos pueden eliminarse. Un nodo se elimina (junto con sus descendientes) si no existe una solución factible; pero si existe solución factible se convierte en una cota inferior. El algoritmo termina cuando todos los nodos han sido revisados y la solución óptima es la de mayor cota inferior [Rocha, L. *et al.* 2011].

El árbol del centro de *k-grados* (*k-degree center tree algorithm*) trabaja con un número fijo de vehículos, una solución factible en el conjunto de aristas se divide en cuatro subconjuntos que son: las aristas que no pertenecen a la solución, las aristas que forman el árbol, las aristas que inciden en el primer vértice y las aristas que no inciden en el primer vértice. Estos subconjuntos se traducen en restricciones en el modelo y la solución objetivo consiste en sumar el costo de todas las aristas en la solución [Rocha, L. *et al.* 2011].

Programación Dinámica: Propuesto por Eilon, Watson-Gandy y Christofides en 1971. En el método se considera un número fijo de m vehículos. Encuentra primero el costo mínimo alcanzable utilizando k vehículos, teniendo en cuenta la función del costo en la longitud de una ruta de vehículos a través de todos los vértices del subconjunto, luego encuentra el

costo de todos los subconjuntos de vértices con m vehículos [Laporte 1991] [Rocha, L. *et al.* 2011].

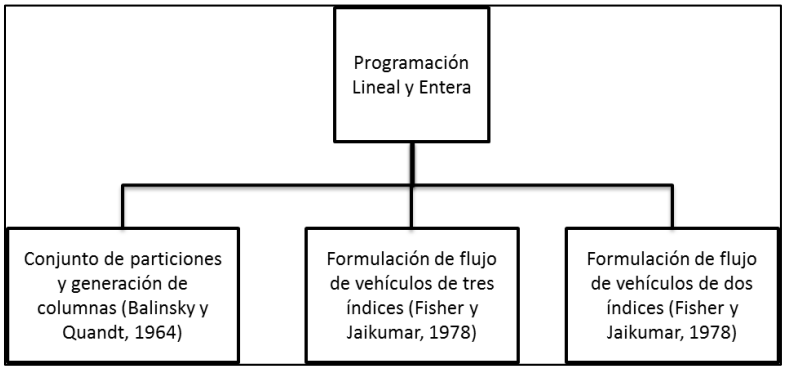
Programación Lineal y Entera: En la figura 5 se muestran las tres técnicas comprendidas dentro de esta clasificación.

El método de partición y generación de columnas se considera un conjunto factible de rutas y un coeficiente binario que es igual a uno sí y sólo sí determinado depósito pertenece a una ruta. También se tiene en cuenta el costo óptimo de una ruta y una variable binaria que es igual a uno sí y sólo sí esa ruta es utilizada en la solución óptima [Rocha, L. *et al.* 2011].

El valor de este costo se obtiene resolviendo un *TSP*. Por su parte el método de flujo de vehículos de 2 índices y 3 índices fue desarrollado para *CVRPTW*. En la formulación de 2 índices x_{ij} representa el camino que une el depósito i con el depósito j. En la formulación de 3 índices la variable x_{ijk} indica el camino que une el depósito i con el depósito j, utilizando el vehículo k. El algoritmo desarrollado está basado en una formulación que garantiza la solución óptima en un número finito de pasos, si se ejecuta hasta finalizarlo [Rocha, L. *et al.* 2011].

La formulación no exige que los vehículos sean idénticos.

Figura 5. Clasificación para los métodos de solución de programación Lineal y Entera.



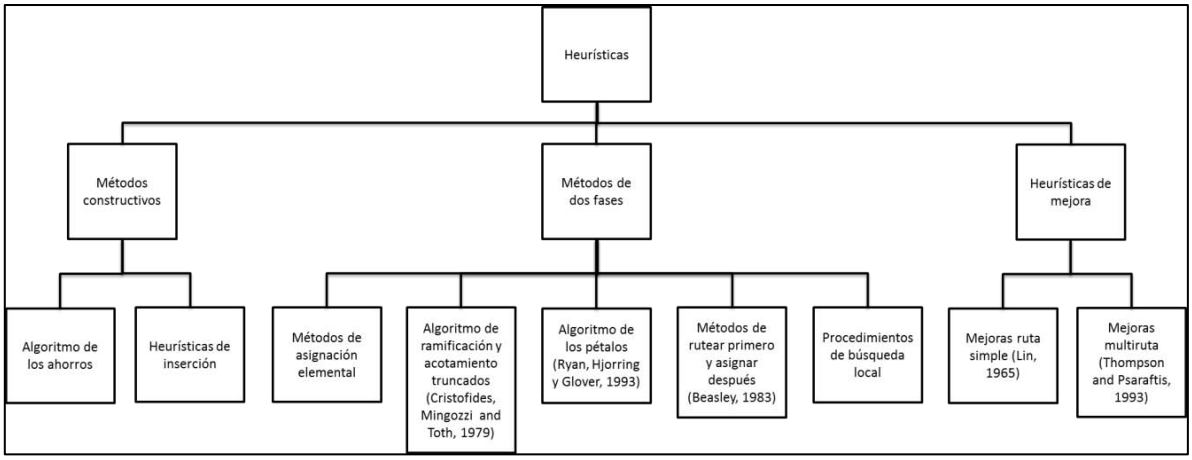
Fuente: elaboración propia, basado en: Rocha, L. et al. 2011

Heurísticas

Las heurísticas son procedimientos que proporcionan soluciones de aceptable calidad mediante una exploración limitada del espacio de búsqueda. Clarke y Wright, propusieron el primer algoritmo que resultó efectivo para resolver el *VRP* en 1964. La mayoría de las heurísticas clásicas para resolver el *VRP* fueron desarrolladas entre 1960 y 1990 [Rocha, L. *et al.* 2011].

Los métodos heurísticos se han clasificado en métodos constructivos, métodos de dos fases y heurísticas de mejora, como se observa en la figura 6.

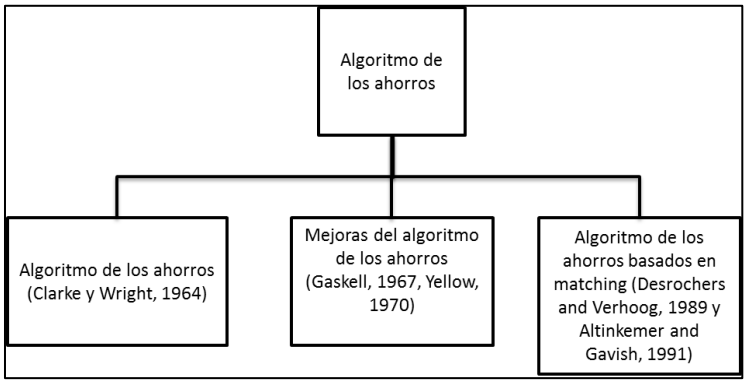
Figura 6. Clasificación para los métodos de solución Heurísticos.



Fuente: elaboración propia, basado en: Rocha, L. et al. 2011.

Métodos Constructivos: En los métodos constructivos se encuentran los algoritmos de los ahorros y las heurísticas de inserción, los cuales son ampliamente conocidos dentro del campo de la Investigación de Operaciones. Los algoritmos de los ahorros (*Savings Algorithms*) comprenden el algoritmo de los ahorros de Clarke y Wright, el algoritmo de los ahorros mejorado y el algoritmo de los ahorros basado en coincidencia. En la figura 7 se muestra la clasificación establecida a partir de los algoritmos de ahorros [Rocha, L. *et al.* 2011].

Figura 7. Clasificación para los métodos de solución mediante algoritmos de ahorros.



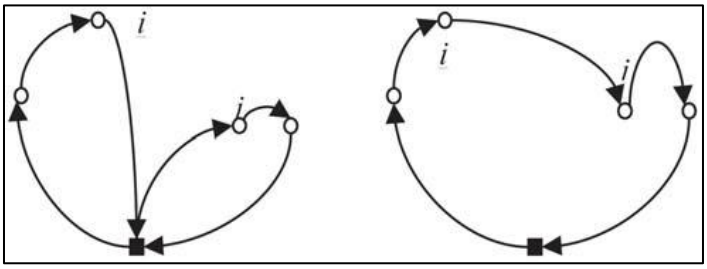
Fuente: elaboración propia, basado en: Rocha, L. et al. 2011

El algoritmo de los ahorros de Clarke y Wright se aplica generalmente a problemas para los cuales el número de vehículos es una variable de decisión, calcula el mayor ahorro en distancia, al utilizar los arcos. Si en una solución se encuentran dos rutas diferentes y estas dos rutas pueden ser combinadas para obtener una nueva en la cual se encuentre mayor

ahorro en sus arcos, entonces se utilizará esta nueva ruta. En la figura 8 se muestra la representación en forma de grafo para esta heurística [Rocha, L. *et al.* 2011].

El algoritmo original de Clarke y Wright produce buenas rutas al inicio pero no hacia el final, pues incluye algunas rutas circulares. El algoritmo de Clarke y Wright mejorado propuesto por Laporte, Toth y Vigo generalizaron los ahorros mediante un parámetro llamado *Shape Parameter* o Parámetro de Forma que penaliza la unión de rutas con clientes lejanos [Rocha, L. *et al.* 2011].

Figura 8. Heurística del algoritmo de los ahorros de Clarke y Wright.



Fuente: tomado de Rocha, L. et al. 2011.

Por su parte los algoritmos de los ahorros basados en coincidencia (en un grafo significa un conjunto de arcos que no tienen extremos en común y el peso de una coincidencia (*matching*) es la suma de los pesos de sus arcos) (*Matching-Based Savings Algorithms*) son una modificación del algoritmo de ahorros estándar y plantean realizar la unión de dos rutas teniendo en cuenta las posibles uniones subsiguientes [Rocha, L. *et al.* 2011].

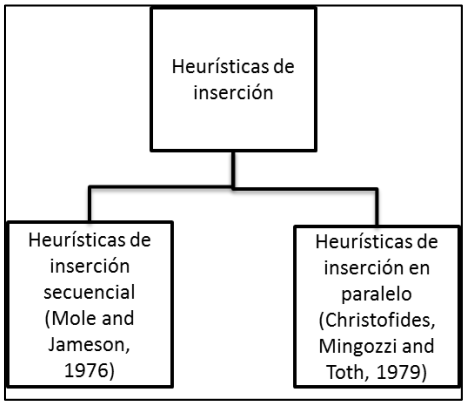
En lo que concierne a las heurísticas de inserción, se encuentran dos algoritmos de dos fases cada uno, que aplican a problemas con un número de vehículos no específico. En la figura 9 se muestra una clasificación de dichas heurísticas. Estas heurísticas crean soluciones mediante sucesivas inserciones de clientes en las rutas, es decir, en cada iteración se tiene una solución parcial cuyas rutas sólo visitan un subconjunto de los clientes y luego se selecciona un cliente no visitado para insertarlo en la última ruta creada [Rocha, L. *et al.* 2011].

La heurística de inserción secuencial de Mole & Jameson utiliza parámetros para expandir una ruta en construcción. Para insertar un cliente se utilizan dos medidas; la primera medida es el costo de insertar el cliente no visitado en la ruta. Esta medida se utiliza para determinar la mejor posición de cada cliente no visitado, así mismo, se calcula teniendo en cuenta únicamente las distancias sin reordenar los nodos que ya están en la ruta [Rocha, L. *et al.* 2011].

La heurística de inserción en paralelo de Christofides, Mingozzi & Toth es una heurística de inserción de dos fases que utiliza dos parámetros controlados λ y μ. En la primera fase

se determina la cantidad de rutas a utilizar aplicando la primera fase del algoritmo de Mole & Jameson para obtener rutas compactas y conservar los clientes iniciales de cada ruta junto con la cantidad de rutas de la solución final. En la segunda fase se crean las rutas y se inserta el resto de los clientes en ellas [Rocha, L. *et al.* 2011].

Figura 9. Clasificación para los métodos de solución con heurísticas de inserción.

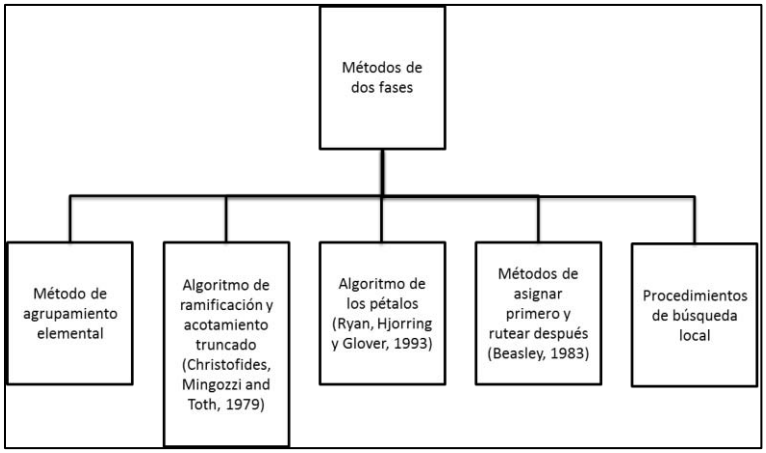


Fuente: elaboración propia, basado en: Rocha, L. et al. 2011

Métodos de dos fases: En los métodos de dos fases se encuentran los métodos de asignación elemental, el algoritmo de ramificación y acotamiento truncados, el algoritmo de los pétalos, el método de asignar primero y rutear después y viceversa y los procedimientos de búsqueda local. En la figura 10 se muestra la clasificación establecida a partir de los autores Paolo Toth y Daniele Vigo [Toth and Vigo. 2002], Alfredo Olivera [Olivera 2004], Gilbert Laporte [Laporte 1991]. Seguidamente se presenta una descripción de cada uno [Rocha, L. *et al.* 2011].

Métodos de Agrupamiento Elemental (Elementary Clustering Methods): En este tipo de métodos se encuentran el algoritmo de barrido (*sweep algorithm*), el algoritmo basado en asignación generalizada (*generalizad-assignment-based algorithm*) y la heurística basada en localización (*location based heuristic*). El algoritmo de Barrido (*sweep algorithm*) consiste en formar inicialmente agrupamientos girando una semirrecta con origen en el depósito e incorporando los clientes hasta violar la restricción de capacidad. Una ruta de vehículos es obtenida para el *cluster* resolviendo un *TSP*. En algunos casos de implementación es necesaria una fase de post-optimización en la cual los vértices se intercambian entre *clusters* adyacentes y las rutas son re-optimizadas [Rocha, L. *et al.* 2011].

Figura 10. Clasificación para los métodos de solución de dos fases.



Fuente: elaboración propia, basado en: Rocha, L. et al. 2011

Por su parte el algoritmo basado en asignación generalizada (*generalizad-assignment-based algorithm*) en lugar de utilizar un método geométrico para formar *clusters* utiliza un Problema de Asignación Generalizada (*GAP*). La primera fase de este algoritmo consiste en escoger los vértices semilla para construir los agrupamientos. En la segunda fase se asignan los vértices a cada agrupamiento sin violar la capacidad del vehículo resolviendo un *GAP* [Rocha, L. *et al.* 2011].

Por último en la heurística basada en localización (*location based heuristic*) las rutas iniciales (semillas) son establecidas como un problema de localización con capacidades y los vértices restantes son incluidos gradualmente en la ruta asignada en una segunda etapa. Para satisfacer las restricciones de capacidad y minimización de los costos se debe decidir sobre las semillas a colocar y los terminales a los cuales se van a conectar cada semilla. Las rutas de los vehículos se construyen entonces, insertando en cada paso, el cliente asignado para que las semillas tengan el menor costo de inserción [Rocha, L. *et al.* 2011].

Ramificación y Acotamiento Truncados (Truncated Branch and Bound). En este algoritmo el árbol de búsqueda tiene tantos niveles como rutas de vehículos y cada nivel contiene un conjunto de rutas de vehículos, para ello Christofides, Mingozzi y Toth proponen una implementación en la cual seleccionan una rama en cada nivel y la otra se descarta bajo algún criterio de selección. Se puede construir un árbol limitado manteniendo pocas rutas en cada nivel [Rocha, L. *et al.* 2011].

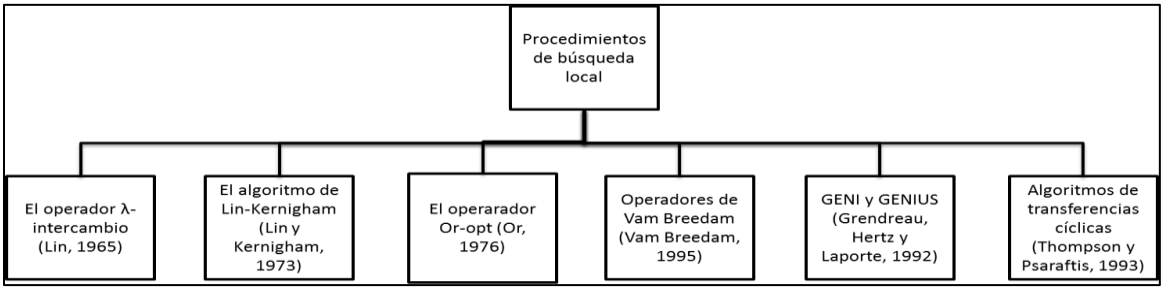
Algoritmos de los Pétalos (Petal Algorithms). Este algoritmo es una extensión del algoritmo de barrido y se utiliza para generar varias rutas llamadas pétalos con el fin de hacer una selección final resolviendo un *Set Partitioning Problem*. Se dispone de un conjunto de rutas R en la que cada cliente es visitado por varias rutas y se debe seleccionar un subconjunto de R que visite exactamente una vez cada cliente [Rocha, L. *et al.* 2011].

Métodos de Ruteo Primero y Asignación Después (Route-First, Cluster-Second Methods). Este método consta de dos fases. En la primera fase se calcula una gran ruta que visita a todos los clientes resolviendo un *TSP* sin tener en cuenta las restricciones del problema. Luego en la segunda fase, esta ruta gigante se descompone en varias rutas factibles, es decir, teniendo en cuenta la solución de la primera fase, se determina la mejor partición teniendo en cuenta la capacidad del vehículo [Rocha, L. *et al.* 2011].

Método de asignar primero, rutear después (cluster-first, route-second method): Los métodos asignar primero y rutear después proceden en dos fases. Primero se busca generar grupos de clientes, también llamados *clusters*, que estarán en una misma ruta en la solución final. Luego, para cada *cluster* se crea una ruta que visite a todos sus clientes. Las restricciones de capacidad son consideradas en la primera etapa, asegurando que la demanda total de cada *cluster* no supere la capacidad del vehículo. Por lo tanto, construir las rutas para cada *cluster* es un TSP que, dependiendo de la cantidad de clientes en el *cluster*, se puede resolver de forma exacta o aproximada. Cabe mencionar que este método en particular, es el que utilizaremos en el desarrollo de esta tesis como parte importante del híbrido que vamos a utilizar para resolver las instancias propuestas, como lo veremos en capítuos posteriores.

Procedimientos de Búsqueda Local (*Local Search Procedures*). Los procedimientos de búsqueda local se aplican para mejorar una solución ya obtenida. En estos procedimientos se define un conjunto de soluciones vecinas y parte de una solución primaria para luego remplazarla por una solución vecina con menor costo. El procedimiento se repite hasta que no se pueda mejorar la solución. Los diferentes procedimientos de búsqueda local se pueden observar en la figura 11, establecida a partir de los autores Paolo Toth y Daniele Vigo [Toth and Vigo. 2002], Alfredo Olivera [Olivera 2004], Gilbert Laporte [Laporte 1991] [Rocha, L. *et al.* 2011].

Figura 11. Clasificación para los métodos de solución con búsqueda local.

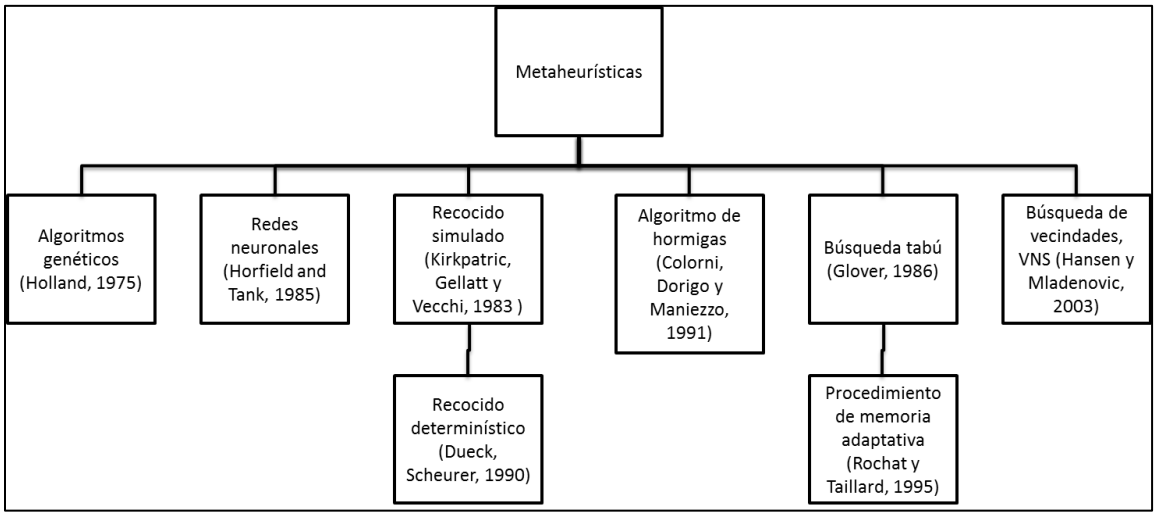


Fuente: elaboración propia, basado en: Rocha, L. et al. 2011

Metaheurísticas

Las metaheurísticas fueron desarrolladas hacia la década de los 90 y se caracterizan por que realizan un procedimiento de búsqueda para encontrar soluciones de aceptable calidad, mediante la aplicación de operadores independientes del dominio que modifican soluciones intermedias guiadas por la idoneidad de su función objetivo. Dentro de estas se encuentran el Recocido Simulado, Redes Neuronales, Búsqueda Tabú, Algoritmos Genéticos, Algoritmos de Hormigas y Búsqueda de vecindades. En la figura 12 se muestra una clasificación de estas metaheurísticas [Rocha, L. *et al.* 2011].

Figura 12. Clasificación para los métodos de solución metaheurísticos.



Fuente: elaboración propia, basado en: Rocha, L. et al. 2011

Algoritmos Genéticos (*Genetic Algorithms*). Inspirado en la teoría de la evolución darwiniana, este algoritmo parte de una población inicial de individuos que representan soluciones iniciales factibles pero subóptimas. Seguidamente el algoritmo evoluciona mediante la aplicación de operadores evolutivos que combinan y modifican a los individuos de la población creando una nueva. Para cada individuo se define una función de aptitud *f(i)* que califica su idoneidad. Usualmente, se trabajan tres operadores: selección, cruzamiento y mutación [Rocha, L. *et al.* 2011].

La forma de operar de estos algoritmos para la solución del *VRP* se resume de la siguiente forma. Se generan soluciones iniciales, las cuales representan cada viaje como una secuencia de ciudades (a diferencia de los algoritmos genéticos tradicionales que utilizan una representación de dígitos binarios). Para cruzar dos soluciones, se toma una subruta que no necesariamente cumpla que inicie y termine en el depósito, y se determina el cliente más cercano que no esté en la subruta. Si la ruta no fuera factible, se particiona. De ésta manera se genera un descendiente, es decir, una copia modificada de una de las soluciones iniciales. Usualmente para este tipo de problemas, se consideran cuatro operadores de mutación: intercambio de la posición de dos nodos en una ruta; inversión

del orden de la ruta; reinserción de un nodo en una ruta diferente a la original y selección de una subruta para insertarla en otro lugar de la solución [Rocha, L. *et al.* 2011].

Algoritmos de Hormigas (*Ant Algorithms*). Estos algoritmos están inspirados en la estrategia que usan las colonias de hormigas en la búsqueda de alimentos. Cuando una hormiga encuentra el camino para ir a la fuente de alimento deposita una sustancia (feromona) que depende de la longitud del camino y la calidad del alimento. Las hormigas tienden a seguir los trayectos con mayor cantidad de feromonas puesto que es más probable que conduzcan más rápido hacia la fuente de alimento, lo que a su vez provoca un refuerzo de los mejores trayectos, es decir, los que demoren menos tiempo y por donde transiten la mayor cantidad de hormigas [Rocha, L. *et al.* 2011].

En el caso de los *VRP*, el modo de funcionamiento de estos algoritmos se resume así: Se inicializa el algoritmo colocando una hormiga en cada nodo. Para la construcción de caminos, se utiliza una regla probabilística que asigna una probabilidad igual a cero si el nodo ya fue visitado y diferente a cero para el caso contrario. La hormiga visita el nodo que tenga una probabilidad mayor. En cada arco, se actualiza la "feromona" y finaliza si se obtiene una solución inferior a una cota preestablecida, de lo contrario se recalculan probabilidades y la hormiga sigue construyendo soluciones [Rocha, L. *et al.* 2011].

Búsqueda Tabú (*Tabu Search*). Consiste en realizar una búsqueda local aceptando soluciones que mejoran el comportamiento del costo de tal manera que en cada iteración el algoritmo se mueve de una solución *(st)* a otra mejor *(st+1)* dentro de un subconjunto de soluciones cercanas. Como *st+1* no necesariamente es el menor costo, se utiliza una memoria de corto plazo que registre algunos atributos de soluciones ya visitadas. Estas soluciones prohibidas se llaman soluciones tabú y los movimientos que llevan a esas soluciones se llaman movimientos tabú. En algunos casos es necesario aceptar soluciones tabú porque poseen mejores atributos que las demás y para esto se utiliza un criterio llamado criterio de aspiración; el criterio también se usa para aceptar soluciones que no son tabú. A estas soluciones por las cuales pasa el criterio de aspiración se llaman soluciones admisibles y la búsqueda se realiza sobre las soluciones admisibles de la vecindad [Rocha, L. *et al.* 2011].

El procedimiento Tabú para un problema de ruteo debe responder a los seis criterios siguientes [Rocha, L. *et al.* 2011]:

• *Algoritmo de búsqueda local:* Se genera una solución inicial de prueba, la cual puede ser cualquier secuencia de nodos, se inician las iteraciones seleccionando el mejor vecino inmediato que no esté descartado de la lista Tabú.

• *Estructura de vecindad*: Se generan dos arcos (que unan dos nodos) y se eliminan dos de la solución actual, debe tenerse cuidado de descartar sub-viajes que solamente inviertan la dirección de la ruta.

• *Forma de los movimientos Tabú*: Enumerar los arcos de tal manera que un sub-viaje inverso se convierta en tabú si los dos arcos que se eliminan se encuentren en la lista.

• *Adición de un movimiento Tabú*: En cada iteración del algoritmo, después de incluir dos arcos a la solución actual, también se incorporan estos dos arcos a la lista tabú.

• *Tamaño máximo de la lista Tabú*: Se debe generar un criterio bajo el cual un par de arcos se inserte a la lista y salgan los que llevan más tiempo en ella.

• *Regla de detención*: Criterio para detener el proceso puede ser después de un número consecutivo de iteraciones, donde no se produzca mejoras en la solución.

Otras metaheurísticas utilizadas en el problema *VRP* son Memoria Adaptativa (*Adaptative Memory*) que es una mejora de la búsqueda tabú propuesta por Rochat y Taillard en 1995. Construye buenas soluciones mediante la combinación de otras buenas soluciones. Una memoria contiene los componentes de las soluciones visitadas y periódicamente se construye una nueva utilizando datos en la memoria y se mejora mediante un procedimiento de búsqueda local, la mejor solución es utilizada para actualizar la memoria. Redes Neuronales (*Neural Networks*) que consiste en un modelo computacional compuesto de unidades interconectadas a través de conexiones fuertes, parecidas a las neuronas del cerebro humano. Se envía una señal desde una unidad a otra mediante una conexión y se modula a través del peso asociado. Recocido Simulado (*Simulated Annealing*) es un método de búsqueda local aleatorio, en el cual una modificación a la solución actual que conduzca a un incremento en el costo solución puede ser aceptado. Por último tenemos Recocido Determinístico (*Deterministic Annealing*) que funciona de una manera similar al recocido simulado, salvo que utiliza una regla determinística para aceptar un movimiento [Rocha, L. *et al.* 2011].

