UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES
ACATLÁN

ALGORITMOS GENÉTICOS APLICADOS A LA TEORÍA DE INVENTARIOS: MODELO DE CANTIDAD ECONÓMICA DE PEDIDO (CASO PRÁCTICO)

T E S I N A

QUE PARA OBTENER EL GRADO DE :
LICENCIADO EN MATEMÁTICAS APLICADAS Y COMPUTACIÓN

PRESENTA:
OMAR ILICH ORTIZ HERNÁNDEZ

ASESOR: DR. LUIS ALEJANDRO TAVERA PÉREZ

SANTA CRUZ ACATLÁN, MÉXICO MAYO DEL 2006

ÍNDICE

RESUMEN

Este trabajo consiste en una aplicación de los Algoritmos Genéticos a la Teoría de Inventarios Determinísticos, particularmente al Modelo de Cantidad Económica de Pedido (EOQ – Economic Order Quantity). Nuestro objetivo es la obtención de uno o más puntos que se acerquen al o a los óptimos para un problema de optimización. Esto es una alternativa práctica, ya que en las aplicaciones reales no siempre la obtención del óptimo es lo mejor y tampoco se puede alcanzar éste en nuestro problema. Esto nos lleva a plantear la alternativa de obtener puntos cercanos al óptimo para tener otras opciones de elección.

Palabras clave: Algoritmos Genéticos, Modelo de Cantidad Económica de Pedido, inventario, óptimo.

ABSTRACT

This job is based upon an application of Genetic Algorithms to Deterministic Inventory Theory, particularly to the model of Economic Order Quantity (EOQ). Our goal is to obtain one or more points closer to the optimum for a particular problem. This is a practical alternative, indeed in real applications not always is the best to get the optimum and also we can't get it in our problem. This take us to plan this alternative to obtain points closer to the optimum for us to have another options.

Key words: Genetic Algorithms, Economic Order Quantity, inventory, optimum.

TEMA

Algoritmos Genéticos aplicados a la Teoría de Inventarios: Modelo de Cantidad Económica de Pedido (caso práctico)

OBJETIVO

Aplicación de los Algoritmos Genéticos en la Teoría de Inventarios para resolver el problema del Modelo de Cantidad Económica de Pedido como una opción para encontrar subóptimos aceptables para esta metodología.

INTRODUCCIÓN

Los Algoritmos Genéticos nacen de una abstracción de la naturaleza. Son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acuerdo con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin. Por imitación de este proceso, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas.

Ya que el problema que nos concierne de inventarios es un problema de optimización de una función continua, el Algoritmo Genético es una opción adecuada. El Algoritmo Genético arroja resultados enteros y esto es sumamente conveniente para nuestro problema, ya que en la producción de artículos no se pueden tener fracciones.

El problema que nos concierne se basa en la teoría de inventarios en su parte determinista de una empresa X que produce un bien no divisible Y. Se aborda el problema con un Algoritmo Genético programado en Visual Fox Pro y se realizaron 50 corridas, eligiéndose las 10 mejores, se compararon con el óptimo obtenido mediante la fórmula del Modelo de Cantidad Económica de Pedido.

En el primer capítulo se presenta un breve introducción a las bases biológicas e históricas de los Algoritmos Genéticos, se habla de genética. Se presentan algunos teoremas acerca de los Algoritmos Genéticos y describimos los pasos para crear un Algoritmo Genético.

En el segundo capítulo hablamos de la importancia que tiene para una empresa tener un buen sistema de inventarios y definimos las características de los sistema de inventario. Al final desarrollamos el Modelo de Cantidad Económica de Pedido que es el modelo que nos interesa.

El tercer capítulo trata acerca de la aplicación del Algoritmo Genético al Modelo de Cantidad Económica de Pedido, mostramos el resultado obtenido mediante la fórmula y explicamos el algoritmo programado en Visual Fox Pro junto con el código.

El cuarto capítulo explica los resultados obtenidos con nuestro Algoritmo Genético y las conclusiones a las que llegamos al aplicar esta metodología.

Definición del problema

Los inventarios son materiales y suministros que una empresa o institución posee, ya sea para vender o para abastecer al proceso productivo. Todas las empresas o instituciones precisan de inventarios, constituyendo una parte importante del activo total de las mismas. En consecuencia el objetivo de una empresa, en cuanto a su política de inventarios, será lograr el balance justo entre los beneficios derivados de mantenerlos y los costos que ellos originan.

Los Algoritmos Genéticos son métodos sistemáticos para la resolución de problemas de búsqueda y optimización que aplican a estos los mismos métodos de la evolución biológica: selección basada en la población, reproducción sexual y mutación.

Debemos aplicar la metodología de los Algoritmos Genéticos a un modelo específico de inventarios llamado Modelo de Cantidad Económica de Pedido.

Una de las herramientas que se utilizan para determinar el monto óptimo de pedido para un artículo de inventario es el modelo de la cantidad económica de pedido (CEP). Tiene en cuenta los diferentes costos financieros y de operación y determina el monto de pedido que minimice los costos de inventario de la empresa. El modelo de cantidad económica de pedido se basa en tres supuestos fundamentales, el primero es que la empresa conoce cuál es la utilización anual de los artículos que se encuentran en el inventario, segundo que la frecuencia con la cual la empresa utiliza el inventario no varía con el tiempo y por último que los pedidos que se colocan para reemplazar las existencias de inventario se reciben en el momento exacto en que los inventarios se agotan.

Hipótesis

Se pretende demostrar que los Algoritmos Genéticos son aplicables a la Teoría de Inventarios para resolver el problema del Modelo de Cantidad Económica de Pedido como una opción para encontrar subóptimos aceptables.

Justificación

Abordamos el Modelo de Cantidad Económica de Pedido ya que parte de una serie de supuestos fuertes (limitamos mucho las variables del problema), los cuales se van suavizando a medida que se avanza en la teoría, sin embargo sus aplicaciones y utilidad son importantes y los desarrollos posteriores que ha permitido, lo hacen un punto de referencia obligado en todos los campos donde se hable de inventarios.

Utilizamos Algoritmos Genéticos ya que su espacio de búsqueda esta delimitado dentro de un cierto rango, se puede definir una función de aptitud que nos indica qué tan buena o mala es una respuesta y las soluciones se pueden codificar de una forma que resulta relativamente fácil de implementar en la computadora.

El Algoritmo Genético nos sirve para encontrar subóptimos aceptables para el Modelo de Cantidad Económica de Pedido.

Capítulo 1

Algoritmos genéticos

1.1 Darwin y la selección natural

Darwin se enfocó en gran medida en el proceso de selección natural y adaptación. Se dio cuenta que muchos organismos tienen más descendencia que llega a la madurez, aún así el número de individuos en determinadas especies no varía notablemente a corto plazo. Notó que la mortalidad juega un papel importante en la naturaleza. Darwin concluyó que la naturaleza mata a los insuficientemente adaptados y lo llamó el proceso de la selección natural.

Algunos de los postulados de Darwin son seguidos en los Algoritmos Genéticos y tienen gran importancia:

- Algunas variantes serán más afortunadas que otras en la lucha por la existencia. La progenie de tales variantes tenderá, por tanto, a ser la más numerosa y a su vez los pocos descendientes de las variantes menos afortunadas en su adaptación al medio estarán en desventaja, por lo que pueden desaparecer.
- Las variaciones en cuestión son heredadas.
- Por lo que las sucesivas generaciones de las variantes mejor dotadas tenderán a constituir una forma modificada.

Para el neodarwinismo se comprende la selección no como la elección del mejor, sino como la eliminación del peor, del poco viable, de los errores de la naturaleza. No se trata ya de la persistencia del más apto, sino de la supresión del menos apto.

La selección ordinaria conserva y acumula las variaciones útiles, las fija y mantiene las especies como son. Por otro lado la herencia de pequeñas variaciones, escogidas por la selección, muestra un tipo de evolución continua. La selección innovadora llega a constituir formas modificadas en los organismos de tal manera que los caracteres son nuevos, útiles y únicos en ese medio ambiente.

1.2 Genética

El estudio de la herencia y sus variaciones se llama genética. Herencia es el fenómeno mediante el cual las características de los padres son transmitidas a sus descendientes. Las grandes corrientes prevalecientes para desentrañar el proceso por el cual los seres vivos son capaces de reproducir organismos semejantes a ellos, con fidelidad casi invariable de una generación a otra, han hecho de la genética una de las áreas más apasionantes de la biología moderna y quizás de toda la ciencia actual.

La genética es el estudio científico de cómo se transmiten los caracteres físicos, bioquímicos y de comportamiento de padres a hijos. Este término fue acuñado en 1906 por el biólogo británico William Bateson. Los genetistas determinan los mecanismos hereditarios por los que los descendientes de organismos que se reproducen de forma sexual no se asemejan con exactitud a sus padres, y las diferencias y similitudes entre padres e hijos que se reproducen de generación en generación según determinados patrones. La investigación de estos últimos ha dado lugar a algunos de los descubrimientos más importantes de la biología moderna.

En los humanos y otros organismos, las células contienen materiales especiales que tienen la información de la herencia. Estas estructuras son llamadas cromosomas. Los genes son estructuras específicas que son contenidas en los cromosomas.

Los humanos normalmente poseemos 46 cromosomas ordenados en pares; de estos 46, 23 vienen del huevo de la madre y 23 del esperma del padre. El proceso mediante el cual es reducido de 46 a 23 es llamado meiosis. En este proceso, los miembros de cada par de cromosomas se acercan, desenrollan e intercambian material genético en un proceso llamado cruce. Sólo una de las cadenas cruzadas es transmitida el huevo o esperma, resultando sólo 23 cromosomas. El cruce no produce nuevos genes, pero produce nuevas combinaciones de material genético. El número de combinaciones posibles es enorme. Cada humano es capaz de producir más de 19 mil millones de huevos o esperma genéticamente diferente.

Ocasionalmente la copia del material genético resulta en algunas imperfecciones, llamadas mutaciones. La velocidad de la mutación es lenta, pero llega a generar diversidad genética adicional en las poblaciones. El efecto a corto plazo en una población no se nota, pero la mutación puede llegar a cambiar una población en períodos largos de tiempo.

El subcampo de la genética que está más relacionado con los Algoritmos Genéticos es la genética de las poblaciones. Los investigadores en este campo estudian las relaciones en la

evolución, la combinación del material genético en y entre las especies y métodos de adaptación al ambiente.

1.3 Bases de los Algoritmos Genéticos

A finales de los 50's y principios de los 60's, muchos biólogos comenzaron a experimentar con simulaciones computacionales de Sistemas Genéticos. A. S. Frauer, en particular, realizó experimentos que tenían una gran similitud con los Algoritmos Genéticos. John Holland fue quien comenzó a desarrollar ideas acerca de sistemas de adaptación durante los 60's. Daba cursos de Sistemas de Adaptación en la Universidad de Michigan y comenzó a publicar muchos escritos en la materia. Gradualmente concretó sus pensamientos, que culminaron en el libro "Adaptation in Natural and Artificial Systems" (Adaptación en Sistemas Naturales y Artificiales), publicado en 1975.

Los Algoritmos Genéticos son rápidos y flexibles. Son aplicables a muy distintas clases de problemas, se utilizan para resolver problemas de búsqueda y optimización. Se basan en procesos de la naturaleza, fundamentalmente genéticos, como su nombre lo indica. Un campo importante para los Algoritmos Genéticos es la optimización de problemas. Si se quiere maximizar o minimizar algo, entonces tenemos un problema de optimización y éste método es apropiado para su resolución. Los Algoritmos Genéticos en algunas ocasiones no encuentran el óptimo, pero la mayoría de las veces encuentran soluciones cercanas al óptimo. Si tenemos un problema que consta de un gran espacio de búsqueda – esto es, un gran conjunto de posibles soluciones– entonces es un buen candidato para esta metodología. Es adecuado porque si el problema es extremadamente complejo, entonces se puede tomar mucho tiempo para el calculo de las soluciones, adicional a esto buscar en todo el espacio llevaría bastante tiempo. El Algoritmo Genético hace más eficiente y rápida esta búsqueda.

