VNIVERSIDAD NACIONAL AVFNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

FACULTAD DE INGENIERÍA

PLANEACIÓN DE INVENTARIOS DE PRODUCTOS MÚLTIPLES CON DEMANDA PROBABILÍSTA EMPLEANDO TÉCNICAS METAHEURÍSTICAS

T E S I S

QUE PARA OBTENER EL GRADO DE:

DOCTOR EN INGENIERÍA

INVESTIGACIÓN DE OPERACIONES

P R E S E N T A

M.I. SALVADOR HERNÁNDEZ GONZÁLEZ

DIRECTOR DE TESIS:

DR. MIGUEL ÁNGEL GUTIÉRREZ ANDRADE

MÉXICO, D.F **SEPTIEMBRE 2010**

JURADO ASIGNADO

Presidente: **Dr. Aceves García Ricardo**

Secretario: **Dra. Flores de la Mota Idalia**

Vocal: **Dr. Gutiérrez Andrade Miguel Ángel**

1er Suplente: **Dr. Ramírez Rodríguez Javier**

2º Suplente: **Dr. De los Cobos Silva Sergio**

Lugar donde se realizó la Tesis:

Universidad Nacional Autónoma de México, Facultad de Ingeniería, División de Estudios de Posgrado.

Tutor de Tesis

Dr. Miguel Ángel Gutiérrez Andrade

Firma

Contenido

Resumen

El problema de reaprovisionamiento de productos múltiples, mejor conocido como problema de reaprovisionamiento conjunto, ha sido estudiado ampliamente en las últimas décadas; en años recientes se ha prestado mucha atención al problema con demanda probabilista o estocástica. Sin embargo, la mayor parte de los métodos propuestos considerando demanda probabilista siguen el enfoque de revisión continua; en comparación, los métodos para el enfoque con demanda periódica (determinar el tiempo entre órdenes consecutivas) son escasos. Suponiendo que el comportamiento de la demanda se ajusta a una función de probabilidad Normal, se obtiene una ecuación de costo aproximada del tipo mixto-entero-no lineal, para el cual sólo se ha reportado un algoritmo heurístico que obtiene buenas soluciones. En este trabajo se propone un algoritmo meta-heurístico acoplado con un algoritmo de búsqueda unidimensional para resolver instancias del problema de reaprovisionamiento conjunto con demanda probabilista. Los resultados obtenidos en una extensa serie de experimentos muestran que el nuevo algoritmo alcanza costos mejores que dicha técnica heurística reportada en la literatura.

Abstract

The problem of replenishment of multiple products, better known as joint replenishment problem has been studied extensively in recent decades, in recent years much attention has been paid to the problem with probabilistic (or stochastic) demand. However, most of the proposed methods considering probabilistic demand are based on the approach of continuous review of inventory.

The methods based on the concept of periodic review of inventory (determining the time between consecutive orders) are scarce. Assuming that the behavior of demand is adjusted by a Normal probability function, one can derive a cost equation for the periodic review approach of the inventory which corresponds to a mixed integer nonlinear function. To solve instances of this variant of the problem has been reported only a heuristic procedure for calculation.

This paper proposes a meta-heuristic algorithm coupled with a one-dimensional search algorithm to solve instances of joint replenishment problem with probabilistic demands. The results obtained in an extensive series of experiments show that the new algorithm achieves better than the heuristic technique reported previously in the literature.

Agradecimientos

Deseo expresar mi agradecimiento al Consejo Nacional de Ciencia y Tecnología (CONACyT) por los recursos proporcionados para el desarrollo de esta investigación.

A mis sinodales por dedicar su tiempo.

Al Dr. Juan Manuel Ricaño, a la Dra.Teresa de la Garza y en especial al M.C. Moisés Tapia por el apoyo, paciencia y confianza depositadas en mí, durante la etapa final.

Dedicatorias

A Quetzalli y Anita, a mis papás, a mi hermano, a mi familia en general, y además:

In the loving memory of Santiago González Hernández

(1950 -2009)

A la UNAM, por sus 100 años.

"Así empiezan las historias, así de fácil. A veces se toma una decisión y, sin reparar mucho en ello, se detona una mina que irá estallando durante varias generaciones…"

Los rojos de ultramar. Jordi Soler.

Notación.

$A_{x,x'}$: Probabilidad de aceptación de la solución *x'* a partir de la solución *x*.

A: Costo de activación mayor.

a_i : Costo de activación del producto *i*.

b: Cantidad de recurso.

C_i: Costo individual.

C_a : Costo total de activación.

C_h : Costo total de acarreo de inventario.

CT: Costo total.

c: Nivel de activación de una orden (sistemas continuos).

c_r: Parámetro de control.

D_i : Demanda del producto *i*.

E', E: Niveles de energía de un sistema físico en dos configuraciones distintas.

f: Función $f: X \to \mathbb{R}^+$.

$f(\chi, \theta)$: Función de densidad de la demanda χ en el tiempo θ.

$G_{x,x'}$: Probabilidad de generación de la solución vecina *x'* a partir de la solución *x*.

g: Función $g: X \to \mathbb{R}^+$.

h_i: Costo individual de acarreo del inventario del producto i.

i: Índice de producto.

K: **N**úmero de factores considerados en el diseño experimental.

k_i : Frecuencia de activación del producto *i*.

$k_{i,\max}$: Cota superior de la frecuencia de activación del producto *i*.

k: Vector de variables$(k_1,...,k_n)$

l: Margen de error.

$N(x)$: Conjunto de soluciones vecinas, estructura de vecindades.

N: Constante.

n: Número de productos.

m: Número de restricciones.

P: Función de probabilidad.

P: Matriz de transición.

Q_i: Tamaño de lote solicitado del producto *i*.

R^2: Coeficiente de correlación.

r: Iteración.

S: Nivel del inventario.

s: Punto de reorden del inventario.

T: Ciclo de tiempo base, tiempo entre pedidos u órdenes consecutivas.

T_0: Ciclo óptimo, caso determinista.

T^*: Ciclo óptimo, caso estocástico.

T_i: Ciclo de tiempo del producto *i*.

t_i: Tiempo de espera del producto *i*.

v_i: Volumen individual.

V: Capacidad de la mochila.

X : Espacio de soluciones.

x_i : Variable de decisión, solución actual.

x' : Solución vecina.

z_β: Multiplicador de la desviación estándar de la demanda σ_χ .

z_i: Multiplicador de la desviación estándar de la demanda del producto i .

α: Rapidez de enfriamiento.

β : Nivel de servicio.

ε: Valor final del parámetro de control.

θ: Instante.

λ : Demanda promedio

μ: Demanda promedio en el instante $t+T$.

σ_χ : Desviación estándar de la demanda.

$\sigma_{\chi,t+T}$: Desviación estándar de la demanda en el instante $t+T$.

σ_i : Desviación estándar de la demanda del producto *i*.

χ: Demanda en el instante $t+T$.

ψ: Costo de órdenes pendientes.

$[\mathbf{a},\mathbf{b}]$: Intervalo de incertidumbre.

Introducción

El inventario se emplea en las empresas de manufactura, servicios, y distribución, debido a que es un factor básico en la medición del desempeño de la rentabilidad de una empresa. Sin embargo, frecuentemente las necesidades de controlar el inventario van más allá de planear un solo producto.

Una tendencia común hoy en día, consiste en reducir el número de entidades que suministran los recursos en una empresa, de tal forma que el distribuidor proporcione la mayor cantidad de suministros. La logística requiere que se realice un solo contrato a largo plazo y que cubra un espectro amplio en la gama de productos que se van a adquirir de dicho distribuidor, lo que implica un costo fijo; posteriormente se realizan las adquisiciones de dichos productos solicitando varios a la vez en una misma orden de compra y cada cierto tiempo; algunos productos se incluirán siempre en la orden de compra, otros lo serán cada dos órdenes, otros cada tres y así sucesivamente.

Otra situación donde intervienen varios productos se presenta cuando es necesario envasar un mismo producto en distintos tipos de contenedores (se pueden tener 5 o más); esto es común en empresas como la de los alimentos y bebidas donde se requiere planear la frecuencia con la que debe mandarse el envasado de cada presentación.

Es posible obtener ahorros en los costos por adquirir o fabricar varios productos a la vez, en lugar de hacerlo tomando cada uno por separado. Determinar el tiempo o la frecuencia de producción-activación de cada producto es un problema de coordinación y se le conoce como problema de reaprovisionamiento conjunto (JRP por sus siglas en inglés: joint replenishment problem).

La demanda es el factor que determina la planeación de los productos y se puede suponer que es determinista, sin embargo en los últimos años se ha observado que es más conveniente considerar que la demanda es probabilista o estocástica y que se ajusta a alguna función de distribución. Las diferencias por considerar o no la demanda probabilista pueden ser considerables en cuanto a las cantidades a adquirir o en los tiempos entre la activación de pedidos a los proveedores; el efecto se verá reflejado al final en los costos debidos al inventario almacenado y los de activación de la orden.

Suponiendo que el comportamiento de la demanda se ajusta a una función de probabilidad Normal, se puede obtener una ecuación de costo aproximada del tipo mixto-entero-no lineal

para el cual sólo se ha reportado un algoritmo heurístico, sin embargo podemos establecer que si bien las soluciones obtenidas son de gran calidad, permanece el problema del tamaño de la instancia, ya que el máximo número de productos empleados para las pruebas es de 10, además las comparaciones realizadas han sido más en el sentido de contrastar los costos de distintas políticas de inventario.

En el caso de las técnicas metaheurísticas, sólo se han reportado implementaciones sobre el problema con demanda determinista, por otro lado no existen implementaciones a la variante que trataremos en este trabajo.

Con base en esto, la relevancia de este trabajo radica principalmente en los siguientes aspectos:

1. Propone aplicar una técnica meta-heurística con un método de búsqueda unidimensional acoplado para obtener muy buenas soluciones del modelo de costo del problema de reaprovisionamiento conjunto,
2. Se realiza una medición del desempeño en cuanto a la calidad de la solución y se compara contra otra técnica heurística reportada en la literatura,
3. Establece criterios para generar las instancias de prueba,
4. Emplea diseños experimentales para realizar el estudio del desempeño, corroborar la influencia de los distintos parámetros del algoritmo y de esta manera tener un mejor panorama sobre el funcionamiento del algoritmo al momento de emplearlo con ciertos parámetros.

El trabajo se divide de la siguiente manera: en el capítulo primero se presenta una revisión del estado del arte sobre los métodos existentes hasta ahora para resolver el modelo con demanda probabilista, en el capítulo segundo se hace una revisión sobre teoría del inventario para modelos de un solo producto y posteriormente se hace la extensión para sistemas de productos múltiples. En el capítulo tercero se revisa lo concerniente a las técnicas metaheurísticas y los métodos de búsqueda unidimensional. En el capítulo cuarto se presenta la implementación del algoritmo propuesto, definición de la estructura de

vecindades, esquema de selección y perturbación, el diseño experimental, las medidas de desempeño y los resultados obtenidos, finalmente se presentan las conclusiones y propuestas de trabajo futuros.

Capítulo Primero: Estado del arte del problema de reaprovisionamiento multiproducto con demanda probabilista.

Introducción

Se han desarrollado gran cantidad de procedimientos heurísticos para determinar el tiempo requerido entre órdenes o pedidos a un costo bajo, tanto para el caso con demanda determinista como el caso con demanda probabilista.

Para este último la mayor parte de los trabajos se han realizado bajo el enfoque de las políticas de revisión continua, comparativamente, los sistemas de revisión periódica (determinar el tiempo de activación entre órdenes consecutivas y qué productos incluir en cada solicitud) han recibido poca atención; además, se usan varias funciones de distribución para ajustar la demanda, siendo las más comunes la función Normal y la función de Poisson.

Cuando se supone que el comportamiento de la demanda sigue una función de distribución Normal, es posible obtener un modelo matemático de costo del tipo mixto-entero-no lineal el cual puede emplearse para determinar el tiempo entre solicitudes o pedidos. Dadas las características del problema, ha sido necesario desarrollar métodos de cálculo que permitan resolver instancias del problema obteniendo la respuesta en un lapso razonable y al mismo tiempo que la calidad de la misma sea adecuada; hasta la fecha se han desarrollado algunas técnicas heurísticas las cuales obtienen soluciones cercanas al óptimo, sin embargo, existe una laguna importante por falta de un buen análisis del desempeño ya que no se han realizado comparaciones entre los distintos procedimientos de solución.

Una de las opciones que se tienen para obtener buenas soluciones, son los procedimientos metaheurísticos los cuales han sido aplicados en una gran cantidad de problemas y al ser comparado su desempeño contra otras técnicas han demostrado ser una opción viable. En el caso del problema de reaprovisionamiento conjunto, las aplicaciones son escasas y se han concentrado únicamente en el problema con demanda determinista siendo el problema con demanda probabilista escasamente estudiado.

A continuación se mencionarán los trabajos relacionados con el problema de reaprovisionamiento conjunto con demanda probabilista.

1.1 Estado del arte

El modelo de reaprovisionamiento conjunto con demanda probabilista, es una variante del modelo de revisión periódica propuesto en Hadley y Within (1963) para un solo producto. La mayor parte de los trabajos para los sistemas multiproducto se han enfocado en las políticas de revisión continua, al contrario de las políticas de revisión periódica que han recibido menos atención; los sistemas continuos se tratan de manera extensa en Balintfy (1964), Ignall (1969), Fredergruen, Groenvelt y Tijms (1984) o Shultz Johansen (1999). En Goyal y Satir (1989) se hace una revisión de los algoritmos para los casos con demanda determinista y aleatoria, sin embargo para esta última únicamente cubre aquellos desarrollados para los sistemas continuos de revisión, dejando una laguna importante por no considerar los sistemas periódicos

Una revisión más reciente del estado del arte para el JRP es la de Khouja y Goyal (2008), donde sí se consideran los sistemas periódicos y los algoritmos desarrollados dentro del período 1989-2005.

Hoy en día, los sistemas periódicos son muy empleados por su fácil aplicación: los compradores prefieren coordinarse con los vendedores y realizar una revisión periódica de sus inventarios, los costos de operación son menores y con frecuencia es más sencillo realizar un monitoreo periódico que uno diario de las existencias (Silver 1985).

En Hadley y Within (1963) se analizan los modelos de revisión periódica y se dan tanto ecuaciones exactas (para el caso de demanda con un comportamiento tipo Normal), como métodos de búsqueda para encontrar los valores de S y T, aunque únicamente lo hacen para el caso de un solo producto. Igualmente en (Silver 1985) se hace una revisión de este sistema de inventarios, se proporcionan también reglas de decisión para obtener políticas de revisión del inventario.

Otros trabajos relativos a los modelos de un producto son el de Donaldson (1984), donde se plantea una ecuación exacta empleando una función de distribución Normal para aproximar el comportamiento de la demanda y obtener T; en Sani y Kingsman (1997) se comparan distintos sistemas de inventario periódico para artículos de lenta rotación; Robb y Silver (1998) proporcionan un método para sistemas que requieren un mínimo en la compra; Tagaras y Vlachos (2001) incorporan costos por pedidos urgentes; y Rao (2003)

proporciona propiedades sobre los sistemas multiproducto así como ejemplos de los sistemas periódicos de inventario en sistemas con demanda probabilista estacionaria.

Para el caso de sistema multiproducto se puede mencionar el trabajo de Atkins e Iyogun (1988) donde se proponen cotas inferiores para el sistema multiproducto con demanda tipo Poisson, órdenes pendientes y tiempo de espera constante. Se debe minimizar el costo esperado a largo plazo; para esto se desarrolla un algoritmo heurístico y además comparan su desempeño contra los sistemas continuos (*S, c, s*). Los resultados demuestran que los sistemas periódicos son mejores para valores de costo de pedido pequeño.

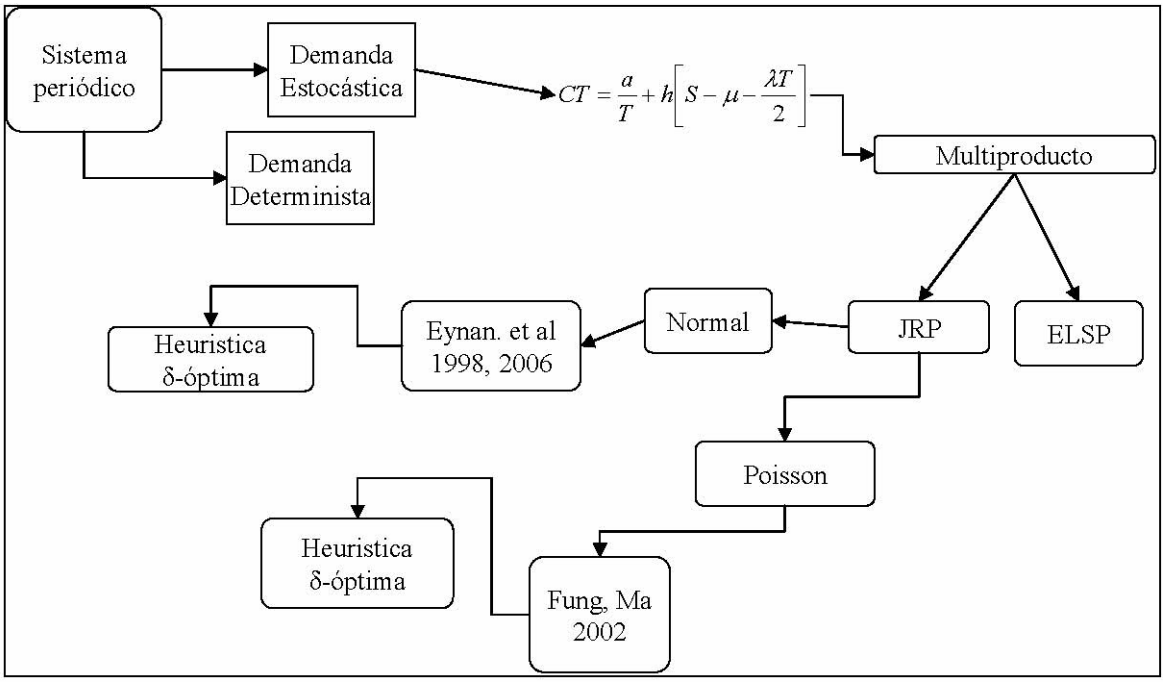


Figura 1.1 Investigación reportada para el problema de reaprovisionamiento conjunto demanda probabilista (Normal y Poisson)

En Pantumsinchai (1992) se comparan tres políticas de control de inventario: (S, c, s) correspondiente al sistema de revisión continua, el sistema propuesto por Atkins (1988) y el modelo propuesto por Renberg y Planche (1967) el cual se conoce como QS. En su investigación encuentran que los sistemas (S, c, s) funcionan mejor cuando los costos de pedido mayor son pequeños, y los sistemas periódicos se desempeñan mejor a medida que este valor se incrementa.

En Viswanathan (1997) se propone un modelo de revisión periódica llamado P(s, S) o sistema punto de re-orden-nivel de inventario periódico, en el cual, el nivel de inventario de

cada producto se monitorea mediante la técnica (s, S); se supone que la demanda sigue un comportamiento tipo Poisson.

En Eynan y Kropp (1998) proponen otra técnica heurística basada en las condiciones de 1er orden y las reglas de Goyal (1974); cabe señalar si bien no se obtiene el óptimo, las soluciones obtenidas son de buena calidad, sin embargo no se realizó un análisis más profundo del mismo y suponen que la diferencia con respecto al óptimo es siempre la misma (Figura 1.1).

En Eynan y Kropp (2007) el mismo algoritmo se aplica con sus debidas modificaciones para resolver el problema considerando órdenes pendientes; en Fung, Ma y Lau (2001) se desarrolla otra técnica heurística para el caso con demanda tipo Poisson, donde los autores comparan el desempeño contra los sistemas continuos, sin embargo el tamaño de la instancia se limitó a 10 productos (Figura 1.1); finalmente en Rao(2003) propone un algoritmo basado en potencias de 2 y realiza un análisis de complejidad, sin embargo no realiza pruebas de la calidad de la solución.

En el caso de los procedimientos meta-heurísticos, no se han reportado, hasta ahora, implementaciones sobre esta variante del problema, por otro lado las comparaciones que se han realizado han sido más en el sentido de contrastar los resultados obtenidos mediante dos políticas de revisión distintas; esto es muy importante ya que siempre que sea posible, hay que realizar una comparación entre los distintos procedimientos existentes para de esta forma poder contar con un elemento de juicio sobre el desempeño de un algoritmo.

Otra laguna detectada, consiste en que el tamaño de la instancia se ha limitado a únicamente 10 productos, es bien conocido que el número promedio de productos con frecuencia sobrepasa los 20 o más elementos; esto es crítico ya que al momento de resolver un problema el tamaño de la instancia es un factor a considerar en cualquier algoritmo. También es importante señalar que en una de las técnicas desarrolladas se encontró que la instancia sobre la cual se realizaron las pruebas consiste en un conjunto de datos que no es representativo de las situaciones que se suelen encontrar en un sistema real, esta instancia, por sí sola, no es en modo alguno suficiente para verificar el desempeño del algoritmo. En este sentido una de las recomendaciones es generar instancias de manera aleatoria,

resolverlas y posteriormente realizar un análisis estadístico para determinar por ejemplo el número de empates.

En el siguiente capítulo se continuará con la descripción de los distintos conceptos de los sistemas de inventario para pasar a describir el problema de reaprovisionamiento conjunto y particularmente el modelo con demanda probabilista.

