# EL PROBLEMA DEL ENRUTAMIENTO DE VEHÍCULOS. PROPUESTAS PARA LA BUSQUEDA DEL CAMINO MAS CORTO. APLICACIÓN AL ENTORNO DOCENTE Y PYMES

**JUAN JESÚS BERNAL GARCÍA**
*Juanjesus.bernal@upct.es*
*Universidad Politécnica de Cartagena (UPCT) / Métodos Cuantitativos e Informáticos*
*C/Real3. 30201 Cartagena*

**ELOY HONTORIA HERNÁNDEZ**
*eloy.hontoria@upct.es*
*Universidad Politécnica de Cartagena / Organización de Empresas.*
*Escuela Técnica Superior de Ingenieros Industriales. 30202 Cartagena*

**DARKO ALEKSOVSKI**
*darko.aleksovski@ijs.si*
*Jožef Stefan Institute / Department of Knowledge Technologies.*
*Ljubljana (Eslovenia*

RESUMEN: El enrutamiento de vehículos o VRP (Vehicle Routing Problem) es un problema típicamente planteado para la optimización de las rutas de transporte y vehículos asociados a ellas. Para su planteamiento y búsqueda de soluciones óptimas se recurre en investigación operativa a modelos matemáticos como *Branch and Bound* o programación dinámica entre otros. Estos modelos proporcionan soluciones exactas pero presentan una gran complejidad para su uso práctico en el entorno docente y de las PYMES. El objetivo del presente trabajo es la presentación de la utilidad informática Solver, que incorporada en Excel y debidamente programada con VBA (*Visual Basic for Applications*) permite la obtención de soluciones de manera sencilla para su uso docente y para la pequeña y mediana empresa.

Como aplicación práctica para mostrar la metodología elaborada y su operatividad, se ha recurrido a un caso práctico consistente en la determinación de la ruta más corta entre ciudades españolas. Para contrastar la validez de los resultados obtenidos, se han comparado éstos con los proporcionados con un algoritmo de búsqueda exhaustiva mediante fuerza bruta (*BSF*) diseñado "ad hoc". Este algoritmo que explota la aplicación de distancias de rutas de *Google Maps*, proporciona mejoras con respecto al modelo anterior basado en Solver [1], suministrando información más actualizada y completa sobre el problema y, pudiendo aportar en un futuro una visión de costes al problema de enrutamiento de vehículos.
*Palabras clave:* Excel, Solver, Google Maps, Optimización de Rutas, Algoritmo de Búsqueda por Fuerza Bruta.

ABSTRACT: The Vehicle Routing Problem (VRP) is a common problem presented to optimize the transport routes and vehicles involved. For its approach and search of optimal solutions in operational research, mathematical models like Branch and Bound

or dynamic programming among other options are used. These tools provide exact solutions, but the disadvantage is it´s difficult practical application for teaching or SME´s. The purpose of this researching work is to present Solver, a deployed application in Excel which properly programmed in BVA (Visual Basic for Applications) allows us to obtain easily solutions for teaching purpose or to be applied in SME´s.

As a practical application to show the designed methodology and its operability, a practical case consisting in the calculation of the shortest path between several Spanish cities will be presented. To test the validity of the results obtained through Solver´s application, they will be contrasted with others obtained through an algorithm based in the Brute Force Search (BFS) specifically designed for this purpose. By using this algorithm, which uses Google Maps´s application for route distances, some improvements are reached compared with the previous model based in Solver, providing more current and complete information about the problem and a future cost perspective to the Vehicle Routing Problem.

*Keywords:* Excel, Solver, Google Maps, Routes Optimization, Brute Force Search Algorithm.

## 1. **Introducción**

El problema del enrutamiento de vehículos o VRP (Vehicle Routing Problem) es un nombre genérico dado a toda aquella clase de problemas que implica la visita a "clientes" mediante vehículos [2]. El VRP ha sido profundamente estudiado por diversos autores como [3], [4], [5] y [6] y en su situación inicial se parte de un conjunto de clientes con localizaciones y requerimientos conocidos, que tienen que ser servidos desde uno o varios depósitos por una flota de vehículos. El problema del diseño de las rutas para esos vehículos que satisfagan los requerimientos de los clientes y al menor coste de distribución posible, es lo que se conoce con el nombre de problema del enrutamiento de vehículos.

Las variantes del VRP son muy diversas y vienen dadas por los requerimientos y restricciones operativas impuestas por el diseño de rutas en situaciones prácticas [7]. Entre estas variantes se puede destacar que los clientes no sean conocidos a priori en el caso del DVRP (Dynamic Vehicle Routing Problem) estudiado por [8] o que las entregas solo puedan ser realizadas en una determinada franja horaria (Time Window), tratándose del VRPTW [9] o que la capacidad del vehículo sea una restricción en el Capacitated Vehicle Routing Problem (CVRP) analizado por [7] y una infinidad de variantes (entrega y recogida simultánea al cliente, demandas deterministas o aleatorias, etc.). En determinadas ocasiones se ha recurrido a técnicas de simulación para la composición de una flota de vehículos heterogénea (Heterogeneous fleet VRP, HVRP), con demanda determinista atendiendo a criterios económicos y restricciones de tiempo máximo para realizar el reparto diario [10].

