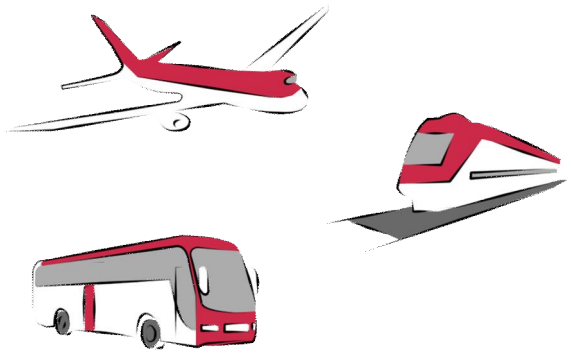

vFlightConnect

Biblioteca de desarrollo para el cálculo
de conexiones de vuelos o rutas de transporte intermodal.

Manual de usuario

Versión 1.1

infozara.es

Control de versiones

Versión	Contenido
1.0	Versión inicial de vFlightConnect. Biblioteca en C++.
1.1	Versión 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 turismo, 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, en particular para el viajero o turista.

Una aplicación del problema de la ruta más corta es el cálculo de las mejores conexiones entre vuelos en particular, y entre medios de transporte intermodal en general.

vFlightConnect[1] es una biblioteca de desarrollo en C++ que permite resolver el problema de encontrar las mejores conexiones entre diferentes medios de transporte entre dos localidades, sin que el programador deba conocer ningún detalle del algoritmo que calcula las conexiones. vFlightConnect es una capa de especialización sobre la biblioteca sPath[2], que también comercializa Infozara a través de la plataforma Vetu.

Este documento es el manual de usuario de la biblioteca vFlightConnect, y presenta toda la información necesaria para poder utilizarla, 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 de redes

  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 vFlightConnect.

- Biblioteca vFlightConnect

  Se ofrece una guía detallada para un correcto uso de la biblioteca vFlightConnect. Al estar basada en sPath, el uso de vFlightConnect es realmente sencillo, porque solo es necesario enviar la lista de los diferentes medios de transporte disponibles, y la biblioteca devuelve las mejores conexiones, bajo diferentes criterios.

- Guía de usuario

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

[1] vFlightConnect es una marca registrada propiedad de Infozara
[2] sPath es una marca registrada propiedad de Infozara

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 vFlightConnect 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++.

vFlightConnect también tiene interface directo con la plataforma .NET. Si se utiliza este interface, el prograador debe tener unos mínimos conocimientos de alguno de los lenguajes copatible con esta plataforma.

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 monetaria.
UT	Unidad de tiempo (por ejemplo segundos, minutos, o cinco minutos).

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

1.5 Soporte

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

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

1.6 Actualización de versiones

vFlightConnect es una biblioteca de resolución de problemas de optimización en red, aplicado a la conexión de medios de transporte, que evolucionará con nuevas funcionalidades. Si usted ha contratado el servicio de mantenimiento de vFlightConnect, recibirá actualizaciones a medida que Infozara vaya publicando nuevas versiones. Los proyectos desarrollados con vFlightConnect no se verán afectados por los cambios de versión, ya que el interface público de las clases de vFlightConnect no cambia con las nuevas versiones.

1.7 Más información

Si desea más información sobre cualquier aspecto de la biblioteca vFlightConnect, 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/vflightconnect.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 predominan en nuestra vida diaria. La representación de redes se utiliza ampliamente en áreas tan diversas como turismo, transporte, 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. Los vuelos (y otros medios de transporte) que ofrece por ejemplo un buscador de Internet, se pueden representar como una red, y el problema de optimización en red consiste en encontrar la mejor combinación de vuelos para ir desde un origen a un destino. La mejor combinación puede ser la de menor coste, o la más rápida, o la de menores tiempos de escala, o la de menores tiempos de vuelo.

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. sPath es una biblioteca de programación en C++ que permite resolver problemas de optimización de redes, sin necesidad de conocer los algoritmos específicos que se utilizan. vFlightConnect es una biblioteca desarrollada sobre sPath, para optimización de redes de transporte intermodal.

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 vFlightConnect.

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 llaman 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 representan la distancia del arco.

