(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 468 795**

(51) Int. CI.:

G01C 21/34 (2006.01)
G08G 1/0968 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **09.07.2010 E 10754899 (2)**

(97) Fecha y número de publicación de la concesión europea: **23.04.2014 EP 2452325**

(54) Título: **Dispositivo de navegación y método para el cálculo de ruta con dependencia temporal**

(30) Prioridad:

09.07.2009 US 213746 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
17.06.2014

(73) Titular/es:

TOMTOM INTERNATIONAL B.V. (50.0%)
De Ruijterkade 154
1011 AC Amsterdam, NL y
TOMTOM DEVELOPMENT GERMANY GMBH (50.0%)

(72) Inventor/es:

SCHILLING, HEIKO;
GAWRILOW, EWGENIJ;
HILGER, MORITZ;
PROFOUS, ANDREAS;
WERBER, JÜRGEN y
TERTOOLEN, SIMONE

(74) Agente/Representante:

DE ELZABURU MÁRQUEZ, Alberto

DESCRIPCIÓN

Dispositivo de navegación y método para el cálculo de ruta con dependencia temporal

Campo de la invención

La presente invención se refiere a un método informatizado de determinación de una ruta usando datos de mapa y sistemas relacionados. En particular, pero no exclusivamente, el método se puede usar en un dispositivo de navegación usado para facilitar una planificación de ruta.

Antecedentes de la invención

Son conocidos algoritmos de planificación de ruta (tales como el método Dijkstra, método A* o métodos Moore/Pape). No obstante, éstos pueden ser algo lentos. Un ejemplo de cómo se puede aumentar la velocidad de tal planificación de ruta se muestra en la US 6 636 800. La US 6 636 800 enseña un procesamiento previo de datos de mapa dividiendo el mapa en regiones y determinando un árbol de trayectos más cortos (trayectos de coste mínimo) para cada región del mapa. Estos datos procesados previamente se almacenan en memoria y se usan durante el cálculo de una ruta.

Más específicamente, cuando un origen está dentro una región y un destino está dentro de otra región, se puede usar el árbol predeterminado de trayectos más cortos para determinar rápidamente el trayecto entre las regiones a ser usadas como parte de la ruta. Tal método reduce el nivel de procesamiento requerido para determinar una ruta debido a que reduce el número de trayectos explorados desde que un usuario solicita la determinación de una ruta.

Se entenderá que trayecto “más corto” no necesariamente significa el trayecto de la distancia más corta sino que significa el trayecto que tiene el coste mínimo. El coste del trayecto dependerá de los factores que se consideren como especificados en una fórmula de coste y pueden tener en cuenta factores tales como la ruta más rápida y el uso de combustible.

Tal método tiene mucho éxito cuando los trayectos más cortos entre regiones permanecen estáticos. No obstante, los cambios en las condiciones del tráfico con el tiempo pueden cambiar el trayecto que es el más corto entre regiones de manera que el árbol procesado previamente ya no identifica los trayectos más cortos verdaderos entre regiones. Esto puede provocar a una ruta que se determina que no sea la ruta óptima para los criterios especificados.

“Engineering and Augmenting Route Planning Algorithms” de Daniel Delling, 10 de febrero de 2009, XP002612394 describe las ideas detrás de varias técnicas de aceleración para encaminamiento estático exacto en redes de carreteras.

La JP 2004-233230 describe un terminal de navegación que puede mostrar la información de congestión de tráfico sobre el pronóstico de una carretera, se realiza la visualización de ruta de dos o más rutas recomendadas teniendo en cuenta la predicción de congestión de tráfico que se puede encontrar cuando se cambia la hora de salida y la hora de llegada, y aspira a mostrar al usuario la diferencia del efecto de dos o más rutas recomendadas teniendo en cuenta la predicción de congestión de tráfico.

La US 2004/0249568 describe un dispositivo de almacenamiento que almacena datos de mapa incluyendo datos de enlace de enlaces respectivos que constituyen las carreteras en un mapa, y datos estadísticos que incluyen el tiempo de viaje (velocidad de movimiento) que se determina por valores estadísticos de información de tráfico recogida en el pasado, con respecto a cada uno de los enlaces. La US 2004/0249568 además describe que el dispositivo de navegación usa para cada candidata de hora de salida, los datos de mapa y los datos estadísticos de una colección de condiciones que corresponden a los estados tras pasar cada uno de los enlaces de constitución de ruta que constituyen la ruta, para obtener el tiempo de viaje para cada uno de los enlaces de constitución de ruta. A partir de entonces, los tiempos de viaje de los enlaces de constitución de ruta respectivos obtenidos de esta manera se suman, y se obtiene el tiempo de viaje entre la posición de salida y el destino.

Compendio de la invención

Según un primer aspecto de la invención se proporciona un método de determinación de perfiles de coste de rutas usando datos de mapa según la reivindicación 1.

El perfil de coste permite a un usuario y/o un aparato de navegación determinar cómo cambiar la hora de viaje alterará el coste de la ruta. De este modo, se puede seleccionar una hora de viaje en base al perfil de coste.

Según un segundo aspecto de la invención se proporciona una portadora de datos que tiene almacenada en la misma instrucciones que, cuando se ejecutan por un aparato de procesamiento, hacen al aparato de procesamiento ejecutar un método según el primer aspecto de la invención.

Según un tercer aspecto de la invención se proporciona un dispositivo informático que comprende una memoria que

tiene almacenados en la misma datos de mapa según la reivindicación 11.

Breve descripción de las figuras

Ahora sigue a modo de ejemplo solamente una descripción detallada de realizaciones de la presente invención con referencia a los dibujos anexos en los que:

La **Figura 1** es una ilustración esquemática de una parte ejemplar de un Sistema de Posicionamiento Global (GPS) utilizable por un dispositivo de navegación;

La **Figura 2** es un esquema de un servidor y dispositivo de navegación en comunicación a través de un canal de comunicaciones;

La **Figura 3a** es un esquema de un dispositivo de navegación;

La **Figura 3b** es una representación esquemática de una pila de arquitectura empleada por el dispositivo de navegación de la Figura 3a;

La **Figura 4** es una vista en perspectiva de un dispositivo de navegación;

La **Figura 5** muestra una parte de un mapa ejemplo que se genera por realizaciones de la invención;

La **Figura 6** muestra el mapa de la Figura 5 en el que se muestran nodos usados para encaminamiento;

La **Figura 7** muestra el mapa de la Figura 5 que sigue el procesamiento;

La **Figura 8** muestra un ejemplo de cómo los nodos, en algunas realizaciones, se asignan al mapa de la Figura 7;

La **Figura 9** ilustra un ejemplo de cómo se puede dividir un mapa en una pluralidad de regiones anidadas;

La **Figura 9a** ilustra una mejora de la división de la Figura 9;

La **Figura 10** muestra un ejemplo de conjunto de vectores de bit utilizados por realizaciones de la invención;

La **Figura 10a** ilustra cómo se utiliza información de perfil de tiempo durante una exploración Dijkstra de una red;

La **Figura 11** muestra un ejemplo de formato de fichero para un fichero de una realización de la invención;

La **Figura 12** muestra una realización de cómo se pueden codificar la tabla de reasignación de región y las listas de ID de región de la Figura 11;

La **Figura 13** muestra un ejemplo de formato de fichero para la región anidada más tosca como se ilustra en la Figura 3;

La **Figura 14** muestra un ejemplo de formato de fichero para regiones anidadas distintas de la más tosca;

La **Figura 15** ejemplifica la coalescencia de bits dentro de un esquema de codificación;

La **Figura 16** ejemplifica el efecto de la coalescencia de bits;

La **Figura 17** también ejemplifica el efecto de la coalescencia de bits;

La **Figura 18** muestra un mapa que resalta rutas consideradas usando las técnicas de encaminamiento de la **TÉCNICA ANTERIOR**;

La **Figura 19** muestra un mapa que resalta rutas consideradas por realizaciones de la invención actual;

La **Figura 20** muestra un ejemplo del método de búsqueda A* (**TÉCNICA ANTERIOR**);

La **Figura 21** muestra una modificación a la Figura 20 usada por al menos algunas realizaciones de la invención;

La **Figura 22** muestra un diagrama de flujo que perfila los pasos del método;

La **Figura 23** muestra un visualizador de un dispositivo de navegación que ilustra las velocidades esperadas en los segmentos navegables de la ruta;

La **Figura 24** muestra un visualizador de un dispositivo de navegación que ilustra un perfil de coste para una o más rutas entre un origen y destino;

La **Figura 25** muestra un visualizador de un dispositivo de navegación que ilustra un perfil de coste de una o más

rutas entre un origen y un destino;

La **Figura 26** muestra un visualizador de un dispositivo de navegación que ilustra una ruta entre un origen y destino para una hora de viaje particular y un control para cambiar la hora de viaje;

La **Figura 27** muestra una actualización del visualizador de la Figura 26 después de que el usuario ha alterado la hora de viaje;

La **Figura 28** muestra una actualización del visualizador de la Figura 27 después de que el usuario ha alterado la hora de viaje de nuevo;

La **Figura 29** muestra una serie de visualizaciones de un dispositivo de navegación, en donde se sugiere al usuario una hora de viaje;

La **Figura 30** muestra un gráfico que ilustra un perfil de coste de dos rutas que compiten en un nodo de intersección; y

La **Figura 31** muestra perfiles de coste de límite superior y de límite inferior derivados de los perfiles de coste mostrados en la Figura 30.

Descripción detallada de las figuras

En toda la descripción siguiente se usarán los mismos números de referencia para identificar partes iguales. En toda la descripción de las diversas realizaciones se hace referencia al diagrama de flujo de la Figura 22 que tiene números de referencia en la serie 19xx.

La Figura 1 muestra esquemáticamente un Sistema de Posicionamiento Global (GPS) que es un sistema de navegación basado en radio por satélite que se puede utilizar para determinar la posición continua, velocidad, tiempo, y en algunos casos información de dirección para un número de usuarios ilimitado. Antiguamente conocido como NAVSTAR, el GPS incorpora una pluralidad de satélites que orbitan la tierra en órbitas extremadamente precisas. En base a estas órbitas precisas, los satélites del GPS pueden retransmitir su ubicación, como datos GPS, a cualquier número de unidades de recepción. No obstante, se entenderá que se podrían usar sistemas de Posicionamiento Global, tales como GLOSNASS, el sistema de posicionamiento Galileo europeo, el sistema de posicionamiento COMPASS o IRNSS (Sistema Indio de Satélites de Navegación Regional).

El sistema GPS se implementa cuando un dispositivo, especialmente equipado para recibir datos GPS, comienza a explorar radiofrecuencias de señales de satélite de GPS. Tras recibir una señal radio desde un satélite de GPS, el dispositivo determina la ubicación precisa de ese satélite a través de una pluralidad de diferentes métodos convencionales. El dispositivo continuará explorando, en la mayoría de los casos, señales hasta que haya adquirido al menos tres señales de satélites diferentes (señalar que esa posición no de manera normal, pero se puede determinar, solamente con dos señales usando otras técnicas de triangulación). Implementando triangulación geométrica, el receptor utiliza las tres posiciones conocidas para determinar su propia posición bidimensional con respecto a los satélites. Esto se puede hacer de una manera conocida. Adicionalmente, la adquisición de una cuarta señal de satélite permite al dispositivo de recepción calcular su posición tridimensional mediante el mismo cálculo geométrico de una manera conocida. Los datos de posición y velocidad se pueden actualizar en tiempo real de una forma continua por un número de usuarios ilimitado.

Como se muestra en la Figura 1, el sistema GPS 100 comprende una pluralidad de satélites 102 que orbitan alrededor de la tierra 104. Un receptor GPS 106 recibe datos GPS como señales de datos de satélite de GPS de espectro expandido 108 desde un número de la pluralidad de satélites 102. Las señales de datos de espectro expandido 108 se transmiten continuamente desde cada satélite 102, las señales de datos de espectro expandido 108 transmitidas cada una comprende un flujo de datos que incluye información que identifica un satélite 102 particular desde el cual se origina el flujo de datos. El receptor GPS 106 generalmente requiere señales de datos de espectro expandido 108 desde al menos tres satélites 102 a fin de ser capaz de calcular una posición bidimensional. La recepción de una cuarta señal de datos de espectro expandido permite al receptor GPS 106 calcular, usando una técnica conocida, una posición tridimensional.

De esta manera, un sistema GPS permite a un usuario de un dispositivo que tiene un receptor GPS 106 determinar su posición en el planeta tierra dentro de unos pocos metros. A fin de hacer uso de esta información ha llegado a ser una práctica común basarse en mapas electrónicos que permiten que la posición del usuario sea mostrada sobre el mismo. Tales mapas se ejemplifican por proveedores tales como TeleAtlas (http://www.teleatlas.com).

Tales mapas electrónicos no solamente permiten que una posición de usuario sea mostrada sobre el mismo usando un sistema GPS (o por otros medios) sino que permiten a un usuario planificar rutas para viajes y similares (propósitos de encaminamiento). A fin de que ocurra esta planificación de ruta el mapa electrónico se procesa por un dispositivo de navegación, el cual puede ser proporcionado por un dispositivo informático general.

Ejemplos específicos de un dispositivo de navegación incluyen dispositivos de navegación por Satélite (Sat. Nav) a

los que es conveniente referirse como Dispositivos de Navegación Portátiles (PND). Se debería recordar, no obstante, que las enseñanzas de la presente invención no están limitadas a los PND sino que en su lugar son aplicables universalmente a cualquier tipo de dispositivo de procesamiento que esté configurado para procesar mapas electrónicos, generalmente para proporcionar planificación de ruta y funcionalidad de navegación. Se desprende por lo tanto que en el contexto de la presente solicitud, un dispositivo de navegación está destinado a incluir (sin limitación) cualquier tipo de dispositivo de navegación y planificación de ruta, con independencia de si ese dispositivo se encarna como un PND, un vehículo tal como un automóvil, un recurso informático portátil, por ejemplo un ordenador personal (PC) portátil, un teléfono móvil o un Asistente Digital Personal (PDA) que ejecuta un software de navegación y planificación de ruta o un servidor, u otro dispositivo informático, que proporcione tal funcionalidad a través de una red.

Un ejemplo de tal dispositivo de navegación en forma de un PND se muestra en las Figuras 3a y 4b y se debería señalar que el diagrama de bloques del dispositivo de navegación 200 no incluye todos los componentes del dispositivo de navegación, sino que solamente es representativo de muchos componentes ejemplo. El dispositivo de navegación 200 está situado dentro de un alojamiento (no mostrado). El dispositivo de navegación 200 incluye circuitería de procesamiento que comprende, por ejemplo, el procesador 202 mencionado anteriormente, el procesador 202 que está acoplado a un dispositivo de entrada 204 y un dispositivo de visualización, por ejemplo una pantalla de visualización 206. Aunque se hace referencia aquí al dispositivo de entrada 204 en singular, los expertos deberían apreciar que el dispositivo de entrada 204 representa cualquier número de dispositivos de entrada, incluyendo un dispositivo de teclado, un dispositivo de entrada de voz, un dispositivo de panel táctil y/o cualquier otro dispositivo de entrada conocido utilizado para introducir información. De la misma manera, la pantalla de visualización 206 puede incluir cualquier tipo de pantalla de visualización tal como un Visualizador de Cristal Líquido (LCD), por ejemplo.

En el dispositivo de navegación 200, el procesador 202 está conectado operativamente a y es capaz de recibir información de entrada desde el dispositivo de entrada 204 a través de una conexión 210, y conectado operativamente a al menos uno de la pantalla de visualización 206 y el dispositivo de salida 208, a través de las conexiones de salida 212 respectivas, para sacar información al mismo. El dispositivo de navegación 200 puede incluir un dispositivo de salida 208, por ejemplo un dispositivo de salida audible (por ejemplo un altavoz). Como el dispositivo de salida 208 puede producir información audible para un usuario del dispositivo de navegación 200, se debería entender igualmente que el dispositivo de entrada 204 puede incluir un micrófono y software para recibir comandos de voz de entrada también. Además, el dispositivo de navegación 200 también puede incluir cualquier dispositivo de entrada adicional 204 y/o cualquier dispositivo de salida adicional, tales como dispositivos de entrada/salida de audio por ejemplo.

El procesador 202 está conectado operativamente a la memoria 214 a través de la conexión 216 y está adaptado además para recibir/enviar información desde/a los puertos de entrada/salida (I/O) 218 a través de la conexión 220, en donde el puerto de I/O 218 es conectable a un dispositivo de I/O 222 externo al dispositivo de navegación 200. El dispositivo de I/O 222 externo puede incluir, pero no está limitado a un dispositivo de escucha externo, tal como un audífono por ejemplo. La conexión al dispositivo de I/O 222 puede ser además una conexión cableada o inalámbrica a cualquier otro dispositivo externo tal como un equipo estéreo de coche para operación manos libres y/o para operación activada por voz por ejemplo, para conexión a un audífono o auriculares, y/o para conexión a un teléfono móvil por ejemplo, en donde la conexión de teléfono móvil se puede usar para establecer una conexión de datos entre el dispositivo de navegación 200 e Internet o cualquier otra red por ejemplo, y/o establecer una conexión a un servidor a través de Internet o alguna otra red por ejemplo.

La memoria 214 del dispositivo de navegación 200 comprende una parte de memoria no volátil (por ejemplo para almacenar un código de programa) y una parte de memoria volátil (por ejemplo para almacenar datos según se ejecuta el código de programa). El dispositivo de navegación también comprende un puerto 228, que comunica con el procesador 202 a través de la conexión 230, para permitir que una tarjeta de memoria extraíble (comúnmente conocida como una tarjeta) sea añadida al dispositivo 200. En la realización que se describe el puerto está dispuesto para permitir que sea añadida una tarjeta SD (Secure Digital). En otras realizaciones, el puerto puede permitir que sean conectados otros formatos de memoria (tales como tarjetas Compact Flash (CF), Memory Sticks™, tarjetas de memoria xD, unidades Instantáneas USB (Canal Principal Serie Universal), tarjetas MMC (Multimedia), tarjetas SmartMedia, micro unidades, o similares).

La Figura 2 ilustra una conexión operativa entre el procesador 202 y una antena/receptor 224 a través de la conexión 226, en donde la antena/receptor 224 puede ser una antena/receptor GPS por ejemplo y como tal funcionaria como el receptor GPS 106 de la Figura 1. Se debe entender que la antena y el receptor designados por el número de referencia 224 están combinados esquemáticamente para ilustración, pero que la antena y el receptor pueden ser componentes situados separadamente, y que la antena puede ser una antena de parche GPS o una antena helicoidal por ejemplo.

Además, el dispositivo de navegación portátil o de mano 200 de las Figuras 3 y 4 se puede conectar o "acoplar" de una manera conocida a un vehículo tal como una bicicleta, una motocicleta, un coche o un barco por ejemplo. Tal dispositivo de navegación 200 entonces es extraíble de la ubicación acoplada para uso de navegación portátil o de

mano. De hecho, en otras realizaciones, el dispositivo 200 se puede disponer que sea de mano para permitir la navegación de un usuario.

Con referencia la Figura 3a, el dispositivo de navegación 200 puede ser una unidad que incluye el dispositivo de visualización y entrada integrado 206 y los otros componentes de la Figura 3a (incluyendo, pero no limitado a, el receptor GPS 224 interno, el procesador 202, una fuente de alimentación (no mostrada), los sistemas de memoria 214, etc.).

El dispositivo de navegación 200 se puede montar en un brazo 252, que en sí mismo se puede asegurar en el salpicadero/ventana/etc. de un vehículo usando una copa de succión 254. Este brazo 252 es un ejemplo de una estación de acoplamiento en la que se puede acoplar el dispositivo de navegación 200. El dispositivo de navegación 200 se puede acoplar o conectar de otro modo al brazo 252 de la estación de acoplamiento mediante presión conectando el dispositivo de navegación 200 al brazo 252 por ejemplo. El dispositivo de navegación 200 entonces se puede girar sobre el brazo 252. Para liberar la conexión entre el dispositivo de navegación 200 y la estación de acoplamiento, se puede presionar un botón (no mostrado) en el dispositivo de navegación 200, por ejemplo. Otras disposiciones igualmente adecuadas para acoplar y desacoplar el dispositivo de navegación 200 a una estación de acoplamiento son bien conocidos por los expertos en la técnica.

Volviendo a la Figura 3b el procesador 202 y la memoria 214 cooperan para soportar una BIOS (Sistema de Entrada/Salida Básico) 282 que funciona como una interfaz entre los componentes hardware funcionales 280 del dispositivo navegación 200 y el software ejecutado por el dispositivo. El procesador 202 entonces carga un sistema operativo 284 desde la memoria 214, que proporciona un entorno en el que se puede ejecutar la aplicación software 286 (que implementa algo de o toda la funcionalidad de navegación y planificación de ruta descrita). La aplicación software 286 proporciona un entorno operacional que incluye la Interfaz Gráfica de Usuario (GUI) que soporta las funciones centrales del dispositivo de navegación, por ejemplo las funciones de visión de mapas, planificación de ruta, navegación y cualesquiera otras funciones asociadas con las mismas. A este respecto, parte de la aplicación software 286 comprende un módulo de generación de vistas 288.

En la realización que se describe, el procesador 202 del dispositivo de navegación está programado para recibir datos GPS recibidos por la antena 224 y, de vez en cuando, almacenar esos datos GPS, junto con un la marca de tiempo de cuando fueron recibidos los datos GPS, dentro de la memoria 214 para crear una serie de posiciones del dispositivo de navegación. Cada registro de datos así almacenado se puede considerar como una posición GPS; es decir es una posición de la ubicación del dispositivo de navegación y comprende una latitud, una longitud, una marca de tiempo y un informe de precisión. Esta serie de posiciones se puede considerar como datos de posicionamiento.

En una realización los datos se almacenan sustancialmente de una forma periódica que es por ejemplo cada 5 segundos. Los expertos apreciarán que serían posibles otros periodos y que haya un equilibrio entre resolución de datos y capacidad de memoria; es decir según se aumenta la resolución de los datos tomando más muestras, se requiere más memoria para mantener los datos. No obstante, en otras realizaciones, la resolución podría ser sustancialmente cada: 1 segundo, 10 segundos, 15 segundos, 20 segundos, 30 segundos, 45 segundos, 1 minuto, 2,5 minutos (o de hecho, cualquier período entre medias de estos periodos). Esta manera, dentro de la memoria del dispositivo hay creado un registro de los paraderos del dispositivo 200 en puntos en el tiempo.

En algunas realizaciones, se puede encontrar que la calidad de los datos capturados se reduce según aumenta el período y aunque el grado de degradación será dependiente al menos en parte de la velocidad a la que está moviéndose el dispositivo de navegación 200 un periodo de aproximadamente 15 segundos puede proporcionar un límite superior adecuado.

Además, el procesador 202 se dispone, de vez en cuando, a actualizar el registro de los paraderos del dispositivo 200 (es decir los datos GPS y la marca de tiempo) a un servidor. En algunas realizaciones en las que el dispositivo de navegación 200 tiene un canal de comunicación permanente, o al menos generalmente presente, que lo conecta al servidor la carga de los datos ocurre de una forma periódica que por ejemplo puede ser una vez cada 24 horas. Los expertos apreciarán que son posibles otros períodos y pueden ser sustancialmente cualquiera de los siguientes períodos: 15 minutos, 30 minutos, cada hora, cada 2 horas, cada 5 horas, cada 12 horas, cada 2 días, semanalmente, o cualquier tiempo entre medias de estos. De hecho, en tales realizaciones del procesador 202 se puede disponer para cargar el registro de los paraderos de una forma sustancialmente en tiempo real, aunque esto puede significar inevitablemente que los datos se transmitan de hecho de vez en cuando con un periodo relativamente corto entre las transmisiones y por tanto se puede considerar más correctamente como que es en seudo tiempo real. En tales realizaciones en seudo tiempo real, el dispositivo de navegación se puede disponer a almacenar temporalmente las posiciones GPS dentro de la memoria 214 y/o en una tarjeta insertada en el puerto 228 y trasmitir éstas cuando se ha almacenado un número predeterminado. Este número predeterminado puede ser del orden de 20, 36, 100, 200 o cualquier número entre medias. Los expertos apreciarán que el número predeterminado está regulado en parte por el tamaño de la memoria 214/tarjeta dentro del puerto 228.

En otras realizaciones, que no tienen un canal de comunicación presente de manera general el procesador se puede disponer para cargar el registro en el servidor cuando se crea un canal de comunicación. Esto por ejemplo, puede ser cuando el dispositivo de navegación 200 está conectado a un ordenador del usuario. De nuevo, en tales

realizaciones, el dispositivo de navegación se puede disponer para almacenar temporalmente las posiciones GPS dentro de la memoria 214 o en una tarjeta insertada en el puerto 228. Al llegar a estar llena de posiciones GPS la memoria 214 o tarjeta insertada en el puerto 228 el dispositivo de navegación se puede disponer para borrar las posiciones GPS más antiguas y por tanto se puede considerar como un almacenador temporal Primero en Entrar Primero en Salir (FIFO).

En la realización que se describe, el registro de los paraderos comprende una o más trazas con cada traza que representa el movimiento de ese dispositivo de navegación 200 dentro de un período de 24 horas. Cada periodo de 24 horas se dispone coincidente con un día natural en pero en otras realizaciones, éste no necesita ser el caso.

El servidor 150 está dispuesto para recibir el registro de los paraderos del dispositivo y almacenar éstos dentro un almacenamiento de datos masivo para su procesamiento. De esta manera, según pasa el tiempo el almacenamiento de datos masivo acumula una pluralidad de registros de los paraderos de los dispositivos de navegación 200 que tienen datos cargados. De manera general, el servidor recogerá los paraderos de una pluralidad de dispositivos de navegación 200.

El servidor entonces se puede disponer para generar datos de velocidad a partir de los paraderos recogidos del o cada dispositivo. Estos datos de velocidad pueden comprender un perfil de velocidad que varía con la hora que muestra cómo varía con el tiempo la velocidad media a lo largo de los segmentos navegables de un mapa.

La Figura 7 muestra un ejemplo de un mapa electrónico que, en la realización que se describe, se ve en un dispositivo de navegación a través de una red que en este caso es Internet. Un primer paso en cualquiera de las realizaciones que se describen en la presente memoria es la generación de tal mapa electrónico 1900. El mapa se puede ver en: http://routes.tomtom.com/?Lid=1#/map/?center=55.01%2C-3.445&zoom=4&map=basic

A fin de que el mapa electrónico de la Figura 7 sea usado para propósitos de encaminamiento tiene una serie de nodos dentro del mismo que no se ven típicamente por un usuario del mapa electrónico. Por tanto, los nodos, etc. pueden ser referidos típicamente como datos de mapa. No obstante, algunos de los nodos 300 se muestran en la Figura 6 por facilidad de comprensión. Estos nodos se proporcionan en intersecciones, regiones finales de segmentos navegables, en este caso segmentos de carretera, etc. y se usan para propósitos de encaminamiento. Cuando un usuario pide a su dispositivo de navegación planificar una ruta entre dos o más puntos el dispositivo de navegación planifica la ruta entre los nodos relevantes del mapa. Por tanto, los datos del mapa se pueden usar para generar una ruta según al menos un criterio fijado por un usuario del dispositivo de navegación.

Los expertos apreciarán que el mapa electrónico también comprende nodos adicionales que proporcionan la forma de los segmentos de carretera, la ubicación de entradas a edificios, etc.

De esta manera cada nodo 300 tiene al menos un único, y generalmente una pluralidad, de segmentos de carretera que un usuario puede recorrer emanando de los mismos. Por ejemplo, el nodo 302 tiene cuatro de tales segmentos de carretera 304a, b, c, d. En otras realizaciones, los segmentos navegables pueden representar los segmentos de trayectos que no son carreteras, tales como aceras, carriles de bicicleta, canales, segmentos de ferrocarril, pistas, o similares. Por comodidad no obstante se hace referencia a un segmento de carretera. De esta manera, el mapa que se muestra en la Figura 7 muestra a un usuario del mismo una pluralidad de segmentos de carretera, cada uno de los cuales representa una parte de una ruta navegable en el área cubierta por el mapa.

Los métodos de planificación de ruta (tales como el método Dijkstra, el método A* o los métodos de Moore/Pape) son conocidos. No obstante, éstos pueden ser prohibitivamente lentos en términos de tiempos de cálculo. Un ejemplo de cómo se puede aumentar la velocidad de tal planificación de ruta se muestra en la US 6 636 800, la enseñanza de la cual se incorpora por este medio por referencia.

Las realizaciones que se describen en la presente memoria generan, en un paso de procesamiento previo, un denominado fichero lateral que contiene datos de coste mínimo que se usan por el dispositivo de navegación para acelerar la generación de una ruta cuando se procesa el mapa electrónico. La información se puede mantener como datos binarios en lo que se puede denominar vector de bit, es decir una cadena de ceros y unos. Por tanto, el fichero lateral también se puede considerar como datos de mapa pero el fichero lateral se puede suministrar con o sin el mapa electrónico (por ejemplo, como se muestra en la Figura 7). De esta manera, algunas realizaciones de la invención pueden proporcionar los datos de mapa como un mapa separable de un fichero lateral, mientras que otras realizaciones de la invención pueden combinar los datos de fichero lateral y mapa.

No obstante, los expertos apreciarán que si se usa un fichero lateral entonces se debería usar con el mapa electrónico para el que fue generado el fichero lateral. Si esto no se realiza entonces es concebible que se obtendrán rutas incorrectas (por ejemplo rutas que no minimizan la fórmula de coste fijada por un usuario del mismo).

También, diferentes realizaciones de la invención pueden especificar diferentes parámetros para la generación de los datos de coste mínimo (en esta realización una serie de vectores de bit). Por tanto, si la planificación de ruta posterior usando los datos de mapa generados usa diferentes parámetros de los usados para crear los datos de

coste mínimo es poco probable que los datos de coste mínimo sean útiles para esa planificación de ruta.

Por ejemplo, algunas realizaciones pueden generar datos de coste mínimo para el viajar a través del mapa en coche. Si se usa una planificación de ruta posterior para generar una ruta para caminar entonces es poco probable que los datos de coste mínimo específicos de coche sean útiles. En otro ejemplo, algunas realizaciones pueden generar los datos de coste mínimo suponiendo que un usuario está feliz viajando a lo largo de una autopista, carretera de peaje, etc. Si, en la planificación de ruta posterior, un usuario solicita una ruta que utilice autopistas, carreteras de peaje, etc. entonces los datos de coste mínimo es poco probable que sean útiles.

Algunas realizaciones la invención pueden generar una pluralidad de conjuntos de datos de coste mínimo cada uno que tiene un conjunto diferente de criterios predeterminados. Por ejemplo, las realizaciones de la invención pueden generar aproximadamente cualquiera de las siguientes: 2, 3, 4, 5, 6, 10, 15, o más conjuntos de datos de coste mínimo.

De esta manera, en algunas realizaciones y para generar el fichero lateral, en un paso de procesamiento previo de mapa los nodos se dividen en una pluralidad de regiones y por tanto, cualquier mapa se divide en un número conocido de regiones - por ejemplo N - y se determinan los trayectos de coste mínimo entre las regiones. Este procesamiento previo genera datos que se pueden utilizar junto al mapa a fin de aumentar la velocidad de los métodos de planificación de ruta.

Típicamente el procesamiento previo se realiza antes de que se usen los datos de mapa con independencia de si los datos de mapa van a ser usados en un sitio web o en un dispositivo tal como un PND. Por tanto, el paso de procesamiento previo a menudo se conoce como un proceso lateral de servidor.

Aunque cualquier dispositivo informático general sería adecuado para realizar el procesamiento previo, los expertos apreciarán que cuanto mayor es el rendimiento del aparato, más rápido se realizará el procesamiento previo. Típicamente se utilizará un dispositivo informático de arquitectura X86 para el procesamiento previo. Tal dispositivo de arquitectura X86 típicamente ejecutará un sistema operativo (OS) tal como Microsoft™ Windows™, UNIX, LINUX, OSX™ etc. No obstante, otras realizaciones pueden usar otras plataformas informáticas tales como una arquitectura RISC.

También, como se trató un en otra parte, el procesamiento previo se puede realizar en paralelo y por tanto se puede realizar en una pluralidad de dispositivos informáticos, o al menos en una pluralidad de núcleos de procesador (que pueden ser núcleos de procesador virtuales o reales).

Como siguiente paso de procesamiento previo, cada segmento de carretera (por ejemplo 304 a-d) dentro de una región se procesa para determinar si es parte de un trayecto de coste mínimo para cada una del número de regiones (N) dentro del mapa y se genera un vector de bit (la valoración de coste mínimo). De esta manera, el vector de bit, para cada segmento de carretera dentro de una región, comprende un bit para cada región del mapa. Es decir el vector de bit comprende N -1 bits (1 para cada región menos la región en cuestión) que se fijan o bien a 0 o bien a 1 dependiendo de si ese segmento de carretera forma parte del trayecto de coste mínimo para la región representada por el bit. Algunas realizaciones, pueden añadir bits adicionales para proporcionar información adicional tales como cabeceras, áreas de visibilidad, áreas contiguas, y similares.

