CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITECNICO NACIONAL.

DEPARTAMENTO DE INGENIERIA ELECTRICA

SECCION COMPUTACION

RUTAS DE DISTANCIA MINIMA SOBRE UN MODELO JERARQUICO DE TERRENO

Tesis que presenta el Ing. Jesús Vázquez Gómez para obtener el grado de MAESTRO EN CIENCIAS en la especialidad de INGENIERIA ELECTRICA. Trabajo dirigido por el Dr. Renato Barrera Rivera.

Becario de COSNET.

México D.F. Junio 1988.

A mis padres.

A mis hermanos.

A mis amigos.

AGRADECIMIENTOS:

Agradezco a mi familia todo el apoyo y comprensión que me brindaron durante los estudios de maestría, así como en la realización de esta tesis.

Al Dr. Renato Barrera, por su dirección y paciencia en el desarrollo de este trabajo.

Al Dr. Gilberto Calvillo y al Dr. Cristóbal Vargas, por el tiempo que dedicaron a la revisión de la presente tesis.

Al Lic. Jorge Buenabad Chávez, por sus comentarios, ideas y ánimos en el transcurso de mis estudios de maestría.

Al Dr. Armando Maldonado, por su apoyo en la terminación del presente trabajo.

A COSNET, por proporcionarme la beca crédito para financiar mis estudios.

A todos mis compañeros y amigos del CINVESTAV por un sinnúmero de buenos detalles.

A mis amigos, quienes de alguna u otra forma siempre me apoyan.

INDICE

RESUMEN

TITULO: Rutas de Distancia Mínima sobre un Modelo Jerárquico de Terreno.

- Se presenta un algoritmo para encontrar Rutas de Longitud Mínima entre dos puntos dados (origen y destino) de un terreno.
- El terreno se representa mediante un poliedro de caras triangulares.
- Se ha considerado conveniente adoptar un modelo Jerárquico de terreno en el que las caras triangulares del poliedro pueden ser subdivididas en triangulaciones más finas, hasta lograr la precisión deseada.
- Este trabajo se apoya principalmente en dos algoritmos :
  1) El de Mount [M-85], que presenta una solución al problema de encontrar una Ruta de Longitud Mínima en a superficie de un poliedro.
  2) El de Dijkstra [AHU-74], para obtener Rutas de Longitud Mínima en Gráficas Dirigidas.
- El algoritmo que se propone aprovecha as peculiaridades de los mocelos jerárquicos para resolver una serie de problemas aproximados. En la solución de tales problemas se eliminan "a priori" ciertas regiones del terreno por las que es imposible que pase la ruta de distancia mínima.
- Finalmente, se aplica un método exacto para obtener el camino más corto entre Origen y Destino, considerando

sólamente la porción de terreno que no ha sido eliminada en la serie de problemas aproximados.

INTRODUCCION.

Este trabajo se refiere a la obtención de una Ruta de Longitud Mínima que conecte un punto Origen con un punto Destino dentro una cierta región de terreno. Se presupone contar con una base de datos con información del terreno, y que esta información está organizada en forma jerárquica. La susodicha base de datos contiene la ubicación y la altura de puntos significativos del terreno.

El problema de rutas mínimas en poliedros ha sido tratado entre otros por Mount [M-85]; O'Rourke, Suri y Booth [OSB-84] y Sharir y Schorr [SS-84].

Sharir y Schorr presentan un algoritmo para encontrar la ruta de longitud mínima sobre un poliedro convexo. La complejidad de dicho algoritmo es de $O(n^3 \log n)$.

O'Rourke, Suri y Booth desarrollaron un algoritmo de complejidad $O(n^5)$, para encontrar la ruta de longitud mínima entre dos puntos sobre un poliedro no convexo.

Por último, Mount propone un algoritmo de orden $O(n^2 \log n)$; este algoritmo mejora los resultados de O'Rourke, Suri y Booth, para obtener la ruta de longitud mínima entre dos puntos sobre un poliedro no necesariamente convexo.

El propósito de esta tesis es el de reducir la complejidad de los algoritmos mencionados, para lo cual se propone un algoritmo del tipo "ramificación y acotamiento" previo a la solución exacta del problema final.

Esta "ramificación y Acotamiento" consta pues de dos fases:

a)- Acotamiento. En la que se genera una aproximación

"buena" a la ruta mínima dentro de la región a considerarse. La longitud de dicha aproximación acota por arriba a la longitud mínima.

b)- Reducción: En la que, con base en la cota anterior, se descartan de la región a considerarse porciones por las que es imposible que pase una ruta mínima.

Para realizar lo anterior se contará con un modelo jerárquico del terreno, en el que la jerarquía está dada por una serie de representaciones de creciente nivel de precisión. En cada nivel de precisión se tiene una partición triangular cuyos vértices tienen altura conocida; para aproximar la altura de cualquier otro se interpola linealmente entre los vértices del triángulo que lo contiene. Los vértices correspondientes a un cierto nivel de precisión son obtenidos seccionando los triángulos del nivel anterior. La razón para organizar el modelo en niveles es la de hacer un compromiso entre exactitud y volumen de información: los primeros niveles tienen información muy gruesa, aunque muy sucinta.

Sobre este modelo jerárquico se resolverán una serie de problemas aproximados de distancia mínima, comenzando con el nivel de precisión más gruesa, y prosiguiendo con los niveles de mejor ajuste. La información lograda al resolver los problemas aproximados en niveles de menor precisión permitirán descartar "a priori" grandes áreas (reducción), con lo que al resolverse el problema "real" mediante un método exacto, se habrá disminuido enormemente el volumen de cálculos involucrados.

En el capítulo I se describe el modelo de terreno utilizado

y los antecedentes en que se basa este trabajo.

En el capítulo II se describe el algoritmo propuesto para encontrar rutas de longitud mínima.

Finalmente, en el capítulo III se presentan los resultados y conclusiones del trabajo realizado.

En el apéndice A, se describe el concepto de "amplificación".

CAPITULO I. ANTECEDENTES

Este trabajo se apoya principalmente en el algoritmo de Mount [M-85]. Dicho autor utiliza una construcción geométrica que llama cuña para obtener la ruta de longitud mínima en un poliedro, y se apoya a su vez en el algoritmo de Dijkstra [AHU-74] para el camino más corto en gráficas.

La representación del terreno utilizado es descrito en la sección 1.1. y el algoritmo de Mount en la sección 1.2.

1.1 REPRESENTACION DEL TERRENO

En esta sección se muestra la forma en que se representa un terreno, así como algunos procedimientos para obtener información del mismo.

En la sección 1.1.1 se presenta el modelo seleccionado para representar el terreno, y en la 1.1.2 se describe la forma en que se almacena la información correspondiente.

1.1.1 MODELO DEL TERRENO

Como se ha dicho ya, en este trabajo el terreno se representa mediante una superfice poliédrica.

Un poliedro se define como un conjunto finitd conexo de polígonos planos (llamados "caras"), tal que cada lado de cada polígono es común con otro polígono.

El poliedro forma una superficie cerrada simple, y descompone el espacio en dos regiones, una de las cuales, llamada "interior", es finita.

Un poliedro es convexo si sus caras o su interior contienen al total de segmentos que unen cada par de los vértices de sus caras.

En este trabajo las caras del poliedro serán triangulares. En la siguiente figura se muestra un poliedro no convexo con caras triangulares.

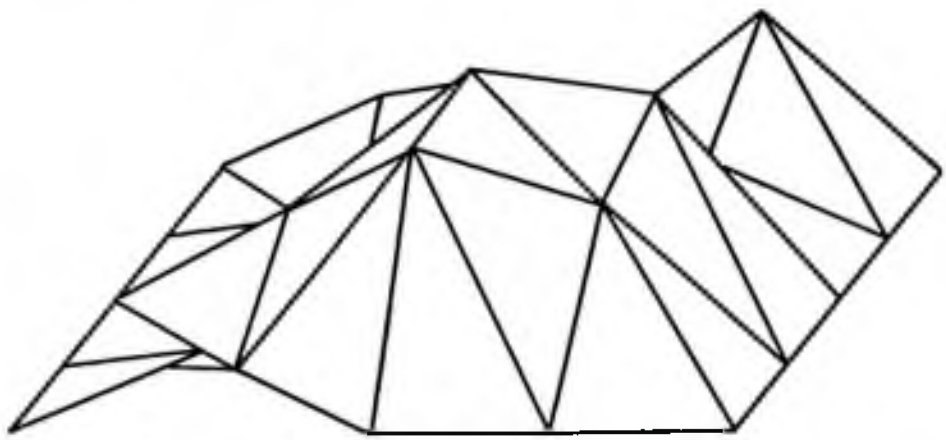


figura 1.1 Poliedro no convexo con caras triangulares.

Representación

La representación de terrero utilizada es una "jerárquica", con crecientes niveles de precisión. Para todo nivel de esta jerarqula la región se subdivide en un conjunto de triángulos rectángulos (cuyo número depende de la precisión), de cuyos vértices se corocen las coordenadas (x,y,z). Para conocer la altura de un punto cualquiera se interpola linealmente entre los vertices del triángulo que lo contiene. Una triangulación de un nivel dado se ilustra en las siguiente figura:

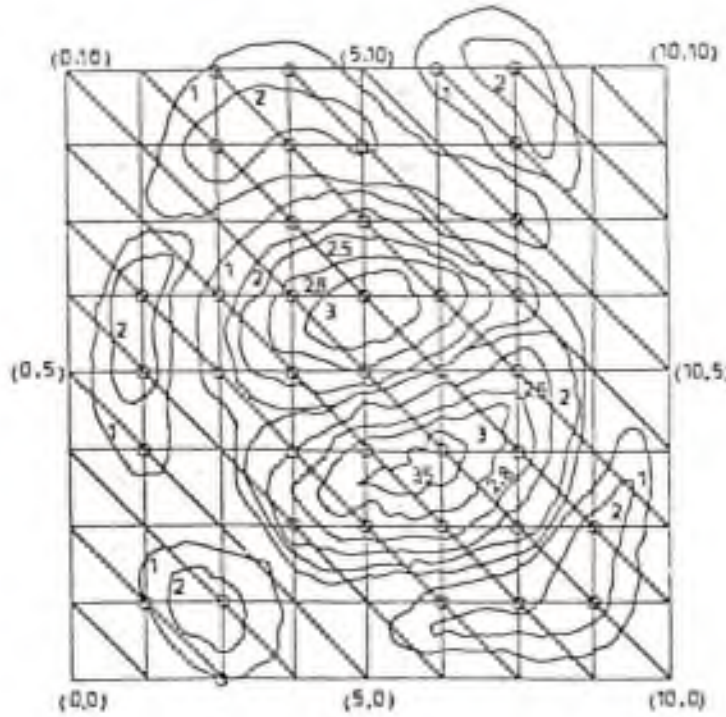


figura 1.2.1 Partición para un nivel dado de precisión

Como se mencionó en la introducción, los triángulos de un nivel de precisión dado se obtienen subdividiendo los del nivel anterior. En particular, los nuevos triángulos se generan utilizando los puntos medios de los lados del triángulo padre, como lo muestra la figura 1.2.2.

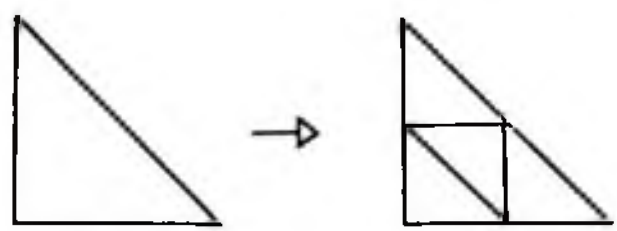


figura 1.2.2 Generación de nuevos triángulos

Así pues, la base de esta jerarquía es una partición inicial RUDA de 8 triángulos, almacenada en el archivo ARCH0; le sigue una partición de 32 triárgulos que está almacenada en un archivo ARCH1, y asi sucesivamente hasta el archivo ARCHn que es el de mejor precisión.

La figura 1.3.1 muestra 3 niveles de aproximación para un terreno, mismos que constan de 8,32 y 128 triángulos.

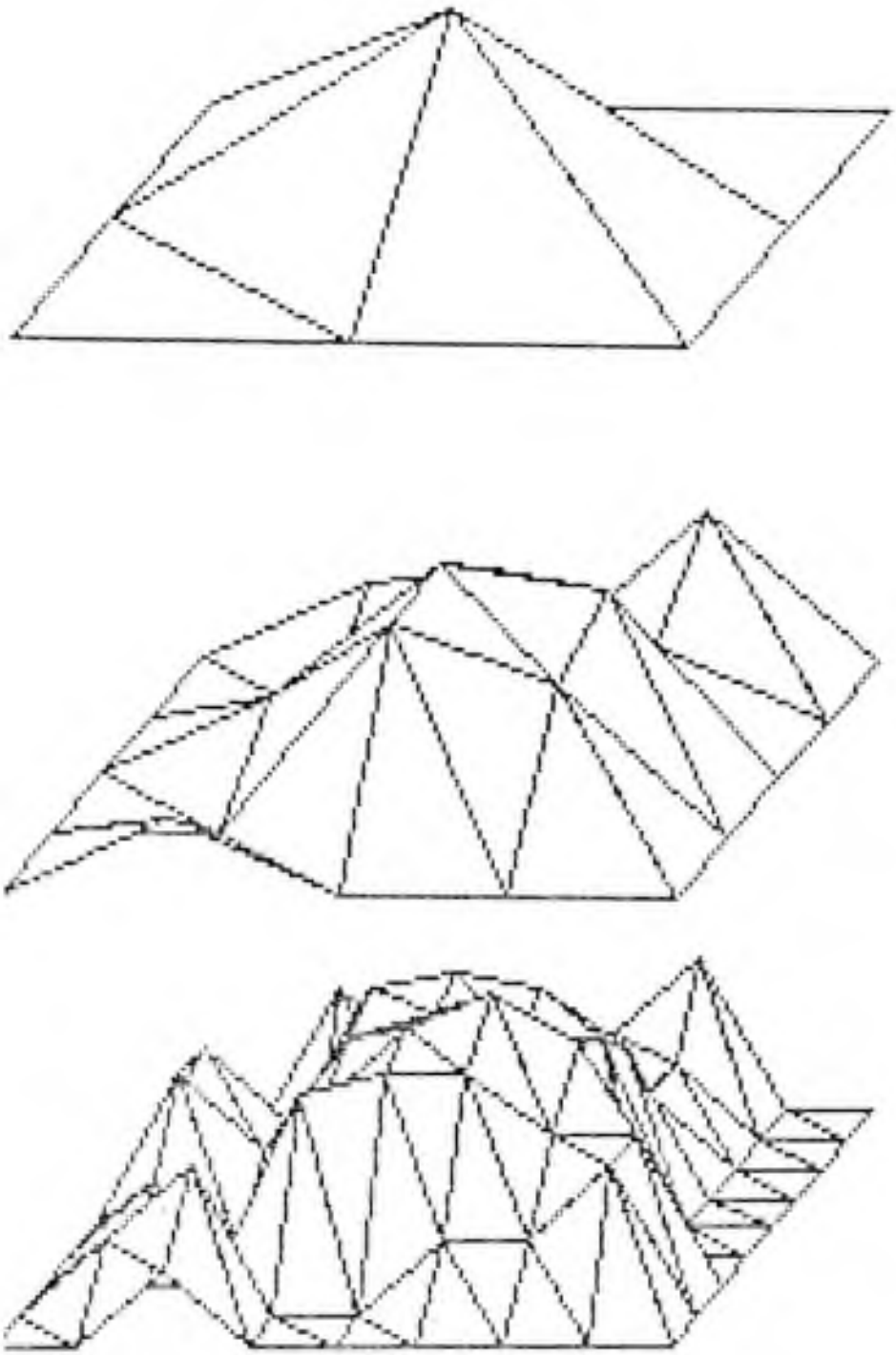


figura 1.3.1 Modelo Jerárquico

1.1.2 CLAVES DE LOS VERTICES Y LOS LADOS

En este trabajo, y por la conveniencia de utilizar el concepto de "monotonicidad" descrito en la sección 1.2.1., un lado que sea común a dos caras será representado mediante dos distintas versiones o "lados internos", uno para cada cara que lo contiene.

Los algoritmos necesarios para implantar este trabajo utilizan muy frecuentemente tanto la operación de accesar la información de un vértice (o lado interno) dado, como la de obtener una lista de sus vecinos. Con el fin simplificar estas operaciones es conveniente asociar claves a cada vértice y a cada "lado interno". Estas claves son obtenidas utilizando las coordenadas (x,y) de los vértices mediante "entretejido de bits" [BV-84]. Este método consiste simplemente en expresar en forma binaria las coordenadas "x" "y" de un punto, y obtener de ellas un nuevo número binario cuyos campos entretejen un bit de la ordenada con uno de la abcisa.

Se considera que los "lados internos" de una cara cualquiera en la triangulación están orientados en el sentido de las manecillas del reloj, lo que permite hablar de un vértice inicial y de uno final de un lado interno.

Como hemos escogido una partición en triángulos rectángulos equiláteros, un lado puede tener seis direcciones, cuyo código es el siguiente:

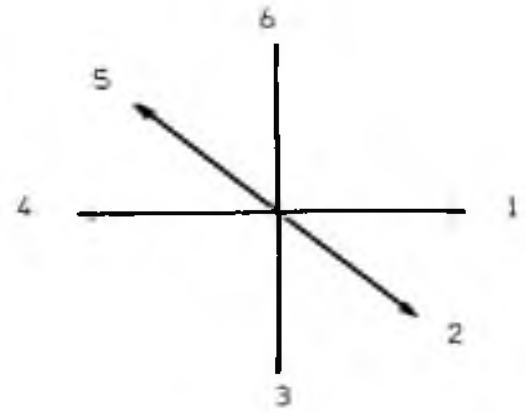


figura 1.3.2 Código de direcciones

La orientación y el vértice inicial de un lado interno nos permiten identificar su vértice final, ya que la longitud de un lado depende de la orientación y el nivel de precisión. Por tanto, una orientación y un vértice inicial dados determinan completamente un lado interno.

Aprovechando tal hecho formaremos la clave de un lado interno concatenando su dirección con las "coordenadas entretejidas" de su vértice inicial.

Así, cuando una clave que inicie con '1' indica que su vértice final se encuentra en la dirección 1, es decir, a la derecha sobre el eje de las X's del vértice inicial del lado.

Para poder representar mediante el mismo formato las claves de vértices y de lados internos asignaremos a los vértices una dirección ficticia '0'.

Ejemplo: Suponiendo que se tiene la clave '41400', entonces:

- Se está hablando de un lado interno
- El vértice final está en la dirección 4, es decir, a la izquierda sobre el eje de las X's del vértice inicial.

- La parte de la clave '1400', nos indica que las coordenadas del vértice inicial son (C/2,C/2).
- Y finalmente, como sabemos que el vértice final está a la izquierda sobre el eje de las X's, las coordenadas del vèrtice final son (0,C/2).

  Donde C es el valor máximo que pueden tomar X o Y.

En nuestro trabajo, utilizamos los siguientes procedimientos para manipular las claves:

- Procedimiento ClaveOri(.), se obtiene la clave de un lado, incluyendo la parte del mismo.
- ClaveaCoor(clave), regresa las coordenadas del vértice representado por clave.
- Adyacente(clave), obtiene las lados adyacentes a clave, donde clave puede ser un lado o un vértice.
- ClaveOpuesta(clave), regresa la clave del lado representado por clave, pero en dirección contraria. (es decir, la otra parte del lado)

De esta forma, con estos procedimientos es posible obtener información del terreno en un cierto nivel de precisión.

1.2 ALGORITMO DE MOUNT.

El problema tratado en este trabajo se refiere a la obtención de rutas de longitud mínima entre dos puntos sobre un terreno. Uno de estos puntos es el Origen de la ruta, el otro es el Destino.

Como se mencionó en la sección 1.1 el terreno es representado por una superficie poliédrica no convexa con caras triangulares.

