Tesis de Licenciatura en Ciencias de la Computación

“Un Algoritmo GRASP para el Problema de Ruteo de Arcos Capacitados No Dirigidos con Beneficios (UCARPP)”

Anacondio, Leandro – L.U. 487/07 – leandro.anacondio@gmail.com
Gajda, Adrián – L.U. 195/91 – adrian@gajda.com.ar

Directora: Dra. Irene Loiseau

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Argentina

Septiembre de 2019

Agradecimientos

Queremos expresar nuestro agradecimiento a aquellas personas que, de una u otra forma, nos ayudaron a concretar este trabajo de tesis.

A nuestra directora, Dra. Irene Loiseau, por su dedicación, paciencia y constante estímulo.

Leandro Anacondio y Adrián Gajda

A mi mamá por acompañarme y guiarme siempre, a mis abuelos y papá desde dónde estén por hacerme quién soy. A mis amigos por el apoyo y ser un ejemplo. A mi compañero Adrian por el empuje, el aliento y el compromiso y dedicación. Dedicado a mis padres y abuelos.

Leandro Anacondio

A mi esposa Elizabeth, por su por su apoyo, comprensión y permanente estímulo durante todo este último tiempo. A mis padres Blanca y Alfredo, por guiarme en el camino del estudio. A mis amigos, por alentarme constantemente. A mi compañero Leandro por la cordialidad y compromiso asumido. Dedicado a mis padres.

Adrián Gajda

Resumen

El Problema de Ruteo de Arcos Capacitados No Dirigidos con Beneficios (UCARPP - Undirected Capacitated Arc Routing Problem with Profits) es un problema de optimización combinatoria que consiste en encontrar un conjunto de rutas para una flota homogénea de vehículos que puedan satisfacer las restricciones de duración de las rutas y de capacidad de cada vehículo, maximizando el total de la ganancia obtenida, sin que sea obligatorio recorrer todos los ejes.

Una ganancia y una demanda son asociadas a cada eje de un grafo. Cada eje representa un camino que puede ser transitado para atender un cliente en él y tiene asociado también un tiempo de viaje. Se dispone de una flota homogénea de vehículos que tienen una capacidad de transporte dada y con ella se deben satisfacer las demandas y recolectar las ganancias de los ejes. Solo un vehículo puede recoger la demanda de un eje y cobrar la ganancia.

UCARPP es un problema de optimización combinatoria que pertenece a la clase NP-Hard. En la literatura se han propuesto algoritmos exactos y heurísticas. Los primeros, obtienen soluciones óptimas sobre instancias de datos de tamaño reducido. Los segundos, en general, alcanzan soluciones cercanas a las óptimas y a bajo costo computacional.

El objetivo de esta tesis es el desarrollo de un algoritmo heurístico, basado en GRASP, que obtenga resultados competitivos y robustos.

Indice

1. Introducción

Los problemas de ruteo de vehículos (VRP - Vehicle Routing Problem) han obtenido mucha atención por parte de la comunidad científica por ser problemas difíciles con múltiples aplicaciones.

Los problemas de esta clase generalmente se caracterizan por la necesidad de atender a un conjunto determinado de clientes con una flota de vehículos a un costo mínimo. Son problemas de optimización combinatoria que pertenecen a la clase de problemas NP-Hard.

Existen distintas variantes del problema VRP. Entre otras podemos considerar VRP con capacidades (CVRP), VRP con Restricciones de Distancias (DVRP), VRP con Ventanas de Tiempo (VRPTW) y VRP con Carga y Entrega (VRPPD).

CVRP

Es la versión básica de VRP y la más estudiada. El problema consiste en entregar a cada cliente la cantidad de mercadería demandada por cada uno de ellos. Los vehículos son idénticos con base en un único depósito. El objetivo es minimizar el costo total del recorrido a realizar para cubrir a todos los clientes, cumpliendo las restricciones de capacidad de cada camión. CVRP pertenece a la clase de problemas NP-hard y generaliza al problema del viajante de comercio (TSP).

DVRP

En esta variante de VRP se agrega la restricción de longitud (o tiempo) máxima T para cada recorrido. A cada arco se le agrega un valor no negativo representando la longitud del mismo. En el caso de que tanto la restricción de capacidad como de longitud máxima de cada recorrido estén presente el problema se denomina CVRP con Restricciones de Distancia (DCVRP).