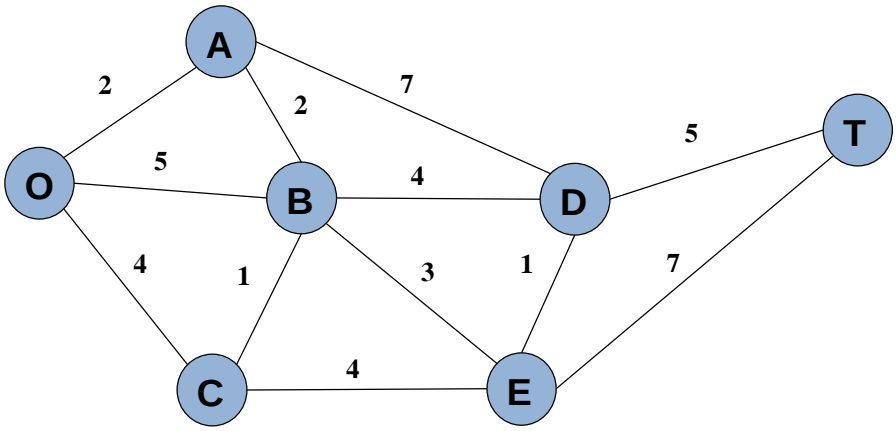

Figura 2-1. Red de ejemplo

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:

- 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.

  vFlightConnect es una biblioteca específica de programación para resolver un problema de optimización de redes, donde los nodos son vuelos (u otros medios de transporte), y los arcos que unen esos vuelos pueden medir coste, tiempo de escala, duración del viaje, o duración de los vuelos.

- 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. Este problema representa 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 pueden realizar según la temporalidad de los trayectos, y el peso de las aristas puede ser la duración o la distancia de cada trayecto. 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.

- 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.

2.4 Rutas de transporte intermodal

Las rutas de transporte intermodal están formadas por diversos medios de transporte, definidos por un origen, un destino, una hora de salida y una hora de llegada. Los medios de transporte pueden ser aéreo, autobús, marítimos, etc. Aunque el nombre de la biblioteca vFlightConnect sugiera su aplicación únicamente al transporte aéreo, se puede aplicar a cualquier otro medio de transporte.

Una conexión entre dos medios de transporte conlleva un coste, un tiempo de escala, o un tiempo de viaje. Diferentes conexiones llevan a diferentes valores de coste, tiempo de escala o tiempo de viaje. En general, la conexión de menor coste es distinta a la conexión de menor tiempo de escala, y distinta a la conexión de menos tiempo de viaje, o distinta a la más rápida

desde el origen al destino. El objetivo que se busca es encontrar simultáneamente las mejores conexiones para cada uno de los criterios. De esta forma, se puede elegir la mejor conexión, sopesando todos los criterios.

Una aplicación directa del problema de rutas de transporte intermodal son los buscadores de vuelos que están disponibles por Internet. En general, estos buscadores muestran una gran cantidad de opciones de vuelos, pero debe ser el cliente el que busque la mejor conexión, en función de sus preferencias, y esa labor puede llevar un tiempo apreciable. Sería muy útil para el cliente, que el buscador le mostrara directamente las mejores conexiones para cada criterio, y así el cliente puede hacer su elección rápidamente. En el apartado 4.7 se discute esta problemática con un ejemplo.

3 Conexiones vuelos o medios de transporte

3.1 Modelo conceptual de vFlightConnect

El modelo conceptual de la biblioteca vFlightConnect es muy sencillo, debido a que tiene un carácter muy específico, y únicamente utiliza la información estrictamente necesaria de cara al usuario, aunque interiormente presente una cierta complejidad, de la que no debe tener conocimiento el usuario. El modelo conceptual se muestra en la Figura 3-1 y se lee de la siguiente forma:

> 'El problema de calcular la mejores rutas de transporte intermodal consiste en definir una serie de vuelos o medios de transporte, y encontrar las mejores rutas para ir de un origen a un destino. La definición y el cálculo de las rutas está encapsulado en la clase vfcConnectSolver, y las soluciones están encapsuladas en la clase vfcConnection, que contiene una secuencia de vuelos (o medios de transporte), y expresa unos niveles de coste, tiempo de escala, duración de los vuelos y rapidez de la conexión.'

