01168
fg. 1

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

DIVISION DE ESTUDIOS DE POSGRADO
FACULTAD DE INGENIERIA

PROBLEMA DEL VIAJERO Y EXTENSIONES

Créditos asignados a la tesis DOCE (12)
letra y número

APROBADO POR EL JURADO

Presidente: DR. MIGUEL COBIAN SELA

Vocal: DR. SERGIO FUENTES MAYA

Secretario: M. en I. ARTURO FUENTES ZENON

Suplente: M. en I. FRANCISCO ALVAREZ CASO

Suplente: M. en I. RUBEN TELLEZ SANCHEZ

CIUDAD UNIVERSITARIA MEXICO 20, D.F. APDO. POSTAL 70-256 TEL. 548-58-77 550-52-15 EXT. 4155

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Universidad Nacional
Autónoma de México

DIVISION DE ESTUDIOS DE POSGRADO
FACULTAD DE INGENIERIA

Profr. SERGIO FUENTES MAYA
P r e s e n t e

Comunico a usted que a propuesta del SUBJEFE DEL AREA DE SISTEMAS ha sido designado como director de tesis del alumno(a) YOU JIANYU para obtener el grado de M EN I EN INV. DE OPERACIONES.

Mucho he de agradecerle su comunicación, por escrito, de la aceptación a esta designación y el nombre de la tesis a desarrollar.

Atentamente,
"POR MI RAZA HABLARA EL ESPIRITU"
Cd. Universitaria a 7 de noviembre de 1985
EL JEFE DE LA DIVISION

DR. GABRIEL ECHAVEZ ALDAPE

E.S.1

RESUMEN

A partir de la Segunda Guerra Mundial, la rama de investigación de operaciones ha tenido grandes avances tanto en el campo teórico como en las diversas áreas de aplicación. Muchos problemas se han resuelto satisfactoriamente, sin embargo, la solución de otros problemas es bastante difícil en el sentido de que al crecer la dimensión de éstos, los procedimientos de solución se vuelven laboriosos, y a veces hasta imposibles. Uno de este tipo de problemas es el Problema de Viajero analizado en este trabajo.

Un problema típico del agente de viajero es el de un comerciante que desea recorrer N ciudades especificadas con una distancia total mínima. La condición del recorrido es visitar solamente una vez a cada ciudad.

El presente trabajo tiene como propósito el análisis de las bases teóricas, descripción, clasificación, implantación y comparación de diversos métodos de solución del problema de viajero, asimismo la regla de selección de tales métodos para casos reales específicos. Adicionalmente se proporcionan algoritmos eficientes en el caso de viajeros multiples vistos como una generalización inmediata del problema de viajero simple. Con objetivo de facilitar el uso de los algoritmos, estos están implantados en la microcomputadora Apple bajo el sistema operativo de USCD Pascal.

INDICE

INTRODUCCION

A partir de la Segunda Guerra Mundial, la rama de investigación de operaciones ha tenido grandes avances tanto en el campo teórico como en las diversas áreas de aplicación. Muchos problemas se han resuelto satisfactoriamente, sin embargo, la solución de otros problemas es bastante difícil en el sentido de que al crecer la dimensión de éstos, los procedimientos la solución se vuelven laboriosos, y a veces hasta imposibles. Uno de este tipo de problemas es el Problema de Viajero analizado en este trabajo.

Un problema típico del agente de viajero es el de un comerciante que desea recorrer N ciudades especificadas con una distancia total mínima. La condición del recorrido es visitar solamente una vez a cada ciudad.

En 1934, Hassler Whitney, de la Universidad Princeton, identificó el problema de viajero, y descubrió sorpresivamente, que hay pocos resultados y propiedades matemáticos relacionados a este tipo de problema.

Desde la identificación del problema de viajero, mucha gente ha concentrado su esfuerzo en el desarrollo de soluciones, ya sea por medios analíticos o bien computacionales. Asimismo, se ha utilizado este problema y su solución para resolver problemas más complejos. Entre ellos, podemos mencionar: problema de viajeros multiples, problema de viajero dependiente de tiempo, problema de trayectoria de Euler, etc.

Hoy, se están empleando los métodos de solución del problema de viajero en varios campos de aplicación tales como: problema de distribución, secuenciación de instalaciones, orden-recolección en un almacén, diseño de ruta para camiones escolares, etc. El progreso obtenido se debe en gran parte al desarrollo paralelo de la computadora digital, con sus tremendas capacidades de velocidad de cómputo y almacenaje de información. En efecto, si no hubiera sido por la computadora digital, los problemas de dimensiones grandes de computación no habrían adquirido el estado actual promisorio en todas las clases de ámbitos operacionales. Al mismo tiempo que se llevan los algoritmos a casos reales, se descubren que los métodos exactos desarrollados originalmente (Ramificaciones y Acotamiento, Programación Lineal, Programación Entera) resultan ineficientes, ya que consumen mucho tiempo y memoria de la computadora. Por otra parte, algunos algoritmos inclusive necesitan justificación intuitiva, cosa que es sumamente difícil para una computadora. Por tanto, se ha abierto otro campo de desarrollo: los métodos aproximados o bien denominados heurísticos. Estos algoritmos tienen la ventaja de ser adecuados para resolver algún tipo específico de problemas. Un algoritmo heurístico bien empleado sólo consume el tiempo de computación en orden de n^3, y ocupa el espacio de memoria en orden de n^2, asimismo logra una solución de menos de 2% de diferencia con la óptima. Mientras que la mayoría de los métodos exactos consume tiempo y memoria exponencialmente.

Actualmente existen numerosos métodos heurísticos, y surge la necesidad de clasificarlos para poder utilizarlos adecuadamente. Los criterios de clasificación pueden ser diferentes para distintos autores, pero los que se emplean con más frecuencia son: orden del número de operaciones, y el error en el peor caso. Este último nos indica la cota de diferencia entre la solución proporcionada por un algoritmo heurístico y la óptima. Basando en la clasificación y comparación de los diversos algoritmos, para un problema determinado, se puede seleccionar el método que conduce a la solución con mayor eficiencia.

Una generalización del problema de viajero es el caso en que existen varios viajeros. Es decir, varios agentes viajeros pueden visitar las N ciudades. Por la estructura especial asociada, el problema es mucho más difícil para resolver. No obstante, se presenta en un número de situaciones reales y, consecuentemente, atrae una atención mayor.

El presente trabajo tiene como propósito el análisis de las bases teóricas, descripción, clasificación, implantación y comparación de diversos métodos de solución del problema de viajero, asimismo la regla de selección de tales métodos para casos reales específicos. Adicionalmente se proporcionan algoritmos eficientes en el caso de viajeros multiples vistos como una generalización inmediata del problema de viajero simple. Con objetivo de facilitar el uso de los algoritmos, estos están implantados en la microcomputadora Apple bajo el sistema operativo de USCD Pascal.

El trabajo está organizado de la siguiente manera:

En el capítulo I se formula el problema de viajero y se describen sus propiedades. Se presentan algunas aplicaciones que se pueden tener en diversos campos. Las diferentes formas de conceptualizarlo nos conducen a la formulación del problema en términos de: programación lineal, entera, árbol de expansión mínima y problema de asignación.

En el capítulo II, se describen y analizan los métodos clásicos de solución exacta. En particular, los métodos de ramificación y acotamiento son tratados detalladamente, desde su base teórica hasta las variaciones usuales derivadas del método.

En el capítulo III, se describen y revisan los algoritmos aproximados para resolver el problema de viajero. Se clasifican tales algoritmos según su estructura y se comparan la calidad de la solución obtenida, el tiempo consumido, asimismo las ventajas y desventajas. Se deriva simultáneamente la aplicabilidad de los métodos.

En el capítulo IV, se generaliza el problema de viajero simple al caso de viajeros múltiples. Se presentan los resultados desarrollados por varios autores sobre este tema y se proporcionan algunos mejoramientos sobre este género de problemas. Este es el

principal resultado del trabajo. También se discute el aspecto computacional de los algoritmos.

En el capítulo V, se presentan las conclusiones y extensiones del trabajo.

Se anexan dos apéndices. En el apéndice 1, se introducen los conceptos básicos de gráficas y los teoremas fundamentales de redes, necesarios durante todo el desarrollo del trabajo. En el apéndice 2, se explica el uso de cada uno de los programas de computadora así como un bosquejo de la estructura de los mismos.

CAPITULO I EL PROBLEMA DEL VIAJERO Y SUS EXTENSIONES

Considerando que una gran variedad de problemáticas reales se pueden modelar como el problema de viajero, se ha despertado el interés en resolver este problema. Antes de encontrar una solución apropiada, varios autores han tratado de simplificar el problema relacionándolo con los problemas asociados actualmente resueltos. Esto hace que mucha gente haya buscado la solución al problema de viajero formulándolo de diferentes maneras. Dichas formulaciones aprovechan las propiedades de problemas de optimización como: asignación de recursos, árbol de expansión mínima, programación entera y problema de trayectoria más larga, entre otros. De la relación que existe al efectuar estas formulaciones se deducen métodos de solución al problema de viajero.

Este capítulo se desarrolla como sigue:

En la sección 1 , se define el problema de viajero y se muestra algunas aplicaciones y extensiones. En la sección 2, se describen las propiedades más importantes del modelo asociado al problema de viajero, y se establece la equivalencia entre la versión ruta y la versión circuito usada en la formulación del problema. En la sección 3, se presenta la reformulación de este problema y se discuten los resultados analíticos relacionados con los problemas de asignación, árbol de expansión mínima, programación entera, y trayectoria más larga.

1.1 DESCRIPCION DEL PROBLEMA DE VIAJERO CLASICO

El Problema de Viajero consiste en: dado un conjunto finito N de nodos (i.e., ciudades, estaciones, etc.) y la matriz de "costos" o "distancias" entre cada par de nodos, se pretende minimizar la longitud de aquella ruta que visita todos los nodos al menos una vez con mínimo costo. Específicamente el problema consiste en:

$$\text{Min } Z(t) = \sum_{(i,j)\in t} C_{ij}$$

donde t representa una ruta, esto es, una sucesión de arcos de la forma

$$(i_1,i_2),\ (i_2,i_3),\ldots,(i_p,i_1)$$

donde k, k=1,2,...,p denota los nodos visitados por la ruta, y cada nodo 1,2,3,...,n es visitado al menos una vez.

En términos de la teoría de gráficas, el problema del viajero es sencillo de explicar si introducimos el concepto de *ruta hamiltoniana*, esto es, una sucesión de p+1 enteros en X=(1,2, ...,n) en la cual cada uno de los elementos de X aparece al menos una vez y donde el primer y último elementos son iguales. Observe que la ruta está formada por los pares consecutivos de elementos de la sucesión de p+1 enteros de X. ([4])

Algunas situaciones reales donde podemos identificar el llamado problema del viajero son:

Ejemplo 1.(Problema de Distribuición) Suponga que el almacén central de una compañía desea distribuir materia prima a cada una de sus sucursales y que el costo de distribución de materia prima es proporcional a la distancia total recorrida por el vehiculo repartidor. Se desea determinar la ruta de costo mínimo.

Ejemplo 2.(Secuenciación de Instalaciones) En las industrias químicas y farmacéuticas surge el siguiente problema: Un número de productos es manufacturado usando un recipiente de reacción. Después de cada operación de producción Pi, el recipiente puede ser limpiado o no, pero antes de que empiece la producción Pj, dependiendo de la combinación de productos (Pi,Pj). El costo de la limpieza es constante. Supongamos que los n productos son manufacturados en un ciclo continuo, esto es, terminando la última operación, empieza nuevamente el ciclo de producción.

El problema que se plantea consiste en definir la sucesión de operaciones de producción Pi (i=1,..,n), de manera de minimizar costos.

I.1 DESCRIPCION DEL PROBLEMA DE VIAJERO CLASICO

El Problema de Viajero consiste en: dado un conjunto finito N de nodos (i.e., ciudades, estaciones, etc.) y la matriz de "costos" o "distancias" entre cada par de nodos, se pretende minimizar la longitud de aquella ruta que visita todos los nodos al menos una vez con mínimo costo. Específicamente el problema consiste en:

$$\text{Min } Z(t) = \sum_{(i,j)\in t} C_{i,j}$$

donde t representa una ruta, esto es, una sucesión de arcos de la forma

$$(i_1,i_2),\ (i_2,i_3)\ldots(i_p,i_1)$$

donde k, k=1,2,...,p denota los nodos visitados por la ruta, y cada nodo 1,2,3,...,n es visitado al menos una vez.

En términos de la teoría de gráficas, el problema del viajero es sencillo de explicar si introducimos el concepto de *ruta hamiltoniana*, esto es, una sucesión de p+1 enteros en X=(1,2, ...,n) en la cual cada uno de los elementos de X aparece al menos una vez y donde el primer y último elementos son iguales. Observe que la ruta está formada por los pares consecutivos de elementos de la sucesión de p+1 enteros de X. ([4])

Algunas situaciones reales donde podemos identificar el llamado problema del viajero son:

Ejemplo 1.(Problema de Distribuición) Suponga que el almacén central de una compañía desea distribuir materia prima a cada una de sus sucursales y que el costo de distribución de materia prima es proporcional a la distancia total recorrida por el vehículo repartidor. Se desea determinar la ruta de costo mínimo.

Ejemplo 2.(Secuenciación de Instalaciones) En las industrias químicas y farmacéuticas surge el siguiente problema: Un número de productos es manufacturado usando un recipiente de reacción. Después de cada operación de producción Pi, el recipiente puede ser limpiado o no, pero antes de que empiece la producción Pj, dependiendo de la combinación de productos (Pi,Pj). El costo de la limpieza es constante. Supongamos que los n productos son manufacturados en un ciclo continuo, esto es, terminando la última operación, empieza nuevamente el ciclo de producción.

El problema que se plantea consiste en definir la sucesión de operaciones de producción Pi (i=1,..,n), de manera de minimizar costos.

Ejemplo 3.(Problema de Carteros) Considere un cartero que parte de la oficina central a recolectar correspondencia de los buzones que están colocados en diferentes lugares de la ciudad. ¿Cómo debe atravesar la ciudad el cartero para recoger las correspondencias minimizando el tiempo del viaje? Una característica de este problema es que el cartero tiene que pasar AL MENOS una vez por cada uno de los buzones.

Ejemplo 4.(Orden-recolección en un almacén rectangular) Uno de los problemas fundamentales asociados con el manejo de los materiales de un almacén es llamado "Problema de orden-recolección". Una orden es en un conjunto de artículos pedidos al almacén. Cuando se recibe la orden, se despacha un vehículo desde el área de embarque a coleccionar los artículos y regresar al área de embarque. El objetivo es minimizar la distancia recorrida o el tiempo usado por el vehículo.

Conviene señalar que en los ejemplos anteriores es sencillo identificar los nodos que se consideran en la ruta hamiltoniana. En el último caso, es necesario pasar el problema original a uno modificado donde es sencillo identificar la ruta hamiltoniana. Una configuración típica del almacén se muestra en la figura I.1 y su modelado en términos del problema de viajero se muestra en la figura I.2.

Ejemplo 5.(Problema de distribución con varios vehículos repartidores) En el problema de distribución de materias primas del ejemplo 1, puede existir varios vehículos repartidores. Ahora la pregunta es: ¿A cuáles consumidores debe distribuir cada uno de los vehículos? Esta extensión del problema del viajero es un ejemplo simple de problema de viajeros multiples. Dicho problema tiene diferentes variaciones. Algunas de las cuales son: a) todos los vehículos tienen que partir de una misma ciudad base; b) al menos r vehículos tienen que ser usados; c) los vehículos pueden partir de diferentes ciudades.

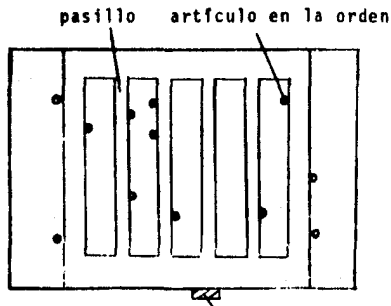


Figura I.1

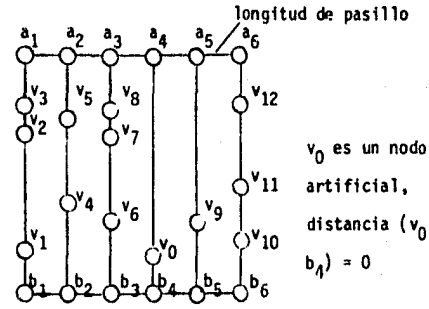


Figura I.2

I.2 PROPIEDADES ESTRUCTURALES DEL PROBLEMA DE VIAJERO

Un aspecto básico del estudio de un problema de optimización es la existencia del soluciones. Usualmente, lo que se desea es determinar las condiciones necesarias y suficientes para la existencia de soluciones. Una vez conocidas éstas, podemos aplicarlas para determinar las soluciones óptimas mediante un procedimiento que sea eficiente. Dicho procedimiento es simplemente un algoritmo.

Una condición necesaria para que el problema del viajero, en una gráfica no dirigida G=[N,A], tenga solución es que G sea 2-conectada, es decir, la cardinalidad de cada nodo es mayor o igual que 2. En el caso de gráficas dirigidas, la condición necesaria es que el grado de entrada y el de salida de cada nodo deben ser mayor o igual que 1. ([2])

La condición anterior se puede generalizar un poco como sigue: Sea G=[N,A] gráfica dirigida. Sea S, $\bar{S}$ una partición no nula de los nodos de N, esto es, S y $\bar{S}$ son no vacios y $S \cup \bar{S}=N$. Defina $x_{i,j}=1$ si $(i,j) \in A$; $x_{i,j}=0$ caso contrario. Entonces una condición necesaria para la existencia de la solución del problema de viajero es:

$$\sum_{i \in S, j \in \bar{S}} x_{ij} \geq 1$$

Conviene señalar que hasta hoy, no se conoce la condición necesaria y suficiente para comprobar eficientemente la existencia de la solución del problema de viajero.

Un caso especial de la ruta hamiltoniana es cuando cada nodo es visitado exactamente una vez. Este caso está reducido al concepto de circuito hamiltoniano. Un *circuito hamiltoniano* (o circuito) es una ruta hamiltoniana en que cada nodo en (1,2,...n) es visitado exactamente solo una vez.

El siguiente resultado garantiza que cada vértice se visita exactamente una vez cuando se cumple cierta condición sobre la matriz de costos.

TEOREMA I.1: Si en el problema de viajero, la matriz de costos C satisface la desigualdad de triángulo, entonces existe un circuito óptimo. ([4])

Prueba: Como C satisface la desigualdad de triángulo, no existe ningún arco (digamos i y j) cuyo costo es infinito; pues la suma de los costos de la travectoria que une i y j tiene que ser mayor que el costo asociado al arco (i,j). Por tanto, la gráfica es completa y existe solución factible (Por ejemplo,{(1,2),(2,3),...(n,1)). Como el número de soluciones factibles es finito, se implica que existe una solución óptima.

Cuando se está resolviendo un problema de optimización, un aspecto importante es saber si se llega a la solución óptima o no. Es posible verificar dicha condición usando la llamada PRUEBA DE OPTIMALIDAD. En nuestro caso, esta prueba se reduce al siguiente resultado cuya demostración es obvia.

TEOREMA I.2: Sean p_1 y p_2 dos permutaciones diferentes de los enteros (2,3,...k+1). Denote tales permutaciones por

$$p_1=(i_1,i_2,\ldots,i_k)$$
$$p_2=(j_1,j_2,\ldots,j_k)$$

y sean los costos totales asociados con los segmentos de las rutas hamiltonianas generadas

$$Z(Pm)= \sum_{(i,j)\epsilon p_m} C_{i,j} \quad m=1,2.$$

Si

$$C_{1i_1} + Z(p_1) + C_{i_k s} < C_{1j_1} + Z(p_2) + C_{j_k s}$$

entonces $(1,p_2,s)$ no puede ser segmento del circuito óptimo.

