Biblioteca de desarrollo C++para la resolución del problema de la ruta más corta

Manual de usuario

Versión 1.1

infozara.es

Control de versiones

Versión	Contenido
1.0	Versión inicial de vPath. Biblioteca en C++.
1.1	Version compatible con .NET.

SOBRE INFOZARA

Infozara es una empresa que se constituyó en 2006 como spin off de la Universidad de Zaragoza a través del Grupo Nóesis, Grupo Consolidado de Investigación Aplicada del Gobierno de Aragón (España) dirigido por el Profesor Eladio Domínguez.

Infozara tiene una amplia experiencia en la realización de proyectos de I+D+I y en la prestación de servicios a clientes, que cuentan con un alto nivel de valor añadido derivado de las investigaciones industriales realizadas en dichos proyectos.

Todos los productos y desarrollos se han realizado para ser explotados a través de la Web, bajo la forma de lo que actualmente se llama servicios SaaS (Software as a Service).

PROPIEDAD Y CONFIDENCIALIDAD

ÍNDICE

1 Introducción

Los problemas de redes surgen en una gran variedad de situaciones. La representación en redes se utiliza ampliamente en áreas tan diversas como producción, transporte y distribución, planificación de proyectos, localización de instalaciones, etc. De hecho, una representación de redes proporciona un panorama general tan poderoso y una ayuda conceptual para visualizar las relaciones entre las componentes de los sistemas que se usa casi en todas las áreas científicas, sociales y económicas.

Uno de los problemas de mayor aplicación en el análisis de redes es el problema de la ruta más corta. Es un problema que se utiliza principalmente para determinar la ruta más corta en un mapa de carreteras, pero tiene otras aplicaciones.

vPath es una biblioteca de desarrollo en C++ que permite resolver el problema de encontrar la ruta más corta entre dos nodos de una red, sin que el programador deba conocer ningún detalle del algoritmo que calcula la ruta.

Este documento es el manual de usuario de la biblioteca vPath, y presenta toda la información necesaria para poder utilizarlo, con varios ejemplos. Este documento es también una guía de usuario que describe todas las clases y funciones de la biblioteca.

Este manual está dirigido a programadores que desarrollan aplicaciones de cálculo, con conocimientos básicos de C++.

1.1 En este manual

Este manual está estructurado es los siguientes capítulos:

- Optimización en red

  En este capítulo se describen los conceptos básicos de optimización en redes en general, y del problema del camino más corto en particular. No es un manual completo de optimización en red, sino que muestra los conceptos generales, que serán útiles para un buen uso de la biblioteca vPath.

- Biblioteca vPath

  Se ofrece una guía detallada para un correcto uso de la biblioteca vPath. Se comienza con el modelo conceptual del sistema, se describen los pasos a seguir para construir problema del camino más corto, y se presentan varios ejemplos de uso.

- Guía de usuario

  Se detallan todas las clases y funciones de la biblioteca vPath.

1.2 ¿Quiénes somos?

Infozara es una empresa que se constituyó en 2006 como spin off de la Universidad de Zaragoza a través del Grupo Nóesis, Grupo Consolidado de Investigación Aplicada del Gobierno de Aragón (España) dirigido por el Profesor Eladio Domínguez.

Infozara tiene una amplia experiencia en la realización de proyectos de I+D+I y servicios con un alto nivel de valor añadido, derivado de las investigaciones industriales realizadas en

dichos proyectos.

Una característica común a todos los proyectos ha sido la construcción, en cada uno de ellos, de productos en estado precompetitivo y el desarrollo de una metodología de construcción industrial del software como parte integral de un servicio.

Desde su fundación, Infozara:

- Ha participado en diversos proyectos de Desarrollo e Investigación Industrial destacando los proyectos SPOCS (www.spocs.es), LISBB (www.lisbb.es), QRP (qrp.infozara.es), AMBÚ (ambu.infozara.es) y SMOTY (www.smoty.es) del Plan nacional de I+D+i.
- Ha desarrollado productos y componentes industriales en estado precompetitivo en el marco de los proyectos anteriores o como desarrollo posterior ante demanda del mercado.
- Ha construido y está construyendo servicios en la Cloud y en el ámbito del Internet de las Cosas (IoT).
- Tiene una cartera de clientes a los que se les está ofreciendo servicios de valor añadido.

Todos los productos y desarrollos se han realizado para ser explotados a través de la Web, bajo la forma de lo que actualmente se llama servicios SaaS (Software as a Service).

1.3 Conocimientos necesarios

Como vPath es una biblioteca escrita para programadores en C++, este manual asume que el lector tiene experiencia en el desarrollo de programas en C++, y tiene conocimientos de uso de algún entorno de desarrollo en C++.

No son necesarios conocimientos de optimización en red ni de algoritmos de cálculo de rutas.

1.4 Nomenclatura

A lo largo del presente documento se hará referencia a los siguientes términos:

UM	Unidad de longitud.

Las direcciones web o direcciones de correo electrónico que se referencian en este documento, se muestran en color azul, como www.vetu.es/webvetu/vpath.do.

1.5 Soporte

Si es usted ha adquirido licencias de vPath, puede obtener soporte técnico sobre el uso de la biblioteca, poniéndose en contacto con el soporte técnico de Infozara.