Los Algoritmos Genéticos trabajan con una población inicial de individuos que se representa de distintas maneras dependiendo del planteamiento del problema. Cada individuo es una solución factible al problema. Para adaptar el problema de modo que el Algoritmo Genético lo pueda usar se tiene que representar como una cadena de valores, generalmente se usa la representación binaria con ceros y unos. La cadena representará posibles soluciones al problema que estamos resolviendo. Para usar los Algoritmos Genéticos no se tiene que saber mucho acerca del problema, simplemente debemos tener alguna forma de evaluar una cadena dada. Esto es, saber como se acerca al óptimo.

Los Algoritmos Genéticos son procedimientos computacionales modelados en base a la genética y evolución, están diseñados para hacer búsquedas de soluciones atractivas a problemas largos y complejos. Una de las partes más importantes de los algoritmos genéticos es la representación del problema y la forma de plantear la función de aptitud.

Los cromosomas se representan por arreglos y cada elemento del arreglo vendría representando un gen. El cromosoma representa una posible solución al problema y con la función de aptitud se le da un valor al cromosoma, dependiendo del acercamiento a la solución buscada. Los AG se basan mayormente en el proceso de unión de los cromosomas del padre con los de la madre. En este proceso se intercambia material genético y algunas veces se producen mutaciones. Estas son dos operaciones básicas en los AG conocidas como cruce y mutación. Como los cromosomas se pueden decodificar para revelar ciertas características de un organismo, una cadena de bits se puede decodificar para revelar una posible solución del problema.

El propósito básico de los operadores genéticos es transformar la población de posibles soluciones y extender la búsqueda por otros puntos del espacio. También tienen relación con la evolución planteada por Darwin y la selección natural, en la cual los organismos mejor adaptados al medio ambiente son los que sobreviven y eventualmente dominarán al resto de la población.

En el Algoritmo Genético Simple los individuos se seleccionan aleatoriamente y se comparan, pasan únicamente los más aptos a las siguientes generaciones. La mutación consiste en elegir un cromosoma aleatoriamente con cierta probabilidad, y cambiar un gen de forma aleatoria por otro. Para la reproducción, se seleccionan aleatoriamente dos padres y de igual forma (aleatoriamente), se decide el lugar donde se va a efectuar el cruce. El cruce se hace aleatoriamente sin tomar en cuenta el grado de adaptación. Esto es, en el número de elemento elegido, se corta el cromosoma y se intercambia con el otro padre, creándose así dos nuevos cromosomas. Para crear la nueva generación, se seleccionan aleatoriamente dos individuos, y el que tenga el mejor valor de acuerdo al esperado pasa a la siguiente generación. Vamos generando nuevas soluciones que van evolucionando hacia el óptimo. Como hemos visto la aleatoriedad es una característica distintiva de los Algoritmos Genéticos.

Un criterio para evaluar el Algoritmo Genético es el de la convergencia, el cual consiste en evaluar la población a lo largo de las sucesivas generaciones y cuando aproximadamente un 95% de los cromosomas son iguales quiere decir que convergió, aproximándose al óptimo de la solución. Si se busca una convergencia rápida se tendrá que pagar el precio. Esto es, que probablemente no se tenga una buena aproximación al óptimo, ya que limitamos el espacio de búsqueda. Y si se busca una muy buena aproximación, con una probabilidad alta tendremos una convergencia lenta, ya que trabajaremos con un espacio de búsqueda muy amplio. Tenemos que buscar un punto medio entre una convergencia rápida y una buena aproximación.

El cruce es un proceso poderoso que extiende la búsqueda en muchas direcciones para que nos podamos acercar más al óptimo. Podríamos imaginar una campana colgada de una cuerda en un cuarto oscuro. Tenemos una cantidad de pelotas. El objetivo es golpear la campana. Comenzamos a lanzar las pelotas indiscriminadamente. Abruptamente escuchamos un ligero repiqueteo de la campana. Esto quiere decir que golpeamos la cuerda que la sostiene en algún punto. En este momento ya tenemos una noción de la dirección que debemos seguir para golpear la campana.

Como consideramos el desempeño en la selección de las generaciones sucesivas, nos pueden interesar ciertos elementos de los cromosomas y querríamos que pasaran a las siguientes generaciones para que eventualmente dominen la población, esto se hace con el cruce. Así que esta búsqueda considera indirectamente el éxito o fracaso de ciertos elementos y cromosomas, tiene algo de memoria y aprende de los éxitos.

1.4 Teoremas acerca de los Algoritmos Genéticos

ESQUEMAS

Se llama esquema a un caso base que acepta variaciones.

3 símbolos: 0, 1 y *

El asterisco (*) es un símbolo que puede variar entre 0 ó 1.

Consideremos la siguiente cadena binaria y dos esquemas:

Cadena A: 11100

Esquema 1: 1***0

Esquema 2: **10*

El esquema 1 es un modelo que requiere un 1 en la primera posición y un 0 en la 5ta. posición, pero no importan los valores que se tomen en las otras posiciones. El esquema 2 requiere un 1 en la tercera posición y un 0 en la cuarta, pero no importan los valores que tome antes o después de estas posiciones. Ambos esquemas coinciden con la cadena A.

Digamos que tenemos una población que consiste en 100 cadenas binarias de tamaño 5. Suponemos que el esquema 1 coincide con 25 miembros de la población y el esquema 2 coincide con 22 de ellos. El esquema 1 y 2 pueden estar en una misma cadena. Asumimos que el desempeño promedio de las 100 cadenas de la población es 20.

25 cadenas contienen 1***0

22 cadenas contienen **10*

También asumamos que el desempeño promedio de aquellas cadenas que contienen el esquema 2 es solamente ¾ más alto que el desempeño promedio de las 100 cadenas (20 * 1.75 = 35). Y el desempeño del esquema 1 es 1½ veces mayor que el promedio de las 100 cadenas (20 * 2.50 = 50).

e.g. Promedio desempeño 100 cadenas: 20

Desempeño esquema 1: 50 → 1½ veces mayor

Desempeño esquema 2: 35 → ¾ veces mayor

El teorema de Holland dice:

- El esquema 1 recibirá un incremento exponencial de la prioridad (de ser seleccionado) en las generaciones siguientes (Figura 1.1).
- El esquema 2 recibirá un decremento exponencial de la prioridad (de ser seleccionado) en las siguientes generaciones (Figura 1.2).

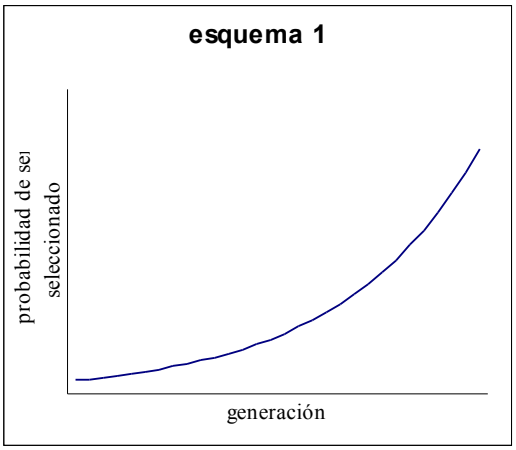

Figura 1.1

Desempeño promedio: 50

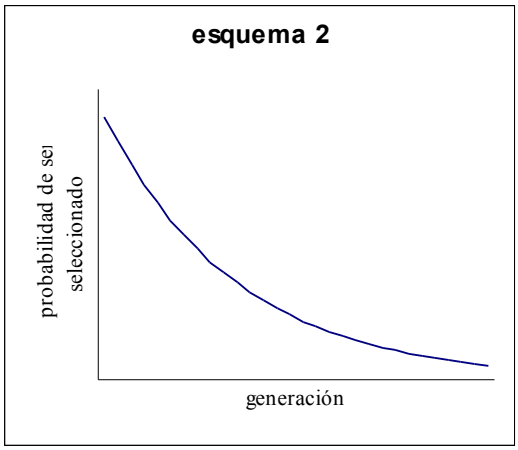

Figura 1.2

Desempeño promedio: 35

Las cadenas con mejor desempeño tienen mayor probabilidad de ser seleccionadas en las siguientes generaciones.

En este caso los individuos con la mejor adaptación se toman en cuenta con más frecuencia para ser los padres de las siguientes generaciones.

CRUCE

El cruce es el intercambio de información entre cadenas para lograr variedad en la población.

Si reexaminamos los esquemas 1 y 2 nos daremos cuenta que es más fácil que el cruce afecte al esquema 1 ya que están más separados los elementos de dicho esquema, no siendo así con el esquema 2. Los teoremas de Holland concluyen lo siguiente: El esquema que tiene una distancia menor entre sus elementos, que tiene pocos elementos y que tiene un mejor promedio de adaptación recibirá un número exponencialmente mayor de oportunidades de pasar a las siguientes generaciones. Estos pequeños esquemas componen bloques de gran importancia en la búsqueda del Algoritmo

Genético. El Algoritmo Genético toma en cuenta estos bloques y los une para encontrar soluciones cercanas al óptimo.

Holland demostró que cuando el Algoritmo Genético trabaja con una población de n estructuras (elementos que conforman la población), el número de esquemas que son procesados es de aproximadamente n^3. Esto hace la búsqueda del Algoritmo Genético muy eficiente. Holland llamó a esto paralelismo implícito.

Una ventaja de los Algoritmos Genéticos es que sólo una pequeña parte del código para la computadora cambia de un problema a otro. Principalmente el cambio que se hace es en la función de aptitud. Los principios que seguimos en el cruce y la mutación no cambian. Debemos estar conscientes de qué es lo que realmente queremos optimizar, si no planteamos adecuadamente la función de aptitud (función que nos ayuda a evaluar una cadena dada, esto es saber como se acerca al óptimo; se le da un valor al cromosoma dependiendo del acercamiento a la solución buscada), nos podríamos encontrar con resultados muy distantes de los esperados. La función de aptitud está directamente relacionada con la cadena. Para cada posible cadena debe haber un valor asignado por la función de aptitud que represente el desempeño de dicha cadena con respecto a la solución que se está buscando.

Recordemos que el AG trata de encontrar la cadena óptima y a través de todo este proceso crea la mejor cadena con límites previamente establecidos. Para saber si nuestro AG se está acercando al óptimo, debemos correr el programa varias veces y analizar los resultados que nos está arrojando. Si tenemos resultados que varían en gran medida, puede ser que el problema tenga varios puntos óptimos o que le debemos hacer algún ajuste a nuestro programa.

Cuando un AG converge en una solución inaceptable se puede deber a diversos factores: esta convergiendo muy rápido y no tiene oportunidad de explorar todo el espacio de búsqueda, la representación del problema esta mal planteada o la función de aptitud no es la adecuada. Puede ser mala suerte, esto es cuando al crear la población inicial únicamente hay malas soluciones y sigue con ellas a lo largo de todo el proceso, en este caso debemos correr el programa más veces.

1.5 Pasos para la creación de un Algoritmo Genético

1. Representación del problema

   Se debe representar el problema de forma que el Algoritmo Genético pueda trabajar con él, se representa con cadenas de símbolos que en su estructura imitan a los cromosomas. Generalmente se trabaja con código binario (ceros y unos). Nuestra cadena representará una solución potencial a nuestro problema.

   Las cadenas, con las sucesivas iteraciones, se irán modificando. Para la modificación de las cadenas se usan operadores biológicos, éstos se basan en procesos llevados acabo en ciertos organismos. Por ejemplo, dos cadenas se pueden cruzar y producir descendencia que contenga las mejores características de cada una de ellas. De acuerdo a la noción de supervivencia del más fuerte, el individuo de la población con mejor desempeño eventualmente dominará.

