

T
519.703
VAS



ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Instituto de Ciencias Matemáticas

Ingeniería en Estadística Informática

“Uso e implementación de métodos meta heurísticos de tipo recocido simulado en problemas de optimización duros”

TESIS DE GRADO

Previa a la obtención del Título de

INGENIERO EN ESTADÍSTICA INFORMÁTICA

Presentada por:

Juan Carlos Vásquez Castillo

GUAYAQUIL – ECUADOR

AÑO

2003



D-31990

AGRADECIMIENTO



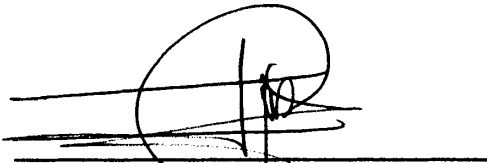
Agradezco principalmente a Dios ya que sin la ayuda de él nada es posible, a mis padres Luis Bolívar y Graciela por el apoyo incondicional que me dieron a lo largo de mi carrera universitaria, a mis hermanos y demás que de una u otra manera me supieron dar la mano cuando lo necesite.

DEDICATORIA



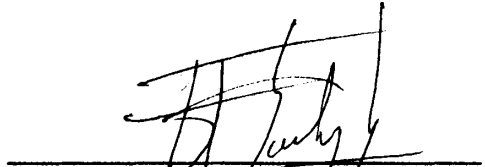
A Dios
A mis Padres, Luis y Graciela
A Karen

TRIBUNAL DE GRADUACIÓN



MAT. JORGE MEDINA

DIRECTOR DEL I.C.M.



MAT. FERNANDO SANDOYA

DIRECTOR DE TESIS



MAT. CESAR GUERRERO

VOCAL



ING. SILVIO SAMANIEGO

VOCAL

DECLARACIÓN EXPRESA

“La responsabilidad del contenido de esta tesis de grado, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL”

(Reglamento de graduación de la ESPOL)


Juan Carlos Vásquez Castillo



RESUMEN

El presente trabajo surge como consecuencia de la dificultad que se tiene para resolver problemas de optimización duros, es decir, problemas complicados o imposibles de resolver con algoritmos de optimización exactos en tiempos razonables.

Se trabajará con métodos aproximados de resolución denominados Meta heurísticos, específicamente con el algoritmo Recocido Simulado. Se iniciará el trabajo hablando acerca de las técnicas de optimización exacta, para de ahí dar paso a los método meta heurísticos.

Dada la importancia de muchas de las técnicas meta heurísticos, se dedica el capítulo 3 a su estudio, en el cual se hablan de los principales métodos que han aparecido en este campo.

En el capítulo 4, hablaremos exhaustivamente de lo que es en sí el Recocido Simulado, dejando para el capítulo 5 el Diseño, Implementación y Aplicación de Método Recocido Simulado a un problema práctico muy importante dentro de la logística de las empresas: el problema de ruteo de vehículos capacitado (VRP o CVRP por sus siglas en inglés).



La complicada orografía y el elevado grado de dispersión en las poblaciones son los dos problemas fundamentales que agravan la difícil tarea de la elaboración de las rutas en la resolución del VRP.

El problema a resolver debe satisfacer un objetivo principal que es el económico, que consiste en la minimización del costo de transporte.

Al final del capítulo 5, se analizan los resultados obtenidos y se comparan con los presentados por otros autores.



CIB - ESPOL

INDICE GENERAL

	Pág.
RESUMEN	II
ÍNDICE GENERAL	IV
ABREVIATURAS	VII
SIMBOLOGIA	VIII
ÍNDICE DE FIGURAS	IX
ÍNDICE DE TABLAS	X
INTRODUCCIÓN	1

I. INTRODUCCIÓN A LOS MÉTODOS METAHEURÍSTICOS

1.1 Introducción.....	1
-----------------------	---

II. MÉTODOS DE OPTIMIZACIÓN EXACTA EN NÚMEROS ENTEROS

2.1 Toma decisiones.....	9
2.2 Programación Lineal.....	11
2.3 Programación Entera, método de Branch and Bound.....	21
2.3.1 Programación Entera.....	21
2.3.2 Método de Branch & Bound.....	23
2.3.3 Comentarios Finales sobre el Método de Branch & Bound.....	35

III. TÉCNICAS DE OPTIMIZACIÓN META HEURÍSTICAS

3.1 Reseña histórica de la evolución de las técnicas de optimización meta heurísticas.....	36
3.2 Los principales métodos meta-heurísticos.....	43
3.2.1 Búsqueda Tabú.....	44



3.2.2 G.R.A.S.P.....	46
3.2.3 Algoritmos Genéticos.....	49

IV. MÉTODO RECOCIDO SIMULADO

4.1 Reseña histórica del Algoritmo Recocido Simulado y sus generalidades.....	54
4.2 Esquema general de un algoritmo Recocido Simulado.....	58
4.3 Convergencia del Algoritmo.....	62

V. DISEÑO, IMPLEMENTACIÓN Y APLICACIÓN DEL MÉTODO RECOCIDO SIMULADO

5.1 Aplicación a la Logística del método Recocido Simulado.....	67
5.1.1. El Problema de Ruteo de Vehículos.....	67
5.1.1.1 Precisión.....	71
5.1.1.2 Velocidad.....	72
5.1.1.3 Simplicidad.....	74
5.1.1.4 Flexibilidad.....	75
5.1.2 Formulaciones del V.R.P.....	77
5.1.2.1. Modelos de Flujo de Vehículos.....	78
5.1.2.2. Problemas de prueba para el Problema de Ruteo de Vehículos Capacitado (CVRP) y otras variantes del V.R.P.....	90
5.1.3 Parámetros principales del V.R.P.....	96
5.1.3.1. Vehículos.....	96
5.1.3.2 Clientes.....	97
5.1.3.3 Depósitos.....	98
5.1.3.4 Funciones Objetivos.....	98
5.2 Plataforma y Código de Programación.....	99
5.2.1 Diagrama de Flujo de Datos.....	99
5.3 Resultados Obtenidos.....	101



CONCLUSIONES Y RECOMENDACIONES

ANEXOS

BIBLIOGRAFÍA



ABREVIATURAS

B&B	Branch and Bound
PL	Programación Lineal
CI	Cota Inferior
PLE	Programación Lineal Entera
MT	Método Búsqueda Tabú
G.R.A.S.P.	Procedimientos de Búsqueda Adaptados basados en funciones "Glotonas" Aleatorizadas.
VRP	Problema de ruteo de vehículos
CVRP	Problema de ruteo de vehículos capacitado
GSEC	Generalizada eliminación de restricciones
CCC	Restricción de corte de capacidad
ACVRP	Problema de ruteo de vehículos asimétricos
SCVRP	Problema de ruteo de vehículos euclidianos
DCVRP	Problema de ruteo de vehículos simétricos

SIMBOLOGIA

e	Constante de Euler
Δ	Delta (mayúscula)
\mathbf{R}	Números Reales
α	Alfa
δ	Delta (minúscula)
∞	Infinito
$[\alpha]$	Entero mayor de α
Σ	Sumatoria
$\forall x$	Todos los elementos de x
$ x $	Valor absoluto de x

INDICE DE FIGURAS

Figura 1.1 Función con Mínimos Locales y Absolutos.....	7
Figura 2.1 Espacio Solución Factible para PL0.....	33
Figura 2.2 Espacio Solución Factible de PL1 y de PL2.....	33
Figura 2.3 Resolución del Problema focalizado en forma de árbol.....	34
Figura 4.1 Función con mínimos desigualmente espaciados.....	65
Figura 5.1 Ejemplo de V.R.P. Cronograma rutinario de los vehículos.....	69
Figura 5.2 Distribución de clientes.- Problema Tipo C.....	94
Figura 5.3 Distribución de clientes.- Problema Tipo R.....	95
Figura 5.3 Distribución de clientes.- Problema Tipo RC.....	95

INDICE DE TABLAS

Tabla 5.1 Símbolos utilizados en los D.F.D.'s..... 100



INTRODUCCIÓN

El transporte y su logística sostienen la actividad más social y económica en una comunidad. Durante los últimos años, su estudio ha llevado al desarrollo de varios modelos y algoritmos que también se han aplicado a una variedad de otros campos científicos e industriales.

El transporte es un campo complejo que involucra a varios factores y niveles de decisión, incertidumbres y gastos considerablemente importantes en una economía. La variedad y complejidad de este campo es reflejada por la riqueza de las investigaciones, modelos, software, y métodos realizados para resolver el problema de transporte.

El presente trabajo se considera como una guía teórica con una aplicación práctica del algoritmo recocido simulado para la optimización de problemas duros, también se observan las principales diferencias entre las técnicas de optimización meta heurísticas.

Con esto, se pretende dar a conocer los beneficios que se obtienen con una sincronización adecuada del empleo de las estadísticas, la programación computacional y el análisis matemático, en un tema tan trascendental como lo es en la investigación de operaciones resolviendo problemas de



optimización combinatoria como el ruteo de una flota de vehículos, y encontrando la solución óptima para la resolución de este tipo de problemas.



optimización combinatoria como el ruteo de una flota de vehículos, y encontrando la solución óptima para la resolución de este tipo de problemas.



CAPITULO I

1. INTRODUCCIÓN A LOS MODELOS META HEURÍSTICOS

1.1 Introducción

En los últimos años se han desarrollado métodos efectivos, denominados meta heurísticos para resolver problemas difíciles de optimización que se presentan en varias aplicaciones de la ingeniería. La siguiente tesis tiene como objetivo específico el análisis del método denominado recocido simulado.

Para la realización de este estudio primeramente describiremos de manera breve algunos conceptos de optimización, luego hablaremos más detalladamente de los fundamentos matemáticos y del algoritmo



de recocido simulado, para finalmente adaptar el algoritmo a una aplicación a la ingeniería.

El algoritmo de recocido simulado ha demostrado ser una opción bastante aceptable como técnica de optimización en problemas difíciles, ya que como es un algoritmo de aproximación, nos otorga unas buenas soluciones a los problemas de optimización con varias restricciones y en un tiempo prudencial. Generalmente los problemas de optimización de esta clase tienen como restricción que varias variables sólo pueden tomar números enteros.

Los desarrolladores de este método de optimización, se dieron cuenta que era posible establecer una analogía, entre los parámetros que intervienen en la simulación termodinámica de recocido de sólidos, y los que aparecen en la problemática de resolver problemas de optimización combinatoria de gran escala, es por esta circunstancia que el algoritmo se lo conoce como Recocido Simulado.



Como vamos a hablar de problemas de optimización combinatoria seria bueno dar algunos ejemplos de los mismo, a continuación se detallan algunos:

- Planificación de tareas
- Asignación de recursos y horarios en instituciones educativas
- Minimización de desperdicios en el corte de materiales
- Localización de plantas o de instalaciones
- Problema de la suma de subconjuntos
- Determinación de caminos mínimos en grafos
- Flujo en redes
- Asignación de tareas
- Ruteo de una flota de vehículos
- Flujo Multiproducto

Es por eso que definimos como proceso de optimización combinatoria, al proceso de hallar el valor óptimo (máximo o mínimo) de una función definida sobre un conjunto finito (pero muy grande) de puntos.

En ingeniería de sólidos, el recocido significa un proceso de calentamiento de un sólido, que puede ser por ejemplo el acero, a una temperatura en la que sus partículas deformadas recristalizan para producir nuevas partículas. La temperatura de recocido o de recristalización depende del tipo de material, del grado de deformación del mismo, además de su uso futuro. Seguida a la fase de calentamiento viene un proceso de enfriamiento en donde la temperatura se baja poco a poco. De esta manera, cada vez que se baja la temperatura, las partículas se acomodan en estados de más baja energía hasta que se obtiene un sólido con sus partículas acomodadas conforme a una estructura de cristal.

Si se comienza con un valor máximo de la temperatura, en la fase de enfriamiento del proceso de recocido, para cada valor de la temperatura T debe permitirse que se alcance su equilibrio térmico. Sin embargo, si el proceso de enfriamiento es demasiado rápido y no se alcanza en cada etapa el equilibrio térmico, el sólido congelará en un estado cuya estructura será amorfa en lugar de la estructura cristalina de más baja energía. La estructura amorfa está caracterizada por una imperfecta cristalización del sólido.



Inicialmente, este algoritmo, permite explorar una buena parte del espacio de estados, de tal forma que la solución final puede resultar insensible al estado inicial. En consecuencia, la probabilidad de quedar atrapado en un máximo o en un mínimo local, se hace cada vez más ínfima.

Así, en resumen, el recocido simulado es un algoritmo computacional que se basa en los pasos establecidos en el proceso físico de tratamiento térmico de materiales. Por su naturaleza, este algoritmo debe ser formulado como un descenso de pico en el que la función objetivo es el nivel energético.

Las sustancias físicas usualmente se mueven desde configuraciones de alta energía a las de menor energía, así que el descenso al valle, ocurre en forma natural. Pero, eventualmente pueden haber transiciones a niveles energéticos más altos como se verá luego, con una probabilidad dada por:

$$p = e^{\frac{-\Delta E}{kT}}$$
$$\Delta E = E_{nuevo} - E_{actual}$$

T: es la Temperatura absoluta, y

K: es la constante de Boltzmann.

El procedimiento que se va a seguir para enfriar el sistema, se llama programa recocido simulado. Su forma óptima depende de cada tipo de problema y usualmente se lo descubre empíricamente. Para que el programa recocido simulado se ejecute de una manera eficaz se deben cumplir los siguientes requerimientos.

- El valor inicial de la temperatura.
- El criterio que será utilizado para decidir cuando reducir la temperatura.
- La cantidad en que será reducida la temperatura.
- Cuando finalizar el proceso.

Observemos ahora la analogía de este procedimiento con el proceso de optimización, para esto analicemos un problema de optimización bastante simple, que será el de hallar el mínimo de una función de una variable real.

$$f: I \subseteq \mathbb{R} \rightarrow \mathbb{R}$$

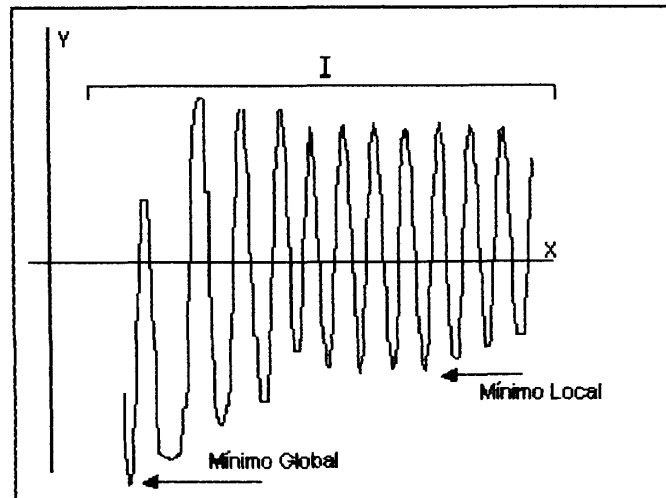


Figura 1.1 Función con Mínimos Locales y Globales

Tal como lo indica la flecha en la figura 1.1, ahí encontramos un mínimo local y es ahí donde todos los métodos de análisis numérico se entrampan y nos lo mostraría como el mínimo absoluto, y como claramente nos podemos dar cuenta esta solución hallada numéricamente no sería el mínimo global de la función en el intervalo I .

Pero el algoritmo Recocido Simulado, no se queda anclado en mínimos locales, sino que examina la vecindad de los puntos para

hallar el mínimo absoluto. El mismo tipo de problema se presenta en el caso de querer hallar un máximo. En el capítulo IV se hablará más ampliamente de los beneficios de este método.

Capítulo II

2. MÉTODOS DE OPTIMIZACIÓN EXACTA EN NÚMEROS ENTEROS

2.1 Toma de decisiones

Decidir significa elegir entre varias alternativas. Para dar una solución más completa, es conveniente construir las alternativas con una clara presentación de las mismas y su respectiva justificación, indicar que se necesita una decisión, e incluso, señalar la elección más recomendable justificándola adecuadamente.

