UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DISEÑO DE PISTAS PARA CIRCUITOS IMPRESOS EN FORMA AUTOMATICA

T E S I S

Que para obtener el título de
INGENIERO EN COMPUTACION
p r e s e n t a n

JOSE OCTAVIO SALAZAR MARTINEZ
MARIO ALEJANDRO MOLINA GODINEZ

Director de tesis: Ing. Luis Alvarez Icaza

México, D. F. 1989

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

página

DISEÑO DE PISTAS PARA CIRCUITOS IMPRESOS EN FORMA AUTOMATICA

PROLOGO

Al iniciar este trabajo se desconocía la existencia de investigaciones con respecto al tema, queríamos diseñar una herramienta que ayude a elaborar circuitos impresos por computadora; se encontró que no éramos los únicos interesados en el tema, que ya existen planteamientos y algunos algoritmos. Estos en general, sólo buscan llegar al destino, sin importar como lo hacen, además de tener limitaciones. Nuestro trabajo se enfocó entonces a desarrollar un algoritmo más inteligente, adecuadamente fundamentado en la programación lineal.

Al término de la investigación pudimos darnos cuenta de que somos capaces de desarrollar cosas nuevas, y de utilizar nuestro ingenio sin necesidad de acoplarnos a métodos e ideologías extranjeras. Ponemos a su consideración nuestra investigación.

INTRODUCCION

"Vivimos en la época de la automatización. La batalla de la competencia obliga a una producción más rápida empleando métodos más racionales que exijan poca mano de obra. Hoy en día el circuito impreso es un componente determinante de la calidad de los aparatos electrónicos, además ofrece la posibilidad de fabricarlos en serie, abatiendo costos de producción y ampliando el mercado de ventas.

En lugar de la técnica de cableado tridimensional, utilizada anteriormente, el circuito impreso necesita una estructura bidimensional. Los conductores se instalan en un plano. A este respecto se utilizan pequeñas superficies de materia aislante cubiertas con líneas metálicas conductoras de corriente, en las que se sueldan directamente los componentes.

La ventaja de las placas de circuitos impresos consiste en que las propiedades eléctricas de los aparatos, en la fabricación en serie, son siempre constantes, así como los valores de capacitancia y de inductancia del cableado.

Incluso cuando son preparadas por personal no calificado, el porcentaje de defectos es muy pequeño. Además se necesita menos mano de obra, particularmente cuando las placas de circuitos impresos están equipadas con componentes que pueden ser soldadas en un ciclo de trabajo.

Para que los circuitos impresos puedan ser fabricados a escala industrial se tienen que efectuar distintos trabajos de diseño para colocar e interconectar los elementos; a menudo también se ha de hacer una elaboración individual en el laboratorio, en el taller técnico, incluso por parte del aficionado." (Ref.1)

El párrafo anterior citado textualmente de la Ref.1 no ha dejado de tener vigencia. Los avances más recientes, tales como los circuitos con alto grado de integraciòn y el cambio de estándares para la distribución de conectores externos, han impuesto restricciones aún más fuertes a los circuitos impresos.

La elaboración de circuitos impresos se realiza con

distintos procedimientos, que se muestran en la figura 0.1 (Ref. 2).

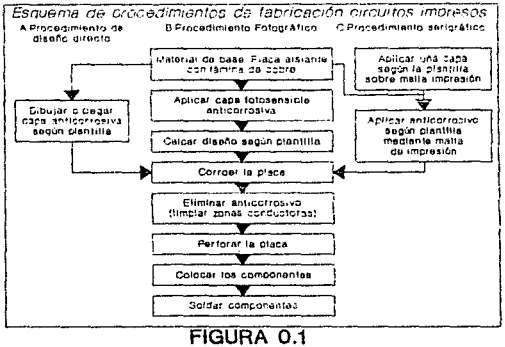

FIGURA 0.1

Como puede notarse, todos los procedimientos que se utilizan para la elaboración de circuitos impresos, tienen en común la utilización de una placa de material aislante revestida de cobre, sobre la cual se dispondrá posteriormente un material protector anticorrosivo con la forma del diseño del conductor; este diseño es una etapa previa, en la cual, se dibuja la ubicación de los componentes sobre la placa y la forma de las pistas que interconectan los elementos, a fin de

lograr el correcto funcionamiento del circuito.

El presente trabajo tiene el propósito de crear una herramienta para el diseño de circuitos impresos. Para ello se presenta una breve historia del surgimiento de los mismos y una revisión de lo que se puede encontrar al respecto actualmente.

Este trabajo está dedicado sólo a una de las fases que involucra la elaboración del circuito impreso, que es el trazo de las interconexiones.

Esta fase de trazado es muy laboriosa; se invierte una gran cantidad de tiempo para decidir cómo se debe realizar el diseño, pues existen diversos factores que intervienen en la solución; entre ellos se pueden mencionar el orden en que se realizan las interconexiones y la distribución de los elementos, por ello se ha intentado realizar el trazado en forma automática.

Se ilustran algunas técnicas para la elaboración del trazado, desde la elaboración manual hasta una

semiautomática; se analizan las ventajas y desventajas de dichas técnicas. A partir de esto se busca proponer una solución adecuada al problema del trazado que deberá satisfacer algunos requisitos mínimos. Con este último fín, se introduce una forma de valuación del trazado a través de los parámetros considerados importantes.

CONTENIDO.

En el primer capítulo se expondrá el surgimiento de los circuitos impresos. Se incluye además un glosario para limitar la semántica de algunos de los términos más empleados.

Debido a que en todas las técnicas para elaborar los circuitos impresos una parte importante lo constituye el diseño de los trazos de las interconexiones, en el segundo capítulo se incluyen algunas técnicas para resolver dicho problema en forma automática. Se describe en especial lo que existe en la actualidad, así como lo que se ha desarrollado en México.

En el tercer capítulo se expondrán alternativas de solución propuestas en este trabajo y se comentarán sus ventajas y desventajas.

En el siguiente capítulo se explica a detalle el algoritmo propuesto para solucionar los trazos de los circuitos impresos así como sus fundamentos matemáticos, se explica la validez del algoritmo de selección de la ruta más corta.

En el último capítulo se muestran la complejidad que podría tener el diseño de circuitos según el grado de independencia que se desee para el desarrollo de un sistema integral para diseño de circuitos impresos.

CAPITULO 1

ANTECEDENTES

1.1 DEFINICIONES.

a) CIRCUITO IMPRESO .- Tableta de material aislante que contiene las conexiones necesarias entre los elementos de un circuito por medio de un metal conductor adherido a la tableta en vez de utilizar cables.

b) PLACA .- Area sobre la cual se localiza el circuito impreso.

c) ISLA .- Elemento del circuito impreso que sirve para conectar un dispositivo electrónico a la placa.

d) PISTA .- Segmento de cobre que enlaza eléctricamente los elementos.

e) TRAZADO DE PISTAS .- Procedimiento para dirigir la

pista sobre la placa con el fin de establecer las interconexiones.

f) MODULO .- Segmento de cobre semejante a una isla pero sin perforación, que permite colocar puntas o pines de prueba para verificación de señales.

g) ZONAS PROHIBIDAS .- Area de la placa sobre la cual no pueden cruzar pistas.

h) ENTROPIA .- Medida de la calidad de una pista en relación con la forma de su recorrido.

i) CARA .- Es uno de los lados de la placa. La placa puede tener dos caras (en circuitos modernos, más de 2)

j) DISTANCIA .- Es la longitud que tiene la pista desde el principio hasta el fin de ésta.

k) ANCHO DE PISTA .- Son los distintos grosores que puede tener una pista, dependiendo de la

corriente que circulará en ésta.

l) HEURISTICO .- Calificativo para un procedimiento algorítmico basado comunmente en la experiencia que permite resolver un problema determinado. Por definición, no se puede tener una justificación total del procedimiento.

m) COLA .- Es una estructura de datos en la que el primer elemento que entra es el primero que sale.

1.2 FABRICACION DE CIRCUITO IMPRESO.

Antes del surgimiento de los circuitos impresos, la manera más común de realizar las conexiones entre componentes era por medio de cables. Esta es una tarea muy laboriosa y representa un gran problema para la fabricación de circuitos en serie, ya que provoca problemas como falsos contactos y errores en las conexiones entre los elementos, por lo que su desempeño requiere de mano de obra especializada.

A continuación se reproduce en la figura 1.1 un circuito realizado con cables. La intención es mostrar que a pesar del reducido número de interconexiones, el circuito involucra mucha mano de obra.

Circuito realizado con cables

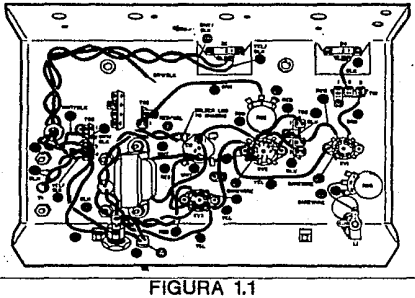

FIGURA 1.1

Desde el principio de la fabricación industrial de aparatos, los ingenieros y técnicos buscaron procedimientos para establecer las conexiones entre los componentes de los circuitos, es decir, para el cableado de un aparato a través de un proceso de trabajo.

En su forma fundamental el circuito impreso tiene ya casi ochenta años de existir. En el año de 1906 Edison y Sprague en los Estados Unidos de América informaron de la posibilidad de aplicar el cableado común con polvo metálico sobre aisladores.

Casi veinte años más tarde, el 19 de Marzo de 1925, Francis T. Harman obtuvo la patente en E.U.A. sobre uno de los procedimientos semejantes a la actual técnica de corrosión. En el año de 1927, Telefunken lanza al mercado el amplificador "Arcollette", cuyo cableado consistía de tiras de chapa de latón perforadas y adecuadamente configuradas. Este dispositivo marca el inicio del desarrollo de los circuitos impresos (Ref. 1).

Con el gran desarrollo de la electrónica, empezaron a surgir circuitos cada vez más complejos y difíciles de alambrar, por lo que se buscó una forma sencilla de realizar su cableado y además ahorrar tiempo y abatir el costo del proceso de fabricación.

Existen varias técnicas para la fabricación de circuitos

impresos, como la técnica de corrosión, la fotográfica y la del estarcido por mencionar algunas. En todas ellas, como ya se mencionó una fase común es el dibujo del circuito para poder después "imprimir" en la tarjeta los componentes y conexiones y facilitar la fabricación de los aparatos.

Una vez adoptada la técnica de circuitos impresos, surgieron algunos problemas en su implementación, que no existían al realizar el cableado manualmente como son las restricciones impuestas porque las rutas de conexión no deben cruzarse, ni pasar demasiado cerca unas de otras pues pueden interferir en el buen funcionamiento de los elementos del circuito.

En la figura 1.2 se muestra un circuito impreso, donde se observan las características antes mencionadas.

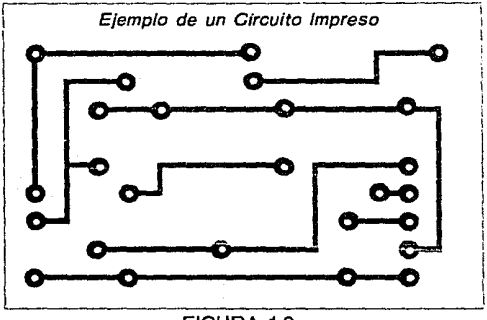

FIGURA 1.2

Al principio, la elaboración de los circuitos impresos se efectuaba en forma manual, pero esto es poco eficiente por el tiempo que requiere, y porque la solución del circuito puede estar "alejada" de la mejor solución. Con el advenimiento de la gran potencia de computación, surge la posibilidad de emplear la computadora como auxiliar para elaborar los circuitos impresos.

Para solucionar el problema del trazado de las conexiones de los circuitos con la ayuda de la computadora,

surgen de las investigaciones distintos algoritmos para su elaboración como son los algoritmos de Lee, Hitchock y Hightower que se describirán más adelante.

CAPITULO 2

ESTADO ACTUAL

2.1 DESCRIPCION DE SISTEMAS EXISTENTES.

Actualmente existen algunos sistemas que resuelven el trazado de las pistas en forma semiautomática, como es el caso del paquete "SMART-WORK".

En este tipo de paquetes el diseñador proporciona la posición de los elementos y la secuencia de conexión entre estos y se usan técnicas heurísticas para solucionar el problema. Sin embargo, son limitados pues no poseen la suficiente inteligencia para ofrecer la posibilidad de realizar cambios de plano (pasar de una cara de la tarjeta a la cara contraria).

Su mayor atractivo reside en que emplean la moderna tecnología de las computadoras personales con lo que aprovechan la velocidad de su procesador.

2.2 INVESTIGACIONES EN MEXICO.

En México, en el CIMAS (Centro de Investigación de Matemáticas Aplicadas y Sistemas, ahora IIMAS) de la UNAM, se desarrolló un sistema que resuelve el problema del trazado de las pistas en forma automática mediante programas implantados en una computadora Burroughs 6700. La investigación se realizó con el propósito de realizar las conexiones para una computadora digital; y de dicha investigación surgieron dos algoritmos, el Palas Atenea y el Alter Ego.

2.3 CARACTERISTICAS DEL ALGORITMO PALAS ATENEA.

Fué diseñado para resolver redes planas (usando una sola cara de la placa), es una variación del algoritmo de Lee. Utiliza una malla fundamental para la creación de las pistas y tiende a buscar la distancia mínima entre los nodos a unir, se dirige directamente del origen al destino, en caso de encontrar algún obstáculo o una celda ya ocupada, trata de esquivarlos en forma paralela y una vez rodeado el obstáculo toma el punto donde está como origen para alcanzar el

destino.

2.4 CARACTERISTICAS DEL ALGORITMO ALTER EGO.

A diferencia del algoritmo anterior, el Alter Ego, maneja las dos caras de una tarjeta para unir los puntos empleando solo dos direcciones (horizontal para una cara y vertical para la otra). Así trata de llegar al destino, ésto con el fin de evitar obstáculos o cruces de pistas.

2.5 ALGORITMO DE LEE.

Además de los algoritmos mencionados anteriormente se describirá el algoritmo de Lee, pues los demás son una versión de éste algoritmo con algunas variantes, ya sea para reducir requerimientos de memoria, como cantidad de operaciones involucradas para solucionar el problema.

