UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Escuela Nacional de Estudios Profesionales
" ACATLAN "

SELECCION Y APLICACION DE UN ALGORITMO PARA UNA RED DE DISTRIBUCION DE MEDICAMENTOS A LAS FARMACIAS DE LA SECRETARIA DE SALUBRIDAD Y ASISTENCIA

T E S I S

Que presenta:

Virginia Rivera Lara

Para Obtener el Título de:

A C T U A R I O

MEXICO 1982

UNIVERSIDAD NACIONAL AVTONOMA

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ACATLAN"

COORDINACION DEL PROGRAMA DE ACTUARIA Y MATEMATICAS APLICADAS Y COMPUTACION

CAMAC-008/83.

SRITA. VIRGINIA RIVERA LARA
Alumna de la carrera de Actuaría,
P r e s e n t e.

De acuerdo a su solicitud presentada con fecha 7 de diciembre de 1982, me complace notificarle que esta Coordinación tuvo a bien asignarle el siguiente tema de tesis: "Selección y Aplicación de un Algoritmo para una Red de Distribución de Medicamentos a las Farmacias de la Secretaría de Salubridad y Asistencia", el cual se desarrollará como sigue:

Antecedentes
1.- Planteamiento del problema
2.- Algoritmos
3.- Aplicación
4.- Caso práctico
5.- Experiencia computacional del caso práctico
Conclusiones

Asímismo fué designado como Asesor de Tesis al Sr. Act. Javier Ramírez Rodríguez.

Ruego a usted tomar nota que en cumplimiento a lo especificado en la Ley de Profesiones, deberá prestar servicio social durante un tiempo mínimo de seis meses como requisito básico para sustentar examen profesional, así como de la disposición de la Dirección General de Servicios Escolares en el sentido de que se imprima en lugar visible de los ejemplares de la tesis el título del trabajo realizado. Esta comunicación deberá imprimirse en el interior de la tesis.

A t e n t a m e n t e
"POR MI RAZA HABLARA EL ESPIRITU"
Acatlán, Edo. de Méx., a 15 de Febrero de 1983.

ENEP ACATLAN
COORDINACION ACTUARIA Y MATEMATICAS APLICADAS Y COMPUTACION

ING. ADALBERTO LOPEZ LOPEZ
Coordinador.

RECONOCIMIENTOS

Reconocer la ayuda de algunas personas, puede resultar injusto para otras, pero a pesar de ello no puedo dejar de subrayar la valiosa ayuda y estímulo del señor Javier Ramírez en la dirección de este trabajo. También deseo dejar constancia de la colaboración del señor Iñaqui Olaizola, y de las importantes observaciones hechas por Guillermo Molina.

A familiares, maestros y amigos de los que he recibido el estímulo, los conocimientos y experiencias indispensables para mi formación.

Para todos mi sincero agradecimiento.

INTRODUCCION

En nuestro lugar y tiempo, México 1982-1983, resulta indispensable aprovechar adecuadamente los avances tecnológicos y científicos disponibles. Sólo el mejor empleo de los recursos hará posible encontrar soluciones a la diversidad y complejidad de los problemas, cuya solución garantice el desarrollo necesario a nuestro país.

Con el propósito de colaborar en este sentido, este trabajo presenta una solución práctica a la necesidad de establecer rutas adecuadas, para distribuir bienes y servicios de centros de distribución a centros de consumo, al mínimo costo posible.

Para mostrar la aplicación del algoritmo seleccionado en un caso práctico, se presenta un proyecto propuesto a la Gerencia General de Farmacias de la Secretaría de Salubridad y Asistencia, por la unidad de Información de la propia Secretaría. El proyecto tiene como objetivo solucionar la distribución de medicamentos a las farmacias, partiendo de almacenes dados.

El tema, en Investigación de Operaciones (IO), se conoce como "Problema del Agente Viajero". Consiste en que, partiendo de la localidad donde esté ubicado un almacén, se visite solo una vez cada localidad distribuidora y regresar al punto de partida; a este recorrido se le llama circuito Hamiltoniano.

La idea básica para solucionar este problema, es encontrar el circuito Hamiltoniano que tenga un valor mínimo.

Para construir circuitos óptimos, se han desarrollado gran número de algoritmos puestos en práctica mediante programas de computadora, cuya principal limitación es el tiempo de cómputo que requieren.

Como antecedente, en este trabajo se mencionan generalidades acerca del problema del agente viajero, y los diferentes tipos de soluciones que han encontrado los investigadores de IO.

En el capítulo 1, se presenta el problema. En el capítulo 2, se mencionan algoritmos que lo resuelven; se hace un exámen de ellos para decidir cual usar, hasta llegar a la elección del algoritmo a emplearse. En el capítulo 3, se resuelve un ejemplo que ilustra el funcionamiento del algoritmo. En el capítulo 4, se da solución al problema práctico de la distribución de medicamentos a farmacias empleando un programa de computadora y se presenta la solución a la distribución a farmacias del almacén de Oaxaca. En el capítulo 5, se muestra la experiencia computacional obtenida del caso práctico.

La presentación tiene el propósito de facilitar la aplicación del algoritmo en la solución de problemas similares de distribución.

I N D I C E

I N D I C E (Cont.)

ANEXOS:

ANTECEDENTES

En 1934, H. Whitney planteó formalmente por primera vez el problema del agente viajero (PAV) 4/. Pero es a partir de la década de los cincuenta cuando fue posible solucionar problemas de 40 a 50 ciudades.

Por su parte, Shen Lin definió matemáticamente el problema del agente viajero (PAV), de las siguientes dos formas equivalentes:

a) Dada una matriz de costos $C = (c_{ij})$, donde c_{ij} es igual al costo de ir de la ciudad i a la ciudad j, $(i, j = 1, \ldots, n)$ encontrar una permutación de un viaje $t = (i_1, i_2, \ldots, i_n)$ de los enteros uno hasta n, que minimicen la cantidad:

$$c_{i_1 i_2} + c_{i_2 i_3} + \ldots + c_{i_n i_1}$$

b) Dada una matriz de costos C como la anterior, determinar las x_{ij} que minimicen la cantidad $Q = \sum_{ij}^{n} c_{ij} x_{ij}$, sujeto a las siguientes restricciones:

i) $x_{ii} = 0$

ii) $x_{ij} = 0, 1$

iii) $\sum_{i}^{n} x_{ij} = 1$ para toda j; $\sum_{j}^{n} x_{ij} = 1$ para toda i

iv) y, para cualquier permutación:

$S = (i_1, i_2, \ldots, i_r)$, $r \leq n$, de los enteros uno hasta n:

$$x_{i_1 i_2} + x_{i_2 i_3} + \ldots + x_{i_{r-1} i_r} + x_{i_r i_1} \begin{cases} < r & \text{si } r < n \\ = n & \text{si } r = n \end{cases}$$

Es conveniente hacer notar que el problema conocido como Problema de Asignación (PA), se define como el PAV, con excepción de la restricción iv), señalada en el párrafo anterior.

El PAV, formulado como problema de Programación Lineal, aún cuando el número de ciudades a visitar sea pequeño, ha mostrado una factibilidad muy cuestionable, tanto por su formulación como por su ejecución.

Hillier y Lieberman, en su libro Operations Research, exponen: "A veces es difícil ver de que manera el redondeo puede ser hecho para retener la factibilidad. Puede ser necesario cambiar el valor de ciertas variables por una o más unidades después del redondeo. Para ilustrar suponga que algunas de las restricciones son:

$$-x_1 + x_2 \leq 3\ 1/2$$

$$x_1 + x_2 \leq 16\ 1/2$$

y que el Método Simplex ha identificado la solución óptima no entera como $x_1 = 6\ 1/2$ y $x_2 = 10$, nótese que es imposible redondear x_1 a 6 ó 7 (o cualquier otro entero), y retener la factibilidad. Esto sólo puede ser hecho cambiando el valor entero de x_2. Es fácil imaginar como muchas dificultades pueden ser combinadas cuando hay decenas y centenas de restricciones y variables".

Es por esto que en la práctica se ha intentado solucionar este problema usando programación Lineal, ignorando la restricción de valores enteros, y una vez establecida la solución redondeando los valores fraccionarios que hayan resultado a tomar valores enteros, lo cual no garantiza que la solución resultante sea factible u óptima (que maximiza o minimiza los objetivos del sistema, según sean beneficios o costos).

De aquí que hayan sido desarrollados gran número de algoritmos heurísticos por personas interesadas en el tema (Flood, Little et al., Gilmore y Gomory, Lin, Bellmore y Malone, Picard y Queyranne, etc.)

Los algoritmos heurísticos son modelos iterativos que en cada nuevo paso, descubren soluciones o valores, cada vez mas cercanos al valor óptimo, hasta encontrar un valor cuya diferencia con el óptimo sea mínima.

Los investigadores mencionados, tomaron un conjunto de viajes y diseñaron algoritmos cuya solución les diera un conjunto más pequeño que todos los viajes posibles.

Por otra parte, los algoritmos conocidos para solucionar el PAV, generalmente se dividen en:

Un punto de partida.

Un modelo generador de soluciones.

Una regla de terminación.

Si la regla de terminación es tal que la iteración se detiene si y sólo si un viaje es óptimo, el método es exacto. Si la regla es tal que se detiene, pero no sólo si un viaje es óptimo, el método es aproximado.