2. Inicializar la población

   No se tienen reglas para el tamaño de la población. Usualmente ésta se compone de 100 a 200 elementos. Cuanto mayor sea la población, tendremos mayor diversidad, pero también requerimos mayor tiempo para que la computadora trabaje con el Algoritmo Genético. Una vez que hemos escogido el tamaño de la población, aleatoriamente generamos las cadenas. La aleatoriedad es básica en el trabajo con Algoritmos Genéticos.

   Para ciertos problemas es necesario rechazar algunas cadenas generadas, ya que se deben ajustar a ciertos parámetros. En estos casos ponemos ciertas condiciones en el algoritmo para que acepte únicamente aquellas que sean válidas.

   Eventualmente la población convergirá. Convergencia significa que la mayoría o todas las cadenas son iguales. Podemos experimentar con el tamaño de la población para evaluar cual es el mejor para nuestro problema.

3. Calcular el desempeño

   Ya que generamos la población, debemos calcular el desempeño de cada cadena con respecto al óptimo que queremos encontrar. Esto lo llevamos a cabo planteando una función de aptitud. Lo que hace esta función es evaluar la cadena bajo ciertos parámetros relacionados con el problema y le otorga un valor. El valor está directamente relacionado con su proximidad al

punto óptimo. Se utiliza con regularidad la función de aptitud en el proceso del Algoritmo Genético.

4. Selección

Al hacer la selección, lo que tratamos de hacer es que sobrevivan o que pasen las cadenas con mejor desempeño a la siguiente generación, para hacer esto utilizamos los valores que nos arrojó la función de aptitud. Se pueden tener diferentes criterios de selección dependiendo del problema.

Si utilizamos métodos muy estrictos, podemos hacer que el algoritmo converja rápidamente. Generalmente a costa de no haber explorado adecuadamente el espacio y quizá sin encontrar cadenas más adecuadas. Esto se conoce como el problema de la convergencia prematura.

5. Cruce

Podríamos decir que el cruce es la parte que le da fuerza al Algoritmo Genético. Es la que logra que las cadenas mejor adaptadas tengan descendencia con sus características y logra extender la búsqueda hacia otros puntos y con el paso del tiempo hace que mejoren las soluciones más adecuadas.

El cruce se lleva a cabo en 4 pasos: Primero se seleccionan 2 padres aleatoriamente. Segundo, aleatoriamente determinamos si se va a llevar a cabo el cruce. Generalmente utilizamos una probabilidad del 60% para la realización del cruce[1]. Tercero, aleatoriamente seleccionamos el lugar de la cadena donde se va a efectuar el cruce. Cuarto, cortamos cada cadena en el punto seleccionado y las intercambiamos. Esto es, si tenemos la cadena A y la cadena B, aleatoriamente se decide hacer el cruce en el 6to. elemento e intercambiamos los elementos a partir de éste. Así la cadena A se queda con sus primeros 6 elementos y el resto son los elementos de la cadena B a partir del séptimo y viceversa.

Algunos estudios muestran que el 60% de probabilidad para el cruce es adecuado en algunas ocasiones, pero esto puede variar dependiendo del problema que estemos resolviendo.

El cruce se basa en la experimentación con distintas combinaciones, a la larga nos quedaremos con las mejores. De cierto modo es un método exhaustivo que va reduciendo los parámetros de búsqueda. Este proceso está altamente relacionado

[1] La probablidad de 0.6 para el cruce y 0.001 para la mutación viene de los trabajos de Kenneth De Jong, uno de los estudiantes de doctorado de John Holland. De Jong hizo una serie de pruebas en problemas de optimización con diferentes parámetros a las que llamó: "De Jong test suite" (grupo de prueba de De Jong) y en estas pruebas notó que eran las probabilidades más convenientes.

con la selección ya que gracias a ésta se pueden hacer combinaciones con las cadenas mejor adaptadas. Esto hace muy eficiente a nuestro Algoritmo Genético.

6. Mutación

La mutación introduce desviaciones aleatorias en nuestra búsqueda del óptimo, amplía nuestro espacio de búsqueda. El proceso es el siguiente: Definimos la probabilidad de mutación, si se decide que se va a efectuar; aleatoriamente seleccionamos una cadena de nuestro espacio. De nuevo, aleatoriamente decidimos que elemento vamos a mutar y lo cambiamos por otro. En algunas variantes de Algoritmo Genético la probabilidad de mutación va cambiando a lo largo de las iteraciones.

7. Convergencia

La convergencia generalmente se mide usando el concepto de "bias", que se define como una medida de evaluación de la población. El bias tiene valores entre el 50 y 100 por ciento. Por ejemplo si tenemos 100 miembros y 50 de ellos tienen un 0 en el sexto elemento y los otros 50 un 1, el promedio es 50-50 y el bias para el elemento es 50%. Si el promedio es 70-30 (70 ceros y 30 unos), entonces el bias para el elemento es 70%. El bias es el número mayor del promedio. De igual forma el bias sería 70% si fuera al revés: 30-70 (30 ceros y 70 unos). También tenemos bias para las cadenas. El bias de la cadena es el promedio de los bias de los elementos, este bias será nuestra medida para la convergencia del algoritmo.

La convergencia es muy influenciada por el proceso de selección, mientras la selección sea más estricta, la convergencia será más rápida. Con el paso de las generaciones la convergencia irá teniendo valores más altos.

La medida para la convergencia generalmente es un bias de 95%. No se puede llegar a un bias del 100% por la mutación. La mutación hace que una cadena sea diferente a las demás, en otras palabras cambia un gen. Después de calcular el bias de la población llegamos a un punto crítico. Podemos tomar 2 decisiones: Si el bias es satisfactorio, en ese momento podemos detener el algoritmo, si no hemos alcanzado el bias esperado, entonces debemos volver al paso 3 y repetir hasta el 7 hasta que converja nuestro algoritmo.

Si el algoritmo se tarda mucho tiempo en convergir, entonces podemos aplicar otro criterio que es fijar un número máximo de iteraciones. Por ejemplo, detener nuestro algoritmo cuando llegue a las 20 iteraciones. Si la convergencia es muy lenta, debemos revisar el algoritmo y ver si se necesitan ajustes en alguno de los pasos.

Variaciones en el cruce

La operación de cruce que hemos mencionado, es llamada cruce en un punto porque solamente se corta la cadena en un punto. También se puede usar el cruce en dos puntos, el cual se realiza en dos partes de la cadena aleatoriamente. Intercambiamos la parte de en medio y lo demás queda igual:

Cadenas originales:

01 : **101** : 10
11 : **010** : 01

Cruce en dos puntos:

01 : **010** : 10
11 : **101** : 01

Podemos usar el cruce en varios puntos, esto es, cortar las cadenas en varias partes e intercambiar las subcadenas. Pero quizá nuestra búsqueda se torne un tanto caótica debido a la heterogeneidad de nuestra población. Claro que esto también dependerá del tipo de problema que estemos resolviendo, la población, etc.

Selección

La selección es un paso fundamental en nuestra metodología, ya que implica el paso de una generación a otra, podemos hacer este proceso muy selectivo o más relajado.

David Goldberg y Kalayanmoy Deb en su estudio "Un Análisis Comparativo en Esquemas de Selección Usados en los Algoritmos Genéticos" (A Comparative Analysis of Selection Schemes Used in Genetic Algorithms) analizaron varios procedimientos de selección e identificaron sus ventajas y desventajas. Veremos algunos de ellos y resumiremos sus aportaciones:

Para los casos que vamos a mostrar la aptitud se obtiene como una representación decimal de la cadena binaria, esto es: tenemos la cadena binaria 1000, en decimal es igual a: $(1 * 2^3) + (0 * 2^2) + (0 * 2^1) + (0 * 2^0) = 8$.

1. Proporción de la ruleta (Roulette wheel proportionate). Este proceso utiliza una selección basada en la adaptación. Para este método sumamos la aptitud de toda nuestra población y le damos porcentajes de acuerdo a los elementos que tengan la misma aptitud. El porcentaje se basa en la

suma de las aptitudes de los elementos iguales. Aleatoriamente generamos un número y vemos en que intervalo cae. Esto lo hacemos las veces que queramos el tamaño de la población. Con este método tendremos en la segunda generación individuos con aptitud alta, ya que es proporcional.

e.g.

Cromosoma / Gen	1	2	3	4
1	1	0	0	0
2	0	1	1	0
3	1	0	0	0
4	1	0	0	0
5	0	0	0	1
6	1	1	1	1
7	1	0	0	1
8	0	1	1	0
9	1	0	1	1
10	0	1	0	1

Aptitud
8
6
8
8
1
15
9
6
11
5

Suma de la aptitud de toda la población: 77

Cromosoma			
1	0	0	0
0	1	1	0
0	0	0	1
1	1	1	1
1	0	0	1
1	0	1	1
0	1	0	1

Suma Aptitud
24
12
1
15
9
11
5

Porcentaje
31.17%
15.58%
1.30%
19.48%
11.69%
14.29%
6.49%

Intervalos	
0	31.17%
31.18%	46.75%
46.76%	48.05%
48.06%	67.53%
67.54%	79.22%
79.23%	93.51%
93.52%	100.00%

Porcentaje aleatorio	*Cromosoma*			
43.07%	0	1	1	0
25.48%	1	0	0	0
48.68%	1	1	1	1
99.67%	0	1	0	1
55.10%	1	1	1	1
68.10%	1	0	0	1
9.66%	1	0	0	0
72.15%	1	0	0	1
1.69%	1	0	0	0
19.17%	1	0	0	0

Aptitud
6
8
15
5
15
9
8
9
8
8

Suma de la aptitud de toda la población: 91

2. Selección del cociente estocástico (Stochastic remainder selection). El primer paso es sacar un porcentaje basado en la suma de las aptitudes de las cadenas iguales. Esto es, si tenemos dos cadenas iguales, su aptitud va a ser la misma. Se agrupan las cadenas iguales y se suman sus aptitudes.

   e.g.

   Si la cadena 3 y la cadena 8 son 1011 y su aptitud es 11, al agruparlas y sumar su aptitud nos queda: cadena 1011 con aptitud igual a 22. Para sacar el porcentaje que le corresponde a dicha cadena primero obtenemos la suma de todas las aptitudes, que en este caso sería 85 (ver tablas debajo), entonces calculamos el porcentaje haciendo a 85 el 100%, por lo tanto 22 sería el 25.88% ((22 * 100) / 85). La suma de los porcentajes calculados debe ser igual a 100.

   Con base en la parte entera del porcentaje, ese número de cadenas pasará a la siguiente generación (siempre proporcional al total). Esto es, recordamos que el porcentaje para nuestra cadena 1011 es de 25.88%, entonces pasarían 25 cadenas si nuestra población contuviera 100 de ellas, pero como no es así dividimos 25.88 / 10 = 2.59, así la parte proporcional que le corresponde es de 2 ya que nuestra población consta de 10 cadenas y redondeando la parte decimal nuestro residuo queda 0.59 (2 + 0.59 = 2.59).

   De los residuos de nuestros porcentajes, sacamos una proporción para que pasen a la siguiente generación las cadenas que nos faltaron para completar la población requerida. De tal modo, sumamos todos los residuos que quedan, para nuestro ejemplo la suma es igual a 4 y esto lo vamos a tomar como el 100%, por lo tanto para nuestra cadena 1011 cuyo residuo es 0.59 el nuevo porcentaje que le corresponde es 14.75% ((0.59 * 100) / 4).

   Ya que tenemos los porcentajes usamos la Proporción de la Ruleta, para esto establecemos los intervalos, generamos los números aleatorios, revisamos en que intervalo caen y anexamos las cadenas que nos faltan.