Este algoritmo es altamente empleado en investigación de operaciones y teoría de gráficas, es un algoritmo sencillo que emplea la búsqueda exhaustiva y localiza la ruta de menor distancia entre dos puntos a unir en un plano, si esta

existe.

El plano empleado, se reduce a una malla cuadriculada donde cada cuadro representa una pista o isla, e incluye la distancia mínima entre dos elementos de un circuito impreso, en ésta malla se basa la búsqueda de las rutas. Dos celdas de la malla contienen los puntos a unir (origen y destino), en la figura 2.1 se muestra una malla indicando origen (A) y destino (B).

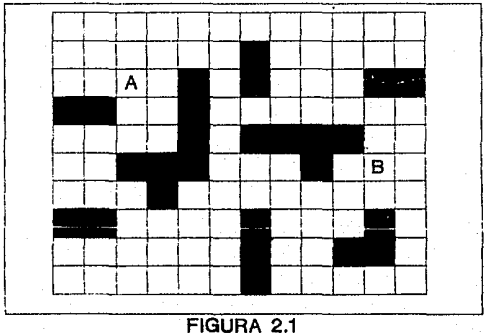

FIGURA 2.1

A partir del origen se asigna a las celdillas contiguas

un número 1 si están libres, pués esto significa que pueden ser ocupadas por una pista (ver figura 2.2.a); una vez numeradas las celdas adyacentes al origen con 1's, se procede a numerar con 2's las celdas contiguas a éstas y no ocupadas, como se puede observar en la figura 2.2.b. Después se asignan 3's a las celdas adyacentes a los 2's y se continúa con el proceso de numeración incrementando el número asignado a la celda en uno si es factible de usar y así hasta encontrar un bloqueo en la ruta o hasta alcanzar el nodo destino, como en el caso mostrado en la figura 2.2.c donde esto ocurre en el paso 13, esto indica la longitud de la ruta más corta.

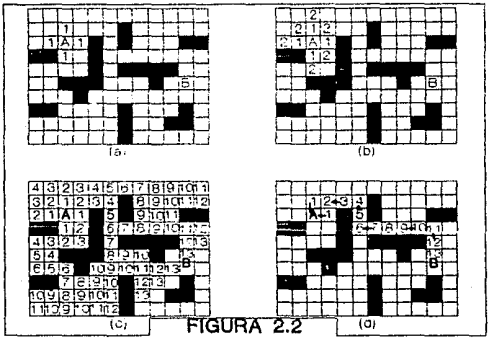

FIGURA 2.2

Una vez alcanzado el destino se procede a trazar la pista siguiendo la numeración asignada a las celdas contiguas en forma descendente, como se observa en la figura 2.2.d.

Con ésta técnica pueden surgir varias rutas, por lo que requiere de una evaluación para decidir cual pista se trazará, de lo contrario resultan trazos muy irregulares. Se sugieren prioridades de dirección por ejemplo Norte-Este-Oeste-Sur, intentando continuar en la misma dirección de movimiento de la celda anterior mientras no encuentre un

bloqueo en la ruta (Ref 4).

El SMART-WORK utiliza este el algoritmo en dos fases, las cuales se detallan a continuación (Ref.5) :

2.5.1 PRIMERA FASE (Fase-I).

1) Guarda la posición y el estado del punto origen.

2) Se verifica en el arreglo de banderas el estado de punto destino. Si es igual al del nodo origen la búsqueda se suspende; sino almacena el valor de la posición actual.

3) Coloca las coordenadas de la celda origen en una fila.

4) Mientras la estructura no esté vacía se repiten los pasos del 5 al 9. Si la fila está vacía, se procede con el paso 10.

5) Toma las coordenadas que están al tope de la fila. Estas coordenadas corresponden al "origen" para los pasos 6 al 12.

6) Verifica en el arreglo de banderas de los elementos la

celda al Norte. Si se trata del destino se completa la fase I y se ejecuta el paso 11. Si el valor de la celda es diferente a cualquier otro valor que "no verificada", entonces la celda ya ha sido revisada y se procede con el paso 7.

7) Si el valor es "no verificada", se revisa la celda para detectar la posible presencia de un conductor. Si un conductor ocupa esta celda, se almacena en el arreglo de banderas un estado de "bloqueo" y el algoritmo procede con el siguiente paso. Si la celda está vacía, el valor Sur es almacenado en el arreglo de banderas de elementos y las coordenadas de esta celda son agregadas al final de la fila. El valor de la bandera Sur será usado como guía para el regreso a la celda origen, si la celda destino es encontrada eventualmente. Esta celda se agrega a la fila de manera que la celda contigua será checada en la búsqueda de la celda destino.
Los pasos 7, 8 y 9 se repiten para cada celda en el sentido Sur, Este y Oeste con respecto al origen.

8) Revisa el arreglo de banderas de elementos de la celda al Este del origen. Si contiene el valor del destino, la fase I

esta completa y el algoritmo procede con paso 11. Si el valor en el arreglo de banderas de elementos al Este del origen contiene un valor diferente de "no verificado", el algoritmo procede con el siguiente paso. Si el valor es "no verificado", se revisa para un conductor. Si un conductor ocupa la celda, el valor "bloqueado" se almacena en el arreglo de banderas de elementos y se continua con el siguiente paso. Si la celda esta vacía, el valor Oeste es almacenado en el arreglo de banderas de elementos y las coordenadas de la celda son agregadas al final de la fila.

9) Verifica el arreglo de banderas de elementos de la celda para el Sur de la celda origen. Si contiene el valor del destino, se completa la fase I y se procede con el paso 11. Si el valor en el arreglo de banderas de elementos al Sur de la celda origen contiene un valor diferente de "no verificado", se contínua con el siguiente paso. Si el valor es "no verificado", la celda se revisa para un conductor. Si un conductor la ocupa, se almacena el valor "bloqueado" en el arreglo de banderas de elementos, y se continúa con el siguiente paso. Si la celda esta vacía, el valor Norte es almacenado en el arreglo de banderas de elementos y las

coordenadas son almacenadas al final de la fila.

10) Revisa el arreglo de banderas de elementos de la celda al Oeste de la celda origen. Si contiene el valor del destino, se termina la fase I, se procede con el paso 11. Si contiene un valor diferente de "no verificado", se procede con la siguiente celda en la fila (paso 4). Si el valor es "no verificado", la celda se verifica para un conductor Si la ocupa un conductor, el valor bloqueado es almacenado en el arreglo de banderas de elementos para la celda y se procede con el paso 4. Si la celda esta vacía, el valor Este se almacena en el arreglo de banderas de elementos, y las coordenadas son anexadas al final de la fila. Se regresa al paso 4.

Si el último paso de la primera fase encuentra que la fila de las celdas está vacía, no existe ruta que conecte el origen con el destino.

2.5.2 SEGUNDA FASE (fase-II).

En esta fase se traza la ruta encontrada, que es la ruta

más corta.

11) Coloca el estado de conductores a las celdas que lo requieran para establecer el trazo, esto se hace regresando del destino al origen.

Cuando la celda destino es alcanzada, se modifica para agregar un conductor del centro de la celda a la celda adyacente a la celda origen y lo mismo se realiza con la celda origen.

12) El contenido del arreglo de banderas de elementos de la celda origen se examina. Si contiene el valor de la celda origen, el algoritmo terminó; sino, contendrá alguno de los valores: Norte, Este, Oeste, Sur. La celda origen y la celda tomada del arreglo de banderas en la dirección indicada son modificadas para agregar un conductor entre los centros de las celdas.

13) Se mueve a la celda adyacente el valor de celda "origen" en la dirección indicada en el arreglo de banderas de elementos. Así la celda alcanzada viene a ser la nueva celda

"origen" y el paso 12 se repite.

2.5.3 Conclusiones.

La principal desventaja consiste en que el número de celdillas que se debe emplear es muy grande; estas son pequeñas para cumplir con los estándares de separación de conductores y elementos.

Se requiere de mucha memoria y el tiempo invertido en la búsqueda de las pistas es muy grande también. Las operaciones necesarias son proporcionales al cuadrado de la longitud de la pista (si es de longitud "n" es proporcional a n cuadrada). Por la cantidad de tiempo de cómputo resulta poco práctico para problemas en los cuales el número de rutas es extenso; sólo resuelve circuitos en un plano, además a pesar de encontrar la ruta más corta no considera el número de quiebres en la pista. Las grandes ventajas son que encuentra la ruta si es que ésta existe y la ruta encontrada tiene la mínima distancia.

CAPITULO 3

ALTERNATIVAS DE SOLUCION

3.1 CARACTERISTICAS DE DISEÑO.

Con base en lo expuesto en el capítulo anterior, se concluye que la parte que más tiempo consume en la elaboración de circuitos impresos es el diseño y dibujo de los mismos.

Se identifican tres etapas principales que influyen en el diseño: la distribución de los elementos, el orden de conexión de los mismos y una vez realizadas las dos etapas anteriores, el trazado de las pistas.

En este trabajo se atacará lo referente al trazado de pistas entre los elementos, y se supondrán dadas las dos primeras etapas.

Así pues, para la tercera etapa no se hablará de una

solución óptima, pues antes se debe definir el sentido de la optimalidad. En este caso se podría considerar como óptima a la solución que proporcione el menor número de trazos posible.

Se propone además dar una solución que ahorre tiempo de desarrollo, por lo que se busca el diseño de los circuitos impresos sin requerir la intervención de una persona especializada. Al respecto, se intenta simplificar el diseño, mediante el uso de una computadora personal.

El tiempo invertido para obtener la solución se considera importante, así como el "buen" diseño de las pistas. El problema se resolverá en partes dada su complejidad. Como estrategia general se intenta resolver el trazado de las pistas por una cara, pero el sistema será capaz de cambiar de cara cuando el trazo de la pista así lo requiera. Se busca también que los trazos tengan la menor distancia y el menor número de cambios de sentido y de cara.

Este sistema no se debe limitar a generar una solución y proporcionarla, sino que generará varias alternativas, las

cuales son evaluadas por una función de costo que selecciona un trazo adecuado. Con esto no se trata de llegar a una solución óptima pero si la mejor solución dentro de las posibilidades, ya que, el trazado depende en gran parte de la distribución de los elementos a interconectar.

3.2 ALTERNATIVAS DE SOLUCION.

En esta sección se describen las ideas generales para el trazo de las pistas en forma automática, muestra las ventajas que ofrecen y se analizan también sus inconvenientes.

Si se busca simplificar el trazo de un circuito en forma, para unir dos puntos una posibilidad consiste en trazar segmentos de recta entre ellos. Para esto resulta adecuado considerar 8 direcciones posibles de movimiento; con respecto a los puntos cardinales y a 45 grados de estos. Así se podrían realizar trazos perpendiculares y diagonales en una malla cuadriculada y analizar para cada celda a la que se llegue, la dirección en que debe efectuarse el movimiento que acerque el trazo al destino. En caso de encontrar un obstáculo, se cambia la dirección para rodearlo y se retoma

después la dirección que lleve hacia el destino. El método descrito no asegura que se llegue al destino y sólo serviría para unir dos puntos. El tiempo que requiere es igual o mayor que el requerido por el algoritmo de Lee, y también la capacidad de almacenamiento necesaria es muy grande.

Otra posibilidad para el trazo consiste en crear unos ejes cartesianos imaginarios, colocar el origen en el centro de dichos ejes y buscar el cuadrante donde se localiza el destino. Se traza una pista sobre los semi-ejes imaginarios cuyo cuadrante encierra al punto destino, y se trata de alcanzar cualquiera de las coordenadas de este. Después se cambia a una dirección perpendicular. Sin embargo surgen problemas con esta forma de trazo cuando se encuentran obstáculos, pues ello obliga a realizar trazos muy irregulares y complica la unión del resto de los puntos. Para evitar este inconveniente, a diferencia del método anterior se reduce a cuatro el número de direcciones de movimiento. Con este método no genera varias alternativas de trazo.

El problema principal persiste y se refiere a que el

número de celdillas es grande y a la falta de "inteligencia" de la búsqueda, se debe analizar celda por celda hasta alcanzar el destino.

Por lo anterior la idea fué crear un método semi-inteligente, que pueda tomar en cuenta su entorno y generar varias opciones de solución. Estas serían evaluadas una a una por una función de costo que relacione características como la distancia, entropía, número de caras en las que se proporcione la solución y número de cambios de cara para realizar la unión del origen y destino.

Para ello se tomó en cuenta el razonamiento intuitivo de una persona para trazar las pistas, se encuentra que este es analizar que puntos quiere unir; si no hay obstáculo emplea una línea recta, pero si hay alguna otra pista que estorbe se buscan las posibles salidas o áreas de paso (Areas Libres) que se tienen en la tarjeta y se pasan por ahí los trazos necesarios.

Para que la computadora pueda emular esto,se pensó en generar rectángulos "imaginarios" cuyas aristas se

construirían tomando como base las pistas, zonas prohibidas, islas y módulos ya localizadas en la tarjeta e identificando las áreas libres más cercanas.

Los rectángulos indican las áreas libres de pistas y se pueden generar de la siguiente manera:

- Si se trata de una ISLA o MODULO, las aristas del rectángulo se forman con base en lo siguiente. Si la isla coincide con alguna de las coordenadas (X o Y) del punto que se desea unir, se genera una recta perpendicular en la coordenada contraria; de otro modo, se genera una recta en cada una de las coordenadas de dicha isla o módulo. Como se ilustra en la figura 3.1.

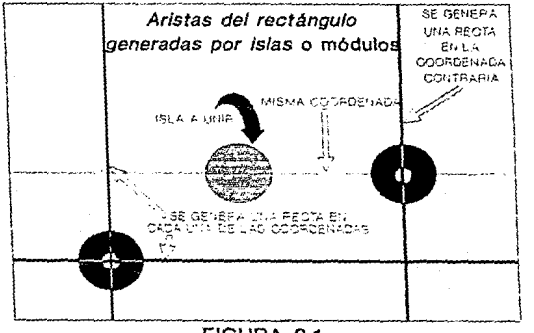

FIGURA 3.1

- Si se trata de una PISTA o ZONA PROHIBIDA, las aristas se forman con base en lo siguiente. Si la pista coincide con una de las coordenadas del punto a unir, se define una línea perpendicular a la pista. En caso contrario, se define una línea que coincide con el sentido de la pista (fig. 3.2).