Los factores intangibles del juicio humano vs. la toma de decisiones son un punto importante a tomar en cuenta. Estudios teóricos sobre toma de decisiones racionales, principalmente en el contexto de la teoría de la probabilidad y la teoría de la decisión, han ido



acompañadas de investigación empírica sobre si el comportamiento humano se ajusta a la teoría. Numerosos estudios empíricos indican que el juicio sobre las situaciones y la toma de decisiones humanas están basados en estrategias intuitivas (heurísticas) en vez de en reglas de razonamiento teóricamente sólidas. Con ello se pretende reducir la carga cognitiva.

El juicio y la elección de los humanos sin ayuda de modelación matemática, exhiben sistemáticas violaciones de los axiomas de la probabilidad.

La superioridad de los modelos, incluso lineales simples (como la programación lineal), sobre el juicio intuitivo humano sugiere que un modo de mejorar la calidad de las decisiones es descomponer el problema de la decisión en componentes simples que sean bien entendidos y definidos. El estudio de un sistema complejo construido con tales componentes puede ser ayudado por una técnica formal teóricamente sólida. Esto permite aplicar conocimiento científico que puede ser transferido a otros problemas de la vida diaria, permitiendo analizar, explicar y argumentar acerca del problema de decisión.

En las diferentes herramientas de modelado desarrolladas, el conocimiento acerca del sistema es representado por ecuaciones o reglas lógicas, mejoradas con una representación explícita de la incertidumbre.

La toma de decisiones bajo incertidumbre, introduce mejoras en base a aproximaciones estadísticas, tales como el análisis de la confiabilidad, la simulación, y la toma de decisiones basada en la teoría estadística. Una vez que un modelo ha sido formulado, pueden usarse una variedad de métodos matemáticos para analizarlo.

Uno de los métodos matemáticos de la rama de la investigación de operaciones más utilizado y con grandes aplicaciones para la toma de decisiones es la programación lineal, la cual la describiremos a continuación.

2.2 Programación Lineal

La programación lineal, es un proceso matemático que se ha venido desarrollando, ya hace algunos siglos atrás, y se conoce por escritos,



que los primeros indicios de estos fueron en los siglos XVII. Pero es en la década de 1.950, donde se forman principalmente en Estados Unidos, distintos grupos de estudio para ir solucionando y desarrollando las diferentes ramas de la programación lineal.

Las principales bases de la programación lineal se deben al matemático estadounidense Von Neuman (1903-1957), quien en el año de 1928 publicó su famoso trabajo *Teoría de Juegos*. Este grandioso matemático desde el año 1930, se ejerció como catedrático de la *Universidad de Princeton de Estados Unidos*, y es ahí donde impulsó que otros investigadores se interesen progresivamente en el desarrollo de esta rama de las matemáticas.

A continuación vamos a profundizar un poco más sobre los conceptos de la programación lineal, que como vamos a mencionar es una de las técnicas de mayor manejo dentro de la investigación de operaciones. En los modelos matemáticos de Investigación de operaciones, las variables de decisión pueden ser enteras o continuas, y el objetivo y las funciones de restricción son lineales o no lineales. Los problemas de optimización planteados por estos modelos dieron origen a una

variedad de métodos de solución, cada uno diseñado para tomar en cuenta las propiedades matemáticas especiales del modelo.

La más prominente y exitosa de estas técnicas es la programación lineal, en la cual todas las funciones, el objetivo y las restricciones son lineales y todas las variables son continuas, es por esta razón que le vamos a dedicar esta parte del segundo capítulo a tratar con mayor profundidad la programación lineal.

La programación lineal es una técnica de modelación matemática, diseñada para optimizar el empleo de recursos limitados. Problemas de programación lineal ocurren en muchos casos de la vida económica real, situaciones donde las utilidades son maximizadas y los costos son minimizados con sus respectivas restricciones. Por ejemplo, una empresa siempre quiere minimizar costos pero sin embargo tiene una restricción que es, que de igual manera debe cumplir su meta de ventas anuales o mensuales y esto indirectamente implicaría aumento de costos. La programación lineal se aplica exitosamente también en varios campos tales como el ejército, la



agricultura, la industria, la transportación, la economía, los sistemas de salud, e incluso, en las ciencias sociales.

La utilidad de la técnica se incrementa mediante la disponibilidad de programas de computadoras altamente eficientes. De hecho la programación lineal, debido a su elevado nivel de eficiencia computacional, es la base para el desarrollo de algoritmos de solución de otros tipos más complejos de modelos de investigación operacional, incluyendo la programación entera, no lineal y estocástica.

Los cálculos en la programación lineal, igual que en la mayor parte de los modelos de investigación operacional, por lo común son voluminosos y tediosos y, por consiguiente, requieren el empleo de la computadora. De hecho, todas las técnicas de investigación de operaciones dan por resultado algoritmos computacionales que son de naturaleza iterativa, es decir, que el problema se resuelve en iteraciones y que cada nueva iteración lleva la solución más cerca de la óptima. La naturaleza iterativa de los algoritmos por lo general da origen a que estos cálculos se realicen por computadoras

Un modelo de programación lineal, igualmente que en cualquier modelo de investigación operacional, incluye tres elementos básicos:

- 1.- **Variables** de decisión que tratamos determinar
- 2.- **Objetivo** (meta) que tratamos de optimizar
- 3.- **Restricciones** que necesitamos respetar

Un primer paso esencial hacia el desarrollo del modelo es la definición apropiada de las variables de decisión. Una vez que se han definido correctamente las variables, tendremos como siguiente tarea construir la función objetivo, para luego seguir con las restricciones, las cuales no deberían ser de mayor dificultad ya que están basadas por el entorno y nos damos claramente cuenta de ellas.

Entonces finalmente un modelo matemático típico de investigación de operaciones por lo común se organiza como sigue:

Maximizar o minimizar (Función objetivo)

Sujeto a (Restricciones)

También es cierto que los problemas de toma de decisiones, como lo habíamos dicho en la primera parte de este capítulo, por lo común incluyen importantes factores intangibles, que tal vez no son fácilmente cuantificables. El principal entre estos factores es la presencia del elemento humano en la mayor parte de los ambientes de toma de decisiones. De hecho, se han reportado situaciones de decisiones donde el efecto de la conducta humana ha influido a tal grado en el problema de decisiones, que la solución obtenida del modelo matemático se considera impracticable.

Una ilustración de estos casos es una versión del *problema del ascensor*, que ha circulado ampliamente. En respuesta a las quejas de los inquilinos acerca del servicio tan lento del ascensor en un edificio de oficinas muy grande, se analizó la situación utilizando un modelos de líneas de espera. Sin embargo, la solución propuesta para apresurar el servicio del ascensor no mitigó el problema. Un estudio adicional de la situación reveló que las quejas de los inquilinos eran más bien un caso de tedio y el problema se resolvió utilizando espejo de cuerpo entero en la entrada al ascensor.

Las quejas desaparecieron cuando los usuarios del ascensor se mantenían ocupados contemplándose a si mismos y a los demás mientras esperaban el servicio del ascensor.

Resulta difícil prescribir cursos de acción específicos para estos factores intangibles, es a causa de esta razón, que solo podemos ofrecer pautas generales para la puesta en practica de la investigación operacional.

Las etapas principales para la puesta en practica de la investigación de operaciones incluyen:

1. Definiciones del problema
2. Construcción del modelo
3. Solución del modelo
4. Validación del modelo
5. Puesta en practica de la solución

Entre la cinco etapas anteriormente descritas solo la tercera, concerniente a la solución del modelo, la que mejor se define y la más fácil de poner en practica en un estudio de investigación de operaciones, debido a que aborda en su mayor parte, teoría

matemática más precisa. Aplicar el resto de las etapas es más un arte que una teoría.

En consecuencia, no podemos prescribir procedimientos para la ejecución de estas etapas.

La primera etapa que es la definición del problema implica establecer el alcance del problema que se está investigando. Esta es una función que debe desempeñar todo el equipo de investigación de operaciones

La construcción del modelo implica traducir la definición del problema de relaciones matemáticas como las vamos a establecer más adelante. Si el modelo resultante se ajusta en uno de los modelos matemáticos estándar, como programación lineal, por lo común es posible encontrar una solución utilizando los algoritmos disponibles. Como una alternativa, si las relaciones matemáticas son demasiado complejas para permitir la determinación de una solución analítica, el equipo de investigación de operaciones puede optar por simplificar el modelo y emplear un enfoque *meta heurístico*, si es apropiado.



La solución del modelo es la parte más sencilla de todas las etapas de investigación de operaciones, debido a que implica el empleo de algoritmos de optimización bien definidos.

La validación del modelo verifica si el modelo propuesto hace lo que ese supone que debe hacer, es decir, ¿el modelo proporciona un predicción razonable del comportamiento del sistema que se está estudiando?. Inicialmente, el equipo de investigación de operaciones debe estar convencido de que el resultado del modelo no contiene "sorpresas".

En otras palabras, ¿Tiene sentido la solución? ¿Los resultados son intuitivamente aceptables?

La puesta en práctica de la solución de un modelo validado implica la traducción de los resultados del modelo a instrucciones de operación, impartidas de una forma que sea comprensible par los individuos que administrarán el sistema recomendado.

Ahora, si analizamos matemáticamente la programación lineal, su forma general que tiene una función objetivo que va a ser minimizada o maximizada, y su conjunto de restricciones sería:

$$\text{Maximizar } C_1X_1 + C_2X_2 + C_3X_3 + \dots + C_nX_n$$

Sujeto a:

$$a_{i1}X_1 + a_{i2}X_2 + a_{i3}X_3 + \dots + a_{in}X_n \leq b_i \quad \text{Restricciones de holgura (} i=1..l)$$

$$a_{j1}X_1 + a_{j2}X_2 + a_{j3}X_3 + \dots + a_{jn}X_n \geq b_j \quad \text{Restricciones de superávit}$$

($j=l+1..l+r$)

$$a_{k1}X_1 + a_{k2}X_2 + a_{k3}X_3 + \dots + a_{kn}X_n = b_k \quad \text{Restricciones de igualdad}$$

($k=l+r+1..l+r+q$)

$$X_1 \geq 0, X_2 \geq 0, \dots, X_n \geq 0$$

Podemos ver que el número de restricciones es $m = l + r + q$, c_j y a_{ij} son coeficientes constantes, los b_i son constantes reales las cuales han sido ajustadas para ser no negativas, y las X_j son las variables a



ser determinadas. En aplicaciones a la ingeniería, las X_j son llamadas como variables de diseño.

Otras técnicas de investigación de operaciones aparte de la programación lineal que abordan otros tipos de modelos matemáticos son la programación dinámica, la programación entera, la programación no lineal y la programación de redes, para mencionar ligeramente solo unas cuantas entre ellas.

2.3 Programación Entera, método de Branch and Bound

2.3.1. Programación Entera

Algunas veces hemos podido observar en casos de la vida real, que una de las más grandes limitaciones dentro de la programación lineal, es que tanto la función objetivo como las restricciones deben ser variables continuas, y es ahí donde muchas aplicaciones no se pueden realizar efectivamente, ya que en muchos procesos de

optimización se requiere asignar personas, animales o cosas, las cuales deberían venir dadas en variables netamente enteras.

En ese instante, cuando tenemos la necesidad de satisfacer problemas de optimización con variables enteras, es donde nace el problema de optimización conocido como Programación Lineal Entera, en el cual vamos a inmiscuirnos a continuación.

Los programas lineales enteros son programas lineales en los cuales algunas o todas las variables están restringidas a valores enteros o discretos. A pesar de décadas de extensas investigaciones, la experiencia de los cálculos con programación lineal entera ha sido algo menos que satisfactoria. A la fecha, no existe un programa de computadora de programación lineal entera que pueda resolver de manera exacta los problemas de programación entera de una manera uniforme.

Se debe tener en cuenta que cuando se restringe las variables de decisión a tomar solo valores enteros, la solución óptima entera que



se encuentre será menos eficiente que la decimal, pero la mejor entera posible.

En esta parte del capítulo vamos a explorar métodos que nos van a proveer soluciones óptimas enteras. Además el algoritmo recocido simulado y los algoritmos genéticos también pueden ser usados.

2.3.2. Método de Branch & Bound

Uno de los métodos de Programación Entera más usado y que ha tenido mayor cantidad de aplicaciones, es sin duda alguna el método de *Branch & Bound* (B&B), al que se refieren a veces en español como método de Ramificación y Acotamiento. B&B es un método de optimización exacto que utiliza a la programación lineal como herramienta para obtener una solución del problema sin tomar en cuenta la restricción de que las variables deben tomar solamente valores enteros. Aquí describiremos B&B brevemente para problemas de maximización debido que para problemas de minimización el proceso es similar.

El enfoque de Branch and Bound es resolver el problema como si se tratara de un problema de programación lineal, sin tomar en cuenta que las variables deben ser enteras. Branch and Bound empieza entonces utilizando alguna rutina de Programación Lineal, el cual a partir de ahora lo llamaremos PL. A esta iteración le llamaremos la iteración inicial. Si en la iteración inicial, la solución es entera, el problema se ha resuelto, de modo que la solución obtenida con PL es la solución óptima del problema entero. De otra manera, habrá que continuar buscando la solución del problema entero.

Si en la iteración inicial, la solución obtenida no es entera, se procede a seleccionar una variable no entera X_i . Se encuentran los dos enteros más próximos de X_i . Sean estos enteros X_{menor} y X_{mayor} los enteros más pequeño y más grande que X_i , respectivamente. B&B forma entonces dos nuevas restricciones que agrega una a una al problema original, de modo que se obtienen dos nuevos problemas. A la obtención de nuevos problemas se le llama ramificación. Acto seguido se procede a buscar una solución entera en cada uno de



estos problemas, para lo cual se procede en cada uno de ellos como en la iteración inicial, usando PL para resolver los problemas de programación lineal correspondientes.

B&B utiliza para su análisis una cota inferior (para problemas de maximización), que denominaremos aquí CI. B&B solo examina aquellos problemas cuya solución proporcionada por PL estén por arriba de CI. De esta forma es necesario que el algoritmo de B&B determine una CI lo más pronto posible. Una forma común es empezar con una CI para los valores de las X's iguales a su cota inferior.

No obstante existen rutinas de B&B que esperan a obtener una CI igual a la primera solución que solo tenga valores enteros. Nótese que en el caso de la iteración inicial el proceso termina si la solución tiene solo valores enteros. B&B cambia su CI cuando detecta que existe un problema con una solución entera cuyo valor de la función objetivo es mayor que la CI actual.



Una vez que B&B ha determinado una CI, si esta no corresponde a la iteración inicial, procede a obtener nuevos problemas mediante el proceso de partición.

Solamente se analizan aquellos problemas que tengan como solución un valor de la función objetivo mayor que la última CI determinada. B&B termina cuando no existen más problemas por analizar.

Ahora vamos a resumir el algoritmo de B&B de manera un poco más estructurada. Suponiendo un problema de maximización, determinaremos una cota inferior inicial $z = -\infty$ en el valor objetivo óptimo de Programación lineal entera (PLE). Determine $i=0$

Paso 1.- (Acotar) Seleccione PL_i , el siguiente sub-problema a ser examinado. Resuelva PL_i y trate de sondearlo, utilizando una de tres condiciones.

- a) El valor óptimo z de PL_i no puede producir un mejor valor objetivo que la cota inferior actual.



- b) PL_i produce una solución entera factible mejor que la cota inferior actual.
- c) PL_i no tiene una solución factible.

Se presentaran dos casos:

- a) Si se sondea PL_i , actualice la cota inferior si se encuentra una mejor solución del PLE. Si se han sondeado todos los sub-problemas, deténgase; el PLE óptimo esta asociada con la cota inferior actual, si la hay. De otra manera, haga $i = i+1$, y repita el paso 1.
- b) Si no se sondea PL_i , vaya el paso 2 para efectuar la ramificación.