Capítulo Segundo: Sistemas de inventario

Introducción

En el presente capítulo se revisarán aspectos teóricos de los sistemas de inventario, primeramente se revisará de forma general el caso con un solo producto y posteriormente se tratará con más detalle el caso de productos múltiples dando énfasis al problema con demanda probabilista.

La administración del inventario, es un conjunto de técnicas que se emplean para controlar los niveles o cantidades de productos almacenados, con el objetivo de reducir los costos y al mismo tiempo dar el nivel de servicio que los clientes requieren. El control de inventarios toma como datos los pronósticos de demanda y sus precios; con ambos parámetros se inicia el proceso de balancear y controlar los niveles de existencias de productos almacenados.

El inventario se emplea en la mayoría de las actividades de manufactura, servicios y distribución, y es un factor determinante en la competitividad de las mismas. Se puede pensar que el inventario es una forma de medir el servicio prestado a un cliente; cabe señalar que tener un nivel de servicio no está libre de costo por lo tanto, estudiar los sistemas de inventario consiste en hacer un análisis de los costos y los beneficios de mantener un inventario; el objetivo final será obviamente maximizar los beneficios y al mismo tiempo minimizar los costos; una tarea muy compleja, sobre todo si se trata de varios productos. Uno de los modelos más empleados es el conocido como EOQ (Economic Order Quantity, por sus siglas en inglés), con el cual se puede determinar la cantidad que se debe solicitar al proveedor de un producto. La idea básica de este modelo consiste en equilibrar los costos asociados por mantener inventario en la bodega y los costos por activación de la orden de compra; este equilibrio se alcanza minimizando el costo total. Sin embargo, el enfoque es para un solo producto, por lo que la pregunta es: ¿qué sucede cuando existen varios productos?

2.1 Sistemas de un solo producto

Considerar que la demanda es constante ha sido una práctica común que cualquier empresa aplica en su sistema de control del inventario, sin embargo, este supuesto no siempre es

recomendable emplearlo, ya que en la realidad, la demanda puede presentar variaciones muy grandes y que tienen como consecuencia que los resultados obtenidos al aplicar alguna técnica de planeación sean erróneos, los efectos más importantes de estas variaciones serán 1.el sistema alcanzará el nivel de cero existencias antes de lo previsto causando una situación difícil ante los clientes o bien 2.la capacidad de almacenaje no será suficiente para confinar todo el material que se ha solicitado.

En cambio si se toma en cuenta que la demanda es aleatoria al momento de establecer los debidos controles, es factible prevenir problemas en el sistema para dar el servicio requerido, además de esta manera se pueden evitar riesgos muy grandes como el de perder clientes o evitar una sobreproducción que rebase la capacidad de almacenaje. A continuación, se definirán algunos conceptos básicos de los sistemas de control de inventarios en general y posteriormente se analizarán los sistemas con demanda probabilista.

2.2 Niveles de inventario

Sin importar el tipo de empresa, el sistema de inventario cuenta con los siguientes elementos:

a. Inventario físico: es aquel que como su nombre lo dice, se encuentra presente en el almacén.
b. Inventario neto: se define como la diferencia entre el inventario físico menos los pedidos pendientes.
c. Posición del inventario: en ocasiones también se conoce como nivel de inventario, se define mediante la siguiente ecuación:

$$PosiciónInventario = Físico + Tránsito - Pendientes \qquad (2.1)$$

d. Inventario de seguridad: la cantidad que se mantiene como reserva justo antes de que el siguiente pedido llegue al almacén, generalmente son unidades que se mantiene para protegerse contra variaciones excesivas de la demanda (Silver 1985).

2.3 Órdenes pendientes y ventas perdidas

Estas situaciones se presentan cuando el inventario está en el nivel de cero existencias y entonces el proveedor cae dentro de alguna de las siguientes situaciones:

1. Orden Pendiente: cuando se da entrada al pedido y se mantiene sin atender o surtir ya sea porque el material se encuentra fabricando o en tránsito hacia el almacén; cuando el producto llega al almacén en ese momento se surte o se completa dicha orden.
2. Venta Perdida: en este caso se solicita el producto pero no se surte por estar el inventario en nivel cero por lo que el cliente se retira generando una pérdida.

En la industria se encuentran combinaciones de estas prácticas, es decir, se pueden presentar y tolerar un pequeño número de órdenes pendientes o de clientes perdidos, sin embargo son situaciones muy específicas (Silver 1985).

2.4 Monitoreo de los inventarios con demanda probabilista

La demanda es el elemento primordial de un sistema de inventario y es una variable no controlable, sin embargo en los sistemas de inventario existen factores que sí son controlables y éstos son la cantidad a ordenar, cada cuándo tiempo ordenar (T) y qué ordenar. Esencialmente controlar el inventario significa dar respuesta a las siguientes preguntas:

1. ¿Con que frecuencia se debe revisar el inventario?
2. ¿Cada cuánto tiempo se debe activar una orden de compra?
3. ¿Cuánto se debe solicitar?

Si la demanda es determinista o conocida, se puede recurrir a los modelos basados en el tamaño económico de lote para realizar los cálculos, sin embargo, hoy en día, uno de los principales factores que se toma en cuenta es el hecho de que la demanda sigue un comportamiento estocástico, por lo que la respuesta a las preguntas ya no es tan obvia. Considerando la demanda probabilista, para controlar el inventario se debe escoger entre alguno de los siguientes sistemas para su monitoreo y control: sistemas de revisión continua

o sistemas de revisión periódica (Silver 1985). Como su nombre lo indica, los sistemas periódicos requieren hacer la revisión del inventario por intervalos de igual duración dentro de un horizonte de tiempo y en cada revisión se activa un pedido de producto el cual llega después de transcurrir un lapso conocido como tiempo de espera o arribo (lead time en inglés), en el otro extremo, el nivel de inventario es monitoreado de forma continua (lo cual es posible hoy en día gracias a los avances de los sistemas de cómputo), y es lo que se conoce como sistema de revisión continua.

Cada uno de ellos tiene ventajas y desventajas: los sistemas periódicos requieren un costo operativo menor a los sistemas continuos, así mismo, existe más certidumbre al momento de dar una predicción aproximada del nivel del inventario al momento de activar una nueva orden. La principal desventaja de este sistema es la necesidad de mantener un inventario de seguridad que permita hacer frente a las fluctuaciones de la demanda, generando la necesidad de adquirir espacio adicional para almacenamiento, por lo que el costo de almacenaje se incrementa.

En los sistemas continuos, la revisión es hecha cada instante y la actualización de la información es inmediata permitiendo un nivel de servicio muy uniforme hacia los clientes, además, ya que el inventario es monitoreado continuamente, la renovación se realiza hasta que se activa la orden, lo cual sucede cuando se alcanza un cierto nivel de existencias conocido como punto de reorden. Una consecuencia directa es que se requiere menos espacio de almacenamiento y por lo tanto el costo por acarreo de inventario es menor, sin embargo, los costos fijos por activar las órdenes de compra tienden a tener un mayor efecto en el costo total en comparación con los sistemas periódicos.

2.4.1 Descripción de los sistemas de control periódico

Al momento de implementar la política de control de inventario, la operación se lleva a cabo de alguna de las siguientes formas:

a. Sistema (T, S)

Consiste en una revisión del nivel de inventario cada *T* unidades de tiempo. Al cumplirse el ciclo, la orden se activa y se solicita una cantidad suficiente para reponer el nivel al punto *S* (Figura 2.1), dicha orden tardará *t* unidades de tiempo en llegar al almacén.

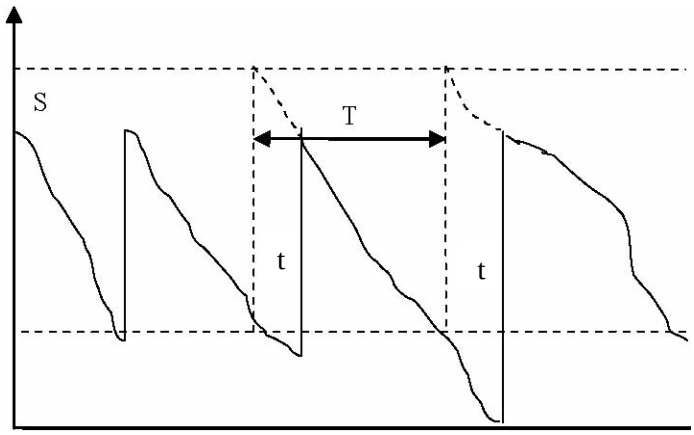


Figura 2.1 Sistema de control (*T*, *S*) con demanda aleatoria

b. Sistema (T, s, S)

Combinación de sistemas punto de reorden y revisión periódica, consiste en revisar el nivel de inventario cada *T* unidades de tiempo, si el nivel de inventario se encuentra por debajo del nivel *s* o punto de reorden se solicita una cantidad suficiente para restablecerlo hasta el punto *S*, por el contrario, si el nivel se encuentra por arriba del punto *s*, no se realiza pedido hasta que han transcurrido nuevamente *T* unidades de tiempo.

2.4.2 Función de costo para sistemas de inventario con demanda probabilista y un solo producto

La medida primordial del desempeño del sistema de control del inventario es el costo total el cual consiste en la mayoría de los casos, en la suma de los costos de activación de las órdenes de compra y los costos de almacenaje (también conocidos como costos de acarreo) y se pueden considerar además, ya sea órdenes pendientes o ventas perdidas. Se analizará únicamente el caso con órdenes pendientes para obtener el modelo definiendo los siguientes parámetros:

a_i: Costo de activación del producto *i*.

D_i : Demanda del producto *i*.

$f(\chi,\theta)$: Función de densidad de la demanda χ en el tiempo θ

h_i: Costo individual de acarreo de inventario del producto *i*.

S: Nivel de inventario.

T: Tiempo entre dos revisiones o dos órdenes consecutivas.

t_i: Tiempo de espera individual del producto *i*.

z_β: Multiplicador de la desviación estándar de la demanda σ_χ (número de desviaciones estándar de la demanda).

μ: Demanda promedio durante el tiempo de espera.

σ_χ : Desviación estándar de la demanda.

$\sigma_{\chi,t+T}$: Desviación estándar de la demanda en el tiempo $t+T$

β: Nivel de servicio (probabilidad de que el inventario de seguridad cubra la demanda).

ψ: Costo por órdenes pendientes.

Los costos a considerar son: costo de activación del pedido, costo de almacenaje, costo de revisión y costo por órdenes pendientes. Es necesario establecer además los siguientes supuestos:

1. El costo por realizar la revisión es independiente de *S* y *T*
2. El costo por unidad es constante
3. Las órdenes pendientes representan cantidades muy pequeñas
4. El costo por cada orden pendiente ψ, es el mismo sin importar la extensión de tiempo en el que existe un orden pendiente

Sea *T* el tiempo que transcurre entre la activación de dos órdenes de compra del inventario, entonces el costo de activación del pedido será $\frac{a}{T}$. El costo del inventario promedio se obtiene determinando el costo por periodo y multiplicarlo por $\frac{1}{T}$. Por conveniencia se emplea el tiempo que transcurre entre dos pedidos. En el momento en el que se activa la orden el inventario alcanza el punto *S*, en este momento, todas las órdenes de compra han llegado y no existe ninguna cantidad en tránsito; el inventario esperado después de que llega la orden compra será la diferencia entre la posición del inventario y la demanda

promedio durante el tiempo de espera:

$$\mathrm{E}[\text{inventario}] = S - \mu \tag{2.2}$$

El promedio de la demanda es constante en todo el período, entonces el nivel de inventario deberá disminuir en forma lineal con respecto al tiempo, por lo que el nivel promedio al momento de llegar el siguiente pedido será:

$$\mathrm{E}[\text{inventario}] = (S - \mu) - DT \tag{2.3}$$

Donde la cantidad solicitada está dada por el producto *DT*. Por el supuesto 3, la integral sobre el tiempo del inventario neto deberá ser aproximada a la integral del inventario físico, por lo que el número de unidades almacenadas en el período será:

$$\mathrm{E}[\text{inventario almacenado}] = \left[(S - \mu) - \frac{DT}{2} \right] \tag{2.4}$$

El costo promedio de acarreo (como también se le conoce al costo de almacenaje) de inventario será:

$$\mathrm{E}[\text{costo acarreo}] = h\left[(S - \mu) - \frac{DT}{2} \right] \tag{2.5}$$

Para obtener el costo anual de las órdenes pendientes, primero debe determinarse el número de órdenes pendientes promedio anuales; las órdenes requieren un tiempo para llegar a su destino que se debe tomar en cuenta al realizar el cálculo del tiempo entre cada activación; para dicho tiempo de espera existen dos situaciones: tiempo de espera constante y tiempo de espera aleatorio.

Se analizará únicamente el primer caso. Cuando se activa un pedido en el instante θ, éste llegará en el tiempo $\theta + t$, igualmente, el siguiente pedido arribará en el momento $\theta + t + T$ además, la posición del inventario se encontrará en el punto *S*. Por lo tanto, se debe obtener el número promedio de órdenes pendientes entre los instantes $\theta + t$ y $\theta + t + T$.

Una orden pendiente se presenta únicamente si se cumple el supuesto (3) y si además la demanda en el período $t+T$ excede la posición *S*. En consecuencia, el número de órdenes pendientes en el período será la integral:

$$\int_{S}^{\infty}(\chi - S)f(\chi;t+T)d\chi \tag{2.6}$$

Hay que señalar que la demanda abarca el instante $t+T$ porque al activarse un pedido en el instante *t*, no puede volver a activarse otro pedido adicional hasta que el anterior ha llegado al sistema en el instante $\theta+t$ (es decir, se tiene físicamente), lo anterior tiene como consecuencia la necesidad de mantener un inventario de seguridad destinado a cubrir la demanda en el instante $t+T$. Se ha señalado que el inventario promedio hasta el instante en que llega un pedido está dado por la cantidad $\mathrm{E}[\text{inventario}]=(S-\mu)-DT$, la cual se conoce como el inventario de seguridad y depende de la demanda en el instante $t+T$. El número promedio de órdenes pendientes anuales será:

$$\mathrm{E}[\text{Pendientes}]=\left(\frac{1}{T}\right)\int_{S}^{\infty}(\chi - S)f(\chi;t+T)d\chi \tag{2.7}$$

Se definió el costo por orden pendiente como ψ, por lo tanto el costo anual será $\mathrm{E}[\text{Costo Pendientes}]=\psi\mathrm{E}[\text{Pendientes}]$. El costo total del sistema está dado por la suma de los costos por activación del pedido, costos de acarreo de inventario y costo por tener órdenes pendientes:

$$CT=\mathrm{E}[\text{Costo Activación}]+\mathrm{E}[\text{Costo Acarreo}]+\mathrm{E}[\text{Costo Pendientes}]$$

$$CT=\frac{a}{T}+h\left[(S-\mu)-\frac{DT}{2}\right]+\frac{\psi}{T}\int_{S}^{\infty}(\chi - S)f(\chi;t+T)d\chi \tag{2.8}$$

La ecuación (2.8) es aplicable para el caso donde se toman en cuenta las órdenes pendientes. Sin embargo, si la cantidad de órdenes pendientes es muy pequeña, su

contribución al costo puede descartarse; además, como ya se mencionó también, no siempre es posible determinar dicho costo, por lo que la ecuación puede simplificarse quedando como:

$$CT = \frac{a}{T} + h\left[(S-\mu) - \frac{DT}{2}\right] \tag{2.9}$$

Son varias las funciones de probabilidad que pueden ajustarse para describir el comportamiento de la demanda *D*, siendo las que se emplean con mayor frecuencia la Normal cuando se tiene una demanda continua y la Poisson para el caso con demanda discreta.

La función Normal es muy empleada por dos razones: con frecuencia es la que mejor ajusta a los datos y permite obtener una ecuación de costo analítica aproximada y sencilla de manejar. En este sentido, el inventario de seguridad es una cantidad que siempre debe estar presente para evitar las órdenes pendientes, y por lo tanto será función de la demanda en el tiempo de espera y de la desviación estándar de la demanda $\sigma_{\chi,t+T}$ durante el instante $T+t$ y que se puede aproximar mediante la siguiente ecuación (Silver 1985):

$$\sigma_{\chi,t+T} = \sigma_{\chi}\sqrt{T+t} \tag{2.10}$$

También es necesario establecer un nivel de servicio β, que debe cumplir el sistema de inventario y se puede obtener a partir de la función normal estándar:

$$z_{\beta}\sigma_{\chi,t+T} = z_{\beta}\sigma_{\chi}\sqrt{T+t} = \chi - \mu \tag{2.11}$$

Como se mencionó anteriormente, z_{β} es la constante que multiplica a la desviación estándar de la demanda σ_{χ} y su valor depende del nivel de servicio deseado β, representa el número de desviaciones estándar de la demanda que el inventario de seguridad será capaz de cubrir con un cierto nivel de probabilidad; el costo total por acarreo del inventario será la suma del costo del inventario promedio y el inventario de seguridad, de tal forma que al sustituir en la ecuación (2.9) se tiene:

$$CT = \frac{a}{T} + h\left[\frac{DT}{2} + z_{\beta}\sigma_{\chi}\sqrt{T+t}\right] \tag{2.12}$$

Esta ecuación corresponde al modelo de costo donde la demanda sigue un comportamiento

Normal y el tiempo de espera para la llegada de una orden es constante. El primer término corresponde al costo de activación del pedido y el segundo término es el costo de acarreo del inventario de seguridad (Silver 1985).

2.4.3 Sistemas multiproducto

Regularmente en la planeación del inventario se emplea el modelo EOQ, éste considera únicamente un solo producto, cosa poco común hoy en día porque en la mayoría de las veces se tienen almacenes donde se confina más de un producto, ya sea por el empleo de distintos tipos de envasado, o simplemente por los distintos productos requeridos por la empresa.

En este caso, se puede aplicar la fórmula de EOQ, lo que equivale a una planeación independiente de cada producto, sin embargo la suma de los costos individuales no necesariamente será la mínima. Es en este momento donde se debe definir la política de planeación a emplear para el control del inventario con varios productos que puede ser, mediante un sistema periódico o mediante un sistema continuo.

Para el caso de revisión periódica, para el control de varios productos, se debe determinar el tiempo requerido para la activación de una orden de compra y además, qué productos incluir en la misma con la finalidad de que el costo sea mínimo.

Una de las políticas se conoce como periodo base de tiempo, la cual, establece que se tiene un periodo de tiempo entre activación de órdenes de compra y el tiempo individual de activación de cada producto será un múltiplo entero de este periodo base de tiempo. Esto es lo que se conoce como reaprovisionamiento conjunto donde la demanda puede considerarse ya sea determinista o estocástica. El hecho de coordinar los productos implica determinar con qué frecuencia se deberán incluir dentro de la orden de compra, lo que tiene las siguientes ventajas (Goyal 1974):

1. Ahorros por compras: cuando se ordenan grupos de productos o se integran varios en una misma orden de compra, generalmente se hacen descuentos en el precio de venta. De lo contrario, si se ordenan por separado, y aún empleando la ecuación de tamaño económico de lote, sumando los costos se verá que por lo general éstos serán mayores que si se solicitan varios al mismo tiempo.

2. Ahorros por traslado: al igual que en el párrafo anterior, es más conveniente realizar los pedidos de varios productos a la vez, de esta forma su traslado es más barato y por cuestiones de logística facilita la operación; de no hacerlo así pueden presentarse problemas: un ejemplo aún muy común en la práctica es la de emplear un transporte con espacio para 30 toneladas para enviar únicamente 10 toneladas de un material, el cual llega a la planta del cliente y tiene que hacer cola detrás de otros tres transportes con capacidad de 30 toneladas que únicamente llevan 10 toneladas cada uno y que en ocasiones inclusive proceden del mismo proveedor.

3. Ahorros por pedido: en ocasiones, realizar un pedido es muy costoso, por lo que es más conveniente agrupar varios materiales para solicitarlos en una sola exhibición y reducir así los costos.

4. Logística: realizar los pedidos de varios productos a la vez, facilita la operación y la planeación del comprador que puede solicitar sus productos de acuerdo a su necesidad con costos más bajos que si los solicita por separado; también el vendedor que puede ahorrarse dinero enviando en un único envío todo el material.

Como ya se mencionó, para este problema se puede considerar la demanda como determinista o estocástica. Ambos se describirán a continuación.

2.4.4 Ecuación de costo con demanda determinista

Para el caso con demanda determinista deben establecerse primero los siguientes supuestos:

- Demanda constante para cada producto
- No se permiten órdenes sin surtir
- El horizonte de tiempo es infinito
- El tiempo de espera para surtir-fabricar es cero
- Se emplea una política de revisión periódica

Los parámetros a emplear son:

CT: Costo total.

A: Costo de activación mayor independiente del número de productos.

a_i: Costo de activación del producto *i*.

D_i: Demanda del producto *i*.

h_i: Costo individual de acarreo de inventario del producto *i*.

n: Número de productos.

T: Ciclo de tiempo base, tiempo entre dos órdenes consecutivas.

T_i: Ciclo de tiempo del producto *i*.

k_i: Frecuencia de activación individual del producto *i*.

Q_i: Tamaño de lote solicitado del producto *i*.