VRP es en definitiva una extensión del problema del Viajante de Comercio (Traveling Salesman Problem, TSP) donde el vendedor debe visitar un conjunto de clientes o

ciudades una sola vez, para después volver a la ruta de partida, construyéndose un grafo donde figuran los clientes (vértices) y las rutas posibles entre un cliente y otro (aristas). El TSP empezó a ser contemplado mediante técnicas informáticas con el trabajo de [11] en el que se diseñó un algoritmo para el cálculo óptimo de una ruta a través de 49 ciudades. El VRP según [12] es considerablemente más difícil de resolver que un TSP para un problema del mismo tamaño. Un problema de TSP implicando cientos o incluso miles de vértices puede ser resuelto mediante algoritmos avanzados de branch-and-cut-and.-price [12] y en contraste [14] creó uno de los más sofisticados algoritmos exactos donde resolvía con una tasa de éxito variable un máximo de 100 clientes.

Otro capítulo a analizar es el método empleado para la resolución del problema planteado. Las técnicas de optimización se enfrentan generalmente a objetivos contrapuestos como son encontrar la solución óptima y además hacerlo en el menor tiempo posible o realizarlo mediante un modelo matemático que pueda imitar fielmente el experimento o conseguirlo a un coste razonable, etc. En determinadas ocasiones se valora más la obtención de soluciones aceptables aunque no sean las óptimas si éstas han sido obtenidas más rápidamente. En este aspecto se debe destacar tres propuestas de resolución: Métodos exactos, métodos basados en el TSP y Heurísticos. Los heurísticos son métodos aproximados cuya versión más compleja se alcanza con los Metaheurísticos. Referido a la primera opción de métodos exactos y para la optimización de las rutas de transporte y vehículos asociados a ellas, se recurre en investigación operativa a modelos matemáticos o a programación dinámica entre otras opciones. Estas herramientas proporcionan soluciones exactas, pero el diseño de un modelo matemático que refleje la realidad con precisión es complicado y además requiere mucho tiempo para la búsqueda de soluciones.

Para la resolución de estos modelos matemáticos en docencia no especializada, se suele recurrir a programas específicos cómo *WinQSB* [15], *LINDO Systems* [16], etc., pero en este trabajo se ha preferido la utilidad informática Solver, que incorporada en Excel y a pesar de su metodología limitada, permite determinar la ruta más corta de forma sencilla y por tanto muy útil para su utilización.
En el presente trabajo se pretende contrastar la validez de los resultados aportados por *Solver* para el cálculo del camino óptimo, mediante su comparación con los obtenidos con un algoritmo diseñado para este propósito, que al utilizarlo conjuntamente con *Google Maps* permite además facilitar la toma de distancias, y aportar información adicional a la solución obtenida con la hoja de cálculo.

## 2. Camino más corto entre varios puntos. Caso práctico con de la herramienta Solver para la obtención de resultados.

El TSP nombrado anteriormente, es uno de los problemas más conocidos de la optimización combinatoria donde la dificultad de buscar soluciones crece de manera exponencial al aumentar el número de ciudades a visitar. A modo de ejemplo y para un planteamiento de 12 ciudades, el viajante deberá decidirse entre las 479.001.600 combinaciones posibles.

El problema del camino más corto planteado en este trabajo es una variación del TSP,

consistente en que en este caso se debe buscar el camino más corto entre un origen y un destino, sin la obligación de visitar o pasar por todos los puntos (ciudades) posibles. En la resolución mediante *Solver* el planteamiento del problema es sencillo y las soluciones son encontradas sin un elevado coste computacional cuando el número de ciudades es relativamente reducido. Estas ventajas en cuanto a facilidad, economía y rapidez hacen de este complemento de Excel una herramienta de gran utilidad para introducir tanto a los alumnos, cómo a los técnicos de las pequeñas empresas en el campo de métodos cuantitativos, al mismo tiempo que se familiarizan con el manejo de hojas de cálculo. Sin embargo, los resultados obtenidos deben ser contrastados convenientemente.

Para la citada validación se recurre al planteamiento de un caso práctico consistente en encontrar el camino más corto entre dos ciudades, una origen y otra destino, sabiendo que existen diversas rutas alternativas dependiendo de las 5 posibles ciudades intermedias por las que se pase. Por tanto se trata de un problema con cinco vértices y 2 aristas, incluidos el origen y el destino, con 120 combinaciones posibles (*Figura 1*), del que se conoce la distancia entre ellos; concretamente se trata de considerar las ciudades españolas: Cartagena (CT-Origen), Albacete (AB), Cuenca (CU), Toledo (TO), Segovia (SG), Soria (SO) y Burgos (BU-Destino).