El algoritmo de Dijkstra podría ser usado en el caso de que se viajara sólo a través de vértices y lados; desgraciadamente, las rutas mínimas sobre un poliedro no siempre están sobre lados y típicamente cruzan caras, lados y vértices.

Para encontrar el camino más corto en un terreno representado por un poliedro nos apoyaremos en el algoritmo propuesto por Mount [M-85], que a continuación se describe.

1.2.1 GENERALIDADES.

Este algoritmo mejora los resultados obtenidos por Sharir y Schorr [SS-84] y los de O'Rourke, Suri y Booth [OSB-84] para el problema de encontrar caminos cortos en la superficie de un poliedro (convexo o no).

El algoritmo es una generalización de las técnicas usadas por Sharir y Schorr [SS-84], que a su vez generalizan el algoritmo de Dijkstra [AHU-74]. Este trabajo es válido para poliedros no convexos como el mostrado en la figura 1.1. Se asume que las caras son triangulares y además, que los puntos origen y destino (de la ruta buscada) coinciden con dos vértices

de la triangulación. Esto no es una restricción, ya que si un punto dado no coincide con un vértice se pueden extender lados a partir de dicho punto hacia los vértices de la cara que lo contiene.

DESDOBLAMIENTO DE UNA CARA

Con el fin de simplificar el cálculo involucrado en la obtención de rutas sobre un poliedro es conveniente "desdoblar" planarmente tal poliedro a lo largo de una ruta

Desdoblar es el procedimiento mediante el cual dos caras adyacentes son rotadas sobre su lado común, hasta que ambas están sobre un mismo plano.

Se puede apreciar que al viajar sobre un poliedro la distancia de un camino que cruza de una cara c1 a otra c2 no es afectada por el ángulo entre caras; esto se observa fácilmente al desdoblar las caras con respecto al lado común entre ellas, de tal manera que ambas queden sobre un mismo plano. La operación de desdoblar un poliedro se conoce como "desdoblamiento planar". Lo anterior se puede generalizar a una cadena de caras, como se muestra en la figura 1.4.b, donde se desdobla el poliedro de la figura 1.4.a. en las caras {A, B, C, D, E y F} y en las caras {A, B, G, H, E y F}.

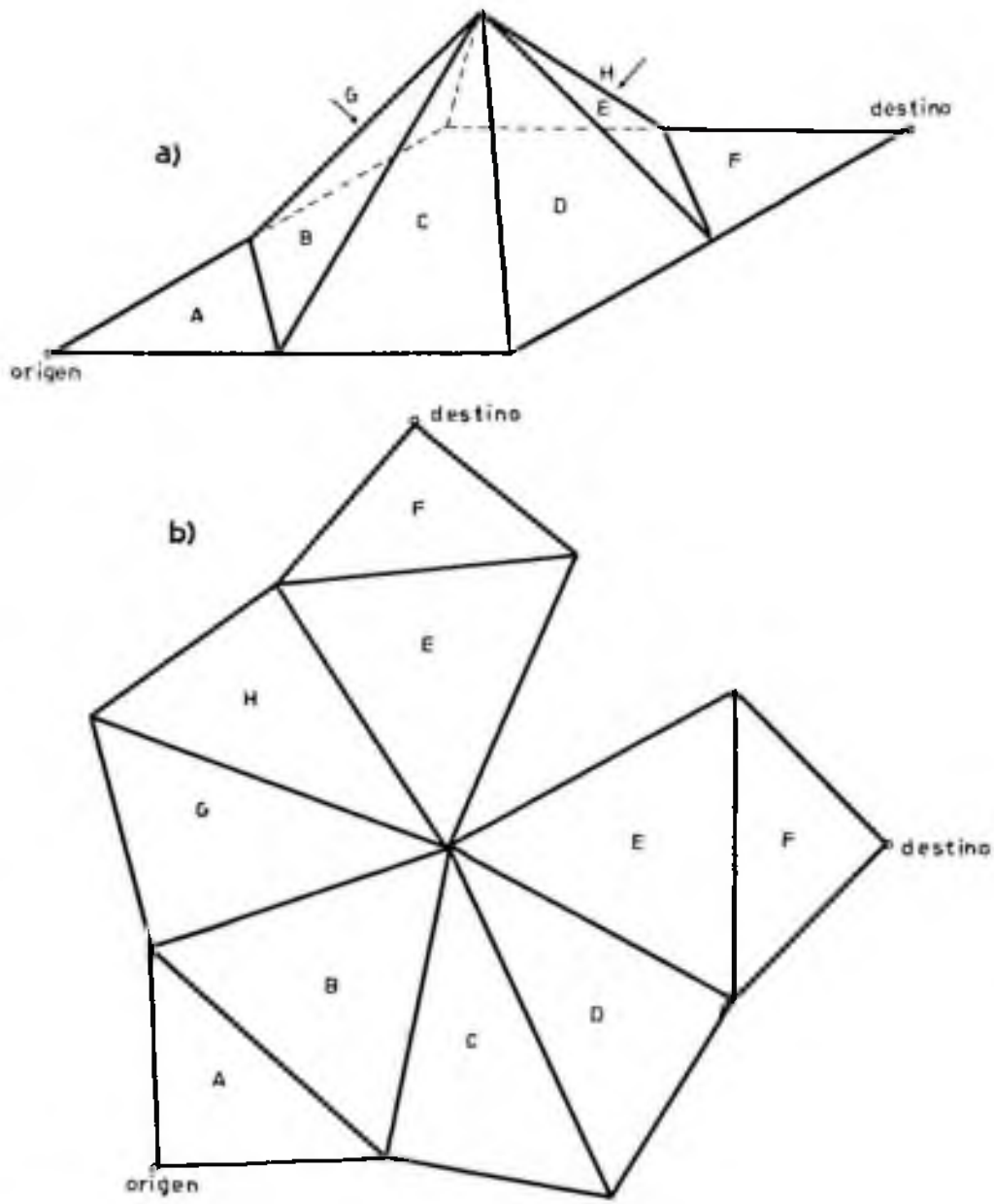


figura 1.4 Desdoblamiento Planar

Para "desdoblar" dos triángulos con vértices ABC y BCD, se procede de la siguiente manera:

-Primero se obtiene la distancia L entre dos vértices del primer triángulo (ABC), digamos C y A. Se considera que cada uno de éstos vértices queda completamente ubicado por medio de sus coordenadas (x,y,z). Ver la figura 1.5.a.

-El vértice C se coloca en las coordenadas (x,y), mientras que A en (x,y+L). Ver figura 1.5.b.

-Posteriormente se calculan las distancias de A a B (r1), y de B a C (r2).

-Con A y C como centros se construyen circunferencias de radios r1 y r2 respectivamente, se obtiene la intersección de ambas circunferencias.

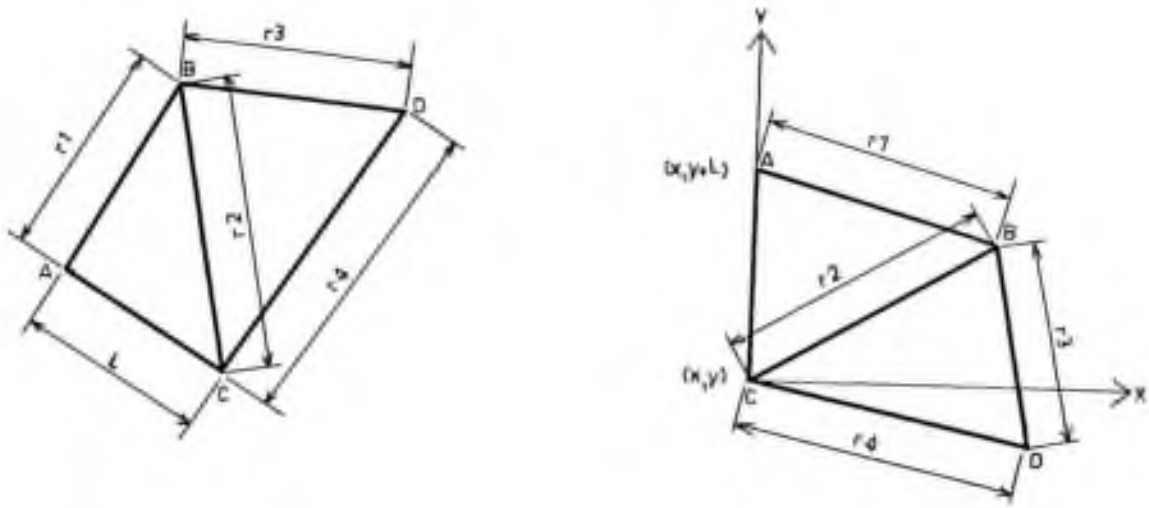
-Finalmente, con los dos puntos obtenidos en la intersección anterior, se prueba cuál de ellos corresponde al vértice B.

Para "desdoblar" al triángulo BCD con respecto al triángulo ABC, debemos considerar que en la representación "desdoblada" obtenida para el triángulo ABC se asignó una posición a los vértices B y C, por lo que sólo resta obtener la ubicación del vértice D de la siguiente manera:

-Se obtienen las distancias de B a D (r3) y de C a D (r4).

-Con B y C como centros se construyen circunferencias de radios r3 y r4 respectivamente, se obtiene la intersección de ambas circunferencias.

-Finalmente, con los dos puntos obtenidos en la intersección anterior, se prueba cuál de ellos corresponde al vértice D.



a) vista tridimensional b) desdoblamiento

figura 1.5. Descoblamiento de una cara.

Lo anterior se implementa en el procedimiento Unflat(A,B,r1,r2), donde A y B son los centros de las circunferencias construidas y, r1 y r2 los radios de las mismas respectivamente.

CUÑAS

Para desarrollar un algoritmo eficiente Mount propone la introducción de un elemento llamado cuña.

La cuña describe una región en una cara dentro de la cual todos los caminos más cortos desde el origen tienen la misma historia. Dos caminos tienen la misma "historia" cuando:

a) Surgen de un mismo origen y

b) Cruzan la misma secuencia de caras, lados y vértices.

Una cuña se define como el cuádruplo <o,b,d,e>, donde:

"e" = lado sobre el que incide la base de la cuña.

"b" = base de la cuña.

"o" = origen de la cuña.

"d" = distancia del camino más corto entre "o" y el punto inicial I del camino. (figura 1.6).

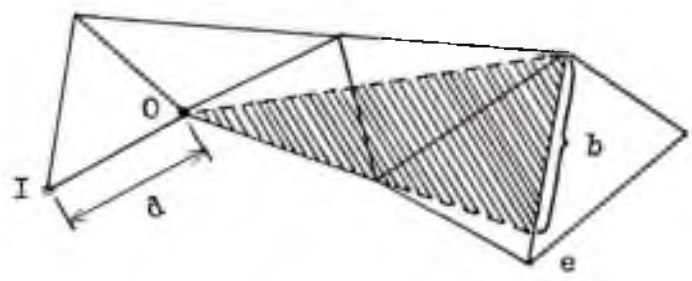


figura 1.6 Una cuña mostrada sobre un desdoblamiento plano.

A continuación se presenta la descripción de los atributos de una cuña para su implementación:

- Clave del lado interno sobre el que incide la cuña.
- Base de la cuña.
- Coordenadas del origen relativo.
- Peso del origen relativo.
- Distancia de la Base al origen relativo.
- Ancestro.

Clave del lado interno. Sirve para identificar a la cuña.

Base de la Cuña. Es el conjunto de puntos de "e" sobre el cual los caminos cortos cruzan la misma secuencia de caras,

lados y vèrtices a partir del origen "o". El origen relativo es ya sea el origen, o el último vèrtice que haya cruzado uno de de los caminos a los que corresponde la cuña. Este origen relativo indica la posición relativa de un vèrtice (el origen de la cuña) respecto al lado donde reposa la base de la cuña, y está especificado por sus coordenadas respecto al lado. Para obtener èstas coordenadas se procece de la manera siguiente:

Coordenadas del origen relativo. Como se vio en 1.1.2, un lado "e" (con longitud L) tiene un extremo inicial y otro final. Sea d1 (d2) la distancia de el extremo inicial (final) de "e" al origen relativo "o". Para obtener las coordenadas de "o" con respecto a "e", se procede asl:

- El extremo inicial del lado "e", es colocado en las coordenacas (0,0).
- El extremo final se coloca en (L,0).
- Se construyen dos circunferencias con centros en los extremos inicial y final de "e", y con radios d1 y d2 respectivamente. Las coordenadas del origen relativo "o", coninciden con la de uno de los puntos de intersección de estas circunferencias.

El cálculo de las nuevas coorcenadas del origen relativo se efectúa con el fin de facilitar las operaciones en los procedimientos de Ajuste de cuñas y Cuña Subtendiente, como se verá más adelante.

Peso del origen relativo. Proporciona la distancia acumulada desde el inicio del camino hasta el vèrtice al que

el origen relativo corresponde.

Ancestro. Es la cuña que al "extenderse" generó a la actual. Cuando se ha llegado al destino, es fácil reconstruir mediante los ancestros el camino andado. El proceso de "extender" una cuña será descrito más adelante.

La estructura de datos (en Pascal) utilizada para almacenar las antedichas características de una cuña es la siguiente :

```
{ ------------------------------------------------------------
   Para almacenar la información de una Cuña.
------------------------------------------------------------}
  ApRecWedge   =  ^RecWedge;   {apuntador al registro del tipo
                                RecWedge}
  RecWedge     = RECORD                    {registro de la cuña }
                   Wed       : integer;     {número de cuña }
                   PointCla  :  str15;      {clave del lado o
                                             vértice a extender}
                   orel3     : Coordenadas; {origen relativo.}
                   orelref   : Coordenadas; {o.relativo desdoblado}
                   DistOrel  :  real;       {distancia mínima al
                                                 origen realtivo}
                   peso,          {distancia de o.relativo a ORIGEN}
                   dist: real;    {distancia de la base de la cuña
                                   al ORIGEN }
                   inicio,       {coordenadas del inicio del lado
                                    desdoblado (en el sentido
                                   de las manecillas del reloj) }
                   final,         {coordenadas del vértice final del
                                   lado desdoblado}
                   baI,baF   :  Coordenadas; {coordenadas del inicio
                                          y  final  de la base de  la cuña}
                   padre,                    {apuntador al padre}
                   next        :   ApRecWedge;    {apuntador  a la
                              siguiente cuña ordenada por el campo dist}
                'END;
```

COMENTARIOS:

La estructura de datos anterior es un registro del lenguaje PASCAL. En ella suponemos que la base de la cuña está conectada, y que por tanto bastan dos puntos para describirla (baI, baF).

Posteriormente, al hablar de "Monotonicidad", se verá que el método utilizado rinde bases conectadas.
"Padre" apunta al la cuña que generó a la actual y permite reconstruir la ruta mínima. Por último, "next" es un apuntador usado por la cola de prioridades que se utiliza en el algoritmo de Mount.

MONOTONICIDAD.

Lee y Preparata [LP-84], observaron que el problema del camino más corto puede ser simplificado considerablemente si los caminos muestran ser monótonos, esto es, si la rutas de longitud mínima siempre cruzan un lado en una sola dirección. Concretamente, demostraron que si un problema era monótono, la base de una cuña era conexa.

Aunque puede ocurrir que los caminos cortos sobre un poliedro no sean monótonos, puede simularse la monotonicidad imaginando que cada lado es dividido en dos copias o partes (lados internos), una asociada con los caminos que cruzan el lado por la izquierda, y la otra con los caminos que lo cruzan por la derecha. Esta técnica es conocida como "monotonicidad estructurada", y aunque incrementa un poco el número de cuñas, el tiempo de procesamiento que requiere cada cuña disminuye considerablemente puesto que se simplifica el tratamiento de las bases.

En este trabajo con el fin de lograr monotonicidad, se considera que un lado consta de dos partes: Una perteneciente a una cara C1 y la otra parte a la cara C2 (fig 1.7.), es decir, una parte será el lado interno de C1 y la otra parte será el lado

interno de C2. (ver sección 1.1.2)

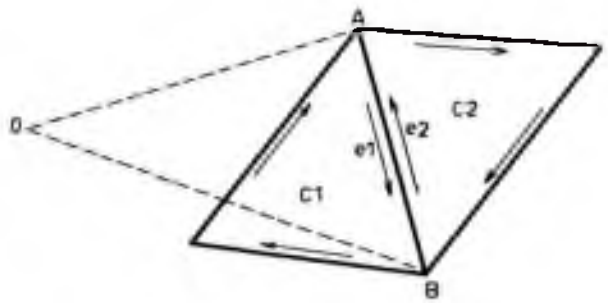


figura 1.7 Partes de un Lado.

Así, en la Fig. 1.7 el lado AB genera dos lados internos: e1, que va de A a B y e2, que va de B hacia A.

GEODESICAS.

Se define una "geodésica" como un camino en el poliedro que es localmente el camino más corto. En [M-85] se establecen las propiedades de las geodésicas de la siguiente manera:

1) La restricción de un camino más corto a una cara es un segmento de línea recta. La restricción de una geodésica a una cara es un segmento de línea.
2) Una geodèsica pasa a través del interior de un lado de tal forma que el desdoblamiento planar del camino corto es una línea recta.
3) Una geodésica que viaja a través de un vértice forma un ángulo mayor o igual a 180^{o} con respecto a èste. Para el caso de un poliedro convexo, la suma de los ángulos con respecto a cualquier vértice es menor a 360^{o}, de ahí que

las geodèsicas en este caso no pasan por los vèrtices (a menos que el Origen y el Destino de la ruta, correspondan a los extremos de un lado).

LINEA DE VISTA

Si se desea cruzar de la cara C1 a la cara C2 a través de un lado "e" a partir de un origen O (figura 1.8.a) es necesario que la parte del lado "e" que pertenece a C1 esté orientada en el sentido de las manecillas del reloj con respecto a O, como se ilustra en la figura 1.8.a. De ocurrir lo anterior se dice que el lado "e" sí presenta **línea de vista** al origen O.

La línea de vista que ofrece un lado a un origen es la parte de ese lado sobre la cual los caminos cortos cruzan la misma secuencia de caras, lados y vèrtices a partir del origen. (figura 1.8.a).

En la representación desdoblada de la figura 1.8.b se muestra el caso en que los caminos que se originan en O no pueden cruzar de la cara C1 a la C2 a través del lado AB. Esto se debe a que la parte del lado AB que pertenece a la cara C1 está orientada en sentido contrario a las manecillas del reloj con respecto a O.

Para verificar que un lado cualquiera ofrezca línea de vista a un origen dado se implementa la función PorDonde(.), la cual regresa un '0' cuando no hay línea de vista y regresa un '1' cuando la hay.

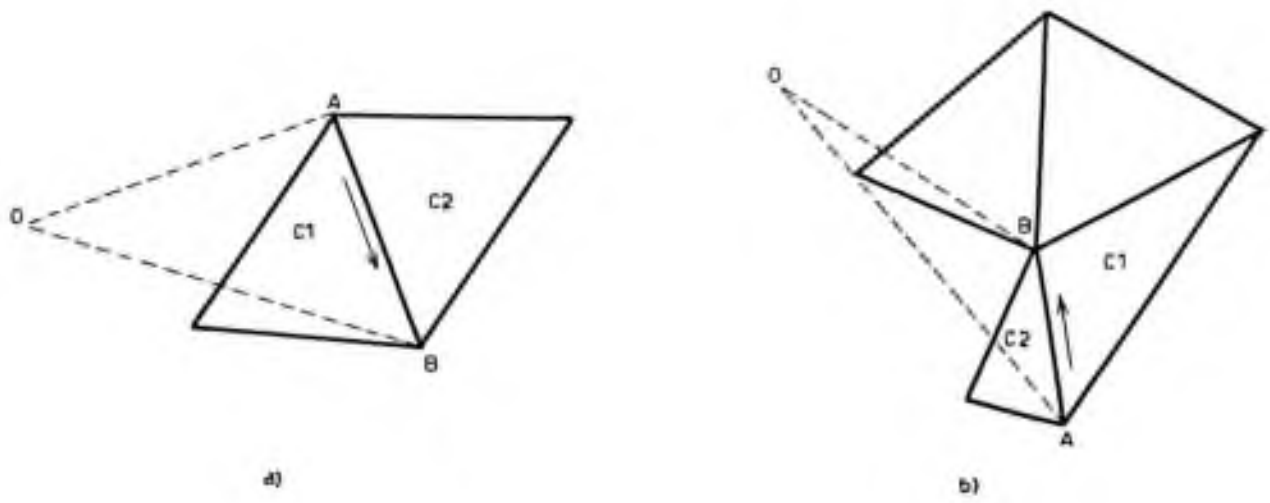


figura 1.8 Linea de Vista

ADYACENCIA

1) Dos caras son adyacentes si tienen un lado en común.