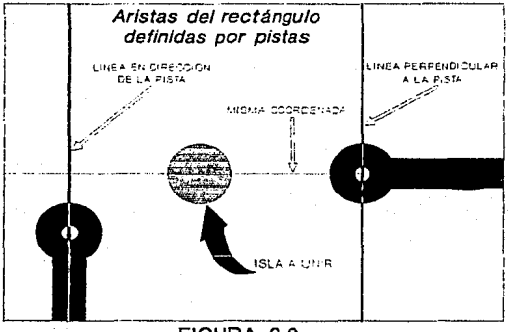

FIGURA 3.2

La información para determinar las aristas, y de ahí los rectángulos se proporciona desde el principio (el conjunto de nodos origen y el conjunto de nodos destino), se generan los rectángulos, y se generan tantos nodos como áreas libres existan. Se analizan los nodos uno a uno; si no hay paso de un a otro regresa al nodo inmediato donde encuentre nodos sin analizar y descoha la ruta por la que no llegó. Si alcanza el destino, marca la ruta como factible y regresa a la última bifurcación que contenga nodos sin analizar, se termina cuando ya no existan nodos por analizar.

Esta es una buena manera buscar las pistas, pero le falta inteligencia para determinar con anterioridad si debe realizar el proceso de búsqueda de la solución. Para ello se debe analizar antes que sea factible alcanzar el destino. Se requiere además mucho espacio de memoria y el método es lento por la generación de información que necesita para encontrar las soluciones.

3.3 CONCLUSIONES.

Si se combinan algunas de las características anteriores se pueden crear una serie de rectángulos en la tarjeta por ambos lados antes de buscar la ruta y analizar si se pueden unir el origen y el destino. Se construiría una malla con los rectángulos, que serán generados no por todos los elementos sobre la tarjeta, sino sólo con las pistas o zonas prohibidas para evitar una generación innecesaria de rectángulos que aumentaría el tiempo de proceso y el espacio para almacenamiento. El número de direcciones para realizar los trazos se restringe a dos (horizontal y vertical).

CAPITULO 4

METODO DE SOLUCION DEL PROBLEMA

Con base en los análisis cualitativos de solución expuestos en el capítulo anterior se ha definido un método que satisface los requerimientos del problema.

A éste método se le ha nombrado Método de los Rectángulos y consiste en dividir la tarjeta en una serie de rectángulos, que son generados con base en las pistas que se definen en la tarjeta, después de generar la primera conexión, y con estos determinar si es factible de unir los puntos deseados, a partir de una matriz cuyos renglones y columnas representan los rectángulos generados, y cuyos elementos indican si existe paso de un rectángulo a otro (Matriz de alcance); se procede a calificar los trazos asignando costos y aplicando la función de evaluación para elegir la mejor de las soluciones generadas y proceder a definirla sobre la tarjeta.

Este proceso continúa hasta tener unidos todos los puntos, ya sea en una sola cara o en dos caras en caso de requerirlo la configuración propuesta, ya que esta depende de la distribución que el usuario asigne a los componentes en la tarjeta.

En la figura 4.1 se ilustra el resultado de generar los rectángulos, se muestran además las áreas de paso para unir las celdas origen ("O") con la celda destino ("D") y los trazos posibles para realizar esto.

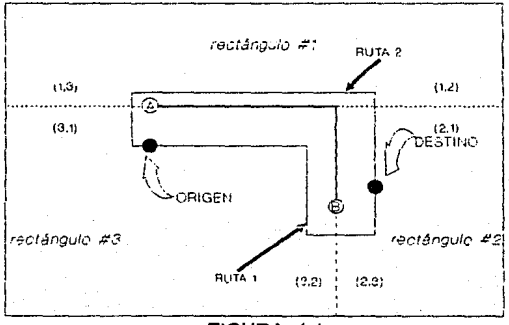

FIGURA 4.1

Los pasos para el desarrollo del algoritmo son los siguientes :

1.- GENERA RECTANGULOS
2.- GENERA MATRIZ DE ALCANCE
3.- OBTENER MATRIZ N-ESIMA
4.- ¿ES ALCANZABLE?
    a) SI : ASIGNAR COSTOS
            EVALUAR RUTAS
            DIBUJAR PISTA
    b) NO : INDICAR QUE NO ES ALCANZABLE

4.1 GENERA RECTANGULOS.

Para generar los rectángulos, se parte de las pistas que ya existen, y se generan líneas que producirán los rectángulos al intersectarse. Los límites de la tarjeta son considerados como pistas.

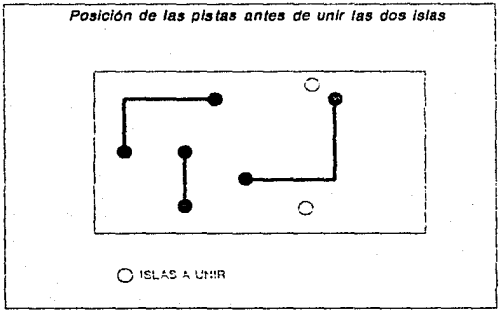

FIGURA 4.2

Si se tiene en cuenta que las pistas están definidas, como se muestra en el ejemplo de la figura 4.2, se pueden generar las líneas y rectángulos con los siguientes algoritmos:

GENERACION DE LINEAS HORIZONTALES

1) Asigna como primera línea la orilla superior de la tarjeta.

2) Selecciona la siguiente pista horizontal.

3) Localiza que pistas verticales cruzan la coordenada en "Y" de la pista horizontal y toma aquellas que se encuentren más cercanas a las coordenadas de inicio y fin de la pista horizontal.

4) Con éstas pistas se definen las coordenadas de la línea horizontal.

5) Si la línea generada está contenida en otra línea previamente definida, no se incluye en el arreglo de líneas horizontales.

6) Se repite a partir del paso 2 hasta agotar las pistas horizontales.

7) Asigna como última línea horizontal el límite inferior de la tarjeta.

GENERACION DE LINEAS VERTICALES

1) Selecciona pista vertical.

2) Localiza las líneas horizontales que cruzan la coordenada en "X" de la pista vertical y toma aquellas que se encuentren más cercanas a las coordenadas de inicio y fin de la pista.

3) Con éstas se definen las coordenadas de la línea vertical.

4) Si la línea generada está contenida en otra línea previamente definida, no se incluye en el arreglo de líneas verticales.

5) Se repite a partir del paso 1 hasta elegir la penúltima pista vertical.

6) Asigna como última línea vertical el límite derecho de la tarjeta.

Una característica importante de los algoritmos anteriores es que las líneas horizontales quedan ordenadas de arriba hacia abajo y de izquierda a derecha, y las verticales de izquierda a derecha y de arriba hacia abajo.

GENERACION DE RECTANGULOS

Es un proceso de tres fases, que genera dos puntos con base en las líneas horizontales y verticales. En la primera fase se genera un vector que contiene el primer par de coordenadas (coordenadas de inicio de la línea) y la coordenada en "X" del segundo punto. En la segunda fase genera un vector auxiliar de coordenadas que sirve para identificar la coordenada en "Y" del segundo punto. En la última se relaciona el segundo vector con el primero de tal forma que definen los rectángulos con base en sus dos puntos diagonales.

PRIMERA FASE :

1) Se toma una línea horizontal no analizada.

2) El primer punto queda definido por el inicio de la línea horizontal.

3) La coordenada en "X" del segundo punto queda definida por la primera línea vertical que intersecta a la derecha del primer punto.

4) Se cambia como inicio de la línea horizontal la coordenada "X" obtenida en el paso anterior.

5) Se repite desde el paso 2 hasta que no haya líneas verticales que la intersecten.

6) Se repite desde el paso 1 hasta la penúltima línea horizontal.

SEGUNDA FASE :

1) Se toma una línea vertical (empezando con la última).

2) Se busca la primera línea horizontal que intersecte

(empezando desde la segunda horizontal).

3) Se almacenan las coordenadas de la intersección en el vector auxiliar.

4) Se repiten los pasos 2 y 3 hasta agotar las líneas horizontales.

5) Se repiten los pasos 1 al 4 hasta analizar la penúltima línea vertical.

TERCERA FASE:

1) Se toma la coordenada en "X" del segundo punto generado en la primera fase, y se busca aquella "X" generada en la segunda fase que sea igual a ésta y entonces se tendrá la coordenada en "Y" del segundo punto del rectángulo. Se repite esto para todos los elementos del vector generado en la primera fase y se eliminan aquellos del vector de la segunda fase que ya han sido analizados. Al terminar este ciclo, se tienen los puntos que conforman los rectángulos.

Con estas estructuras se logra definir los rectángulos por medio de dos puntos en diagonal, posteriormente se desechan todos aquellos cuya distancia sea unitaria y se consideran únicamente los que permitan movimiento en su interior. La figura 4.3 muestra la placa con las pistas y los puntos a unir señalando los rectángulos generados.

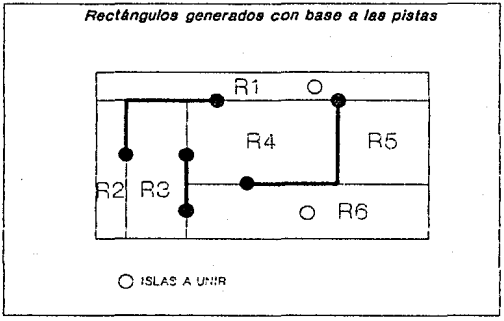

FIGURA 4.3

Se define después la matriz de alcance al analizar los lados de cada rectángulo para identificar aquellos rectángulos, los cuales se pueden interconectar en la misma

cara y definir aquellos cuya área se traslape con otros rectángulos en la cara opuesta, de ésta manera se tienen todas las alternativas de salida de cada uno de los rectángulos.

Al estar definidas todas las interconexiones, se genera la que se llama matríz de adyacencia, que es muy utilizada en programación lineal.

En nuestro caso se denomina matríz de alcance y se define como se explica a continuación:

4.2 TEORIA DE GRAFICAS (Definiciones).

GRAFICA : Una gráfica G = <V,E,O> consiste de un conjunto no vacío V, denominado conjunto de nodos de la gráfica, E es el conjunto de bordes de la gráfica, y O es un mapeo del conjunto de bordes E al conjunto de pares de elementos de V.

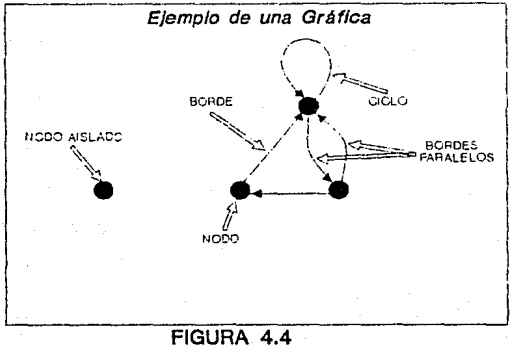

FIGURA 4.4

BORDE : Elemento de una gráfica que enlaza un par de nodos.

BORDES PARALELOS : En algunas gráficas se tendrán nodos unidos por más de un borde, éstos, se les llama bordes paralelos.

GRAFICA PESADA : A las gráficas cuyos bordes tienen asignado un valor llamado "costo" se conocen como gráficas pesadas.

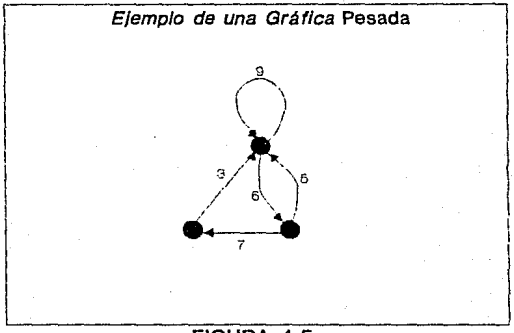

FIGURA 4.5

NODO AISLADO : Si en una gráfica existe algún nodo que no es alcanzable desde cualquier otro nodo de la gráfica, se conoce como nodo aislado.

TRAYECTORIA : Secuencia de bordes de una gráfica, tal que el nodo terminal de un borde en la secuencia es el nodo inicial del siguiente borde; definen una trayectoria.

LONGITUD DE LA TRAYECTORIA : Es el número de bordes que

aparecen en la secuencia de una trayectoria.

CICLO : Trayectoria en la cual el nodo incial es el mismo que el nodo final y puede tener longitud mayor o igual a uno.

Una gráfica puede estar unilateralmente conectada o fuertemente conectada, si para cada par de nodos de la gráfica al menos uno de los nodos del par es alcanzable desde el otro nodo se dice que está unilateralmente ligada, si cada par de nodos son alcanzables uno desde el otro, entonces se dice que está fuertemente conectada (Ref. 6).

4.3 INTERPRETACION DE LA MATRIZ DE ADYACENCIA.

Una forma de representar una gráfica son las matrices. Si tenemos una gráfica G = (V,E,O) donde V = (v_1,v_2,...,v_n) y se asume que los nodos estan conectados de v_1 a v_n. Se puede definir una matriz de n x n cuyos elementos (a_{ij}) cumplen lo siguiente :

$$a_{ij} = \begin{cases} 1 & , \text{ si } (v_i, v_j) \in E \\ 0 & , \text{ en caso contrario} \end{cases}$$

A esta matriz se le conoce como matriz de adyacencia de la gráfica G y una característica importante a simple vista es que ésta resulta ser una matriz simétrica.

En la figura 4.6 se muestra la gráfica que representa la relación entre los rectángulos de la tarjeta.

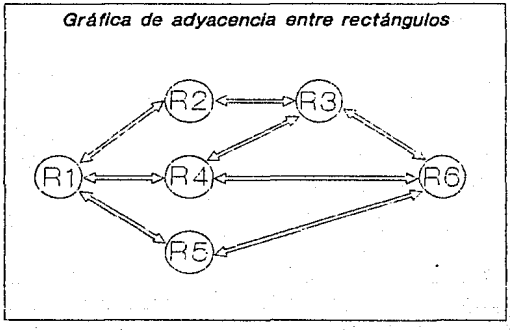

FIGURA 4.6

Si se consideran las potencias de la matriz de

adyacencia; y se toma en cuenta que un valor de 1 en el i-ésimo renglón y la j-ésima columna indican la existencia de una trayectoria del nodo v_i a v_j, entonces:

$$a_{ij}^2 = \sum_{k=1}^{n} a_{ik} * a_{kj}$$

Para cualquier k, $a_{ik}*a_{kj} = 1$ sí y sólo si $a_{ik}=a_{jk}=1$, o sea (v_i,v_k) y (v_k,v_j) son bordes de la gráfica, esto implica que la trayectoria de v_i a v_j es de longitud 2. Al calcular la potencia 2, se determina el número de trayectorias de longitud 2 que unen los nodos (v_i,v_j) y los elementos de la diagonal principal dan el número de ciclos de longitud 2.