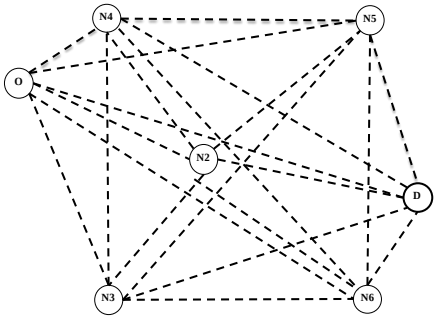



Figura 1: Siete nodos y sus conexiones posible

Cómo información de partida se dispone de las distancias entre las distintas ciudades, que pueden ser introducidas manualmente, tecleándolos en la celda correspondiente, o tomarlas directamente de una tabla de distancias entre ciudades importada a Excel desde una página Web (*Tabla 1*).

Tabla 1.Matriz de distancias entre las ciudades del problema

| | CT | AB | CU | TO | SG | SO | BU |
|---|---|---|---|---|---|---|---|
| **CT** | | 192 | 352 | 438 | 541 | 649 | 677 |
| **AB** | 192 | | 164 | 249 | 353 | 461 | 489 |
| **CU** | 352 | 164 | | 179 | 260 | 272 | 398 |
| **TO** | 438 | 249 | 179 | | 158 | 299 | 318 |
| **SG** | 541 | 353 | 260 | 158 | | 191 | 200 |
| **SO** | 649 | 461 | 272 | 299 | 191 | | 143 |
| **BU** | 677 | 489 | 398 | 318 | 200 | 143 | |

La situación real suele implicar el planteamiento inicial de una serie de rutas o conexiones convenientes, de entre todas las posibles (*Figura 1*). En este caso, el grafo con las posibles rutas, con líneas de trazo discontinuo, quedaría de la forma que se muestra en la *Figura 2*, tanto con aristas, de forma general, como con las ciudades del caso práctico. La programación mediante macros en *VBA* [17] realizada para dibujar las conexiones se adjunta en el *Anexo B.*

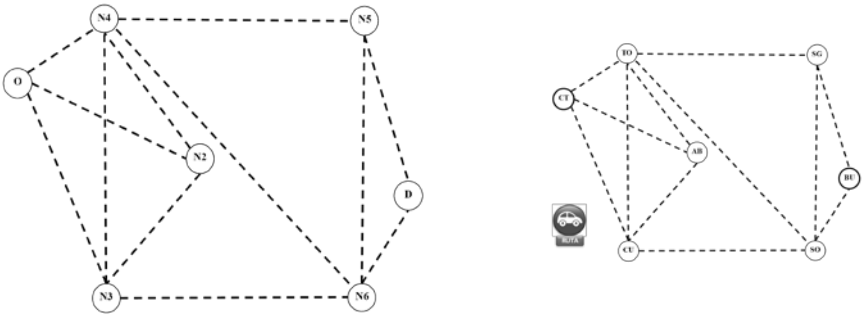



Figura 2:Conexiones deseadas entre nudos y entre ciudades

La forma práctica de selección de caminos entre ciudades en el modelo elaborado en Excel, es en la tabla de doble entrada, sustituir el valor "0" por un "1" en la celda intersección de ambas ciudades, lo que establece la existencia de conexión deseada entre ellas, siendo necesario hacerlo solamente en la parte superior de la misma, ya que al tratarse de una matriz transpuesta, la inferior se rellena automáticamente (*Tabla 2*). Los caminos posibles que sean señalados determinarán la matriz de distancias a utilizar entre las ciudades conectadas, haciéndose cero el resto (*Tabla 3*). A modo de ejemplo, al teclear un 1 en el elemento intersección CT-AB, aparece la distancia de 192 km., que es la existente entre ambas ciudades.

Tabla 2. Matriz (transpuesta) de "unos y fondo gris" para indicar las conexiones posibles

| | 1 | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| | **CT** | **AB** | **CU** | **TO** | **SG** | **SO** | **BU** |
| **CT** | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| **AB** | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| **CU** | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| **TO** | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| **SG** | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| **SO** | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| **BU** | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

Seguidamente al invocar la macro, o pulsar el icono elaborado para ello con forma de coche (*Figura 2*), se ejecuta el Solver programado con la ecuación a minimizar y distintas restricciones (*Anexo A)* [6], obteniéndose la solución de ruta óptima encontrada en forma de tabla (*Tabla 4*). La solución encontrada incluye las ciudades intermedias de paso, en este caso: N3 (CU) y N6 (SO), y la distancia total de dicha ruta óptima: 352 (CT-CU) + 272 (CU-SO) + 143 (SO-BU) = 767 kms.