1.5. COMPLEJIDAD COMPUTACIONAL DEL PROBLEMA DE RUTEO VEHICULAR CAPACITADO *(CVRP)*

Para comenzar diremos que un algoritmo es un método para resolver una instancia específica de un problema. La eficiencia de un algoritmo se mide de acuerdo al tiempo total de cómputo que consume la resolución del algoritmo o a la cantidad de memoria utilizada. Importa en ello el tamaño de la instancia medida en número de operaciones a través de una función de complejidad. En función de lo anterior se definen las siguientes clases de problemas [ELIZONDO 2014]:

- Clase *P*: estos problemas de decisión son relativamente sencillos y se resuelven mediante algoritmos en tiempo polinomial, para ello existen algoritmos eficientes.
- Clase *NP*: Problemas Polinomiales no deterministas[4] que incluyen todos los problemas "razonables" de importancia teórica y práctica. No es necesario resolver el problema: es suficiente con mostrar que tal verificación de la solución existe. Un problema de decisión de *A* esta en la clase *NP* si existe un polinomio *p(n)* y un algoritmo *a* para certificar y revisar la existencia de un algoritmo de solución.

  x es una instancia "si" de $A \Leftrightarrow \exists$ una secuencia de símbolos en $\sum$. la certificación $c(x)$, $|c(x)| \leq p(|x|)$considerando que si se provee la entrada $x\$c(x)$ al algoritmo a este alcanza una respuesta "sí" después de a lo más de $p(|x|)$ pasos.

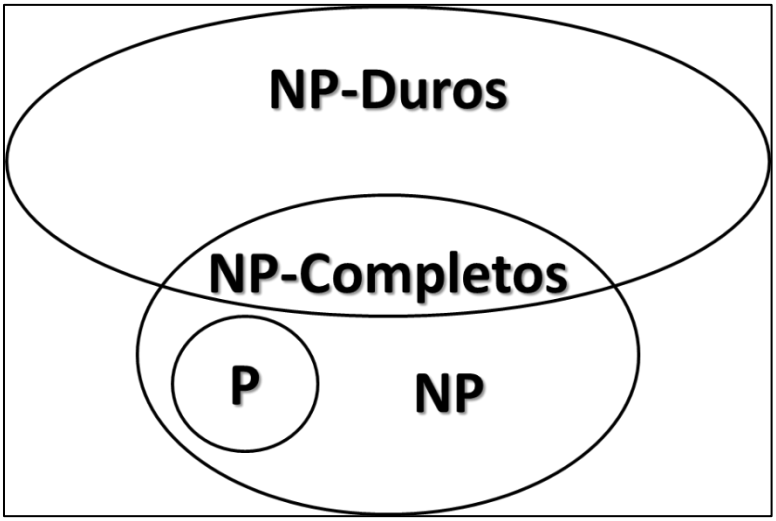
  Los problemas *P* pertenecen al conjunto de problemas *NP*.

- Clase *NP-Duros*: un problema de decisión *A* pertenece a esta categoría si para todo problema *B* que sea *NP*, existe un algoritmo que funciona en tiempo polinomial, capaz de reinterpretar toda instancia del problema *A* a una instancia del problema *B*. Es la categoría de todos los problemas que son al menos tan difíciles como cualquier otro de la Clase *NP*.
- Clase *NP-Completo*:
  1. No pueden resolverse con ningún algoritmo polinomial conocido.
  2. Si existe un algoritmo polinomial (eficiente) para cualquier problema *NP-Completo* y por tanto existen algoritmos polinomiales para todos los problemas en *NP*.

  Los problemas *NP-Completo* son la intersección de *NP* con *NP-Duros*.

[4] Término acuñado por Stephen Arthur Cook.

Figura 13. Clases básicas de complejidad.



Fuente: elaboración propia, basado en Denef et al.2007

Resumiendo, los problemas de la clase *P*, son aquellos que se resuelven fácilmente y para los que existe un algoritmo determinista que se resuelve en tiempo polinomial. Sin embargo, los problemas de ruteo vehicular caen en la clasificación de *NP-Duros* como lo mencionan Toth y Vigo [Toth and Vigo. 2002], para mayor referencia dirigirse a [Toth and Vigo. 2002], aquellos para los que se cree que no existe algoritmo determinista que lo pueda resolver en tiempo polinomial.

Se "cree" que estos problemas son intratables computacionalmente; significa que no son susceptibles al uso de algoritmos de solución eficientes, y en el peor de los casos, se requeriría una cantidad de tiempo exponencial para resolverlo, impráctico para todos los problemas, salvo instancias muy pequeñas.

De lo anterior se deriva la importancia de seguir desarrollando herramientas que nos permitan resolver o al menos dar "buenas" soluciones a problemas de este tipo en tiempos razonables, y a su vez es una parte que fundamenta este trabajo de tesis ya que lo que pretendemos es hacer una aportación en el campo de la simulación, mediante la hibridación de una heurística combinada con un método de dos fases mediante *ABS.*

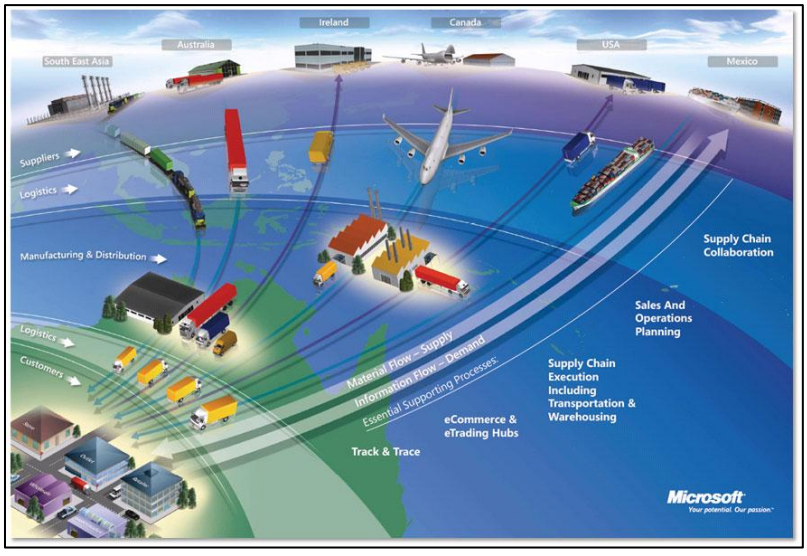
1.6. APLICACIONES DEL PROBLEMA DE RUTEO VEHICULAR CAPACITADO (*CVRP*)

La importancia de encontrar herramientas que nos ayuden a resolver problemas de ruteo de manera más eficiente y a menor costo, radica, entre otras cosas, en la aplicación que estas herramientas pueden tener en la vida real, por citar un ejemplo; la producción en la rama transporte en México según el Instituto Mexicano del Transporte (IMT) es una de las actividades más importantes dentro de la rama económica y ocupó el sexto lugar en importancia en el 2013 [IMT 2014].

Por otro lado, según el Manual Estadístico del Transporte 2013 cita "La posición de ésta actividad (Transporte) en la generación de Valor Agregado Bruto, se había mantenido a lo largo del período con montos del 7% de participación con respecto al total de cada año. La interpretación de los resultados sugiere el relevante papel del transporte dentro de la economía nacional, pero también puede significar que existen posibles ineficiencias en el sector, por ejemplo, que los precios del servicio son demasiado elevados, o que en el traslado de las mercancías se efectúan recorridos innecesarios o demasiado extensos" [IMT 2014].

No sólo en México, sino a nivel mundial, y en todos los niveles de la industria y el comercio, observamos que en la cadena de suministro es indispensable el transporte de mercancía, materia prima, producto terminado, etc. De lo anterior nace la inquietud de seguir desarrollando y analizando técnicas y herramientas que nos ayuden a resolver los problemas de distribución de manera eficaz y a bajo costo.

Ilustración 1. Cadena de suministro.



Fuente: Parag Khanna[5]

[5] Disponible en: http://paragkhanna.com/page/5/

TENDENCIAS

En la década de los 90´s se presenta una variación de *VRP* que se refiere a la idea de *VRP* con múltiples usos de vehículos (*VRP multiple use of vehicles*), basada en el supuesto de que un vehículo puede hacer más de un viaje en un periodo de planeación. Esta idea fue introducida por Fleischmann en 1990. En 1997 Brandao y Mercer lo trabajaron como *VRP* con viajes múltiples (*VRP with multiple trips*) resolviéndolo mediante una Búsqueda Tabú [Rocha, L. *et al.* 2011].

En 2002 Prins [Prins 2002] introdujo el *VRP* con viajes múltiples de flota heterogénea mediante un caso real a gran escala, luego en 2008 aparece el *VRP* con viajes múltiples periódico y el *VRP* con viajes múltiples independientes del sitio (*site-dependent*). El último trabajo encontrado sobre *Multi–trip VRP* es el *VRPTW* con múltiples usos de vehículos, resuelto mediante métodos exactos [Rocha, L. *et al.* 2011].

Los métodos de solución utilizados para resolver estas variaciones recientes del *VRP* han sido en su mayoría metaheurísticas tales como la Búsqueda Tabú y los Algoritmos Genéticos, pero se han empleado otros métodos como el algoritmo de memoria adaptativa, algoritmo genético híbrido, búsqueda de vecindades, algoritmo de ramificación y valor, estrategia de guía auto-adaptativa, heurística multifase y heurística basada en ahorros [Rocha, L. *et al.* 2011].

Entre los años 1999 y 2009, a partir del Problema *VRPPD* se desarrollaron variaciones de dicho problema de tal manera que se pueden considerar dos grandes clasificaciones: la primera son problemas estáticos donde los datos del problema son conocidos antes de construir las rutas y la segunda son problemas dinámicos en los cuales algunos datos sólo son conocidos durante el periodo de tiempo de operación, además que el horizonte de planeación puede ser no acotado [Rocha, L. *et al.* 2011].

A partir del año 2002 y hasta el 2011, se desarrollaron métodos exactos de solución aplicables al *CVRP* y al *VRPTW* basados principalmente en dos técnicas: la primera es la formulación de algoritmos de partición de conjuntos que permiten incorporar restricciones adicionales que pueden ser aplicables en la modelación de situaciones específicas en la industria y la segunda son algoritmos basados en la generación de columnas que han sido derivados de pequeñas modificaciones de los algoritmos originales de generación de columnas para solucionar el *VRP* [Rocha, L. *et al.* 2011].

Los métodos de solución para el *SVRP* pueden ser aplicables a problemas que no tengan parámetros estocásticos pero con una estructura similar. En las últimas revisiones (2005-2010) se presenta un especial interés por los tiempos de viaje y de servicio aleatorios [Rocha, L. *et al.* 2011].

Es importante resaltar que en Colombia se ha presentado interés en los últimos años para el estudio de los problemas *VRP*, en especial en aplicaciones concretas, lo anterior se evidencia en la participación con ponencias en congresos internacionales, tal es el caso de lo presentado en el XVI congreso Latinoamericano de Investigación de operaciones (XVI CLAIO/ XVIV SPBO) [Rocha, L. *et al.* 2011].

1.7. PLANTEAMIENTO DEL PROBLEMA

En resumen, hemos visto la importancia que suponen los problemas de tipo *VRP* en la vida diaria de las industrias, ya que el transporte de mercancías y/o servicios es vital en el desarrollo de éstas. Realizando un análisis profundo de lo que acabamos de reseñar, podemos llegar a la conclusión de que, dada la importancia de los problemas de transporte de tipo *CVRP* y a su carácter de tipo NP-Duro, nos lleva a seguir desarrollando herramientas y métodos que nos ayuden a resolver o al menos a dar buenas soluciones a estos problemas en tiempos razonables y de manera más eficiente. Además de lo que la literatura consultada nos indica, mi paso por la maestría en Investigación de Operaciones en materias tan específicas como Simulación, Métodos Heurísticos, Teoría de Redes, Programación Entera, Programación Dinámica y Metodología de la Investigación de Operaciones, me ha llevado a formular una propuesta que podría ayudarnos a resolver estos problemas mediante el uso de la Simulación Basada en Agentes y una heurística apoyada en *Ant Systems.*

Podemos plantear el problema de la siguiente forma:

1. En la literatura no existe mucha información acerca del uso de las heurísticas y algoritmos utilizados para la realización de este trabajo, debido a la especificidad del tema, lo cual dificulta que el usuario no especializado en el tema, haga una revisión rápida y objetiva de las herramientas matemáticas disponibles para la solución de problemas de tipo CVRP.
2. El uso de la simulación basada en agentes representa en sí misma una barrera en su uso e implementación, ya que, como se verá en los capítulos siguientes algunos autores ubican a esta herramienta como una de las más difíciles de utilizar comparada con otras que sirven de manera parecida al mismo propósito, ya que se requiere un conocimiento muy puntual tanto del problema como del uso de la herramienta en sí.
3. El problema principal que aquí se trata (*CVRP*) como ya se determinó en el apartado de complejidad computacional del *CVRP,* en un problema difícil de tratar por su pertenencia a la clase NP-Duro, y por su naturaleza de carácter exponencial, lo cual nos lleva a seguir desarrollando técnicas, herramientas, métodos o cualquier procedimiento que nos ayude a resolver estos problemas de una manera más eficiente y a bajo costo.

4. El uso de una heurística basada en *Ant Systems* obedece la complejidad que supone utilizar un *ACO* o más aun un *AS* formal ya que al aumentar los parámetros y acercarse a la formalización de *AS*, aumenta la complejidad en el uso de dicha herramienta, lo cual supone una dificultad aún más grande en su uso e implementación.

Por lo tanto, apoyándonos en lo expuesto con antelación, podemos llegar a formular el objeto de estudio de la tesis, que se define de la siguiente manera:

Solución de problemas de tipo CVRP utilizando un híbrido que consta de un método de dos fases y una heurística basada en Ant Systems mediante la Simulación Basada en Agentes.

El objeto de estudio se desarrollará ampliamente en el capítulo siguiente (Capítulo II), en él se expondrá todo lo relacionado con el objeto de estudio, estado del arte, marco teórico y la estrategia de investigación, todo esto con la finalidad de dar certeza al lector respecto del tema a desarrollar y el uso de las herramientas aquí aplicadas.

Capítulo 2

II. MARCO DE REFERENCIA

En este capítulo dejaremos en claro el objeto de estudio, y desarrollaremos lo referente al estado del arte, el marco teórico y la estrategia de investigación, en cada punto haremos un esfuerzo por sentar las bases teóricas que sustentan este trabajo de tesis.

2.1. OBJETO DE ESTUDIO

El objeto de estudio de esta tesis es: la solución de problemas de tipo *CVRP* utilizando un híbrido que consta de un algoritmo de dos fases (primero agrupar, luego rutear) y de una heurística basada en *Ant Systems* mediante la Simulación Basada en Agentes.

2.2. INVESTIGACIONES SOBRE EL OBJETO DE ESTUDIO

Con respecto al objeto de estudio, a lo largo del tiempo se han desarrollado técnicas tanto exactas como heurísticas para obtener buenas soluciones referentes a cada tipo de *VRP*, al respecto tenemos que, nuestra referencia más inmediata se encuentra en las técnicas metaheurísticas, con el algoritmo de las hormigas desarrollado por Colorni, Dorigo y Maniezzo en 1991 [Rocha, L. *et al.* 2011], da origen a los procedimientos de memora adaptativa desarrollados por Rochat y Taillard en 1995 como se muestra en la tabla 1:

Tabla 1. Soluciones meta-heurísticas de VRP.

Metaheurísticas	Búsqueda de vecindades	VNS, Hansen y Mlandenovik, 2003		
	Búsqueda Tabú	Glober, 1986		
	Algoritmo de Hormigas	Colorni, Dorigo y Maniezzo, 1991	Procedimiento de memoria adaptativa	Rochat y Taillard, 1995
	Recocido Simulado	Kirkpatric, Gellatt y Vecchi, 1983		
	Redes Neuronales	Horfield and Tank 1985	Recocido determinístico	Dueck, Scheurer, 1990
	Algoritmos Genéticos	Holland 1975		

Fuente: elaboración propia, basado en: Rocha, L. et al. 2011.

Al respecto de la tabla 1, podemos decir lo siguiente: una de las áreas de estudio más importante y prometedora de los últimos años es la investigación de algoritmos basados en comportamientos observados en la Naturaleza, los cuales permiten la obtención de métodos heurísticos no deterministas para la resolución de problemas de optimización combinatoria *NP-Hard.* La Naturaleza presenta dos grandes mecanismos: la selección, que premia a los individuos más fuertes y penaliza a los más débiles; y la mutación, la cual introduce elementos aleatorios y permite el nacimiento de nuevos individuos. Aplicado a la obtención de heurísticas, la selección implementa la optimización y la mutación permite la realización de una búsqueda no determinista. Por otro lado, las heurísticas provenientes de la observación de la Naturaleza son adaptativas, ya que emplean técnicas de retroalimentación de información para así poder modificar los parámetros que describen el funcionamiento del modelo [De La Fuente *et al.* 2011].

En los últimos años se ha producido una fuerte tendencia a la resolución de problemas de optimización combinatoria mediante la utilización de algoritmos *ACO* (*Ant Colony Optimization*). Estos algoritmos pertenecen al campo de *Swarm Intelligence* o inteligencia de enjambre, y están compuestos por individuos simples que cooperan de forma auto-organizada, es decir, sin ninguna forma de control central sobre los miembros del enjambre [De La Fuente *et al.* 2011].

ACO es una técnica de optimización de propósito general basada en el comportamiento de colonias de hormigas reales, concretamente en las feromonas que depositan entre la comida y el nido para marcar el mejor camino encontrado, Beckers *et al.* (1992) y Goss *et al.* (1989). La cantidad de feromonas en un camino aumenta cada vez que una hormiga lo atraviesa. A medida que esta cantidad incrementa, la probabilidad de que una hormiga siga ese camino también lo hace, por lo que la cantidad de feromonas en el camino más corto será mayor después de un determinado tiempo y, como consecuencia, un mayor número de hormigas tenderán a seleccionar dicho camino. Sin embargo, la decisión de seguir un camino o no, nunca es determinista, por lo que se permite la continua exploración de rutas alternativas. Estos algoritmos utilizan también un procedimiento de evaporación que reduce la cantidad de feromona a lo largo del tiempo, poniendo así más énfasis en nuevas direcciones de búsqueda para evitar quedarse estancado por decisiones pasadas [De La Fuente *et al.* 2011].

En la mayoría de estos algoritmos, cada hormiga construye una solución según una regla de decisión basada en dos parámetros: valores de las feromonas locales (qué tan bueno era el movimiento en el pasado) e información heurística local (basada en una información a priori como por ejemplo la distancia del movimiento). Por tanto, uno de los aspectos más importantes en el estudio de algoritmos heurísticos es el balance entre intensificación y diversificación. Demasiado énfasis en la primera puede producir que los agentes converjan

en un óptimo local y demasiado énfasis en la segunda puede causar un estado inestable. Sin embargo, estos dos factores son esenciales, ya que es necesario acelerar la convergencia y utilizar la diversificación para encontrar mejores soluciones [De La Fuente *et al.* 2011].

El primer algoritmo *ACO*, propuesto por Colorni, Dorigo y Maniezzo (1991-1992), recibe el nombre de *Ant System* (*AS*). Sus autores definen el término "hormiga" como un agente simple local que interactúa con el resto de agentes y que presenta las siguientes características [De La Fuente *et al.* 2011]:

- Libera una sustancia denominada *trail* o rastro a lo largo del camino al viajar desde una ciudad a otra (las feromonas en las hormigas reales);
- Elige la próxima ciudad a visitar con una probabilidad que es función de la distancia a la ciudad (su inversa recibe el nombre de "visibilidad") y de la cantidad de rastro presente en el camino que las conecta;
- Realiza un recorrido legal, ya que las transiciones hacia ciudades ya visitadas son inhibidas hasta que la ruta sea terminada, empleando para ello una lista tabú.

En el algoritmo *AS*, cada hormiga genera una ruta completa seleccionando las ciudades de acuerdo con una regla probabilística de transición entre estados, las hormigas prefieren moverse hacia ciudades más cercanas y con un alto contenido de feromonas en su camino [De La Fuente *et al.* 2011].

Esta regla de transición presenta parámetros configurables que determinan la importancia relativa de la feromona (rastro) frente a la longitud del trayecto (visibilidad). Posteriormente, se aplica una regla de actualización global de feromonas: se evapora una fracción de la feromona de los caminos no recorridos y cada hormiga deposita una cierta cantidad de la misma en los caminos pertenecientes a su ruta, en proporción a lo corta que ésta resultara. Finalmente, se itera el mismo proceso [De La Fuente *et al.* 2011].

Las diferentes formas posibles de computar la variación de feromonas a lo largo del tiempo y la selección del instante de actualización del rastro producen diferentes instancias de este algoritmo basado en hormigas: *Ant-density*, *Ant-quantity* y *Ant-cycle*. Una explicación más amplia de estas variaciones del algoritmo original, así como un mayor número de resultados computacionales obtenidos tras su aplicación para la resolución del bien conocido problema del *TSP*, pueden ser hallados en un artículo escrito por los mismos autores, Colorni *et al*. (1991). Posteriormente, Bullnheimer, Hartl y Strauss (1997) aplican el algoritmo *AS* a la resolución del problema *VRP* en su forma más básica, es decir, con restricciones de capacidad y distancias, un único almacén central y vehículos homogéneos. Para ello proponen un algoritmo *AS* híbrido, combinado el método original con una heurística 2-opt. Esta heurística consiste en que una ruta es mejorada borrando

dos arcos, es decir, invirtiendo uno de los caminos resultantes y luego reconectándolos hasta que no pueda obtenerse ninguna mejora adicional [De La Fuente *et al.* 2011].

Dos años más tarde (1999), los mismos autores presentan una mejora para este algoritmo híbrido, la cual consiste en la introducción en la fase inicial de una lista de candidatos para la selección de los clientes más prometedores. Esta lista de candidatos es obtenida ordenando el conjunto de las localizaciones de menor a mayor distancia. Dorigo y Gambardella (1997) desarrollaron, basándose en *AS* y con el objetivo de mejorar su eficiencia, el algoritmo *ACS* (*Ant Colony System*), cuya estructura ya presenta un cambio más sustancial con respecto al original. *ACS* difiere de *AS* en tres aspectos principales: i) la regla de transición entre estados proporciona una forma directa de balanceo entre exploración de nuevos caminos y explotación del conocimiento acumulado a priori; ii) la regla de actualización global es únicamente aplicada a los caminos pertenecientes a la mejor ruta y iii) mientras las hormigas construyen una solución se aplica una regla de actualización local de feromona. El algoritmo comienza con el posicionamiento de m hormigas en n ciudades elegidas de acuerdo a alguna regla de inicialización (por ejemplo de forma aleatoria). Cada hormiga construye una ruta aplicando la regla de transición entre estados. A medida que construye su ruta, la hormiga actualiza la cantidad de feromona en los caminos recorridos aplicando la regla de actualización local. Una vez que todas las hormigas han terminado su ruta, la cantidad de feromona en los caminos es modificada de nuevo aplicando la regla de actualización global. Stüzle y Hoos (2000) propusieron el algoritmo *MMAS* (*MAX-MIN Ant System*) como una mejora del *AS* inicial. Sus características principales son que únicamente la mejor hormiga actualiza el rastro de feromonas y que el valor de la feromona está acotado, tanto superior como inferiormente. Estas cotas son típicamente obtenidas de forma empírica y ajustadas según el problema específico considerado, Socha *et al.* (2002) [De La Fuente *et al.* 2011].

Recientemente, se han propuesto nuevos algoritmos basados en colonias de hormigas con el objetivo de obtener mejores resultados y rendimiento en la búsqueda de soluciones de problemas de optimización combinatoria más complejos. A pesar de que los algoritmos *ACS* y *MMAS* han demostrado tener un buen rendimiento, pueden presentar el fenómeno de quedarse estancados en un mínimo local, como resultado de la acumulación de feromonas [De La Fuente *et al.* 2011].

Muchos algoritmos *ACO* nuevos han sido propuestos mediante la introducción de los operadores genéticos (cruce, mutación y selección) en los métodos tradicionales, Kaveh *et al.* (2008) y Lee *et al.* (2008), para solventar los inconvenientes anteriormente mencionados, sin embargo, presentan una elevada complejidad y suponen un alto tiempo de computación. Por este motivo, algunos autores han decidido incluir en los algoritmos *ACO* únicamente alguno de los operadores. Es el caso de la reciente propuesta del algoritmo

MACO (*Mutated Ant Colony Optimization*), Zhao *et al.* (2010), en el cual se añade únicamente el operador de mutación. Esta mutación es utilizada para modificar de forma aleatoria uno o más elementos de la mejor solución local después de cada iteración. Si la solución mutada es mejor que la solución original, la primera sustituye a la segunda. En caso contrario, la mejor solución local permanece inalterada. De esta forma, se mejora el comportamiento de la búsqueda local, se expande la diversidad de soluciones y se evita la convergencia prematura. La mutación se aplica al algoritmo *MMAS* (*mutated MMAS* o *M-MMAS*) y al algoritmo *ACS* (*mutated ACS* o *M-ACS*) [De La Fuente *et al.* 2011].

Otro algoritmo propuesto para solventar el enrutamiento de vehículos es *EA* (*Evolutionary Ant Algorithm*), propuesto por Tsai *et al.* (2002), el cual introduce también mecanismos genéticos (operadores de cruce y mutación) para evitar los mínimos locales y un método llamado *NN* (*Nearest Neighbour*) para obtener una solución de forma más rápida. La heurística *NN* es un método de búsqueda local que consiste en que el viajante comienza en una ciudad y posteriormente visita la ciudad más cercana a la inicial. Seguidamente vuelve a visitar la ciudad más cercana a la última en la que se encontraba y así hasta que todos los nodos sean visitados y el viajante vuelva a la ciudad de origen. La ruta creada de esta forma sirve a modo de ruta inicial que posteriormente será mejorada por el algoritmo *EA*. *HK-ACO.* Li *et al.* (2008), es otro algoritmo mejorado de optimización con colonias de hormigas que emplea límites inferiores de *Held-Karp* para estimar la longitud de la ruta óptima. Esta medida, relativamente rápida y fácil de calcular, resulta muy práctica a la hora de evaluar soluciones cercanas a la óptima en grandes problemas donde el verdadero óptimo es desconocido [De La Fuente *et al.* 2011].

PDACO (*Population Declining ACO*), Wu *et al.* (2008), es una propuesta de algoritmo que pretende aumentar la capacidad de búsqueda global sin aumentar la complejidad computacional. Como ya se ha comentado con anterioridad, los algoritmos *ACO* presentan a veces la desventaja de no obtener un óptimo local debido a que la búsqueda cesa de forma prematura a medida que la cantidad de feromona va en aumento. Por otro lado, la población de hormigas se mantiene constante a lo largo de todas iteraciones, lo que supone un gasto de computación innecesaria en las fases finales del algoritmo ya que la población se mantiene muy grande y el espacio de búsqueda va decreciendo. En el algoritmo *PDACO* la población de la colonia es mucho más grande en las primeras iteraciones para aumentar así el rango de búsqueda. Posteriormente, a medida que se realizan iteraciones, la población va descendiendo para así disminuir la complejidad computacional. Esta propuesta se combina con diferentes algoritmos *ACO*, como *MMAS* (*PDMMAS*) y *ACS* (*PDACS*), para mejorar su funcionamiento. No modifica el algoritmo original ya que únicamente controla la población de la colonia [De La Fuente *et al.* 2011].