Si A es la matriz de adyacencia de la gráfica G. El elemento del i-ésimo renglón y j-ésima columna de A^n $(n > 1)$ es igual al número de trayectorias de longitud n del i-ésimo nodo al j-ésimo nodo.

Con esto se puede determinar el número de rutas de longitud menor o igual a r, si se define la matriz B_r, como:

$$B_r = A + A^2 + ... + A^r$$

Esta matriz B_r que llamaremos "Matriz de Alcance" indica la existencia o ausencia de rutas entre un par de nodos, pero no indica cual es la trayectoria a seguir para alcanzar un nodo (Ref. 6).

4.3.1 CONCLUSIONES.

Se introdujo la matriz de alcance para determinar la existencia de al menos una ruta que una dos nodos al tomar como base que los nodos son rectángulos en los que se localizan los elementos que se desean interconectar. La matriz de alcance determina si es o no posible unir los elementos y así realizar la búsqueda de trayectorias teniendo la certeza de que existe la conexión entre dichos elementos.

4.4 FUNCION DE COSTOS.

Una vez que se determina que existe la posibilidad de unir origen y destino, se procede a determinar el costo de cada uno de los bordes de la matriz de adyacencia. Se asignan

costos "infinitos" a aquellos bordes que la ligan a nodos que no son alcanzables. Para los otros casos el costo se determina con el valor que implica desplazarse del centro del rectángulo origen al centro del rectángulo con que se conecta, el número de cambios de dirección que tiene, el punto de unión de los rectángulos, la cara empleada y si hay necesidad de realizar cambios de cara. En la siguiente figura se muestran los costos que implican el trazo de una ruta que va de un rectángulo a otro.

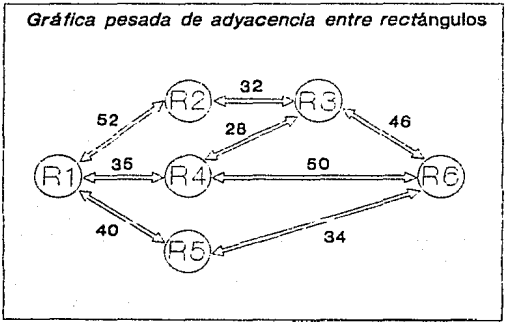

FIGURA 4.7

Con las características mencionadas en el párrafo

anterior se definió la siguiente función para evaluar el costo de las pistas:

$$C = a(D) + b(E) + g(K) + d(N)$$

donde:

C - Costo del borde.
D - Longitud del borde.
E - Entropía del borde.
K - Número de cara empleada.
N - Número de cambios de cara.
a,b,g,d - Factores de peso.

Los factores de peso sirven para lograr una tendencia en alguna de las características que se quiera tengan las pistas resultantes; para evitar alguna de estas, se asigna un valor grande al factor de peso correspondiente a la característica de tal forma que se eliminen las pistas que contengan dicho rasgo o se elija en caso de ser la única posibilidad. Por ejemplo, si se desea obtener trazos por una sola cara se debe asignar al factor d un valor grande para que las pistas que requieran de cambio de cara sean eliminadas. En cuanto a los

valores adecuados para dichos factores de peso requiere de una investigación profunda, no incluida en este trabajo.

Con la función de costo descrita antes se asignan los pesos de los distintos bordes de las trayectorias encontradas en la matriz de alcance, y define una matriz de costos.

Con la matriz de costos y con la seguridad de que se puede de alguna manera llegar al destino, se procede a determinar la ruta tomando como base el problema de redes de la ruta o trayectoria más corta descrito a continuación.

4.5 RUTA MAS CORTA.

Si se tiene una red G con "m" nodos y "n" arcos, y un costo C_{ij} asociado con cada arco (i,j) en G. El problema de la ruta más corta es: encontrar la ruta más corta (de menor costo) del nodo 1 al nodo m en G. El costo resulta de la suma de los costos de los arcos para cubrir la ruta.

Se puede pensar en el problema de la ruta más corta dentro del contexto de flujo en una red, al diseñar una red

en la que se desea enviar una sola unidad de flujo del nodo 1 al nodo m con un costo mínimo. Se define $b_1 = 1$, $b_m = -1$ y $b_i = 0$ para $i \neq 1$ ó m, se tiene que:

$$\text{Minimizar} \sum_{i=1}^{m} \sum_{j=1}^{m} C_{ij} X_{ij}$$

$$\text{sujeta a:} \quad \sum_{j=1}^{m} X_{ij} - \sum_{k=1}^{m} X_{ki} = \begin{cases} 1 \text{ si } i = 1 \\ 0 \text{ si } i \neq 1 \text{ ó } m \\ -1 \text{ si } i = m \end{cases}$$

$$X_{ij},\ X_{ki} = 0 \text{ ó } 1 \qquad i,j,k = 1,\ 2,\ \ldots,\ m$$

en donde las sumas y los requerimientos 0-1 se toman sobre los arcos existentes en G. Las restricciones $X_{ij} = 0$ ó 1 indican que cada arco está o no en la ruta.

Al considerar el dual del problema de la ruta más corta:

$$\text{Maximizar} \quad w_1 - w_m$$

$$\text{sujeta a:} \quad w_i - w_j \leq C_{ij} \qquad i,j = 1,2,\ldots,m$$

$$w_i, w_j \text{ sin restricciones} \quad i = 1,2,\ldots,m$$

Es más conveniente realizar la sustitución de $w'_i = -w_i$.

Como se verá dentro de poco, en la solución óptima $w'_1 - w'_1$ es la distancia más corta del nodo 1 al nodo i, por lo tanto, se puede obtener la distancia más corta del nodo 1 a todos los nodos de la red.

Si se toma en cuenta que no existen costos negativos o sea $C_{ij} \geq 0$.

PASO INICIAL $\quad w'_1 = 0$ y $X = \{ 1 \}$

PASO PRINCIPAL $\quad \overline{X} = N - X$ considérense los arcos en el conjunto $(X,\overline{X}) = \{ (i,j): i \in X, j \in \overline{X} \}$

Sea $w'_p + C_{pq} =$ Mínimo $\{ w'_i + C_{ij} \}$

$(i,j) \in (X,\overline{X})$

Se toma $w'_q = w'_p + C_{pq}$ y se coloca el nodo q en X. Se repite el paso principal exactamente m-1 veces (incluyendo la primera vez) y después se termina; se tiene a la mano la solución óptima (Ref. 7).

La figura 4.8 muestra la ruta más corta que resulta de aplicar el algoritmo.

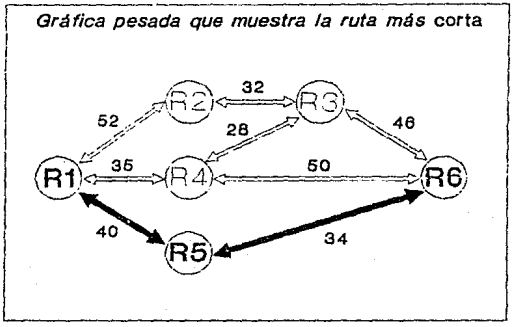

FIGURA 4.8

4.5.1 VALIDEZ DEL ALGORITMO.

Supóngase inductivamente que w'_i para $i \in X$ representa el costo de la ruta más corta del nodo 1 al nodo i. Considérese al algoritmo en algún punto en el que un nuevo nodo q se va añadir a X.

Supóngase que $w'_p + C_{pq}$ = Mínimo { $w'_i + C_{ij}$ } -(ec. 4.1)

$$(i,j) \in (X,\overline{X})$$

se mostrará que la ruta más corta del nodo 1 al nodo q tiene longitud $w'_q = w'_p + C_{pq}$ y que se puede construir iterativamente como la ruta más corta del nodo 1 al nodo p mas el arco (p,q).

Sea p cualquier ruta del nodo 1 al nodo q. Basta mostrar que la longitud de p es mayor o igual a w'_q. Puesto que el nodo 1 está en X y el nodo q se encuentra actualmente en $\overline{X}$, entonces p debe contener un arco (i,j) en donde $i \in X$ y $j \in \overline{X}$ (i,j podrían ser p y q respectivamente). La longitud de la ruta p es, por lo tanto, igual a la suma de las siguientes longitudes:

1.- La longitud del nodo 1 al nodo i.
2.- La longitud del arco (i,j), es decir, C_{ij}.
3.- La longitud de j a q.

Por la hipótesis de inducción, la longitud del nodo 1 al nodo i es mayor o igual que w'_i. Como se ha supuesto que los

costos de todos los arcos son no negativos, entonces, la longitud de p es mayor o igual que $w'_i + C_{ij}$.

En vista de la ecuación (4.1) y puesto que $w'_q = w'_p + C_{pq}$ es claro que la longitud de p $\geq w'_q$. Esto completa el argumento de inducción y se verifica el algoritmo.

Con este algoritmo se garantiza que la ruta que une el rectángulo que contiene al origen con el que incluye al destino es la que representa un menor costo (Ref. 7).

4.6 CONCLUSIONES

Se realizaron pruebas al algoritmo, se hicieron distintas pruebas colocando al algoritmo en situaciones diferentes para las distintas posibilidades que se podían presentar incluyendo aquellas en las que el problema no tenía solución. Se logró buen éxito en los resultados, pues el algoritmo en todos los casos indica la ruta más corta que une los dos puntos. Los programas que ejecutan este algoritmo están elaborados en lenguaje Turbo Pascal y se anexan al final del trabajo (apéndice 1). En la figura 4.9 se ilustra

el resultado obtenido de aplicar el algoritmo para unir dos islas, se muestran los rectángulos generados y estan sombreados los que sugieren el mejor trazo para la pista.

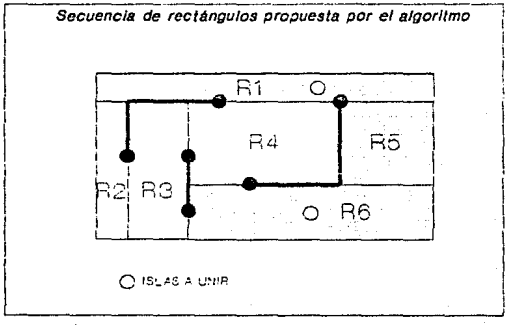

FIGURA 4.9

CAPITULO 5

ALCANCES

5.1 ALCANCES DEL ALGORITMO.

Se pueden realizar mejoras a este algoritmo para reducir el espacio de memoria requerido si se simulan las matrices con vectores e inclusive se efectúe la potencia de las matrices en este tipo de estructura, las cuales se pueden encontrar bien explicadas en un libro de Estructuras de Datos.

Se puede elaborar el algoritmo que determina los traslapes entre los rectángulos de una y otra cara, para incluírlos en la matríz de Adyacencia de tal manera que en esta quedarían representadas las relaciones existentes en una cara, en la otra y entre ambas caras.

Al determinar los costos, lógicamente se dará alto costo a un cambio de cara, por lo que estos en principio se desechan, a menos de que no exista otra posibilidad.

Otro factor que incrementa el costo de una relación entre rectángulos es realizar primero los trazos por la cara superior, por lo que la cara inferior tendrá un factor de peso mayor tal que excluya las rutas de dicha cara a menos de que realmente representen un mejor trazo.

Al determinar la ruta más corta se tomarán en cuenta las relaciones entre los rectángulos y los pesos asignados a éstos serán determinantes para que el algoritmo obtenga como resultado la ruta cuyo costo sea menor al incluir los factores arriba mencionados. Con ello inclusive se consigue realizar todos los cambios de cara necesarios para llegar al destino.

5.2 COMPLEMENTOS AL ALGORITMO.

A partir del algoritmo descrito y de las ideas planteadas, se puede proponer completar un algoritmo que pudiera ahorrar aún más trabajo al diseñador. El algoritmo que se acaba de exponer supone que el diseñador determina la ubicación de los elementos y el orden en que éstos deben

interconectarse, pero pudiera elaborarse un algoritmo al cual se le diera la ubicación de los elementos a interconectar y que decida el órden en el que se deben realizar las interconexiones agotando todas las posibilidades y obtener al final del análisis el arreglo de pistas en el cual se tiene en suma las rutas con menor costo.

Otro algoritmo aún más complejo sería aquel en el que únicamente se alimentara a la computadora con los elementos a interconectar pero no con su ubicación; entonces el programa debe decidir cómo ubicar los elementos en la placa y en que orden se interconectan.

Otra sofisticación sería que se proporcione a la computadora el conjunto de bloques que deseamos en el circuito, por ejemplo: una fuente de poder, un multiplicador, un reloj, un amplificador, un inversor y que el programa elija que elementos electrónicos utilizar, la ubicación sobre la tarjeta y el orden de interconexión de éstos.

Por último, se podría diseñar un algoritmo en el cual se indican las señales de entrada al circuito y el tipo de

señales o la respuesta que se desea que éste entregue. Entonces la computadora tendría que realizar un análisis minucioso de los requerimientos del circuito para determinar el tipo de elementos electrónicos necesarios, su ubicación sobre la placa y el orden de interconexión para lograr la respuesta deseada a las señales de entrada. Evidentemente la tarea tiene un alto grado de complejidad.

Todas estas modificaciones requieren de una cantidad de tiempo considerable para su desarrollo y obtendría como resultado un grado de independencia cada vez mayor del diseñador con la computadora de tal manera que éste puede darle los datos iniciales sin importarle cuanto tiempo tarde ésta en dar la solución; lo importante sería que llegase a ésta de manera segura y que fuese la solución "óptima" buscada.

Estos desarrollos deben ser objeto de futuras investigaciones.