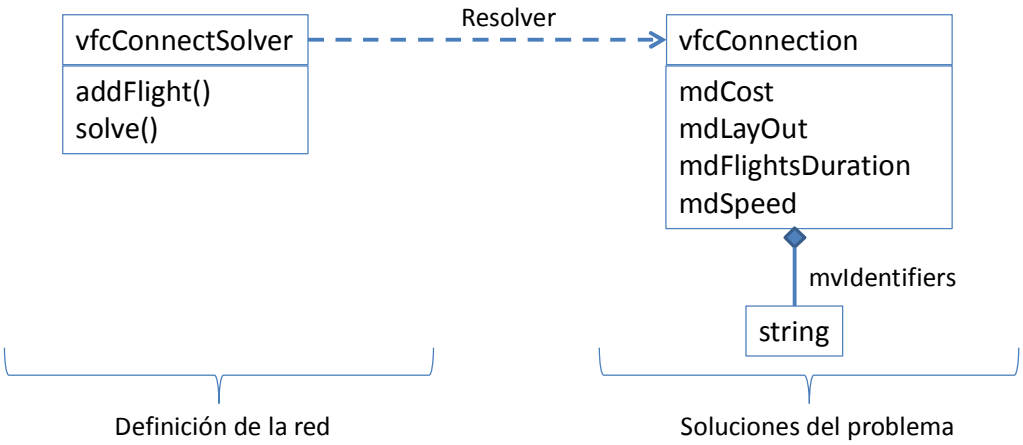

Figura 3-1. Modelo conceptual de vFlightConnect

3.2 El problema al detalle

En los sistemas de transporte intermodal en general, y en los buscadores de vuelos en particular, existe una gran combinación de medios de transporte o vuelos que conectan un origen y un destino. Cada una de estas combinaciones tiene diferentes valores objetivo. Estos valores objetivos pueden medir coste, tiempo de escala, duración de los viajes y rapidez del desplazamiento. Para definir estos cuatro valores objetivo, nos apoyamos en el ejemplo de red de transporte que muestra la Figura 3-2, donde se desea viajar desde AGP (Málaga) a VGO (Vigo). Se dispone de siete vuelos, que aterrizan en MAD (Madrid), MEL (Melilla) y BCN (Barcelona). Cada vuelo viene representado por un rectángulo azul, que muestra el identificador del vuelo (o medio de transporte) en la parte superior, y el origen y destino en la parte inferior. Sobre cada rectángulo se muestra el coste del vuelo, y bajo los rectángulos se muestran la salida y llegada de cada vuelo. Como ejemplo de lectura del diagrama, el vuelo de identificador ID505 sale de MAD en el instante 900 y llega a BCN en el instante 950, y cuesta 85 UM.

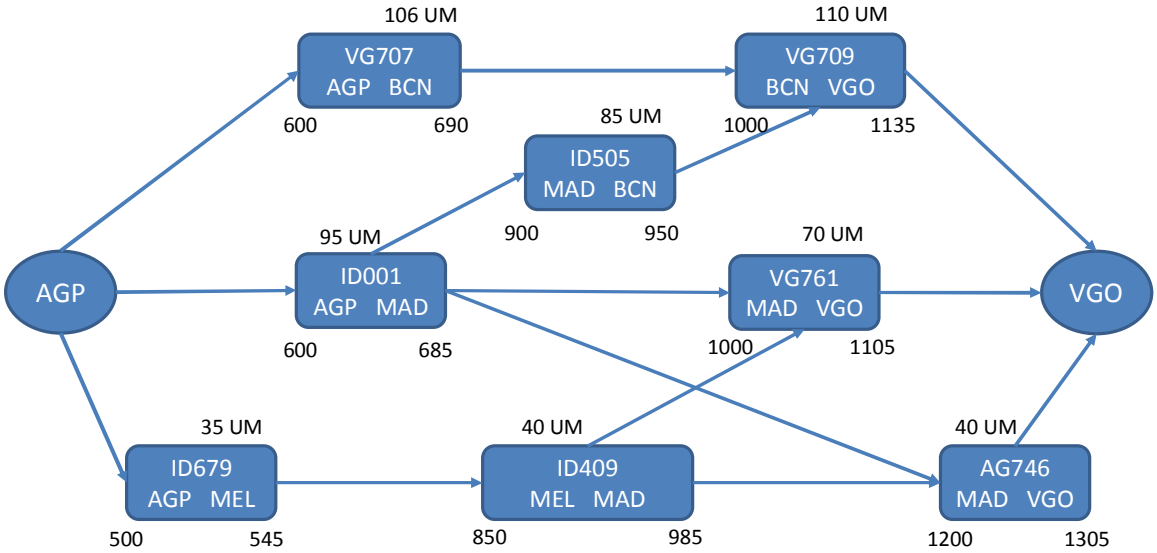

Figura 3-2. Red de vuelos

A continuación se van a definir los conceptos que representan los cuatro valores objetivos enumerados en el párrafo anterior. El coste de una conexión de vuelos es la suma de los costes de los vuelos individuales de la conexión. Por ejemplo el coste de la conexión de vuelos VG707-VG709 es 106+110=216 UM. El tiempo de escala de una conexión de vuelos es la suma de los tiempos de escala intermedios. Por ejemplo el tiempo de escala de la conexión de vuelos ID001-ID505-VG709 es (900-685)+(1.000-950)=265 UT. La duración de los vuelos de una conexión es la suma de las duraciones de los vuelos que la componen. Por ejemplo la duración de vuelos de la conexión ID679-ID409-AG746 es 45+135+105=285 UT. La rapidez de una conexión es el tiempo que transcurre desde la salida del primer vuelo en el origen a la llegada del último vuelo en el destino. Por ejemplo, en la conexión ID001-VG761, la rapidez es de 1.105-600= 505 UT.