En los métodos aproximados , el viaje encontrado depende generalmente del punto de partida, de aquí que sea posible producir diferentes viajes finales usando diferentes puntos de partida.

Ya que la mayoría de los puntos de partida y reglas de terminación dependen en parte del modelo empleado para producir soluciones, los algoritmos de solución al PAV se pueden clasificar, de acuerdo al método de producir soluciones, en los tres modos siguientes:

1) Mejoras viaje a viaje.

Su punto de partida es un viaje arbitrario. Este modelo productor de soluciones es una regla para encontrar un mejor viaje que es vecino del primero.

Los algoritmos de este tipo deben juzgarse por su eficiencia computacional, (calidad de la solución contra el tiempo utilizado).

2) Construcción de viaje.

El viaje que se busca se va construyendo empezando de un punto arbitrario (ciudad o actividad a desarrollarse), siguiendo con el más cercano y así sucesivamente.

3) Eliminación de subviajes. (viajes que enlazan a menos de las n ciudades o actividades que involucra el problema).

El punto de partida es una solución óptima que puede contener subviajes. Si no los contiene, el viaje es óptimo para el PAV.

En el caso de que la solución contenga subviajes, se emplea un mode lo iterativo para eliminarlos, iterando hasta alcanzar una solución que es un viaje factible.

CAPITULO 1

PLANTEAMIENTO DEL PROBLEMA

1.1 Situación actual

Hoy en día, no son óptimas las rutas que se siguen para distribuir adecuadamente los medicamentos de los almacenes a las farmacias de la Secretaría de Salubridad y Asistencia (SSA); lo que ocasiona que algunos medicamentos no se tengan en existencia por períodos que varían.

1.2 Objetivo

Establecer rutas óptimas de distribución a las farmacias de dicha Secretaría a través de sus almacenes en un tiempo mínimo; estableciendo un programa para distribución de medicamentos de un almacén a un número determinado de farmacias. El logro del objetivo planteado se reflejará en un mejor servicio por parte de las farmacias al público demandante.

1.3 Consideraciones generales

Es cierto que la variable costo se tradujo para el caso que aquí se presenta, en la variable distancia, y fueron considerados los costos de las rutas como directamente proporcionales a las distancias recorridas.

Sin embargo, es posible obtener un modelo más completo, adicionando a las distancias de recorrido otras variables (diferencia de altura entre ciudades, tipo de carreteras, etc.) que intervienen en la composición de los costos, aunque en forma indirecta y mínima.

La adición de estas últimas variables complica de tal modo el modelo, que las mejoras que pudieran obtenerse, no justifican dicha complicación, sobre todo, tomando en cuenta la carencia de información confiable y registros adecuados.

En el presente trabajo se propone un modelo tomando como base el algoritmo de Little et al., Este algoritmo proporciona las mejores soluciones para establecer rutas o circuitos óptimos. Es pertinente mencionar que además, el tiempo de computadora necesario para la solución de problemas específicos es mínimo.

La distribución de medicamantos, en el caso presentado, tiene la restricción de la capacidad de los transportes, esto limita los puntos de entrega a no más de 20.

CAPITULO 2

2.0 ALGORITMOS PROPUESTOS

Algunos algoritmos que han sido propuestos para resolver el PAV son los siguientes:

EL PROBLEMA DEL AGENTE VIAJERO (PAV)

Merrill M. Flood 8/

Para solucionar el PAV, se propone solucionar primero el PA asociado a él, y tomar la solución que se obtenga como solución inicial para el cálculo de la solución al PAV correspondiente.

Este trabajo propició la investigación de métodos de solución al PAV.

UN ALGORITMO PARA EL PROBLEMA DEL AGENTE VIAJERO.

Little et al., (Little J., Murty K., Sweeney D., y Karel C. 9/

Los autores, refiriéndose a trabajos anteriores, mencionan que algunos de los métodos para solucionar el problema han sido ineficaces, otros dan soluciones no necesariamente óptimas, y algunos más requieren de juicios intuitivos que hacen difícil ejecutar programas en computadora.

Se restringen los autores a métodos que:

-Garantizan optimización.

-Parecen razonables de programarse.

-Son generales (no fallan ante problemas de tamaño específico).

Little et al., presentan un algoritmo de ramificación y acotamiento* que incrementa su tiempo de cómputo a lo menos en forma exponencial, pero

* En la página 21 c se presenta una descripción sucinta de la técnica de ramificación y acotamiento.

aumenta considerablemente también el número de ciudades que involucra.

Los autores de este algoritmo comentan que:

"Los subconjuntos de viajes son convenientemente representados como los nodos de un árbol y el proceso de partición como una ramificación del árbol, de aquí que se ha llamado el método de ramificación y acotamiento". 9/

Los autores aclaran también, que la eficiencia del proceso de ramificación y acotamiento descansa en la división usada para elegir los subconjuntos y encontrar cotas bajas (valores mínimos de cada paso).

La causa de la ramificación y el cálculo de las cotas bajas se origina en ideas frecuentemente usadas en la solución a problemas de asignación.

La técnica de ramificación y acotamiento es usada ampliamente en la solución de problemas de programación entera.

Experiencia computacional

Problemas de hasta diez ciudades pudieron resolverse a mano en menos de una hora.

Las principales pruebas del algoritmo se hicieron en una máquina IBM 7090, estudiándose problemas de distancias obtenidas aleatoriamente, así como problemas publicados por otros autores, y subproblemas construídos de ellos suprimiendo varias ciudades.

La mayoría de los problemas publicados, se construyeron tomando las distancias registradas en atlas de carreteras y por lo tanto son simétricos.**Los tiempos de cómputo promedio mostraron que para pro -

**Simétricos: Distancia A B = B A

blemas de hasta veinte ciudades sólo se necesitó menos de un minuto de computadora. Sin embargo, el tiempo creció exponencialmente y para cuarenta ciudades el tiempo aumentó hasta un promedio de 8 minutos.

Las desviaciones estándar de estos tiempos de cómputo también - crecieron con el tamaño del problema:
Para problemas de 10 ciudades, la desviación observada fue de menos de un centésimo de minuto; en cambio, para 40 ciudades la desviación- fue mayor a los 2 minutos.

Los problemas simétricos tomaron un tiempo más largo que los - problemas de distancias asimétricas de igual tamaño.

UN CASO DEL PROBLEMA DEL AGENTE VIAJERO

Gilmore y Gomory 10/

Se plantea el problema de hallar la secuencia de costo mínimo para los n trabajos de una máquina.

c_{ij} define el costo de pasar la máquina de un trabajo i a un - trabajo j, el cual puede representar tiempo, dinero, etc.; y si el - primero y el último trabajo son el mismo, se considera este problema como un caso especial del PAV.

SOLUCIONES COMPUTACIONALES AL PROBLEMA DEL AGENTE VIAJERO

Shen Lin 11/

Lin propone un algoritmo de mejoras viaje a viaje.

Dicho algoritmo proporciona subviajes óptimos para problemas hasta de 105 ciudades.

Experiencia computacional.

Lin reporta que el tiempo de cómputo promedio en la máquina - IBM 7094 modelo II para obtener un subviaje óptimo, es menor o - igual a $30\ n^3$ microsegundos, y la probabilidad de que dicho viaje sea verdaderamente óptimo es aproximadamente $p(n) = 2^{-n/10}$.
n = puntos de entrega

EL PROBLEMA DEL AGENTE VIAJERO: UNA EXPLORACION

Bellmore y Nemhauser 12/

Entre sus investigaciones, estos autores mencionan que han sido propuestas posibilidades de reducir el tamaño del PAV, imponiendo restricciones en el orden en el cual las ciudades pueden visitarse.

En el algoritmo de Lin, por ejemplo, se reduce el tamaño del problema suponiendo que una vez que el enlace de un par de ciudades - (un arco), aparece en las soluciones obtenidas por algoritmos aproximados, lo hará en un viaje óptimo, reduciendo así la magnitud de un problema.

ALGORITMOS DE ELIMINACION DE SUBVIAJES PARA EL PROBLEMA DEL AGENTE VIAJERO.

Bellmore y Malone 14/

Los autores afirman que la experiencia computacional reportada- de los algoritmos publicados varía ampliamente, pero que no se ha ex - plicado adecuadamente esta variación, tampoco se han ofrecido teorías que indiquen sus aproximaciones.

Los autores desarrollan una teoría que subraya este problema, la cual se basa en la obtención de soluciones mediante la eliminación de subviajes, probándose esta teoría con problemas generados aleatoriamente.

Se proponen dos algoritmos de eliminación de subviajes para el PAV -para los casos simétrico y asimétrico-. Ambos se programaron en lenguaje Fortran IV, y mostraron un crecimiento en tiempo de cómputo, con respecto al número de ciudades.

Experiencia computacional.

Para el caso asimétrico, el algoritmo se probó para problemas hasta de 80 ciudades en una máquina IBM 7094. Se obtuvo que para problemas de 10 a 30 ciudades, el tiempo promedio varía de décimas de segundo a 13 segundos. Para problemas de 40 ciudades el tiempo se aproxima a 20 segundos.

El algoritmo para el caso simétrico se probó en una máquina UNIVAC 1108.

Para problemas de 10 a 30 ciudades, el tiempo promedio de corrida varió de aproximadamente 3 segundos hasta más de 300 respectivamente.