Tabla 3.Matriz de distancias entre ciudades conectadas

| | CT | AB | CU | TO | SG | SO | BU |
|---|---|---|---|---|---|---|---|
| **CT** | 0 | 192 | 352 | 438 | 0 | 0 | 0 |
| **AB** | 192 | 0 | 164 | 249 | 0 | 0 | 0 |
| **CU** | 352 | 164 | 0 | 179 | 0 | 272 | 0 |
| **TO** | 438 | 249 | 179 | 0 | 158 | 299 | 0 |
| **SG** | 0 | 0 | 0 | 158 | 0 | 191 | 200 |
| **SO** | 0 | 0 | 272 | 299 | 191 | 0 | 143 |
| **BU** | 0 | 0 | 0 | 0 | 200 | 143 | 0 |

Tabla 4.Matriz con las distancias de la ruta óptima

| | CT | AB | CU | TO | SG | SO | BU |
|---|---|---|---|---|---|---|---|
| **CT** | 0 | 0 | **1** | 0 | 0 | 0 | 0 |
| **AB** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **CU** | 0 | 0 | 0 | 0 | 0 | **1** | 0 |
| **TO** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **SG** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **SO** | 0 | 0 | 0 | 0 | 0 | 0 | **1** |
| **BU** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Como aportación a otros modelos análogos existentes en Excel, se ha programado también la solución en forma gráfica, ya que de manera automática se muestra la ruta más corta mediantes conexiones de trazo continuo (en color verde en el programa de hoja de cálculo) (*Figura 5*). Macro en *Anexo B.*

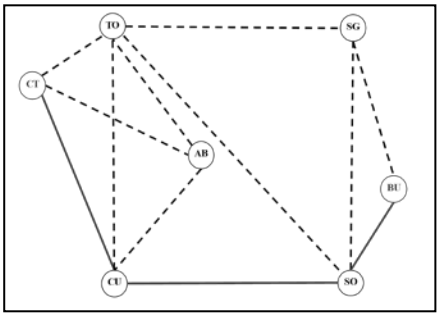



Figura 5: Solución gráfica y con iconos de la ruta más corta: CT-BU.

Aunque se ha hablado inicialmente de vértices para generalizar el uso del modelo, como el ejemplo tratado hace referencia a siete ciudades concretas, se ha pensado representar sobre una imagen con mapa de España, o un mapa de Google, dichas ciudades así como la solución, resultando de esta forma más evidente la lógica de la misma (*Figura 6*). Se debe comentar en este punto que Excel no cuenta en las versiones actuales con la inclusión de *Microsoft Maps*, que permitía incluir directamente mapas en la hoja de cálculo e insertar datos en ellos.

Figura 6: Solución del problema sobre mapa con siete ciudades

Una vez encontrada la solución óptima con Solver, se estimó conveniente someter al modelo a diversas pruebas para comprobar su correcto funcionamiento. En primer lugar se quiso analizar su comportamiento ante variaciones en las conexiones existentes entre ciudades, y en segundo, comprobar la reversibilidad de la solución obtenida. Para la primera prueba, y dado que el establecimiento de estas conexiones son fundamentales a la hora de plantear y resolver el problema, se ha probado a marcar como posible la conexión que une directamente el origen "O" (CT) con el destino "D" (BU), marcando con 1 el valor correspondiente en la matriz de conexiones (*Tabla 5*). Se comprueba que efectivamente la nueva solución óptima encontrada es ésta, ya que implicaría menos kilómetros que la anterior, con varias ciudades, de 767 km., y por tanto supondría un ahorro de 90 kms. (*Figura 7*).

Tabla 5. Conexión directa entre origen y destino

| | O | N2 | N3 | N4 | N5 | N6 | D |
|---|---|---|---|---|---|---|---|
| **O** | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| **N2** | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| **N3** | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| **N4** | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| **N5** | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| **N6** | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| **D** | 0 | 1 | 0 | 0 | 1 | 1 | 0 |

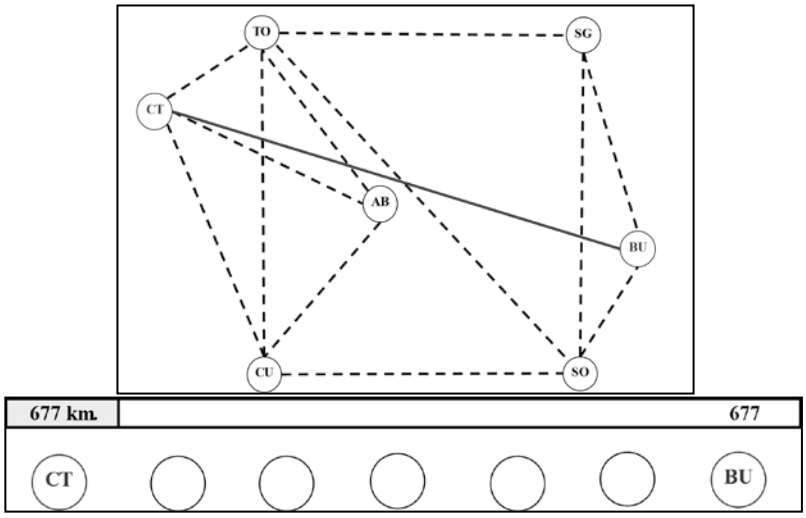



Figura 7: Nueva solución tras añadir una nueva conexión CT-BU.

Para la segunda comprobación sobre si el modelo responde correctamente a analizar el camino en sentido inverso se programó la posibilidad de permutar origen y destino. Para ello se cambió simplemente el "1" por "0" sobre la O de origen (CT), y se comprueba que la solución implica los mismos nodos N6 (SO) y N3 (CU), y la misma distancia suma (767 Kms.) (*Tabla 6)*, lo cual demuestra nuevamente el correcto funcionamiento del modelo planteado y la herramienta Solver.