Conocidos los vuelos de la Figura 3-2, si un viajero quiere viajar desde AGP a VGO, un buscador de vuelos usualmente podría mostrar la información de la Tabla 3-1, que el viajero debe estudiar, para seleccionar la opción que más le conviene.

Salida	Llegada	Vuelos	Coste	Escalas
600	1.135	VG707-VG709	216 UM	1 (310 UT)
600	1.115	ID001-ID505-VG709	290 UM	2 (265 UT)
600	1.105	ID001-VG761	165 UM	1 (315 UT)
600	1.305	ID001-AG746	135 UM	1 (515 UT)
500	1.105	ID679-ID409-VG761	145 UM	2 (320 UT)
500	1.305	ID679-ID409-AG746	115 UM	2 (520 UT)

Tabla 3-1. Lista de opciones de vuelo

Sería muy útil para el viajero un resumen de las mejores opciones de vuelos según diversos criterios. La solución del problema que plantea la Figura 3-2 es la que muestra la Tabla 3-2. La mejor combinación de vuelos depende del criterio seleccionado. La secuencia de vuelos más barata es ID679-ID409-AG746. La secuencia de menor tiempo de escala es VG707-VG709. La secuencia de menor tiempo de vuelo es ID001-AG746. La secuencia de vuelos más

rápida es ID001-AG761.

Secuencia	Coste	Tiempo de escala	Duración de los vuelos	Rapidez	Comentario
ID679, ID409,AG746	115 UM	520 UT	285 UT	805 UT	Menor coste
VG707, VG709	216 UM	310 UT	225 UT	535 UT	Menor tiempo de escala
ID001, AG746	135 UM	515 UT	190 UT	705 UT	Menor duración de los vuelos
ID001, AG761	165 UM	315 UT	190 UT	505 UT	Más rápida

Tabla 3-2. Solución del problema de transporte

A la vista de la información que muestra la Tabla 3-2, un cliente del buscador de vuelos tiene la información necesaria para tomar la mejor decisión. Las ideas que puede plantear el cliente dependen del tipo de viaje que desee realizar. Por ejemplo, el cliente debe valorar si está dispuesto a pagar 165-115=50 UM por reducir el tiempo de viaje (rapidez) 805-505=300 UM, o si está dispuesto a ahorrar 135-115=20 UM, reduciendo además 520-515=5 UT el tiempo de escala, o si está dispuesto a pagar 135-115=20 UM por reducir 805-705=100 UT el tiempo total de viaje, estando 285-190=95 UT menos de vuelo.

La información de la Tabla 3-2 se hace más importante a medida que aumenta la cantidad de vuelos, porque esta tabla siempre tendrá cuatro filas, mientras que la cantidad de vuelos y conexiones de vuelos, puede ser de varios centenares.

4 Manual de usuario

4.1 Introducción

vFlightConnect es una librería C++ para resolver problemas de cálculo de rutas en redes de transporte intermodal. En este capítulo se explica todo lo necesario para poder utilizar vFlightConnect. También se muestra el contenido de vFlightConnect, y se explica su uso, mediante un ejemplo sencillo.

4.2 Contenido

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

La biblioteca vFlightConnect está escrita en ANSI 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. vFlightConnect está escrita en C++ estándar, y solo el proceso de verificación de licencias hace llamadas al sistema operativo.

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

4.3 Instalación

La biblioteca vFlightConnect 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 vFlightConnect.lib para desarrollar programas C++, y el ensamblado vFlightConnectWrapper.dll, para desarrollar programas sobre la plataforma .NET de Microsoft.
\doc	Contiene el manual de usuario de vFlightConnect.

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

4.4 Licencias

vFlightConnect 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 Infozara.
- 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 vFlightConnect.

4.5 Configuración de un proyecto C++

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

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

4.6 Configuración de un proyecto .NET

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

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

  Al crear la referencia al ensamblado, se tiene acceso al namespace vFlightConnectWrapper.