COTAS PARA EL PROBLEMA DEL AGENTE VIAJERO.

Nicos Christofides 15/

Se establece una cota baja al PAV, y se espera que los algoritmos de ramificación y acotamiento para este problema puedan beneficiarse de ella. La cota se calcula con un proceso iterativo que converge y requiere un tiempo de cómputo 9% mayor que el tiempo requerido para solucionar un PA equivalente.

Experiencia computacional.

Esta nueva cota se probó para problemas de 10 a 100 ciudades,- - en promedio se establece que las cotas obtenidas fueron solo 4.7 % inferiores a la solución óptima para el caso simétrico y 3.8 % inferiores al óptimo para el caso asimétrico.

UN ALGORITMO HEURISTICO PARA EL PROBLEMA DEL AGENTE VIAJERO

Lin y Kernighan 16/

Estos autores presentan un algoritmo para producir soluciones óptimas y cercanas a la óptima para el PAV simétrico.

Los tiempos de cómputo para este algoritmo heurístico crecen como n^2 microsegundos aproximadamente, aunque la efectividad de predecir el crecimiento disminuye al incrementarse n.

Experiencia computacional.

El método propuesto por Lin y Kernighan se probó en la solución de problemas con distintos números de puntos. Se obtuvieron 4 soluciones distintas para problemas de 48 puntos. En el caso de 57 puntos el método produjo 8 soluciones distintas. La comparación de los resultados con otros, obtenidos de métodos de optimalidad probada, mostró - aproximación en los valores mínimos.

Para problemas con datos obtenidos aleatoriamente, la frecuencia de obtener el óptimo disminuye, varía entre el 1% y el 68%, por que el número medio de soluciones aproximadas varía más que en los otros tipos de problemas probados.

EL PROBLEMA DEL AGENTE VIAJERO DEPENDIENTE DEL TIEMPO Y SU APLICACION AL PROBLEMA DE SECUENCIA DE TRABAJOS DE UNA MAQUINA. Picard y Queyranne 17/

El problema del agente viajero dependiente del tiempo es una generalización del PAV y del PA, que se define como el problema de secuencia de n trabajos de una máquina, en la que los n trabajos deben procesarse a un costo mínimo.

El costo asociado con cada trabajo no sólo depende del trabajo que lo precede, sino también de su posición en el orden que define la secuencia de trabajos.

Los autores proponen un método de ramificación y acotamiento para este problema. Por otra parte, cuando los costos de cada trabajo no son dependientes del tiempo, el problema se convierte en el PAV, con el que los autores no reportan experiencia computacional.

EL PROBLEMA DE LOS M AGENTES VIAJEROS

Hong y Padberg 18/, Steele 19/, Ali y Kennington 20/

A partir de la década pasada, un caso más general del PAV ha empezado a analizarse, el problema de los m agentes viajeros (PMAV). Según investigaciones realizadas recientemente por los autores mencionados, el problema ha sido planteado en la siguiente forma:

Se tienen n ciudades (1, ..., n), y m agentes disponibles, todos localizados en la ciudad k. Se desea establecer un conjunto de viajes, de tal modo que cada ciudad sea visitada una sola vez por exactamente un vendedor, y cada vendedor sea asignado a un viaje, partiendo y terminando su viaje en la ciudad k.

c_{ij} es igual a la distancia de la ciudad i a la ciudad j.

El objetivo es encontrar el número de agentes a ser empleados y sus respectivas rutas, de modo que la distancia total recorrida por ellos sea mínima.

Si la solución óptima debe componerse de m viajes, entonces se llama a este modelo, el problema de los m agentes viajeros fijos (PMAVF). Si m es una cota máxima en el número de viajes en la solución óptima, entonces a este modelo se le llama el problema de los m agentes viajeros variables (PMAVV).

Para el caso de m variables, pueden fijarse cargas asociadas con los diferentes agentes.

Si m es igual a 1, los dos modelos coinciden con el PAV.

$c_{ij} = c_{ji}$, implica que el problema es simétrico, en otro caso es asimétrico.

A diferencia del problema del único vendedor, para valores dados de n y m, el número total de viajes para el problema asimétrico puede ser sustancialmente más grande (cuatro veces o más), que el número de viajes para el problema simétrico, hecho que llama la atención a la investigación de ...

*Ramificación y acotamiento es una técnica de programación entera, que se utiliza para resolver problemas que tienen un número finito de soluciones. Las soluciones se pueden encontrar por el simple procedimiento de enumeración y selección; desafortunadamente el número finito de soluciones, usualmente es muy grande, esto hace necesario utilizar los beneficios de las computadoras para encontrar las soluciones.

El número total de posibles soluciones de un problema con 10 variables, cada variable con 10 valores factibles, es de 10^{10}.

A partir de la página 27 un ejemplo ilustra la técnica mencionada. Hillier & Lieberman 2/, tratan y explican extensamente la técnica.

CAPITULO 3

APLICACION

El presente capítulo se compone de dos apartados: El primero describe el funcionamiento del algoritmo de Little et al., con la ayuda de un diagrama de bloques y la descripción de cada bloque, que hacen el planteamiento matemático del Problema del Agente Viajero.

El segundo apartado tiene el propósito de facilitar la comprensión del funcionamiento del algoritmo mediante la demostración en una aplicación práctica. El ejemplo incluye cinco puntos de entrega y un centro distribuidor.

3.1 Algoritmo de Little et al.

El algoritmo de Little et al., incluídas las mejoras sugeridas por Lin, se basa en un principio de separación dicotómica. Consiste en aumentar o nó, a un conjunto ya determinado, un nuevo elemento. Esto se hace midiendo la eficacia de cada una de las dos decisiones por la evaluación de la cota inferior de un costo.

PLANTEAMIENTO MATEMATICO DEL PROBLEMA DEL AGENTE VIAJERO

Función objetivo:

$$\text{Min } Q = \sum_{ij}^{n} C_{ij} X_{ij}$$

Restricciones:

a) $X_{ii} = 0$

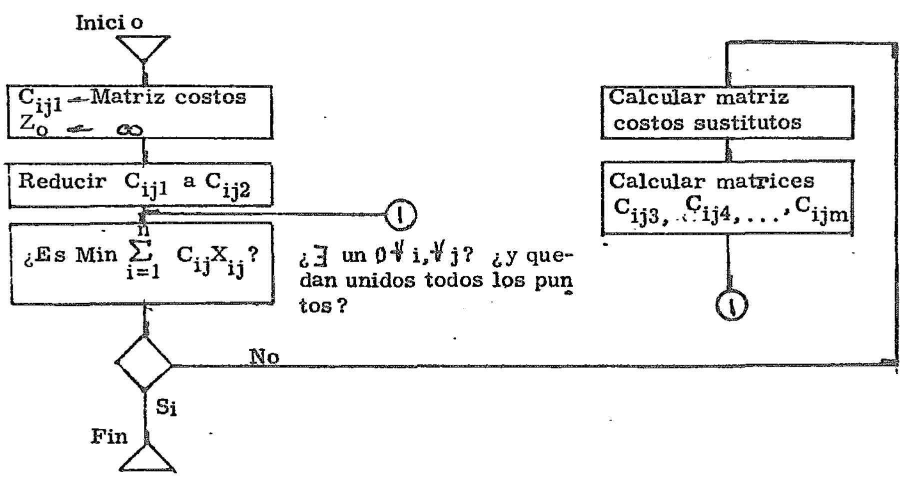
b) $X_{ij} = 0, 1$

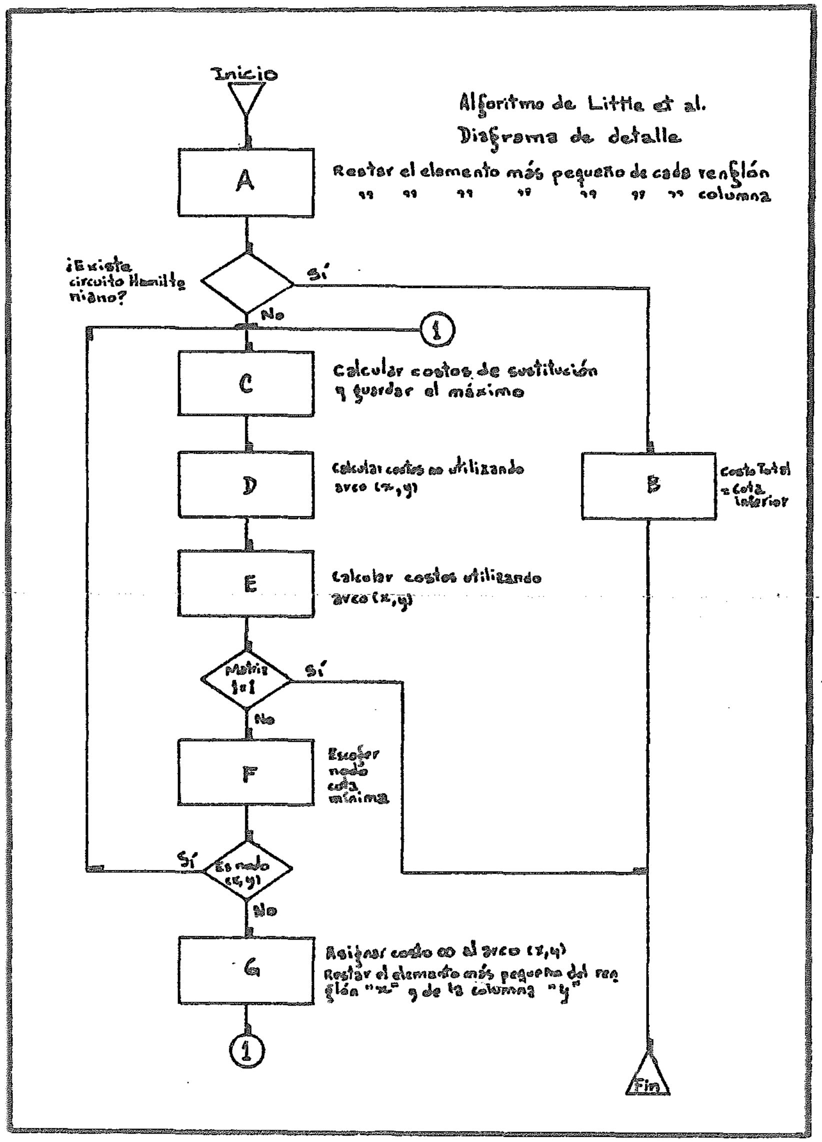
c) $\sum_{i=1}^{n} X_{ij} = 1; \forall j; \quad \sum_{j=1}^{n} X_{ij} = 1, \forall i$