Tabla 6: Marcar rutas en sentido inverso (permutar origen y destino). Ruta BU-CT.

| | 0 | | | | | | 1 |
|---|---|---|---|---|---|---|---|
| | **CT** | **AB** | **CU** | **TO** | **SG** | **SO** | **BU** |
| **CT** | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| **AB** | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| **CU** | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| **TO** | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| **SG** | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| **SO** | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| **BU** | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

## 3. Propuesta de solución alternativa. Algoritmo de Búsqueda mediante Fuerza Bruta *y Google Map.*

Como se ha podido apreciar en el ejemplo anterior, Solver es una herramienta válida para conseguir soluciones de manera rápida y económica. Sin embargo, el administrador del sistema puede requerir de soluciones más restrictivas en cuanto a la exactitud,

fundamentalmente para distancias pequeñas como puede ocurrir en rutas urbanas, y/o que la actualización de las mismas se efectué de manera automática sin requerir introducir nuevos valores en la hoja Excel. Por todo ello, se planteó la posibilidad de buscar la solución óptima por otros métodos alternativos al Solver, que a su vez sirviesen de contrate de la solución obtenida anteriormente para el caso práctico planteado.

El método elegido está basado en el Algoritmo de Búsqueda mediante Fuerza Bruta (en inglés *Brute Search Force, BSF*), aplicado al problema del camino más corto. La búsqueda por Fuerza Bruta también es conocida como búsqueda exhaustiva y consistiría en generar todas las soluciones factibles, calcular para todas ellas el coste asociado (distancia, tiempo, unidades monetarias) y posteriormente elegir el menor de estos costes.

De una manera más formal sería equivalente a definir una función de coste $f$: $S \rightarrow R$, siendo $S$ el espacio de soluciones factibles y pudiéndose calcular $f(s) \quad \forall s \in S$ y dar como solución final el óptimo, que para el problema que se intenta resolver se correspondería con el camino más corto.

La generación de una secuencia con todas las soluciones factibles no es labor fácil, pero para el ejemplo considerado con un número reducido de nodos, es posible encontrar todo el espacio de soluciones mediante el modelo diseñado y además elegir el óptimo dentro de él.

Cuando el número de ciudades intermedias aumenta de manera considerable, este procedimiento no es viable debido a la magnitud del espacio de soluciones *S*, desvaneciéndose el objetivo de encontrar el óptimo. En estos casos adquiere mayor importancia la obtención de buenas soluciones aunque no sean las óptimas, recurriéndose al empleo de heurísticos o metaheurísticos en su versión más sofisticada.

Abordaremos el mismo ejemplo anterior de siete ciudades combinando el uso del servidor de aplicaciones de mapas en la web *Google Maps* y *BSF*. La ventaja es que la toma de distancias entre ciudades se hace de manera automática y exacta, al ser obtenida mediante la posibilidad existente en *Google Maps* de disponer de la matriz de distancias. La aplicación Web *Google Maps* permite al usuario mediante su *API*[1] generar una matriz de distancias que será la base para la resolución del problema del camino más corto entre dos ciudades. Para ello todas las direcciones o coordenadas de las ciudades son introducidas en esta *API* de *Google Maps* y la mencionada aplicación devuelve unos resultados correspondientes a las distancias entre cada una de los nodos pero en un formato no demasiado operativo para el ámbito docente y profesional. Para obtener las distancias en formato de hoja de cálculo Excel, se precisa de un "script"[2] desarrollado "ad hoc" por los autores (*Figura 8*).

[1] Application Programming Interface (Interfaz de Programación de Aplicaciones)

[2] Programa que proporciona un formato estándar.

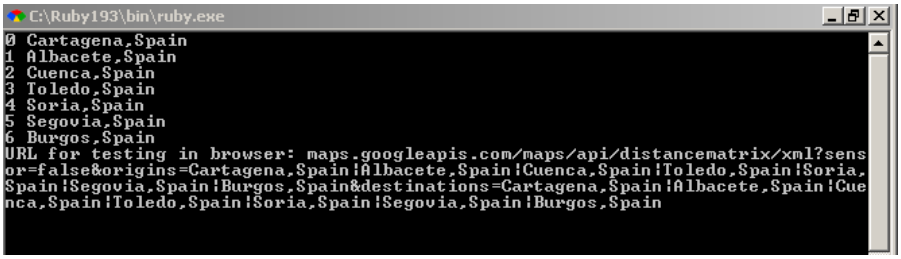



Figura 8: Script para la obtención de las matrices de tiempos y distancias en formato Excel

El resultado devuelto mediante la aplicación de este script a los valores proporcionados por *Google Maps* es una Matriz de distancias (Distance Matrix) en formato Excel con las distancias en Kms. entre cada una de las 7 ciudades tal y como muestra la *Tabla 7*. Se debe aclarar que el resultado puede depender de cómo sean medidas las distancias entre ciudades, siendo el centro de la ciudad el punto de referencia desde el cual toma medidas Google Maps.