2) Un lado interno es adyacente a otro si tienen un vértice en común y pertenecen a caras adyacentes. Asi pues en la fig. 1.9.a los lados internos "AC" y "CB" de la cara C2 son adyacentes al lado interno AB de la cara C1.

3) Por último, un lado es adyacente a un vértice si presenta linea de vista a éste y además pertenece a una cara que contiene dicho vértice (figura 1.9.b.)

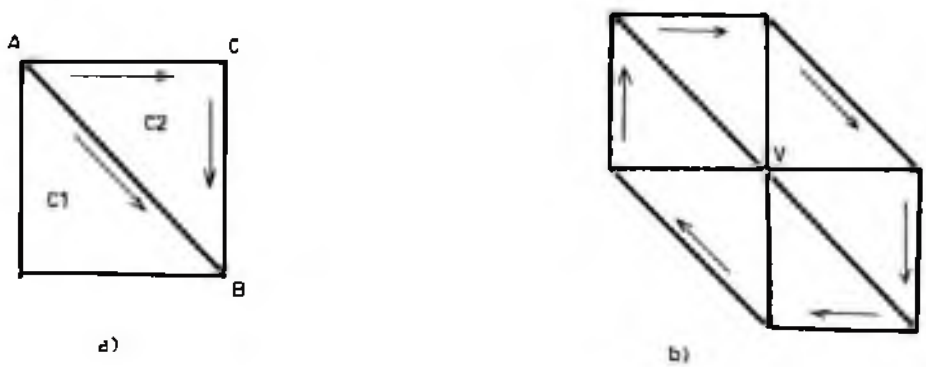


figura 1.9 Adyacencia

EXTENSION DE UNA CUÑA

Los elementos que componen una cuña fueron descritos en esta misma sección. Para extender una cuña se siguen los siguientes pasos: (fig 1.10)

1.- Se obtienen los lados adyacentes al lado "l" en que incide la base de la cuña que se desea extender .

2.- Se obtiene el desdoblamiento planar de la cara que contiene a los lados adyacentes a "l".

3.- Se prolongan las líneas que van del origen de la cuña a su base hasta intersectar con los lados adyacentes a "l". Esta intersección delimita unas nuevas bases sobre los lados adyacentes a "l".

4.- Se evaluan los atributos de las nuevas cuñas obtenidas, y se agregan a una cola de cuñas ordenada en forma creciente de la distancia al origen de la ruta buscada.

Al proceso anterior se le llama Extender una Cuña.

Al extender cuñas se pueden presentar dos tipos de evento: evento lado y evento vèrtice.

El "evento lado" sucede cuando se extiende una cuña sobre un lado "l" y èste presenta línea de vista al origen relativo de la cuña. En caso contario se presenta un evento vèrtice, es decir, se genera un nuevo origen.

En la figura 1.11, 'O' representa un Nuevo Origen. El Nuevo Origen es el vèrtice común entre la base de la cuña recièn extendida y el lado "l" que no presentó línea de vista.

Al extender una cuña a partir de la base de otra se dice que

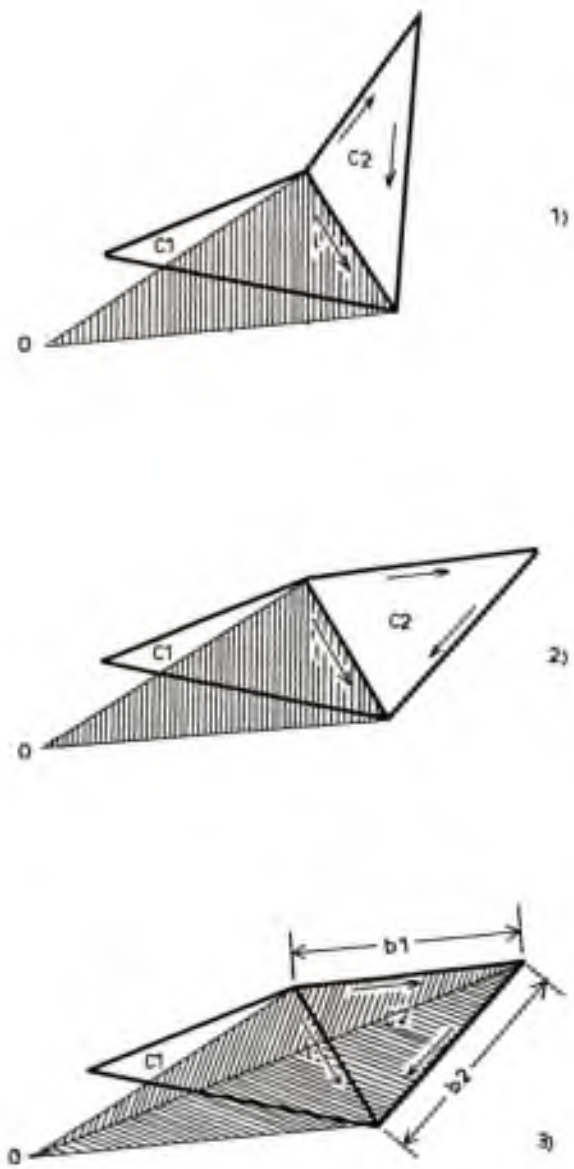


figura 1.10 Extensiôn de una Cuña.

O origen de la cuña (la cual se muestra sombreada).
C1 y C2 son dos caras de la triangulaciôn.
b1 y b2 son las bases de las cuñas generadas al extender la cuña con base en ; y origen en O.

se extiende a partir de un lado, mientras que al hacerlo a partir de un vértice se dice que se extiende un vértice. Lo anterior se implementa con los procedimientos Extiende_Lado(.) y Extiende_Vertice(.) en el programa, como se muestra: (en seudo código).

```
Extiende_Lado (Clave, Lado)

{  Clave     -  clave del lado que contiene la base de la
                cuña que se extiende.
   Lado      -  clave del lado adyacente a Clave, y sobre
                el que se desea extender                    }
BEGIN
  Obten Origen Relativo(Lado).
  Genera Nueva Cuña(Lado)
  IF Lado presenta línea de vista al origen relativo THEN
     CASE
       CASE la Nueva Cuña se Traslapa con otra
            IF Cota Inferior
               Cuña_Equivalente(Lado)
            ELSE
               Ajusta_Cuña(Lado) {método exacto}
            ENDIF

       CASE la Nueva Cuña Regresa     {llega a un lado
                                        sobre el cual ya hay
            Elimina Cuña.             una cuña, pero por
                                        la otra parte del
                                        lado}

       CASE la Nueva Cuña no tiene competencia
            Colocarla en la cola ordenada de cuñas.
     ENDCASE
  ELSE
     Genera Nuevo Origen.
     IF no se repite THEN
      Coloca el Nuevo Origen en la cola ordenada de cuñas.
     ENDIF
  ENDIF
END.
```

```
Extiende_Vertice (Clave, Lado)

{  Clave    -  clave del vértice, a partir del cual se
              extenderán cuñas.
   Lado     -  clave del lado adyacente a Clave  }
BEGIN
  Obten Origen Relativo(Lado).
  Genera Nueva Cuña(Lado)
  CASE
    CASE la Nueva Cuña se Traslapa con otra
            IF Cota Inferior
               Cuña_Equivalente(Lado)
            ELSE
               Ajusta_Cuña(Lado) {método exacto}
            ENDIF
    CASE la Nueva Cuña Regresa     {llega a un lado, sobre el
                                  cual ya hay una cuña, pero en
            Elimina Cuña.         la otra parte del lado}
    CASE la Nueva Cuña no tiene competencia
            Colocarlo en la cola de cuñas.
  ENDCASE
END.
```

Cabe hacer notar que los procedimientos anteriores no sólo se utilizan en el algoritmo de Mount, sino que también se emplean en el algoritmo propuesto en esta tesis explicado en el capítulo II. Así pues, esta sección será referida más adelante.

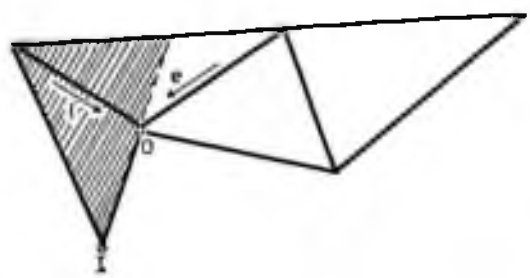


figura 1.11 Nuevo Origen

En la figura anterior el lado "e" no presenta línea de vista al origen I de la cuña que se extiende sobre el lado "l", por lo que se genera el nuevo origen O.

Cuardo se extiende una cuña genera varios eventos tras lo que muere, es decir, se extrae de la cola ordenada antes mencionada. La figura 1.12.a muestra que un evento vértice o Nuevo Origen puede extenderse hasta a 6 lados, mientras que un evento lado (figura 1.12.b) se extiende como máximo a 2.

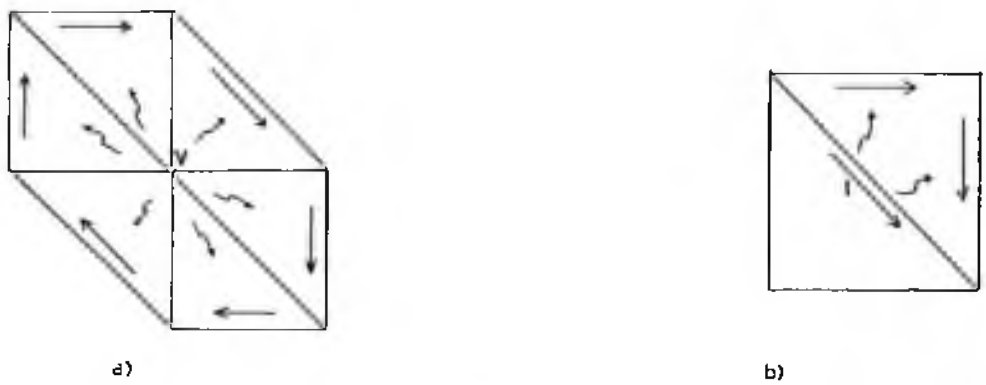


figura 1.12
a) Extender un vértice
b) Extender un lado.

1.2.2 METODO.

A continuación se describe el algoritmo de Mount.

- Mantiene actualizada una cola ordenada de cuñas que serán extendidas en orden creciente de su distancia al punto origen del camino.
- Cada vez que se extiende una cuña genera nuevas cuñas que son incluidas en la cola ordenada, la cual puede crecer indefinidamente.
- Se quita de la cola ordenada la cuña que acaba de ser extendida
- Si al generar una nueva cuña su base se traslapa con

la de otra residente en la cola ordenada, es necesario ajustar las bases de ambas cuñas. El ajuste se lleva a cabo mediante la curva bisectriz entre los orígenes relativos de las cuñas que se traslapan, delimitando las bases de estas.

- El proceso se repite hasta que la cuña que se desea extender contiene al Destino de la ruta.

1.2.3 PROBLEMA DEL CAMINO MONOTONO MAS CORTO.

LEMA 1 [M-85]

Las bases de las cuñas generadas en el problema monótono, están conectadas.

Prueba:

La prueba es por inducción en el número de lados por los que ha pasado la cuña. Una cuña degenarada cuya base iguale a su origen está trivialmente conectada. Considere una cuña cuya base reposa sobre un lado "e" de la cara "f", figura 1.13. La cuña entra a la cara "f" por alguno de los otros lados o vértices de ésta cara.

La región que delimita la intersección de la cuña con el lado por donde se entra a la cara "f" es, por inducción, un subconjunto de una base conectada. Sólo una cuña sobre "e" puede entrar a "f" a través de ésta región, ya que las bases de las cuñas no se traslapan.

Suponiendo lo contrario, que la cuña no está conectada. De los comentarios anteriores y del hecho de que hay cuñas que

cubren a "e", debe haber dos cuñas cuyos desdoblamientos planares se intersecten.

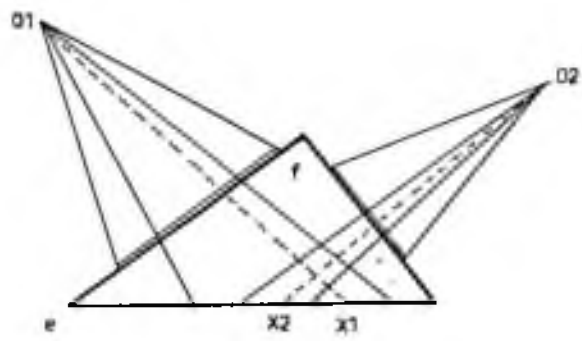


figura 1.13 Intersección de una Cuña.

Ya que "f" es convexa ésta intersección ocurre dentro de "f". Existen puntos x1 y x2 que están sobre las bases de las cuñas intersectadas cuyos orígenes son O1 y O2 respectivamente, y cuyas geodésicas más cortas intersectan en "f" con un ángulo diferente a cero. Sin embargo, esto conduce a una contradicción inmediata al construir los caminos más cortos, a través de "f", desde x1 a O2 y de x2 a O1. Esto implica que existe una geodésica más corta que llega a X cruzando "f".

LEMA 2. [M-85]

El número de cuñas incidentes sobre un lado en el problema

monótono es O(n) y por tanto el número total de cuñas es $O(n^2)$.

Considere el conjunto de cuñas incidentes sobre una parte de un lado "e". Cada origen de las cuñas está en un vértice. Al seleccionar, arbitrariamente, uno de los posibles caminos cortos a un vértice podemos considerar que el camino es único. Entonces si dos cuñas diferentes comparten un origen común, en algún lugar (entre el origen común y el lado "e") las cuñas pasan por lados diferentes de un vértice V tras haber pasado por una cara "f" incidente en V. Ningún otro par de cuñas que compartan un origen común pueden ser separadas primero por V mientras pasan sobre "f", ya que esto implicaría que dos geodésicas se cruzan en "f" con un ángulo diferente a cero. En el lema 1 se demostró que esto no puede ocurrir. De ésto resulta que el número de cuñas incidentes sobre "e" está limitado por el número de orígenes más el número de pares (V,f) (donde V es uno de los vértices de "f"). El número de orígenes esta limitado por el número de vértices y el número de pares (V,f), está limitado por dos veces el número de lados. Entonces, hay O(n) cuñas sobre "e".

1.2.4 ALGORITMO DEL CAMINO MONOTONO MAS CORTO

Se presenta el algoritmo del camino más corto considerando que los lados de la triangulación sólo pueden ser cruzados en una dirección.

Se basa en el algoritmo de Dijkstra [AHU-74].

Utiliza cuñas para almacenar la función de distancia.

El algoritmo es controlado por una cola de prioridades

ordenada por distancias. Las operaciones de insertar, extraer el de menor distancia, borrar y cambiar prioridad de un evento en la cola, se lleva a cabo en un tiempo O(log n).[AHU-74].

Se pueden distinguir dos tipos de eventos: cuando una cuña se extiende a través de un lado a otro se tiene un evento lado. Cuando una cuña arriva a un vértice se presenta un evento vértice.

Inicialmente la cola contiene un evento vértice, el origen del camino, con una distancia 0. Para cada uno de los vértices restantes se agrega en la cola un evento vértice con una distancia infinita.

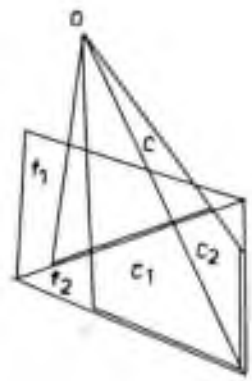
Conforme se vaya conociendo el valor de la distancia al origen de cada cuña, su prioridad se modifica.

Si se retira de la cola un evento vértice se genera una nueva cuña para cada lado de las caras que inciden en dicho vértice y que le presenten línea de vista. La base para estas nuevas cuñas es el lado mismo.

Si se retira de la cola un evento lado se genera una nueva cuña al extender la cuña actual, es decir, al prolongar los rayos que delimitan a la cuña que está en la cabeza de la cola.

Considerando que la cuña actual incide en el lado "e", sobre una cara f1, y f2 es la cara opuesta sobre "e", entonces las nuevas cuñas son generadas para los otros dos lados de f2.

La cuña actual es el ancestro de las cuñas recién generadas. Ver figura 1.14.



C es padre de C1 y C2
figura 1.14

Si se traslapa la base de una nueva cuña con la de alguna cuña ya existente en la cola, sus bases deben ser ajustadas.

AJUSTE DE CUÑAS

Cuando dos cuñas tienen sus bases sobre un mismo lado "e", se presentan dos casos:

1.- Sus bases no se traslapan.

2.- Sus bases se traslapan.

Para el primer caso, el camino más corto que llega a un punto cualquiera del lado "e" está contenido en sólo una cuña, por lo que no es necesario ajustar las bases de las cuñas incidentes.

Cuando la base de una nueva cuña se traslapa con la de una cuña ya existente sobre "e" (caso 2) éstas deben ser ajustadas para evitar el traslape, ya que un punto X contenido en la parte del lado "e" en que las bases de las cuñas se traslapan, puede tener dos caminos: uno que viene de la cuña izquierda y otro que

viene de la cuña derecha. Para definir qué puntos sobre "e" están más cerca de la cuña izquierda, que de la derecha, se procede de la siguiente manera:

-Inicialmente debe determinarse la posición relativa de la nueva cuña con respecto a las otras cuñas sobre "e".

-Se puede calcular la posición relativa de la nueva cuña en un tiempo O(log n) por bisección.

BISECCION.

Se ha visto que una nueva cuña cuya base incide sobre un lado "e" puede traslaparse con las cuñas ya existentes sobre "e". (figura 1.15.)

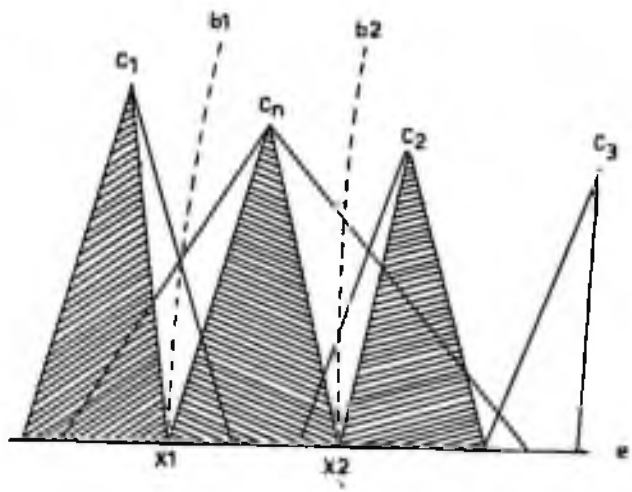


figura 1.15 Bisección

En la figura 1.15 la nueva cuña Cn se traslapa con C1, C2 y C3. C1 está a la izquierda de Cn, y C2 está a la derecha, por lo que los denominaremos Ci y Cd respectivamente.

Primero ajustaremos Cn con Ci que está a la izquierda, obteniendo el punto x1, obtenido mediante la intersección con "e" del lugar geométrico de los puntos que están a la misma distancia de Ci que de Cn (cuando Ci y Cn tienen el mismo peso, dicho lugar geométrico es una recta; en caso contrario es un segmento hiperbólico). En la figura 1.15 la linea punteada "b1" representa este lugar geométrico. Los puntos a la izquierda de x1, están más cerca del origen de Ci; los de la derecha, están más cerca del origen de Cn.