Si no es usted no ha adquirido licencias de la biblioteca de Infozara, y tiene cualquier duda o sugerencia, puede ponerse en contacto con el equipo de Infozara por los mecanismos descritos en el apartado 1.7.

1.6 Actualización de versiones

vPath es una biblioteca de resolución de problemas de optimización en red, que va a

incorporar nuevos algoritmos. Si usted ha contratado el servicio de mantenimiento de vPath, recibirá actualizaciones a medida que los solvers vayan publicando nuevas versiones. Los proyectos desarrollados con vPath no se verán afectados por los cambios de versión, ya que el interface público de las clases de vPath no cambia con las nuevas versiones.

1.7 Más información

Si desea más información sobre cualquier aspecto de la biblioteca vPath, puede ponerse en contacto con Infozara, a través de los siguientes medios:

Teléfono: +34 976 25 43 76

Correo electrónico: informa@infozara.es

También puede consultar las siguientes direcciones web:

www.infozara.es

www.vetu.es

www.vetu.es/webvetu/vpath.do

2 Optimización de redes

2.1 Introducción

Los problemas de redes surgen en una gran variedad de situaciones. Las redes de transporte, eléctricas y de comunicaciones predomina en nuestra vida diaria. La representación de redes se utiliza ampliamente en áreas tan diversas como producción, distribución, gestión de recursos, localización de instalaciones, por nombrar solo unos cuantos ejemplos. De hecho, una representación de redes proporciona un panorama general tan poderoso y una ayuda conceptual para visualizar las relaciones entre los componentes de los sistemas, que se usa casi en todas las áreas científicas, sociales y económicas.

Uno de los mayores desarrollos en investigación de operaciones ha sido el rápido avance tanto en la metodología como en la aplicación de los modelos de optimización de redes. vPath es una bliblioteca de programación en C++ que permite resolver problemas de optimización de redes, sin necesidad de conocer los algoritmos específicos que se utilizan.

2.2 Terminología

Se ha desarrollado una terminología relativamente extensa para describir los tipos de redes y sus componentes. A continuación se describe la terminología de redes, que es básica para trabajar con la biblioteca vPath.

Una gráfica (o red, o árbol) consiste en un conjunto de puntos y un conjunto de líneas que unen ciertos pares de puntos. Los puntos de llaman nodos (o vértices). Por ejemplo, la red de la Figura 2-1 tiene 7 nodos, presentados por 7 círculos. Las líneas se llamana arcos (o aristas, o ramas). Por ejemplo la red de la Figura 2-1 tiene 12 arcos, que corresponden a todos los caminos que unen nodos. Los arcos de una red pueden tener un flujo de algún tipo que pasa por ellos. Por ejemplo, en la red de la Figura 2-1, los numero escritos sobre cada arco representran la distancia del arco.

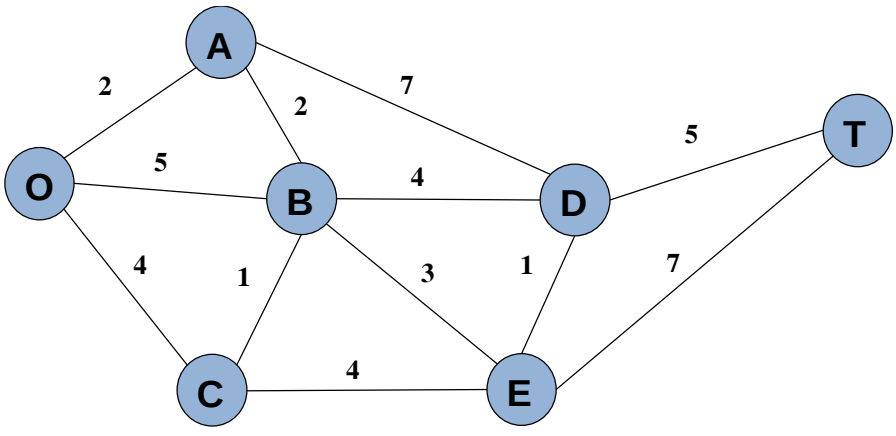

Figura 2-1. Red de ejemplo

2.3 Problema de la ruta más corta

El problema de optimización de redes más utilizado es sin duda, el problema del camino más corto, que se resuelve mediante el algoritmo de Dijkstra. La formulación del problema consiste en que se dispone una red de nodos y arcos, y se conoce también el peso de cada arco. El problema consiste en determinar la ruta más corta entre dos nodos del árbol.

Algunos ejemplos del problema de la ruta más corta son los siguientes:

- Ruta entre dos localidades

  Dado un mapa de carreteras, el problema consiste en encontrar la ruta más corta (en distancia) entre dos localidades.

  En este caso, los nodos son las localidades, las aristas son las carreteras, y el peso de las aristas es la distancia. Es la mayor aplicación del problema de la ruta más corta.

- Encaminamiento de paquetes

  Conocidas los trayectos que realizan los vehículos de transporte, el problema consiste en definir la ruta más costa que debe seguir un paquete que debe transportarse desde un origen a un destino

  En este caso, los nodos son los trayectos de los vehículos, las aristas enlazan trayectos que se puede realizar según la temporalidad de los trayectos, y el peso de las aristas puede ser la duración o la distancia de cada trayecto, en función de si el objetivo es minimizar el tiempo de transporte del paquete, o la distancia recorrida por el paquete.