Los expertos apreciarán que en este sentido el trayecto de coste mínimo se puede determinar frente a un número de diferentes criterios de coste. Por ejemplo el coste mínimo se puede considerar frente a cualquiera de los siguientes criterios: la distancia más corta; el tiempo de viaje más corto; el menos caro (en términos de impacto ambiental); el que usa menos gasolina; el de menos CO_2 generado, etc. En la realización actual se considera el coste mínimo frente al tiempo de viaje más corto.

De esta manera, los expertos apreciarán que para un mapa que cubre un área significativa N es probable que sea grande. De esta manera, el procesamiento previo puede tardar una cantidad significativa de tiempo.

De esta manera, en un mapa ejemplo que cubre Europa Occidental y Central podría haber típicamente 40.000.000 de nodos que tienen 100.000.000 de segmentos de carretera. Supongamos que hay 100 regiones dentro del mapa entonces N es igual a 100 y hay 100.000.000 x 100 (N); es decir se necesitan $10x10^9$ bits para almacenar los 99 (es decir N-1) bits para cada segmento de carretera del mapa. Usando realizaciones de la invención descrita más adelante, tal mapa puede usar aproximadamente 500 Mb de almacenamiento.

La WO2009/053410 trata de un método que asigna perfiles de velocidad que dan la velocidad a lo largo de un segmento navegable de un mapa como una función del tiempo. Por tanto, para propósitos de encaminamiento, si un segmento navegable constituye o no parte de la ruta más rápida en otra región variará como una función del tiempo. Es decir, según aumenta/disminuye la densidad del tráfico, por ejemplo en horas punta y similares, llegará a ser menos/más deseable respectivamente usar un segmento navegable.

Algunas realizaciones pueden almacenar una pluralidad de vectores de bit para cada segmento de carretera. Los expertos apreciarán que cuanto mayor es la resolución a la que se graba la velocidad mayor será el número de

vectores de bit requeridos. No obstante, otras realizaciones pueden hacer una valoración de la ruta más rápida utilizando una función que varía con la hora mucho como se describe en la WO2009/053410.

Cuando se planifica una ruta a través de los mapas electrónicos se entiende que es necesario ser capaz de hacer esto usando una gradualidad temporal de una precisión de aproximadamente centésimas de segundo; es decir ser capaz de especificar la hora de salida desde un nodo con una precisión de 0,01 segundo a fin de determinar correctamente la ruta de coste más bajo.

De esta manera, cuando se considera este nivel de precisión se apreciará que para el mapa considerado anteriormente que tiene 40.000.000 de nodos y 100.000.000 de segmentos de carretera tiene $100.000.000^2$ de rutas posibles a través del mismo. Cuando se considera además la dimensión temporal el número de la rutas aumenta más: 7 (es decir días por semana) x 24 (es decir horas por día) x 3600 (es decir segundos por hora) x 100 (es decir centésimas de segundo por segundo) x $100.000.000^2$ = 604.800.000.000.000.000.000.000 (sextillones) de rutas posibles. En los niveles de procesamiento actual y usando un planteamiento ingenuo en el que cada segmento se explora en cada intervalo de tiempo tardaría 19.178.082.191.780.821 años. El mapa de Europa Occidental y Central que el solicitante vende actualmente tiene 120.000.000 segmentos de carretera aumentado por ello más esto.

De esta manera, se apreciará que se requiere una cantidad significativa de datos a fin de almacenar y procesar los datos en las etapas de procesamiento previo.

Por tanto, las realizaciones de la presente invención usan técnicas a fin de reducir significativamente la cantidad de procesamiento previo. Estas técnicas se describen ahora en más detalle y algunas realizaciones de la invención pueden utilizar todas, mientras que otras realizaciones pueden utilizar solamente algunas de las técnicas.

Diferentes realizaciones de la invención pueden usar combinaciones diferentes de los rasgos perfilados más adelante. Por tanto, algunas realizaciones pueden utilizar todas las técnicas descritas para reducir la cantidad de información. En el otro extremo, otras realizaciones pueden no utilizar ninguna de las técnicas para reducir los datos.

Algunas realizaciones de la invención no calculan los vectores de bit para los elementos de carretera que no cumplen un criterio predeterminado - como se muestra en el paso 1902. Por ejemplo, segmentos de carretera pueden no tener vectores de bit calculados para ellos si no es posible viajar a lo largo del segmento de carretera en una clase de transporte predeterminado.

De esta manera, en un ejemplo donde se fija a un coche el criterio de modo de transporte los siguientes segmentos de carretera pueden no tener vectores de bit calculados para ellos:

- ferrocarriles;
- segmentos presentes en datos de mapa que no corresponden a segmentos de carretera;
- una carretera de un sentido en la dirección equivocada (ilegal);
- segmentos de carretera que no son navegables en una forma de transporte (por ejemplo zonas peatonales, aceras, cuando se considera conducción en coche);
- segmentos de carretera en un punto de no decisión (es decir no se aleja del segmento de carretera);

Reducción de red

Una técnica es reducir el tamaño de la red que se considera para la valoración en cuanto a si los segmentos de carretera forman parte de un trayecto de coste mínimo; reducir el número de segmentos de carretera reduce la cantidad de tiempo que lleva hacer el cálculo como se muestra en el paso 1904.

Para ilustrar esta técnica, la red de carreteras que se muestra la Figura 7 también se muestra la Figura 6 pero con segmentos de carretera que no son útiles para la valoración de coste mínimo borrados de la misma. Esta reducción de la red aspira a proporcionar la subred más pequeña en la que se debería hacer la valoración de coste mínimo.

En términos matemáticos la red que se deja, y que se ilustra en la Figura 7, se puede describir como el componente bi conectado más grande de la representación no dirigida del componente de intensidad más grande de la red de carreteras de la Figura 7. La red central se determina aplicando técnicas estándar para ambas conectividades intensas (también conocidas como bi conectividad) con adaptación a las regulaciones del tráfico rodado tales como restricciones de giro y acceso local. Los expertos apreciarán que una conectividad intensa indica que el viaje es posible en ambas direcciones entre los nodos (es decir desde A → B así como B → A).

De esta manera, con referencia a ambas Figuras 5 y 7 se puede ver que las calles sin salida tales como los fondos de saco 250, bucles redundantes 252, etc. se han borrado. No obstante se verá que un nodo (por ejemplo 400 en la

Figura 7), permanece en el origen de los segmentos de carretera que han sido eliminados. Esto es por las razones tratadas en lo sucesivo.

Posterior a la reducción de la red central (por ejemplo como se muestra en la Figura 7), la red central se divide para proporcionar las regiones descritas anteriormente. En la realización que se describe, la red se divide según separadores de arco de múltiples vías. Los expertos apreciarán que en tal red si se eliminan los segmentos de carretera (es decir los arcos) que se dividen por la mitad por los límites de región entonces los segmentos de carretera que permanecen en cualquier en cualquier región no se deberían conectar con ninguna otra región.

División de red - paso 1906

No obstante, antes de que se realice la división se hacen los siguientes pasos:

1. Las islas se determinan y dividen independientemente. Las islas tienden a estar separadas físicamente de otras partes de la red y se enlazan por ello por medios tales como enlaces de transbordador, etc. Éstos se comportan de manera diferente que los segmentos de carretera típicos (es decir las velocidades medias son significativamente inferiores, etc.) y si tales enlaces se incluyen en la división entonces el método no se realiza tan bien como puede esperarse.

2. Los cruces, rotondas y otras confluencias, que se representan por más de un nodo en la red se contraen de manera que los nodos que pertenecen al mismo cruce, etc. no finalicen en regiones diferentes.

3. Los trayectos simples (es decir conexiones entre dos nodos sin posibilidades de giro) donde todos los elementos navegables comparten el mismo conjunto de características (por ejemplo transbordador, No de conducción, No atravesables, etc.) se contraen para reducir el tamaño de entrada de la red pasada al sistema de división. Por ejemplo, mirando la Figura 6, el nodo 308 puede caer sobre la carretera 310.

Las realizaciones de la invención que se describen emplean un planteamiento de divide y vencerás que se perfila en http://labri.fr/perso/pelegrin/scotch. Los métodos perfilados en la presente memoria dividen el mapa para comprender un número de regiones que no están basadas en un área del mapa sino que se basan en el número de nodos contenidos dentro de una región. Tener nodos que están así organizados tiene la ventaja de que puede ayudar a hacer el encaminamiento usando las regiones más eficientes debido a que ayuda a entregar regiones que tienen diámetros de búsqueda local similares.

Cada nodo tiene números de identidad asociados con el mismo incluyendo un ID de región y por tanto, nodos en la misma región tienen los mismos números de ID de región. Los nodos pertenecen a regiones en niveles jerárquicos L. El nivel 0 es por convenio el nivel más tosco (usado para encaminamiento de larga distancia). El nivel L -1 es el nivel detallado.

En lugar de establecer un número absoluto de nodos a estar contenidos dentro de cualquier región es conveniente ajustar un tope en el número máximo de nodos y permitir alguna flexibilidad por debajo de éste. Por ejemplo, el método puede especificar un número máximo de 100 nodos por región lo que podría provocar una dispersión de nodos por región de típicamente 60 a 100 nodos. Un tope se muestra ventajoso por encima de un número fijo de nodos dado que un número fijo puede provocar regiones de formas forzadas que a su vez conduce a un encaminamiento por debajo del óptimo cuando se usan las regiones.

Múltiples niveles

En la realización que se describe, se realiza la división para proporcionar una pluralidad de niveles de regiones anidadas las cuales se ejemplifican conceptualmente en la Figura 9. Quedará claro a partir de la Figura 9 que existe una naturaleza jerárquica con el nivel más bajo (0) que se subdivide para proporcionar el nivel 1, que a su vez está subdividido para proporcionar el nivel 2.

El número de nodos en cada región varía entre niveles y los niveles más toscos (que tienden a cubrir un área geográfica más grande) tienen más nodos dentro de los mismos que los niveles inferiores. Por tanto, el nivel más tosco en la realización que se describe (por ejemplo el nivel 0) se usa típicamente para propósitos de encaminamiento de larga distancia mientras que los niveles de más detalle (por ejemplo el nivel 2) se usan para encaminamiento de corta distancia.

Una parte del vector de bit para cada uno de los segmentos de carretera comprende bits para cada uno de los niveles. El número de bits es el número de regiones en cada nivel.

Para dar una idea de la cantidad de datos a codificar (en una realización típica el número de niveles sería típicamente L = 3):

- nivel global = 0 tendría típicamente 100 regiones.
- nivel intermedio = 1 tendría típicamente 10 regiones para cada región de nivel cero (es decir 100 x 10).
- nivel más detallado = 2 tendría típicamente 5 regiones para cada región de nivel 1 (es decir 100 x 10 x 5).

El uso de niveles jerárquicos es ventajoso dado que reduce, quizás significativamente, la cantidad de procesamiento que se requiere. En la realización actual que se describe 100x10x5 regiones (es decir 5000 regiones). De esta manera, en una estructura plana, que tiene el mismo número de regiones requeriría métodos perfilados en la presente memoria para referirse a 5000 regiones. No obstante, en las realizaciones descritas en la presente memoria se pueden reducir esto refiriéndose a nodos en el nivel adecuado.

El número de niveles y el número de regiones por nivel típicamente se ajusta una vez que se conozcan diversos factores para un mapa. Por ejemplo, cuánto almacenamiento se puede asignar al mapa y la velocidad a la que en se debería realizar el encaminamiento. Los expertos apreciarán que hay un compromiso en que según se aumenta la cantidad de procesamiento previo entonces mayor llega a ser el tamaño del mapa para soportar los datos, pero se pueden calcular rutas más rápidas usando ese mapa.

En una realización, hay 3 niveles de regiones. No obstante, en otras realizaciones puede haber cualquier número de niveles. Por ejemplo, puede haber aproximadamente cualquiera de los siguientes números de niveles: 1, 2, 4, 5, 6, 7, 8, 9 o 10.

De esta manera, usando el ejemplo anterior, un mapa de Europa Occidental y Central que típicamente tiene 40.000.000 de nodos y 100.000.000 de segmentos de carretera, usando la codificación más ingenua usaría una codificación de tamaño fijo para ID de región para cada nodo en cada nivel, y un vector de bit de tamaño fijo (marcas arp) para cada segmento de carretera en cada uno de los niveles. De esta manera, el tamaño de tal codificación básica se puede calcular fácilmente como sigue:

- cada ID de región de nodo en el nivel 0 usaría log2(100) bits = 7 bits
- cada ID de región de nodo en el nivel 1 usaría log2(10) bits = 4 bits
- cada ID de región de nodo en el nivel 2 usaría log2(5) bits = 3 bits
- cada vector de bit (marcas arp) en el nivel 0 usaría 100 bits (100 regiones menos 1 para la región actual)
- cada vector de bit (marcas arp) en el nivel 1 usaría 10 bits (10 regiones menos 1 para la región actual)
- cada vector de bit (marcas arp) en el nivel 2 usaría 5 bits (5 regiones menos 1 para la región actual)

La Figura 9 muestra un nivel más tosco de la región 600, que en la Figura proporciona 6 regiones 1-6. Cada una de estas regiones se subdivide en regiones adicionales, en esta realización 9 subregiones, como se representa por el segmento de carretera discontinuo dentro del nivel más tosco de la región 600. En la realización que se describe se usa un nivel adicional de subdivisión en el que también se subdivide cada una de las regiones proporcionadas por los segmentos de carretera discontinuos, proporcionando de esta manera tres niveles de división pero éstos no se muestran en la Figura 9 para facilidad de referencia. Otras realizaciones pueden usar por supuesto más niveles (por ejemplo 4, 5, 6, 7, o más niveles) o menos niveles (por ejemplo 1 o 2 niveles).

De esta manera, las realizaciones de la invención introducen las denominadas áreas de visibilidad para regiones de nivel de más detalle paso 1908. Un área de visibilidad de una región k es un conjunto de (k -1) regiones, donde la región es distinguible en su propio bit en los conjuntos de marcas. Naturalmente, el área de visibilidad siempre incluye las (k -1) regiones a las que pertenece la región k. Adicionalmente, puede contener algunas regiones (k -1) contiguas. Las áreas de visibilidad van a ser calculadas durante el procesamiento previo y almacenadas en asociación con el vector de bit.

La relación entre las regiones k y sus áreas de visibilidad de nivel (k -1) también se puede ver desde el lado opuesto: cada región (k -1) conoce sus alrededores que constan de regiones de nivel k que están en las inmediaciones. Visto de esta manera, se puede ver que los conjuntos de marcas de los segmentos de carretera no son más de una longitud fija en todo el mapa completo; en su lugar, los vectores de bit en cada región 1 tienen su longitud específica A + B + C + Afueras 0 + Afueras 1. Donde A se refiere al nivel más tosco de regiones, B se refiere a las regiones de granularidad intermedia y C se refiere a la granularidad de más detalle de regiones (en una realización en la que hay tres niveles).

El procesamiento previo posterior calcula las áreas de visibilidad anterior a los cálculos del trayecto más corto para generar los vectores de bit. El método de encontrar el área de visibilidad se puede describir como sigue:

Para cada región k, iniciar una primera búsqueda de amplitud en el gráfico de proximidad de la región; entonces para cada región k visitada, que está lo bastante cerca de la región de inicio, añadir su región (k-1) de contenido

al área de visibilidad.

La relación de “cercanía” debería tener en cuenta tanto la métrica geográfica (por ejemplo la distancia entre los puntos medios de la región) como la distancia teórica del gráfico. (De esta manera, una región puede estar “cerca” si está geográficamente lejos pero enlazada a la región actual mediante conexiones rápidas. Del mismo modo, una región se puede considerar como “distante” incluso si está cerca geográficamente si es difícil viajar entre regiones).

Los valores umbral exactos pueden estar sometidos a ajuste experimental, ya que dependen de características específicas de mapa como el diámetro medio de áreas metropolitanas, que son más susceptibles de los efectos negativos descritos anteriormente tales como aumento de procesamiento previo, etc. Los segmentos navegables con tiempo de viaje extremadamente largo (como transbordadores o segmentos de carretera de No de conducción) se ocultan durante el recorrido de gráfico de adyacencia, de esta manera las áreas de visibilidad están siempre confinadas a islas únicas o pequeños archipiélagos que pertenecen a la misma región.

De esta manera, las regiones visitadas durante la primera búsqueda de amplitud constituyen los alrededores de *R*. El conjunto de inclusión mínimo de las regiones del siguiente nivel más tosco que cubre completamente los alrededores se llama área de visibilidad de *R*. La relación inversa se llama inmediaciones: la lista de inmediaciones de una región Q (en el nivel L) consta de todas las regiones en el nivel L+1 que tienen a *Q* en sus áreas de visibilidad.

Para dar un ejemplo específico y con referencia a la Figura 9, ejemplos de áreas de visibilidad son los siguientes:

Si decidimos que la distancia mínima entre cada región 1 y la frontera de su área de visibilidad debería ser al menos 1, entonces las áreas de visibilidad para algunas regiones 1 seleccionadas constarán de las siguientes regiones 0 (la propia región 0 se puede omitir, ya que siempre está allí):

- 15:

  (es decir la región de nivel 1 nº 15 no tiene áreas de visibilidad dado que está en el centro de su región de nivel 0).

- 28: 5

  (es decir la región de nivel 1 nº 28 tiene una única región de visibilidad que es la región nº 5 (nivel 0))

- 37: 2, 5, 6

  (es decir la región de nivel 1 nº 37 tiene tres regiones de visibilidad que son las regiones de nivel 0, 2, 5 y 6)

Además de considerar las contiguas de nivel 0 de una región de nivel 1, las contiguas de nivel 1 también se determinan para cada región de nivel 0. De esta manera como ejemplo,

- 1: 21, 24, 27, 28, 41, 42, 43, 51

  (es decir para la región de nivel 0 nº 1 las regiones de nivel 1 son: 21, 24, 27, 41, 42, 43, 51).

  De esta manera, en este ejemplo, se puede ver que la región 28 está enumerada a pesar de no estar en la columna de más a la izquierda de la región 2. Esto por ejemplo pudiera ser debido a que la región 28 tiene enlaces rápidos a la región 1 y por tanto está cerca cuando se considera en términos de tiempo más que de distancia. Los expertos apreciarán que la cercanía se puede considerar frente a otras métricas tales como la ecología (menos CO_2 o similar), etc.

Codificación de listas contiguas

La Figura 9a muestra más detalle de los niveles que se pueden emplear en comparación con la Figura 9. El nivel 0 es el nivel más tosco y se puede considerar como el nivel k-1. Por facilidad, en la Figura 9a solamente se muestran las regiones 1 a 4 (es decir 602; 604; 606; 608). Cada una de estas regiones k-1 se divide además en 9 regiones (1-9) que se pueden conocer como regiones de nivel k o regiones de nivel 1. Además, cada una de estas regiones de nivel 1 se divide en 4 regiones de nivel k+1 (es decir regiones de nivel 2).

Generación de vector de bit

Una vez que la red se ha dividido en los tres niveles de regiones se procesa para determinar el trayecto de coste

mínimo para cada una de las regiones y se crea un vector de bit para cada segmento de carretera dentro de una región paso 1910. De esta manera, como se trató anteriormente cada segmento de carretera dentro de cualquier región se analiza con respecto a una fórmula de coste para determinar si es parte de un trayecto de coste mínimo para cada otra región. Se genera un vector de bit para cada segmento de carretera en una región como se muestra en la Figura 7 la cual, por facilidad de comprensión, muestra un vector de bit simplificado.

También se debería apreciar que se determina el trayecto de coste mínimo, no sólo un único periodo de tiempo, sino que se calcula para una pluralidad de periodos de tiempo como se describe en lo sucesivo.

Este cálculo se realiza para la red mostrada en la Figura 7 (es decir la red reducida). Las partes de la red que se han eliminado caen eficazmente sobre el nodo desde el cual se originan. Por ejemplo, la región 250 mostrada en la Figura 5 cae sobre el nodo 400 en la Figura 7.

Se verá que cada vector de bit comprende tres columnas: una columna de más a la izquierda 700 que contiene un número de identidad para el nodo en el inicio del segmento de carretera en consideración; una segunda columna 702 que contiene el número de identidad para el nodo al final del segmento de carretera en consideración y una tercera columna 704 que contiene el vector de bit para ese segmento de carretera. De esta manera, se verá que cada segmento de carretera se identifica por dos nodos, uno en cada extremo del mismo.

Se puede usar cualquier método de encaminamiento adecuado para determinar si un segmento de carretera es parte de la ruta de coste más bajo. En esta realización particular el método Dijkstra bien conocido se usa para explorar la red entera.

No obstante, se puede reducir la cantidad de tiempo de procesamiento empleando diversas estrategias. Los expertos apreciarán que las técnicas para reducir la red descrita anteriormente también reducirán la cantidad de tiempo de procesamiento. Las realizaciones de la invención pueden emplear una cualquiera o más de las siguientes:

- calcular todas las entradas de vectores de bit que corresponden a una región de una vez. El árbol de trayecto más corto en el gráfico inverso para cada nodo límite. Tal planteamiento es ventajoso ya que cada región se puede procesar en paralelo disminuyendo por ello el tiempo de procesamiento.
- reducir los re cálculos de sub árboles de trayecto más corto similares.
- no volver a calcular los vectores de bit que ya han sido generados.

De esta manera, en resumen una realización puede realizar lo siguiente a fin de generar los vectores de bit:

Pasos de preparación:

1. Como los expertos apreciarán la búsqueda a través del mapa electrónico se puede representar por un gráfico acíclico dirigido (DAG). Este gráfico y las estructuras de datos contiguas (tablas de extensión largas y conversión de costes) se invierten.

2. Los segmentos de carretera simples con respecto al nivel de más detalle se contraen (es decir los nodos extremos caen sobre otro como se trató en otra parte). Un segmento de carretera se denomina *simple* si consta de uno o más nodos de grado = 2, todos que están en la misma región del nivel dado, y los segmentos navegables tienen atributos idénticos: por ejemplo “es un transbordador”, “es No Atravesable”, y “es No de Conducción”.

3. Para cada región, los segmentos de carretera de la red de carreteras se clasifican en tres grupos: segmentos de carretera *salientes*, *entrantes*, e *interiores*, dependiendo de si el nodo de cabecera (es decir el nodo_ desde) y/o de cola (es decir a nodo_a) pertenece a la región. Es decir, si tanto la cabecera como la cola están dentro de la misma región el segmento de carretera se denomina segmento de carretera interior; si la cabecera está dentro de la región pero la cola no está entonces el segmento de carretera se denomina segmento de carretera saliente; y si la cola está en la región pero la cabecera no está entonces el segmento de carretera se denomina segmento de carretera entrante.

4. Se ponen en todos los segmentos de carretera restricciones de giro especiales que salen de las áreas No Atravesables y No de Conducción.

La rutina de procesamiento previo procede de manera ascendente, comenzando con el nivel de división de más detalle – es decir el nivel 2 en la realización que se describe. En cada nivel de regiones, se repiten los siguientes pasos para cada región *R*:

1. Determinar el área de exploración de *R*. En el nivel más alto (por ejemplo la región 0 en la realización que se describe) es el gráfico entero, en los niveles de más detalle es el sub gráfico confinado por el *área de visibilidad* de *R* (como se describió anteriormente).

De esta manera, en el nivel intermedio (es decir el nivel 1) se realizan solamente los siguientes pasos para el área de visibilidad de la región de nivel 0 en la que están contenidas esas regiones de nivel 1. En el nivel 0, que se debería recordar que se usa para encaminamiento de larga distancia, se consideran los segmentos de carretera del gráfico entero.

Reunir los segmentos de carretera entrantes del área de visibilidad, es decir, los segmentos de carretera que conducen desde los nodos fuera del área dentro del área. Entonces reunir los segmentos de carretera *frontera*, es decir, los segmentos de carretera *siguientes* a los segmentos de carretera entrantes. El segmento de carretera L 2 se llama *siguiente* a L 1 (y L 1 *precedente* de L 2) si cabecera (L 1) = cola(L 2) y el giro desde L 1 en L 2 no está prohibido. Los cruces complejos se contraen a nodos únicos con el propósito de encontrar los segmentos de carretera frontera. Los segmentos de carretera de transbordador y segmentos de carretera con un atributo “No de Conducción” se consideran como segmentos de carretera frontera.

Reunir los segmentos de carretera entrantes de esta manera puede reducir, quizás significativamente, la cantidad de procesamiento requerida en los pasos de exploración descritos en lo sucesivo. El(los) paso(s) de exploración puede(n) reducir la exploración del gráfico para considerar esas rutas en una región dada que incluye una ruta entrante.

2. Para cada segmento de carretera saliente de *R*, encontrar los segmentos de carretera raíz y el número de pasos de exploración. Si el segmento de carretera saliente es un único sucesor de al menos uno de sus predecesores que están dentro de *R*, este segmento de carretera saliente es el único segmento de carretera raíz, y se realiza un único paso de exploración. De otro modo, si el segmento de carretera saliente es bidireccional (es decir, conducible en ambas direcciones), entonces el segmento de carretera en sí mismo y su segmento de carretera opuesto (entrante) se toman como segmentos de carretera raíz de un único paso de exploración. De otro modo, si el segmento de carretera saliente es unidireccional, entonces cada segmento de carretera interior precedente (y, si está presente, su segmento de carretera opuesto) se toma como un segmento de carretera raíz para un paso de exploración separado. Finalmente, con independencia de la clase de segmento de carretera saliente, para cada trayecto de extensión de tráfico que se ejecuta a través del segmento de carretera saliente, el segmento de carretera de inicio de la extensión (si está dentro de *R*) se toma como un segmento de carretera raíz para un paso de exploración separado.

   En niveles de más detalle (es decir todos los niveles excepto el nivel 0), los segmentos de carretera de transbordador salientes se tratan especialmente. Como se señaló anteriormente, los segmentos de carretera de transbordador se ignoran cuando se determinan los alrededores de la región. Si la región de cabecera de un segmento de carretera de transbordador no pertenece al área de visibilidad de *R*, entonces se realizará un único paso de exploración con el segmento de carretera de transbordador que es el único segmento de carretera raíz y el área de exploración que se restringe a la región de cabecera en sí misma.

3. Realizar los pasos de exploración programados (descritos más adelante).

4. Rastrear los trayectos más cortos desde los segmentos de carretera dentro del área de exploración a los segmentos de carretera raíz. En todos los niveles excepto el más alto (es decir el Nivel 0), la ordenación de los segmentos de carretera en los que la traza respectiva comienza se afecta a los resultados. Suponemos que R es una región de nivel L (donde L>0), entonces los segmentos de carretera salientes de regiones de nivel (L-1) se procesan primero, luego los segmentos de carretera salientes restantes en el nivel L, y así sucesivamente hacia el nivel de más detalle; los segmentos de carretera que son interiores con respecto al nivel de más detalle (y no han sido aún eliminados) se procesan finalmente. Siempre que un segmento de carretera se encuentre que ya ha sido visitado en una traza anterior, no hay necesidad de seguir ese trayecto una segunda vez. Mientras se rastrean, los vectores de bit objetivo adecuados se fijan para cada segmento de carretera visitado. En todos excepto en el nivel más alto (es decir el nivel 0), tienen lugar acciones adicionales:

   después de que el trayecto ha cruzado primero algún límite de región de nivel L, ese segmento de carretera límite y todos los segmentos de carretera adicionales en el camino al segmento de carretera raíz se marcan con una marca de segmento de carretera de tránsito;

   los nodos donde el trayecto más corto hace un giro en U se marcan con una marca de giro en U.

   Finalmente, en todos excepto en el nivel de más detalle, se rellenan las entradas de la matriz de correlación.

Después de que se procesan todas las regiones del nivel, el gráfico se simplifica para el siguiente nivel. En primer lugar, se eliminan todos los segmentos de carretera no marcados como segmentos de carretera de tránsito. Entonces se contrae la acumulación de nuevos trayectos simples; se conservan los nodos marcados como giro en U.

Finalmente, los vectores de bit se propagan a los segmentos de carretera eliminados antes de procesar este nivel, según la matriz de correlación.

Matriz de correlación

Algunas realizaciones de la invención usan una matriz de correlación que almacena las relaciones entre las regiones objetivo necesarias para fijar los vectores de bit en los segmentos de carretera. En cada nivel *L* excepto el nivel de más detalle, se crea y usa una nueva matriz de correlación. En el nivel *L*, las filas de la matriz se indexan por regiones de nivel *(L + 1)*, las columnas se indexan por regiones de nivel *L*, y cada entrada de matriz es un conjunto de regiones de nivel *(L + 1)* cero o más. En los niveles inferiores, la mayoría de entradas de matriz son iguales al conjunto vacío, es decir las matrices son escasas.

El propósito de una matriz de correlación es ayudar a ajustar los vectores de bit en segmentos de carretera que han sido borrados en etapas anteriores. Estos segmentos de carretera no están contenidos en los gráficos acíclicos dirigidos resultantes de los pasos de exploración de nivel *L*, pero lo estarían si no han sido borrados. Con más precisión, la matriz de correlación se usa para fijar los vectores de bit a alguna región *S* de nivel *L* (donde *L* no es el nivel de más detalle) en los segmentos de carretera en el área de exploración de *S* que se ha borrado durante los cálculos en algún nivel > L. Para una región *R* de nivel *(L + 1)* contenida en el área de exploración de *S*, el elemento de matriz M[R, S] finalmente será un conjunto de regiones de nivel *(L + 1)* en el límite del área de visibilidad de *R*, de manera que todos trayectos más cortos (en el gráfico invertido) desde *S* a *R* pasen a través de una de esas regiones.

Cuando se crea la matriz, todas las entradas se inicializan al conjunto vacío. Entonces, para todas las regiones *S* de nivel *L*, se realizan las dos acciones siguientes.

1. Creación de matriz: Para cada paso de exploración con un segmento de carretera raíz *I* en o sobre *S* y el gráfico acíclico dirigido resultante *D*, para cada región *R* de nivel *(L + 1)* contenida en el área de exploración de *S*, y para cada segmento de carretera entrante *I'* de *R*, el elemento de matriz M[R, S] se actualiza como sigue:

   Indicamos mediante *A* el área de exploración *R* (que se calculó anteriormente para el nivel *L + 1*). Rastrear el trayecto en *D* desde *I'* de vuelta a *I* y comprobar si deja *A*: si un segmento de carretera en este trayecto va desde una región *T* de nivel *(L + 1)* a una región *T'* de nivel *(L – 1)*, donde *T* aún está contenida en A, pero T' no, entonces T se añade al conjunto M[R, S], y se procesa el siguiente *I'*.

2. Lectura de la matriz: para cada segmento de carretera en el área de exploración de *S* que se ha borrado antes de que comenzasen los cálculos de nivel *L*, permitamos que *R* sea la región de nivel *(L + 1)* donde ese segmento de carretera termina (de nuevo, con respecto al gráfico invertido). Ahora el bit del vector de bit para la región *S* en ese segmento de carretera se fija al OR lógico de los bits del vector de bit para región *T*, donde *T* se cubre todos los elementos de M[R, S] .

Señalar que el bit del vector de bit para cada *T* habrá sido fijado en un nivel anterior o bien directamente desde algún gráfico acíclico dirigido o bien en el procedimiento análogo que implica una matriz de correlación en algún nivel inferior.

Exploración

El paso de exploración consta de construir un gráfico acíclico dirigido de trayectos de coste mínimo arraigado en el segmento de carretera dado (par). Esto se logra usando una variante del método de Dijkstra bien conocido para cálculo de coste mínimo. La función objetivo a ser minimizada se puede elegir libremente, por ejemplo, según el tiempo de viaje o consumo de combustible estimado, o cualquiera de los otros factores tratados en otra parte.

Una realización usa una suma ponderada de tiempo de viaje, duración de trayecto, y otros términos de penalización suprimiendo giros y maniobras indeseados.

Se hacen las siguientes modificaciones al método clásico (Dijkstra):

1. Funciona en el gráfico de segmento de carretera, es decir los elementos que se visitan, relajan, etc. son segmentos de carretera, no nodos. Esto es útil para permitir al método considerar restricciones de giro y/o extensiones de tráfico.

2. La función objetivo en la etiqueta es el vector de pares (tiempo de viaje, coste) para un conjunto fijo de intervalos de tiempo de llegada en el segmento de carretera raíz. Los intervalos de tiempo se eligen de manera que se cubran todos los modos de tráfico relevantes (flujo libre, hora punta de mañana de día

laborable, hora punta por la tarde, etc.).

Ésta se expande además más adelante en la discusión acerca de valorar el coste en una pluralidad de periodos de tiempo.