Tabla 7: Matriz de distancias (Google Maps)

| **DISTANCE MATRIX: (In Kilometers)** | | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Cartagena** | **Albacete** | **Cuenca** | **Toledo** | **Soria** | **Segovia** | **Burgos** |
| **Cartagena** | 0 | 192 | 352 | 438 | 649 | 541 | 677 |
| **Albacete** | 190 | 0 | 164 | 249 | 461 | 353 | 489 |
| **Cuenca** | 351 | 163 | 0 | 179 | 272 | 260 | 398 |
| **Toledo** | 439 | 250 | 182 | 0 | 299 | 158 | 318 |
| **Soria** | 651 | 462 | 274 | 300 | 0 | 191 | 143 |
| **Segovia** | 541 | 353 | 260 | 165 | 190 | 0 | 200 |
| **Burgos** | 677 | 488 | 400 | 321 | 144 | 200 | 0 |

Como se puede apreciar en la anterior tabla, la matriz devuelta no es simétrica, lo cual tiene su lógica debido a que los arcos en el recorrido de ida y vuelta no utilizan en todo momento el mismo camino. Aunque para el problema en cuestión, estas diferentes distancias en ambos sentidos no tiene demasiada importancia, sí que es un factor a considerar en la logística urbana, donde algunas calles son utilizadas plenamente en un sentido y usándose otras en sentido contrario. La asimetría en la matriz de distancias afecta a los cálculos de distancias, tiempos y costes de las empresas de transporte exclusivamente urbano. Para la depuración de la matriz de distancias convirtiéndola en simétrica para una mejor operatividad, se ha optado por el criterio de tomar la menor de las dos distancias como se aprecia en la *Tabla 8*.

Tabla 8: Matriz de distancias simétrica (Google Maps)

| DISTANCE MATRIX: (In Kilometers) | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Cartagena | Albacete | Cuenca | Toledo | Soria | Segovia | Burgos |
| Cartagena | 0 | 190 | 351 | 438 | 649 | 541 | 677 |
| Albacete | 190 | 0 | 163 | 249 | 461 | 353 | 488 |
| Cuenca | 351 | 163 | 0 | 179 | 272 | 260 | 398 |
| Toledo | 438 | 249 | 179 | 0 | 299 | 165 | 321 |
| Soria | 649 | 461 | 272 | 299 | 0 | 190 | 143 |
| Segovia | 541 | 353 | 260 | 165 | 190 | 0 | 200 |
| Burgos | 677 | 488 | 398 | 321 | 143 | 200 | 0 |

En el ejemplo a resolver y al tratarse de un número reducido de 7 ciudades, de las cuales dos son inalterables en el orden de visita (Origen y destino) se buscará la solución óptima y para ello se recurrirá al algoritmo BSF. Éste buscará todas y cada una de las combinaciones posibles entre todas las ciudades con la única restricción de no repetir el recorrido a través de alguna de ellas. Una vez obtenido el espacio de soluciones, éstas se filtrarán, quedando sólo aquellas que tengan su origen en Cartagena y terminen en Burgos. Con las soluciones resultantes se procederá a cuantificarlas kilométricamente y para ello y mediante una aplicación informática desarrollada por los autores de este trabajo, se establecerá una conexión entre ellas y la matriz de distancias convirtiéndose las soluciones en distancias numéricas. El siguiente paso es la selección de la menor de ellas, que será el camino óptimo más corto.

#1|Cartagena,Spain

#2|Albacete,Spain

#3|Cuenca,Spain

#4|Toledo,Spain

#5|Soria,Spain

#6|Segovia,Spain

#7|Burgos,Spain

# CT, AB, CU, TO, SO, SG, BU

La solución óptima encontrada para el cálculo del camino más corto mediante el método propuesto es:
CT - CU- SO - BU = 351+ 272 + 143 = **766 Kms,** ruta coincidente con la obtenida mediante Solver (*Figura 9*).


Figura 9: Camino más corto entre las 7 ciudades. Ruta CT-BU (Mapa Google)

Como aportación y mejora del modelo inicial, destacar que junto a la matriz de distancias se obtiene la "Matriz de Tiempos" (Time Matrix) que permite conocer los tiempos que emplearía un vehículo en recorrer esas distancias (*Tabla 9).*

Tabla 9: Matriz de tiempos de recorrido de rutas (Google Map)

| Time Matrix: (in minutes) | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Cartagena | Albacete | Cuenca | Toledo | Soria | Segovia | Burgos |
| Cartagena | 0 | 115 | **206** | 243 | 363 | 292 | 365 |
| Albacete | 115 | 0 | 106 | 143 | 261 | 191 | 266 |
| Cuenca | 206 | 106 | 0 | 110 | **190** | 151 | 216 |
| Toledo | 243 | 143 | 110 | 0 | 179 | 97 | 183 |
| Soria | 363 | 261 | 190 | 179 | 0 | 142 | **107** |
| Segovia | 292 | 191 | 151 | 97 | 142 | 0 | 127 |
| Burgos | 365 | 266 | 216 | 183 | 107 | 127 | 0 |

CT- CU- SO - BU = 206 + 190 + 107 = 503 Minutos = 8, 38 HORAS

Esta posibilidad de obtener los tiempos necesarios de recorrido, permite un cálculo más completo de los costes totales de transporte al posibilitar la incorporación el estudio de los costes laborales u otros.