VRPTW

En esta versión del problema cada cliente i tiene asociado un intervalo de tiempo [ai,bi] de recepción de mercadería (llamado ventana de tiempo) y un tiempo de servicio S_i, cada arco (i,j) tiene asociado un tiempo de trayecto t_{ij} . Se agrega la restricción de que el servicio de cada cliente debe comenzar dentro de su ventana de tiempo y debe permanecer en el mismo S_i instantes de tiempo. VRPTW es una generalización de CVRP.

VRPPD

En esta variante cada cliente i tiene asociadas dos cantidades d_i y p_i representando las cantidades de entrega y de recolección de mercadería respectivamente. Además para cada cliente i, Oi denota el vértice que es origen de la mercadería a ser entregada y Di denota el vértice destino de la mercadería a cargar en el nodo i. Se agrega la restricción de que los nodos Oi y Di deben ser visitados por el mismo vehículo y en ese mismo orden. VRPPD es una generalización de CVRP.

Los algoritmos para resolver este problema pueden dividirse en dos grandes grupos: Algoritmos Exactos y Algoritmos Heurísticos.

Los algoritmos exactos garantizan encontrar la solución óptima, sin embargo requieren mucho tiempo de proceso. Para mayor detalle sobre métodos exactos para VRP consultar en [Lap92], [Nob87], [Tot00], [Vig98], [TotVi].

Los algoritmos heurísticos generan soluciones en un tiempo menor, pero sin poder establecerse que tan lejos se halla la solución obtenida del valor óptimo al finalizar el algoritmo. Para una lectura adicional sobre heurísticas de ruteo y TSP ver [Bod83], [Chr85], [Min79], [Hel00], [Joh95].

Las técnicas metaheurísticas son procedimientos generales para explorar el espacio de soluciones e identificar buenas soluciones y que pueden ser aplicados a diferentes problemas de optimización con pocas modificaciones. Por lo general están inspiradas en procesos naturales. Las mismas ponen énfasis en la ejecución de una exploración profunda de las regiones más prometedoras del espacio de soluciones. Estos métodos combinan reglas de exploración de vecindarios sofisticadas usando conocimiento específico del problema, estructuras de memoria y recombinación de soluciones. La calidad de las soluciones obtenidas por las metaheurísticas es superior a las producidas mediante las heurísticas clásicas pero a un costo de mayor tiempo de procesamiento.

En todos los modelos para los problemas citados anteriormente, los clientes están representados por vértices de un grafo. Los casos de problemas de ruteo donde los clientes están representados por arcos de un grafo generalmente es más limitada.

Los problemas de ruteo que asocian un beneficio a cada cliente tienen aplicaciones muy interesantes en el contexto del transporte, ya que ayudan a la toma de decisiones de los transportistas.

Consideremos el caso de un transportista con una flota de vehículos de carga. Cada cliente está representado por un arco del grafo. Atender a un cliente significa atravesar el arco desde el origen del viaje hasta el destino. El mismo vehículo puede atender a varios clientes, pero después de un límite de tiempo máximo tiene que volver al depósito, debido, por ejemplo, a las restricciones en el tiempo de conducción del conductor. El transportista tiene un conjunto de clientes que atender, pero estos clientes no utilizan completamente el tiempo disponible para el servicio de uno o más vehículos. Por lo tanto, el transportista está interesado en encontrar nuevos clientes que, junto con los demás, llenen la capacidad de tiempo de los vehículos. Los transportistas podrían presentar ofertas a clientes potenciales. Para hacerlo, deben poder evaluar el impacto de atender a uno o más de estos clientes en su flota. El problema para un operador se convierte en el problema de identificar el subconjunto de clientes potenciales que maximizan el beneficio obtenido por el operador al tiempo que satisface la restricción sobre la duración de cada ruta. Este problema puede ser modelado como un Problema de Ruteo de Arcos Capacitados No Dirigidos con Beneficios (UCARPP - Undirected Capacitated Arc Routing Problem with Profits).

Algunos de los trabajos que abordan este problema son:

- The Undirected Capacitated Arc Routing Problem With Profits, Claudia Archetti, Dominique Feillet, Alain Hertz, M. Grazia Speranzaa. Este trabajo presenta un algoritmo exacto, un VNS y un Tabu Search [Arc10].

- Local Search For The Undirected Capacitated Arc Routing Problem With Profits, E.E. Zachariadis, C.T. Kiranoudis. Este trabajo presenta un algoritmo de local-search con dos conceptos de vecindad [Zac11].

- An artificial bee colony approach for the undirected capacitated arc routing problem with profits, Tunchan Cura. Este trabajo presenta un algoritmos de colonia de abejas [Cur13].