Por otro lado, cuando cada punto en la base de Ci está más cerca del origen de Cn, entonces Ci debe ser borrada. Una vez que Ci es retirada de la cola de prioridades antes mencionada, se debe considerar a la siguiente cuña hacia la izquierda como Ci y se repite el proceso de ajuste. (Fig 1.16.)

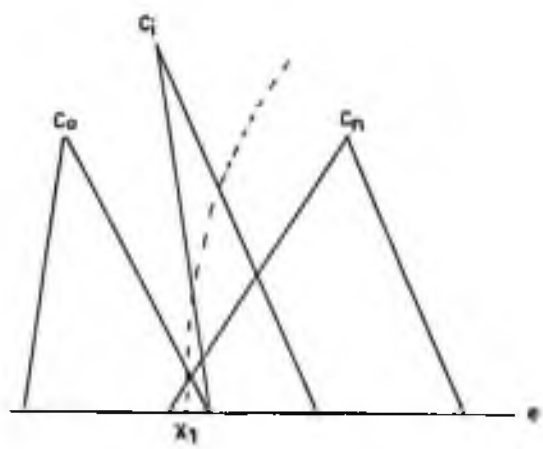


figura 1.16 Ajustando a la Izquierda.

Ci debe ser borrada porque todos los puntos de su base están más cerca del origen de Cn. Co pasa a ser Ci y se repite el proceso de ajuste.

Después se procede de igual manera con la cuña de la derecha

(Cd), obteniendose el punto x2 mediante "b2" en la figura 1.15.

Una vez que se ha posicionado la nueva cuña al ajustar su base, tanto a la izquierda como a la derecha, sobre un lado "e" se agrega a la cola ordenada de cuñas.

Si "d" es la distancia del punto más cercano de la base de la cuña a su origen y "p" el peso al Origen de la ruta que se busca, entonces la cola de cuñas será ordenada por "d"+"p".

El costo de ajustar cuñas es proporcional al número de cuñas borradas. A su vez, éste está limitado por el número de cuñas creadas, que se mostrará es de $O(n^2)$.

Modificar las cuñas de la cola requiere un tiempo de $O(n^2 \log n)$, por lo que costo total del ajuste es $O(n^2 \log n)$.

A continuación se presenta un ejemplo para ilustrar la forma en que se extienden las cuñas sobre el desdoblamiento planar de una representación de terreno dada.

1.2.5. EJEMPLO

En la figura 1.17 se muestra la forma en que se extienden las cuñas sobre una partición de terreno que ha sido previamente desdoblada. Se desea llegar al destino M a partir del origen O.

Inicialmente (figura 1.17.a) a partir del vértice O, se extiende una cuña, a la que le llamaremos AB, sobre el lado común a las caras A y B. La base de la misma cubre completamente el lado AB, por lo que se especifica que cubre de 0.0 a 7.1 del lado (7.1 es la longitud de dicho lado).

El origen de la cuña que en éste caso es O está ubicado con respecto al lado AB en las coordenadas (3.5,-3.5). El peso de

este origen relativo es 0.0, ya que es en él donde inicia el camino.

La distancia más corta a la base de la cuña es de 5.0. Y por último el ancestro es el origen mismo.

La cuña recién generada es colocada en una cola que está ordenada por la distancia al origen relativo más el peso del mismo.

Supongamos que ahora extendemos la cuña AB sobre el lado BC (figura 1.17.b). En éste caso la cuña generada tiene como base el lado BC, cuyo origen relativo se encuentra en (8.2,-4.7) con un peso de 0.0, y la distancia más corta a la base es de 5.0. El ancestro es la cuña AB. La cuña BC es ordenada en la cola de prioridades y la cuña AB es retirada de la misma (si no tiene más lados adyacentes sobre los cuales extender nuevas cuñas).

Al tratar de extender la cuña BC sobre el lado CD (fig. 1.17.c) ocurre que CD no presenta línea de vista al origen O, por lo que es necesario generar un nuevo origen (evento vértice).

El nuevo origer relativo N tiene un peso igual a 5.0 (que es la distancia de N al origen O), y por no tener base su distancia a ella es 0.0. Las coordenadas de O respecto a N son (0,-5), y tiene como ancestro a BC. El evento vértice es agregado a la cola de prioridades, mientras que la cuña BC es retirada.

A partir de N es fácil llegar al destino M, como se ilustra en la figura 1.17.d.

Cuando la base de la cuña a extenderse (la cuña que esta a la cabeza de la cola) contiene al punto destino de la Ruta se considera que el proceso ha terminado.

RANURA	BASE	ORIGEN	PESO	DIST	ANCEST
AB	0.0,7.1	(3.5,-3.5)	0.0	5.0	0
BC	0.0,7.1	(8.2,-4.7)	0.0	5.0	AB

Al extender la ranura BC, no hay línea de vista de la orilla CD con respecto al origen O. Se genera un nuevo origen N.

N	0.0,0.0	(0.0,-5.0)	5.0	0.0	BC

Al extender N se llega al destino M.

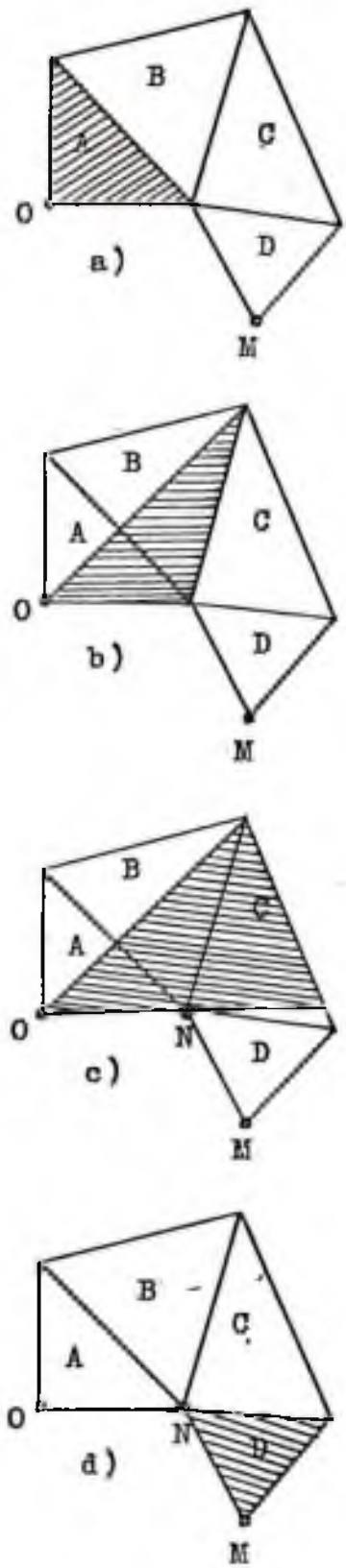


figura 1.17 Extensión de Cuñas

Nota:

Aunque una ruta de distancia mínima puede ser como la mostrada en la figura 1.18, en la cual se une el origen con el destino mediante una línea recta a través de un desdoblamiento, puede haber rutas que unan al origen con el destino mediante una línea recta sin que ésto implique que sea una ruta de distancia mínima.

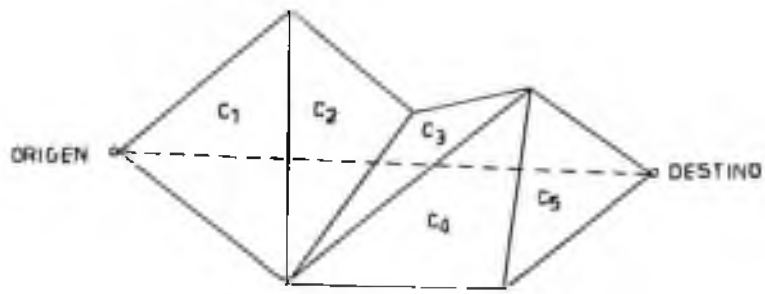


figura 1.18 Una Posible Ruta de Distancia Mínima.

Camino corto entre dos vértices
sobre un desdoblamiento planar.

1.2.6 CORRECCION DEL ALGORITMO

Lema 3 [M-85]

Sea d la distancia del siguiente evento en la cola.

1) (las cuñas corresponden a geodésicas). Si una cuña C es creada por el algoritmo y un punto X reposa sobre la base de C, entonces hay una geodésica a X desde el origen cuya distancia es el peso del origen de C más la distancia

desde X al origen de C.

2) (Las cuñas cubren todos los puntos con distancia menor o igual que d). Para un punto X, el cual está a una distancia d' < d del origen más cercano y pertenece a un lado "e" hay una cuña C, cuya base contiene a X tal que la distancia desde X al origen de C es d' y C da el desdoblamiento planar del camino más corto a X.

3) (Solo las cuñas verdaderas son extendidas). Si una cuña C ha sido extendida a través de un lado "e" y X es el punto más cercano sobre la base de C a su origen al tiempo de extensión, entonces C es el desdoblamiento planar de la geodésica más corta a X desde esta parte de "e".

4) (Los caminos más cortos a vértices son correctos hasta la distancia d). Si un vértice V ha sido etiquetado con distancia d' <= d, entonces la distancia desde el origen es d'.

PRUEBA.

La prueba es por inducción en el número de eventos extraidos de la cola, notando que los eventos son extraidos en orden creciente de distancia.

1) Proviene simplemente de que las cuñas son extendidas al desdoblar caras tal como lo son las geodésicas.

2) Considere un punto X. El camino más corto desde X hasta el origen intersecta a otro lado (o vértice) en el punto Y, que debe estar más cerca al origen. Se sabe que alguna cuña contiene a Y, y debido a que la distancia a Y

es menor que d, ésta cuña ya ha sido extendida. De lo anterior se deduce que alguna cuña contiene a X, y que por (1) ésta cuña debe corresponder a una geodésica verdadera, y ya que el camino a través de Y es hipotéticamente la geodésica más corta, la cuña extendica a través de Y debe contener X. figura 1.19.

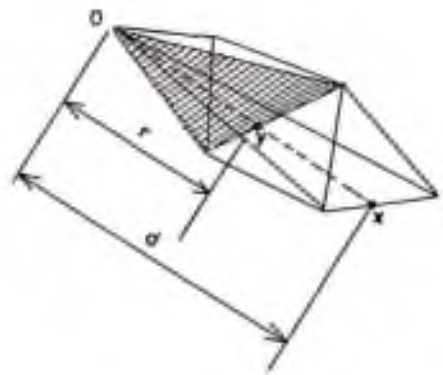


figura 1.19. Las cuñas cubren todos los puntos con distancia menor o igual que d

3) Supóngase que X, sobre el lado "e", es el punto más cercano al origen de la cuña C al tiempo de extensión. De (1), existe un camino desde X a algún origen, cuya distancia es el peso del origen más la distancia desde X a su origen. Por (2) no hay camino más corto a cualquier origen. Entonces hay por lo menos un punto de la cuña cue sobrevive el ajuste, y por tanto la cuña es permanente.

4) Se sigue de (1) y (2).

CAPITULO II. ALGORITMO PROPUESTO

En este capitulo se propone un algoritmo diseñado con vistas a disminuir el tiempo de obtención de rutas de distancia mínima.

El método de Mount, explicado en el capitulo I, demostró tener una complejidad $O(n^2 \log(n))$ donde "n" es el número de vértices en la triangulación. El presente método intenta reducir la complejidad del citado algoritmo, descartando a priori grandes regiones del terreno en donde no es factible que pase una ruta mínima.

El método aqui propuesto presupone un modelo jerárquico del terreno como el explicado en el capitulo I. Este modelo consta de una serie de "m" triangulaciones :

$$\{T_m\} < \{T_{m-1}\} < \ldots < \{T_0\}$$

Cada triangulación $\{T_i\}$ está compuesta de w_i lados, y la inclusión de una triangulación en otra $\{T_j\} < \{T_i\}$ significa que la proyección en el plano x-y de cada vértice de $\{T_j\}$, coincide ya sea con la proyección de un vértice de $\{T_i\}$ o yace en la proyección de un punto intermedio de alguno de sus lados.

Partiendo de $\{T_0\}$, una descripción sumamente gruesa de la topografía, se aplica un algoritmo del tipo "ramificación y acotamiento" sobre descripciones más y más exactas del terreno con el fin de reducir en cada paso el área a ser considerada. Así, cuando el algoritmo llega al nivel de mayor precisión $\{T_m\}$ se habrán descartado amplias regiones de terreno. En ese punto se puede usar un método exacto, como el de Mount, sobre un volumen

muy reducido de datos.

El algoritmo propuesto consta de dos etapas:

.De Reducción, en que se merman las regiones a considerarse mediante la aplicación secuencial de procesos de reducción sobre triangulaciones cada vez más finas.

.De Solución final.

Así pues, en la etapa de Reducción, se descartan regiones por las que no puede pasar una ruta mínima entre el origen y el destino. Para marcar como descartada una región de una triangulación se recurre al concepto de "barrera", esto es, un lado de una triangulación al cual es imposible que cruce una ruta mínima. La etapa de Reducción está orientada a detectar barreras. Una región será descartada sólo cuando:

.No contenga al origen y al destino
.Este rodeada por barreras

Inicialmente, en la triangulación $\{T_0\}$, las únicas barreras son los bordes de la representación.

En la figura 2.1 los lacos 4,5,6,9,10,13,14, y 16 son barreras, ya que representan el borde de la región y por tanto el camino más corto debe cruzar por los lados activos (esto es, los que no son barreras) para poder llegar al destino.

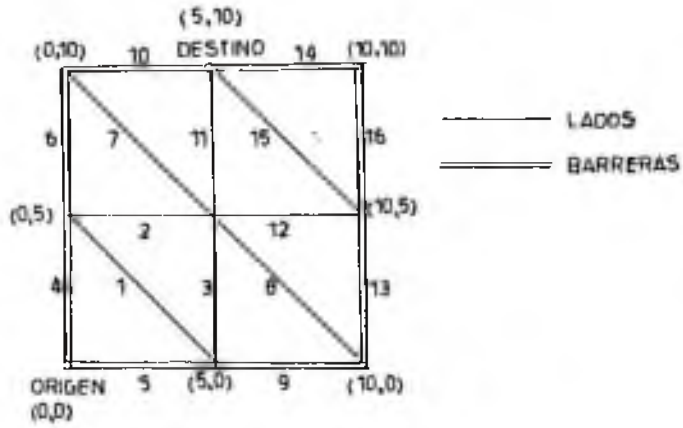


figura 2.1 Representación con Barreras

Dos características muy convenientes de la representación del terreno utilizada son:

.Solo los lados activos (esto es, los que no son barreras) necesitan ser incluidos en las triangulaciones. Todo lado no incluido se declara implícitamente como barrera.

.No es necesario determinar al principio de un procedimiento de reducción las regiones que están completamente rodeadas por barreras, ya que los lados contenidos en esta región desaparecen al terminar el procedimiento de reducción.

La organización del capítulo es la siguiente: en la sección 2.1 se hace una descripción general del algoritmo propuesto. En esa sección se discute sólamente la etapa de reducción. No se discutirá la segunda etapa, puesto que consiste en la aplicación del algoritmo de Mount que ya ha sido descrito en 1.2.

La etapa de reducción se basa en tres algoritmos principales, el de Obtención de Cotas Inferiores, el de Obtención de Cota Superior, y el de Refinamiento de una triangulación, que son descritos respectivamente en las secciones 2.2, 2.3 y 2.4

2.1 DESCRIPCION GENERAL

Como se dijo anteriormente el algoritmo tiene 2 etapas:

De Reducción, en que se eliminan regiones por donde es imposible que pase una ruta de longitud mínima.

De solución final, que encuentra la ruta de longitud mínima dentro de la región reducida anteriormente.

En este inciso será descrita la etapa de Reducción. Para ello son necesarias las siguientes definiciones:

Sean:

$\{T_t\}$ una triangulación de trabajo con w_t lados, que almacena los lados que no han sido descartados (no declarados como barreras) en el transcurso de la etapa de Reducción

o el origen de la ruta buscada.

d el destino de la ruta buscada.

x una variable definida sobre cada lado "l" de una triangulación, en tal forma que x=0 en uno de sus vértices y x=L, la longitud de "l", en el otro.

$O_l(.)$ una función definida sobre cada lado "l" de una triangulación $\{T_l\}$ que acota por abajo la distancia de cada punto "x" (sobre un lado "l") al origen de

la ruta.

$D_l(.)$ una función definida sobre cada lado "l" de una triangulación $\{T_t\}$ que acota por abajo la distancia de cada punto "x" (sobre un lado "l") al destino de la ruta.

S La distancia recorrida por una cierta ruta desde "o" hasta "d" en una ruta factible dentro de la triangulación $\{T_t\}$, y que se usará como cota superior.

L_l Min $[O_l(.) + D_l(.)]$, para el lado "l".

La obtención de las funciones $O_l(.)$ y de $D_l(.)$ se hace aplicando el procedimiento EXTIENDE(origen) o EXTIENDE(destino) respectivamente.

Obsérvese que la suma $O_l(.)+D_l(.)$ es menor que la longitud de cualquier ruta de "o" a "d" que cruza por "l". En este sentido es que se dice que son cotas inferiores.

El procedimiento EXTIENDE(.) se describe en la sección 2.2.1. La cota superior S se obtiene mediante la función RUTA(.), descrita en la sección 2.3.

Como se mencionó anteriormente, la triangulación de trabajo $\{T_t\}$ almacena la región que no ha sido descartada en la parte de reducción del terreno. Por tanto todo lado que no esté contenido en la triangulación $\{T_t\}$ es una barrera, por la que es imposible que pase la ruta de longitud mínima.

A continuación se muestra el algoritmo propuesto, en seudo código, para la parte de Reducción. Además de EXTIENDE(.) y

RUTA(.), el seudocódigo utiliza el procedimiento REFINA(.), el cual se describe en detalle en la sección 2.4. y que tiene como entrada los lados de una triangulación T_i, y da como resultado los lados correspondientes a la triangulación T_{i+1}

```
ALGORITMO REDUCCION(.);
BEGIN
1     i = 0;                          {nivel inicial = 0}
2     {T } <- {T };                     {inicialmente sólo los bordes
        t       i
                                       son barreras}
3     WHILE   (i  <  m)  DO           {m es el nivel de mejor
       BEGIN                            precisión}
                          i+3
4       S = RUTA({T },2    );       {Se obtiene una Cota Superior, con
                    t
                                        el   algoritmo   de   Mount
                                       limitando el tamaño de la
                                              i+3
                                       cola a 2    }

5       EXTIENDE( Origen);             {Aqui se obtiene la Cota
6       EXTIENDE(Destino);             Inferior  "L"  }
7       FOR EACH ({l} ∈ {T })             {Selecciona Lados}
                          t
8          IF L(l) > S THEN  {T } <- {T } - {l};
                           t       t
                                 {"l" es eliminado por ser barrera}
9        i = i+1;                          {pasa al siguiente nivel}
10       {T } <- REFINA({T });          {obtine una representación más
           t                t
                                       precisa    para    T ,    que
                                                           t
                                       corresponde   al    siguiente
                                       nivel}
       END;
END.
```

COMENTARIOS.

Instrucción 4 (RUTA(.))

La función RUTA(.) procede a obtener un camino suficientemente bueno entre el origen y el destino. El

camino así obtenido evita cruzar barreras. Esta función, que obtiene el citado camino y su longitud, se describe en 2.3. De esta ruta se utiliza solamente su longitud "S", que servirá como una cota superior a la distancia mínima.

Instrucciones 5, 6 (EXTIENDE(Origen), EXTIENDE(Destino))

La obtención de las cotas O (.), D (.) se describe en la sección 2.2.1.

Instruccion 8 (Eliminación de Barreras)

Se verifica si el lado "l" es una barrera comparando la cota inferior O (.)+D (.) con la cota superior "S".

De esta forma, al eliminar lados "barrera", se pueden descartar las porciones de terreno rodeadas por barreras, porciones por las que la ruta de longitud mínima no puede atravesar.