El producto *i* es solicitado cada cierto tiempo T_i, el cual es k_i veces el ciclo base *T*, dado que la demanda es determinista, la cantidad ordenada del producto *i* será:

$$D_i k_i T \tag{2.13}$$

El costo anual individual de acarreo de inventario está dado por la siguiente ecuación:

$$\frac{1}{2} Q_i h_i = \frac{1}{2} D_i k_i h_i T \tag{2.14}$$

El costo total para *n* productos será:

$$C_h = \frac{1}{2} T \sum_{i=1}^{n} D_i k_i h_i \tag{2.15}$$

El costo por solicitar el producto *i* está dado por:

$$C_a = \frac{a_i}{T k_i} \tag{2.16}$$

Y el costo total de activación(o pedido) para *n* productos será la suma de los costos individuales más el costo fijo *A* por activar la solicitud de compra:

$$C_a = \frac{1}{T}\left(A + \sum_{i=1}^{n} \frac{a_i}{k_i} \right) \tag{2.17}$$

El costo total anual se obtiene sumando (2.15) y (2.17):

$$C_a + C_h = \frac{1}{T}\left(A + \sum_{i=1}^{n} \frac{a_i}{k_i} \right) + \frac{1}{2} T \sum_{i=1}^{n} D_i k_i h_i \tag{2.18}$$

Hay que puntualizar que el costo mayor *A* por activación de la orden, es independiente del número de productos incluidos en la misma; el problema de optimización se puede plantear

como sigue:

$$\text{Min}$$

$$CT = \frac{1}{T}\left(A + \sum_{i=1}^{n} \frac{a_i}{k_i} \right) + \frac{1}{2} T \sum_{i=1}^{n} D_i k_i h_i \qquad (2.19)$$

Sujeta a :

$$T > 0$$

$$k_i \geq 1 \text{ y enteros } \forall i = 1,2,...n$$

En el problema de reaprovisionamiento multiproducto se requiere encontrar los valores enteros k_i que corresponden a la frecuencia con la que debe solicitarse cada producto, también se requiere determinar el ciclo base T (Goyal 1974)y que corresponde al tiempo que debe transcurrir entre cada pedido del grupo de productos. A partir de esta ecuación de costo se puede obtener una, en función únicamente de los valores de k_i y a partir de estos valores encontrar el valor óptimo de *T*, sin embargo, resolver una instancia por esta vía tiene un costo computacional considerable ya que el número de combinaciones posibles crece de forma abrupta a medida que el número de productos se incrementa; a este fenómeno se le conoce como explosión combinatoria y es uno de los principales retos a los que se enfrentan los investigadores. Por el contrario, para este caso, es menos costoso resolver el problema encontrando los valores óptimos de k_i dado un valor de *T*.

2.4.5 Ecuación de costo con demanda probabilista

Si se considera que la demanda es aleatoria, el problema de coordinar puede parecer más complejo. Los supuestos que se manejan adicionalmente son (Silver 1985):

- La demanda sigue un comportamiento que se ajusta a una función tipo Normal.
- Existe un tiempo de espera entre órdenes consecutivas.
- Todos los pedidos llegan antes de realizar el siguiente pedido.

Siguiendo la notación anterior, se considerarán los siguientes parámetros del modelo de costo de un solo producto con demanda probabilista: nivel de servicio, desviación estándar de la demanda del producto *i* y el tiempo de espera.

Cada producto tiene un ciclo de pedido T_i; dentro del modelo de reaprovisionamiento

conjunto, éste valor se restringe a ser un múltiplo entero de un ciclo base *T*:

$$T_i = k_i T \tag{2.20}$$

El costo de acarreo de inventario será:

$$C_h = \sum_{i=1}^{n}\left(\frac{TD_i k_i h_i}{2} + h_i z_i \sigma_i \sqrt{k_i T + t_i}\right) \tag{2.21}$$

El costo total anual por activación de la orden que contiene *n* productos es:

$$C_a = \frac{1}{T}\left(A + \sum_{i=1}^{n}\frac{a_i}{k_i}\right) \tag{2.22}$$

Donde *A* es el costo de activación mayor que es independiente del número de productos incluido en el pedido. El costo total está dado por la suma de los costos de activación y acarreo de inventario:

$$CT = C_a + C_h = \frac{1}{T}\left(A + \sum_{i=1}^{n}\frac{a_i}{k_i}\right) + \sum_{i=1}^{n}\left(\frac{TD_i k_i h_i}{2} + h_i z_i \sigma_i \sqrt{Tk_i + t_i}\right) \tag{2.23}$$

Finalmente, el problema de reaprovisionamiento conjunto y demanda con función distribución de probabilidad tipo Normal se puede plantear como un problema de optimización de la siguiente forma:

$$\text{Min} \tag{2.24}$$

$$CT = \frac{1}{T}\left(A + \sum_{i=1}^{n}\frac{a_i}{k_i}\right) + \sum_{i=1}^{n}\left(\frac{TD_i k_i h_i}{2} + h_i z_i \sigma_i \sqrt{Tk_i + t_i}\right)$$

sujeta a :

$$T > 0$$

$$k_i \geq 1 \text{ y enteros } \forall i = 1,2,...n$$

La función objetivo *CT* corresponde a una función mixta-entera no lineal es no-convexa, sin embargo, al fijar los valores de k_i, la función es convexa para la variable continua *T*, y a diferencia el modelo con demanda determinista mostrado en la sección anterior, incorpora el costo por mantener inventario de seguridad.

Para el caso determinista, en el óptimo, los costos de activación dados por el término $\left(A + \sum_{i=1}^{n}\frac{a_i}{k_i}\right)\left(\frac{1}{T}\right)$ igualan a los costos por acarreo del inventario $\sum_{i=1}^{n}\left(\frac{TD_i h_i k_i}{2}\right)$. Sin embargo en el

caso de demanda probabilista, se debe tomar en cuenta el costo del inventario de seguridad necesario para hacer frente a las fluctuaciones de la demanda y que está dado por la expresión $\sum_{i=1}^{n}\left(h_i z_i \sigma_i \sqrt{Tk_i + t_i}\right)$, dicho parámetro no siempre es recomendable descartarlo ya que la diferencia en costo así como del ciclo base de tiempo *T* puede ser importante e inclusive puede acarrear problemas que ya se mencionaron anteriormente (Eynan y Kropp 1998; Silver 1985).

Con respecto a los modelos mixto-enteros no lineales, están clasificados dentro del conjunto NP-duro, es decir, son problemas para los que no se sabe si existirá un algoritmo que permita resolverlos en un tiempo acotado por un polinomio (Floudas 2000), por lo que regularmente es necesario emplear alguna técnica heurística para obtener una buena solución, ya que las técnicas exactas pueden presentar problemas al quedar atrapadas en mínimos locales; de hecho existe un procedimiento heurístico de cálculo reportado para resolver instancias de este problema, el cual se describirá en el capítulo cuarto.

Una estrategia de solución consiste en emplear una técnica metaheurística para la búsqueda sobre las variables discretas y combinarlo con un método que no utilice el cálculo de derivadas (como el de Fibonacci o el de Sección Dorada) para aproximar el valor de la variable continua *T* que minimiza el costo. Las técnicas metaheurísticas han sido implementadas en gran número de problemas y la evidencia experimental ha mostrado que son una opción viable, sin embargo, se requiere un profundo estudio del problema para sacar las mayores ventajas de estos procedimientos de búsqueda y además se deben especificar varios aspectos relativos a su funcionamiento. El capítulo tercero estará dedicado a las técnicas metaheurísticas, las recomendaciones generales para la implementación y la combinación con otros métodos de búsqueda.

Capítulo Tercero: Las técnicas metaheurísticas

Introducción

En el presente capítulo se revisará lo concerniente a generalidades de los problemas de optimización y las técnicas heurísticas a modo de introducción, posteriormente se describirán las técnicas metaheurísticas en general y al recocido simulado en particular, se describirá el algoritmo de sección dorada y al final del capítulo se presentará un procedimiento para realizar pruebas sobre un algoritmo metaheurístico implementado.

Hoy en día, las implementaciones de técnicas metaheurísticas son muy profusas; la razón de este éxito son los resultados obtenidos al resolver problemas que se sabe pertenecen a los problemas NP-duros. Esta clase de problemas, para los que no se sabe si existirá un algoritmo que lo resuelva en tiempo acotado por un polinomio, han sido uno de los principales motores para desarrollar e implementar técnicas y procedimientos nuevos para obtener una solución a instancias de este tipo de problemas.

Se han desarrollado varias técnicas que se catalogan como metaheurísticas: Algoritmos Genéticos, Búsqueda Tabú y Recocido Simulado son las más empleadas; en años recientes se han desarrollado otras técnicas, como ejemplos tenemos las inspiradas en las Colonias de Hormigas o los Enjambres de Partículas; sin embargo, implementarlas requiere primeramente un buen estudio del problema y posteriormente un análisis para hacerlas funcionar de forma eficiente, siendo esto último una etapa a la que en años recientes se le ha dado una creciente importancia.

El análisis de su desempeño ha ido evolucionando con los años, hoy en día se identifican aspectos muy concretos sobre el mismo y que han permitido profundizar en el conocimiento sobre el comportamiento de las técnicas metaheurísticas: tamaño del espacio de búsqueda, valores en los parámetros del algoritmo, tipo de búsqueda (perturbación o construcción).

En fechas recientes se ha hecho recomendable realizar la implementación y análisis de estas técnicas empleando, por ejemplo, diseños experimentales, de esta manera es posible identificar aquellos elementos que más influencia tienen sobre el comportamiento del algoritmo y no se desperdicia tanto tiempo modificando o estudiando una variable a la vez, adicionalmente será posible obtener un algoritmo altamente eficiente en su búsqueda y

capaz de devolver una solución de elevada calidad en un tiempo razonable.

3.1 Problemas de optimización

En los problemas de decisión que es necesario enfrentar en la dinámica empresarial por lo general existen una serie de recursos que son escasos (espacio de almacén, presupuesto, tiempo, etc.) o bien requisitos mínimos que hay que cumplir (satisfacer la demanda, producción), y donde existe un conjunto de personas que deben tomar las decisiones necesarias sobre la forma de usar dichos recursos; por lo general, es conveniente que las decisiones sobre los planes o actividades se tomen de manera óptima. Los problemas de optimización se pueden plantear de forma matemática como sigue:

$$\begin{aligned} &\text{Max} \quad f(x) \\ &\text{sujeta a:} \\ &\qquad g_j \le b_j \quad j = 1,2,\ldots,m \end{aligned} \tag{3.1}$$

Donde $f(x)$ es la función objetivo, $g_j(x)$ representa la restricción j del problema, b_j representa la cantidad de recurso j disponible, las variables de decisión están representadas por el vector x (número de unidades a solicitar de un producto, flujo en una red, etc.) y pueden ser continuas o discretas.

Una instancia de un problema de optimización es una pareja (X, f) donde X es el conjunto de soluciones que cumplen con las restricciones (soluciones factibles) y f es la función de costo:

$$f: X \to \mathbb{R}. \tag{3.2}$$

Encontrar el óptimo consiste en hallar $x \in X$ tal que dé el mínimo (máximo) valor de f satisfaciendo al mismo tiempo las restricciones, es decir, $x \in X$ debe cumplir con:

$$f(x) \le f(x'), \forall x' \in X \tag{3.3}$$

A x se le conoce como óptimo global de la instancia o simplemente solución óptima; hay que aclarar que un problema de optimización consiste en un conjunto de instancias y una instancia del problema de optimización se refiere a un problema específico, es decir, cuando se fijan los parámetros del problema.

Ejemplo 3.1 En el problema de la mochila hay que seleccionar de un conjunto de n elementos con beneficio C_i y volumen v_i, un subconjunto de elementos tal que quepan en una mochila de volumen V y que maximicen el beneficio total. El modelo matemático es:

$$\begin{aligned} &\max \quad CT = \sum_{i=1}^{n} C_i x_i \\ &\text{sujeto a :} \\ &\quad \sum_{i=1}^{n} v_i x_i \le V \\ &\quad x_i = \begin{cases} 1 & \text{si se selecciona el elemento } i \\ 0 & \text{en otro caso} \end{cases} \end{aligned} \tag{3.4}$$

Una instancia del problema consistirá en asignarle valores a n, C_i, v_i, y V.

Cuando las variables son discretas se les conoce como problemas combinatorios y las soluciones consisten en objetos que se seleccionan a partir de un conjunto finito o infinito; los objetos pueden ser ciclos, permutaciones, particiones, etc., y las técnicas empleadas para encontrar el óptimo de los problemas combinatorios (y en general de cualquier problema de optimización), son en su mayor parte iterativos (Papadimitriou y Steiglitz 1998).

3.2 Técnicas heurísticas

Uno de los inconvenientes que se presenta al resolver problemas combinatorios es lo que se conoce como explosión combinatoria; ésta se encuentra en situaciones donde las elecciones

están compuestas secuencialmente, es decir, dado un conjunto de elementos, se pueden obtener diferentes arreglos ordenados o combinaciones de éstos, pero conforme crece el tamaño del problema se incrementa también lo hace y, de forma muy rápida el número de arreglos o combinaciones posibles, y precisamente a este rapidísimo crecimiento se le ha llamado explosión combinatoria.

El efecto de la explosión combinatoria es evidente sobre todo cuando se desea resolver un problema; para uno con pocas variables es factible emplear por ejemplo la enumeración exhaustiva: se enumeran todas las posibles combinaciones, se calcula el costo de cada una de ellas y se escoge la mejor, el tiempo invertido siempre será razonable; sin embargo para problemas de gran tamaño, emplear dicha técnica se vuelve totalmente impráctico, ya que el número de combinaciones posibles puede ser tal que a una persona no le alcanzaría la vida entera para encontrar el óptimo, inclusive aún con computadoras poderosas. La explosión combinatoria la podemos encontrar en todo tipo de problemas: inversión financiera, operación de manufactura, control de inventario, etc.

En este sentido se ha optado por desarrollar y aplicar métodos que empleen estrategias de búsqueda local, es decir, que al momento de resolver un problema combinatorio no generen todas las opciones relevantes y que al mismo tiempo den una solución "satisfactoria" en un tiempo razonable y práctico (Figura 3.1).

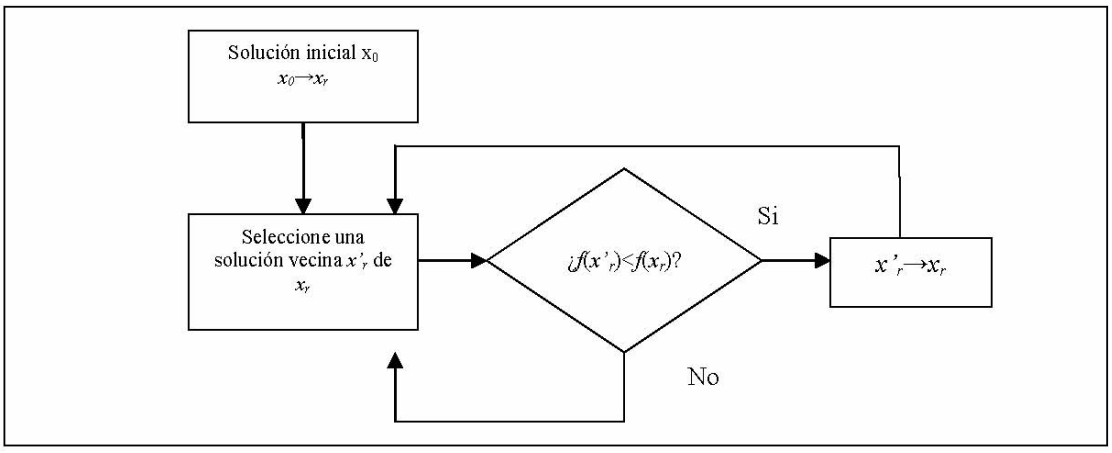


Figura 3.1 La búsqueda local emplea información del entorno.

La forma de realizar la búsqueda local se basa en reglas o métodos empíricos que buscan un buen arreglo y dan una solución en un tiempo razonable. Un procedimiento de búsqueda

que hace uso de estas reglas se le llama búsqueda heurística, palabra de origen griego que significa descubrir; esencialmente consiste en generar a partir de una solución *x* con costo $f(x)$, una solución vecina *x*' con costo $f(x')$ y compararlas. Las técnicas heurísticas enfocan su atención en aquellas estrategias más promisorias e intentan encontrar una solución sin explorar todas posibilidades; hay técnicas heurísticas que se diseñan para un problema concreto y también hay técnicas generales que son conocidas como metaheurísticas, las cuales se describirán más adelante en este capítulo.
Los factores que hacen necesaria la utilización de las técnicas heurísticas son:

1. Cuando no existe un método exacto de solución, o en caso de existir, requiere mucho tiempo de cálculo o memoria.
2. Cuando existen limitaciones de tiempo o espacio de almacenamiento de datos y por lo tanto es necesaria una respuesta rápida a costa de la precisión.
3. Como paso intermedio en la ejecución de otro algoritmo.

Las técnicas heurísticas son muy flexibles para el manejo de problemas ya que se pueden determinar distintas reglas de acuerdo al problema estudiado; sin embargo, el principal inconveniente de estas técnicas es la falta de información sobre la calidad de la solución devuelta, en otras palabras, qué tan lejos se encuentra dicha solución del óptimo; en contrapartida las técnicas heurísticas son capaces de devolver la mejor solución dentro de cierto entorno, a esta solución se le conoce como mínimo local (Díaz. Et al 1996).

3.2.1 Función de costo

Para dirigir al algoritmo hacia la mejor solución, debe existir una medida de desempeño dada por la función de costo *f* y es indicativa de la calidad de una solución. Las unidades monetarias de costo son las más frecuentes en el ámbito de aplicaciones prácticas aunque se puede pensar en otras funciones como el número de elementos, desviaciones, etc. Las funciones pueden ser lineales o no lineales.

3.2.2 Enumeración exhaustiva y búsqueda local

Existen dos conceptos importantes dentro del ámbito de los algoritmos y son la enumeración exhaustiva y la búsqueda local. En el primer caso, se considera que el algoritmo realiza la exploración sobre todo el espacio de soluciones y proporciona un resultado óptimo pero en el caso de los problemas de gran tamaño, como ya vimos, el número de combinaciones posibles puede ser tan grande que el algoritmo tardaría mucho tiempo en devolver el resultado (en este caso el óptimo) aún empleando una computadora poderosa. Por el contrario, en el caso de la búsqueda local, a partir de una solución, el algoritmo realiza la exploración de las soluciones vecinas, es decir, las que caen o se encuentran dentro de un espacio de búsqueda previamente definido, pero lo más importante es que sólo se explora un número relativamente pequeño de vecinos y se emplea únicamente información local de la región o entorno, dicha exploración se lleva a cabo hasta que se cumplen los criterios de paro del algoritmo (Figura 3.2).

Figura 3.2 La búsqueda local emplea información del entorno.

Ya que no abarca la totalidad del espacio de soluciones y sólo se limita a un entorno (de ahí su nombre de búsqueda local), es una búsqueda incompleta y por lo tanto no hay la seguridad de alcanzar una solución óptima; sin embargo y precisamente al no tener que explorar todo el espacio su tiempo de ejecución tiende a ser menor que una enumeración exhaustiva y además, si dicho algoritmo es eficiente en su búsqueda, con regularidad la solución encontrada será indudablemente la mejor posible pero únicamente dentro del entorno, es decir, será un mínimo local.

La forma de realizar esta búsqueda se hace siguiendo alguno de los siguientes esquemas o paradigmas: por construcción o por perturbación los cuales se describirán a continuación.

3.2.3 Paradigmas de búsqueda

La idea fundamental de la búsqueda de una solución radica en generar y evaluar soluciones de forma iterativa. La evaluación de una solución de un problema combinatorio implica verificar si la solución candidata x' con costo $f(x')$ es mejor que la solución x y costo $f(x)$, en caso afirmativo se actualiza convirtiendo la solución candidata en la mejor solución obtenida, o simplemente en la solución actual, en caso contrario la solución candidata se descarta y se evalúa una nueva solución la cual se escoge nuevamente del entorno.

Las soluciones candidatas deben obtenerse de alguna manera, en este caso existen dos esquemas: búsqueda por construcción y búsqueda por perturbación; en el primer caso a partir de una solución parcial, se obtiene una solución candidata completa agregando los elementos faltantes hasta que se obtiene una solución factible, un ejemplo es el algoritmo tipo "glotón" el cual va agregando aquellos elementos que presentan las mejores características; en el segundo caso a partir de una solución completa se selecciona un elemento y se modifica para obtener una nueva solución, dicha modificación o perturbación puede ser por ejemplo efectuando una operación de suma o resta; en ambos casos existen criterios aplicables que determinan la forma en que se realiza la selección de la solución candidata.

3.2.4 Vecindades, movimientos de búsqueda y estrategias de búsqueda

La estructura de vecindades $N(x)$ se refiere al conjunto de soluciones a las que se puede acceder desde la solución actual x; la estructura o entorno de vecindades que se defina es crucial para el desempeño del algoritmo ya que: 1. es específica del problema y 2. debe aprovechar de forma eficiente la información del entorno.

A partir de la solución actual x, es posible desplazarse o "moverse" a una solución candidata $x' \in N(x)$ efectuando una operación elemental (añadir o quitar elementos, intercambiar elementos, realizar una operación aritmética, etc.); conforme transcurren las iteraciones el algoritmo va recorriendo un "camino" $x_0, x_1, ..., x_r$ donde en la r-ésima iteración la solución será la mejor obtenida para cierta vecindad; el algoritmo se detendrá cuando se cumpla el criterio de paro establecido y la solución obtenida será de buena

calidad.

Si la estructura de vecindades es adecuada, el algoritmo aprovechará de forma eficiente la información local y la calidad de la solución obtenida será alta. Ahora bien, es necesario resaltar que en la búsqueda dentro de cierta vecindad $N(x)$se obtiene al final un mínimo local; existe desafortunadamente la posibilidad de que los algoritmos de búsqueda local queden atrapados en alguno de estos mínimos locales y que, dadas las características de la función, el algoritmo sea incapaz de salir o escapar, trasladarse a otra y continuar la búsqueda e inclusive encontrar una solución mejor, por lo que también se presta cierta atención a incorporar alguna técnica o un medio para hacer que el algoritmo salga de estas vecindades (u "hoyos") y se traslade a otra donde continuar la exploración.