Paso2.-(Ramificación) Seleccione una de las variables enteras X_p cuyo valor optimo x_j^* en la solución de PL_i no es un entero. Elimine la región $[x_j^*] < x_j < [x_j^*] + 1$ (donde $[\alpha]$ es el entero más grande $\leq \alpha$) creando dos sub-problemas de PL que corresponden a

$$x_j \leq [x_j^*] \quad \text{y} \quad x_j \geq [x_j^*] + 1$$

Determine $i = i + 1$, y vaya al paso 1

Los pasos dados aplican a los problemas de maximización. Para minimización, reemplazamos la cota inferior con una cota superior cuyo valor inicial es $z = +\infty$.

El algoritmo de B&B se puede extender directamente a los problemas mixtos en los cuales sólo algunas variables son enteras. Si una variable es continua, simplemente nunca la seleccionamos como una variable de ramificación. Un sub-problema factible proporciona una nueva cota sobre el valor objetivo si los valores de las variables discretas son enteros y el valor objetivo se mejora en relación con la cota actual.

Para entender con mayor facilidad el método de Branch and Bound se explicará mediante un ejemplo numérico.

Considere el siguiente problema detallado a continuación:

$$\text{Maximize } Z = 5X_1 + 4X_2$$

Sujeta a

$$X_1 + X_2 \leq 5$$

$$10X_1 + 6X_2 \leq 45$$

$X_1, X_2 \geq 0$ y enteras

Este problema asociado de Programación Lineal (PL), PL0, se define quitando las restricciones enteras y sus solución óptima se da como:

$$X_1 = 3.75, X_2 = 1.25, \text{ y } Z = 23.75$$

Debido a que la solución óptima no satisface los requerimientos enteros, el algoritmo de Branch and Bound modifica el espacio de la solución de tal manera que a la larga identifica la óptima de Programa Lineal entero (PLE).

Siguiendo los pasos estructurados en el algoritmo, primero seleccionamos una de las variables enteras cuyo valor en PL0 no es entero. Al seleccionar arbitrariamente $X_1 = 3.75$ la región $3 > X_1 > 4$; del espacio de la solución PL0 ya no contiene valores enteros de X_1 , y ,

por tanto, se pueden eliminar como no prometedores. Esto es equivalente a reemplazar la PL0 original con dos nuevas PL, PL1, y PL2, definidos como

$$\text{Espacio PL1} = \text{Espacio PL0} + (X_1 \leq 3)$$

$$\text{Espacio PL2} = \text{Espacio PL0} + (X_1 \geq 4)$$

Los espacios de PL1 y PL2 contienen los mismos puntos enteros factibles del PLE original, lo que significa que tratar con PL1 Y PL2 es lo mismo que tratar con la PL0 original.

Si seguimos eliminando de la *mejor* manera las regiones que no incluyen soluciones enteras, imponiendo las restricciones apropiadas tales como, $3 < X_1 < 4$, a la larga produciremos PL_s cuyos puntos extremos óptimos satisfagan las restricciones enteras. En efecto, estaremos resolviendo el PLE al tratar con una sucesión de PL (continuas).

Las nuevas restricciones $X_1 \leq 3$ y $X_1 \geq 4$, se excluyen mutuamente, de manera que PL1 y PL2 se deben tratar como PL separados. Esta dicotomía da origen al concepto de la ramificación en el algoritmo de Branch and Bound. En este caso, X_1 es la variable de ramificación.

El PLE óptimo se encuentra ya sea en PL1 o en PL2. Por tanto, se deben examinar ambos sub problemas. Primero examinemos arbitrariamente PL1 (asociada con $X_1 \leq 3$)

$$\text{Maximize } Z = 5X_1 + 4X_2$$

Sujeta a

$$X_1 + X_2 \leq 5$$

$$10X_1 + 6X_2 \leq 45$$

$$X_1 \leq 3; X_1, X_2 \geq 0$$



La solución PL1 produce la solución óptima $X_1 = 3$, $X_2 = 2$, y $Z = 23$.

La solución de PL1 satisface los requerimientos de enteros para X_1 y X_2 , por lo tanto, ya no es necesario investigar más a PL1, porque no incluye ninguna solución mejor de PLE.

En este punto no podemos juzgar la "calidad" de la solución entera de PL1, ya que PL2 puede producir una solución entera con un valor más elevado de z , es decir, una solución entera mejor. Lo que si podemos decir es que $Z = 23$ es una cota inferior sobre el valor objetivo(máximo) del PLE original. Esto significa que cualquier sub - problema no examinado que no puede producir una valor objetivo mejor se debe descartar como no prometedor. Si un sub-problema no examinado puede producir una solución entera mejor, entonces la cota inferior se debe actualizar posteriormente.

Dada la cota inferior $Z = 23$, examinamos PL2, en este caso, el único sub problema restante no examinado. Debido a que la óptima $Z = 23$ en PL0 y a que todos los coeficientes de la función objetivo son enteros, es imposible que PL2, que es más restrictiva que PL0,

producirá una solución entera mejor. Como resultado final, descartaremos PL2 y concluimos el proceso.

El método de Branch and Bound ahora está completo, por que se han examinado y sondeado tanto PL1 como PL2, la primera para producir una solución entera y la segunda para mostrar que no puede producir una solución entera mejor. Por ello, concluimos que la solución óptima del PLE es la asociada con la cota inferior, es decir, $X_1 = 3$, $X_2 = 2$ y $Z = 23$.

Además el procedimiento del problema anterior se puede ver en los siguientes gráficos:

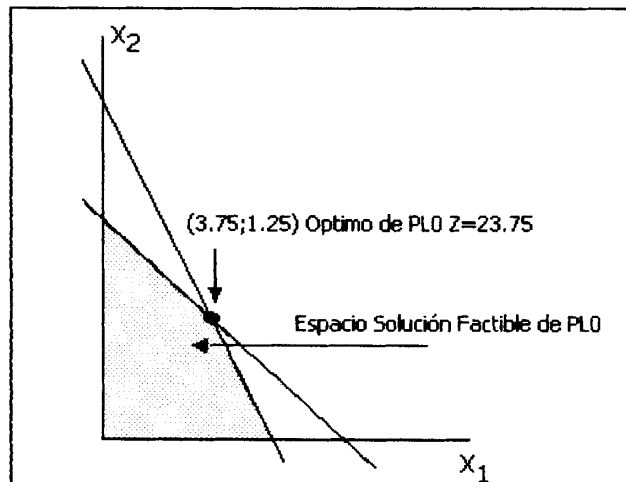


Figura 2.1 Espacio Solución Factible para PL0

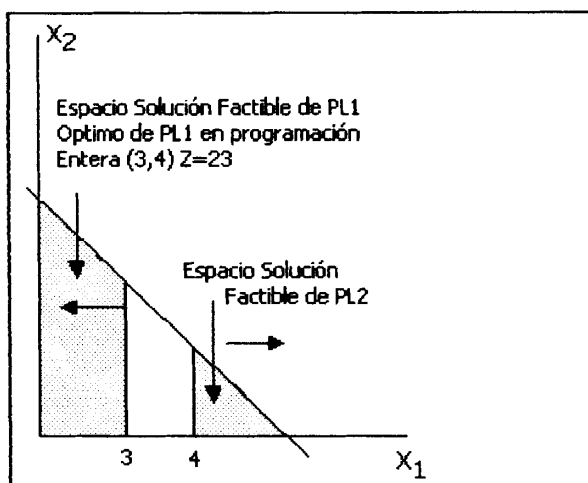


Figura 2.2 Espacio Solución Factible de PL1 y de PL2

El siguiente árbol también nos describe el desarrollo del método

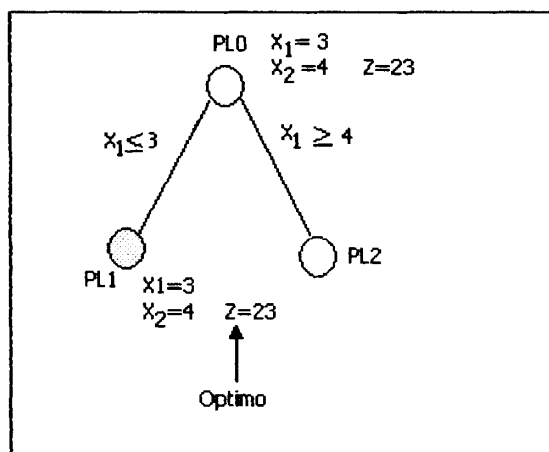


Figura 2.3 Resolución del Problema focalizado en forma de árbol

2.3.3. Comentarios Finales sobre el Método de Branch & Bound

Al momento de decidir que variable utilizar para la ramificación no es trivial, es una cuestión que depende de la estructura del problema. Los problemas enteros puros se resuelven más eficazmente que los enteros mixtos. En la actualidad se pueden resolver eficazmente los problemas de programación Entera de más de 10.000 variables, siempre que tengan una estructura especial que se pueda explotar



Capítulo III

3. TÉCNICAS DE OPTIMIZACIÓN META HEURÍSTICAS

3.1 Reseña histórica de la evolución de las técnicas de optimización meta heurísticas

Existen tipos de problemas que no pueden resolverse de manera exacta usando un algoritmo que requiere tiempo polinomial. De hecho, en muchas aplicaciones prácticas, no podemos siquiera decir si existe una solución eficiente del mismo.

En pocas palabras, existe una diversa cantidad de problemas para los cuales el mejor algoritmo que se conoce para obtener una solución, requiere de tiempo exponencial para resolverlo.

Es en ese momento, cuando tenemos la necesidad de resolver estos problemas “*diffciles*”, donde tenemos que usar métodos denominados heurísticos.

A continuación, vamos a explicar los diferentes tipos de problemas de optimización que existen, para determinar en cuales es necesario desarrollar un algoritmo heurístico.

- Clase P
- Clase NP

Clase P

Un problema pertenece a la clase P si agrupa el conjunto de problemas de decisión y existe un algoritmo de resolución para el mismo que puede ser resuelto en tiempo polinomial en una computadora determinística, es decir, capaz de dar para cualquier instancia la respuesta en un intervalo de tiempo prudente.

También se puede entender como que la clase P agrupa el conjunto de problemas de decisión “*fáciles*”, es decir, aquellos para los cuales



existe al menos un algoritmo eficaz. La clase P puede ser formalizada muchos más rigurosamente introduciendo un formalismo matemático como la Máquina de Turing.

Existen problemas de optimización combinatoria que pertenecen a la clase P, estos son los problemas para los que existe al menos un algoritmo de complejidad polinomial para encontrar una versión de decisión o una solución óptima.

Como acotación podemos decir que los problemas de la programación lineal pertenece a la clase P, el método Simplex de Dantzig(1.947) funciona muy bien en la práctica, pero no es de complejidad polinomial, sin embargo el algoritmo de punto interior de Karmarkar para la programación lineal si es de complejidad polinomial.

Clase NP

Los problemas NP-Complejos son aquellos cuya complejidad es descrita por una función no polinomial y no pueden ser ejecutados para entradas grandes en una cantidad de tiempo razonable, y por tanto son de poca utilidad excepto para entradas pequeñas.



Más adelante vamos a tratar los problemas cuya complejidad es descrita por funciones exponenciales, problemas para los cuales el mejor algoritmo conocido requeriría de muchos años o centurias de tiempo de cálculo para entradas moderadamente grandes. De esta forma se presentarán definiciones que pretenden distinguir entre los problemas tratables (aquellos que no son tan duros) y los problemas intratables (duros o que consumen mucho tiempo). La mayoría de estos problemas ocurren como problemas de optimización combinatoria.

Clases P vs. NP

La clase P contiene problemas que pueden resolverse rápidamente, mientras que como hemos podido observar, la clase NP contiene problemas cuya solución puede requerir un tiempo demasiado largo. Alrededor de los años '70, varios investigadores se plantearon la interrogante: ¿Es $P = NP$? Desde entonces, sigue siendo una pregunta abierta para los teóricos. Cabe mencionar, que hasta el momento la conjetura es que $P \neq NP$.

Problemas NP Completos

Todos los algoritmos requeridos para resolverlos requieren tiempo exponencial en el peor de los casos, es decir, estos problemas son sumamente difíciles de resolver.

Estos problemas "difíciles" se los denomina NP-duros o NP-completos, debido a sus siglas en inglés (Non Polinomial), es decir, problemas de optimización que no pueden ser resueltos por un algoritmo en tiempo polinomial.

Un problema de optimización se dice NP-duro, si el problema decisión que esta asociado a el, es NP-completo. Un problema P se lo reconoce como de la clase NP-duro o NP-completo, si todo el problema en la clase NP se reduce polinomialmente a la clase P.

Para el desarrollo de este tipo de problemas no se pueden aplicar métodos exactos, pues su resolución requeriría de un tiempo extremadamente largo (años o incluso siglos), de ahí la importancia de los métodos heurísticos para lograr su resolución.



La palabra "heurística" se deriva del griego heuriskein, que significa "encontrar" o "descubrir", sin embargo, el significado del término ha variado históricamente. Algunos han usado el término como un antónimo de "algorítmico". Por ejemplo, Newell dice: "A un proceso que puede resolver un cierto problema, pero que no ofrece ninguna garantía de lograrlo, se le denomina una 'heurística' para ese problema". Las heurísticas fueron un área predominante en los orígenes de la Inteligencia Artificial.

Actualmente, el término suele usarse como un adjetivo, refiriéndose a cualquier técnica que mejore el desempeño en promedio de la solución de un problema, aunque no mejore necesariamente el desempeño en el peor de los casos (Russell & Norvig, 1995).

Una definición más precisa y adecuada para el desarrollo de esta tesis es la proporcionada por Reeves (1993): "Un algoritmo heurístico es una técnica que busca soluciones "BUENAS", es decir, casi óptimas, y que lo logra a un costo computacional razonable, aunque sin garantizar optimalidad o factibilidad de las mismas". En algunos

casos, ni siquiera puede determinar qué tan cerca del óptimo se encuentra una solución factible en particular.

Muchos investigadores llegaron a pensar en cierto instante que las heurísticas no eran necesarias, pero se dieron cuenta que cuando enfrentamos espacios de búsqueda muy grandes, y que además los algoritmos más eficientes que existen para resolver el problema requieren tiempo exponencial, resulta obvio pensar que las técnicas clásicas de búsqueda y optimización son insuficientes.

Los procedimientos Meta heurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son ni efectivos ni eficientes.

Las técnicas Meta heurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de: inteligencia artificial, evolución biológica y mecanismos estadísticos.

Los métodos meta-heurísticos dieron sus primeras apariciones en la década de los 50 con el método Recocido Simulado, específicamente se registraron en 1.953 los primeros trabajos de este método.

Luego en la década de los 60 nacieron las primeras ideas de los Algoritmos Genéticos, en los siguientes años se desarrollo el método de Búsqueda Tabú con sus primeras aplicaciones a optimización combinatoria en 1.986, basado en algunas ideas de los años 70. En el año de 1.989 también tuvieron lugar las primeras apariciones del método del G.R.A.S.P. y de igual manera el método Búsqueda con umbral.

3.2 Los principales métodos meta-heurísticos

En esta parte del capítulo vamos a hablar más acerca de los diferentes métodos meta-heurísticos y describir cada uno de ellos.

3.2.1 Búsqueda Tabú

Los orígenes del Método Búsqueda Tabú (M.T.) puede situarse en diversos trabajos publicados hace alrededor de 20 años específicamente en los años 1.986. Oficialmente, el nombre y la metodología fueron introducidos posteriormente por Fred Glover (1989).

El Método Búsqueda Tabú es una técnica para resolver problemas combinatorios de gran dificultad que está basada en principios generales de la Inteligencia Artificial. En esencia es un meta-heurístico que puede ser utilizado para guiar cualquier procedimiento de búsqueda local.

Una de las principales características del M.T. es la de poseer una memoria flexible de búsqueda. Básicamente, el empleo de dicha memoria flexible consiste en modificar, restringir y expandir sobre la marcha el criterio de vecindad, de acuerdo a observaciones realizadas durante el proceso de búsqueda.

Las principales características del Método de Búsqueda Tabú es que permite elegir una solución vecina que no sea estrictamente mejor que la actual para "salir" de un mínimo local. Además, este método usa una lista Tabú de soluciones o movimientos para evitar que el algoritmo cicle en si mismo y usa una función de aspiración que permita en algunos casos elegir un elemento o movimiento Tabú.