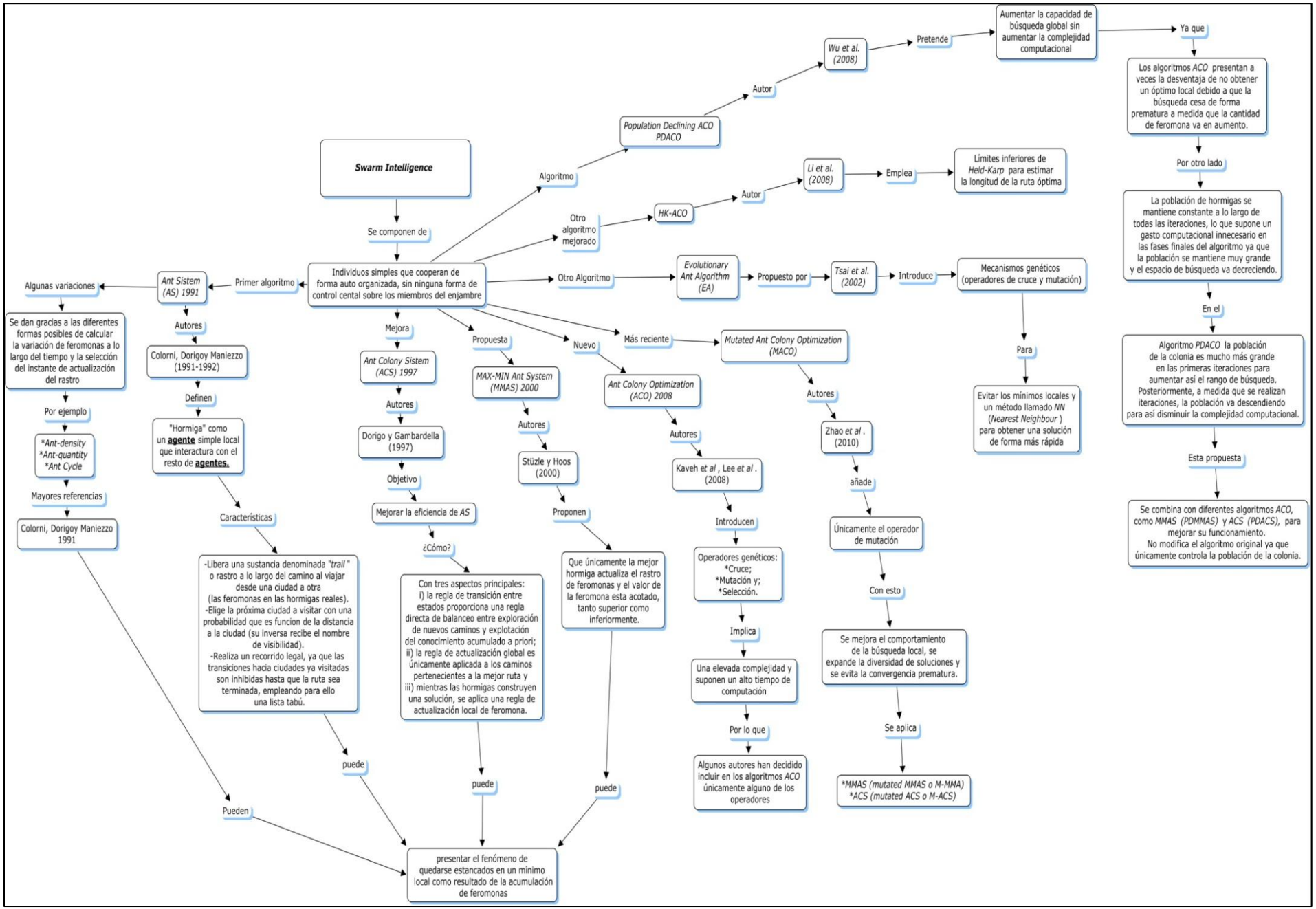
Al día de hoy, se han propuesto multitud de variantes de los algoritmos *ACO* originales, resultando algunas de ellas más exitosas que otras. La optimización basada en colonias de hormigas se trata de una metodología relativamente joven en comparación con otras tales como computación evolutiva, búsquedas tabú o *simulated annealing.* Aun así, se ha demostrado que estos algoritmos son bastante flexibles y eficientes [De La Fuente *et al.* 2011].

Lo anterior, nos da una idea clara de la complejidad que implica trabajar con algoritmos inspirados y diseñados a partir de la imitación de la Naturaleza, en nuestro caso *Ant Systems*, lo interesante es ver que el comportamiento natural de algunas clases de hormigas a la hora de buscar su alimento optimiza de manera instintiva la búsqueda (encontrar la mejor ruta para traer al nido el alimento, así como la exploración del espacio), esto ya lo dicen Toth y Vigo [Toth and Vigo. 2002] y nos perfila hacia la idea que nos conduce en este trabajo.

Más recientemente, podemos encontrar algunos artículos relacionados con la heurística *ACO* aplicada a *CVRP* tales como [Redd *et al.* 2014][Gajpal *et al*. 2009][Mazzeo *et al.* 2004]. Si hacemos un análisis de por qué se utiliza más ampliamente *ACO* que *AS*, debemos recordar que la primera heurística en ser desarrollada fue *AS* y esta a su vez fue mejorada para convertirse en *ACO*, cabe mencionar también que *ACO* por su sofisticación, es más difícil de usar e implementar ya que requiere el uso de más elementos y un conocimiento a detalle del problema, en nuestro caso al utilizar una heurística basada en *AS* lo que hacemos es que no utilizamos tanta sofisticación como en *AS* o *ACO,* pero seguimos aprovechando la ventaja que nos da usar heurísticas basadas en los comportamientos naturales como los que nos muestran los enjambres de algunos insectos, en nuestro caso, las hormigas.

A manera de resumen podemos ver la figura 14, misma que nos explica de manera general la evolución de los diferentes algoritmos relacionados con *AS* y *ACO.*

Figura 14. **Mapa conceptual del estado del arte *AS* y *ACO***



Fuente: elaboración propia, baso en [De La Fuente et al. 2011]

Este trabajo de tesis, encuadra en las líneas de investigación de *ACO*, para resolver problemas de tipo *CVRP*, y el aporte principal es la utilización de un híbrido entre un algoritmo de dos fases y una heurística basada en *AS* para solucionar *CVRP*. Además de que su utilización e implementación no requieren un conocimiento exhaustivo del problema, por lo cual es relativamente fácil de utilizar y ha demostrado tener buenos resultados.

2.3. MARCO TEÓRICO

En resumen, las disciplinas mediante las cuales se debe abordar el tema son: Simulación, Métodos Heurísticos, Teoría de Redes, Programación Lineal Entera y Metodología de la Investigación de Operaciones.

SIMULACIÓN

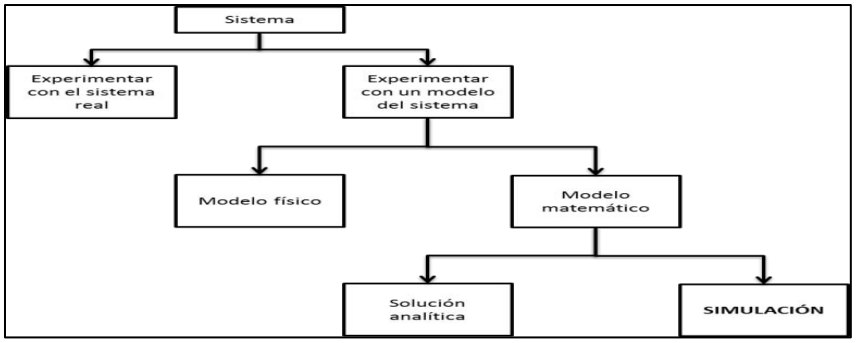
Algunos de los problemas de optimización no se pueden resolver fácilmente de manera analítica; una alternativa a esta problemática es utilizar la simulación. La simulación es una técnica con aplicaciones en distintos campos, utilizada para el análisis y estudio de sistemas complejos [Flores *et al.* 2007].

La simulación, es una herramienta de la Investigación de Operaciones que sirve para analizar problemas de optimización en sistemas donde los recursos son limitados o difíciles de resolver mediante un algoritmo. "Los modelos de simulación dividen el sistema representado en módulos básicos entrelazados por relaciones lógicas" [Flores *et al.* 2007]. Es una técnica que construye un modelo, lo más parecido al sistema real bajo estudio, y lo hace funcionar a semejanza del sistema real en condiciones de prueba, para analizar todos los posibles desempeños del sistema.

Cuándo usar la simulación

Para analizar las razones del uso de la simulación, es interesante explorar las alternativas existentes para dicha herramienta, es decir, los diferentes métodos que pueden usarse para resolver el mismo problema como lo ilustra el diagrama de la figura 15 [Flores *et al.* 2007]:

Figura 15. Diagrama de decisión para el uso de modelos.



Fuente: elaboración propia, basado en: Flores et al. 2007

En la mayoría de los casos, la simulación se usa cuando las alternativas matemáticas, son pobres, es decir, es el "último recurso" algo así como: "cuando todo falle, use la simulación".

MÉTODOS HEURÍSTICOS

Al respecto se abordan temas tales como la complejidad computacional importante en este tema de tesis y de los temas fundamentales que justifican este trabajo, así como los algoritmos glotones, de búsqueda local, con pérdida de memoria, con memoria adaptativa, de búsqueda aleatoria, de búsqueda sistemática y algunos problemas de aplicación.

TUNING

El *tuning* consiste en la selección y afinación de los parámetros adecuados para la simulación ya que según [Dorigo *et al.* 1996] las diferentes combinaciones en la selección de los parámetros hacen que el rendimiento del algoritmo varíe. Es importante resaltar que el *tuning* o "afinación" de los parámetros de la simulación es indispensable en la obtención de buenos resultados, y además el *tuning* es muy específico de cada problema y de cada instancia. El ajuste es necesario ya que cada problema tiene características diferentes. El *tuning* es difícil de lograr ya que, los parámetros dependen del problema y los mejores valores de los parámetros no son estables a lo largo de la búsqueda [B. Crawford 2013].

Por otro lado, como lo demuestra [Gaertner *et al.* 2005] no existe una correlación, estadísticamente significativa, entre los parámetros, ya que en algunos casos, relativamente similares entre sí, mientras algún parámetro permanece casi sin variación, los otros se mueven de manera muy amplia. Mientras que en otros casos al azar, se requieren combinaciones de parámetros en una gama mucho más estrecha de valores.

TEORÍA DE REDES

Desde los conceptos básicos de la Teoría de Redes, así como los análisis de flujos, flujos y potenciales y los tópicos especiales.

PROGRAMACIÓN LINEAL ENTERA

Indispensable como introducción en el conocimiento, planteamiento y manejo de los diferentes problemas generales, tales como: *Integer knapsak Problem, Facility Location Problem, Scheduling,* problemas de cubrimiento y partición de conjuntos, *TSP*, así como algunos algoritmos y heurísticas para su solución.

METODOLOGÍA DE LA INVESTIGACIÓN DE OPERACIONES.

Básicamente el objetivo de dicha materia es: "Proporcionar las bases metodológicas que permitan desarrollar el análisis y solución de problemas reales en el campo de la Investigación de Operaciones, en especial los problemas relacionados con la recopilación de información, construcción de modelos y la implantación de la solución, además de las

diferencias, similitudes y relaciones entre la Investigación de Operaciones, Ingeniería de sistemas y Planeación, destacando los problemas metodológicos comunes." El objetivo es bastante claro y podemos decir que es parte primordial para la construcción y desarrollo de toda esta tesis, ya que toda la investigación realizada esta basada en los conceptos fundamentales de esta materia.

TEORÍAS SOBRE EL OBJETO DE ESTUDIO

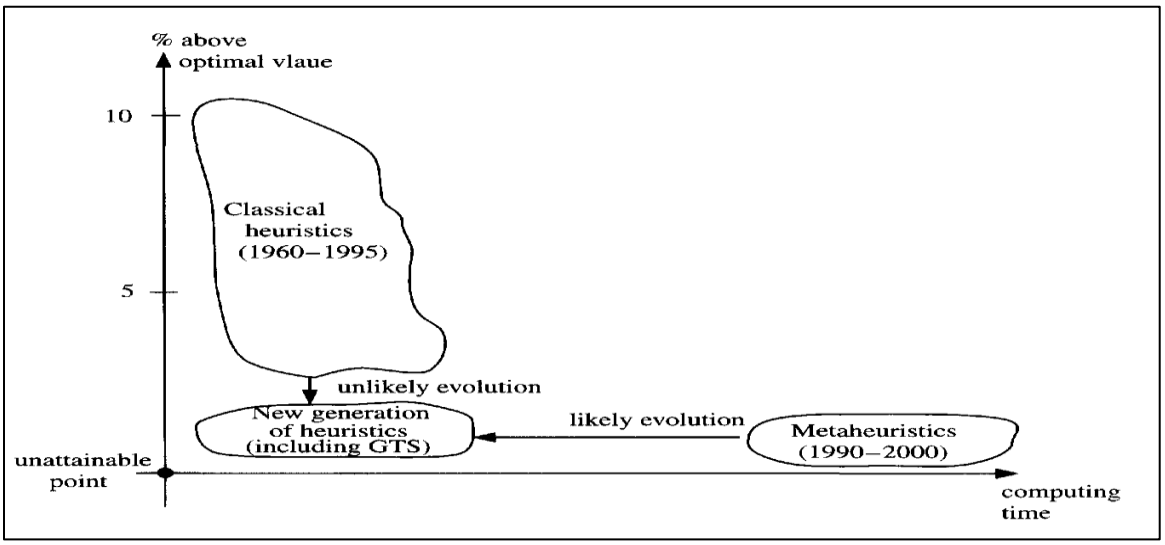
Ant Systems (AS)

Los métodos basados en Sistemas de hormigas (*AS*) están inspirados por la analogía con las colonias de hormigas reales en su búsqueda por alimento. En esa búsqueda por alimento, las hormigas marcan las rutas por donde viajan, colocando una esencia aromática llamada feromona. La cantidad de feromona establecida en una ruta, depende de la longitud de la ruta y de la calidad de la fuente de comida. Esta feromona provee información a otras hormigas que se sienten atraídas a ella. Con el tiempo, las rutas conducen a las fuentes de alimento más atractivas, esto es, las más cercanas al nido y con mayor cantidad de alimento, ya que son más frecuentadas y están marcadas con mayores cantidades de feromona. En general, este proceso conduce a un procedimiento eficiente para la adquisición de alimentos por las colonias de hormigas [Toth and Vigo. 2002].

El método se refinó mediante a adición de varias características en aplicaciones tales como el *TSP* simétrico y asimétrico [Colorni *et al.* 1991; Dorigo *et al.* 1997]. Una conclusión general se puede sacar de estos documentos, si bien el método puede producir excelentes resultados, este no suele competir con otras metaheurísticas o heurísticas especializadas en búsqueda local, a menos que se hibride de una u otra manera con un optimizador local [Toth and Vigo. 2002].

Al respecto los autores Toth y Vigo [Toth and Vigo. 2002] presentan una gráfica que muestra la evolución de las heurísticas para el VRP.

Figura 16. Evolución de las heurísticas para VRP.



Fuente: tomado de Toth and Vigo. 2002

Métodos Asignar Primero - Rutear Después

Los métodos asignar primero y rutear después (*cluster-first, route-second*) proceden en dos fases. Primero se busca generar grupos de clientes, también llamados *clusters*, que estarán en una misma ruta en la solución final. Luego, para cada *cluster* se crea una ruta que visite a todos sus clientes. Las restricciones de capacidad son consideradas en la primera etapa, asegurando que la demanda total de cada *cluster* no supere la capacidad del vehículo. Por lo tanto, construir las rutas para cada *cluster* es un TSP que, dependiendo de la cantidad de clientes en el *cluster*, se puede resolver de forma exacta o aproximada.

Simulación Basada en Agentes (ABS)

En las últimas décadas, debido en gran parte al desarrollo de las computadoras, han aparecido innovadoras técnicas de modelado de sistemas complejos que hacen uso de la nueva tecnología disponible. Dos de estas técnicas son la simulación basada en agentes y la dinámica de sistemas [R. Izquierdo *et al.* 2008].

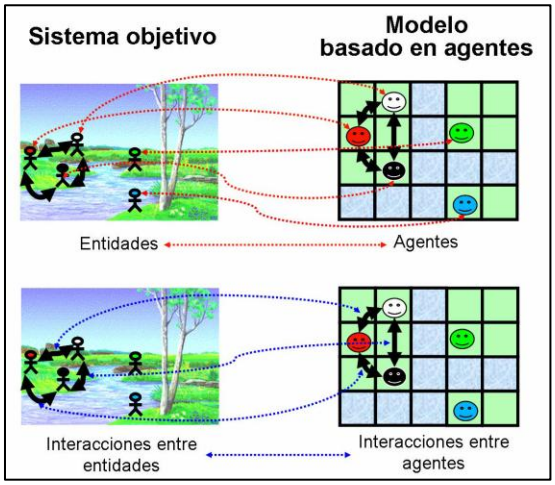
Con el desarrollo de las computadoras han aparecido nuevos enfoques de modelado científico que hacen uso de la simulación computacional como proceso inferencial. En las simulaciones computacionales, el modelo se codifica en un lenguaje de programación formal, y la inferencia se lleva a cabo ejecutando el programa informático desarrollado. Así, el actual desarrollo de la informática nos permite crear y estudiar mediante simulación computacional, modelos formales que, debido a su complejidad, venían siendo intratables matemáticamente [R. Izquierdo *et al.* 2008].

La simulación basada en agentes ha demostrado ser una técnica tremendamente útil para modelar sistemas complejos [Conte *et al.* 1997; Gilbert & Troitzsch 1999; Gilbert & Terna 2000; Gilbert 2007]. Mediante la simulación basada en agentes, el modelador reconoce

explícitamente que los sistemas complejos, son producto de comportamientos individuales y de sus interacciones [R. Izquierdo *et al.* 2008].

Lo que distingue a la simulación basada en agentes de otras técnicas de modelado es la forma en que se construye la primera abstracción del sistema real y, consecuentemente, el modelo formal. En los modelos formales construidos mediante simulación basada en agentes, los componentes básicos del sistema real están explícita e individualmente representados en el modelo [Edmonds *et al.* 2001]. De esta forma, como se muestra en la figura 17, las *fronteras* que definen a los componentes básicos del sistema real se corresponden con las *fronteras* que definen a los agentes del modelo, y las *interacciones* que tienen lugar entre los componentes básicos del sistema real se corresponden con las *interacciones* que tienen lugar entre los agentes del modelo [Edmonds *et al.* 2001; Galán *et al.* 2008]. Esta correspondencia directa contrasta con el tradicional uso de "agentes representativos" y es capaz de aumentar el realismo y el rigor científico de los modelos formales así construidos [R. Izquierdo *et al.* 2008].

Figura 17. Los componentes básicos del sistema real y las interacciones entre ellos están explícita e individualmente representados en el modelo.



Fuente: tomado de R. Izquierdo et al. 2008

Los sistemas basados en agentes se caracterizan por comprender varios agentes que son (en mayor o menor grado) autónomos, heterogéneos e independientes, que muestran cada uno sus propias metas y objetivos, y que generalmente son capaces de interaccionar entre sí y con su entorno [Torsun 1995]. En muchas ocasiones, pero no siempre, son sistemas caracterizados por la existencia de un número grande de agentes relativamente simples, que pueden evolucionar a lo largo del tiempo para adaptarse a nuevas condiciones del entorno o a nuevos objetivos. En particular, la simulación basada en agentes es especialmente relevante en sistemas complejos con las siguientes características [R. Izquierdo *et al.* 2008]:

- Sistemas con componentes individuales heterogéneos; especialmente aquéllos en los que las implicaciones de esta heterogeneidad no se han estudiado en profundidad [Axtell, 2000]. Varias disciplinas (por ejemplo: la economía neoclásica) utilizan frecuentemente la hipótesis de 'individuo prototipo' o 'agente representativo', confiando en que los resultados obtenidos bajo esta hipótesis puedan aplicarse satisfactoriamente a sistemas reales en los que los individuos son, en realidad, heterogéneos. Sin embargo, en sistemas caracterizados por fuertes externalidades (por ejemplo: la explotación del medio ambiente, la gestión de recursos comunes…), los resultados obtenidos usando modelos que asumen la existencia de un individuo representativo están, en muchas ocasiones, muy alejados de las observaciones empíricas (ver por ejemplo: Ostrom *et al*. 1994). Ignorar por principio la existencia de heterogeneidad entre individuos es una fuerte premisa que debe ser contrastada; para ello, resulta necesario estudiar el sistema con heterogeneidad, y el modelado basado en agentes es particularmente útil para acometer este análisis;
- Sistemas adaptativos, i.e. sistemas en los que los componentes individuales del sistema son capaces de aprender (adaptación a escala individual), o bien pueden ser seleccionados y reemplazados de acuerdo con algún criterio (adaptación a nivel poblacional). En cualquiera de estos dos casos parece claro que resulta conveniente representar explícita e individualmente cada componente del sistema;
- Sistemas en los que el espacio geográfico puede tener una influencia significativa. En muchos sistemas, el hecho de que dos individuos estén separados en el espacio supone una probabilidad de interacción inferior. El modelado basado en agentes facilita la representación del espacio físico en el que se mueven e interactúan los agentes;
- Sistemas en los que existen redes sociales de interacción. Las interacciones entre componentes del sistema pueden estar influenciadas por diversos factores además de por el espacio físico. El modelado basado en agentes facilita la representación explícita de redes de interacción social que no están necesariamente estructuradas espacialmente;
- Sistemas en los que se desea analizar en profundidad la relación existente entre los atributos y comportamientos de los individuos (la 'micro-escala') frente a las propiedades globales del grupo (la 'macro-escala') [Gilbert & Troitzsch 1999; Squazzoni 2008].

La metodología basada en agentes ha sido utilizada ampliamente para modelizar sistemas en un amplio rango de disciplinas científicas: por ejemplo, economía [Tesfatsion 2002, 2003], finanzas [LeBaron, 2000], gestión de recursos naturales y ecología [Bousquet & Le Page 2004; López & Hernández 2008], ciencias políticas [Axelrod 1997; Johnson 1999], antropología [Kohler & Gumerman 2000], sociología [Conte *et al.* 1997; Gilbert & Troitzsch 1999; Gilbert 2007], biología [Paton *et al.* 2004; Walker *et al.* 2004a; Walker *et al.* 2004b] o medicina [Mansury *et al.* 2002; Mansury & Deisboeck, 2004], en los que partiendo de reglas que determinan el comportamiento individual de los agentes se pretende inferir las propiedades globales de

todo el sistema [Holland 1998]. Los métodos basados en agentes facilitan el estudio y modelado de sistemas complejos a partir de las unidades que los componen, permitiéndonos construir modelos experimentales de la realidad desde un punto de vista diferente al tradicional: desde lo más simple hacia lo más complejo [R. Izquierdo *et al.* 2008].

Sin duda, uno de los puntos fundamentales de la simulación basada en agentes es el concepto de *emergencia*. Los fenómenos emergentes son patrones macroscópicos que surgen a partir de las interacciones descentralizadas de componentes individuales más simples [Holland 1998]. Lo que caracteriza a estos fenómenos emergentes es que su presencia o aparición no resulta evidente a partir de una descripción del sistema consistente en la especificación del comportamiento de sus componentes individuales y de las reglas de interacción entre ellos [Gilbert & Terna 2000; Gilbert 2002; Squazzoni 2008]. Un ejemplo típico de fenómeno emergente es la formación de grupos diferenciados en el modelo de segregación de Schelling (1971); la aparición de patrones claros de segregaciónno está explícitamente impuesta en la definición del modelo, sino que emerge de las interacciones locales de individuos con tendencias segregacionistas en ocasiones sorprendentemente débiles. Otro ejemplo son los patrones migratorios de *Sugarscape* [Epstein & Axtell 1996][R. Izquierdo *et al.* 2008].

Existen multitud de fenómenos emergentes en diversas disciplinas (ver por ejemplo: Reynolds 1987; Holland 1998; Johnson 2001), pero sin duda es en las ciencias sociales donde la idea de emergencia cobra una dimensión adicional de complejidad e importancia. En muchos sistemas sociales en los que interviene la dimensión humana cabe la posibilidad de que cada uno de los componentes individuales del sistema tome cierta consciencia del fenómeno emergente del que es causa parcial y que, como consecuencia de esta percepción consciente, reaccione modificando su comportamiento. Este fenómeno (conocido como emergencia de segundo orden [Gilbert 2002; Squazzoni 2008]) subyace tras la complejidad de muchos sistemas sociales [R. Izquierdo *et al.* 2008].

Los mercados financieros constituyen un ejemplo claro de emergencia de segundo orden. En estos mercados, el precio (fenómeno emergente) surge de la interacción de varios inversores (componentes individuales del sistema); los inversores perciben de forma consciente el precio que ellos mismos generan, y modifican su comportamiento en respuesta a esta observación, creando así un complejo lazo de realimentación entre los dos niveles jerárquicos del sistema [R. Izquierdo *et al.* 2008].

Aplicaciones de la Simulación Basada en Agentes (ABS)

El área de investigación de Simulación Basada en Agentes (*ABS*) continúa produciendo técnicas, herramientas y métodos, además de un gran número de aplicaciones de ABS que se han desarrollado.

Las distintas áreas de aplicación son [UTN 2010]:

- **Comunicaciones:** La aplicación de la simulación en las industrias de las comunicaciones es cada vez más vital. Redes *LAN*, redes *WAN*, inalámbricas, sistemas telefónicos, sistemas de comunicaciones satelitales nacionales e internacionales, redes de televisión por cable y teléfonos celulares son ejemplos de los complejos sistemas que demandan la capacidad de la simulación para lograr un diseño y operación eficientes.
- **Educación:** Estudios relacionados a los efectos de cambios en los niveles de inscripción, procesos de registro, ubicación y *scheduling* de aulas, planeación del inventario de la biblioteca y operaciones de diseño de sistemas para escuelas y universidades pueden ser realizados por simulación.
- **Entretenimientos:** Las técnicas de simulación están siendo muy usadas en el diseño de la estructura y operación de los parques de diversiones, estudios de producción y sistemas de cines y teatros, sistemas de venta de *tickets*, diseño del estacionamiento de autos, diseño de la capacidad y *scheduling* de paseos, equipamiento y *scheduling* de producción de películas, son algunos de los típicos propósitos de aplicación de la simulación en la industria del entretenimiento.
- **Servicios Financieros:** Existen muchos reportes de aplicaciones de simulación en un banco, en la bolsa de valores y en las compañías de seguros. Análisis de las transacciones, de *cash-flow*, diseño de sistemas de oficina, planeación de los materiales y suministros, procesamiento de datos, diseño de redes, diseño de los sistemas de manejo de los cajeros automáticos son algunas de las actividades que pueden ser realizadas por la simulación.
- **Servicios Alimenticios:** Sistemas de pagos en restaurantes, en locales de comida rápida y sistemas de almacenaje de comestibles, pueden ser sujetos a estudios de simulación con propósitos como planeación del inventario y de provisiones, planeación de la distribución, selección del sitio, *layout*, planeación y *scheduling* de mano de obra.
- **Sistemas de Salud:** Hospitales, consultorios de emergencia, oficinas de médicos, son frecuentemente estudiados por la simulación para determinar los cambios de horarios de médicos y enfermeras, inventario de medicamentos y alimentos, planeación de la capacidad de recursos como camas, capacidad de las salas de espera, de quirófanos, equipos y ambulancias. También estudios de epidemiología, como pronósticos de las tasas de propagación de enfermedades y análisis de

políticas alternativas de control de enfermedades, todas estas son realizadas por la simulación.

- **Hotelería:** Sistemas de hoteles, hostal y resort son estudiados por la simulación para determinar factores como son capacidad, políticas de administración de los recursos de inventario, planeación de la mano de obra y métodos de *scheduling*, sistemas de reservas y contratación.
- **Transportes:** Estos sistemas involucran uno o más tipos de vehículos (Por ejemplo: taxis, ómnibus, trenes, barcos, aviones), pasajeros, rutas de transporte y carga. El objetivo de la simulación puede ser obtener la capacidad del vehículo, del personal, planeación y *scheduling*, planeación de recambio de partes, de mantenimiento, planeación urbana, rutas de los vehículos, diseño de nuevas autopistas, diseño de sistemas de control de crecimiento del espacio aéreo y diseño de lugares de estacionamiento.
- **Pronósticos del tiempo, medio ambiente y ecología:** Los pronósticos del tiempo, rutinaria e intensivamente usan la simulación. Un gran número de variables son utilizadas por programas de simulación para predecir la situación del tiempo local y global. Estudios relacionados con el control de la polución, el efecto invernadero, población de insectos y otros flujos ambientalistas y ecologistas son también desarrollados a través de la simulación.

Los sistemas de producción y manufactura son otras de las aplicaciones de la simulación. Algunos de los sistemas típicos son las siguientes:

- **Extracción/cosecha de recursos naturales:** industrias como la minería, maderera, de perforación y pesquera usan la simulación para planificar las actividades relacionadas a la creación de políticas de elección oportuna de recursos, como grandes maquinarias, máquinas cargadoras, ascensores, máquinas excavadoras, grúas, trituradoras, cintas transportadoras y barcos.
- **Crecimiento de plantas y animales:** los sistemas de crecimiento pueden ser simulados para pronósticos de producción, planeación de recursos como la tierra, fertilizantes, alimentos para los animales, medicamentos, tractores, cosechadoras, vehículos de transporte y para el estudio y diseño de procedimientos operacionales para determinar factores como producción, crecimiento, almacenaje y distribución.
- **Generación de energía:** sistemas de generación de energía eléctrica basados en fuentes como son vapor, combustible fósil, termal, nuclear, solar o de viento usan la simulación para diseñar sistemas de capacidad, configuración y distribución y para el análisis y diseño de sistemas operacionales los cuales pueden tener salidas como la programación de la tasa de generación, la planificación de la distribución, el diseño de sistemas de control, el diseño de sistemas seguros y fiables, la programación de mantenimiento y de control de impacto ambiental.