Los lados que pasan la prueba anterior, y que por tanto siguen "vivos", son almacenados en una cola de prioridades la cual está ordenada por la distancia al ORIGEN (ó al DESTINO) cuando se esté "extendiendo a partir del origen" ("extendiendo a partir del destino").

Instrucción 10 (REFINA(.))

Con los lados "vivos" se pasa al proceso de Refinamiento para obtener una aproximación mejor. Esto se realiza mediante el procedimiento REFINA(.), descrito en la sección 2.4.

El Refinamiento consiste en "dividir" o "refinar" nuestra última partición, para acomodarla al siguiente nivel de precisión.

A continuación se presenta un ejemplo del efecto de las instrucciones descritas anteriormente al ser aplicadas sobre el terreno de la figura 2.1.

Ejemplo:

En la figura 2.2 se ilustra un "camino suficientemente bueno" obtenido mediante la función RUTA(.). (instrucción 4)

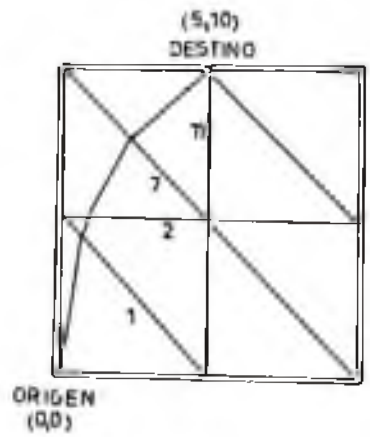


figura 2.2 Camino Suficientemente Bueno.

Suponiendo que el camino suficientemente bueno recién obtenido nos proporciona una cota superior S = 14.

El resultado de aplicar las instrucciones 5 y 6 del algoritmo al lado 1 es:

O (1) = distancia más corta desde el origen hasta 1

D (1) = distancia más corta desde el destino hasta 1

Suponiendo que se obtienen las cotas inferiores que a continuación se listan, podemos identificar las barreras.

O (1) = 3.53 en el punto 2.5,2.5

D (1) = 7.07 en el punto 0,5

L = Min(O (1)+D (1)) = 12.24

Ya que L <= S para el lado 1, éste no es una barrera.

Los lados que son barreras se muestran en la figura 2.3 con doble línea.

Al descartar las regiones rodeadas por las barreras de la figura 2.3. obtenemos la representación de la figura 2.4.

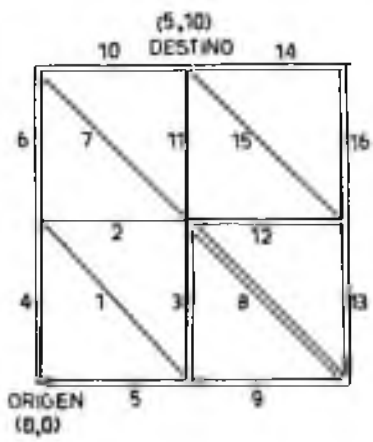


figura 2.3 Barreras en una Representación.

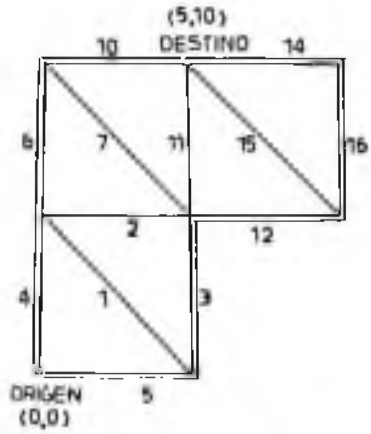


figura 2.4 Eliminación de Barreras.

En la figura anterior, los lados 3 y 12 pasan a ser "bordes" de la representación. Los bordes aunque no han sido eliminados de la representación son considerados barreras.

Una vez que se han eliminado las regiones rodeadas de barreras, esto es, si la partición resultante del proceso de "eliminación" fuese la de la figura 2.4, el resultado del proceso de "REFINA(.)" (instrucción 10) sería el que se muestra en la figura 2.5.

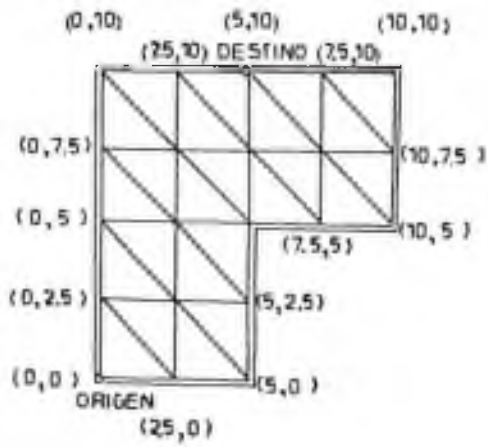


figura 2.5 Refinamiento.

2.2 OBTENCION DE LAS COTAS INFERIORES O (.) y D (.).
                                        | |

Para obtener la cota inferior a la distancia entre los puntos de un lado "l" de una triangulación y el origen (el destino), es necesario obtener una función O (.) (D (.)). Para realizar estas dos funciones se implantó el procedimiento EXTIENDE(meta). Dicho procedimiento, descrito en la siguiente sección, obtiene para cada punto de un lado de una triangulación, una cota inferior a su distancia a "meta" (donde "meta" puede ser el origen o el destino de la ruta buscada). Así, O (.) y D (.), son el resultado de aplicar los procedimientos
| |
EXTIENDE(origen) y EXTIENDE(destino) respectivamente.

La filosofía de este procedimiento se basa en la usada en el algoritmo de Mount, y al igual que éste trabaja con cuñas.

También como en el algoritmo de Mount, se considera que cada lado "l" de la triangulación puede ser considerado como formado por dos copias ("ld" y "li") del mismo. La copia "ld" es la que presenta "l" al ser cruzado de derecha a izquierda, y "li" es la copia presentada por "l" al ser cruzado de izquierda a derecha. Lo anterior fué descrito con más detalle en la sección 1.2.1.

En el algoritmo de Mount se podían extender (en diversos momentos) hasta "n" cuñas sobre un lado "l", donde "n" es el número de puntos de la triangulación. En el algoritmo EXTIENDE(meta) sólo se puede expandir una vez una cuña para cada lado. Para lograr esto, una vez que se han obtenido diversas cotas inferiores a las distancias de un lado a "meta", se obtiene una "cuña mínima subtendiente" o "cuña equivalente", que acota por abajo a todas las posibles cotas ya existentes en el lado

"l".

2.2.1 DESCRIPCION DEL PROCESO EXTIENDE.

EXTIENDE(meta), obtiene para cada lado "l" de una triangulación {T} una cota inferior de su distancia a "meta". Esta cota inferior es una función del parámetro "x" definido en 2.1, y es de la forma:

$$O(x) = \sqrt{(h - i(x))^2 + (k - j(x))^2} + A$$

donde :

(i(x),j(x)) son las coordenadas de un punto "x" sobre "l".

(h,k) son las coordenadas del origen relativo de la cuña que incide sobre "l".

A es el peso del origen relativo de la cuña.

Esta fórmula es la distancia al origen de los puntos "x" de la base de una cuña, la cual incide sobre el lado "l". Para obtener recursivamente estas cotas inferiores se usa también un mètodo que utiliza cuñas y propagaciones. Difiere del de Mount en que antes de que se propague la cota inferior de un lado:

.Todos los puntos de ese lado deber tener cotas inferiores vàlidas.(Que no puedan ser modificadas).

.Se obtenga una cuña subtendiente que, sin gran desperdicio, acote por abajo a las cotas inferiores ya existentes en ese lado.

Además, las cuñas que se usan en este algoritmo, difieren de las usadas por Mount, en que sus bases cubren lados completos de

la triangulación. (Fig 2.6)

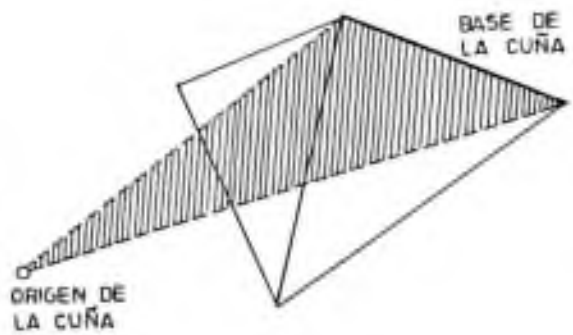


figura 2.6 Cuñas que cubren lados completos.

Al igual que el algoritmo de Mount se mantiene actualizada una cola de prioridades, ordenada en orden creciente de la distancia. Al inicio de esta cola se haya la cuña de menor distancia a la "meta" (el Origen o el Destino de la ruta).

Antes de pasar a la especificación del algoritmo "EXTIENDE(meta)", son necesarias ciertas definiciones:

"l"	el lado de la triangulación, en el que yace la cuña que está a la cabeza de la cola.
d	La distancia mínima desde la cuña, que está a la cabeza de la cola, hasta "meta".
{C}	El conjunto de cuñas residentes en la cola de prioridades que inciden en "l".
D	La máxima distancia a "meta " entre las cuñas de {C}.
{P}	El conjunto de cuñas que irciden en lados cuya

proyección en el plano x-y, se encuentra a una distancia menor que D-d.

{R} Conjunto de cuñas recién generadas y que aún no se incluyen en la cola de prioridades.

A continuación se describe el algoritmo EXTIENDE(.) en seudo código. Este algoritmo necesita de los siguientes procedimientos:

```
Vecinos(.)
Expande(.)
Iteración(.)
Cuña_subtendiente(.)
Agrega_a_Cola(.)
Origen(.)
```

COMENTARIOS:

. VECINOS(elemento), obtiene los lados vecinos a un elemento. Por elemento se entiende, un vértice o un lado. Se describe en la sección 2.2.1.a.

. EXPANDE(vertice), expande cuñas a partir de un vértice. Este procedimiento se describe en la sección 2.2.1.b.

. ITERACION({P},{C}), verifica que se incluyan en el conjunto {C}, todas las cuñas de {P} que pueden afectar la Cota Inferior. Se describe en la sección 2.2.1.c.

. CUNA_SUBTENDIENTE({C}), obtiene a partir de las cuñas en {C} una única Cuña Equivalente, se describe en más detalle en la sección 2.2.1.d.

. AGREGA_A_COLA({R}), inserta en la cola de prioridades las cuñas {R} generadas en el procedimiento EXPANDE(.).

- ORIGEN(ñ), nos proporciona el origen de la cuña ñ.

En seudo código:

```
EXTIENDE(meta);
BEGIN
1 {E} <- VECINOS(meta);
2 {R} <- EXPANDE(meta, {E});
3 AGREGA_A_COLA({R});
WHILE (cola no vacia) DO
 BEGIN
4   toma elemento superior de la cola. Obten {C}, D, d, u=D-d,{P}
  /* propagamos los efectos de P a C*/
5  {C} <- ITERACION({P},{C});
6  ñ = CUÑA-SUBTENDIENTE({C});
7  {E} = VECINOS(l)
8  {R} <-EXPANDE(origen(ñ),{E})
9  AGREGA_A_COLA({R});
 END;
END;
```

Lema 4

"La cota obtenida en el proceso EXTIENDE(.) acota por debajo a la longitud de la Ruta Mínima".

Prueba.

Por inducción, en que una cuña "r" que incide sobre un lado "l" no se extenderá hasta que se haya verificado que ninguna otra cuña, que arrive al lado "l", lo haga en forma más corta que "r". Cuando un conjunto de cuñas {P} incide sobre "l", cada una de ellas lo hace recorriendo un camino más corto a "l", siendo generalmente este camino diferente para cada cuña. De esta forma, cada cuña en {P} proporciona una cota de la distancia a "l" (la longitud del camino), algunas de estas cotas son mejores que otras. Para obtener una buena cota inferior se obtiene una "Cuña Equivalente" (sección 2.2.1.d), la cual "substituye" a

todas las anteriores proporcionando, además, una cota inferior a "l" que es menor a cualquiera de las cotas de las cuñas en {P}.

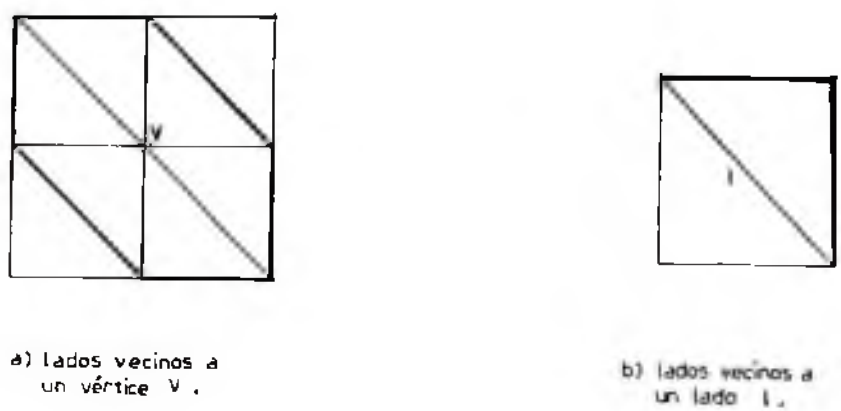
2.2.1.a PROCEDIMIENTO VECINOS

Obtiene los lados vecinos a un "elemento" que puede ser un vértice o un lado.

Lo anterior se lleva a cabo al obtener los lados "adyacentes" al elemento. Para que un lado sea adyacente es necesario que presente "línea de vista" al origen de la cuña que se desea expandir.

El procedimiento LINEA_DE_VISTA(.) fué descrito en la sección 1.2.1. En el caso de que el elemento sea un vértice, y debido a la triangulación elegida para representar el terreno, se pueden tener hasta 6 lados adyacentes. Si el elemento es un lado, se puede tener como máximo 2 lados adyacentes.

Lo anterior fué descrito tambien en el capítulo I. (Fig 2.7)



a) lados vecinos a un vértice V.

b) lados vecinos a un lado l.

figura 2.7 Lados Vecinos.

2.2.1.b PROCEDIMIENTO EXPANDE

"Expandir una cuña" es prolongar los rayos que unen al origen de la misma con su base, hasta que estos inciden en un lado adyacente sobre un "desdoblamiento planar". El "desdoblamiento planar" fue descrito en la sección 1.2.1.

Como se mencionó en la sección 2.2.1, las cuñas en la parte de Reducción deben cubrir con sus bases lados completos de la triangulación.

Cuando se expande una cuña pueden darse los siguientes casos:

. Que su base presente línea de vista.
. Que su base no presente línea de vista.

El primer caso se ilustra en la figura 2.8, en donde la cuña con origen en "vertice" se expande pasando sobre el lado "1" hasta sus lados adyacentes "2" y "3".

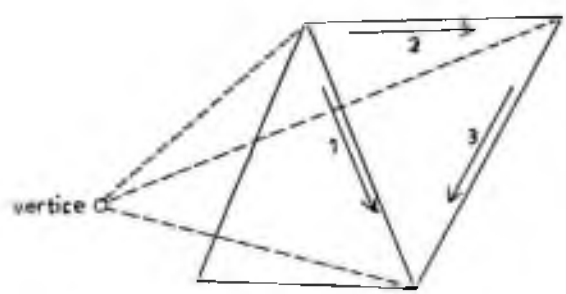


figura 2.8 Cuña que se expande sobre de un lado.

El segundo caso se muestra en la figura 2.9, en la que al expandir sobre el lado "l" una cuña con origen en "vertice" no se tiene línea de vista sobre "c". En este caso se genera "O", un "nuevo origen relativo". Este fué descrito en la sección 1.2.1.

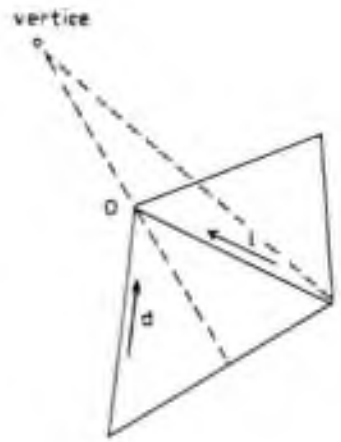


figura 2.9 Nuevo Origen.

La expansión de una cuña se implanta mediante el procecimiento EXPANDE(vertice,{A}), para lo cual son necesarias las siguientes definiciones:

Entrada: vertice Es el origen de la cuña a partir del cual ésta se extenderá.

{A} Conjunto de lados adyacentes a "vertice"

Salida : {R} Conjunto de "Nuevas cuñas" listas para ser agregadas a la cola de prioridades.

Este proceso se implementó meciante el siguiente procecimiento, expresado en seudo código:

```
EXPANDE(vertice,{A});
BEGIN
  FOR CADA lado {l}∈{A} DO
    IF Presenta_Linea_de_Vista({l}) THEN {R} <- Genera_Cuña({l})
                                    ELSE {R} <- Nuevo_Origen({l});
  RETURN({R});
END;
```

COMENTARIOS:

En los procedimientos Genera_Cuña y Nuevo_Origen se obtienen los datos del evento generado, es decir, los atributos de la cuña.

Cuando se genera un nuevo origen se dice que se genera un evento vértice; cuando se crea una nueva cuña se dice que se genera un evento lado.

2.2.1.c PROCEDIMIENTO ITERACION

ITERACION(.) incrementa a {C} si fuese necesario, para incluir en él todas las cuñas que pueden llegar a "l" y afectar su cota inferior.

Entrada: {C} Conjunto de cuñas que se hayan sobre un mismo lado "l". La cuña más cercana está a una distancia "d" y la más lejana a una distancia "D".

{P} Conjunto de cuñas que estan a una distancia D-d del lado "l".

Salida: {C} Con todas las posibles modificaciones provocadas por {P}.

Es necesario observar cómo afectan al conjunto {C} todas las cuñas que se encuentren a una distancia de "l" igual o menor que D-d. Cualquiera de estas cuñas, que forman el conjunto {P}, deben pasar a formar parte de {C} si al ser extendida modifica la cota inferior. Lo anterior se realiza en el procedimiento ITERACION({C}).

Una vez que se ha verificado que nada más puede afectar a {C}, se procede a obtener la "Cuña Subtendiente", la cual acota por abajo a todas las cuñas en {C}. El proceso de "Cuña Minima Subtendiente" se describe en la sección 2.2.1.d.

A continuación se presenta el seudo código del algoritmo:

```
ITERACION({P},{C});
BEGIN
 l= lado en  que esta {C};
 copia en una cola todos los miembros de {P},{C};
 sin_cambio = 0
 WHILE( sin_cambio <= cardinalidad de la cola) DO
  BEGIN
   q <- cuña de la cabeza de la cola;
   {E} <- VECINOS(l);
   {R} <- EXPANDE(origen(q),{E});
   IF (hubo cambio) THEN  {C} <- {R}; sin_cambio = 0  ELSE sin_cambio++;
  END;
 Return(conjunto de cuñas en el lado l)
END;
```

COMENTARIOS:

Los procedimientos VECINOS(.) y EXPANDE(.) fueron descritos en las secciones 2.2.1.a y 2.2.1.b respectivamente.

2.2.1.d CUNA MINIMA SUBTENDIENTE

Como se mencionó anteriormente en 2.2.1, las cuñas generadas en esta parte del algoritmo tienen como base a lados completos de

la triangulación.

Cuanco las cuñas C1, C2,..Cn, se traslapan sobre un lado "l", el proceso de Cuña Subtendiente obtiene una cuña "ñ" tal que "subtienda" o acote por abajo a C1, C2,..,Cn. "ñ" substituirá posteriormente a C1, C2,...,Cn.

Cuando las bases de dos o más cuñas inciden sobre un mismo lado "l" durante los procedimientos de "extender hacia adelante o hacia atrás", debe obtenerse una Cuña Mínima Subtendiente que substituya a las anteriores. Esto es con el fin de tener sólo una cuña por lado.

El origen equivalente Oe de ésta nueva cuña es obtenido de la manera siguiente: (fig 2.10)

Si la curva bisectriz de O1 y O2 (los orígenes relativos de las cuñas C1 y C2) no intersecta al lado "l" implica que uno de los orígenes está siempre más cerca de "l" que el otro, y este se elige como origen de la nueva cuña mínima subtendiente "ñ". (fig 2.10.a.)