d) Para cualquier permutación: $S = (i_1, i_2, \dots, i_r)$, $r \leq n$, de los enteros 1 hasta n:

$$X_{i_1 i_2} + X_{i_2 i_3} + \dots + X_{i_{r-1} i_r} + X_{i_r i_1} \begin{cases} \leq r & \text{si } r < n \\ = n & \text{si } r = n \end{cases}$$

DIAGRAMA DE PRIMER NIVEL




Algoritmo de Little et al.
Diagrama de detalle
Inicio
A
Restar el elemento más pequeño de cada renglón
" " " " " " " columna
¿Existe circuito Hamiltoniano?
Sí
No
1
C
Calcular costos de sustitución y guardar el máximo
D
Calcular costos no utilizando arco (x,y)
E
Calcular costos utilizando arco (x,y)
Matriz 1x1
Sí
No
F
Escoger nodo cota mínima
Es nodo (x,y)
Sí
No
G
Asignar costo ∞ al arco (x,y)
Restar el elemento más pequeño del renglón "x" y de la columna "y"
1
B
Costo Total = cota inferior
Fin

A continuación se presenta el algoritmo dividido en bloques:

BLOQUE A. Restar el elemento más pequeño de cada renglón; después restar el elemento más pequeño de cada columna. La suma de estos valores es la cota inferior.

BLOQUE B. Si a partir de la matriz resultante en el bloque A, se puede encontrar un circuito Hamiltoniano, utilizando únicamente los arcos de costo cero, este circuito será la solución y su costo la cota inferior antes calculada. Si no es así, pasar al Bloque C.

BLOQUE C. Calcular los costos de sustitución c(x,y), de los arcos de costo mínimo (cero) y retener el máximo (cualquiera en caso de igualdad). Pasar al bloque D.

BLOQUE D. En este momento se sabe que después de la evaluación de los costos de sustitución de los ceros de la matriz, queda la posibilidad de:

a) Renunciar a utilizar el arco (x,y), de costo de sustitución mayor.

En este caso, el costo de sustitución será sumado a la cota inferior calculada anteriormente.

b) Utilizar el arco (x,y) , y examinar las consecuencias de esta selección.

Así se desarrollará la arborescencia, por la creación de un nodo de tipo I: No (x,y) y de un nodo de tipo II: (x,y). Como se dijo en a), el valor de un nodo de tipo I , será:

Cota inferior: igual a cota inferior más c(x,y)

Pasar al bloque E.

BLOQUE E.

1) Si se utiliza el arco (x, y), se suprime el renglón "x" y la columna "y" de la matriz de costos reducidos; posteriormente, por la introducción de costos infinitos, prohibir los circuitos parásitos (arcos que no pueden ser tomados sin impedir la continuación de la búsqueda de un circuito Hamiltoniano).

2) Verificar si hay en la matriz resultante un cero por línea y un cero por columna, si no es así, restar el elemento mas pequeño de los renglones y columnas que no tengan ceros.

3) La cota del nodo (x, y) de la arborescencia será:

cota inferior:= cota inferior + (suma de los elementos restados en 2) del Bloque E).

4) Si se está en una matriz de 1 x 1 parar (se ha llegado a la solución), si no es así, pasar al bloque F.

BLOQUE F.

1) Revisar las cotas de los nodos pendientes de la arborescencia y escoger el nodo de cota mínima (cualquiera si hay mas de un mínimo).

2) Si el nodo es de tipo II, ir al bloque C, si no, ir al Bloque G.

BLOQUE G.

Prohibir el arco (x, y), escribiendo un costo infinito en el cruce correspondiente de la matriz de costos reducidos. Después sustraer el elemento más pequeño de la línea "x" y de la columna "y" e ir al Bloque C.

3.2 Ejemplo.

Elegir la ruta óptima entre las poblaciones A, B, C, D, E y F.

Los costos calculados aparecen en la siguiente matriz:

1.
Matriz de costos.

	A	B	C	D	E	F
A	-	6	7	3	1	8
B	7	-	8	2	9	7
C	5	10	-	10	1	3
D	8	6	5	-	5	1
E	3	7	6	7	-	4
F	9	8	8	5	3	-

Las rutas posibles son 120 (5!)

Las sumas de los costos de cada una de las 120 rutas son distintas.

Se busca encontrar la ruta de costo mínimo.

Desarrollo de la solución aplicando el algoritmo de Little et al.

Ejecutando el Bloque A, obtenemos las matrices 2 y 3:

Matriz 2. Sustracción del elemento más pequeño de cada renglón.

	A	B	C	D	E	F
A	-	5	6	2	0	2
B	5	-	6	0	7	5
C	4	9	-	9	0	6
D	7	5	4	-	4	0
E	3	3	2	3	-	0
F	6	5	5	2	0	-

Matriz 3. Sustracción del elemento más pequeño de cada columna (Matriz de costos reducidos).

	A	B	C	D	E	F
A	-	2	4	2	0	2
B	2	-	4	0	7	5
C	1	6	-	9	0	6
D	4	2	2	-	4	0
E	0	0	0	3	-	0
F	3	2	3	2	0	-

Con una cota inferior de: $1+2+1+1+4+3+3+3+2=20$

Al revisar las áreas de la matriz 3, no es posible formar un circuito, empleando solo las áreas de costo cero. Se pasa al Bloque C.

Se calculan los costos de sustitución de las áreas de costo cero de la matriz 3. Por ejemplo:

El costo de sustitución del arco (A, E) se calcula sumando el valor más pequeño del renglón A y de la columna E.

La suma de estos dos valores $2+0=2$

De igual forma se calculan los costos de sustitución, para los otros ceros de la matriz 3, obteniéndose la matriz 4, cuyo costo de sustitución máximo es 4 y corresponde al arco (B, D). Pasamos al Bloque D.

Matriz 4. Costos de sustitución correspondientes a los ceros de la matriz 3.

	A	B	C	D	E	F
A					2	
B				4		
C					1	
D						2
E	1	2	2			0
F					2	

En este momento queda la posibilidad de:

i) Renunciar a usar el arco (B, D).

En este caso, la cota del nodo no(B,D) de la arborescencia será:

cota inferior := cota inferior + C(B,D) = 20 + 4 = 24

ii) Utilizar el arco (B,D), en cuyo caso se suprime el renglón B y la columna D de la matriz 3, se introduce un costo infinito en el lugar correspondiente al arco (D, B). Se obtiene así la matriz 5.

Al verificar que haya un cero por línea y un cero por columna, la matriz queda igual (no es necesario restar ningún elemento).

Matriz 5

	A	B	C	E	F
A	–	2	4	0	2
C	1	6	–	0	6
D	4	∞	2	4	0
E	0	0	0	–	0
F	3	2	3	0	–

La cota del nodo (B, D) será:

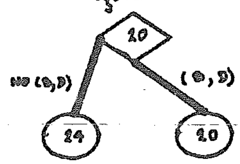
Cota inferior : = cota inferior + (suma de loselementos restados en 2) del Bloque E).

Cota inferior:= 20 + 0 = 20

En este momento se revisan las cota se los nodos pendientes (figura 1) y se elige la mínima; en este caso:

Mínima $\{24, 20\} = 20$, correspondiente al nodo tipo II: (B, D)..

figura 1



De aquí que se ejecute el Bloque C obteniéndose la matriz 6.

Matriz 6

	A	B	C	E	F
A				2	
C				1	
D					2
E	1	2	2		0
F				2	

Tomemos en consideración el arco (A,E) desarrollando la arborescencia a partir del nodo (B,D).

Pasamos al Bloque D.

La cota del nodo no(A,E) es: 20 + 2 = 22

Pasamos al Bloque E.

Para calcular la cota del nodo (A,E) suprimimos la línea A y la columna E de la matriz 5, prohibimos el arco (E,A), de donde resulta la matriz 7.

Matriz 7

	A	B	C	F
C	1	6	–	6
D	4	∞	2	0
E	∞	0	0	0
F	3	2	3	–

Restando uno del renglón C y dos del renglón F, obtenemos la matriz 7 bis, que tiene al - menos un cero por renglón y un cero por columna.