- Variable Neighborhood Search Algorithm for Solving UCARPP Problem, JIN Qian-Qian, LIN Dan. Este trabajo presenta un algoritmo VNS utilizando seis tipos de vecindario [Qui12].

En este modelo un beneficio y una demanda están asociados con cada arco, que representan a los clientes, de un grafo dado. Se asocia un tiempo de viaje a cada arco del grafo. Hay una flota de vehículos con capacidades para servir a los clientes. También se da una duración máxima de la ruta de los vehículos. El beneficio de un arco es cobrado por un solo vehículo. Si un vehículo recoge la demanda del arco, cobra el beneficio. El objetivo de este problema, es encontrar un conjunto de rutas que satisfagan las restricciones de la duración de la ruta y la capacidad del vehículo y maximizar el beneficio recaudado total.

En este documento, proponemos un algoritmo que utiliza la metaheurística GRASP [Feo95] para UCARPP.

En la sección 3.2 detallaremos con profundidad el algoritmo.

2. Metaheurística GRASP

GRASP proviene de las siglas de "Greedy Randomized Adaptative Search Procedure", que en castellano sería algo así como: Procedimientos de Búsqueda basados en funciones "Golosas" Aleatorias Adaptativas. Se trata de una metaheurística basada en un procedimiento de búsqueda goloso, aleatorio y adaptativo el cual, aplicado a una amplia gama de problemas de optimización, garantiza una buena solución, aunque no necesariamente la óptima.

La metodología GRASP fue propuesta al final de la década de los ochenta. El término GRASP fue introducido por Feo y Resende en 1995 [Feo95] como una nueva técnica metaheurística de propósito general y continua aun siendo objeto de estudio [Res16].

GRASP es un procedimiento iterativo donde cada paso consiste en una fase de construcción y una de mejora.

En la fase de construcción se construye iterativamente una solución factible, añadiendo un elemento en cada paso. En cada paso la elección del próximo elemento para ser añadido a la solución parcial viene determinada por una función golosa. Esta función mide el beneficio de añadir cada uno de los elementos y se elige la mejor. Es necesario resaltar el hecho de que esta medida es "miope" en el sentido que no tiene en cuenta qué ocurrirá en los pasos sucesivos una vez que se hace una elección, sino únicamente en el paso actual.

Por otro lado, la heurística es aleatoria porque no selecciona el mejor candidato según la función golosa, sino que, con el objeto de diversificar y no repetir soluciones en dos construcciones diferentes, se construye una lista con los mejores candidatos de entre los que se selecciona uno al azar.

Dado que la fase inicial no garantiza obtener la mejor solución (notar que hay selecciones aleatorias), se aplica un procedimiento de búsqueda local como para mejorar la solución obtenida.

Hemos de tener en cuenta que, en un problema de optimización combinatoria dado, cada solución del problema tiene un conjunto de soluciones asociadas al que se denomina entorno o vecindad de la solución. Así, dada una solución, podemos calcular cualquier solución perteneciente a su entorno mediante una operación denominada movimiento.

Al realizar el GRASP muchas iteraciones, nos está ofreciendo un muestreo del espacio de soluciones existente.

El funcionamiento de GRASP se puede estructurar en tres etapas:

1. Una etapa Constructiva, en la que se construye una solución en forma iterativa. Para ello se genera una lista de candidatos mediante la utilización de una función golosa, considerando una lista restringida de los mejores candidatos, seleccionando aleatoriamente un elemento de la misma.

2. Una etapa de Mejora. En ella se realiza un proceso de búsqueda local a partir de la solución construida hasta cumplir con el criterio de parada.

3. Una etapa de Actualización. Si la solución obtenida mejora a la mejor almacenada, actualizarla.

En la fase de construcción se va generando la solución de partida de forma iterativa, es decir, eligiendo un elemento en cada iteración. En esta fase, la elección del siguiente elemento que formará parte de la solución se determina elaborando una lista de candidatos (LC) con todos los elementos que pueden entrar a formar parte de la solución. Los elementos de esta lista de candidatos se ordenan con respecto a la función golosa que mide, como ya hemos comentado, el beneficio asociado a cada uno de los elementos, creándose a partir de estos valores la llamada lista restringida de candidatos (RCL), cuyo tamaño siempre es menor al de la lista de candidatos y es definido paramétricamente. Esta lista incluirá a aquellos elementos cuyos valores de la función golosa sean más beneficiosos desde el punto de vista del criterio de optimización definido. Una vez constituida esta lista, se seleccionará aleatoriamente un elemento de la misma, que automáticamente pasará a formar parte de la solución de partida.