4.7 Ejemplo de programación en C++

En este apartado se va a mostrar el uso de la biblioteca vFlightConnect en una aplicación C++, a través de un sencillo ejemplo, que resuelve el problema planteado en el apartado 3.2.

En la línea 1 se crea la instancia de la clase `vfcConnectSolver`, que es la responsable de calcular las mejores conexiones de vuelos. El proceso completo de cálculo se compone de dos pasos: indicar los vuelos y resolver el problema. Desde las líneas 3 a la 18 se indican los vuelos, y el problema se resuelve en las líneas 21 y 22.

Para añadir un vuelo al problema, se utiliza la función `vfcConnectSolver::addFlight()` , que recibe como parámetros el identificador único del vuelo, el origen, el destino, el instante de salida, el instante de llegada y el coste. Como ejemplo, en las líneas 3 y 4 se crea el vuelo de identificador VG707, que sale de AGP (Málaga) y llega a BCN (Barcelona), sale en el instante 600 (10:00 si la unidad de tiempo es el minuto), llega en el instante 690 (11:30), y tiene un coste de 106 UM. En el apartado 5.1 se muestra la sintaxis de la función `addFlight()`.

El problema se resuelve en la función `vfcConnectSolver::solve()`, que recibe como parámetros los aeropuertos origen y destino de la ruta de viaje, y devuelve la solución. Como ejemplo, en la línea 22 se indica que se desea encontrar las mejores conexiones de vuelos para viajar desde AGP a VGO. En el apartado 5.1 se muestra la sintaxis de la función `solve()`. Cada conexión de vuelo que calcula el solver está encapsulada en la clase `vfcConnection`. En el apartado 5.2 se muestra la declaración de la clase `vfcConnection`. La función `solve()` devuelve un vector con cuatro conexiones de vuelos. La primera conexión es la de menor coste. En la línea 27 se accede a esta conexión de menor coste, y entre las líneas 28 y 35 se

muestra por pantalla toda la información de la conexión. En concreto, entre las líneas 28 y 31 se muestran los cuatro valores objetivo de la conexión, y entre las líneas 33 y 35 se muestran los identificadores de los vuelos que forman la conexión. La segunda conexión corresponde a la de menor tiempo de escala, y el código fuente que la muestra por pantalla está escrito entre las líneas 38 y 46. La tercera conexión corresponde a la de menor tiempo de vuelo, y el código fuente que la muestra por pantalla está escrito entre las líneas 49 y 57. La cuarta conexión corresponde a la más rápida, y el código fuente que la muestra por pantalla está escrito entre las líneas 60 y 68.

```
 1  vfcConnectSolver oSolver;
 2
 3  oSolver.addFlight(   string("VG707"), string("AGP"), string("BCN"),
 4                       600, 690, 106.0);
 5  oSolver.addFlight(   string("ID001"), string("AGP"), string("MAD"),
 6                       600, 685, 95.0);
 7  oSolver.addFlight(   string("ID679"), string("AGP"), string("MEL"),
 8                       500, 545, 35.0);
 9  oSolver.addFlight(   string("ID505"), string("MAD"), string("BCN"),
10                       900, 950, 85.0);
11  oSolver.addFlight(   string("ID409"), string("MEL"), string("MAD"),
12                       850, 985, 40.0);
13  oSolver.addFlight(   string("VG709"), string("BCN"), string("VGO"),
14                       1000, 1135, 110.0);
15  oSolver.addFlight(   string("VG761"), string("MAD"), string("VGO"),
16                       1000, 1105, 70.0);
17  oSolver.addFlight    string("AG746"), string("MAD"), string("VGO"),
18                       1200, 1305, 40.0);
19
20  // Resuelve el problema
21  vector<vfcConnection> vSoluciones =
22                              oSolver.solve(string("AGP"), string("VGO"));
23
24  vector<string>::iterator itf;
25
26  // Escribe la solución por coste
27  vfcConnection cCoste = vSoluciones[0];
28  std::cout     << "\n\nSolucion por coste. Coste: " << cCoste.cost()
29                << "   Escalas: " << cCoste.layOut()
30                << "   Duracion vuelos: " << cCoste.flightsDuration()
31                << "   Rapidez: " << cCoste.speed();
32  std::cout << "\n Vuelos: ";
33  for (  itf = cCoste.identifiers().begin();
34         itf != cCoste.identifiers().end(); itf++)
35         std::cout << *itf << "  ";
36
37  // Escribe la solución por tiempos de escala
38  vfcConnection cEscala = vSoluciones[1];
39  std::cout     << "\n\nSolucion por escalas. Coste: " << cEscala.cost()
40                << "   Escalas: " << cEscala.layOut()
41                << "   Duracion vuelos: " << cEscala.flightsDuration()
42                << "   Rapidez: " << cEscala.speed();
43  std::cout << "\n Vuelos: ";
44  for (  itf = cEscala.identifiers().begin();
45         itf != cEscala.identifiers().end(); itf++)
46         std::cout << *itf << "  ";
47
48  // Escribe la solución por duración de los vuelos
49  vfcConnection cDuracion = vSoluciones[2];
```

```
50  std::cout     << "\n\nSolucion por duración de los vuelos. Coste: "
51                << cDuracion.cost() << "   Escalas: " << cDuracion.layOut()
52                << "   Duracion vuelos: " << cDuracion.flightsDuration()
53                << "   Rapidez: " << cDuracion.speed();
54  td::cout << "\n Vuelos: ";
55  for (  itf = cDuracion.identifiers().begin();
56         itf != cDuracion.identifiers().end(); itf++)
57         std::cout << *itf << "  ";
58
59  // Escribe la solución por la rapidez de los vuelos
60  vfcConnection cRapidez = vSoluciones[3];
61  std::cout     << "\n\nSolucion por rapidez de los vuelos. Coste: "
62                << cRapidez.cost() << "   Escalas: " << cRapidez.layOut()
63                << "   Duracion vuelos: " << cRapidez.flightsDuration()
64                << "   Rapidez: " << cRapidez.speed();
65  std::cout << "\n Vuelos: ";
66  for (  itf = cRapidez.identifiers().begin();
67         itf != cRapidez.identifiers().end(); itf++)
68         std::cout << *itf << "  ";
```