Matriz 7 bis

	A	B	C	F
C	0	5	–	5
D	4	∞	2	0
E	∞	0	0	0
F	1	0	1	–

La cota del nodo (A,E) es:

20 + 3 = 23

Se calcula:

Mínimo { 24,22,23} = 22, correspondiente al nodo no(A,E), para continuar la ar- - borescencia.

Pasamos al Bloque G.

Volvemos a la matriz 5, que es la matriz de costos relativa al nodo ante rior que se acaba de examinar (A, E).

Ejecutando lo indicado en este bloque obtenemos la matriz 8.

Matriz 8

	A	B	C	E	F
A	-	0	2	∞	0
C	1	6	-	0	6
D	4	∞	2	4	0
E	0	0	0	-	0
F	3	2	3	0	-

Pasamos al Bloque C.

Aplicamos lo indicado en este Bloque a la matriz 8, obtenemos así, la matriz 9.

La cota para el nodo no(D, F) es:

22+2 = 24

Matriz 9

	A	B	C	E	F
A		0			0
C				1	
D					2
E	1	0	2		0
F				2	

La cota del nodo (D, F) se calcula aplicando el Bloque E; se suprime el renglón D y la columna F. Se prohibe el arco (F, B). Se obtiene así, la matriz 10.

Matriz 10

	A	B	C	E
A	-	0	2	∞
C	1	6	-	0
E	0	0	0	-
F	3	∞	3	0

La cota para el nodo (D, F) es:

22 + 0 = 22

Se calcula

Mínimo $\{24, 24, 22, 23\} = 22$, correspondiente al nodo tipo II: (D, F).

Se ejecuta el bloque C a la matriz 10, obteniéndose la matriz 11.

Matriz 11

	A	B	C	E
A		2		
C				1
E	1	0	2	
F				3

La matriz 11 designa el arco (F, E) a analizarse.

Matriz 12

	A	B	C
A	-	0	2
C	1	6	-
E	0	∞	0

Matriz 12 bis

	A	B	C
A	-	0	2
C	0	5	-
E	0	∞	0

La cota de no(F, E) es: $22 + 3 = 25$

La cota de (F, E) es: $22 + 1 = 23$

(pasando de la matriz 12 a la matriz 12 bis).

Se calcula

Mínimo $\{24, 24, 25, 23, 23\} = 23$.

Se decide seguir a partir de (F, E), de cota 23.

Matriz 13

	A	B	C
A		7	
C	5		
E	0		2

Se ejecuta el Bloque C obteniéndose la matriz 13. Se designa (A, B) como el arco a analizarse.

La cota de no(A, B) es: $23 + 7 = 30$

La cota de (A, B) es: $23 + 0 = 23$

(matriz 14).

Se calcula

Mínimo $\{24, 24, 25, 30, 23, 23\} = 23$. Se decide continuar a partir de (A, B) de cota 23.

Matriz 14

	A	C
C	0	-
E	∞	0

La matriz 15 permite elegir entre los arcos (C, A) y (E, C). Arbitrariamente se elige (C, A).

Matriz 15

	A	C
C	∞	
E		∞

Cota de no(C, A) es: $23 + \infty = \infty$

Cota de (C, A) es: $23 + 0 = 23$

(matriz 16).

La matriz 16 es de forma 1 x 1: Se debe emplear el arco (E, C).

Tomando (E, C):

Cota de no(E, C) es: $23 + \infty = \infty$

Cota de (E, C) es: $23 + 0 = 23$

Matriz 16

	C
E	0

Se tiene el circuito Hamiltoniano de costo mínimo 23 que es:

B – D – F – E – C – A – B $\qquad (2 + 1 + 3 + 6 + 5 + 6 = 23)$

EN EL DESARROLLO APARECEN SUMAS DE COSTOS IGUALES.

Existe otro circuito Hamiltoniano usando el arco (A,E) de cota 23. Regresamos a la matriz 7 bis, evaluamos los costos de sustitución de los ceros y se-elige el arco (C, A).

Matriz de costos de sustitución

	A	B	C	F
C	6			
D				2
E		0	1	0
F		1		

Se tiene enseguida:

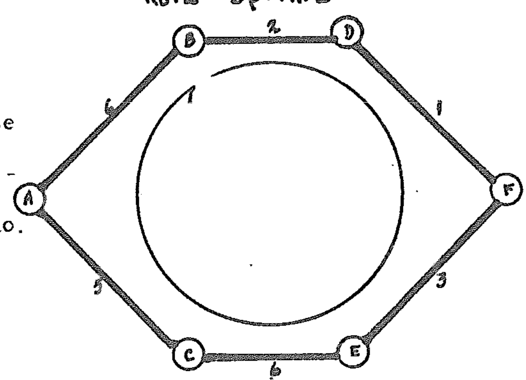
Cota de no(C,A) es: $23+6=29$

Cota de (C , A) es: $23+1=24$

	D	C	F
D	∞	2	0
E	0	∞	0
F	0	1	–

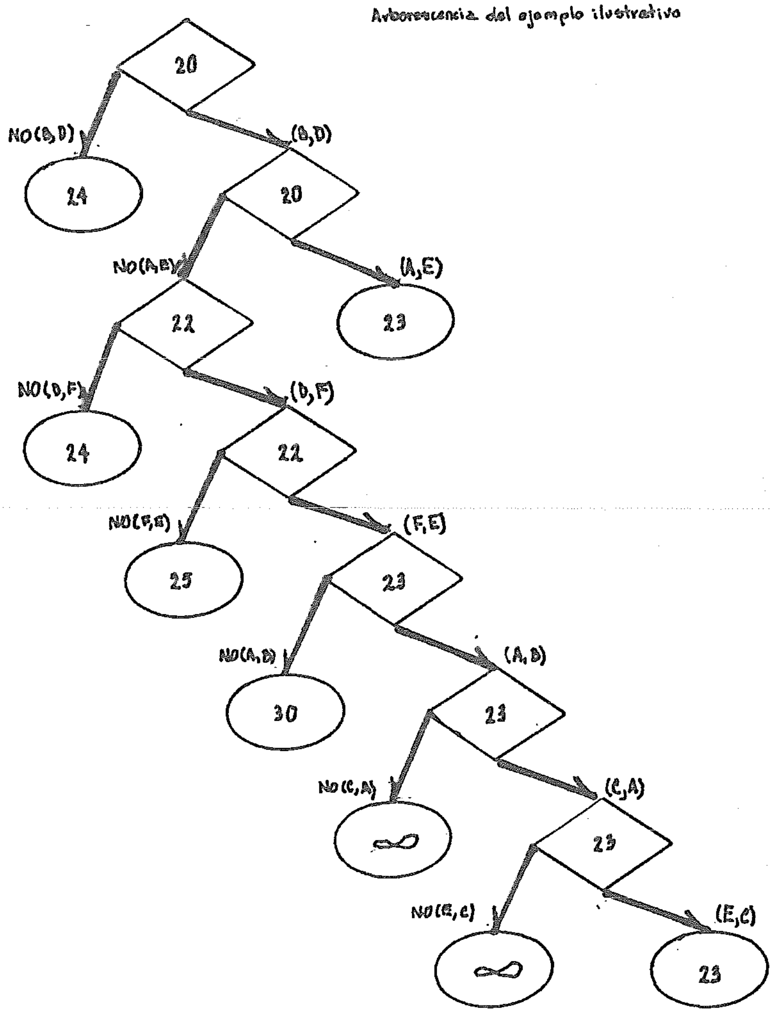
De aquí que todas las otras cotas de los nodos pendientes de la arborescencia son estrictamente mayores que 23. Esto indica que el circuito encontrado es óptimo.

Rota óptima



La arborescencia que enseguida se muestra es el resultado de la ejecución de los Bloques del algoritmo.

Arborescencia del ejemplo ilustrativo



CAPITULO 4.

CASO PRACTICO.

Distribución de medicamentos a las farmacias de la Secretaría de Salubridad y Asistencia.

Metodología:

Se hizo la localización de almacenes y farmacias en la República Mexicana. Tomando como base los datos anteriores, se estableció el radio de acción de cada almacén con relación a las farmacias. Dicho de otra manera, para cada farmacia se determinó el almacén más cercano, según la red de carreteras 21/, 22/ y 23/.

Una vez determinadas las farmacias a surtir por cada almacén, se elaboró la matriz de distancias de las localidades en que está ubicado el almacén correspondiente.

Después de establecer las matrices de distancias, que en este caso resultaron simétricas, (la distancia de la localidad A a la localidad B es igual a la distancia de B a A), la información se encontró estructurada para aplicar un algoritmo y determinar el orden en que conviene entregar medicamentos a cada una de las farmacias consideradas anteriormente.

Se seleccionó el algoritmo de Little et al.

El programa de computadora se elaboró en lenguaje FORTRAN y se utilizó una computadora CYBER 7071.

Para ejemplificar la bondad del procedimiento se puede mencionar como ejemplo las localidades atendidas por el Almacén de Oaxaca.