- Ruta de vagones sin carga

  Existen unos trenes con horarios ya definidos, y existe la necesidad de transportar vagones de una estación a otra. El problema consiste en encontrar la ruta de trenes que deben transportar los vagones, para minimizar la distancia recorrida por los vagones, o el tiempo de viaje.

  En este caso, los nodos son los trayectos de los trenes (o las tuplas estación-hora de salida del tren), las aristas son la conexión entre trenes consecutivos con el mismo destino -> origen (contando el tiempo de cambio de vagones), y el peso de las aristas puede ser la duración o la distancia de cada trayecto de tren, en función de si el objetivo es minimizar el tiempo de transporte del vagón, o la distancia recorrida por el vagón.

- Encaminamiento de paquetes de información por los routers

  Un mensaje puede tardar un cierto tiempo en atravesar una línea de comunicación (por ejemplo, por efectos de congestión). El objetivo del problema es encontrar un camino entre estos dos nodos cuyo tiempo de envío del menaje sea el mínimo.

  En este caso, los nodos de la red son los routers, las aristas son las líneas de comunicación, y los pesos de las aristas serían los tiempos previstos de paso del mensaje por las líneas.

- Enrutamiento de aviones y tráfico aéreo

  En este caso, el problema consiste en encontrar la mejor secuencia de vuelos para llegar antes a un destino. Un agente de viajes tiene acceso a los datos de vuelos (origen, destino,

hora de salida, hora de llegada) y debe determinar la hora de llegada más temprana para llegar a un destino, dado un aeropuerto de origen y hora de inicio.

En este caso, los nodos son los vuelos (o las tuplas aeropuerto-hora de salida del vuelo), las aristas son la conexión entre vuelos consecutivos con mismo destino -> origen (contando el tiempo de tránsito en el aeropuerto intermedio), y el peso de las aristas es la duración de los vuelos.

- Movimiento de piezas en fábricas

  En una fábrica hay máquinas que tratan piezas, y estas piezas deben pasar por diversas máquinas para llegar a ser un producto terminado. Hay un flujo continuo de piezas por un mapa de carriles. Se trata de determinar el camino más corto que siguen las piezas por los carriles en cada momento, porque los carriles pueden estar bloqueados por operaciones de trabajo.

  En este caso, los nodos son las máquinas, las aristas son los carriles, y el peso de las aristas es la distancia de los carriles.

3 Manual de usuario

3.1 Introducción

vPath es una librería C++ para resolver problemas de optimización en red. En este capítulo se exlica todo lo necesario para poder utilizar vPath. También se explica el contenido de vPath, y se explica su uso, mediante un ejemplo.

3.2 Contenido

Al adquirir una licencia de vPath, Infozara envía al cliente la biblioteca de desarrollo vPath, con todo lo necesario para desarrollar aplicaciones con procesos de optimización en red.

La biblioteca vPath está escrita en C++, y no está concebida para ningún entorno de desarrollo ni sistema operativo en especial. El lenguaje C++ se puede compilar y ejecutar en todos los sistemas operativos. vPath está escrita en C++ estándar, y solo el proceso de verificación de licencias hace llamadas al sistema operativo.

Desde la versión v1.1, vPath es compatible con el desarrollo en la plataforma .NET de Microsoft.

3.3 Instalación

El framework Lin2any se distribuye como un fichero comprimido con el código fuente y la documentación, según la siguiente estructura:

\include	Contiene los ficheros de cabecera .h.
\lib	Contiene le fichero de biblioteca vPath.lib para desarrollar programas C++, y el ensamblado vPathWrapper.dll, para desarrollar programas sobre la plataforma .NET de Microsoft.
\doc	Contiene el manual de usuario de vPath.

La instalación consiste en descomprimir el fichero para obtener la estructura anterior, y copiar los ficheros al directorio de proyecto de desarrollo.

3.4 Licencias

vPath se licencia por cada ordenador donde se utiliza la biblioteca. Las licencias van asociadas a la mac address de cada ordenador.

El proceso de validación de licencias es el siguiente:

- Identificar la MAC ADDRESS del ordenador donde se va a instalar la licencia. Para obtener la MACC ADDRESS en un ordenador son sistema operativo Windows, se debe teclear ipconfig /all en la línea de comandos, y observar el valor 'Physical Address'.
- Enviar la MAC ADDRESS al servicio técnico de Inforaza.

- El servicio técnico de Infozara enviará al cliente un fichero de licencia (.lic).
- Colocar el fichero de licencia en el directorio donde esté cada ejecutable que incluye a la librería vPath.

3.5 Configuración de un proyecto C++

Para poder hacer uso de vPath en un proyecto C++:

- Se deben añadir las directivas #include con los ficheros .h correspondientes a las clases de vPath.
- Se debe añadir al proyecto la biblioteca vPath, y asegurar que el proyecto enlaza con ella.