4.8 Ejemplo de programación en .NET

En este apartado se va a mostrar el uso de la biblioteca vFlightConnect en una aplicación escrita para la plataforma .NET, a través de un sencillo ejemplo escrito en C#, que resuelve el problema planteado en el apartado 3.2.

Una vez creada la referencia al ensamblado vFlightConnectWrapper.dll, se dispone de acceso al espacio de nombres vFlightConnectWrapper. En este espacio de nombre existen tres clases públicas, que se utilizan para plantear un problema de conexiones de medios de transporte y acceder a la solución.

En las líneas 1 y 2 se crea la instancia de la clase `vFlightConnectCpp`, que es la responsable de calcular las mejores conexiones de vuelos. El proceso completo de cálculo se compone de dos pasos: indicar los vuelos y resolver el problema. Desde las líneas 6 a la 13 se indican los vuelos, y el problema se resuelve en las líneas 16 y 17.

Para añadir un vuelo al problema, se utiliza la función `vFlightConnectCpp::addFlight()` , que recibe como parámetros el identificador único del vuelo, el origen, el destino, el instante de salida, el instante de llegada y el coste. Como ejemplo, en la línea 6 se crea el vuelo de identificador VG707, que sale de AGP (Málaga) y llega a BCN (Barcelona), sale en el instante 600 (10:00 si la unidad de tiempo es el minuto), llega en el instante 690 (11:30), y tiene un coste de 106 UM. En el apartado 6.1 se muestra la sintaxis de la función `addFlight()`.

El problema se resuelve en la función `vFlightConnectCpp::solve()`, que recibe como parámetros los aeropuertos origen y destino de la ruta de viaje, y devuelve la solución. Como ejemplo, en la línea 17 se indica que se desea encontrar las mejores conexiones de vuelos para viajar desde AGP a VGO. En el apartado 6.1 se muestra la sintaxis de la función `solve()`. Cada conexión de vuelo que calcula el solver está encapsulada en la clase `spaConnection`. En el apartado 6.2 se muestra la declaración de la clase `spaConnection`. La función `solve()` devuelve un vector con cuatro conexiones de vuelos. La primera conexión es la de menor coste. En la línea 20 se accede a esta conexión de menor coste, y entre las líneas 21 y 26 se muestra por pantalla toda la información de la conexión. En concreto, entre las líneas 21 y 23 se muestran los cuatro valores objetivo de la conexión, y entre las líneas 25 y 26 se muestran los identificadores de los vuelos que forman la conexión. La segunda conexión corresponde a la de menor tiempo de escala, y el código fuente que la muestra por pantalla está escrito entre las líneas 29 y 35. La tercera conexión corresponde a la de menor tiempo de vuelo, y el