Observe que la aplicación recursiva de este resultado puede servir de base para determinar el circuito óptimo. ([4])

COROLARIO I.1:

Si C es simétrica y, t_1 y t_2 son dos circuitos en los cuales los vértices son visitados en orden reverso, esto es:

$$t_1=(i_1,i_2,\ldots,i_n,i_1)$$
$$t_2=(i_1,i_n,\ldots,i_2,i_1)$$

entoces $Z(t_1)=Z(t_2)$.

Escribimos un circuito hamiltoniano del siguiente modo:

$$t_1=(i_1,i_2,\ldots,i_n,i_1)$$

Para calcular el número de permutaciones de los vértices que forman un circuito hamiltoniano, se procede como sigue: se debe fijar un vértice, digamos i_1. Observe que para seleccionar i_2, hay n-1 maneras; para seleccionar i_3, hay n-2 maneras; así sucesivamente. En general, hay (n-1)! circuitos, pero para problemas simétricos, sólo hay (n-1)!/2. Si se resuelve el problema de viajero mejorando circuito por circuito, el número de operaciones se crece exponencialmente mientras crece el valor de n, pues $k!>a^k$ cuando k tiende a infinito.

EQUIVALENCIA ENTRE VERSION RUTA Y VERSION CIRCUITO

En la definición del problema de viajero, cada nodo puede aparecer más de una vez en la solución. Sin embargo, casi en todos los algoritmos desarrollados, para su solución, se pide que cada nodo sea visitado EXACTAMENTE UNA VEZ. Equivalentemente, la ruta hamiltoniena óptima es un CIRCUITO. ([6]) Por lo anterior, existen dos versiones de la solución del problema de viajero. En la versión ruta, un nodo aparece una o más veces, mientras que en la versión circuito, el nodo aparece una sola vez.

Los métodos de solución de la versión circuito pueden ser

aplicados a la versión ruta usando un argumento de la siguiente naturaleza. Asociada a la versión ruta de una gráfica G, se puede definir otra gráfica G' con la propiedad de que una ruta óptima en G corresponde a un circuito óptimo en G'. La gráfica G' es construída del siguiente modo:

1. Los nodos de G' son los mismos de G.
2. En G', los nodos i y j son unidos por un arco dirigido (i,j), si y sólo si existe una trayectoria en G denotada
   $\gamma(i,k_1),(k_1,k_2),\ldots,(k_m,j))$.
3. Las longitudes d'(i,j) en G' son las longitudes de la trayectoria más corta entre los nodos i y j de G.

Los siguientes lemas demuestran que la equivalencia de optimalidad entre las gráficas G y G'.

Lema I.2.1: A cualquier circuito factible denotado R en G' corresponde una ruta factible T en G, y las longitudes de T y de R son iguales.

Prueba: Suponga que $C=(i_1=1,i_2=2,\ldots,i_n)$ es un circuito en G', entonces la ruta correspondiente en G es

$$T=(i_1=1,P1,i_2=2,\ldots,Pn,i_{n+1})$$

donde (i_k,Pk,i_{k+1}) es la trayectoria dirigida más corta en G entre i_k y i_{k+1}. Evidentemente T es factible y por la definición de las longitudes en G', las longitudes de T y R son iguales.

Lema I.2.2: Si T es una ruta óptima en G, entonces existe un circuito R en G' tal que T=t(R). Donde t es la transformación que definimos anteriormente.

La prueba es obvia por la definición de t. Evidentemente R no es necesariamente único.

Lema I.2.3: Un circuito R en G' es óptimo si y solo si la ruta T=t(R) es óptima en G.

Prueba: Sean R el circuito óptimo en G', y L su longitud. Supongase que t(R) no es óptimo; entonces existe otra ruta T' tal que T' es óptimo y su longitud L'<L. Por el lema I.2.2, existe un circuito R' tal que t(R')=T'. La longitud de R' es también la de L', así R no es óptimo. Esto es una contradicción.

Reciprocamente, sea T una ruta óptima, y L su longitud. Por el lema I.2.2 existe un circuito R tal que t(R)=T, y por el lema I.2.1 la longitud de R es L también. Supngamos ahora que R no es óptimo, entonces existe otro circuito R' optimo tal quo L'<L. Entonces la longitud de t(R') es L', T no es óptima, lo cual es una contradicción otra vez y la prueba termina.

El lema I.2.3 implica que la versión ruta del problema de viajero puede transformarse en la versión circuito.

I.3 RELACION CON OTROS PROBLEMAS DE OPTIMIZACION

Debido a las propiedades estudiadas en la sección anterior, se observa que el problema de viajero tiene ciertas similitudes con otros problemas de programación matemática. Auxiliándonos de la solución de estas programaciones, se puede obtener la solución del problema mencionado de una manera mas fácil. La relación con otros problemas de optimización también nos ayuda a encontrar ciertos parámetros del problema de viajero tales como: la cota inferior de la función objetivo, la probabilidad de que una solución de otros problemas sea un circuito hamiltoniano, etc. Los casos más usuales de problemas de programación matemática que tienen relación con el problema del viajero se describen a continuación.

I.3.1 ASIGNACION

El problema de asignación está definido como:

$$\begin{aligned} \text{Minimizar } w= & \sum_{ij} c_{i,j}\, x_{i,j} \\ \text{Sujeto a} & \sum_{i} x_{i,j}=1 \quad \forall j=1,\ldots,n \\ & \sum_{j} x_{i,j}=1 \quad \forall i=1,\ldots,n \\ & x_{i,j}=0,1 \quad \forall i,j \end{aligned}$$

Es otras palabras, el problema de asignación se puede explicar como: Se tienen n individuos y n trabajos. Si el individuo i es asignado al trabajo j se incurre en el costo $c_{i,j}$. Se desea encontrar el costo mínimo de asignar los individuos a los trabajos. Observe que si $x_{i,j}=1$ significa que el individuo i ha sido asignado al trabajo j, y que $x_{i,j}=0$ indica que el individuo i no está asignado al trabajo j.

Se puede interpretar la solución del problema mencionado en forma gráfica como sigue: primero se dibujan n nodos en la gráfica. Luego se traza un arco (i,j) si el individuo i es asignado al trabajo j. En tal gráfica, se observa que el grado de entrada de cada nodo es 1 y el de salida también. Esta característica cumple la condición necesaria de la existencia de la solución del problema de viajero. No obstante, en la gráfica pueden existir subcircuitos aislados que no cumplen la condición de una solución de problema de viajero.

Sea t cualquier circuito en el cual cada nodo es visitado exctamente una vez. Entonces $x_{i,j}=0$ si (i,j) no está en el circuito y $x_{i,j}=1$ si (i,j) sí está. Entonces $(x_{i,j})$ es una solución factible del problema de asignación con valor de la función

$$Z(t)= \sum_{i}\sum_{j} c_{i,j}x_{i,j}=w.$$

Obviamente, la inversa de este teorema no es cierta.

El problema de asignación ha sido estudiado por mucha gente. La obtención de la solución es sencilla aplicando algún método existente, por ejemplo: el método húngaro. Esta solución puede ser un circuito factible del problema de viajero. Pero, ¿qué tan probable es que la solución de asignación sea un circuito hamiltoniano? Basando en el siguiente teorema, podemos encontrar esta probabilidad suponiendo que cada elemento de la matriz de costos sea una variable aleatoria.

Sea el problema de asignación modificado

$$\text{Minimizar } w=\sum_i\sum_j c_{i,j}\,x_{i,j}$$

$$\text{Sujeto a } \sum_i x_{i,j}=1 \ \forall j=1,\ldots,n$$

$$\sum_j x_{i,j}=1 \ \forall i=1,\ldots,n$$

$$x_{i,i}=0 \ \forall i$$

donde $c_{i,j}$ son variables aleatorias independientes e idénticamente distribuídas. ([3])

TEOREMA I.3: Supongamos que las soluciones del problema anterior tiene la misma posibilidad de ser óptimas, entonces la probabilidad de que la solución óptima sea un circuito factible es

$$Pn=(n-1)!/[n!e^{-1}+0.5]\simeq e/n \quad \text{para n grande}$$

Prueba: Primero demostramos que el número de soluciones enteras es $[n!e^{-1}+0.5]$.

Si resolvemos el problema:

$$\text{Minimizar } w=\sum_i\sum_j c_{i,j}\,x_{i,j}$$

$$\text{Sujeto a } \sum_i x_{i,j}=\sum_j x_{j,i}=1 \ \forall j=1,\ldots,n$$

existe exactamente n! soluciones factibles.

Consideramos que se obtiene del problema anterior otro problema haciendo $X_{kk}=0$, y eliminando la columna k y el renglón k. El problema modificado obtenido tiene (n-1)! soluciones, y hay $\binom{n}{1}$ maneras para escoger k.

Ahora consideramos el problema obtenido por escoger 2 índices k y l, haciendo $X_{kk}=X_{ll}=0$. De esta manera, vamos a resolver un problema de asignación de tamaño (n-2)x(n-2). El número de soluciones es (n-2)! y hay $\binom{n}{2}$ maneras para escoger k y l.

Aplicando el principio de Inclusión y Exclusión, se obtiene el número de soluciones del problema de asignación modificado que es:

$$n!-\binom{n}{1}(n-1)!+\binom{n}{2}(n-2)!-\binom{n}{3}(n-3)!+\ldots+(-1)^n\binom{n}{n}(n-n)!$$

$$=n!(1-1/1!+1/2!-1/3!+\ldots+(-1)^n/n!)$$

Los términos que están dentro de paréntesis son los primeros n términos de la serie de expansión de e^{-1}. Se puede demostrar que el error de estimación de la expresión es menor que 0.5 para n>1. Por tanto, el número de soluciones es dado por $n!e^{-1}+0.5$ redondeado al entero mas cercano, o sea $[n!e^{-1}+0.5]$.

Pn=(número de circuitos factibles)/(número de soluciones enteras)
$=(n-1)!/[n!e^{-1}+0.5]$

y esto termina la prueba.

I.3.2 ARBOL DE EXPANSION MINIMA

El problema de Arbol de expansión mínima está definido como:

Dada la matriz de costos de una gráfica no dirigida, encontrar el árbol que incluye todos los vértices en la gráfica, y tal que el costo total del árbol sea mínimo.

En una solución factible del problema de viajero versión circuito, el grado de cualquier vértice es 2, pero la solución del árbol de expansión mínima no cumple esta condición aunque la gráfica esté conectada. Sin embargo, se puede obtener la solución del problema de viajero desde este árbol, transformándolo hasta que la cardinalidad de todos los nodos sea 2.

En el caso de una gráfica no dirigida con matriz de costo simétrica, una cota inferior para la solución del problema de viajero es derivada usando el correspondiente árbol de expansión mínima de la gráfica de la siguiente manera. Supongamos que el arco (x_1,x_2) está en el circuito óptimo del problema de viajero. Si este arco es borrado del circuito, se genera una trayectoria de n-1 arcos que recorre todos los vértices empezando de x_1 y terminando en x_2. Como el costo del árbol de expansión mínima L(AEM) es la cota inferior de esta trayectoria, L(AEM) más $c(x_1,x_2)$ es la cota inferior del costo del problema de viajero. ([1])

En general se desconoce (x_1,x_2) en el circuito óptimo, por tanto se selecciona $\max[c(x_1,s)]$, donde s es el vértice segundo cercano a x_1. Entonces una cota inferior válida de la solución del problema de viajero se tiene por medio de

$$L(AEM) + \max_{x_1}[c(x_1,s)]$$

1.3.3 PROGRAMACION ENTERA

El problema de programación entera lineal consiste en:

$$\begin{aligned} &\text{Minimizar } CX \\ &\text{Sujeto a } AX=b \\ &\qquad X\geq 0 \\ &\qquad X \text{ entero,} \end{aligned}$$

donde X es un vector (n x 1), C es un vector (1 x n), b es un vector (n x 1) y A es una matriz (n x n).

Conviene señalar que el problema de viajero denotado por PV puede ser formulado como una programación entera. De esta manera, se puede aplicar cualquier técnica de solución programacion entera para resolver el problema en cuestión. ([4], [8])

La formulación del PV que tiene que visitar exactamente una vez cada una de las n ciudades indexadas 1, 2, ..., n partiendo de la ciudad base 0, y cuyos costos de transporte de una ciudad i a una j, son dados por la matriz $C=(c_{i,j})$ es dado como:

$$\text{Minimizar } z=\sum_i \sum_j c_{i,j}\, x_{i,j}$$

$$\text{Sujeto a} \quad \sum_{i\neq j} x_{i,j}=1 \qquad \forall j$$

(PE1)

$$\sum_{j\neq i} x_{i,j}=1 \qquad \forall i$$

$$x_{i,j}=0,1 \quad \forall i,j$$

$$u_i-u_j+nx_{i,j}\leq n-1 \quad \forall i,j \text{ excepto } i=j=0$$

donde u_i son números reales que permiten la equivalencia de la solución del PV con el problema de programación entera planteado.

La idea de la formulación anterior es como sigue:

El viajero va de la ciudad i a j si y solo si $x_{i,j}=1$. Entonces, lo importante de la formulación consiste en demostrar que se corresponden los dos conjuntos de soluciones factibles, es decir, una solución factible para PE tiene variables $x_{i,j}=1$ que definen un recorrido legítimo en PV, y recíprocamente, un recorrido legítimo de PV define un conjunto de variables $x_{i,j}=1$ que satiface las restricciones de PE1 junto con valores de u_i apropiados.

Con el propósito de establecer esta equivalencia, considere primero una solución factible de PE1. Demostraremos que todos los circuitos incluven el nodo 0. Las restricciones de la forma

$$\sum_j x_{i,j}=1 \quad \forall\, x_{i,j}\geq 0$$

representan las condiciones de que cada ciudad (diferente de 0) es visitada exactamente una vez. Los u_i sirven para eliminar los

circuitos que no empiezan ni finalizan en el nodo 0 y los circuitos que visitan más de n ciudades. Considere cualquier $x_{r0,r1}=1$ ($r_1 \neq 0$). Existe un r2 unico tal que $x_{r1,r2}=1$. A no ser que r2=0, también existe un r3 unico tal que $x_{r2,r3}=1$. Así sucesivamente hasta encontrar un $r_k=r_j$, $j<k-1$. Como ningún r es cero, tenemos que

$$u_{ri} - u_{ri+1} + nx_{ri,ri+1} \leq n-1$$

or

$$u_{ri} - u_{ri+1} \leq -1$$

Sumando de i=j hasta k-1 tenemos que $u_{rj} - u_{rk} = 0 \leq j+1-k$ que es una contradicción. Por tanto, todos los circuitos incluven el nodo 0. Demostraremos ahora que ninguno de los circuitos tiene longitud mayor que n. Supongase que existe un circuito de longitud mayor que n, digamos $x_{0,r1}$, $x_{r1,r2}$, $x_{rn,rn+1}=1$, $r_i \neq 0$. Entonces

$$u_{ri} - u_{rn+1} \leq -n$$

or

$$u_{rn+1} - u_{ri} \geq n$$

pero

$$u_{rn+1} - u_{ri} + nx_{rn+1,ri} \leq n-1$$

or

$$u_{rn+1} - u_{ri} \leq n(1-x_{rn+1,ri}) - 1 \leq n-1$$

que es una contradicción.

Reciprocamente, demostraremos que un recorrido legítimo en el PV define una solución factible de PE1. Si $x_{i,j}$ corresponden a un recorrido legitimo, se puede ajustar u_i tal que $u_i=j$ si i es la j-ésima ciudad que se visita, y

$$u_i - u_j = -1 \text{ si } x_{i,j}=1$$

y siempre se cumple que

$$u_i - u_j \leq p-1.$$

La discusión anterior demuestra el teorema de equivalencia entre las soluciones de PV y PE1:

TEOREMA 4: El circuito óptimo de PV puede encontrarse resolviendo el Programación entera PE1.

Otra formulación equivalente, pero diferente del problema de viajero en términos de Programación entera se presenta a continuación para el caso de matriz de costos simétricos. Antes de presentar el resultado, conviene aclarar el concepto de partición de un conjunto (1, 2, ..., n). Una partición S, $\bar{S}$ es no vacía, si cada conjunto es no vacío y $S \cap \bar{S} = 0$ y $S \cup \bar{S} = (1,\ldots,n)$. En la figura I.3, se muestran unos ejemplos de las particiones.

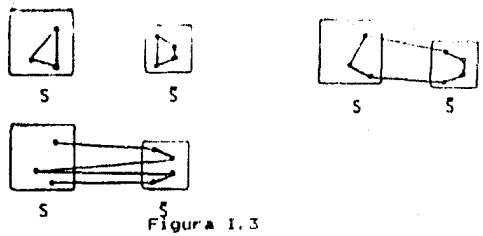


Figura I.3

TEOREMA 5: Sea C una matriz simétrica, sean S, $\bar{S}$ una partición de los enteros i=1,...,n. Por costos simétricos podemos asignar $x_{i,j}=0$ si el arco n dirigido (i,j) no está en un circuito y $x_{i,j}=1$ si el arco no dirigido (i,j) está. Un circuito óptimo puede ser encontrado al resolver el programación entera

PE2

$$\text{Minimizar } Z=\sum_i\sum_j c_{i,j}x_{i,j}$$
$$\text{Sujeto a } x_{i,j}=0,1 \quad (i=1,\ldots,j-1;\ j=2,\ldots,n)$$
$$\sum_{i\in S,\,j\in\bar{S}} x_{i,j}\geq 2$$

para todas las particiones no nulas $(S,\bar{S})$ tal que si $(S,\bar{S})$ está considerado, $(\bar{S},S)$ ya no. (NOTA: especificamente, en una gráfica dirigida, las restrincciones serán

$$\sum_{i\in S,\,j\in\bar{S}} x_{i,j}\geq 1\ )$$

Prueba: Similar a la demostración anterior, se establece la correspondencia de la definición de variables entre el problema de viajero y el problema de programación entera.

La solución del PV evidentemente es una solución factible de PE2. Ya que en la gráfica generada por la solución de PV, la cardinalidad de cada nodo es 2, y la gráfica es conectada, de tal manera que

$$\sum_{i\in S,\,j\in\bar{S}} x_{i,j}\geq 2.$$

En otro sentido de la prueba, nótese que en la solución de PE2, la cardinalidad de cada nodo es mayor que 2, porque se puede particionar S que contiene únicamente i. Si existe un nodo de cardinalidad mayor 2, existe un arco que conecta a otro nodo de cardinalidad mayor que 2 también. Pero por la minimización, este arco no puede existir. Por tanto, cardinalidad de cada nodo es 2 exactamente. Por la misma condición

$$\sum_{i\in S,\,j\in\bar{S}} x_{i,j}\geq 2.$$

la gráfica es conectada. Por tanto, la solución de PE2 también es solución de PV.

I.3.4 TRAYECTORIA MAS LARGA

El problema de travectoria más larga consiste en:

Dado un conjunto finito de nodos (1,2,...,i,...n), las distancias d(i,j) entre cualquier par ordenado (i,j) y dos nodos específicos (digamos 1 y n), encontrar una sucesión

$$t=(i_1=1,i_2,\ldots,i_n)$$

tal que:

i) ningún nodo puede aparecer más de una vez, es decir, la sucesión es una travectoria.

ii) la longitud de esta travectoria es máxima.