APENDICE 1

LISTADO DEL PROGRAMA

```
{$U+}
PROGRAM PROG(INPUT,OUTPUT);
TYPE

    { TIPO PARA ARREGLOS DE PISTAS O AREAS LIBRES
              X,Y : COORDENADAS
                D : DISTANCIA DE LA PISTA }
    REG=RECORD X,Y,D:INTEGER; END;

    { TIPO PARA ARREGLOS DE RECTANGULOS
              X1, Y1 : COORDENADAS DE INICIO
              X2, Y2 : COORDENADAS DEL PUNTO EN DIAGONAL }
    REC=RECORD X1,Y1,X2,Y2:INTEGER; END;

    { ARREGLO DE NUMERO MAXIMO PARA RECTANGULOS }
    VREC=ARRAY [1..10] OF REC;

    { ARREGLO DE NUMERO MAXIMO PARA PISTAS O AREAS LIBRES }
    VREG=ARRAY [1..10] OF REG;

    { TIPO PARA ARREGLOS DE AREAS LIBRES }
    NL=RECORD X,Y,D,R: INTEGER; END;

    { ARREGLO DE AREAS NO LIBRES }
    NOLIBRE=ARRAY [1..50] OF NL;

    { TIPO PARA AREAS LIBRES }
    L=RECORD X,Y,D: INTEGER; END;

    { ARREGLO DE AREAS LIBRES }
    LIBRE=ARRAY [1..50] OF L;

    { TIPO PARA MATRIZ DE COSTOS, INDAL : INDICE QUE APUNTA A
    EL NO. DE AREA LIBRE }
    MAT= RECORD COSTO,INDAL: INTEGER; END;

    { MATRIZ DE COSTOS }
    MATACESO=ARRAY [1..50,1..50] OF MAT;
```

```
    { TIPO DE ARREGLO PARA ALMACENAR LA RUTA MAS CORTA }
    RT=ARRAY [0..50] OF INTEGER;

VAR

    FILVAR: TEXT;

    { ARREGLOS DE PISTAS Y LINEAS H-HORIZONTALES V-VERTICALES}
    PISTASH,PISTASV,LINEASH,LINEASV:VREG;

    {ARREGLO PARA ALMACENAR LOS RECTANGULOS GENERADOS}
    RECTANG:VREC;

    {NPH, NPV - NUMERO DE PISTAS HORIZONTALES Y VERTICALES
     NLH, NLV - NUMERO DE LINEAS HORIZONTALES Y VERTICALES
           NR - NUMERO DE RECTANGULOS GENERADOS
     ALFA,BETA - PARAMETROS PARA FUNCION EVALUACION DE COSTO.
     LAS DEMAS SON VARIABLES AUXILIARES PARA GENERACION DE LA
     RUTA, POR EJEMPLO AREALRV - AREAS LIBRES VERTICALES
     RESUELTAS. SI TIENE UNA N ANTES SIGNIFICA NO RESUELTA }
    NPH,NPV,NLH,NLV,NR,ANRH,ALFA,BETA,
    ORIGEN,DESTINO,ARH,ANRV,ARV:INTEGER;
    AREALRV,AREALRH: LIBRE;
    AREALNRV,AREALNRH: NOLIBRE;
    MATACCESO:MATACESO;

    {BANDERA PARA INDICAR SI EXISTE O NO ACCESO DE UN
    RECTANGULO A OTRO }
    BAN:BOOLEAN;
    RUTA:RT;
    I,J:INTEGER;
    FILENAME: STRING[14];

{ PROCEDIMIENTO GENLIN }
{ OBJETIVO: GENERAR LINEAS HORIZONTALES Y VERTICALES PARA
  CREAR RECTANGULOS
  PARAMETROS: PISTASH - ARREGLO DE SEGMENTOS DE PISTA HORIZ.
              PISTASV - ARREGLO DE SEGMENTOS DE PISTA VERTIC
              LINEASH - ARREGLO DE LINEAS HORIZONTALES
              LINEASV - ARREGLO DE LINEAS VERTICALES
              NPH,NPV - NUMERO DE PISTAS HORIZONTALES
                        Y VERTICALES RESPECTIVAMENTE
              NLH,NLV - NUMERO DE LINEAS HORIZONTALES
                        Y VERTICALES RESPECTIVAMENTE
```

```
  EXPLICACION: CON LAS PISTAS HORIZONTALES GENERA UNA LINEA
               HORIZONTAL POR CADA PISTA HORIZONTAL Y ELIMINA
               LAS QUE SE DUPLICAN; EN BASE A ESTAS LINEAS
               HORIZONTALES Y LAS PISTAS VERTICALES CREA LAS
               LINEAS VER- TICALES, ELIMINANDO LAS VERTICALES
               DUPLICADAS)

PROCEDURE GENLIN(PISTASH,PISTASV:VREG;NPH,NPV:INTEGER;
                 VAR LINEASH,LINEASV:VREG; VAR NLH,NLV:
                 INTEGER);
VAR
   I,J,K:INTEGER; { INDICES }
BEGIN

{ INICIALIZACION DE ARREGLO DE LINEAS HORIZONTALES Y
VERTICALES }

FOR I:=1 TO 10 DO
    BEGIN
    LINEASH[I].X:=0;
    LINEASH[I].Y:=0;
    LINEASH[I].D:=0;
    LINEASV[I].X:=0;
    LINEASV[I].Y:=0;
    LINEASV[I].D:=0;
    END;

{ ASIGNACION DE PRIMERA LINEA HORIZONTAL }

LINEASH[1].X:=PISTASH[1].X;
LINEASH[1].Y:=PISTASH[1].Y;
LINEASH[1].D:=PISTASH[1].D;
K:=2;

{ DEFINICION DE LAS LINEAS HORIZONTALES }

FOR I:=2 TO NPH DO
    FOR J:=1 TO NPV DO
        BEGIN

        { LOCALIZACION DE LAS COORDENADAS DE INICIO DE LINEA
          HORIZONTAL }

        IF (PISTASV[J].X <= PISTASH[I].X) AND ((PISTASV[J].Y
```

```
            < PISTASH[I].Y)
            AND (PISTASH[I].Y < PISTASV[J].Y+PISTASV[J].D))
            THEN
            BEGIN
               LINEASH[K].X:=PISTASV[J].X;
               LINEASH[K].Y:=PISTASH[I].Y;
            END;

         { LOCALIZACION DE LA LONGITUD DE LA LINEA HORIZONTAL}

         IF (PISTASV[J].X >= PISTASH[I].X+PISTASH[I].D) AND
            ((PISTASV[J].Y < PISTASH[I].Y) AND (PISTASH[I].Y <
            PISTASV[J].Y+PISTASV[J].D)) THEN
               BEGIN
               LINEASH[K].D:=ABS(PISTASV[J].X-LINEASH[K].X);

               { VERIFICACION LINEAS HORIZONTALES DUPLICADAS }

               IF (K <> 1) AND (LINEASH[K].Y = LINEASH[K-1].Y)
                  AND
                  (LINEASH[K].X+LINEASH[K].D = LINEASH[K-
                  1].X+LINEASH[K-1].D) THEN
               ELSE

               { INCREMENTO DEL INDICE DE LINEAS HORIZONTALES}

                  K:=K+1;
               J:=NPV;
               END;
         END;

{ ASIGNACION DE ULTIMA LINEA HORIZONTAL }

LINEASH[K].X:=PISTASH[NPH].X;
LINEASH[K].Y:=PISTASH[NPH].Y;
LINEASH[K].D:=PISTASH[NPH].D;
NLH:=K;

{ DEFINICION DE LAS LINEAS VERTICALES }

K:=1;
FOR I:=1 TO (NPV-1) DO
    FOR J:=1 TO NLH DO
        BEGIN
```

```
        { LOCALIZACION DE LAS COORDENADAS DE INICIO DE LINEA
          VERTICAL }

        IF (LINEASH[J].Y <= PISTASV[I].Y) AND ((LINEASH[J].X
           <= PISTASV[I].X) AND (PISTASV[I].X <=
           LINEASH[J].X+LINEASH[J].D)) THEN
           BEGIN
              LINEASV[K].X:=PISTASV[I].X;
              LINEASV[K].Y:=LINEASH[J].Y;
           END;

        { LOCALIZACION DE LA LONGITUD DE LA LINEA VERTICAL }

        IF (LINEASH[J].Y >= PISTASV[I].Y+PISTASV[I].D) AND
           ((LINEASH[J].X <= PISTASV[I].X) AND (PISTASV[I].X
           <= LINEASH[J].X+LINEASH[J].D)) THEN
           BEGIN
           LINEASV[K].D:=ABS(LINEASV[K].Y-LINEASH[J].Y);

              { VERIFICACION DE LINEAS VERTICALES DUPLICADAS }

              IF (K <> 1) AND (LINEASV[K].X = LINEASV[K-1].X)
                 AND (LINEASV[K].Y+LINEASV[K].D = LINEASV[K-1].Y
                 +LINEASV[K-1].D) THEN
              ELSE

                 { INCREMENTO DE CONTADOR DE LINEAS VERTICALES }

                 K:=K+1;
              J:=NLH;
              END;
           END;

{ ASIGNACION DE LA ULTIMA LINEA VERTICAL }

LINEASV[K].X:=PISTASV[NPV].X;
LINEASV[K].Y:=PISTASV[NPV].Y;
LINEASV[K].D:=PISTASV[NPV].D;
NLV:=K;
WRITELN(FILVAR,'NUM  LIN. H.     LIN. V.');
WRITELN(FILVAR,'LIN  X  Y  D     X  Y  D');
WRITELN(FILVAR,'========================');
IF NLV>NLH THEN
```

```
   J:=NLV
ELSE
   J:=NLH;
FOR I:=1 TO J DO
WRITELN(FILVAR,I:2,LINEASH[I].X:4,LINEASH[I].Y:3,LINEASH[I].D
        :3,' ',LINEASV[I].X:4,LINEASV[I].Y:3,LINEASV[I].D:3);
WRITELN(FILVAR);
END;

{ PROCEDIMIENTO GENREC }
{ OBJETIVO: GENERAR RECTANGULOS
  PARAMETROS:    R - ARREGLO DE COORDENADAS DE LOS RECTANGULOS
                 NR - NUMERO DE RECTANGULOS
                 LH - ARREGLO DE LINEAS HORIZONTALES
                 LV - ARREGLO DE LINEAS VERTICALES
                NLH - NUMERO DE LINEAS HORIZONTALES
                NLV - NUMERO DE LINEAS VERTICALES

  EXPLICACION: ES UN PROCESO DE TRES FASES, QUE GENERA DOS
               PUNTOS EN BASE A LAS LINEAS HORIZONTALES Y
               VERTICALES.EN LA PRIMERA FASE GENERA EL PRIMER
               PAR DE COORDENADAS Y LA COORDENADA EN "X" DEL
               SEGUNDO PAR. EN LA SEGUNDA FASE GENERA UN
               VECTOR AUXILIAR DE COORDENADAS QUE SIRVE PARA
               IDENTIFICAR LA COORDENADA EN "Y" DEL SEGUNDO
               PUNTO. EN LA ULTIMA PARTE SE ASIGNA LA SEGUNDA
               COORDENADA DEL PUNTO AL RECTANGULO
               CORRESPONDIENTE }

PROCEDURE GENREC(LH,LV:VREG; NLH,NLV:INTEGER; VAR R:VREC; VAR
                 NR:INTEGER);
VAR
   I,J,K:INTEGER; { INDICES }
   LOOP:BOOLEAN; { BANDERA PARA ROMPER CICLO DE GENERACION DE
                   RECTANGULO }

   { ARREGLO DE COORDENADAS AUXILIARES EN LA GENERACION DE
     RECTANGULO }

   LAUX:VREG;
BEGIN
```

```
{ INICIALIZACION DE ARREGLO DE COORDENADAS DE RECTANGULOS }

FOR I:=1 TO 10 DO
    BEGIN
    RECTANG[I].X1:=0;
    RECTANG[I].Y1:=0;
    RECTANG[I].X2:=0;
    RECTANG[I].Y2:=0;
    END;
K:=1;

{ CICLO PARA DEFINIR LAS COORDENADAS DEL PRIMER PUNTO Y DE LA
  COORDENADA EN 'X' DEL SEGUNDO BASANDOSE EN LAS LINEAS
  HORIZONTALES Y LAS LINEAS VERTICALES QUE LAS INTERSECTAN }

FOR I:= 1 TO (NLH-1) DO
    BEGIN
    J:=2;

    { ASIGNACION DEL PRIMER PUNTO DEL RECTANGULO }

    R[K].X1:=LH[I].X;
    R[K].Y1:=LH[I].Y;
    LOOP:=TRUE;
    WHILE LOOP { CICLO DE BUSQUEDA CONTRA LAS LINEAS VERT.}
          BEGIN

          { LOCALIZACION PARA LA ASIGNACION DEL PUNTO INCIAL
            DEL RECTANGULO }

            IF (R[K].X1 < LV[J].X) AND (LV[J].X <=
               R[K].X1+LH[I].D) AND (LV[J].Y <= R[K].Y1) AND
               (R[K].Y1 < LV[J].Y+LV[J].D) THEN
               BEGIN

               {ASIGNACION DE LA PRIMERA COORDENADA DEL
                SEGUNDO PUNTO }

                  R[K].X2:=LV[J].X;
                  IF R[K].X2 < LH[I].X+LH[I].D THEN
                     BEGIN
```

```
                    { ASIGNACION DE PRIMER PUNTO DEL
                      SIGUIENTE RECTANGULO }

                       R[K+1].X1:=R[K].X2;
                       R[K+1].Y1:=LH[1].Y;
                    END
                 ELSE
                    LOOP:=FALSE;  { FIN DE BUSQUEDA }
                 K:=K+1;  {INCREMENTA CONTADOR RECTANGULOS}
              END;
           J:=J+1;  { INCREMENTA EL INDICE DE LINEAS VERTI}
        END;
    END;

{ INICIALIZACION DEL VECTOR AUXILIAR }

FOR I:=1 TO 10 DO
    BEGIN
    LAUX[I].X:=0;
    LAUX[I].Y:=0;
    LAUX[I].D:=0;
    END;

{ GENERA EL VECTOR AUXILIAR PARA DETERMINAR LA COORDENADA "Y"
DEL 2o PUNTO }

J:=1;
FOR I:=NLV DOWNTO 2 DO
    BEGIN
       FOR K:= 2 TO NLH DO
           BEGIN

           { BUSQUEDA DEL SEGUNDO PUNTO DEL RECTANGULO }

           IF (LH[K].X < LV[I].X) AND (LV[I].X <=
              LH[K].X+LH[K].D) AND
              (LV[I].Y < LH[K].Y) AND (LH[K].Y <=
              LV[I].Y+LV[I].D) THEN
              BEGIN

                 { ASIGNACION DE COORDENADAS DEL PUNTO }

                 LAUX[J].Y:=LH[K].Y;
                 LAUX[J].X:=LV[I].X;
```

```
                J:=J+1;  { INCREMENTA EL INDICE PUNTOS AUXI }
            END;
        END;
    END;
NR:=J-1;

{ BUSQUEDA DE CORRESPONDENCIA ENTRE LA 1a. COORDENADA DEL 2o.
  PUNTO DEL RECTANGULO Y LAS COORDENADAS QUE CONTIENE EL
  VECTOR DE PUNTOS AUXILIARES }

FOR I:=1 TO NR DO
    BEGIN
       J:=1;

       { CICLO DE BUSQUEDA DE COORDENADA "Y" EN ARREGLO AUX }

       WHILE R[I].X2 <> LAUX[J].X
          J:=J+1;
       R[I].Y2:=LAUX[J].Y; { ASIGNACION DE LA COORDENADA "Y"
                             PARA EL RECTANGULO}
       LAUX[J].X:=0;  { ELIMINA EL PUNTO ANALIZADO }
    END;

FOR I:=1 TO NR DO
    WRITELN(FILVAR,'RECTANGULO No. ',I:2,' X1: ',R[I].X1:2,
    'Y1: ',R[I].Y1:2,' X2: ',R[I].X2:2,' Y2: ',R[I].Y2:2);
END;


{ PROCEDIMIENTO ALCANCE }
{ OBJETIVO: DETERMINAR EL ACCESO A LO PORCION DE LA TARJETA
            QUE CONTIENE EL NODO DESTINO.
  PARAMETROS : PH,PV - ARREGLOS DE COORDENADAS DE PISTAS
                       HORIZONTALES Y VERTICALES
                       RESPECTIVAMENTE.
               NPH,NPV - NUMERO DE PISTAS HORIZONTALES Y
                       VERTICALES RESPECTIVAMENTE.
                     R - ARREGLO DE COORDENADAS DE RECTANGULOS
                       GENERADOS.
                    NR - NUMERO DE RECTANGULOS GENERADOS.