El método Búsqueda Tabú comienza de la misma forma que cualquier procedimiento de búsqueda local, procediendo iterativamente de una solución x a otra y en el entorno de la primera: $N(x)$. Sin embargo, en lugar de considerar todo el entorno de una solución, MT define el entorno reducido $N^*(x)$ como aquellas soluciones disponibles del entorno de x . Así, se considera que a partir de x , sólo las soluciones del entorno reducido son alcanzables.

Los requerimientos para usar este método serían:

- Determinar el conjunto de soluciones factibles S .
- Determinar la función objetivo f .

- Definir el tamaño de la lista Tabú T.
- De ser posible definir una cota inferior para la función objetivo f.
- Definir criterios de parada

3.2.2 G.R.A.S.P.

Los métodos G.R.A.S.P. fueron desarrollados al final de la década de 1.980 con el objetivo inicial de resolver problemas de recubrimiento de conjuntos (Feo y Resende, 1989). El término G.R.A.S.P. fue introducido por Feo y Resende (1995) como una nueva técnica meta-heurística de propósito general.

La palabra G.R.A.S.P. proviene de las siglas de Greedy Randomized Adaptive Search Procedures que en castellano sería algo así como: Procedimientos de Búsqueda Adaptados basados en funciones "Glotonas" Aleatorizadas.

G.R.A.S.P. es un procedimiento iterativo en donde cada paso consiste en una fase de construcción y una de mejora. En la fase de



construcción se aplica un procedimiento heurístico constructivo para obtener una buena solución inicial. Esta solución se mejora en la segunda fase mediante un algoritmo de búsqueda local. La mejor de todas las soluciones examinadas se guarda como resultado final.

A continuación detallamos los elementos de este procedimiento.

En la fase de construcción se construye iterativamente una solución posible, considerando un elemento en cada paso. En cada iteración la elección del próximo elemento para ser añadido a la solución parcial viene determinada por una función glotona. Esta función mide el beneficio de añadir cada uno de los elementos según la función objetivo y elegir la mejor. Cabe recalcar que esta medida es miope en el sentido que no tiene en cuenta qué ocurrirá en iteraciones sucesivas al realizar una elección, sino únicamente en esta iteración.

Se dice que el heurístico G.R.A.S.P. se adapta porque en cada iteración se actualizan los beneficios obtenidos al añadir el elemento seleccionado a la solución parcial. Es decir, la evaluación que se

tenga de añadir un determinado elemento a la solución en la iteración j , no coincidirá necesariamente con la que se tenga en la iteración $j+1$.

El siguiente esquema muestra el funcionamiento global del algoritmo:

Mientras (Condición de parada)

Fase Constructiva

- Seleccionar una lista de elementos candidatos.
- Considerar una lista restringida de los mejores candidatos.
- Seleccionar un elemento aleatoriamente de la lista restringida.

Fase de Mejora

- Realizar un proceso de búsqueda local a partir de la solución construida hasta que no se pueda mejorar más.

Actualización

- Si la solución obtenida mejora a la mejor almacenada, actualizarla.

Una de las ventajas del método G.R.A.S.P., es que al realizar muchas iteraciones es una forma de realizar un muestreo del espacio de soluciones.

3.2.3 Algoritmos Genéticos

Los Algoritmos Genéticos fueron introducidos por John Holland en 1970 inspirándose en el proceso observado en la evolución natural de los seres vivos.

Los Biólogos han estudiado en profundidad los mecanismos de la evolución, y aunque quedan parcelas por entender, muchos aspectos están bastante explicados. De manera muy general podemos decir que en la evolución de los seres vivos el problema al que cada individuo se enfrenta cada día es la supervivencia. Para ello cuenta con las habilidades innatas provistas en su material genético. A nivel de los genes, el problema es el de buscar aquellas adaptaciones beneficiosas en un medio hostil y cambiante. Debido en parte a la selección natural, cada especie gana una cierta cantidad de

"conocimiento", el cual es incorporado a la información de sus cromosomas.

Así pues, la evolución tiene lugar en los cromosomas, en donde está codificada la información del ser vivo. La información almacenada en el cromosoma varía de unas generaciones a otras. En el proceso de formación de un nuevo individuo, se combina la información cromosómica de los progenitores aunque la forma exacta en que se realiza es aún desconocida.

Aunque muchos aspectos están todavía por discernir, existen unos principios generales ampliamente aceptados por la comunidad científica.

Algunos de estos son:

1. La evolución opera en los cromosomas en lugar de en los individuos a los que representan.



2. La selección natural es el proceso por el que los cromosomas con "buenas estructuras" se reproducen más a menudo que los demás.
3. En el proceso de reproducción tiene lugar la evolución mediante la combinación de los cromosomas de los progenitores. Llamamos Recombinación a este proceso en el que se forma el cromosoma del descendiente. También son de tener en cuenta las mutaciones que pueden alterar dichos códigos.
4. La evolución biológica no tiene memoria en el sentido de que en la formación de los cromosomas únicamente se considera la información del período anterior

Los algoritmos genéticos establecen una analogía entre el conjunto de soluciones de un problema y el conjunto de individuos de una población natural, codificando la información de cada solución en un vector binario a modo de cromosoma. En palabras del propio Holland: *"Se pueden encontrar soluciones aproximadas a problemas de gran complejidad computacional mediante un proceso de evolución simulada"*.

A tal efecto se introduce una función de evaluación de los cromosomas, que llamaremos calidad ("fitness") y que está basada en la función objetivo del problema. Igualmente se introduce un mecanismo de selección de manera que los cromosomas con mejor evaluación sean escogidos para "*reproducirse*" más a menudo que los que la tienen peor.

Los algoritmos desarrollados por Holland inicialmente eran sencillos pero dieron buenos resultados en problemas considerados difíciles. Los Algoritmos Genéticos están basados en integrar e implementar eficientemente dos ideas fundamentales: Las representaciones simples como vectores binarios de las soluciones del problema y la realización de transformaciones simples para modificar y mejorar estas representaciones.

Para llevar a la práctica el esquema anterior y concretarlo en un algoritmo, hay que especificar los siguientes elementos:

- Una representación cromosómica



- Una población inicial
- Una medida de evaluación
- Un criterio de selección / eliminación de cromosomas
- Una o varias operaciones de recombinación
- Una o varias operaciones de mutación

Capítulo IV

4. MÉTODO RECOCIDO SIMULADO

4.1 Reseña histórica del Algoritmo Recocido Simulado y sus generalidades

Las primeras ideas que dieron origen a la técnica meta-heurística conocida como Recocido simulado fueron descritas en el año de 1.953, con la publicación de "Equation of state calculation by fast computing machines", J. of Chemistry Physics, Metropolis, A., Rosenbluth, M., Rosenbluth A., Teller, A., Teller, E, y otros trabajos que evolucionaron lentamente hasta que los investigadores Kirkpatrick, Gelatt y Vecchi proponen en 1.983 un procedimiento para obtener soluciones aproximadas a problemas de optimización, llamado Recocido Simulado, con su publicación de "Optimization by simulating

annealing". Este procedimiento se basa en una analogía con el comportamiento de un sistema físico al someterlo a un baño de agua caliente. Recocido Simulado ha sido probado con éxito en numerosos problemas de optimización, mostrando gran "habilidad" para evitar quedar atrapado en óptimos locales. Debido a su sencillez de implementación así como a los "buenos" resultados que iban apareciendo, experimentó un gran auge en la década de los 80.

Pero son finalmente los investigadores Kirkpatrick, y Cerny, en el año 1.985 que propusieron en forma independiente usar este algoritmo en problemas de optimización combinatoria, haciendo un paralelo entre el proceso de enfriamiento y los elementos de este tipo de problemas.

Como un esquema general, es un algoritmo para simular el proceso de enfriamiento de un material en un baño de calor. Si un material sólido se calienta por encima de su punto de fundición y después se enfría hasta quedar en estado sólido, las propiedades de este sólido dependen de la velocidad de enfriamiento.

Una de las diferencias de este algoritmo con otros, es que este algoritmo propone mejorar el algoritmo básico de descenso o

búsqueda local, permitiendo movimientos donde el valor de la función empeora con una frecuencia gobernada por una función de probabilidad que cambia a lo largo del proceso.

Kirpatrick y otros estudiosos trabajaron en el diseño de circuitos electrónicos considerando aplicar el algoritmo de Metrópolis en alguno de los problemas de optimización combinatoria que aparecen en este tipo de diseños.

Para ello, pensaron que era posible establecer una analogía entre los parámetros que intervienen en la simulación termodinámica y los que aparecen en los métodos de optimización local, tal y como los mostramos a continuación:

Termodinámica vs. Optimización

- Configuración, Solución Factible
- Configuración Fundamental, Solución óptima
- Energía de la Configuración, Costo de la Solución

Se establece un paralelismo entre el proceso de las moléculas de una sustancia que van colocándose en los diferentes niveles energéticos buscando un equilibrio y las soluciones visitadas por un procedimiento de búsqueda local. La distribución de las partículas sigue la función de Boltzmann, por lo que cuando una molécula se mueve, ese movimiento será aceptado en la simulación si la energía disminuye, o bien aumenta con una probabilidad proporcional a la constante de Boltzmann en caso contrario.

Finalmente podemos mencionar que, Recocido Simulado es un procedimiento basado en búsqueda local en donde todo movimiento de mejora es aceptado, pero que además permite movimientos de no mejora de acuerdo con unas probabilidades. Dichas probabilidades están basadas en la analogía con el proceso físico de enfriamiento y se obtienen como función de la temperatura del sistema.

Existen varios problemas prácticos en los que se necesita encontrar la solución de más bajo costo total que minimice ciertas funciones de costos dadas.



A continuación describiremos el esquema general de un Algoritmo Recocido Simulado para un problema de minimización.

4.2 Esquema general de un algoritmo Recocido Simulado

Elegir una solución inicial s_0

Elegir una temperatura inicial $t_0 > 0$

Elegir una función de reducción de temperatura $\alpha(t)$

Repetir ***mientras no se verifique la condición de parada***

Repetir ***hasta icount = nrep(t)***

Elegir al azar $s \in N(s_0)$

$$\delta = f(s) - f(s_0)$$

Si $\delta < 0$

entonces $s := s_0$

Sino

Generar x al azar en $(0, 1)$

Si $x < \exp(-\delta/t)$ entonces $s := s_0$

Fin

Poner $t = \alpha(t)$

Fin

El algoritmo de recocido simulado no es más que un procedimiento de generación de soluciones seguido de la aplicación del criterio de Metrópolis de manera repetitiva.

En la práctica para poder utilizar este esquema de algoritmo, nos va a ser necesario cumplir con los requerimientos siguientes:

- Definir el conjunto de soluciones
- Definir la función de costo
- Definir vecinos
- “Elegir” parámetros: temperatura, nrep, α
- “Elegir” criterio de parada

Para poder realizar de la mejor manera el método tendríamos que tomar en cuenta que la temperatura inicial debe ser suficientemente

“alta” como para permitir un intercambio casi libre de las soluciones vecinas, además la forma de reducir la temperatura es vital para el éxito de la meta-heurística.

Hay que determinar:

- Número de repeticiones en cada temperatura.
- Forma de reducción de la temperatura

Muchas iteraciones por temperatura para pocos valores de temperatura o al revés.

Por ejemplo: $\alpha(t) = a^t$; con $a < 1$

Se acostumbra usar $0.8 < a < 0.99$

La variable n_{rep} puede crecer geoméricamente o aritméticamente en cada temperatura, y en algunos casos se usan los datos históricos para determinar n_{rep} .



Las leyes de la termodinámica dicen que a temperatura t la probabilidad de un crecimiento de la energía de magnitud ΔE está dada por:

$$p = e^{\frac{-\Delta E}{kT}}$$

$$\Delta E = E_{\text{nuevo}} - E_{\text{actual}}$$

donde k es la constante de Boltzman, y

T es la Temperatura absoluta

La estrategia de Recocido Simulado es comenzar con una **temperatura inicial** "alta", lo cual proporciona una probabilidad también alta de aceptar un movimiento de no mejora. En cada iteración se va reduciendo la temperatura y por lo tanto las probabilidades son cada vez más pequeñas conforme avanza el procedimiento y nos acercamos a la solución óptima. De este modo, inicialmente se realiza una diversificación de la búsqueda sin controlar demasiado el coste de las soluciones visitadas. En iteraciones posteriores resulta cada vez más difícil el aceptar malos movimientos, y por lo tanto se produce un descenso en el costo.

De esta forma, Recocido Simulado tiene la habilidad de salir de óptimos locales al aceptar movimientos de no mejora en los estados intermedios. Al final del proceso estos son tan poco probables que no se producen, con lo que, sin no hay movimientos de mejora, el algoritmo finaliza.

El criterio de parada es a menudo $T < T_{\min}$, pero pueden ser incluidos aquí factores adicionales. Además, el resultado de la expresión $e^{-\Delta E / (k^*T)}$, no necesariamente se encuentra entre 0 y 1, este valor puede ser evaluado con la comparación del número aleatorio generado.

4.3 Convergencia del Algoritmo

Bajo ciertas condiciones y después de un número elevado de iteraciones el algoritmo encontrará el óptimo, dicho de otra manera, el algoritmo converge asintóticamente al conjunto de soluciones óptimas.

Definición.- Sea (S, f) una instancia de un problema de optimización combinatoria y sean $c_k \hat{A}^+$ y S_i una vecindad de i . Llamamos *probabilidad de generar j desde i* a:

$$G_{ij}(c_k) = \frac{1}{|S_i|} \chi_{S_i}(j)$$

$$\forall i, j \in S$$

Definición.- Sea (S, f) una instancia de un problema de optimización combinatoria y sea $c_k \hat{A}^+$. Llamamos *probabilidad de aceptar j desde i* al valor:

$$A_{ij}(c_k) = \begin{cases} 1 & \text{si } f(j) \leq f(i) \\ \exp\left(\frac{f(i) - f(j)}{c_k}\right) & \text{si } f(j) > f(i) \end{cases}$$

$$\forall i, j \in S$$

Definición.- Sea (S, f) una instancia de un problema de optimización combinatoria y sean $c_k \hat{A}^+$. Llamamos *probabilidad de transición desde i hasta j* al valor:



$$P_{ij}(c_k) = \begin{cases} G_{ij}(c_k)A_{ij}(c_k) & \text{si } i \neq j \\ I - \sum_{l \in S, l \neq i} G_{il}(c_k)A_{il}(c_k) & \text{si } i = j \end{cases}$$

Definición.- Una *cadena de Markov* es una sucesión de experimentos donde la probabilidad de un resultado depende sólo del resultado en el experimento anterior. Si $X(k)$ es una variable aleatoria que denota el resultado en el k -ésimo experimento, entonces se define la *matriz de transición* aquella que tiene por componentes:

$$P_{ij}(k) = P\{X(k) = j | X(k-1) = i\}, \forall i, j \in S$$

Es decir, la matriz de probabilidades de transición es:

$$P = \begin{pmatrix} p_{00} & p_{01} & \cdot & \cdot & \cdot & p_{0m} \\ \cdot & \cdot & & & & \cdot \\ \cdot & & \cdot & & & \cdot \\ \cdot & & & \cdot & & \cdot \\ p_{m0} & p_{m1} & \cdot & \cdot & \cdot & p_{mm} \end{pmatrix}$$

Siendo $i=0, \dots, m$ los estados posibles

Obviamente, las probabilidades de transición del algoritmo de recocido simulado conforman una matriz de transición de una cadena de Markov, por lo que es aplicable toda la teoría desarrollada en este contexto para analizar el algoritmo.

En concreto, estaremos interesados en ver bajo que condiciones el algoritmo converge al conjunto de soluciones óptimas.