3.6 Configuración de un proyecto .NET

Para poder hacer uso de vPath en un proyecto .NET:

- Se debe añadir referencia al ensamblado vPathWrapper.dll, que contiene clases escritas en C++ administrado, para su uso como cualquier ensamblado .NET.

3.7 Modelo conceptual

La Figura 3-1[1] muestra un diagrama de clases con el modelo conceptual de vPath. El modelo está dividido en dos partes muy diferenciadas: la primera parte corresponde a la definición del problema de red como un árbol, y la segunda corresponde a la resolución de los diferentes problemas que se pueden plantear sobre una misma representación de problema de red como árbol.

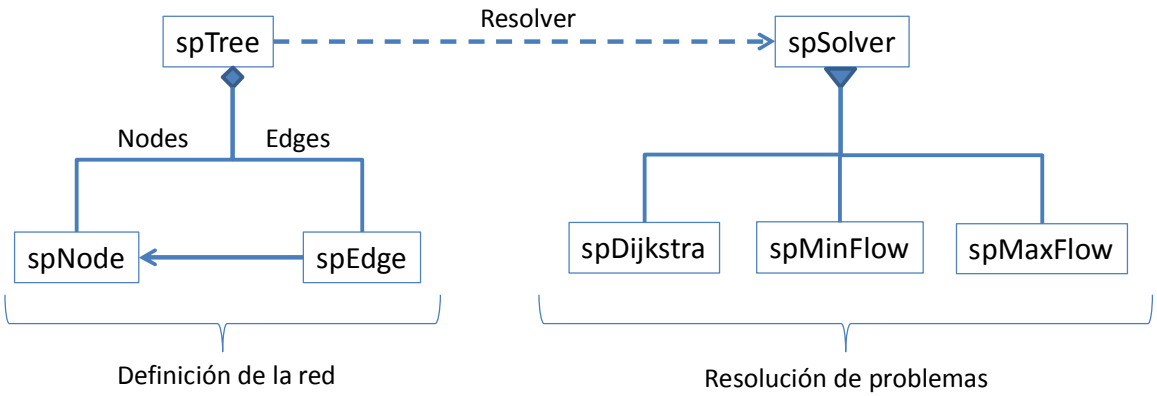

Figura 3-1. Modelo conceptual de vPath

La definición del problema de red se hace según la terminología de la optimización en red y no tiene relación con ninguna técnica de resolución. El problema de red se representa como un árbol (spTree), que se compone de nodos (spNode), enlazados por aristas (spEdge).

La resolución del problema de red se hace mediante clases específicas para cada tipo de

[1] La versión actual vPath v1.0 no incluye spMinFlow ni spMaxFlow

problema a resolver sobre el problema representado como un árbol. En función del tipo de problema a resolver, se utilizará un solver específico para ese problema.

La unión de la representación del modelo de red y la resolución de un problema de red viene dada por la relación entre spTree y spSolver. Como spSolver es una clase abstracta, spTree nunca sabrá cuál es el solver que va a resolver el problema. Para añadir un nuevo algoritmo en un programa, se debe crear una nueva clase que derive de spSolver y que sobrescriba todos sus métodos abstractos. Desde ese momento, se podrá utilizar el nuevo solver sin que se haya modificado para nada la representación del problema.

El modelo de vPath es una variante de bien conocido patrón de diseño Bridge[2], cuya enfoque es desacoplar una abstracción de su implementación, para que ambas puedan variar independientemente. En el caso de vPath, la abstracción es el árbol que representa a un problema, y la implementación corresponde al solver que resuelve el problema. Con este modelo, se puede cambiar el solver sin tocar el árbol, o incluso se pueden resolver varios tipos de problema sobre una misma red.

3.8 Ejemplo en C++

En este apartado se va a mostrar el uso de la biblioteca vPath en una aplicación C++, a través de un sencillo ejemplo. Sea el siguiente problema:

> 'Dada la red de la Figura 3-2, se desea saber la ruta más corta desde el nodo O al nodo T [3]. Los números que se muestran en las aristas representan la distancia entre dos nodos.'

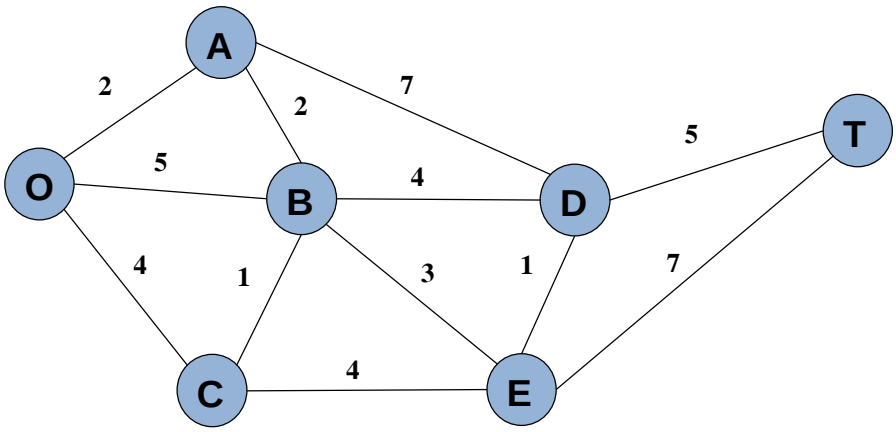

Figura 3-2. Red para el problema de ejemplo

Se parte de la instalación de la biblioteca vPath, por ejemplo en el directorio C:\vPathLib. Este directorio contiene dos subdirectorios: el subdirectorio \include contiene todos los ficheros de cabecera (.h) de vPath, y el subdirectorio \lib contiene el fichero con la biblioteca de vPath (vPath.lib). Se deben establecer los mecanismos, en función de compilador que se utilice, para que el programa pueda acceder a los ficheros de cabecera y que pueda enlazar con la biblioteca.