Las estrategias varían de acuerdo al problema y el algoritmo; en Recocido Simulado se puede por ejemplo reiniciar el parámetro de control si hay un cambio a una solución de peor calidad de tal forma que aumente la probabilidad de aceptar una solución aún peor.

La búsqueda local no realiza una exploración completa por lo que parecería que ser muy inferior a un algoritmo de enumeración exhaustiva, sin embargo, no es así. El tiempo es un factor determinante en el mundo real, y en este sentido con frecuencia existe una limitación de este recurso; los algoritmos exactos corren el riesgo de tener que ser detenidos sin haber terminado el proceso de búsqueda debido a que han consumido demasiado tiempo y cabe la posibilidad de no obtener ninguna solución, en cambio con los algoritmos de búsqueda local una vez que se ha consumido el tiempo disponible devuelven la mejor solución encontrada hasta entonces.

Toda búsqueda local debe ser realizada de forma inteligente, es decir, el algoritmo debe tomar la información del entorno de tal manera que le permita arribar una nueva solución, mejor que la anterior y, de esta forma, tendrá el tiempo suficiente para explorar el nuevo entorno antes de que se satisfaga el criterio de paro; sin embargo es muy importante lograr un equilibrio entre la profundidad de la exploración y el tiempo consumido, dicho equilibrio está determinado por los parámetros con los que funciona el algoritmo los cuales se determinan vía experimentación.

3.2.5 Búsqueda local aleatoria

Algunos de los algoritmos de búsqueda local que con frecuencia se emplean tienen como

característica distintiva el seleccionar la soluciones candidatas en función de una probabilidad; estos algoritmos se conocen como algoritmos de búsqueda local aleatoria. Las componentes de un algoritmo de esta clase son (Hoos y Stützle 2005, Figura 3.3):

1. Espacio de soluciones
2. Conjunto de soluciones factibles
3. Definir la estructura de vecindades
4. Una función de inicialización que especifica la distribución de probabilidad a lo largo de las distintos estados
5. Una función de probabilidad para la estructura de vecindades
6. Una función de paro para establecer el momento en que el algoritmo debe detenerse

Acoplar estos elementos y hacer que funcionen eficientemente es un trabajo donde el estudio del problema absorbe la mayor parte, además es muy probable tener que evaluar distintos aspectos del algoritmo, por ejemplo para arribar a una solución vecina en el caso de la estructura de vecindades, o el tamaño o extensión del espacio de soluciones o bien, los valores de los parámetros necesarios para su funcionamiento.

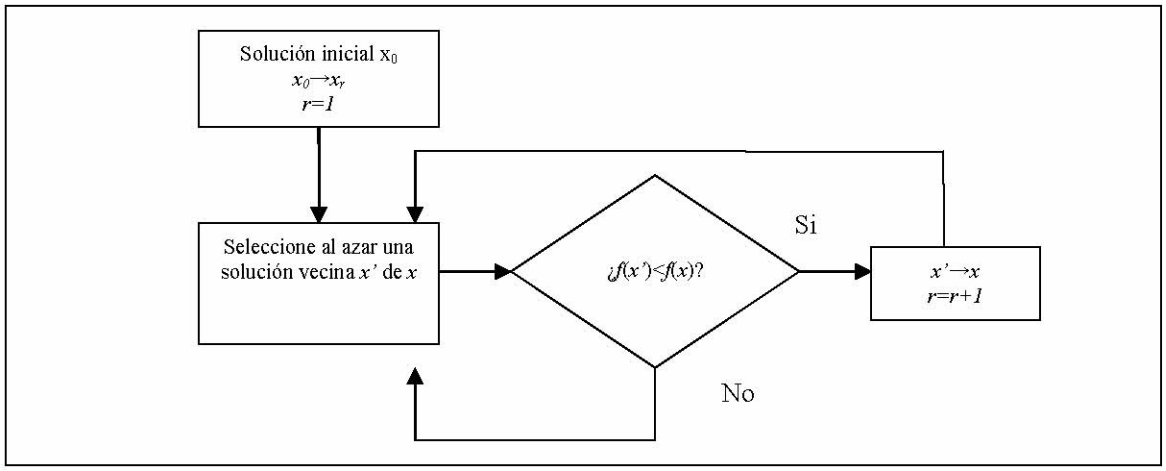


Figura 3.3 Búsqueda local aleatoria

3.3 Técnicas metaheurísticas

Las técnicas metaheurísticas son un conjunto de algoritmos generales para optimización

combinatoria cuya principal característica es que no están diseñados para un problema concreto, son procedimientos generales y muy flexibles lo que permite aplicarlos a la gran mayoría de los problemas combinatorios, por otro lado, estos procedimientos hacen más eficiente la búsqueda que realiza un método heurístico, permitiéndole explorar otras regiones del espacio de búsqueda.

Algunas tienen como característica el emular algún proceso (el recocido) o fenómeno de la naturaleza(los algoritmos genéticos, la búsqueda de alimento por parte de las colonias hormigas), sin embargo, y a diferencia de una técnica heurística, consisten en un procedimiento general y sólo la forma de seleccionar las soluciones o arreglos se basan en criterios empíricos acordes a cada tipo de problema, de ahí la razón por la que se catalogan como metaheurísticas, es decir, conceptualmente se encuentran por encima de una técnica heurística.

A través de estas reglas de selección de soluciones y/o arreglos, dichas técnicas dirigen la búsqueda heurística realizando movimientos hacia regiones donde la calidad de la solución sea más prometedora escapando de óptimos locales, algo que no siempre es posible únicamente con un procedimiento heurístico.

Cabe señalar sin embargo que en la implementación de dichos procedimientos es necesario tomar en cuenta varios aspectos que se deben definir: la estructura de vecindades, el número de iteraciones o el criterio de paro por mencionar algunos. Como ya mencionó al inicio del capítulo, Algoritmos Genéticos, Búsqueda Tabú, Recocido Simulado, Colonia de Hormigas y Enjambre de Partículas son ejemplos de técnicas metaheurísticas. A continuación se presentará con más detalle la técnica de Recocido Simulado.

3.3.1 Recocido Simulado

El Recocido Simulado (RS) es un procedimiento metaheurístico introducido por Kirkpatrick, Gelatt y Vecci (1982) y que ha sido muy empleado para resolver problemas de optimización. La idea surge del proceso físico conocido como recocido en el cual, se eleva la temperatura de un sólido hasta el punto que se vuelve líquido, a continuación la temperatura se disminuye de forma paulatina para obtener una estructura cristalina sin defectos y que puede considerarse como un estado de mínima energía o estado basal.

Cada descenso de temperatura debe ser lo suficientemente pequeño para que el sistema no adquiera una estructura cristalina con defectos, además el sistema debe permanecer un tiempo suficiente a una misma temperatura para permitirle alcanzar un estado estacionario, dicho en otras palabras, que las partículas vuelvan a reacomodarse y adquieran una configuración estable a esa temperatura. Existe un algoritmo para simular dicho proceso el cual, es adaptado por Kirkpatrick, Gelatt y Vecci (1982) para resolver problemas combinatorios.

Para adaptar el algoritmo de RS a problemas de optimización, primero se necesita realizar la analogía con el sistema físico y que se presenta en la Tabla 3.1. Los estados del sistema son los posibles acomodos que pueden adquirir las partículas, cada arreglo tendrá un nivel de energía y es posible cambiar de un estado a otro; x y x' son dos configuraciones, f es la función de costo y equivale a la energía E del sistema, el parámetro de control c equivale a la temperatura T a la que se encuentra el sistema físico.

La implementación del algoritmo RS (y en general de cualquier metaheurística) requiere primeramente la definición de tres aspectos fundamentales y que son propios del problema:

1. Representación del problema
2. Mecanismo de transiciones
3. Estructura de vecindades

Tabla 3.1

Analogías entre los estados de un sistema físico y un problema combinatorio

Problema de optimización	Sistema físico
Soluciones del modelo	Estados del sistema
Función de costo f	Energía en un estado del sistema
Parámetro de control, c	Temperatura del sistema
Dos soluciones x, x' con costo $f(x)$y$f(x')$	Configuraciones con estados de energía E y E'

Hay que recordar que un problema combinatorio se puede representar como una pareja

(X, f), donde X es el conjunto de todas las soluciones (o configuraciones) y f es la función de costo o simplemente el costo; también ya se vio anteriormente que en vez de explorar todas las configuraciones posibles sólo hay que realizar la búsqueda limitándose a una vecindad o entorno, entonces a partir de una solución x y costo $f(x)$ se selecciona a continuación un vecino dentro de un intervalo o vecindad $N(x)$ previamente definido y empleando únicamente información local, a continuación se evalúa la calidad de la solución. Si el valor de la función $f(x')$ es mejor se acepta, en otro caso y concretamente con el algoritmo RS dicha solución no se descartará a la primera, sino que se evalúa la probabilidad de aceptación dada por (3.5) la cual es conocida como el criterio de Metrópolis.

$$P[\text{aceptar } x'] = \begin{cases} 1 & f(x') < f(x) \\ \exp\left(-\dfrac{f(x') - f(x)}{c}\right) & f(x') \geq f(x) \end{cases} \tag{3.5}$$

Solución inicial: $x_0 \rightarrow x$
Fijar parámetro de control c_0, c_f, *iteraciones*
establecer esquema de enfriamiento
Hacer mientras $c > c_f$
    Hacer mientras *r<iteraciones*
        Seleccione solución vecina $x' \in N(x)$
        Calcule $f(x')$
        Si $f(x') < f(x)$, entonces $x' \rightarrow x$
        En otro caso
            Si $\exp\left(-\dfrac{f(x') - f(x)}{c}\right) \geq$ Aleatorio (0, 1)
                Entonces $x' \rightarrow x$
    $r = r + 1$
    Regresa
actualizar parámetro de control c
Regresa

Figura 3.4 Pseudocódigo del algoritmo de de Recocido Simulado

También se debe generar un número aleatorio $U(0,1)$ siguiendo una distribución de probabilidad uniforme; si la probabilidad de aceptación es mayor a dicho número aleatorio la solución x' se acepta. Al parámetro *c* se le conoce como el parámetro de control (en el caso del sistema físico es la temperatura); por lo general se escoge un valor c_0 grande para que las soluciones de mala calidad tengan una alta probabilidad de ser seleccionadas, a medida que el valor de *c* disminuye, las soluciones de mala calidad que se pueden aceptar caen dentro de un intervalo que cada vez es más restringido, a medida que el valor de *c* se aproxima más a cero, igualmente la probabilidad de aceptar una solución de mala calidad se aproxima también a cero (Figura 3.4).

Para cierto valor de *c*, la secuencia de soluciones (configuraciones) exploradas puede visualizarse como una cadena de Markov con matriz de transición $\mathbf{P} = \mathbf{P}(c)$ y que tiene como elementos la probabilidad $G_{x,x'}$ de generar una solución *x'* a partir de una solución *x* dada, y la probabilidad $A_{x,x'}$ de aceptar dicha solución, de tal forma que la probabilidad de aceptar una solución *x'* generada a partir de una solución *x* será:

$$\mathrm{P}(c) = \begin{cases} \mathrm{G}_{x,x'}\mathrm{A}_{x,x'}(c) & x \neq x' \\ 1 - \sum_{l=1}^{|X|} \mathrm{G}_{j,j'}\mathrm{A}_{j,j'}(c) & x = x' \end{cases} \tag{3.6}$$

Donde $G_{x,x'}$ es la probabilidad de generación que está dada por la siguiente ecuación:

$$\mathrm{G}_{x,x'}(c) = \begin{cases} \dfrac{1}{|N(x)|} & x' \in N(x) \\ 0 & \text{en otro caso} \end{cases} \tag{3.7}$$

La probabilidad $A_{x,x'}$ de aceptar una solución será:

$$\mathrm{A}_{x,x'}(c) = \min\left\{1, \exp\left(-\frac{f(x') - f(x)}{c}\right)\right\} \tag{3.8}$$

Conforme desciende el valor de *c*, la probabilidad de aceptar un movimiento hacia una

configuración desfavorable o solución de mala calidad es cada vez menor y el algoritmo convergerá con probabilidad 1 a una configuración de mínimo costo; de igual manera después de realizar un descenso en el valor de *c*, se deben visitar un gran número de soluciones vecinas, de tal forma que el sistema alcance un nuevo estado estacionario (o de pseudo-equilibrio). Sin embargo ya que el estado estacionario se alcanza después de un número infinito de transiciones (cadena de Markov infinita) y en una aplicación práctica el algoritmo debe devolver una solución en un tiempo razonable, es necesario realizar las adecuaciones pertinentes del algoritmo: definir el valor inicial de c_0, definir el número de soluciones exploradas para cada valor de *c* (longitud de la cadena de Markov), seleccionar el esquema de enfriamiento y determinar el valor final del *c* (Laarhoven, Aarts, Lenstra 1992).

Generalmente el valor inicial del parámetro de control c_0 se determina en base a las características del problema, como regla general se selecciona un valor inicial tal que la probabilidad de aceptar una solución de mala calidad sea alta, de hecho se busca que la mayoría de las soluciones consideradas malas se acepten; en cuanto al enfriamiento, éste generalmente se realiza empleando la ecuación (3.9); si bien a lo largo de los años se han desarrollado otros sistemas de enfriamiento éste es de los más empleados.

$$c = \alpha \times c \tag{3.9}$$

Por regla general el enfriamiento debe ser lento y así se previene que el algoritmo quede atrapado en una solución de mala calidad, en la ecuación (3.9) la rapidez de descenso se controla mediante el valor de α el cual se escoge dentro del intervalo $\alpha = (0.8, 0.99)$; cabe señalar que determinar el esquema de enfriamiento adecuado es una etapa en la que se puede experimentar con varios sistemas; sin embargo si la implementación es nueva o se hace por primera vez, entonces conviene emplear (3.9) y posteriormente en las etapas de calibración para mejorar el desempeño se puede probar alguno de los otros sistemas (Aarts y Korst 1990; Laarhoven, Aarts, Lenstra 1992, Fouskakis, Dimitris. Et al 2002).

Igualmente, el número de puntos a explorar o transiciones que se realizan a un valor dado de *c* dependerá en parte del tamaño de la instancia a resolver tomando en cuenta que: si el

número de transiciones realizado no es suficiente, el algoritmo dejará zonas sin explorar y esto provocará que la posibilidad de quedar atrapado en una solución de mala calidad sea alta, en cambio si se fija un número muy elevado de transiciones se corre el riesgo de que el algoritmo ejecute más de las que son necesarias sin devolver ninguna mejora y desperdiciando en cambio tiempo que como ya se mencionó con frecuencia es escaso; un valor inicial adecuado puede ser igual al número de variables *n*, posteriormente durante la experimentación se pueden realizar distintas pruebas hasta encontrar un valor satisfactorio; en cuanto al criterio de paro se puede por ejemplo, fijar cierto número de transiciones en las que no se presente cambio en la solución.

No es de sorprenderse que este algoritmo tenga como principal desventaja el consumo de tiempo, por ello es necesario definir de forma adecuada la estructura de vecindades, el número de transiciones y la rapidez de enfriamiento, de tal forma que su desempeño sea bueno.

El desempeño (o comportamiento) del algoritmo se mide mediante el tiempo de cómputo y/o la calidad de la solución (existen otras medidas del desempeño, sin embargo éstas son las más comunes) y existe además abundante literatura con estudios donde se hacen recomendaciones para la selección de parámetros pero que dependen del problema que se esté tratando (coloración, asignación cuadrática, localización); sin embargo los valores para cierto problema pueden no ser recomendables para otro; por lo que es necesario llevar a cabo una serie de experimentos para calibrar de forma adecuada el algoritmo.

Con frecuencia a los algoritmos metaheurísticos es necesario acoplarlos con otros métodos de búsqueda para hacerlos más eficientes, o para poder emplearlos únicamente en cierta parte de la búsqueda. Recientemente se han realizado implementaciones con métodos de búsqueda sin cálculo de derivadas. A continuación, se presentarán algunos aspectos generales sobre estos últimos.

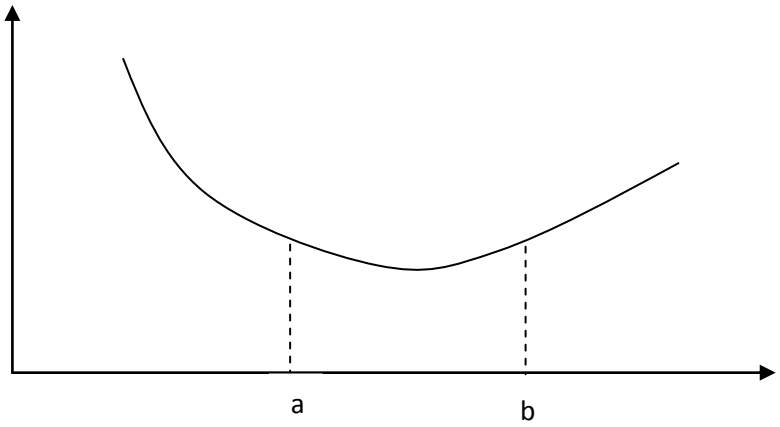


Figura 3.5 Función cuasiconvexa

3.4 Búsqueda unidimensional y el método de sección dorada

La búsqueda unidimensional es la columna vertebral de varios algoritmos de optimización no-lineal. Estos métodos no requieren que la función tenga derivada, únicamente que sea cuasi-convexa dentro de un intervalo $[\mathbf{a},\mathbf{b}]$(Figura 3.5).

Los métodos se clasifican en dos categorías: métodos de búsqueda simultánea y métodos secuenciales. En el primer caso los puntos a evaluar se conocen a priori, en el segundo, los siguientes puntos a evaluar se obtienen a partir de la iteración anterior.

3.4.1 Intervalo de incertidumbre

Suponga que deseamos minimizar la función $f(\rho)$ dentro del intervalo $\mathbf{a} \le \rho \le \mathbf{b}$. Como se desconoce la localización exacta del mínimo dentro del intervalo $[\mathbf{a},\mathbf{b}]$ a éste se le da el nombre de intervalo de incertidumbre. Sin embargo, es posible eliminar secciones dentro del intervalo para reducirlo y de esta forma aproximar el punto que minimiza el valor de la función $f(\rho)$.

Teorema 3.1 Sea una función $f: X \to \mathbb{R}$ estrictamente cuasi-convexa (Figura 3.5), dentro del intervalo $[\mathbf{a},\mathbf{b}]$.Sean $\rho,\zeta \in [\mathbf{a},\mathbf{b}]$ tales que $\rho < \zeta$. Si $f(\rho) > f(\zeta)$, entonces $f(\upsilon) \ge f(\zeta)$ para cualquier $\upsilon \in [\mathbf{a},\rho)$. Por el contrario, si $f(\rho) \le f(\zeta)$, entonces $f(\upsilon) \ge f(\rho)$ para toda

$\upsilon \in (\zeta, \mathbf{b}]$.

Prueba. Suponga que $f(\rho) > f(\zeta)$ y sea $\upsilon \in [\mathbf{a}, \rho)$. Por contradicción, suponga también que $f(\rho) < f(\zeta)$. Ya que ρ es una combinación convexa de υ y de ζ, y además si f es estrictamente cuasi-convexa, entonces:

$$f(\rho) = \max[f(\upsilon), f(\zeta)] = f(\zeta) \tag{3.10}$$

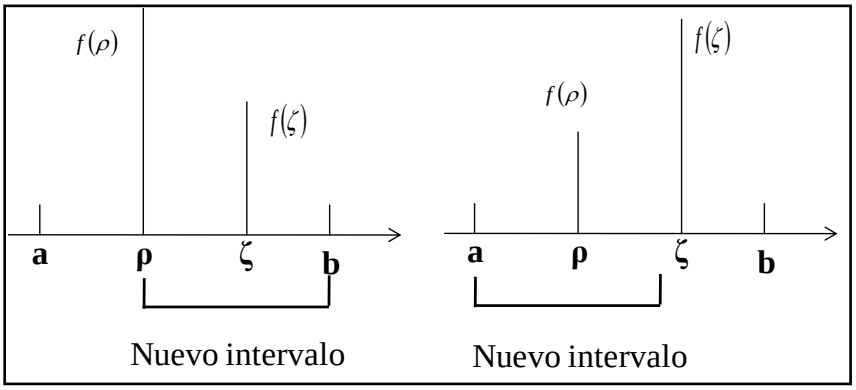


Figura 3.6 Selección de los intervalos de búsqueda

Lo cual contradice que $f(\rho) > f(\zeta)$. En consecuencia $f(\upsilon) \geq f(\zeta)$. A partir del teorema 3.1 y bajo la suposición de la estricta cuasi-convexidad $f(\rho) > f(\zeta)$, el nuevo intervalo de incertidumbre será $\upsilon \in (\rho, \mathbf{b}]$; por el contrario, si $f(\rho) \leq f(\zeta)$ entonces el nuevo intervalo de incertidumbre será $\upsilon \in [\mathbf{a}, \zeta)$ (Bazaraa, Sherali, Shetty 2006).