El problema de travectoria más larga es mucho más difícil que el de la travectoria más corta, pero similar al problema de viajero. ([6])

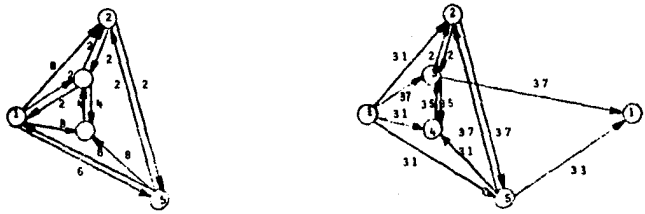
El problema de viajero puede transformarse a un problema de travectoria más larga. Se demostrará que para cada gráfica G, existe una gráfica L(G) de tal manera que la trayectoria más larga en L(G) corresponda a un circuito óptimo en G. Construcción de la gráfica L(G) asociada a la gráfica G del problema de viajero:

i) Los nodos de L(G) son los mismos de G más uno adicional denotado por 1'.
ii) Los arcos de L(G) son los mismos de G, excepto que para los arcos de forma (j,1) en G, se sustituye por (j,1') en L(G).
iii) Las distancias D(i,j) en L(G) están definidas por:

$$D(i,j)=\begin{cases}\infty & \text{si } i=j\\ K-d(i,j) & \text{si } i\neq j \text{ y } j\neq 1'\\ K-d(i,1) & \text{si } i\neq j \text{ y } j=1'\end{cases}$$

donde K una constante estrictamente mayor que S el cual es la suma de los n valores d(i,j) más grandes. La travectoria en L(G) significa la trayectoria entre los nodos 1 y 1' en L(G).

Un ejemplo de la construcción de L(G) se muestra en la figura I.4.



S=8+8+8+8+6=38

Figura I.4

En seguida, demostraremos que existe la gráfica L(G) tal que la trayectoria más larga en L(G) corresponda a un circuito hamiltoniano óptimo en la gráfica G.

TEOREMA 6: Una trayectoria más larga desde 1 a 1' en la nueva red contiene cualquier nodo intermediario (1',...,n), y si denotamos esta trayectoria más larga (1 ,i_1 ,..., i_{n-1}, 1'), (1, i_1 ,..., i_{n-1}, 1) es el circuito óptimo.

Basando en los siguientes lemas, se puede demostrar el teorema.

Lema I.6.1: Cualquier trayectoria más larga en L(G) contiene n arcos, y por lo tanto, todos los nodos de L(G).

Prueba: Sea Lk la longitud de cualquier trayectoria que contiene k arcos, y sea $m \leq n-1$. Entonces por definición de K, tenemos que

$$\max Lm \leq mK \leq (n-1)K = nK-K < nK-S \leq \min Ln$$

por tanto cualquier trayectoria de n' arcos es más larga que cualquier trayectoria de $m<n$ arcos. En particular, cualquier trayectoria más larga tiene n arcos.

Lema I.6.2: Existe una correspondencia uno-uno entre circuitos factibles en G y trayectoria de n arcos en L(G).

Prueba: Sea (i_1=1, i_2,..., i_n) un circuito factible en G, entonces la secuencia (i_1=1, i_2, ..., i_n) en G' obviamente es una trayectoria de n arcos, entonces (i_1=1, i_2, ..., i_n) es un circuito factible en G, y esta correspondencia es uno-uno.

Lema I.6.3: Una trayectoria en L(G) es una trayectoria más larga si y solo si el circuito correspondiente es óptimo.

Prueba: Sean P1 la trayectoria más larga en L(G) y P2 cualquier trayectoria que contiene n arcos (pero no la más larga), Sean L1 y L2 las longitudes de P1 y P2 respectivamente. Por el lema I.6.2, P1 tiene n arcos, y además por el lema I.6.2, existe dos circuitos factibles C1 y C2 en G correspondientes a P1 y a P2 respectivamente. Entonces

$$Lk = nK - \sum_{C_k} d(i,j) \qquad (k=1,2)$$

y

$$0 < L1-L2 = \sum_{C_2} d(i,j) - \sum_{C_1} d(i,j)$$

Por tanto C1 es un circuito óptimo.

En otro sentido, sea C1 un circuito óptimo y C2 cualquier circuito factible (pero no óptimo), sean P1 y P2 trayectorias correspondientes en L(G), y sean L1 y L2 las longitudes de P1 y P2 respectivamnete. Entonces por el lema I.6.2, P1 y P2 ambos contienen n arcos y bajo la hipótesis $d(i.j)<d(i,j)$, tenemos que:

$$L1 = nk - \sum_{C_2} d(i,j) > nk - \sum_{C_1} d(i,j) = L2,$$

y por el lema I.6.2, la trayectoria más larga en L(G) contiene n arcos. Por tanto, P1 es la trayectoria más larga.

En virtud del lema I.6.3, un algoritmo para la trayectoria más larga puede resolver el problema de viajero, debido a que encontrar un circuito óptimo en G, sólo se necesita encontrar una trayectoria más larga en L(G).

CAPITULO II ALGORITMOS DE SOLUCION EXACTOS

Una vez identificado el problema del viajero (PV), se empezaron a proponer métodos de solución. Entre estos métodos, podemos distinguir los llamados exactos, qye determinan la solución óptima del problema del PV. En esta clase de métodos, se tiene a los llamados métodos de Ramificación y Acotamiento (R-A), los cuales, por su estructura, resultan los más adecuados para este tipo de problemas. Conviene señalar que los métodos de R-A combinados con otras técnicas de optimización proporcionan diferentes maneras de resolver el PV. Estos algoritmos son de gran utilidad cuando resolvamos problemas de dimensión menor.

En este capítulo, se estudian los algoritmos exactos para resolver el problema de viajero. Dentro de los algoritmos, el más usual y clásico es el método de R-A basado en la matriz reducida, aunque se tiene los combinados con árbol de expansión mínima y problema de asignación, entre otros. La primera sección analiza el aspecto teórico de los métodos de ramificación y acotamiento. Las secciones de dos a cuatro muestran las aplicaciones de métodos R-A al problema de viajero. En la quinta sección, se analiza otro método de solución como el de penalización. Finalmente, en la última sección, se analiza el método de programación dinámica.

II.1 RAMIFICACION Y ACOTAMIENTO: ASPECTOS GENERALES

Los métodos de ramificación y acotamiento son planteamientos enumerativos para resolver problemas de optimización discreta. Dichos métodos tienen la característica de jerarquizar las distintas alternativas de solución al considerar, en general, una pequeña parte de todas las alternativas posibles, pues las restantes son eliminadas por criterios que establecen que tales alternativas no son óptimas. La idea del método es describir el conjunto de soluciones del problema en cuestión en forma de un árbol y proceder a analizarlo sin omitir vértice. El concepto de la cota inferior sobre los valores de la función objetivo hace posible que una solución factible buena elimine un volumen considerable de soluciones factibles para propósitos de análisis.

Considere el problema de optimización discreta:

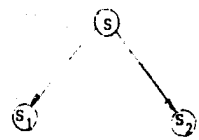
Minimizar $Z=f(x)$ Sujeto a $x \in E$

donde E es el conjunto de soluciones factibles y f es una función real.

El método de ramificación y acotamiento consiste en la aplicación recursiva de las operaciones de ramificación y acotamiento con el objeto de eliminar aquellos conjuntos de soluciones que no contienen la solución óptima hasta determinar el que la contiene, si dicha slución óptima existe. ([2])

La operación de ramificación del método consiste en dividir el conjunto de soluciones factibles E en subconjuntos ajenos de manera que se aisle la solución óptima en uno de los subconjuntos. Para representar esta división, se construye un árbol cuyo vértice básico corresponde a E, y los restantes vértices corresponden a los subconjuntos de E.

Sea S un subconjunto del conjunto de las soluciones factibles E. El conjunto S puede ser separado en un número de subconjuntos ajenos S1,...Sp tal que S=S1U...USp. En la mayoría de los casos, S es separado en dos subconjuntos ajenos que son representados por los vértices de un árbol. Por ejemplo, en la siguiente figura anexa, (S) representa el subconjunto S, y los vertices (S1) y (S2) los subconjuntos S1 y S2.



Después de dividir S en dos subconjuntos, se puede pensar que la solución óptima tiene más posibilidades en uno de los subconjuntos.

La operación de acotamiento del método consiste en determinar una cota inferior de f(x) en S para cada subconjunto S⊆E. Esta cota puede ser considerada como una evaluación de f(x) en S, que se denota por ev(S) y se tiene que ev(S)≤f(x), ∀x∈S. La utilidad de la evaluación de la cota inferior es la siguiente: si tenemos una solución del problema de optimización discreta (x̄∈E) y si

$$ev(S) \geq f(\bar{x})$$

entonces no existen mejores soluciones en el subconjunto S, pues $f(x) \geq ev(S) \geq f(\bar{x})$, ∀x∈S. La ventaja de esta observación es que prácticamente se requiere que calculemos una "buena" evaluación de f(x) para todos los subconjuntos de S, para obtener una "buena" solución factible de f(x) en E.

En resumen, para un problema dado, se requiere buscar una buena evaluación de la cota inferior. Frecuentemente, se usa el siguiente método: suponer que E=E'∩E", S=S'∩S" con S'⊆E' y S"⊆E", y resolver el problema más sencillo:

$$\min_{x \in S'} g(x)$$

donde g(x) es una función definida en E' tal que $g(x) \leq f(x)$ ∀x∈E. (usualmente g(x) es igual a f(x)). Entonces para obtener una evaluación de la cota inferior de f(x) en S, se relaja la condición del problema en

$$ev_1(S) = \min_{x \in S'} g(x)$$

Los métodos más usados de exploración del árbol asociado al método de ramificación y acotamiento son:

a.Método de ramificación y acotamiento progresivo:
Consiste en ramificarse a partir del vértice de evaluación mínima de árbol. Esto significa una búsqueda amplia. Generalmente es complejo, pero es de interés cuando existen pocas restricciones o si uno sabe cómo obtener una buena solución.

b.Método de ramificación y acotamiento secuencial:
Consiste en ramificar aquel vértice del árbol que está más cerca al último vértice ramificado. Esto significa una exploración de profundidad. Es empleado aquí el concepto de Regreso al nodo anterior, o sea que la regla para la solución de vértices es UEPS (última entrada, primera salida). Este método generalmente es fácil de operar y por lo tanto se usa con mayor frecuencia.

Un método de R-A para un problema en particular queda definido al especificar sus operaciones de ramificación y acotamiento. En el caso del problema de viajero, existen tres métodos de ramificación

y acotamiento:

1. Construcción del circuito según la matriz reducida.
2. Eliminación de subcircuitos a partir de la solución de problema de asignación.
3. Construir la travectoria basando en el árbol de expansión mínima.

En seguida, se explica el funcionamiento de cada uno de ellos.

II.2 METODO R-A BASADO EN LA MATRIZ REDUCIDA

El funcionamiento de este método se basa en la construcción del circuito hamiltoniano según la matriz reducida ([7], [2]). Para ilustrar la idea de este método, veamos el siguiente ejemplo:

Sea la matriz de tiempos

	A	B	C	D	E	F
A	-	27	43	16	30	26
B	7	-	16	1	30	25
C	20	13	-	35	5	0
D	21	16	25	-	18	18
E	12	46	27	28	-	5
F	23	5	5	9	5	-

Tabla II.1 tiempos entre localidades

Empezamos por observar que si T es la duración de un recorrido hamiltoniano asociado a la matriz de tiempos de la tabla II.1, entonces, la duración de ese mismo recorrido con la matriz de tiempo obtenida de restar un escalar hi a la hilera i es dado por T-hi. Esta observación es inmediata pues cada recorrido considera uno y sólo un elemento de la hilera i. Lo mismo sucede si restamos un escalar hj de la columna j (j=A,B,C,D,E,F). Una cota inferior del recorrido hamiltoniano óptimo es sencilla de obtener si restamos de cada hilera de la matriz de tiempos entre localidades, el mínimo elemento correspondiente. Por ejemplo, si en la tabla II.1, restamos el escalar 16 de la primera hilera y los escalares: 1, 0, 16, 5 y 5 de las respectivas hileras restantes se tiene como nueva matriz de tiempos entre localidades:

	A	B	C	D	E	F
A	-	11	27	0	14	10
B	6	-	15	0	29	24
C	20	13	-	35	5	0
D	5	0	9	-	2	2
E	7	41	22	23	-	0
F	18	0	0	4	0	-

Tabla II.2 tiempos modificados entre localidades

Un recorrido hamiltoniano de duración T asociado a la tabla II.1, tiene una duración T-43 cuando se calcula con los tiempos entre localidades de la tabla II.2. Si restamos de cada columna de la tabla II.2 el mínimo elemento correspondiente, se obtiene

	A	B	C	D	E	F
A	-	11	27	0	14	10
B	1	-	15	0	29	24
C	15	13	-	35	5	0
D	0	0	9	-	2	2
E	2	41	22	23	-	0
F	13	0	0	4	0	-

Tabla II.3 tiempos modificados entre localidades

Un recorrido hamiltoniano de duración T asociado a tabla II.1. tiene una duración T-48 cuando se calcula con tiempos entre localidades de la tabla II.3.

Un aspecto importante de la discusión anterior es que cada recorrido hamiltoniano asociado a la tabla II.3 tiene una duración no negativa y que difiere del tiempo de recorrido original en 46 unidades de tiempo. Equivalentemente una cota inferior de la duración del recorrido mínimo es 48.

APLICACION DEL METODO DE RAMIFICACION Y ACOTAMIENTO:

Una manera de proceder a la determinación del recorrido hamiltoniano de mínima duración es particionar el conjunto de recorridos hamiltonianos como sigue:

a1. Recorridos hamiltonianos que usan el arco AD
a2. Recorridos hamiltonianos que no usan el arco AD

En el primer caso la matriz de tiempos entre localidades se reduce a una nueva matriz en donde se elimina la hilera A y la columna D. Asimismo, el tiempo entre localidad D y A se hace infinito (o un número grande) para evitar usar el arco DA; pues sabemos que no forma parte del recorrido hamiltoniano mínimo. Si no hacemos este tiempo infinito, existe la posibilidad de la aparición de subcircuitos.

	A	B	C	E	F
B	1	-	15	29	24
C	15	13	-	5	0
D	-	0	9	2	2
E	2	41	22	-	0
F	13	0	0	0	-

Tabla II.4 tiempos entre localidades

Una cota inferior de la duración de los recorridos hamiltonianos asociados con esta tabla es sencilla de obtener si restamos una unidad a cada elemento de hilera uno para obtener:

	A	B	C	E	F
B	0	-	14	28	23
C	15	13	-	5	0
D	-	0	9	2	2
E	2	41	22	-	0
F	13	0	0	0	-

Tabla II.5 tiempos modificados entre localidades

Conviene señalar que como resultados de las manipulaciones anteriores podemos decir que los recorridos hamiltonianos asociados a la tabla II.1, que usen el arco AD tiene una duración no menor de 49 unidades de tiempo: 48 unidades acumuladas hasta la obtención de la tabla II.3 y una unidad de tiempo al pasar la tabla II.4 a II.5. Equivalentemente, 49 unidades de tiempo es una cota inferior a la duración de los recorridos hamiltonianos que usan el arco AD.

Una cota inferior a la duración de los recorridos hamiltonianos que no usan el arco AD es sencilla de obtener. Si no usamos el arco AD tenemos que usar un arco que va de A a alguna de las localidades del conjunto {B,C,E,F} y otro arco que parte de {B,C,E,F} y llegar a la localidad D. De la tabla II.3 se observa que el tiempo mínimo para ir de alguna de estas localidades a D es cero.

Al evitar el arco AD es necesario pagar un retardo mínimo de 10 unidades de tiempo. Equivalentemente, 58 unidades de tiempo es una cota inferior a la duración de los recorridos hamiltonianos que no usan el arco AD.

Una forma esquemática de la ramificación y acotación realizada sobre los recorridos hamiltonianos que se analizan se muestran en la figura II.1.

Una cuestión que conviene analizar en este momento es: ¿por qué se efectuó la ramificación sobre el arco AD y no sobre otro que tuviese el tiempo entre localidades igual a cero en la tabla II.3 ? La razón es como sigue: estrictamente debimos haber analizado para cada arco (i,j) que tiene tiempo entre localidades igual a cero en la tabla II.3, el correspondiente retardo asociado al eliminar ese arco y elegir el que tenga máximo retardo (que es el arco AD) para garantizar una mayor cota inferior de la duración de los recorridos hamiltonianos.

Como siguiente paso consideramos una ramificación del nodo AD, esto es, recorridos hamiltonianos que usan el arco AD cuya cota inferior de duración es 49. Para ello partimos de la tabla II.5 y calculamos los retardos mínimos asociados con cada uno de los arcos cuyo tiempo asociado es cero. Dichos retardos son sencillos de calcular. Si se trata del arco (i,j) lo que necesitamos hacer es sacar el mínimo de la hilera i así como el mínimo de la columna j en la tabla II.5 y sumarlos para obtener una cota inferior a los retardo asociados con los recorridos hamiltonianos que no usan el arco (i,j). específicamente:

Retardo (BA) = 14 + 2 =16
Retardo (CF) = 5 + 0 = 5
Retardo (DB) = 2 + 0 = 2
Retardo (EF) = 0 + 0 = 0
Retardo (FB) = 0 + 0 = 0
Retardo (FC) = 0 + 9 = 9
Retardo (FE) = 0 + 2 = 2

Es por ello que escogemos el arco BA para efectuar la ramificación como sigue:

b1. Recorrido hamiltoniano que usan el arco BA
b2. Recorrido hamiltoniano que no usan el arco BA

En el primer caso partimos de la tabla II.5, eliminando la hilera B y la columna A y hacemos que el tiempo de A a B sea infinito para evitar circuitos innecesarios. En este momento, como AD y BA han sido seleccionados para formar el recorrido hamiltoniano, hacemos que el tiempo de D a A sea infinito por implicación. La nueva tabla de tiempos es:

	B	C	E	F
C	13	-	5	0
D	-	9	2	2
E	41	22	-	0
F	0	0	0	-

Tabla II.6 Tiempos entre localidades

Con el propósito de tener un elemento cero en cada hilera y columna así como mejorar la cota inferior de los recorrido hamiltonianos procedemos a restar dos unidades de la hilera 2 en la tabla II.6 para obtener:

	B	C	E	F
C	13	-	5	0
D	-	9	2	2
E	41	22	-	0
F	0	0	0	-

Tabla II.7 Tiempos modificados entre localidades

y observar que una cota inferior de los recorridos hamiltonianos que usan BA es 51 unidades de tiempo.

Una cota inferior de la duración de los recorridos que no usan el arco BA es 51 unidades de tiempo.

Una forma esquemática de las ramificaciones y acotaciones desarrolladas sobre los recorridos hamiltonianos se tiene en la figura II.1 en donde se muestra el resto de las operaciones de manera resumida para llegar a demostrar que el recorrido hamiltoniano óptimo es A-D-C-E-F-B-A con una duración de 63 unidades de tiempo.

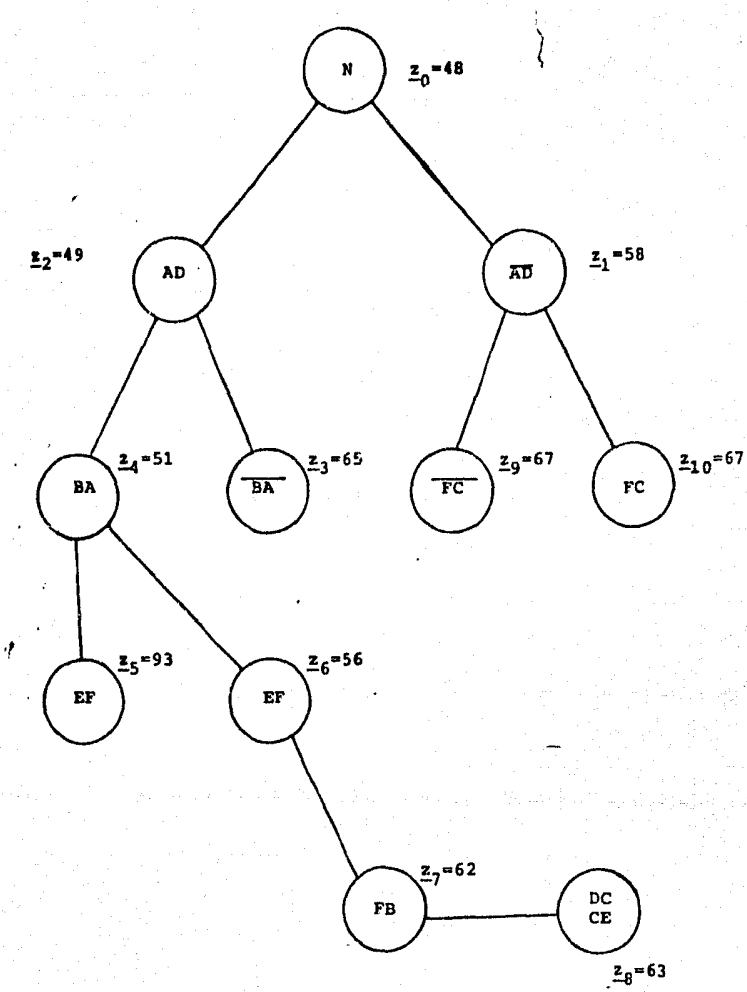


Figura II.1 Arbol de búsqueda del problema.