## 4. Conclusiones

La herramienta Solver incorporada en Excel ha demostrado su utilidad tanto en su aspecto docente para introducir a los alumnos en el campo de los métodos cuantitativos como para su uso empresarial a niveles no demasiado complejos, como los requeridos con frecuencia en PYMES, al mostrar los resultados de forma sencilla, analítica y gráficamente.

A su vez y con el propósito de testar los resultados obtenidos mediante Solver se ha desarrollado un algoritmo que ha servido para comprobar la fiabilidad de ambos

métodos. El algoritmo basado en la Búsqueda por Fuerza Bruta invoca a la aplicación de distancias de Google Maps obteniendo de manera rápida y automática las distancias entre los vértices. Este punto es interesante ante cualquier variación kilométrica acaecida por apertura de nuevas rutas, evitando la incorporación manual necesaria para el desarrollo mediante Solver. Otra de las aportaciones del algoritmo diseñado es la búsqueda de todas las combinaciones posibles y la selección posterior de la más idónea, es decir, el camino más corto buscado.

Junto con las distancias obtenidas mediante este segundo método, se ha conseguido una información adicional de gran utilidad para el ámbito del enrutamiento de vehículos como es el tiempo necesario de recorrido de un vehículo entre los diferentes puntos, así como el de la solución final. Esta información representada en la matriz de tiempos, permite cálculos de disponibilidad de flotas, así como costes relacionados con ellas, lo que permitirá ampliar los resultados y la aplicabilidad del modelo diseñado a las compañías de transporte.

## 5. Referencias Bibliográficas

[1] Bernal García, JJ. (2000). *"Los modelos de red y la minimización de costes. Monta S.A., desea conocer la mejor forma de organizar el transporte de abastecimiento de sus fábricas, así como de elegir la ruta más corta para hacerlo a su nueva planta"*. Estrategia Financiera (CISS-Especial Directivos). Volumen Nº. 160, pp 22-28. Madrid.

[2] Christofides, Nico (1976). *"The Vehicle Routing Problem"*. Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle, 10 (2), pp. 55-70

[3] Laporte, G. (1992). *"The Vehicle Routing Problem: An overview of exact and approximate algorithms"*. European Journal of Operational Research, 59 (3), pp. 345-358.

[4] Laporte, G., Gendreau, M., Potvin, J.Y., Semet, F. (2000). *"Classical and Modern Heuristics for the Vehicle Routing Problem*". International Transactions in Operational Research 7, pp 285-300.

[5] Fisher, Marshall L. (1994). *"Optimal solution of vehicle routing problems using minimum K-trees"*. Operations Research, 42 (4), pp. 626-642.

[6] Bachelet, B., Yon, L. (2007). *"Model enhancement: Improving theoretical optimization with simulation"*. Simulation Modelling Practice and Theory 15 (6), pp 703-715.

[7] Toth, P., Vigo, D. (2002). *"Models, relaxations and exact approaches for the capacitated vehicle routing problem"*. Discrete Applied Mathematics, 123 (1-3), pp. 487-512.

[8] Montemanni, R., Gambardella, L.M., Rizzoli, A.E., Donati, A.V. (2005). *"Ant*

*Colony System*
*for a Dynamic Vehicle Routing Problem"*. Journal of Combinatorial Optimization 10 (4) pp 327-343.

[9] Solomon, Marius M. (1987). "*Algorithms for the Vehicle Routing and scheduling problems with time window constraints"*. Operations Research, 35 (2), pp. 254-265.

[10] De la Fuente-Aragon, MV., Hontoria, E., Ros-McDonell, L. (2012). "*A Simulation-Based Solution for Optimal Logistics of Heavy and Variable-Size Items*". Industrial Engineering: Innovative Networks. Springer Link pp 291-298

[11] Dantzig, G., Fulkerson, R., Johnson, S. (1954). "*Solution of a large scale traveling salesman problem*". Oper. Res. 2, pp 393-410.

[12] Laporte, G. (2007)." *What you should know about the vehicle routing problem"*. Naval Research Logistics 54 (8), pp. 811–819.

[13] Applegate, D.L., Bixby, R.E., Chvátal, V., Cook , W.J. (2007). "*The traveling salesman problem, A computational study*". Princeton. University Press, Princeton, NJ.

[14] Baldacci, R., Christofides, N., Mingozzi, A. (2008). "*An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts*". Mathematical Programming .115 (2), pp 351–385.

[15] http://www.wiley.com/college/tech/winqsb.htm (03/04/2013).

[16] http://www.lindo.com/ (03/04/2013).

[17] Walkenbach, J. (2011). "*Excel 2010. Programación con VBA"*. Anaya Multimedia.-Wiley. Madrid.