3. Se puede almacenar más de una etiqueta para un segmento de carretera dado. Si los conjuntos de extensiones de tráfico (inacabadas) en dos etiquetas no son iguales, entonces las etiquetas en sí mismas se llaman independientes y ambas siguen propagándose sobre los segmentos de carretera sucesivos. De otro modo, si la relación entre los valores de función de coste de diferentes intervalos de tiempo de llegada no están alternando, es decir una etiqueta es mejor que otra de manera no ambigua, se descarta la etiqueta peor. De otro modo, se crea una nueva etiqueta fundiendo los mejores valores para cada intervalo de tiempo, la cual se propaga en lugar del original. El conjunto predecesor de la etiqueta fundida entonces es la unión de los conjuntos predecesores de las etiquetas originales.

4. Se crean etiquetas de giro en U especiales en cada segmento de carretera bidireccional. Ellas codifican la posibilidad de iniciar la ruta real (no invertida) en ambas direcciones. Las etiquetas de giro en U no se propagan y no se pueden fundir con etiquetas normales. No obstante, influyen la fase de retro seguimiento cuando se fijan los vectores de bit: se marca el trayecto más corto solamente si la etiqueta de inicio no es peor que la etiqueta de giro en U en el mismo segmento de carretera.

En los niveles de más detalle, donde el área de exploración está restringida a un conjunto de regiones, los segmentos de carretera frontera, que se definieron anteriormente, se observan permanentemente. Tan pronto como se alcanzan todos los segmentos de carretera frontera por la parte delantera de búsqueda, la cresta de observación se crea a partir de los valores de función de coste más grande (= peor) por intervalo de tiempo. Entonces una etiqueta en un segmento de carretera fuera del área de exploración, cuando se extrae del montón, se propaga solamente si su valor de función de coste está por debajo de la cresta de observación actual en al menos un intervalo de tiempo. Si un área de exploración se extiende sobre varias islas, se mantiene una cresta de observación separada para cada isla.

Funciones que varían con el tiempo

Realizaciones de la invención actual calculan el vector de bit mostrando los trayectos de coste mínimo a través de la red en una pluralidad de periodos de tiempo en lugar de en un único momento. Los expertos apreciarán que la ruta de coste más bajo a través de una red de carreteras puede variar con la hora debido a la influencia de la densidad de tráfico, etc. Por lo tanto, para cualquier nodo puede haber dos o más trayectos de coste mínimo, cada uno para una hora diferente. En esta realización, el vector de bit no está codificado con una referencia temporal para cuando son aplicables los trayectos de coste mínimo. El vector de bit se fija simplemente para identificar un segmento navegable como que o bien es parte de un trayecto de coste mínimo o bien no. Por lo tanto, cuando se encamina usando los datos de coste mínimo, el algoritmo de encaminamiento tendrá que considerar todos los trayectos de coste mínimo posibles desde un nodo. Este proceso se describe ahora brevemente con ayuda de la Figura 10a.

En una exploración Dijkstra estándar de una red, según se explora la red, el método usa el coste total incurrido hasta la fecha a obtener para ese punto en la red más el coste esperado aún a ser incurrido.

De esta manera, algunas realizaciones utilizan una función en contraposición a un valor discreto para hacer la evaluación de coste en cada nodo. De esta manera, en la Figura 10a cada nodo (por ejemplo el 750) del gráfico 751 que está siendo explorado tiene una función de coste asociada con el mismo que identifica cómo varía con el tiempo el parámetro que está siendo explorado (por ejemplo tiempo, combustible gastado o similares).

La función de coste se puede asociar con un segmento de carretera en contraposición al nodo.

El método entonces procesa el coste gastado para añadir los costes estimados sumando la función en el nodo actual con el coste que ha sido acumulado hasta la fecha para generar una nueva función. El ejemplo mostrado la Figura 10a en 752 muestra una función de coste que ha sido generada en el nodo 750 mediante el método de búsqueda y muestra cómo varía el tiempo de viaje (eje y) con la hora de salida (eje x). Se verá que la función de coste aumenta en los puntos 754 y 756 debido a las horas punta de la mañana y la tarde.

En una realización particular, la función de coste (por ejemplo la velocidad media en un segmento de carretera) se almacena en intervalos de 5 minutos; es decir es una función cuantificada en lugar de la función que varía continuamente con un periodo de tiempo de 5 minutos.

El vector de bit para un segmento de carretera se fija si ese segmento de carretera es parte de la ruta de coste más bajo en cualquier periodo de tiempo.

Proyección de datos centrales a la red completa

Anteriormente fue descrito como la red contenida en el mapa fue reducida a fin de reducir el número de segmentos de carretera y los nodos que deben ser considerados por el método de división. No obstante, los nodos que fueron borrados en el paso de reducción también se deberían considerar además a fin de que los métodos de encaminamiento aún puedan generar rutas a y desde los nodos y segmentos de carretera borrados.

Por tanto, los nodos y segmentos de carretera borrados están asignados a las mismas regiones que el nodo al que están conectados en la red central.

Comprensión

Como se trató, los vectores de bit son significativos en tamaño y por lo tanto es deseable comprimir la información. Realizaciones de la invención pueden realizar esto de diferentes maneras.

No obstante, una realización utiliza diversas técnicas para comprimir, coalescencia y/o correlación de los vectores de bit seguidos por una codificación de Huffman posterior de los vectores de bit.

De esta manera, algunas realizaciones pueden intentar y asegurar que hay una distribución desigual de vectores de bit dado que esto puede ayudar a asegurar que la codificación de Huffman es más eficiente que de otro modo por el caso.

Por ejemplo si los vectores de bit se distribuyen como sigue:

00000…. (49% del tiempo)

11111…. (49% del tiempo)

?????…. (2% del tiempo)

entonces puede ser deseable manipular los vectores de bit anterior a la codificación de Huffman para tener una distribución más desigual tal como:

00000…. (79% del tiempo)

11111…. (19% del tiempo)

?????…. (2% del tiempo)

Reducción en vectores de bit generados

A fin de reducir la cantidad de vectores de bit que necesitan ser generados las realizaciones de la invención pueden usar una cualquiera o más de las siguientes estrategias:

- no se usan ID de región para todos los nodos y se generan solamente para nodos navegables (por ejemplo se ignoran los nodos que corresponden a ferrocarriles).
- no se necesitan vectores de bit para todos los segmentos de carretera y se pueden usar para segmentos de carretera de decisión alrededor de nodos de decisión. Los nodos de decisión y los segmentos de carretera de decisión se pueden determinar mirando los datos de segmento de carretera (como se describe en este documento).
- incluso aunque hay muchos vectores de bit posibles, algunos son mucho más frecuentes que otros, así se puede usar una codificación especial para los más frecuentes (por ejemplo 000…000 y 111…111).
- los vectores de bit que no lo son 000…000 o 111...111 tienen aún a menudo o bien la mayoría de sus bits fijados a 1, o bien la mayoría de sus bits fijados a 0. Así el código de Huffman de bloques parciales debería codificarlos bastante eficientemente.
- los vectores de bit son a menudo idénticos en nodos cercanos unos de otros, así la codificación delta puede codificarlos eficientemente.
- se puede hacer que diferentes regiones tengan más patrones de vector de bit similares reordenando los ID de región destino por región fuente (la idea se describe en este documento)

- o de todos los vectores de bit alrededor de segmentos de carretera de un nodo deberían siempre dar 111…111. Eso se puede usar adecuadamente para codificar vectores de bit más eficientemente.

Algunos de éstos se tratan en más detalle más adelante.

Vale la pena señalar en este punto que las técnicas descritas aquí aspiran a reducir el tamaño de los vectores de bit. No obstante, se señala que se requiere acceso aleatorio a los datos por los dispositivos que usan los datos de mapa para propósitos de encaminamiento. De manera general, una codificación eficiente de datos requiere un tamaño variable que impediría no obstante un acceso aleatorio a los datos.

Por tanto, las realizaciones de la invención pueden usar un compromiso en el que los datos se codifican como una serie de páginas, que se indexan, y entonces utilizan codificación variable dentro de esas páginas. En tales realizaciones, el acceso aleatorio es alcanzable para cada página (a través de la indexación). Una vez que se ha accedido a una página por las realizaciones, pueden decodificar posteriormente la página entera. Esto proporciona un compromiso entre eficiencia y tamaño de mapa - aumentar el número de nodos por página reduce el tamaño de mapa pero ralentiza el acceso a los datos. Una realización particular la invención usa 16 nodos por página. Se apreciara que cualquier nodo bien puede tener un número diferente de segmentos de carretera que salen de ese nodo. Por tanto, incluso aunque puede haber el mismo número de nodos puede haber una cantidad variable de segmentos de carretera por página. Además, bien puede darse una compresión diferente de cada uno de los vectores de bit almacenados en cada página.

Tal estructura puede conducir a un formato de mapa como se muestra en la Figura 11. En esta Figura, la referencia a Marca Arp es sinónimo de vector de bit. El número ‘n’ se almacena dentro de la cabecera y se puede alterar para diferentes mapas a fin de optimizar el rendimiento de ese mapa.

Ajustar ‘n’ es un compromiso entre tamaño de mapa y velocidad de acceso cuando se decodifican datos de mapa:

- un valor grande de n agrupará muchos nodos juntos lo cual es bueno para la compresión del mapa, pero malo para la velocidad de acceso aleatorio a los datos.

- un valor pequeño de n agrupará pocos nodos juntos lo cual es bueno para la velocidad de acceso aleatorio de los datos pero malo para la compresión del mapa.

- ‘n’ se puede fijar a 4 por ejemplo es decir páginas de 16 nodos-desde (un nodo-desde que está en un extremo inicial de un segmento de carretera - es decir la columna 700 de la Figura 10) pero tener en mente que cada nodo-desde tiene varios segmentos de carretera-a así que suponiendo 3 segmentos de carretera-a de media, cada uno significa que cada página almacena el equivalente de ~48 segmentos de carretera.

En este formato, hay un formato diferente para datos que dependen del nivel de la región que se codifica. La Figura 12 muestra un ejemplo de un formato para nivel 0 (es decir las regiones más toscas) y la Figura 12 muestra un ejemplo de formato para otros niveles de región.

Los vectores de bit y la información relacionada se almacenan en una estructura de datos, el formato de la cual se muestra en la Figura 11 que comprende lo siguiente: una cabecera 800; árboles de Huffman 802 usados en codificación de Huffman descrita más tarde; recuento de regiones en cada jerarquía 804 (vacía si es constante el número de regiones por nivel); contiguas de la región 806 (vacía si no hay contiguas de la región); tablas de reasignación de ID de región 808; índice de página de vector de bit (2^n nodos) 810; y el ID de región y vectores de bit 812. La estructura de datos que soporta los vectores de bit se puede mantener dentro de un fichero único o se puede mantener dentro de una pluralidad de ficheros.

En algunas realizaciones la cabecera de mapa 800 se dispone para contener información adicional que indica lo siguiente:

- número máximo de niveles

- longitud de la lista de contiguas más corta.

- longitud de la lista de contiguas más larga.

- desplazamiento de byte de la sección que contiene todas las listas de contiguas.

El o cada fichero que soporta la información puede tener una sección para codificar las listas de contiguas. El tamaño de todas las listas se codifica en primer lugar. Cada longitud de lista se codifica relativamente a la longitud de la lista más corta, en un número fijo de bits determinado por BitsRequired(longestListLength – shorterListLength). Señalar que si todas las listas tienen la misma longitud, entonces no se necesita ningún bit para codificar las longitudes.

Entonces sigue el contenido de todas las vistas. Cada lista se hace de varias tuplas de ID de regiones contiguas: pares en el nivel 0, tuplas de 3 elementos en el nivel 2, etc.

Señalar que no se codifican las tuplas de la región-desde (parte antes de ':' en el fichero ASCII). Ellas están implícitas dado que las listas están almacenadas para todas las regiones en orden ascendente. Por ejemplo, si un mapa tiene 3 niveles con 100x10x5 (100 regiones en el nivel 0,10 regiones en el nivel 1, 5 regiones en el nivel 2), entonces:

- en el nivel 0, las listas se almacenan para las regiones-desde 1, 2, 3, ... 100 (100 listas en ese orden). Cada una de estas listas contiene pares.
- en el nivel 1, las listas se almacenan para las regiones-desde 1.1, 1.2, 1.3, … 1.10, 2.1, 2.2, … 2.10, 3.1, … 100.9, 100.10 (1000 listas en este orden). Cada una de estas listas contiene tuplas de 3 elementos.
- en el nivel 2: no se almacena nada dado que es el último nivel así que no hay contiguas.

Cada componente en una tupla se almacena como n bits. El número de bits para cada nivel se determina a partir del número de regiones en el nivel correspondiente. Así es posible acceder a una lista aleatoriamente. En el caso de 3 niveles 100x10x5, codificar una tupla a.b.c usaría 7 bits para a (debido a que hay 100 regiones en el nivel 0), 4 bits para b (debido a que hay 10 regiones en el nivel 1), 3 bits para c (debido a que hay 5 regiones en el nivel 2).

Ejemplo: Supongamos una división de 100x10x5 regiones: 100 regiones en el nivel 0 más tosco, 10 en el nivel 1 intermedio y 5 en el nivel detallado 2.

En el fichero en el nivel 0, la sección que contiene las listas de contiguas contendrá:

- 100 números que indican la longitud de las listas para las 100 regiones en el nivel 0. El número de bits se calcula a partir de BitsRequired(longestListLength – shortestListLength). Cada número es relativo a la lista más corta en el nivel (la lista más corta que se almacena en la cabecera).
- Entonces sigue el contenido de las 100 listas (100 pares). El primer elemento de cada par se codifica en 7 bits (debido a que hay 100 regiones en el nivel 0) y el segundo elemento de cada par se codifica en 4 bits (debido a que hay 10 regiones en el nivel 1).

En el fichero en el nivel 1, la sección que contiene las listas de contiguas contendrá:

- 100*10 = 1000 números que indican la longitud de las listas para las 1000 regiones en el nivel 1.
- Sigue el contenido de las 1000 listas (1000 tuplas de 3 elementos). El primer elemento de cada tupla se codifica sobre 7 bits (debido a que hay 100 regiones en el nivel 0), el segundo elemento de cada tupla se codifica sobre 4 bits (debido a que hay 10 regiones en el nivel 1 en cada región de nivel 0) y el tercer elemento de cada tupla se codifica sobre 3 bits (debido a que hay 5 regiones en el nivel 2 en cada región de nivel 1).

En el fichero en el nivel 2, no necesita ser almacenado nada dado que es el último nivel.

Para el fichero de entrada codificador, se puede adoptar cualquier forma de listas.

De esta manera, una vez que se ha realizado la división en los tres niveles, cada nodo se asigna a una región por nivel; es decir tres regiones.

Cabecera 800

Típicamente, la cabecera usada en realizaciones de la invención es pequeña, y por tanto el tamaño no necesita ser optimizado a fin de reducir su tamaño. Típicamente todo está alineado en bytes o palabras por comodidad:

- versión de codificación (4 bytes) (se aumenta cada vez que cambia el formato de mapa)
- marcas de mapa (4 bytes) (para encender o apagar rasgos, 0 inicialmente pero se puede usar más tarde si necesitamos añadir rasgos opcionales)
- número total de nodos en el mapa (4 bytes)
- número total de segmentos de carretera en el mapa (4 bytes)
- desplazamiento de byte de sección de árboles de Huffman (4 bytes)

- desplazamiento de byte de sección de blob de región (4 bytes)
- desplazamiento de byte de sección de informaciones de página de región (4 bytes)
- desplazamiento de byte de sección de informaciones de página de vector de bit (4 bytes)
- desplazamiento de byte de sección de registros de tamaño variable (4 bytes)
- página de vector de bit máximo (marca arp) en bits (4 bytes) (se puede usar mediante métodos de planificación de ruta para asignar previamente el caso peor para un decodificador de flujo de bits en el arranque).
- tamaño de página de vector de bit medio (marca arp) en bits (4 bytes) (usado para interpolar la posición de página de vector de bit)
- delta de página de vector de bit mínimo (marca arp) (4 bytes) (usada para hacer todas las deltas > = 0, evitando almacenar bit de signo)
- tamaño máximo (2 bytes) de historia de vector de bit (marca arp) (se puede usar mediante métodos de planificación de ruta para asignar previamente el almacenador temporal de historia en el arranque)
- número máximo de segmentos de carretera por página (2 byte) (no usado actualmente)
- nivel Apolo de este fichero (1 byte)
- bits por vector de bit (marca arp) (1 byte)
- bits por delta de página de vector de bit (marca arp) (1 byte) (campo en registro de tamaño fijo de páginas de vector de bit (marca arp))
- bits por índice de blob (1 byte) (campo en registro de tamaño fijo de información de página de región)
- bits por recuento de regiones (1 byte) (campo en registro de tamaño fijo de información de página de región)
- bits por bloque de vector de bit (marca arp) no trivial (1 byte)
- log_2() de tamaño de página de nodo de región (1 byte)
- log_2() de tamaño de página de vector de bit (marca arp) (1 byte)
- número de árboles de Huffman para codificar los ID de región local (1 byte)
- número de árboles de Huffman para codificar códigos de historia de vector de bit (marca arp) (1 byte)

Árboles de Huffman 802

- árbol de Huffman para codificar el número de segmentos de carretera alrededor de cada nodo: diminuto, solamente 10 códigos o así, solamente presente en fichero en el nivel 0)
- árbol de Huffman para almacenar un bloque de vector de bit (marca arp) no trivial: árbol de Huffman más grande, cuanto más grande mejor para la compresión pero se requiere más memoria en los métodos de planificación de ruta (compromiso entre compresión de mapa y uso de memoria en métodos de planificación de ruta).
- árbol de Huffman de códigos delta de vector de bit (marca arp) cuando el tamaño de historia es 0: diminuto, solamente 3 códigos
- árbol de Huffman de códigos delta de vector de bit (marca arp) cuando el tamaño de historia es 1: diminuto, solamente 4 códigos
- árbol de Huffman de códigos delta de vector de bit (marca arp) cuando el tamaño de historia es > = n: diminuto (número de árboles de Huffman almacenados en la cabecera)
- árbol de Huffman para ID de región cuando hay 3 regiones en una página de región: diminuto, solamente 3 códigos

- árbol de Huffman para ID de región cuando hay 4 regiones en una página de región: diminuto, solamente 4 códigos
- árbol de Huffman para ID de región donde hay > = n regiones en la página de región: diminuto (número de árboles de Huffman almacenados en la cabecera)

Tabla de reasignación de región 804 y listas de ID de región 806

Aunque más pequeños que otras partes del formato de fichero de la Figura 11, los ID de región 806 también se pueden comprimir como se ejemplifica en la Figura 14. En ésta, se puede usar una correlación geográfica a fin de reducir la cantidad de datos usados.

Cada página de región almacena una lista de las distintas regiones en esa página de región. Esta lista se espera que sea más pequeña en la mayoría de los casos (de hecho muchas páginas están conteniendo probablemente solamente 1 región al menos en el nivel 0). Las listas de región tienen un tamaño variable. Los datos dentro de las páginas deberían estar accesibles de una forma aleatoria (es decir Acceso Aleatorio) y por tanto se usa un tamaño de tabla fijo para permitir esto.

Cada lista de regiones distintas se ordena por la frecuencia de nodos en cada región: el primer elemento de la lista corresponde a la región con el número más grande de nodos en la página, el último elemento de la lista es la región con el número menor de nodos en la página.

Cada nodo en las páginas de región, puede codificar su ID de región usando un ID de región local (local para la página). Este ID local es el índice de la región en la página (que es un entero pequeño, a menudo 0 dado que 0 corresponde a la región más popular en la página de región).

Los ID de región de los nodos se almacenan como sigue:

- Una Colección de Regiones, que contiene los ID de región, almacena todas las listas de solapamiento posibles de las regiones en las páginas. Las listas de región son los ID de región consecutivos en esa colección. Las listas pueden (y lo hacen) solaparse. La colección no almacena el inicio y final de cada lista (esto se hace por la tabla de información de página de región).
- La tabla de información de página de región es una tabla de registro de tamaño fijo (por lo tanto accesible aleatoriamente) y cada registro contiene el índice en la colección del comienzo de una lista + el número de elementos en la lista.
- Cada nodo contiene un ID de nodo local (local en su página).

Cada uno de estos conceptos se define además en lo sucesivo.

Colección de regiones

La colección de regiones codifica todas las listas de regiones posibles de páginas. Es una colección simple de ID de región en la que puede solaparse la lista de los ID de región. Su tamaño debería ser pequeño dado el solapamiento de listas. La colección de regiones se describe además en la sección de informaciones de páginas de región.

Informaciones de páginas de región

La especificación de una lista de ID de región en una tabla de página de región usa 2 campos en el registro de tamaño fijo de la tabla de información de página de región:

- un recuento de regiones (número de regiones en esta página, se espera que sea pequeño).
- un desplazamiento en una colección de listas de regiones (donde comienza la lista de región)

En una realización, esto se describe en la Figura 12.

El campo de desplazamiento apunta en la colección de regiones: registros de tamaño fijo con 1 byte por ID de región son suficientes, por ejemplo, suponiendo que hay siempre menos de 256 regiones en cada nivel, lo cual es una suposición razonable (pero hacerlo mayor de 8 bits es posible fácilmente si 256 bits por nivel se considera demasiado restrictivo). El recuento de regiones en el registro de tabla de páginas de región indica cuántas regiones van a ser consideradas en la colección en el desplazamiento especificado.

Si varias regiones tienen la misma lista, pueden apuntar a la misma ubicación, que debería ser compacta dado que podemos esperar muchas páginas o bien para compartir en las mismas listas, o bien para compartir partes de las mismas listas.

Esto se explica en más detalle con referencia la Figura 12 que muestra una realización que tiene páginas de 2^nr nodos (nr = 9 por ejemplo para agrupar 512 nodos)

Señalar cuán compacta puede ser la colección 900 de ID de región debido a que varias páginas pueden apuntar a la misma ubicación o solapar ubicaciones en la colección (etiquetada blob de región en la figura). De hecho, aumentar el número de páginas puede no aumentar el tamaño de la colección debido a que cada página entonces solapa menos regiones de manera que se reduce la posibilidad de combinaciones. Así esta colección debería permitir la creación de muchas páginas sin requerir demasiado espacio de mapa o demasiada memoria en el dispositivo en el que se cargan los datos de mapa generados.

El método aspira a mantener la colección 900 que contiene la lista de ID de región tan pequeña como sea posible. El método aspira a reutilizar la misma lista de ID de región tan a menudo como sea posible en la colección. El método es libre de reordenar los últimos 2 elementos de la lista dado que no influirían en el tamaño de los códigos de Huffman.

Por ejemplo, cuando una lista contiene 2 regiones 10 y 34, es equivalente a almacenar la lista 10, 34 o 34,10 (con independencia de las frecuencias) dado que el árbol de Huffman usa solamente 1 bit en todos los casos cuando hay 2 nodos solamente. En otras palabras, la reordenación en frecuencia se relaja para las 2 últimas regiones. De manera similar, una lista de 3 elementos 10, 34, 40 también se puede almacenar como 10, 40, 34, dado que solamente el primer código 10 (el más frecuente) usará 1 bit y los otros códigos 40 y 34 y ambos usan 2 bits (con independencia del orden).

De esta manera, mirando la Figura 12, se puede ver que la colección 900 almacena dos valores: una longitud y un desplazamiento desde el comienzo de los datos de región. Esta manera, tomando la primera fila (3: 0), ésta se refiere a tres piezas de desplazamiento de datos por 0 desde el inicio del fichero (es decir 10, 34, 40). Tomando como ejemplo adicional, la entrada de la colección (1:2) se refiere a una única región (es decir una longitud de 1) con el desplazamiento de dos desde el comienzo del fichero; (es decir la región 40).

En una realización alternativa, la información de página de región se codifica según el siguiente método:

Esta sección codifica el número de subregiones en cada región. El número de subregiones puede ser variable por nivel. No obstante, a menudo el número de subregiones es constante para cada nivel. El número de subregiones se codifica relativamente al número mínimo de regiones en cada nivel y usando log_2 (max_number_of_regions – min_number_of_regions) bits. Así si el número de regiones es constante, entonces se usan 0 bits para codificar el recuento de regiones y esta sección está vacía. El número mínimo y máximo de regiones se almacena en la cabecera del fichero lateral.

Codificación de la sección de contiguas de la región (alrededores que se tratan en relación a las Figuras 6 y 6a)

Esta sección codifica para cada jerarquía de región en un nivel L dado una lista de contiguas de región en el nivel L +1 más detallado. Por ejemplo, una región 3.9 en el nivel L = 1 puede tener la siguiente lista de contiguas en el nivel L = 2: 3.5.4 3.6.3 3.6.4 4.7.1 4.7.3. Como se trató en otra parte la lista de contiguas se puede usar para aumentar la velocidad del procesamiento previo usado para generar el o cada fichero lateral.

Esta sección se divide en 2 sub secciones:

- una sub sección para codificar la longitud de todas las listas de contiguas (tantas como regiones haya en el nivel dado). La longitud se codifica relativamente a la lista más corta y entonces se calcula el número de bits como log_2 (length_longest_list – length_shortest_list). Si todas las listas tienen la misma longitud, entonces se usan 0 bits para codificar las longitudes (y de esta manera la sección entonces está vacía).
- una sub sección para codificar todas las listas de contiguas (tantas como regiones hay en el nivel dado).

Codificación de tablas de reasignación de ID de región

Esta sección se codifica en el fichero lateral de nivel 0 solamente. Codifica una tabla bidimensional que se usa para reordenación y coalescencia de bits en cada región en el nivel 0 (a fin de codificar eficientemente los vectores de bit, la coalescencia y reordenación de bits se describen además más adelante en este documento. La reordenación y coalescencia de bits en los vectores de bit se hace para optimizar la codificación de Huffman de vectores de bit. Esta tabla se usa por los métodos de planificación de ruta para encontrar la posición de bit en el vector de bit para

decodificar cuando se conoce:

- el ID de región-desde del nodo actual
- el índice de bit-a original (es decir el índice de bits después de la des coalescencia de bits del vector de bit)

Los 2 índices de la tabla bidimensional son:

- el ID de región-desde
- el índice de bit de destino (región de destino)

Esta sección está hecha de 2 sub secciones:

- una sección para codificar el número de bits juntados para cada región en el nivel 0. El número de bits usado para codificar cada número es log_2(max_number_of_coalesed_bits)
- una sección para codificar la tabla de reasignación de bits. Dado que cuando se encamina, el bit de destino no cambia (el destino sigue siendo el mismo mientras que se encamina) pero el ID de región-desde cambia (dependiendo de los nodos explorados mientras que se encamina) la matriz almacena por filas el bit de destino. En cada fila, se almacena el número de reordenación de bit para cada región-desde. Los métodos de planificación de ruta típicamente deberían cargar solamente 1 fila en memoria correspondiente a un destino de encaminamiento dado mientras que se encamina. El método de planificación de ruta no necesita almacenar la matriz bidimensional entera en memoria. Los métodos de planificación de ruta pueden acceder a esa fila aleatoriamente. El número de bits usados para codificar cada entrada de reasignación se calcula como log_2(número máximo de bits juntados).

La Figura 13 expande la sección de vector de bit 812 del fichero mostrado en la Figura 11 para la región de nivel 0 (ver la Figura 9). Se ve que cada página comprende un recuento de segmentos de carretera, ID de región y los vectores de bit.

La Figura 14 expande la sección de vector de bit 812 para el fichero mostrado en la Figura 11 para niveles distintos del nivel 0. Se puede ver que para otras regiones en los vectores de bit se almacenan o no los ID de región o recuento de segmentos de carretera que se almacenan para el nivel 0.

En vista de la diferencia en el área cubierta por las regiones de cada nivel, se puede variar por nivel el número de nodos por página. Por ejemplo, según una región dentro del nivel 0 cubre un área grande puede haber más nodos por página por región para el nivel 0. Por ejemplo, algunas realizaciones pueden almacenar 512 nodos por página por región para el nivel 0. Por tanto, un nivel más detallado puede tener un número menor de nodos – por ejemplo 256 nodos, 128 nodos, 64 nodos, 32 nodos, 16 nodos, o similar. De hecho, alguna realización puede utilizar 1024 nodos, 2048 nodos, o 4096 nodos por página.

Codificación de vectores de bit 810, 812

La tabla 810 contiene registros de tamaño fijo. Los ID de nodo-desde se agrupan en páginas de 2^n juntas.

Es conveniente agrupar datos en páginas de múltiples nodos consecutivos debido a que se espera que los vectores de bit tengan patrones similares para varios segmentos de carretera en los mismos alrededores. Usando páginas, es posible codificar varios segmentos de carretera en delta y lograr una buena compresión. De manera similar, es posible codificar los ID de región de nodos en delta en las páginas. Por otra parte, significa que los datos de acceso de un segmento de carretera requieren desempaquetar datos de varios segmentos de carretera (sin acceso aleatorio directo).

Tener que desempaquetar varios nodos o segmentos de carretera para acceder a los datos de un segmento de carretera o nodo se puede considerar aceptable dado que:

- los datos se pueden almacenar en caché de manera que los datos extra leídos cuando se accede a un segmento de carretera a menudo no son útiles. Puede ser que estos datos extra sean útiles poco después (esto es similar a leer anticipadamente un parámetro de almacenamiento en caché de disco).
- el encaminamiento usando los vectores de bit reduce el tamaño de la búsqueda de expansión en un orden de magnitud en comparación con un encaminamiento A* Dijkstra. Agrupando los datos en páginas, solamente una pequeña parte del mapa aún necesita ser decodificada (páginas a lo largo del trayecto real).
- debería reducir significativamente el tamaño de datos codificados gracias a compresión delta de vectores de bit e ID de región.

- las páginas reducen el tamaño de indexación dado que los datos serán almacenados en un fichero lateral como se describió.

Cada registro dentro de la tabla 810 contiene un campo 'delta' que se usa para encontrar la posición del comienzo del tamaño variable de cada página (delta para una posición interpolada). El número de bits para cada delta se almacena en la cabecera.

A fin de acceder al ID de región y vectores de bit 812 un decodificador puede calcular el desplazamiento estimado de la página de inicio haciendo una interpolación lineal:

intepolated_offset = from_node_id*avg_page_size

Donde avg_page_size es el tamaño de página medio en bits almacenado en la cabecera (posiblemente en un punto fijo para mejorar la precisión). El desplazamiento de los datos se puede calcular entonces como sigue:

offset = intepolated_offset + min_delta + delta

Donde min_delta es el valor mínimo de todos los campos delta para todas las páginas (almacenadas en la cabecera) y delta es el campo sin signo almacenado en la página. El valor min_delta asegura que todos los campos delta son un valor positivo (sin bit de signo para almacenar).

Se accede a los registros de tamaño variable a través del campo 'delta' de las informaciones de página de vector de bit descritas previamente.

Cada registro contiene los datos de 2^n nodos (ID de región de nodos-desde y vectores de bit de sus segmentos de carretera unidos en todos los niveles). El mismo esquema de indexación se usará de esta manera para todos los niveles.

Los registros de tamaño variable almacenan:

- código de página – un código que indica para la página entera en cuanto a si los nodos dentro de esa página son parte o no de la misma región;
- el número de segmentos de carretera alrededor de cada nodo en la página de vector de bit (solamente almacenada en el nivel 0 dado que sería la misma para todos los niveles).
- los ID de región de los nodos-desde en la página, un ID por nivel (información para todos los niveles almacenados en el fichero en el nivel 0 {{ap_0_*.dat)
- vectores de bit de segmentos de carretera alrededor de nodos en la página (solamente alrededor de nodos que tienen > 2 segmentos de carretera unidos) en el nivel i solamente. Ésta es la parte de datos más grande.
- Codificación del número de segmentos de carretera alrededor de cada nodo

Para cada uno de los 2^n nodos en una página de vector de bit, un código de Huffman codifica el número de segmentos de carretera alrededor del nodo. Esta información no es específica para todos los niveles y solamente se almacena en el fichero en el nivel 0 (ap_0_*.dat).

El conocimiento del número de segmentos de carretera alrededor de un nodo se usa para decodificar los vectores de bit (ver 1000, Figura 13). Esta información es redundante con la información ya presente en otros ficheros pero hace más fácil y más rápido decodificar vectores de bit en páginas sin tener que buscar esa información en otra parte; por ello un aumento pequeño en tamaño proporciona un aumento en rendimiento.

Codificación de ID de región en registros de tamaño variable

Los ID de región de nodos 1002 se codifican en los registros de tamaño variable después de la codificación del número de segmentos de carretera 1000 alrededor de cada nodo (ver disposición de codificación). A fin de realizar un encaminamiento usando los vectores de bit generados en el procesamiento previo se necesita generalmente acceso a los ID de región 1002 en todos los niveles para un nodo dado, los ID de región de todos los niveles se almacenan en el mismo fichero cerca de otro más que dividirlos en diferentes ficheros por nivel.