   Este método es más estricto y únicamente tiene un poco de aleatoriedad al final.

   e.g.

Cromosoma / Gen	*1*	*2*	*3*	*4*
1	1	1	1	1
2	0	1	0	0
3	1	0	1	1

Aptitud
15
4
11

4	1	0	1	0	10
5	0	0	1	1	3
6	0	1	1	1	7
7	1	1	0	0	12
8	1	0	1	1	11
9	0	0	1	1	3
10	1	0	0	1	9

Suma de la aptitud de toda la población: 85

Cromosoma				*Suma Aptitud*	*Porcentaje*
1	1	1	1	15	17.65%
0	1	0	0	4	4.71%
1	0	1	1	22	25.88%
1	0	1	0	10	11.76%
0	0	1	1	6	7.06%
0	1	1	1	7	8.24%
1	1	0	1	12	14.12%
1	0	0	1	9	10.59%

% Cadenas que pasarán	*# Cadenas que pasarán*	*% que queda*	*% de prob. con relación al 100%*
1.76%	1	0.76	19.00
0.47%	0	0.47	11.75
2.59%	2	0.59	14.75
1.18%	1	0.18	4.50
0.71%	0	0.71	17.75
0.82%	0	0.82	20.50
1.41%	1	0.41	10.25
1.06%	1	0.06	1.50

Intervalos	
0	19.00
19.01	30.75
30.76	45.50
45.51	50.00
50.01	67.75
67.76	88.25
88.26	98.50
98.51	100.00

Porcentaje aleatorio	*Cromosoma*				*Aptitud*
72.28%	0	1	1	1	7
2.33%	1	1	1	1	15
2.46%	1	1	1	1	15
48.95%	1	0	1	0	10

Cromosoma / Gen	*1*	*2*	*3*	*4*	*Aptitud*
1	1	1	1	1	15
2	1	0	1	1	11
3	1	0	1	1	11
4	1	0	1	0	10
5	1	1	0	1	13
6	1	0	0	1	9
7	0	1	1	1	7
8	1	1	1	1	15
9	1	1	1	1	15

10	1	0	1	0	10

Suma de la aptitud de toda la población: 116

3. Selección estocástica universal (Stochastic universal selection). Para empezar, igual que en los otros métodos, debemos dar porcentajes a las cadenas iguales basados en la suma de las aptitudes (ver Selección del Cociente Estocástico). Establecemos intervalos con dichos porcentajes. Esto es, si nuestras 3 primeras cadenas tienen los porcentajes: 16.90%, 16.90% y 9.86% respectivamente, nuestros intervalos serán: 0 – 16.90%, el siguiente 16.91% - 33.80% (16.90 + 16.90 = 33.80) y el último irá de 33.81% a 43.66% (33.80 + 9.86 = 43.66). El intervalo 0 – 16.90% corresponde a nuestra primera cadena, el intervalo 16.91% - 33.80% corresponde a nuestra segunda cadena, el intervalo 33.81% - 43.66% corresponde a nuestra tercera cadena; y así sucesivamente hasta llegar al 100%.

   Después dividimos 100 (el porcentaje total) entre el número original de individuos del que consta nuestra población, a este número le llamaremos x. Para nuestro ejemplo, nuestra población consta de 10 individuos, entonces x = 100 / 10, por lo tanto x = 10. Cuando tengamos el valor de x, generamos un número aleatorio entre 0 y x. En este caso nuestro número aleatorio será 3.

   A partir de nuestro número aleatorio vamos incrementando en x nuestros siguientes números tal que, el primero será 3 (nuestro número aleatorio), el segundo 13 (3 + x = 3 + 10 = 13), el tercero 23 (13 + x = 13 + 10 = 23) y así sucesivamente. Ya teniendo los números, buscamos cada uno dentro de los intervalos que habíamos establecido previamente, así el 3 se encuentra dentro del intervalo 0 – 16.90 que corresponde a nuestra primera cadena, el 13 de igual modo, el 23 se encuentra en el intervalo 16.91 – 33.80 que corresponde a nuestra segunda cadena y así con todos los números. En la parte del ejemplo que explicamos anteriormente tendríamos 3 cadenas que pasarían a la siguiente generación: 2 correspondientes a la primera cadena y 1 correspondiente a la segunda.

Cromosoma / Gen	1	2	3	4
1	1	1	0	0
2	0	1	1	0
3	0	1	1	0
4	0	1	1	1
5	1	1	0	1
6	0	0	0	0
7	1	0	0	1

Aptitud
12
6
6
7
13
0
9

8	0	1	0	0	4
9	0	1	0	1	5
10	1	0	0	1	9

Suma de la aptitud de toda la población: 71

Cromosoma				*Aptitud*	*Porcentaje*	*Intervalos*	
1	1	0	0	12	16.90%	0	16.90%
0	1	1	0	12	16.90%	16.91%	33.80%
0	1	1	1	7	9.86%	33.81%	43.66%
1	1	0	1	13	18.31%	43.67%	61.97%
0	0	0	0	0	0.00%	-	-
1	0	0	1	18	25.35%	61.98%	87.32%
0	1	0	0	4	5.63%	87.33%	92.96%
0	1	0	1	5	7.04%	92.97%	100.00%

$x = 100 / 10 = 10$

Número aleatorio: 3

Números	*Cromosoma*				*Aptitud*
3%	1	1	0	0	12
13%	1	1	0	0	12
23%	0	1	1	0	6
33%	0	1	1	0	6
43%	0	1	1	1	7
53%	1	1	0	1	13
63%	1	0	0	1	9
73%	1	0	0	1	9
83%	1	0	0	1	9
93%	0	1	0	1	5

Suma de la aptitud de toda la población: 88

4. Selección Genitor (Genitor selection).Ésta selección la desarrolló Whitley en 1989, está basada en la función de aptitud. Se le dan valores a las cadenas basándose en la adaptación, así los peores individuos son reemplazados por los mejores.

5. Selección por rango con ruleta (Ranking selection with roulette wheel). Comenzamos ordenando la población de acuerdo a su adaptación. Luego, una función de asignación le da a cada cadena una probabilidad de pasar a la siguiente generación, proporcional a la adaptación. Se simula una ruleta con las posiciones determinadas por la función de asignación. La siguiente generación de n elementos se construye dando a la ruleta n vueltas. Con esta selección suponemos que pasen los mejores elementos sin forzar la selección de ninguno.

6. Selección por torneo (Tournament selection). El torneo es como una lucha entre miembros para determinar cual pasará a la siguiente generación. Puede involucrar a 2 o más individuos al mismo tiempo. Generalmente, se seleccionan 2 individuos aleatoriamente de la población y se comparan los valores de adaptación. El que tenga mejor desempeño pasará a la siguiente generación. También podemos usar porcentajes, por ejemplo, que el 70% de las veces pase el que tenga mejor desempeño. O simplemente puede ser al azar. Repetimos este proceso hasta que la población para la siguiente generación se complete.

Debe haber un balance entre la exploración y la explotación. Nos referimos a la explotación como el aprovechamiento de la información que hemos obtenido en anteriores generaciones. Supongamos que hay cadenas con ciertas características y hemos notado que éstas nos llevan a una mejor aproximación al óptimo. Y lo que hace el Algoritmo Genético es explotar las características de estas cadenas para convergir a la mejor solución. Pero lo que puede pasar en el afán del Algoritmo Genético por convergir rápidamente es, que no explore otras partes del espacio que pueden ofrecer mejores aproximaciones. Es común el problema entre exploración y explotación.

El método de selección que se elija es fundamental en el balance entre exploración y explotación. Un método de selección más fuerte sin dejar tanto al azar tenderá a basarse más en la explotación, ya que nos fijamos en nuestra población con mejores resultados para que pase a la siguiente generación. En cambio un método basado un poco más en la aleatoriedad tiene oportunidad de explorar más el espacio de búsqueda.

Los Algoritmos Genéticos tienen la ventaja de ofrecer un mecanismo adaptativo de resolución de problemas, de forma que aunque el problema cambie, éste se pueda seguir resolviendo. “Resolver un problema cambiante”, llevado al límite, es igual a “resolver cualquier problema”. Según este planteamiento podríamos pensar en diseñar un esquema común que facilite al programa aprender en cualquier entorno del problema.

Capítulo II

Inventarios

2.1 Motivos para mantener un inventario

Es claro que hay un potencial enorme para mejorar la eficiencia de la economía si se controlan los inventarios de manera inteligente. Las empresas que poseen métodos científicos de control de inventarios tienen una ventaja competitiva apreciable en el mercado. El inventario es un mal necesario, demasiado poco causa costosas interrupciones y el exceso da por resultado un capital ocioso.

Una parte importante en la decisión del tipo de inventario a elegir es la demanda (por tiempo de unidad) de un artículo que puede ser determinística (conocida con cierto grado de certidumbre) o probabilística (descrita mediante una distribución de probabilidad).

Los motivos para mantener un inventario son:

Economías de escala. Si se produce una línea de artículos semejante y cada corrida requiere reconfigurar la línea de producción y recalibrar las máquinas, generalmente se invierte bastante tiempo y dinero en preparar la producción. Entonces podría ser más económico producir una cantidad grande de artículos en cada corrida y almacenarlos para el uso posterior. Esto nos permitiría amortizar los costos fijos de preparación, repartidos en una cantidad mayor de unidades (requerimos que el costo de preparación sea una constante fija).

Incertidumbre. La incertidumbre generalmente es un motivo importante para almacenar inventarios. La incertidumbre de la demanda externa es primordial. Si no disponemos de un artículo que un cliente demanda, probablemente lo consiga en otro lugar, y lo peor es que quizá perdamos a ese cliente. Si tenemos un inventario adecuado, podremos responder a las demandas de los clientes.

Otro tipo de incertidumbre es el tiempo de demora. Éste se define como el tiempo que tarda en llegar un pedido a partir del momento en que se coloca. Aún cuando tengamos certeza de la demanda a futuro, necesitamos tener existencias de amortiguamiento para tener un flujo uniforme de producción cuando el tiempo de demora sea incierto.

Otra fuente de incertidumbre es el abastecimiento. Si de alguna forma el abastecimiento de materia prima del que dependemos sufre algún colapso, nos puede afectar enormemente, incluso puede llegar a poner en riesgo la operación.

Especulación. Si es posible que aumente el valor de un artículo básico para la empresa, es mejor comprar y almacenar grandes cantidades de este artículo para no pagar precios más elevados a futuro. Ya que el aumento en el costo de la materia prima conllevará al aumento en el precio de los bienes producidos y quizá la reducción en el consumo de éstos por parte de los clientes.

Transporte. Los inventarios relacionados con el transporte son los llamados inventarios en tránsito. Debido a que el tiempo que se tardan en llegar los productos que necesitamos es muy largo, se requiere almacenar éstos en puntos estratégicos del trayecto (del lugar donde se producen a donde los requerimos) y en esos lugares mantenemos el inventario en tránsito para tener un abastecimiento uniforme. Otra opción es establecer el abastecimiento localmente.

Suavizamiento. Pueden surgir cambios en la demanda del producto. Tener inventarios previendo etapas en las que la demanda sea alta evita las interrupciones debidas a cambios en la producción.

Logística. Podemos tener problemas en las compras, producción o distribución, por esto es necesario mantener inventarios. En la logística de la manufactura, necesitamos los inventarios para tener continuidad en éste proceso (de manufactura).

Costos de control. Éste es el costo de mantener el sistema de control de inventarios. Un sistema que tiene una mayor cantidad de inventario no necesita el mismo nivel de control que otro que mantiene el nivel de éste al mínimo posible.