Fabricación: plantas de procesamiento químico, industrias de automóviles, de aviones, electrónicas, de maquinarias, de herramientas y otros, usan la simulación extensivamente en aplicaciones como son la planeación estratégica, planeación de la capacidad y producción media, *layout* de la planta, selección y reemplazo del equipo, diseño de políticas de mantenimiento y reemplazo, planeación y control de inventario, planeación de la producción, balance de la línea de ensamblaje, almacenamiento y manipulación de materiales, diseño de sistemas de manufactura, y numerosas salidas relacionadas con el diseño, fabricación, ensamblaje, control de calidad, *packaging*, almacenaje y distribución. La popularidad de los estudios de simulación en sistemas de manufactura han incrementado, por lo que, un considerable número de herramientas de simulación, de propósitos especiales, están disponibles comercialmente para el diseño y análisis de estos sistemas.

Tabla 2. Ventajas y desventajas de los diferentes tipos de simulación.

Complejidad	Enfoque: Simulación de eventos discretos	Simulación de Dinámica de Sistemas	Simulación basada en Agentes
No linealidad	↑	↑	↕
Interacciones	←	←	↑
Agentes Inteligentes	←	←	↕
Comportamientos emergentes	←	←	↕
Adaptación	←	←	↕
Comportamientos Dinámicos	↑	↑	↑
Fácil creación	←	→	↓
Fácil de validar y verificar	→	→	↓

↕Excelente ↑Muy bueno →Bueno ←Pobre ↓Muy pobre

Fuente: Elaboración Propia, basado en: S. Balestrini et al. 2009

Como se puede observar en la tabla anterior (tabla 2), la simulación basada en agentes (*ABS*) presenta muchas ventajas sobre la simulación discreta y continua, básicamente la flexibilidad que presenta *ABS*, sólo se ve opacada por su dificultad en su creación y por su dificultad en la validación y verificación.

Desde un punto de vista formal, se puede definir la simulación basada en agentes como un método informático que permite construir modelos constituidos por agentes que interaccionan entre sí dentro de un entorno para llevar a cabo experimentos virtuales [Gilbert 2008b]. Entender esta definición implica aclarar qué son los experimentos virtuales, quiénes son los agentes y cuál es su entorno [García-Valdecasas 2011].

Un experimento consiste en aplicar algún tratamiento a una parte aislada de la realidad y observar qué ocurre; posteriormente, el objeto de análisis que ha sido tratado es comparado con otro objeto equivalente que no ha recibido ningún tratamiento (llamado «control»). La gran ventaja de la experimentación es que permite asegurar que el tratamiento aplicado al objeto de estudio es de hecho la causa de los efectos observados, ya que solo el tratamiento es lo que difiere entre el objeto de análisis y el de control [Gilbert 2008b] [García-Valdecasas 2011].

Asimismo, los modelos basados en agentes pueden también realizar experimentos porque permiten aislar virtualmente determinadas actividades de otros procesos e investigar las causas de los efectos observados. Además, dichos experimentos no plantean problemas éticos porque utilizan agentes y sociedades artificiales. Así pues, es posible diseñar un experimento virtual y llevarlo a cabo tantas veces como se desee, pudiéndose usar un rango amplio de parámetros o incluso permitir, por ejemplo, que algunos factores varíen al azar [Gilbert 2008b], comparando posteriormente posibles resultados [García-Valdecasas 2011].

Actualmente, encontramos diversas opciones de *software* para implementar la simulación basada en agentes (*ABS*) como se observa en la tabla 3. Por otro lado, tenemos que recordar que, construir un modelo basado en agentes (*ABM*) es una tarea muy compleja si hay que utilizar las técnicas de programación clásicas.

Tabla 3. Software de simulación ABMS.

➢ ABLE	➢ GPU Agents	➢ MASON
➢ AgentBuilder	➢ GROWlab	➢ MAS-SOC
➢ AgentSheets	➢ iEcho	➢ MOOSE
➢ Anylogic	➢ iGem	➢ NetLogo
➢ AOR	➢ JABM	➢ OBEUS
➢ Ascape	➢ JCA-Sim	➢ OCARO-T
➢ Brahms	➢ JADE	➢ Omonia
➢ Breve	➢ JANUS	➢ Repast
➢ Construct	➢ JAMEL	➢ SeSam
➢ Cornas	➢ JAS, JASA	➢ Simport
➢ Cougar	➢ jES	➢ Soar
➢ Dex	➢ JESS	➢ Sugarscape
➢ Digihive	➢ LSD	➢ Swarm
➢ ECHO_ECJ	➢ Madkit	➢ ZEUS
➢ FAMOJA	➢ MAGSY	
➢ Framsticks	➢ MAML	

Fuente: Elaboración propia, Tomado de: Huerta 2014

Por lo tanto, nuestro problema de decisión, se reduce a elegir alguno(s) de entre tantos. Sin embargo, ¿con base en qué criterios podeos seleccionar el *software* más adecuado a nuestros propósitos de modelación y simulación? Una respuesta a esta pregunta la sugieren Nikoukaran *et al. (*2009) quienes recomiendan que al seleccionar un *software* se tomen en cuenta algunos de los siguientes siete criterios: *el fabricante, el tipo de entradas del modelo, la ejecución de las simulaciones, el tipo y calidad de la animación de la simulación, la eficiencia de la simulación, el tipo de salidas requeridas para análisis* o bien, *el usuario* [Huerta 2014].

En nuestro caso los criterios de decisión que nos hicieron definirnos por el *software* NetLogo, fueron: *el fabricante, la ejecución de las simulaciones, la eficiencia de la simulación, el tipo de salidas requeridas para análisis y el propio usuario.*

Los modelos basados en agentes se pueden también desarrollar utilizando lenguajes de programación orientados hacia objetos (con sus correspondientes bibliotecas), tales como Java, C++ o Visual Basic; o empleando entornos de programación que permiten crear, ejecutar y visualizar resultados sin salir del sistema, como NetLogo, *Swarm*, *Mason* o *Repast*. Los modelos construidos con lenguajes de programación poseen ciertas ventajas con respecto a los diseñados con entornos de programación, por ejemplo se ejecutan más rápidamente, y además son mucho más flexibles que los entornos de programación, es decir, permiten programar cualquier modelo por complejo que sea. Sin embargo, los entornos de programación son mucho más fáciles de aprender, y requieren menos tiempo para desarrollar un modelo que los lenguajes de programación. Entre los lenguajes de programación, los compilados (C++) se ejecutan más rápido que los interpretados (Java y Visual Basic); sin embargo, los lenguajes interpretados permiten moverse entre las tareas de escribir, probar y modificar el código con mayor facilidad. NetLogo es posiblemente el entorno de programación más rápido de aprender, más fácil de usar y más simple de instalar, pero es ligeramente más lento que el resto y por lo tanto no es adecuado para modelos excesivamente complejos y grandes [Gilbert 2008b] [García-Valdecasas 2011]. Si a lo que acabamos de leer le agregamos que NetLogo es de distribución libre, entenderemos por qué para desarrollar nuestra investigación utilizamos NetLogo, y no algún otro *software* relacionado con *ABS*.

NETLOGO

NetLogo es una plataforma que ofrece *ABS* permitiendo comparar en tiempo real de ejecución, y dentro de la misma plataforma, un modelo diseñado con un enfoque basado en agentes y otro diseñado bajo la óptica de dinámica de sistemas. NetLogo es apto para modelar sistemas complejos a lo largo del tiempo. Los modeladores pueden dar instrucciones a través de una gran cantidad de "agentes" que operan independientemente. Esto permite explorar la conexión a niveles detallados de comportamiento y reflejar patrones a gran escala en su interacción.

Es un entorno de modelado programable multi-agente. Es utilizado por decenas de miles de estudiantes, profesores e investigadores de todo el mundo. Es una herramienta muy útil que nos permite hacer de manera fácil y rápida, simulaciones basadas en agentes. Cuenta con una "librería" muy amplia que consta de programas diseñados para tareas generales o para ilustrar los atributos de dicha plataforma que, al ser libre, nos permite modificar completamente cualquier programa para adaptarlo a un problema específico, lo cual nos brinda mucha flexibilidad a la hora de programar en dicha plataforma.

Sirve para simular fenómenos naturales y sociales. Fue desarrollado por Uri Wilensky en 1999 y ha estado en constante desarrollo desde entonces en *The Center for Connected Learning (CCL) and Computer-Based Modeling*.

NetLogo es especialmente adecuado para modelar sistemas complejos de desarrollo en el tiempo. Los modeladores pueden dar instrucciones a cientos o miles de "agentes" que actúan todos de forma independiente. Esto hace que sea posible explorar la conexión entre el comportamiento a nivel micro de los individuos y los patrones de nivel macro que surgen de su interacción. Permite a los usuarios simulaciones abiertas y la manipulación de estas, así como explorar su comportamiento bajo diversas condiciones. También es un entorno de edición que permite a los usuarios, y desarrolladores crear sus propios modelos. Es bastante simple para los estudiantes y maestros, pero lo suficientemente avanzada para servir como una poderosa herramienta para los investigadores en muchos campos.

Cabe señalar que NetLogo, funciona de la siguiente forma: divide su espacio virtual en parcelas, estas parcelas son unitarias y el usuario puede editar el tamaño del espacio virtual que está en función del número de parcelas. Estas parcelas forman un plano cartesiano, en donde cada parcela está identificada por medio de sus coordenadas en R^2, lo que quiere decir que, el tamaño del espacio virtual, en nuestro caso, está en función de las coordenadas de los clientes. El nido hace la función del centro de distribución CEDIS y éste está situado en el origen del plano cartesiano con coordenadas (0,0). Es importante señalar que, el hecho de tener cuadriculado el espacio nos facilita mucho el trabajo de identificación de los clientes ya que es muy sencillo situarlos en el espacio virtual gracias a

sus coordenadas. Por el otro lado, es importante no perder de vista que al estar cuadriculado el espacio, conforme las coordenadas con las que queremos trabajar tienen valores de en x e y más grandes, el tiempo de computo también aumenta ya que el programa realiza un barrido de cada una de las parcelas contenidas en el espacio virtual y ejecuta las instrucciones correspondientes en cada parcela, por lo que, si tenemos valores pequeños en las coordenada, el espacio virtual es pequeño y el tiempo de cómputo también lo es, en cambio, si los valores de las coordenadas son grandes, el tiempo de cómputo será mayor. Por lo anterior, es necesario poner atención a los valores de las coordenadas que estamos manejando, ya que estas influyen en el tiempo de cómputo de nuestra simulación.

Por otro lado, NetLogo tiene una amplia documentación y tutoriales. También contienen una Biblioteca de modelos, una gran colección de simulaciones pre-escritas que se pueden utilizar y modificar. Estas simulaciones abordan las áreas de contenido en las ciencias naturales y sociales, incluyendo la biología y la medicina, la física y la química, las matemáticas y la informática, y la economía y la psicología social.

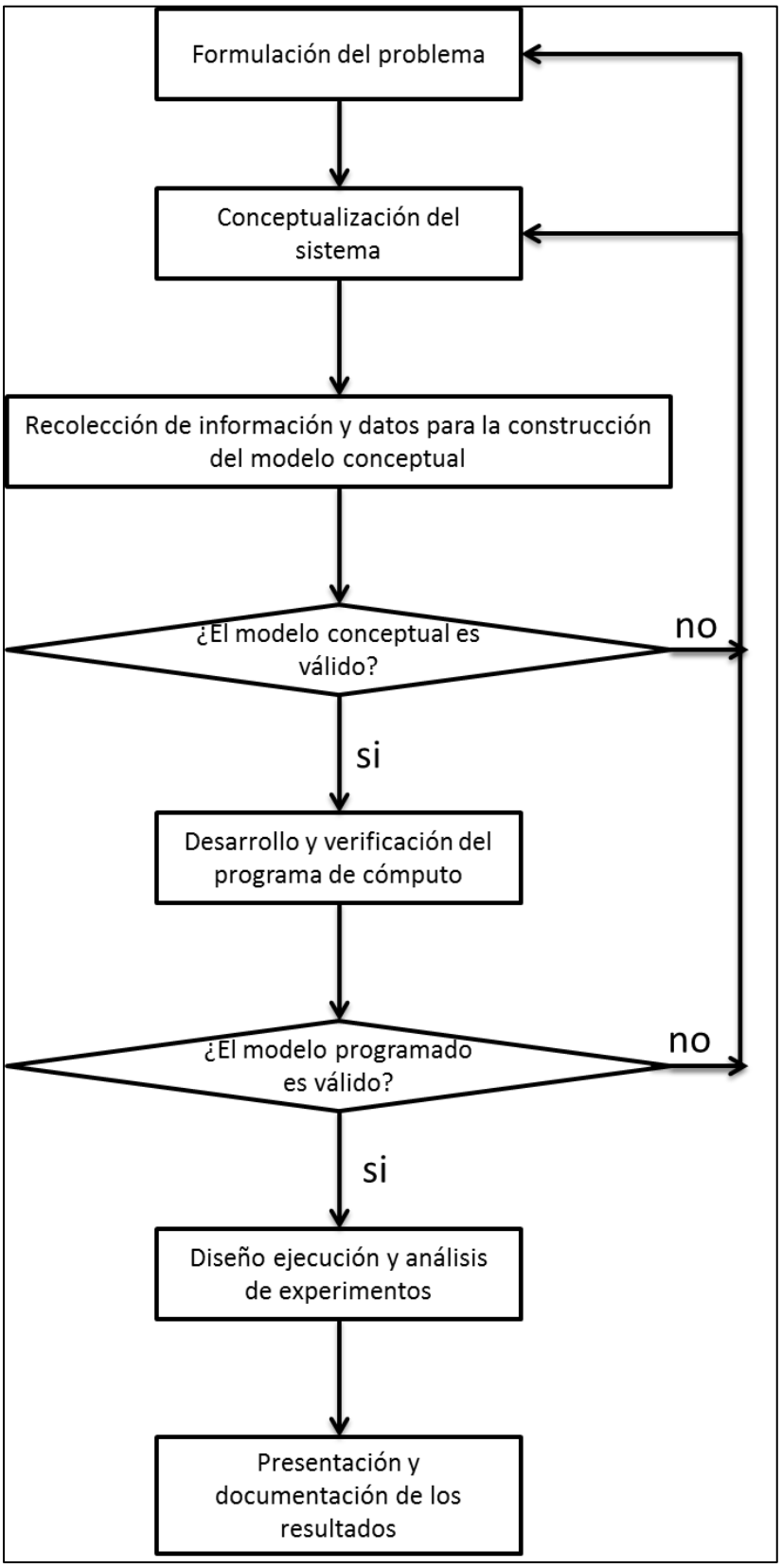
NetLogo es la próxima generación de la serie de lenguajes de modelado multiagente incluyendo StarLogo y StarLogoT. NetLogo se ejecuta en la máquina virtual Java, por lo que funciona en todas las plataformas (Mac, Windows, Linux, y otros). Se ejecuta como una aplicación de escritorio.

2.4. ESTRATEGIA DE INVESTIGACIÓN DE LA TESIS

La metodología que utilizaremos como base, en este trabajo, se apoya en la "metodología de la simulación" de [Flores *et al.* 2007] la cual propone una serie de pasos a seguir para poder hacer un estudio de simulación adecuado, ver figura 18, los pasos que propone dicha metodología son los siguientes:

1. Formulación del problema;
2. Conceptualización del sistema;
3. Recolección de información y datos para la construcción del modelo conceptual;
4. Validación del modelo conceptual;
5. Desarrollo y verificación del programa de cómputo;
6. Validación del modelo;
7. Diseño, ejecución y análisis de experimentos;
8. Presentación y documentación del análisis de la simulación.

Figura 18. Pasos de la metodología de investigación de ésta tesis.

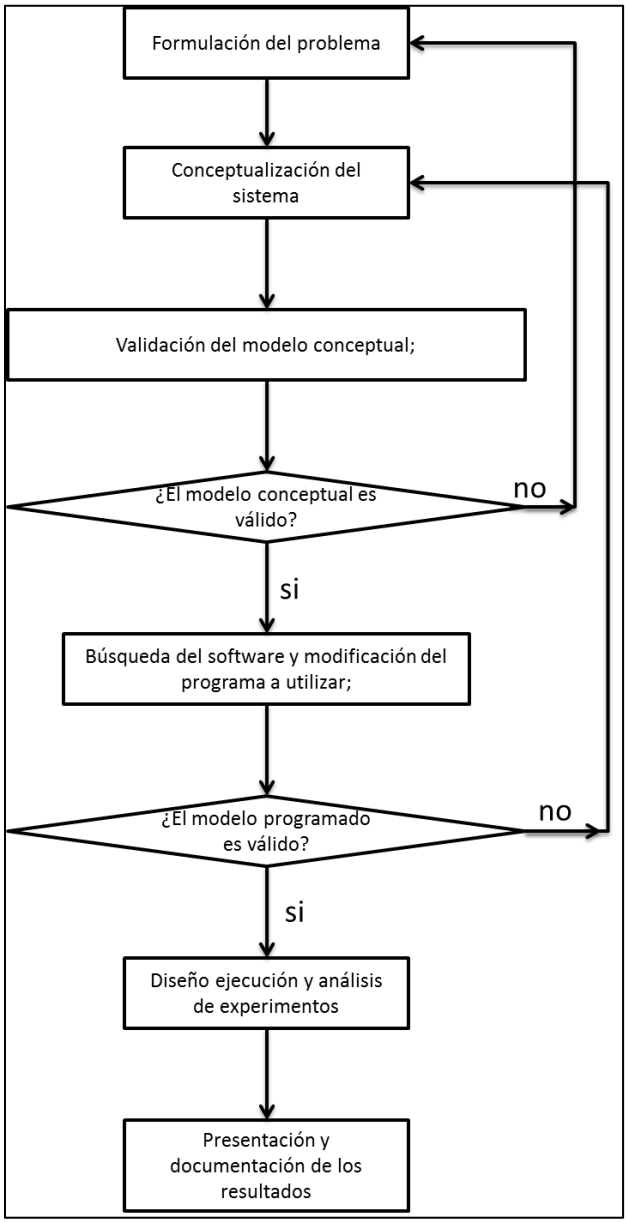


Fuente: elaboración propia, basado en Flores et al. 2007

Para nuestro caso, la metodología que proponemos se muestra en la figura 19 y los pasos a seguir son los siguientes:

1. Formulación del problema;
2. Conceptualización del sistema;
3. Validación del modelo conceptual;
4. Búsqueda del *software* y modificación del programa a utilizar;
5. Validación del programa de cómputo;
6. Diseño, ejecución y análisis de experimentos;
7. Presentación y documentación del análisis de la simulación.

Figura 19. Metodología de investigación propuesta para este trabajo.



Fuente: elaboración propia.

1. Formulación del problema

En específico el caso que pretendemos abordar es el problema de ruteo vehicular capacitado (*CVRP*), que como ya lo definimos en capítulos anteriores, sabemos es un problema de optimización combinatoria cuya dificultad aumenta de manera exponencial, conforme la instancia del problema crece, sabemos también que pertenece a la clase NP-Duro y al no existir un algoritmo que pueda resolver el problema de manera eficiente en un tiempo polinomial, es necesario seguir desarrollando técnicas y herramientas que puedan ayudarnos a resolver este tipo de problemas de manera rápida y eficiente, ya que en la vida real el problema de ruteo afecta a la mayoría de las empresas a nivel mundial.

Por las razones expuestas anteriormente se ha llegado a la definición general de la siguiente problemática:

> En la actualidad es indispensable el traslado de mercancías para cualquier tipo de empresa, la eficiencia o en su defecto ineficiencia en el traslado de las mercancías impacta de manera positiva o negativa, según sea el caso, en las utilidades de las empresas. De tal manera que el problema aquí tratado (*CVRP*) es de evidente relevancia para casos de la vida real y además se sigue estudiando en los círculos académicos dada su naturaleza NP-Duro. Para instancias grandes (más de 50 clientes [Toth and Vigo. 2002]) del problema *CVRP* los métodos exactos ya no nos son de mucha ayuda de aquí que el desarrollo de las computadoras, que cada día son más poderosas para procesar información de manera más eficiente, nos permite utilizar la simulación basada en agentes para tratar de resolver un problema de tipo NP-Duro de una manera ingeniosa y relativamente sencilla ya que se incluye y emula el comportamiento de las hormigas a la hora de buscar comida en la vida real y se hace una analogía con el hecho de que el centro de distribución sea el nido y los clientes sean la "comida".

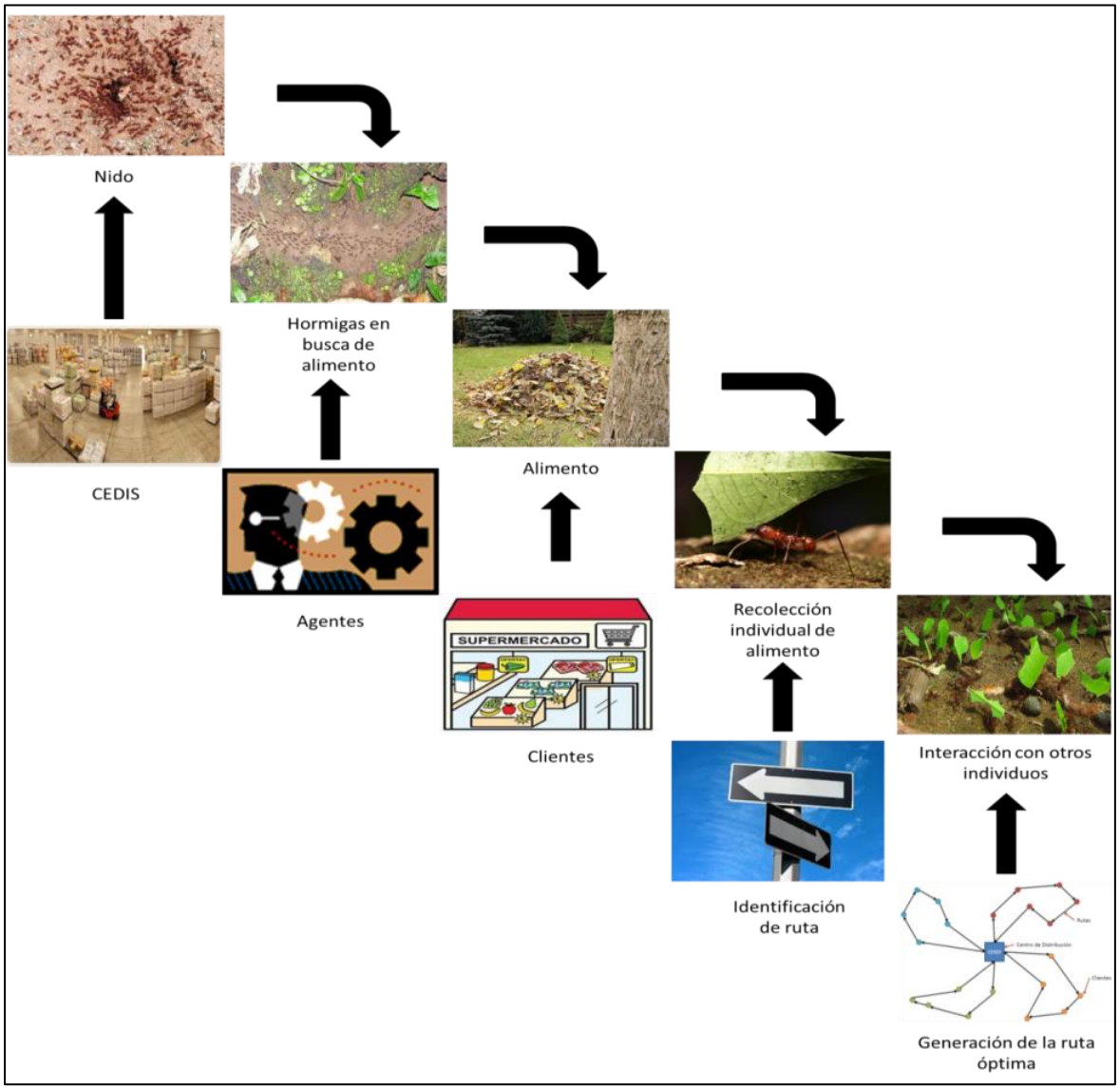
2. Conceptualización del sistema

Básicamente, el modelo que se utilizará simula la forma en que las hormigas exploran y buscan alimento en la vida real, las hormigas son los agentes, el número de agentes puede ser modificado desde uno hasta doscientos, la manera en que los agentes interactúan o se comunican entre sí, es mediante la "feromona", esta puede ser modificada en cuanto a su radio de acción y el tiempo que dura activa. Los puntos distribuidos geográficamente en el espacio virtual, simulan ser la "comida" en la vida real, la "comida" puede distribuirse en el espacio virtual de manera aleatoria o en coordenadas específicas si el usuario así lo requiere, la interpretación que le damos en el modelo a la "comida", es que representan a los clientes que deben ser atendidos en el problema de ruteo vehicular. Una vez que el modelo empieza a correr, los agentes buscan de manera individual la "comida", el lugar o lugares en donde encuentran mayor cantidad de "alimento" es reforzado mediante el

rastro de "feromona" que los demás agentes identifican y siguen, entendiendo que en esa dirección existe una mayor probabilidad de encontrar una fuente de alimento.

Lo que comentamos en el párrafo anterior se puede ver de manera más clara en el siguiente modelo conceptual:

Figura 20. Modelo conceptual del problema.



Fuente: elaboración propia.

3. Validación del modelo conceptual

La validación se llevará a cabo mediante el análisis de modelos y la literatura existente haciendo referencia a los expertos en temas de simulación y problemas de (*CVRP*), apoyándonos en sus estudios, opiniones y experiencia analizando casos similares, para la construcción, corrección y mejora del modelo conceptual.

4. Búsqueda y modificación del software y programa a utilizar

Para este trabajo, se modificará un programa existente en la biblioteca del *software* NetLogo, llamado "*ants*"; las modificaciones se realizarán en función de lo requerido para resolver nuestro problema, estos requerimientos son:

- Número de clientes iniciales;
- Las posiciones de los clientes para las 30 corridas que se realizan para cada problema son fijas;
- Los clientes deben tener una etiqueta indicando su número de cliente o "identificador";
- Los clientes sólo serán de un tipo;
- El programa deberá reportar los resultados en una hoja de Excel® para su respectivo análisis;
- Cada hoja deberá tener un identificador, diferente para cada una.

5. Verificación del programa de cómputo;

Para verificar el programa de cómputo, el *software* cuenta con su propio indicador de errores, tanto de texto como en comandos, por lo que se facilita la tarea de verificación del programa. Por otro lado se realizaron pruebas piloto y de escritorio, con una instancia pequeña de *CVRP* y se analizaron los resultados de esas pruebas, de manera visual y analítica.

6. Validación del modelo programado

Para asegurarnos que el híbrido que estamos proponiendo resuelve *CVRP*, el programa que estamos utilizando no sólo entrega resultados analíticos, también permite ver la simulación de manera gráfica en tiempo real, lo que nos permite observar que los comandos e instrucciones que le estamos ingresando al programa se estén ejecutando de

manera adecuada, por ejemplo, que el número de clientes que ingresamos en la sintaxis del programa sea igual al que observamos en pantalla con su respectivo identificador y en las coordenadas que le indicamos, esto se logra observando la ventana de datos de salida que contiene el mismo NetLogo. En nuestro caso, para validar el programa, trabajamos con una instancia mediana del problema, tomada de la librería TSPLIB, misma que ya fue resuelta mediante métodos exactos, para posteriormente compararlas con la simulación realizada por el programa y validar que la brecha que existe entre ambos métodos no este demasiado alejada del óptimo, a lo más 20% con respecto del óptimo.

7. Diseño, ejecución y análisis de experimentos

El diseño de los experimentos está basado en la literatura existente, analizando trabajos similares y tomando como dato los resultados obtenidos, en nuestro caso tomaremos como referencia el trabajo de Bratton [Bratton *et al.* 2007] y Toth y Vigo [Toth and Vigo. 2002] quienes nos dicen que para problemas relacionados con *Particle Swarm Optimization* (*PSO*) y *Ant Algorithms,* respectivamente, problema similar al que estamos tratando, se ha verificado que con 30 corridas del problema, este tiende a minimizar la brecha entre la simulación y el óptimo.

8. Presentación y documentación de los resultados de la simulación

Los procedimientos y resultados de la simulación serán detallados y asentados en este trabajo de tesis.

Capítulo 3

III. DESARROLLO DE LA ESTRATEGIA DE INVESTIGACIÓN.

En este capítulo, describiremos el desarrollo de los pasos que se plantearon en la metodología de investigación de este trabajo, cada punto será justificado, aclarando el procedimiento e intentando no dejar dudas de lo que se hizo.