Una página de vector de bit de 2^n nodos (n = 4 por ejemplo) y con 3 niveles Apolo almacenaría de esta manera los ID de nodo como sigue:

node#0: local_region_id_level_0 local_region_id_level_1 local_region_id_level_2

node#1: local_region_id_level_0 local_region_id_level_1 local_region_id_level_2

node#2: local_region_id_level_0 local_region_id_level_1 local_region_id_level_2

…

node#15: local_region_id_level_0 local_region_id_level_1 local_region_id_level_2

Adicionalmente:

- Nada se codifica alrededor de nodos con 1 o 2 segmentos de carretera unidos (es decir para nodos para los que almacenamos 0 como número de segmentos de carretera unidos).
- Cuando el bit en el código de página se fija en un nivel dado, entonces se sabe que todos los nodos están en el mismo ID de región en ese nivel y el ID de región en ese nivel entonces se codifica solamente una vez (para el primer nodo con > = 3 segmentos de carretera unidos). El número de bits para codificar el ID de región es log_2(regionCount – 1).
- Excepto para el primer nodo en una página donde se codifica un ID de región, también se codifica un bit antes de codificar cada ID de región. Este bit indica si el ID de región es idéntico al ID de nodo codificado previamente en el mismo nivel. Cuando se fija este bit, no hay necesidad de codificar el ID de región dado que es el mismo que se codificó previamente en ese nivel. Cuando este bit es 0, codificamos un ID de región con log_2 (regionCount – 1) bits. Dado que muchos nodos consecutivos están en la misma región, a menudo solamente necesitamos 1 bit para codificar el ID de región.

La codificación de Huffman del índice de región local es eficiente debido a que:

- las regiones se clasifican por frecuencias en cada página de región (así el índice local 0 es más frecuente que el índice local 1,…)
- hay distintos árboles de Huffman especializados para cada número de regiones en una página (1 árbol de Huffman para 3 regiones en la página, 1 árbol de Huffman para 4 regiones en la página, etc.). Los árboles de Huffman son pequeños y por tanto es posible almacenar varios sin usar cantidades significativas de memoria.
- tener 3 regiones o más debería ser bastante raro de todos modos, al menos en el nivel 0 (pero no sería raro en otros niveles).

Codificación de vectores de bit en registros de tamaño variable

Cada registro de tamaño variable contiene vectores de bit para todos los segmentos de carretera alrededor de la página en la página. Los vectores de bit se codifican solamente alrededor de los nodos con 3 o más segmentos de carretera unidos (segmentos de carretera). Para nodos con 1 o 2 segmentos de carretera unidos, los métodos de encaminamiento pueden dar implícitamente vectores de bit (valores 111...111) a esos nodos.

Los nodos-a no se codifican.

Con referencia a Figura 10, se señala que un segmento de carretera se puede especificar por 2 nodos; uno en cada extremo del mismo. De esta manera, cuando se considera la dirección un nodo en el comienzo de una dirección se puede conocer como un nodo_desde y un nodo en el final se puede conocer como un nodo_a.

Se usan diversas propiedades alrededor de los vectores de bit dentro de la codificación para hacerla eficiente:

- Muchos de los vectores de bit son 000…000 o 111…111.
- Para otros valores de vectores de bit (es decir no 000...000 o 111…111) no es probable que sea una repetición y el mismo valor probablemente se repetirá.
- También el OR de todos los vectores de bit alrededor de un nodo dado debería ser 111…111.

Los vectores de bit se codifican como un primer código de Huffman y códigos de Huffman adicionales opcionales. El primer código de Huffman indica si el vector de bit es:

- un código 0 para el vector de bit trivial 000…000
- un código 1 para el vector de bit trivial 111…111

- un código 2 para indicar un vector de bit no trivial aún no encontrado en la página. En ese caso, y solamente en este caso, otro código de Huffman sigue a codificar el vector de bit recién encontrado.

- un código > = 2 cuando el vector de bit es idéntico a un vector de bit previamente encontrado en la página actual (ignorar los vectores de bit triviales 000…000 y 111…111). Esta codificación de esta manera usa la historia de código encontrado previamente en la página. El código entonces da realmente el índice en la historia de todos los códigos encontrados previamente.

Aparte de este código de Huffman, necesita ser codificada más información solamente en el caso de vectores de bit no triviales no encontrados en la historia (código = 2). En ese caso, justo después del código de Huffman = = 2, se codifica:

- 1 bit de negación

- Varios códigos de Huffman para codificar en bloques de N bits los vectores de bit para n regiones (N y n se dan en la cabecera de mapa). Por ejemplo, codificar 100 regiones (vectores de bit de 99 bits) usando bloques de 11 bits requiere codificar 9 códigos de Huffman (9x11 = 99).

Dado que la mayoría de los vectores de bit contienen o bien mayoritariamente ceros o bien mayoritariamente unos, el bit de negación indica si el vector de bit se almacena negado o no. Esto permite el almacenamiento de códigos en el árbol de Huffman que contiene por mucho mayoritariamente ceros (de esta manera mejorando la codificación de Huffman de los bloques). El bit de negación existe solamente si el tamaño de los bloques es menor que el número de regiones, que es el caso en la práctica en el nivel 0 pero en el nivel 1 o 2, el vector de bit entero se podría codificar en 1 bloque solamente así que no es necesario el bit de negación.

Si hay 100 regiones; N = 100 (por lo tanto vectores de bit de 99 bits), el primer bloque codifica bits para las regiones destino 1 a 11, el segundo bloque codifica las regiones 12 a 22, etc. En el primer bloque, el LSB (0x1) corresponde a la región destino 1, el siguiente bit (0x2) corresponde a la región 2, el siguiente bit (0x4) corresponde a la región 3, etc.

Para vectores de bit que usan la historia, la profundidad de la colección de historia es el número de vectores de bit distintos encontrados previamente en la página (sin tener en cuenta los vectores de bit triviales 000…000 y 111…111). Se usa un árbol de Huffman diferente si el vector de historia contiene 0 elementos, 1 elemento, 2 elementos, 3 elementos, etc. Multiplicar los árboles de Huffman es aceptable dado que todos los árboles de Huffman son pequeños y por tanto no se requiere un almacenamiento significativo:

- cuando la historia tiene 0 elementos, el árbol de Huffman tiene 3 códigos: 0 para 000…000, 1 para 111…111, 2 para un nuevo vector de bit.

- cuando la historia tiene 1 elemento, el árbol de Huffman tiene 4 códigos: 0 para 000…000, 1 para 111…111, 2 para un nuevo vector de bit, 3 para el mismo que el elemento #0 en la historia.

- cuando la historia tiene 2 elementos, el árbol de Huffman tiene 5 códigos: 0 para 000…000, 1 para 111…111, 2 para un nuevo vector de bit, 3 para el mismo que el elemento #0 en la historia, 4 para el elemento #1 en la historia

- etc.

El tamaño de la página de vector de bit se espera que sea pequeño (2^n nodos-desde por ejemplo) así que el número de árbol de Huffman se espera que sea pequeño.

No obstante, es posible limitar el tamaño del árbol de Huffman a un valor máximo: por ejemplo siempre que la historia contenga más de H elementos, se usará un árbol de Huffman único (el valor H se almacena en la cabecera de mapa).

Esta codificación codifica solamente distintos vectores de bit en cada página + algunos códigos.

La codificación es más eficiente en tamaño con páginas de índice grandes pero a costa de disminuir la decodificación a fin de usar los vectores de bit para propósitos de encaminamiento (más vectores de bit para decodificar en las páginas).

Estadísticas

Aquí se detallan las estadísticas para un formato de fichero cuando se codifica el Benelux en 254 regiones (1 nivel). Los siguientes parámetros de entrada fueron usados:

número de bits por bloque de vector de bit: 11

número de nodos por página de vector de bit: 2^4 = 16

número de nodos por página de región: 2^9 = 512

Las estadísticas se proporcionan para dar una idea de la forma del mapa en términos de tamaño de mapa e ilustrar la descripción del formato de mapa sobre algunos datos reales:

- - - Contadores de estadísticas globales:

recuento de nodos 1598632

recuento de líneas 1598632 (100,000%)

saltar líneas alrededor de nodos con 1 línea 220180 (13,773%)

saltar líneas alrededor de nodos con 2 líneas 727138 (45,485%)

- - - Comienza en nivel = [0] :

Contadores de mapa. 87437736 (100,000%)

marcas arp triviales codificadas 000 . . . 000 . . . . . 1310914 (31,651%)

marcas arp triviales codificadas 111 . . . 111 . . . . . 913348 (22,052%)

marcas arp codificadas en la historia 362432 (8,751%)

marcas arp codificadas no en la historia 607717 (14,673%)

bloques negados 235171 (5,678%)

Tamaño de mapa (en bits). 87437736 (100,000%)

cabecera global 496 (0,001%)

árboles de Huffman 28808 (0,033%)

blob de región 52664 (0,060%)

informaciones de páginas de región 56216 (0,064%)

informaciones de páginas de marca arp 2497880 (2,857%)

registros de tamaño variable 84801672 (96,985%)

recuento de líneas alrededor de nodos 2847844 (3,257%)

ID de región de nodo 2112451 (2,416%)

marcas arp 79841370 (91,312%)

código trivial 000 . . . 000 . . . . . 1689322 (1,932%)

código trivial 111 . . . 111. . . . . 1826696 (2,089%)

código encontrado en la historia. 1668053 (1,908%)

no encontrado en la historia. 74657299 (85,383%)

código no encontrado en la historia 1463183 (1,673%)

bit de negación 607717 (0,695%)

bloques. 72586399 (83,015%)

Todos los tamaños están en bits. El tamaño de mapa total es de 87.437.736 bits (10.929.717 bytes).

El sangrado refleja la jerarquía. La información de registro de tamaño variable es por mucho la parte más grande de información (96,985% del tamaño del mapa). En los registros de tamaño variable, los sub elementos (con sangrado) dan más detalles. Los vectores de bit son por mucho la parte más grande de información para almacenar en registros de tamaño variable (91,312%). Y en los vectores de bit, almacenar los vectores de bit no triviales no encontrados aún en la historia constituye la parte más grande del mapa (83,015%).

Estadísticas en árboles de Huffman

Esta sección da las estadísticas para árboles de Huffman cuando se codifica el Benelux en 255 regiones (es decir para los datos de mapa mostrados anteriormente).

Árbol de Huffman del número de segmentos de carretera alrededor de cada nodo

- - - [árboles de Huffman : NrLines] - - -

bits:	1	valor:	3	código	0
bits:	2	valor:	2	código	10
bits:	3	valor:	1	código	110
bits:	4	valor:	4	código	1110
bits:	5	valor:	5	código	11110
bits:	6	valor:	6	código	111110
bits:	7	valor:	7	código	1111110
bits:	7	valor:	8	código	1111111

La mayoría de los nodos tienen 3 segmentos de carretera, pero en la posición 2ª y 3ª en el árbol de Huffman encontramos nodos con 2 y 1 segmentos de carretera unidos (que no son nodos de decisión).

Árbol de Huffman de bloques de vectores de bit no triviales

Este es el árbol de Huffman más grande dado que los bloques de almacenamiento de vectores de bit triviales son el contribuyente de tamaño de mapa más grande por mucho (83,015% en el ejemplo del Benelux de 255 regiones).

- - - [árboles de Huffman : NonTrivialArpFlagBlock] - - -

bits:	1	valor:	0	código	0
bits:	6	valor:	1	código	100000
bits:	6	valor:	2	código	100001
bits:	6	valor:	4	código	100010
bits:	6	valor:	8	código	100011
bits:	6	valor:	16	código	100100
bits:	6	valor:	32	código	100101
bits:	6	valor:	64	código	100110

bits: 6 valor: 512 código 100111

bits: 6 valor: 1024 código 101000

bits: 7 valor: 128 código 1010010

bits: 7 valor: 256 código 1010011

bits: 7 valor: 384 código 1010100

bits: 8 valor: 5 código 10101010

. . . recorte, demasiado largo . . .

bits: 24 valor: 1534 código 111111111111111111111011

bits: 24 valor: 1717 código 111111111111111111111100

bits: 24 valor: 1741 código 111111111111111111111101

bits: 24 valor: 1830 código 111111111111111111111110

bits: 24 valor: 1973 código 111111111111111111111111

Almacenar un bloque hecho de todo 0 es el patrón más frecuente y se codifica solamente en 1 bit según el árbol de Huffman anterior (lo que significa que un 50% o más de los bloques codifican el valor 0, incluso aunque no se codifiquen por bloques los vectores de bit triviales 000…000). Esto es debido a que la mayoría del vector de bit no trivial contiene o bien

- mayoritariamente ceros (y unos pocos unos)
- o bien mayoritariamente unos (y unos pocos ceros)

El esquema de codificación niega (~) los vectores de bit que contienen mayoritariamente unos así en el final, la codificación de bloques de vectores de bit mayoritariamente codifica bloques que contienen 000…000 en 1 bit solamente. Los siguientes bloques más frecuentes son bloques donde solamente se fija 1 bit (1, 2, 4, 8, 16, 32, 64...). Tienen más o menos las mismas frecuencias por lo tanto el mismo (o casi el mismo) número de bits.

Árboles de Huffman de ID de región local

Dado que la lista de regiones se almacena por frecuencia en cada página, podemos ver que almacenar el ID de región local 0 lleva menos bits (de hecho solamente 1 bit) que otros ID de región de ubicación. Los diferentes árboles de Huffman corresponden a páginas con 3 regiones, 4 regiones, 5 regiones, etc.

- - - [árboles de Huffman : Regions_0] - - -

bits: 1 valor: 0 código 0

bits: 2 valor: 1 código 10

bits: 2 valor: 2 código 11

- - - [árboles de Huffman : Regions_1] - - -

bits: 1 valor: 0 código 0

bits: 2 valor: 1 código 10

bits: 3 valor: 2 código 110

bits: 3 valor: 3 código 111

- - - [árboles de Huffman : Regions_2] - - -

bits:	1	valor:	0	código 0
bits:	2	valor:	1	código 10
bits:	3	valor:	2	código 110
bits:	4	valor:	3	código 1110
bits:	4	valor:	4	código 1111

- - - [árboles de Huffman : Regions_3] - - -

bits:	1	valor:	0	código 0
bits:	2	valor:	1	código 10
bits:	3	valor:	2	código 110
bits:	4	valor:	3	código 1110
bits:	5	valor:	4	código 11110
bits:	5	valor:	5	código 11111

. . . recorte . . .

Árboles de Huffman de códigos de historia de vector de bit

El código 0 (que significa vector de bit trivial 000…000) es el más frecuente (y codificado en 1 bit solamente en la mayoría de los casos). El código 1 (vector de bit trivial 111…111) es entonces el siguiente más frecuente (y codificado en 1 bit solamente). El siguiente código más frecuente (2) es para el vector de bit no trivial codificado por bloques. Los otros códigos (> 2) para el vector de bit encontrado en la historia.

- - - [árboles de Huffman : ArpFlag_0] - - -

bits:	1	valor:	0	código 0
bits:	2	valor:	1	código 10
bits:	2	valor:	2	código 11

- - - [árboles de Huffman : ArpFlag _1] - - -

bits:	1	valor:	0	código 0
bits:	2	valor:	1	código 10
bits:	3	valor:	2	código 110
bits:	3	valor:	3	código 111

- - - [árboles de Huffman : ArpFlag _2] - - -

bits:	1	valor:	0	código 0
bits:	2	valor:	1	código 10
bits:	3	valor:	2	código 110
bits:	4	valor:	3	código 1110

bits: 4 valor: 4 código 1111

- - - [árboles de Huffman : ArpFlag _3] - - -

bits: 1 valor: 0 código 0
bits: 2 valor: 1 código 10
bits: 3 valor: 2 código 110
bits: 4 valor: 3 código 1110
bits: 5 valor: 4 código 11110
bits: 5 valor: 5 código 11111

- - - [árboles de Huffman : ArpFlag _4] - - -

bits: 1 valor: 0 código 0
bits: 2 valor: 1 código 10
bits: 3 valor: 2 código 110
bits: 4 valor: 3 código 1110
bits: 5 valor: 4 código 11110
bits: 6 valor: 5 código 111110
bits: 6 valor: 6 código 111111

- - - [árboles de Huffman : ArpFlag _5] - - -

bits: 2 valor: 0 código 00
bits: 2 valor: 1 código 01
bits: 2 valor: 2 código 10
bits: 4 valor: 3 código 1100
bits: 4 valor: 4 código 1101
bits: 4 valor: 5 código 1110
bits: 5 valor: 6 código 11110
bits: 5 valor: 7 código 11111

. . . recorte . . .

Influencia de los parámetros de entrada en el tamaño de mapa

Hay un número de parámetros de entrada que controlan el formato de fichero mostrado la Figura 11 y que pueden influir en el tamaño del mapa. Retocar los parámetros puede ser un compromiso entre tamaño de mapa y uso memoria o velocidad de descompresión dependiendo de los parámetros.

Las realizaciones de la invención pueden usar los siguientes parámetros de entrada:

- tamaño de página de región
- tamaño de página de vector de bit

- tamaño del bloque de vector de bit
- recuento de códecs de Huffman de vector de bit
- recuento de códecs de Huffman de región

Influencia del tamaño del bloque de vector de bit

valor	tamaño de mapa (bits)	
4	35548192	
5	33648792	
6	32290344	
7	30853616	
8	31103200	
9	30436696	(por defecto)
10	30051792	
11	29266784	
12	28934696	

Aumentar el tamaño del bloque de Huffman para los vectores de bit mejora la comprensión del mapa. Cuanto mayor es el tamaño del bloque, mejor es la compresión. Pero el aumento del tamaño del bloque es a costa de requerir más memoria para almacenar un árbol de Huffman mayor de 2^n valores. La tabla anterior ilustra esto.

La influencia este parámetro se espera que llegue a ser más significativa cuando introducimos la optimización para reasignar los ID de región por región fuente: esta optimización provocará de manera optimista una reducción de tamaño de mapa significativa cuando se usa un tamaño de bloque de vector de bit grande.

Influencia del tamaño de página de vector de bit

valor	tamaño de mapa (bits)	
2^1	55563944	
2^2	42502936	
2^3	34898840	
2^4	30436696	(por defecto)
2^5	27389952	
2^6	25165032	
2^7	23635936	

Aumentar el tamaño de página ayuda a comprimir mejor los mapas. Pero las páginas grandes desafortunadamente ralentizan la descompresión de los datos en el formato de fichero por los métodos de encaminamiento, dado que acceder al vector de bit de un segmento de carretera aleatorio requiere decodificar todos los segmentos de carretera en la página. La tabla anterior ilustra esto.

Influencia del recuento de códecs de Huffman de vector de bit

valor	tamaño de mapa (bits)	
1	30866920	
2	30748024	
3	30634168	
5	30504504	
7	30467944	
9	30436696	(por defecto)
11	30423688	

Aumentar el número de códecs de Huffman de vector de bit ayuda a aumentar la compresión ligeramente y esto se ilustra en la tabla anterior. No hay casi ningún inconveniente en aumentar el valor dado que esos árboles de Huffman son pequeños en cualquier caso. Aumentar más allá de 9 árboles de Huffman (valor por defecto) no da ninguna mejora significativa. Aumentar este parámetro podría ser más efectivo con páginas de vector de bit más grandes.

Coalescencia y Reordenación de bits en vectores de bit

Los vectores de bit tienen patrones. Esos patrones difieren significativamente para cada región fuente (es decir la región del nodo donde se almacena el vector de bit). El número de bits para almacenar vectores de bit de N bits se puede reducir almacenando tablas de traducción pequeñas para cada región fuente. Esas tablas realizan 2 funciones adicionales descritas en esta sección:

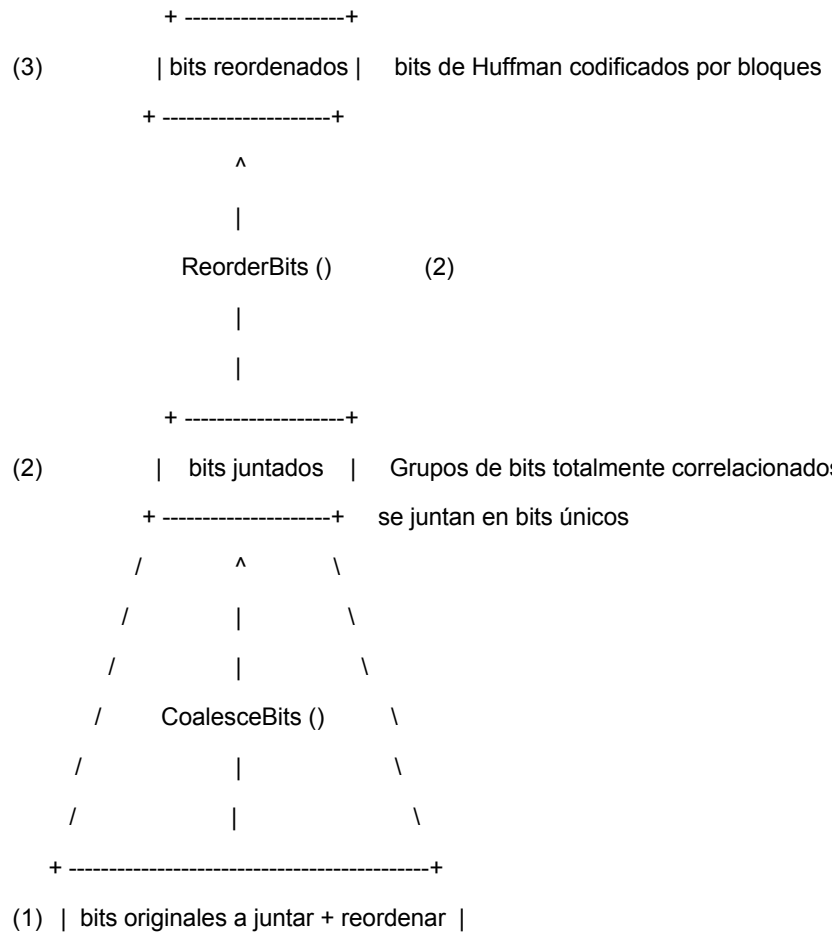
- coalescencia de bit
- reordenación de bit

La idea se puede entender intuitivamente como sigue: cuando está en España, está claro que un segmento de carretera que conduce a Suecia (bit = 1 para región de destino Suecia) es más probable que conduzca también a Noruega (es decir el bit para el destino Noruega es entonces también 1). Si otro segmento de carretera no conduce a Suecia (bit = 0), entonces tampoco conduce a Noruega en la mayoría de los casos. Así cuando está en España, los valores de los bits del vector de bit para las regiones de destino Suecia y Noruega son casi siempre iguales. De hecho, son incluso siempre estrictamente iguales para muchas regiones de destino. Qué bits están bien correlacionados con qué bit depende en gran medida de la región desde.

Cuando está en Finlandia por ejemplo, los bits para la región de destino Noruega y Suecia están mucho menos correlacionados. Por otra parte, en Finlandia, los bits para la región de destino España y Portugal es probable que estén 100% correlacionados (o al menos muy cerca al 100%).

La **coalescencia de bits** explota la propiedad de que algunos bits son siempre iguales (completamente correlacionados). Esos bits se pueden juntar en un único bit. La coalescencia de bits reduce el número de bits a codificar en cada región.

La **reordenación de bits** explota la propiedad de que algunos bits están bastante bien correlacionados (pero no el 100% correlacionados) a fin de barajar los bits de tal manera en cuanto optimizar la codificación de Huffman de vectores de bit (y/o reducir el tamaño de los árboles de Huffman).



Antes de calcular las tablas para juntar bits y reordenar bits, se calculan las correlaciones de todos los pares de bits en todas las regiones-desde. El número de distintos pares es:

C(N, 2) = N! / (2!*(N – 2)!) = N*(N-1)/2

…donde N es el número de regiones. Este número es de esta manera bastante alto cuando el número de regiones es alto. Calcular todas las correlaciones de bit es la parte más lenta del método para generar el formato de fichero mostrado en la Figura 11. La complejidad del método es n*N*N donde n es el número de vectores de bit y N es el número de regiones. Para cada par de bits (es decir par de distintas columnas en los vectores de bit) en cada región, calculamos una tabla tridimensional:

bitCorrelation[fromRegionId][bitI][bitJ].

Cada entrada en la tabla contiene una estructura de 4 campos que cuenta:

- el número de veces de bitI = 0 y bitJ = 0 en todos los vectores de bit de fromRegionId
- el número de veces de bitI = 0 y bitJ = 1 en todos los vectores de bit de fromRegionId
- el número de veces de bitI = 1 y bitJ = 0 en todos los vectores de bit de fromRegionId
- el número de veces de bitI = 1 y bitJ = 1 en todos los vectores de bit de fromRegionId

Aunque caro en términos de tiempo de procesador (y por lo tanto lento) para calcular, este proceso puede ser fácilmente puesto en paralelo dado que el cálculo de las correlaciones de bit para cada región-desde es completamente independiente de otras regiones-desde.

Realizaciones de la presente invención usan subprocesamiento múltiple para acelerar el cálculo de correlación de todos los bits para usar eficientemente máquinas de SMP (Multiprocesamiento Simétrico). En un sistema, una máquina de 1 CPU que calcula la correlación de bit de Benelux con 300 regiones tarda alrededor de 8 minutos. Pero el paralelismo escala bien y cuando se permiten subprocesos y usando las 4 CPU calcular la correlación de bit

entonces tarda 2 minutos (4 veces menos).

Coalescencia de bits

Cuando varios bit están completamente correlacionados (es decir siempre todos 0 o todos 1), podemos juntarlos en solamente 1 bit sin perder ninguna información. Refiriéndose a un conjunto de bits que están completamente correlacionados en una región dada como un 'grupo' entonces los vectores de bit de una región dada se hacen de varios grupos. Si los vectores de bit de N bit se hacen de n grupos, entonces la codificación de vectores de bit de N bits usa n bits + una tabla pequeña para indicar qué bits son equivalentes en cada región.

Tal coalescencia de bits tiene la ventaja de que se reduce el tamaño del fichero resultante de una manera sin pérdidas. Esta ventaja es probable que se aumente según aumenta el número de regiones dado que es probable que se junten más bits. Así el tamaño del mapa aumenta de una forma sub lineal con el número de regiones.

En una realización fueron obtenidos los siguientes datos:

	número mínimo de grupos	número medio de grupos	número máximo de grupos
Benelux de 255 regiones	12	84	152
Benelux de 300 regiones	13	90,017	163

Por tanto en el ejemplo del Benelux que tiene 255 regiones hay al menos 1 región que tiene solamente 12 grupos (es decir requiere solamente 12 bits para codificar sus vectores de bit de 255 bits (es decir el vector de bit tiene la misma longitud que el número de regiones) incluso antes de la codificación de Huffman (y de esta manera incluso menos después de la codificación de Huffman).

En media, las regiones requieren 84 bits para Benelux de 255 regiones. En este ejemplo, la peor región requiere 152 bits (152 grupos) antes de la codificación de Huffman.

Según otro ejemplo y tomando el Id de región = 3 en el ejemplo de Benelux de 255 regiones anterior el cual tiene los siguientes 18 grupos que se toman a partir de los datos codificados.

regionId=[3] tiene [18] grupos:

#1 (0)

#1 (1)

#1 (2)

#141 (3 4 5 7 8 11 12 15 16 19 20 21 22 23 24 25 26 27 28 29 30

31 32 33 36 37 38 39 40 42 43 44 45 46 50 51 59 60 62 63 66

67 69 71 72 73 74 :75 76 80 81 82 83 84 86 88 89 90 95 96 97

98 100 101 102 103 104 105 106 108 110 112 113 119 120 123

127 129 130 133 136 138 139 142 143 147 148: 149 150 151 154

155 156 157 159 160 165 166 167 171 172 173 174 176 177 178

179 181 182 183 190 192 200 203 205 207 210 211 212 213 214

215 218: 219 220 221 222 226 227 229 236 237 241 242 243 244

245 247 249 250 252)

#1 (6)

#1 (8)

#1 (10)

#30 (13 14 35 48 49 57 68 85 93 109 115 117 118 128 131 134 136

153 158 164 168 169 187 189 195 209 217 224 238 239)

#59 (17 18 47 53 54 56 65 91 92 99 107 114 116 121 124 125 126 132

137 140 141 144 145 146 152 161 162 163 170 175 180 184 185

186 191 193 194: 196 198 199 201 202 204 206 208 216 223 225

228 230 231 232 233 234 236 240 246 248 251)

#1 (34)

#5 (41 78 79 112 197)

#3 (52 77 87)

#1 (55)

#3 (58 61 94)

#1 (64)

#1 (70)

#1 (322)

#1 (188)

Cada segmento de carretera representa un grupo (18 grupos para regionId = 3). Los números entre paréntesis son índices de bit que se juntan en cada grupo. El número después de # es el número de bits en cada grupo.

En este ejemplo, 1 grupo es muy grande y contiene tantas como 141 regiones. Esto no es inusual. En general, las regiones en la frontera del mapa juntan más bits que la región en el medio del mapa.

En este ejemplo, el número de bits se ha reducido en media en un factor ~ 3 (~ = 255/84). Por supuesto eso no significa que el tamaño de mapa se reduzca en un factor 3 dado que los restantes bits a codificar tienen más entropía que los bits originales (así más son más difíciles de codificar con bloques de Huffman). Pero los bits de coalescencia de bits aún pueden reducir el tamaño de mapa, quizás significativamente.

Cuando se juntan bits en cada región, los vectores de bit en cada región tienen diferente número de bits como resultado (pero un número idéntico de bits para los vectores de bit de una región-desde dada) como se ilustra en la Figura 15. El sombreado ilustra el ID de región-desde de cada vector de bit.

Una vez que se ha calculado la tabla bitCorrelation, identificar grupos de bits completamente correlacionados es bastante fácil y rápido. Todos los pares de bits donde no encontramos los patrones 01 o 10 están completamente correlacionados y se pueden juntar en 1 bit.

Para dar un ejemplo adicional de coalescencia de bits, la tabla de más adelante da un ejemplo de 6 vectores de bit, cada uno de 15 bits. Estos no han sido juntados.

100111101111100

000001101110100

011001010010111

111110010001011

011000010000011

000000101100000

…

Estos vectores de bit de 15 bits se pueden juntar en solamente 4 grupos – por lo tanto 4 bits por vector de bit antes de la codificación de Huffman. Esto se muestra en la siguiente tabla:

1	2	3	4	5	6	7	8	9	10	11		1 + 3 + 9	2 + 11	5 + 7	4 + 8 + 10
1	00	11	1	1	0	11	1	1	1	00	→	1	0	1	1
0	00	00	1	1	0	11	1	0	1	00	→	0	0	1	1
0	11	00	1	0	1	00	1	0	1	11	→	0	1	0	1
1	11	11	0	0	1	00	0	1	0	11	→	1	1	0	0
0	11	00	0	0	1	00	0	0	0	11	→	0	1	0	0
0	00	00	0	1	0	11	0	0	0	00	→	0	0	1	0

La cabecera de la parte derecha de la tabla muestra qué bits del vector juntó a 4 bits en la parte izquierda de la tabla. Para clarificar, el primer bit del vector juntado a 4 bits representa la 1ª, 3ª y 9ª columnas de la parte izquierda. Por tanto, dado que los bits 1º, 3º y 9º de la parte izquierda (es decir el vector original de 15 bits) son siempre los mismos en el ejemplo anterior, se pueden representar por la 1ª columna del lado derecho de la tabla anterior.

De esta manera, en resumen una tabla de concordancia entre los lados izquierdo y derecho es como sigue:

bit original	bit juntado
0	0
1	1
2	1
3	0
4	0
5	3
6	2
7	1
8	2
9	2
10	3
11	0
12	3
13	1
14	1

Lo que la coalescencia de bits hace eficazmente, es agrupar regiones de destino cercanas y regiones de destino en el mismo sector de ángulo de la región-desde según se ilustra en la Figura 16.

El sombreado en la Figura 16 indica grupos de regiones que se incorporan juntas. Las 2 Figuras (es decir la Figura 16a y 16b) muestran las mismas regiones juntadas desde el punto de vista de 2 regiones-desde diferentes: A y B. Qué regiones se juntan depende del ID de región del nodo desde según se muestra por la Figura. En esta Figura, las 8 regiones se juntan en 4 grupos. La coalescencia de bits es diferente desde el punto de vista de la región-desde A o la región-desde B en estas Figuras. Señalar cómo las regiones rayadas verticalmente (1300) se juntan debido a que están en la misma dirección cuando se ven desde la región-desde A.

Esto se ejemplifica además en la Figura 17 que ilustra que la coalescencia de bits agrupa regiones de destino cercanas y regiones de destino en +/- el mismo sector de ángulo, desde el punto de vista de cada región-desde.

Después de la coalescencia de bits, el número de bits para codificar vectores de bit difiere para cada región. En el caso de Benelux de 255 regiones por ejemplo, el número de bits del vector de bit para cada región es:

región → bits juntados

desde

1 → 57

2 → 93

3 → 18

4 → 12

5 → 63

6 → 75

… recorte …

251 → 21

252 → 46

253 → 117

254 → 80

El método es exacto (no es heurístico). De esta manera encuentra el número ideal de grupos de bits.