código fuente que la muestra por pantalla está escrito entre las líneas 38 y 44. La cuarta conexión corresponde a la más rápida, y el código fuente que la muestra por pantalla está escrito entre las líneas 47 y 53.

```
1  // Crea el solver
2  vFlightConnectWrapper.vFlightConnectCpp oSolver =
3                              new vFlightConnectWrapper.vFlightConnectCpp();
4
5  // Crea los vuelos
6  oSolver.addFlight( "VG707", "AGP", "BCN", 600, 600 + 90, 106.0 );
7  oSolver.addFlight( "ID001", "AGP", "MAD", 600, 600 + 85, 95.0);
8  oSolver.addFlight( "ID679", "AGP", "MEL", 500, 500 + 45, 35.0);
9  oSolver.addFlight( "ID505", "MAD", "BCN", 900, 900 + 50, 85.0);
10 oSolver.addFlight( "ID409", "MEL", "MAD", 850, 850 + 135, 40.0);
11 oSolver.addFlight( "VG709", "BCN", "VGO", 1000, 1000 + 135, 110.0);
12 oSolver.addFlight( "VG761", "MAD", "VGO", 1000, 1000 + 105, 70.0);
13 oSolver.addFlight( "AG746", "MAD", "VGO", 1200, 1200 + 105, 40.0);
14
15 // Resuelve el problema
16 List<vFlightConnectWrapper.spaConnection> lstConnections = null;
17 oSolver.solve( "AGP", "VGO", ref lstConnections);
18
19 // Escribe la solución por coste
20 vFlightConnectWrapper.spaConnection cCoste = lstConnections[0];
21 Console.WriteLine(  "Solucion por coste. Coste: " + cCoste.cost() + "
22 Escalas: " + cCoste.layOut() + "   Duracion vuelos: " +
23 cCoste.flightsDuration() + "   Rapidez: " + cCoste.speed() );
24 Console.WriteLine( " Vuelos: " );
25 foreach( string strIdentifier in cCoste.identifiers() )
26        Console.WriteLine( strIdentifier );
27
28 // Escribe la solución por tiempos de escala
29 vFlightConnectWrapper.spaConnection cEscala = lstConnections[1];
30 Console.WriteLine("Solucion por escalas. Coste: " + cEscala.cost() + "
31 Escalas: " + cEscala.layOut() + "   Duracion vuelos: " +
32 cEscala.flightsDuration() + "   Rapidez: " + cEscala.speed() );
33 Console.WriteLine(" Vuelos: ");
34 foreach (string strIdentifier in cEscala.identifiers())
35        Console.WriteLine(strIdentifier);
36
37 // Escribe la solución por duración de los vuelos
38 vFlightConnectWrapper.spaConnection cDuracion = lstConnections[2];
39 Console.WriteLine("Solucion por duracion de los vuelos. Coste: " +
40 cDuracion.cost() + "   Escalas: " + cDuracion.layOut() + "  Duracion vuelos:
41 " + cDuracion.flightsDuration() + "   Rapidez: " + cDuracion.speed() );
42 Console.WriteLine(" Vuelos: ");
43 foreach (string strIdentifier in cDuracion.identifiers())
44        Console.WriteLine(strIdentifier);
45
46 // Escribe la solución por la rapidez de los vuelos
47 vFlightConnectWrapper.spaConnection cRapidez = lstConnections[3];
48 Console.WriteLine("Solucion por rapidez del viaje. Coste: " +
49 cRapidez.cost() + "   Escalas: " + cRapidez.layOut() + "  Duracion vuelos: "
50 + cRapidez.flightsDuration() + "   Rapidez: " + cRapidez.speed() );
51 Console.WriteLine(" Vuelos: ");
52 foreach (string strIdentifier in cRapidez.identifiers())
53        Console.WriteLine(strIdentifier);
```

5 Manual de referencia C++

Este capítulo hace referencia a todas las clases públicas de la biblioteca C++ vFlightConnect.

5.1 vfcConnectSolver

Categoría Clase de desarrollo.

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

Esta clase dispone únicamente de una función que permite añadir los vuelos (o medios de transporte) del problema, y otra función para obtener las mejores secuencias de vuelos.

Fichero include

```
class vfcConnectSolver
{

public:
	vfcConnectSolver();
	virtual ~vfcConnectSolver();

	void addFlight(string& strIdentifier,
			string& strOrigin, string& strDestination,
			int nDeparture, int nArrival,
			double dCost);

	vector<vfcConnection>& solve( string& strOrigin,
					string& strDestination );
};
```

Constructores

```
vfcConnectSolver (void);
```

El constructor no recibe ningún parámetro.