Como ejemplos numéricos podemos citar, la minimización de una función de prueba con mínimos desigualmente espaciados, que viene dada por:

$$f(x) = 1 - \exp\left(-2 \log(2) \left(\frac{x - 0.08}{0.854}\right)^2\right) \operatorname{sen}^6(5\pi(x^{3/4} - 0.05))$$

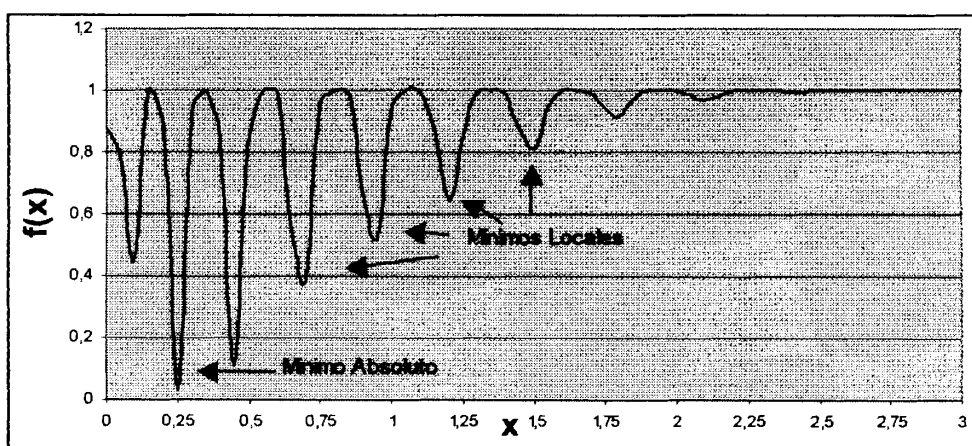


Figura 4.1 Función con mínimos desigualmente espaciados

Si en esta función aplicamos algún método numérico para la búsqueda del mínimo absoluto, nuestro algoritmo puede estancarse en un mínimo local y no hallaría el mínimo absoluto real, más bien nos mostraría como resultado un mínimo local, debido a que los métodos numéricos suelen estancarse en ellos.



Capítulo V

5. DISEÑO, IMPLEMENTACIÓN Y APLICACIÓN DEL MÉTODO RECOCIDO SIMULADO

5.1 Aplicación a la Logística del método Recocido Simulado.-

5.1.1.- El Problema de Ruteo de Vehículos

El problema de ruteo de vehículos V.R.P., por sus siglas en inglés, (Vehicle Routing Problem), también conocido como problema de ruteo de vehículos capacitado (C.V.R.P.) introducido por Dantzig y Ramser en 1959 bajo el nombre original de "The truck dispatching problem", ocupa un lugar central en la logística y es uno de los problemas más ampliamente estudiados en optimización combinatoria. El V.R.P. clásico puede definirse formalmente como lo describimos a continuación.

Sea $G = (V, A)$ un grafo donde $V = \{v_0, v_1, \dots, v_n\}$ es un conjunto de vértices, y $A = \{(v_i, v_j); v_i, v_j \in V, i \neq j\}$ el conjunto de arcos. El vértice v_0 representa un depósito, mientras que los vértices restantes corresponden a clientes. Con A está asociada una matriz de costos (c_{ij}) y una matriz de tiempos de viaje (t_{ij}) . Si estas matrices son simétricas, como lo son generalmente, entonces lo estándar es definir el V.R.P. en un grafo no dirigido $G = (V, E)$, donde $E = \{(v_i, v_j); v_i, v_j \in V, i < j\}$ es un conjunto de aristas.

Cada cliente tiene una demanda no negativa q_i y un tiempo de servicio t_i . En el depósito se tiene una flota de m vehículos idénticos de capacidad Q . El número de vehículos es o bien conocido de antemano o tomado como una variable de decisión. El V.R.P. consiste en designar un conjunto de a lo máximo m entregas o colección de rutas tal que:

1. Cada ruta inicia y finaliza en el depósito .
2. Cada cliente es visitado exactamente una vez por exactamente un vehículo.
3. La demanda total de cada ruta nunca excede a Q .



4. La duración total de cada ruta (incluyendo tiempos de viaje y servicio) no deberá exceder un límite preestablecido D .
5. El costo total del ruteo es minimizado.

El V.R.P. es un problema de optimización combinatoria duro, y únicamente puede resolverse de manera exacta en casos relativamente pequeños. Actualmente, ningún algoritmo exacto es capaz de resolver consistentemente casos con más de 50 clientes, pero en cambio si puede ser tratado eficientemente con meta heurísticas.

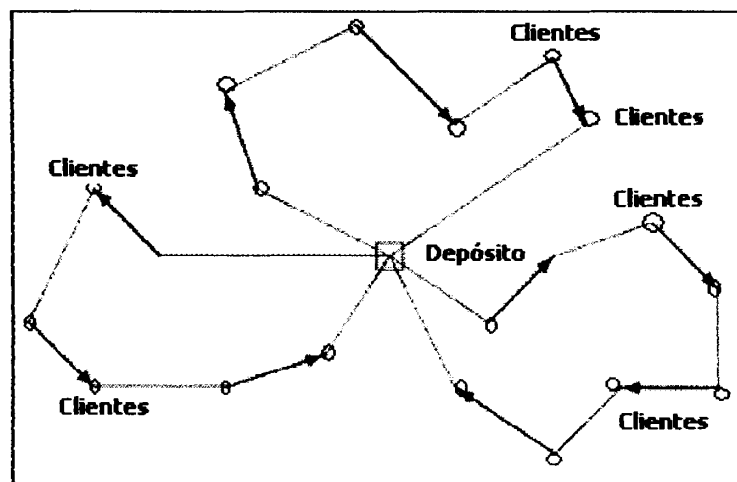


Figura 5.1 Ejemplo de V.R.P. Cronograma rutinario de los vehículos

En la figura 5.1 podemos observar como los vehículos salen del depósito y se dirigen a varios clientes, para finalmente regresar al vértice V_0 o depósito, una secuencia de clientes es asignado a un vehículo en particular.

El tratamiento del V.R.P. por medio de métodos exactos es muy complicado, sino imposible, esto se debe al hecho que es muy difícil hallar buenas cotas inferiores de la función objetivo, lo cual quiere decir que los algoritmos exactos basados en enumeración parcial usando branch and bound o programación dinámica, tendrán una baja tasa de convergencia. Ya que los enfoques exactos son en general inadecuados, para esta aplicación usaremos los métodos meta heurísticos, más específicamente el algoritmo Recocido Simulado.

Todavía la mayor parte del software utilizado en el ámbito comercial usado por las empresas está basado en metodologías no sofisticadas, muchas de ellas provenientes de los años 60's. Hay algunas razones para esta situación. Una de ellas es que el componente de software que hace la optimización del V.R.P. es únicamente una pequeña parte del producto, la mayor parte del esfuerzo ha sido dedicado a la

administración y manejo de datos y a sofisticadas interfaces de usuario.

En lo siguiente describiremos lo que se cree son los cuatro atributos esenciales para la transferibilidad de software y su adopción por parte de los usuarios finales.

Los Cuatro Atributos de las Buenas Heurísticas V.R.P.

Como la mayoría de las heurísticas, las heurísticas para ruteo de vehículos están usualmente medidas bajo dos criterios: precisión y velocidad. Actualmente la opinión también son atributos esenciales de buenas heurística la simplicidad y la flexibilidad. Expliquemos ahora estos cuatro criterios.

5.1.1.1. Precisión

La precisión mide el grado de desviación de una solución heurística con respecto al valor óptimo. Ya que en el caso del V.R.P. usualmente no se dispone del óptimo y de buenas cotas inferiores, la

mayoría de las comparaciones deben hacerse con los mejores valores conocidos. Otra cuestión relacionada a la precisión es la consistencia.

Finalmente, los usuarios preferirán a menudo un algoritmo que produzca una buena solución en un primer instante, y luego exhiba soluciones de calidad creciente todo el tiempo de ejecución, en lugar a un algoritmo que se desarrolla con solamente una respuesta final, posiblemente después de un largo tiempo de cálculo, esto es más que todo, mucho más atractivo para el usuario. Esto da a los usuarios una mejor sensación de cuánto vale el esfuerzo adicional invertido dada la tasa de evolución de la solución.

5.1.1.2. Velocidad

La Velocidad computacional juega un papel muy preponderante a la hora del ruteo de vehículos. Todo depende del nivel de planificación en el cual el problema es resuelto y del grado de precisión requerido. En un extremo, aplicaciones en tiempo real tales como correo expreso

de recolección y entrega o planificación de la ubicación de ambulancias requieren rapidez, y algunas veces acción casi instantánea.

En el otro extremo, en decisiones de planificación a largo plazo hechas cada cierto número de meses, tales como determinar el tamaño de la flota, tiene sentido invertir algunas horas o incluso algunos días de tiempo de computación, particularmente si están en juego grandes sumas de dinero.

Muchas aplicaciones caen entre estos dos extremos. No parece descabellado invertir diez o veinte minutos de tiempo de computación en un problema de ruteo que debe resolverse diariamente. Los sistemas interactivos deben por supuesto reaccionar mucho más rápidamente. El problema de la velocidad no siempre es analizado en esta perspectiva.

Como en el área de la precisión, los investigadores de V.R.P. no respetan consistentemente estándares estrictos cuando se evalúa la velocidad de los algoritmos.



5.1.1.3. Simplicidad

Muchas heurísticas VRP son raramente implementadas ya que son muy complicadas de entender y mucho más difíciles de codificar. Aunque no es realista esperar artículos científicos que proporcionen una descripción minuciosa de cada detalle algorítmico, se debe proporcionar la suficiente información para permitir que un programador razonablemente experto pueda desarrollar un código que trabaje.

Adicionalmente, las heurísticas deberán ser razonablemente robustas para asegurar que trabajan apropiadamente, aunque no todo detalle pequeño sea implementado. Códigos simples, preferiblemente cortos y con soporte autónomo, tienen una mejor oportunidad de ser adoptados, aunque se espera un mínimo de complejidad para lograr buenos resultados.

También los algoritmos que contienen muchos parámetros son difíciles de comprender y difícilmente son usados. Este problema es constante en muchos desarrollos meta heurísticos de los últimos años.

En su búsqueda por lograr mejores soluciones, los investigadores han incrementado el número de parámetros que contienen sus algoritmos mucho más allá de lo que se pudiera considerar razonable, particularmente en vista del hecho de que se usan relativamente pocos casos en las pruebas.

Hay dos maneras fáciles de evitar la proliferación de parámetros. Una es fijar de una vez por todas un valor significativo, especialmente si las pruebas muestran que el algoritmo es bastante insensible a un parámetro particular elegido. Otra posibilidad es hacer uso de parámetros que se auto ajustan durante el desarrollo del algoritmo.

5.1.1.4 Flexibilidad

Una buena heurística deberá ser lo bastante flexible como para acomodar los variados tipos de restricciones adicionales encontradas en la mayoría de la aplicaciones de la vida real(en nuestro caso aplicación a la Ingeniería).



Comparadas con las heurísticas clásicas, los métodos meta heurísticas realizan una búsqueda mucho más completa en el espacio solución, permitiendo movimientos inferiores y algunas veces no factibles, así como recombinaciones de soluciones para crear nuevas.

Esta área de investigación ha experimentado un formidable crecimiento en los últimos diez años y ha producido algunas heurísticas V.R.P. altamente efectivas y flexibles. Es justo decir, sin embargo, que la ganancia en la calidad de la solución obtenida con estas heurísticas modernas ha sido obtenida a expensas de la velocidad y simplicidad, aunque este no es el caso de algunas de las más recientes implementaciones. El Método Meta heurístico Recocido Simulado aparece como una herramienta muy eficiente para el desarrollo del problema de ruteo de vehículos. Es por esta razón que nosotros vamos a realizar nuestra aplicación para el algoritmo Recocido Simulado.

5.1.2. Formulaciones del V.R.P.

En esta parte de este trabajo presentaremos las principales formulaciones de programación matemática que deben ser usadas en el modelo básico del V.R.P.

En general, analizaremos los modelos para el V.R.P. y discutiremos como pueden ser extendidos para incorporar restricciones adicionales o diferentes funciones objetivos.

Existen tres diferentes modelos básicos que han sido propuestos para el V.R.P.

- 1.- Formulaciones del flujo de vehículos.
- 2.- Modelos de flujo de Mercancía
- 3.- Configuración de modelos de partición

5.1.2.1. Modelos de Flujo de Vehículos

Para este modelo comenzaremos describiendo la formulación de la programación lineal entera para V.R.P. El modelo es una formulación del flujo de vehículos de dos índices que utiliza variables binarias para indicar si atraviesa un arco en la solución óptima. En otras palabras, la variable x_{ij} toma valores 1 si el arco $(i,j) \in A$, es decir, pertenece a la solución óptima y toma valor de 0 en caso contrario.

La formulación es la siguiente:

$$(5.1.2.1.1) \quad \min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

sujeto a :

$$(5.1.2.1.2) \quad \sum_{i \in V} x_{ij} = 1; \quad \forall j \in V / \{0\}$$

$$(5.1.2.1.3) \quad \sum_{j \in V} x_{ij} = 1; \quad \forall i \in V / \{0\}$$

$$(5.1.2.1.4) \quad \sum_{i \in V} x_{i0} = K$$

$$(5.1.2.1.5) \quad \sum_{j \in V} x_{0j} = K$$

$$(5.1.2.1.6) \quad \sum_{i \in S} \sum_{j \in S} x_{ij} \geq r(s) \quad \forall S \subseteq V / \{0\}, S \neq \{ \}$$

$$(5.1.2.1.7) \quad x_{ij} \in \{0,1\} \quad \forall i, j \in V$$

Las restricciones de grado interno y externo (5.1.2.1.2) y (5.1.2.1.3) imponen que exactamente un arco ingresa y abandona en cada vértice asociado con un cliente, respectivamente. Análogamente, las restricciones (5.1.2.1.4) y (5.1.2.1.5) imponen el grado requerido para el vértice de depósito.

La denominada restricción de corte de capacidad (C.C.C.) (5.1.2.1.6) impone la conectividad de la solución y los requerimientos de la capacidad del vehículo. De hecho, estipulan que cada corte $(V \setminus S, S)$ definido por un conjunto de clientes es cruzado por un número de arcos no tan pequeños como $r(S)$ (el mínimo número de vehículos que se necesitan para servir al conjunto S). La restricción de corte de capacidad también permanece válida si $r(S)$ es remplazado por el límite trivial más bajo BPP.

Se puede observar que cuando $|S| = 1$ o $S = V \setminus \{0\}$ las CCCs (5.1.2.1.6) están debilitando las formas de las correspondientes restricciones de grado (5.1.2.1.2) hasta (5.1.2.1.5). Note además que debido a las restricciones de grado, se tiene que:

$$(5.1.2.1.8) \quad \sum_{i \notin S} \sum_{j \in S} x_{ij} = \sum_{i \in S} \sum_{j \notin S} x_{ij} \quad \forall S \subseteq V \setminus \{0\}, S \neq \{ \}$$

En otras palabras, cada corte $(V \setminus S, S)$ es cruzado en ambas direcciones el mismo número de veces.

De (5.1.2.1.8) también se puede reiterar (5.1.2.1.6) como

$$(5.1.2.1.9) \quad \sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(V \setminus S) \quad \forall S \subset V, 0 \in S$$

Una formulación alternativa puede ser obtenida transformando las CCCs, bajo el significado de las restricciones de grado (5.1.2.1.2) hasta (5.1.2.1.5), dentro de la bien conocida GSECs (Generalizada eliminación de restricciones):

$$(5.1.2.1.10) \quad \sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - r(S) \quad \forall S \subseteq V \setminus \{0\}, S \neq \{ \}$$

la cual impone que al menos $r(S)$ arcos abandonan cada conjunto de clientes S .