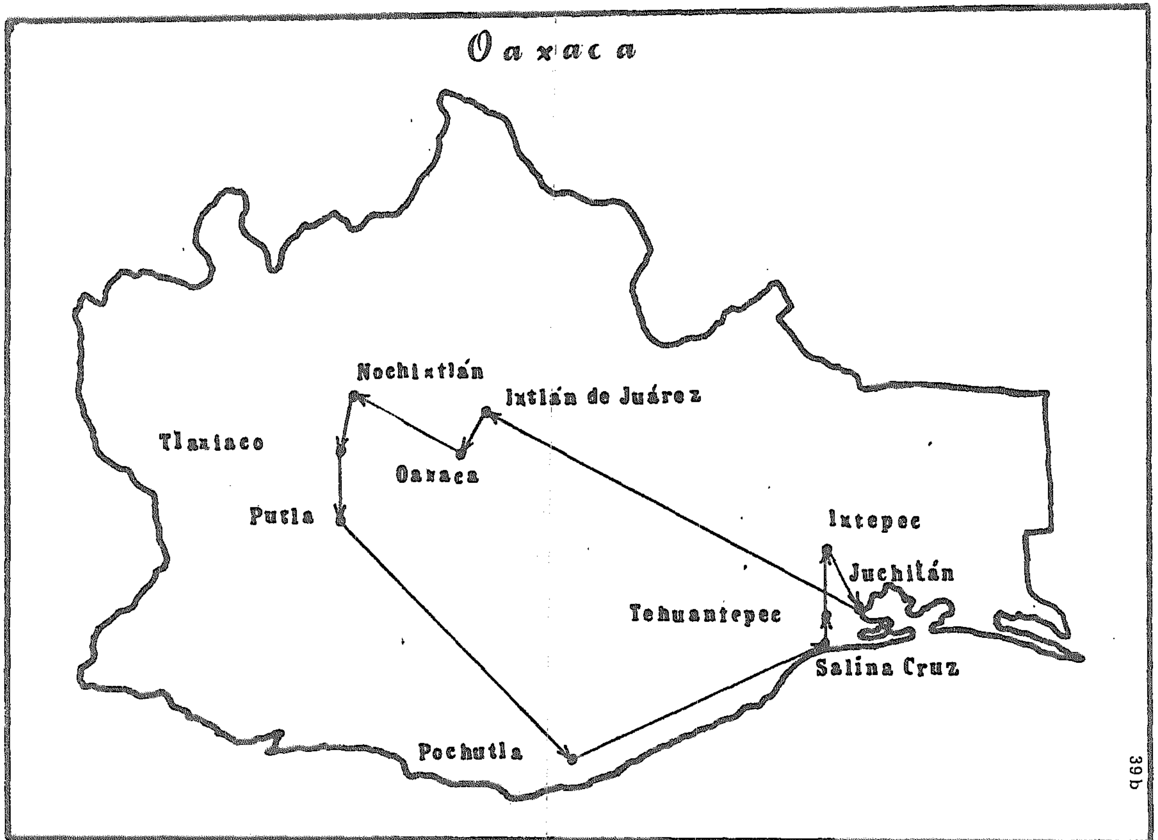
Las localidades atendidas por el Almacén de Oaxaca son nueve.

Las rutas posibles son: 362 880 (9!)

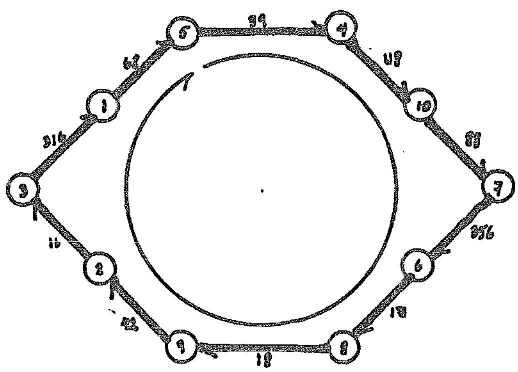
El resultado obtenido para el Almacén de Oaxaca, se presenta en la siguiente tabla:

	Nombre de la localidad:	Número de orden de visita.
1	Ixtlán de Juárez	9
2	Ixtepec	7
3	Juchitán	8
4	Nochixtlán	1
5	OAXACA	10
6	Pochutla	4
7	Putla	3
8	Salina Cruz	5
9	Tehuantepec	6
10	Tlaxiaco	2

El valor mínimo de este circuito es: 1,227 km.


Oaxaca
Nochixtlán
Ixtlán de Juárez
Tlaxiaco
Oaxaca
Putla
Ixtepec
Juchitán
Tehuantepec
Salina Cruz
Pochutla

A continuación se muestra una representación gráfica del circuito obtenido señalando las distancias que separan a las localidades en cada línea. El número de cada círculo corresponde al número de cada localidad.



CAPITULO 5.

EXPERIENCIA COMPUTACIONAL DEL CASO PRACTICO.

Con el propósito de estimar el tiempo esperado en la solución de - problemas de diferentes magnitudes, se efectuaron pruebas con el algoritmo. Los datos para las pruebas, fueron producidos aleatoriamente, tomando números de tres dígitos. La dimensión de las matrices fue desde $n = 4$, hasta $n = 20$.

El número de pruebas con matrices de cada tamaño fue de 10, a causa del aumento del tiempo de cómputo requerido en obtener soluciones, al variar el tamaño de las matrices probadas.

Con los tiempos medios (medidos en segundos), de cada prueba (desde - $\bar{t}=1$, hasta $\bar{t}=17$), se buscó una función que representara al fenómeno. La función que obtuvo el mejor coeficiente de correlación (0.99), fue:

$$\bar{t} = ae^{bn} \quad ; \qquad \text{donde:} \quad \begin{array}{l} a = 0.02156 \\ b = 0.3542 \\ n = \text{número de ciudades.} \end{array}$$

La figura 1, muestra los tiempos de cómputo promedio y la curva ajustada.

A partir del ajuste, se extrapolan valores de tiempo promedio para matrices de diferentes órdenes. A continuación se muestran los valores:

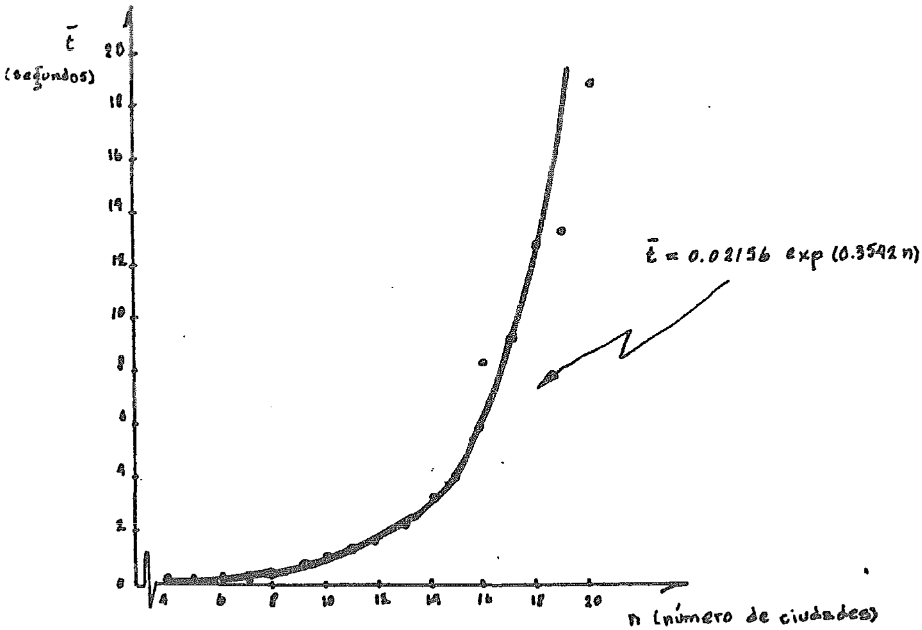


fig.1 Tiempo de cómputo promedio en una CYBER 7071, para datos de 3 dígitos generados aleatoriamente.

n	$\bar{t}$ estimado (segundos)
25	152
30	889
35	5220
40	30672
45	180248
50	1059264

Por lo tanto la ecuación antes indicada, puede ser un auxiliar muy útil para calcular los tiempos de computadora, esperados en la solución de un problema específico.

C O N C L U S I O N E S

Explorar en la compleja dimensión de lá cultura buscando nuevos puntos de partida a las soluciones de las necesidades sociales, nos lleva sin remedio al mundo de la tecnología. Este ámbito muestra dos aspectos destacados; el primero, un gran número de caminos que se pueden tomar para la solución de cada necesidad, el segundo es la velocidad que se incrementa cada día, en la creación de nuevos caminos.

La aparición de las computadoras ha hecho posible la solución a problemas que antes ni siquiera se imaginaban. Por otra parte, cada día aparecen innovaciones a los sistemas, que hacen caer en la obsolescencia, cada vez con mayor rapidez, los métodos en uso.

Este trabajo de investigación nos permitió ver con mayor claridad lo antes mencionado. Existen muchas maneras de solucionar el problema del "Agente viajero". Por ello, una parte que considero muy importante del trabajo fue la selección de un camino, dentro de los muchos que existen. La solución obtenida y aplicada al establecimiento de una ruta óptima para surtir medicamentos del Almacén de Oaxaca, modifica el sistema actual, que consiste en surtir cada farmacia cuando solicita medicamentos, con todas las consecuencias inherentes. La distancia recorrida para surtir a todas las farmacias alcanza 3716 km cantidad tres veces mas grande que 1227 km, resultado de optimizar el sistema.