  EXPLICACION: ANALIZA CADA UNO DE LOS LADOS DE LOS
               RECTANGULOS GENERADOS HASTA EL MOMENTO Y
               VERIFICA QUE EXISTA ENTRADA Y/O SALIDA DEL
```

```
                MISMO. SE AUXILIA CON PROCEDIMIENTO LIBRES.
                ADEMAS MANDA IMPRIMIR LA MATRIZ DE ACCESO Y
                LOS ARREGLOS DE AREAS LIBRES RESUELTAS
                HORIZONTALES Y VERTICALES. }

PROCEDURE ALCANCE(PH,PV:VREG; NPH,NPV:INTEGER; R:VREC; NR:
                  INTEGER);
VAR
   IR:INTEGER; { INDICE PARA CICLO DE BUSQUEDA EN ARREGLO DE
                 RECTANGULOS }


{ PROCEDIMIENTO LIBRES }
{ OBJETIVO: DETERMINAR LOS SEGMENTOS LIBRES DE EL LADO DE UN
            RECTANGULO.
  PARAMETROS: VEC - ARREGLO DE PISTAS HORIZONTALES O VERTIC.
              BAN - INDICA SI EL ARREGLO "VEC" CONTIENE
                    PISTAS HORIZONTALES O VERTICALES.
               NP - INDICA EL NUMERO DE PISTAS QUE SE DEBEN
                    ANALIZAR.
            X,Y,D - COORDENADAS (X,Y) Y DISTANCIA (D) QUE
                    DEFINEN EL RECTANGULO A ANALIZAR.

  EXPLICACION: PARA CADA UNA DE LAS PISTAS DEFINIDAS LOCALIZA
               CUALES SON LAS AREAS DE PASO QUE DEFINEN CON
               LOS LADOS DE LOS DE LOS RECTANGULOS, Y LAS
               ALMACENA EN UN ARREGLO DE AREAS LIBRES NO
               RESUELTAS. ESTAS QUEDAN RESUELTAS AL
               ANALIZARLAS CON EL PROCEDIMIENTO BUSCA. PARA
               SABER SI YA ESTA DEFINIDA O ESTA CONTENIDA EN
               OTRA AREA LIBRE PREVIAMENTE DEFINIDA }

PROCEDURE LIBRES(VEC:VREG; BAN:CHAR; NP,X,Y,D:INTEGER);
VAR
    J,K:INTEGER; {INDICES ARREGLO Y NUMERO DE PISTAS RESPEC}
    ALIBRE:NL;   {ARREGLO DE AREAS LIBRES O DE PASO}
    AUXLIB:REG;  {VARIABLE AUXIL. PARA INTERCAMBIAR LAS
                  COORDENADAS (X,Y) PARA HACER EL ANALISIS EN
                  FORMA HORIZONTAL O VERTICAL}

{ PROCEDIMIENTO BUSCA }
{ OBJETIVO: IDENTIFICAR QUE SEGMENTOS DE AREAS DE PASO DEBEN
            SER INCLUIDAS EN EL ARREGLO DE AREAS LIBRES
            RESUELTAS.
```

```
  PARAMETROS: ALIBNR - CONTIENE LA INFORMACION (X,Y,D) QUE
                       DEFINE UN AREA LIBRE NO RESUELTA A
                       ANALIZAR.
              ALNR - CONTIENE LA INFORMACION (X,Y,D) QUE
                     DEFINA UN AREA LIBRE RESUELTA.
              AREA - ARREGLO DE AREAS LIBRES NO RESUELTAS.
             AREAL - ARREGLO DE AREAS LIBRES RESUELTAS.

  EXPLICACION: IDENTIFICA SI EL AREA LIBRE ES HORIZONTAL O
               VERTICAL PARA SELECCIONAR EL ARREGLO
               CORRESPONDIENTE Y VERIFICA QUE NO ESTE
               CONTENIDA O PREVIAMENTE DEFINIDA. SI ES NUEVA
               SE INCLUYE EN EL ARREGLO CORRESPONDIENTE
               (HORIZONTAL O VERTICAL). }

PROCEDURE BUSCA(ALIBNR:NL; VAR ALNR,ALR:INTEGER; VAR
                AREA:NOLIBRE; VAR AREAL:LIBRE);
VAR
   {INDICES Y VARIABLES AUXILIARES PARA EL ANALISIS DE
    COORDENADAS Y DEFINICION DE AREAS LIBRES}
   I,J,X1,X2,X3,Y1,Y2,Y3,D1,D2,D3,R1,R2,R3:INTEGER;

   AUX1,AUX2:NL;{INTERCAMBIO DE COORDENADAS SEGUN EL SENTIDO
                 HORIZONTAL O VERTICAL.}
BEGIN
   J:=ALNR+1; {INICIALIZA INDICE PARA ASIGNAR AREA LIBRE N R}
   I:=1; {APUNTA A LA PRIMERA AREA LIBRE NO RESUELTA (ALNR)}
   WHILE I<=ALNR DO {CICLO PARA ANALISIS DE ALNR}
      BEGIN
      IF BAN='V' THEN {VERIFICA ANALISIS HORIZ O VERTICAL}
         BEGIN

         {ASIGNA COORDENADAS PARA ANALISIS VERTICAL}
         AUX1.X:=AREA[I].Y;
         AUX1.Y:=AREA[I].X;
         AUX2.X:=ALIBNR.Y;
         AUX2.Y:=ALIBNR.X;
         END
      ELSE
         BEGIN

         {ASIGNA COORDENADAS PARA ANALISIS HORIZONTAL}
         AUX1.X:=AREA[I].X;
         AUX1.Y:=AREA[I].Y;
```

```
        AUX2.X:=ALIBNR.X;
        AUX2.Y:=ALIBNR.Y;
        END;

    {VERIFICA SI ESTA CONTENIDA EN EL ALNR ANALIZADA}
    IF (AUX1.Y<=AUX2.Y) AND (((AUX1.X<=AUX2.X) AND
       (AUX2.X < AUX1.X+AREA[I].D)) OR ((AUX2.X<=AUX1.X)
       AND (AUX1.X<AUX2.X+ALIBNR.D))) THEN
       BEGIN
          J:=I; {QUEDA APUNTANDO EN EL ARREGLO AL AREA QUE
                 LA CONTIENE}
          I:=ALNR+1; {ROMPE CICLO DE BUSQUEDA}
       END
    ELSE
       I:=I+1; {INCREMENTA INDICE PARA BUSCAR EN ARREGLO
                DE ALNR}
    END;
IF J>ALNR THEN {VERIFICA LA EXISTENCIA DE UNA NUEVA ALNR}

   {ALMACENA LAS CARACTERISTICAS DE LA NUEVA ALNR EN EL
   ARREGLO DE ALNR}
   BEGIN
      AREA[J].X:=ALIBNR.X;
      AREA[J].Y:=ALIBNR.Y;
      AREA[J].D:=ALIBNR.D;
      AREA[J].R:=ALIBNR.R;
      ALNR:=ALNR+1; {INCREMENTA TOPE DE ARREGLO ALNR}
   END
ELSE
   BEGIN
   IF BAN='V' THEN {VERIFICA SI SE REALIZA ANALISIS
                    HORIZONTAL O VERTICAL}

      {ASIGNA COORDENADAS PARA ANALISIS VERTICAL}
      BEGIN
         AUX1.X:=AREA[J].Y;
         AUX1.Y:=AREA[J].X;
         AUX2.X:=ALIBNR.Y;
         AUX2.Y:=ALIBNR.X;
      END
   ELSE

      {ASIGNA COORDENADAS PARA ANALISIS HORIZONTAL}
      BEGIN
```

```
    AUX1.X:=AREA[J].X
    AUX1.Y:=AREA[J].Y;
    AUX2.X:=ALIBNR.X;
    AUX2.Y:=ALIBNR.Y;
END;

{INICIALIZA LA COORDENADA EN "Y" DEL ALNR QUE SE
DIVIDE}
Y1:=AUX2.Y;
Y2:=AUX2.Y;
Y3:=AUX2.Y;
IF AUX1.X<=AUX2.X THEN{IDENTIFICA EL INICIO DE ALNR}

    {ASIGNA CARACTERISTICAS DE ALNR VERTICAL}
    BEGIN
       X1:=AUX1.X;
       X2:=AUX2.X;
       R1:=AREA[J].R;
    END
ELSE

    {ASIGNA CARACTERISTICAS DE ALNR HORIZONTAL}
    BEGIN
       X1:=AUX2.X;
       X2:=AUX1.X;
       R1:=ALIBNR.R;
    END;
D1:=X2-X1; {CALCULO DE LA LONGITUD DE LA ALNR RESUL}

{VERIFICA QUE SE TERMINO DE ANALIZAR TODA LA ALNR}
IF AUX1.X+AREA[J].D<=AUX2.X+ALIBNR.D THEN
    BEGIN
       X3:=AUX1.X+AREA[J].D;
       D3:=AUX2.X+ALIBNR.D-X3;
       R3:=ALIBNR.R;
    END
ELSE
    BEGIN
       X3:=AUX2.X+ALIBNR.D;
       D3:=AUX1.X+AREA[J].D-X3;
       R3:=AREA[J].R;
    END;
D2:=X3-X2; {CALCULO LONGITUD DE ALNR RESULTANTE}
```

```
R2:=AREA[J].R; {ASIGNACION DEL RECTANGULO AL QUE
                PERTENECE LA ALNR}
IF D1<>0 THEN {VERIFICA QUE LONGITUD MAYOR A CERO}
   BEGIN

      {IDENTIFICA ANALISIS HORIZONTAL O VERTICAL}
      IF BAN='V' THEN

         {ASIGNA COORDENADAS ANALISIS VERTICAL}
         BEGIN
            AREA[J].X:=Y1;
            AREA[J].Y:=X1;
         END
      ELSE

         {ASIGNA COORDENADAS ANALISIS HORIZONTAL}
         BEGIN
            AREA[J].X:=X1;
            AREA[J].Y:=Y1;
         END;
      AREA[J].D:=D1; {CALCULO LONG ALNR RESULTANTE}
      AREA[J].R:=R1; {ASIGNA RECTANGULO A QUE
                      PERTENECE EL ALNR}
      J:=ALNR+1;
   END;

{VERIFICA QUE LA DISTANCIA DEL OTRO SEGMENTO DE ALNR
RESULTANTE ES DIFERENTE DE CERO}
IF D3<>0 THEN
   BEGIN

      {VERIFICA ANALISIS HORIZONTAL O VERTICAL}
      IF BAN='V' THEN
         BEGIN

            {ASIGNA COORDENADAS ANALISIS VERTICAL}
            AREA[J].X:=Y3;
            AREA[J].Y:=X3;
         END
      ELSE
         BEGIN

            {ASIGNA COORDENADAS ANALISIS HORIZONTAL}
            AREA[J].X:=X3;
```

```
            AREA[J].Y:=Y3;
          END;
       AREA[J].D:=D3; {ASIGNA LONGITUD DE ALNR RESUL}
       AREA[J].R:=R3; {NUMERO DE RECTANG A LA ALNR}

       {VERIFICA EXISTENCIA DE UNA NUEVA ALNR POR
       DIVISION}
       IF J>ALNR THEN
          ALNR:=ALNR+1;{INCREMENTA TOPE ARREGLO ALNR}
    END;

{VERIFICA QUE ALNR RESULTANTE POR DIVISION NO TIENE
LONGITUD CERO}
IF (D1=0) AND (D3=0) THEN

    {CICLO PARA REACOMODO DE ARREGLO DE ALNR}
    BEGIN
       FOR I:=J+1 TO ALNR DO
          BEGIN
             AREA[I-1].X:=AREA[I].X;
             AREA[I-1].Y:=AREA[I].Y;
             AREA[I-1].D:=AREA[I].D;
             AREA[I-1].R:=AREA[I].R;
          END;
    ALNR:=ALNR-1; {DECREMENTA TOPE ARREGLO DE ALNR}
    END;
ALR:=ALR+1; {AJUSTE TOPE DE ARREGLO DE ALNR}

{VERIFICA EN QUE SENTIDO HORIZONTAL O VERTICAL PARA
INCLUIR EN EL ARREGLO DE AREAS LIBRES RESUELTAS}
IF BAN='V' THEN
    BEGIN

       {ASIGNA COORDENADAS PARA SENTIDO VERTICAL}
       AREAL[ALR].X:=Y2;
       AREAL[ALR].Y:=X2;
       ALR:=ALR*(-1);
    END
ELSE
    BEGIN

       {ASIGNA COORDENADAS PARA SENTIDO HORIZONTAL}
       AREAL[ALR].X:=X2;
       AREAL[ALR].Y:=Y2;
```

```
            END;
          AREAL[ABS(ALR)].D:=D2; {LONGITUD AREA LIBRE RESUELTA}

          {ASIGNACION DE COSTOS EN MATRIZ DE ACCESO}
          MATACCESO[R2,ALIBNR.R].COSTO:=1;
          MATACCESO[R2,ALIBNR.R].INDAL:=ALR;
          MATACCESO[ALIBNR.R,R2].COSTO:=1;
          MATACCESO[ALIBNR.R,R2].INDAL:=ALR;
          ALR:=ABS(ALR);
       END;
END;

BEGIN
   K:=NP; { INICIALIZA TOPE PARA ANALIZAR EN PISTAS }
   J:=1;

   {CICLO PARA ANALIZAR EL ARREGLO DE PISTAS}
   WHILE J<=NP DO
       BEGIN

       {VERIFICA SI ES ANALISIS DE PISTAS HOR O VER}
       IF BAN='V' THEN
          BEGIN

             {VERTICAL}
             AUXLIB.X:=VEC[J].Y;
             AUXLIB.Y:=VEC[J].X;
             AUXLIB.D:=VEC[J].D;
          END
       ELSE
          BEGIN

             {HORIZONTAL}
             AUXLIB.X:=VEC[J].X;
             AUXLIB.Y:=VEC[J].Y;
             AUXLIB.D:=VEC[J].D;
          END;

       {IDENTIFICA SI HAY PISTA QUE GENERE AREA LIBRE}
       IF (AUXLIB.Y=Y) AND (AUXLIB.X<X+D) AND
          (X<AUXLIB.X+AUXLIB.D) THEN
          BEGIN
               K:=J; {GUARDA INDICE DE PISTA QUE GENERA AREA
                      LIBRE}
```

```
            J:=NP+1; {ROMPE EL CICLO DE BUSQUEDA}
         END;
      J:=J+1; {SELECCIONA LA SIGUIENTE PISTA}
      END;

{IDENTIFICA SI ES ANALISIS DE PISTAS VERTICAL U HORIZONT}
IF BAN='V' THEN
   BEGIN

      {VERTICAL}
      AUXLIB.X:=VEC[K].Y;
      AUXLIB.Y:=VEC[K].X;
      AUXLIB.D:=VEC[K].D;
   END
ELSE
   BEGIN

      {HORIZONTAL}
      AUXLIB.X:=VEC[K].X;
      AUXLIB.Y:=VEC[K].Y;
      AUXLIB.D:=VEC[K].D;
   END;

{CICLO IDENTIFICACION CARACTERISTICAS DEL AREA LIBRE}
WHILE (AUXLIB.Y=Y) AND (AUXLIB.X<X+D) AND
      (X<AUXLIB.X+AUXLIB.D) AND (K<=NP) DO
      BEGIN

         {IDENTIFICA SI ES ANALISIS HORIZONT O VERTICAL}
         IF BAN='V' THEN
            BEGIN

               {VERTICAL}
               AUXLIB.X:=VEC[K].Y;
               AUXLIB.Y:=VEC[K].X;
               AUXLIB.D:=VEC[K].D;
            END
         ELSE
            BEGIN

               {HORIZONTAL}
               AUXLIB.X:=VEC[K].X;
               AUXLIB.Y:=VEC[K].Y;
               AUXLIB.D:=VEC[K].D;
```

```
    END;
{IDENTIFICA LA COORDENADA EN "X"}
IF AUXLIB.X<=X THEN
    BEGIN
        {INICIO DEL AREA LIBRE}
        D:=X+D-AUXLIB.X-AUXLIB.D;
        X:=AUXLIB.X+AUXLIB.D;
    END
ELSE
    BEGIN
        {CARACTERISTICAS DE LONGITUD Y RECTANGULO
        ASOCIADO AL AREA LIBRE}
        ALIBRE.D:=AUXLIB.X-X;
        ALIBRE.R:=IR;
        {IDENTIFICA ANALISIS HORIZON O VERTICAL}
        IF BAN='V' THEN
            {VERTICAL}
            BEGIN
                ALIBRE.X:=Y;
                ALIBRE.Y:=X;
                {VERIFICA SI SE INCLUYE AREA LIBRE}
                BUSCA(ALIBRE,ANRV,ARV,AREALNRV,
                      AREALRV);
            END
        ELSE
            BEGIN
                {ASIGNACIONES PARA ANALISIS HORIZON}
                ALIBRE.X:=X;
                ALIBRE.Y:=Y;
                {VERIF SI SE INCLUYE EL AREA LIBRE}
                BUSCA(ALIBRE,ANRH,ARH,AREALNRH,
                      AREALRH);
            END;
```

```
                    {IMPRESION AREA LIBRE GENERADA}
                    WRITELN (FILVAR,ALIBRE.X,' - ',ALIBRE.Y,' -
                             ',ALIBRE.D,' - ',IR);