En algunas realizaciones, la coalescencia de bits se implementa de una forma sin pérdidas: las regiones solamente se juntan cuando tienen siempre los mismos bits. Pero en otras realizaciones, esto se puede extender para hacerlo con pérdidas; es decir de manera que la recuperación completa de los datos originales no es posible. Tales realizaciones con pérdidas pueden introducir un umbral: si un par de bits difieren en menos de X veces, entonces podríamos juntar los bits si se sustituyen uno o más 0 con un 1 en algunos de los vectores de bit permitiendo a esos vectores de bit ser juntados.

Un ejemplo de tal coalescencia con pérdidas sería para juntar 0000001100 con 0000001110. Una ventaja de tal planteamiento es que hay mejor compresión de los datos pero una desventaja es que se valorarán segmentos de carretera adicionales mediante métodos de encaminamiento usando los vectores de bit dado que unos extra indicarán que un segmento de carretera es parte de una ruta más rápida.

En el caso de coalescencia sin pérdidas, el umbral es 0. Aumentar el umbral permitiría juntar regiones adicionales, a costa de introducir unos pocos bits 1 en los vectores de bit. Pero si el umbral se mantiene bajo, no tendría casi impacto en la velocidad a la que se puede realizar el encaminamiento usando el formato de datos. Por ejemplo, si 2 bits son casi siempre idénticos en miles de vectores de bit, excepto para digamos solamente 1 vector de bit, puede ser aceptable para alterar este único vector de bit donde difiere para hacer posible juntar más bits. Cuanto mayor es el umbral, podemos esperar más compresión del mapa

Reordenación de bits

En algunas realizaciones, una vez que se han juntado bits como se describió anteriormente, se pueden reordenar los bits. La reordenación de bits no reduce el número de bits a codificar (antes de la codificación de Huffman) pero ayuda a hacer la codificación de Huffman más eficiente, por lo tanto reduciendo el tamaño de mapa y también ayuda a reducir el número de códigos distintos en códigos de Huffman de vector de bit, por lo tanto reduciendo el requerimiento de memoria de los árboles de Huffman.

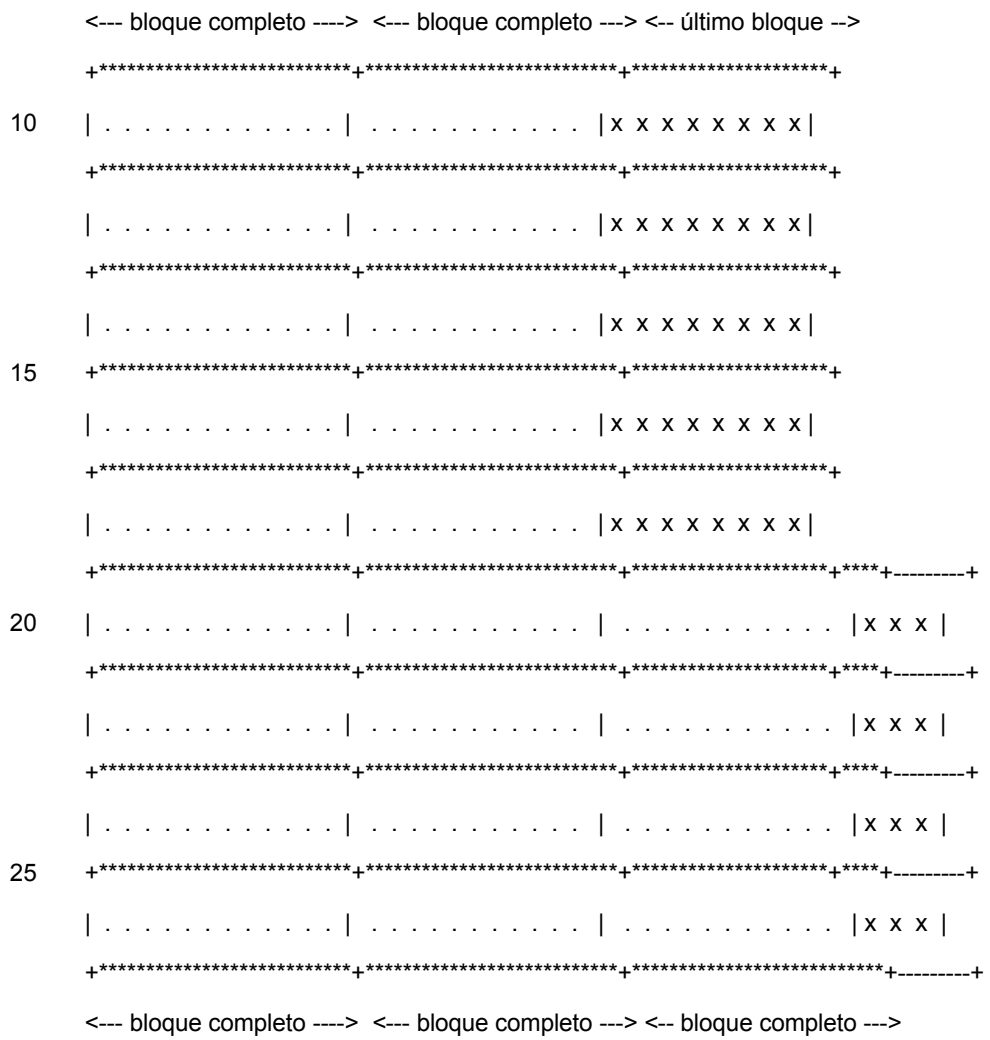
A diferencia de la coalescencia de bit, la ordenación de bit es un método heurístico de manera que encuentra una buena reordenación pero no puede encontrar la mejor.

Los vectores bit se codifican por bloques. El tamaño de bloque es personalizable y se almacena en la cabecera de mapa 800. En la explicación dada más adelante, y como ejemplo, el tamaño de bloque se fija a 11 bits. El número de bits en los vectores de bit no puede ser divisible por 11, así que el último bloque puede ser menor que 11 bits. El

último bloque usa un árbol de Huffman diferente que el de bloques completos de 11 bits. La codificación de cada vector de bit de esta manera codifica:

- varios bloques completos que usan un código de Huffman para bloques de 11 bit (en este ejemplo)
- 1 último bloque que puede usar otro código de Huffman cuando quedan menos de 11 bits.

La imagen siguiente representa los bits para codificar con Huffman por bloques con vectores de bit en 2 regiones (que tienen diferente número de bits después de la coalescencia):



Tener árboles de Huffman para cada región sería probablemente más caro en memoria. Así en algunas realizaciones, se comparte el mismo árbol de Huffman para todas las regiones. Dado que los bits correlacionados son diferentes en cada región, compensa reordenar los bits en cada región, de manera que los bits correlacionados se ponen sistemáticamente en las mismas posiciones en todas las regiones. Así la compartición del árbol de Huffman a través de todas las regiones llega a ser más eficiente.

La tabla para ordenar bits para cada región se almacena en datos de mapa. Señalar que no necesitamos codificar una tabla para juntar bits y una tabla para reordenar bits separadas sino podemos codificar en su lugar solamente 1 tabla que es la composición de ambas transformaciones (coalescencia y reordenación).

El método para reordenar bits en bloques completos procede como sigue: para cada región, encontrar el par de bits más correlacionado. Dado que los bits correlacionados completamente ya han sido juntados, ninguno de los bits restantes está correlacionado completamente. No obstante, algún par de bits está mucho más correlacionado que otro. El par más correlacionado de bits (a, b) se reasigna al bit #0 y #1 en el primer bloque completo:

+**************************+**************************+**************************+---------+

| a b . . . . . | . . . . . | . . . . . |x x x |

+**************************+**************************+**************************+---------+

Como resultado de agrupar los pares de bits más correlacionados, los árboles de Huffman de bloques completos tienen menos códigos distintos (lo cual tiene la ventaja de usar menos memoria para almacenar los datos) y las estadísticas son más sesgadas (haciendo la codificación de Huffman más eficiente). En lugar de contener secuencias aleatorias de bits, los primeros 2 bits por ejemplo contienen en la gran mayoría de los casos o bien 00 o bien 11 pero casi nunca 10 o 01 (lo mismo para otros bits). Sin reordenación de bits, los primeros 2 bits contendrían todo tipo de patrones 00, 01, 10, 11.

Después de reasignar los primeros 2 bits, el método entonces encuentra el siguiente par de bits más correlacionado (c, d) y los reasigna para almacenarlos en los primeros 2 bits del segundo bloque:

+**************************+**************************+**************************+---------+

| a b . . . . . | c d . . . . . | . . . . . |x x x |

+**************************+**************************+**************************+---------+

El método encuentra de nuevo el siguiente par de bits más correlacionado (e, f) y los reasigna para almacenarlos en los primeros 2 bits del tercer bloque:

+**************************+**************************+**************************+---------+

| a b . . . . . | c d . . . . . | e f . . . . . |x x x |

+**************************+**************************+**************************+---------+

Cuando se alcanza el último bloque completo, el método vuelve al primer bloque completo y reasigna el siguiente par de bits más correlacionado (g, h). Si varios pares tiene la misma correlación, un interruptor de enlace selecciona el par más correlacionado con el par previo en el mismo bloque (es decir el par más correlacionado con (a, b)):

+**************************+**************************+**************************+---------+

| a b g h . . . . . | c d . . . . . | e f . . . . . |x x x |

+**************************+**************************+**************************+---------+

El algoritmo del método procede como anteriormente hasta que todos los pares de bits en bloques completos han sido reasignados:

+**************************+**************************+**************************+---------+

|a b g h m n s t y z.| c d i j o p u v A B.| e f k l q r w x C D.|x x x |

+**************************+**************************+**************************+---------+

En este ejemplo, dado que el tamaño de bloque tiene un número impar de bits (11 bits) aún existe un bit no reasignado en cada bloque completo. El método entonces encuentra el bit más correlacionado con ‘z’ y lo almacena en el primer bloque. Entonces encuentra el bit más correlacionado con B y lo almacena en el segundo bloque, etc. hasta que se reasignan todos los bloques completos:

+**************************+**************************+**************************+---------+

|a b g h m n s t y zE | c d i j o p u v A BF| e f k l q r w x C DG|x x x |

+**************************+**************************+**************************+---------+

En este punto, todos los bits están reasignados en bloques completos. El método entonces reasigna los bits en el último bloque intentando agrupar el par de bits más correlacionado como fue hecho para los bloques completos. La reordenación de bits en el último bloque puede ayudar a reducir el tamaño de mapa pero no ayuda tanto como reordenar bits en bloques completos por 2 razones:

- Los bloques completos son más importantes. En este ejemplo, cada código usa 3 códigos de Huffman para bloques completos mientras que usa solamente 1 código de Huffman para el último bloque, así que es normal que los bloques completos contribuyan más al tamaño de mapa total que el último bloque incompleto y es más útil para optimizar la codificación de Huffman de bloques completos.

- Dado que ya escogimos todos los bits más correlacionados para asignarlos en los bloques completos, los bits que quedan por reasignar en el último bloque están menos correlacionados. Así la entropía de los bits en el último bloque es más alta de esta manera que para los bits en los bloques completos. En otras palabras, la codificación de Huffman del último bloque no es tan eficiente como la codificación de Huffman de los bloques completos.

+*****************************+*****************************+*****************************+---------+

| a b g h m n s t y z E | c d i j o p u v A B F | e f k l q r w x C D G | H I J |

+*****************************+*****************************+*****************************+---------+

Se debería recordar que el mismo árbol de Huffman se usa para todos los bloques completos de todas las regiones. La codificación del vector de bit en el ejemplo anterior de esta manera codifica con los mismos códigos de Huffman todos los bloques completos y finalmente codifica el último bloque con un códec de Huffman diferente:

+*****************************+

| a b g h m n s t y z E |

+*****************************+

| c d i j o p u v A B F | (3 bloques completos que usan el mismo códec de Huffman)

+*****************************+

| e f k l q r w x C D G |

+*****************************+

+---------+

| H I J | (1 último bloque que usa otro códec de Huffman)

+---------+

La razón para que el método reasigne los primeros bits de cada bloque (en lugar de reasignar todos los bits en el primer bloque antes de saltar al segundo bloque) debería estar más clara cuando se ve la figura anterior. Dado que se usa el mismo códec para todos los bloques completos, es deseable tener todos los códigos para todos los bloques tan idénticos como sea posible. Si fuéramos a reasignar todos los bits en el primer bloque, luego todos los bits en el segundo bloque (etc.), entonces cada bloque tendría patrones muy diferentes: el primer bloque tendrían la mayoría de bits correlacionados, el segundo bloque tendría menos bits correlacionados, el tercer bloque tendría incluso menos bits correlacionados, etc. Sería ser posible crear varios códec de Huffman para cada columna de los bloques completos pero eso se considera que es demasiado caro en memoria. El método perfilado hasta aquí funciona bien mientras que se comparte el mismo códec de Huffman para todos los bloques completos.

Otras posibles optimizaciones

Páginas de páginas

La información de página de vector de bit almacena un campo delta que se usa para encontrar el desplazamiento del comienzo de cada página de vector de bit. El campo delta se almacena usando un interpolador lineal de carretera del desplazamiento. Pero los desplazamientos no son muy lineales en carreteras debido a que los vectores de bit alrededor de los ID de nodo pequeños (nivel 0, autopistas) requieren más bits que los vectores de bit alrededor de los ID de nodo altos (nivel 5, carreteras de destino).

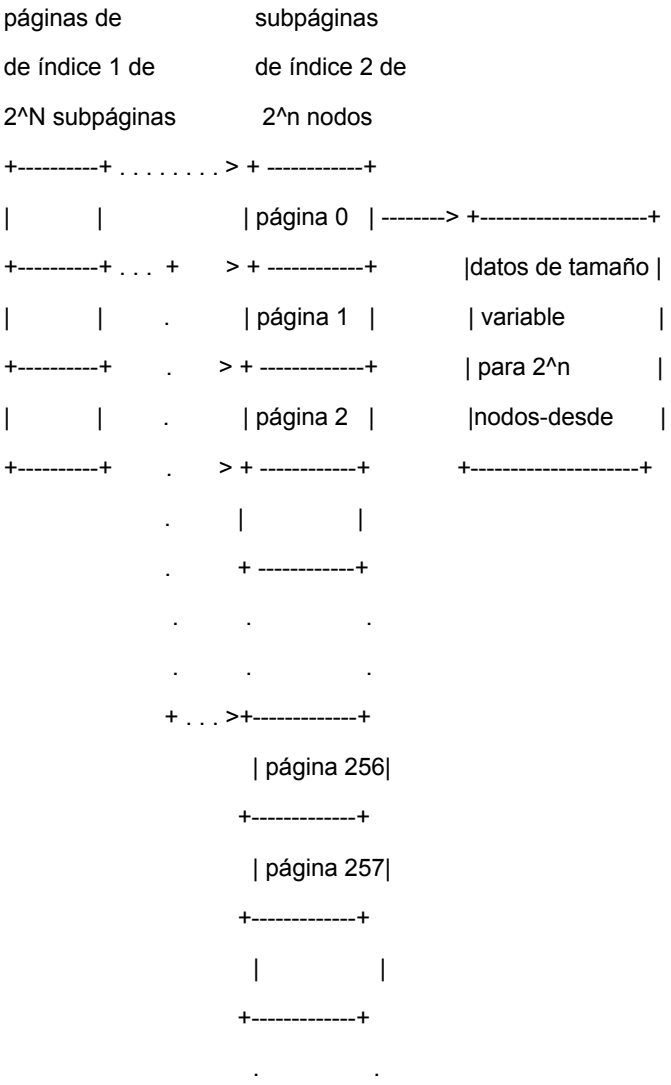
La codificación de información de página de vector de bit no es tan ventajosa como se puede desear debido a que el interpolador no predice de manera precisa el desplazamiento real. Mejorando el interpolador, sería posible mejorar la codificación de la tabla de información de página de vector de bit para hacerla más eficiente.

Algunas realizaciones de la invención pueden usar páginas de vector de bit (y posiblemente páginas de región) que tienen 2 niveles en lugar de solamente 1 nivel. Las páginas de 2^n nodos para agrupar los datos (llamémoslas subpáginas) se pueden agrupar en páginas de 2^n subpáginas.

Por tanto, los parámetros de interpolación lineal se almacenarían por página en lugar de globalmente (en la cabecera). Por ejemplo, el índice de nivel 2 podría agrupar 2^4 = 16 nodos como antes en subpáginas, y el índice 1 podría agrupar 2^{16} = 1023 de esas subpáginas en páginas. La interpolación lineal entonces ocurre para 1024*16 =

16K nodos en lugar de en el recuento de nodos entero (40.000.000 de nodos en un mapa de Europa Occidental y Central) así la interpolación lineal de desplazamientos de tamaño variable llega a ser mucho más precisa y los campos delta en índice 2 son más pequeños de esta manera.

El tamaño de índice 1 extra es pequeño si las páginas son grandes (lo bastante pequeño para caber en la memoria). Ser capaz de caber dentro de una memoria de un dispositivo es ventajoso dado que no debería ralentizar el encaminamiento usando los datos. En lugar de almacenar el tamaño de página medio en la cabecera, el tamaño de página medio entonces se podría almacenar para cada entrada en la tabla de índice 1.



Intercalar páginas de vectores de bit

Como se describió anteriormente, el interpolador para desplazamiento de página no es preciso debido a que los vectores de bit son diferentes para carreteras importantes (muchas marcas no triviales) y carreteras secundarias (muchas marcas triviales). Una forma simple para hacer el interpolador más lineal es intercalar páginas de diferentes niveles de red y esto se puede usar en realizaciones de la invención.

El fichero descrito en las realizaciones anteriores almacena las siguientes páginas:

0

1

2

3

4

…

n-3

n-2

n-1

En otras realizaciones que pueden ser más eficientes es posible almacenar de una manera intercalada como sigue:

0

n-1

1

n-2

2

n-3

3

n-4

…

Para acceder a la página # x (por ejemplo mediante una aplicación de planificación de ruta) se accede a la página cargando la página # x' donde:

- x' = 2*x (cuando x es par)
- x' = 2*(x – (n - 1) (cuando x es impar)

Tal realización debería ser ventajosa dado que reducirá el tamaño de indexación en varios bits por página. No obstante, los datos se pueden agrupar peor para almacenamiento en caché lo cual puede ralentizar el acceso a los datos (menos bits en la caché del sistema de ficheros).

No almacenar los ID de región para todos los nodos

Los ID de región no necesitan ser almacenados para nodos en las calles cortadas y nodos con 2 segmentos de carretera unidos. Esto nodos se pueden ignorar para el encaminamiento. El ir a uno de estos nodos se puede transformar en ir a sus nodos de decisión contiguos.

Almacenar información extra a nivel de página o a nivel de nodo

Mirando los datos de mapa, hay montones de páginas de vector de bit que solamente contienen los vectores de bit triviales 000…000 o 111…111. Algunas realizaciones pueden almacenar 1 bit para cada página para marcar esas páginas, entonces almacenar vectores de bit en esas páginas puede ser más eficiente dado que solamente necesitamos un único bit para cada vector de bit para indicar si es 000…000 o 111…111. No solamente reducirá el tamaño de las páginas que solamente contienen vectores de bit triviales, sino que también hará el árbol de Huffman de los códigos de vector de bit más optimizado para páginas que tienen vectores de bit no triviales (el número de bits para indicar vectores no triviales se reducirá dado que la frecuencia de esos códigos aumentará significativamente en porcentaje). En el nivel de red de más detalle (por ejemplo el nivel 3), la mayoría de las páginas solamente contienen vectores de bit triviales de manera que puede haber solamente 1 vector de bit por página en alrededor de la mitad de las páginas.

Almacenar vectores de bit solamente en nodos de decisión

Como se trató anteriormente, algunas realizaciones no podrían almacenar vectores de bit para nodos con 1 o 2 segmentos de carretera unidos. No obstante, otras realizaciones pueden ser más agresivas y generalizar la idea solamente a los vectores de bit alrededor de los nodos de decisión.

Los conceptos de **nodo de decisión** y **segmento de carretera de decisión** se introducen debido a que pueden ser ventajosos en la codificación de datos de mapa: los vectores de bit no necesitan ser codificados para segmentos de carretera de no decisión como se trata ahora.

- Un **nodo de decisión** es un nodo donde hay un segmento de carretera entrante para encaminamiento de manera que hay múltiples opciones para salir del nodo (sin hacer un giro en U).
- Un **nodo de no decisión** es un nodo que no es un **nodo de decisión**. Así con independencia de desde dónde viene el encaminamiento, hay siempre solamente una manera de salir del nodo.
- Un **segmento de carretera de decisión** es un segmento de carretera que es legal a fin de salir de un **nodo de decisión**.
- Un **segmento de carretera de no decisión** es un segmento de carretera que no es un **segmento de carretera de decisión.**

Todos los segmentos de carretera de decisión de este modo están alrededor de nodos de decisión. Pero no todos los segmentos de carretera alrededor de nodos de decisión son segmentos de carretera de decisión. Todos los segmentos de carretera alrededor de nodos de no decisión son segmentos de carretera de no decisión.

Los vectores de bit se codificarán solamente para segmentos de carretera de decisión. Los vectores de bit están implícitos (000…000 o 111…111) para los segmentos de carretera de no decisión dado que las técnicas de encaminamiento que usan los datos pueden hacer una determinación a partir de información ya presente en los segmentos de carretera.

¿Cómo determinar si un nodo es un nodo de decisión? El criterio lo podemos reducir a:

isDecisionNode = (lineCount >=3) && (lineGoingOutCount >= 2)

Donde:

lineCount: es el número total de líneas unidas al nodo

ignorando tipos de línea no encaminables (líneas ferroviarias, líneas de referencia),

ignorando líneas cerradas en direcciones e ignorando

carreteras no atravesables (áreas residenciales).

lineGoingOutCount: es el número de líneas unidas a

un nodo que es legal tomar para salir del nodo.

Si es o no legal tomar un segmento de carretera para salir del nodo depende de los atributos del segmento de carretera:

- tipo de segmento de carretera (los segmentos de carretera de referencia y línea ferroviaria son siempre ilegales)
- flujo hacia delante/hacia atrás de vía (almacenado en marcas de segmento de carretera)
- atributo no atravesable en marcas de segmento de carretera (los segmentos de carretera no atravesables no tienen ningún vector de bit)

En algunas realizaciones ignorar los segmentos de carretera de no decisión puede desechar aproximadamente el 40% de los segmentos de carretera. Se ha encontrado que este porcentaje es bastante coherente con independencia del mapa. Evitar la codificación del 40% de los vectores de bit es ventajoso pero ahorra menos del 40% del tamaño de mapa, dado que elimina mayoritariamente vectores de bit triviales.

La eliminación de vectores de bit alrededor de nodos con menos de 3 segmentos de carretera unidos (nodos ficticios) elimina vectores de bit no triviales, así que el ahorro de tamaño de mapa para esta categoría de segmentos de carretera de no decisión puede ser mayor que para los segmentos de carretera de no decisión. Por otra parte, el filtrado requiere un decodificador (tal como un método de encaminamiento que usa el mapa) para decodificar los tipos de segmento de carretera y marcas de segmento de carretera de segmentos de carretera y aplicar una lógica sobre ellos a fin de averiguar los vectores de bit que están implícitos, lo cual puede ralentizar el proceso.

En realizaciones teóricas también podríamos mirar las maniobras (es decir restricciones de giro) para decidir si un segmento de carretera que sale es legal, pero tal técnica añade complejidad. Ignorar las maniobras significa que la realización puede codificar más vectores de bit que los estrictamente necesarios pero se logra una simplificación en el método.

Ejemplo de nodos de no decisión

a - - < - > - - b - - < - > - - c

En este ejemplo, (b) está unido a 2 segmentos de carretera navegables en ambas direcciones. (b) no es un nodo de decisión, debido a que hay < = 2 segmentos de carretera unidos.

Así los vectores de bit de ambos segmentos de carretera que salen del nodo (b) no se codificarán. El decodificador puede fijarlos implícitamente a 111…111.

a - - < - > - - b - - < - > - - c

|

^

|

d

Las flechas > en este ejemplo muestran la dirección legal de flujo. (b) no es un nodo de decisión, debido a que hay solamente una salida. De esta manera ninguno de los segmentos de carretera alrededor de (b) necesita vectores de bit.

Los vectores de bit para el segmento de carretera (b) - > (c) que salen del nodo (b) no está codificados, implícitamente serán 111…111. Los vectores de bit para segmentos de carretera ilegales que salen de (b) no están codificados tampoco, serán implícitamente 000…000.

Ejemplos de nodos de decisión

a - - < - > - - b - - < - > - - c

|

^

|

d

(b) es un nodo de decisión debido a que cuando se viene desde (d), hay una elección: el encaminamiento puede continuar hacia (a) o hacia (c).

Señalar que en este ejemplo, cuando se viene desde (a), no hay elección: el encaminamiento solamente puede continuar hacia (c). Pero el nodo (b) es todavía un nodo de decisión debido a que hay al menos una elección cuando se viene desde (d) de manera que los vectores de bit se deberían almacenar para los 2 segmentos de carretera alrededor del nodo (b).

Los vectores de bit se almacenan para el segmento de carretera (b) - > (a) y el segmento de carretera (b) - > (a) dado que son segmentos de carretera de decisión.

Un vector de bit no se almacena para el segmento de carretera (b) - > (d) dado que este segmento de carretera es ilegal de tomar según el flujo de tráfico hacia atrás/hacia delante. Es implícitamente 000…000.

Operación lógica OR de vectores de bit

Supongamos que un nodo tiene 3 segmentos de carretera unidos y que los primeros 2 segmentos de carretera decodificados tienen los siguientes vectores de bit:

0000000000000000000

0000000000000000000

Entonces el tercer vector de bit no necesita ser codificado debido a que solamente puede ser:

1111111111111111111

Esto puede funcionar solamente si los vectores de bit de los segmentos de carretera alrededor del nodo pasan a ser en este orden: todos los vectores de bit 000…000 y el último es 111…111. En la práctica parece que pasa bastante frecuentemente en la red de nivel de más detalle (por ejemplo nivel 0) (que es donde están la mayoría de los segmentos de carretera).

Aceptando que los primeros 2 vectores de bit sean:

0000000000000000110

0000000000000000010

Entonces el tercer vector de bit solamente puede tener todos sus bits fijados a 1 excepto 2 bits que son desconocidos y necesitan ser codificados de alguna manera.

1111111111111111??1

Dado que en el ejemplo anterior, la mayoría de los bits ya se conocen, debería ser posible usar esta información para codificar los 2 bits desconocidos más eficientemente que codificar el vector de bit entero. De esta manera, en este ejemplo, solamente es necesario codificar 2 bits.

Un posible esquema de decodificación rápido que usa esta propiedad sería calcular una máscara de bit de todos los vectores de bit codificados en los segmentos de carretera de los nodos actuales mediante la operación lógica OR de todos los vectores de bit previos en el segmento de carretera. Usando el mismo ejemplo que anteriormente, si un nodo tiene 3 segmentos de carretera unidos y si los 2 segmentos de carretera previos tienen los siguientes vectores de bit:

0000000000000000110

0000000000000000010

… entonces la máscara de bit OR es:

0000000000000000110

Tomado el 3º y último vector de bit para codificar alrededor del nodo como:

1111111111111111001

En lugar de codificar el código de Huffman de 1111111111111111001 que puede ser raro, el codificador es libre de codificar cualquier otro código (otherCode) siempre que:

value_to_encode = ~ bitMask | otherCode

En este ejemplo, otherCode = 0000000000000*0000000 capacita dado que:

value_to_encode = 1111111111111111001 = 0000000000000*0000110 | 0000000000000*0000000

Codificar 0000000000000000000 es mucho más eficiente que codificar 1111111111111111001 dado que 0000000000000000000 es mucho más frecuente. La decodificación es rápida, dado que la decodificación solamente necesita calcular la máscara de bit (u operación) siempre que decodifica un vector de bit y aplicar la máscara de bit al último vector de bit decodificado:

vector de bit real = máscara de bit & vector de bit decodificado

= ~ 0000000000000000110 & 0000000000000000000

= 1111111111111111001

Esta optimización funciona bien si los segmentos de carretera alrededor de los nodos se almacenan de tal manera para tener el de mayoría de 1 en el vector de bit en una región final. No obstante, esto puede resultar difícil:

- la clasificación de los segmentos de carretera alrededor de un nodo se realiza antes de calcular los vectores

de bit (a menos que usemos la información de bit a posteriori para volver a codificar)

- los segmentos de carretera ya están clasificados por nombres de carreteras. Es posible clasificar los segmentos de carretera alrededor de nodos cuando se identifican nombres de carreteras.

Usar niveles de red

Debería haber una correlación fuerte entre vectores de bit y niveles de red de segmentos de carretera alrededor de nodos-desde: el encaminamiento a otra región preferirá de manera general tomar la red de nivel más tosco (el nivel 1 será generalmente la red de nivel más tosco y el 0 en la red de nivel de más detalle).

El siguiente ejemplo representa una intersección con segmentos de carretera en niveles de red diferentes:

```
                4
                |              La línea (2) - > (1) está en la red 4 (importante)
                |              La línea (2) - > (3) está en la red 5 (secundario)
  1 ======= 2 ======= 3        La línea (2) - > (3) está en la red 5 (secundario)
```

Los patrones de bit van a ser probablemente como sigue:

```
2   1   ????????????
2   3   ????????????
3   4   000000000000
```

No almacenar vectores de bit para nodos en niveles de red de más detalle

El encaminamiento casi nunca va a través de niveles de red de segmentos de carretera no importantes, tales como nivel de red 5, excepto cerca del punto de destino o de inicio. De esta manera, es posible no almacenar vectores de bit en el nivel 5 y por tanto, debería haber un impacto mínimo en el encaminamiento, incluso aunque el número de segmentos de carretera en nivel 5 sea grande. El encaminamiento en la mayoría de los casos solamente explorará pocos segmentos de carretera en nivel de red menos importante en el inicio y el final de una ruta dado que alcanzará rápidamente algunos niveles de red de segmento de carretera más importantes, y los datos de aceleración de búsqueda en esos nodos casi siempre dirán al encaminador que omita un segmento de navegación que vuelve al nivel de red menos importante y usar un segmento de navegación que conduce a o permanece en los niveles de red más importantes. Esto es cierto en el nivel de red más importante dado que el nivel de red se usa cuando el destino está todavía lejos.

Además de dejar caer vectores de bit en el segmento de carretera menos importante (por ejemplo nivel 5), también podríamos no codificar los ID de región en esa red.

Transformar unos pocos ceros en unos en vectores de bit

Esto también es un esquema de compresión con pérdidas.

Si un vector de bit sólo contiene solamente un 0 (o posiblemente menos de un umbral pequeño), entonces puede ser posible transformarlo en 1 si provoca (es decir ajustar ese segmento de carretera para ser parte de una ruta más rápida):

- un vector de bit trivial 111…111 (dado que los vectores de bit triviales se codifican de una forma más compacta que los vectores de bit no triviales)
- o un vector de bit ya encontrado en la historia de la página de vectores de bit actuales (dado que esos también están codificados más eficientemente)

La transformación de ceros en unos en un vector de bit no afecta la salida del encaminamiento, solamente la hace más lenta teniendo en cuenta el encaminamiento de más segmentos de carretera (debido a que ahora se fijan para ser parte de la ruta más rápida). No obstante, algunas realizaciones de la invención pueden emplear este método – particularmente si el ahorro de mapa es grande con un impacto de rendimiento pequeño en términos de velocidad de encaminamiento.

Encaminamiento

La Figura 18 muestra un mapa 1600 que cubre un área y que tiene un nodo de inicio 1602 y un nodo de destino 1604 junto con una pluralidad de segmentos de carretera que se han explorado por el método de búsqueda A* de la técnica anterior. La ruta seleccionada se muestra por la línea de puntos 1606.

La Figura 19 muestra un mapa 1650 que cubre la misma área que el mapa de la Figura 18 y también que muestra el mismo nodo de inicio 1602 y nodo de destino 1604. La Figura también destaca las carreteras que fueron exploradas usando una realización de la presente invención que utilizó la misma red de carreteras y el mismo criterio que se usó para generar la ruta de la Figura 18 (por ejemplo ambas que se recorren en coche, deseando usar la velocidad como la función de coste a minimizar, etc.). La ruta seleccionada por el método se muestra de nuevo con el segmento de carretera de puntos 1606 y se debería señalar que el segmento de carretera es el mismo que el calculado en la Figura 18. Por tanto, se señala que las rutas calculadas por realizaciones de la invención actual se pueden denominar exactas matemáticamente y encuentran la ruta mejor/óptima con respecto al modelo de coste dado que fue seleccionado para ser minimizado. De esta manera, será evidente que los segmentos de carretera explorados por la realización de la invención actual son significativamente menores en comparación con la técnica anterior. Por tanto, el método de la realización de la invención es más rápido, y generalmente significativamente más rápido, que el método A* de la técnica anterior.