3.4.2 El método de sección dorada

El método de sección áurea o sección dorada, es un procedimiento de búsqueda secuencial que utiliza la información de iteraciones previas para obtener nuevos puntos, otros métodos son el Método de Fibonacci y el Método de la Dicotomía. El método surge cuando en el algoritmo de búsqueda de Fibonacci el número de puntos de medición se hace tender al infinito; en otras palabras, se genera una sucesión de intervalos de incertidumbre cuyas anchuras tienden a cero más rápido que por otros métodos. El procedimiento de búsqueda mediante sección dorada es como sigue (Bazaraa, Sherali, Shetty 2006):

Inicio

Hacer $r \leftarrow 1$

Seleccione $l > 0, [\mathbf{a}, \mathbf{b}]; \mathbf{a}_1 = \mathbf{a}, \mathbf{b}_1 = \mathbf{b}$

Evalúe

$$\rho_1 = \mathbf{a}_1 + (1-\gamma)(\mathbf{b}_1 - \mathbf{a}_1), \zeta_1 = \mathbf{a}_1 + \gamma(\mathbf{b}_1 - \mathbf{a}_1)$$

Evalúe $f(\rho_1), f(\zeta_1)$

Paso principal

1. Si $\mathbf{b}_r - \mathbf{a}_r < l$ entonces terminar = cierto, x^* se encuentra dentro del intervalo $[\mathbf{a}_r, \mathbf{b}_r]$, si $f(\rho_r) > f(\zeta_r)$ entonces ir a paso 2
2. Hacer $\mathbf{a}_{r+1} = \rho_r, \mathbf{b}_{r+1} = \mathbf{b}_r, \rho_{r+1} = \zeta_r, \zeta_{r+1} = \mathbf{a}_{r+1} + \gamma(\mathbf{b}_{r+1} - \mathbf{a}_{r+1}); f(\zeta_{r+1})$, ir al paso 4
3. Hacer $\mathbf{a}_{r+1} = \mathbf{a}_r, \mathbf{b}_{r+1} = \zeta_r, \zeta_{r+1} = \rho_r, \rho_{r+1} = \mathbf{a}_{r+1} + (1-\gamma)(\mathbf{b}_{r+1} - \mathbf{a}_{r+1}); f(\rho_{r+1})$ ir al paso 4
4. $r = r + 1$ y repite paso 1

Donde el valor de la constante es $\gamma = 0.618$ y recibe el nombre de sección dorada.

3.5 Análisis empírico de algoritmos metaheurísticos

Mediante el análisis experimental se determinan los valores o parámetros del algoritmo que hacen que su desempeño sea el mejor bajo ciertas condiciones al momento de emplearse para resolver un problema. Se debe recordar que si bien no existe una garantía acerca de que la solución que proporciona al final sea óptima, mediante un buen análisis experimental es posible hacer que el algoritmo realice una búsqueda eficiente.

3.5.1 Experimentación con algoritmos

No hay una metodología bien definida que permita analizar de forma experimental un algoritmo, por un lado están los diversos análisis de tipo teórico propuestos con la salvedad de que se basan en suposiciones difíciles de llevar a la práctica, si bien nada asegura que no se encuentre un caso extremo en el que un algoritmo presentará problemas, los casos

promedio pueden llegar a ser los más comunes; por otro lado está el análisis experimental donde se estudia el comportamiento del algoritmo a través de alguna medida de su desempeño, los resultados de la experimentación permitirán establecer un conjunto de parámetros que permitan al algoritmo realizar la búsqueda de forma eficiente .

Tabla 3.2

Ejemplo de variables a considerar del algoritmo de Recocido Simulado

Nombre	**Descripción**
Factor de control	c_0, "temperatura" inicial del sistema
Factor de enfriamiento	α, rapidez de descenso del factor de control
Esquema de enfriamiento	Geométrico, Logarítmico
Transiciones	Número de perturbaciones realizadas a un determinado valor de c_0
Iteraciones	Número de descensos del parámetro de control
Criterio de paro	Número de iteraciones, número de iteraciones sin cambio en la solución, etc.

Las medidas de desempeño que se emplean con mayor frecuencia para analizar cómo se comporta un algoritmo son el tiempo de ejecución para resolver una instancia dada y la calidad de la solución. Las variables que determinan el desempeño del algoritmo son básicamente los valores de los parámetros con los que funciona o se ejecuta (Tabla 3.2).

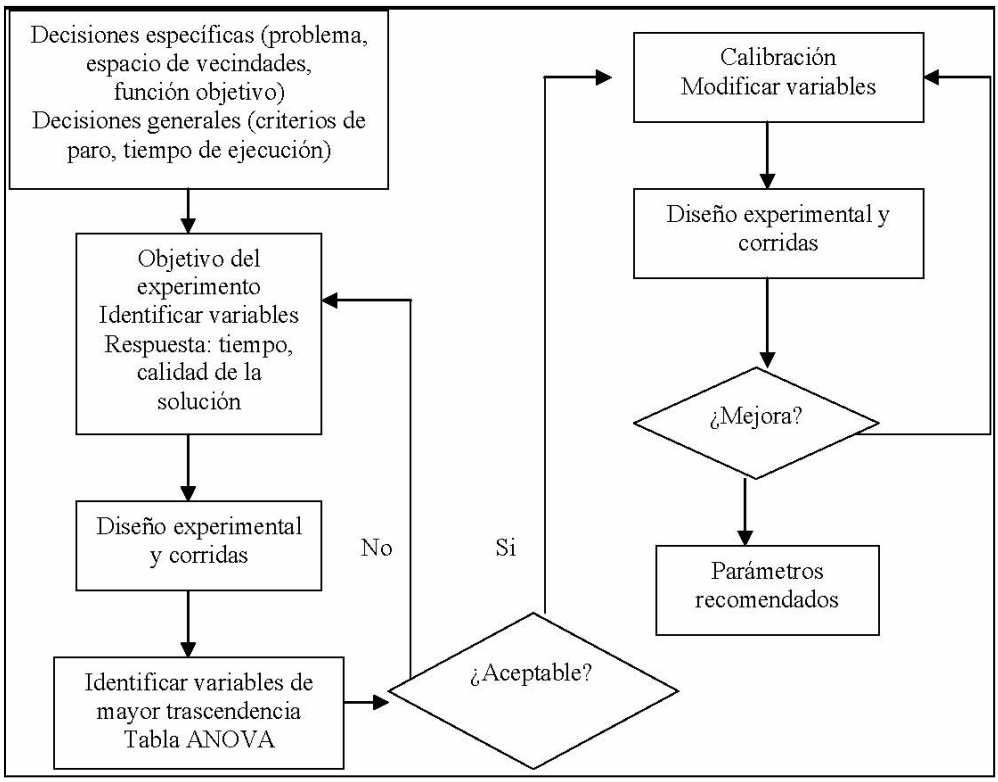
Para estos parámetros hay reportados valores que varían de acuerdo a la aplicación, si bien a algunos es posible acotarlos a ciertos intervalos y reducir así el número de posibilidades de selección, aún es amplio el margen de movilidad para establecerlos, más aún, un valor que se sabe es adecuado para un tipo de problema, no debe suponerse que lo será para resolver otro. Esto ha generado, en fechas recientes, que se empiecen a aplicar técnicas estadísticas para analizar los algoritmos y una de las más empleadas es el diseño de experimentos.

En la figura 3.7 se bosqueja el procedimiento para realizar el análisis de un algoritmo, dicho análisis depende mucho del problema que se está resolviendo y de cuál algoritmo se desea implementar.

Primeramente deben establecerse los objetivos de la experimentación: explorar el comportamiento bajo una serie de parámetros, mejorar soluciones reportadas anteriormente o disminuir el tiempo de cómputo de un algoritmo son algunas posibilidades; adicionalmente se debe trabajar en adaptar el algoritmo al problema que se va a resolver y por lo tanto hay que tomar dos tipos de decisiones: las decisiones específicas y las decisiones generales (Johnson. et al 1989).

De las primeras se puede mencionar por ejemplo la estructura de vecindades; el diseñar una estructura de vecindades es una parte crítica ya que buena parte del desempeño del algoritmo se basa en ella; si es una estructura que saca el máximo provecho de la información del problema el algoritmo realizará una búsqueda eficiente, de lo contrario, en la medida en que esta estructura no sea aprovechada el algoritmo disminuirá su eficiencia; otro tipo de decisión específica es la función objetivo con la que se evaluará la calidad de la función o bien la forma de determinar los pesos de cada variable presente, etc.

Figura 3.7 Implementación de un algoritmo



En el caso de las decisiones generales deberá definirse cuál será la medida de desempeño del algoritmo: tiempo de ejecución, calidad de la solución, porcentaje de mejora son algunas de las más empleadas; igualmente los criterios de paro del algoritmo o la forma de seleccionar una solución vecina son otras decisiones generales.

Una vez que se han resuelto las cuestiones relativas al problema y al algoritmo, el siguiente paso consistirá en identificar un primer conjunto de variables que deberán estudiarse para determinar la forma en que afectan el desempeño del algoritmo, este análisis puede simplificarse si por ejemplo se conocen de antemano algunos factores que de antemano se sabe afectan el desempeño (por ejemplo, en RS el factor de enfriamiento es uno de los parámetros que siempre debe considerarse). En caso de que la información disponible no sea satisfactoria se puede complementar el análisis del sistema con herramientas como el "diagrama de pescado" para identificar aquellos que hayan sido pasados por alto. Cuando se ha resuelto la cuestión sobre los factores a estudiar, el siguiente paso será realizar la experimentación necesaria para comprobar los efectos.

Por lo general se llega a experimentar moviendo una variable a la vez, sin embargo es sabido que esta estrategia no es muy recomendable ya que implica mucho tiempo y la

información que se obtiene será incompleta. Hoy en día se recomienda el empleo de diseños experimentales, los cuales permiten estudiar los efectos de varios factores a la vez; uno de los más empleados son los diseños factoriales que en las etapas iniciales de la experimentación permiten identificar las variables de mayor trascendencia o influencia sobre la respuesta; la tabla ANOVA brindará una ayuda inestimable ya que permitirá identificar qué variables son las de mayor peso.

Una vez que se ha determinado de forma satisfactoria los factores y su influencia sobre la respuesta, se podrá entonces pasar a lo que se conoce como la fase de calibración del algoritmo; en ésta se buscará refinar los valores asignados a los parámetros para resolver el problema y de los que se pueda tener la certeza que son adecuados, es decir la calidad de la solución obtenida será alta y el algoritmo devolverá dicha solución después de transcurrido un tiempo que se considerará justo; es importante resaltar que estos valores sólo serán aplicables para los límites de experimentación, no se recomienda extrapolar los resultados fuera de estos límites ya que se corre el riesgo de no obtener un buen desempeño o un resultado distinto.

Si se tiene más de una variable o factor a analizar, es recomendable emplear herramientas como las superficies de respuesta; conviene además manejar intervalos o regiones pequeñas para de esta forma, disminuir la incertidumbre existente acerca de la región donde se experimenta.

Todo el trabajo descrito tiene como resultado final el que el algoritmo implementado no se vea afectado de forma importante por cambios en el problema a resolver, aprovechará los recursos de forma adecuada y la solución obtenida será de gran calidad. En el siguiente capítulo se presenta la implementación del recocido simulado al problema de reaprovisionamiento multiproducto aplicando las recomendaciones señaladas anteriormente y realizando una comparación con otro procedimiento heurístico para verificar su desempeño.

Capítulo Cuarto: Implementación del algoritmo recocido simulado-sección dorada (RSSD) para el problema de reaprovisionamiento multiproducto con demanda probabilista

Introducción

En el presente capítulo, se comparará el algoritmo RS-SD contra el algoritmo de Eynan con la finalidad de tener elementos de juicio que permitan dar un panorama más amplio sobre la manera que se comportan ambos procedimientos.

Hasta ahora, la mayor parte de los trabajos desarrollados para resolver instancias del modelo JRP en ambientes de demanda probabilista se han enfocado a los sistemas continuos de control de inventario. Los más recientes relacionados con el problema de reaprovisionamiento conjunto son los de Eynan y Kropp (1998) donde la demanda se aproxima mediante una función de distribución Normal, y Fung y Ma (2001) donde se supone que la demanda se ajusta mediante una función de Poisson.

En ambos se trabaja con una función aproximada de costo y la solución se obtiene mediante técnicas heurísticas; sin embargo las pruebas realizadas en el primer caso se limitaron a un problema de 6 productos y en el segundo, el máximo número de productos fue 10. El resto de los trabajos realizados se han enfocado más en comparar políticas de control del inventario que en estudiar el desempeño de los algoritmos; también en el caso del trabajo de Eynan y Kropp (1998) el estudio se limitó a resolver una sola instancia, lo que genera una laguna ya que siempre es recomendable resolver varias instancias y además, se deben comparar los resultados contra otros procedimientos; en este caso particular ya que no existe librería de instancias (como puede ser el problema del agente viajero o el problema de cobertura de conjuntos) es recomendable generarlas de forma aleatoria procurando que el procedimiento para crearlas sea lo más sencillo y práctico posible para poder ser aplicado en trabajos posteriores.

4.1 Algoritmo heurístico de Eynan y Kropp (1998)

El procedimiento desarrollado por estos autores realiza una búsqueda local basándose en las condiciones de 1er orden, sin embargo el procedimiento no fue analizado con la debida profundidad, no se generaron instancias ni hay evidencia sobre su comportamiento para

distintos tamaños de instancias.

4.1.1 Caso con un solo producto

El análisis está basado en la ecuación de costo (2.12) que aquí se reproduce nuevamente suprimiendo únicamente el subíndice de la desviación estándar de la demanda:

$$CT = \frac{a}{T} + h\left[\frac{DT}{2} + z\sigma\sqrt{T+t}\right]$$

La función de costo se puede derivar con respecto a T obteniendo:

$$\frac{dCT}{dT} = -\frac{a}{T^2} + h\frac{D}{2} + \frac{hz\sigma}{2\sqrt{T+t}} = 0 \qquad (4.1)$$

Reacomodando se tiene:

$$h\frac{D}{2} + \frac{hz\sigma}{2\sqrt{T+t}} = \frac{1}{2}h\left(D + \frac{z\sigma}{2\sqrt{T+t}}\right) = \frac{a}{T^2} \qquad (4.2)$$

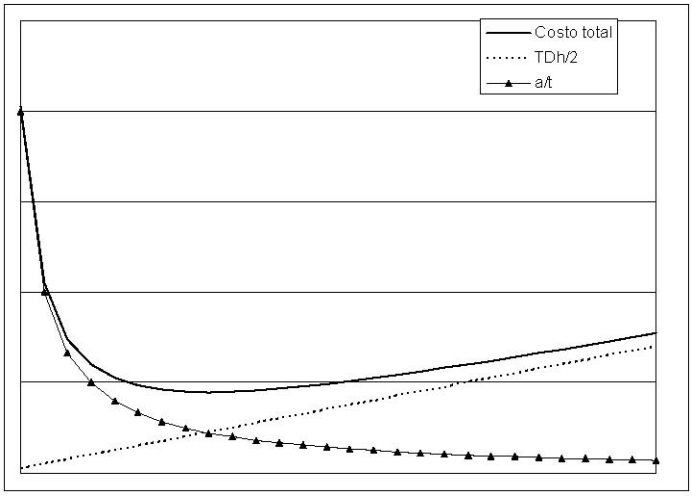


Figura 4.1. Gráfica de costos de inventario, caso determinista y un producto

Se denotará el ciclo de tiempo que minimiza el costo como T_0 para el caso determinista y T^* para el caso estocástico. El valor T_0 que minimiza el costo total en el caso determinista

se alcanza cuando la derivada del costo de acarreo de inventario iguala a la derivada del costo de pedido (Figura 4.1). Para el caso estocástico la derivada del costo de activación deberá igualar a la suma dada por el término $h\frac{D}{2}+\frac{hz\sigma}{2\sqrt{T+t}}$ (Figura 4.2 y ecuación 4.2); además el valor de T^* que minimiza el costo para el caso estocástico será menor al que se obtiene en el caso determinista: en el caso estocástico existe una penalización debida al inventario de seguridad el cual depende del ciclo de tiempo base; en otras palabras a medida que transcurre más tiempo entre la activación de dos pedidos, el tamaño del inventario de seguridad se incrementa para hacer frente a las fluctuaciones de la demanda.

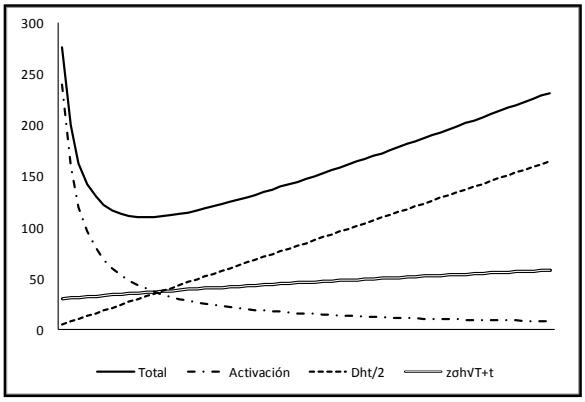


Figura 4.2. Gráfica de costos de inventario, caso estocástico y un producto.

Para obtener el valor de T^* que minimiza el costo de la ecuación (2.12) se puede proceder como sigue, resolviendo las condiciones de primer orden y reacomodando la expresión (4.1) se tendrá:

$$T=\left[\frac{2a}{h\left(D+\frac{z\sigma}{\sqrt{T+t}}\right)}\right]^{1/2} \qquad (4.3)$$

Se tiene una ecuación de la forma $x=g(x)$; para una ecuación de esta naturaleza la siguiente fórmula de iteración $x_{r+1}=g(x_r)$ produce en ocasiones una sucesión de aproximaciones $x_1, x_2, \ldots$ que convergen a una raíz a la cual se le llama punto fijo. En este

caso concreto, si se sustituye en la primera iteración a T_0 se tendrá entonces:

$$T_1 = \left[\frac{2a}{h\left(D + \frac{z\sigma}{\sqrt{T_0 + t}} \right)} \right]^{1/2} \tag{4.4}$$

Si $T_0 > T^*$, entonces $T_0 > T_r > T^*$, y substituyendo de forma iterativa los valores de T calculados con la ecuación (4.4) se obtiene una sucesión $T_0 > T_1 > \ldots > T_{r-1} > T_r > T^*$ y donde T^* es el valor que minimiza el costo total. Para $r \to \infty$ la sucesión convergerá a T^*, aunque dicho valor podrá aproximarse lo suficiente mediante la siguiente fórmula iterativa.

$$f(T_{r+1}) = \left[\frac{2a}{h\left(D + \frac{z\sigma}{\sqrt{T_r + t}} \right)} \right]^{1/2} \tag{4.5}$$

4.1.2. Caso multiproducto

La ecuación de costo para el problema de varios productos es:

$$CT = \frac{1}{T}\left(A + \sum_{i=1}^{n} \frac{a_i}{k_i} \right) + \sum_{i=1}^{n} \left(\frac{TD_i k_i h_i}{2} + h_i z_i \sigma_i \sqrt{k_i T + t_i} \right) \tag{4.6}$$

Como ya se mencionó anteriormente, en el problema multiproducto se debe determinar el ciclo de tiempo base T y el ciclo de tiempo individual T_i^* para cada i; sin embargo el valor T_i^* se restringe a ser un múltiplo entero del ciclo base T, y se expresa como $Tk_i = T_i^*$, por lo que las variables de decisión en el problema de reaprovisionamiento multiproducto son k_i y T. El valor de óptimo de k_i será aquel que cumpla con la siguiente condición:

$$CT(Tk) \leq CT(T(k+1)) \tag{4.7}$$

En adelante se suprimirá subíndice *i* para analizar el caso individual; realizando la substitución de las ecuaciones de costo individuales en (4.7):

$$\frac{a}{kT} + h\left[\frac{DkT}{2} + z\sigma\sqrt{Tk+t}\right] < \frac{a}{T(k+1)} + h\left[\frac{DT(k+1)}{2} + z\sigma\sqrt{T(k+1)+t}\right] \tag{4.8}$$

Reacomodando se tiene:

$$\frac{a}{k(k+1)T} < h\left[\frac{DT}{2}\right] + zh\sigma\left[\sqrt{T(k+1)+t} - \sqrt{Tk+t}\right] \tag{4.9}$$

El teorema del valor medio establece que existe un punto x $(kT \leq x \leq T(k+1))$ tal que:

$$\left[\sqrt{T(k+1)+t} - \sqrt{Tk+t}\right] = \frac{T}{2\sqrt{x+t}} \tag{4.10}$$

Substituyendo en (4.9) y reacomodando se tiene:

$$\left(\frac{1}{T}\right)\left[\frac{2a}{h\left(D + \frac{z\sigma}{\sqrt{x+t}}\right)}\right]^{1/2} < \sqrt{k(k+1)} \tag{4.11a}$$

O bien:

$$\frac{T_i^*}{T} < \sqrt{k(k+1)} \tag{4.11b}$$

Si $\frac{T_i^*}{T} < \sqrt{k(k+1)}$ entonces el valor de *k* será $k = k$, por otro lado si $\frac{T_i^*}{T} \geq \sqrt{k(k+1)}$, entonces el valor a emplear será $k = k+1$. Se puede entonces plantear la siguiente regla para decidir el valor de cada k_i:

$$\sqrt{k(k-1)} \leq \frac{T_i^*}{T} < \sqrt{k(k+1)} \tag{4.12}$$

Para obtener un método iterativo de cálculo de los valores de ciclo base de tiempo *T* es necesario hacer la siguiente puntualización: existe al menos un producto que se solicita cada ciclo, en este caso su valor T_i^* es el menor de todos los productos; para dicho producto, al que se le asignara el índice i =1, la frecuencia de activación será $k = 1$. Retomando la ecuación (4.6) e identificando el producto i =1 tenemos:

$$CT = \frac{1}{T}\left(A + a_1 + \sum_{i=2}^{n} \frac{a_i}{k_i}\right) + \frac{1}{2}\left(\frac{TD_1h_1}{2} + h_1z_1\sigma_1\sqrt{T+t_1}\right) + \sum_{i=2}^{n}\left(\frac{TD_ik_ih_i}{2} + h_iz_i\sigma_i\sqrt{k_iT+t_i}\right) \tag{4.13}$$