La formalización del algoritmo del ramificación y acotamiento (Little) es:

Propósito: Determinar el circuito hamiltoniano de costo mínimo dada una matriz de costos. El método utiliza la propiedad de la matriz de costos reducida para probar la inclusión o exclusión de un arco en el circuito.

Descrioción:
Paso 1: Dada la matriz de costos C, se efectúan subtracciones en las hileras y las columnas de la matriz C, sin permitir que aparezcan valores negativos. Con esto obtenemos una cota inferior del problema de viajero al sumar los elementos que se restaron a las hileras y columnas. La matriz C' obtenida es la matriz reducida de C.

Paso 2: Sean S un vértice del árbol y ev(S) la la cota inferior de este vértice S. Con cada par (i,j) tal que $d_{ij}=0$ asociar el número $w_{ij}=u_i+v_j$, donde u_i,v_j son definidos como sigue: u_i (o v_j) es el elemento mínimo diferente que está en la hilera i (o la columna j) de la matriz reducida D'. Seleccionar el arco que tiene el máximo de los w_{ij}.

Paso 3: Si el arco (i,j) no está seleccionado, ev(S)+w_{ij} es una cota inferior. Si el arco (i,j) está seleccionado, entonces la matriz será reducida omitiendo la hilera i y la columna j. Buscar la condición adicional sobre D' para excluir la introducción de subcircuito.

Paso 4: Seleccionar el vértice que tiene el menor costo, ir al paso 1.

II.3 METODO R-A BASADO EN EL PROBLEMA DE ASIGNACION

La idea principal del algoritmo es eliminar de subcircuitos a partir de la solución de problema de asignación. ([1])

En la sección I.3 del capítulo uno, se indica que la solución del problema de asignación es una cota inferior del problema de viajero. Sin embargo, en la solución de problema de asignación, existe subcircuitos, esto es, puede existir una solución de la forma ((i(1),i(2)) , ..., (i(k),i(1)), (i(k+1),i(k+2)) , ..., (i(n),i(k+1))). En este caso, los vértices i(1), l=1,...k forman un subcircuito; y los vértices i(1), l=k+1,...,n forman otro. La esencia fundamental que este método de ramificación y acotamiento es eliminar estos subcircuitos para formar un circuito hamiltoniano.

Metodo de R-A basado en el problema de asignación:

Propósito: Determinar el circuito hamiltoniano de costo mínimo, dada una matriz de costos. Primero se obtiene la solución de problema de asignación, y luego se toma la decisión sobre los arcos de los subcircuitos para convertirlos en un circuito hamiltoniano.

Descripción:
La aplicación del método de ramificación y acotamiento tiene diferentes formas dependiendo de la operación de las operaciones de ramificación, pero la operación de acotamiento siempre es la misma y está basada en la solución óptima del problema de asignación bajo la matriz de costos supuesta. (método húngaro)

Las tres maneras diferentes de efectuar la operación de ramificación son:

a) Ramificación simple
Romper el subcircuito de la solución de problema de asignación suponiendo que uno de los arcos del subcircuito tiene costo infinito.

b) Ramificación disjunta
Suponer que uno de los arcos del subcircuito tiene costo infinito pero un subconjunto de los arcos del subcircuito tiene que estar conectado forzosamente.

c) Ramificación mejorada
Romper el subcircuito suponiendo que todos los arcos tienen costo infinito excepto uno de ellos.

Ejemplo: En la figura II.2.a, se muestra una solución del problema de asignación. En la figura II.2.b,c,d, se muestran diferentes las operaciones de ramificación mencionadas anteriormente.

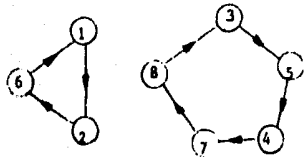


Figura II.2.a solución de prob. asignación PO

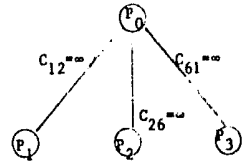


Figura II.2.b ramificación simple

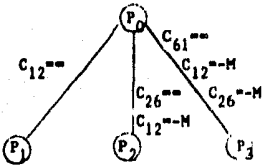


Figura II.2.c ramificación disjunta

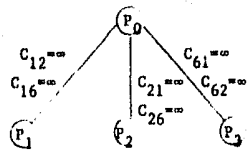


Figura II.2.d ramificación mejorada

II.4 METODO DE R-A BASADO EN EL ARBOL DE EXPANSION MINIMO

En ciertos casos del problema del viajero, se necesita obtener una trayectoria hamiltoniana con costo mínimo. En otras palabras, una trayectoria que pase todos los nodos sin cerrarse. Para la solución de este problema se puede aprovechar la propiedad del árbol de expansión mínima. En la sección II.4, hemos visto que el costo total del árbol de expansión mínima es una cota inferior del problema del viajero. Observe que en el árbol de expansión mínima puede haber más de dos arcos incidentes en un nodo, lo que pretende este algoritmo es eliminar estos nodos.

Metodo de R-A basado en el árbol de expansión mínima:

Propósito: Dada la matriz de costos simétricos, se encuentra la trayectoria hamiltoniana entre cualquier par de nodos. El nodo inicial y el final no son especificados. El algoritmo aprovecha la conectividad de un árbol y trata de transformarlo a una trayectoria hamiltoniana mediante la toma de decisiones sobre los arcos conectados al nodo que tiene grado mayor que dos.

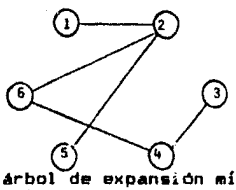
Descripción:

La aplicación del método R-A en este caso consiste en efectuar la operación de acotamiento por medio de la solución del árbol de expansión mínima de la matriz de costos actual. La operación de ramificación es: suponer cada uno de los arcos excesos en un nodo tiene el costo infinito, de esta manera generar varias alternativas.

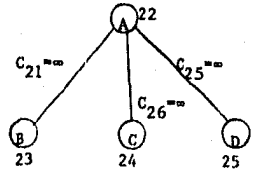
Ejemplo: considere la matriz de costos siguiente y resuelva usando el método anterior

		1	2	3	4	5	6
	1	-	4	10	18	5	10
	2	4	-	12	8	2	6
C=	3	10	12	-	4	18	16
	4	18	8	4	-	14	6
	5	5	2	18	14	0	16
	6	10	6	16	6	16	0

El árbol de expansión mínima genera la solución Ct=22, observamos d(2)=3>2.

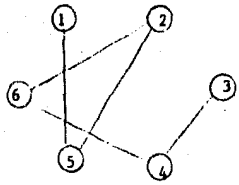


Árbol de expansión mínima bajo la matriz de costo actual

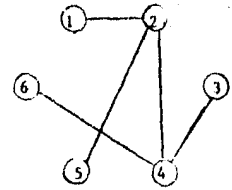


Árbol de ramificación

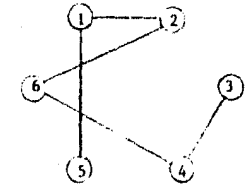
Las tres alternativas se muestran como sigue:



B: costo actual=23
solución óptima



C: costo actual=24



D: costo actual=25

II.5 METODO DE PENALIZACION

Un caso especial del problema de trayectoria hamiltoniana es cuando están fijos los vértices extremos de la trayectoria. Este tipo de problemas se puede resolver por medio del método de penalización. La idea fundamental del método es generar el árbol de expansión mínima y luego asignar un mayor costo al los arcos conectados al nodo de grado mayor que dos. La introducción de esta penalización es sencilla de contabilizar en el problema que se analiza como puede observar del lema siguiente.

Lema II.1: Si la matriz de costo C es transformada a otra matriz C' tal que:

$$c'(i,j)=c(i,j)+p(i)+q(j) \quad \forall i,j=1,2,\ldots n$$

donde p(k) es un número real no negativo, entonces el costo total bajo la matriz C' de todas las trayectorias hamiltonianas que tienen los mismos nodos inicial y final es la suma del costo total bajo C y una constante.

Prueba: Sea Fc el costo de una trayectoria hamiltoniana bajo la matriz de costo C que tiene el nodo inicial x(1) y el final x(2). Entonces, x(1) y x(2) se unen una sola vez a la trayectoria, y los demás nodos se unen dos veces a la trayectoria. El costo Fc' de la misma trayectoria hamiltoniana bajo la matriz de costo C' difiere Fc por:

$$Fc' = Fc + p(1) + p(2) + 2 \sum_{j\neq 1,2} p(j)$$

los p(k) son constantes, por tanto queda demostrado el lema.

Basando en este lema, se desarrolla el algoritmo de penalización como sigue:

Metodo de Penalización:

Propósito: Dada la matriz de costos simétrica, determinar la trayectoria hamiltoniana entre el nodo inicial y el final especificados. En este algoritmo, es necesario definir un valor de penalización que normalmente está entre 5 y 10.

Descripción:

Paso 1: Penalizar los 2 nodos extremos M,N. Asignar un valor grande a p(M), p(N); calcular nueva matriz de costo:

$$c'(i,j)=c(i,j)+p(i)+q(j) \quad \forall i,j$$

Paso 2: Obtener la solución de árbol de expansión mínima con la matriz de costo actual.

Paso 3: Si no existe nodos de grado mayor que 2, termina. Penalizar los nodos que no satisfacen las condiciones de ser solución del problema de viajero.

$$p(i) = r \max(d(i)-2,0)$$

donde r es constante, experimentalmente, se escoge entre 5 y 10.

Paso 4: Calcular nueva matriz de costo

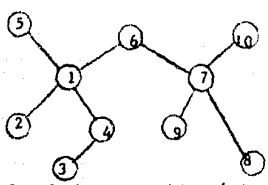
$$c'(i,j)=c(i,j)+p(i)+p(j) \quad \forall i,j$$

Ir al paso 2.

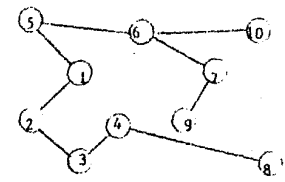
Ejemplo: determinar la trayectoria hamiltoniana óptima dados los nodos terminales 8 y 9, y la matriz de costos:

	1	2	3	4	5	6	7	8	9	10
1	-	28	31	28	22	36	50	67	40	74
2	28	-	31	40	41	64	74	80	63	101
3	31	31	-	14	53	53	53	50	42	83
4	28	40	14	-	50	41	39	41	28	69
5	22	41	53	50	-	40	61	86	53	78
6	36	64	53	41	40	-	24	58	22	39
7	50	74	53	39	61	24	-	37	11	30
8	67	80	50	41	86	58	37	-	36	60
9	40	63	42	28	53	22	11	36	-	41
10	74	101	83	69	78	39	30	60	41	-

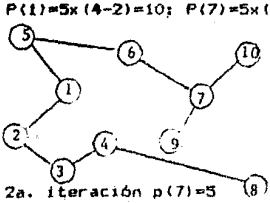
Durante las iteraciones, el árbol de expansión mínima se transforma a la solución del problema de viajero de la siguiente manera:



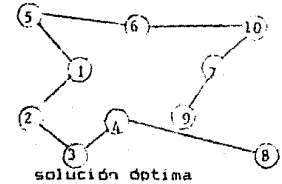
Arbol de expansión mínima,
P(1)=5x(4-2)=10; P(7)=5x(4-2)=10



1a.iteración P(6)=5x(3-2)=5



2a. iteración p(7)=5



solución óptima

Conviene hacer algunos observaciones adicionales al algoritmo de penalización que implica una mayor eficiencia. Para el paso 3 del algoritmo, se puede aplicar diferentes estrategias de penalizacion:

1. Penalización fija
   a) Con valores positivos unicamente:
      para los nodos de grado mayor que 2, la penalización es:
      $r(d(i)-2)$
   b) Con valores nagativos unicamente:
      Para los nodos de grado 1, la penalización es:
      $-r$
   c) Combinado:
      para los nodos de grado mayor que 2, o de grado 1, las penalizaciones son respectivamente:
      $r(d(i)-2)$, $-r$

2. Penalización decreciente reversible:
   Para evitar la divergencia del algoritmo, en la iteración k, se usa en lugar de r, el valor $v^{k-1}r$ donde $0<v<1$.

3. Penalización calculada:
   a) Con valores positivos únicamente:
      Se calcula p(i) de la siguiente manera:
      Borrar desde el árbol $T(X,A)$ uno de los arcos (x_i,x_r) incidentes en el nodo x_i.
      Esto deja que T está separado por dos partes T1 y T2.
      Encontrar el arco de costo mínimo que ligue estos dos subárboles:

$$C(x_j^r,x_k^r) = \min_{x_j \in T_1 \neq x_i} \; \min_{x_k \in T_2 \neq x_r} \{C(x_j,x_k)\}$$

      entonces

$$p(i) = \min_{(x_i,x_r)\in A} \{C(x_j^r,x_k^r)-C(x_i,x_r)\}$$

      es la penalización mínima.

   b) Con valores negativos únicamente:
      Supongamos que $d(i)=1$.
      Consideramos la adición de un arco (x_i,x_r) al árbol T, inmediatamnete se formará un ciclo.
      Sea S_{ri} el conjunto de arcos de T en la ruta (X_r,X_i)

$$p(i) = \min_{x_r \in X} \{C(x_i,x_r) - \min_{(x_j,x_k)\in S_{ri}} \{C(x_j,x_k)\}\}$$

      es decir, p(i) es el costo adicional mínimo por agregar un arco de x_i a cualquier otro x_r, y borrar el arco menor en la ruta de x_r a x_i, entonces, d(i) será 2.

   c) Método combinado:
      Para los nodos de grado mayor o igual que 2, usar a)
      Para los nodos de grado 1, usar b).

II.6 SOLUCION POR PROGRAMACION DINAMICA

Una manera de resolver el problema de N etapas de decisión es proceder a descomponerlo en N subproblemas cuya solución es equivalente al problema original. A grandes razgos lo que se pretende es resolver primero la última etapa y usar dichos resultados para resolver la penúltima etapa y así sucesivamente. Este procedimiento se denomina PROGRAMACION DINAMICA.

La programación dinámica para resolver el problema de viajero consiste en:

Sea

$$f_{M-1}((i(m)|i(1),\ldots,i(m-1),i(m+1),\ldots,i(k-1))$$

el costo total mínimo de la trayectoria que parte del nodo 1, pasando por $(i(1),\ldots,i(m-1),i(m+1),\ldots,i(k-1))$ y termina en el nodo $i(m)$.

En el etapa k de solución, la trayectoria de costo mínimo de 1 a j está determinada por:

$$f_M((j|i(1),\ldots,i(k-1))$$

$$= \min \{f_{M-1}[(i(m)|i(1),\ldots,i(m-1),i(m+1),\ldots,i(k-1)]+c(i(m),j)\}$$

Se aplica la ecuación anterior, empezando con la condición inicial:

$$f_2(j|i(1))=c(1,i(1))+c(i(1),j) \quad \forall j\neq 1,\ j\neq i(1)$$

El circuito hamiltoniano óptimo se obtiene por resolver:

$$f_M((j|i(1),\ldots,i(n-1))$$

$$= \min \{f_{M-1}[(i(m)|i(1),\ldots,i(m-1),i(m+1),\ldots,i(n-1)]+c(i(m),1)\}$$

La dificultad para resolver las ecuaciones recursivas en una computadora digital es el almacenamiento de los valores de f_M. Para calcular f_{M+1}, se debe tener presente el valor de todos los f_M's. Una vez que f_{M+1}'s han sido calculados, f_M pueden ser eliminados de la memoria. Se puede calcular que el número de valores de f_M es:

$$g(n,k)=(n-1)!/[(k-1)!(n-k-1)!]$$

cuando $(n-1)/2 \leq k \leq (n+1)/2$, g llega a su valor máximo.

Entonces, en una computadora Apple que tiene 17K de momoria para el usuario, se puede resolver un problema de n=14, ya que g(14,7)=12,012.

Para un problema de matriz de costos simétrica, con un número par n, sólo es necesario calcular $f_{n/2}$. En particular

$$f_{n/2}(j|i(1),\ldots,i(n/2-1))+f_{n/2}(j|i(n/2+1),\ldots,i(n-1))$$

es el costo toal del circuito optimo que procede desde nodo 1 pasando por (i(1),...,i(n/2-1)) sucesivamente, y regresa al nodo 1 por (i(n/2+1) ,...,i(n-1)).

Ejemplo: Obtener el circuito hamiltoniano óptimo usando la siguiente matriz de costos:

	1	2	3	4
1	-	13	5	-
2	13	-	-	11
3	-	6	-	4
4	2	11	-	-

$f_2(3|2)=00$ $\quad$ $f_2(4|2)=c(1,2)+c(2,4)=24$

$f_2(2|3)=c(1,3)+c(3,2)=11$ $\quad$ $f_2(4|3)=c(1,3)+c(3,4)=9$

$f_2(2|4)=00$ $\quad$ $f_2(3|4)=00$

$$f_3(2|(3,4))=\min\begin{cases} f_2(4|3)+c(4,2) \\ f_2(3|4)+c(3,2) \end{cases}$$

$$=\min\begin{cases} 00 \\ 9+11 \end{cases}$$

$$=20$$

$$f_3(4|(2,3))=\min\begin{cases} f_2(2|3)+c(2,4) \\ f_2(3|2)+c(3,4) \end{cases}$$

$$=\min\begin{cases} 11+11 \\ 00 \end{cases}$$

$$=22$$

$$f_4(1|(2,3,4))=\min\begin{cases} f_3(3|(2,3))+c(4,1) \\ f_3(2|(3,4))+c(2,1) \\ f_3(3|(2,4))+c(3,1) \end{cases}$$

$$=\min\begin{cases} 22+2 \\ 20+13 \\ 00 \end{cases}$$

$$=24$$

El circuito hamiltoniano óptimo es (1,3,2,4).

CAPITULO III ALGORITMOOS DE SOLUCION HEURISTICOS

Los métodos exactos para resolver el problema de viajero, en la práctica, resultan ineficientes en un gran número de situaciones. Es decir, consumen demasiado tiempo en el proceso de solución, o pueden llegar a saturar la memoria de la computadora. Por ello, para problemas de dimensión práctica, sólo se desea obtener una buena solución, que usualmente no es óptima. Estos métodos de solución aproximada se denominan heurísticos, específicamente, el procedimiento para obtener una solución factible sin enfatizar en la función objetivo se denomina *algoritmo heurístico (o aproximado)*. A la solución producida por un algoritmo heurístico se denomina *solución heurística*.

La medida principal para los métodos proximados es la experiencia computacional que ofrece. De hecho, uno toma en cuenta la calidad de solución obtenida. No obstante, un método que produce casi siempre soluciones excelentes puede proporcionar una solución pobre para un tipo particular de problemas. Es por elllo que se ha introducido el concepto de "cota del peor caso" para medir la certeza de la calidad de solución.