Se debería señalar también que según se acerca la ruta 1606 al destino 1604 se exploran más rutas en comparación con el inicio: es decir cuando la ruta 1606 pasa más allá del punto 1652 se exploran carreteras distintas de la ruta del menor coste. Una explicación de esto es que el método ha conmutado desde la región de nivel 0 a la región de nivel 1 en el punto 1652. Se debería ver además que se exploran segmentos de carretera adicionales en las inmediaciones del destino 1604. De nuevo, esto se puede explicar señalando que el método de planificación de ruta conmutará desde el nivel 1 al nivel 2 en estas inmediaciones.

De esta manera, cuando se encamina a larga distancia, típicamente habrá solamente una ruta más rápida. No obstante, según conmuta el método a un nivel de más detalle (por ejemplo desde el nivel 0 al 1) puede haber más de una ruta que se marque como la de 'coste más bajo' y por lo tanto necesita ser explorada.

Una vez que se han creado los datos de mapa y calculado y almacenado los vectores de bit, se pueden usar para calcular rutas entre dos puntos.

La Figura 20 muestra el método A* de la técnica anterior en el que una red se explora por una aplicación de planificación de ruta en la que se toma una decisión en un nodo para cuyo segmento de carretera explorar más. A fin de lograr esto el coste de viajar al siguiente nodo se añade al coste estimado de moverse desde el siguiente nodo al objetivo. El trayecto que tiene el mínimo coste desde un nodo dado entonces se explora más. Los métodos mantienen el total del coste que se ha incurrido hasta ahora según el método pasa y en cada iteración se considera el coste mínimo.

Por ejemplo, empezando en el nodo B, se puede ver que ambos nodos C y D están alejados 2 unidades de coste de B, el nodo de inicio. No obstante, desde el nodo D el coste estimado de alcanzar el nodo F es de 3 unidades y desde el nodo C el coste estimado de alcanzar el nodo F es de 5 unidades. De esta manera, el método A* exploraría la ruta al nodo D con preferencia a la ruta al nodo C dado que el coste al siguiente nodo más el coste estimado al objetivo es menor para el segmento de carretera al nodo D. (es decir BC = 2 + 5 = 7 mientras que BD = 2 + 3 = 5).

Cuando la consideración de la planificación de ruta se realiza usando los vectores de bit de coste mínimo de realizaciones de la presente invención, los bits para cada segmento de carretera (es decir la columna 704 de la Figura 10) identifican qué segmentos de carretera son candidatos para ser parte de un trayecto de coste mínimo a una región de destino. Por tanto, el método A* descrito en relación con la Figura 20 se modifica a aquél mostrado en la Figura 21.

Típicamente la planificación de ruta se realiza usando los vectores de bit como se describe en relación con la Figura 21. No obstante, algunas realizaciones de la invención pueden permitir a una planificación de ruta recurrir a A* (u otro método de la técnica anterior) en diversas circunstancias. Diferentes realizaciones pueden utilizar cualquiera de los siguientes: se omite información de vectores de bit (permitiendo por ello al sistema funcionar incluso en ausencia de los vectores de bit); se usa una función de coste; el encaminamiento distinto de la función de coste para el que fueron calculados los vectores de bit (típicamente la ruta más rápida pero no necesariamente así); un usuario especifica un criterio diferente a aquéllos usados para calcular previamente los vectores de bit (por ejemplo un usuario desea evitar autopistas); el tipo de vehículo es diferente a aquél usado para el cálculo previo de los vectores de bit (por ejemplo el usuario va a pie y no en coche); y la ruta está por debajo de una longitud umbral (por ejemplo los puntos de inicio y fin de la ruta están dentro de la misma región).

Dos dígitos extra, 1800, 1801 se muestran en la Figura 21 en comparación con la Figura 20. Adicionalmente, las regiones se indican en la Figura mediante los segmentos de carretera troceados 1802 y 1804. El '1' 1800 indica que el segmento de carretera BD es parte de un trayecto de coste mínimo desde el nodo B a la región en la que está situado el nodo objetivo. El '0' 1801 indica que el segmento de carretera BC no es parte de ningún trayecto de coste

mínimo desde el nodo B a la región en la que se sitúa el nodo objetivo.

Por tanto, cuando se empieza desde el nodo A las realizaciones de la invención usarían el A* perfilado anteriormente para explorar los nodos desde A. No obstante, una vez que el nodo B ha sido alcanzado no hay necesidad de usar la exploración más dado que hay una anotación explícita de que solamente el segmento de carretera BD es una parte posible del trayecto de coste mínimo. Por tanto, una vez que el método A* ha sido usado y encontrado el trayecto de coste mínimo seleccionará posteriormente el trayecto de coste mínimo mirando los bits relevantes dentro de los vectores de bit.

No obstante, los datos de coste mínimo pueden identificar más de un trayecto de coste mínimo entre un par de las regiones si existen diferentes trayectos de coste mínimo entre el par de regiones en momentos diferentes. Por lo tanto, un encaminamiento entre un nodo inicial y un nodo objetivo (origen y destino) puede encontrar más de un trayecto de coste mínimo y tener que decidir entre trayectos de coste mínimo que compiten, por ejemplo si ambos dígitos 1800 y 1801 para los segmentos de carretera BD y BC tienen un valor de “1” que indica que cada segmento de carretera es parte de un trayecto de coste mínimo en un cierto momento. En esta realización, el vector de bit no identifica la hora en la que cada segmento de carretera es parte de un trayecto de coste mínimo. Por consiguiente, el algoritmo de encaminamiento lleva a cabo un análisis de coste para ambos trayectos de coste mínimo posibles en base a una hora a la que se recorren los segmentos. Esta hora se puede determinar a partir de una hora de salida desde el nodo de origen/inicio.

Asociado con cada segmento está un perfil de velocidad, como se muestra en la Figura 10a, de cómo la velocidad esperada de viaje a través del segmento varía con la hora. A partir de estos perfiles de velocidad, se puede determinar la velocidad esperada en la hora relevante y ésta se puede usar para determinar el coste de recorrer a lo largo de ese segmento de carretera a esa hora. Sumando los costes de recorrer los segmentos de carretera de cada trayecto de coste mínimo, se puede determinar el coste de cada trayecto de coste mínimo. El dispositivo de navegación entonces puede seleccionar para la ruta el trayecto de coste mínimo que tiene el coste más bajo para esa hora.

La ruta determinada se puede mostrar en el visualizador 206. La Figura 23 es un ejemplo de cómo se puede mostrar la ruta en los datos de mapa.

En esta realización, el visualizador comprende una indicación de la velocidad esperada en al menos alguno de los segmentos navegables de la ruta. Para esta realización, estas indicaciones tienen un coloreado de la ruta y una barra de tiempo 2000 al lado de la imagen del mapa para mostrar la velocidad esperada junto a la ruta. Por ejemplo, los segmentos navegables pueden ser coloreados de un primer color, por ejemplo verde, si la velocidad esperada está por encima de un valor umbral y un segundo color, por ejemplo rojo, si la velocidad esperada está por debajo de un segundo valor umbral.

Un método alternativo de determinación de una ruta se tratará ahora con referencia a las Figuras 10a y 24 a 31. En lugar de determinar un coste para las rutas a una hora particular, se podría determinar un perfil de coste que representa costes que cambian con la hora. En esta realización, se usan múltiples valores en lugar de un único valor del perfil de velocidad para cada segmento navegable para determinar un perfil de coste para las rutas.

Por ejemplo, con referencia a la Figura 21, en lugar de tener un valor único para cada segmento de carretera BC, CE, etc., el coste de cada ruta es una distribución de cómo varía el coste con la hora. Estas distribuciones de coste se suman para determinar un perfil de coste para la ruta. La propagación de perfiles de coste a través de un árbol también se muestra en la Figura 10a.

En esta realización, cada segmento navegable tiene un perfil de velocidad asociado con el mismo que comprende 12 compartimentos de tiempo para cada hora, cada compartimento de tiempo que contiene una velocidad esperada de viaje a través del segmento navegable para esa hora. Se determina un coste para la velocidad esperada de cada compartimento y se suman los costes para cada compartimento relevante para los segmentos navegables de una ruta para determinar el perfil de coste. Los valores que se suman para cada segmento pueden no ser de los compartimentos para la misma hora ya que la hora esperada de viaje a lo largo de cada segmento variará dependiendo de cuándo se predice al usuario llegar a cada segmento navegable. Por ejemplo, se puede predecir que el usuario tardará 5 minutos en viajar a lo largo del segmento navegable BD, por lo tanto, los valores de compartimento sumados juntos para los segmentos BD y DF pueden ser los valores de compartimento que están separados en 5 minutos, es decir el valor de compartimento de tiempo T para BC se añade al valor de compartimento de tiempo T+5 para DF.

Solamente los segmentos navegables que son parte de un trayecto de coste mínimo se exploran sobre segmentos navegables que compiten que no son parte de un trayecto de coste mínimo. Por lo tanto, para que ambos trayectos BDF y BCEF sean considerados, ambos trayectos se deben identificar por los datos de coste mínimo como trayectos de coste mínimo.

Si ambos trayectos BDF y BCEF son trayectos de coste mínimo a diferentes horas, entonces en el nodo F en el que se cruzan las dos rutas, los perfiles de coste para BDF y BCEF necesitan ser retenidos en algún grado y

posiblemente propagados adicionalmente. Para lograr esto, el procesador lleva a cabo cálculos que son equivalentes a superponer los dos perfiles de coste, C1 y C2 en la parte superior uno de otro y determinar una distribución de límite superior UB (mostrada en líneas continuas) y una distribución de límite inferior LB (mostrada en líneas discontinuas). Esto se muestra en las Figuras 30 y 31. Las distribuciones UB y LB entonces se usan para propagación adicional permitiendo un perfil de coste máximo y mínimo para cada ruta a ser determinada.

Una propagación adicional de la distribución de límite superior y la distribución de límite inferior ocurre de una manera similar con valores de coste para rutas navegables posteriores que se añaden a esos perfiles. Si, las rutas exploradas se dividen y entonces se cruzan de nuevo en un nodo más tarde similar al nodo F, entonces los dos perfiles de límite superior y los dos perfiles de límite inferior para cada trayecto entrante se procesan de nuevo solapando los dos perfiles de límite superior para determinar un único perfil de límite superior nuevo y solapando los dos perfiles de límite inferior para determinar un único perfil de límite inferior nuevo. De esta manera, se mantiene un aumento en el número de perfiles en un nivel que no perjudica indebidamente el tiempo que tarda en determinar una ruta o requerir cantidades grandes de memoria (ya que la memoria es limitada en los PND).

En otra realización, solamente se retiene uno de los perfiles de límite superior e inferior para propagación adicional a lo largo de la ruta. En una realización adicional, se determina una media de los dos perfiles entrantes para provocar un perfil promediado para propagación adicional a lo largo de la ruta.

Una vez que se ha determinado un perfil de coste para una ruta entera, se selecciona uno de los perfiles superior e inferior como el perfil de coste final, el cual se usa para determinar una visualización en el dispositivo de navegación, como se explicará ahora.

Las Figuras 24 y 25 ilustran formas en las que se pueden mostrar múltiples valores del perfil de coste.

En la Figura 24, el perfil de coste se ilustra mostrando la hora de salida desde el origen y la hora de llegada al destino junto con barras de tiempo para horas de salida discretas separadas por una cantidad predeterminada, en esta realización 15 minutos. Como la barra de tiempo 2000 en la Figura 23, cada barra de tiempo está codificada por colores para indicar las velocidades esperadas a lo largo de diferentes partes de la ruta. Se proporcionan las flechas 2001 y 2002, las cuales cuando se seleccionan por un usuario hacen al visualizador mostrar pares de hora de salida y hora de llegada y barras de tiempo para horas de inicio anterior o posterior (la selección de la flecha izquierda 2001 que causa que horas de inicio anteriores sean visualizadas y la selección de la flecha de la derecha 2002 que causa que horas de inicio más tardías sean visualizadas). Seleccionar una barra de tiempo puede hacer que un visualizador, como el mostrado en la Figura 23, sea mostrado para la hora de inicio seleccionada.

En la Figura 25, el perfil de coste se muestra como un gráfico continuo sobre un periodo fijo de tiempo, en este ejemplo durante 168 horas (equivalente a una semana). Este gráfico se muestra junto con un mapa que ilustra las rutas desde el origen al destino que contribuyen a formar este perfil de coste. El gráfico y las rutas se codifican por colores (como se ilustra por las líneas continuas y discontinuas) de manera que un usuario puede determinar qué rutas van a ser usadas a qué horas. Por ejemplo, el gráfico entre 7 horas y 40 minutos y 9 horas y 15 minutos está coloreado para mostrar que en el inicio de la ruta se debería seguir la ruta discontinua 2003 en lugar de la primera parte de la ruta de puntos 2004. Tal visualizador permite al usuario ver fácilmente cómo el coste de viajar entre el origen y destino cambia con la hora y cómo afecta esto a la ruta.

Las Figuras 26 a 28 ilustran una realización alternativa. En esta realización, se ha determinado un perfil de coste, como se describió anteriormente, pero solamente se muestra una imagen de una ruta determinada para una hora de viaje especificada por el usuario. No obstante, en el visualizador está un recuadro 2010 que muestra el día y hora para los que es aplicable la ruta visualizada. Las flechas proporcionadas en cualquiera de los dos lados del día y la hora se pueden seleccionar por un usuario mientras que ve la imagen de los datos de mapa mostrando la ruta determinada para cambiar el día y/o la hora. Cambiar el día y/o la hora puede hacer a la ruta cambiar y el visualizador se actualiza para visualizar la nueva ruta así como cualquier cambio en el tiempo para recorrer la ruta y la distancia recorrida. Ejemplos de estas visualizaciones actualizadas se muestran en las Figuras 27 y 28.

La actualización del visualizador ocurre en “tiempo real”, por ejemplo del orden de milisegundos, por ejemplo en menos de 1000 milisegundos y preferiblemente menos de 500 milisegundos. Tales tiempos de recalculo rápidos se pueden lograr a través del uso de los datos de coste mínimo procesados previamente y/o debido a que las rutas para estas otras horas ya se han determinado a través de una búsqueda de perfil.

La Figura 29 ilustra una función adicional del dispositivo de navegación que usa búsquedas de perfil. En esta realización, en la selección de una hora de viaje, por ejemplo una hora de inicio, el procesador del dispositivo de navegación puede comparar el coste de viajar a esa hora con valores del perfil de coste en una ventana alrededor de esa hora de viaje, por ejemplo, 30 minutos en cualquiera de los dos lados de la hora de viaje. Si el procesador encuentra una hora de viaje dentro de esa ventana que tiene un coste inferior entonces el procesador puede hacer al visualizador mostrar una indicación de la hora de viaje que da un resultado mejor. Por ejemplo, en la realización ilustrada, se muestra una nota 2015 que informa al usuario que si viaja 15 minutos más tarde el tiempo de viaje será más corto, en este caso en 10 minutos.

A fin de realizar el encaminamiento descrito hasta ahora, se obtiene la información de vector de bit desde el fichero lateral codificado descrito en relación con las Figuras 11 a 13 usando un proceso de decodificación.

La salida del procesamiento previo descrito anteriormente comprende:

- una asignación de regiones jerárquicas para la mayoría de los nodos, y
- una asignación de vectores de bit para los segmentos de carretera salientes en estos nodos.

Carga de mapa

Comprobaciones de coherencia

Cuando se cargan datos de mapa, el conjunto de ficheros laterales se debería presentar en el directorio de mapa, de otro modo el decodificador desactivará los datos de aceleración de búsqueda de manera que la búsqueda de ruta se lleva a cabo sin datos de aceleración de búsqueda. Hay varias comprobaciones que ayudan a asegurar la integridad de los datos, las cuales se enumeran ahora.

Los ficheros laterales siguen un convenio de nomenclatura.

El número de niveles se da por el número de ficheros laterales, pero también se almacena en la cabecera para cada fichero lateral y es igual al número de fichero lateral.

Cada fichero lateral almacena el número de regiones en su nivel respectivo, que es el mismo que se especifica en los nombres de fichero lateral. Además, los ficheros laterales almacenan la siguiente información, que debería ser la misma para todos los ficheros laterales:

- la versión de fichero lateral.
- el número de nodos por "página de vector de bit" (explicado más adelante).

Hay también sumas de comprobación en cada fichero lateral las cuales identifican

- el conjunto de fichero lateral particular como un todo
- el mapa como un todo.

Éstos se deberían corregir para los ficheros laterales asociados para un mapa electrónico dado. Sí cualquiera de las comprobaciones anteriores falla, el rasgo de datos de aceleración de búsqueda no se activará para este mapa.

Estructuras de datos que se cargan en memoria

El decodificador lee los ficheros laterales en una implementación de memoria externa. Esto significa que los contenidos de los ficheros laterales de vector de bit no se cargan totalmente en memoria, sino que solamente se leen según se necesita. No obstante, algunos datos generales se cargan en memoria al principio y se mantienen allí durante el tiempo que se carga el mapa.

Información de cabecera

Cada fichero lateral comienza con una sección de cabecera, que contiene los datos descritos en relación con las Figuras 8 a 10 anteriores. Esta información se almacena en memoria para cada fichero lateral.

Definiciones de árboles de Huffman

Los ficheros laterales contienen las definiciones de varios árboles de Huffman. Cada definición de árbol de Huffman da la descripción completa de una codificación de Huffman particular y se usa más tarde para decodificar una parte del flujo de bits de fichero lateral en los datos originales (es decir para decodificar una secuencia de bits desde un fichero lateral en un número o algún otro valor particular). Las siguientes definiciones de árbol de Huffman se leen a partir de cada fichero lateral y se mantienen en memoria.

- Un árbol de Huffman para decodificar el número de segmentos de carretera en cada nodo. (El número codificado de segmentos de carretera puede ser menor que el mapa subyacente. Señalar que el

decodificador es independiente del mapa subyacente es decir no necesita leer el mapa.) Este árbol de Huffman solamente se almacena en el fichero lateral de nivel 0.

- Un árbol de Huffman para decodificar el código de página (explicado más adelante).
- Varios árboles de Huffman para decodificar los bloques de vector de bit. El número de definiciones de árbol se da por la longitud del bloque de vector de bit habitual como almacenada en la cabecera del fichero lateral; es igual a la longitud del bloque menos uno. (La cadena de bits entera de vectores de bit para un segmento de carretera se divide en bloques. Si la longitud de la cadena de bit de vector de bit no es un múltiplo de la longitud de bloque habitual, entonces el bloque final es más corto. Hay un árbol de Huffman para cada longitud de bloque desde 2 hasta la longitud de bloque habitual. No se usa un árbol de Huffman para un bloque de longitud uno, debido a que éste es sólo un bit y se almacena directamente.)
- Varios árboles de Huffman para seleccionar el método de decodificación para vectores de bit. El número de estos árboles se especifica la cabecera; su uso se explica más adelante.

Listas de contiguos

En cada nivel excepto el nivel de más detalle, el fichero lateral codifica las listas de contiguas. Una lista de contiguas para una región en un nivel k es una lista de regiones de nivel cero o más (k +1) que se llaman contiguas de la región de nivel k. La sección de lista de contiguas en el fichero lateral de nivel k tiene dos partes.

- La primera parte contiene el número de regiones contiguas para cada subregión de nivel k. Por ejemplo (k = 1), si hay 4000 regiones en el nivel 0 global, y cada región global se subdivide en 10 regiones de nivel 1, entonces el fichero lateral de nivel 1 contiene 4000*10 subregiones de nivel 1. Para cada una de éstas, se da la longitud de la lista de contiguas individual (la cual consta de regiones de nivel 2).
- La segunda parte contiene las listas de contiguas reales, cuyas longitudes respectivas se conocen a partir de la primera parte. Cada lista de contiguas es una lista de subregiones de nivel (k+1).

Tabla de reasignación de regiones

Los ficheros laterales almacenan vectores de bit en un formato comprimido por ejemplo mediante coalescencia y reordenación, etc. El fichero lateral en el nivel k tiene una tabla de reasignación de regiones que se usa para descomprimir vectores de bit codificados por bloques. Consta de dos partes:

- La primera parte es una colección que está indexada por las subregiones de nivel k. Para cada subregión en el nivel k, contiene la longitud del vector de bit comprimido. Esto es relevante para aquellos segmentos de carreteras salientes de nodos en la subregión respectiva para los cuales el vector de bit se codifica por bloques.
- La segunda parte es una colección bidimensional indexada por (1) la posición de bit en el vector de bit descomprimido y (2) la subregión de nivel k. Las entradas de la colección especifican la posición de bit en el vector de bit comprimido para la posición de bit dada en la cadena de bits no comprimida leída por bloques.

Señalar que solamente la primera parte se mantiene en memoria; la segunda parte se usa solamente bajo demanda durante la decodificación debido a que es grande. Actualmente, la tabla de reasignación de regiones se almacena solamente en el fichero lateral de nivel global (k = 0).

Iniciar una consulta de ruta

El decodificador se usa para acelerar la búsqueda de ruta desde una posición de salida a un conjunto de nodos de destino. (Un ejemplo en el que el destino consta de más de un nodo es cuando se usa una extensión de carretera entera como destino.) Los nodos de destino definen un conjunto de una o más regiones objetivo.

Señalar que una región objetivo es una secuencia de ID de región, una para cada nivel de división, comenzando en el nivel 0. Al comienzo de cada nueva búsqueda, el conjunto de regiones objetivo se construye borrando el conjunto previo de regiones objetivo y pasando los nuevos nodos de destino al decodificador. El decodificador determinará una lista de regiones objetivo sin duplicados. El mecanismo para encontrar la región de un nodo dado es el mismo que durante la búsqueda; ver más adelante para detalles.

Exploración de un nodo durante la búsqueda de ruta

La discusión que queda supone que se ha fijado un conjunto de regiones objetivo. Un rasgo del codificador es una función que, dado un ID de nodo de una red de carreteras, devuelve una colección de bits (los vectores de bit para este nodo y la regiones objetivo dadas) que está indexada sobre los segmentos de carretera que salen de este nodo. La ordenación de los nodos y segmentos de carretera en un nodo se define por el mapa. Siempre que un valor de bit es cero, esto significa que el segmento de carretera correspondiente se puede ignorar durante la búsqueda. (Esto normalmente conduce a una reducción considerable del espacio de búsqueda y de esta manera el tiempo de ejecución de la búsqueda.)

Para un número pequeño de nodos, ni la información de región ni los datos de vector de bit se almacenan en los ficheros laterales. Para estos nodos, el decodificador devuelve una cadena de bits en la que todos los bits son 1. (Esto impedirá que la búsqueda de ruta omita ningún segmento de carretera en este nodo.) El decodificador tiene una función de consulta Booleana que dice si éste es el caso para un nodo dado. Además, hay una función de consulta Booleana que dice si un nodo dado está situado en una de las regiones objetivo previamente fijadas. Los vectores de bit devueltos por el decodificador para un nodo en una región objetivo son de nuevo cadenas de bits en las que todos los bits son 1. Estas funciones de consulta Booleanas se usan para optimizaciones en la búsqueda de ruta.

Según el formato de fichero lateral, los datos del vector de bit para un nodo dado se decodifican en varios pasos. En cada fichero lateral, la información vector de bit y región se organiza en de las denominadas páginas. El número de nodos por página es una potencia fija de 2 (por ejemplo 16) e idéntica para cada fichero lateral de un conjunto de ficheros laterales. Esto significa que, para un ID de nodo dado, el índice de página de una página de vector de bit se puede calcular por un desplazamiento de bit simple. La información para los nodos se almacena consecutivamente por ID de nodo.

Encontrar el desplazamiento de página para un nodo dado

Para cada fichero lateral, el número medio de bytes por página se almacena en la cabecera del fichero lateral. Se usa para aproximar el desplazamiento de bytes de la página multiplicando el índice de página con el tamaño medio. Un término de corrección se almacena en una tabla indexada por el índice de página. Esta tabla se almacena en una sección separada del fichero lateral. Cuando se consulta una página, el término de corrección se busca en la tabla y añade al desplazamiento de página aproximado, dando la posición de la página en el fichero lateral.

Decodificación de una página

Almacenamiento en caché

Cuando se consulta una página por primera vez, las cadenas de vectores de bit para los nodos en la página se decodifican (con respecto al conjunto de regiones objetivo fijas) y se almacenan en caché. Los datos de la siguiente vez se consultan para un nodo desde la misma página, los datos almacenados en caché se pueden usar sin ningún acceso al fichero lateral. Un bit marcador especial en la cadena de bits de vector de bit decodificada se usa para recordar si el nodo es un nodo sin información.

Código de página

Para cada página, un denominado bit de código de página especifica si todos los nodos en la página tienen el mismo ID de región. El código de página contiene un bit por nivel, pero todos los bits se almacenan como un símbolo de Huffman común al principio de la página en el fichero lateral de nivel 0 solamente.

Decodificación de recuentos de segmentos de carretera salientes

Como se mencionó anteriormente, cada página contiene información para un número fijo de nodos. Este número de nodos se almacena en la cabecera de cada fichero lateral. Al principio de una página (o después del código de página, para el nivel 0), la página enumera el número de segmentos de carretera salientes para cada nodo en la página. Siempre que el número de segmentos de carretera salientes es 0, esto significa que no se almacena ninguna información en absoluto para el nodo correspondiente. Mientras que se decodifica la página, los números se almacenan en una colección temporal.

Decodificación de regiones

La sección de recuento de segmentos de carretera es seguida por la sección de regiones. La región de un nodo se da por una secuencia de ID de región, uno en cada nivel. El ID de región de un nivel particular se almacena en el fichero lateral correspondiente. El decodificador lee las regiones en todos los niveles para todos los nodos en la página. Cuando no se almacena ninguna información para un nodo (es decir, el recuento de segmentos de carretera del nodo es cero), la información de región se deja vacía. El decodificador lee la secuencia de ID de región para el primer nodo que tiene un recuento de segmentos de carretera mayor que cero. Si el código de página en un nivel dado especifica que todos los ID de región son los mismos en ese nivel, entonces en ese nivel se fija el mismo ID de región para todos los nodos. De otro modo, los ID de región en los niveles correspondientes se leen para todos los nodos de recuento de segmentos de carretera positivo. Al final de este proceso, el decodificador ha llenado una tabla con las secuencias completas de ID de región para todos los nodos de la página (donde algunas secuencias pueden estar vacías).

Decodificación de vectores de bit

Encontrar el conjunto de posiciones de bit relevantes

Para una región objetivo y nodo dado, un bit particular en un nivel particular determina el valor de los bits de vector de bit para los segmentos de carretera que salen de este nodo. Cuando hay más de una región objetivo, a los bits resultantes se les aplica una operación lógica OR juntos. Para cada nodo, el decodificador calcula el conjunto de posiciones de bit relevantes. El conjunto de posiciones de bit relevantes es el mismo para cada segmento de carretera saliente en ese nodo. Solamente depende de la región del nodo y el conjunto de regiones objetivo. Si hay sólo una región objetivo, habrá solamente una posición de bit relevante en un nivel; en otras palabras, la información almacenada en los otros niveles se puede ignorar para este nodo. En el caso de más de una región objetivo, algunas de las posiciones de bit relevantes pueden coincidir, de manera que siempre hay a lo sumo tantas posiciones de bit relevantes como regiones objetivo haya. A continuación, trataremos cómo determina el decodificador la posición de bit relevante para una región objetivo dada. Para más de una región objetivo, las posiciones de bit relevantes se encuentran de la misma forma y se combinan en un conjunto.

Cuando no se definen regiones contiguas, hay un bit de vector de bit (por segmento de carretera saliente) para cada región objetivo en cada nivel. (Por simplicidad, nos olvidamos del hecho de que no está almacenado ningún bit para el ID de región del nodo en sí mismo). El bit relevante está en el primer nivel, contando desde el nivel 0, en el que el ID de región del nodo difiere del ID de región de la región objetivo. Por ejemplo, si el ID de región del nodo en el nivel global es igual al ID de región objetivo en el global, pero los dos ID de región son diferentes en el nivel 1, entonces la posición de bit relevante está en el nivel 1, y es igual al ID de región objetivo. Es una posición de bit en la cadena de vectores de bit descomprimidos en ese nivel; esta cadena contiene un bit para cada ID de región objetivo posible. La tabla de reasignación de región se usa para transformar esta posición en una posición en la cadena de vectores de bit comprimida, la cadena que se codifica realmente en ese nivel.

Cuando se definen regiones contiguas, entonces se determina el bit relevante en el nivel “de más detalle” en el que la región objetivo es una contigua de la región del nodo. Tomando cuatro niveles como ejemplo, permitamos que la secuencia de ID de región objetivo sea (a, b, c, d), y permitamos que la secuencia de ID de región del nodo sea (e, f, g, h). Si (a, b, c, d) es una contigua de (e, f, g) como se define en la sección de listas contiguas, y (a, b) es una contigua de (e), entonces el bit relevante se determina por (a, b, c, d) y se sitúa en el nivel 2, el nivel del prefijo de (e, f, g) que contiene (a, b, c) como contigua. Más concretamente, la posición de bit relevante es el índice de (a, b, c, d) como una contigua de la región (e, f, g) en el fichero lateral de nivel 2. El bit habitual para el ID de región h, como se explicó en el párrafo previo, es irrelevante en este caso. De nuevo, esta posición de bit relevante es con respecto a la cadena de vectores de bit descomprimida. El decodificador usa la tabla de reasignación de regiones para transformarla a una posición de bit en la cadena de bits comprimida en el nivel 2. Más información sobre las listas contiguas está contenida en el documento “Proposal for enhanced multi-level graph partitioning”.

Decodificación de los bits de vector de bit

Dado el conjunto fijo de regiones objetivo, el vector de bit para cada nodo en la página constará de un bit por segmento de carretera saliente. Si el recuento de segmentos de carretera del nodo es cero (es decir el nodo es un nodo sin información), cada bit se fijará a 1 sin decodificación adicional para ese nodo. Si el nodo se sitúa en una de las regiones objetivo, el vector de bit será de nuevo todo 1; en este caso, los datos codificados podrían tener que ser omitidos a fin de decodificar datos para los nodos posteriores.

Si el vector de bit para el nodo actual no es trivialmente todo 1, el decodificador determina el conjunto de posiciones de bit relevantes, como se explicó anteriormente. Cada posición de bit es con respecto a un nivel particular, es decir el bit está situado en un fichero lateral particular. En cada nivel, el fichero lateral contiene la cadena completa de bits

comprimidos, pero el decodificador necesita extraer los bits en las posiciones de bit relevantes solamente. Durante la decodificación, la información no usada se lee (se omite), pero solamente si es seguida por información que se necesita realmente. De otro modo, la parte no usada del flujo de bit no se lee en absoluto.

El conjunto de posiciones de bits relevantes se agrupa según el nivel. En lugar de decodificar la cadena de bits comprimida y descomprimirla para leer los bits en las posiciones relevantes, las posiciones relevantes se transforman en posiciones en la cadena de bits comprimida. Si hay bits relevantes en algún nivel, primero se omiten los datos para nodos precedentes (si es adecuado el decodificador recuerda hasta donde se ha llegado en el fichero lateral en cada nivel). Cuando la sección para el nodo en cuestión se alcanza en un fichero lateral dado, los vectores de bit se decodifican línea a línea. Para cada segmento de carretera saliente, se almacena un bit; es el resultado de una operación lógica OR del bit decodificado para todas las posiciones de bit relevantes. La información para un segmento de carretera particular en el nodo actual comienza con un símbolo de Huffman que especifica el método de decodificación. Tiene uno de los siguientes valores:

- Un símbolo que especifica que todos los bits de vector de bit para el segmento de carretera en este nivel son 0 (incluyendo todos los bits contiguos a este nivel). No tienen que ser codificados bits adicionales para este segmento de carretera.
- Un símbolo que especifica que todos los bits de vector de bit para el segmento de carretera en este nivel son 1 (incluyendo todos los bits contiguos en este nivel). No tienen que ser codificados bits adicionales para este segmento de carretera.
- Un símbolo que especifica que la cadena de bits de vector de bit se da explícitamente por bloques. La decodificación de la cadena de bits de vector de bit comprimida por bloques se explica más adelante. El bit de vector de bit para el segmento de carretera se pone en una “pila de historia” que se construye durante la codificación de la página entera.
- Un símbolo que especifica un índice en la pila de historia. En el índice dado, la pila de historia contiene el valor de bit de vector de bit deseado para el segmento de carretera.

Se usa un árbol de Huffman diferente para el selector del método de decodificación, dependiendo del tamaño actual de la pila de historia. Hay un valor límite en la cabecera del fichero lateral que especifica el número de árboles de Huffman; cuando el tamaño de la pila de historia excede este valor, entonces se reutiliza el último árbol de Huffman.