2.2 Características de los sistemas de inventario

1. *Demanda*. Generalmente el análisis que se hace del comportamiento y las características de la demanda es muy importante para determinar la complejidad del modelo de inventario:

   a. Constante o variable. Para los modelos simples de inventario se toma una demanda constante. El modelo de cantidad económica de pedido (EOQ, Economic Order Quantity) y sus variaciones se fundamentan en estas bases. Para modelos más complejos en los cuales no tenemos certeza acerca de la cantidad de artículos demandados a futuro, ésta (la demanda) se toma variable. En este caso podemos tomar modelos de inventario probabilísticos en los que la demanda se describe con una distribución de probabilidad.

   b. Conocida o desconocida. La demanda puede ser constante y aleatoria al mismo tiempo. La mayoría de los modelos de demanda estocástica consideran la tasa promedio de demanda como constante.

2. *Tiempo de demora*. Si nuestro proveedor es externo, el tiempo de demora es el intervalo de tiempo transcurrido desde el instante en que se realiza el pedido, hasta el momento de llegada. El tiempo de demora se debe expresar en las mismas unidades que el tiempo de demanda (intervalo de tiempo que tardan nuestros clientes en pedir el producto que realizamos o el material que necesitamos).

3. *Tiempo de revisión*. A veces podemos conocer el nivel exacto de inventario en cada momento, para esto podríamos suponer que nuestro sistema registra todas las transacciones del inventario al instante. En este caso, le damos el nombre de "revisión continua". En caso contrario se llama "revisión periódica", esto es, conocemos el nivel de inventario en puntos discretos del tiempo, en otras palabras, es cuando hacemos un levantamiento físico de existencias cada determinado tiempo.

4. *Exceso de demanda*. Otra cuestión importante, es la reacción del sistema hacia el exceso de demanda, esto es, cuando no podemos satisfacer la demanda al instante con las existencias que tenemos. Las opciones que más se utilizan son:

   - Se corre y se acumula el exceso de demanda. Se agrega a nuestro inventario la cantidad de artículos que no se tenían en existencia. Ése ahora será nuestro inventario óptimo, por si algún día se presenta el mismo caso podamos satisfacer la demanda.
   - Se pierde el exceso de demanda. Se satisface fuera del sistema con una producción extra o el cliente adquiere

los artículos en otro lugar. En este caso nuestro inventario futuro es el mismo que el actual, vamos a mantener la misma cantidad de inventario.

- Acumulación parcial. Se agrega a nuestro inventario futuro solo una parte de los artículos que no se tenían en existencia. Lo demás no se toma en cuenta.
- Impaciencia del cliente. Si no satisfacemos la demanda en un tiempo determinado el pedido es cancelado. Nuestro inventario futuro no sufre modificaciones.

5. *Inventario cambiante*. Cuando los inventarios sufren cambios a lo largo del tiempo, éstos pueden dañar la utilidad. Algunos artículos pueden ser afectados por la obsolescencia o en otros casos, pueden ser perecederos, y allí podemos tener perdidas significativas.

COSTOS RELEVANTES

Nuestro interés se centra en la optimización del sistema de inventarios, por lo tanto debemos encontrar un método adecuado de optimización. La mayoría de los modelos de inventario utilizados se basan en la minimización del costo como criterio de optimización. Otro criterio que podemos usar es la maximización de la ganancia. La mayoría de los costos de inventario se pueden ubicar en alguna de las siguientes clases: costo de mantener inventario, costo de pedido o costo de penalización.

Costo de mantener el inventario

Es llamado también costo de almacén o costo de inventario. Es el total de los costos en relación a la cantidad física de inventario en cualquier momento. Algunos de los componentes del costo de mantener el inventario son:

- El costo del espacio físico para el almacenaje de los productos.
- Impuestos y seguros.
- Daños y obsolescencia.
- Costo de oportunidad de una inversión alternativa.

Generalmente el último punto es el de mayor valor para el cálculo de los costos de mantener el inventario. Debemos invertir dinero para la compra o producción de inventario, y si disminuye nuestro inventario, esto nos arroja mayor capital. Con este dinero podríamos invertir en mejoras para la empresa.

Costo de pedido

El costo de pedido se basa en la cantidad de inventario producido o pedido. Generalmente el costo de pedido consta de dos partes: una fija y una variable. El costo fijo, K, no depende del tamaño del pedido, mientras no sea cero. El costo variable, c, depende del número de unidades. Otro nombre que recibe K es el de costo de preparación y a c también se le llama costo proporcional de pedido. Definiendo C(x) como el costo de pedir (o producir) x unidades, tenemos:

$$C(x)=\begin{cases} 0 & si\ \ x=0 \\ K+cx & si\ \ x>0 \end{cases}$$

Al estimar el costo de preparación únicamente debemos incluir los costos relevantes a la decisión actual de pedir. Algunos de los costos comprendidos en K son: los gastos de contabilidad relacionados con el pedido, costos fijos requeridos por el vendedor que no dependen del tamaño del pedido, costos de generación y recepción del pedido y costos de su manejo.

Costo de penalización

Este costo recibe diversos nombres como: costo de escasez, costo de faltante o costo de agotamiento de inventario, y se debe a la falta de existencias necesarias a la mano para satisfacer la demanda al momento que se presenta. Este costo se interpreta de forma variable dependiendo de la demanda, es decir, si el exceso de demanda se acumula, el costo de penalización incluirá todos los costos contables y/o de demora en que pudiera incurrirse, y si el exceso de demanda se pierde, el costo incluye la ganancia perdida que pudiera haberse acumulado con la venta. En ambos casos, se debe incluir el costo de "pérdida por buena voluntad", que es un indicador de la falta de satisfacción del cliente; es difícil determinar este indicador.

Suponemos que el costo de penalización se carga con base en las unidades. Así, cada vez que no podemos satisfacer la demanda de inmediato, incurrimos en un costo de penalización independiente del tiempo que se tarde en abastecer la demanda. Otro método para llevar un control de los faltantes es cargar el costo de penalización en una base por unidad y por unidad de tiempo. Usamos este método cuando debemos tener en cuenta el tiempo que permanece un pedido no surtido en los libros.

2.3 El modelo de cantidad económica de pedido

Este modelo es el más sencillo de los modelos de inventario. Describe el importante compromiso entre los costos fijos y los costos de mantener el inventario, y es la base para el análisis de sistemas más complejos.

Para utilizar el modelo básico, hacemos las siguientes suposiciones:

1. Conocemos la tasa de demanda y es una constante igual a λ unidades por unidad de tiempo (podemos determinar la unidad de tiempo de acuerdo a nuestras necesidades: día, semana, mes, año, etc. Se deben expresar todas las variables relevantes en la misma unidad de tiempo).

2. No se permiten faltantes.

3. No hay tiempo de demora de pedido (al momento que se coloca el pedido es abastecido).

4. Los costos incluyen:
   a. Costo de preparación K por pedido colocado.
   b. Costo proporcional de pedido c por unidad pedida.
   c. Costo de mantener el inventario h por unidad mantenida por unidad de tiempo.

Suponemos que en el tiempo cero el nivel de inventario también es cero. Como establece el segundo punto, no se permiten faltantes, así que cuando el tiempo es cero, hacemos un nuevo pedido. El tamaño del pedido lo representamos por Q. Así cuando el tiempo t es igual a cero, el inventario disponible aumentará en forma instantánea de cero a Q.

Al observar el siguiente tiempo, nos damos cuenta que se pueden reducir los costos de mantener el inventario si esperamos a que éste llegue a cero antes de ordenar un nuevo pedido. Los cambios descritos en los niveles de inventario a través del tiempo son representados por la siguiente gráfica (Figura 2.1).

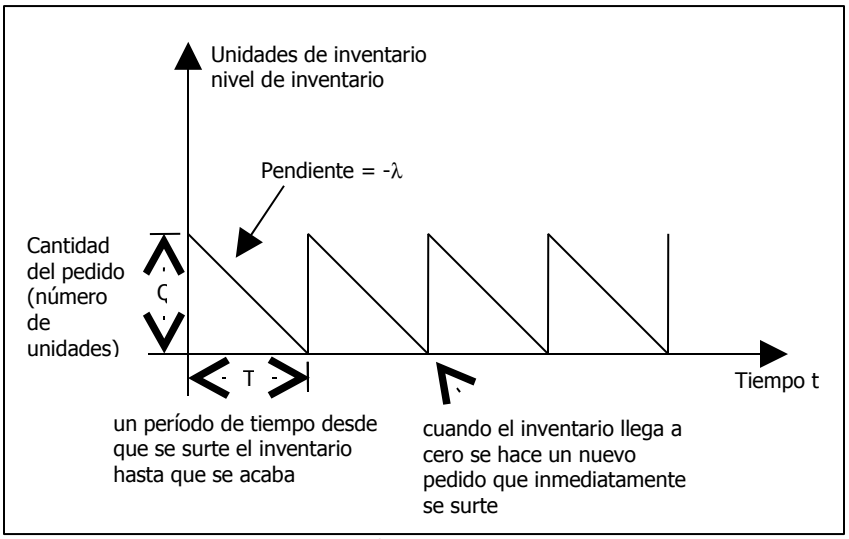

Figura 2.1

Ahora, representaremos mediante una ecuación el costo promedio anual en función del tamaño de lote *Q*. En cada ciclo, tenemos el costo de preparación por pedido colocado más el costo proporcional de pedido por unidad pedida por el tamaño del pedido, esto es: $C(Q) = K + cQ$. Para obtener el costo de pedido por unidad de tiempo se divide $C(Q)$ entre la longitud del ciclo T. Como se consumen Q unidades cada ciclo a una tasa λ, en consecuencia $T = Q/\lambda$. Otra forma de obtener esta igualdad es al observar la pendiente de la curva ($-\lambda$) de la Figura 2.1 que es igual a la relación $-Q/T$.

En cada ciclo el nivel de inventario decrece linealmente de Q a 0. Por lo tanto el nivel promedio en cada ciclo es de $Q/2$. Como todos los ciclos son iguales, el promedio se mantiene en $Q/2$. Entonces el costo anual promedio $G(Q)$ es:

$$G(Q) = \frac{K + cQ}{T} + \frac{hQ}{2}$$
$$= \frac{K + cQ}{Q/\lambda} + \frac{hQ}{2}$$

$G(Q)$ es el costo total por tiempo de unidad (costo promedio anual en función del tamaño de lote Q.

K es el costo de preparación asociado con la colocación de un pedido (pesos por pedido).

c es el costo proporcional de pedido por unidad pedida.

Q es el tamaño de pedido o de lote.

T es la longitud del ciclo de pedido.

h es el costo de almacenamiento (pesos por unidad de inventario por tiempo de unidad).

λ es la tasa, es decir el índice de demanda (unidades por tiempo de unidad).

$$G(Q) = \frac{K\lambda}{Q} + \lambda c + \frac{hQ}{2}.$$

Los tres términos que quedan al simplificar *G(Q)* son el costo anual de preparación, el costo anual de compra y el costo anual de mantener el inventario, respectivamente.

El valor óptimo de la cantidad *Q* del pedido se determina minimizando *G(Q)* respecto a *Q*. Suponemos que *Q* es continua. Una condición necesaria para encontrar el valor óptimo de *Q* es la derivada y examinando la forma de la curva *G(Q)* apreciamos que

$$G'(Q) = -K\lambda / Q^2 + h/2$$

y

$$G''(Q) = 2K\lambda / Q^3 > 0 \quad \text{para} \quad Q > 0$$

Como *G''(Q)* > 0, se sigue que *G(Q)* es una función convexa de *Q*.

El valor óptimo de *Q* se presenta cuando *G'(Q)* = 0. Esto es cierto cuando $Q^2 = 2K\lambda/h$, que da como resultado

$$Q^* = \sqrt{\frac{2K\lambda}{h}}.$$

Q^* es la cantidad económica de pedido (EOQ).