Un pseudocódigo de la fase de construcción se puede ilustrar como sigue:

```
Procedimiento de ConstrucciónSolución (Solución) Solución = {}
    MIENTRAS La solución no esté completa Hallar
          RCL (RCL)
          a = SelecciónAleatoriaElemento (RCL) Solución =
          Solución + {a}
    FIN MIENTRAS
Fin
```

La fase de construcción de un GRASP no garantiza que las soluciones generadas sean optimas, de aquí el interés de realizar una búsqueda local posterior a la fase construcción.

Un procedimiento de búsqueda local parte de una solución inicial y utilizando un criterio de vecindad, escoge una mejor solución perteneciente a dicho entorno. Dicho de otro modo, realiza un movimiento que, aplicado sobre la solución de partida, da como resultado otra solución que pertenece a la vecindad. Con esta solución, se realiza la fase de actualización. En el GRASP, este proceso es utilizado reiteradamente: en cada iteración el algoritmo hace un movimiento, "visitando" una nueva solución.

Un pseudocódigo genérico del GRASP podría ser el siguiente:

```
Procedimiento    GRASP    (NumIter,    Semilla)
    Valor(MejorSolución) = 0
    Para k = 1 hasta  NumIter
          P^S  = ConstruirSolución  (Semilla)
          P^L  = BúsquedaLocal (P^S)
          Si Valor (P^L) > Valor (MejorSolución)  entonces
              MejorSolución     = P^L
          Fin si
          Devolver  (MejorSolución)  Fin
      Para
Fin
```

3. El Problema de Ruteo de Arcos Capacitados No Dirigidos con Beneficios (UCARPP)

3.1. Modelo

Consideremos un grafo no dirigido G=(V,E) donde V={1,...,n} es el conjunto de vértices y E es el conjunto de ejes.

El vértice 1, el depósito, es el punto de inicio y final de cada recorrido.

Un tiempo de viaje t_e está asociado con cada eje e∈E. Los tiempos de viaje satisfacen la desigualdad triangular.

Un beneficio positivo p_e y una demanda d_e están asociados con cada eje e∈E.

Un conjunto limitado de vehículos está disponible. Cada vehículo tiene una capacidad Q y está sujeto a un límite de tiempo T_{max} en la duración de la ruta. El beneficio de cada eje e∈E puede ser cobrado solo por un vehículo. Si la demanda es recolectada por un vehículo el beneficio también debe ser cobrado por el mismo.

El objetivo del UCARPP es encontrar una ruta para cada uno de los m vehículos de tal manera que se maximice la ganancia total y cada ruta satisfaga las limitaciones de capacidad y límite de tiempo de los vehículos. No necesariamente todos los clientes son atendidos.

3.2. Algoritmo propuesto

El algoritmo propuesto para encontrar la solución de UCARPP se basa en la metaheurística GRASP, con una fase de inicialización que construye una solución factible, y luego realiza una búsqueda local.

Para la solución inicial diseñamos un mecanismo de rankeo parametrizable con variables que pondera el beneficio en función de los tiempos y demandas.

Basándonos en ese ranking se agrega un factor de aleatoriedad entre los posibles candidatos para realizar el próximo avance y se construye así iterativamente la solución.

La fase de búsqueda local utiliza un concepto de vecindad partiendo de la solución construida anteriormente, agregando un segundo factor de aleatoriedad para la selección del camino y aristas a modificar.

El algoritmo finaliza si no se encuentra una solución que mejore un porcentaje definido en una cantidad dada de iteraciones. Si esta solución es encontrada, se reinicia el contador de iteraciones.

En síntesis la estructura general de nuestro enfoque de UCARPP se inicia con el uso de una heurística de construcción simple y eficiente. La solución inicial se mejora luego aplicando búsqueda local.

Para evitar procesamiento innecesario, agregamos una lista que almacena las soluciones iniciales, descartando repeticiones.