Si el símbolo de selector del método especifica que la cadena de vectores de bit se debería codificar explícitamente, entonces el decodificador lee la cadena de bits de vector de bit comprimida bloque por bloque. Reúne los valores de bit en las posiciones de bit relevantes (con respecto a la cadena de vectores de bit comprimida) en este nivel. A los valores de los bits se les aplica una operación lógica OR. Tan pronto como el resultado intermedio de la OR llega a ser 1, se puede omitir el resto de los bits en todos los bloques adicionales para este segmento de carretera. El árbol de Huffman usado para decodificar cada bloque depende del número de bits en el bloque. Hay un árbol de Huffman para la longitud de bloque habitual como se especifica en la cabecera del fichero lateral. El último bloque para el segmento de carretera puede ser más corto; dependiendo de su tamaño, se usa un árbol de Huffman adecuado para decodificación. Un bloque de longitud uno es sólo un bit, que se lee directamente sin decodificación de Huffman.

Si el número de bloques es al menos 2, entonces el flujo de bit generado a partir del vector de bit contiene un bit adicional que llega antes que los símbolos de Huffman para los bloques. Su valor especifica si la cadena decodificada entera se entiende que es la negación en forma de bit del valor real. (El propósito de esta negación es una mejor compresión de Huffman).

REIVINDICACIONES

1. Un método de determinación de perfiles de coste de rutas que usa datos de mapa divididos en una pluralidad de regiones (600; 602, 604, 606, 608), los datos de mapa que comprenden:

una pluralidad de segmentos navegables (304a, 304b, 304c, 304d) que representan segmentos de trayectos navegables de los datos de mapa, cada segmento navegable que tiene una función de coste que varía con la hora asociada; y

datos de coste mínimo (704) para los segmentos navegables dentro de cada región de los datos de mapa que identifican, para cada una de las otras regiones, si un segmento navegable es parte de un trayecto de coste mínimo para la otra región en cualquier periodo de tiempo,

el método que comprende usar al menos un aparato de procesamiento para:

buscar rutas desde un origen a un destino, la búsqueda que comprende determinar si uno o más segmentos navegables de un conjunto de segmentos navegables conectados a un nodo se identifican por los datos de coste mínimo como parte de un trayecto de coste mínimo para regiones que comprenden el origen y destino y, si uno o más de los segmentos navegables del conjunto se identifican como que son parte de un trayecto de coste mínimo, explorar desde el conjunto solamente los uno o más segmentos navegables que se identifican como que son parte de un trayecto de coste mínimo;

caracterizado por que el método comprende usar el al menos un aparato de procesamiento para:

determinar un perfil de coste de las rutas en el tiempo a partir de las funciones de coste que varían con la hora de los segmentos navegables que se exploran,

en donde determinar un perfil de coste de las rutas en el tiempo comprende:

combinar un primer perfil de coste (C1) asociado con un primer trayecto de coste mínimo y un segundo perfil de coste (C2) asociado con un segundo trayecto de coste mínimo cuando el primer y segundo trayectos de coste mínimo se cruzan en un nodo.

2. Un método según la reivindicación 1, en donde el paso de combinar el primer (C1) y segundo (C2) perfiles de coste comprende superponer los dos perfiles de coste y determinar al menos uno de un perfil de coste máximo (UB) y mínimo (LB) para propagación adicional.

3. Un método según la reivindicación 1, en donde el paso de combinar el primer (C1) y segundo (C2) perfiles de coste comprende determinar un perfil de coste medio para propagación adicional.

4. Un método según cualquier reivindicación precedente, en donde la función de coste que varía con la hora asociada con un segmento navegable comprende datos de perfil de velocidad que identifican la velocidad esperada en el segmento navegable a diferentes horas.

5. Un método según cualquier reivindicación precedente, en donde el perfil de coste de las rutas con el tiempo se determina en respuesta a recibir una petición de perfil de coste, la petición de perfil de coste que incluye un periodo de tiempo dentro del cual está confinada la búsqueda de rutas.

6. Un método según cualquiera de las reivindicaciones 1 a 4, que comprende recibir una petición de una ruta entre el origen y el destino a una hora de viaje particular, el perfil de coste de las rutas con el tiempo que se determina para una ventana de tiempo que incluye esa hora de viaje particular, el método que además comprende:

determinar a partir del perfil de coste si a otra hora dentro de esa ventana de tiempo el coste de la ruta es menor que el coste para la hora particular y, si el coste es menor, causar la visualización de un mensaje (2015) que informa al usuario de la otra hora de viaje.

7. Un método según cualquiera de las reivindicaciones 1 a 4, que comprende recibir desde un usuario a través de una interfaz de usuario una selección de una hora de viaje y causar una visualización para mostrar una imagen de la ruta determinada a partir del perfil de coste a la hora de viaje recibida.

8. Un método según la reivindicación 7, que comprende permitir al usuario cambiar la hora de viaje mientras que ve la imagen de la ruta, y, en respuesta a un cambio en la hora de viaje, actualizar la visualización con una imagen de una nueva ruta determinada a partir del perfil de coste para la hora de viaje cambiada.

9. Un método según la reivindicación 8, que comprende causar la visualización de un control deslizante que representa la hora de viaje y actualizar la visualización con la imagen de la nueva ruta en respuesta a la interacción del usuario con el control deslizante.

10. Una portadora de datos que tiene almacenadas sobre la misma instrucciones que, cuando se ejecutan por un

aparato de procesamiento, causan al aparato de procesamiento ejecutar un método según cualquier reivindicación precedente.

11. Un dispositivo informático que comprende una memoria que tiene almacenados en la misma datos de mapa divididos en una pluralidad de regiones (600; 602, 604, 606, 608), los datos de mapa que comprenden:

una pluralidad de segmentos navegables (304a, 304b, 304c, 304d) que representan segmentos de trayectos navegables de los datos de mapa, cada segmento navegable que tiene una función de coste que varía con la hora asociada; y

datos de coste mínimo (704) para los segmentos navegables dentro de cada región de los datos de mapa que identifican, para cada una de las otras regiones, si un segmento navegable es parte de un trayecto de coste mínimo para la otra región en cualquier periodo de tiempo,

el aparato de procesamiento dispuesto para:

buscar rutas desde un origen a un destino, la búsqueda que comprende determinar si uno o más segmentos navegables de un conjunto de segmentos navegables conectados a un nodo se identifican por los datos de coste mínimo como parte de un trayecto de coste mínimo para regiones que comprenden el origen y destino y, si uno o más de los segmentos navegables del conjunto se identifican como que son parte de un trayecto de coste mínimo, explorar desde el conjunto solamente el uno o más segmentos navegables que se identifican como que son parte de un trayecto de coste mínimo;

caracterizado por que el aparato de procesamiento está dispuesto para:

determinar un perfil de coste de las rutas en el tiempo a partir de las funciones de coste que varían con la hora de los segmentos navegables que se exploran,

en donde determinar un perfil de coste de las rutas en el tiempo comprende:

combinar un primer perfil de coste (C1) asociado con un primer trayecto de coste mínimo y un segundo perfil de coste (C2) asociado con un segundo trayecto de coste mínimo cuando el primer y segundo trayectos de coste se cruzan en un nodo.

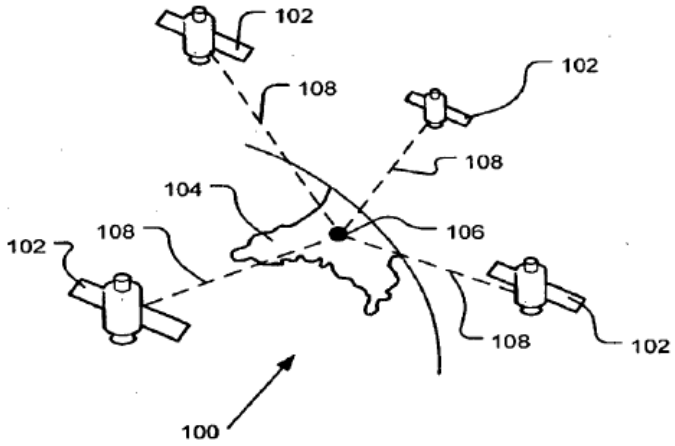


Figura 1

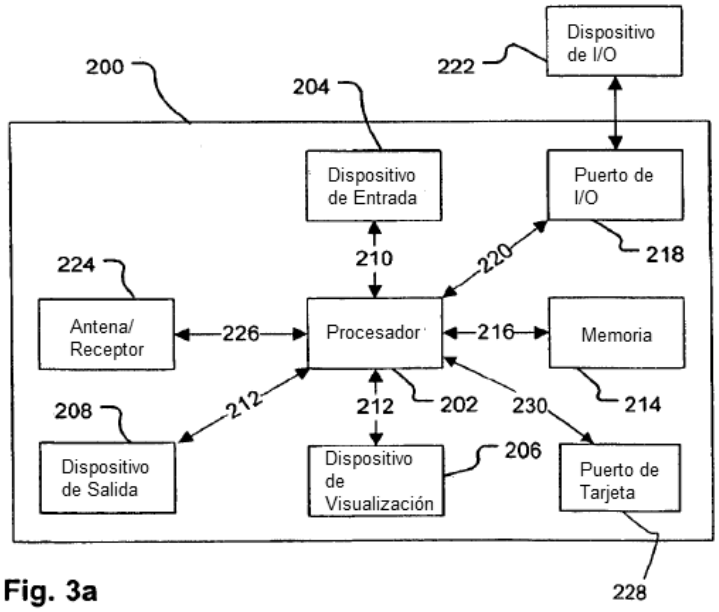
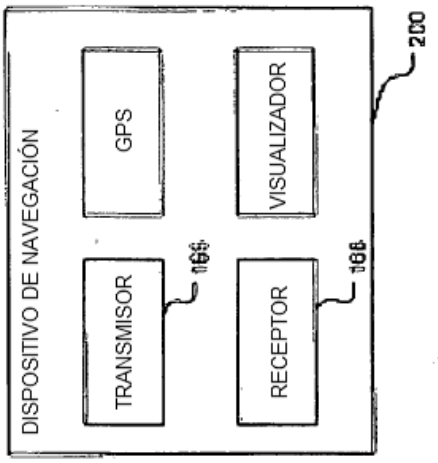
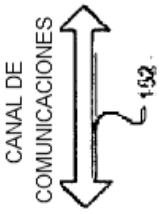


Fig. 3a


DISPOSITIVO DE NAVEGACIÓN
GPS
VISUALIZADOR
TRANSMISOR
RECEPTOR
200


CANAL DE
COMUNICACIONES
152

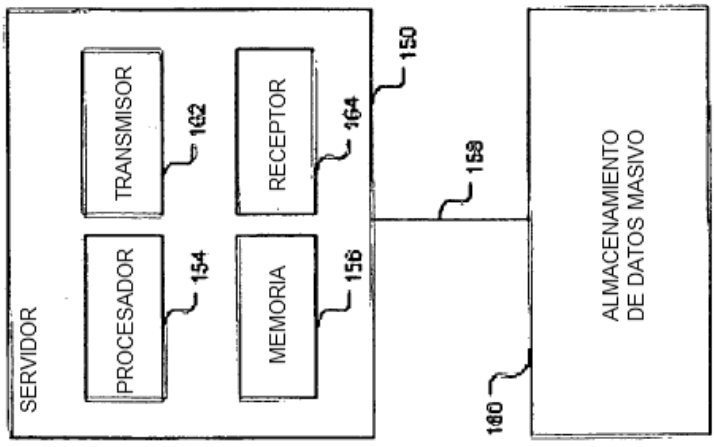

SERVIDOR
PROCESADOR
154
TRANSMISOR
162
MEMORIA
156
RECEPTOR
164
150
158
160
ALMACENAMIENTO
DE DATOS MASIVO

Figura 2

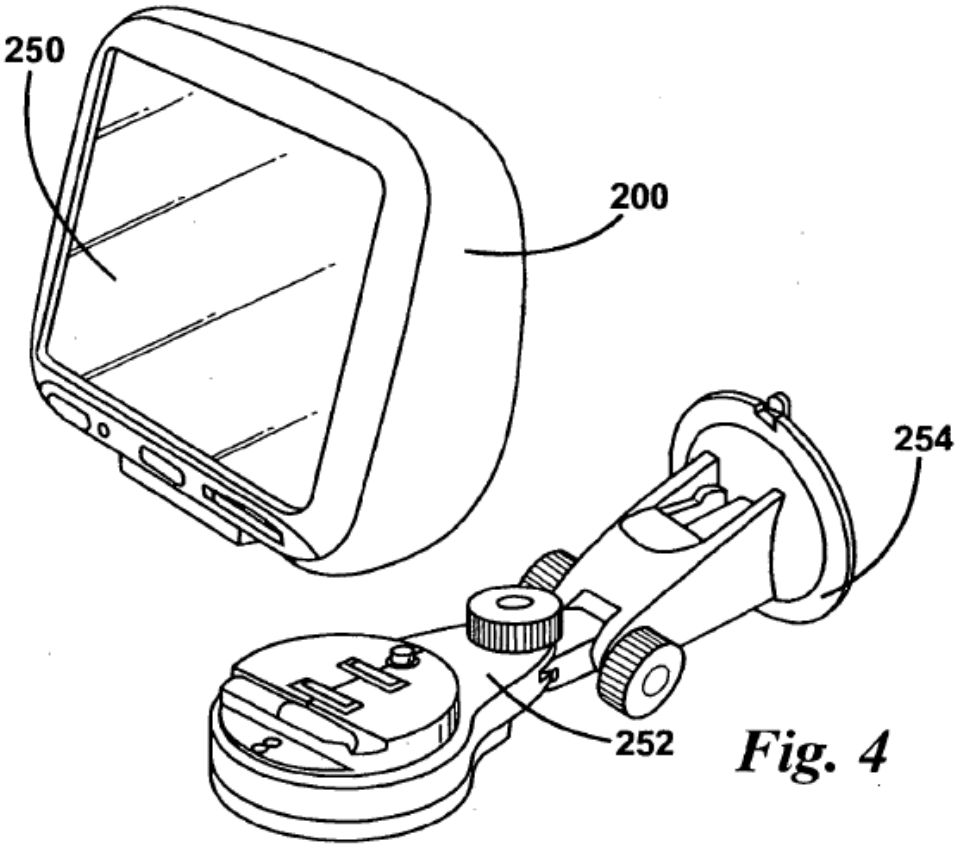

250
200
254
252

Fig. 4

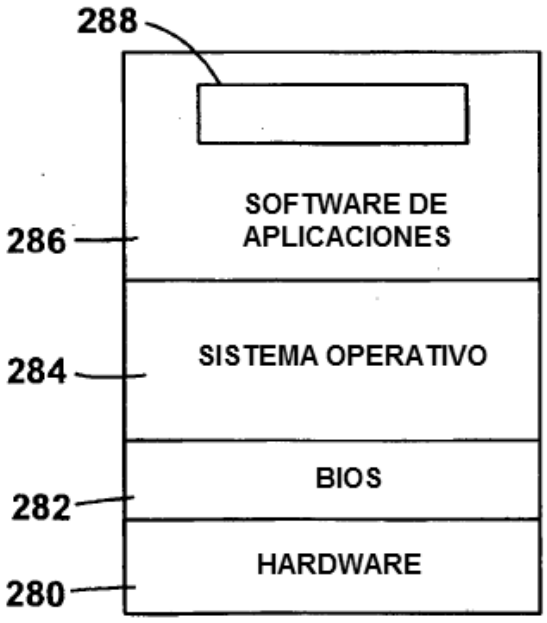

288
SOFTWARE DE APLICACIONES
286
SISTEMA OPERATIVO
284
BIOS
282
HARDWARE
280

Fig. 3b

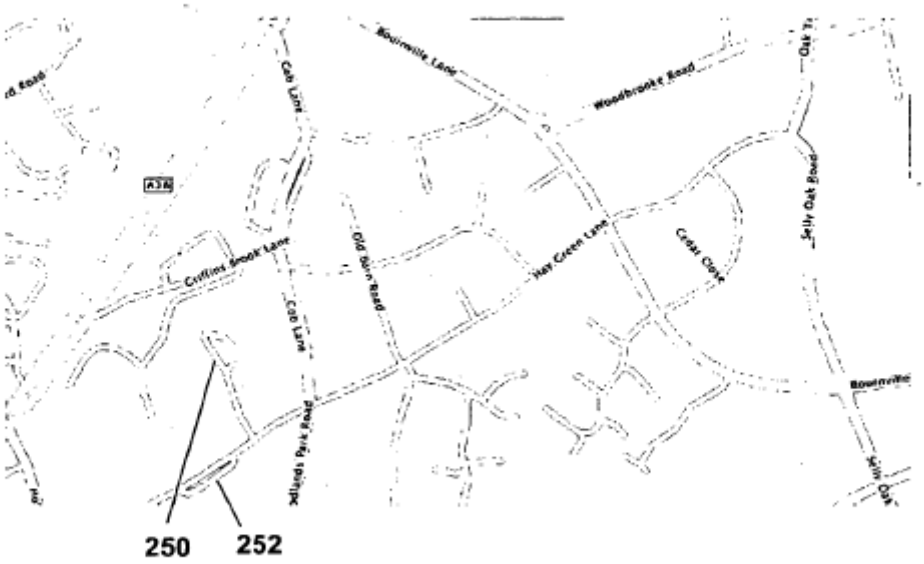

Woodbrooke Road
Griffins Brook Lane
Old Barn Road
Hay Green Lane
Cedar Copse
Selly Oak Road
250
252

Fig. 5

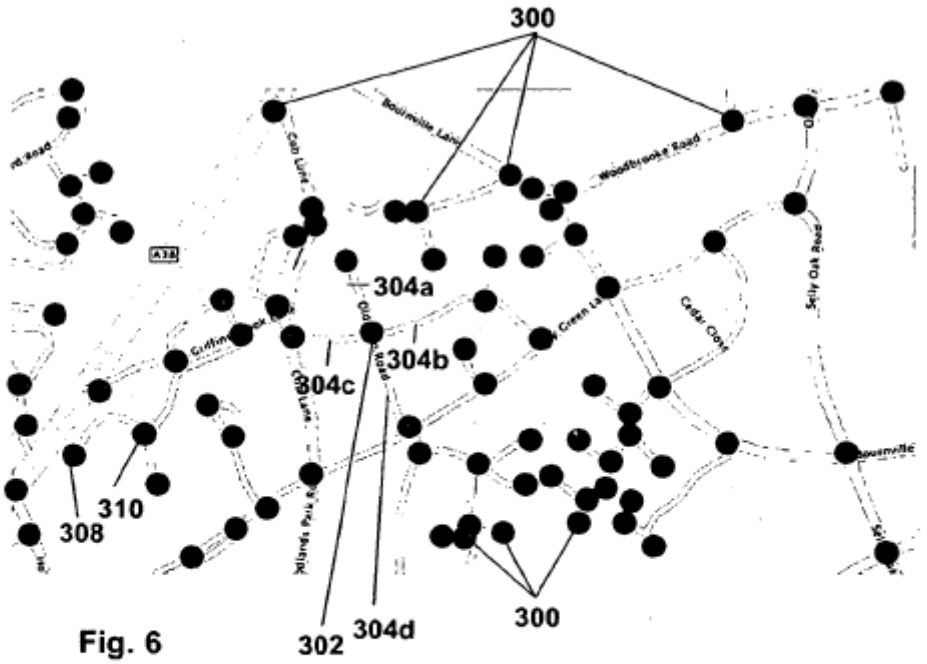

300
Woodbrooke Road
304a
304b
304c
310
308
302
304d
300

Fig. 6

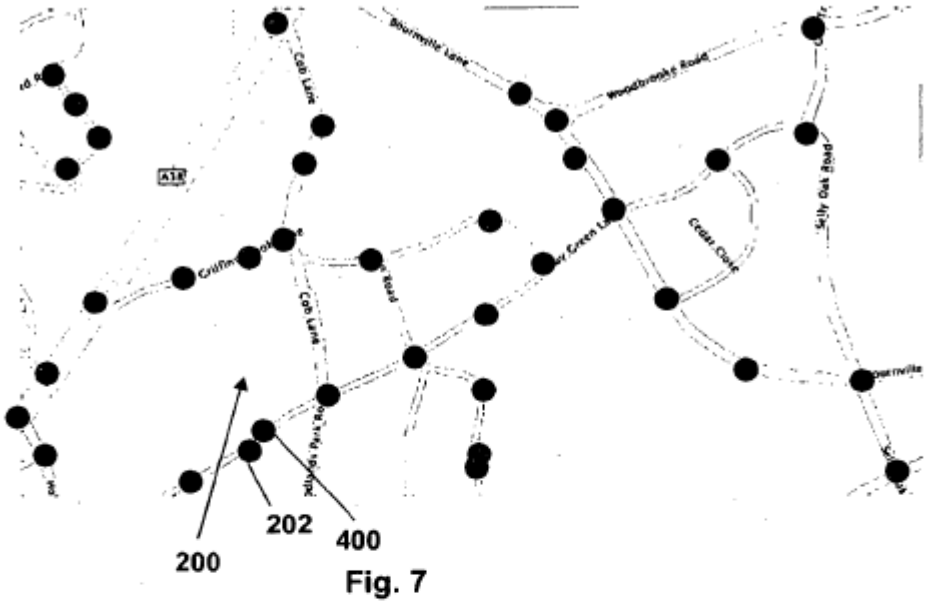


Fig. 7

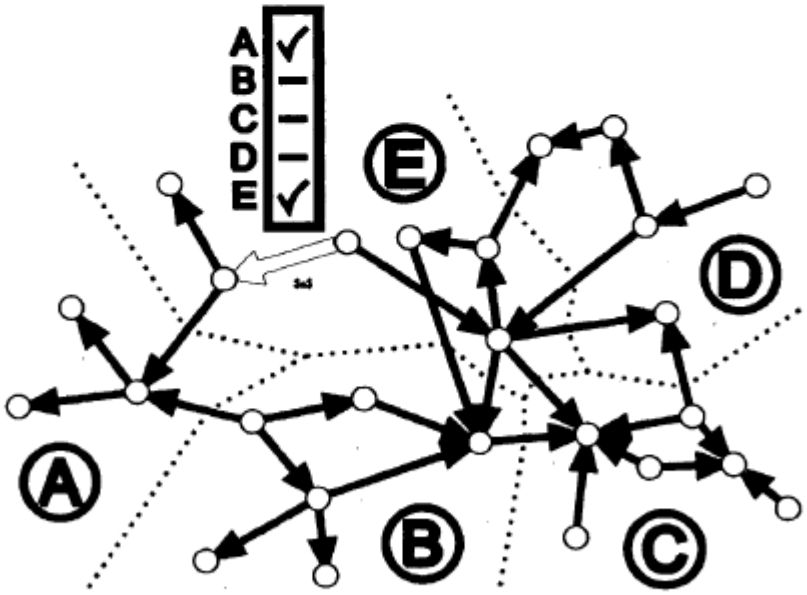


Fig. 8

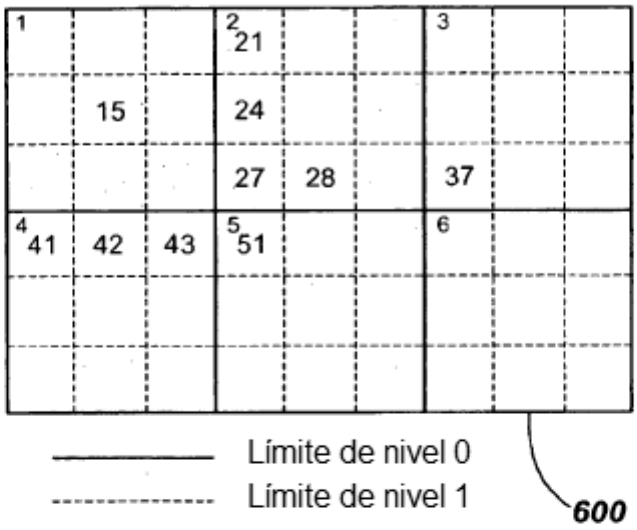

1
2
21
3
15
24
27
28
37
4
41
42
43
5
51
6
Límite de nivel 0
Límite de nivel 1
600

Fig. 9

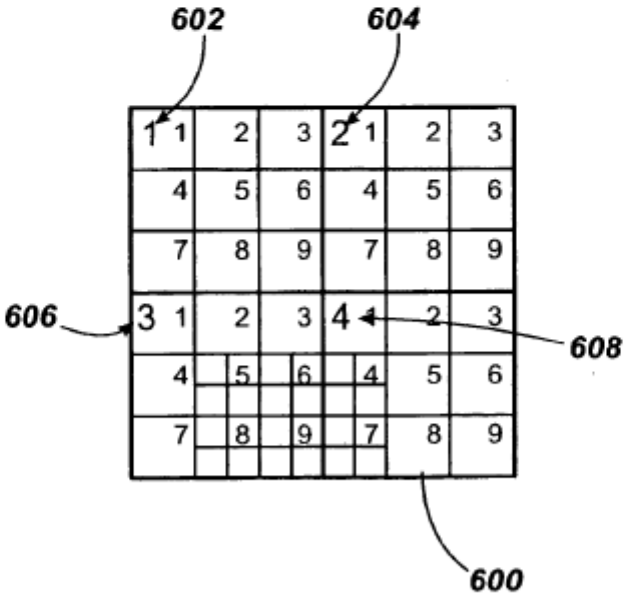

602
604
606
608
600

Fig. 9a

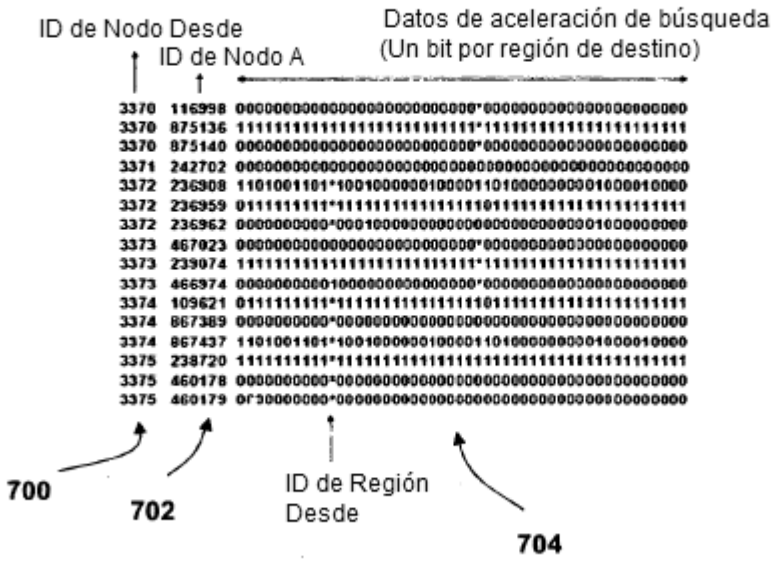


Fig. 10

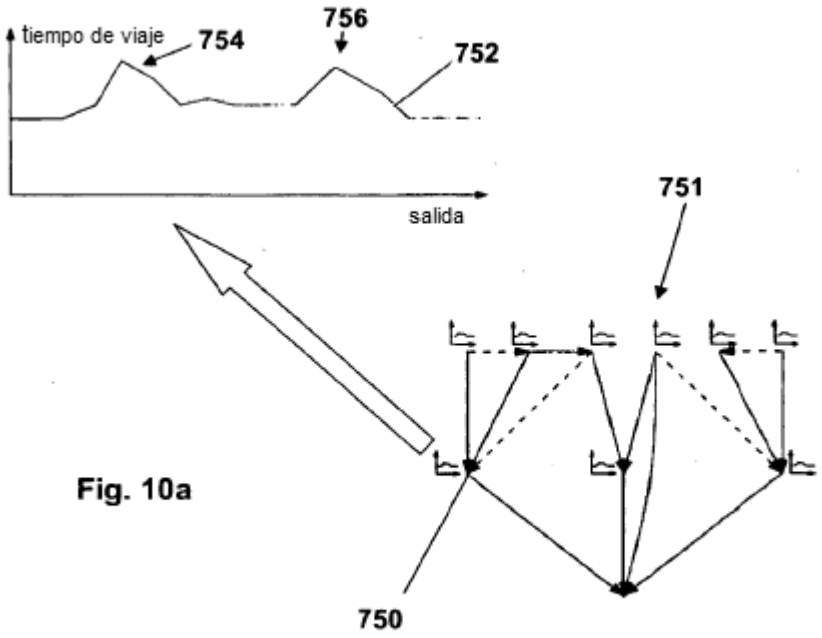


Fig. 10a

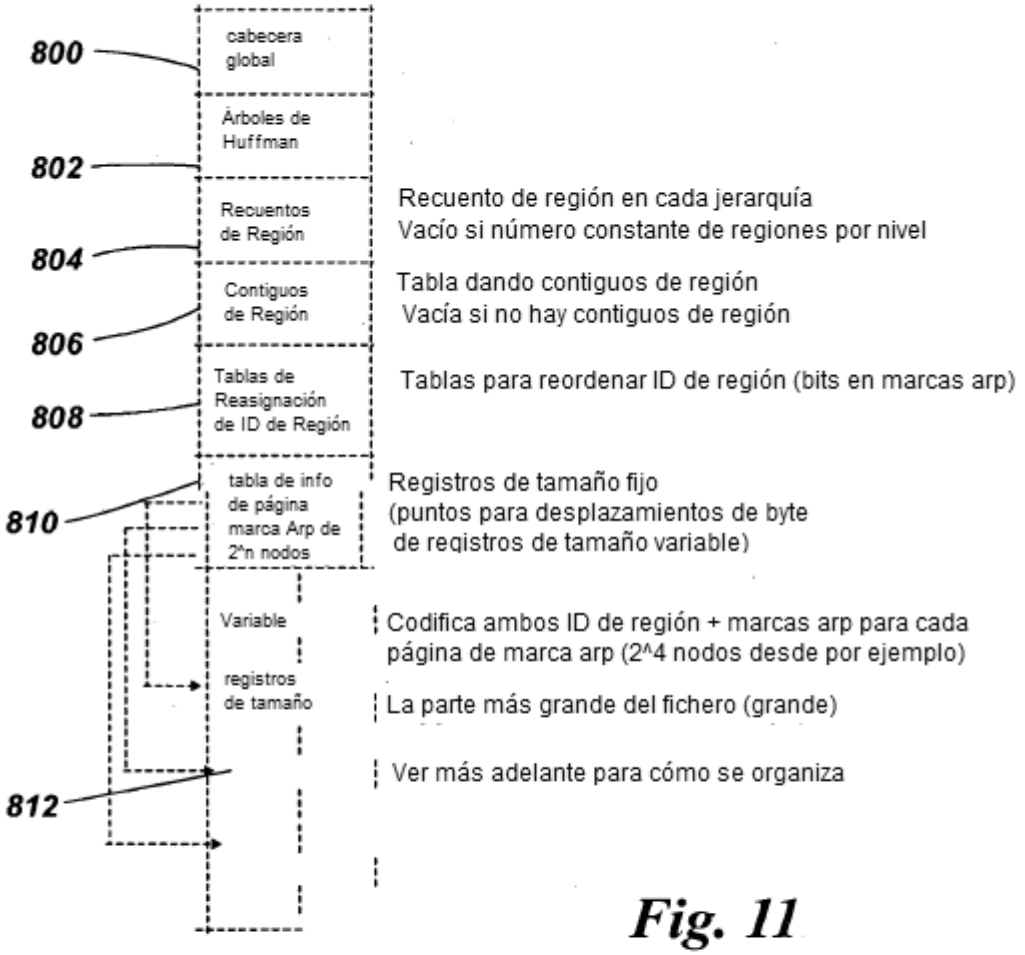

800
cabecera global
Árboles de Huffman
802
Recuentos de Región
804
Contiguos de Región
806
Tablas de Reasignación de ID de Región
808
tabla de info de página marca Arp de 2^n nodos
810
Variable
registros de tamaño
812
Recuento de región en cada jerarquía
Vacío si número constante de regiones por nivel
Tabla dando contiguos de región
Vacía si no hay contiguos de región
Tablas para reordenar ID de región (bits en marcas arp)
Registros de tamaño fijo
(puntos para desplazamientos de byte
de registros de tamaño variable)
Codifica ambos ID de región + marcas arp para cada página de marca arp (2^4 nodos desde por ejemplo)
La parte más grande del fichero (grande)
Ver más adelante para cómo se organiza

Fig. 11

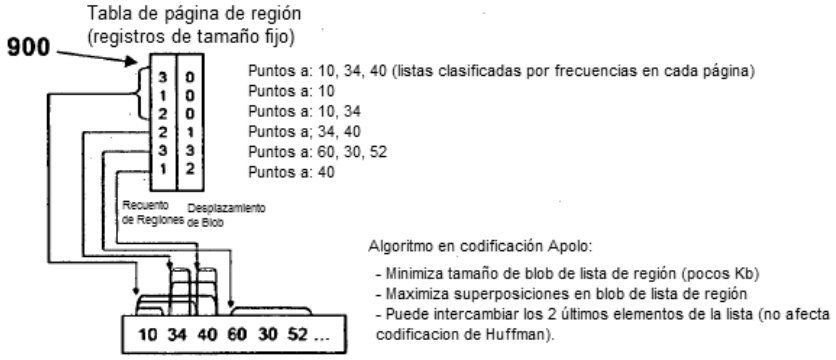


Tabla de página de región
(registros de tamaño fijo)
900
3 0
1 0
2 0
2 1
3 3
1 2
Puntos a: 10, 34, 40 (listas clasificadas por frecuencias en cada página)
Puntos a: 10
Puntos a: 10, 34
Puntos a; 34, 40
Puntos a: 60, 30, 52
Puntos a: 40
Recuento de Regiones
Desplazamiento de Blob
10 34 40 60 30 52 ...
Lista de región de solapamiento (blob)
Algoritmo en codificación Apolo:
- Minimiza tamaño de blob de lista de región (pocos Kb)
- Maximiza superposiciones en blob de lista de región
- Puede intercambiar los 2 últimos elementos de la lista (no afecta codificacion de Huffman).