                    D:=X+D-AUXLIB.X-AUXLIB.D;
                    X:=AUXLIB.X+AUXLIB.D;
                 END;
              K:=K+1; {ROMPE CICLO BUSQUEDA EN ARREGLO PISTAS}
           END;

     {VERIFICA LONGITUD ULTIMA AREA LIBRE NO IGUAL A CERO}
     IF D>0 THEN
        BEGIN
           ALIBRE.D:=D;
           ALIBRE.R:=IR;

           {IDENTIFICA ANALISIS HORIZONTAL O VERTICAL}
           IF BAN='V' THEN
              BEGIN

                 {ASIGNACIONES ANALISIS VERTICAL}
                 ALIBRE.X:=Y;
                 ALIBRE.Y:=X;

                 {VERIFICA SI SE INCLUYE EN ARREGLO A LIBRES}
                 BUSCA(ALIBRE,ANRV,ARV,AREALNRV,AREALRV);
              END
           ELSE
              BEGIN

                 {ASIGNACIONES ANALISIS HORIZONTAL}
                 ALIBRE.X:=X;
                 ALIBRE.Y:=Y;

                 {VERIFICA SI SE INCLUYE EN ARR AREAS LIBRES}
                 BUSCA(ALIBRE,ANRH,ARH,AREALNRH,AREALRH);
              END;

           {IMPRESION DEL AREA LIBRE GENERADA}
           WRITELN (FILVAR,ALIBRE.X,' - ',ALIBRE.Y,' - ',
                    ALIBRE.D,' - ',IR);
        END;
END;
```

```
BEGIN
   FOR IR:=1 TO NR DO  { ANALIZA LOS RECTANGULOS }
       BEGIN

          {ANALIZA SI GENERA A LIBRES PARA CADA LADO RECTANG}
          LIBRES(PISTASH,'H',NPH,R[IR].X1,R[IR].Y1,R[IR].X2-
                 R[IR].X1);
          LIBRES(PISTASH,'H',NPH,R[IR].X1,R[IR].Y2,R[IR].X2-
                 R[IR].X1);
          LIBRES(PISTASV,'V',NPV,R[IR].Y1,R[IR].X1,R[IR].Y2-
                 R[IR].Y1);
          LIBRES(PISTASV,'V',NPV,R[IR].Y1,R[IR].X2,R[IR].Y2-
                 R[IR].Y1);
       END;

   {IMPRESION DE MATRIZ DE ADYACENCIA}
   WRITELN(FILVAR,'MATRIZ DE ADYACENCIA');
   FOR I:=1 TO 10 DO
       FOR J:=1 TO 10 DO
           IF MATACCESO[I,J].COSTO<>0 THEN
           WRITELN(FILVAR,'(',I:2,',',J:2,') = ',
           MATACCESO[I,J].COSTO,' : ',MATACCESO[I,J].INDAL);

   {IMPRESION AREAS LIBRES RESUELTAS GENERADAS (HOR Y VERT)}
   WRITELN(FILVAR,'AREAS LIBRES RESUELTAS HORIZONTALES');
   FOR I:=1 TO ARH DO
       WRITELN(FILVAR,I:2,' - ',AREALRH[I].X:2,',',
                AREALRH[I].Y:2,',',AREALRH[I].D);
       WRITELN(FILVAR,'AREAS LIBRES RESUELTAS VERTICALES');
   FOR I:=1 TO ARV DO
       WRITELN(FILVAR,I:2,' - ',AREALRV[I].X:2,',',
                AREALRV[I].Y:2,',',AREALRV[I].D);
END;

{PROCEDURE CALCULA_COSTO}
{OBJETIVO: ASIGNAR LOS COSTOS ASOCIADOS A LOS BORDES DE LA
           GRAFICA RESULTANTE.
 PARAMETROS : NINGUNO
 EXPLICACION: SE CALCULA LA DISTANCIA DEL PUNTO MEDIO ENTRE
              RECTANGULOS PARA ASOCIAR EL COSTO QUE IMPLICA
              MOVERSE DE UN RECTANGULO A OTRO, EN BASE A LOS
              PARAMETROS DE LA FUNCION DE COSTO.

                COSTO = A*D + B*E      D=DISTANCIA E=ENTROPIA
```

```
                    COSTO = A*D + B*E      D=DISTANCIA E=ENTROPIA
                    A Y B FACTORES INDICAN PESO CARACTERISTICA)
PROCEDURE CALCULA_COSTO;
VAR
   XL1,XL2,YL1,YL2 : INTEGER;

{ FUNCION COSTO }
{ OBJETIVO : CALCULA EL COSTO PARA LA MATRIZ DE ACCESO
  PARAMETROS: I,J - INDICES DE LOS RECTANGULOS A ANALIZAR
              XL1,XL2,YL1,YL2 - COORDENADAS DE RECTANGULOS
  EXPLICACION : GENERA LOS COSTOS ASOCIADOS DE PASAR DE LA
                PARTE MEDIA DE UN RECTANGULO A OTRO.}
FUNCTION PESO(I,J,XL1,XL2,YL1,YL2:INTEGER):INTEGER;
VAR
   X13,Y13,XL3,YL3,X23,Y23,DX1,DY1,DX2,DY2,E,D:INTEGER;
BEGIN
      X13:=TRUNC((RECTANG[I].X2-
           RECTANG[I].X1)/2)+RECTANG[I].X1;
      Y13:=TRUNC((RECTANG[I].Y2-
           RECTANG[I].Y1)/2)+RECTANG[I].Y1;
      XL3:=TRUNC((XL2-XL1)/2)+XL1;
      YL3:=TRUNC((YL2-YL1)/2)+YL1;
      X23:=TRUNC((RECTANG[J].X2-
           RECTANG[J].X1)/2)+RECTANG[J].X1;
      Y23:=TRUNC((RECTANG[J].Y2-
           RECTANG[J].Y1)/2)+RECTANG[J].Y1;
      DX1:=ABS(XL3-X13);
      DY1:=ABS(YL3-Y13);
      DY2:=ABS(Y23-YL3);
      DX2:=ABS(X23-XL3);
      E:=0;
      IF (DX1<>0) AND (DY1<>0) THEN
         E:=1;
      IF (DX2<>0) AND (DY2<>0) THEN
         E:=E+1;
      D:=DX1+DX2+DY1+DY2;
      PESO:=(ALFA*D)+(BETA*E);
END;

BEGIN
      FOR I:=1 TO NR-1 DO
          FOR J:=I+1 TO NR DO
          IF MATACCESO[I,J].COSTO<>0 THEN
             BEGIN
```

```
                IF MATACCESO[I,J].INDAL<0 THEN
                   BEGIN
            {RECTIFICA INDICES MATRIZ ACCESO A AREAS LIBRES}
            MATACCESO[I,J].INDAL:=ABS(MATACCESO[I,J].INDAL);
            MATACCESO[J,I].INDAL:=ABS(MATACCESO[J,I].INDAL);
                   XL1:=AREALRV[MATACCESO[I,J].INDAL].X;
                   YL1:=AREALRV[MATACCESO[I,J].INDAL].Y;
                   XL2:=XL1;
                   YL2:=YL1+AREALRV[MATACCESO[I,J].INDAL].D;
                   END
                ELSE
                   BEGIN
                   XL1:=AREALRH[MATACCESO[I,J].INDAL].X;
                   YL1:=AREALRH[MATACCESO[I,J].INDAL].Y;
                   YL2:=YL1;
                   XL2:=XL1+AREALRH[MATACCESO[I,J].INDAL].D;
                   END;

                {ASIGNACION DE COSTO MATRIZ DE ACCESO}
                MATACCESO[I,J].COSTO:=PESO(I,J,XL1,XL2,
                                           YL1,YL2);
                MATACCESO[J,I].COSTO:=MATACCESO[I,J].COSTO;

                {IMPRESION COSTO ASICIADO DE RECTAN I A J}
                WRITELN(FILVAR,'COSTO ',I,' A ',J,' :=
                ',MATACCESO[I,J].COSTO);
          END;
END;

{PROCEDURE RUTAC - GENERA LA RUTA MAS CORTA}
{OBJETIVO : LOCALIZAR LA RUTA MAS CORTA PARA IR DE UN
            RECTANGULO A OTRO.
 PARAMETROS: MATRIZ - MATRIZ DE ACCESO ENTRE RECTANGULOS
             RUTA   - INDICES DE RECTANGULOS RUTA MAS CORTA
             O, D   - RECTANGULO ORIGEN Y DESTINO
 EXPLICACION: PARTIENDO DE QUE NO EXISTEN COSTOS NEGATIVOS,
              SE ALMACENAN LOS RECTANGULOS QUE DEBEN SER
              INCLUIDOS EN LA TRAYECTORIA DE LA RUTA MAS
              CORTA, TOMANDO EN CUENTA CUAL TIENE EL MENOR
              COSTO}
PROCEDURE RUTAC(MATRIZ:MATACESO;VAR RUTA:RT; O,D:INTEGER);
VAR
   W,X1,X2 : RT; {ARREGLOS PARA MANEJO RECTAN Y TRAYECTORIAS}
```

```
   I,J,K,DE,A,C : INTEGER; {INDICES PARA ARREGLOS}
BEGIN
   W[O]:=0; {COSTO INICIAL DEL NODO ORIGEN}
   X1[1]:=0; {ARREGLO DE RECTANGULOS (NODOS) "ORIGEN"}
   X2[1]:=0; {ARREGLO DE RECTANGULOS (NODOS) "DESTINO"}
   X1[0]:=1; {NUMERO DE NODOS EN LA RUTA}

   {COLOCA EN CEROS LOS COSTOS DE LA MATRIZ}
   FOR I:=1 TO NR DO
       MATRIZ[I,O].COSTO:=0;

   {CICLO PARA ANALIZAR CADA UNO DE LOS NODOS}
   REPEAT
     C:=9999;
     FOR I:=1 TO X1[0] DO
         BEGIN
         K:=X2[I];
         FOR J:=1 TO NR DO
             IF J<>K THEN

                 {VERIFICA QUE EL COSTO ES DIFERENTE DE CERO}
                 IF MATRIZ[K,J].COSTO<>0 THEN

                       {VERIFICA QUE EL COSTO ES EL MENOR}
                       IF W[K]+MATRIZ[K,J].COSTO<C THEN
                          BEGIN

                             {ALMACENA COSTO ARCO ENTRE
                             RECTANGULOS}
                             C:=W[K]+MATRIZ[K,J].COSTO;
                             DE:=K;
                             A:=J;
                          END;
         END;
      FOR I:=1 TO NR DO
          MATRIZ[I,A].COSTO:=0;

      {INCREMENTA NUMERO DE NODOS INCLUIDOS EN RUTA}
      X1[0]:=X1[0]+1;
      X1[X1[0]]:=DE;
      X2[X1[0]]:=A;
      W[A]:=C;
UNTIL A=D;
```

```
{IMPRIME SECUENCIA DE RECTANGULOS ACCESIBLES ENTRE SI}
FOR K:= 1 TO X1[0] DO
    WRITELN(FILVAR,'X1 ',X1[K],' X2 ',X2[K]);
RUTA[1]:=X2[X1[0]];
DE:=X1[X1[0]];
J:=1;

{ASIGNA RUTA MAS CORTA A ARREGLO DE SALIDA}
FOR I:=(X1[0]-1) DOWNTO 1 DO
    IF X2[I]=DE THEN
       BEGIN
          J:=J+1;
          RUTA[J]:=X2[I];
          DE:=X1[I];
       END;

{IMPRIME SECUENCIA DE RUTA MASA CORTA}
WRITELN(FILVAR,'SECUENCIA DE RUTA MAS CORTA');
FOR I:=1 TO J DO
    WRITELN(FILVAR,I:3,': RECTANGULO ',RUTA[I]);
END;

{PROCEDURE POTMAT}
{OBJETIVO: CALCULAR POTENCIA N-ESIMA DE UNA MATRIZ
 PARAMETROS: ORIGEN,DESTINO - CONTIENE NUMERO DEL RECTANGULO
                              ORIGEN Y DESTINO
                         A - MATRIZ DE ACCESO
                       BAN - BANDERA PARA DETECTAR FIN DE
                              CALCULO DE POTENCIA
                        NR - POTENCIA A ELEVAR MATRIZ
 EXPLICACION: SE GENERA EN BASE A SUMAS DE LAS
              MULTIPLICACIONES SUCESIVAS DE MATRICES}
PROCEDURE POTMAT(ORIGEN,DESTINO:INTEGER; A:MATACESO;VAR
BAN:BOOLEAN; NR:INTEGER);
VAR
    I,J,K,L:INTEGER; {INDICES AUXILIARES PARA CALCULOS}
    C,B:MATACESO; {MATRICES AUXILIARES PARA CALCULO}
BEGIN

   {INICIALIZACION MATRIZ CALCULO DE POTENCIA}
   FOR I:= 1 TO NR DO
       FOR J:= 1 TO NR DO
          B[I,J].COSTO:=A[I,J].COSTO;
```

```
    L:=1;
    WHILE L< NR DO {CICLO PARA LA POTENCIA DE MATRIZ}
       BEGIN

          {MULTIPLICACION DE MATRICES}
          FOR I:=1 TO NR DO
              FOR J:=1 TO NR DO
                  BEGIN
                     C[I,J].COSTO:=0;
                     FOR K:=1 TO NR DO
          C[I,J].COSTO:=B[I,K].COSTO*A[K,J].COSTO+C[I,J].COSTO;
                  END;

          {INTERCAMBIO PARA LA MULTIPLICACION}
          FOR K:=1 TO NR DO
              FOR J:=1 TO NR DO
                  B[K,J].COSTO:=C[K,J].COSTO;
          IF B[ORIGEN,DESTINO].COSTO<>0 THEN
             L:=NR;
          L:=L+1;
       END;
       IF B[ORIGEN,DESTINO].COSTO<>0 THEN
          BAN:='TRUE';

    {IMPRESION DE POTENCIA DE MATRIZ}
    WRITELN(FILVAR,'MATRIZ A');
    FOR I:=1 TO NR DO
        BEGIN
        FOR J:=1 TO NR DO
         WRITE(FILVAR,'A[',I:2,',',J:2,']=',A[I,J].COSTO,' ');
        WRITELN(FILVAR);
        END;
    WRITELN(FILVAR);
{    WRITELN(FILVAR,'MATRIZ B');
    FOR I:=1 TO NR DO
        FOR J:=1 TO NR DO
WRITELN(FILVAR,'B[',I:2,',',J:2,']=',B[I,J].COSTO);}
END;

{ PROGRAMA PRINCIPAL }

BEGIN
   {SE ASIGNA ARCHIVO DE SALIDA PARA IMPRESION}
   READLN(FILENAME);
```

```
ASSIGN(FILVAR,FILENAME);
REWRITE(FILVAR);

{ DATOS DE LAS PISTAS HORIZONTALES }

 PISTASH[1].X:=0;
 PISTASH[1].Y:=0;
 PISTASH[1].D:=11;
 PISTASH[2].X:=1;
 PISTASH[2].Y:=1;
 PISTASH[2].D:=3;
 PISTASH[3].X:=5;
 PISTASH[3].Y:=4;
 PISTASH[3].D:=3;
 PISTASH[4].X:=0;
 PISTASH[4].Y:=6;
 PISTASH[4].D:=11;

{ DATOS DE LAS PISTAS VERTICALES }

 PISTASV[1].X:=0;
 PISTASV[1].Y:=0;
 PISTASV[1].D:=6;
 PISTASV[2].X:=1;
 PISTASV[2].Y:=1;
 PISTASV[2].D:=2;
 PISTASV[3].X:=3;
 PISTASV[3].Y:=3;
 PISTASV[3].D:=2;
 PISTASV[4].X:=8;
 PISTASV[4].Y:=1;
 PISTASV[4].D:=3;
 PISTASV[5].X:=11;
 PISTASV[5].Y:=0;
 PISTASV[5].D:=6;

 {DATOS INICIALES PARA GENERACION RUTA ORIGEN-DESTINO}
 NPH:=4;
 NPV:=5;
 ANRH:=0;
 ANRV:=0;
 ARH:=0;
 ARV:=0;
 BAN:=FALSE;
```

```
    ORIGEN:=1;
    DESTINO:=6;
    ALFA:=6; {ASIGNA UN 60% DE PESO A LA DISTANCIA}
    BETA:=4; {ASIGNA UN 40% DE PESO A LA ENTROPIA}

    {INICIALIZA A CERO MATRIZ DE COSTOS}
    FOR I:=1 TO 10 DO
    FOR J:=1 TO 10 DO
       BEGIN
           MATACCESO[I,J].COSTO:=0;
           MATACCESO[I,J].INDAL:=0;
       END;
    GENLIN(PISTASH,PISTASV,NPH,NPV,LINEASH,LINEASV,NLH,NLV);
    GENREC(LINEASH,LINEASV,NLH,NLV,RECTANG,NR);
    ALCANCE(PISTASH,PISTASV,NPH,NPV,RECTANG,NR);
    POTMAT(ORIGEN,DESTINO,MATACCESO,BAN,NR);
    IF BAN THEN {VERIFICA QUE EXISTA PASO DE ORIGEN-DESTINO}
       BEGIN
          CALCULA_COSTO;
          RUTAC(MATACCESO,RUTA,ORIGEN,DESTINO);
          WRITELN(FILVAR);
          CLOSE(FILVAR);
       END;
END.
```