Relajando el requerimiento de variables enteras se pueden obtener las derivadas con respecto a *T* y con respecto a k_i para $i = 1,2,...,n$:

$$\frac{d(CT)}{dT} = -\frac{1}{T^2}\left(A + a_1 + \sum_{i=2}^{n} \frac{a_i}{k_i}\right) + \left(\frac{D_1h_1}{2} + \frac{h_1z_1\sigma_1}{2\sqrt{T+t_1}}\right) + \sum_{i=2}^{n}\left(\frac{D_ik_ih_i}{2} + \frac{h_iz_i\sigma_i}{2\sqrt{T+t_i}}\right) = 0 \tag{4.14}$$

$$\frac{d(CT)}{dk_i} = -\frac{a_i}{Tk_i^2} + \frac{TD_ih_i}{2} + \frac{Th_iz_i\sigma_i}{2\sqrt{Tk_i+t_i}} = 0; i = 1,2,...,n \tag{4.15}$$

Multiplicando (4.15) por $\frac{k_i}{T}$, substituyendo en (4.14) e igualando a cero:

$$-\left(\frac{A+a_1}{T^2}\right)+\frac{D_1h_1}{2}+\frac{h_1z_1\sigma_1}{2\sqrt{T+t_1}}=0 \qquad (4.16)$$

La ecuación (4.16) es similar a la ecuación (4.1) y con ella es factible generar un primer valor de *T*, manipulando la ecuación se tiene:

$$T=\left[\frac{2(A+a_1)}{h_1\left(D_1+\frac{z_1\sigma_1}{\sqrt{T_0+t_1}}\right)}\right]^{1/2} \qquad (4.17)$$

Y para la primera iteración se puede sustituir el ciclo de tiempo determinista del producto 1, es decir:

$$T_0=\left[\frac{2a_1}{h_1D_1}\right]^{1/2}$$

Una vez que se ha evaluado el valor *T* se deben recalcular los valores k_i; tomando la ecuación (4.15) y reacomodando se obtiene:

$$(k_iT)=\left[\frac{2a_i}{h_i\left(D_i+\frac{z_i\sigma_i}{\sqrt{(k_iT)_0+t_i}}\right)}\right]^{1/2} \qquad (4.18)$$

Y para la primera iteración se emplea:

$$(k_i T)_0 = \left[\frac{2a_i}{h_i D_i} \right]^{1/2}$$

El valor de cada k_i puede obtenerse con (4.18) y recordando que $Tk_i = T_i^*$, a continuación se redondea aplicando (4.12), posteriormente se debe evaluar el nuevo ciclo base; hay que hacer notar que la ecuación (4.14) puede reacomodarse como una ecuación de la forma $x = g(x)$, la ecuación queda:

$$T = \left[\frac{2\left(A + \sum_{i=1}^{n} \frac{a_i}{k_i} \right)}{\sum_{i=1}^{n} \left(k_i h_i \left(D_i + \frac{z_i \sigma_i}{\sqrt{T_0 k_i + t_i}} \right) \right)} \right]^{1/2} \qquad (4.19)$$

Para la primera iteración se sustituye el ciclo base de tiempo determinista:

$$T_0 = \left[\frac{2\left(A + \sum_{i=1}^{n} \frac{a_i}{k_i} \right)}{\sum_{i=1}^{n} (k_i h_i D_i)} \right]^{1/2}$$

Para cada valor de T calculado con (4.19), se calcula la combinación de valores de k_i aplicando (4.12) y este procedimiento se puede continuar hasta que no exista cambio en dos iteraciones consecutivas.

A continuación se muestra el procedimiento de cálculo tal y como aparece en (Eynan y Kropp 1998):

Paso 1 Calcular:

$$T_i^* = \left[\frac{2a_i}{h_i \left(D_i + \frac{z_i \sigma_i}{\sqrt{T_{0i} + t_i}} \right)} \right]^{1/2} \qquad (4.20)$$

$$\text{Donde: } T_{0i} = \left[\frac{2a_i}{h_i D_i} \right]^{1/2}$$

Paso 2 Identifique el producto con la T_i^* mínima y asigne i= 1 y k_1=1.

Paso 3 Calcular:

$$T = \left[\frac{2(A + a_1)}{h_1 \left(D_1 + \frac{z_1 \sigma_1}{\sqrt{T_0 + t_1}} \right)} \right]^{1/2} \qquad (4.21)$$

$$\text{Donde: } T_0 = \left[\frac{2(A + a_1)}{h_1 D_1} \right]^{1/2}$$

Paso 4 Redondear a entero los valores k_i de acuerdo a las siguientes reglas propuestas por Goyal (1974):

$$\sqrt{k(k-1)} \leq \frac{T_i^*}{T} < \sqrt{k(k+1)} \qquad (4.22)$$

Paso 5 Calcular:

$$T = \left[\frac{2\left(A + \sum_{i=1}^{n} \frac{a_i}{k_i} \right)}{\sum_{i=1}^{n} \left(k_i h_i \left(D_i + \frac{z_i \sigma_i}{\sqrt{T_0 k_i + t_i}} \right) \right)} \right]^{1/2} \qquad (4.23)$$

Donde: $T_0 = \left[\frac{2\left(A + \sum_{i=1}^{n} \frac{a_i}{k_i} \right)}{\sum_{i=1}^{n} \left(k_i h_i D_i \right)} \right]^{1/2}$

Paso 6 Repita 4 y 5 hasta que no exista mejora

Las pruebas realizadas sobre una instancia con 6 productos muestran que el algoritmo heurístico devuelve una solución muy cercana al óptimo, sin embargo al no realizarse experimentos adicionales ni comparaciones con otros algoritmos no hay mayor prueba de su desempeño.

Comienza
  Entrada $(c_0, c_f, \alpha, transiciones)$
  $c \leftarrow c_0$
  $K \leftarrow$ Genera solución inicial
  $T \leftarrow$ Búsqueda sección dorada
  $CT(T,K) \leftarrow$ Calcula costo de (T,K)
    Mientras $c > c_f$ hacer
      Para $cont \leftarrow 1$ a transiciones hacer
        $K' \leftarrow$ Genera solución vecina
        $T' \leftarrow$ Búsqueda sección dorada
        $CT(T',K') \leftarrow$ Calcula costo de (T',K')
        $\Delta \leftarrow CT(T',K') - CT(T,K)$
          Si $U(0,1) < \exp(-\Delta / c)$ hacer
            $K \leftarrow K', T \leftarrow T', CT(T,K) \leftarrow CT(T',K')$
      $c \leftarrow c\alpha$
  Salida $T, K, CT(T,K)$
Fin

Figura 4.3 Algoritmo RS-SD

4.2 Implementación del algoritmo Recocido simulado –Sección dorada

Implementar el algoritmo Recocido simulado-Sección dorada (y en general cualquier meta-heurística) implicó tomar un conjunto de decisiones previas divididas en decisiones específicas y decisiones generales.

Las primeras se refieren al problema con el que se está trabajando como la estructura de vecindades o el espacio de soluciones candidatas. Con respecto a las decisiones generales son aquellas que tienen que ver con el algoritmo en sí: número de iteraciones para alcanzar el pseudo-equilibrio o el criterio de paro (Johnson. et al 1989).

4.2.1 Decisiones específicas

En el caso de decisiones específicas, se definieron una serie de aspectos relacionados con el problema y la forma en que realizaría la búsqueda.

Función de costo: La ecuación (2.24) se emplea directamente. Como ya se vio, para el caso determinista, en el óptimo los costos de activación dados por el término $\left(A+\sum_{i=1}^{n}\frac{a_i}{k_i}\right)\left(\frac{1}{T}\right)$ igualan a los costos por acarreo del inventario $\sum_{i=1}^{n}\left(\frac{TD_i h_i k_i}{2}\right)$. Sin embargo en el caso de demanda aleatoria, hay que tomar en cuenta la expresión $\sum_{i=1}^{n}\left(h_i z_i \sigma_i \sqrt{Tk_i + t_i}\right)$ dentro de los costos de inventario; este parámetro no siempre es recomendable descartarlo ya que la diferencia en el valor de T o en las frecuencias de pedido en la práctica puede generar problemas o inclusive no ser viable, además la diferencia en costo puede ser significativa (Eynan y Kropp 1998, Silver 1985).

Cabe señalar que el valor de T para el caso aleatorio será menor que para el caso determinista porque al tomarse en cuenta la variabilidad de la demanda, el tiempo que transcurre entre dos pedidos consecutivos deberá ser menor. Ya que existe además un mínimo con respecto a T se puede emplear un algoritmo de búsqueda unidimensional para obtener una solución mejorada para cierta combinación de k_i, basta con fijar un intervalo

de búsqueda $[T_{\min}, T_{\max}]$ es decir, el valor de *T* que minimiza el costo se encontrará dentro del intervalo $T_{\min} \leq T \leq T_{\max}$.

En este trabajo se propone implementar la técnica Recocido Simulado para obtener los valores k_i y complementada con Sección Dorada para aproximar el valor de *T*, el algoritmo se presenta en la figura 4.3.

Espacio de soluciones para valores de k_i: Para cada producto, la frecuencia mínima de pedido será $k_i = 1$ que equivale a solicitar el producto *i* cada ciclo de tiempo y corresponde a la cota inferior de la frecuencia de activación de cada producto; para la cota superior primero mediante (4.24) se determina el valor del ciclo de tiempo óptimo individual:

$$T_i = \left[\frac{2a_i}{h_i D_i} \right]^{1/2} \tag{4.24}$$

$$T_{\min} = \min_i T_i \tag{4.25}$$

El ciclo de tiempo óptimo de cada producto T_i^* deberá encontrarse en el intervalo $T_{\min} \leq T_i^* \leq T_{i,\max}$, la cota inferior para el ciclo de tiempo se seleccionará a partir del valor T_i más pequeño empleando (4.25) y la cota superior será $T_{i,\max} = T_i$. Sin embargo el ciclo de tiempo individual se restringe a ser un múltiplo entero del ciclo de tiempo base, es decir $T_i = Tk_i$; al realizar las substituciones pertinentes se obtiene el intervalo de búsqueda de las frecuencias $1 \leq k_i \leq k_{i,\max}$. Con (4.26) se evalúa la cota superior de la frecuencia $k_{i,\max}$, en otras palabras, la frecuencia máxima de pedido posible de cada producto *i*:

$$k_{i,\max} = \left\lfloor \frac{T_i}{T_{\min}} \right\rfloor \tag{4.26}$$

Estructura de vecindades y esquema de perturbación: es conocido que RS tiene como

desventaja el consumo de tiempo de cómputo por lo que es de crucial importancia la definición de una estructura de vecindades adecuada que facilite la búsqueda. Para esta implementación, la solución vecina $(k_1,...,k_i^{'},...,k_n)$ obtenida de una solución $(k_1,...,k_i,...,k_n)$ se definió como el costo obtenido al modificar en una unidad el valor k_i de un sólo producto.

Cabe señalar que la cota superior no es la misma para todos los productos por lo que es necesario establecer un criterio de selección que:

1. Considere aquellos productos con frecuencias de pedido mayores a 1.
2. Favorezca aquellos índices con una cota superior más grande y en consecuencia más un espacio de soluciones más amplio.

Como la selección es aleatoria primeramente mediante la ecuación (4.27) se obtuvo la probabilidad de selección de cada producto y así, los índices con un espacio de búsqueda mayor tendrán una probabilidad mayor de ser seleccionados:

$$P(i)=\frac{k_{i,\max}}{\sum_i k_{i,\max}}, \forall k_{i,\max} \geq 2 \qquad (4.27)$$

En este caso se empleó el paradigma de perturbación, las perturbaciones son k_i+1 o k_i-1, lo que permite controlar la búsqueda de manera eficiente; el algoritmo encontrará tres casos posibles:

• Únicamente es posible incrementar el valor de k_i en 1 unidad: es decir k_i+1, lo que implica que el valor actual del producto *i* es 1, y por lo tanto la probabilidad de seleccionar esta operación será 1.

• Únicamente es posible disminuir el valor de k_i en 1 unidad: implica que el valor actual del índice se encuentra en su cota superior por lo que sólo es posible realizar la perturbación k_i-1 con probabilidad 1.

• Es posible incrementar o disminuir k_i en 1 unidad: implica que el valor actual del índice se encuentra dentro del intervalo $1<k_i<k_{i,\max}$; en este caso la probabilidad de seleccionar alguna de las dos operaciones es 0.5.

Búsqueda en la variable continua: el método de sección dorada se acopló al RS para

mejorar la solución; una vez que se ha perturbado la variable discreta a continuación se emplea el método de sección dorada para aproximar el valor de T que minimiza el costo para la combinación actual de valores de k_i. La búsqueda sobre T se realiza dentro del siguiente intervalo:

$$\mathbf{a} = T_{\min}; \mathbf{b} = T_{\max} \tag{4.28}$$

Donde **b** se obtiene empleando el ciclo base de tiempo determinista y fijando los valores $k_i = 1$, la expresión es: $T_{\max} = \left[\frac{2\left(A + \sum_{i=1}^{n} a_i \right)}{\sum_{i=1}^{n} (h_i D_i)} \right]^{1/2}$

4.2.2 Decisiones generales

Si bien los valores de los parámetros se seleccionaron de acuerdo a un diseño experimental, a continuación se describe de forma general aspectos relevantes de los mismos.

Esquema de enfriamiento: se aplicó la siguiente ecuación para disminuir el valor del parámetro control en cada iteración:

$$c_{r+1} = \alpha c_r \text{ para } r = 1,2,... \tag{4.29}$$

Donde $0.8 \leq \alpha \leq 0.99$; ya que se trata de una primera implementación, se realizó una exploración experimentando con distintos valores. De la misma forma, el valor inicial c_0 se debe seleccionar tal que la probabilidad de aceptar soluciones de mala calidad sea alta, permitiendo así al algoritmo escapar y explorar adecuadamente el entorno y no caer prematuramente en un mínimo local; también se experimentó con distintos valores.

Criterio de paro: El factor de control desciende hasta que se considera que el sistema está "congelado", para el algoritmo se fija un valor tal que $c_r < \varepsilon$, donde $\varepsilon = 0.01$.

Transiciones: Cada descenso de temperatura requiere generar un cierto número de

transiciones (o soluciones vecinas) para alcanzar un nuevo estado estacionario; ya que se manejaron instancias con distintos tamaños, la búsqueda deberá evaluar más puntos en aquellas con mayor número de productos, se manejó la siguiente ecuación para calcular el número de transiciones a c_r constante:

$$transiciones = N * n \tag{4.30}$$

Donde *n* es el número de productos y *N* es una constante que varía de acuerdo al tamaño de la instancia; para un valor pequeño de *n* entonces el número de transiciones necesarias será menor que para una instancia con un número de productos mayor. A medida que el tamaño de la instancia crece, el algoritmo requiere explorar una mayor cantidad de puntos (o soluciones); ya que se trata de una primera implementación, se optó por realizar pruebas con distintos valores de *N*.

4.3 Experimentos

Los algoritmos fueron programados en Visual Basic 2005 y se empleó una computadora con procesador Intel a 1.7 Ghz y 512 Mb de memoria RAM. Debido a que no existe una librería de instancias para probar el algoritmo, fue necesario generarlas aplicando algunos lineamientos usados en el caso determinista, pero además se incorporaron los parámetros del tiempo de espera y la desviación estándar.

Las instancias se generaron de forma aleatoria para *n*= 10, 20, 30, 40 y 50 productos; la demanda se generó en el intervalo (100-100,000), los costos de acarreo se generaron en el intervalo (0.5- 5) y los costos de activación en el intervalo (2- 3). El tiempo de espera se generó en el intervalo (1/40 – 1/6) unidades de tiempo y la desviación estándar de la demanda σ se obtuvo empleando la siguiente fórmula:

$$\sigma = D * factor \tag{4.31}$$

Donde *factor* es un número aleatorio entre (0.1 – 0.4). Se generaron instancias para valores de costo de activación mayor de *A* = 5, 10, 15, 20 y 30. Se generaron 100 instancias por

cada combinación de *n* y *A*, en total se resolvieron 2500 problemas.

En este trabajo el objetivo principal del experimento fue medir el desempeño del algoritmo comparando la solución obtenida con respecto a la que se obtiene empleando el algoritmo de Eynan (1998), para lo cual se empleó la siguiente clasificación: problemas tipo I, cuando $CT_{RS-SD} < CT_{Eynan}$, problemas tipo II cuando $CT_{RS-SD} > CT_{Eyna}$; adicionalmente se obtuvo el tiempo de ejecución del algoritmo.

4.3.1 Análisis preliminar

Se realizaron corridas empleando para el algoritmo RSSD los valores $c_0 = 50{,}100$, $\alpha = 0.90{,}0.95{,}0.99$ y $N = 1{,}10{,}50{,}100$ resolviendo las 2500 instancias observándose en todos los casos se obtenían entre 95.3 y 97.52% de problemas tipo I, también se observó que para los valores $\alpha = 0.99$ los tiempos de ejecución resultaron elevados, en contraparte, los tiempos correspondientes al algoritmo de Eynan fueron menores a 0.05 segundos en todos los casos (Tabla 4.1).

En base a esto, para las siguientes pruebas se decide restringir el intervalo de la rapidez de enfriamiento al intervalo $\alpha = [0.90{,}0.95]$; si bien la literatura sugiere que pueden emplearse valores menores, no es muy recomendable ya que se incrementa la probabilidad de que el algoritmo presente una convergencia prematura a un óptimo local.

Tabla 4.1

Tiempos de ejecución para el algoritmo RS-SD, análisis preliminar

Combinación	**Transiciones**	***t* (seg.)**	***n***
c_0	100	1.45	10
$\alpha = 0.99$	200	4.81	20
$N = 10$	300	10.04	30
	400	17.07	40
	500	26.28	50

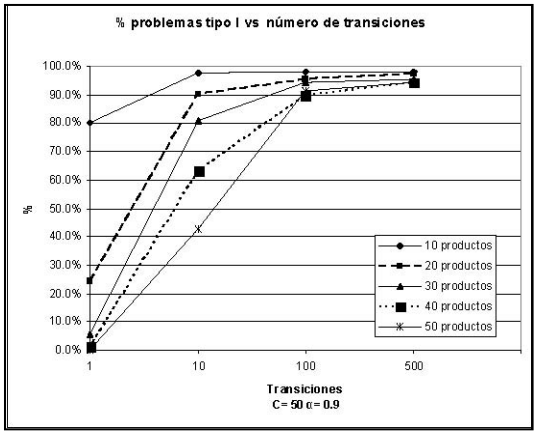


Figura 4.4 Porcentaje de problemas tipo I por tamaño de la instancia, fijos c_0=50 y $\alpha = 0.90$

Las siguientes pruebas se realizaron con distintos valores del número de transiciones y detectar así, el punto a partir del cual el % de problemas tipo I ya no se incrementa, en otras palabras, un punto de estancamiento. Se resolvieron 500 problemas para cada número de productos. La exploración muestra que el porcentaje de problemas tipo I se estanca a partir de 100 transiciones para 10 productos, 400 transiciones para 20 y 30 productos y 500 transiciones para 40 y 50 productos (Figura 4.4).

Los tiempo de ejecución observados disminuyeron de forma considerable, disminuyendo hasta en un 78.5% (Tabla 4.2).

Tabla 4.2

Tiempos de ejecución en segs. para RS-SD, fijos $c_0, \alpha = 0.99$.

# productos	***Transiciones***	**t (seg.)**	***Transiciones***	**t (seg.)**	**Mejora en tiempo (%)**
10	500	0.0685	100	0.0625	8.76%
20	1000	0.21	400	0.1090	48.10%
30	1500	0.44	400	0.1562	64.50%
40	2000	0.75	500	0.2031	72.92%
50	2500	1.15	500	0.2343	79.63%

4.3.2 Propuesta de modelo.

El análisis experimental del desempeño de un algoritmo es muy importante ya que para

seleccionar valores para los parámetros casi siempre se da el caso en que los intervalos (y por lo tanto, la cantidad de opciones) son muy grandes, si bien y como ya se mencionó, existen algunos lineamientos generales, aún así queda un gran margen de selección. Hoy en día los diseños experimentales se han revelado como una herramienta muy útil para analizar de forma empírica las implementaciones de metaheurísticas y estudiar los efectos de los distintos parámetros.

Para la nueva exploración se delimitó la nueva región recurriendo al diseño factorial conocido como 2 factorial, o $2^{\mathbf{K}}$; y donde **K** es el número de factores o variables que se incluyen en la experimentación para medir su efecto sobre la respuesta y se delimita a una región fijando dos niveles: nivel superior y nivel inferior (Montgomery 2000); los factores o variables que se estudiaron fueron nuevamente el parámetro de control c_0 inicial, el parámetro de enfriamiento α y la constante *N*.