Pseudocódigo del algoritmo propuesto:

```
PROCEDIMIENTO GRASP (NumIter, %Mejora)
      MejorSolucion = {}
      PARA K=1 hasta NumIter
            Repetida = False
            MIENTRAS No Repetida
                  SolucionInicial = ConstruccionSolucion()
                  Repetida = Chequear(SolucionInicial)
            FIN MIENTRAS

            SolucionLocal = BusquedaLocal(SolucionInicial)

            SI Beneficio(SolucionLocal) >
                   Beneficio(MejorSolucion)
            ENTONCES
                  MejorSolucion = SolucionLocal
            FIN SI

            SI Comparar(MejorSolucion, SolucionLocal) >
                  %Mejora
            ENTONCES
                  Resetear (NumIter)
            FIN SI
      FIN PARA
      Devolver (MejorSolucion)
FIN
```

3.2.1. Adaptación del grafo para optimizar procesamiento

UCARPP tiene una serie de restricciones con las que se debe cumplir. Entre ellas, el recorrido de cada vehículo no debe exceder el límite temporal propuesto. Para facilitar la toma de decisiones cuando la fase "Golosa" se está ejecutando, decidimos utilizar el algoritmo de Floyd [Flo62], que calcula el mínimo tiempo de tránsito entre todo par de nodos y guardar en cada nodo el tiempo al resto, (en particular al inicial) de modo que en cada paso sabremos cuál es el costo temporal de regresar al depósito.

3.2.2. Obteniendo una solución inicial factible de UCARPP

Inicialmente el objetivo es encontrar un camino para cada uno de los vehículos, maximizando el beneficio y cumpliendo con las restricciones temporales y de capacidad.

```
PROCEDIMIENTO ConstruccionSolucion (Solucion)
      Solución = {}
      MIENTRAS Algun camión le queda tiempo
            PARA CADA Camion con tiempo
            Hallar RCL (Camion)
                  a = SeleccionAleatoria (RCL)
                  Camion.Camino.Agregar (a)
            FIN PARA CADA
      FIN MIENTRAS
      Solucion = caminos de camiones
FIN
```

Los caminos de cada camión se construyen en paralelo.

La idea principal es utilizar un algoritmo goloso con un factor de aleatoriedad para lograr una solución óptima y factible.

El algoritmo es constructivo, en cada paso se elige alguna de las mejores opciones posibles según la función de rankeo. Una vez seleccionado el arco para el siguiente paso, el camión tiene la opción de tomar o no el beneficio que este otorga. La toma de esa decisión es aleatoria y su función parametrizada.

Cuando el camión se queda sin capacidad temporal para continuar por un nuevo eje, se emprende el camino de regreso definido por el algoritmo de Floyd. Una vez que esto ocurre se queda en el depósito y su viaje finaliza.

Para entender por qué el algoritmo brinda una solución inicial nos centraremos en dos factores principales que son necesarios para explicar la factibilidad del mismo:

1) Rankeo de arcos

Los arcos son seleccionados aleatoriamente. En cada iteración se revisan cuales son los arcos disponibles y se ordenan según un concepto de "rentabilidad".

La ecuación que determina el ranking de "rentabilidad" es la siguiente:

El beneficio, el tiempo y la demanda son valores asociados a cada arco del problema. El parámetro que denominamos CostRelation busca darle un peso relativo a cada una de las relaciones dentro del ROI.

ROI = ((beneficio / tiempo) * CostRelation)+((beneficio / demanda) * (1-CostRelation));

Una vez que los arcos disponibles son ordenados según lo que denominamos ROI, seleccionamos los primeros "X %" (siendo X también un parámetro ajustable) y se elige al azar uno de ellos como candidato para el siguiente paso.

2) Cada iteración preserva factibilidad

El objetivo es siempre brindar una solución factible para el problema, por lo que esta debe cumplir con las restricciones ya mencionadas.

El algoritmo avanza en cada paso obteniendo alguno de los arcos mejores rankeados según el criterio antes descripto.

Definimos como arco disponible a aquel arco para el cuál alguno de sus vértices coincide con el nodo actual sobre el cual se encuentra el vehículo, y cuyo otro vértice se encuentra a un costo temporal menor o igual al que dispone el camión, sin tener en cuenta la demanda del mismo respecto de la capacidad restante en el vehículo. En otras palabras, tomamos como disponibles desde un punto dado, a aquellos arcos que llevarán a un nodo del cual se puede volver al depósito sin violar la restricción temporal. Recordemos que inicialmente cada nodo tiene la información de la distancia temporal al resto (en particular al depósito) por eso siempre se puede saber en O(1) cuánto costaría volver si tomo ese arco.

En cada paso solo se toman los arcos que llevan a nodos de los cuales se puede retornar, esto implícitamente hace que a medida que la capacidad temporal de cada camión disminuye, en el momento que solo queda tiempo para emprender el regreso, el camión vuelve indefectiblemente al depósito.