La política de inventario óptimo para el modelo propuesto es:

Pedido $Q^* = \sqrt{\frac{2K\lambda}{h}}$ unidades cada $T^* = \frac{Q^*}{\lambda}$ unidades de tiempo.

Incluso no es indispensable recibir el pedido al instante que se coloca. Podemos tomar un tiempo de entrega positivo, *L* entre el momento en el que se hace y se recibe el pedido, como lo muestra la Figura 2.2.

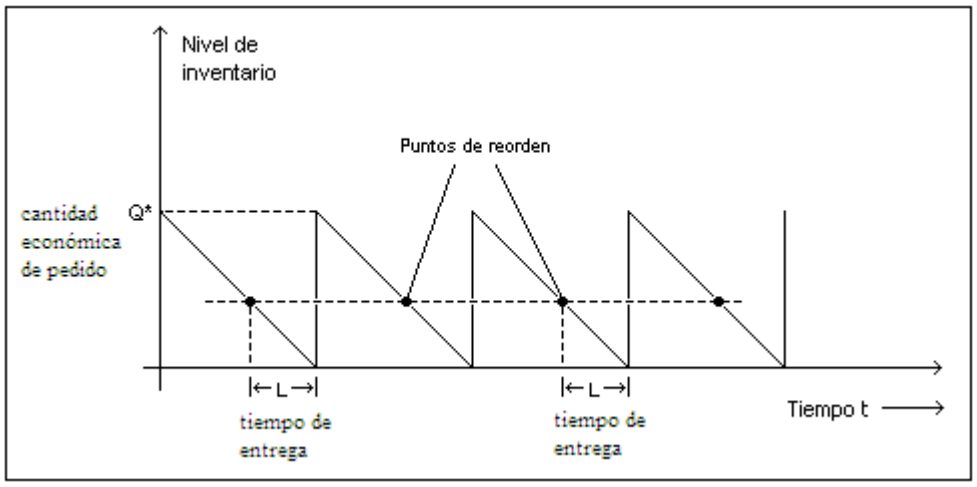

Figura 2.2

En este caso particular, el punto de reorden ocurre cuando el nivel de inventario desciende a $L\lambda$ unidades y suponemos que el tiempo de entrega L es menor que la duración del ciclo T^* lo que por lo general no ocurre. Para definir con mayor precisión tenemos que el tiempo de entrega efectivo es (cuando el tiempo de entrega excede la duración del ciclo, es decir, $L > T^*$).

$$L_e = L - nT^*$$

Cuando n es el entero más grande sin exceder $\frac{L}{T^*}$. Es así debido a que después de n ciclos de T^* cada uno, el inventario se comporta como si el intervalo entre hacer y recibir un pedido fuera L_e. Entonces, el punto del nuevo pedido se da en $L_e\lambda$ unidades, reescribiendo la política de inventario tenemos:

Ordenar la cantidad Q^* cuando el nivel de inventario descienda a $L_e\lambda$ unidades.

Ejemplo:

Un periódico ordena pliegos de papel cada lunes para cubrir la demanda semanal de 100 millares. El costo fijo por pedido es de $25000.^{00}$. Cuesta alrededor de $5.^{00}$ por millar, por semana, almacenar el papel. El tiempo entre colocar y recibir un pedido es de 12 semanas.

$K = 25000$
$Q = 100$
$T = 1$
$h = 5$
$L = 12$

$$\lambda = \frac{Q}{t} = \frac{100}{1} = 100$$

$$Q^* = \sqrt{\frac{2k\lambda}{h}} = \sqrt{\frac{2(25000)(100)}{5}} = 1000$$

$$T^* = \frac{Q^*}{\lambda} = \frac{1000}{100} = 10$$

n (entero más grande) $< \frac{L}{T^*}$

$$n < \frac{12}{10}$$

$$n < 1.2$$

$$\therefore\ n = 1$$

$$L_e = L - nT^*$$

$$= 12 - 1\ (10)$$

$$= 2$$

Ordenar la cantidad de 1000 millares cuando el nivel de inventario descienda a 200 millares.

Capítulo III

Aplicación

3.1 Aplicación de la metodología del Algoritmo Genético

Para nuestra prueba suponemos que alguna compañía tiene una demanda fija de 32767 artículos, que debe cubrir en un período de un año. Los datos son los siguientes:

- El costo de preparación por pedido colocado es de $6000.^{00}$.
- El costo de mantener el inventario es de $6.^{00}$ por unidad al año.

Tenemos entonces:

K = 6000
λ = 32767
h = 6

Utilizando la fórmula para la Cantidad Económica de Pedido (EOQ):

$$Q^* = \sqrt{\frac{2K\lambda}{h}}.$$

Se obtiene que hay que ordenar Q^* = 8095.307 unidades.

Q^* es la Cantidad Económica de Pedido, es una política de inventario óptima que implica un índice de demanda constante con un reabastecimiento instantáneo de pedidos y sin faltante. Hay que pedir Q^* unidades cada determinado tiempo.

1. Representación del problema.
   Se utilizaron cadenas de números binarios de tamaño 14 con un valor máximo en decimal de 16383 (el valor en decimal de la cadena binaria representa a Q^*).

2. Inicializar la población.
   El tamaño de la población fue de 30 individuos y se inicializaron con valores aleatorios.

3. Calcular el desempeño.
   El desempeño se calculó con la función de beneficio económico en un mercado de competencia pura:

$$\text{Beneficio} = (32767 * Q) - \left[(Q * 3) + \frac{32767 * 6000}{Q} \right]$$

4. Selección.
   Se realizó 30 veces (para crear la nueva población). Aleatoriamente se eligieron 2 cromosomas y se quedó con el que tuvo el mejor desempeño (Selección por Torneo entre 2 individuos).

5. Cruce.
   El porcentaje de población incluida en el cruce fue de 60%.

6. Mutación.
   En cada iteración se mutó un individuo.

7. Convergencia.
   Nuestro algoritmo se basó en un número máximo de iteraciones (determinado por el usuario). En este caso fueron 30 iteraciones.

Programa para el Algoritmo Genético en Visual Fox Pro

Tenemos el programa principal (master_inv.prg) donde desarrollamos los pasos del Algoritmo Genético y controlamos el número de iteraciones.

```
***  Programa:  master_inv.prg
***  Programa general para el Algoritmo Genético de Inventarios.


PUBLIC e_contgeneral, e_nopob, e_numiteraciones

e_nopob = 30

USE historico_inv

    FOR e_contblancos = 1 TO RECCOUNT ()

        GO 1

            DELETE
            PACK

    ENDFOR

CLOSE DATABASES ALL

e_contgeneral = 0

DO genera_pob_inv                  && Creación de la población.
DO aptitud_decimal_inv             && Calcula el valor decimal.
DO evaluacion_inv                  && Calcula el desempeño.

FOR e_contgeneral = 1 TO e_numiteraciones

    DO seleccion_inv               && Ejecuta la función de
selección.
    DO cruce_inv                   && Ejecuta el cruce.
    DO mutacion_inv                && Ejecuta la mutación.
    DO cambia_tabla_inv            && Actualiza las tablas de
población.
    DO aptitud_decimal_inv         && Calcula el valor decimal.
    DO evaluacion_inv              && Calcula el desempeño.

ENDFOR
```

Este programa genera los genes que compondrán cada cromosoma.

Tenemos un número determinado de cromosomas (en este caso 30). El cromosoma se compone de un número determinado de genes (en este caso 14).

A los genes se les asigna un valor aleatorio entre 0 y 1 mediante la instrucción MOD (INT ((RAND() * 10), 2).

```
***  Programa:  genera_pob_inv.prg
***  Programa que genera los cromosomas de la población.

USE poblacion_inv

RAND (-1)

FOR e_cromosoma = 1 TO RECCOUNT ()

    GO e_cromosoma

    FOR e_gen = 1 TO FCOUNT ()

            REPLACE (FIELD (e_gen)) WITH MOD (INT ((RAND () *
10)), 2)

    ENDFOR

ENDFOR
```

En cada cromosoma hacemos el cambio del número binario a número decimal.

En la tabla decimal_inv guardamos los valores decimales de los cromosomas.

```
***  Programa:  aptitud_decimal_inv.prg
***  Programa que cambia a decimal el número binario (función de
aptitud).

USE poblacion_inv

FOR e_cromo = 1 TO RECCOUNT ()

    USE poblacion_inv

        GO e_cromo

            e_binario_1 = (Gen_01 * 8192) + (Gen_02 * 4096) +
                (Gen_03 * 2048) + (Gen_04 * 1024) + (Gen_05 * 512)
```

```
            e_binario_2 = (Gen_06 * 256) + (Gen_07 * 128) +
                (Gen_08 * 64) + (Gen_09 * 32) + (Gen_10 * 16)
            e_binario_3 = (Gen_11 * 8) + (Gen_12 * 4) + (Gen_13 *
2) + Gen_14

            e_binario = e_binario_1 + e_binario_2 + e_binario_3

    USE decimal_inv

        GO e_cromo

            REPLACE num_decima WITH e_binario

ENDFOR
```

Calcula la función de beneficio económico en un mercado de competencia pura.

Para el valor de cada cromosoma en decimal hacemos: Si el número es igual a 0, le asignamos una ganancia de –3000000 para que tenga un valor muy bajo y sea difícil que pase a la siguiente generación), si no, la ganancia será igual a:

(32767 * 30) – [(número decimal * 3) + (32767 * 6000) / número decimal]

Busca la mejor ganancia de todas y se la asigna a mejor (definido como variable), y el número decimal se lo asigna a cron_mejor. Esto es para llevar un historial de la evolución de nuestro algoritmo a través de los mejores cromosomas.

```
***  Programa:  evaluacion_inv.prg
***  Programa que calcula la función de beneficio económico
***  en un mercado de competencia pura.


USE decimal_inv

e_mejor = 0

FOR e_cromos = 1 TO RECCOUNT ()

    USE decimal_inv

        GO e_cromos

            IF num_decima = 0

                ganancia = - 3000000
```

```
            ELSE

                resta = num_decima * 3 + 32767 * 6000 / num_decima

                ganancia = 32767 * 30 - resta

            ENDIF

            IF e_mejor < ganancia

                e_mejor = ganancia
                e_cromejor = num_decima

            ENDIF

    USE beneficio_inv

        GO e_cromos

            REPLACE beneficio WITH INT (ganancia)

ENDFOR

USE historico_inv

    APPEND BLANK

        REPLACE valoptimo WITH e_cromejor

USE
```

Elige aleatoriamente entre 2 cromosomas y se queda con el que tenga mayor desempeño.

Seleccionamos 2 cromosomas (e_crom1 y e_crom2) aleatoriamente entre nuestra población.

Obtenemos de la tabla beneficio_inv los valores de la Función de Beneficio Económico de nuestros cromosomas y los comparamos. Nos quedamos con el que tenga el valor más alto. Con el procedimiento llena_matriz, llenamos una matriz con los genes del cromosoma seleccionado e insertamos estos valores en la tabla pobdepurada_inv.

Esto lo hacemos tantas veces como el número de cromosomas que queremos (30).