En el caso de problema de viajero, supongamos que V(a) es la longitud de una ruta producida por un algoritmo heurístico, y V(p) es la longitud de una ruta óptima, entonces un buen indicador es el valor máximo de la razón V(a)/V(p) para cualquier problema al que se aplica el método heurístico. Este valor máximo es la "cota del peor caso" de un algoritmo específico. Un análisis probabilístico nos ayuda a saber el comportamiento de esta razón bajo la suposición de la distribución de costo. Hasta hoy, se han diseñado numerosos algoritmos heurísticos, resultado del gran esfuerzo desarrollado por los investigadores en esta area.

En este capítulo, revisamos y comparamos los algoritmos heurísticos más populares para resolver el problema de viajero. Los algoritmos para resolver un problema de viajero están clasificados de la siguiente manera:

1. Mejoramiento circuito por circuito. (III.1 ~ III.2)
2. Eliminación de subcircutos. (III.3 ~ III.4)
3. Construcción de circuito. (III.5 ~ III.10)
4. Algoritmos combinados. (III.11 ~ III.12)

donde el número indicado dentro de de la paréntesis, significa la numeración de algoritmos que se describen en este capítulo.

III.1 METODOS DE MEJORAMIENTO DE CIRCUITO

La idea principal de estos métodos consiste en lo siguiente: empezar a partir de un circuito inicial arbitrario y se busca un circuito "vecino" del circuito actual que es mejor. Es decir, se trata de efectuar un número mínimo de intercambios de arcos para tener un mejoramiento en la solución. Este tipo de algoritmo es rápido ya que se efectúan comparaciones que no consumen mucho tiempo de la computadora. Dado que se está optimizando con los circuitos factibles, se puede terminar el algoritmo en cualquier momento. Sin embargo, pueden resultar ineficientes cuando el numero de arcos a intercambiar es grande. Por ejemplo, el algoritmo k-óptimo, que se mostrará a continuación, sólo es bueno cuando k=2 y 3. Si k es un número grande, el algoritmo se vuelve lento debido a la recursividad.

III.1 Algoritmo k-óptimo: ([4])

Propósito: Dada la matriz de costos simétrica, encontrar un circuito hamiltoniano de menor costo. Decimos "menor costo", porque el método es aproximado, y en muchas ocasiones no se obtiene la solución óptima. El valor de k indica el número de arcos a intercambiar durante cada iteración.

Descripción:
Paso1: Encontrar un circuito hamiltoniano arbitrario;
Paso2: Asignar x=2
Paso3: Intercambiar x arcos. (es decir, eliminar x arcos que están actualmente en el circuito y agregar x arcos que no están en él)

Paso4: Si $\sum_{i \in S} C_i - \sum_{i \in S'} C_i > 0$, sustituir la solución con el remplazo de los x arcos, ir al paso 2; (S: conjunto de arcos a eliminar; S': conjunto de arcos a insertar)

Si $\sum_{i \in S} C_i - \sum_{i \in S'} C_i \leq 0$, y no han agotado las combinaciones de x arcos, ir al paso 3;

Si $\sum_{i \in S} C_i - \sum_{i \in S'} C_i \leq 0$, y han agotado las combinaciones de x arcos, se incrementa x en 1.
Paso5: Si x es mayor que k, termina; si no, ir al paso 3.

A continuación, se explica el proceso del intercambio de arcos en el algoritmo k-optimo con k=2. En la figura III.1 para un circuito inicial (V1, V3, V2, V4), se selecciona un vértice V1 y un arco incidente en V1, que no está en el circuito, digamos (V1,V2). Si agregamos este arco al circuito, un arco del circuito original que está conectado con V2 tiene que ser eliminado, digamos (V2,V3). Para que el grado del nodo V3 sea 2, se tiene que agregar un arco incidente en V3 al circuito. El otro extremo que este arco mencionado debe ser un nodo que está conectado al V1. En la figura III.1, se nota que este nodo es único. Es decir, por haber escogido (V1,V2) y (V2,V3), se requiere que los últimos dos arcos a remplazar sean únicos.

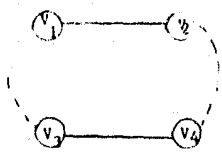
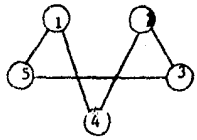


Figura III.1

Ejemplo: Considere el problema del viajero con matriz de costo y solución inicial mostrados a continuación:

	1	2	3	4	5
1	-	2	3	5	4
2	2	-	2	4	2
3	3	2	-	6	2
4	5	4	6	-	4
5	4	2	2	4	-



El costo del circuito inicial (1,5,3,2,4,1) es 17. Seleccionamos Vi=1, (V1,V2)=(1,4) y (V2,V3)=(4,5), de esta manera, V3 queda seleccionado por la implicación. Pero la reducción de costo es igual a C(5,3)-C(3,1)+C(1,4)-C(4,5) = 2-3+5-4=0, por lo que está opción no nos ofrece ningún mejoramiento. Otra alternativa es (V1,V2)=(1,5), (V2,V3)=(5,2), entonces el arco (3,2) será remplazado por (3,1). La reducción ahora es C(1,5)-C(5,2)+C(3,2)-C(3,1)=1, el circuito nuevo va a ser (1,3,5,2,4,1) con el costo 16.

Reinicializando con este circuito, se repite el mismo procedimiento. Seleccionamos (V1,V2)=(1,4) y (V2,V3)=(4,5). Esto implica el cambio adicional de (5,2) por (2,1), y la reducción de costo es un valor positivo 1. De este modo el circuito nuevo es (1,2,4,5,3,1) con costo 15. Continuando el procedimiento, ya no ofrece ningún mejoramiento. Por tanto (1,2,4,5,3,1) es 2-óptimo.

III.2 Algoritmo ALGO IV(r)

Propósito: Encontrar un circuito hamiltoniano de menor costo dada una matriz de costos. El método trata de intercambiar la posición de los segmentos de la trayectoria para mejorar la soución inicial. Para entender mejor el algoritmo, se divide la explicación en dos etapas.

Descripción:
Etapa I:cuando r=1, es decir ALGO IV(1)
Paso 1: Encontrar un circuito arbitrario t(i(1), i(2), ...,i(n)), y asignar el contador x el valor 1;
Paso 2: Intercambiar la posición de i(1) con i(2), i(1) con i(3), ..., i(1) con i(n). De las n-1 comparaciones efectuadas, se denota el mejor resultado después del

cambio para (j(1), j(2), ... j(n)).

Paso 3: Se hace la siguiente asignación:
i(k)=j(k+1) para $1 \leq k < n$; i(n)=j(1)
Se incrementa x en 1.

Paso 4: Si $x \leq n$, ir al paso 2; Si $x > n$, termina.

Etapa II: ALGO IV(r)

Paso 1: Usar el circuito obtenido por ALGO IV(r-1): (i(1),...,i(n)), y asignar el contador x el valor 1.

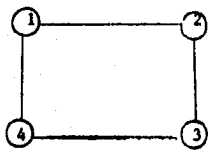
Paso 2: Insertar (i(1), i(2),... i(r)) en la secuencia (i(r+1), i(r+2),...,i(n)). Se tiene n-r maneras para la inserción. De estas n-r comparaciones efectuadas, se denota el mejor resultado (j(1), j(2),...,j(n).

Paso 3: Insertar (i(r), ... i(2), i(1)) en la secuencia (i(r+1), i(r+2),...,i(n)). De estas n-r comparaciones efectuadas, se denota el mejor resultado (j(1), j(2),...,j(n)).

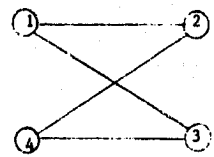
Paso 4: Hacer la siguiente asignación:
i(k)=j(k+1) $1 \leq k < n$
i(n)=j(1)
Incrementar x en 1.

Paso 5: Si $x \leq n$, ir al paso 2;
Si $x > n$, termina.

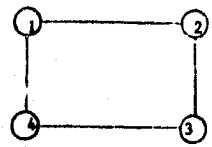
Un ejemplo de ALGO IV(1) se muestra en la figura III.2.



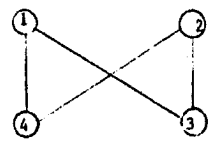
solución inicial (1,2,3,4)



intercambia 1 con 2 (2,1,3,4)



Intercambia 1 con 3 (3,2,1,4)



intercambia 1 con 4 (4,2,3,1)

Figura III.2

III.2 METODOS DE ELIMINACION DE SUBCIRCUITOS

La solución de ciertos algoritmos nos conduce una gráfica que consiste de varios subcircuitos. En este caso, cada subcircuito es óptimo para un subconjunto de nodos. Por ejemplo, cuando se resuelve el problema de asignación, si la solución es un circuito, ésta también es la solución del problema de viajero. Si la solución de problema de asignación no es un circuito, se tiene que aplicar un esquema iterativo para eliminar los subcircuitos. Se observa que este tipo de algoritmos puede particionar el conjunto de n nodos en varios subconjuntos menores. La obtención de la solución localmente óptima para cada subconjunto es fácil. La ventaja de este tipo de algoritmo es mediante esta decomposición, se usa menos memoria de computadora y se disminuye el tiempo consumido, ya que el número de operaciones de un algoritmo exacto de problema de viajero crece exponencialmente. En muchas situaciones reales, las soluciones localmente óptimas son adecuadas.

III.3 Algoritmo de adecuación:

Propósito: Dada una matriz de costos, encontrar un circuito de menor costo. Primero, se resuelve el problema de asignación, se elimina el arco más costoso en cada subcircuito, y se agregan arcos necesarios para convertirlo en un circuito hamiltoniano.

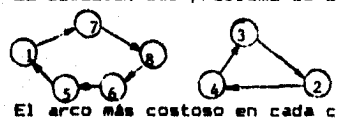
Descripción:

- Paso 1: Resolver el problema original como un problema de asignación.
- Paso 2: Borrar el arco más costoso en cada uno de los subcircuitos.
- Paso 3: Escoger algunos arcos con los que se forma un circuito.

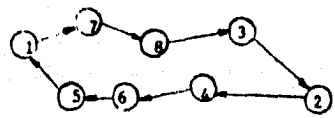
Ejemplo: Considere el problema de viajero con la matriz de costo:

	1	2	3	4	5	6	7	8
1	-	76	43	38	51	42	19	80
2	42	-	49	26	78	52	39	87
3	48	28	-	36	53	44	68	61
4	72	31	29	-	42	49	50	38
5	30	52	38	47	-	64	75	82
6	66	51	83	51	22	-	37	71
7	77	62	93	54	69	38	-	26
8	42	58	66	76	41	52	83	-

La solución del problema de asignación es:



El arco más costoso en cada ciclo es (8,6), (4,3).
Por tanto la solución se transforma en:



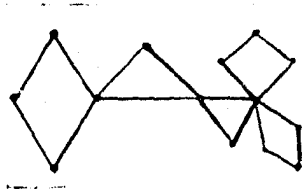


Figura III.3

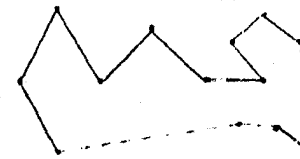


Figura III.4

III.4 Algoritmo de Karp:

Propósito: Dada una matriz de costos, determinar un circuito hamiltoniano de menor costo. El método procede a dividir geométricamente los nodos en varios subconjuntos, para obtener la solución óptima en cada subconjunto. Finalmente, se ligan todos los subcircuitos para obtener la solución.

Descripción:

Paso 1: Dibujar los n vértices en un rectángulo plano. Dividir este rectángulo en varios sub-rectángulos Rj tal que
- A. no existe traslape entre ellos;
- B. el número de los vértices colocados dentro de él no sea mayor que t (una constante);
- C. al menos existe un vértice en cada Rj en el límite con un rectángulo adyacente.

Paso 2: Construir el circuito óptimo en cada uno de los sub-rectángulos Rj.

Paso 3: Unir todos los circuitos en los rectángulos para conducir a un circuito que visite todos los nodos.

Ejemplo: se observa que en la figura III.3, los nodos son particionados en varios subrectángulos, y la solución de viajero para cada subconjunto de nodos es obtenida. En la figura III.4, se unen todos subcircuitos para obtener una solución del problema de viajero.

III.3 METODOS DE CONSTRUCCION DEL CIRCUITO

Los algoritmos de este tipo parten de un nodo arbitrario, digamos i(1). Desde este nodo, se construye una sucesión (i(1), i(2), ...,i(k) por la inclusión de nodos bajo cierta regla. El proceso termina cuando el circuito es obtenido. Un esquema muy simple es la regla de "nodo más cercano próximo. Inicialmente se tiene el nodo i(1). Se busca el nodo más cercano al nodo i(1), digamos i(2). Luego se busca el nodo mas cercano al nodo i(2), y así sucesivemente. Finalmente, de i(n) se regresa a i(1). La ventaja de estos algoritmos es que se obtienen soluciones bastante aproximadas al óptimo. El tiempo consumido es fácil de determinar. Pero, no se dispone de solución factible hasta que termine toda la aplicación del algoritmo.

III.5 Algoritmo de Inserción.

Propósito: Dada una matriz, encontrar un circuito hamiltoniano de menor costo. La idea principal consiste en lo siguiente: se forma primero el circuito hamiltoniano para un subconjunto de nodos, y se busca el nodo que no está dentro de este subconjunto, pero que está mas cercano de él. SE agrega este nodo entre un par de nodos del subconjunto de manera que minimiza el costo total actual.

Descropción:

Paso 1: Empezar con una subgráfica que consiste de nodo i.
Paso 2: Encontrar el nodo k tal que C(i,k) es mínimo y se forma un subcircuito i-k-i.
Paso 3: Dado un subcircuito, eleccionar un nodo k que no está dentro de la subgráfica.
Paso 4: Encontrar el arco (i,j) en la subcircuito de manera que minimiza C(i,k)+C(k,j)-C(i,j). Insertar k entre i y j.
Paso 5: Ir al paso 3 hasta tener un circuito hamiltoniano.

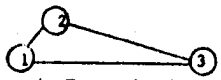
Tres políticas usuales para la selección del nodo k en el paso 3 son:

A. Inserción mas cercana:
   Que el nodo k está más cerca de cualquier nodo en la subcircuito.
B. Inserción con costo mínimo:
   Encontrar (i,j) en el subcircuito y k no está en él, tal que C(i,k)+C(k,j)-C(i,j) sea mínimo.
C. Inserción arbitraria:
   Seleccionar el nodo k arbitrario que no está en el subcircuito.

Ejemplo: considere la matriz de costo dada:

	1	2	3	4	5
1	-	2	3	5	4
2	2	-	2	4	2
3	3	2	-	6	2
4	5	4	6	-	4
5	4	2	2	4	-

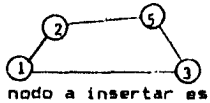
Inicialmente la subgráfica consiste del nodo 1. Se escoge el nodo 2 que está más cerca de 1 formando un subcircuito 1-2-1. El nodo 3 es el nodo más cercano de (1,2). Por tanto, se inserta 3 entre 1 y 2, y el resultado es:



Ahora el nodo 5 es el más cercano de la subgráfica, y
C(1,5)+C(5,3)-C(1,3)=3
C(2,5)+C(5,3)-C(2,3)=2

C(1,5)+C(5,2)-C(1,2)=4

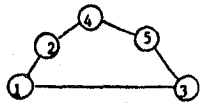
El valor mínimo en esto caso es i=2, j=3. Se inserta 5 entre (2,3). La subgráfica queda:



El último nodo a insertar es 4, y

C(1,4)+C(2,4)-C(1,2)=7
C(1,4)+C(3,4)-C(1,3)=8
C(1,4)+C(5,4)-C(1,5)=5
C(2,4)+C(3,4)-C(2,3)=8
C(2,4)+C(5,4)-C(2,5)=6
C(3,4)+C(5,4)-C(3,5)=8

El valor mínimo es 5. Se inserta 4 entre 1 y 5. El circuito óptimo se muestra a continuación:



III.6 Algoritmo de inserción más lejana:

Propósito: Dada una matriz de costos, determinar un circuito hamiltoniano de menor costo. La idea es similar a la de método de inserción, excepto la política de inserción.

Descripción:

Paso 1: Empezar la subgráfica con nodo i.
Paso 2: Encontrar el nodo k más lejana a i, formar i-k-i.
Paso 3: Encontrar el nodo k más lejana a cualquier nodo de la subgráfica, pero que k no está en ella.
Paso 4: Encontrar el arco (i,j) en la subgráfica tal que minimice C(i,k)+C(k,j)-C(i,j). Insertar k entre i y j.
Paso 5: Ir al Paso 3 hasta tener un circuito hamiltoniano.

III.7 Algoritmo Doble giro:

Propósito: Dada una matriz de costos simétrica, determinar un circuito hamiltoniano de menor costo. Se trata de convertir el árbol de expansión mínima a un circuito hamiltoniano mediante cambios de arcos.

Descripción: Es un método muy simple surgido para los casos simétricos, convirtiendo el árbol de expansión mínima a una solución de problema de viajero. Específicamente, primero se construye un árbol de expansión mínima YCE. Una trayectoria de visita a cada nodo es creada por trazar arcos dirigidos (i,j) y (j,i) para cada (i,j)CY. Empezando con la lista de nodos, formamos otra lista nueva eliminando los nodos encontrados anteriormente hasta que el último nodo de la lista sea encontrado.

III.8 Algoritmo Vecino más cercano:

Propósito: Dada una matriz de costos, determinar un circuito hamiltoniano de menor costo. El esquema es el más fácil entre los algoritmos heurísticos.

Descripción:

Paso 1: Empezar con un nodo arbitrario.
Paso 2: Encontrar el nodo más cercano al último nodo agregado a la trayectoria. Agregar este nodo a la trayectoria.
Paso 3: Repetir el paso 2 hasta que todos los nodos sean contenidos en la trayectoria. Entonces, juntar el primer nodo y el último.

III.9 Algoritmo Clarke & Savings

Propósito: Dada una matriz de costos, determinar un circuito hamiltoniano de menor costo. Se aprovecha las propiedades geométricas cuando los costos son euclidianos.

Descripción:

Paso 1: Seleccionar arbitrariamente un nodo como el depósito central al cual denotamos como nodo 1.
Paso 2: Calcular $S(i,j)=C(1,i)+C(1,j)-C(i,j)$ $\forall i,j=2,3,\ldots,n$
Paso 3: Listar $S(i,j)$ en orden decendiente.
Paso 4: Empezar con el primer elemento de la lista y mover hacia abajo, formando subcircuito mayores por ligar nodos apropiados i y j. Repetirlo hasta formar un circuito.

III.10 Algoritmo de Christofides:

Propósito: Dada una matriz de costos simétrica, determinar el circuito hamiltoniano de menor costo. Se combinan el árbol de expansión mínima y el apareamiento perfecto de los nodos de grado non para obtener el circuito hamiltoniano.

Descripción:

Paso 1: Encontrar el árbol de expansión mínima T de G.
Paso 2: Identificar todos los nodos de grado non en T. Resolver el apareamiento perfecto de costo mínimo en los nodos de grado non usando la matriz de costo original. Agregar los arcos de la solución al árbol T, obtener un ciclo Euler. En esta subgráfica, todos los nodos son de grado par. (Puede ser mayor que 2)
Paso 3: Eliminar los arcos con los cuales el grado es mayor que 2 y transformar el ciclo euclidiano al circuito hamiltoniano.

III.4 METODOS COMBINADOS

Generalmente, los algoritmos de este tipo aprovechan un circuito inicial obtenido por un método de construcción de circuito, y después, buscan una solución mejor usando uno o más algoritmos de mejoramiento de circuitos. Según los reportes de varios autores, los

algoritmos combinados son relativamente rápidos computacionalmente y logran una excelente aproximación a la solución óptima. Se tiene experiencia que un circuito inicial de "buena" calidad conduce mucho más rápido a una solución buena en el etapa de aplicar algún algoritmo de mejoramiento de circuitos.