Ambas familias de restricciones (5.1.2.1.6) y (5.1.2.1.10) tienen una cardinalidad de crecimiento exponencial de grado n . Esto significa que es prácticamente imposible resolverlas directamente con la programación lineal. Una posible forma para cubrir parcialmente este inconveniente es considerar solo un limitado subconjunto de estas restricciones y añadir las restantes solo si son necesitadas, usando procedimientos apropiados de separación. Las restricciones consideradas pueden ser suavizadas en una forma lagrangiana, como fue hecho por Fisher y Miller, o pueden ser explícitamente incluidas en la programación lineal de relajación, como fue hecho en los trabajos de B&B. Alternativamente una familia de restricciones equivalentes a (5.1.2.1.6) y (5.1.2.1.10) y teniendo una cardinalidad polinomial puede ser obtenida considerando el GSEC propuesto por Miller, Tucker, y Zemlin y extendiéndolas a ACVRP:

$$(5.1.2.1.11) \quad u_i - u_j + c_j \leq c - d_j \quad \forall i, j \in V \setminus \{0\}, i \neq j \quad \text{tal que } d_i + d_j \leq c$$

$$(5.1.2.1.12) \quad d_i \leq u_i \leq c \quad \forall i \in V \setminus \{0\}$$

Donde u_i , $i \in V \setminus \{0\}$, es una variable continua adicional representando la carga del vehículo después de la visita del cliente i . Es fácil ver que las restricciones (5.1.2.1.11) y (5.1.2.1.12) imponen la capacidad y los requerimientos de conectividad de ACVRP. De hecho, cuando $x_{ij} = 0$, la restricción (5.1.2.1.11) no está ligada a $u_i \leq c$ y $u_j \leq d_j$, considerando que cuando $x_{ij} = 1$, se impone la siguiente restricción: $u_j \geq u_i + d_j$.

El modelo VRP1 puede ser fácilmente adaptado a el problema simétrico. Se debe notar que en SCVRP las rutas no están orientadas (es decir, el cliente a lo largo de una ruta puede ser visitado indiferentemente en el sentido de las agujas del reloj o en contra de las mismas). Además, no es necesario conocer que bordes direccionados están atravesados por los vehículos, y por cada borde no direccionado $(i, j) \in A$, $i, j \neq 0$, solo uno de las dos variables x_{ij} y x_{ji} deben ser usadas, por ejemplo, con $i < j$.

En los siguientes modelos se asume que las rutas para clientes simples son permitidas. La versión simétrica del modelo VRP1 entonces queda de la siguiente forma:

$$(5.1.2.1.13) \quad (\text{VRP2}) \quad \min \sum_{i \in V \setminus \{n\}} \sum_{j > i} c_{ij} x_{ij}$$

sujeto a:

$$(5.1.2.1.14) \quad \sum_{h < i} x_{hj} + \sum_{j > i} x_{ij} = 2; \quad \forall i \in V \setminus \{0\}$$

$$(5.1.2.1.15) \quad \sum_{j \in V \setminus \{0\}} x_{0j} = 2K;$$

$$(5.1.2.1.16) \quad \sum_{i \in S} \sum_{\substack{h < i \\ h \notin S}} x_{hi} + \sum_{i \in S} \sum_{\substack{j > i \\ j \notin S}} x_{ij} \geq 2r(S) \quad \forall S \subseteq V \setminus \{0\}, S \neq \{ \}$$

$$(5.1.2.1.17) \quad x_{ij} \in \{0,1\} \quad \forall i, j \in V \setminus \{0\}, i < j$$

$$(5.1.2.1.18) \quad x_{0j} \in \{0,1,2\} \quad \forall j \in V \setminus \{0\}.$$

Las restricciones de grado (5.1.2.1.14) y (5.1.2.1.15) imponen que exactamente dos bordes están casualmente dentro de cada vértice asociado con un cliente y que 2K bordes están casualmente dentro del vértice del depósito, respectivamente. El CCCs impone tanto la conectividad de la solución así como los requerimientos de la capacidad del vehículo forzando que un número suficiente de aristas

entran en cada subconjunto de vértices. Restricciones como las (5.1.2.1.8) hasta (5.1.2.1.10) pueden ser adaptadas al SCVRP de similar manera.

La versión simétrica de los modelos de dos índices es más frecuentemente definida usando variables con un índice simple e asociado con las aristas no direccionadas $e \in E$. Si las rutas de los clientes simples no son permitidas, todas las variables usadas son binarias; si no, si $e \notin \delta(0)$, entonces $x_e \in \{0, 1\}$, considerando que si $x_e \in \delta(0)$, entonces $x_e \in \{0, 1, 2\}$.

$$(5.1.2.1.19) \quad (\text{VRP3}) \quad \min \sum_{e \in E} C_e x_e$$

sujeto a:

$$(5.1.2.1.20) \quad \sum_{e \in \delta(i)} x_e = 2; \quad \forall i \in V \setminus \{0\}$$

$$(5.1.2.1.21) \quad \sum_{e \in \delta(0)} x_e = 2K;$$

$$(5.1.2.1.22) \quad \sum_{e \in \delta(S)} x_e \geq 2r(S); \quad \forall S \subseteq V \setminus \{0\}, S \neq \{ \}$$

$$(5.1.2.1.23) \quad x_e \in \{0,1\} \quad \forall e \notin \delta(0)$$

$$(5.1.2.1.24) \quad x_e \in \{0,1,2\} \quad \forall e \in \delta(0)$$

Además en este caso, debido a (5.1.2.1.20), la CCCs (5.1.2.1.22) puede ser escrito de la siguiente forma:

$$(5.1.2.1.25) \quad \sum_{e \in E(S)} x_e \leq |S| - r(S) \quad \forall S \subseteq V \setminus \{0\}, S \neq \{ \}$$

donde $r(S)$ puede ser reemplazado por el límite inferior trivial BPP.

Los modelos de flujo de vehículos de dos índices han sido extensamente usados que los modelos de versiones básicas de

SCVRP y ACVRP y algunas otras variantes, como el VRPB, pero esos generalmente son inadecuados para versiones más complejas de VRP. De hecho, como se ha mencionado, pueden ser usados solo cuando el costo de la solución puede ser expresada como la suma de los costos asociados con los arcos atravesados. Además, no es posible conocer directamente que vehículo atraviesa un arco usado en la solución. Por lo tanto, estos modelos no son satisfactorios para los casos donde el costo de un circuito depende en la secuencia del vértice global o en el tipo de vehículo asignado a la ruta.

Una manera de cubrir parcialmente algunos inconvenientes asociados con el modelo de dos índices es indicando el vehículo que atraviesa un arco, entonces las restricciones involucradas pueden ser impuestas en las rutas. De esta forma se obtiene los denominados formulación de flujo de vehículo de tres índices de SCVRP y ACVRP, los cuales usan $O(n^2K)$ variables binarias x : Las variables x_{ijk} cuentan el número de veces que el arco $(i, j) \in A$ es atravesado por un vehículo k ($k = 1, \dots, K$) toma valores de 1 si el cliente i está servido por un vehículo k en la solución óptima y toma valores de 0 , si no. El modelo de tres índices para ACVRP está dado por lo siguiente:

$$(5.1.2.1.26) \quad (\text{VRP4}) \quad \min \sum_{i \in V} \sum_{j \in V} c_{ij} \sum_{k=1}^K x_{ijk}$$

sujeto a :

$$(5.1.2.1.27) \quad \sum_{k=1}^K y_{ik} = 1; \quad \forall i \in V \setminus \{0\}$$

$$(5.1.2.1.28) \quad \sum_{k=1}^K y_{ok} = K;$$

$$(5.1.2.1.29) \quad \sum_{j \in V} x_{ijk} = \sum_{j \in V} x_{jik} = y_{ik} \quad \forall i \in V, k=1, \dots, K$$

$$(5.1.2.1.30) \quad \sum_{i \in V} d_i y_{ik} \leq c \quad \forall k=1, \dots, K$$

$$(5.1.2.1.31) \quad \sum_{i \in S} \sum_{j \notin S} x_{ijk} \geq y_{hk} \quad \forall S \subseteq V \setminus \{0\}, h \in S \\ k=1, \dots, K$$

$$(5.1.2.1.32) \quad y_{ik} \in \{0,1\} \quad \forall i \in V, k=1, \dots, K$$

$$(5.1.2.1.33) \quad x_{ijk} \in \{0,1\} \quad \forall i, j \in V, k=1, \dots, K$$

Restricciones como (5.1.2.1.27) hasta (5.1.2.1.29) imponen que cada cliente es visitado exactamente una vez, que K vehículos dejan el depósito, y que el mismo vehículo entra y abandona un cliente dado, respectivamente. Restricciones como (5.1.2.1.30) son la restricción de capacidad para cada vehículo k, considerando que la restricción (5.1.2.1.31) impone la

conectividad de la ruta realizada por k . Las últimas restricciones pueden ser reemplazadas por la eliminación de restricciones:

$$(5.1.2.1.34) \quad \sum_{i \in S} \sum_{j \in S} x_{ijk} \leq |S| - 1 \quad \forall S \subseteq V \setminus \{0\}, |S| \geq 2, k = 1, \dots, K$$

La que impone que para cada vehículo k al menos un arco abandona cada conjunto de vértices visitados por k y no contenidos en el depósito. Alternativamente, la versión de tres índices de la generalizada eliminación de restricciones de Miller-Tucker-Zemlin (5.1.2.1.11) puede ser utilizada.

$$(5.1.2.1.35) \quad u_{ik} - u_{jk} + cx_{ijk} \leq c - d_j \quad \forall i, j \in V \setminus \{0\}, i \neq j$$

tal que $d_i + d_j \leq c, k = 1, \dots, K$

$$(5.1.2.1.36) \quad d_i \leq u_{ik} \leq c \quad \forall i \in V \setminus \{0\}, k = 1, \dots, K$$

Estas restricciones reemplazan además los requerimientos de capacidad de (5.1.2.1.30).

La versión no direccionada del modelo anterior puede ser obtenido fácilmente usando variables binarias x_{ek} , $e \in E$ y $k = 1, \dots, K$.

$$(5.1.2.1.37) \quad (\text{VRP5}) \quad \min \sum_{e \in E} c_e \sum_{k=1}^K x_{ek}$$

sujeto a :

$$(5.1.2.1.38) \quad \sum_{k=1}^K y_{ik} = 1; \quad \forall i \in V \setminus \{0\}$$

$$(5.1.2.1.39) \quad \sum_{k=1}^K y_{0k} = K;$$

$$(5.1.2.1.40) \quad \sum_{e \in \delta(i)} x_{ek} = 2 y_{ik} \quad \forall i \in V, k=1, \dots, K$$

$$(5.1.2.1.41) \quad \sum_{i \in V} d_i y_{ik} \leq c \quad \forall k=1, \dots, K$$

$$(5.1.2.1.42) \quad \sum_{e \in \delta(S)} x_{ek} \geq 2 y_{hk} \quad \forall S \subseteq V \setminus \{0\}, h \in S \\ k=1, \dots, K$$

$$(5.1.2.1.43) \quad y_{ik} \in \{0,1\} \quad \forall i \in V, k=1, \dots, K$$

$$(5.1.2.1.44) \quad x_{ek} \in \{0,1\} \quad \forall e \notin \delta(0), k=1, \dots, K$$

$$(5.1.2.1.45) \quad x_{ek} \in \{0,1,2\} \quad \forall e \in \delta(0), k=1, \dots, K$$

Los modelos del flujo de vehículo de tres índices han sido extensivamente usados para modelar versiones más restringidas del V.R.P., como el V.R.P.T.W. (Vehicle Routing Problem with time window), debido a sus grados de flexibilidad en incorporar rasgos adicionales. El principal inconveniente de estos modelos está representado por número de variables incrementadas. Por otro lado,

éstos generalizan los modelos de dos índices, los cuales pueden ser obtenidos por simple definición

$$X_{ij} = \sum_{k=1}^K x_{ijk} \quad \text{para todo } (i, j) \in A, \text{ o}$$

$$x_e = \sum_{k=1}^K x_{ek} \quad \text{para todo } e \in E,$$

así permitiendo el uso directo de todas las inecuaciones propuestas por modelos de dos índices y el desenvolvimiento de formulaciones adicionales y fuertes.

5.1.2.2. Problemas de prueba para el Problema de Ruteo de Vehículos Capacitado (CVRP) y otras variantes del V.R.P.

Debido al gran interés en los V.R.P.'s por la comunidad científica y por los profesionales, las pruebas computacionales de los métodos de solución para el V.R.P. generalmente han sido llevadas a cabo solo en un grupo limitado de problemas de prueba Euclidianos, que fueron

propuestos por Christofides y Eilon, y por Christofides, Mingozzi y Toth.

Estos problemas son identificados con una variedad de nombres por los varios autores que los han usado en sus publicaciones y esto puede ser causa de confusión. Es por esta razón, que en este trabajo adoptaremos la unificación de esquemas de nombres descritos por Vigo para identificar los problemas de prueba utilizados para CVRP y DCVRP.

El esquema de nombres a utilizar se basará en los datos del problema, esto nos ayudaría a determinar rápidamente sus características. El primer campo, t , es un carácter alfabético que identifica el tipo de problema y es igual a:

- E para problemas Euclidianos SCVRP
- S para problemas No Euclidianos SCVRP
- A para problemas asimétricos ACVRP, y
- D para problemas simétricos DCVRP

El segundo campo del nombre, nnn, son tres dígitos enteros que denotan el número de vértices del grafo del problema, es decir, incluido el vértice del depósito. El tercer campo, v, es normalmente igual a "-", pero puede ser un carácter alfabético utilizado para distinguir varios problemas que son caracterizados por el mismo número de vértices y vehículos disponibles. Finalmente, el último campo del nombre, p, es un carácter alfabético que identifica la publicación donde el problema fue publicado inicialmente o una alternativa de donde encontrarlo, como sigue a continuación:

- **a** Hays y Eilon, Watson-Gandy, y Christofides
- **c** Christofides, Mingozi, y Toth
- **d** Dantzig y Ramser y Eilon, Watson-Gandy, y Christofides
- **e** Christofides y Eilon
- **f** Fisher
- **g** Gaskell y Eilon, Watson-Gandy, y Christofides
- **h** Hadjiconstantinou, Christofides, y Mingozi
- **m** Christofides, Mingozi, y Toth
- **n** Noon, Mittenthal, y Pillai



- **v** Fischetti, Toth y Vigo, y
- **w** Clarke y Wright y Eilon, Watson-Gandy, y Christofides

Por ejemplo, de acuerdo al esquema de nombres, E051-05e identifica el clásico problema de 50 clientes, Euclidiano con 5 vehículos disponibles propuesto por Christofides y Eilon, y A073-03v identifica un problema ACVRP con 72 clientes, con 3 vehículos y descrito por Fischetti, Toth, y Vigo.

Estos consisten en problemas diseñados intencionalmente con cierta complejidad, para probar la convergencia de los problemas reales y que se pueden encontrar en librerías de internet, si este problema de prueba desarrollado, llega a funcionar adecuadamente esto significa que cualquiera de los demás problemas estarán en capacidad de funcionar bajo este mismo mecanismo de resolución, sin ningún inconveniente mayor.

Estos problemas de prueba están divididos en 3 diferentes categorías, nombrados de la siguiente manera por el significado de sus siglas en inglés, estas son:

- Tipo C, Conglomerado de clientes
- Tipo R, Clientes distribuidos uniformemente
- Tipo RC, Combinación de los Tipos R y de los Tipos C.

Para detallar y tener un enfoque más real de los tipos de problemas de prueba, a continuación graficaremos los 3 diferentes categorías de los mismos.