*** Programa: seleccion_inv.prg

```
***  Aleatoriamente elige 2 cromosomas y se queda con el que tenga
el
***  mejor desempeño.

SET PROCEDURE TO procedimientos_inv

RAND (-1)

DIMENSION em_cambia (14)

USE beneficio_inv

FOR e_pob = 1 TO RECCOUNT ()

    USE beneficio_inv

    e_crom1 = INT (1 + RAND () * 1000 % RECCOUNT ())
    e_crom2 = INT (1 + RAND () * 1000 % RECCOUNT ())

    GO e_crom1

        e_val1 = beneficio

    GO e_crom2

        e_val2 = beneficio

    ***  Elige el mejor valor de los 2 cromosomas

    IF e_val1 > e_val2

        e_cromopasa = e_crom1

    ELSE

        e_cromopasa = e_crom2

    ENDIF

    ***

    USE poblacion_inv

    ***  Llena la matriz con los genes del cromosoma seleccionado

        GO e_cromopasa

            DO llena_matriz WITH em_cambia

    ***

    USE pobdepurada_inv
```

```
    ***  Llena el cromosoma de pobdepurada_inv.dbf con los valores
de la matriz

        e_lugarcruce = 1

        GO e_pob

            DO reemp_mat WITH em_cambia, e_lugarcruce

    ***

ENDFOR
```

Aquí hacemos el cruce aleatorio de dos cromosomas.
Usamos la tabla pobdepurada_inv.
Generamos un número aleatorio entre 0 y 1, si el número es menor que 0.5 realizamos el cruce, para este proceso, seleccionamos aleatoriamente 2 padres de la población. Con el procedimiento revisa_padres checamos si el padre y la madre son el mismo cromosoma, si es así, los cambia. Después determinamos el lugar donde se va a realizar el cruce (entre el gen 2 y el 13). Llenamos una matriz con los genes de un cromosoma y otra matriz con los genes del otro.
Con el procedimiento realiza_cruce, realizamos el cruce (cambiamos los genes de lugar). Y con el procedimiento reemp_mat insertamos los genes de los cromosomas cruzados en la tabla pobdepurada_inv.

```
***  Programa:  cruce_inv.prg
***  Cruce aleatorio de dos cromosomas.


SET PROCEDURE TO procedimientos_inv

RAND (-1)

FOR e_cont = 1 TO 18          && Porcentaje de cruce:  60%
población.

    e_probcruce = RAND ()

    IF e_probcruce < 0.5

        USE pobdepurada_inv

        e_padresrepetidos = 1
```

```
        e_padre = INT (1 + RAND() * 1000 % RECCOUNT ())
        e_madre = INT (1 + RAND() * 1000 % RECCOUNT ())

        DO revisa_padres WITH e_padresrepetidos, e_padre, e_madre
        *  Revisa si el padre y la madre son el mismo cromosoma.

        e_lugarcruce = INT (2 + RAND() * 1000 % (FCOUNT () - 2))
        *  El lugar del cruce está entre el gen 2 y el 13.

        DIMENSION em_padre (FCOUNT ())
        DIMENSION em_madre (FCOUNT ())

            GO e_padre

                DO llena_matriz WITH em_padre

            GO e_madre

                DO llena_matriz WITH em_madre

            DO realiza_cruce WITH e_lugarcruce, em_padre, em_madre
            *  Realiza el cruce, cambia los cromosomas de lugar.

            GO e_padre

                DO reemp_mat WITH em_padre, e_lugarcruce

            GO e_madre

                DO reemp_mat WITH em_madre, e_lugarcruce

        USE

    ENDIF

ENDFOR
```

Este programa muta un gen y realiza este procedimiento una vez.
Usa la tabla pobdepurada_inv.
Seleccionamos aleatoriamente el cromosoma y el gen que vamos a mutar. Si el gen tiene el valor 0 lo cambiamos a 1 y viceversa.

```
***  Programa:  mutacion_inv.prg
***  Programa que realiza la mutación de un gen
***  aleatoriamente (una vez).


SET PROCEDURE TO procedimientos_inv
```

```
RAND (-1)

USE pobdepurada_inv

DIMENSION em_cromosoma (FCOUNT ())

    e_cromosomamuta = INT (1 + RAND() * 1000 % RECCOUNT ())
    e_genmuta = INT (1 + RAND() * 1000 % FCOUNT())

    USE pobdepurada_inv

        GO e_cromosomamuta

            DO llena_matriz WITH em_cromosoma

            IF em_cromosoma (e_genmuta) = 1

                REPLACE (FIELD (e_genmuta)) WITH 0

            ELSE

                REPLACE (FIELD (e_genmuta)) WITH 1

            ENDIF

    USE
```

Este programa actualiza la tabla de población. Inserta la tabla pobdepurada_inv en poblacion_inv. Con el procedimiento llena_matriz, almacena los cromosomas de pobdepurada_inv en una matriz y con el procedimiento reemp_matriz inserta la matriz que tenemos en población_inv.

```
***  Programa:  cambia_tabla_inv
***  Programa que actualiza las tablas:  hace que la tabla
***  pobdepurada_inv se convierta en poblacion_inv

SET PROCEDURE TO procedimientos_inv

DIMENSION em_cambio (14)

USE pobdepurada_inv

e_principio = 1

FOR e_ctcontador = 1 TO RECCOUNT ()
```

```
    USE pobdepurada_inv

        GO e_ctcontador

            DO llena_matriz WITH em_cambio

    USE poblacion_inv

        GO e_ctcontador

            DO reemp_mat WITH em_cambio, e_principio

ENDFOR
```

```
***  Programa:  procedimientos_inv.prg
***  Procedimientos que se utilizan en todo el programa del
***  Algoritmo Genético
***  para el Inventario (Visual Fox Pro).


***  Procedimiento que llena una matriz con el valor de los genes
***  de determinado
***  cromosoma.

PROCEDURE llena_matriz
    LPARAMETER em_cromosomas

    em_cromosomas (1)  = Gen_01
    em_cromosomas (2)  = Gen_02
    em_cromosomas (3)  = Gen_03
    em_cromosomas (4)  = Gen_04
    em_cromosomas (5)  = Gen_05
    em_cromosomas (6)  = Gen_06
    em_cromosomas (7)  = Gen_07
    em_cromosomas (8)  = Gen_08
    em_cromosomas (9)  = Gen_09
    em_cromosomas (10) = Gen_10
    em_cromosomas (11) = Gen_11
    em_cromosomas (12) = Gen_12
    em_cromosomas (13) = Gen_13
    em_cromosomas (14) = Gen_14

ENDPROC

***


***  Procedimiento que llena los campos de genes con los valores
```

```
***  de la matriz en el programa de reemplazamiento y cruce.

PROCEDURE reemp_mat
    LPARAMETER em_crom, e_inicia

    FOR e_gen = e_inicia TO FCOUNT ()

        REPLACE (FIELD (e_gen)) WITH em_crom (e_gen)

    ENDFOR

ENDPROC

***


***  Procedimiento que revisa si el padre y la madre son el mismo
***  cromosoma en el programa de cruce.

PROCEDURE revisa_padres
    LPARAMETER e_progrepetidos, e_progmas, e_progfem

    DO WHILE e_progrepetidos = 1

        IF e_progmas = e_progfem

            e_progfem = INT (1 + RAND() * 1000 % RECCOUNT ())

        ELSE

            e_progrepetidos = 0

        ENDIF

    ENDDO

ENDPROC

***


***  Procedimiento que realiza el cruce, cambia los cromosomas de
***  lugar en el programa de cruce.

PROCEDURE realiza_cruce
    LPARAMETER e_crucelugar, em_progmas, em_progfem

    FOR e_cambiogen = e_crucelugar TO FCOUNT ()

        e_guardar = em_progmas (e_cambiogen)
        em_progmas (e_cambiogen) = em_progfem (e_cambiogen)
        em_progfem (e_cambiogen) = e_guardar
```

```
    ENDFOR

ENDPROC

***
```

Capítulo IV

Resultados y conclusiones

Resultados

Se realizaron 50 corridas, las cuales se muestran a continuación.

• 7679	• 8084	• 8097	• 8129	• 8193
• 7935	• 8087	• 8098	• 8130	• 8193
• 8053	• 8090	• 8098	• 8136	• 8194
• 8056	• 8093	• 8098	• 8138	• 8196
• 8059	• 8093	• 8099	• 8188	• 8196
• 8062	• 8095	• 8101	• 8192	• 8198
• 8062	• 8096	• 8102	• 8192	• 8202
• 8081	• 8096	• 8106	• 8192	• 8211
• 8081	• 8096	• 8113	• 8192	• 8256
• 8084	• 8096	• 8128	• 8192	• 8337

Si sumamos todos los números y los dividimos entre 50 (que es el número de corridas) tenemos: 8119.50, que es una buena aproximación.

Si tomamos en cuenta los resultados de las 10 mejores aproximaciones

- 8098
- 8098
- 8097
- 8096
- 8096
- 8096
- 8096
- 8095
- 8093
- 8093

En estos resultados podemos observar que prácticamente alcanzamos el óptimo (8095.307). Y tenemos algunos otros valores que pudiéramos usar en caso de ser necesario.

Lo que nos brinda este método es la flexibilidad en la elección de la cantidad de artículos a ordenar de acuerdo a nuestros intereses.

Sacando el promedio de nuestros 10 mejores resultados observamos que el valor es 8095.8, casi el óptimo. Esto nos puede dar una idea de la confiabilidad de este método en nuestra aplicación.

Conclusiones

Al aplicar el Algoritmo Genético al problema de teoría de inventarios en su parte determinista de una empresa X que produce un bien no divisible Y, al analizar los resultados presentados anteriormente (las 10 mejores corridas) y compararlas con el resultado que nos arrojó el Modelo de Cantidad Económica de Pedido, nos damos cuenta que obtuvimos el óptimo y valores muy cercanos a él.

Al adentrarnos en esta metodología, nos damos cuenta que es muy sencillo trabajar con ella. Es una abstracción de la naturaleza y los pasos son sumamente simples: inicializar la población, darle valores según una función de aptitud, aplicar un método para definir que individuos pasarán a la siguiente generación, cruzar los individuos, mutarlos y elegir con cuales nos vamos a quedar para la siguiente generación. Repetimos estos pasos todas las veces necesarias hasta que nuestra población converja o lleguemos a un número determinado de iteraciones.

Al trabajar con este método y en caso que se tengan varios óptimos para nuestro problema, podemos escoger alguno de ellos. La fórmula de Cantidad Económica de Pedido solamente nos da uno, mientras que con el Algoritmo Genético podemos encontrar varios subóptimos. Esto nos ayuda mucho ya que en aplicaciones reales no siempre podemos conseguir el óptimo, en estos casos tenemos otras opciones de donde elegir.

La propuesta es aceptable porque es un método eficiente que nos puede dar una mejor aproximación en caso de que haya varios óptimos e incluso cuando hay un óptimo podemos elegir entre varios resultados subóptimos.

De tal forma podemos tomar el óptimo (que es un número entero) o alguno de los valores encontrados, ya que sabemos que son subóptimos, en caso de no poder producir el óptimo tenemos más opciones de donde elegir.

Los Algoritmos Genéticos se usaron para obtener subóptimos en un caso real, el Modelo de Lote Económico nos arroja un resultado, mientras que el Algoritmo Genético puede arrojar varios. En este trabajo lo primero que hacemos es obtener el valor a través del EOQ. Lo que hace nuestro algoritmo es probar con posibles soluciones. Se generan cadenas y se evalúan para ver cual se acerca más al óptimo y se sigue iterando el Algoritmo Genético hasta alcanzar el número máximo de iteraciones.

La solución por medio del EOQ es factible en la medida que es útil para el tomador de decisiones y se ajusta a los requisitos de inventario. Esta es una ventaja del A.G. con respecto al EOQ, ya que tenemos varias aproximaciones, por lo tanto tenemos varias

soluciones de donde elegir. Este trabajo sirve para aquellos que se encargan de manejar los inventarios y que buscan diferentes cantidades de pedido.

Algunas de las cuestiones en las que se tuvo que poner mayor atención al hacer el programa fueron definir los programas que se debían utilizar, aplicar procedimientos para no repetir partes de código y el manejo de las matrices para insertar registros en las tablas.

Un punto que se podría mejorar en el programa es el manejo de la memoria, para hacer los cálculos más rápidos y eficientes.

Esta aplicación es un método novedoso para tratar la teoría de inventarios y sienta precedentes para seguir abordando ésta a través de los Algoritmos Genéticos. Este trabajo pone las bases para entender mejor esta metodología y desarrollarla en la aplicación en inventarios. Sin hacer muchos cambios, podemos adaptar el Algoritmo Genético a diferentes problemas principalmente variando la función de aptitud.