III.11 Algoritmo combinado 1:

Propósito: Dada una matriz de costos simétrica, determinar el circuito hamiltoniano de menor costo. Se obtiene un circuito inicial por el método de construcción, y se mejora el resultado por el metodo de mejoramiento de circuitos.

Descripción:
- Paso 1: Resolver el problema con un algoritmo de construcción que se menciona anteriormente.
- Paso 2: Aplicar método k-óptimo.

III.12 Algoritmo combinado 2:

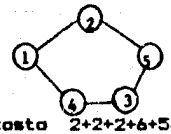
Propósito: Dada una matriz de costos simétrica, determinar el circuito hamiltoniano de menor costo.

Descripcion:
- Paso 1: Sólo aplicar el proceso de construcción para pocos nodos, y se agregan los nodos restante arbitrariamente al circuito.
- Paso 2: Usar k-óptimo para mejorar la solución.

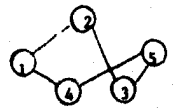
Ejemplo: resolver el problema de viajero usando la matriz de costos:

	1	2	3	4	5
1	-	2	3	5	4
2	2	-	2	4	2
3	3	2	-	4	2
4	5	4	6	-	4
5	4	2	2	4	-

Aplicando el algoritmo EL NODO PROXIMO MAS CERCANO mencionado al inicio de los métodos de insersión, nos da un circuito inicial:



con costo 2+2+2+6+5=17. Aplicando el algoritmo 2-óptimo, se eliminan (2,5), (3,4), se insertan (2,3), (4,5), el circuito resulta:



con costo=2+2+2+4+5=15

III.5 ASPECTOS COMPARATIVOS

Debido a los equipos limitados y el tiempo que tenemos, es imposible realizar todos los experimentos para probar el esfuerzo computacional de cada algoritmo. Sólo podemos citar los resultados obtenidos por otros autores que se mostrarán a continuación. Cabe señalar que la comparación a veces es difícil ya que los autores usan diferentes computadoras. ([10]) En la tabla 1, se muestran las longitudes de los circuitos obtenidos por diferentes algoritmos:

Tabla 1.

	Numeración de problema / No. nodos		
	1/25	2/42	3/70
Mejor solucion conocida	1711	699	684
2-optimo	1711	707	684
3-optimo	1719	699	698
Clarke & Wright	1750	702	700
Vecino mas cercano	1772	864	719
Insersion mas cercana	1917	776	748
Insersion mas lejana	1711	717	689
Insersion arbitraria	1787	720	698

En la tabla 2, se exhiben los porcentages de la solución de los métodos sobre la mejor solución conocida. Por ejemplo: la mejor solución conocida es 100, y la longitud obtenida por un algoritmo es 102, entonces el porcentage es 2%.

Revisando las tablas 1 y 2, notamos que los algoritmos de construcción tienen dificultades con la calidad de la solución. El porcentage es mayor que 3%. El Vecino más cercano, el Insersión más cercana no funcionan bien en todos los casos. El algoritmo de Clarke & Wrigth, el Insersión más lejana y el Insersión arbitraria sorpresivamente trabajan bien apesar de sus limitaciones. La técnica de Urna convexa y la de 2-óptimo son eficientes y obtienen buena calidad de solución.

Los métodos compuestos probados trabajan muy bien. El porcentage sobre la mejor solución conocida es menor que 3%. En todas las pruebas, solo hay tres casos en que el porcentage es mayor que 3%. Durante los experimentos, se detecta también que una solución inicial buena conduce a un buen resultado para los algoritmos compuestos.

Tabla 2

	Numeración de problema / No.nodos				
	3/100	4/100	5/100	6/100	7/100
Mejor solución conocida	21282	22148	20749	21294	22068
2-óptimo	1.11%	3.02%	0.51%	3.27%	3.24%
3-óptimo	7.82%	2.84%	3.30%	1.15%	1.40%
Clarke & Wright	1.62%	3.74%	6.37%	3.41%	2.85%
Vecino más cercano	16.67%	16.85%	13.35%	16.51%	13.27%
Inserción más cercana	18.69%	17.78%	22.96%	14.44%	20.33%
Inserción más lejana	5.14%	6.94%	3.17%	1.99%	7.42%
Inserción arbitraria	4.46%	3.47%	3.28%	2.90%	5.03%
Algoritmo de Christofides	7.53%	3.90%	5.36%	14.44%	8.51%
Urna convexa	3.64%	2.49%	2.54%	2.35%	3.45%
Métodos compuestos					
Vecino más cercano, 2-opt, 3-opt	3.64%	2.49%	2.54%	2.35%	3.45%
Urna convexa, 2-opt	0.94%	1.91%	1.60%	2.04%	3.22%
Urna convexa, 3-opt	0.37%	1.43%	1.06%	0.35%	2.46%
2-opt, 3-opt	0.81%	1.41%	0.53%	1.74%	0.18%
Christofides, 2-opt, 3-opt	2.51%	1.37%	1.53%	0.17%	3.03%
Inserción arbitraria desde 10 nodos, 3-opt para la mejor	1.42%	1.48%	2.57%	1.13%	1.59%
Inserción arbitraria desde 10 nodos, 3-opt para cada una	0.56%	1.30%	1.20%	0.66%	1.06%
Inserción más lejana desde 10 nodos, 3-opt para la mejor	1.17%	3.06%	0.58%	2.34%	2.60%
Inserción más lejana desde 10 nodos, 3-opt para cada una	0.46%	1.54%	0.49%	0.45%	0.98%

Hasta ahora, las distancias cumplen la desigualdad de triangulo, y las gráficas son dirigidas. En la tabla 3, se muestran los resultados de los algoritmos aplicados a cinco problemas. Las distancias de los 5 problemas son simétricas, y son generadas desde una distribución probabilística uniforme y discreta. Los rangos de la distancia de los cinco problemas son: 0-99, 0-200, 0-300, 50-350, 100-500.

En la tabla 3, se observa que en general, los algoritmos compuestos operan mejor que otras técnicas. Sin embargo, las ventajas no son tan consistentes de un problema a otro como en el caso donde se cumpla la desigualdad de triangulo.

Tabla 3

	Numeración de problema / No.nodos				
	9/100	10/100	11/100	12/100	13/100
2-óptimo	338	657	837	6065	11227
3-optimo	241	451	573	5736	10779
Vecino más cercano	373	748	906	6037	11200
Inserción más cercana	603	1104	1681	6780	12188
Inserción más lejana	574	1201	1651	6874	12300
Inserción arbitraria	623	1087	1677	6810	12288
Inserción arbitraria desde 10 nodos iniciales, 3-ópt desde la mejor solución	211	415	520	5703	10787
Inserción arbitraria desde 10 nodos iniciales, 3-ópt desde cada solución	207	406	520	5703	10787
Inserción más lejana desde 10 nodos, 3-opt para la mejor	222	424	561	5721	10846
Inserción más lejana desde 10 nodos, 3-ópt para cada una	199	410	554	5696	10803

En la tabla 4, se ilustran la eficiencia y características de algunos algoritmos.

Otros criterios para la selección de un algoritmo son: la cota superior del error y el orden de números de cálculos requeridos. La cota de error se obtiene por suponer una distribución de probabilidad de las distancias, y frecuentemente usa la propiedad de triángulo. El orden de cálculo implica el tiempo consumido cuando se implanta el algoritmo en una computadora. Uno puede deducir fácilmente el número de operaciones a realizar para estimar el orden de tiempo consumido. Sin embargo, para algunos métodos, es muy laborioso calcular la cota del error. Es decir, se desconoce todavía el comportamiento del error de algunos algoritmos.

Tabla 4

Algoritmos	N máximo resuelto	computadora	tiempo (min)	comentario
Programación Dinámica	13	7090	0.28	1.Consume mucha memoria para n=13, usa 32K 2.Tiempo de computo determinístico 3.Es bueno cuando n<10
Ramificación y Acotamiento	70	1620	103.5	1.Variación grande en el tiempo consumido 2.Es bueno cuando n es menor que 70 3.Cuando hay muchos ciclos con 2 arcos, se muestra mucha desventaja
Programación lineal	42	7094	5	1.Variación grande del tiempo consumido 2.El orden recomendable es menor que 30 3.Varios autores reportan desafavorable
r-óptimo	100	7094	30	1.Cuando r=3, $T=30n^3$ con probabilidad de ser optimo $2-n/10$ 2.Bueno cuando n está dentro de (20,100) 3.Normalmente, se toma r=2 o r=3
Método de penalización	60	CDC 6600	0.23	1.Para n<30, muchas veces resulta mejor la penalización positiva 2.Orden recomendable es (5,60) 3.Esta computadora es más rápida.

Nota: la computadora 7090 es aproximadamente 50 veces más rápida que 1620.

En la siguiente tabla, se muestran el orden de número de operaciones y la cota superior del error.

Algoritmo	Orden de número de operaciones	Cota de peor caso
Ramif. y acotamiento	exponencial	1
Vecino mas cercano	n^2	$0.5\times(\lg(n)+1)$
Clarke & Wright	$n^2\times\lg(n)$	-
Insersion más cercana	n^2	2
Insersion con costo mínimo	$n^2\times\lg(n)$	2
Insersion arbitraria	n^2	$2\ln(n)+0.16$
Insersion más lejana	n^2	$2\ln(n)+0.16$
Urna convexa	$n^2\times\lg(n)$	-
Christofides	n^3	1.5
k-óptimo (k<n/4)	n^k	$2(1-1/n)$
ALGO IV(k)	n^k	-
Doble giro	n^3	2
Karp*	$2(n-1)/(t-1)S(t)+O(n\lg(n))$	$1+t^{-0.5}$
Método de adecuación	n^3	-

- significa desconocido.
* t es el número de vértices en un rectángulo, S(t) es el orden de cálculos del algoritmo para resolver un subproblema.

En la siguiente tabla, se listan la calidad y el tiempo consumido de algunos algoritmos de solución. Todos ellos están implantados en Apple IIe con el lenguaje USCD-Pascal. El primer dato es el valor de función objetivo, y el segundo es el tiempo consumido.

ALGORITMOS	n = 10	n = 20	n = 50
2-óptimo	223 / 23"	536 / 80"	853 / 910"
Método de insersión	246 / 23"	360 / 95"	512 / 810"
Método de adecuación	231 / 32"	374 / 175"	447 / 815"
Método compuesto	173 / 27"	360 / 125"	512 / 810"
Branch & Bound	148 / 60"	- / >1800"	

CAPITULO IV PROBLEMA DE VIAJEROS MULTIPLES

El problema del viajero clásico considera que un solo viajero debe realizar la visita a todas las ciudades. En el problema de viajeros multiples se disponen de m viajeros que deben visitar n ciudades. Cada ciudad tiene que ser visitada exctamente una vez por uno de los m viajeros. Se puede suponer que existe un costo adcional fijo por tener activo un viajero. El problema es determinar cuántos viajeros usar y cuales son las rutas a recorrer para que el costo total sea mínimo.

El problema de viajeros multiples tiene diferentes variaciones. Estas se clasifican de la siguiente manera:

1) Todos los viajeros tienen que partir de una "Ciudad Base".
2) Al menos r viajeros tienen que ser activos.
3) En un caso más general, los viajeros pueden partir de diferentes ciudades

En este capítulo, analizamos, en la primera sección, la solución del problema de viajeros multiples que tienen que partir de una ciudad base. La idea principal de este método es transformar el problema actual en un problema de viajero simple, y después aplicar cualquier método que explicó en el capítulo III. La segunda sección consiste de una modificación de dicha transformación para resolver el caso de usar al menos r viajeros. En la tercera sección, se proporcionan los métodos para resolver la tercera variación del problema de viajeros multiples.

IV.1 LOS VIAJEROS PARTEN DE UNA CIUDAD BASE

Una manera eficiente para resolver el problema de viajeros multiples con la misma ciudad base es transformarlo en un problema de viajero simple (o estandar). Para ello se define el concepto de *r-ruta* que vamos a usar más adelante en esta sección.([14])

Un conjunto de r subcircuitos dirigidos en una gráfica G(N,A) se denomina una *r-ruta* si:

i) Cada subcircuito debe tener el nodo 0 como su nodo inicial;
ii) Cualquier nodo de la gráfica diferente del nodo 0 es visitado exactamente una vez por uno de los r subcircuitos.

Por la definición anterior, una r-ruta ($1 \leq r \leq m$) es una solución factible para el problema de viajeros. En particular, una 1-ruta llamada simplemente "circuito", es una solución factible para el problema de viajero simple (o estandar).

La transformación del problema de viajeros multiples al caso simple es como sigue: suponga que los m viajeros están indexados desde 0 a m-1 tal que

$$D(0) \leq D(1) \leq D(2) \leq \ldots \leq D(m-1)$$

Donde D(i) es el costo adicional si el viajero i es usado para visitar ciudades. Entonces el problema es equivalente a encontrar una r-ruta que minimiza:

$$Z = \sum_{k=1}^{r} Z(k) + \sum_{k=0}^{r-1} D(k) \qquad \text{(IV-1)}$$

para todo r ($1 \leq r \leq m$) donde Z(k) es el costo del k-ésimo subcircuito en la r-ruta. Se demostrará que r-ruta que minimice Z sobre r se puede obtener desde el circuito mínimo en una gráfica extendida G(N',A').

Construcción de la gráfica G(N',A'):

El conjunto de nodos N' es obtenido por agregar m-1 nodos adicionales a N, los cuales se denotan por -1, -2,..., -(m-1). El conjunto de arcos A' consiste de:

(1) Todos los nodos en A;
(2) Arco (-i,j) para cada nodo adicional (-i) si A contiene el arco (0,j) para j=1,..n.
(3) Arco (j,-i) para cada nodo adicional (-i) si A contiene el arco (j,0) para j=1,...,n.
(4) Arco (-i,-(i-1)) para i=1,2,..,(m-1).

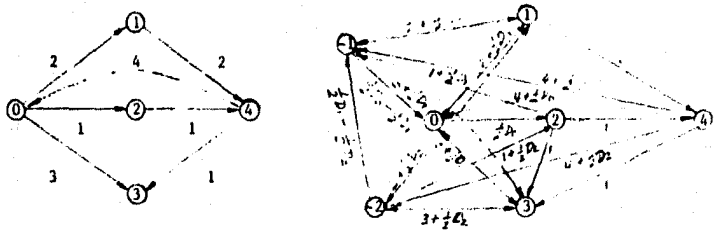


Figura IV.1

Los costos asociados a los arcos son:

$$C'(i,j)=C(i,j) \text{ para } i=1,\ldots,n;\quad j=1,\ldots,n \qquad \text{(IV-2)}$$

$$C'(-i,j)=C(0,j)+0.5D(i) \qquad \text{para} \begin{cases} i=0,1,\ldots,(m-1) \\ j=1,2,\ldots,n \end{cases}$$

$$C'(j,-i)=C(j,0)+0.5D(i) \qquad \text{(IV-3)}$$

$$C'(-i,-(i-1))=0.5D(i-1)-0.5D(i) \quad \text{para } i=1,2,\ldots,(m-1) \qquad \text{(IV-4)}$$

donde C(i,j) es la distancia de arco (i,j) en A y D(i) es el costo adcional del i-ésimo viajero.

Conviene señalar que las ecuaciones (IV-3) y (IV-4) son para sumar D(i) al costo del circuito en G(N',A') si y solo si un grupo de nodos positivos siguen inmediatamente al nodo (-i) en la visita. Esto implica que el viajero i visita este grupo de ciudades que siguen al nodo (-i).

Ejemplo: Considere la gráfica siguiente con tres viajeros mostrada en figura IV.1. Si consideramos el circuito (0,1,4,-2,-1,2,3,0) en G(N',A'). La interpretación es : El viajero 0 visita la ciudad 1 y la 4; El viajero 2 no visita ninguna ciudad; El viajero 1 visita la ciudad 2 y la 3. Se observa que D(0) es sumado al costo total, la mitad está en (0,1), y la otra mitad en (3,0). Un argumento semejante pasa con D(1), pero 0.5D(2) es sumado en (4,-2) y -0.5D(2) es sumado en (-2,-1), por tanto se cancela.

La optimalidad de un circuito en G=(N',A') respecto a la gráfica original es como sigue:

Teorema IV.1: Para cualquier r-ruta en G(N,A) para $1 \leq r \leq m$, existe un circuito en G(N',A') tal que

$$Z=\sum_{k=1}^{r} Z(k)+\sum_{k=0}^{r-1} D(k) \qquad \text{(IV-5)}$$

donde Z es el costo del circuito en G(N',A'), Z(k) es el costo del k-ésimo subcircuito en la r-ruta en G(N ,A) y D(k) es el costo asociadc con el viajero k.

Prueba: Sean los subcircuitos en la r-ruta numerados arbitrariamente de 1 a r. Se construye un circuito en G(N',A') de la siguiente manera:

A partir del nodo 0, el circuito recorre a lo largo de los nodos en el subcircuito 1 y regresa al nodo (-1); luego desde el nodo (-1) recorre a lo largo de los nodos en el subcircuito 2 y regresa al nodo (-2); continúa este proceso hasta recorrer el subcircuito (r-2) regresando al ciclo -(r-2); luego desde -(r-2) recorre los nodos en el subcircuito (r-1) y regresa al nodo (m-1); después recorre -(m-2), -(m-3), ..., -(r-1), y finalmente desde el nodo -(r-1) recorre a lo largo de los nodos del subcircuito r y regresa al nodo 0. Este recorrido es posible ya que en G(N',A') existen el arco (j,-i) y el (-i,j) para i=0,1,..-(m-1); en G(N,A) existen (j,0) y (0,j) respectivamente. De esa manera, D(k) es sumado al costo del circuito para k=0, 1,..., (r-1) por ecuaciones (IV-3) y (IV-4), Se cumple (IV-5) y termina la prueba.

Teorema IV.2: Para cada circuito en G(N',A') con costo Z, hay una r-ruta en G(N ,A) para algún r ($1 \le r \le m$) tal que

$$Z \ge \sum_{k=1}^{r} Z(k) + \sum_{k=0}^{r-1} D(k)$$

donde Z(k) es la distancia del k-ésimo ciclo en r-ruta.

Prueba: Si un grupo de nodos positivamente indexados siguen al nodo (-i) en un circuito dado en G(N',A') para $0 \le i \le (m-1)$, entonces el viajero i visita a ese grupo de ciudades; en caso contrario, el viajero i es ocioso. Dado un circuito en G(N',A'), sea r el número de viajeros activos para visitar ciudades de acuerdo con la interpretación anterior. Se construye r subcircuitos dirigidos, cada grupo de nodos es visitado por un viajero desde el nodo 0, y un número arbitrario de subcircuitos 1 a r. Entonces tenemos:

$$Z = \sum_{k=1}^{r} Z(k) + \sum_{k \in S} D(k) \ge \sum_{k=1}^{r} Z(k) + \sum_{k=0}^{r-1} D(k)$$

donde S es el conjunto de viajero activos, Z(k) es el costo del subcircuito k, y la desigualdad es derivada de la relación

$$D(0) \le D(1) \le \ldots \le D(r)$$

y esto termina la prueba.

Corolario 1: Si un circuito en G(N',A') es de costo mínimo, entonces

$$Z = \sum_{k=1}^{r} Z(k) + \sum_{k=0}^{r-1} D(k)$$

donde Z es el costo del circuito en G(N',A').

Prueba: Supongamos que Z es mínimo y $Z > \sum_{k=1}^{r} Z(k) + \sum_{k=0}^{r-1} D(k)$. Entonces existe un circuito de costo Z' tal que (Teorema IV.1)

$$Z' = \sum_{k=0}^{r} Z(k) + \sum_{k=0}^{r-1} D(k)$$

de donde $Z' < Z$, que es una contradicción.