Otro resultado importante de esta investigación es el encuentro con posibilidades que pueden facilitar la solución de problemas en nuestro país. Utilizar estas posibilidades aumentarían la eficiencia y la eficacia del trabajo como factor de la producción. Desafortunadamente son pocas las empresas que utilizan la tecnología moderna y menos aún las que destinan recursos para investigación, proporcionalmente hablando.

El otro aspecto es que existe un campo propicio para desarrollar y poner en práctica, los resultados de la investigación, adaptarlos a las necesidades de nuestro país, e iniciar nuevos puntos de partida en el campo de la investigación, desde los puntos en que otros han finalizado.

Utilizar nuestras mejores aptitudes y lo mejor de otros, abre un horizonte necesariamente optimista a nuestro país.

R E F E R E N C I A S

I LIBROS.

1 Berge, Claude: Graphes et hypergraphes, Dunod Université, 2 ème édition.

2 Hillier, Frederick S., y Lieberman, Gerald: Operations Research, Ed. Holden Day, 2nd. edition.

3 Taha, Hamdy A.: Operations Research an introduction, Ed. Collier Mc. Millan, 2nd. edition.

4 Blumenkron Hernández, Edgar: Estudio de algoritmos heurísticos para la solución al problema del agente viajero, Tesis UNAM - 1975.

5 Prawda, Juan: Métodos y Modelos de Investigación de Operaciones, - Vol. 1, Modelos determinísticos, 3a. reimpresión, 1981.

6 Faure, Robert: Précis de recherche opérationnelle, Dunod Décision, 1978.

II ARTICULOS.

7 Dantzig G., Fulkerson R., y Johnson S.: "Solution of a large scale traveling-salesman problem", ORSA, Vol. 2, pp 393-410 - (1952-1954).

8 Flood Merrill: "The traveling-salesman problem", ORSA, Vol. 4,- pp 61-75, 1956.

9 Little J., Murty K., Sweeney D., y Karel C.: "An algorithm for - the traveling salesman problem" ORSA, Vol. 11, pp 972-989,- 1963.

10 Gilmore P., y Gomory R.: "Sequencing a one state-variable Machine: A Solvable Case of the Traveling-Salesman Problem", ORSA, Vol. 12, pp 655-679, 1964.

11 Lin Shen: "Computer Solutions of the Traveling Salesman Problem", BSTJ, Vol. 44, pp 2245-2269, 1965.

12 Bellmore M., y Nemhauser G.: "The traveling-salesman problem: a Survey", ORSA, Vol. 16, pp 538-557, 1968.

13 Akkanad M., y Turban E.: "Some comments on the Traveling-Salesman Problem", ORSA, Vol. 17, pp 543-545, 1969.

14 Bellmore M., y Malone J.: "Pathology of Traveling-Salesman Subtour elimination algorithms", ORSA, Vol. 19, pp278-307, 1971.

15 Christofides Nicos:" Bounds for the Travelling-Salesman Problem", ORSA, Vol.20, pp 1044-1056, 1972.

16 Lin S., y Kernighan W.: "An Effective Heuristic Algorithm for the Traveling-Salesman Problem", ORSA, Vol. 21, pp 498-516, 1973.

17 Picard J., y Queyranne M.: "The Time-Dependent Traveling-Salesman and it Application to the Tardiness Problem in One-Machine Scheduling", ORSA, Vol. 26, pp 86-110, 1978.

18 Hong S., y Padberg M.: "A Note on the Symmetric Multiple Traveling Salesman Problem with Fixed Charges", ORSA, Vol. 25, pp 871-875, 1977.

19 Steele J. Michael: "Complete Convergence of Short Paths and Karp's Algorithm for the TSP", Stanford University CA, Dept. of Statistics, 1980.

20 Ali I., y Kennington J.: "The m-Traveling Salesman Problem", Texas University at Austin, Center for Cybernetic Studies, 1980.

III MAPAS

21 Mapa de carreteras. Dirección de Programación, SOP, 1976.

22 Mapa turístico de carreteras, SAHOP 1979.

23 Mapa de los Estados, Serie Patria.

ANEXO 1

LOCALIDADES CON FARMACIAS A SURTIR POR EL ALMACEN DE OAXACA.

	Nombre de la Localidad.
1	Ixtlán de Juárez
2	Ixtepec
3	Juchitán
4	Nochixtlań
5	OAXACA
6	Pochutla
7	Putla
8	Salina Cruz
9	Tehuantepec
10	Tlaxiaco

Matriz de distancias

	Ixtlán de Juárez	Ixtepec	Juchitán	Nochixtlán	OAXACA	Pochutla	Putla	Salina Cruz	Tehuantepec	Tlaxiaco
Ixtlán de Juárez	—	332	316	161	62	249	367	308	290	279
Ixtepec	332	—	16	369	270	231	∞	60	42	487
Juchitán	316	16	—	353	254	215	∞	44	26	471
Nochixtlán	161	369	353	—	99	336	206	346	327	118
OAXACA	62	270	254	99	—	237	305	246	228	217
Pochutla	249	231	215	336	237	—	357	171	289	454
Putla	367	∞	∞	206	305	357	—	463	445	88
Salina Cruz	308	60	44	347	246	171	463	—	18	463
Tehuantepec	290	42	26	327	228	289	445	18	—	445
Tlaxiaco	279	487	471	118	217	454	88	463	445	—

Si $a_{ij} \geq 500$ se consideró como ∞

ANEXO 3

ALGORITMO DE LITTLE ET AL.

PROGRAMA DE COMPUTADORA

COMPUTADORA:	CYBER 7071
NOMBRE DEL PROGRAMA:	HAMIL
LENGUAJE:	FORTRAN
CENTRO DE PROCESAMIENTO DE DATOS:	UNIDAD DE INFORMACION DE LA S S A.

```
      PROGRAM HAMIL(INPUT,OUTPUT)
      COMMON IB(20,20)
      COMMON/PARAM/INF,N,ICOT,ICOJ,IEN
      COMMON/A/IA(20,20)
      COMMON/ARBOL/IAR(5000),IOP(2000),IAP(100)
      READ *,N,INF
      ICOJ=INF+1
      ICOT=100*ICOJ
      IEN=ICOT*100
      CALL LEEMA
      CALL ACASIR(ICOTA,IA)
      CALL COPIA
      CALL RUTJL
      CALL COSU(LCO)
      CALL DECODI(LCO,LL,I,J,MCO)
      IAR(1)=LCO+ICOTA
      CALL MENOS1(I,J)
      IB(J,I)=INF
      CALL ACASIR(MCOTA,IB)
      IAR(2)=LCO+ICOTA-MCO+MCOTA
      INUL=3
      MINAC=INF
   45 MINI=INF
      IF(INUL.GT.5000.OR.NODO.GT.2000) STOP
      IM1=INUL-1
      DO 22 I=1,IM1
      DO 2 J=1,NODO
    2 IF(I.EQ.IOP(J)) GO TO 22
      CALL DECODI(IAR(I),LL,M,L,MCO)
      IF(MCO.GT.MINI) GO TO 22
      IEFE=I
      MINI=MCO
   22 IFLAG=0
      IPAPA=IEFE
      IO=1
      NODO=NODO+1
      IOP(NODO)=IEFE
      LARCO=0
  220 CALL DECODI(IAR(IEFE),IE,I,J,MC)
      IF(IEFE.EQ.(IEFE/2*2)) LARCO=LARCO+1
      IAP(IO)=IEFE
      IF(IE.EQ.0) GO TO 2222
      IEFE=IE
      IO=IO+1
      GO TO 220
 2222 CONTINUE
      CALL COPIA
      DO 4433 MN=1,IO
      IF(IAP(MN).NE.(IAP(MN)/2*2)) GO TO 4433
      CALL PARA(IO,LARCO,MN)
 4433 CONTINUE
      LARCO=0
      DO 111 I=1,IO
      M=IO-I+1
      CALL DECODI(IAR(IAP(M)),LL,NN,MM,NC)
      IF(IAP(M).NE.(IAP(M)/2*2)) GO TO 110
      LARCO=LARCO+1
```

```
PROGRAM HAMIL        73/173   OPT=1                         FTN 4.6+460        1

           CALL MENOS1(NN.MM)
           IB(MM.NN)=INF
           CALL ACASIB(MA.IB)
           GO TO 111
      110  IB(NN.MM)=INF
           CALL ACASIB(MA.IB)
      111  CONTINUE
           IF(LARCO.EQ.N) GO TO 455
           CALL DECODI(IAR(IAP(1)).LL.MA.ME.MINAC)
           CALL COSU(LCO)
           CALL DECODI(LCO.LL.I.J.MCO)
           IAR(INUL)=LCO+MINAC+IEN*IPAPA
           IF(MCO.EQ.INF) IAR(INUL)=LCO+IEN*IPAPA
           CALL MENOS1(I.J)
           IB(J.I)=INF
           INUL=INUL+1
           IO=IO+1
           LARCO=LARCO+1
           IAP(IO)=INUL
           IAR(INUL)=IAR(INUL-1)
           DO 898 MM=1.IO
      898  IF(IAP(MM).EQ.(IAP(MM)/2*2)) CALL PARA(IO.LARCO.MM)
           CALL ACASIB(MCOTA.IB)
           IAR(INUL)=IEN*IPAPA+ICOI*I+ICOJ*J+MINAC+MCOTA
           INUL=INUL+1
           GO TO 45
      455  PRINT 902
           DO 4554 I=1.IO
      902  FORMAT(1H1./////.10X.*<<<<<<<<<<LA RUTA OPTIMA ES :<<<<<<<<<*.///)
           IF(IAP(I).NE.(IAP(I)/2*2)) GO TO 4554
           CALL DECODI(IAR(IAP(I)).LL.K.J.MCO)
           PRINT 1011.K.J
     1011  FORMAT(25X.2(I2.2X))
     4554  CONTINUE
           CALL DECODI(IAR(IAP(1)).LL.K.J.MCO)
           PRINT *.≠<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<≠
           PRINT *.≠<<<<<<<<<<<<<<<< EL COSTO ES: ≠.MCO.≠ <<<<<<<<<<<<<<<<<<<<<≠
           PRINT *.≠<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<≠
           PRINT *.≠EL NUMERO DE NODOS GENERADOS FUE :≠.(INUL-1)
           END
```