El código fuente escrito en C++ que resuelve el problema definido anteriormente con la

[2] Design Patterns: Elements of Reusable Object-Oriented Software. Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. Addison-Wesley Professional Computing Series. ISBN: 0-201-63361-2.
[3] Introducción a la investigación de operaciones. Frederick Hillier, Gerald Lieberman. McGraw-Hill.

biblioteca vPath es el siguiente (se muestra el número de las líneas de código para facilitar la posterior explicación):

```
1  // Definición del árbol
2  spTree oTree;
3
4  // Se crean los vértices del árbol
5  spVertex* pO = oTree.addVertex("O");
6  spVertex* pA = oTree.addVertex("A");
7  spVertex* pB = oTree.addVertex("B");
8  spVertex* pC = oTree.addVertex("C");
9  spVertex* pD = oTree.addVertex("D");
10 spVertex* pE = oTree.addVertex("E");
11 spVertex* pT = oTree.addVertex("T");
12
13 // Se crean la aristas
14 oTree.setEdge(pO, pA, 2);
15 oTree.setEdge(pO, pB, 5);
16 oTree.setEdge(pO, pC, 4);
17 oTree.setEdge(pA, pB, 2);
18 oTree.setEdge(pA, pD, 7);
19 oTree.setEdge(pB, pC, 1);
20 oTree.setEdge(pB, pD, 4);
21 oTree.setEdge(pB, pE, 3);
22 oTree.setEdge(pC, pE, 4);
23 oTree.setEdge(pD, pE, 1);
24 oTree.setEdge(pD, pT, 5);
25 oTree.setEdge(pE, pT, 7);
26
27 // Crea el algoritmo del camino más corto desde el nodo O
28 spDijkstra oSolver;
29 oSolver.setOrigen( pO );
30
31 // Resuelve el problema
32 oTree.solve(&oSolver);
33
34 // Obtiene el resultado
35 vector<spVertex*> vRoute;
36
37 // Ruta más corta desde O a T
38 cout << "\n\n";
39 double dDistancia = oSolver.getBest(pT, vRoute);
40 cout << "\n\nRuta mas corta desde " << pO->name() << " hasta " <<
41 pT->name() << " con distancia " << dDistancia;
42 for (vector<spVertex*>::iterator itv = vRoute.begin();
43      itv != vRoute.end(); itv++)
44      cout << "\n" << (*itv)->id() << " " << (*itv)->name();
45
46 // Ruta más corta desde O a E
47 cout << "\n\n";
48 dDistancia = oSolver.getBest(pE, vRoute);
49 cout << "\n\nRuta mas corta desde " << pO->name() << " hasta " <<
50 pE->name() << " con distancia " << dDistancia;
51 for (vector<spVertex*>::iterator itv = vRoute.begin();
```

```
52        itv != vRoute.end(); itv++)
53        cout << "\n" << (*itv)->id() << " " << (*itv)->name();
```

Los pasos para resolver un problema de encontrar la ruta más corta con vPath son los siguientes: crear el árbol, definir el algoritmo que se aplica sobre el árbol, resolver el problema y acceder al resultado. A su vez, definir el árbol consiste en crear primero los vértices y después las aristas que conectan los vértices.

En la línea 1 del código fuente del ejemplo anterior, se crea una instancia de árbol, que corresponde a la clase spTree. Entre las líneas 1 y 11 se crean todos los vértices del árbol. Los vértices los crea el árbol, que devuelve un puntero al vértice creado. Los vértices se crean accediendo a la función addVertex de la clase spTree. Entre las líneas 14 y 25 se crean las aristas. Para crear una arista, se utiliza la función setEdge de la clase de árbol spTree. Los argumentos de la función setEdge son el vértice origen de la arista, el vértice destino, y el peso de la arista. En este problema, el peso de una arista es la distancia entre los nodos, y el problema es minimizar la distancia entre ciertos nodos.

El siguiente paso es crear el algoritmo que se va a aplicar sobre el árbol. En este caso se quiere resolver el problema de encontrar la ruta más corta, por tanto se utiliza el algoritmo de Dijkstra, dado por la clase spDijkstra. En la línea 28 se crea una instancia del algoritmo, y en la línea 29 se configura la clase de algoritmo. La configuración del algoritmo de Dijkstra consiste en especificar el nodo origen, que se hace en la línea 29. En la línea 32 se resuelve el problema, indicando al árbol el algoritmo a utilizar. En este momento, se lanza el algoritmo de Dijkstra, que calcula el camino más corto desde el nodo origen a todos los nodos del árbol.