## Anexo A
## Programación del Solver

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 33 | | 1 | | | | | | 0 | |
| 34 | | CT | AB | CU | TO | SG | SO | BU | Total |
| 35 | CT | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 36 | AB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 37 | CU | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 38 | TO | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 39 | SG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 40 | SO | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 41 | BU | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 42 | Total | 0 | 0 | 1 | 0 | 0 | 1 | 1 | |
| 43 | | *1* | *0* | *1* | *0* | *0* | *1* | *0* | |
| 44 | Dif | 1 | 0 | 0 | 0 | 0 | 0 | -1 | 0 |
| 45 | Neto | 1 | 0 | 0 | 0 | 0 | 0 | -1 | 0 |

Figura 10: Celdas del problema

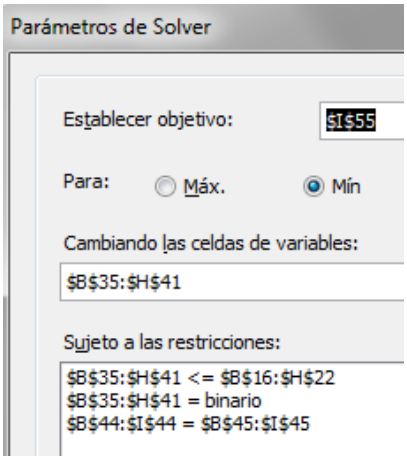



Figura 11: Celda objetivo y restricciones del Solver

## Anexo B

**Sub RUTA()**

```
' RUTA Macro
   SolverOk SetCell:="$I$55", MaxMinVal:=2, ValueOf:=0,
ByChange:="$B$35:$H$41", _
      Engine:=2, EngineDesc:="Simplex LP"
   SolverOk SetCell:="$I$55", MaxMinVal:=2, ValueOf:=0,
ByChange:="$B$35:$H$41", _
      Engine:=2, EngineDesc:="Simplex LP"
   SolverSolve UserFinish:=True
  Linea = 1
```

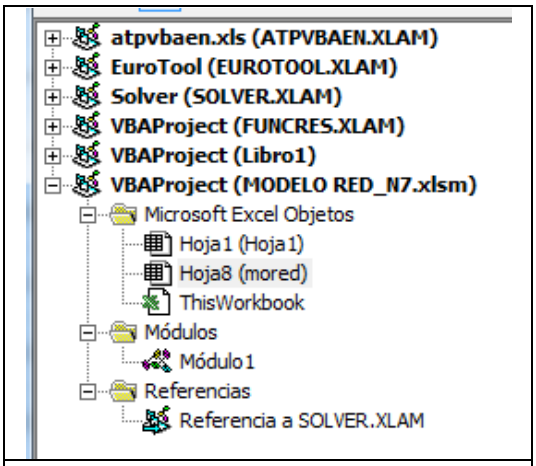
```
FilaIni = 35
ColIni = 3
Inij = 0
Application.ScreenUpdating = False
For i = 0 To 6
   For j = Inij To 5
   If Cells(FilaIni + i, ColIni + j) <> 0 Then
      ActiveSheet.Shapes.Range(Array("linea" & Linea)).Select
      Selection.ShapeRange.Line.ForeColor.RGB = RGB(255, 0, 0)
      Selection.ShapeRange.Line.DashStyle = msoLineSolid
      Selection.ShapeRange.Line.Weight = 2.25
   Else
      ActiveSheet.Shapes.Range(Array("linea" & Linea)).Select
      Selection.ShapeRange.Line.ForeColor.RGB = RGB(0, 0, 0)
      Selection.ShapeRange.Line.DashStyle = msoLineDash
      Selection.ShapeRange.Line.Weight = 2#
   End If
   Linea = Linea + 1
   Next j
   Inij = Inij + 1
Next i
ActiveSheet.Shapes.Range(Array("Botón 1")).Select
Application.ScreenUpdating = True
End Sub

Private Sub Worksheet_Change(ByVal Target As Range)
If InRange(Target, Range("B16:H22")) Then
Linea = 1
FilaIni = 16
ColIni = 3
Inij = 0
Application.ScreenUpdating = False
For i = 0 To 6
   For j = Inij To 5
   If Cells(FilaIni + i, ColIni + j) <> 0 Then
      ActiveSheet.Shapes.Range(Array("linea" & Linea)).Select
      Selection.ShapeRange.Line.Visible = msoTrue
   Else
      ActiveSheet.Shapes.Range(Array("linea" & Linea)).Select
      Selection.ShapeRange.Line.Visible = msoFalse
   End If
   Linea = Linea + 1
   Next j
   Inij = Inij + 1
Next i
Application.ScreenUpdating = True
Target.Select
```

```
End If
End Sub

Function InRange(Range1 As Range, Range2 As Range) As Boolean
' returns True if Range1 is within Range2
Dim InterSectRange As Range
   Set InterSectRange = Application.Intersect(Range1, Range2)
   InRange = Not InterSectRange Is Nothing
   Set InterSectRange = Nothing
End Function
```

**Nota:** Para programar Solver es preciso activar la referencia correspondiente en VB (*)




*(*) Referencia SOLVER.XLAM*