En caso de que la curva bisectriz corte a "l", se procede a calcular el origen equivalente de la siguiente forma:

-Se traza un radio r1 a partir del Inicio del lado al origen relativo más cercano.

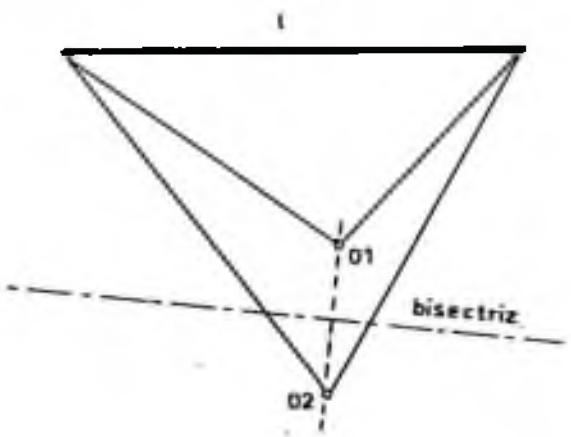
-Se traza un radio r2 a partir del Final del lado al otro origen rélativo (fig 2.10.b).

-La posición del nuevo origen relativo (origen equivalente) de la cuña "ñ" está dada por la intersección de las circunferencias de radio r1 y r2, y con centros en Inicio y Final respectivamente.

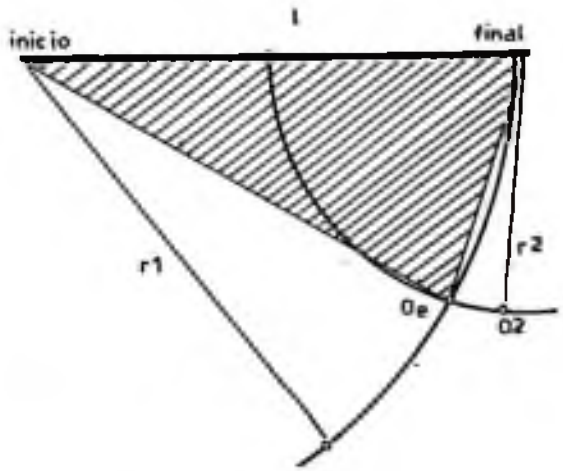
-La cuña "ñ" hereda la historia de cualquiera de las cuñas (C1 o C2) que esté más cercana al punto ORIGEN de la ruta buscada.

Como se aprecia en la figura 2.10.c, el peso del origen

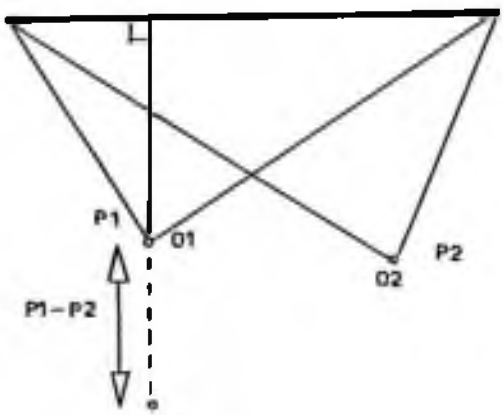
relativo también es involucrado para obtener el Origen Equivalente. En esta figura, el peso P1 es mayor que el peso P2.



a) La bisectriz no intersecta con L.



b) Origen Equivalente Oe.



c) P1 es mayor que P2.

figura 2.10 Cuña Mínima Subtendiente

2.2.2 SELECCION DE LADOS

Se procede ahora a seleccionar los lados que sobrevivirán la prueba de acotamiento. Esto se efectúa comparando la suma de los campos Dorigen y Ddestino de la estructura de datos descrita en 1.2.1 contra la Cota Superior "S", cuya obtención se describe en la sección 2.3. Así pues, los lados que cumplen con:

$$MIN(Dorigen + Ddestino) <= S$$

sobreviven para ser refinados a una partición más pequeña; los que no cumplen son considerados "barreras" y sirven para descartar regiones.

2.3 COTA SUPERIOR

2.3.1 OBTENCION DE UNA RUTA "BUENA"

El problema aquí planteado es el de encontrar una ruta que pase sólo por zonas factibles y que tenga una longitud razonablemente pequeña, para usar esta longitud como cota superior.

El procedimiento seguido utiliza una versión del algoritmo de Mount que limita el número de cuñas residentes en la cola de prioridades. . En esta versión, a diferencia de la original, el número de cuñas que pueden residir en la cola es una función de la precisión de la triangulación.

Tras de varias pruebas, se encontró que al incrementar el tamaño de la cola en forma exponencia , de acuerdo a la de finura de la

triangulación daba resultados satisfactorios. En concreto, para los ejemplos utilizados en esta tesis, y para un nivel "i", se utilizó un tamaño de la cola igual a 2^{i+3}. Así, para el nivel i=0, por ejemplo, el tamaño máximo de la cola es 8.

El propósito de reducir el tamaño de la cola es el disminuir el tiempo requerido para obtener la ruta estimada. La heurística de limitar el tamaño de la cola se basa en que generalmente las opciones "malas", a diferencia de las "buenas", permanecen mucho tiempo en la cola. Desgraciadamente, esto no siempre es cierto : algunas veces, la opción óptima puede estar "sumergida" por bastante tiempo. Así, dictar un tope en el tamaño de la cola limita también el número de opciones que pueden ser consideradas, pudiéndose descartar, entre otras, a la solución óptima, con la consiguiente pérdida de exactitud en el cálculo de la Ruta "Buena".

Si se selecciona juiciosamente el tamaño de la cola esta simplificación no es muy perjudicial, ya que el objetivo de este algoritmo es el de acotar por arriba la distancia mínima entre origen y destino para utilizar esta cota, en combinación con cotas inferiores a las distancias a esos dos puntos, en la eliminación de regiones. Tener una precisión extrema en la cota superior no produce necesariamente disminuciones extra en las áreas descartadas.

Lo descrito anteriormente se implementó en la función RUTA(tamcola):real a la cual se le pasa el parámetro "tamcola" (tamaño de la cola) que limita el número de cuñas que pueden ser extendidas y regresa la longitud de la Ruta "Buena".

A continuación se describe el procedimiento RUTA(tamcola):

Entrada: tamcola el tamaño máximo que puede tomar la cola.

Salida: Proporciona como salida la ruta y la distancia de la misma.

Algunas características del procedimiento son:

- Mantiene actualizada una cola ordenada de cuñas que serán extendidas en orden creciente de su distancia al punto origen del camino.

- La cuña a la cabeza de la cola se extiende, tras lo cual es retirada de la cola, disminuyendo su tamaño. Aún cuando al extender una cuña varias nuevas cuñas pueden ser admitidas en la cola incrementando su tamaño, la longitud de ésta no puede sobrepasar a "tamcola".

- Cuando la cola ha crecido hasta "tamcola" se elimina el último elemento de la misma.

- Si al generar una nueva cuña su base se traslapa con la de otra ya existente es necesario hacer un ajuste de las mismas. El ajuste se lleva a cabo mediante la curva bisectriz entre los orígenes relativos de las cuñas que se traslapan, delimitando las bases de estas.

- El proceso se repite hasta que la cuña que se va a extender contenga al punto destino de la ruta buscada.

Para describir RUTA(.) son necesarios los siguientes procedimientos:

VECINOS(.)
EXPANDE(.)
AGREGA_A_COLA(.)

EXTIENDE_VERTICE(.)
EXTIENDE_LADO(.)

A continuación se describe el algoritmo de obtención de la ruta estimada en seudo código:

Sea:

{A} conjunto de cuñas de la cola.
{R} conjunto de cuñas generadas.
{E} conjunto de vecinos a un vértice o a un lado.

```
funcion Ruta(tamcola) : REAL;
BEGIN
1 {E}  <- VECINOS(origen);          /*  Obtiene los vecinos del
                                        origen*/
2 {R}  <- EXPANDE(origen,{E});      /*  Genera Cuñas a partir del
                                        origen*/
3 AGREGA_A_COLA({R});               /* Se incrementa el tamaño de la
                                    cola */
 WHILE (no llegues al Destino) DO
   BEGIN
4     elemento <- TOMA_CABEZA_DE_COLA; /* Se decrementa la cola */
5     {E} <- VECINOS(elemento);
6     IF elemento = vertice THEN  {R} <- Extiende_Vertice(elemento)
7     ELSE  {R} <- Extiende_Lado(elemento);
8     AGREGA_A_COLA({R});
   END;
 RETURN (Ruta y Distancia del camino corto);
END;
```

Esta programación del algoritmo RUTA(.) permite utilizarlo también para obtener la distancia mínima mediante el metodo de Mount (etapa de Solución final).

COMENTARIOS:

Los procedimientos VECINOS(.) y EXPANDE(.) fueron descritos en las secciones 2.2.1.a y 2.2.1.b respectivamente.

Los procedimientos EXTIENDE_VERTICE(.) y EXTIENDE_LADO(.) fueron descritos en la sección 1.2.1. En estos dos procedimientos se efectúa el "ajuste de cuñas", también descrito

en el capitulo I.

En la figura 2.11 se presenta: a) Ruta "Buena" y b) la Ruta de Longitud Minima cuando se desea un camino entre los puntos (0,0) y (10,5).

Se puede observar en la figura 2.11 que la Ruta "Buena" es mayor que la Ruta de Longitud Minima.

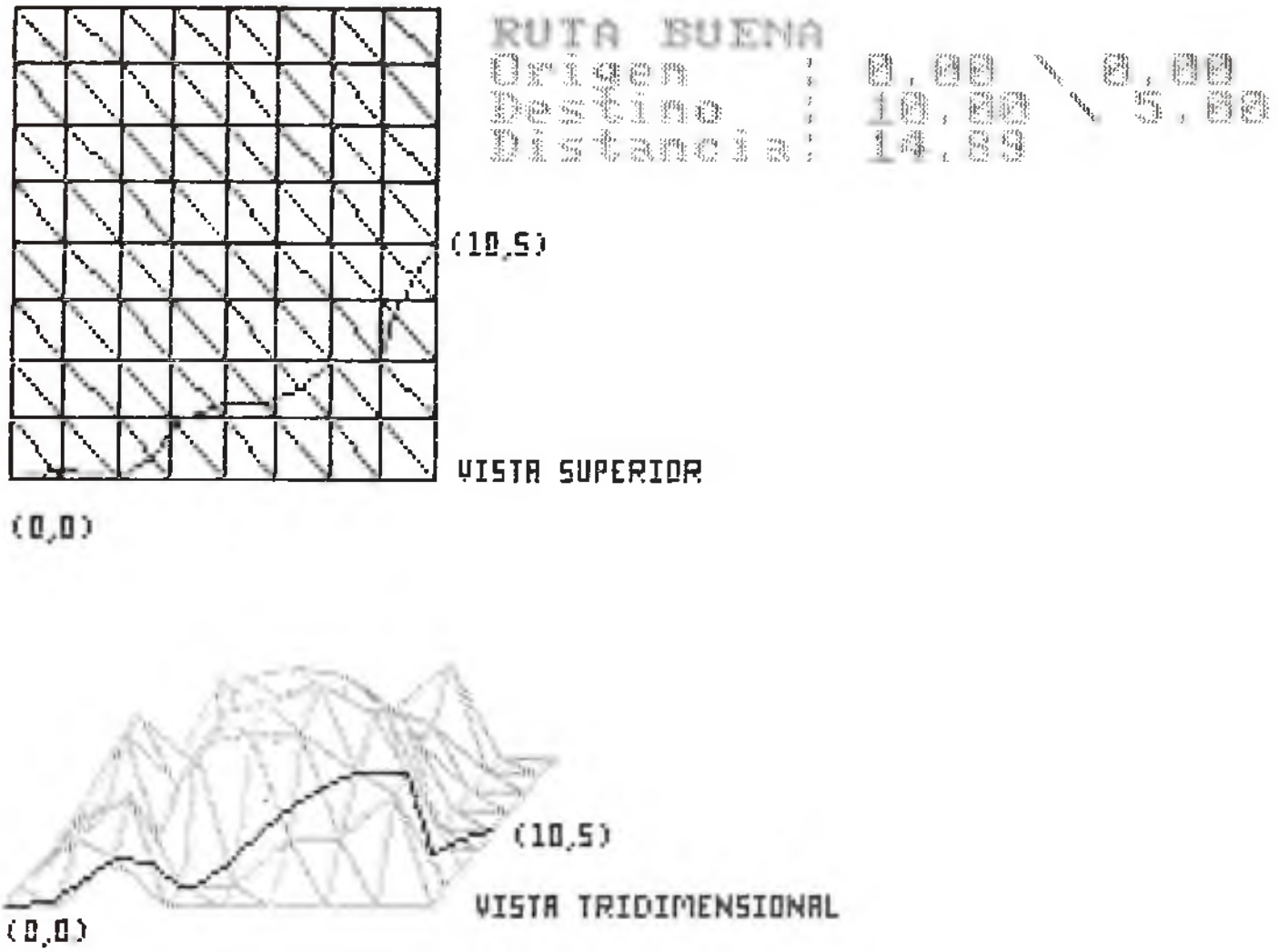


figura 2.11.a Ruta "Buena"

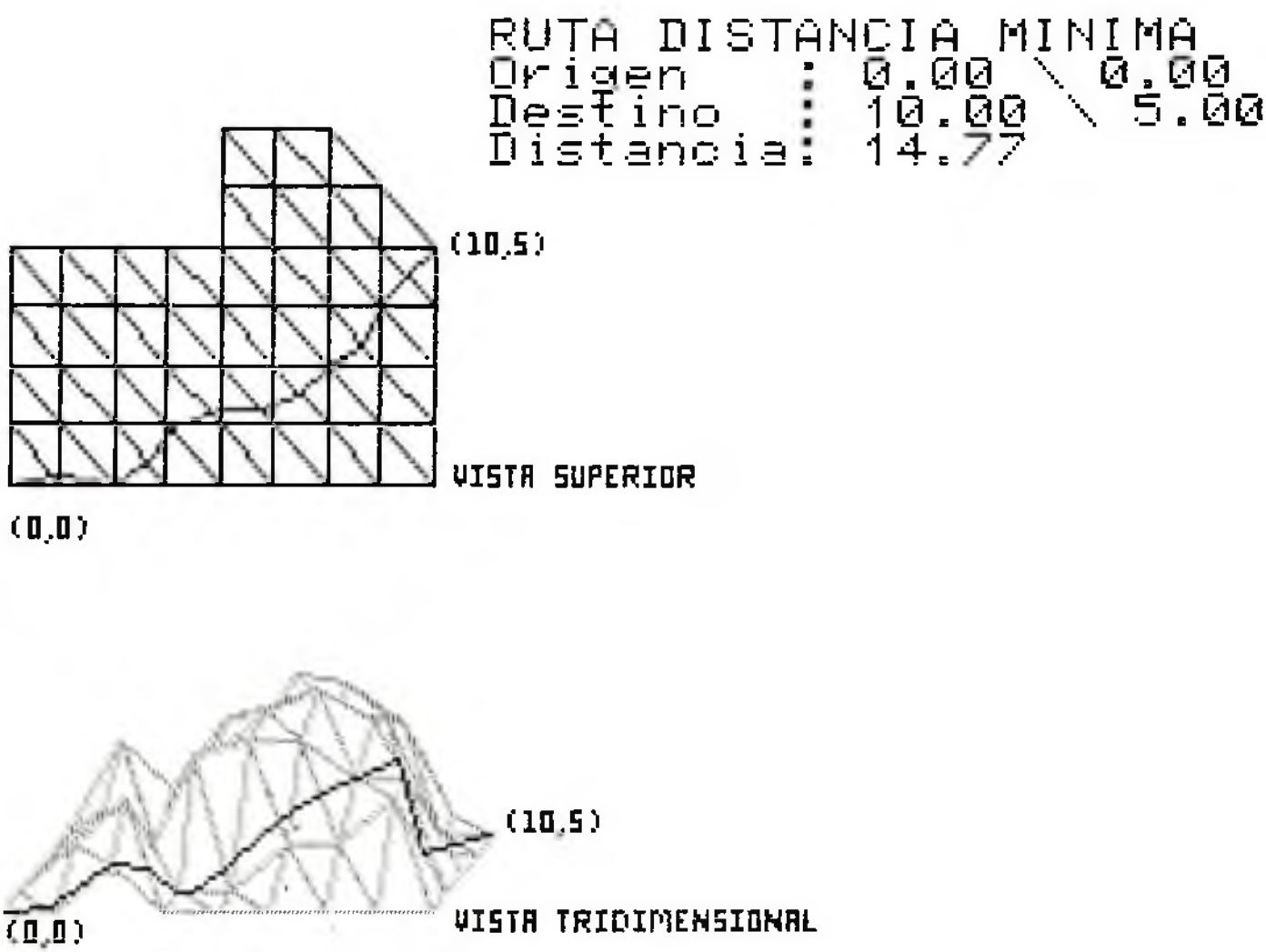


figura 2.11.b Ruta de Longitud Minima

2.4 REFINAMIENTO

Una triangulación más fina se obtiene subdividiendo los lados sobrevivientes al proceso de reducción.

El refinamiento se lleva a cabo de la siguiente manera: (en seudo código)

Sea:

{L} el conjunto de lados de una triangulación.

{T } triangulación de trabajo.
 t

```
REFINA({L});
BEGIN
  {j} <- Divide/2({L});
                        /*En {j} se tienen los lados que son creados
                          al dividir los lados de {L} en dos.*/

  {k}  <- genera_lados({j});
                            /*Con los lados de {j}, se obtienen los
                              que hacen falta para completar la
                              triangulación.*/

  {Tt} <- {j} + {k};
                          /*Los   lados nuevos son almacenados en la
                            triangulación    de    trabajo    para
                            subsecuentes procesos*/
END;
```

A continuación se muestra un ejemplo:

En la figura 2.12.a se muestra la partición original {L}, en la cual se tienen 13 lados. Sea {j} el conjunto de lados que son generados al dividir en dos cada lado de {L}. En la figura 2.12.b se ilustra la misma partición después de encontrar los puntos intermedios de cada lado. Encontrar estos puntos no es

suficiente, como se muestra en fig 2.12.b, y es necesario obtener los nuevos lados que completen la triangulación (Fig 2.12.c)

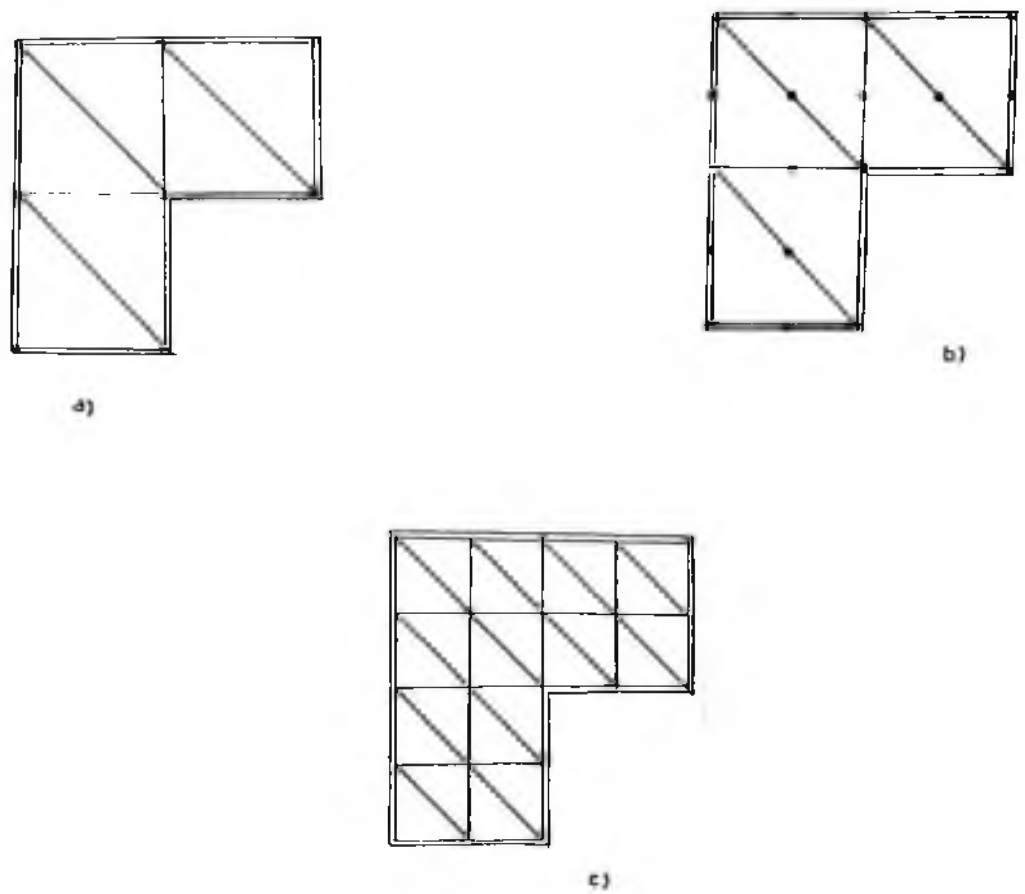


figura 2.12 Triangulación Refinada.