3.1. FORMULACIÓN DEL PROBLEMA

La idea de utilizar la simulación basada en agentes o (*ABS*) por sus siglas en Inglés *Agent Based Simulation,* para tratar de resolver un problema de ruteo, en específico un problema de ruteo vehicular capacitado (*CVRP*), surge como parte da la idea de seguir desarrollando técnicas y herramientas que puedan ayudarnos resolver problemas del tipo NP-duro de manera eficiente. De lo que se pudo analizar en la literatura encontrada, logramos deducir que la *ABS* se ajusta de manera adecuada en cuanto a su enfoque en el planteamiento y solución de problemas de optimización, que son tratados por la Investigación de Operaciones. El hecho de poder relacionar el uso de la *ABS* para solucionar problemas complejos surge de la observación, investigación y análisis del comportamiento de los enjambres de distintos insectos y de sus propiedades emergentes cuando actúan de manera colectiva para satisfacer una necesidad, que en la mayoría de los casos es la necesidad de alimentarse [Toth and Vigo. 2002].

Para este trabajo, se tenía conocimiento de que en el *software* llamado NetLogo existe una librería que contiene varios modelos de simulación programados para determinados fines, y con un poco de investigación se observó un programa, dentro de esa librería, llamado "*ants*" que simula la forma en que las hormigas se alimentan, sin llegar a ser propiamente un *Ant Sistem* o un *Ant Colony Optimization.* Con un poco de curiosidad, se comenzó a cuestionar la manera en que se podría utilizar este programa para resolver el problema *CVRP.* El diseño de la estrategia para tratar de resolver este problema de tipo NP-duro utilizando la simulación basada en agentes, consiste esencialmente en aplicar un método de dos fases: primero agrupar y luego rutear (*cluster-first, route-second*); en la primera fase agrupamos un determinado número de clientes, dependiendo de la capacidad de los vehículos, para esto se utilizó una macro diseñada en Excel® que hace este proceso de manera rápida y sencilla, en la segunda fase resolvemos el *TSP* asociado al problema de *CVRP* para lo cual se aplicó la simulación basada en agentes, mediante el programa NetLogo, mismo que indicó las rutas generadas para que cada vehículo visitara todos los clientes, después se reportaron los resultados.

3.2. CONCEPTUALIZACIÓN DEL SISTEMA

Como lo describen Paolo Toth y Daniele Vigo [Toth and Vigo. 2002], la conceptualización consiste en hacer referencia a la analogía de la búsqueda de alimento que realizan algunas especies de hormigas, lo cual se describe de la siguiente manera: las hormigas salen del hormiguero o nido y comienzan una búsqueda de manera aleatoria hasta encontrar comida. En su camino van dejando un rastro de "feromona", de manera que cuando alguna hormiga encuentra un lugar prometedor donde exista alimento, regresa al nido y refuerza el camino con más feromona, si el lugar se encuentra cercano al nido la feromona no se ha evaporado del todo, por lo cual el rastro es más fuerte, y por lo tanto las otras hormigas tienden a seguir ese rastro más marcado de feromona y de esta manera optimizan la búsqueda de alimento. En nuestra conceptualización del sistema, los agentes representan a las hormigas que en el espacio virtual identificamos por dibujos de pequeñas hormigas rojas. En el centro de nuestro universo virtual tenemos un hormiguero o nido, del cual salen las hormigas (agentes), los puntos en el espacio virtual representan la comida que a su vez representa a los clientes ubicados geográficamente de manera determinista (se pueden modificar sus coordenadas), los agentes se mueven de manera aleatoria en el espacio virtual y cuando encuentran comida regresan al nido y dejan un rastro de "feromona" que los demás agentes encuentran, y así, toman ese camino ya que es la forma en que los agentes indican que en esa dirección existe alimento.

Atendiendo a que nosotros utilizaremos un híbrido entre un algoritmo de dos fases que consiste en primero agrupar y luego rutear (*cluster-first, route-second*) y una heurística basada en *Ant Sistem*, primero desarrollamos una macro en Excel® agrupar a los clientes en función de la capacidad dada para cada ejemplo, una vez hecho esto, la segunda fase consiste en ingresar al programa NetLogo las coordenadas de los clientes en el orden en que fueron agrupados, para hacer las corridas correspondientes.

En cada corrida, los agentes en el modelo ubican y reportan la coordenada y orden en que encontraron la "comida", que representa a los clientes en el modelo; los clientes tienen una etiqueta que los identifica y de esta manera es más fácil hacer el análisis de los resultados. Cada que se corre el modelo, se resuelve un *TSP* y es comparado con el *TSP* óptimo de las instancias medianas, mismo que se obtiene previamente con un *software* llamado Lingo[6]®. El programa se corre treinta veces y se ha visto que el resultado tiende al óptimo, observando un *GAP*[7] de menos del 20%, lo cual consideramos bueno para este método sin llegar a ser concluyente todavía.

[6] Recordar que lingo utiliza por defecto Branch and Cut para resolver de manera exacta los problemas de PLE. [Maroto *et al. 2002*].

[7] GAP: Distancia o diferencia que existe entre elementos relacionados entre sí.

3.3. VALIDACIÓN DEL MODELO CONCEPTUAL

El proceso de búsqueda de alimento de las colonias de hormigas, en general, conduce a un procedimiento eficiente para la adquisición de alimento [Toth and Vigo. 2002] esta observación le permitió a Colorni, Dorigo y Maniezzo proponer una nueva clase de metaheurística para resolver problemas combinatorios basados en la siguiente correspondencia: las hormigas artificiales que buscan en el espacio de soluciones, simulan hormigas reales que exploran su entorno, los valores de la función objetivo se asocian con la calidad de las fuentes de comida, y los valores registrados en una memoria adaptativa, imitan los rastros de feromona [Toth and Vigo. 2002].

De lo anterior, que está registrado en literatura, se deriva en gran medida el trabajo que se presenta en este documento. La necesidad de búsqueda de comida es un problema de tipo combinatorio, por lo tanto es fácil relacionar esta conducta con la solución de problemas de tipo NP-duro, que es lo que estamos pretendiendo hacer en este trabajo de tesis, además, si a esto le agregamos que actualmente tenemos la opción de representar este tipo de conductas de enjambre con la simulación basada en agentes (*ABS*), ahora nuestro trabajo se enfoca en analizar qué tan bien se apega este tipo de simulación, de la cual no existe mucha información puntual registrada en la literatura, a la solución de problemas de optimización combinatoria, que en nuestro caso en un problema de ruteo vehicular capacitado (*CVRP*).

Por lo anterior, se valida el modelo conceptual aquí propuesto y que nos ayudará a desarrollar este trabajo de tesis.

3.4. BÚSQUEDA Y MODIFICACIÓN DEL *SOFTWARE* Y PROGRAMA A UTILIZAR

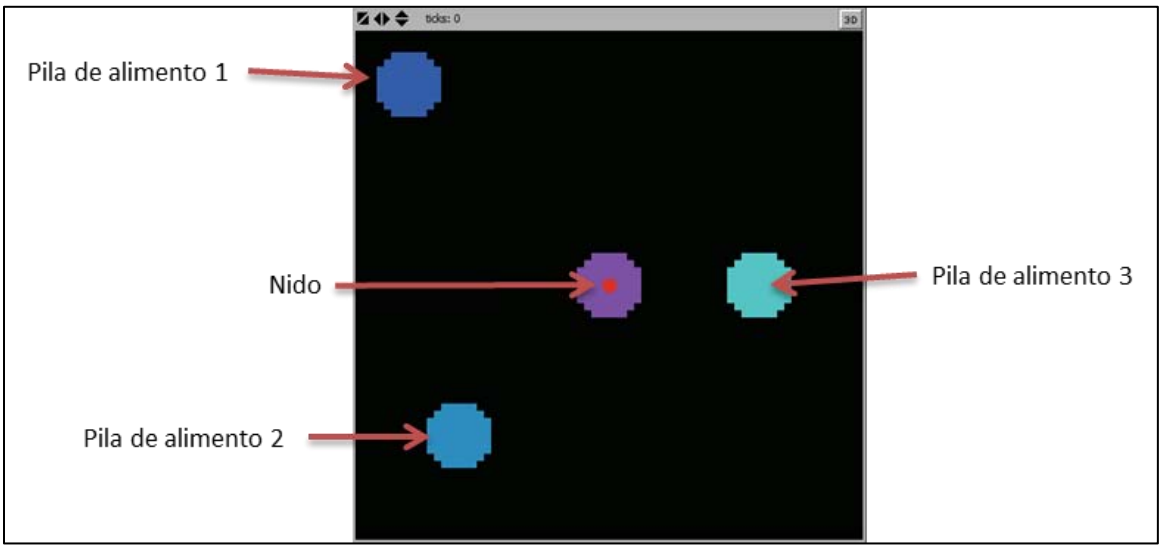
Las herramientas que actualmente existen para trabajar con la simulación basada en agentes, no son muy conocidas si las comparamos con las que utilizan procedimientos exactos. Además, el *software* comercial que existe en el mercado no es muy flexible y es hasta cierto punto limitante, por estas razones entre algunas otras, se decidió trabajar con un *software* libre llamado Netlogo. Este *software* permite buscar en su biblioteca una gran cantidad de programas y al elegir alguno de nuestro interés, este puede ser modificado hasta adecuarse al problema particular de estudio. NetLogo es muy flexible, fácil de entender y manipular, además es gratuito y está orientado a la ABS.

El desarrollo del programa de cómputo que se utiliza en este trabajo está basado en uno ya existente dentro de la librería del *software* NetLogo, este programa se llama "*Ants*" y consiste en la simulación de la manera en que las hormigas salen de su nido, buscan y recolectan alimento. Este programa fue modificado para retomar dicho comportamiento y

de esta forma, poder resolver un problema *CVRP* aplicando la heurística de primero agrupar y luego rutear (*Cluster-First, Route-Second*).

En primer lugar, el programa preestablecido consiste de tres pilas de alimento, ubicadas en diferentes lugares, dentro del espacio virtual, y un nido ubicado al centro del plano, como se muestra en la figura 21.

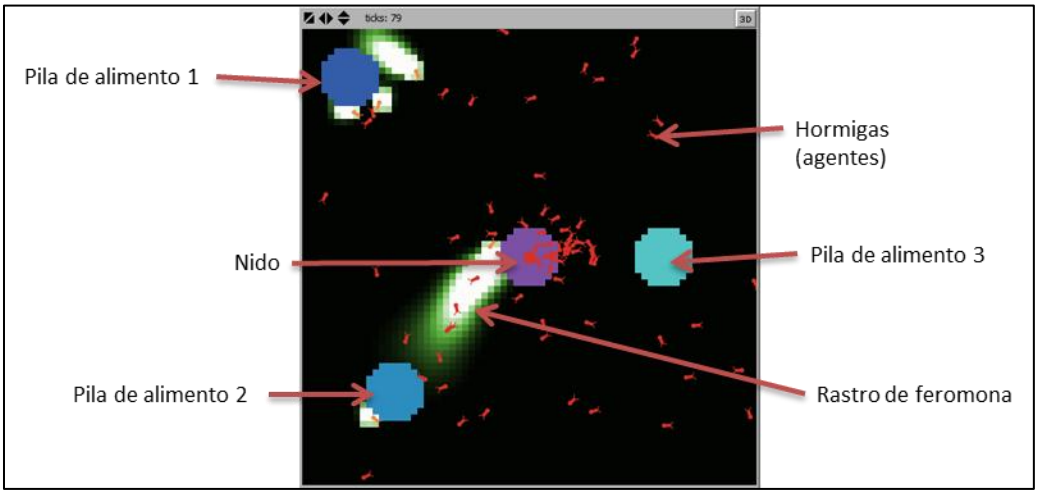
Figura 21. Programa "*Ants*" original.



Fuente: Netlogo.

Del nido salen las hormigas y comienzan su búsqueda de alimento en el espacio virtual, las hormigas se mueven de manera aleatoria, ver figura 22.

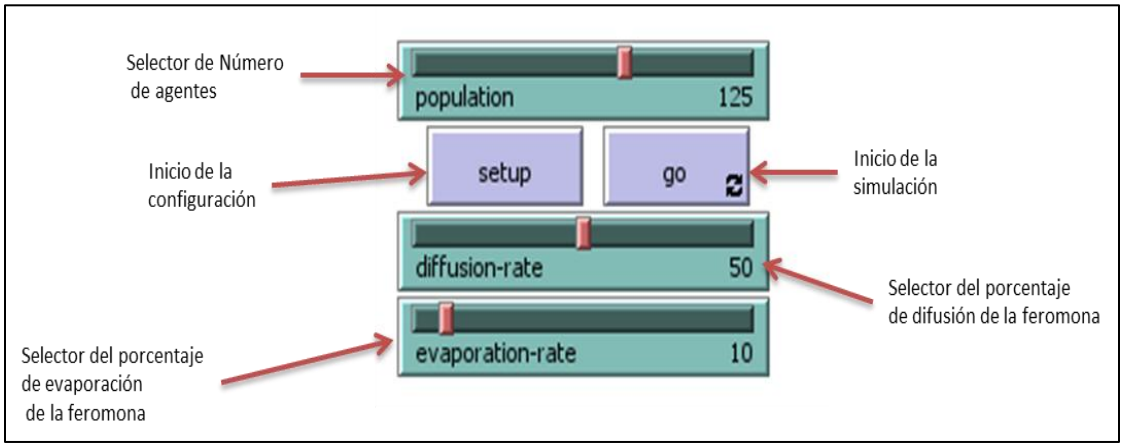
Figura 22. Búsqueda aleatoria de alimento.



Fuente: Netlogo.

El usuario puede elegir el número de agentes, el radio de acción de la feromona y el tiempo que ésta dura activa en el ambiente, ver figura 23.

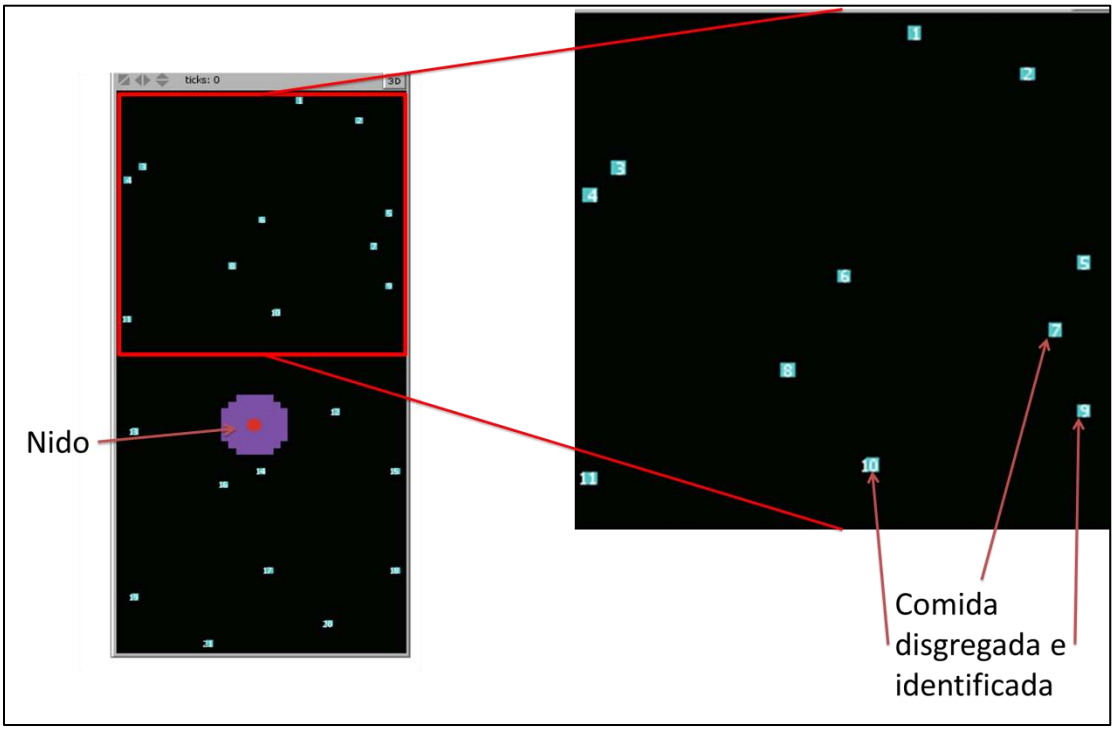
Figura 23. Selección de parámetros.



Fuente: Netlogo.

Lo que se hizo fue; disgregar las pilas de comida en unidades, cada unidad se colocó en un lugar predeterminado y se le asignó un identificador, ver figura 24.

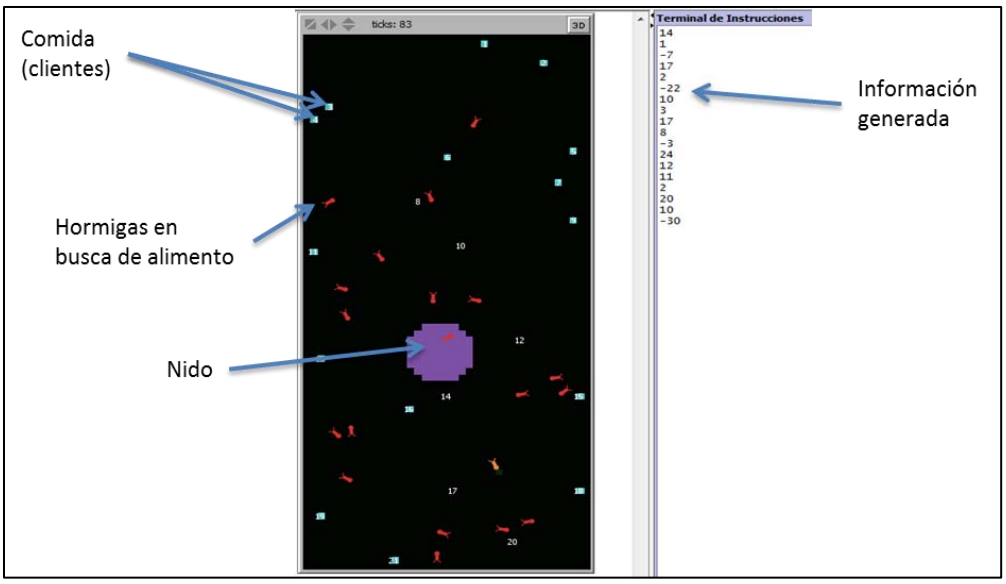
Figura 24. Pilas de comida disgregadas e identificadas.



Fuente: Elaboración propia.

Cuando una hormiga hace contacto con alguna unidad de alimento "recolección", la información generada se guarda en un archivo adjunto con la ubicación en R^2, el número de cliente del que se trata, y el orden en que fue encontrado, figura 25.

Figura 25. Del lado izquierdo observamos la simulación, del lado derecho la información generada por la simulación.



Fuente: elaboración propia.

El programa, ya modificado, nos permite ingresar el número de clientes que necesitemos mediante la definición de sus coordenadas cartesianas en el plano, a cada cliente se le asigna una etiqueta de identificación que va desde uno hasta el número de clientes a ingresar en el programa, además, nos permite seleccionar el número de agentes (hormigas) que estarán en el espacio virtual y la manera en que estas interaccionan, mediante el radio de difusión de la feromona y su porcentaje de evaporación en el tiempo. Una vez que la corrida termina, los datos recabados se almacenan en un archivo que después puede ser exportado a una hoja de Excel® para poder ser analizados.

El código en Java del programa final ya modificado se muestra en el anexo 1.

3.5. VERIFICACIÓN Y VALIDACIÓN DEL PROGRAMA DE CÓMPUTO

Para realizar la verificación del programa de cómputo, éste se hizo correr en varias ocasiones para observar que no se tuviera ningún error de sintaxis en el código. Cabe mencionar, que el compilador de NetLogo tiene un corrector automático que no permite que el programa corra si existe algún error de sintaxis en la escritura de las sentencias o de algún otro tipo, lo cual facilita la tarea de verificación del programa.

Por otra parte, para la validación del programa, se tomó una instancia mediana de la sub-librería de TSP-LIB llamada *CVRP data*, el nombre de la instancia es "eil22", que consta de

21 clientes y un centro de distribución (CEDIS); la demanda por cliente es variable, y la capacidad de los vehículos es homogénea, en este caso, de 6000 (unidades de carga), el número de vehículos no es una restricción. Cabe señalar, que a los datos que están en los archivos de la librería de TSPLIB se les da un tratamiento previo, que consiste en verificar cuál es el valor de las coordenadas del CEDIS, tanto en el eje *x* como en el eje *y*, para poder restar ese valor a todas las coordenadas, la explicación es la siguiente: i) con la operación mencionada se obliga a que las coordenadas del CEDIS sean las mismas que las del "nido" y ii) la malla que utiliza el programa NetLogo para desarrollar su espacio virtual está compuesta, como ya se mencionó en el capítulo anterior, de "parcelas"; estas parcelas son cuadradas y unitarias, de tal manera que el número de parcelas aumenta conforme las coordenadas de los clientes son mayores.

Dicho lo anterior, podemos comentar que para todos los ejercicios decidimos restar el valor de las coordenadas *x* e *y* del CEDIS para disminuir en gran medida los valores de las coordenadas que tienen los clientes que conforman las instancias que estamos utilizando. Para este ejercicio, se tomaron los valores de la pareja ordenada del cliente[8] 0(145, 215) y se le restaron dichos valores a cada una de las coordenadas de los clientes que integran este ejercicio, quedando como resultado los valores que se encuentran en la tabla 4:

Tabla 4. Datos para el ejercicio de calibración.

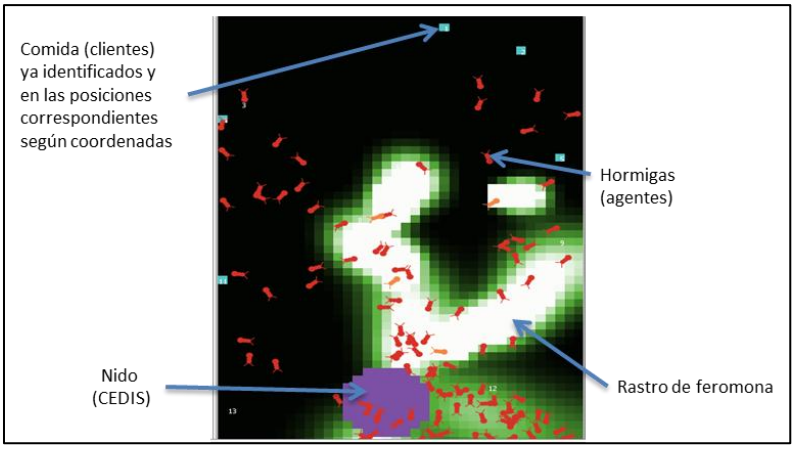
Cliente	x	y	Demanda
0	0	0	0
1	6	49	1100
2	14	46	700
3	-15	39	800
4	-17	37	1400
5	18	32	2100
6	1	31	400
7	16	27	800
8	-3	24	100
9	18	21	500
10	3	17	600
11	-17	16	1200
12	11	2	1300
13	-16	-1	1300
14	1	-7	300
15	19	-7	900
16	-4	-9	2100
17	2	-22	1000
18	19	-22	900
19	-16	-26	2500
20	10	-30	1800
21	-6	-33	700

Fuente: elaboración propia.

[8] El cliente 0 indica el centro de distribución en todos los casos usados en este trabajo.

Una vez pre-tratados los datos y ya ingresados en el programa, se analizó el comportamiento de la simulación para ver que todo estuviera funcionando de acuerdo con lo esperado, y que el programa realizara las tareas para las cuales fue diseñado. En efecto, se pudo comprobar que los agentes buscan de manera aleatoria, ver figura 26, la comida (clientes).

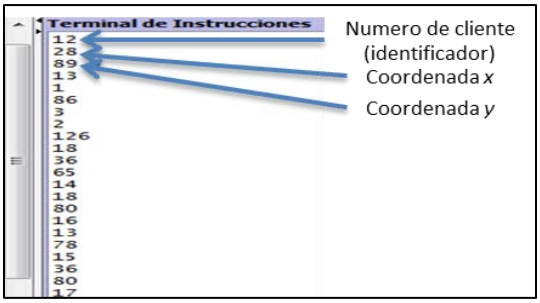
Figura 26. Búsqueda aleatoria de la "comida".



Fuente: elaboración propia.

Cada que un agente se encuentra con un cliente (recolecta comida), este reporta el orden de visita y las coordenadas de su ubicación, ver figura 27, mismas que son guardadas en un archivo adjunto de Excel® que después es analizado con una macro, diseñada específicamente para este problema. La macro empleada para el análisis de los resultados de la simulación, reporta el porcentaje de error comparado con el valor óptimo obtenido anteriormente con el *software* Lingo9®.

Figura 27. Reporte de orden de visita y coordenadas.



Fuente: elaboración propia.

Los resultados preliminares de la simulación, basados en la observación y análisis de la información, nos ayudan a verificar y validar el programa de cómputo, dándonos la certeza de que la utilización del programa sirve a los fines y propósitos para los que fue diseñado, generando la confianza de poderlo utilizar para el desarrollo de las pruebas en este trabajo.

Capítulo 4

IV. DISEÑO, EJECUCIÓN Y ANÁLISIS DE EXPERIMENTOS

Para el desarrollo de este capítulo se decidió, con base en [Bratton *et al.* 2007] y [Toth and Vigo. 2002], trabajar de la siguiente manera: se utilizarán las instancias existentes en la librería *TSPLIB* y COIN-OR diseñadas para el análisis y comparación de herramientas que puedan solucionar de manera más eficiente los problemas de tipo *CVRP*. En la librería de *TSPLIB*, existe una sub-librería especializada en problemas de *CVRP* que es utilizada de la misma manera que para los problemas de *TSP*, en la cual existen instancias ya trabajadas del problema *CVRP* que nos ayudan a probar las herramientas que se proponen para la solución de algunas instancias. Por otro lado, la librería para comparar CVRP que se encuentra en COIN-OR funciona de la misma manera, cabe aclarar que a gusto personal, la librería de COIN-OR es más amigable y contiene más información que la librería TSPLIB. Con base en lo anterior se seleccionaron 5 instancias que cumplen con las características que nosotros necesitamos para ingresar a nuestra simulación, las características son:

- Sólo un centro de distribución;
- Que mostraran las coordenadas cartesianas en R^2 para poder obtener las distancias euclidianas;
- Que los vehículos fueran homogéneos;
- Que nos indicaran la capacidad de los vehículos.

Las instancias adecuadas para nuestro análisis, y fueron las siguientes:

Tabla 5. Instancias seleccionadas para el análisis de las corridas.

Nombre de la Instancia	No. de Clientes	No. de CEDIS	Tipo de coordenadas	Procedencia
eil22	21	1	Cartesianas R^2	TSPLIB
eil23	22	1	Cartesianas R^2	TSPLIB
eil30	29	1	Cartesianas R^2	TSPLIB
eil33	32	1	Cartesianas R^2	TSPLIB
An54k7	53	1	Cartesianas R^2	COIN-OR

Fuente: elaboración propia con base en TSPLIB y COIN-OR

Es importante recordar que el tamaño de las instancias (medianas o grandes) ya lo definen Toth y Vigo, para mayores referencias dirigirse a Toth and Vigo 2002.

Para cada instancia, el proceso a seguir fue el siguiente:

1. Primero se pre-tratan los datos, como ya se indicó en el capítulo anterior.
2. Se corrió una macro que agrupó (*clustering*), a los vehículos en función de la capacidad de cada uno de ellos y de la demanda de los clientes.
3. Ya agrupados los vehículos se procedió a ingresar sus coordenadas en el programa modificado de NetLogo.
4. Se realizaron 30 corridas para cada uno de los vehículos asignados y con eso se generaron las rutas que deberían seguir.
5. Se analizaron esas rutas con otra macro diseñada en Excel®, que compara las rutas obtenidas por el *software,* con el resultado óptimo obtenido en Lingo®, se selecciona la ruta con el menor porcentaje de error y esa es la que utilizamos como solución del problema.

Cabe destacar que en la mayoría de los caso obtuvimos porcentajes de error del 0% con respecto del óptimo.

EJECUCIÓN DE EXPERIMENTOS Y ANÁLISIS DE RESULTADOS

En este punto ya con los antecedentes de la literatura revisada [Bratton *et al.* 2007][Toth and Vigo. 2002], se decidió realizar un total de treinta corridas por cada instancia y analizar los resultados con las macros diseñadas para tal efecto. En función de los resultados obtenidos, se fueron modificando los parámetros (*tuning*) y se observó si estos mejoraban o empeoraban, de tal manera que se pudiera evaluar el comportamiento y reportar cuáles son los mejores parámetros para cada tipo de problema.

A continuación, se presentan los resultados obtenidos para cada instancia del problema, para cada una se realizó un análisis de las tabas y gráficas obtenidas en el proceso, y se hicieron los comentarios correspondientes.

INSTANCIA EIL22

Para la instancia eil22 se realizó el pretratamiento de los datos obteniendo las coordenadas de los clientes y sus respectivas demandas, ya con la información de la capacidad de los vehículos se corrió la macro correspondiente en Excel® y nos dio el número de camiones que se deberían de utilizar y por lo tanto el número de clientes en cada ruta, dándonos como resultado los datos de la tabla 6:

Tabla 6. Resultados de la aplicación de la primera fase del algoritmo a la instancia eil 22.