Una vez resuelto el problema, queda acceder al resultado. En este tipo de problema, acceder al resultado significa especificar el nodo destino, y el resultado es la ruta más corta a ese nodo destino. La ruta más corta se obtiene accediendo a la función getBest del algoritmo dado por la clase spDijkstra. En la línea 39 se pregunta la ruta más corta al nodo T (partiendo del nodo O), y en la línea 48 se acede a ruta más corta al nodo E (partiendo del nodo O). La función getBest devuelve la distancia, y recibe como parámetros el nodo destino, y un vector donde se almacenará la ruta más corta al nodo destino, expresada como la lista ordenada de nodos de la ruta más corta.

Entre las líneas 40 y 44, se muestra por pantalla la ruta más corta para ir desde el nodo O al nodo T, y entre las líneas 49 y 53 se muestra por pantalla la ruta más corta para ir desde el nodo O al nodo E.

3.9 Ejemplo en C++ con Microsoft Visual Studio®

En este apartado del manual de usuario de vPath, se muesta un ejemplo de uso completo con el compliador C++ de Microsoft Visula Studio Express 2015, paa resolver el siguiente problema:

> 'Dada la red de la Figura 3-2, se desea saber la ruta más corta desde el nodo O al nodo T [4]. Los números que se muestran en las aristas representan la distancia entre dos nodos.'

[4] Introducción a la investigación de operaciones. Frederick Hillier, Gerald Lieberman. McGraw-Hill.

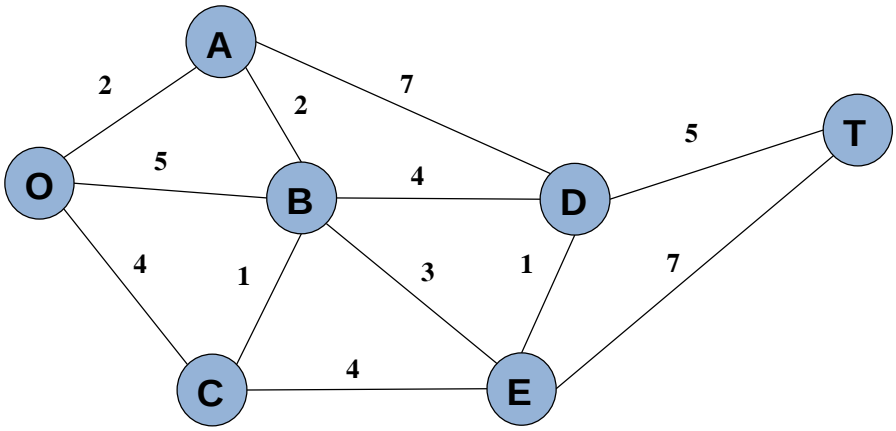

Figura 3-3. Red para el problema de ejemplo

Se parte de la instalación de la biblioteca vPath en el directorio C:\vPathLib. Este directorio contiene dos subdirectorios: el subdirectorio \include contiene todos los ficheros de cabecera (.h) de vPath, y el subdirectorio \lib contiene el fichero con la biblioteca de vPath (vPath.lib).

Para desarrollar una aplicación de consola en C++ que resuelva el problema anterior, se pueden seguir los siguientes pasos:

1. Abrir Visual Studio Express 2015 para escritorio de Windows.
2. Seleccionar la opción de menú Archivo -> Nuevo proyecto …
3. Dentro de Plantillas, seleccionar Visual C++, y elegir el tipo de proyecto 'Proyecto vacío'.
4. Fijar el nombre de la solución y del proyecto como vPathCpp.
5. En el explorador de soluciones, pulsar con el botón derecho del ratón sobre el proyecto vPathCpp.
6. Dentro de las propiedades de configuración, seleccionar la propiedad 'Directorio de VC++'. Incluir el path C:\vPathLib\include dentro de 'Directorios de archivos de inclusion'.
7. Dentro de las propiedades de configuración, desplegar la opción 'Vinculador', y pulsar sobre 'General'. Añadir el path C:\vPathLib\lib dentro de 'Directorios de biblioteca adicionales'.
8. Dentro de las propiedades de configuración, desplegar la opción 'Vinculador', y pulsar sobre 'Entrada'. Añadir vPath.lib dentro de 'Dependencias adicionales'.
9. Pulsar el botón 'Aplicar', y después puñlsar el botón 'Aceptar'.
10. En el explorador de soluciones, pulsar con el botón derecho del ratón sobre el nombre del proyecto. Selccionar la opción del menú 'Agregar nuevo elemento …'. Seleccionar el tipo de fichero 'Archivo C++ (.cpp)', e introducir el nombre main.cpp

3.10 Ejemplo en C#

En este apartado se va a mostrar el uso de la biblioteca vPath en una aplicación C#, a través de un sencillo ejemplo. Sea el siguiente problema:

'Dada la red de la Figura 3-2, se desea saber la ruta más corta desde el nodo O

al nodo T [5]. Los números que se muestran en las aristas representan la distancia entre dos nodos.'