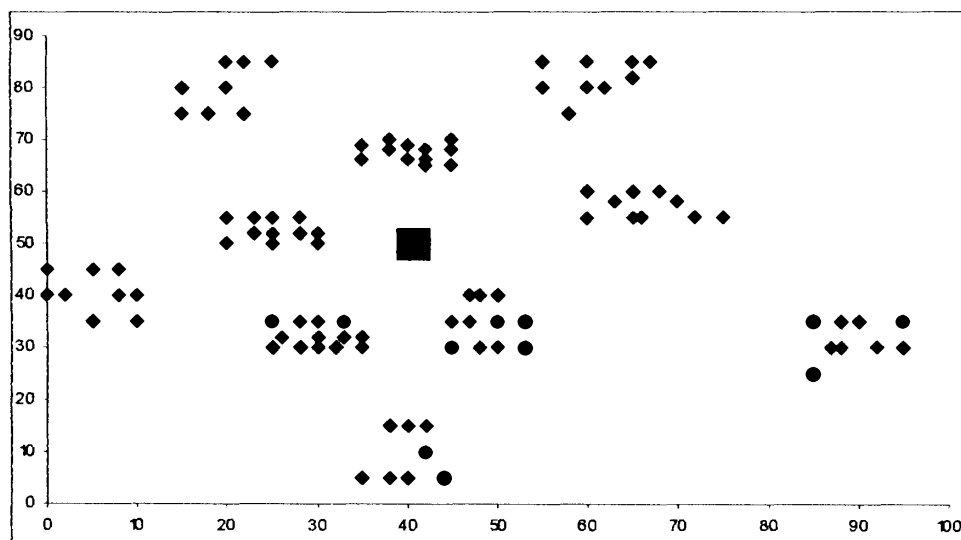


Figura 5.2 Distribución de clientes.- Problema Tipo C

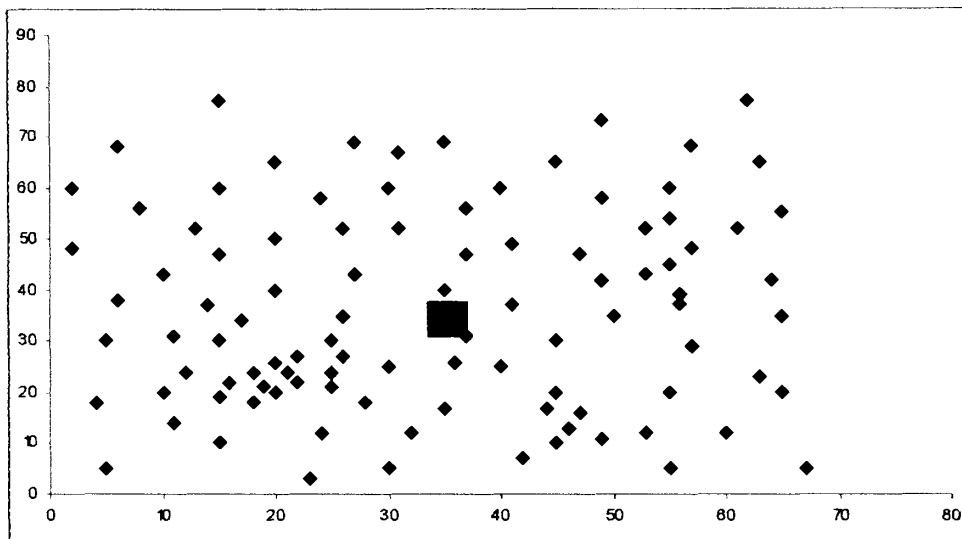


Figura 5.3 Distribución de clientes.- Problema Tipo R

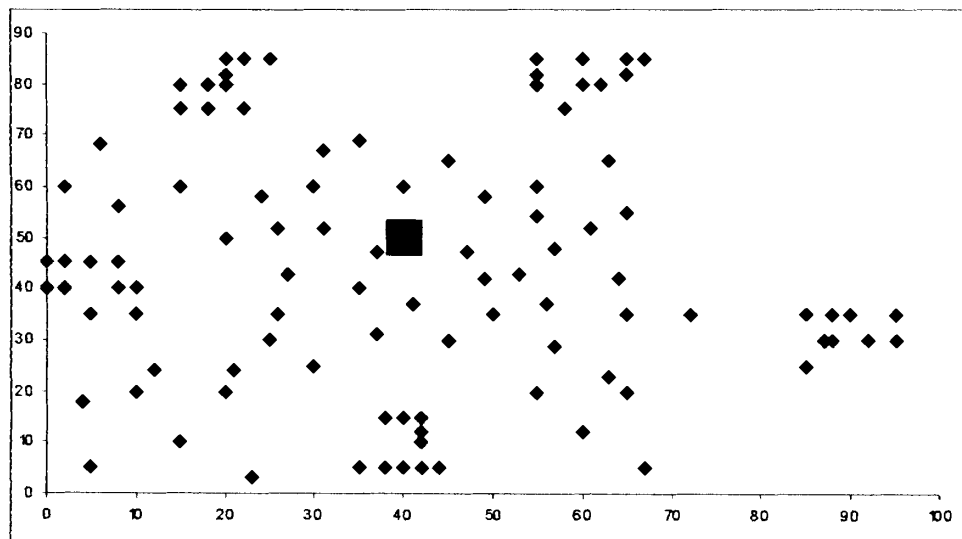


Figura 5.3 Distribución de clientes.- Problema Tipo RC

5.1.3 Parámetros principales del V.R.P.

En la práctica se involucra o se pueden tener muchas restricciones en las rutas de los vehículos, por ejemplo, una de ellas puede ser el límite en el número de horas que un chofer puede trabajar. En lo que sigue de este trabajo vamos a considerar algunas de las restricciones más comunes y las trataremos de clasificar en relación de los vehículos con los clientes, aunque no son consideradas en el presente trabajo.

Nótese que no en todos los casos estas restricciones pueden ser iguales, sin embargo pensando genéricamente sobre este problema, será útil nombrar todas las restricciones que pueden presentarse potencialmente.

5.1.3.1. Vehículos

- ✓ Cada vehículo tiene un límite (capacidad, esto puede estar dado por peso y/o volumen) dependiendo de las unidades de medición de lo que transporta, es decir, un vehículo que transporta gasolina

su capacidad esta dada en volumen, mientras que los autobuses tienen un límite permitido por ley del número de personas a bordo.

- ✓ Cada vehículo tiene un tiempo de funcionamiento desde la salida del depósito hasta su regreso al mismo, típicamente esto obedece a las restricciones legales en horas de trabajo del chofer.
- ✓ Cada vehículo tiene varios periodos de tiempo durante los que no hace nada (periodo de descanso del chofer).
- ✓ Cada vehículo tiene un costo asociado con su uso para las entregas.

5.1.3.2. Clientes

- ✓ Cada cliente tiene una cierta cantidad que debe ser despachada y/o recolectada, típicamente nosotros pensamos en puros funcionamientos de este tipo, pero hay funcionamientos que sólo involucran recolecciones como la recolección de basura doméstica, y funcionamientos que involucran una mezcla de recolecciones y entregas como los funcionamientos de las empresas DHL, Federal Express, UPS (United Parcel Service). A veces esta cantidad es

exactamente conocida (el caso determinístico) y a veces conocida con un grado de incertidumbre (el caso estocástico). En nuestra aplicación vamos a trabajar con el caso determinístico.

- ✓ Cada cliente tiene un grupo de vehículos que no pueden usarse para la entrega (restricciones de acceso).
- ✓ Cada cliente puede tener una prioridad por la entrega, si los vehículos no pueden visitar a todos los clientes). En este trabajo se omite esta restricción.

5.1.3.3. Depósitos

- ✓ Número de depósitos donde los vehículos puedan comenzar, visitar y/o finalizar su recorrido. (En el presente trabajo se realizará la aplicación con 1 solo depósito).
- ✓ Localización de cada uno de los n depósitos existentes.

5.1.3.4. Funciones Objetivos

- ✓ Minimizar la distancia total del viaje.
- ✓ Minimizar el tiempo total del viaje.
- ✓ Minimizar el número de vehículos a utilizar.

5.2 Plataforma y Código de Programación

Para la realización de este trabajo vamos a utilizar como plataforma de programación a Lenguaje C++ el cual constituye una herramienta eficaz para implementar este tipo de aplicación científica.

5.2.1. Diagrama de flujo de datos

Para detallar de una mejor manera nuestro programa, haremos uso de los diagramas de flujos de datos.

Estos diagramas conocidos como D.F.D.'s (por el significado de las siglas) son herramientas que ayudan a el programador mostrarle al usuario como está desarrollado todo el sistema. Hay diferentes tipos de diagramas de flujo de datos, como son: de contexto, de nivel o, de nivel uno.

El diagrama de contexto es el nivel más alto y contiene solamente un proceso que representa todo el sistema completo. En este muestran sólo las entidades externas y los flujos de datos que entran y salen del

sistema. En el diagrama de nivel 0 se muestran procesos pero en una forma más general, también se muestran los almacenamientos de datos.

Finalmente, en el diagrama de nivel 1 es un diagrama hijo para cada uno de los procesos del diagrama de nivel 0. Los símbolos que se utilizan para graficar los D.F.D.'s son los siguientes:

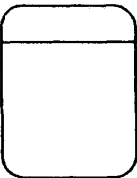
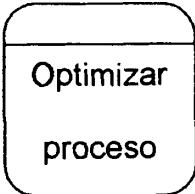
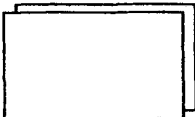
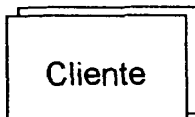




Símbolo	Significado	Ejemplo
	Proceso	
	Entrada	
	Flujo de Datos	
	Almacenamiento de datos	

Tabla 5.1 Símbolos utilizados en los D.F.D.'s

Los D.F.D.'s utilizados en este programa se los puede observar en el anexo 1.

5.3. Resultados Obtenidos

Durante la realización de este trabajo hemos intentado encontrar el mejor ruteo para minimizar los costos para un caso clásico existente (Ver anexo 2).

El recorrido óptimo para los 3 tipos de problema R, C, RC estarían dados como se observa en el anexo 3

El tiempo computacional para la resolución del problema tipo R fue de 14 minutos y 56 segundos, para el tipo C fue de 10 min y 50 segundos, mientras que para el tipo RC fue de 14 min. y 15 segundos. Además todos los intentos fueron realizados en una P.C. Pentium III 1.2 Ghz. con 128 Mb. de R.A.M.

En el anexo 4 podemos observar la comparación de nuestros resultados, con los obtenidos por otros algoritmos en la resolución del V.R.P., los mismos que fueron diseñados por otros autores.



CONCLUSIONES

1. El V.R.P. es un problema difícil de optimización combinatoria basado en aplicaciones reales, y se exigen algoritmos meta heurísticos como el Recocido Simulado, el mismo que fue analizado en este trabajo, para lograr obtener “buenas” soluciones en una cantidad razonable de tiempo para el tamaño real del problema.
2. Como se puede ver el algoritmo Recocido Simulado trabaja de una manera más eficaz en los tipos de problemas C con respecto a la distancia total recorrida, en comparación con los problemas R y RC.
3. El tiempo de ejecución del algoritmo recocido simulado crece exponencialmente y es directamente proporcional de acuerdo al número de clientes que se deseen atender o nodos de recorrido.
4. Debido a que el algoritmo Recocido Simulado es dependiente de un proceso de selección de azar, podemos lograr beneficios con otros tipos de generadores de números aleatorios.

5. El algoritmo recocido simulado nos demuestra que es una buena herramienta para la resolución de este tipo de problemas de optimización duros, ya que nos da una “buena” solución con respecto a los demás algoritmos conocidos, cada uno de ellos con sus ventajas y desventajas.
6. Sin embargo, estas soluciones son consideradas como las más “buenas” pero no existen fundamentos teóricos para demostrar que son las óptimas.
7. No se puede establecer una comparación exacta en los tiempos de ejecución de los diferentes algoritmos presentados, debido a que cada autor ha usado computadoras con diferentes características.
8. Al probar que el algoritmo planteado funciona con los denominados problemas de prueba, se puede concluir que ofrece una solución adecuada en problemas de aplicaciones reales, esto se debe a que los problemas de prueba son diseñados intencionalmente con cierta complejidad, es decir, sirven para probar la convergencia del algoritmo en los problemas de la vida diaria.

- Existirán ocasiones en las que no solo la minimización de la función objetivo será de gran importancia para tomar la decisión final, y es por este motivo, que hay que priorizar otros resultados obtenidos como podría ser el tiempo de ejecución.



RECOMENDACIONES

1. Sería apropiado investigar un poco más acerca del algoritmo recocido simulado para así poderlo aplicar en las diferentes áreas de la estadística, así también aplicarlo a otros problemas de optimización en investigación de operaciones.
2. Incentivar a los estudiantes de pregrado, a investigar más a profundidad acerca de la investigación de operaciones, ya que es una rama que nos muestra una gran cantidad de aplicaciones a la vida diaria.
3. Considero buenas aplicaciones para esta implementación del problema de ruteo de vehículos con recocido simulado en empresas de correo, recolectoras de basura y quizás una manera de darle mayor complejidad al problema sería añadiéndole restricciones como ventanas de tiempo para la hora de servicio.
4. El transporte es uno de los factores más determinantes en el desarrollo económico de un país, por lo que se deben mejorar las técnicas para el mejor ruteo del mismo, y quizás si habláramos del



transporte público, un mejor ruteo se pudiera ver reflejado en el precio final del mismo al consumidor.

5. Creación de un centro de capacitación, en las entidades llamadas a realizar este tipo de trabajos a diario, como las empresas de correo, transportadoras de valores, recolectores de basura, que orienten a los protagonistas que intervienen en este tipo de actividades, indicándoles sus ventajas y desventajas.

6. En problemas de ruteo de vehículos en los cuales el tiempo de respuesta de los vehículos es indispensable, como es el caso del ruteo de las ambulancias de una unidad médica, es recomendable utilizar los algoritmos que muestren resultados en los que si bien la función objetivo puede tomar valores mayores con respecto a las soluciones de otros algoritmos, tengan un tiempo de ejecución menor, es decir, una más rápida obtención de resultados.

7. El algoritmo recocido simulado será de gran utilidad en nuestra comunidad, debido a que en nuestro país y en la mayoría de países de América del Sur, existen empresas que toman decisiones bajo incertidumbre, sin tener en cuenta posibles violaciones de modelación matemática, incurriendo en una mayor cantidad de costos.



8. Implementar un sistema de información geográfica que vinculado con el presente algoritmo permita resolver el problema de ruteo de vehículos en redes físicas reales.

9. Promover a los estudiantes de pregrado a una mayor capacitación en la programación de modelos matemáticos y no concentrarse únicamente en el desarrollo de aplicaciones computacionales, ya que la programación de estos modelos pueden ser utilizados en una gran variedad de aplicaciones a la ingeniería.

10. Como recomendación final, cabe recalcar que en la toma de decisiones resulta difícil prescribir cursos de acción específicos para factores intangibles, el principal de ellos es la presencia del elemento humano, es por este motivo, que solo podemos ofrecer pautas generales para la puesta en práctica de la investigación operacional, y existirán situaciones donde el efecto de la conducta humana influirá en un alto grado para escoger la decisión más acertada.