APENDICE 2

RESULTADO DEL PROGRAMA

GENERACION DE LINEAS HORIZONTALES Y VERTICALES

NUM LIN	LIN-HOR X	Y	D		LIN-VER X	Y	D
1	0	0	11	-	0	0	6
2	0	1	11	-	1	1	5
3	3	4	8	-	3	1	5
4	0	6	11	-	8	1	3
5	0	0	0	-	11	0	6

RECTANGULOS GENERADOS

RECTANGULO No.1	X1:0	Y1:0	X2:11	Y2:1
RECTANGULO No.2	X1:0	Y1:1	X2: 1	Y2:6
RECTANGULO No.3	X1:1	Y1:1	X2: 3	Y2:6
RECTANGULO No.4	X1:3	Y1:1	X2: 8	Y2:4
RECTANGULO No.5	X1:8	Y1:1	X2:11	Y2:4
RECTANGULO No.6	X1:3	Y1:4	X2:11	Y2:6

AREAS LIBRES DE LOS RECTANGULOS

PUNTO (X , Y)	LONGITUD	RECTANGULO
(0 , 1) -	1 -	1
(4 , 1) -	7 -	1
(0 , 1) -	1 -	2
(1 , 3) -	3 -	2
(1 , 3) -	3 -	3
(3 , 1) -	2 -	3
(3 , 5) -	1 -	3
(4 , 1) -	4 -	4
(3 , 4) -	2 -	4
(3 , 1) -	2 -	4
(8 , 1) -	3 -	5
(8 , 4) -	3 -	5
(3 , 4) -	2 -	6
(8 , 4) -	3 -	6
(3 , 5) -	1 -	6

MATRIZ QUE INDICA EL AREA LIBRE QUE UNE DOS RECTANGULOS

(R_1, R_2)	CVE	# AREA LIBRE	
(1, 2)	= 1 :	1	
(1, 4)	= 1 :	2	
(1, 5)	= 1 :	3	(NEGATIVA = AREA VERTICAL)
(2, 1)	= 1 :	1	
(2, 3)	= 1 :	-1	
(3, 2)	= 1 :	-1	
(3, 4)	= 1 :	-2	
(3, 6)	= 1 :	-3	(POSITIVA = AREA HORIZONTAL)
(4, 1)	= 1 :	2	
(4, 3)	= 1 :	-2	
(4, 6)	= 1 :	4	
(5, 1)	= 1 :	3	
(5, 6)	= 1 :	5	
(6, 3)	= 1 :	-3	
(6, 4)	= 1 :	4	
(6, 5)	= 1 :	5	

AREAS LIBRES RESUELTAS HORIZONTALES

# A. LIBRE	(X, Y , Distancia Horizontal)
1	- (0, 1 , 1)
2	- (4, 1 , 4)
3	- (8, 1 , 3)
4	- (3, 4 , 2)
5	- (8, 4 , 3)

AREAS LIBRES RESUELTAS VERTICALES

# A. LIBRE	(X, Y , Distancia Vertical)
1	- (1, 3 , 3)
2	- (3, 1 , 2)
3	- (3, 5 , 1)

MATRIZ DE ADYACENCIA (LOS INDICES REPRESENTAN RECTANGULOS)

A[1,1]=0	A[1,2]=1	A[1,3]=0	A[1,4]=1	A[1,5]=1	A[1,6]=0
A[2,1]=1	A[2,2]=0	A[2,3]=1	A[2,4]=0	A[2,5]=0	A[2,6]=0
A[3,1]=0	A[3,2]=1	A[3,3]=0	A[3,4]=1	A[3,5]=0	A[3,6]=1
A[4,1]=1	A[4,2]=0	A[4,3]=1	A[4,4]=0	A[4,5]=0	A[4,6]=1
A[5,1]=1	A[5,2]=0	A[5,3]=0	A[5,4]=0	A[5,5]=0	A[5,6]=1
A[6,1]=0	A[6,2]=0	A[6,3]=1	A[6,4]=1	A[6,5]=1	A[6,6]=0

DETERMINACION DE COSTOS ENTRE LOS RECTANGULOS

	R1		R2		COSTO
	==		==		=====
COSTO	1	A	2	:=	52
COSTO	1	A	4	:=	32
COSTO	1	A	5	:=	40
COSTO	2	A	3	:=	32
COSTO	3	A	4	:=	28
COSTO	3	A	6	:=	46
COSTO	4	A	6	:=	50
COSTO	5	A	6	:=	34

SECUENCIA DE RECTANGULOS ACCESIBLES PARA RUTA MAS CORTA

	R1		R2
	==		==
X1	0	X2	1
X1	1	X2	4
X1	1	X2	5
X1	1	X2	2
X1	4	X2	3
X1	5	X2	6

SECUENCIA DE RUTA MAS CORTA

SECUENCIA		RECT
=========		====
1	:	RECTANGULO 6
2	:	RECTANGULO 5
3	:	RECTANGULO 1

REFERENCIAS

1 - Sutaner Hans. "Circuitos Impresos" Fabricación Ed. Marcombo, S.A., 1969. Segunda Edición.

2 - Bauer Alfred. "Circuitos Impresos" Diseño y Realización, Ediciones CEAC, 1982. Primera Edición.

3 - Hillier & Lieberman. "Introducción a la Investigación de Operaciones", Ed Mc Graw Hill, 1982. Tercera Edición.

4 - Breuer Melvin A. "Design Automation of Digital System" Theory and Techniques, Ed. Prentice Hall, 1972. Volume I

5 - Belter Stephen E. "Computer Aided Routing of Printer Circuit Boards" An examination of Lee's Algorithm and possible enhancements, Revista BYTE de Junio de 1987.

6 - Tremblay & Sorenson. "An Introduction to Data

Structures with Applications", Ed. Mc Graw Hill, 1981.

7 - Bazaraa & Jarvis. "Programacion Lineal Y Flujo en Redes", Ed. Limusa, 1986. Versión traducida al español por Onésimo Hernandez Lerma.