Tabla 4.3

Diseño experimental

Parámetro	**Superior**	**Inferior**
$\boldsymbol{A}$	0.95	0.9
$\boldsymbol{c_0}$	100	50
N	10	1

Tabla 4.4

Número de transiciones para cada tamaño de instancia

	transiciones	
n	***N*= 1**	***N*= 10**
10	10	100
20	20	200
30	30	300
40	40	400
50	50	500

Se realizaron dos experimentos por cada combinación; adicionalmente se agregó un punto central para complementar el análisis y verificar la influencia de curvatura en la región, se resolvieron las 2500 instancias con cada una de las 17 combinaciones de parámetros (Tablas 4.3 y 4.4).

Tabla 4.5

Resultados de la experimentación

				Problemas tipo I		Problemas tipo II	
Combinación	c_0	α	N	**Frecuencia**	**%**	**Frecuencia**	**%**
1	100	0.9	1	2395	95.80	105	4.20
				2398	95.92	102	4.08
2	50	0.9	1	2396	95.84	104	4.16
				2389	95.56	111	4.44
3	100	0.9	10	2430	97.20	70	2.80
				2433	97.32	67	2.68
4	50	0.9	10	2429	97.16	71	2.84
				2433	97.32	67	2.68
5	100	0.95	1	2411	96.44	89	3.56
				2413	96.52	87	3.48
6	50	0.95	1	2410	96.40	90	3.60
				2408	96.32	92	3.68
7	100	0.95	10	2438	97.52	62	2.48
				2434	97.36	66	2.64
8	50	0.95	10	2434	97.36	66	2.64
				2434	97.36	66	2.64
9	75	0.95	75	2430	97.20	70	2.80

En la tabla 4.5 se muestran los resultados obtenidos agrupados para cada combinación de parámetros del algoritmo, se reportan tanto la frecuencia como el porcentaje de problemas tipo I y tipo II. El algoritmo RS-SD mejoró los resultados obtenidos por la técnica de Eynan para todas las combinaciones de parámetros; se observa que el porcentaje de problemas tipo I varió entre 95.56 y 97.36 %, el % de problemas tipo II varió entre 4.46 y 2.8. Se puede decir que todas las combinaciones de parámetros del algoritmo dentro de los límites del diseño experimental hacen que el algoritmo tenga un buen desempeño.

Tabla 4.6

Diferencia promedio, problemas tipo I

n	Diferencia
10	3.87
20	4.81
30	10.56
40	12.49
50	15.88

En la tabla 4.6 se muestra la diferencia promedio entre las soluciones obtenidas con cada algoritmo al resolver las instancias, un aspecto acerca del algoritmo de Eynan que no había sido evaluado de manera más extensa fue el de la calidad de la solución, como se ve en dicha tabla, cuando se compara con el algoritmo RS-SD, la solución se degrada conforme aumenta el tamaño de la instancia.

En la tabla 4.7 se muestran los tiempos obtenidos para los niveles superior e inferior del experimento y que corresponden a las cotas de tiempo observadas para los valores explorados dentro del diseño experimental. Una característica del algoritmo RS es el consumo de tiempo, para el caso del algoritmo de Eynan los tiempos de ejecución siempre se mantuvieron por debajo de 0.05 segundos.

Tabla 4.7

Diferencia promedio, problemas tipo I

Combinación	***Transiciones***	**Tiempo (seg.)**	***n***
c_0=100	100	0.8442	10
α=0.95	200	3.22	20
N=10	300	7.614	30
	400	13.483	40
	500	21.53	50
c_0=50	10	0.044	10
α=0.90	20	0.173	20
N=1	30	0.383	30
	40	0.67	40
	50	1.06	50

A continuación se verificaron los resultados mediante el análisis de la varianza; el diseño experimental original se muestra en el apéndice 2.

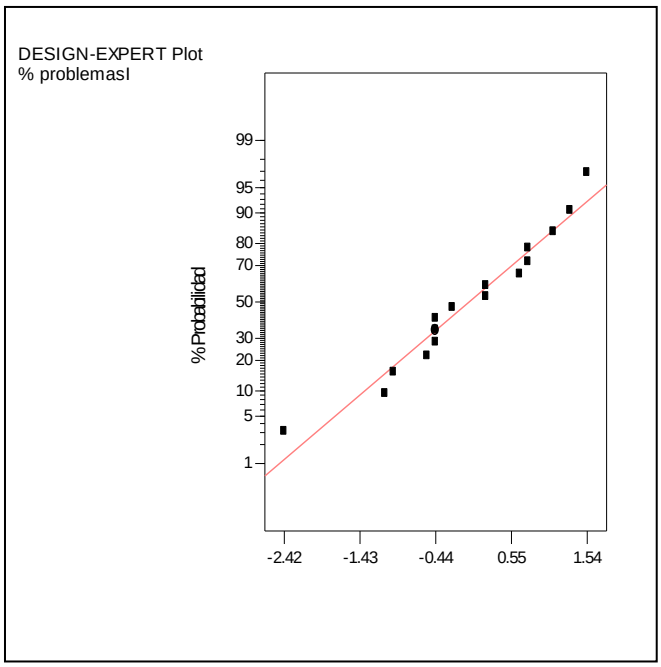


Figura 4.5 Gráfica de probabilidad Normal, Problemas Tipo I

4.3.3 Verificación del experimento

El análisis toma como respuesta el porcentaje de problemas tipo I. La figura 4.5 de distribución de los residuales muestra que éstos se comportan siguiendo una distribución Normal, en la figura 4.6 se observan los residuales por experimento; no hay evidencia aparente de un patrón que indicara que las condiciones del experimento varíen para cada combinación.

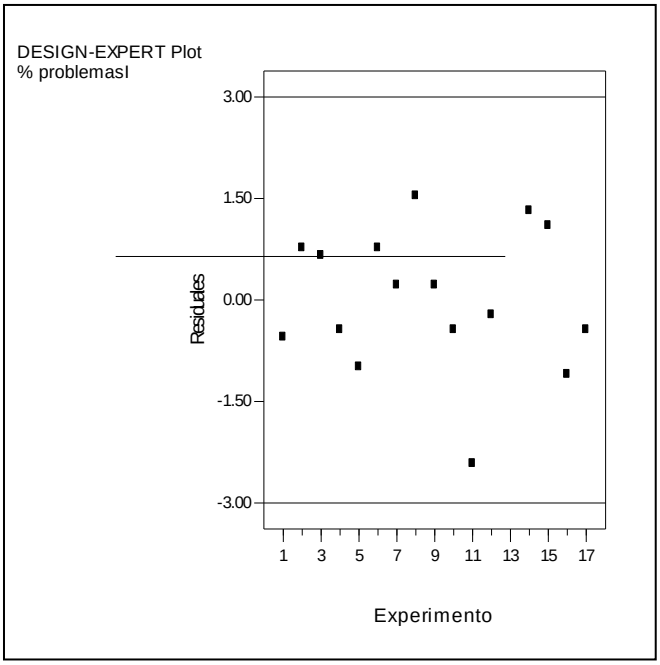


Figura 4.6 Residuales por experimento, Problemas Tipo I

Tabla 4.8

ANOVA, Problemas tipo I

	Suma de	**Cuadrado**		
Fuente	**cuadrados**	**Medio**	**Prob > F**	**Conclusión**
Modelo	6.87	2.29	< 0.0001	significativo
A	0.62	0.62	< 0.0001	
N	6	6	< 0.0001	
α*N*	0.24	0.24	0.0005	
Curvatura	0.22	0.22	0.0007	significativa
Residual	0.13	0.011		
Falta de Correlación	0.047	0.012	0.4218	no significativa

En la tabla 4.8 de análisis de la varianza podemos observar que las variables significativas del experimento son la rapidez de enfriamiento α y la constante N, así como el efecto combinado de ambos (interacción); el valor inicial del parámetro de control c_0 no es significativo en el experimento. El modelo es significativo, es decir, el fenómeno puede explicarse con las variables presentes; la curvatura en la zona es significativa en la región de experimentación, sin embargo la falta de correlación que pudiera deberse a dicha curvatura no es significativa y por lo tanto no se requieren términos de orden superior para ajustar los datos.

Se ajustaron otros modelos de regresión lineal, observando cómo cambia el coeficiente de regresión, dichos ajustes se muestran en la tabla 4.9:

Tabla 4.9

Modelos de regresión

No.	Modelo	R^2
1	$\%\,\text{problemas tipo I} = 95.99 + 0.14N$	83.10
2	$\%\,\text{problemas tipo I} = 88.49 + 8.11\alpha + 0.14N$	92.23
3	$\%\,\text{problemas tipo I} = 88.35 + 0.002c_0 + 8.11\alpha + 0.14N$	92.74
4	$\%\,\text{problemas tipo I} = 82.95 + 14.10\alpha + 1.14N - 1.09\alpha N$	95.56

Tabla 4.10

Análisis del modelo 4

Desv. Std.	0.11	**R^2**	95.56
Media	96.74	**R^2 ajustado**	95.46
C.V.	0.11		

La tabla 4.10 tenemos datos adicionales del modelo 4; además de que el valor del coeficiente de regresión R^2 fue el mejor, se observó que valor ajustado de R^2 cambia muy poco con respecto al valor del coeficiente de regresión original, lo que indica que el modelo no presenta variaciones importantes al agregarse nuevos términos, todo lo contrario, el valor del coeficiente de regresión baja ligeramente, por lo tanto los resultados observados pueden explicarse únicamente con las variables que ya han sido incluidas.

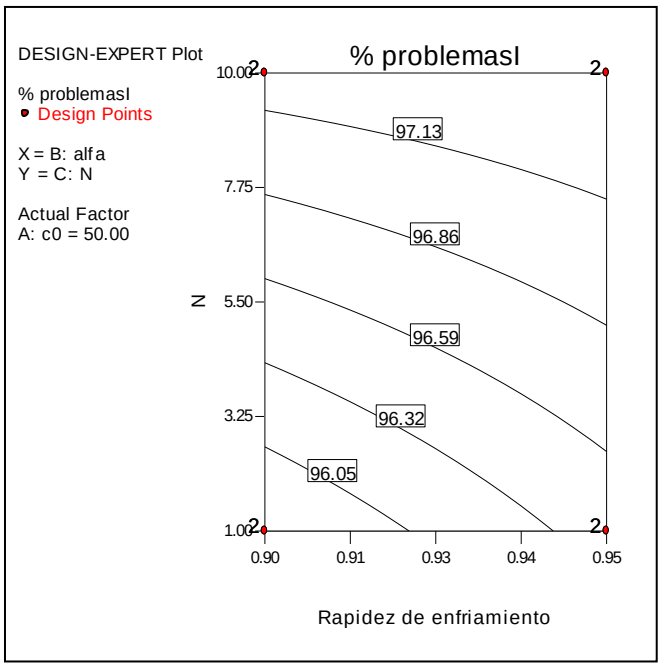


Figura 4.7 Gráfica de contornos, Problemas Tipo I

4.3.4 Análisis del modelo

En la figura 4.7 se muestran los contornos dentro de la región de experimentación, se observa que los problemas tipo I aumentan a medida que se incrementa el valor de α, lo que se traduce en un descenso cada vez más lento en el valor del parámetro de control c_0, sin embargo como se verá más adelante, el tiempo de ejecución del algoritmo se incrementa de forma importante al modificar este parámetro.

En la figura 4.8 se muestra la superficie la región de experimentación. Se observa que el porcentaje de problemas tipo I dentro de los límites asignados al experimento es elevado, de hecho con la combinación en los niveles inferiores es factible obtener más del 95% de problemas tipo I.

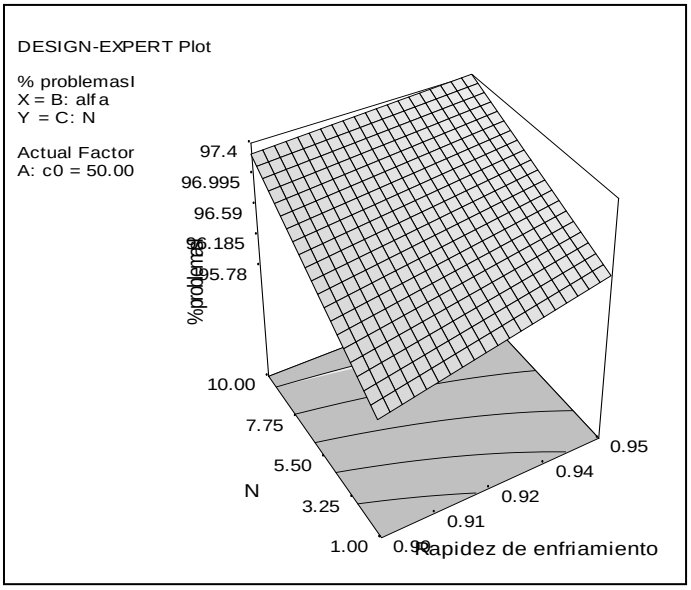


Figura 4.8 Superficie de respuesta, Problemas Tipo I

Los valores de los parámetros de RS-SD que se seleccionen dentro el intervalo del experimento, aseguran que en no menos del 95.52% de los casos el algoritmo devolverá mejor solución que el procedimiento de Eynan.

En el modelo 4 se puede observar que el coeficiente de la rapidez de enfriamiento α indica que al modificarse su valor, los problemas tipo I aumentarán de forma importante, siendo la variable con el mayor efecto sobre la respuesta. El valor de la constante N (con el que se obtiene el número de puntos evaluados) tiene también efecto significativo sobre la respuesta sin embargo, es pequeño comparado con el del parámetro de enfriamiento, finalmente el efecto debido a la interacción de ambos factores es la menor.

4.3.5 Experimentación con los parámetros

El elevado % de problemas tipo I observado sugiere en un principio que los valores de los parámetros fueron muy buenos, es decir el algoritmo realiza una búsqueda muy eficiente de la región(el criterio para la selección de la solución vecina es adecuado), el número de transiciones es evidentemente el necesario para explorar la región y la rapidez con la que desciende el valor de c es adecuada para evitar un "enfriamiento" abrupto que tenga como

consecuencia que el algoritmo caiga en un óptimo local; sin embargo es recomendable continuar explorando con nuevos valores para verificar si es posible modificarlos, la información obtenida permitirá en un momento dado sugerir ya sea nuevos valores o bien, concluir que cualquier posterior modificación no genere un beneficio patente; todo se puede realizar empleando la información obtenida del diseño experimental y del modelo, los cuales dan una pauta sobre la dirección en que (en su caso) debe modificarse cada parámetro y una predicción del resultado.

El objetivo fue observar cambios del % de problemas tipo I dentro de la región de experimentación; se fijaron los valores de $N = 1$ y $c_0 = 50$ y se perturbó la rapidez de enfriamiento a $\alpha = 0.925$ ya que es la variable con el mayor efecto. Al sustituir los valores en el modelo de regresión éste predice que el porcentaje de problemas tipo I será 96.13% global. Al correr el experimento se obtiene un resultado global de 96.14% de problemas tipo I.

La distribución de los problemas tipo I de acuerdo al número de productos se muestra en la tabla 4.11 y la figura 4.9, y se aprecia que dicho porcentaje varía con el tamaño del problema disminuyendo a medida que el número de productos crece.

Tabla 4.11

% de problemas tipo I, por número de productos, fijos N, = 1 c_0 =50

	n	**10**	**20**	**30**	**40**	**50**
α	0.90	98.6	97.4	95.6	93.2	94.4
	0.925	98.6	97.4	96.0	94.0	94.8
	0.95	98.6	97.8	96.4	94.0	95.2

Por otro lado, α se puede disminuir por debajo de 0.9 sin embargo, si bien dicho factor impacta fuertemente en el tiempo de ejecución, dicha disminución tampoco es recomendable ya que se corre el riesgo de que la rapidez de enfriamiento sea brusca provocando que el algoritmo caiga en un óptimo local.

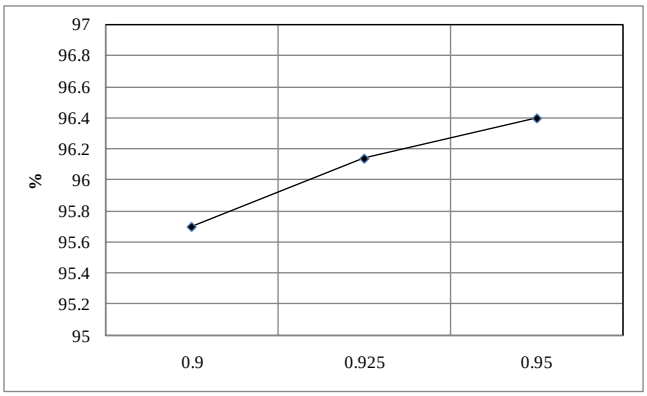


Figura 4.9 % de problemas tipo 1, fijos N=1 y c_0 = 50

La siguiente variable que se puede perturbar es el valor de la constante N que se emplea para determinar el número de transiciones (o soluciones) que se exploran a un valor c_r constante. En este caso existe un mayor margen de movilidad, de hecho sólo se sugiere en la literatura que el número de transiciones sea tal que permita una buena exploración de la región (véase por ejemplo Hoos y Stützle 2005), por lo cual debe determinarse experimentalmente para encontrar un valor conveniente de puntos explorados y que al mismo tiempo sea el necesario evitando así invertir tiempo de forma excesiva.

Se perturbó el valor de la constante realizándose una prueba con el valor N=5; el modelo de regresión indica que se obtendrán 96.67% de problemas tipo I, el resultado global del experimento es de 97 %. La forma en que se distribuyeron los porcentajes de acuerdo al tamaño de la instancia se muestran en la tabla 4.12 y en las figuras 4.10 y 4.11.

Tabla 4.12

% de problemas tipo I, por número de productos, fijos α = 0.9, c_0=50

	n	**10**	**20**	**30**	**40**	**50**
N	1	98.6	97.4	95.6	93.2	94.4
	5	98.8	97.8	97.4	94.8	96.2
	10	98.8	98	97	95.4	96.6

Se puede observar que el número de transiciones realizado tiene un efecto en el porcentaje de problemas tipo I obtenido para cada tamaño del problema, así para 10, 20 y 30 productos

deben realizarse al menos un número de transiciones igual al tamaño de la instancia para obtener 98.6, 97.4 y 95.6 % de problemas tipo I respectivamente, en este caso equivale asignar un valor a la constante *N*=1; sin embargo para *n*= 40 y 50 productos la situación es distinta ya que se requieren al menos 400 y 500 transiciones para obtener 95.4 y 96.6% respectivamente, lo que equivale a un valor de la constante *N*=10, a partir de este punto se observa un comportamiento asintótico en la respuesta obtenida.

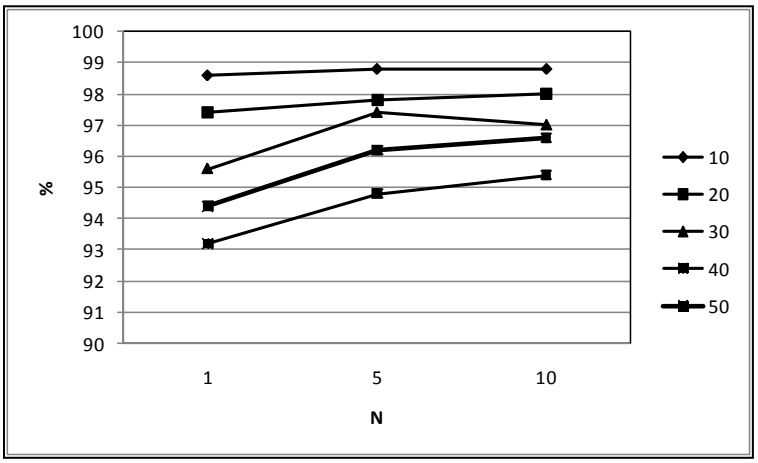


Figura 4.10 Distribución de problemas tipo I obtenido por número de productos y distintos valores de *N*, fijos α=0.9 y $c_0 = 50$

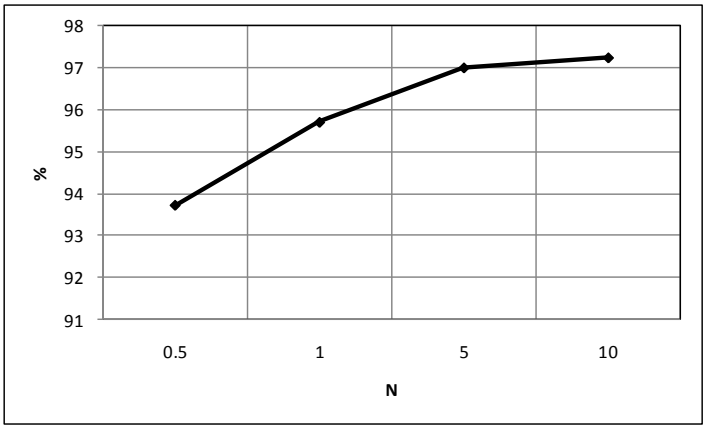


Figura 4.11 % de problemas tipo I y distintos valores de *N*, fijos α=0.9 y $c_0 = 50$

Como se observa, el valor de la constante se puede disminuir por debajo de N=1 (y en consecuencia el número de transiciones), sin embargo el nuevo valor caerá fuera del nivel inferior del diseño experimental, por lo que no es recomendable realizar una extrapolación con el modelo para predecir el resultado.

El nuevo valor de N se fijó en 0.5 y se realizó la corrida experimental, el % global de problemas tipo I obtenido fue 93.73, que está por debajo del obtenido con los valores del diseño experimental (figura 4.11).

Tabla 4.13

Parámetros sugeridos para el algoritmo RS-SD

c_0	50
N	1
α	0.90

En base a la exploración realizada, cabe la posibilidad de proponer un conjunto de valores para los parámetros del algoritmo RS-SD y que aseguren una buena solución. Como referencia puede considerarse que el porcentaje de problemas tipo I no sea menor a un cierto valor, para la situación que nos ocupa se puede tomar no menos del 95% global. Los parámetros se muestran en la tabla 4.13.