ANEXOS



ANEXO 1

DIAGRAMA DE FLUJO DE DATOS

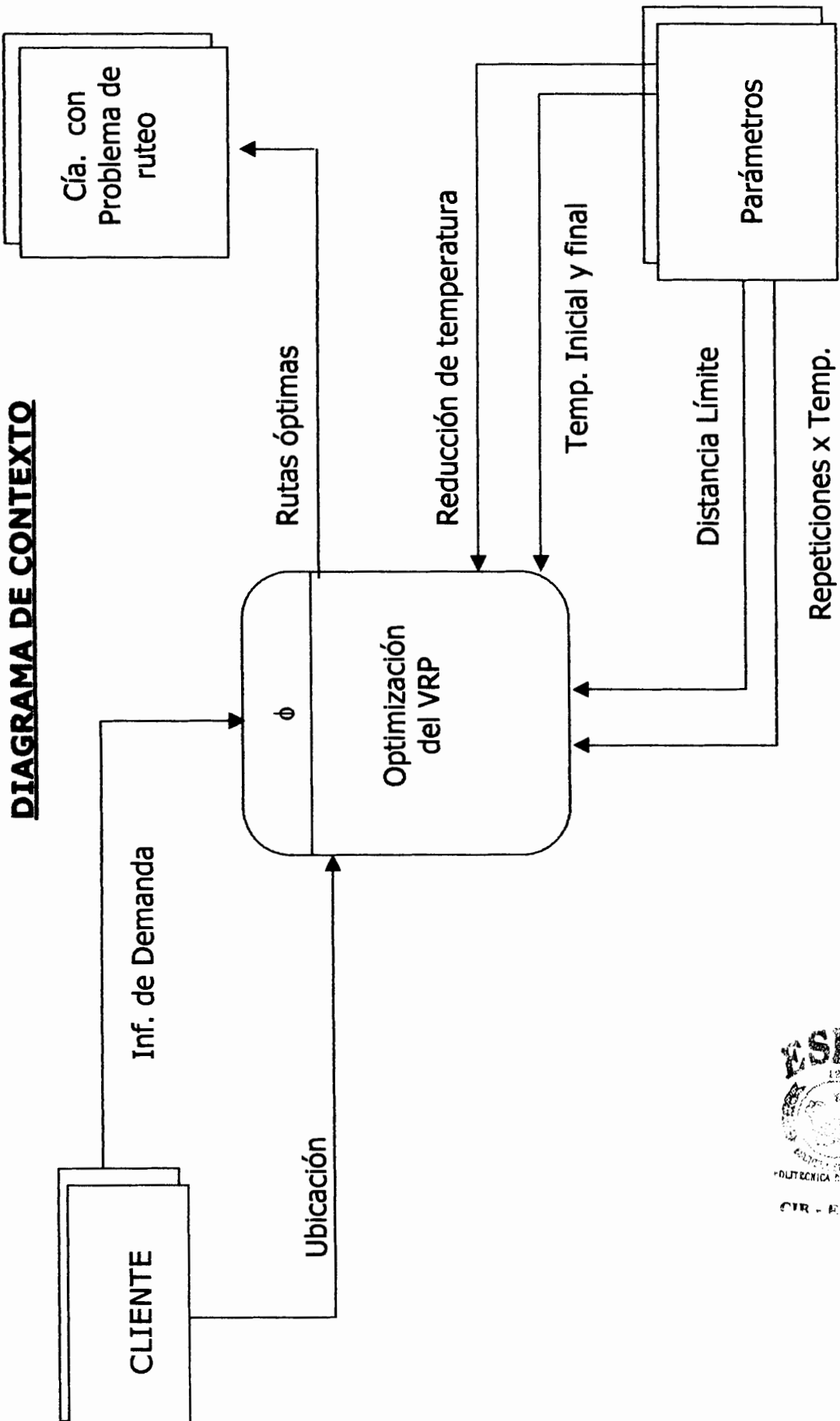
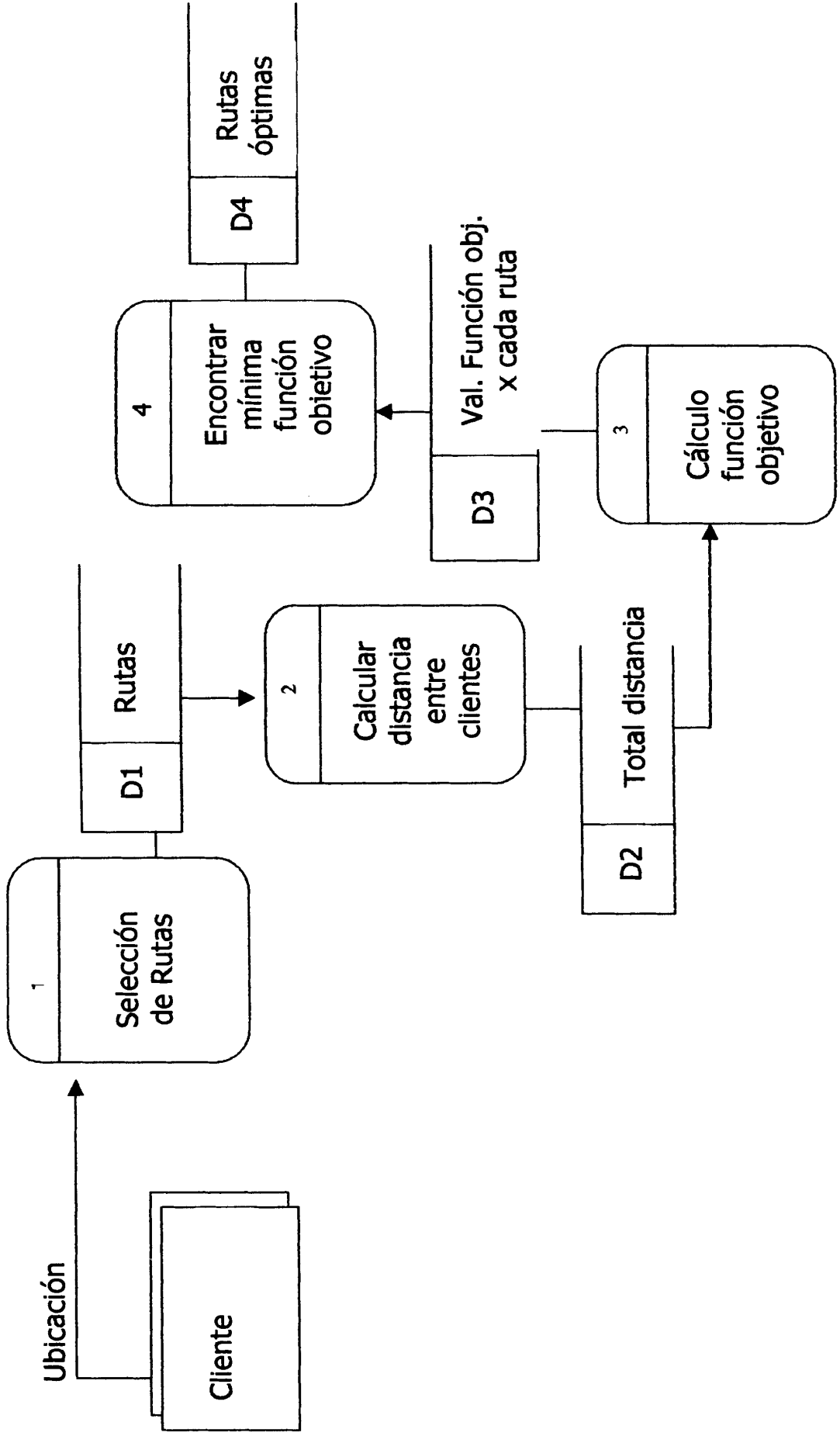


DIAGRAMA DE NIVEL 0



ANEXO 2

DATOS DEL PROBLEMA TIPO R

CLIENTE	XCOORD.	YCOORD.	DEMANDA
0	35	35	0
1	41	49	10
2	35	17	7
3	55	45	13
4	55	20	19
5	15	30	26
6	25	30	3
7	20	50	5
8	10	43	9
9	55	60	16
10	30	60	16
11	20	65	12
12	50	35	19
13	30	25	23
14	15	10	20
15	30	5	8
16	10	20	19
17	5	30	2
18	20	40	12
19	15	60	17
20	45	65	9
21	45	20	11
22	45	10	18
23	55	5	29
24	65	35	3
25	65	20	6
26	45	30	17
27	35	40	16
28	41	37	16
29	64	42	9
30	40	60	21
31	31	52	27
32	35	69	23
33	53	52	11
34	65	55	14
35	63	65	8
36	2	60	5
37	20	20	8
38	5	5	16
39	60	12	31



40	40	25	9
41	42	7	5
42	24	12	5
43	23	3	7
44	11	14	18
45	6	38	16
46	2	48	1
47	8	56	27
48	13	52	36
49	6	68	30
50	47	47	13
51	49	58	10
52	27	43	9
53	37	31	14
54	57	29	18
55	63	23	2
56	53	12	6
57	32	12	7
58	36	26	18
59	21	24	28
60	17	34	3
61	12	24	13
62	24	58	19
63	27	69	10
64	15	77	9
65	62	77	20
66	49	73	25
67	67	5	25
68	56	39	36
69	37	47	6
70	37	56	5
71	57	68	15
72	47	16	25
73	44	17	9
74	46	13	8
75	49	11	18
76	49	42	13
77	53	43	14
78	61	52	3
79	57	48	23
80	56	37	6
81	55	54	26
82	15	47	16
83	14	37	11
84	11	31	7



CIB - ESPOL

85	16	22	41
86	4	18	35
87	28	18	26
88	26	52	9
89	26	35	15
90	31	67	3
91	15	19	1
92	22	22	2
93	18	24	22
94	26	27	27
95	25	24	20
96	22	27	11
97	25	21	12
98	19	21	10
99	20	26	9
100	18	18	17



DATOS DEL PROBLEMA TIPO C

CLIENTE	XCOORD	YCOORD	DEMANDA
0	40	50	0
1	45	68	10
2	45	70	30
3	42	66	10
4	42	68	10
5	42	65	10
6	40	69	20
7	40	66	20
8	38	68	20
9	38	70	10
10	35	66	10
11	35	69	10
12	25	85	20
13	22	75	30
14	22	85	10
15	20	80	40
16	20	85	40
17	18	75	20
18	15	75	20
19	15	80	10
20	30	50	10
21	30	52	20
22	28	52	20
23	28	55	10
24	25	50	10
25	25	52	40
26	25	55	10
27	23	52	10
28	23	55	20
29	20	50	10
30	20	55	10
31	10	35	20
32	10	40	30
33	8	40	40
34	8	45	20
35	5	35	10
36	5	45	10
37	2	40	20
38	0	40	30
39	0	45	20
40	35	30	10
41	35	32	10
42	33	32	20

43	33	35	10
44	32	30	10
45	30	30	10
46	30	32	30
47	30	35	10
48	28	30	10
49	28	35	10
50	26	32	10
51	25	30	10
52	25	35	10
53	44	5	20
54	42	10	40
55	42	15	10
56	40	5	30
57	40	15	40
58	38	5	30
59	38	15	10
60	35	5	20
61	50	30	10
62	50	35	20
63	50	40	50
64	48	30	10
65	48	40	10
66	47	35	10
67	47	40	10
68	45	30	10
69	45	35	10
70	95	30	30
71	95	35	20
72	53	30	10
73	92	30	10
74	53	35	50
75	45	65	20
76	90	35	10
77	88	30	10
78	88	35	20
79	87	30	10
80	85	25	10
81	85	35	30
82	75	55	20
83	72	55	10
84	70	58	20
85	68	60	30
86	66	55	10
87	65	55	20
88	65	60	30



89	63	58	10
90	60	55	10
91	60	60	10
92	67	85	20
93	65	85	40
94	65	82	10
95	62	80	30
96	60	80	10
97	60	85	30
98	58	75	20
99	55	80	10
100	55	85	20

DATOS DEL PROBLEMA TIPO RC

CLIENTE	XCOORD	YCOORD	DEMANDA
0	40	50	0
1	25	85	20
2	22	75	30
3	22	85	10
4	20	80	40
5	20	85	20
6	18	75	20
7	15	75	20
8	15	80	10
9	10	35	20
10	10	40	30
11	8	40	40
12	8	45	20
13	5	35	10
14	5	45	10
15	2	40	20
16	0	40	20
17	0	45	20
18	44	5	20
19	42	10	40
20	42	15	10
21	40	5	10
22	40	15	40
23	38	5	30
24	38	15	10
25	35	5	20
26	95	30	30
27	95	35	20
28	92	30	10
29	90	35	10
30	88	30	10
31	88	35	20
32	87	30	10
33	85	25	10
34	85	35	30
35	67	85	20
36	65	85	40
37	65	82	10
38	62	80	30
39	60	80	10
40	60	85	30
41	58	75	20



42	55	80	10
43	55	85	20
44	55	82	10
45	20	82	10
46	18	80	10
47	2	45	10
48	42	5	10
49	42	12	10
50	72	35	30
51	55	20	19
52	25	30	3
53	20	50	5
54	55	60	16
55	30	60	16
56	50	35	19
57	30	25	23
58	15	10	20
59	10	20	19
60	15	60	17
61	45	65	9
62	65	35	3
63	65	20	6
64	45	30	17
65	35	40	16
66	41	37	16
67	64	42	9
68	40	60	21
69	31	52	27
70	35	69	23
71	65	55	14
72	63	65	8
73	2	60	5
74	20	20	8
75	5	5	16
76	60	12	31
77	23	3	7
78	8	56	27
79	6	68	30
80	47	47	13
81	49	58	10
82	27	43	9
83	37	31	14
84	57	29	18
85	63	23	2
86	21	24	28
87	12	24	13



88	24	58	19
89	67	5	25
90	37	47	6
91	49	42	13
92	53	43	14
93	61	52	3
94	57	48	23
95	56	37	6
96	55	54	26
97	4	18	35
98	26	52	9
99	26	35	15
100	31	67	3

ANEXO 3
TABLAS DE LAS RUTAS ÓPTIMAS PARA DIFERENTES
PROBLEMAS DE PRUEBA

Ruta 1	90	32	20	66	65	35	71	9	33	1				
Ruta 2	63	64	10	69	70	30	51	68	80	54	26	53	27	28
Ruta 3	48	47	36	49	19	11	62	52	31	88				
Ruta 4	85	61	16	86	17	45	46	8	7	82	83	18	89	
Ruta 5	93	99	59	94	6	95	100	44	37	98	92	96		
Ruta 6	57	15	43	42	14	87	2	13	97	58				
Ruta 7	40	72	73	4	56	74	21	75	22	41				
Ruta 8	5	84	60	91	38	23	67	39	25	55	24	29	50	76
Ruta 9	81	34	78	79	3	77	12							

Rutas para recorrido óptimo. Tipo R

Ruta 1	43	5	21	67	47	91	10	32	98	55				
Ruta 2	81	57	100	99	28	26	23	6	25	27	29	20		
Ruta 3	89	88	85	82	83	19	18	17	13	44	40	41	11	
Ruta 4	96	95	94	33	31	35	37	38	39	36	34			
Ruta 5	69	66	68	64	8	9	4	48	62	74	72	97	14	
Ruta 6	51	50	52	49	46	45	3	7	75	42	22	24		
Ruta 7	59	60	58	65	54	53	56	71	76	78				
Ruta 8	70	73	77	79	63	84	80	12	15	2				
Ruta 9	86	87	90	61	16	30	92	93	1					

Rutas para recorrido óptimo. Tipo C

Ruta 1	5	3	7	8	11	9	6	4	2	1	75	34		
Ruta 2	27	25	24	22	20	17	13	18	89					
Ruta 3	81	91	21	23	67	90	14	12	10	65	63			
Ruta 4	57	59	54	53	60	92	94	95	62	66				
Ruta 5	98	99	100	80	84	83	87	33	42	43	86	85		
Ruta 6	47	49	52	19	15	16	41	88	29	35				
Ruta 7	40	50	51	48	45	74	26	28	30	31	32	39		
Ruta 8	78	76	71	46	70	69	68	64	61	72	77	73	82	79
Ruta 9	55	56	58	97	93	96	44	36	37	38				

Rutas para recorrido óptimo. Tipo RC



ANEXO 4

LAS MEJORES SOLUCIONES COMPUTADAS POR DIFERENTES ALGORITMOS VRP

Tipos de Problemas	R			C			RC		
	Vehículo	Distancia	Tiempo	Vehículo	Distancia	Tiempo	Vehículo	Distancia	Tiempo
SA	9	1203,56	896	9	833,5	650	9	1389,25	855
MACS	13	1214,8	100	10	828,4	100	12	1395,47	100
RT	13	1208,43	450	10	832,59	540	13	1381,33	430
KPS	13	1200,33	2900	10	830,75	2900	12	1388,15	2900
CW	12	1241,89	1382	10	834,05	649	12	1408,87	723
TB	13	1233,88	2296	10	830,41	2926	12	1404,59	1877

SA= Simulated Annealing , MACS= Multi Ant Colony Sistem, RT=Rochat and Taillard), KPS= Kilby, Prosser and Shaw , CW = Cordone and Wolfler-Calvo, TB= Taillard et al.

BIBLIOGRAFÍA

1. Zbigniew Michalewicz & David B. Fogel, 2.000 *How to Solve It: Modern Heuristics*, Springer, Berlin.
2. M. Balinski and R. Quandt., 1.964 *On an integer program for a delivery problem. Operations Research*.
3. Paolo Toth, Daniele Vigo, SIAM, 2.002, *The Vehicle Routing Problem*
4. Breedam, Alex Van, 1.995, "Improvement Heuristics for the Vehicle Routing Problem based on Simulated Annealing"
5. 2.003, <http://www.femuni-hagen.de/winf/touren/probleme.html>