Los Algoritmos Genéticos nos enseñan que podemos adaptar modelos de la naturaleza para resolver una gran cantidad de problemas, en nuestro caso uno relacionado a inventarios, obteniendo resultados adecuados a nuestro problema.

Una aplicación especialmente interesante relacionada con este trabajo es la siguiente:

Como informa Lemley, United Distillers and Vintners, una empresa escocesa que es el mayor y más rentable distribuidor de licores del mundo y es responsable de más de un tercio de la producción mundial de whisky de grano, utiliza un algoritmo genético para administrar su inventario y sus suministros. Esto es una tarea desalentadora que exige almacenar y distribuir eficientemente más de 7 millones de barriles, que contienen 60 recetas distintas, entre un enorme sistema de almacenes y destilerías, dependiendo de una multitud de factores como la edad, el número de malta, el tipo de madera y las condiciones del mercado. Anteriormente, coordinar este complejo flujo de suministro y demanda requería de cinco empleados a tiempo completo. Hoy, unas cuantas pulsaciones de teclado en un ordenador solicitan a un algoritmo genético que genere un programa cada semana, y la eficiencia de almacenamiento casi se ha duplicado.

Otras aplicaciones interesantes de los Algoritmos Genéticos son las siguientes:

Astronomía y astrofísica

Se utilizó un Algoritmo Genético para sacar los valores de los parámetros críticos de un modelo magnetohidrodinámico del viento

solar en este problema se obtienen los seis parámetros críticos del viento solar. El AG determinó con éxito el valor de tres con una precisión de menos del 0,1% y los otros tres con precisiones entre el 1 y el 10%. (Aunque siempre serían preferibles unos errores experimentales menores para estos tres parámetros, Charbonneau señala que no existe ningún otro método eficiente y robusto para resolver experimentalmente un problema no lineal 6-dimensional de este tipo; un método de gradiente conjugado funciona "siempre que se pueda proporcionar un valor inicial muy acertado". En contraste, los AGs no requieren un conocimiento del dominio tan bien afinado).

Basándose en los resultados obtenidos hasta ahora, Charbonneau sugiere que los AGs pueden y deben encontrar uso en otros problemas difíciles de astrofísica, en particular, problemas inversos como las imágenes por Doppler y las inversiones heliosísmicas. Para terminar, Charbonneau sostiene que los AGs son un "contendiente poderoso y prometedor" en este campo, del que se puede esperar que complemente (no sustituya) a las técnicas tradicionales de optimización, y concluye que "el punto decisivo, si es que tiene que haber alguno, es que los algoritmos genéticos funcionan, y a menudo colosalmente bien".

Mercados financieros

La utilización de los AGs en los mercados financieros ha empezado a extenderse en las empresas de corretaje bursátil del mundo real. Naik informa de que LBS Capital Management, una empresa estadounidense con sede en Florida, utiliza algoritmos genéticos para escoger las acciones de los fondos de pensiones que administra. Coale, Begley y Beals informan de que First Quadrant, una empresa de inversiones de California que mueve más de 2.200 millones de dólares, utiliza AGs para tomar decisiones de inversión en todos sus servicios financieros.

Matemáticas y algoritmia

Haupt y Haupt describen el uso de AGs para resolver ecuaciones de derivadas parciales no lineales de alto orden, normalmente encontrando los valores para los que las ecuaciones se hacen cero, y dan como ejemplo una solución casi perfecta para los coeficientes de la ecuación de quinto orden conocida como Super Korteweg-de Vries.

Robótica

El torneo internacional RoboCup es un proyecto para promocionar el avance de la robótica, la inteligencia artificial y los campos relacionados, proporcionando un problema estándar con el que probar las nuevas tecnologías -concretamente, es un campeonato anual de fútbol entre equipos de robots autónomos. (El

objetivo fijado es desarrollar un equipo de robots humanoides que puedan vencer al equipo humano de fútbol que sea campeón del mundo en 2050). Los programas que controlan a los miembros del equipo robótico deben exhibir un comportamiento complejo, decidiendo cuándo bloquear, cuándo tirar, cómo moverse, cuándo pasar la pelota a un compañero, cómo coordinar la defensa y el ataque, etcétera. En la liga simulada de 1997, David Andre y Astro Teller inscribieron a un equipo llamado Darwin United cuyos programas de control habían sido desarrollados automáticamente desde cero mediante programación genética, un desafío a la creencia convencional de que "este problema es simplemente demasiado difícil para una técnica como ésa".

Para resolver este difícil problema, Andre y Teller le proporcionaron al programa genético un conjunto de funciones de control primitivas como girar, moverse, tirar, etcétera. (Estas funciones estaban también sujetas al cambio y refinamiento durante el curso de la evolución). Su función de aptitud, escrita para que recompensara el buen juego en general en lugar de marcar goles expresamente, proporcionaba una lista de objetivos cada vez más importantes: acercarse a la pelota, golpear la pelota, conservar la pelota en el campo contrario, moverse en la dirección correcta, marcar goles y ganar el partido. Debe señalarse que no se suministró ningún código para enseñar específicamente al equipo cómo conseguir estos objetivos complejos. Luego los programas evolucionados se evaluaron utilizando un modelo de selección jerárquico: en primer lugar, los equipos candidatos se probaron en un campo vacío y, si no marcaban un gol en menos de 30 segundos, se rechazaban. Luego se evaluaron haciéndoles jugar contra un equipo estacionario de "postes pateadores" que golpeaban la pelota hacia el campo contrario. En tercer lugar, el equipo jugaba un partido contra el equipo ganador de la competición RoboCup de 1997. Finalmente, los equipos que marcaron al menos un gol contra este equipo jugaron unos contra otros para determinar cuál era el mejor.

De los 34 equipos de su división, Darwin United acabó en decimoséptima posición, situándose justo en el medio de la clasificación y superando a la mitad de los participantes escritos por humanos. Aunque una victoria en el torneo sin duda habría sido más impresionante, este resultado es competitivo y significante de pleno derecho, y lo parece aún más a la luz de la historia. Hace unos 25 años, los programas informáticos que jugaban al ajedrez estaban en su infancia; por primera vez, una computadora había sido inscrita recientemente en una competición regional, aunque no ganó. Pero "una máquina que juega al ajedrez a un nivel medio de la capacidad humana es una máquina muy capaz", y podría decirse que lo mismo es cierto para el fútbol robotizado. Si las máquinas de ajedrez actuales compiten al nivel de los grandes maestros, ¿qué tipo de sistemas producirá la programación genética dentro de 20 o 30 años?

Diseño de rutas y horarios

La compañía de telecomunicaciones U.S. West (ahora fusionada con Qwest) se enfrentó a la tarea de desplegar una red de fibra óptica. Hasta hace poco, el problema de diseñar la red para minimizar la longitud total de cable desplegado era resuelto por un ingeniero experimentado; ahora la compañía utiliza un algoritmo genético para realizar la tarea automáticamente. Los resultados: "El tiempo de diseño para las redes nuevas ha caído de dos meses a dos días, y le supone un ahorro a U.S. West de 1 millón a 10 millones de dólares cada una".

Beasley, Sonander y Havelock utilizaron un AG para programar los aterrizajes del London Heathrow, el aeropuerto más transitado del Reino Unido. Esto es un problema multiobjetivo que implica, entre otras cosas, minimizar los retrasos y maximizar el número de vuelos mientras se mantiene la suficiente distancia de separación entre los aviones (los vórtices de aire que se forman en la estela de un avión pueden ser peligrosos para otro avión que vuele demasiado cerca). Comparado con los horarios reales de un periodo intensivo del aeropuerto, el AG fue capaz de reducir el tiempo de espera medio en un 2-5%, implicando dos o tres vuelos extra despegando y aterrizando por cada hora -una mejora significativa. Sin embargo, se han logrado mejoras mayores: como se informa en Wired, aeropuertos internacionales y líneas aéreas importantes como Heatrhow, Toronto, Sydney, Las Vegas, San Francisco, America West Airlines, AeroMexico (ver http://www.ascent.com/case-aeromexico.html) y Delta Airlines están utilizando algoritmos genéticos para programar los despegues, aterrizajes, mantenimiento y otras tareas, mediante el software del Ascent Technology's SmartAirport Operations Center. Cruzando y mutando las soluciones en forma de horarios que incorporan miles de variables, "Ascent vence con comodidad a los humanos, aumentando la productividad hasta en un 30 por ciento en todos los aeropuertos en los que se ha implementado".

Ingeniería de sistemas

Lee y Zak utilizaron un algoritmo genético para evolucionar un conjunto de reglas para controlar un sistema de frenos antibloqueo automovilístico. Aunque la capacidad que tienen los sistemas de freno antibloqueo de reducir la distancia de frenada y mejorar la maniobrabilidad ha salvado muchas vidas, el rendimiento del ABS depende de las condiciones de la superficie de la carretera: por ejemplo, un controlador ABS que esté optimizado para el asfalto seco no funcionará igual de bien en carreteras mojadas o heladas, y viceversa. En este artículo, los autores proponen un AG para ajustar un controlador ABS que pueda identificar las propiedades de la superficie de la carretera (monitorizando el patinaje y aceleración de las ruedas) y pueda actuar en consecuencia, liberando la cantidad adecuada de fuerza de frenado para maximizar

la tracción de las ruedas. En las pruebas, el ABS puesto a punto genéticamente "exhibe características de rodada excelentes" y fue "muy superior"" a los otros dos métodos de maniobras de frenado, encontrando con rapidez nuevos valores óptimos para el patinaje de las ruedas cuando cambia el tipo de terreno bajo un coche en movimiento, y reduciendo la distancia total de frenada. "La lección que hemos aprendido de nuestro experimento... es que un AG puede ayudar a ajustar incluso un controlador bien diseñado. En nuestro caso, ya teníamos una buena solución del problema; sin embargo, con la ayuda de un AG, conseguimos mejorar significativamente la estrategia de control.

Finalmente, como cita Ashley, empresas de la industria aeroespacial, automovilística, fabril, turbomaquinaria y electrónica están utilizando un sistema de software propietario conocido como Engineous, que utiliza algoritmos genéticos, para diseñar y mejorar motores, turbinas y otros dispositivos industriales. En palabras de su creador, el Dr. Siu Shing Tong, Engineous es "un maestro 'toqueteador', ensayando incansablemente las puntuaciones de escenarios de tipo "y-si" hasta que emerge la mejor solución posible". En un ensayo del sistema, Engineous consiguió producir un incremento del 0,92 por ciento de la eficiencia de una turbina experimental en sólo una semana, mientras que diez semanas de trabajo de un diseñador humano sólo produjeron un 0,5 por ciento de mejora.

BIBLIOGRAFÍA

- Holland, J. H. Adaptation in Natural and Artificial Systems. Ann Arbor, MI: The University of Michigan, 1975.

- Goldberg, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley, 1989.

- Davis, L., ed. Handbook of Genetic Algorithms. New York: Van Nostrand Reinhold, 1991.

- Michalewicz, Zbigniew, Genetic Algorithms + data structures = evolution programs. 3rd rev. and extended ed., Springer-Verlag, 1996.

- Bauer, Richard J. Jr., Genetic Algorithms and investment strategies. John Wiley & Sons, 1994.

- Nahmias, Steven, Análisis de la producción y las operaciones. Ed. CECSA, Primera edición, México, 1999.

- Taha, Hamdy A., Investigación de Operaciones, una introducción. Prentice Hall, México, 1998.

- Cruz Ulloa Susana y Arechavaleta Yolanda, Textos de Biología. Colegio de Ciencias y Humanidades.

- Nason Alvin, Biología. 27ª reimpresión, México, Editorial Limusa, 1992.