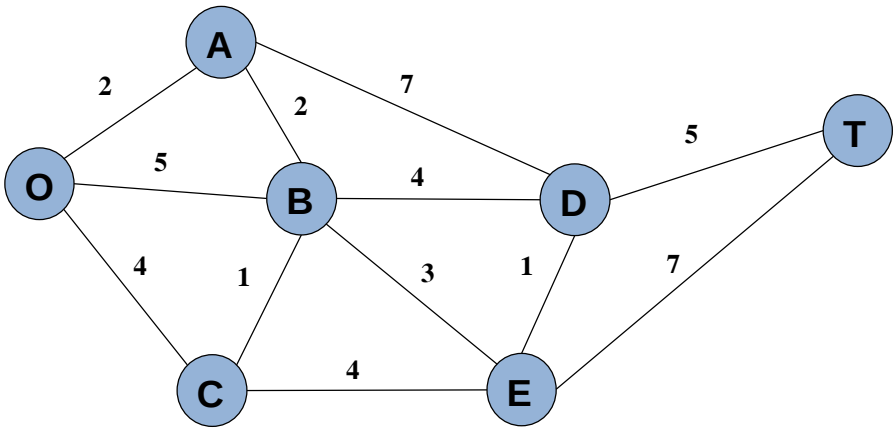

Figura 3-4. Red para el problema de ejemplo

El programa C# que resuelve este problema con la biblioteca vPath es el siguiente (se muestra el número de las líneas de código para facilitar la posterior explicación):

```
1  // Se crea el acceso a vPath
2  vPathWrapper.vPathCpp oPath = new vPathWrapper.vPathCpp();
3
4  // Se crean los vértices del árbol
5  spaVertex pO = oPath.addVertex("O");
6  spaVertex pA = oPath.addVertex("A");
7  spaVertex pB = oPath.addVertex("B");
8  spaVertex pC = oPath.addVertex("C");
9  spaVertex pD = oPath.addVertex("D");
10 spaVertex pE = oPath.addVertex("E");
11 spaVertex pT = oPath.addVertex("T");
12
13 // Se crean la aristas
14 oPath.setEdge(pO, pA, 2);
15 oPath.setEdge(pO, pB, 5);
16 oPath.setEdge(pO, pC, 4);
17 oPath.setEdge(pA, pB, 2);
18 oPath.setEdge(pA, pD, 7);
19 oPath.setEdge(pB, pC, 1);
20 oPath.setEdge(pB, pD, 4);
21 oPath.setEdge(pB, pE, 3);
22 oPath.setEdge(pC, pE, 4);
23 oPath.setEdge(pD, pE, 1);
24 oPath.setEdge(pD, pT, 5);
25 oPath.setEdge(pE, pT, 7);
26
27 // Resuelve desde el nodo 0
28 oPath.solveFromVertex(pO);
29
30 // Calcula la menor distancia
31 List<spaVertex> lstPath = null;
```

[5] Introducción a la investigación de operaciones. Frederick Hillier, Gerald Lieberman. McGraw-Hill.

```
32  double dCoste = oPath.getBest(pT, ref lstPath);
33
34  lbResultado.Items.Add( pO.getName() + " -> " + pT.getName() );
35  lbResultado.Items.Add( "Ruta:" );
36  foreach (spaVertex spav in lstPath)
37    cout << spav.getName() << "\n";
38  cout << "Coste: " << dCoste;
```

Los pasos para resolver un problema de encontrar la ruta más corta con vPath son los siguientes: crear el árbol, definir el algoritmo que se aplica sobre el árbol, resolver el problema y acceder al resultado. A su vez, definir el árbol consiste en crear primero los vértices y después las aristas que conectan los vértices.

En la línea 1 del código fuente del ejemplo anterior, se crea una instancia de árbol, que corresponde a la clase vPathCpp. Esta clase está escrita en C++ administrado y es el puente al motor de cáculo de vPath, escrito en C++. El espacio de nombres de vPath, que incorpora el ensamblado vPathWrapper.dll es PathWrapper.

Entre las líneas 5 y 11 se crean todos los vértices del árbol. Los vértices los crea el árbol, que devuelve una referencia al vértice creado. Los vértices se crean accediendo a la función addVertex de la clase vPathCpp. Los vértices delárbol están representados por la clase spaVertrex, que es una clase escrita en C++ adminsitrado.

Entre las líneas 14 y 25 se crean las aristas. Para crear una arista, se utiliza la función setEdge de la clase de spPathCpp. Los argumentos de la función setEdge son el vértice origen de la arista, el vértice destino, y el peso de la arista. En este problema, el peso de una arista es la distancia entre los nodos, y el problema es minimizar la distancia entre ciertos nodos.

Una vez configurado el árbol, el siguiente paso consiste en resolver el problema. El problema del camino más costro se resulve en dos pasos: en la línea 28 se indica que el nodo origen es el nodo pO, que corresponde al nodo O, y el segundo paso es obtener la ruta más corta a un nodo. En este ejemplo, en la línea 32 se indica que se busca la ruta más corta al nodo pT, que corresponde al nodo T. En la línea 32 también se indica que la solución se debe almacenar en la variable lstPath, que es una lista con los vértices ordenados de la solución.

Una vez resuelto el problema, queda acceder al resultado. Entre las líneas 34 y 37, se muestra por pantalla la ruta más corta para ir desde el nodo O al nodo T, y en la línea 38 se muestra el coste de la solución.