CAPITULO III. RESULTADOS

En este capítulo se muestran los resultados obtenidos con el algoritmo descrito en el capítulo II.

.En la sección 3.1 se presenta el terreno utilizado.
.En la sección 3.2 se muestran los resultados y algunas comparaciones.

3.1 Terreno Utilizado.

Las pruebas que se muestran en este capítulo fueron realizadas sobre el terreno que se ilustra en la figura 3.1.

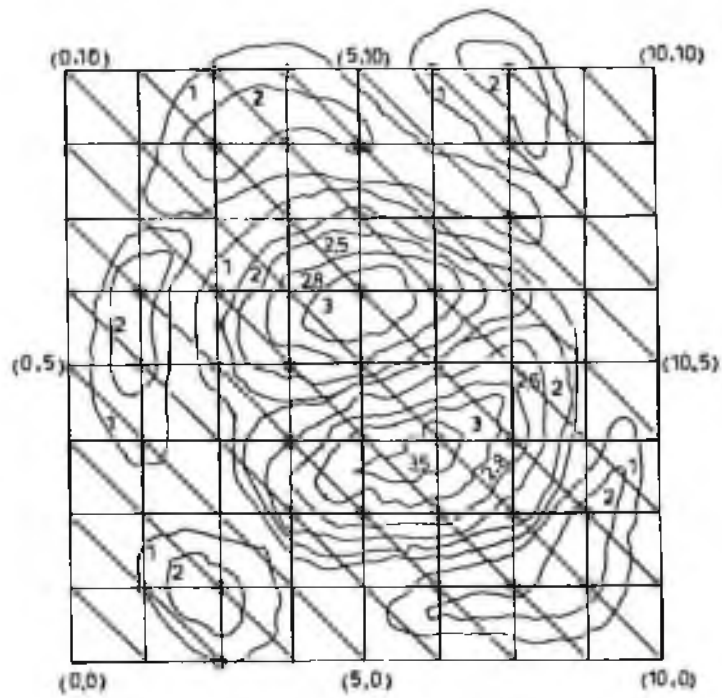


figura 3.1 Terreno de Pruebas.

Para representar dicho terreno se tomaron tres niveles de precisión. Las representaciones obtenidas para cada nivel son

las mostradas en la figura 3.2.

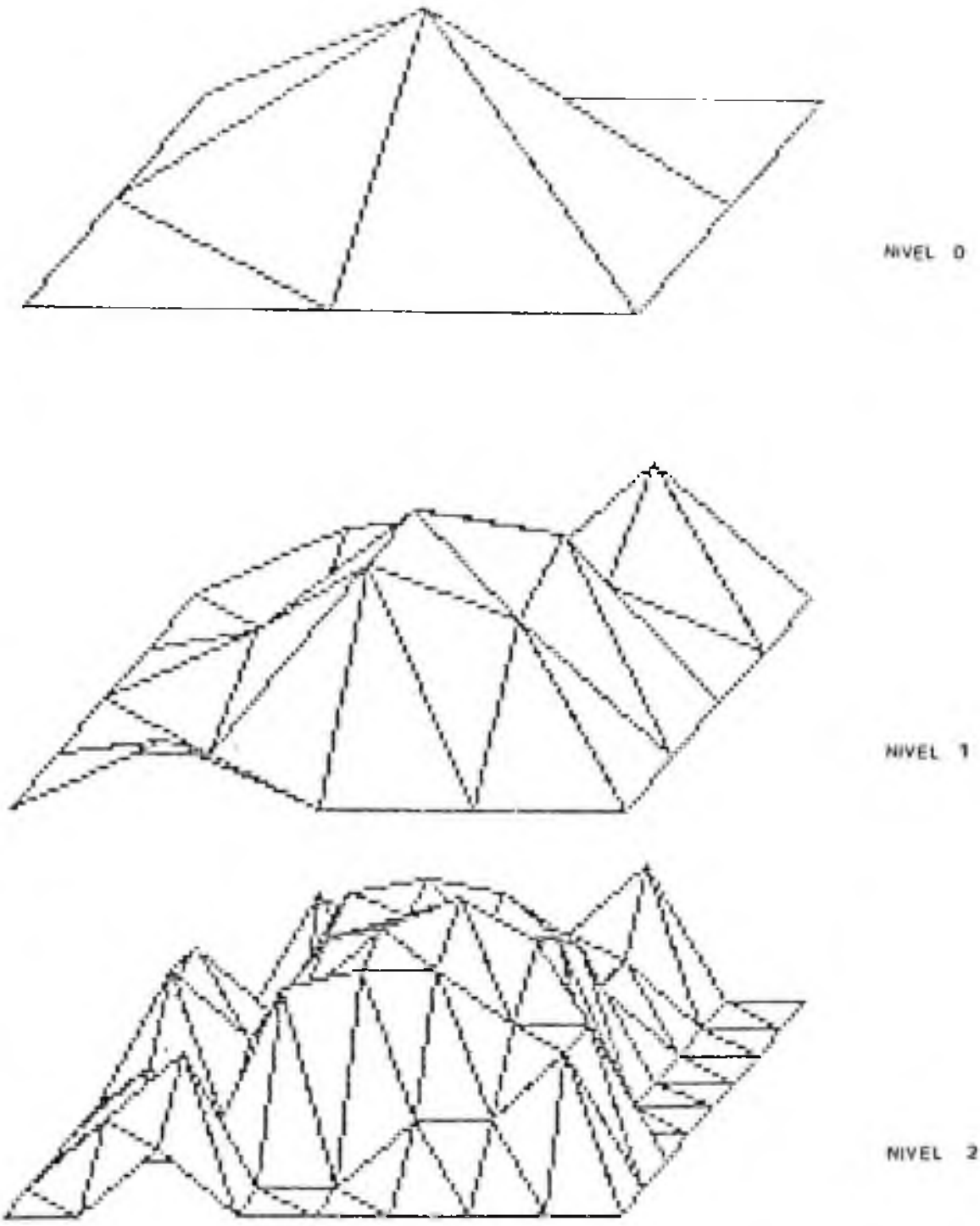


figura 3.2 Niveles de Precisión

Se puede apreciar en la fig 3.2 que conforme se tienen más niveles de precisión nuestra representación se acerca más a la realidad.

En la siguiente sección se muestran las pruebas realizadas.

3.2 Pruebas

La presentación gráfica de los resultados del algoritmo consta de :

- Una vista superior del terreno que muestra la ruta de distancia mínima entre el origen y el destino.
- Una vista en un marco de referencia tridimensional, incluyendo la ruta obtenida y
- Las coordenadas del Origen y Destino así como la distancia del camino obtenido entre dichos puntos.

Las pruebas realizadas se dividen en dos partes:

. Las que utilizan Reducción de regiones.
. Las que no usan Reducción de regiones.

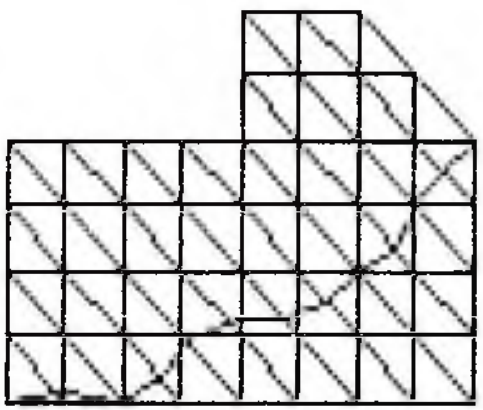
Los resultados son presentados en el siguiente orden, para un Origen Y Destino dados:

a) Resultado aplicando algoritmo de Reducción desde el nivel de menor precisión hasta el de mejor precisión. Se muestra la región final, con la ruta de longitud mínima.

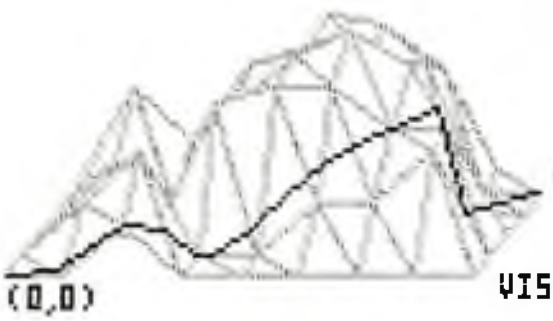
b) Resultado sin Reducción de terreno, aplicando el algoritmo de Mount sobre la representación de mejor precisión. Se muestra la región completa (sin reducciones) y la ruta de longitud mínima.

c) Comparación de los incisos a) y b).

RUTA DISTANCIA MINIMA
Origen : 0.00 \ 0.00
Destino : 10.00 \ 5.00
Distancia: 14.77



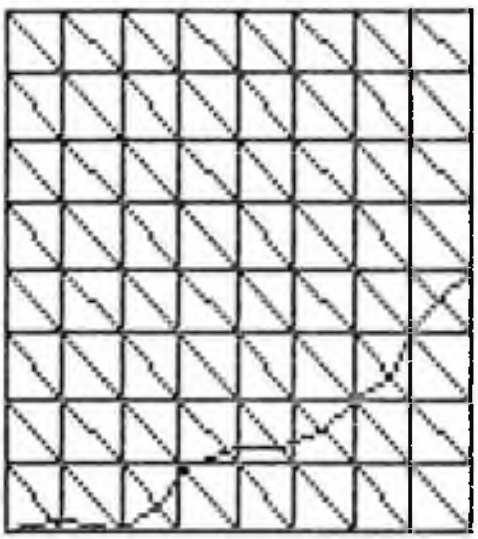
VISTA SUPERIOR



VISTA TRIDIMENSIONAL

Figura e.1.a
RUTA DE DISTANCIA MINIMA
del Origen (0,0)
al Destino (10,5)
Aplicando reducciones al terren

figura Ejemplo 1 a)



RUTA DISTANCIA MINIMA
Origen : 0.00 \ 0.00
Destino : 10.00 \ 5.00
Distancia: 14.77

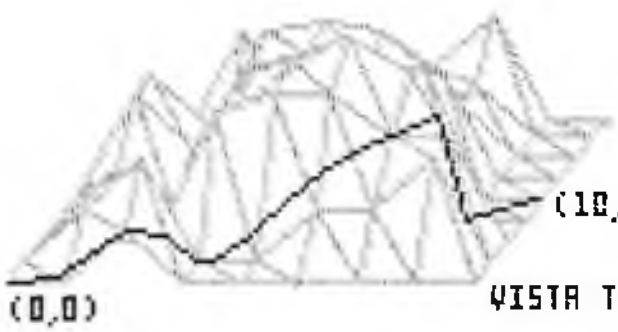
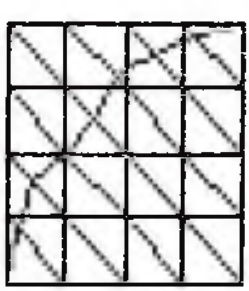


figura e.1.b
RUTA DE DISTANCIA MINIMA
del Origen (0,0)
al Destino (10,5)
Sin aplicar reducciones al terr

figura Ejemplo 1 b)

```
RUTA DISTANCIA MINIMA
Origen    : 0.00 \ 0.00
Destino   : 5.00 \ 5.00
Distancia: 9.26
```



VISTA SUPERIOR

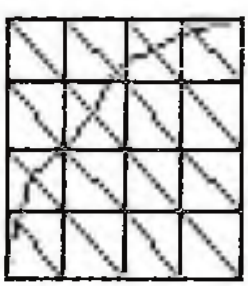
VISTA TRIDIMENSIONAL

Figura e.2.a
RUTA DE DISTANCIA MINIMA
del Origen (0,0)
al Destino (5,5)

Aplicando reducciones al terreno

figura Ejemplo 2 a)

RUTA DISTANCIA MINIMA
Origen : 0.00 \ 0.00
Destino : 5.00 \ 5.00
Distancia: 9.26



VISTA SUPERIOR

VISTA TRIDIMENSIONAL

figura e.2.a
RUTA DE DISTANCIA MINIMA
del Origen (0,0)
al Destino (5,5)

Aplicando reduccionez al terreno

figura Ejemplo 2 a)

RUTA DISTANCIA MINIMA
Origen : 0.00 \ 0.00
Destino : 5.00 \ 5.00
Distancia: 9.26

VISTA SUPERIOR

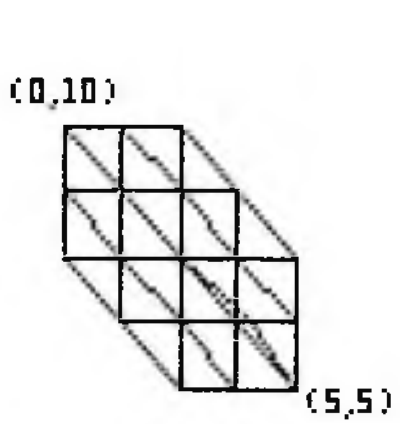
(0,0)

(0,0)

VISTA TRIDIMENSIONAL

figura e.2.b
RUTA DE DISTANCIA MINIMA
del Origen (0,0)
al Destino (5,5)
Sin aplicar reducciones al terre

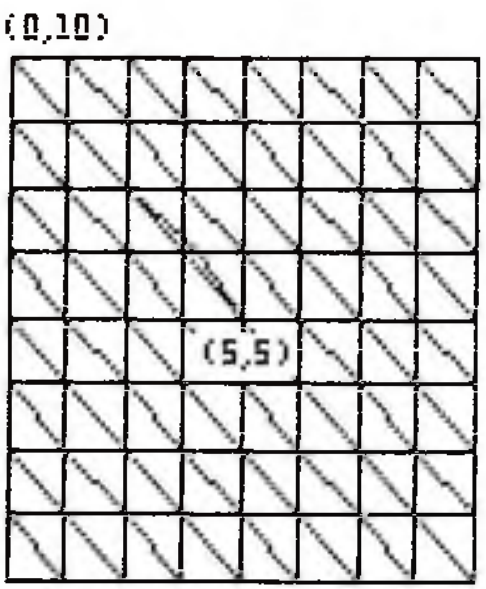
figura Ejemplo 2 b)



VISTA TRIDIMENSIONAL

Figura e.3.a
RUTA DE DISTANCIA MINIMA
del Origen (0,10)
al Destino (5,5)
Aplicando reducciones al terr

figura Ejemplo 3 a)



VISTA SUPERIOR

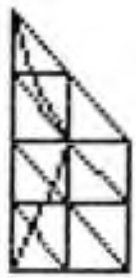
```
RUTA DISTANCIA MINIMA
Origen    : 0.00 \ 10.00
Destino   : 5.00 \ 5.00
Distancia: 8.59
```

VISTA TRIDIMENSIONAL

Figura e.3.b
RUTA DE DISTANCIA MINIMA
del Origen (0,10)
al Destino (5,5)
Sin aplicar reducciones al terre

figura Ejemplo 3 b)

```
RUTA DISTANCIA MINIMA
Origen   : 5.00 \ 0.00
Destino  : 5.00 \ 5.00
Distancia: 7.23
```

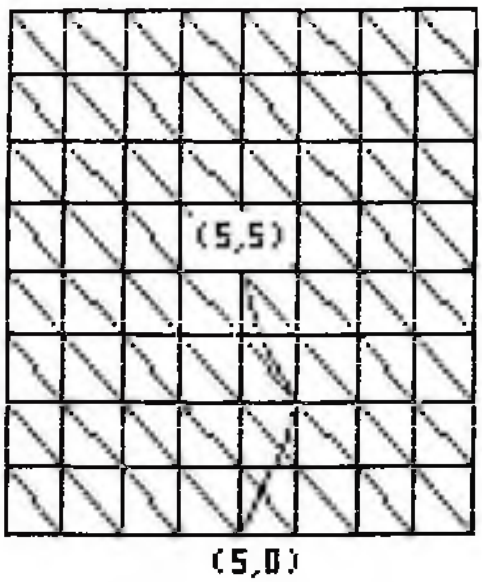


VISTA SUPERIOR

Figura e.4.a
RUTA DE DISTANCIA MINIMA
del Origen (5,0)
al Destino (5,5)
Aplicando reducciones al terren

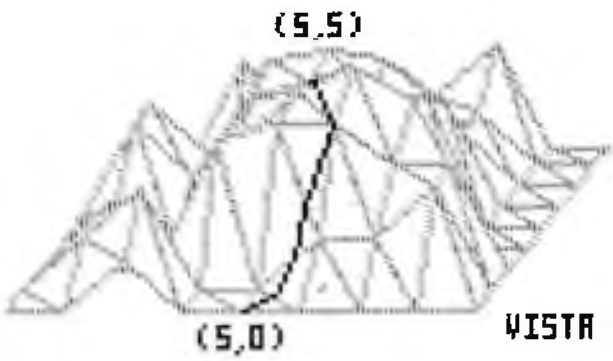
VISTA TRIDIMENSIONAL

figura Ejemplo 4 a)



VISTA SUPERIOR

RUTA DISTANCIA MINIMA
Origen : 5.00 \ 0.00
Destino : 5.00 \ 5.00
Distancia: 7.23



VISTA TRIDIMENSIONAL

Figura e.4.b
RUTA DE DISTANCIA MINIMA
del Origen (5,0)
al Destino (5,5)
Sin aplicar reducciones al terre

figura Ejemplo 4 b)

C) COMPARACION DE TIEMPOS OBTENIDOS EN LAS PRUEBAS ANTERIORES

La tabla siguiente muestra los tiempos obtenidos en las pruebas anteriores. En ella se puede apreciar el ahorro en tiempo al aplicar el algoritmo propuesto en este trabajo.

COORDENADAS ORIGEN	COORDENADAS DESTINO	TIEMPO (min) CON REDUCCION	TIEMPO (min) SIN REDUCCION
0,0	10,5	6	8
0,0	5,5	1.32	2.27
0,10	5,5	1.05	2.30
5,0	5,5	0.34	1.48

Los tiempos obtenidos sin reducir regiones del terreno son tomados al obtener la ruta de longitud minima sobre la representación del terreno de mejor presición, en este caso, la del nivel 3.

De los resultados anteriores se concluye que el algoritmo propuesto reduce el tiempo de obtención de rutas de longitud minima.

APENDICE A "AMPLIFICACION".

.Este concepto se aplica en la obtención de la cota inferior.

.No siempre la distancia viajada en el modelo rudo del terreno nos proporciona una buena cota inferior.

.Es necesario introducir un factor de amplificación que proporcione cotas más precisas.

Debido a que existe error en la representación del terreno utilizada, se introduce el concepto de Amplificación. El Factor de Amplificación K, es el error estimado de esta representación con respecto al terreno real, por lo que se asigna un valor de K a cada triángulo de la misma.

Cuando se representa con un triángulo una porción de terreno, y sobre todo cuando la representación es gruesa (ver sección 1.1), se encuentra que las distancias mínimas calculadas sobre dicho triángulo pueden ser aún menores, es decir, en el caso real la distancia podría ser menor a la calculada sobre el triángulo.(ver figura a.1).