Funciones miembro públicas

```
void addFlight(string& strIdentifier,
		string& strOrigin, string& strDestination,
		int nDeparture, int nArrival, double dCost)
```

Añade un vuelo o medio de transporte, que viene definido por un identificador único, el origen y el destino, las horas de salida y llegada, y el coste.

```
vector<vfcConnection>& solve( string& strOrigin,
				string& strDestination );
```

Devuelve las cuatro mejores conexiones de vuelos, cada una bajo un criterio diferente. La primera conexión es la de menor coste, la segunda conexión es

la de menor tiempo de escala, la tercera la de menor tiempo de vuelo, y la cuarta es la conexión más rápida.

Cada conexión de vuelos está encapsulada en la clase `vfcConnection`.

5.2 vfcConnection

Categoría Clase de desarrollo.

Descripción `vfcConnection` representa una conexión de vuelos.

Fichero include

```
class vfcConnection
{
public:
	vfcConnection();

	vector<string>& identifiers(void);

	double cost(void) const;

	double layOut(void) const;

	double flightsDuration(void) const;

	double speed(void) const;
};
```

Constructores `vfcConnection()`

No recibe parámetros.

Funciones miembro públicas `vector<string>& identifiers(void)`

Devuelve un vector con los identificadores de los vuelos o medios de transporte que forman la conexión. El identificador de un vuelo es un dato de tipo `string`.

`double cost(void) const`

Devuelve el coste total de los vuelos de la conexión de vuelos.

`double layOut(void) const`

Devuelve el tiempo de escala total de la conexión de vuelos.

`double flightsDuration(void) const`

Devuelve la duración total de los vuelos de la conexión de vuelos.

```
double speed(void) const
```

Devuelve la rapidez de la conexión de vuelos.

6 Manual de referencia .NET

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

6.1 vFlightConnectCpp

Categoría Clase de desarrollo.

Descripción vFlightConnectCpp es la representación del problema de encontrar las mejores rutas de vuelos y/o otros medios de transporte.

Esta clase dispone únicamente de una función que permite añadir los vuelos (o medios de transporte) del problema, y otra función para obtener las mejores secuencias de vuelos.

Es una clase escrita en lenguaje C++ administrado, y es accesible desde cualquier lenguaje de desarrollo administrado de la plataforma .NET.

Fichero include

```
public ref class vFlightConnectCpp
{

public:

	vFlightConnectCpp();

	virtual ~vFlightConnectCpp(void);

	spaVertex^ addFlight(String^ strName, String^ strOrigin,
	                     String^ strDestination, int nDeparture,
	                     int nArrival, double dCost);

	void solve(String^ strFrom, String^ strTo,
	           List<spaConnection^>^% lstConnections);
};
```

Constructores vFlightConnectCpp (void);

El constructor no recibe ningún parámetro.

Funciones miembro públicas

```
spaVertex^ addFlight(String^ strName, String^ strOrigin,
                     String^ strDestination, int nDeparture,
                     int nArrival, double dCost);
```

Añade un vuelo o medio de transporte, que viene definido por un identificador único, el origen y el destino, las horas de salida y llegada, y el coste.

```
void solve(String^ strFrom, String^ strTo,
           List<spaConnection^>^% lstConnections);
```

Devuelve las cuatro mejores conexiones de vuelos, cada una bajo un criterio diferente. La primera conexión es la de menor coste, la segunda conexión es la de menor tiempo de escala, la tercera la de menor tiempo de vuelo, y la cuarta es la conexión más rápida.

Cada conexión de vuelos está encapsulada en la clase `spaConnection`.

6.2 spaConnection

Categoría Clase de desarrollo.

Descripción `spaConnection` representa una conexión de vuelos.

Fichero include

```
public ref class spaConnection
{
public:

        spaConnection(double dCost, double dLayOut,
                      double dFlightsDuration, double dSpeed );

        List<String^>^ identifiers(void);

        double cost(void);

        double layOut(void)

        double flightsDuration(void)

        double speed(void)
};
```

Constructores

`spaConnection()`

No recibe parámetros.

Funciones miembro públicas

`List<String^>^ identifiers();`

Devuelve un vector con los identificadores de los vuelos o medios de transporte que forman la conexión. El identificador de un vuelo es un dato de tipo `String`.

`double cost(void)`

Devuelve el coste total de los vuelos de la conexión de vuelos.

```
double layOut(void)
```

Devuelve el tiempo de escala total de la conexión de vuelos.

```
double flightsDuration(void)
```

Devuelve la duración total de los vuelos de la conexión de vuelos.

```
double speed(void)
```

Devuelve la rapidez de la conexión de vuelos.