Eil 22						
Coordenadas		CLIENTE	DEMANDA	RUTA	Capacidad de los vehículos	Total de camiones
X	Y				6000	5
6	49	1	1100	1		
14	46	2	700	1		
-15	39	3	800	1		
-17	37	4	1400	1		
18	32	5	2100	2		
1	31	6	400	2		
16	27	7	800	2		
-3	24	8	100	2		
18	21	9	500	2		
3	17	10	600	2		
-17	16	11	1200	2		
11	2	12	1300	3		
-16	-1	13	1300	3		
1	-7	14	300	3		
19	-7	15	900	3		
-4	-9	16	2100	3		
2	-22	17	1000	4		
19	-22	18	900	4		
-16	-26	19	2500	4		
10	-30	20	1800	5		
-6	-33	21	700	5		

Fuente: elaboración propia

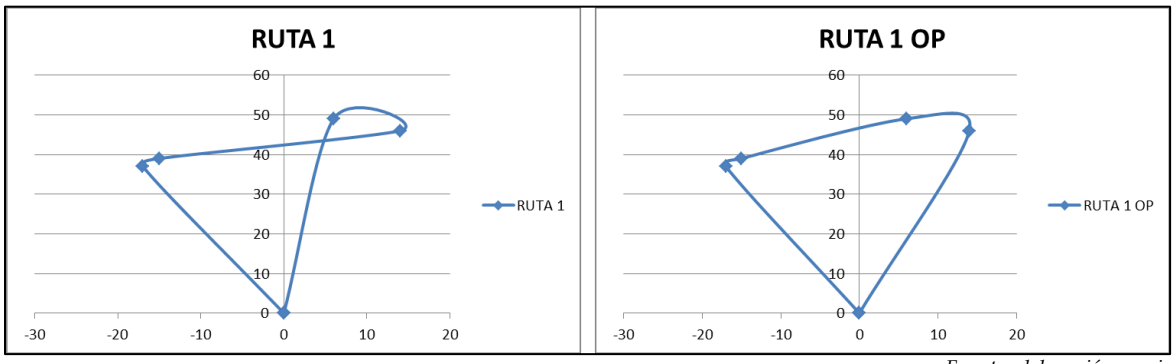
De la tabla 6 podemos observar lo siguiente:

i) El número de camiones a utilizar son 5, con lo cual se generan 5 rutas.
ii) Los clientes que se encuentran en cada ruta se identifican con la columna "RUTA", es decir: los clientes 1, 2, 3, 4 pertenecen a la ruta 1, y así sucesivamente.

Una vez obtenidos los datos como resultado de la aplicación de la primera fase del algoritmo (primero agrupar), se toman las coordenadas de los clientes por grupos, es decir, para este caso tenemos 5 grupos como se observa en la tabla 6, cada grupo se formó debido a la capacidad de los camiones, y cada grupo pertenece a una ruta. Cada grupo de coordenadas es ingresado, en orden, al programa NetLogo, mismo que realizará la simulación con las coordenadas que le estamos ingresando, esto es parte de la segunda

fase del algoritmo que consiste en "después rutear", ya ingresadas las coordenadas en el programa, se realizan las 30 corridas de la simulación para cada ruta y obtenemos las rutas ya optimizadas mismas que presentan un cambio en cuanto al orden en el que deben visitarse los clientes en cada ruta, para ilustrar lo que estamos diciendo, presentamos las siguientes gráficas:

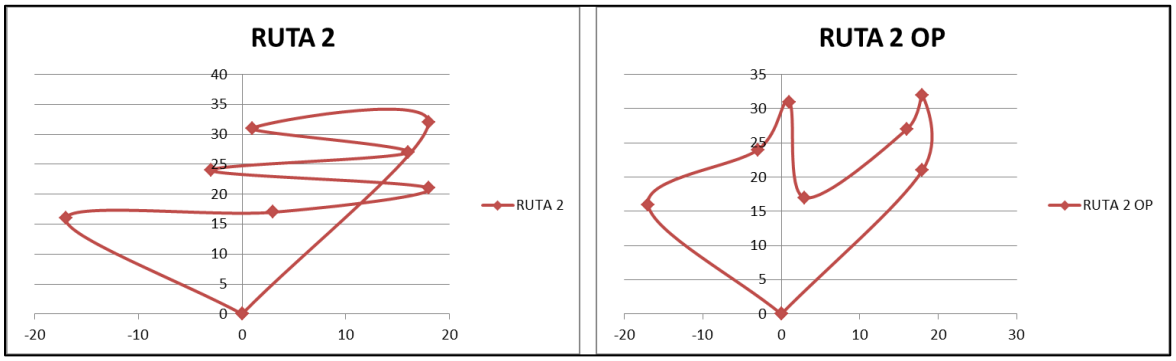
Gráfica 1. Instancia eil 22, Ruta 1 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 1, ver (gráfica 1) se realizaron 30 corridas de la simulación, con los parámetros siguientes: No. de agentes 20, radio de acción de la feromona 0, duración de la feromona 99. Para este caso, el óptimo del *TSP*, obtenido previamente con Lingo®, nos dio un resultado de 123.43, al analizar los datos de la simulación con la macro en Excel®, obtuvimos un valor de 123.43, lo cual nos da un *GAP* de 0%.

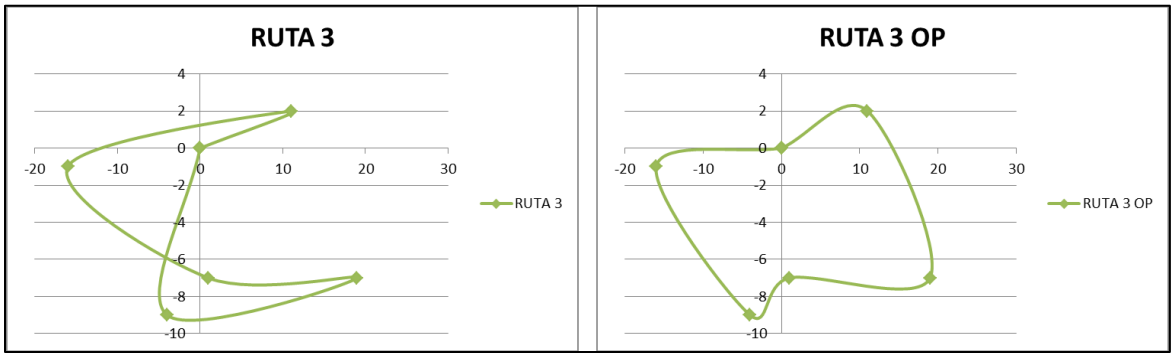
Gráfica 2. Instancia eil 22, Ruta 2 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 2, ver (gráfica 2) se realizaron 30 corridas de la simulación, con los parámetros siguientes: No. de agentes 20, Radio de acción de la feromona 0, duración de la feromona 99. Para este caso, el óptimo del *TSP*, obtenido previamente con Lingo®, nos dio un resultado de 109.058, al analizar los datos obtenidos de la simulación con la macro en Excel®, obtuvimos un valor de 122.12, lo cual nos da un *GAP* de 11.98%.

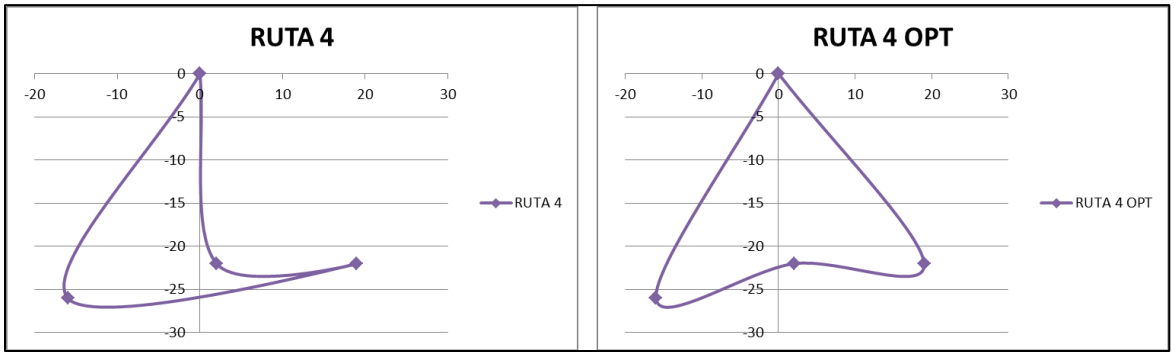
Gráfica 3. Instancia eil 22, Ruta 3 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 3, ver (gráfica 3) se realizaron 30 corridas de la simulación, con los parámetros siguientes: No. de agentes 20, Radio de acción de la feromona 0, duración de la feromona 99. Para este caso, el óptimo del *TSP*, obtenido previamente con Lingo®, nos dio un resultado de 77.06, al analizar los datos obtenidos de la simulación con la macro en Excel®, obtuvimos un valor de 77.06, lo cual nos da un *GAP* de 0%.

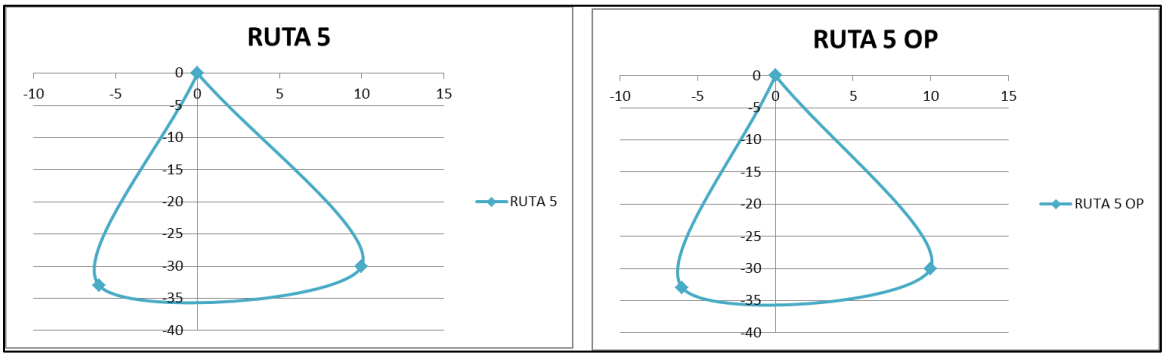
Gráfica 4. Instancia eil 22, Ruta 4 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 4, ver (gráfica 4) se realizaron 30 corridas de la simulación, con los parámetros siguientes: No. de agentes 20, Radio de acción de la feromona 0, duración de la feromona 99. Para este caso, el óptimo del *TSP*, obtenido previamente con Lingo®, nos dio un resultado de 95.03, al analizar los datos obtenidos de la simulación con la macro en Excel®, obtuvimos un valor de 95.03, lo cual nos da un *GAP* de 0%.

Gráfica 5. Instancia eil 22, Ruta 5 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 5, ver (gráfica 5) se realizaron 30 corridas de la simulación, con los parámetros siguientes: No. de agentes 20, Radio de acción de la feromona 0, duración de la feromona 99. Para este caso, el óptimo del *TSP*, obtenido previamente con Lingo®, nos dio un resultado de 81.44, al analizar los datos obtenidos de la simulación con la macro en Excel®, obtuvimos un valor de 81.44, lo cual nos da un *GAP* de 0%.

Los parámetros utilizados en la instancia eil 22 los presentamos en la tabla 7 que se muestra a continuación:

Tabla 7. Resultados de la simulación para la instancia eil 22.

Instancia eil 22						
Ruta	**No. de agentes**	**Radio de acción de la feromona**	**Duración de la feromona**	**Resultado de la simulación**	**Óptimo**	**GAP**
1	20	0	99	123.43	123.43	0.00%
2	20	0	99	122.12	109.058	11.98%
3	20	0	99	77.06	77.06	0.00%
4	20	0	99	95.03	95.03	0.00%
5	20	0	99	81.44	81.44	0.00%

Fuente: elaboración propia.

Ya para finalizar podemos observar que para esta instancia tenemos un *GAP* promedio de 2.40%, el cual está muy por debajo del que habíamos planteado como objetivo, y que es de menos del 20%, por lo que hasta el momento estamos cumpliendo con el objetivo propuesto para esta tesis.

Es importante señalar que el procedimiento descrito para la instancia Eil22 es el mismo para las demás instancias, por lo que en las siguientes sólo daremos a conocer las tablas y gráficas, esto con el fin de evitar ser repetitivos en las todas las imágenes, en el caso de existir alguna particularidad, se dará la explicación correspondiente.

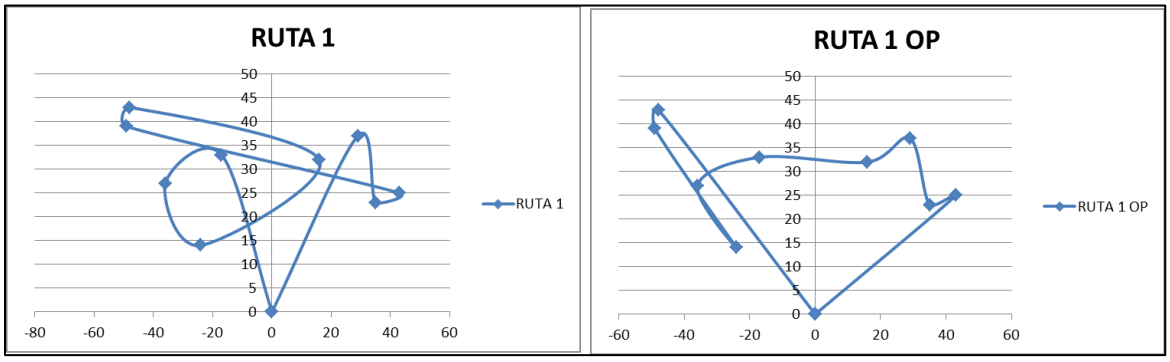
INSTANCIA EIL23

Tabla 8. Resultados de la aplicación de la primera fase del algoritmo a la instancia eil 23.

Eil 23						
Coordenadas		CLIENTE	DEMANDA	RUTA	Capacidad de los vehículos	TOTAL DE CAMIONES
X	Y				4500	3
29	37	1	125	1		
35	23	2	84	1		
43	25	3	60	1		
-49	39	4	500	1		
-48	43	5	300	1		
16	32	6	175	1		
-24	14	7	350	1		
-36	27	8	150	1		
-17	33	9	1100	1		
-10	32	10	4100	2		
-1	22	11	225	2		
1	7	12	300	3		
-7	30	13	250	3		
49	-2	14	500	3		
63	17	15	150	3		
52	17	16	100	3		
63	-11	17	250	3		
1	-22	18	120	3		
9	-43	19	600	3		
37	-34	20	500	3		
-58	-18	21	175	3		
60	-54	22	75	3		

Fuente: elaboración propia

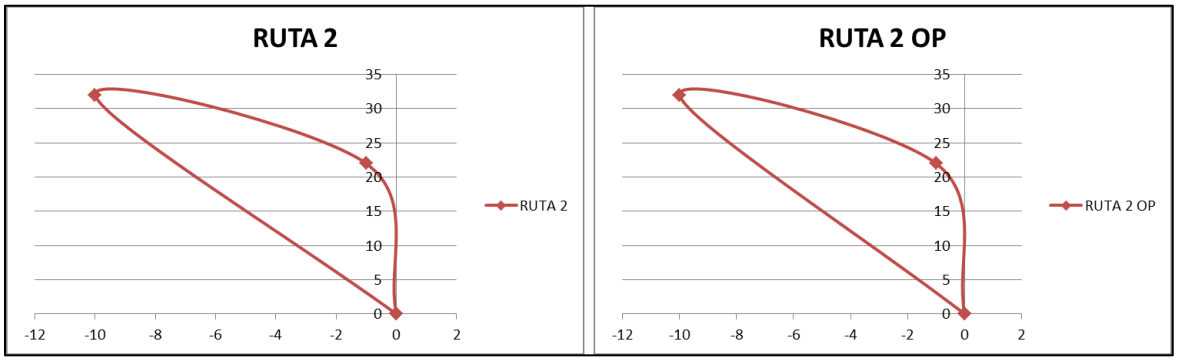
Gráfica 6. Instancia eil 23, Ruta 1 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 1, ver (gráfica 6) el óptimo del *TSP*, nos dio un resultado de 215.37, al analizar los datos de la simulación, obtuvimos un valor de 261.70, lo cual nos da un *GAP* de 21.51%.

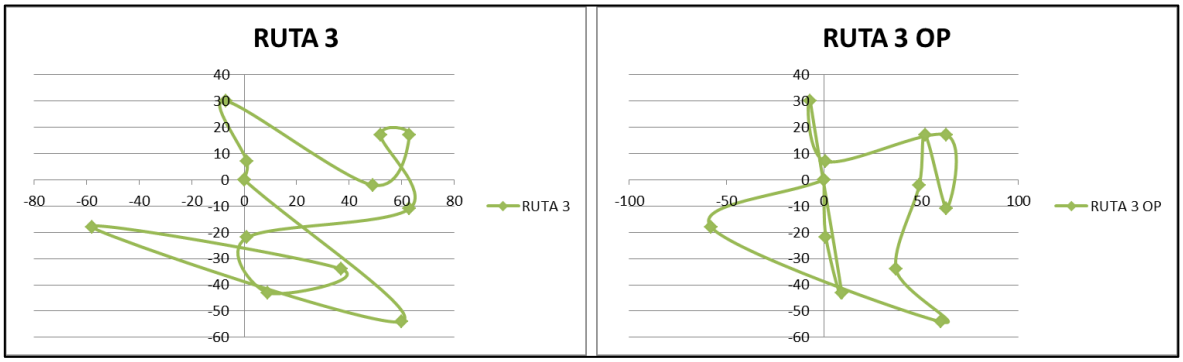
Gráfica 7. Instancia eil 23, Ruta 2 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 2, ver (gráfica 7) el óptimo del *TSP*, nos dio un resultado de 69, al analizar los datos obtenidos de la simulación, obtuvimos un valor de 69, lo cual nos da un *GAP* de 0%.

Gráfica 8. Instancia eil 23, Ruta 3 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 3, ver (gráfica 8) el óptimo del *TSP*, nos dio un resultado de 388.41, al analizar los datos obtenidos de la simulación, obtuvimos un valor de 532.47, lo cual nos da un *GAP* de 37.09%.

Los parámetros utilizados en la instancia eil 23 los presentamos en la tabla 9 que se muestra a continuación:

Tabla 9. Resultados de la simulación para la instancia eil 23.

Instancia eil 23						
Ruta	**No. de agentes**	**Radio de acción de la feromona**	**Duración de la feromona**	**Resultado de la simulación**	**Óptimo**	**GAP**
1	200	65	2	261.70	215.37	21.51%
2	200	65	2	69.00	69.00	0%
3	200	65	2	532.47	388.41	37.09%

Fuente: elaboración propia.

GAP promedio igual a 19.53%.

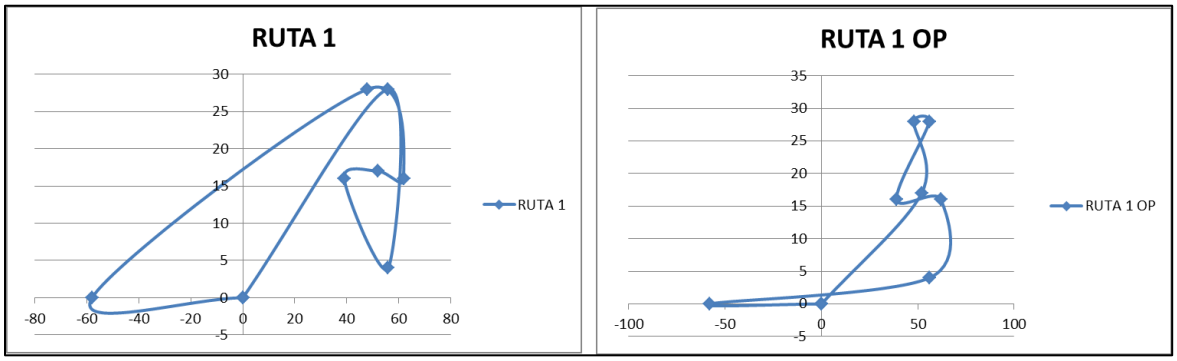
INSTANCIA EIL30

Tabla 10. Resultados de la aplicación de la primera fase del algoritmo a la instancia eil 30.

Eil 30						
Coordenadas		**CLIENTE**	**DEMANDA**	**RUTA**	**Capacidad de los vehículos**	**TOTAL DE CAMIONES**
X	Y				4500	4
56	28	1	300	1		
56	4	2	3100	1		
39	16	3	125	1		
52	17	4	100	1		
62	16	5	200	1		
48	28	6	150	1		
-58	0	7	150	1		
-36	-16	8	450	2		
-43	-14	9	300	2		
-33	-5	10	100	2		
-36	-7	11	950	2		
-37	-8	12	125	2		
-46	1	13	150	2		
-36	-19	14	150	2		
-37	1	15	550	2		
-43	3	16	150	2		
-47	-13	17	100	2		
-9	-3	18	150	2		
13	9	19	400	2		
18	6	20	300	2		
-3	-23	21	1500	3		
26	3	22	100	3		
-10	-5	23	300	3		
53	35	24	500	3		
50	40	25	800	3		
26	39	26	300	3		
45	52	27	100	3		
22	56	28	150	3		
45	38	29	1000	4		

Fuente: elaboración propia

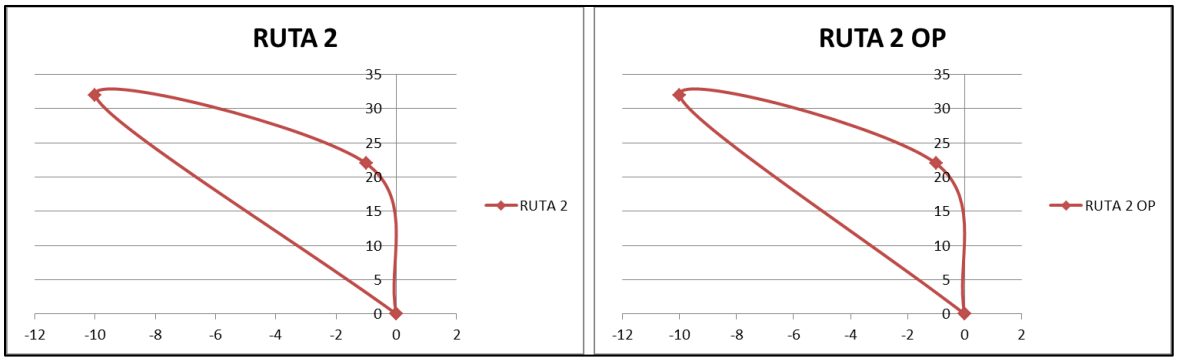
Gráfica 9. Instancia eil 30, Ruta 1 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 1, ver (gráfica 9) el óptimo del *TSP*, nos dio un resultado de 270.62, al analizar los datos de la simulación, obtuvimos un valor de 303.72, lo cual nos da un *GAP* de 12.23%.

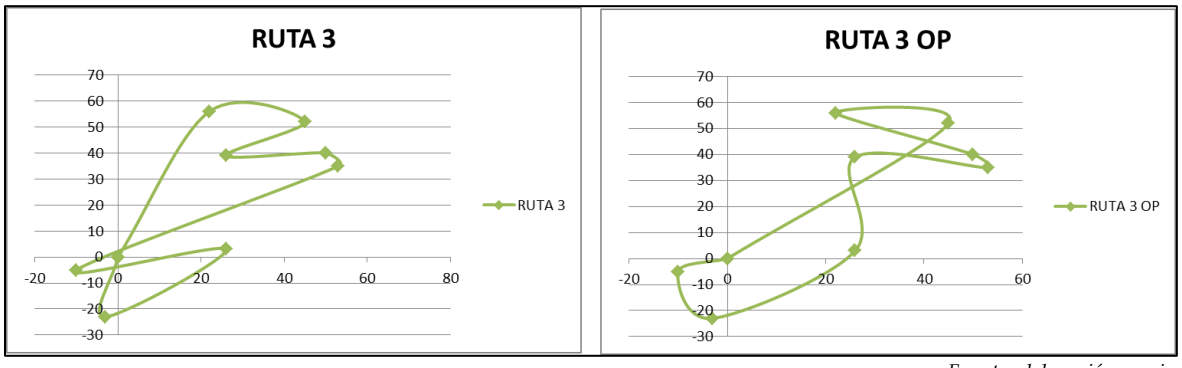
Gráfica 10. Instancia eil 30, Ruta 2 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 2, ver (gráfica 10) el óptimo del *TSP*, nos dio un resultado de 23.96, al analizar los datos obtenidos de la simulación, obtuvimos un valor de 29.70, lo cual nos da un *GAP* de 23.96%.

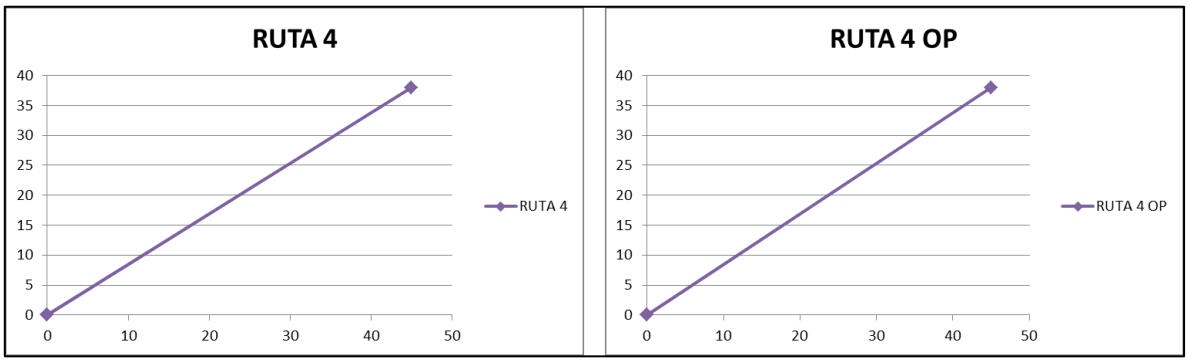
Gráfica 11. Instancia eil 30, Ruta 3 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 3, ver (gráfica 11) el óptimo del *TSP*, nos dio un resultado de 20.71, al analizar los datos obtenidos de la simulación, obtuvimos un valor de 25, lo cual nos da un *GAP* de 20.71%.

Gráfica 12. Instancia eil 30, Ruta 4 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 4, ver (gráfica 12) el óptimo del *TSP*, nos dio un resultado de 117.79, al analizar los datos obtenidos de la simulación, obtuvimos un valor de 117.79, lo cual nos da un *GAP* de 0%.

Los parámetros utilizados en la instancia eil 30 los presentamos en la tabla 11 que se muestra a continuación:

Tabla 11. Resultados de la simulación para la instancia eil 30.

Instancia eil 30						
Ruta	**No. de agentes**	**Radio de acción de la feromona**	**Duración de la feromona**	**Resultado de la simulación**	**Óptimo**	**GAP**
1	200	60	3	303.72	270.62	12.23%
2	200	60	3	29.70	23.96	23.96%
3	200	60	3	25.00	20.71	20.71%
4	200	60	3	117.79	117.79	0.00%

Fuente: elaboración propia.

GAP promedio de 14.23

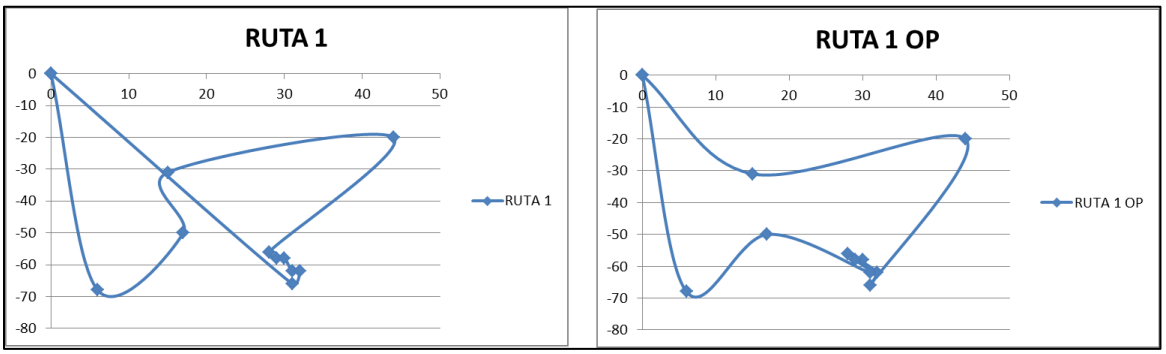
INSTANCIA EIL33

Tabla 12. Resultados de la aplicación de la primera fase del algoritmo a la instancia eil 33.