Debido a lo anterior es necesario introducir en el cálculo de distancias mínimas, sobre triángulos, algún factor de "corrección" para acercarnos más al caso real. Este factor de corrección es la función M(.), con la cual se calcula el máximo ahorro en distancia que se puede tener respecto a un lado.

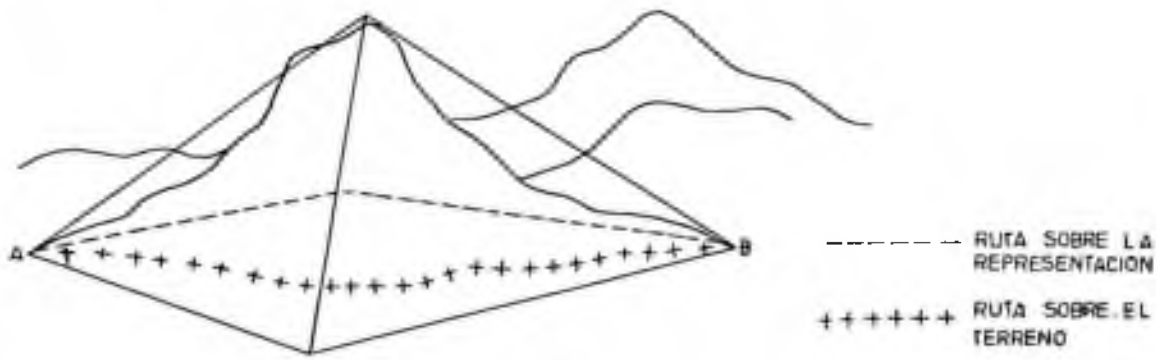


figura a.1

DESCRIPCION

Básicamente la amplificación consiste en lo siguiente: Dos medios de iguales o distintos Factores de Amplificación, a través de los cuales se desea navegar, proporcionan una distancia mínima en la cual se considerará una disminución de la distancia, con respecto a la que se obtendría si no se considerase el error introducido en la representación. La función M(.) se auxilia de la ley de Snell de la refracción de la luz, para obtener el máximo ahorro.

Esta ley esta dada por: $\cos(fi1) = k \cos(fi2)$

Entonces, una distancia mínima que pasa a través de dos triángulos de diferentes factores de amplificación, sufrirá una disminución proporcional a k. (ver figura a.2)

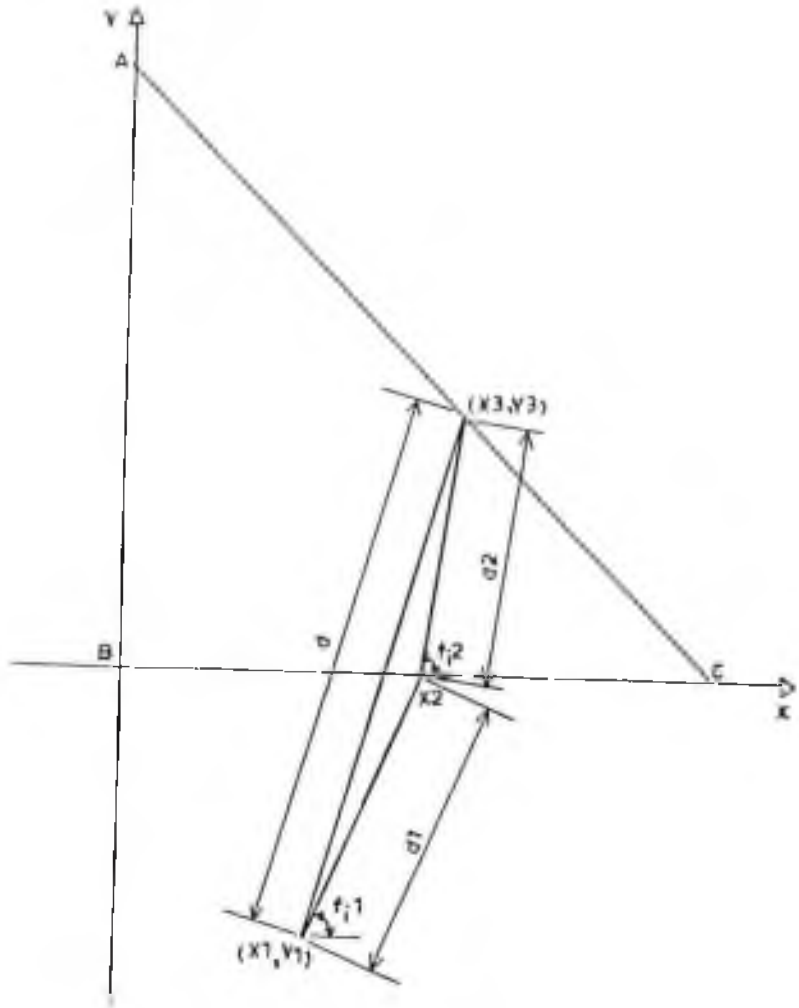


figura a.2

El lado que esta sobre el eje de las x's, es el último sobre el que se ha navegado y se desea observar cómo es afectada la distancia al navegar sobre el triángulo ABC.

La función M(.) está dada por:

$$M(.) = \min_{0<=x<=L} |d - d1 - kd2| \qquad [1]$$

donde d,d1 y d2 estan en función de fi2.

L es la longitud de dicho lado.

$$d = \left((x3 - x1)^2 + (y3 - y1)^2 \right)^{1/2}$$

$$d1 = \left((x2 - x1)^2 + y1^2 \right)^{1/2}$$

$$d2 = \left((x3 - x2)^2 + y3^2 \right)^{1/2}$$

La ecuación de la recta esta dada por:

$$y = mx + b \qquad [2]$$

si alpha = tg(fi2) la ecuación de la recta que va de x2 a (x3,y3) es:

$$y = (x - x2)alpha \qquad [3]$$

igualando [2] y [3] obtenemos:

$$x3 = ((alpha)(x2) + b)/(alpha-m)$$

$$y3 = (m((alpha)(x2) + b)/(alpha-m)) + b$$

y para x2:

$$x2 = x1 - (y1/tg(fi1))$$

Con esto se logra disminuir la distancia de la cuña al punto Origen sobre el cual se esta trabajando, a fin de hacer más rigurosa la prioridad entre las cuñas para ser exterdidas y acercarnos al caso real.

El factor de amplificación, puede ser obtenido de la

siguiente forma:

$$k = \frac{\text{pendiente del triángulo}}{\text{mínima pendiente del terreno}}$$

La amplificación no se implementó en este trabajo.

APENDICE B ALGORITMO DE DIJKSTRA

GRAFICAS

Muchas situaciones del mundo real pueden ser descritas convenientemente empleando un diagrama, el cual esta compuesto de un conjunto de puntos y líneas. Las líneas unen a ciertos pares de puntos no necesariamente diferentes. Por ejemplo, los puntos pueden ser ciudades y las líneas enlaces de comunicación entre ellas. La abstracción matemática de situaciones de este tipo da origen al concepto de gráfica.

Una gráfica G se compone del triple siguiente: $(V(G),E(G),Y_G)$. Donde V(G) es un conjunto no vacío de vértices (los puntos de la gráfica), E(G) es un conjunto de orillas o lados (las líneas en la gráfica) y; Y_G es una función de incidencia, la cual asocia con cada orilla de G un par de vértices, no necesariamente diferentes, de G. Si e es una orilla, y u y v son vértices tales que $Y_G(e) = uv$, entonces se dice que e une a u y v; u y v son llamados extremos de e.

Un ejemplo servirá para aclarar la definición:

$$G = (V(G),E(G),Y_G)$$

donde $V(G) = \{v1,v2,v3,v4,v5\}$

$$E(G) = \{e1,e2,e3,e4,e5,e6,e7,e8\}$$

y Y_G es definida como:

$Y_G(e1) = v1v2$, $Y_G(e2) = v2v3$, $Y_G(e3) = v3v3$, $Y_G(e4) = v3v4$

$Y_G(e5) = v2v4$, $Y_G(e6) = v4v5$, $Y_G(e7) = v2v5$, $Y_G(e8) = v2v5$

En la figura b.1 se muestra la gráfica G.

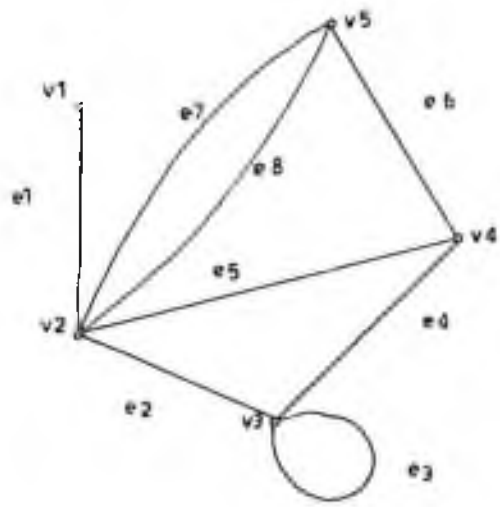


figura 6.1 Gràfica G.

Puede haber muchas formas de dibujar una gràfica, sinembargo el diagrama de una gràfica solo describe la relaciòn de incidencia entre sus vèrtices y orillas.

EL PROBLEMA DEL CAMINO MAS CORTO

Si a cada orilla e de una gràfica G le asociamos un nùmero real w(e), llamado su peso, entonces G es llamada Gràfica Pesada. Las gràficas pesadas ocurren frecuentemente en las aplicaciones de teoria de gràficas.

El problema del camino màs corto puede enunciarse de la siguiente manera: dada una red ferroviaria que conecta varios pueblos, determine la ruta màs corta entre dos pueblos especìficos de la red.

Debe encontrarse, en una gràfica pesada, un camino de mìnimo peso que conecta dos vèrtices especìficos u y v ; los pesos representan las distancias entre dos pueblos conectados o

comunicados por la vìa ferrea, y por lo tanto, estos pesos no pueden ser negativos.

Para claridad en la exposiciòn, nos referiremos al peso de un camino en una grà fica pesada como su "longitud"; similarmente, el peso mìnimo entre los vèrtices u y v serà llamado la "distancia" entre u y v, y se denotarà por d(u,v). Tambièn se adoptarà la convenciòn de que w(u,v) = oo si uv no pertenece a E(G).

El algoritmo aquì descrito fue descubierto por Dijkstra (1959). Este no solo encuentra el camino màs corto entre u y v_o, sino que tambièn encuentra los caminos màs cortos a todos los demàs vèrtices de G. La idea bàsica es la siguiente:

En el algoritmo, cada vèrtice v lleva asociado una etiqueta l(v), la cual es un lìmite superior a $d(u_o,v)$. Inicialmente $l(u_o) = 0$ y $l(v) = oo$ para $v \neq u_o$. (Para implementar este algoritmo en computadora, oo se reemplaza por un nùmero suficientemente grande). Conforme es procesado el algoritmo, estas etiquetas son modificadas de tal forma que al final de la etapa i,

$$l(u) = d(u_o,u) \quad \text{para} \quad u \in S_i$$

y

$$l(v) = \min(d(u_o,u) + w(uv)) \quad \text{para} \quad v \in \bar{S}_i$$

Algoritmo de Dijkstra.

El algoritmo trabaja construyendo un conjunto S de vèrtices, para los que la distancia màs corta a un vèrtice u_o, que se dice es el origen del camino, es conocida.

a) En cada paso del algoritmo se selecciona un vértice $v \in V$, cuya distancia es la más corta al origen y se agrega al conjunto S.

b) Para cada vértice $v \in V-S$, se obtiene el costo mínimo. Para lo cual se considera al vértice recién agregado al conjunto S. En otras palabras, el algoritmo de Dijkstra se comporta de la siguiente manera:

Para una gráfica de n vértices:

1. Se hace $l(u_o) = 0$, $l(v) = \infty$ para $v \neq u_o$, $S_o=\{u_o\}$ e $i = 0$.
2. Para cada $v \in \bar{S}_i$, se reemplaza $l(v)$ por $\min(l(v), l(u_i)+w(u_i v))$.
   Se obtiene el $\min_{v \in \bar{S}_i}(l(v))$, y al vértice v, para el que este mínimo fue obtenido, se le denota u_{i+1}.
   Se hace $S_{i+1} = S_i \cup \{u_{i+1}\}$.
3. Si $i = n-1$, alto. si $i < n-1$, se reemplaza i por i+1, se regresa al paso 2.

Cuando el algoritmo termina, la distancia desde u_o a v está dada por el valor final de $l(v)$. Si nuestro interés es determinar la distancia a un vértice específico v_o, el algoritmo debe parar cuando u_j iguala a v_o. Este algoritmo tiene una complejidad $O(n^2)$. En la forma en que se ha descrito, el algoritmo de Dijkstra determina solamente las distancias desde u_o a todos los otros vértices, y no el camino más corto a cada uno de ellos. Sinembargo, los caminos más cortos pueden ser determinados al seguirles la pista a los predecesores de los vértices, en el árbol de caminos cortos que se forma sobre la gráfica. Un árbol

es una gràfica en la que todos los vèrtices estan conectados, sin que haya lazos cerrados o ciclos. Entonces se puede considerar que éste algoritmo es un procedimiento para obtener o generar árboles sobre una gráfica. El predecesor de un vèrtice v, en el camino más corto a u_o, es aquel que está unido mediante una orilla con v, y además está contenido en el camino más corto a u_o.

A continuación se presenta, en la figura b.2, un ejemplo del resultado de aplicar el algoritmo de Dijkstra a una gráfica pesada. En esta figura se ilustran los pasos para obtener los caminos más cortos a cada uno de los vèrtices de la gràfica, cada paso del algoritmo es detallado en la tabla 1.

(S)	v_{i+1}	$l(u_o)$	l(a)	l(b)	l(c)	l(d)	l(e)	l(f)	l(g)
$\{u_o\}$	-	0	∞	∞	∞	∞	∞	∞	∞
$\{u_o\}$	u	0	2	∞	∞	4	1	3	∞
$\{u_o,e\}$	e	0	2	∞	∞	4	1	2	3
$\{u_o,e,a\}$	a	0	2	5	5	4	1	2	3
$\{u_o,e,a,f\}$	f	0	2	5	5	4	1	2	3
$\{u_o,e,a,f,g\}$	g	0	2	5	4	4	1	2	3
$\{u_o,e,a,f,g,c\}$	c	0	2	5	4	4	1	2	3
$\{u_o,e,a,f,g,c,d\}$	d	0	2	5	4	4	1	2	3
$\{u_o,e,a,f,g,c,d,b\}$	b	0	2	5	4	4	1	2	3

TABLA 1.

En la tabla anterior, podemos obtener el camino más corto a cualquier vèrtice, por ejemplo a c:

Si se analiza la columna l(c), se observa que la distancia más corta a c es de 4 y que ese 4 se repite, en la misma columna, desde tres renglones arriba; esto es, cuando $v_{i+1} = g$, lo que nos indica que el predecesor de c es g. Si ahora se analiza la columna l(g) como se hizo con l(c), se obtiene que el predecesor de g es el vèrtice e; el cual a su vez (analizando l(e)), tiene como predecesor a u_o, que es el origen del camino.

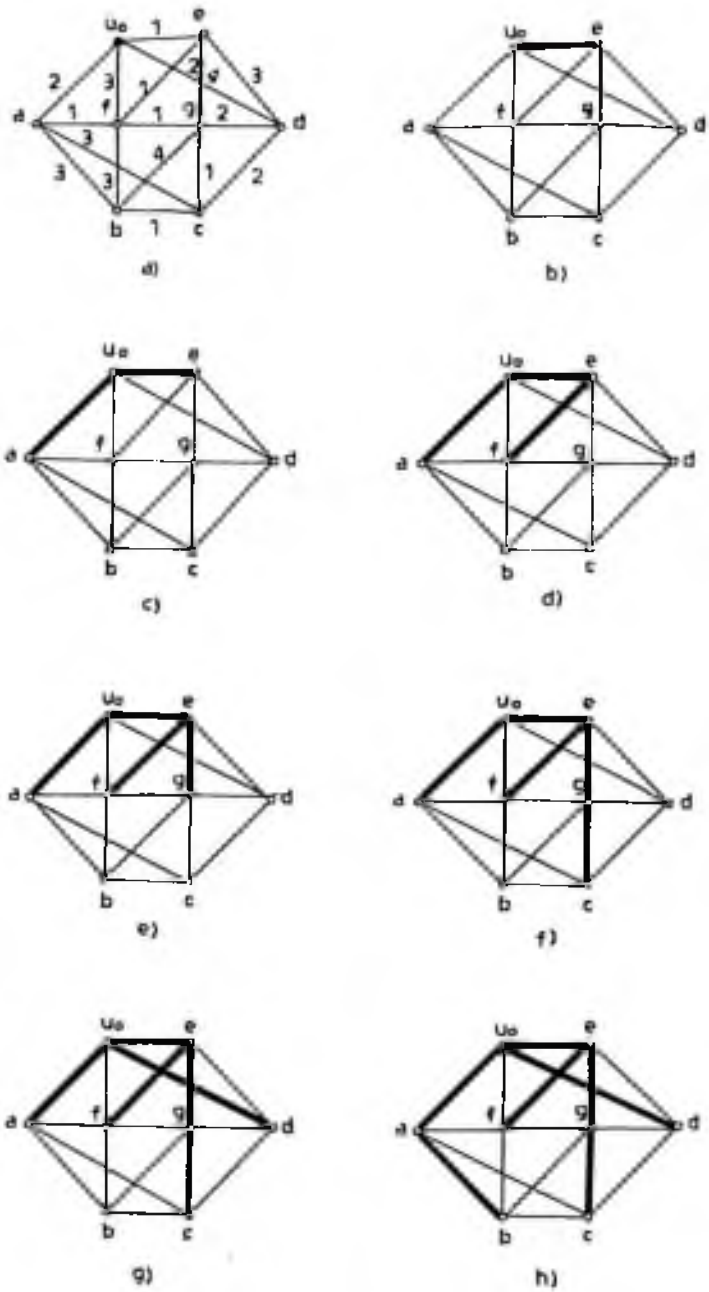


figura 5.2 Pasos para obtener los caminos más cortos a partir del vértice uo, a todos los demás vértices en la gráfica.

REFERENCIAS.

[AHU-74] Aho, A.V.; Hopcroft, J.E. y Ullman, J.D. "The Design and Analysis of Computer Algorithms", Addison-Wesley, 1974.

[BH-] Barrera, Renato; Hinojosa, Alejandro. "Compression Methods for Terrain Relief". Por publicarse C.I.E.A. I.P.N.

[BV-84] Barrera, Renato; Vázquez, A.M. "A Hierarchical Method for Representing Terrain Relief". IIMAS-UNAM, Computer Systems Dept. 1984.

[C-73] Coxeter, H.S.M. "Regular Polytopes". Third Edition. Dover Publications, Inc. , New York. 1973

[LP-84] Lee, D.T.; Preparata, F.P. "Euclidean Shortest Paths in the Presence of Rectilinear Boundaries", Networks, 14 (1984).

[M-85] Mount, David. "Voronoi Diagrams on the Surface of a Polyhedron". Dept. of Computer Science, University of Maryland. report CAR-TR-121, CS-TR-1496. 1985.

[OSB-85] O'Rourke, Joseph; Suri, Subhash; Booth, Heather. "Shortest Paths on Polyhedral Surfaces". Dept. of Electrical Engineering and Computer Science, Johns Hopkins University. 1985.

[SS-84] Sharir,M. y Schorr, A. "On Shortest Paths in Polyhedral Spaces". 16th. ACM Symposium on Theory of Computing (1984).

El jurado designado por la Sección de Computación del Departamento de Ingeniería Eléctrica del Centro de Investigación y de Estudios Avanzados del I.P.N., aprobó esta tesis el 17 de Junio de 1988.

Dr. Renato Barrera R.

Dr. Gilberto Calvillo V.

Dr. Cristóbal Vargas.