Es importante aclarar que el hecho de obtener un modelo de regresión lineal no implica necesariamente que no exista un modelo del tipo no-lineal que presente un mejor ajuste de los datos y que además considere todos los parámetros.

Por otro lado, también es viable construir un diseño de superficie de respuesta, aunque ya se observó que el modelo tipo lineal ajusta de manera adecuada, puede ser recomendable dada la evidencia de la presencia de una curvatura dentro de la región.

Conclusiones

El algoritmo de recocido simulado se implementó para resolver el modelo del problema de reaprovisionamiento conjunto con demanda aleatoria, el cual es un problema de optimización global, con variables mixtas y no lineal. Existe un procedimiento heurístico en la literatura, sin embargo ha sido poco probado.

De la misma forma, a la fecha existen pocas implementaciones de técnicas metaheurísticas a éste problema pero todas al modelo con demanda determinista.

El algoritmo implementado en el presente trabajo, requirió complementarse con una búsqueda unidireccional, se escogió Sección Dorada por su rápida convergencia y que no requiere la evaluación de derivadas.

Se realizó un extenso análisis del desempeño del algoritmo empleando diseños experimentales; se comparó el algoritmo nuevo con otra técnica heurística desarrollada por Eynan. Et al (1998) para comparar la calidad de la solución y el tiempo de ejecución.

Los resultados obtenidos mostraron que el algoritmo RSSD alcanza mejores soluciones al ser comparadas con el algoritmo de Eynan. Et al (1998). Alrededor del 96% del total de las 2500 instancias resueltas fueron resueltas con un costo más bajo con el nuevo algoritmo; el nivel de mejora fue desde 0.1 hasta 656 unidades. Se hizo una calibración posterior para bajar el tiempo de ejecución del algoritmo y se logró disminuir el tiempo de ejecución hasta en 80%. Cabe señalar sin embargo que el algoritmo de Eynan devuelve una solución en un tiempo menor.

Este trabajo contribuyó en los siguientes aspectos:

1. Se obtiene un algoritmo que mejora soluciones reportadas en la literatura.
2. Se realizó por primera vez un análisis comparativo del algoritmo RSSD contra el desarrollado por Eynan. Et al. (1998).
3. Se establecen criterios para generar instancias de prueba para el modelo de reaprovisionamiento conjunto con demanda probabilista.
4. Se implementa por primera vez una técnica metaheurística a esta variante del problema de reaprovisionamiento conjunto.
5. Se aplicó el diseño de experimentos para probar y calibrar una técnica meta-heurística. Aún es común encontrar implementaciones con poco manejo de

herramientas estadísticas.

Líneas futuras de investigación

El control del inventario es una faceta crucial en cualquier organización. Hoy en día, es necesario que las industrias establezcan controles que les permitan dar un alto grado de servicio, esto les permitirá mantenerse dentro de la competencia.

Los inventarios sin embargo, cuentan con una gran variedad de productos, estos deben planearse de tal forma que los costos en los que se incurre sean mínimos. Dar un alto grado de servicio y al mismo tiempo minimizar costos no es una tarea fácil, menos aún si se requiere coordinar más un artículo.

El problema de reaprovisionamiento conjunto se ha estudiado por muchos años, sin embargo para el caso con demanda aleatoria se han dedicado pocos trabajos, al grado de que existe únicamente un algoritmo heurístico, desafortunadamente éste ha sido poco probado. Se considera que es un problema de optimización global con variables mixtas y además es no lineal; estos son un tipo de problemas muy difícil de resolver ya que cuentan con características como el ser no-convexos y contar con más de un mínimo local. Si bien se han desarrollado técnicas exactas, éstas aún presentan algunas dificultades que los hacen converger hacia soluciones subóptimas.

Una alternativa para resolver estos modelos es recurrir a las técnicas metaheurísticas, que se han revelado como una magnífica opción para dar solución a variedad de problemas combinatorios; algoritmos genéticos, búsqueda tabú y recocido simulado son algunas de las más empleadas.

El recocido simulado pertenece al conjunto de los Algoritmos de Búsqueda Local Aleatoria; su implementación a primera vista sencilla, requiere sin embargo una gran esfuerzo: tiempo de ejecución, estructura de vecindades, criterios de paro son algunos tópicos que deben cubrirse para que el algoritmo sea eficiente.

En el presente trabajo, se implementó el recocido simulado acoplado con un algoritmo de búsqueda unidireccional al problema de reaprovisionamiento conjunto con demanda aleatoria. Existe una variante con demanda tipo Poisson, para la cual se ha reportado únicamente una técnica heurística; la instancia más grande empleada en el análisis fue de

10 productos. Sería interesante realizar implementaciones de otras técnicas igualmente para resolver el problema. Para el caso con demanda determinista se han implementado Algoritmos Genéticos y Búsqueda Dispersa.

El modelo de costo del problema de reaprovisionamiento conjunto es similar a otro problema muy trabajado en la literatura y que se conoce como Coordinación del Tamaño Económico de Lote (ELSP por sus siglas en inglés). Algoritmos Genéticos y Búsqueda Tabú han sido las técnicas metaheurísticas empleadas con buenos resultados. Convendría implementar RSSD a éste problema para verificar su desempeño.

Como se mencionó anteriormente, en años recientes se ha comenzado a aplicar técnicas estadísticas para probar y calibrar algoritmos metaheurísticos; los diseños experimentales son un valioso auxiliar ya que permiten economizar recursos y al mismo tiempo se realizan pruebas que arrojan la máxima información acerca del sistema que se está analizando. Se pueden incluir variables categóricas como el tipo de esquema de enfriamiento o alguna estructura de vecindades distinta a la manejada.

Apéndice 1

Código del Algoritmo RSSD

```
For i = 1 To n

                tind(i) = System.Math.Sqrt(2 * s(i) / (d(i) * h(i)))

            Next
            'evalua la t minima

            smallest = 1000000

            For i = 1 To n


                If tind(i) < smallest Then smallest = tind(i)

            Next i

            'evalua la t maxima

            biggest = -5

            For i = 1 To n

                If biggest < tind(i) Then biggest = tind(i)

            Next i

            For i = 1 To n

                kup(i) = System.Math.Floor(tind(i) / smallest)

            Next

            sumakups = 0

            For i = 1 To n
                If kup(i) >= 2 Then

                    sumakups = sumakups + kup(i)

                End If

            Next

            For i = 1 To n

                If kup(i) >= 2 Then

                    kprob(i) = kup(i) / sumakups
                Else
```

```
        kprob(i) = 0

    End If

Next

revisa = 1

Do Until revisa > (n - 1)

    For cuenta = 1 To n - revisa

        If tind(cuenta) >= tind(cuenta + 1) Then

            kprobcambio = kprob(cuenta)
            tindcambio = tind(cuenta)
            kcambio = kup(cuenta)
            dcambio = d(cuenta)
            hcambio = h(cuenta)
            scambio = s(cuenta)
            ltimecambio = ltime(cuenta)
            sigmacambio = sigma(cuenta)

            kprob(cuenta) = kprob(cuenta + 1)
            tind(cuenta) = tind(cuenta + 1)
            kup(cuenta) = kup(cuenta + 1)
            d(cuenta) = d(cuenta + 1)
            h(cuenta) = h(cuenta + 1)
            s(cuenta) = s(cuenta + 1)
            ltime(cuenta) = ltime(cuenta + 1)
            sigma(cuenta) = sigma(cuenta + 1)

            kprob(cuenta + 1) = kprobcambio
            tind(cuenta + 1) = tindcambio
            kup(cuenta + 1) = kcambio
            d(cuenta + 1) = dcambio
            h(cuenta + 1) = hcambio
            s(cuenta + 1) = scambio
            ltime(cuenta + 1) = ltimecambio
            sigma(cuenta + 1) = sigmacambio

        End If

    Next

    revisa = revisa + 1

Loop

sumaprob = 0

For i = 1 To n
```

```
                kprob(i) = sumaprob + kprob(i)
                sumaprob = kprob(i)



            Next
            ' inicia algoritmo de recocido simulado

            'genera solucion inicial

            For i = 1 To n

                kactual(i) = kup(i) '1 'Int(1 + kup(i) * Rnd())

            Next

            setupcost = 0
            holdingcost = 0

            For i = 1 To n
                setupcost = setupcost + s(i)
                holdingcost = holdingcost + d(i) * h(i)

            Next

            'emplea como solucion inical la solución determinista
calculada por goyal

            tsup = System.Math.Sqrt((2 * (smayor + setupcost) /
holdingcost))
            t = tsup
            z = 1.64
            costofinal = 0
            costovecino = 0
            costocooling = 0
            criterio = 0
            frio = 0
            factor = 0.93
            temperatura = 35
            lengt = 500


            suma1 = 0
            suma2 = 0

            For i = 1 To n

                suma1 = suma1 + s(i) / kactual(i)
                suma2 = suma2 + (t * d(i) * h(i) * kactual(i)) / 2 +
sigma(i) * z * h(i) * System.Math.Sqrt(t * kactual(i) + ltime(i))

            Next


            setupcost = (smayor + suma1) / t
            holdingcost = suma2
```

```
tc = setupcost + holdingcost


costocooling = tc
For i = 1 To n
    kcooling(i) = kactual(i)

Next

Do Until criterio = 10000 Or temperatura < 0.01
    m = 0


    Do Until m = lengt

        'selecciona una k para perturbarse

        naleatorio = Rnd()
        sumaprob = 0
        i = 0

        Do Until naleatorio <= sumaprob Or i = n
            i = i + 1
            sumaprob = kprob(i)
            If naleatorio <= sumaprob Then
                l = i
            End If

            If i = n Then
                l = n
            End If

        Loop


        Select Case kcooling(l)

            Case Is = kup(l)

                perturba = 0.25


            Case Is = 1

                perturba = 0.75


            Case Else

                perturba = Rnd()



        End Select


        If perturba >= 0.5 Then
```

```
                    For y = 1 To l - 1
                        kvecina(y) = kcooling(y)
                    Next

                    kvecina(l) = kcooling(l) + 1


                    For y = l + 1 To n
                        kvecina(y) = kcooling(y)

                    Next

                Else


                    For y = 1 To l - 1
                        kvecina(y) = kcooling(y)
                    Next
                    kvecina(l) = kcooling(l) - 1


                    For y = l + 1 To n
                        kvecina(y) = kcooling(y)

                    Next


                End If


                sumavecina1 = 0
                sumavecina2 = 0


                For i = 1 To n


                    sumavecina1 = sumavecina1 + s(i) / kvecina(i)
                    sumavecina2 = sumavecina2 + t * d(i) * h(i) *
kvecina(i) / 2 + sigma(i) * z * h(i) * System.Math.Sqrt(t * kvecina(i) +
ltime(i))


                Next

                setupcostvecino = (smayor + sumavecina1) / t
                holdingcostvecino = sumavecina2
                costovecino = setupcostvecino + holdingcostvecino

                'seleccion del resultado

                If costovecino <= costocooling Then

                    costocooling = costovecino

                    For i = 1 To n
```

```
                        kcooling(i) = kvecina(i)

                    Next


                Else

                    'criterio de boltzman

                    aleatorio = Rnd()

                    boltz = System.Math.Exp(-(costovecino -
costocooling) / temperatura)

                    If boltz >= aleatorio Then

                        costocooling = costovecino

                        For i = 1 To n

                            kcooling(i) = kvecina(i)

                        Next

                    Else

                        costocooling = costocooling

                        For i = 1 To n

                            kcooling(i) = kcooling(i)

                        Next

                    End If

                End If

                m = m + 1

            Loop
            alfa = 1 / 0.5
            beta = 1 / (smallest * 0.5)
            g = beta - alfa
            t1 = beta - 0.618034 * g
            t2 = alfa + 0.618034 * g
            tc1 = 0
            tc2 = 0
            setupcost = 0
            holdingcost = 0
            epsilon = 100

            Do Until (epsilon < 0.00005)

                suma1t1 = 0
```

```
                    suma2t1 = 0

                    For i = 1 To n

                        suma1t1 = suma1t1 + s(i) / kcooling(i)
                        suma2t1 = suma2t1 + (d(i) * h(i) * kcooling(i)) /
(2 * t1) + sigma(i) * z * h(i) * System.Math.Sqrt(kcooling(i) * (1 / t1)
+ ltime(i))

                    Next i

                    setupcost1 = (smayor + suma1t1) * t1
                    holdingcost1 = suma2t1
                    tc1 = setupcost1 + holdingcost1

                    suma1t2 = 0
                    suma2t2 = 0

                    For i = 1 To n

                        suma1t2 = suma1t2 + s(i) / kcooling(i)
                        suma2t2 = suma2t2 + (d(i) * h(i) * kcooling(i)) /
(2 * t2) + sigma(i) * z * h(i) * System.Math.Sqrt(kcooling(i) * (1 / t2)
+ ltime(i))

                    Next i

                    setupcost2 = (smayor + suma1t2) * t2
                    holdingcost2 = suma2t2
                    tc2 = setupcost2 + holdingcost2

                    'criterio de evaluacion de la funcion

                    If tc1 < tc2 Then

                        beta = t2
                        t2 = t1
                        tcdorada = tc1
                        tdorada = t1

                    Else
                        alfa = t1
                        t1 = t2
                        tcdorada = tc2
                        tdorada = t2

                    End If
                    g = beta - alfa
                    t1 = beta - 0.618034 * g
                    t2 = alfa + 0.618034 * g

                    epsilon = System.Math.Abs(tc1 - tc2)

                Loop

                If criterio = 0 Then
```

```
        t = 1 / tdorada

        For i = 1 To n
            kactual(i) = kcooling(i)

        Next
        costoactual = tcdorada

        frio = 0

    Else

        If costoactual <= tcdorada Then

            frio = frio + 1
        Else

            t = 1 / tdorada
            costoactual = tcdorada
            For i = 1 To n

                kactual(i) = kcooling(i)

            Next

            frio = 0

        End If

    End If


    criterio = criterio + 1

    temperatura = factor * temperatura

Loop
```

Apéndice 2

Diseño experimental y resultados obtenidos de acuerdo al orden en que se realizaron las pruebas.

	Variable			
Exp.	**A:c0**	**B:alfa**	**C:N**	**% problemasI**
1	100	0.9	10	97.2
2	100	0.9	10	97.32
3	50	0.9	1	95.84
4	50	0.95	10	97.36
5	50	0.9	10	97.16
6	50	0.9	10	97.32
7	100	0.9	1	95.8
8	100	0.9	1	95.92
9	100	0.95	1	96.44
10	50	0.95	10	97.36
11	50	0.9	1	95.56
12	50	0.95	1	96.4
13	75	0.93	5.5	97.2
14	100	0.95	10	97.52
15	100	0.95	1	96.52
16	50	0.95	1	96.32
17	100	0.95	10	97.36

Bibliografía

1. Aarts, E; Host, J. Simulated Annealing and Boltzman Machines. Wiley, New York. 1990.
2. Atkins, D; Iyogun, P. Periodic Versus "can-order" Policies for Coordinated Multi-item Inventory Systems. Management Science 1988; 34; No. 6; pp. 791-796.
3. Balintfy, Joseph. On a basic class of multi-item inventory problems. Management Science 1964; 10; No. 2; 287 - 296.
4. Barr, Richard; Golden, Bruce L; Kelly, James; Resende Mauricio G.C; Stewart, William R. JR. Designing and reporting on computational experiments with metaheuristics. Journal of Heuristics 1995; 1; pp. 9-32.
5. Bazaraa, Mokhtar; Sherali, Hanif; Shetty, C.M. Nonlinear programming. Wiley Interscience, New Jersey. 2006.
6. Cormen, Thomas; Leiserson, Charles; Rivest, Ronald; Stein Clifford. Introduction to algorithms. MIT Press, USA. 2001.
7. De-los-Cobos, Sergio G. La Búsqueda Tabú y sus aplicaciones. Tesis Doctorado en Ingeniería; Facultad de Ingeniería UNAM 1994.
8. Díaz, Adenso; Glover, Fred; Ghaziri, M; González, J. L; Laguna, Manuel; Moscazo, Pablo; Tseng, Fan. Optimización heurística y redes neuronales. Ed. Paraninfo, Madrid. 1996.
9. Donaldson, W.A. An equation for the optimal value of T, the inventory replenishment review period when demand is normal. J. of Op. Res. Soc 1984; 35; No. 2; pp. 137-139.
10. Eynan, A; Kropp, D. Periodic Review and Joint Replenishment in Stochastic Demand Environments. IIE Transactions 1998; 30; No. 11; pp. 1025-1033.
11. Eynan, A, Kropp, D. Effective and Simple EOQ-like Solutions for Stochastic Demand Periodic Review Systems. European Journal of Operational Research 2007; 180; No 31; pp. 1135 - 1143.
12. Floudas, C. Deterministic Global Optimization. Springer-Verlag, New York. 2005.
13. Fouskakis, Dimitris; Draper, David. Stochastic Optimization: A Review. International Statistical Review 2002; 70; No. 3; pp. 315-349.

14. Federgruen, A; Groenevelt, H; Tijms, H.C. Coordinated replenishment in a multi-item inventory system with compound Poisson demands. Management Science 1984; 30; No. 3. pp. 344 – 357.
15. Fung, R Y K; Ma, X; Lau, H C W. (T, S) Policy for Coordinated Inventory Replenishment Systems under Compound Poisson Demand. Production, Planning and Control 2001; 12; No. 6; pp. 575-583.
16. Glover, Fred; Kochenberger, Gary A. Handbook of metaheuristics. Kluwer, New York. 2003.
17. Goyal, S. K. Determination of Optimum Packaging Frequency for Items Jointly Replenished. Management Science 1974; No. 21; pp. 436-443.
18. Goyal, S K;Satir, AT. Joint Replenishment Inventory Control: Deterministic and Stochastic Models. European Journal of Operational Research 1989; 38; No 1; pp. 2- 13.
19. Goldstein, Larry; Waterman, Michael. Neighborhood size in the simulated annealing algorithm. American Journal of Mathematical and Management Sciences 1988; 8; No. 3 y 4; pp. 409 – 423.
20. Gutiérrez, Miguel A. La técnica de Recocido Simulado y sus aplicaciones. Tesis de Doctorado en Ingeniería; Facultad de Ingeniería, UNAM 1991.
21. Hadley, G; Within, T M: Analysis of Inventory Systems. Prentice Hall, New Jersey. 1963.
22. Hoos, H; Stützle, T. Stochastic Local Search: Foundations and Applications. Elsevier, San Francisco. 2005.
23. Ignall, Edward. Optimal continuous review policies for two product inventory systems with joint setup costs. Management Science 1969; 15; No. 5. pp. 278 – 283.
24. Johnson, D, Aragon, C, McGeoch, L, Schevon, C. Optimization by Simulated Annealing: an Experimental Evaluation; Part I, Graph Partitioning. Operations Research 1989; 37; No. 6; pp. 865-892.
25. Khouja, M; Goyal, S. A Review of the Joint Replenishment Problem Literature: 1989-2005. European Journal of Operational Research 2008; 186; No. 1; pp. 1 - 16.
26. Kirkpatrick, S; Gellat, C D; Vecci, M.P. Optimization by Simulated Annealing. Science 1983; No. 220; pp. 671-680.

27. Laarhoven, Peter; Aarts, Emile; Lenstra Jan. Job Shop Scheduling by Simulated Annealing. Operations Research 1992; 40; No. 1; pp. 113-125.
28. Montgomery, Douglas C. Design and analysis of experiments. John Wiley and Sons, New York. 2000.
29. Myers, Raymond H; Montgomery, Douglas C. Response surface methodology: process and product optimization using designed experiments. John Wiley and Sons, New York. 2002.
30. Papadimitriou, Christos H; Steiglitz, Kenneth. Combinatorial optimization. Algorithms and complexity. Dover Publications, New Jersey. 1998.
31. Pantumsinchai, P. A Comparison of Three Joint Ordering Inventory Policies. Decision Sciences 1992; No. 23; pp. 111-127.
32. Pearl, Judea. Heuristics: intelligent search strategies for computer problem solving. Addison-Wesley. Boston. 1994.
33. Rao, Uday. Properties of periodic review (R, T) inventory control policy for stationary, stochastic demand. Manufacturing and Service Operations Management 2003; 5; No. 1; pp. 37 - 53.
34. Robb, DJ; Silver EA. Inventory management with periodic ordering and minimum order quantities. Journal of Operational Research Society 1998; 49; No. 10. pp. 1085 – 1994.
35. Sani, B; Kingsman B.G. Selecting the Best Periodic Inventory Control and Demand Forecasting Methods for Low Demand Items. Journal of the Operational Research Society 1997; 48; No. 7; pp. 700-713.
36. Shultz, H; Johansen, S.G. Can-order policies for coordinated inventory replenishment with Erlang distributed times between ordering. European Journal of Operational Research 1999; 113; pp. 30-41.
37. Silver, E; Peterson, R. Decision Systems for Inventory Management and Production Planning. John Wiley and Sons, New York. 1985.
38. Tagaras, Gorge; Vlachos, Dimitrios. A periodic review system with emergency replenishments. Management Science 2001; 47; No. 3. pp. 4125 – 429.
39. Viswanathan, S. Periodic Review (s,S) Policies for Joint Replenishment Inventory Systems. Management Science 1997; 43; No. 10; pp. 1447 – 1454.

40. Zheng, Y.S; Federgruen, A. Finding optimal (s, S) policies is as simple as evaluating a single policy. Operations Research 1999; 39; No. 4. pp 654 – 665.