Corolario 2: Si un circuito en $G(N',A')$ es de costo mínimo, entonces la r-ruta correspondiente en el teorema IV.2 minimiza Z sobre todo r.

Prueba: Como el circuito es de costo mínimo, la r-ruta obtenida en el teorema IV.2 satisface

$$Z = \sum_{k=1}^{r} Z(k) + \sum_{k=0}^{r-1} D(k) \quad \text{por el corolario 1.}$$

Supongamos que existe una r'-ruta para algún $r'(1 \leq r' \leq m)$ que satisface:

$$\sum_{k=1}^{r'} Z'(k) + \sum_{k=0}^{r'-1} D(k) < \sum_{k=1}^{r} Z(k) + \sum_{k=0}^{r-1} D(k)$$

donde $Z'(k)$ es el costo del k-ésimo ciclo en la r'-ruta. Existe un circuito en $G(N',A')$ con distancia Z' satisfaciendo (Teorema IV.1)

$$Z' = \sum_{k=1}^{r'} Z'(k) + \sum_{k=0}^{r-1} D(k)$$

Entonces $Z' < Z$, que es una contradicción.

NOTA ACLARATIVA:

Por el corolario 2, el problema de viajeros multiples puede ser resuelto como un problema de viajero simple en $G(N',A')$.

IV.2 AL MENOS r VIAJEROS DEBEN ESTAR ACTIVOS

En el caso de que al menos r viajeros tengan que salir a efectuar las visitas, la solución es bastante sencilla si usamos la metodología de la sección IV.1. Específicamente, La transformación es como sigue:

El conjunto de nodos N' en la gráfica $G(N',A')$ es obtenido al agregar m-1 nodos adicionales a N, los cuales se denotan por -1, -2,..., -(m-1). El conjunto de arcos A' en $G(N',A')$ contiene:

(1) Todos los nodos en A;
(2) Arco (-i,j) para cada nodo adicional (-i) si A contiene el arco (0,j) para j=1,..n.
(3) Arco (j,-i) para cada nodo adicional (-i) si A contiene el arco (j,0) para j=1,...,n.
(4) Arco (-i,-(i-1)) para i=r+1,r+2,..,(m-1).
(5) Arco (-r,0)

Los costos asociados a los arcos son:

$C'(i,j) = C(i,j)$ para $i=1,...,n$; $j=1,...,n$

$C'(-i,j) = C(0,j) + 0.5D(i)$ $\quad i=0,1,...,(m-1)$ para $j=1,2,...,n$

$C'(j,-i) = C(j,0) + 0.5D(i)$

$$C'(-i,-(i-1))=0.5D(i-1)-0.5D(i) \quad \text{para } i=r+1,r+2,\ldots,(m-1)$$

$$C'(-r,0)=0.5(D(0)-D(r))$$

Se puede observar que si se usa el arco $(-i,-(i-1))$, se desactiva el viajero i, ya que se suma un costo negativo $0.5D(i)$ al costo total.

En nuestro caso, como r viajeros deben estar activos para recorrer ciudades. Si no agregamos el arco $(-i,-(i-1))$, no se suma $0.5D(i)$ al costo total, y por propiedadad del problema de viajero, el nodo -i tiene que ser visitado una vez. Por lo anterior, el costo total se incrementa una cantidad igual a:

$$[0.5D(i)+C(i,j)]+[0.5D(i)+C(k,i)]$$

donde j es el nodo sucesor de i, k es el nodo antecesor de i. Por tanto, el viajero i es activo.

IV.3 LOS VIAJEROS PUEDEN PARTIR DE DIFERENTES CIUDADES

Cuando los viajeros pueden estar en diferentes bases, se tiene un caso más general dentro de los problemas de viajeros multiples. De hecho, el problema de viajero con una sola base no es aplicable en algunas situaciones reales. Por ejemplo: se requieren que varios agentes viajeros efectúen una visita a n ciudades. Obligar a los viajeros que regresen a un mismo lugar origen puede resultar más costoso que usar un solo viajero ([16]). En este tipo de problema, el esquema de solución es colocar los r viajeros en r ciudades diferentes, y asignar a cada uno de ellos las ciudades a visitar. En esta sección, se proporcionarán algunos algoritmos para resolver el problema de viajeros multiples con diferentes ciudades base.

IV.3.1 Algoritmo de transformación:

Propósito: Determinar la solución de problema de viajeros multiples cuando el número de viajeros es dos. Los viajeros tienen pueden partir de diferentes ciudad base. La idea principal es transformar el problema de viajero multiple a un problema de viajero simple. Una vez que el problema en cuestión está transformado, puede ser resuelto por un método exacto o heurístico. Pero este algoritmo tiene un limitante, que es aplicable sólo para m=2. ([16])

Descripción:

Paso 1: Para cada $j \in N$, eliminar los arcos $(j,1)$ y $(j,0) \in A$;
Paso 2: Para cada $j \in N$, si $(j,0) \in A$, agregar un arco dirigido $(j,1)$ con el costo $C'(j,1)=C(j,0)$;
Paso 3: Para cada $j \in N$, si $(j,1) \in A$, agregar un arco dirigido $(j,0)$ con el costo $C'(j,0)=C(j,1)$;
Paso 4: La distancia de los otros arcos está dada por:
$C'(i,j)=C(i,j)$;
Paso 5: Aplicar un algoritmo mencionado en el capítulo III para resolver el problema de viajero simple.

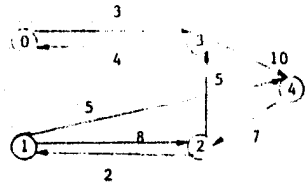


Figura IV.2

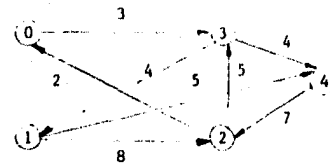


Figura IV.3

Ejemplo: En la figura IV.2, se muestra la grafica original de un problema de viajeros multiples con m=2. En la figura IV.3, se muestra la conversión a un problema de viajero simple.

IV.3.2 Ramificación y acotamiento:

En este algoritmo, primero se resuelve el problema de asignación, y después tratar de eliminar los subcircuitos innecesarios que sólo queden m subcircuitos.

Propósito: Dada una matriz de costos, determinar la solución óptima cuando de disponen de m viajeros. Ellos pueden partir de diferentes ciudades no especificadas. El costo asociado para tener activo a un viajero es cero. Los viajeros activos tienen que visitar más de una ciudad.

Descripción:

Paso 1. Operación de acotamiento:
Resolver el problema de asiganción bajo la matriz de costos. La solución del problema es la cota inferior del problema de viajeros.

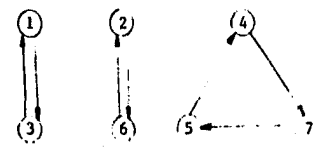
Paso 2. Operación de ramificación:
Si el número de de subcircuitos en la solución de asignación es menor o igual que m, entonces, la solución óptima es obtenida.

En caso contrario, la operación de ramificación es la siguiente: para uno de los subcircuitos, suponer que el costo de cada uno de los arcos es infinito, de esta manera generar diferentes alternativas.

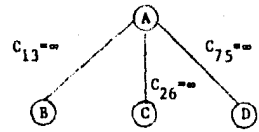
Ejemplo: Supongamos que tenemos 2 viajeros y la matriz de costo es la siguiente:

	1	2	3	4	5	6	7
1	-	11	3	8	21	9	12
2	11	-	19	20	14	2	9
3	3	19	-	15	24	10	20
4	8	20	15	-	1	14	3
5	21	14	24	1	-	19	2
6	9	2	10	14	19	-	10
7	12	9	20	3	2	10	-

La solución de problema de asignación es:



La forma de ramificación está mostrada en la siguiente figura:



IV.3.3 Algoritmo de insersión:

Propósito: Dada una matriz de costos, encontrar los circuitos hamiltonianos que visitan las n ciudades usando r viajeros. Los viajeros pueden partir de diferentes ciudades base. Se supone que no hay costo adicional por activar un viajero. El algoritmo funciona de la siguiente manera: se obtinen primero r ciudades que tienen una mayor distancia entre si, se incluyen las otras ciudades en una de las r ciudades bajo una regla definida.

Descripción:

Paso 1: Empezar con una subgráfica Si que consiste de un nodo i_1, k=1.

Paso 2: Encontrar el nodo i_{k+1} que está más lejana de Si,...Sk, o sea

$$\max_{i_{k+1}} \sum_{l=1}^{k} C(i_{k+1},i_l)$$

Paso 3: Si k=m-1, entonces ir al paso 4; caso contrario, incrementa k en 1, ir al paso 2.

Paso 4: Para cada Se, encontrar un nodo más cercano je al nodo ie, formar ie - je - ie.

Paso 5: Encontrar el nodo k que no está dentro de los Se, y el arco (i,j) de tal manera que minimice

$$C(i,k)+C(k,j)-C(i,j)$$

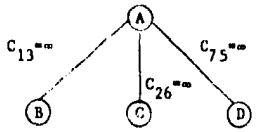
para todo i,j∈Se, e=1,2,...,m.
Insertar k entre i y j.

Paso 6: Si todos los nodos están incluidos en uno de los subconjuntos, termina el proceso; caso contrario, ir al paso 5.

Ejemplo: Resolver el problema de 2 viajeros con la matriz de costo dada a continuación:

	1	2	3	4	5	6	7
1	-	8	3	12	7	6	9
2	8	-	5	6	3	19	5
3	3	5	-	18	8	20	3
4	12	6	18	-	17	2	7
5	7	3	8	17	-	3	10
6	6	19	20	2	3	-	9
7	9	5	3	7	10	9	-

La forma de ramificación está mostrada en la siguiente figura:



IV.3.3 Algoritmo de inserción:

Propósito: Dada una matriz de costos, encontrar los circuitos hamiltonianos que visitan las n ciudades usando r viajeros. Los viajeros pueden partir de diferentes ciudades base. Se supone que no hay costo adicional por activar un viajero. El algoritmo funciona de la siguiente manera: se obtinen primero r ciudades que tienen una mayor distancia entre sí, se incluyen las otras ciudades en una de las r ciudades bajo una regla definida.

Descripción:
Paso 1: Empezar con una subgráfica Si que consiste de un nodo i_1, k=1.
Paso 2: Encontrar el nodo i_{k+1} que está más lejana de Si,...Sk, o sea

$$i_{k+1} \quad \text{Max} \sum_{l=1}^{k} C(i_{k+1}, i_l)$$

Paso 3: Si k=m-1, entonces ir al paso 4;
caso contrario, incrementa k en 1, ir al paso 2.
Paso 4: Para cada Se, encontrar un nodo más cercano je al nodo ie, formar ie - je - ie.
Paso 5: Encontrar el nodo k que no está dentro de los Se, y el arco (i,j) de tal manera que minimice

$$C(i,k)+C(k,j)-C(i,j)$$

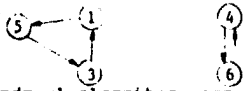
para todo i,j∈Se, e=1,2,...,m.
Insertar k entre i y j.
Paso 6: Si todos los nodos están incluidos en uno de los subconjuntos, termina el proceso;
caso contrario, ir al paso 5.

Ejemplo: Resolver el problema de 2 viajeros con la matriz de costo dada a continuación:

	1	2	3	4	5	6	7
1	-	8	3	12	7	6	9
2	8	-	5	6	3	19	5
3	3	5	-	18	8	20	3
4	12	6	18	-	17	2	7
5	7	3	8	17	-	3	10
6	6	19	20	2	3	-	9
7	9	5	3	7	10	9	-

Los subconjuntos iniciales son:

Siguiendo el paso 5, se encuentra que k=5, i=1, j=3,
Por tanto, se inserta 5 entre 1 y 3.



Aplicando el algoritmo, nos resulta finalmete:

IV.3.4 Algoritmo de Partición:

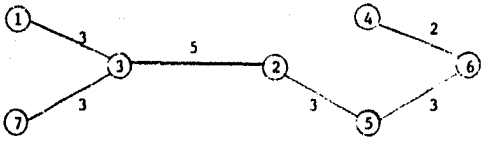
Propósito: Dada una matriz de costos simétrica, encontrar los circuitos hamiltonianos que visitan las n ciudades usando r viajeros. Los viajeros pueden partir de diferentes ciudades base. Se supone que no hay costo adicional por activar un viajero. La ideal fundamental del algoritmo es particionar el conjunto de nodos en r subconjuntos aprovechando la propiedad del árbol de expansión mínima. Se resuelve los r subproblemas individualmente.

Descripción:
Paso 1: Obtener el árbol de expansión mínima bajo la matriz de costo C.
Paso 2: Eliminar los m-1 arcos más pesados. De este modo, se divide el conjunto de nodos en m subconjuntos.
Paso 3: Aplicar cualquier algoritmo desarrollado en el capítulo III para resolver el problema de viajero simple para cada uno de los m subconjuntos de nodos.

Ejemplo: Sean m=2, y la matriz de costos:

	1	2	3	4	5	6	7
1	-	8	3	12	7	6	9
2	8	-	5	6	3	19	5
3	3	5	-	18	8	20	3
4	12	6	18	-	17	2	7
5	7	3	8	17	-	3	10
6	6	19	20	2	3	-	9
7	9	5	3	7	10	9	-

El árbol de expansión mínima es:

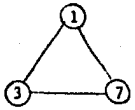


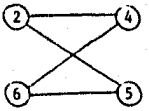
El arco más pesado es (2,3), se elimina. Se divide en

	1	3	7
1	-	3	9
3	3	-	3
7	9	3	-

	2	4	5	6
2	-	6	3	19
4	6	-	17	2
5	3	17	-	3
6	19	2	3	-

Entonces, la solución es:





Costo total=34.

CAPITULO V CONCLUSIONES Y EXTENSIONES

En esta tesis, hemos analizado las bases teóricas del problema del viajero y sus métodos de solución. Junto con los programas implantados hemos efectuado una clasificación y comparación tanto para los métodos exactos como para los aproximados, basándoros en su estructura y eficiencia. Se ha ilustrado la solución del problema de viajeros multiples por dos maneras: conversión al caso de viajero simple, y el tratamiento directo del problema.

El método exacto clásico analizado con mayor detalle es ramificación y acotamiento desarrollado por John Little. Tal método es de gran utilidad cuando se requiere una solución exacta. En el programa implantado en el trabajo, se almacenan la matriz reducida y los arcos de decisión que salen de cada nodo en el árbol de solución. De este modo, se consume menos tiempo en la obtención de la solución. Sin embargo, se ocupa mucho espacio de memoria. Otra manera de implantar el algoritmo es almacenar únicamente los arcos de decisión que salen de cada nodo en el árbol de solución. El objetivo de esta implantación es ocupar menos espacio de memoria. Pero el proceso de solución es más lento porque cuando se recupera la información de un árbol de solución, se tiene que reducir nuevamente la matriz de costos. Con esto, quisieramos decir que aunque los algoritmos están desarrollados, aún se puede hacer muchas cosas en la implantación en una computadora.

Dentro de los algoritmos heurísticos, la clase de mejoramiento de circuito nos ofrece la ventaja de que se puede terminar el proceso de solución en cualquier momento, ya que siempre estamos trabajando con un circuito hamiltoniano factible. Los métodos de eliminación de subcircuitos obtienen inicialmente soluciones localmente óptimas, y luego conectan los subcircuitos para formar un circuito hamiltoniano. Este tipo de algoritmos decomponen un problema en varios suboproblemas, la solución exacta de cada subproblema es fácil de obtener. Muchas veces, las soluciones locales son adecuadas. Los algoritmos de constricción del circuito proporcionan soluciones bastante aproximadas a la óptima, además de comsumir poco tiempo y memoria de computadora. Por último, los métodos combinados aprovechan un circuito inicial obtenido por un método de construcción, y buscan una solución mejor usando algún método de mejoramiento de circuitos. Estos métodos logran una excelente aproximación a la solución óptima y el tiempo consumido es considerablemente menor comparando con el del método exacto.

El problema de viajero con diferentes ciudades base es difícil de resolver. El método de ramificación y acotamiento es presentado en el trabajo. Se proporcionan dos métodos heurísticos: el Algoritmo de Insersión y el Algoritmo de Partición. Con estos métodos, podemos resolver el problema de viajeros multiples con diferentes ciudades base satisfactoriamente. Usando los programa

implantados, podemos estimar la eficiencia de estos algoritmos. Sin embargo, todavía se desconoce el comportamiento de la cota de error en el peor caso. Para calcular esta cota, se requiere un análisis complicado de probabilidad.

Los programas están implantados en la microcomputadora Apple y sus compatibles. Debido a la naturaleza de ésta, no se puede esperar una gran capacidad de almacenamiento y velocidad de proceso. Por lo que sólo recomendamos usar los programas para problemas pequeños o medianos. Se puede observar que los algoritmos presentados en el trabajo tiene muchas relaciones con otros problemas de optimización. Esto puede servir para motivar estudiantes que desean ampliar su conocimiento en el campo de optimización discreta.

APENDICE I CONCEPTOS BASICOS DE GRAFICA

El lenguaje de gráficas ha hecho posible representar la estructura de un gran número de situaciones de una manera simple. Ejemplos clásicos de tales representaciones son:

a) Red de comunicaciones tales como líneas de ferrocarril, flujos de información, etc.
b) Relación binaria asociada a relaciones algebraicas, sociológicas, regla de juegos y control de instalaciones.
c) Representación de las transiciones de estados en una cadena markoviana.

El estudio básico de los conceptos de gráficas que se describe en este apéndice hace especial énfasis en las definiciones que se usan en este trabajo.

En este apéndice se describe el concepto de gráfica dirigida y no-dirigida así como los parámetros típicos que se le asocian como grado de entrada y salida de un nodo y la manera de nombrar conjuntos específicos de nodos y arcos como trayectoria, ciclo, circuito, entre otras.

Una *gráfica* denotada G=[N,A] consiste de un conjunto N cuyos elementos se denominan *vértices* (o nodos) y un conjunto A de pares ordenados de N denominados *arcos*. ([2])

Si denotamos por M=|A| es el número de vértices, se dice que la gráfica es de *orden M*. En general, se supone que los vértices son numerados i=1,...,M. Si N y A son finitos, se dice que G es una *gráfica finita*. Si a=(i,j) es un arco de G, entonces i es el nodo *inicial* del arco a y j es el *nodo final* de a.

Gráficamente, los vértices pueden ser representados por puntos, y cada arco a=(i,j) se representa por una flecha que conecta los puntos i y j. La dirección de tal flecha es de i a j. En el caso de que i=j, o sea que los nodos inicial y final de un arco coinciden, el arco se llama *rizo*. Sean a, b dos arcos diferentes, si a=(i,j), b=(i,j), entonces a y b son *paralelos*.

El nodo j es llamado un *sucesor* de i si existe un arco con el nodo inicial i y el nodo final j. El conjunto de sucesores de un nodo i∈N es denotado por q(i). La transformación q:N --> R es *mapeo multi-valuado*, esto es, mapea subconjuntos de N a los números enteros positivos. Si existe un arco (j,i), j es *antecesor* de i. El conjunto de antecesores de i ∈ N se denota por $q^{-1}(i)$ donde q^{-1} es mapeo inverso del multi-valuado q.

Ejemplo A.I.1: En la siguiente gráfica, 2 es el nodo inicial del arco a_3 y 3 el nodo final: a_4 es un rizo; $q(1)=\{2\}$, $q^{-1}(1)=\{2,5\}$; y finalmente a_4 y a_5 son arcos paralelos.

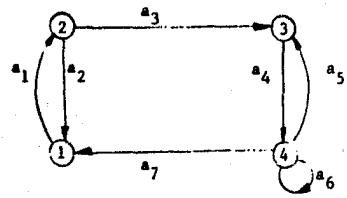


Figura A.I.1

En el estudio de ciertas propiedades de gráficas, la dirección de los arcos no es tan relevante. Por eso, se introduce el concepto de gráficas no dirigidas y algunas definiciones asociadas con éstas.