SUBROUTINE MENOS1 73/173 OPT=1

```
      SUBROUTINE MENOS1(I,J)
      COMMON IB(20,20)
      COMMON/PARAM/INF,N,ICOI,ICOJ,IEN
      DO 1 L=1,N
      IB(I,L)=-1
      IB(L,J)=-1
    1 CONTINUE
      RETURN
      END
```

SUBROUTINE COSU 73/173 OPT=1

```
      SUBROUTINE COSU(LCO)
      COMMON IB(20,20)
      COMMON/PARAM/INF,N,ICOI,ICOJ,IEN
      IM=0 $ JM=0
      ICOCO=0
      DO 1 I=1,N
      DO 1 J=1,N
      IF(IB(I,J).NE.0) GO TO 1
      LX=INF
      LY=LX
      DO 10 L=1,N
      IF(L.EQ.J.OR.L.EQ.I) GO TO 10
      IF(IB(I,L).GE.0) LX=MIN0(LX,IB(I,L))
      IF(IB(L,J).GE.0) LY=MIN0(LY,IB(L,J))
   10 CONTINUE
      IF(LY.EQ.INF.OR.LY.EQ.INF) GO TO 20
      ICO=LX+LY
      IF(ICO.LE.ICOCO) GO TO 1
      ICOCO=ICO
      IM=I
      JM=J
    1 CONTINUE
      IF(IM.EQ.0) GO TO 2
      LCO=ICOI*IM+ICOJ*JM+ICOCO
      RETURN
   20 LCO=ICOI*I+ICOJ*J+INF
      RETURN
    2 DO 22 I=1,N
      DO 22 J=1,N
   22 IF(IB(I,J).EQ.0) GO TO 222
  222 LCO=ICOI*I+ICOJ*J
      RETURN
      END
```

SUBROUTINE COPIA 73/173 OPT=1

```
      SUBROUTINE COPIA
      COMMON /A/IA(20,20)
      COMMON IB(20,20)
      COMMON/PARAM/INF,N,ICOI,ICOJ,IEN
      DO 1 I=1,N
      DO 1 J=1,N
    1 IB(I,J)=IA(I,J)
      RETURN
      END
```

SUBROUTINE DECODI 73/173 OPT=1

```
      SUBROUTINE DECODI(ICO,IA,I,J,LCO)
      COMMON/PARAM/INF,N,ICOI,ICOJ,IEN
      IA=ICO/IEN
      I=(ICO-IA*IEN)/ICOI
      J=(ICO-IA*IEN-I*ICOI)/ICOJ
      LCO=ICO-IA*IEN-I*ICOI-J*ICOJ
      RETURN
      END
```

SUBROUTINE ACASIB 73/173 OPT=1

```
      SUBROUTINE ACASIB(ICOTA,IB)
      INTEGER XMIN
      DIMENSION IB(20,20)
      COMMON/PARAM/INF,N,ICOI,ICOJ,IEN
      ICOTA=0
      DO 1 J=1,N
      XMIN=INF
      DO 2 I=1,N
      IF(IB(J,I).LT.0) GO TO 2
      XMIN=MIN0(XMIN,IB(J,I))
    2 CONTINUE
      IF(XMIN.EQ.INF) XMIN=0
      DO 3 I=1,N
      IF(IB(J,I).EQ.INF.OR.IB(J,I).LE.0) GO TO 3
      IB(J,I)=IB(J,I)-XMIN
    3 CONTINUE
    1 ICOTA=ICOTA+XMIN
      DO 4 I=1,N
      XMIN=INF
      DO 5 J=1,N
      IF(IB(J,I).LT.0)GO TO 5
      XMIN=MIN0(XMIN,IB(J,I))
    5 CONTINUE
      IF(XMIN.EQ.INF) XMIN=0
      DO 6 L=1,N
      IF(IB(L,I).EQ.INF.OR.IB(L,I).LE.0) GO TO 6
      IB(L,I)=IB(L,I)-XMIN
    6 CONTINUE
    4 ICOTA=ICOTA+XMIN
      RETURN
      END
```

```
SUBROUTINE RUTJL        73/173  OPT=1

      SUBROUTINE RUTJL
      COMMON IB(20,20)
      COMMON/PARAM/ INF,N,ICOI,ICOJ,IEN
      DIMENSION IND(60)
      IND(1)=1
      IVAR=1
      K=1
      J=1
   40 I=IVAR
   70 IF(K-1)20,20,15
   15 DO 25 L=1,K
   25 IF(IND(L).EQ.I)GO TO 10
   20 IF(IB(J,I).NE.0)GO TO 10
      K=K+1
      IND(K)=I
      J=I
      I=0
      IF(K.EQ.N)GO TO 30
   10 I=I+1
      IF(I.GT.N)GO TO 60
      GO TO 70
   60 IF(K.EQ.N)GO TO 30
   61 IF(K.EQ.1)GO TO 31
      IF(IND(K).EQ.N)GOTO62
      IVAR=IND(K)+1
      K=K-1
      J=IND(K)
      GO TO 40
   62 K=K-1
      GOTO61
   30 IF(IB(J,1).EQ.0)GOTO88
      IF(IND(2).LT.N) GOTO61
   31 RETURN
   88 PRINT *,≠LA RUTA ES : ≠,(IND(KK),KK=1,N)
      STOP
   99 CONTINUE
      RETURN
      END
```

```
SUBROUTINE LEEMA        73/173   OPT=1

      SUBROUTINE LEEMA
      COMMON/A/IA(20,20)
      COMMON/PARAM/INF,N,ICOI,ICOJ,IEN
      X=SECOND(CP)
      CALL RANSET(X)
      DO 1 I=1,N
      DO 1 J=1,N
      X=RANF(X)
      IA(I,J)=X*1000
    1 IF(IA(I,J).EQ.0) IA(I,J)=1
      DO 2 I=1,N
    2 IA(I,I)=INF
      RETURN
      END
```

```
SUBROUTINE ESCPI        73/173   OPT=1

      SUBROUTINE ESCPI(N,IA)
      DIMENSION IA(20,20)
      DO 1 I=1,N
      PRINT 5555,(IA(I,J),J=1,N)
 5555 FORMAT(/,11X,10(3X,I3))
    1 CONTINUE
      RETURN
      END
```

SUBROUTINE PARA 73/173 OPT=1


```
      SUBROUTINE PARA(IO,LARCO,I)
      COMMON IB(20,20)
      COMMON/PARAM/INF,N,ICOI,ICOJ,IEN
      COMMON/ARBOL/IAR(5000),IOP(2000),IAP(100)
      NF=0
      MF=0
      CALL DECODI(IAR(IAP(I)),MI,NI,NJ,MA)
      DO 10 M=1,IO
      NARCO=0
      IF(IAP(M).NE.(IAP(M)/2*2)) GO TO 10
      CALL DECODI(IAP(IAP(M)),MI,NN,MM,MA)
      IF(NJ.EQ.NN) GO TO 21
      IF(NI.NE.MM) GO TO 10
      IB(NJ,NN)=INF
      LO=1
  770 IF(LO.GT.IO) GO TO 101
      IF(IAP(LO).NE.(IAP(LO)/2*2)) GO TO 300
      CALL DECODI(IAR(IAP(LO)),LL,NL,ML,MA)
      IF(NN.NE.ML) GO TO 300
      NARCO=NARCO+1
      IF(NARCO.EQ.(N-3)) GO TO 300
      IB(NJ,NL)=INF
      LO=1
      NN=NL
      GO TO 770
  300 LO=LO+1
      GO TO 770
   21 CONTINUE
      IB(MM,NI)=INF
      LO=1
  660 IF(LO.GT.IO) GO TO 102
      IF(IAP(LO).NE.(IAP(LO)/2*2)) GO TO 200
      CALL DECODI(IAR(IAP(LO)),LL,NL,ML,MA)
      IF(MM.NE.NL) GO TO 200
      NARCO=NARCO+1
      IF(NARCO.EQ.(N-3)) GO TO 200
      IB(ML,NI)=INF
      LO=1
      MM=ML
      GO TO 660
  200 LO=LO+1
      GO TO 660
  101 NF=NN
      GO TO 10
  102 MF=MM
   10 CONTINUE
      IF((NF*MF).EQ.0) GO TO 999
      IF(LARCO.EQ.(N-1)) GO TO 999
      IB(MF,NF)=INF
  999 RETURN
      END
```