Fig. 12

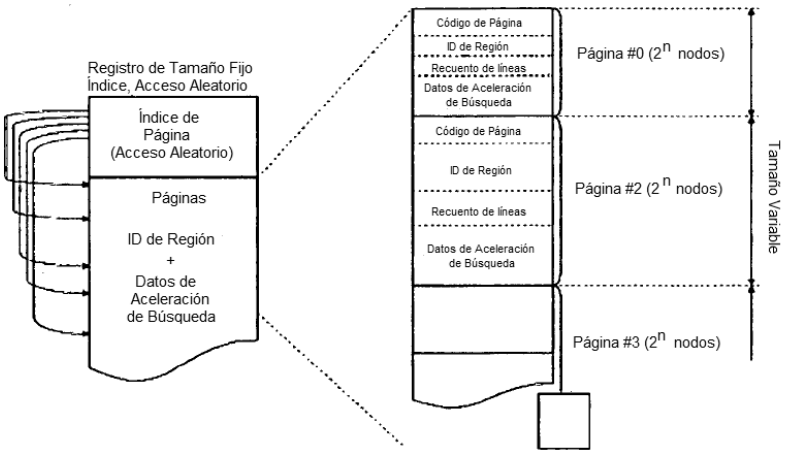

Registro de Tamaño Fijo
Índice, Acceso Aleatorio
Índice de Página (Acceso Aleatorio)
Páginas
ID de Región + Datos de Aceleración de Búsqueda
Código de Página
ID de Región
Recuento de líneas
Datos de Aceleración de Búsqueda
Página #0 (2n nodos)
Página #2 (2n nodos)
Página #3 (2n nodos)
Tamaño Variable

Fig. 13

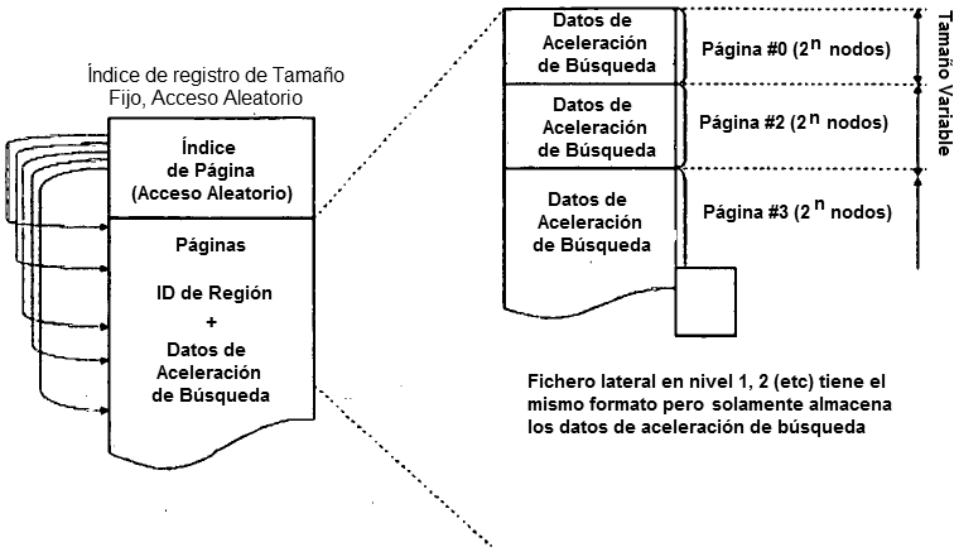


Fig. 14

Datos de tamaño variable después de incorporación:

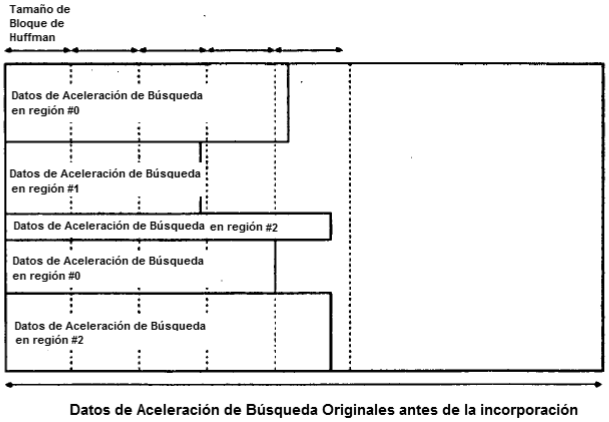


Fig. 15

8 regiones, agrupando para usar marcas de vector de bit de 4 bits

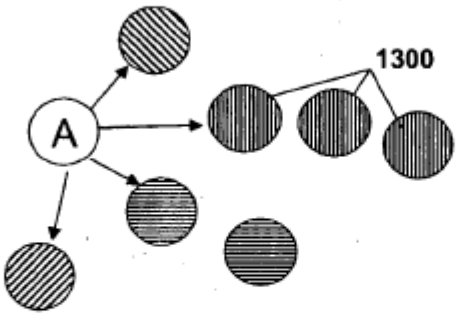


Fig. 16a

Agrupación desde el punto de vista de la región A

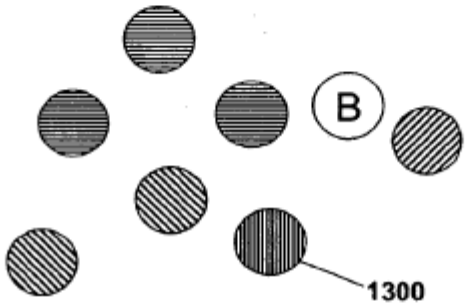


Fig. 16b

Agrupación desde el punto de vista de la región B

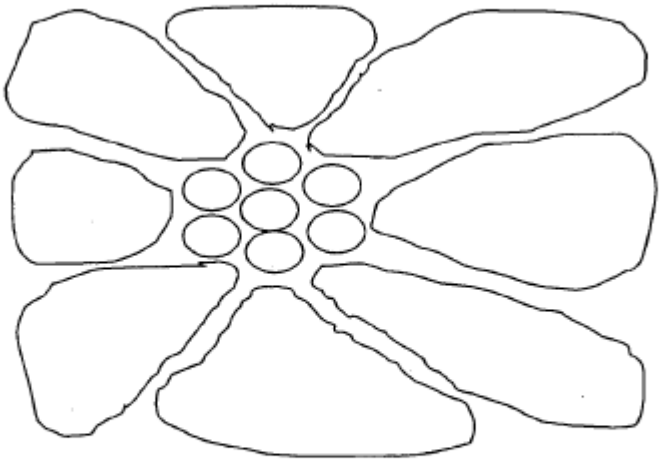


Fig. 17

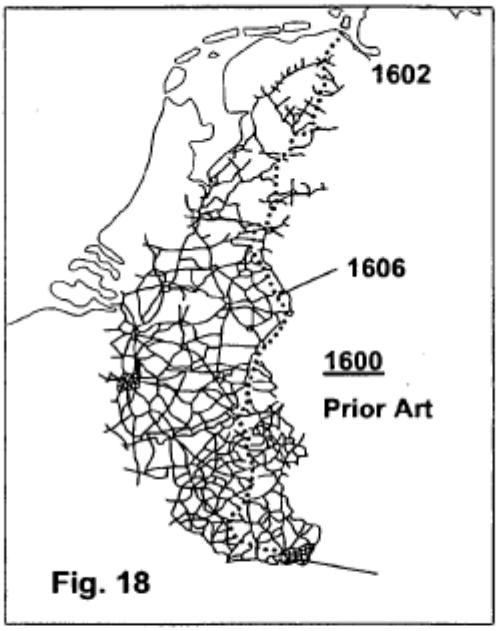


Fig. 18

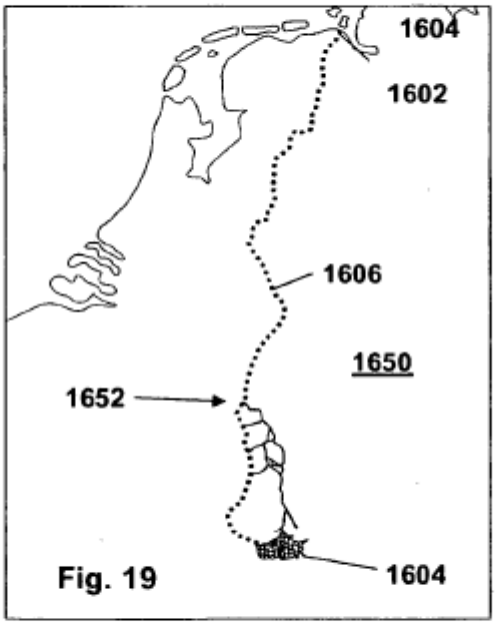


Fig. 19

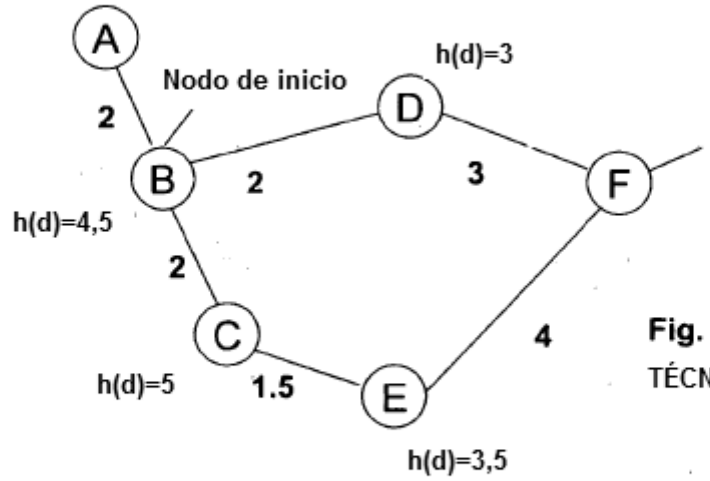


Fig. 20
TÉCNICA ANTERIOR

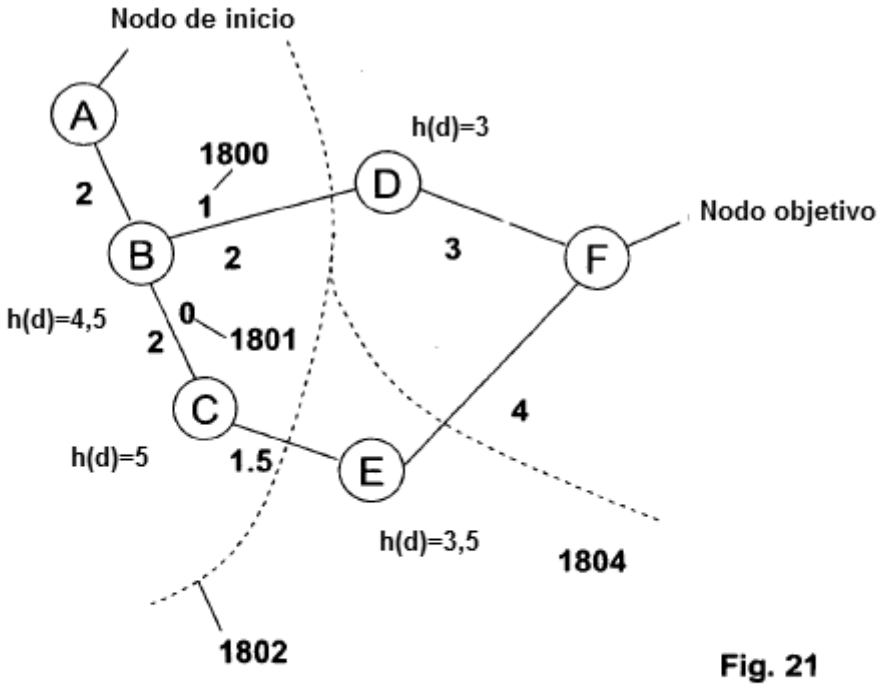


Fig. 21

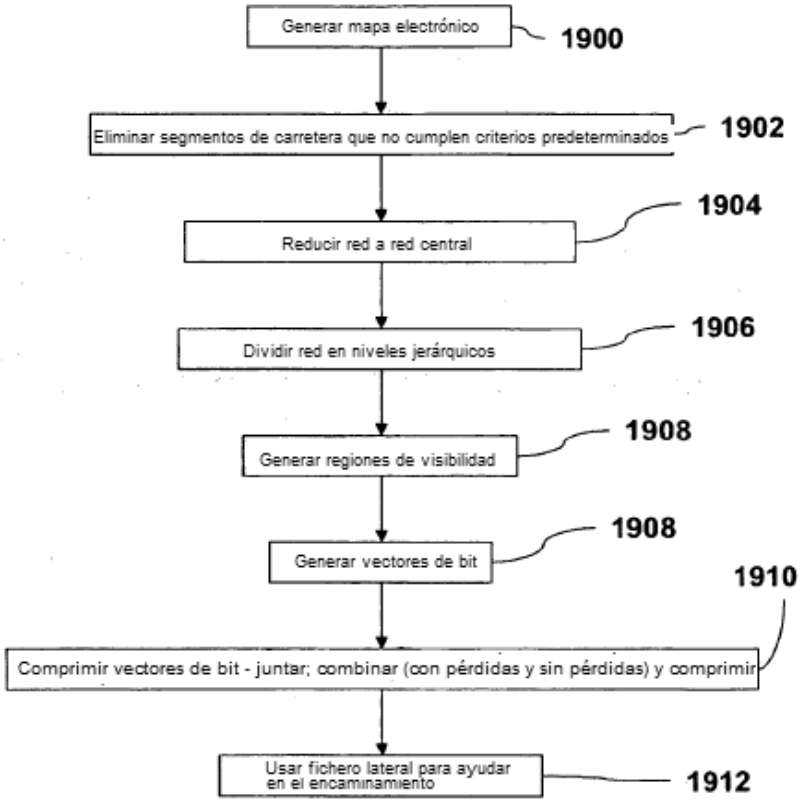

Generar mapa electrónico
1900
Eliminar segmentos de carretera que no cumplen criterios predeterminados
1902
Reducir red a red central
1904
Dividir red en niveles jerárquicos
1906
Generar regiones de visibilidad
1908
Generar vectores de bit
1908
1910
Comprimir vectores de bit - juntar; combinar (con pérdidas y sin pérdidas) y comprimir
Usar fichero lateral para ayudar en el encaminamiento
1912

Fig. 22

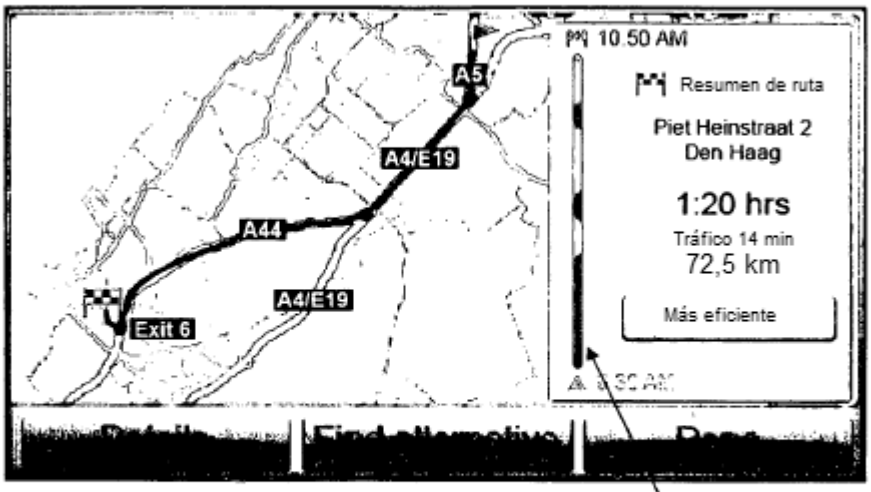

10.50 AM
A5
A4/E19
A44
A4/E19
Exit 6
Resumen de ruta
Piet Heinstraat 2
Den Haag
1:20 hrs
Tráfico 14 min
72,5 km
Más eficiente
2000

Fig. 23

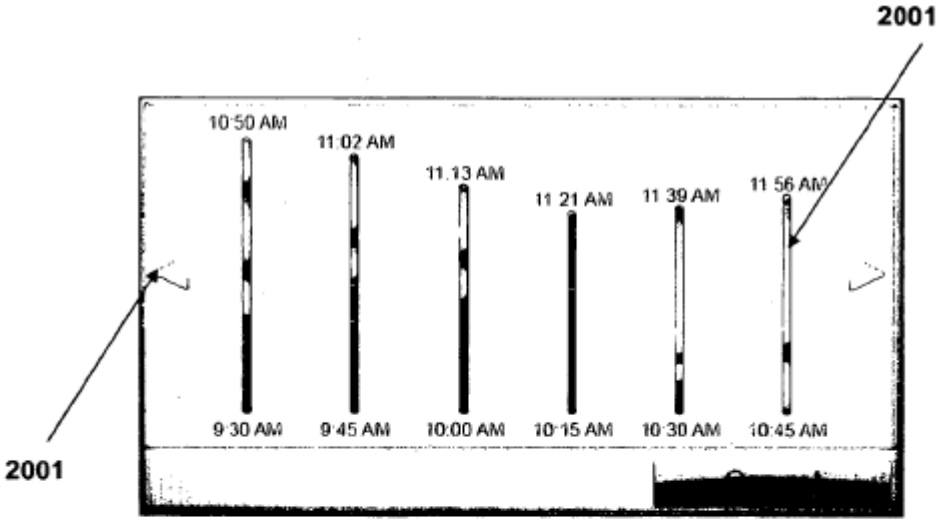

2001
2001
10:50 AM
11:02 AM
11:13 AM
11:21 AM
11:39 AM
11:56 AM
9:30 AM
9:45 AM
10:00 AM
10:15 AM
10:30 AM
10:45 AM

Fig. 24

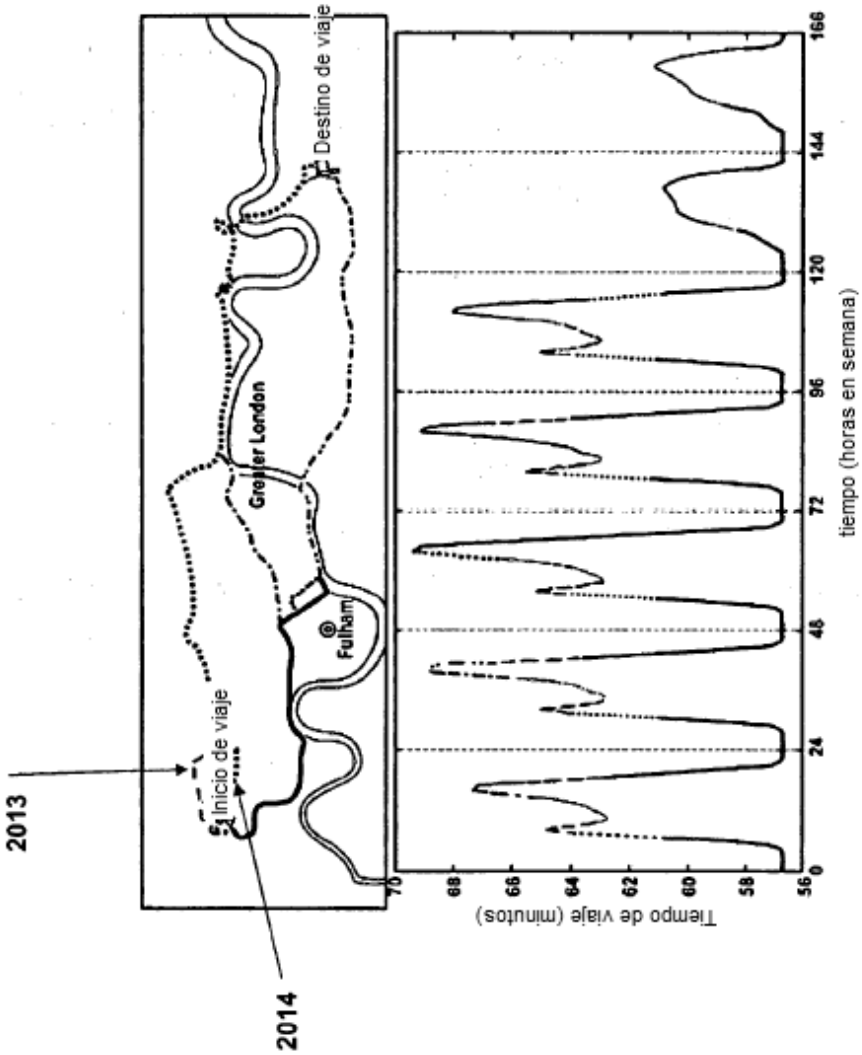

2013
2014
Inicio de viaje
Destino de viaje
Greater London
Fulham
Tiempo de viaje (minutos)
tiempo (horas en semana)
70
68
66
64
62
60
58
56
0
24
48
72
96
120
144
166

Fig. 25

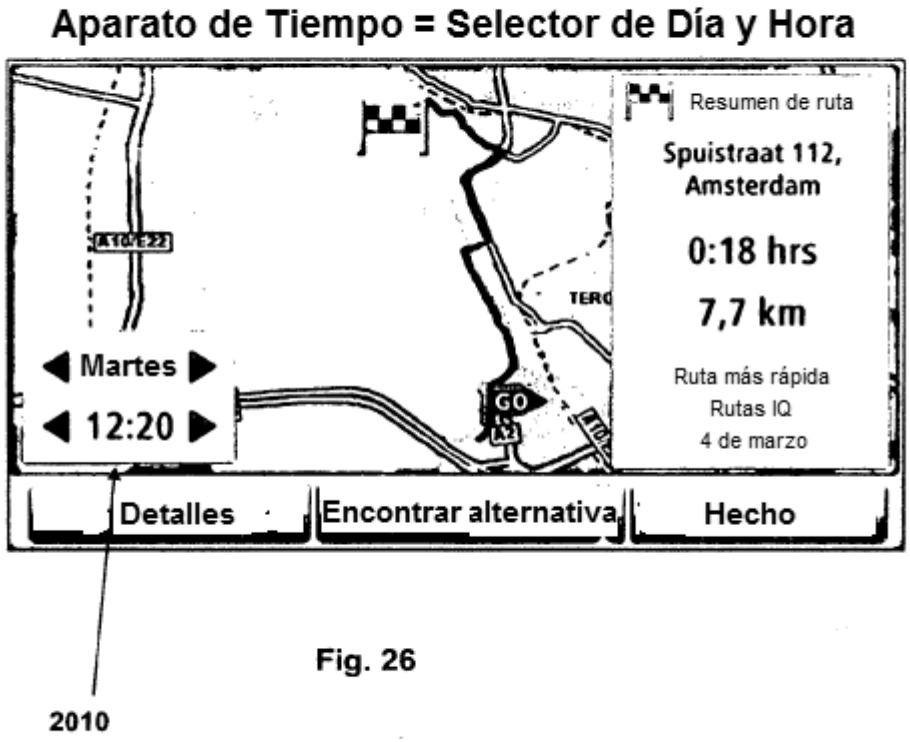

Aparato de Tiempo = Selector de Día y Hora
Resumen de ruta
Spuistraat 112, Amsterdam
0:18 hrs
7,7 km
Ruta más rápida
Rutas IQ
4 de marzo
Martes
12:20
GO
Detalles
Encontrar alternativa
Hecho
2010

Fig. 26

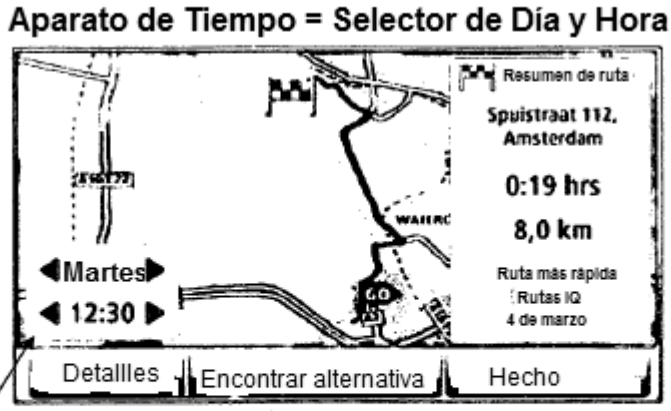

Aparato de Tiempo = Selector de Día y Hora
Resumen de ruta
Spuistraat 112,
Amsterdam
0:19 hrs
8,0 km
Ruta más rápida
Rutas IQ
4 de marzo
Martes
12:30
Detallles
Encontrar alternativa
Hecho
2010

Fig. 27

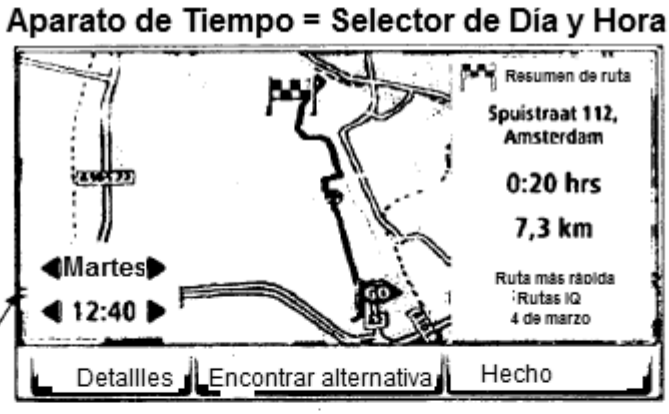

Aparato de Tiempo = Selector de Día y Hora
Resumen de ruta
Spuistraat 112,
Amsterdam
0:20 hrs
7,3 km
Ruta más rápida
Rutas IQ
4 de marzo
Martes
12:40
Detallles
Encontrar alternativa
Hecho
2010

Fig. 28

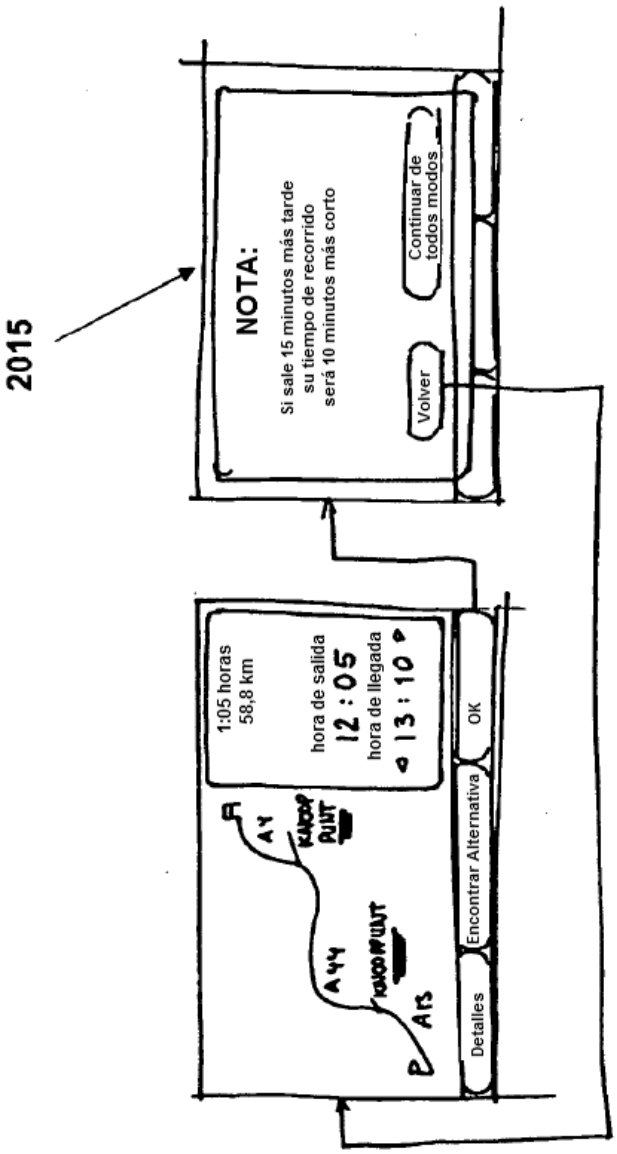

2015
NOTA:
Si sale 15 minutos más tarde su tiempo de recorrido será 10 minutos más corto
Continuar de todos modos
Volver
1:05 horas
58,8 km
hora de salida
12:05
hora de llegada
13:10
OK
Encontrar Alternativa
Detalles

Fig. 29

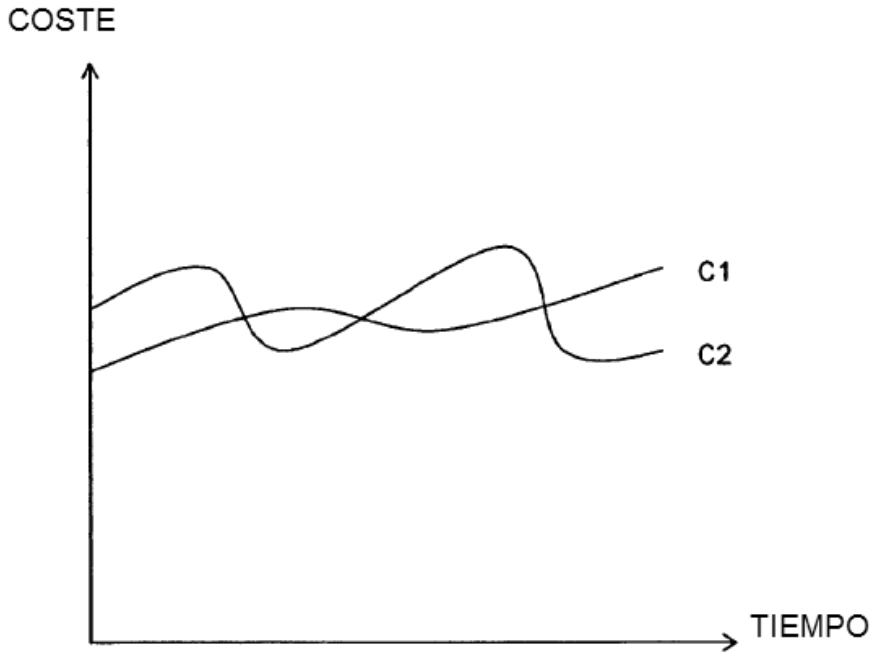


Fig.30

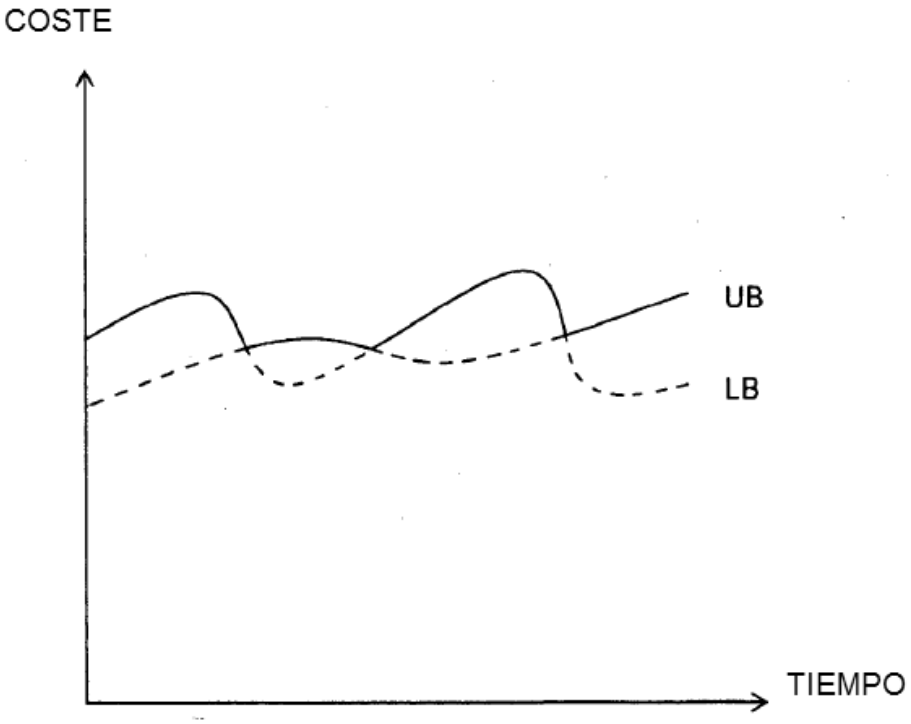


Fig.31