Un par no ordenado (i,j) se llama *arista*, esto es, una arista es un arco sin importar la dirección. Gráficamente, el arista a=(i,j) es representado por una línea que une dos puntos i y j. Si una gráfica contiene únicamente aristas, se dice que es una *gráfica no dirigida*. En caso contrario, la gráfica es *dirigida*.

Una gráfica es *simple* si no contiene rizos, y no existe más de un arista entre cualquier par de vértices. Si a=(i,j) y b=(i,k) entonces se dice que las aristas a y b son *adyacentes*.

Ejemplo A.I.2: En la figura A.I.2 se muestra una gráfica no dirigida, donde $a_1,\ldots,a_7$ son aristas: la gráfica no es simple, ya que tiene rizo a_6 y además tiene más de un arista entre 2 y 1; finalmente a_4 y a_5 son adyacentes.

El *grado de salida* de un nodo i denotado por $d^+(i)$ es el número de arcos (o aristas) que tienen i como su nodo inicial.

El *grado de entrada* de i denotado por $d^-(i)$ es el número de arcos (o aristas) que tienen i como su nodo final.

El *grado del nodo i*, denotado por d(i) es el número de arcos (o aristas) que tienen i como su nodo inicial o final. De inmediato, se tiene que:

$$d(i)=d^+(i)+d^-(i)$$

Dado el subconjunto X c N, la *subgráfica generada* por X es la gráfica Gx=[X,Ax] cuyos vértices son elementos de X y cuyos arcos Ax son los arcos de G con vértice en X.

Sea G=[N,A] y V c A. La *gráfica parcial* generada por V c A es la gráfica con el mismo conjunto de vértices N como lo de G, pero sus arcos son de V (omitiendo los arcos A-V).

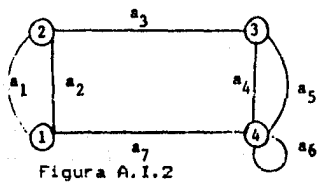


Figura A.I.2

Dados una gráfica G=[N,A], un subconjunto de vértices X c N, y un subconjunto de arcos V c A, la *subgráfica parcial* generada por X y V es la gráfica parcial de G: generada por V.

Ejemplo A.I.3: En la figura A.I.3, se muestra el concepto de subgráfica y gráfica parcial.

Una gráfica G=[N,A] es *completa* si para cualquier par de i,j, existe el arco (i,j).

Una colección de arcos (no necesariamente distintos) $a_1,\ldots,a_N$ de una gráfica se dice que forma una *progresión* de longitud N, si existe una sucesión apropiada de vértices $V_0,\ldots,Vn$ tal que $a_i=(Vi-1,Vi)$. Si $V_0 \neq Vn$, la progresión es *abierta*. En caso contrario, la progresión es *cerrada*. Si $Vi \neq Vj$ para todo i,j, la progresión es una *cadena*. La longitud N también se llama la *cardinalidad* de la cadena. Un *ciclo* es una cadena cuyos extremos se coinciden. Una *trayectoria* es una cadena cuyos arcos están dirigidos en la misma dirección.

Un *circuito* es una trayectoria en la cual el nodo inicial y nodo final se coinciden. O sea que, el punto inicial de la cadena es el punto final de la misma.

Una *ruta* es una progresión en que un nodo de la gráfica puede aparecer más de una vez, y que todos los arcos están dirigidos en la misma dirección.

Una gráfica es *conectada* si para todos los pares de vértices i y j, existe una cadena que une i y j.

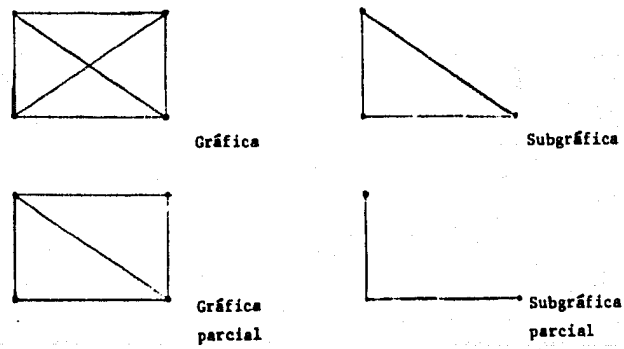


Figura A.I.3

Dados una gráfica G=[N,A], un subconjunto de vértices X c N, y un subconjunto de arcos V c A, la *subgráfica parcial* generada por X y V es la gráfica parcial de G= generada por V.

Ejemplo A.I.3: En la figura A.I.3, se muestra el concepto de subgráfica y gráfica parcial.

Una gráfica G=[N,A] es *completa* si para cualquier par de i,j, existe el arco (i,j).

Una colección de arcos (no necesariamente distintos) $a_1,\ldots,a_N$ de una gráfica se dice que forma una *progresión* de longitud N, si existe una sucesión apropiada de vértices $V_0,\ldots,Vn$ tal que $a_i=(Vi-1,Vi)$. Si $V_0 \neq Vn$, la progresión es *abierta*. En caso contrario, la progresión es *cerrada*. Si $Vi \neq Vj$ para todo i,j, la progresión es una *cadena*. La longitud N también se llama la *cardinalidad* de la cadena. Un *ciclo* es una cadena cuyos extremos se coinciden. Una *trayectoria* es una cadena cuyos arcos están dirigidos en la misma dirección.

Un *circuito* es una trayectoria en la cual el nodo inicial y nodo final se coinciden. O sea que, el punto inicial de la cadena es el punto final de la misma.

Una *ruta* es una progresión en que un nodo de la gráfica puede aparecer más de una vez, y que todos los arcos están dirigidos en la misma dirección.

Una gráfica es *conectada* si para todos los pares de vértices i y j, existe una cadena que une i y j.

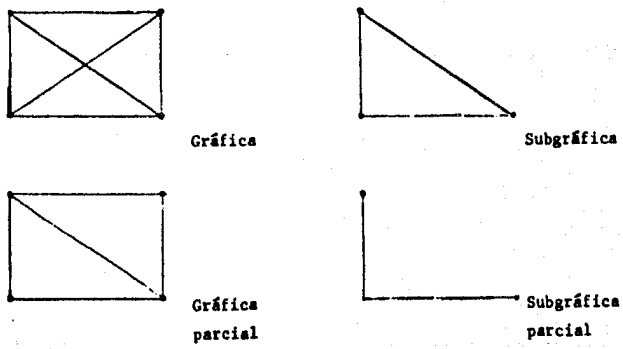


Figura A.I.3

Ejemplo A.I.4: En la figura A.I.1, la progresión ((1,2), (2,3), (3,4), (4,1) es cerrada; esta progresión también es una trayectoria, ya que los arcos tienen un mismo sentido; y es un circuito porque sus extremos coiciden. La cardinalidad es 4.

APENDICE II PROGRAMAS DE COMPUTADORA

Los programas de cómputo relacionados con los métodos de solución del problema del viajero están implantados en las microcomputadoras Apple II y sus compatibles. El lenguaje usado es USCD Pascal Ver.1.1. Sin embargo debido a la alta estandarización del lenguaje Pascal, estos programas son fáciles para trasladar en otras máquinas. El disco de programa es autoejecutable. Con los discos montados en los drives, sólo se necesita prender la máquina, aparece el menú de operaciones que guía al usuario.

En la figura A.2.1, se muestra un bosquejo de los programas implantados en la computadora. Para usar cualquiera de los programas, primero se tiene que crear un archivo donde se almacenan los elementos de la matriz de costos. El archivo consiste de varios registros. Un registro a su vez consiste de varios campos. En cada registro se almacena un renglón de la matriz. En cada campo se almacena un elemento del renglón. El nombre del archivo se forma por ocho caracteres alfanuméricos (el primero tiene que ser una letra), y una extensión. La extensión también contiene ocho caracteres alfanuméricos. Cuando se corre un programa, la computadora pide el archivo de datos. Entonces el usuario sólo necesita teclear el nombre del archivo seguido por <Return>, y la máquina empieza a correr el programa.

Al principio de la ejecución de cada programa, el usuario tiene opción que enviar la salida por video o por la impresora. Pues la computadora indica "Oprime I si desea imprimir el resultado". Se puede oprimir cualquier otra tecla en caso de querer observar el resultado en la pantalla.

En la biblioteca de la DEPFI (División de Estudios de Posgrado de Facultado de Ingeniería), se puede consultar los listados de los programas implantados.

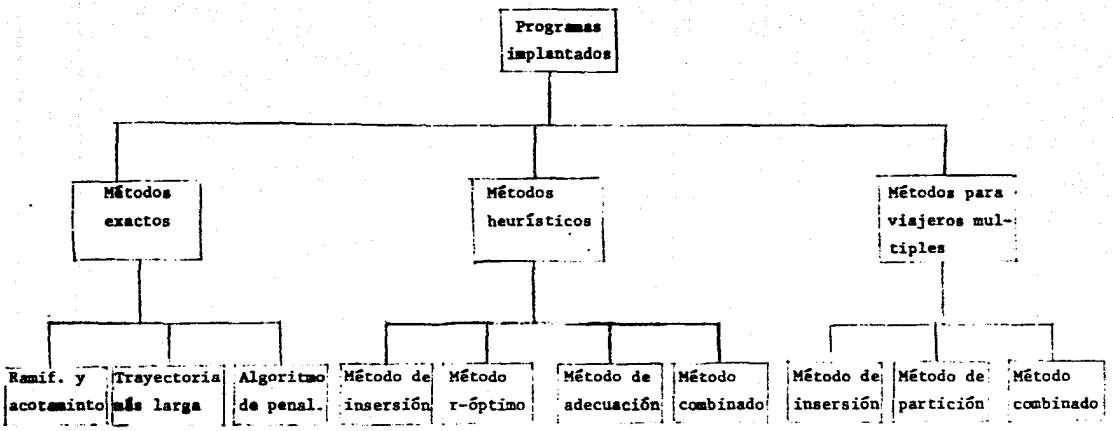


Figura A.2.1

1.RAMIFICACION Y ACOTAMIENTO
Nombre: BR&BND

Descripción: es un algoritmo exacto para resolver el problema del viajero y fue desarrollado por John Little. La ramificación se efectua sobre un elemento de la matriz reducida de acuerdo con cierta regla (ver Cap.II.2). La acotación se obtiene reduciendo la matriz de costos después de eliminar las hileras y columnas innecesarias.

Comentario: Este programa guarda la matriz de costos de todos los estados del árbol de solución, de esta manera, el programa corre con mayor velocidad, pero, usa mucha memoria. La gráfica puede ser dirigida o no.

Salida: Se imprime la matriz de costos de cada vértice del árbol de solución, junto con la trayectoria generada y el costo acumulado. Se imprime también el elemento sobre el cual se efectua la ramificación.

2.METODO E PENALIZACION
Nombre: PENALIZ

Descripción: El objetivo de este programa es el siguiente: dados el nodo inicial y el final, encontrar la trayectoria hamiltoniana óptima. Se genera primero el árbol de expansión mínima, y luego son penalizados los nodos que tienen más de dos arcos incidentes. Se modifica la matriz con la penalización, para volver a generar el árbol de expansión mínima hasta que el grado de los nodos sea dos (excepto el inicial y el final).

Comentario: En este programa, la computadora pregunta al usuario cual es el nodo inicial, el final y el valor de penalización que normalmente está entre 5 y 10. Un valor inadecuado puede provocar divergencia del algoritmo. La gráfica es no dirigida.

Salida: En cada iteración, la computadora imprime la conexión de los nodos para generar el árbol de expansión mínima bajo la matriz de costos actuales. Se indican los nodos que tienen más de dos arcos incidentes y el valor de penalización, y se imprime la nueva matriz de costos.

3.TRAYECTORIA MAS LARGA
Nombre: TRAML

Descripción: El programa genera la trayectoria más larga del nodo 1 al nodo n+1 (n es el número de ciudades del problema del viajero). El nodo n+1 es artificial. Se tiene que convertir el problema del viajero en uno de trayectoria más larga. Se cambia el nodo final a n+1 para todos los arcos que llegan al nodo 1.

Comentario: La matriz de costos es de dimensión n+1, y tiene que estar triangularizada. Es decir, después de permutar ciertos renglones y columnas, los elementos inferiores de la diagonal principal son de valor infinito. La gráfica es dirigida.

Salida: La computadora imprime la trayectoria que va desde el nodo 1 hasta el nodo i (i=2,...,n+1). Si la trayectoria que incluye todos los nodos es (1, j, ..., k, n+1), la solución del problema del viajero es (1, j, ..., k, 1).

4.METODO DE INSERSION (caso simple)
Nombre: MINSERS

Descripción: El programa se basa en el método de Inserción Más Cercana, funciona de la siguiente manera: se busca el nodo que está más cerca al subconjunto de nodos ya resueltos. Luego compara los valores de C(i,k) + C(j,k) - C(i,j). Seleccionar el par (i,j) que minimiza este valor. Insertar k entre ellos.

Comentario: En este algoritmo, la gráfica debe ser no dirigida.

Salida: Cada vez que se inserta k entre i y j, la computadora imprime estos 3 índices, y actualiza conexión de los nodos del subconjunto.

5.R-OPTIMO
Nombre:ROPTIMO

Descripción: El algoritmo parte de cualquier circuito inicial. Luego prueba eliminando r arcos del circuto y adiciona r arcos que no están en éste para obtener una nueva solución. Si este intercambio de arcos ofrece mejoramiento, se usa este circuito para seguir intentando la prueba. El algoritmo termina, cuando por intercambio de r arcos ya no se puede mejorar más y además se ha contemplado que el circuito también es (r-1)-óptimo.

Comentario: Debido a la recursividad del algoritmo, cuando r es grande, el programa corre con mayor lentitud. Por lo que el método sólo es implantado para r=2 y 3. La gráfica debe ser no dirigida.

Salida: Cuando un intercambio produce mejoramiento, la computadora imprime los arcos que serán eliminados y los arcos agregados, e imprime nuevamente la conexión entre los arcos.

6.METODO DE ADECUACION
Nombre: MADEC

Descripción: Primero, se obtiene la solución del problema de asignación usando el método húngaro. Si la solución es un circuito, entonces es óptima también para el problema del viajero. Si la solución consiste de subcircuitos, entonces se borra el arco más pesado de cada subcircuito, y se ligan los subcircuitos por adición de algunos arcos.

Comentario: La selección de los arcos agregados es arbitraria. El algoritmo trata de obtener soluciones localmente óptimas. La gráfica puede ser dirigida o no.

Salida: La computadora imprime todo el procedimiento de la generación de la solucion del problema de asignación. Después indica el arco más costoso en cada subcircuito. Al final imprime la solución del probleme de viajero.

7.METODO COMBINADO
Nombre: COMB

Descripción: Se usa inicialmente el algoritmo de Insersión Más Cercana para generar la solución inicial. Luego se aplica el algoritmo 2-óptimo para mejorar la solución.

Comentarios: En muchas ocaciones, el 2-óptimo ya no mejora nada la solución porque el circuito generado por Insersión Más Cercana es bastante bueno. La gráfica debe ser no dirigida.

Salida: La computadora imprime primero toda la salida mencionada en el algoritmo de Insersión Más Cercana, y luego la salida de r-óptimo.

8.METODO DE INSERSION (caso multiple)
Nombre: MINSERM

Descripción: El programa se basa en la implantación del método de Insersión Más Cercana, funciona de la siguiente manera:
Sea m el numero de viajeros. Primero se forman m subconjuntos distantes iniciales. Cada uno consiste de dos nodos. Después, se usa cualquier nodo k para calcular el mínimo de C(i,k)+C(j,k)-C(i,j). Donde i,j deben pertenecer al mismo subconjunto. Se inserta k entre i y j.

Comentario: Al principio, la computadora pide el número de viajeros en el problema. La gráfica debe ser no dirigida.

Salida: Cada vez que se inserta k entre i y j, la computadora imprime estos tres índices, y la conexión actualizada de los nodos del subconjunto.

9.METODO DE PARTICION (caso multiple)
Nombre: MPART

Descripción: Primero, se genera el árbol de expansión mínima, se elimina los m-1 arcos más costosos para dividir el árbol en m partes (donde m es el numero de viajeros). Se resuelven estos m subproblemas individualmente.

Comentarios: Inicialmente, la computadora pregunta al usuario el número de viajeros en el problema. La matriz de costos debe ser simétrica.

Salida: Se imprime el proceso de la generación del árbol de expansión mínima. Se indican los m-1 arcos mas costosos del árbol. Finalmente se imprime el procedimiento de solución de cada uno de

los subproblemas usando el método de 2-óptimo. Si una subgráfica contiene un solo nodo, ya no se imprime la solución de dicha subgráfica.

10.METODO COMBINADO (caso multiple)
Nombre: COMBM

Descripción: Se usa inicialmente el algoritmo de Insersión para generar la solución inicial. Luego se aplica el algoritmo 2-óptimo para mejorar la solución.

Comentarios: Al principio, el usuario tiene que proporcionar el número de viajeros. En muchas ocaciones, el 2-óptimo ya no mejora nada la solución porque el circuito generado por Insersión es bastante bueno. La gráfica debe ser no dirigida.

Salida: La computadora imprime primero toda la salida mencionada en el algoritmo de Insersión Más Cercana, y luego la salida de r-óptimo.

Bibliografía

[1] N.Chritofides, "Graph Theory: an algorithm approach", Academic Press, 1975. Cap.10

[2] M. Gordon, M. Minoux, "Graphs and Algorithms", Wiley Interscience, 1979, Cap 6,8,11,12

[3] Mandell Bellmore, John C. Malone, "Pathology of traveling salesman subtour elimination algotithms", Vol.19, 1971, pp 278-307

[4] Mandell Bellmore, G.L. Nemhauser, "The Traveling Salesman Problem: a survey", Op.Res. Vol.16, 1968, pp 538-558

[5] J.C. Picard, M.Queyranne, "The time-dependent traveling salesman problem and its application to the tardiness problem in the one-machine scheduling", Op.Res.Vol.26, 1978, pp 86-109

[6] Hardgrave, W.W., J.F. Pierce, "On then relation between the traveling salesman problem", Op.Res. Vol.10, 1967, pp 647-657

[7] Fuentes Maya, "Programación entera", DEPFI, Sept. de 1985

[8] C.E. Miller, R.A. Zemlin, "Integer programming formulation of traveling salesman Problem", A.C.M. vol.7, 1960, pp 326-329

[9] John D.C. Little et.al., "An algorithm for the traveling salesman problem", Op.Res. Vol.11, 1963, pp 972-989

[10] B.Golden et.al., "Approximate Traveling Salesman Algorithms", Op.Res. Vol.28, 1980, pp 694-711

[11] Gonzales R.H. "Solution to the traveling salesman problem by dinamic programming on the hypercube", Tech.Rep. No.18, O.R. Center, M.I.T., 1962

[12] M.L. Fisher et.al. "An analysis of approximations for finding a maximun weight hamiltonian circuit", Op.Res. Vol.27, 1979, pp 799-809

[13] M.H. Webb, "Some methods of producing approximate solutions to traveling salesman problem with hundreds of thousands of cities", Op.Res.Quaterly Vol.22, 1971, pp 49-66

[14] Mandell Bellmore, Saman Hong, "Transformation of multisalesmen problem to then standard traveling salesman problem", A.C.M. Vol.21, 1974, pp 500-504

[15] Saman Hong, Manfred W. Padberg, "A note on then symmetric Multiple Traveling Salesman Problem", Op.Res. Vol.25, 1977, pp 871-874

[16] M. R. Rao, "A note on the Multiple Traveling Salesman Problem", Op.Res. Vol.28, 1980, pp 628-632

[17] Merrill M. Flood, "The traveling salesman problem", Op.Res. vol.4, 1956, pp 61-75