4 Manual de referencia C++

Este capítulo hace referencia a todas las clases públicas de la biblioteca vPath.

4.1 spTree

Categoría Clase de desarrollo.

Descripción spTree es la representación como un árbol de un problema de optimización en red.

Fichero include

```
#include "vector"
using namespace std;

#include "spCommon.h"
#include "spVertex.h"
#include "spSolver.h"

class spTree
{
   vector<spVertex*> mvVertices;

   vector<vector<Node>> ady;

public:
   spTree(void);
   virtual ~spTree(void);

   spVertex* addVertex(string strName);

   void setEdge(spVertex* pOrigen, spVertex* pDestination,
                int nWeight);

   void solve(spSolver* pSolver);
};
```

Constructores

```
spTree (void);
```

El constructor no recibe ningún parámetro.

Funciones miembro

```
spVertex* addVertex(string strName);
```

Añade un vértice al árbol, dado por su nombre.

```
void setEdge(spVertex* pOrigen, spVertex* pDestination,
             int nWeight);
```

Añade una arista en el árbol, dada por los nodos origen y destino (clase

spVertex) y el peso de la arista.

```
void solve(spSolver* pSolver);
```

Resuelve un problema de optimización en red sobre el árbol, ejecutando el algoritmo que recibe como argumento.

4.2 spVertex

Categoría Clase de desarrollo.

Descripción spVertex representa un nodo dentro del árbol definido por la clase spTree.

Estas variables no se pueden crear directamente, llamando al constructor de la clase, sino que debe crearlas la clase spTree.

Fichero include

```
class spVertex
{
   friend class spTree;

   string    mstrName;
   int       mnId;

   spVertex(int nId, string& strName);

public:

   virtual ~spVertex();

   int id(void) const;

   bool es(string& strName)

   string name(void) const;
};
```

Constructores `spVertex(int nId, string& strName);`

Crea una instancia de la clase spVertex, que representa un vértice del árbol del problema de optimización en red.

Es un constructor privado, que solo se puede instanciar desde la clase spTree.

mnId es el identificador numérico único de la variable, y lo fija la clase spTree.

strName es el nombre único que identifica al vértice.

Funciones miembro

`int id(void) const;`

Devuelve el índice único del vértice.

`bool es(string& strName)`

Indica si este vértice corresponde al nombre que recibe como argumento.

`string name(void) const;`

Devuelve el nombre del vértice.

4.3 spSolver

Categoría Clase abstracta.

Descripción Define el comportamiento básico de un algoritmo de optimización en red que se lanza sobre un árbol, compuesto por nodos y aristas.

Fichero include

```
#include <vector>
using namespace std;

#include "spCommon.h"
#include "spVertex.h"

class spSolver
{
public:

   spSolver(void)

   virtual ~spSolver(void)

   virtual void solve(vector<spVertex*>& vVertices,
                      vector<vector<Node>>& vEgdes) = 0;
};
```

Constructores

```
spSolver(void)
```

Constructor público que no recibe ningún argumento.

Funciones miembro

```
virtual void solve(vector<spVertex*>& vVertices,
                   vector<vector<Node>>& vEgdes) = 0;
```

Función virtual pura que implementa un algoritmo de optimización en red. Todas las clases derivadas de spSolver deben definir esta función.

vVertices es un vector que contiene los vértices del árbol.

vEdges es un vector que contiene todas las aristas que salen de cada vértice del árbol.

4.4 spDijkstra

Categoría Clase de desarrollo.

Descripción Implementa el algoritmo de Dijkstra, que resuelve el problema de la ruta más corta sobre un árbol de optimización en red.

Fichero include

```
#include "spSolver.h"

class spDijkstra : public spSolver
{
public:
   spDijkstra(void);
   virtual ~spDijkstra(void);

   void setOrigen(spVertex* pOrigen);

   virtual void solve( vector<spVertex*>& vVertices,
                       vector<vector<Node>>& vEdges );

   double getBest(spVertex* pDestination,
                  vector<spVertex*>& vRoute);
};
```

Constructores

```
spDijkstra(void);
```

Constructor público que no recibe ningún parámetro.

Funciones miembro

```
void setOrigen(spVertex* pOrigen);
```

Indica el vértice origen del árbol, sobre el que se van calcular la ruta más corta.

pOrigen es el vértice origen de la ruta más corta que se va a calcular.

```
virtual void solve( vector<spVertex*>& vVertices,
                    vector<vector<Node>>& vEdges );
```

Función virtual que redefine a la función virtual pura de la clase base spSolver.

Implementa el algoritmo de Dijkstra para encontrar el camino más corto desde el nodo origen a cualquier otro nodo del árbol.

vVertices es un vector que contiene los vértices del árbol.

vEdges es un vector que contiene todas las aristas que salen de cada vértice del árbol.

```
double getBest(spVertex* pDestination, vector<spVertex*>& vRoute);
```

Devuelve el valor de la distancia más corta desde el nodo origen al nodo destino que recibe como argumento.

pDestination es el nodo destino al que se quiere conocer la ruta más corta,

desde el nodo origen.

vRoute es el vector con la secuencia de nodos que suponen la ruta más corta desde el nodo origen al nodo destino *pDestination*.

5 Manual de referencia .NET