Eil 33						
Coordenadas		**CLIENTE**	**DEMANDA**	**RUTA**	**Capacidad de los vehículos**	**TOTAL DE CAMIONES**
x	y				8000	5
6	-68	1	700	1		
17	-50	2	400	1		
15	-31	3	400	1		
44	-20	4	1200	1		
28	-56	5	40	1		
29	-58	6	80	1		
30	-58	7	2000	1		
31	-62	8	900	1		
32	-62	9	600	1		
31	-66	10	750	1		
22	-60	11	1500	2		
19	-53	12	150	2		
12	-68	13	250	2		
1	-74	14	1600	2		
4	-77	15	450	2		
-31	-111	16	700	2		
5	-85	17	550	2		
23	-88	18	650	2		
22	-89	19	200	2		
29	-104	20	400	2		
29	-97	21	300	2		
22	-101	22	1300	3		
21	-117	23	700	3		
12	-113	24	750	3		
3	-93	25	1400	3		
-9	-89	26	4000	4		
-13	-96	27	600	4		
-21	-94	28	1000	4		
-28	-81	29	500	4		
-15	-56	30	2500	5		
-2	-61	31	1700	5		
27	-62	32	1100	5		

Fuente: elaboración propia

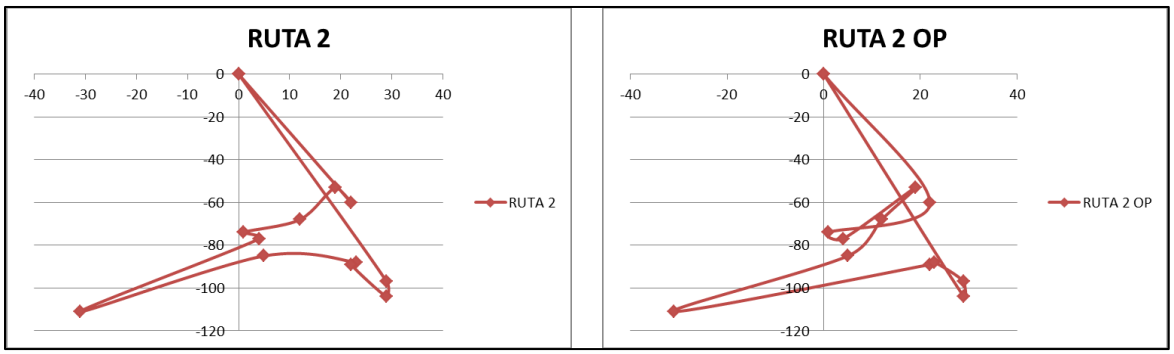
Gráfica 13. Instancia eil 33, Ruta 1 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 1, ver (gráfica 13) el óptimo del *TSP*, nos dio un resultado de 199.92, al analizar los datos de la simulación, obtuvimos un valor de 240.68, lo cual nos da un *GAP* de 20.39%.

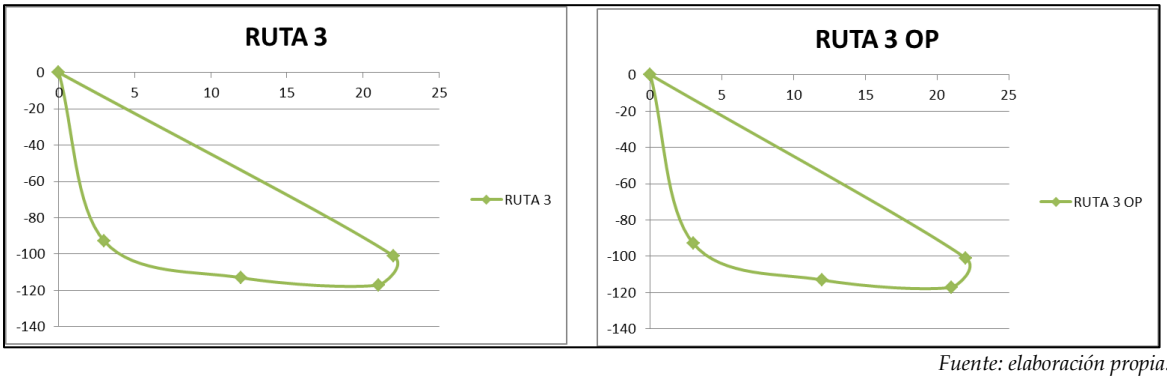
Gráfica 14. Instancia eil 33, Ruta 2 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 2, ver (gráfica 14) el óptimo del *TSP*, nos dio un resultado de 309.68, al analizar los datos obtenidos de la simulación, obtuvimos un valor de 385.61, lo cual nos da un *GAP* de 24.52%.

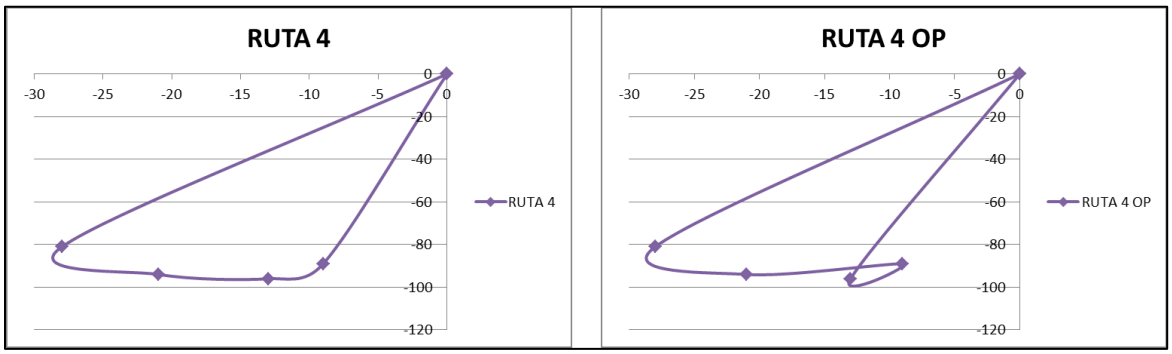
Gráfica 15. Instancia eil 33, Ruta 3 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 3, ver (gráfica 15) el óptimo del *TSP*, nos dio un resultado de 244.22, al analizar los datos obtenidos de la simulación, obtuvimos un valor de 244.22, lo cual nos da un *GAP* de 0%.

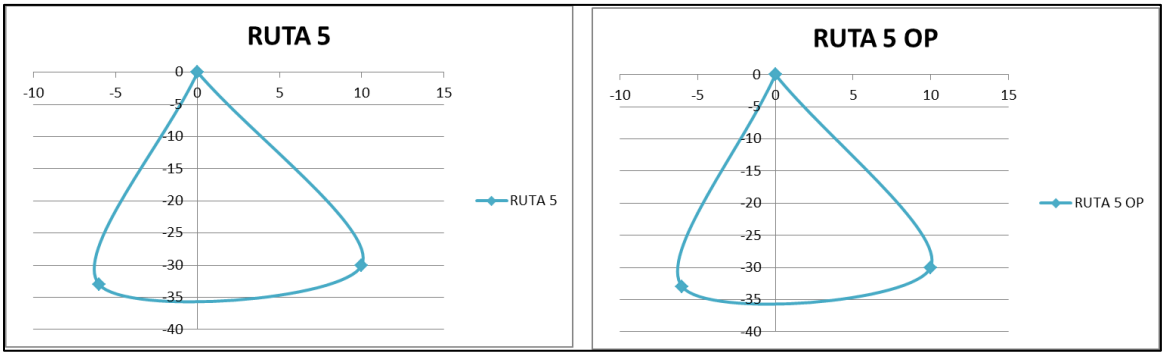
Gráfica 16. Instancia eil 33, Ruta 4 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 4, ver (gráfica 16) el óptimo del *TSP*, nos dio un resultado de 206.23, al analizar los datos obtenidos de la simulación, obtuvimos un valor de 218.40, lo cual nos da un *GAP* de 0%.

Gráfica 17. Instancia eil 33, Ruta 5 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 5, ver (gráfica 5) el óptimo del *TSP*, nos dio un resultado de 81.44, al analizar los datos obtenidos de la simulación, obtuvimos un valor de 81.44, lo cual nos da un *GAP* de 0%.

Los parámetros utilizados en la instancia eil 33 los presentamos en la tabla 13 que se muestra a continuación:

Tabla 13. Resultados de la simulación para la instancia eil 33.

Instancia eil 33						
Ruta	**No. de agentes**	**Radio de acción de la feromona**	**Duración de la feromona**	**Resultado de la simulación**	**Óptimo**	**GAP**
1	200	60	3	240.68	199.92	20.39%
2	200	60	3	385.61	309.68	24.52%
3	200	60	3	244.22	244.22	0.00%
4	200	60	3	218.40	206.23	5.90%
5	200	60	3	168.54	168.54	0.00%

Fuente: elaboración propia.

GAP promedio igual a 10.16%.

INSTANCIA AN54K7

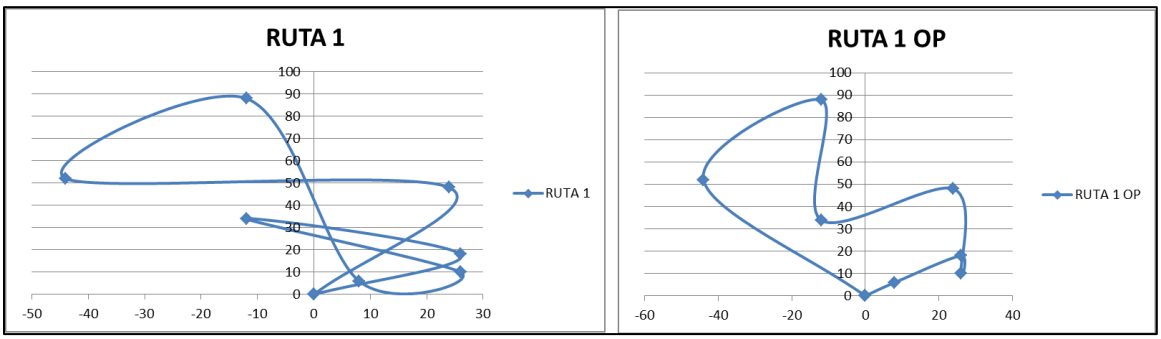
Tabla 14. Resultados de la aplicación de la primera fase del algoritmo a la instancia An54k7.

An54k7						
Coordenadas		CLIENTE	DEMANDA	RUTA	Capacidad de los vehículos	TOTAL DE CAMIONES
X	Y				100	8
24	48	2	24	1		
-44	52	3	9	1		
-12	88	4	15	1		
8	6	5	17	1		
26	10	6	2	1		
-12	34	7	19	1		
26	18	8	10	1		
-42	78	9	17	2		
8	82	10	20	2		
8	38	11	16	2		
-12	62	12	8	2		
-44	56	13	12	2		
-16	56	14	3	2		
-40	48	15	23	2		
10	32	16	4	3		
-8	18	17	23	3		
16	58	18	20	3		
28	2	19	2	3		
-40	78	20	19	3		
16	20	21	2	3		
24	90	22	23	3		
-18	88	23	23	4		
14	20	24	5	4		
-60	38	25	12	4		
-54	2	26	15	4		
20	64	27	9	4		
-38	52	28	13	4		
20	10	29	18	4		
16	30	30	16	5		
-12	-2	31	7	5		
-40	88	32	6	5		
-20	32	33	2	5		
10	86	34	8	5		
-30	8	35	2	5		
8	28	36	2	5		
30	42	37	4	5		
-48	64	38	13	5		
4	70	39	18	5		
30	22	40	9	5		
-52	80	41	19	6		
-46	14	42	3	6		
-54	32	43	14	6		

0	6	44	19	6
-2	78	45	21	6
24	64	46	4	6
-46	24	47	6	6
-60	8	48	22	7
-60	78	49	13	7
24	26	50	10	7
34	20	51	18	7
-56	28	52	5	7
-10	6	53	9	7
-10	80	54	36	8

Fuente: elaboración propia.

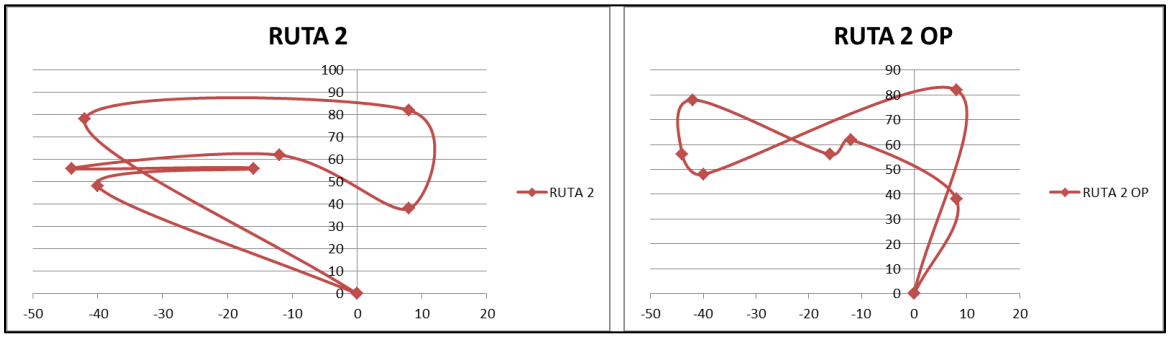
Gráfica 18. Instancia An54k7, Ruta 1 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 1, ver (gráfica 18) el óptimo del *TSP*, nos dio un resultado de 241.25, al analizar los datos de la simulación, obtuvimos un valor de 286.61, lo cual nos da un *GAP* de 18.80%.

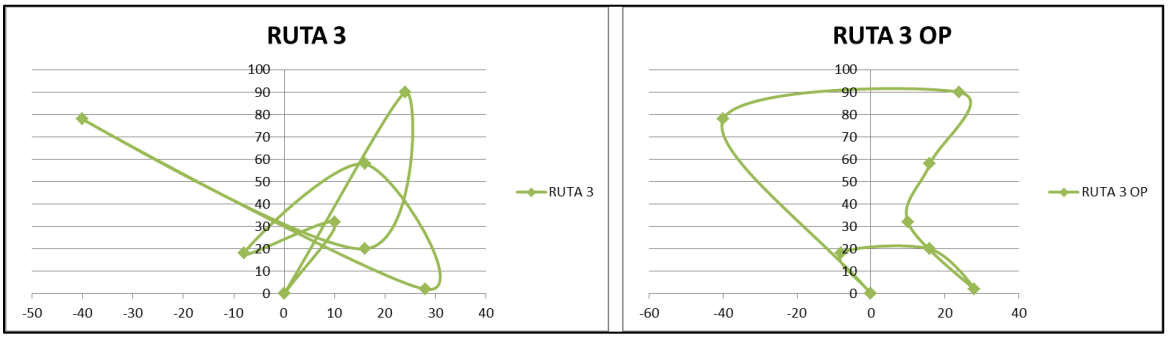
Gráfica 19. Instancia An54k7, Ruta 2 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 2, ver (gráfica 19) el óptimo del *TSP,* nos dio un resultado de 245.9, al analizar los datos obtenidos de la simulación, obtuvimos un valor de 283.60, lo cual nos da un *GAP* de 15.33%.

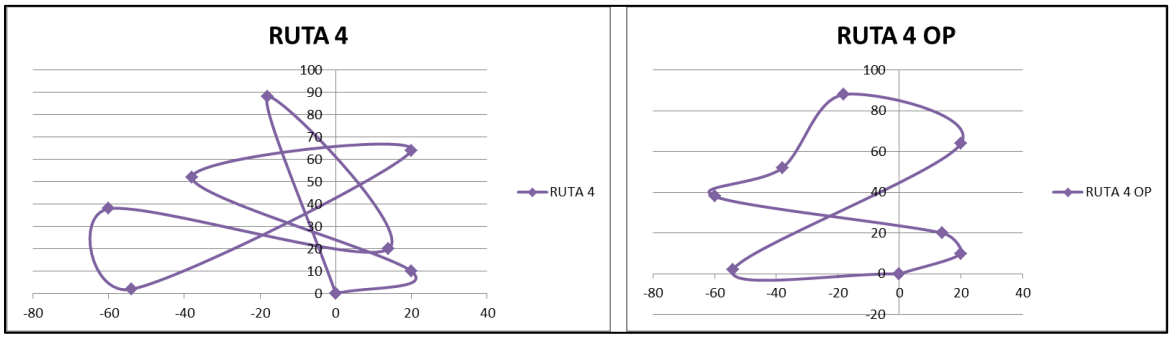
Gráfica 20. Instancia An54k7, Ruta 3 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 3, ver (gráfica 20) el óptimo del *TSP*, nos dio un resultado de 275.6, al analizar los datos obtenidos de la simulación, obtuvimos un valor de 312.83, lo cual nos da un *GAP* de 13.51%.

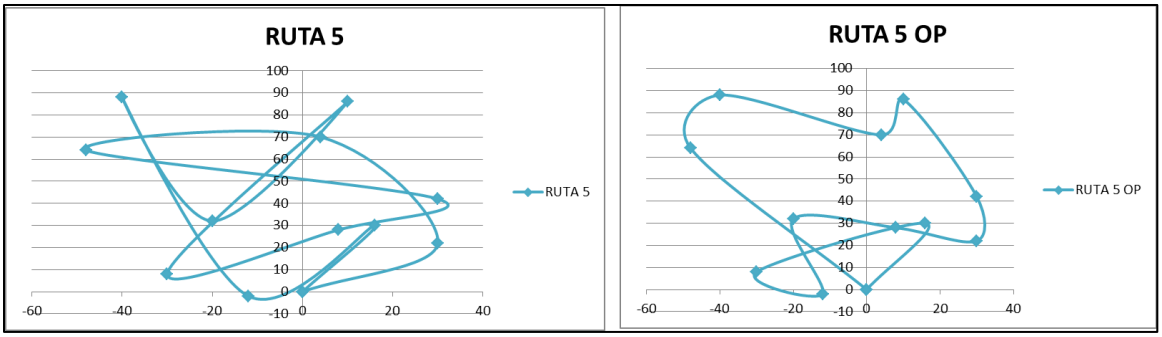
Gráfica 21. Instancia An54k7, Ruta 4 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 4, ver (gráfica 21) el óptimo del *TSP*, nos dio un resultado de 281.16, al analizar los datos obtenidos de la simulación, obtuvimos un valor de 372.96, lo cual nos da un *GAP* de 32.65%.

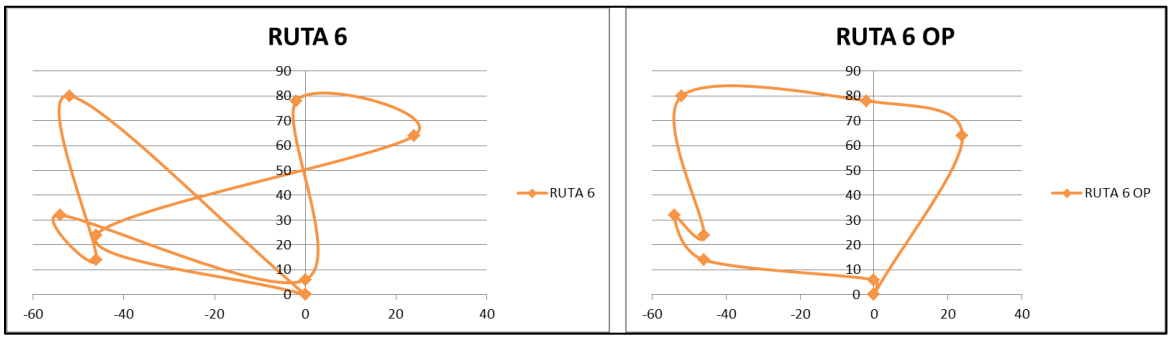
Gráfica 22. Instancia An54k7, Ruta 5 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 5, ver (gráfica 22) el óptimo del *TSP*, nos dio un resultado de 305.4, al analizar los datos obtenidos de la simulación, obtuvimos un valor de 429.85, lo cual nos da un *GAP* de 40.75%.

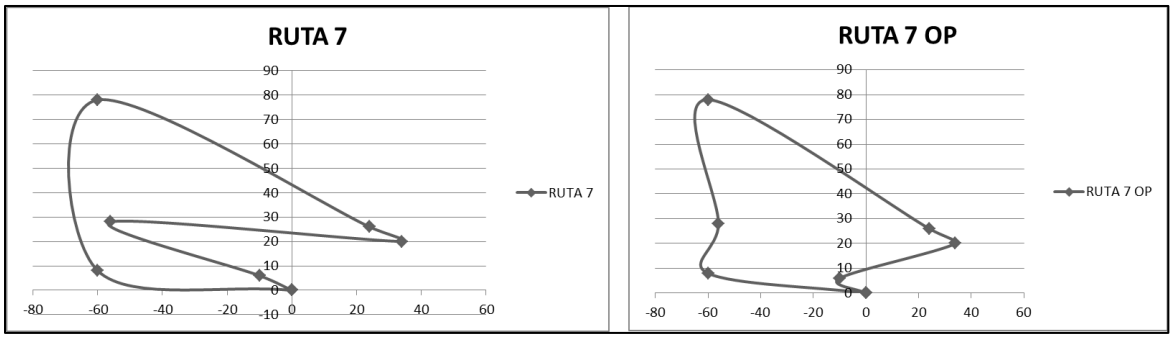
Gráfica 23. Instancia An54k7, Ruta 6 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 6, ver (gráfica 23) el óptimo del *TSP*, nos dio un resultado de 265.778, al analizar los datos de la simulación, obtuvimos un valor de 287.94, lo cual nos da un *GAP* de 8.34%.

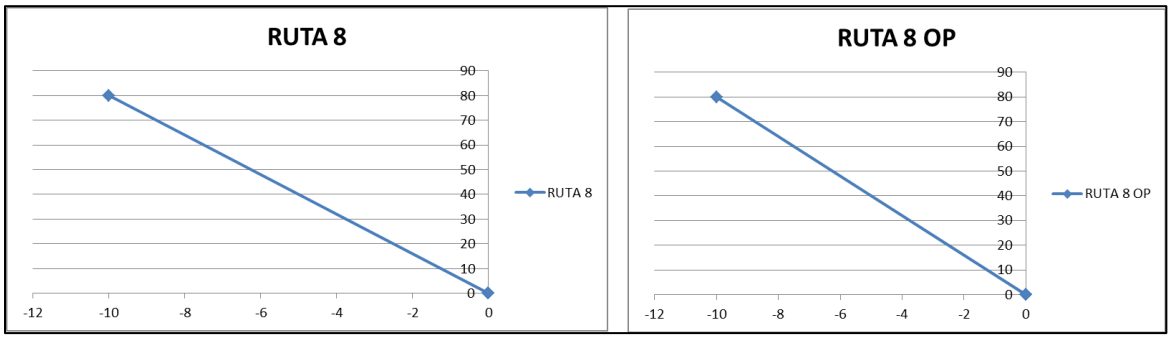
Gráfica 24. Instancia An54k7, Ruta 7 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 7, ver (gráfica 24) el óptimo del *TSP*, nos dio un resultado de 282.157, al analizar los datos obtenidos de la simulación, obtuvimos un valor de 299.37, lo cual nos da un *GAP* de 6.10%.

Gráfica 25. Instancia An54k7, Ruta 8 antes de ser optimizada (izquierda) y después de ser optimizada (derecha).



Fuente: elaboración propia.

Para la ruta 8, ver (gráfica 25) el óptimo del *TSP*, nos dio un resultado de 161.245, al analizar los datos obtenidos de la simulación, obtuvimos un valor de 161.245, lo cual nos da un *GAP* de 0%.

Los parámetros utilizados en la instancia An54k7 los presentamos en la tabla 13 que se muestra a continuación:

Tabla 15. Resultados de la simulación para la instancia An54k7.

Instancia an54k7						
Ruta	**No. de agentes**	**Radio de acción de la feromona**	**Duración de la feromona**	**Resultado de la simulación**	**Óptimo**	**GAP**
1	200	58	4	286.61	241.25	18.80%
2	200	58	4	283.60	245.9	15.33%
3	200	58	4	312.83	275.6	13.51%
4	200	58	4	372.96	281.16	32.65%
5	200	58	4	429.85	305.4	40.75%
6	200	58	4	287.94	265.778	8.34%
7	200	58	4	299.37	282.157	6.10%
8	200	58	4	161.25	161.245	0.00%

Fuente: elaboración propia.

GAP promedio igual a 16.94%.

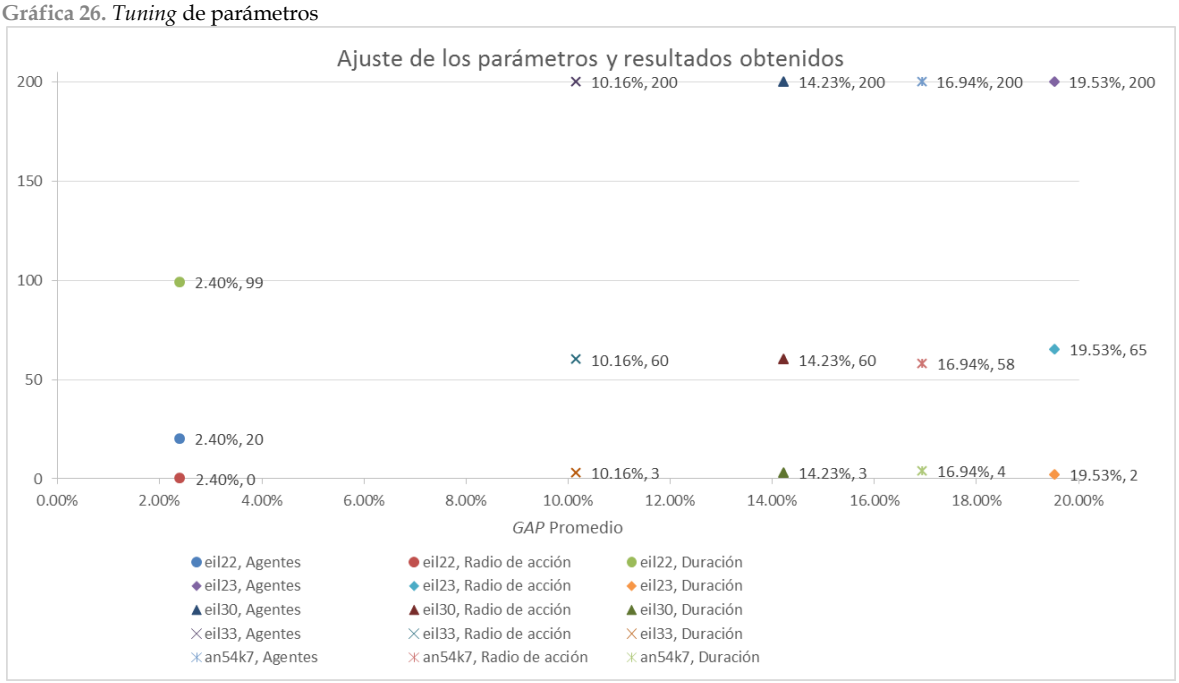
ANÁLISIS DE RESULTADOS

Los resultados aquí mostrados son bastante buenos en términos de nuestro objetivo, el cual consiste en lograr un *GAP* de menos del 20%. A pesar de que el programa no fue diseñado explícitamente para tal efecto, estos resultados podrían mejorar, si ponemos atención a cuestiones importantes como:

1. El principio básico por el cual funciona el programa, se basa en una heurística de *ANT SYSTEMS*, y como trabajo a futuro se pretende pulir el programa no perdiendo de vista el objetivo principal que consiste en tener un programa basado en agentes que sirva para resolver problemas de tipo *CVRP* de una manera relativamente sencilla, fácil de implementar, a bajo costo y de manera rápida;
2. Poner más atención en el *tuning* del programa para obtener resultados con un *GAP* menor a los ya obtenidos.

Es interesante resaltar que, como resultado de la observación y análisis de los resultados, pudimos percibir que el *tuning* es particular de cada problema como se observa en la tabla XX, no se pudo llegar a concluir el hecho de que existan parámetros generales para toda la simulación, cada instancia es particular e independiente de las anteriores y la distribución de los nodos en el espacio afecta, o mejor dicho, es determinante en la selección de los parámetros.

Gráfica 26. *Tuning* de parámetros

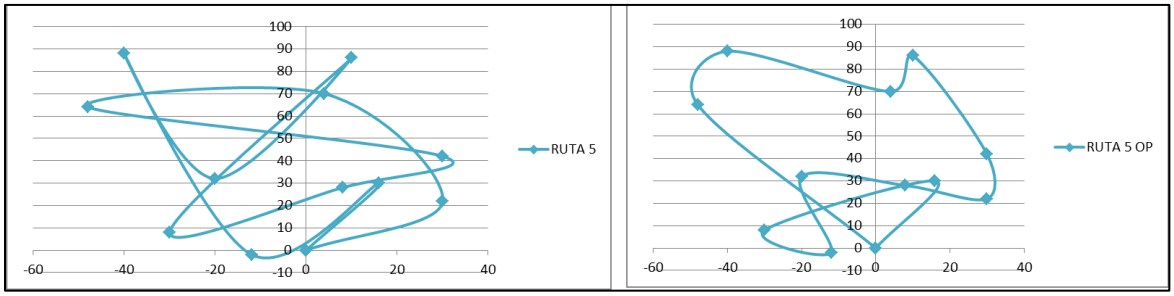


Fuente: elaboración propia

Es importante señalar que como resultado del estudio del desempeño de la simulación, pudimos detectar algunos patrones al momento de resolver distintas instancias del problema, al respecto podemos comentar lo siguiente:

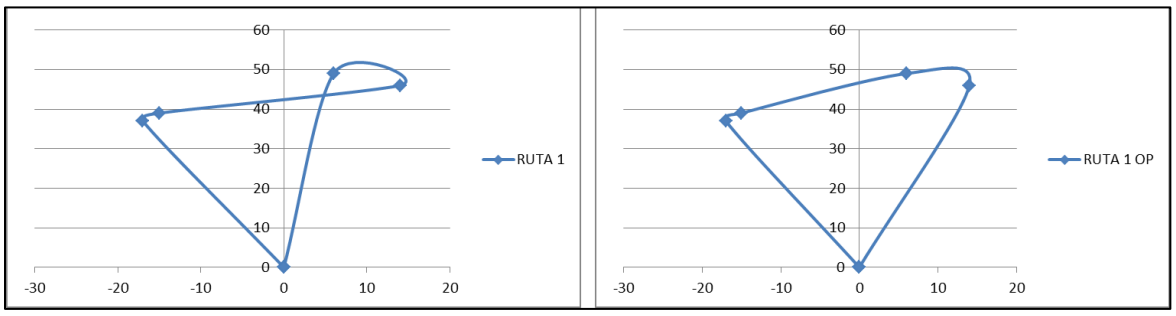
1) Se observa que cuando se encuentran muy cerca entre sí varios puntos (clientes), en el espacio, la solución de se aleja del óptimo, ya que no se logra establecer una ruta claramente definida, la ruta no logra "abrirse" (ver figura 28) del todo, en cambio;
2) Cuando los puntos se encuentran alejados (ver figura 29), es cuando se obtienen mejores resultados, incluso las rutas óptimas.

Figura 28.**La ruta ya optimizada no logra "abrirse" del todo.**



Fuente: Elaboración propia

Figura 29. **La ruta ya optimizada logra "abrirse".**



Fuente: Elaboración propia

Por último, cabe señalar que algunas rutas generadas por la simulación se ven todavía enredadas, podría aplicarse otra heurística a las soluciones obtenidas para tratar de mejorar los resultados, pero eso queda como trabajo a futuro.

CONCLUSIONES

Parte importante de este trabajo, es resaltar el valor que tiene el hecho de seguir desarrollando herramientas que nos ayuden a resolver problemas de optimización complejos y que además son parte cotidiana de la vida real. El desarrollo que implicó la elaboración de esta tesis nos amplía el panorama en este campo, este documento no sólo persigue el interés académico, sino que además, busca su aplicación a las tareas de la vida diaria, ya que no planea ser sofisticado ni caro, sino que impulsa la investigación y desarrollo de herramientas que sean fáciles de utilizar, fáciles de implementar y de fácil acceso al público en general, además de dejar líneas de investigación a futuro abiertas, para ser retomadas y mejoradas.

Como parte medular de este trabajo se propuso la idea de utilizar la *ABS* mediante una heurística basada en *AS* para resolver problemas *CVRP*, este objetivo se cumplió totalmente ya que se pudo demostrar mediante el análisis de la literatura, y la replicación de experimentos y posterior comparación con modelos ya establecidos, que la idea aquí propuesta da buenos resultados. Cabe señalar que los algoritmos basados en comportamientos de la naturaleza, no son nuevos, lo que hace que estos algoritmos mejoren o empeoren su rendimiento, es la utilización de un conjunto de herramientas que nos ayudan a resolver un problema en específico, en nuestro caso es la utilización de la Simulación Basada en Agentes con la combinación de una heurística inspirada en Sistemas de Hormigas y un método de dos fases, específicamente, el método de primero agrupar, luego rutear.

Cabe resaltar que, el programa que se utilizó para resolver las instancias de los problemas *CVRP* no estaba diseñado explícitamente para tal efecto, el programa fue adaptado y utilizado con este fin, producto de:

i) La inquietud de tratar de establecer un antecedente en cuanto a la solución de este tipo de problemas con simulación basada en agentes y;
ii) El análisis de la información recabada para la realización de esta tesis, que arrojó indicios de que el presente trabajo sí podía ser llevado a cabo con bases teóricas y sustentadas por especialistas en el tema.

El hecho de decir que nuestra propuesta da buenos resultados no es sinónimo de que no se pueda mejorar, existen muchas áreas de oportunidad y mejora que se pueden explotar como trabajo a futuro, dentro de las cuales se cuentan:

1. Automatizar el proceso del ingreso de datos al modelo y su posterior análisis para cada una de las corridas;
2. Mejorar el programa para obtener resultados más precisos;

3. Análisis de las distintas combinaciones que se pueden generar con la modificación de los parámetros (*tuning*) ya que esto podría mejorar los resultados.

En dado caso que como trabajo a futuro se generen nuevas soluciones con brechas, entre el óptimo y los resultados obtenidos, más pequeñas, sería conveniente evaluar la posibilidad de re-optimizar las rutas obtenidas con alguna heurística desarrollada para tal efecto.

Para finalizar podemos resumir que:

1. Logramos adaptar y mejorar un programa que imita el comportamiento natural de las hormigas, basado en *ABS*, para poder resolver problemas de optimización combinatoria, en específico un problema *CVRP*, en combinación con un método de dos fases, lo cual mejoro bastante los resultados esperados.
2. Pudimos cumplir con el objetivo de lograr un *GAP* promedio de menos del 20%.
3. Desarrollamos una herramienta que es fácil de entender, fácil de usar, y fácil de implementar además de ser gratuita
4. Logramos resolver varias instancias, tanto medianas como grandes de problemas tipo *CVRP* cuya complejidad computacional es de tipo NP-duro, obteniendo buenos resultados.
5. Se espera que este trabajo pueda tener continuidad, dejando como trabajo a futuro el probar con más instancias, mejorando el *tining* de los parámetros de la simulación así como el programa de cómputo.

BIBLIOGRAFÍA

1. ANTONIO BRANDÃO, JOSÉ. MERCER, ALAN. (1997). "A tabu search algorithm for the multi-trip vehicle routing and scheduling problem"; European Journal of Operational Research, Vol. 100, pp. 180-191.
2. AXELROD, R. (1997). "The complexity of cooperation: Agent based models of competition and collaboration". Princeton NJ: Princeton University Press.
3. AXTELL, R. (2000). "Why agents? On the varied motivations for agent computing in the social sciences". En Macal, C. M. & Sallach, D. (eds.), Proceedings of the Workshop on Agent Simulation: Applications, Models, and Tools, pp. 3-24. Argonne, IL: Argonne National Laboratory.
4. B. CRAWFORD, R. SOTO, E. MONFROY, W. PALMA, C. CASTRO, AND F. PAREDES. (2013). "Parameter tuning of a choice-function based hyperheuristic using particle swarm optimization". Expert Systems with Applications, 40 (5), pp. 1690-1695.
5. BECKERS, R.; DENEUBOURG, J.L.; GOSS, S. (1992). "Trails and U-turns in the selection of the shortest path by the ant". Lasius Niger. J. Theoretical Biology, Vol. 159, pp. 397–415.
6. BOUSQUET, F. & LE PAGE, C. (2004). "Multi-agent simulations and ecosystem management: A review". Ecological Modelling 176(3-4), pp. 313-332.
7. BRATTON, DANIEL; KENNEDY, JAMES. (2007). "Defining a Standard for Particle Swarm Optimization"; Proceedings of the 2007 IEEE Swarm Intelligence Symposium, (SIS 2007).
8. BULLNHEIMER, B.; HARTL, R.F.; STRAUSS, C. (1997). "Applying the Ant System to the Vehicle Routing Problem". Paper presented at 2nd International Conference on Metaheuristics, Sophia-Antipolis, France.
9. BULLNHEIMER, B.; HARTL, R.F.; STRAUSS, C. (1997). "A New Rank Based Version of the Ant System: A Computational Study". Central European Journal for Operations Research and Economics, Vol. 7, No. 1, pp. 25-38.
10. COLORNI, A. DORIGO, M. AND MANIEZZO, V. (1991). "Distributed optimization by ant colonies". In. F Varela and P. Bourgine, editors, *Proceedings of the European Conference on artificial Life.* Elsevier, Amsterdam.
11. **CO**mputational **IN**frastructure for **O**perations **R**esearch (COIN-OR); (2014), disponible en: http://www.branchandcut.org/
12. CONTE, R., HEGSELMANN, R. & TERNA, P. (1997). "Simulating Social Phenomena". Lecture Notes in Economics and Mathematical Systems 456.
13. DAVENDRA, DONALD. (2013). "Traveling Salesman Problem, Theory and Applications". Published by InTech, Janeza Trdine 9, 51000 Rijeka, Croatia. 2010.
14. DAVIDSSON, PAUL; HOLMGREN, JOHAN; KYHLBÄCK, HANS; MENGISTU, DAWIT; PERSSON, MARIE. (2007). "Applications of Agent Based Simulation". School of Engineering, Blekinge Institute of Technology Soft Center, 372 25 Ronneby, Sweden.
15. DE LA FUENTE, DAVID. LOZANO, JESÚS. OCHOA DE OLANO, EVA. DÍAZ, MAGÍN. (2011). "Estado del arte de algoritmos basados en colonias de hormigas para la resolución del problema VRP". 5th International Conference on Industrial

Engineering and Industrial Management. XV Congreso de Ingeniería de Organización Cartagena.
16. DENEF, FREDERIK. DOUGLAS, MICHAEL R. (2007). "Computational complexity of the landscape: Part I". Annals of Physics 322 (2007) 1096–1142.
17. DORER, KLAUS; CALISTI, MONIQUE. (2005). "An Adaptive Solution to Dynamic Transport Optimization". Whitestein Technologies AG Pestalozzistrasse 24 8032 Zürich, Switzerland.
18. DORIGO, M.; GAMBARDELLA, L.M. (1997). "Ant colony system: A cooperative learning approach to the travelling salesman problem". IEEE Transactions On Evolutionary Computation, Vol.1, No. 1, pp.53–66.
19. DORIGO, MARCO. MANIEZZO, VITTORIO AND COLORNI, ALBERTO. (1996). "The ant system: Optimization by a colony of coorperating agents". IEEE Trans. on Systems, Man and Cybernetics-Part B, 26(1):29–41.
20. E. BELL, JOHN; R. MCMULLEN, PATRICK. (2004). "Ant colony optimization techniques for the vehicle routing problem". Advanced Engineering Informatics 18 (2004) 41–48.
21. EDMONDS, B., MOSS, S. & DAVIDSSON, P. (2001). "The Use of Models - making MABS actually work". In, *Multi-Agent-Based Simulation, Lecture Notes in Artificial Intelligence 1979*, pp. 15-32. Berlin: Springer-Verlag.
22. EILON, S., WATSON-GANDY, C.D.T., AND CHRISTOFIDES, N. (1971). "Distribution Management: Mathematical Modelling and Practical Analysis". Griffin, London.
23. ELIZONDO CORTÉS, MAYRA. (2014); "Apuntes de Complejidad Computacional". Universidad Nacional Autónoma de México, Facultad de Ingeniería, Posgrado, pp.69.
24. EPSTEIN, J. M. & AXTELL, R. L. (1996). "Growing Artificial Societies: Social Science from the Bottom Up". The MIT Press.
25. FLORES, I., ELIZONDO, M. (2007). "Apuntes de simulación"; Universidad Nacional Autónoma de México, Facultad de Ingeniería, Posgrado, pp.167, (2007).
26. G. B. DANTZIG AND J. H. RAMSER. (1959). "The Truck Dispatching Problem". Management Science, Vol. 6, No. 1, pp. 80-91.
27. G. CLARKE AND J. WRIGHT. (1964). "Scheduling of vehicles from a central depot to a number of delivery points", Operations Research, 12 #4, 568-581.
28. GAERTNER, DORIAN AND CLARK, KEITH. (2005). "On Optimal Parameters for Ant Colony Optimization algorithms" Dept of Computing, Imperial College London, 180 Queens Gate, London, SW7 2AZ, UK
29. GAJPAL, YUVRAJ. ABAD, P.L. (2009). "Multi-ant colony system (MACS) for a vehicle routing problem with backhauls". Michael G DeGroote School of Business, McMaster University, Hamilton, Ontario, Canada L8S4M4. European Journal of Operational Research 196 (2009) 102–117.
30. GALÁN, J. M., IZQUIERDO, L. R., IZQUIERDO, S. S., SANTOS, J. I., OLMO, R., LÓPEZPAREDES, A. & EDMONDS, B. (2008). "Errors and artefacts in agent-based modelling". *Journal of Artificial Societies and Social Simulation*.
31. GARCÍA-VALDECASAS MEDINA, JOSÉ IGNACIO. (2011). "La simulación basada en agentes: una nueva forma de explorar los fenómenos sociales"; Reis 136, pp. 91-110, (2011).

32. GILBERT, N. & TERNA, P. (2000). "How to build and use agent-based models in social science". Mind and Society 1(1), pp. 57-72.
33. GILBERT, N. & TROITZSCH, K. G. (1999). "Simulation for the Social Scientist". Buckingham, UK: Open University Press.
34. GILBERT, N. (2002). "Varieties of Emergence". En MACAL, C. & SALLACH, D. (eds.), Social Agents: Ecology, Exchange, and Evolution. Agent 2002 Conference, pp. 41-50. Chicago: University of Chicago and Argonne National Laboratory.
35. GILBERT, N. (2007). "Agent-Based Models. Quantitative Applications in the Social Sciences". London: SAGE Publications.
36. GILBERT, N. (2008a). "AGENT-BASED MODELS". University of Surrey, Guildford, UK.
37. GILBERT, N. (2008b). "Agent-Based Models". Quantitative Applications in the Social Science 153, Londres: Sage.
38. GOSS, S.; ARON, S.; DENEUBOURG, J.L.; PASTEELS, J.M. (1989). "Self-organized shortcuts in the argentine ant". Naturwissenschaften, Vol. 76, pp. 579–581.
39. HILLIER, F. S., LIEBERMAN, G. J. (1994). Introducción a la investigación de operaciones. Editorial McGraw-Hill.5ª ed. (1994).
40. HOLLAND, J. H. (1998). "Emergence. From chaos to order". Reading, MA: Addison-Wesley.
41. HUERTA BARRIENTOS, AIDA. (2014). "Metodología basada en Modelos de Simulación para el Análisis de Sistemas Complejos (MoSASCoM)"; Universidad Nacional Autónoma de México; Posgrado de Ingeniería; México D.F. Octubre 2014.
42. Instituto Mexicano del Transporte (IMT). (2014). "Manual Estadístico del Sector Transporte 2013". Disponible en: http://imt.mx/archivos/Publicaciones/Manual/mn2012.pdf
43. J. K. LENSTRA, A.H.G. RINNOOY KAN. (1981). "Complexity of vehicle routing and scheduling problems", Networks, 11, pp. 221-227.
44. JOHNSON, P. E. (1999). "Simulation modeling in political science". American Behavioral Scientist 42(10), pp. 1509-1530.
45. JOHNSON, S. (2001). "Emergence: the connected lives of ants, brains, cities, and software". New York: Scribner.
46. KAVEH, A.; SHAHROUZI, M. (2008). "Optimal structural design family by genetic search and ant colony approach". Engineering Computations, Vol. 25, No.3, pp. 268-288.
47. KOHLER, T. & GUMERMAN, G. J. (2000). "Dynamics in human and primate societies: Agent-based modeling of social and spatial processes". New York: Oxford University Press & Santa Fe Institute.
48. LAPORTE SOTO, GILBERT. (1991). "The Vehicle Routing Problem: An overview of exact and approximate algorithms", *European Journal of Operational Research*, Vol. 59, 1991, pp 345-358.
49. LEBARON, B. (2000). "Agent Based Computational Finance: Suggested Readings and Early Research". Journal of Economic Dynamics & Control 24(5-7), pp. 679-702.
50. LEE, Z. L.; SU, S. F.; CHUANG, C. C.; LIU, K. H. (2008). "Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment". Applied Soft Computing Journal, Vol. 8, No. 1, pp. 55-78.
51. LI, L.; JU, S.; ZHANG, Y. (2008). "Improved Ant Colony Optimization for the Traveling Salesman Problem". International Conference on Intelligent Computation Technology and Automation, pp. 76-80.

52. LÓPEZ PAREDES, A. & HERNÁNDEZ IGLESIAS, C. (2008). "Agent Based Modelling in Natural Resource Management". Insisoc. España. ISBN 978-84-205-4560-8.
53. M. BALINZKI and R. QUANDT. (1964). "On an Integer Program for a Delivery Problem", *Operational Research*, Vol. 12, No. 2, pp 300-304.
54. M. GUERIN, STEPHEN. (2004). "PEEKING INTO THE BLACK BOX: SOME ART AND SCIENCE TO VISUALIZING AGENT-BASED MODELS". Proceedings of the 2004 Winter Simulation Conference.
55. MANSURY, Y. & DEISBOECK, T. S. (2004). "Simulating the time series of a selected gene expression profile in an agent-based tumor model". Physica D: Nonlinear Phenomena 196(1-2), pp. 193-204.
56. MANSURY, Y., KIMURA, M., LOBO, J. & DEISBOECK, T. S. (2002). "Emerging Patterns in Tumor Systems: Simulating the Dynamics of Multicellular Clusters with an Agent based Spatial Agglomeration Model". Journal of Theoretical Biology 219(3), pp. 343-370.
57. MAROTO ÁLVAREZ, CONCEPCIÓN. ALCARAZ SORIA, JAVIER, RUIZ GARCÍA, RUBÉN. (2002). "Investigación operativa: Modelos y técnicas de investigación". Universidad Politécnica de Valencia. I. S. B. N.: 84-9705-239-0. pp. 176.
58. MAZZEO, SILVIA. LOISEAU, IRENE. (2004). "An Ant Colony Algorithm for the Capacitated Vehicle Routing". Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina. Electronic Notes in Discrete Mathematics 18 (2004) 181–186.
59. NIKOUKARAN, J., HLUPIC, V & PAUL, R. (1999). "A hierarchical Framework for evaluating simulation software, Simulation Practice and theory". 7, 219-231.
60. OLIVERA, ALFREDO. (2004). "Heurísticas para problemas de ruteo de vehículos", reporte de investigación, Instituto de Computación – Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay, disponible en: http://www.fing.edu.uy/inco/pedeciba/bibliote/reptec/TR0408.pdf.
61. OSTROM, E., GARDNER, R. & WALKER, J. (1994)."Rules, Games, and Common-Pool Resources". Ann Arbor: University of Michigan Press.
62. PATON, R., GREGORY, R., VLACHOS, C., SAUNDERS, J. & WU, H. (2004). "Evolvable social agents for bacterial systems modeling". IEEE Transactions on Nanobioscience 3(3), pp. 208-216.
63. PRINS, CHRISTIAN. (2002). "Efficient heuristics for the heterogeneous fleet multitrip VRP with application to a large-scale real case", Journal of Mathematical Modelling and Algorithms, Vol. 1, 2002, pp 135-150.
64. R. IZQUIERDO, LUIS; M. GALÁN, JOSÉ; I. SANTOS, JOSÉ; DEL OLMO, RICARDO. (2008). "Modelado de sistemas complejos mediante simulación basada en agentes y mediante dinámica de sistemas". EMPIRIA. Revista de Metodología de Ciencias Sociales. No. 16, julio-diciembre, 2008, pp. 85-112.
65. REED, MARTIN. YIANNAKOU, ALIKI. EVERING, ROXANNE. (2014). "An ant colony algorithm for the multi-compartment vehicle routing problem". Department of Mathematical Sciences, University of Bath, Bath, United Kingdom. Applied Soft Computing 15 (2014) 169–176.
66. REYNOLDS, C. W. (1987). "Flocks, Herds, and Schools: A Distributed Behavioral Model". Computer Graphics, 21(4), pp. 25-34.

67. ROCHA, L.; GONZÁLEZ, C. y ORJUELA, J. (2011). "Una revisión al estado del arte del problema de ruteo de vehículos: Evolución histórica y métodos de solución". Ingeniería, Vol. 16, No. 2, pp. 35 - 55.
68. Rochat, Y. and Taillard, É. D. (1995), Probabilistic diversification and intensification in local search for vehicle routing, to appear in Journal of Heuristics 1.
69. S. BALESTRINI, ROBINSON; J. M. ZENTNER; T. R. ENDER. (2009). "On modeling and simulation methods for capturing emergent behaviors for systems of systems". 12th Annual Systems Engineering Conference National Defense Industrial Association; San Diego C.A; (2009).
70. SANDOYA SANCHÉZ, FERNANDO. (2007). "Métodos Exactos y Heurísticos para resolver el Problema del Agente Viajero (TSP) y el Problema de Ruteo de Vehículos (VRP)". Decimocuartas Jornadas en Estadística e Informática. ESCUELA SUPERIOR POLITECNICA DEL LITORAL. Guayaquil, Ecuador. Disponible en: http://www.icm.espol.edu.ec/jornadas/14/archivos/Diapositivas/SandoyaFernando/conferencia/SandoyaFernando_M%C3%A9todos_exactos_y_heur%C3%ADsticos_para_el%20VRP_jornadas.pdf
71. SCHELLING, T. C. (1971). "Dynamic Models of Segregation". Journal of Mathematical Sociology, 1(2), pp. 143-186.
72. SHEHORY, ONN; STURM, ARNON. (2001). "Evaluation of Modeling Techniques for Agent-Based Systems". IBM Research Lab in Haifa - the Tel-Aviv site, IBM Building, Tel Aviv 61336, ISRAEL. Technion - Israel Institute of Technology, Haifa 32000, ISRAEL.
73. SOCHA, K.; KNOWLES, J.; SAMPELS, M. (2002). "A MAX –MIN ant system for the university timetabling problem". Lecture Notes in Computer Science, Vol. 2463, pp. 63-77.
74. SQUAZZONI, F. (2008). "The Micro-Macro Link in Social Simulation". Sociologica 1/2008, doi: 10.2383/26578.
75. STÜTZLE, T.; HOOS, H. (2000). "MAX-MIN Ant System". Future Generation Computer Systems, Vol. 16, No. 8, pp. 889–914, 2000.
76. TAVERA MARÍN, ANDREA. (2013). "Simulación basada en agentes para resolver un problema de localización"; Universidad Nacional Autónoma de México, Posgrado de Ingeniería, México D.F.
77. TESFATSION, L. (2002). "Agent-based computational economics: growing economies from the bottom up". Artificial life 8(1), pp. 55-82.
78. TESFATSION, L. (2003). "Agent-based computational economics: modeling economies as complex adaptive systems". Information Sciences 149(4), pp. 263-269.
79. TORSUN, I. S. (1995). "Foundations of Intelligent Knowledge-based Systems". New York: Academic Press.
80. TOTH, PAOLO and VIGO, DANIELE. (2002) "The Vehicle Routing Problem". Society of Industrial and Applied Mathematics (SIAM) monographs on discrete mathematics and applications, Philadelphia, USA, pp 1-23, 109-149.
81. TSAI, CHENG-FA. TSAI CHUN-WEI. (2002). "A New Approach for Solving Large Traveling Salesman Problem Using Evolutionary Ant Rules". Department of Management Information Systems. National Pingtung University of Since and Technology. Pingtung, Taiwan. 91201.
82. Universidad Tecnológica Nacional (UTN), Facultad Santa Fe, Argentina. (2010); Apuntes de Teoría Unidad I; "Planteo General de la Simulación"; pp.30; Disponible en: http://www.frsf.utn.edu.ar/matero/visitante/index.php?id_catedra=150&ver=4

83. W. M. GARVIN, H. W. CRANDALL, J.B. JOHN and R. A. SPELLMAN. (1957). "Aplications of Linear Programming in the Oil Industry", *Management Science*, Vol. 3, pp 407-430.
84. WALKER, D. C., HILL, G., SMALLWOOD, R. H. & SOUTHGATE, J. (2004a). "Agent-based computational modelling of wounded epithelial cell monolayers». IEEE Transactions on Nanobioscience 3, pp. 153-163.
85. WALKER, D. C., SOUTHGATE, J., HILL, G., HOLCOMBE, M., HOSE, D. R., WOOD, S. M., MAC NEIL, S. & SMALLWOOD, R. H. (2004b). "The epitheliome: agent-based modelling of the social behaviour of cells". Biosystems 76(1-3), pp. 89-100.
86. WU, Z.; ZHAO, N.; REN, G.; QUAN, T. (2008). "Population declining ant colony optimization algorithm and its applications". Expert Systems with Applications, Vol. 36, No. 2, pp. 6276-6281.
87. YU BIN; YANG ZHONG-ZHEN; YAO BAOZHEN. (2009). "An improved ant colony optimization for vehicle routing problem". European Journal of Operational Research 196 (2009) 171–176.
88. ZHAO, N.; WU, Z.; ZHAO, Y.; QUAN, T. (2010). "Ant colony optimization algorithm with mutation mechanism and its applications". Expert Systems with Applications, Vol. 37, No. 7, pp. 4805-4810.

ANEXOS

I. Código en Java del programa final ya modificado y realizado en NetLogo.

```
patches-own
[
  chemical
  food
  nest?
  nest-scent
  food-source-number
]

to setup
  clear-all
  set-default-shape turtles "bug"
  crt population
[
 set size 2
 set color red
]

  setup-patches
  reset-ticks
  reset-timer

end

to setup-patches
  ask patches
[
 setup-nest
 setup-food
 recolor-patch
]

end

to setup-nest

  set nest? (distancexy (0) (0)) < 5
  set nest-scent 200 - distancexy (0) (0)

 end

to setup-food

  let x [] ;; ingresar las coordenadas x de los clientes
  let y [] ;; ingresar las coordenadas y de los clientes
  let s [] ;; ingresar el contador empezando con 0
(
foreach x y s
 [
  ask patches with [pxcor = ?1 and pycor = ?2] [set food-source-number 1 set plabel ?3 + 1]
  ]
)
    if food-source-number > 0
[
  set food one-of [1 1]
```

```
 ]

end

to recolor-patch
   ifelse nest?
  [ set pcolor violet ]
  [ ifelse food > 0
    [ if food-source-number = 1 [ set pcolor cyan ]
     ]
  [ set pcolor scale-color green chemical 0.1 5 ] ]
end

to go

  ask turtles
  [ if who >= ticks [ stop ]
    ifelse color = red
    [ look-for-food  ]
    [ return-to-nest ]
    wiggle
    fd 1 ]
  diffuse chemical (diffusion-rate / 100)
  ask patches
  [ set chemical chemical * (100 - evaporation-rate) / 100
    recolor-patch ]
  tick

end

to return-to-nest
  ifelse nest?
  [
    set color red
    rt 180
   ]
  [
 set chemical chemical + 60
    uphill-nest-scent
]
end

to look-for-food
  if food > 0
  [ set color orange + 1
    set food food - 1


    output-print plabel
    output-print pxcor
    output-print pycor
export-output "clientes30.csv" ;;cambiar manualmente el destino del archivo con los datos de la
                                          corrida.


    rt 180
    stop]
   if (chemical >= 0.05) and (chemical < 2)
  [ uphill-chemical ]


end
```

```
  let scent-ahead chemical-scent-at-angle   0
  let scent-right chemical-scent-at-angle  90
  let scent-left  chemical-scent-at-angle -90
  if (scent-right > scent-ahead) or (scent-left > scent-ahead)
  [ ifelse scent-right > scent-left
    [ rt 45 ]
    [ lt 45 ] ]
end

 let scent-ahead nest-scent-at-angle   0
  let scent-right nest-scent-at-angle  45
  let scent-left  nest-scent-at-angle -45
  if (scent-right > scent-ahead) or (scent-left > scent-ahead)
  [ ifelse scent-right > scent-left
    [ rt 45 ]
    [ lt 45 ] ]
end

to wiggle
  rt random 40
  lt random 40
  if not can-move? 1 [ rt 180 ]
end

to-report nest-scent-at-angle [angle]
  let p patch-right-and-ahead angle 1
  if p = nobody [ report 0 ]
  report [nest-scent] of p
end

to-report chemical-scent-at-angle [angle]
  let p patch-right-and-ahead angle 1
  if p = nobody [ report 0 ]
  report [chemical] of p
end
```