En cuanto a la restricción de capacidad, el beneficio otorgado por servir a cada arco solo es cobrado cuando la capacidad disponible del camión es mayor que la demanda, por lo que esa restricción también es cumplida en cada paso.

Este procedimiento es realizado en cada paso para cada uno de los vehículos.

Se puede comprobar que cada paso preserva factibilidad por lo cual al finalizar el proceso, obtendremos una solución inicial factible.

3.2.3. Los operadores de búsqueda local de la metaheurística propuesta.

Una vez que la fase de construcción inicial es terminada, la solución es sometida a una serie de modificaciones para realizar una búsqueda local en el espacio cercano a esta, con la motivación de encontrar una alternativa que aumente el beneficio.

Como toda metodología de búsqueda local, el procedimiento consiste en moverse entre los vecinos del punto original en la medida en que la nueva solución de UCARPP mejora a la anterior.

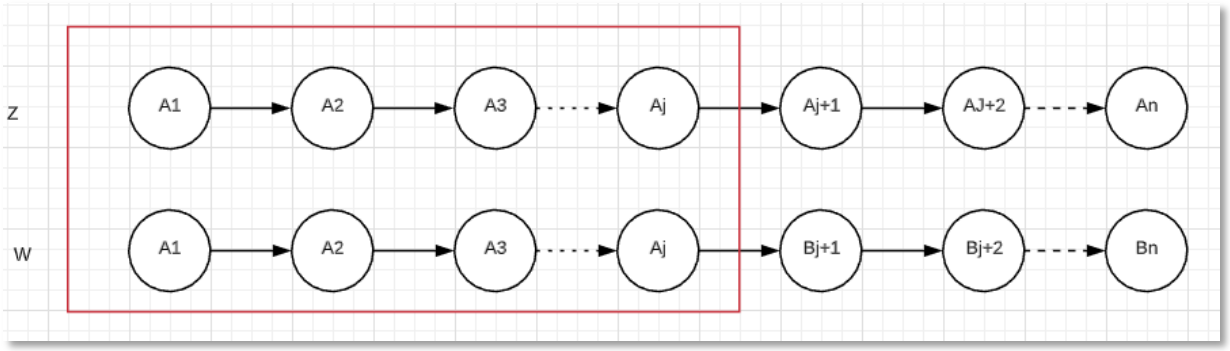
Para esto es necesario definir un concepto de vecindad que aplique al problema. El criterio de vecindad adoptado es el siguiente:

Dada una solución inicial A, una solución B es vecina cuando difieren en el recorrido de uno de los camiones.

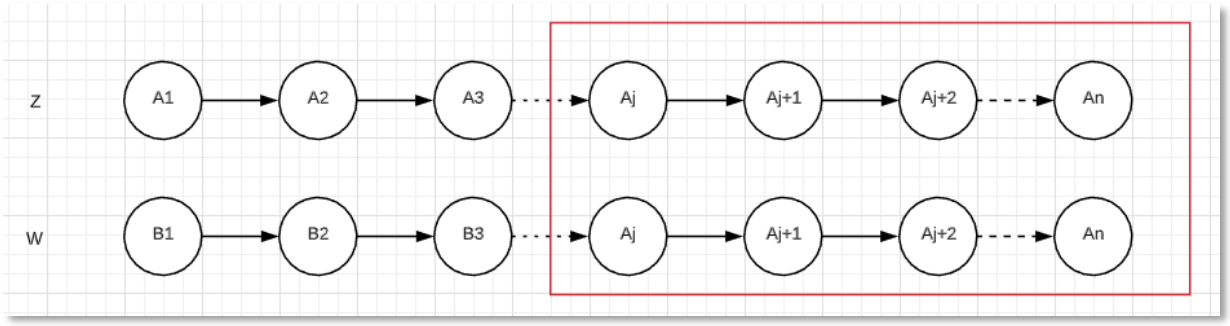
Es decir, para todo camino X,Y,Z pertenecientes a recorridos de A, existen U,V,W pertenecientes a recorridos de B, tal que X=U , Y=V y Z comparte tramos con W.

Llamaremos camino a una secuencia de nodos. Entonces sean $a_1,...,a_n$ y $b_1,...,b_n$ caminos del grafo, podemos escribir a $Z= a_1,...,a_j,...,a_k,...,a_n$ y $W= b_1,...,b_j,...,b_k,...,b_n$ y consideramos Z vecino de W cuando se cumplen alguna de las siguientes condiciones:

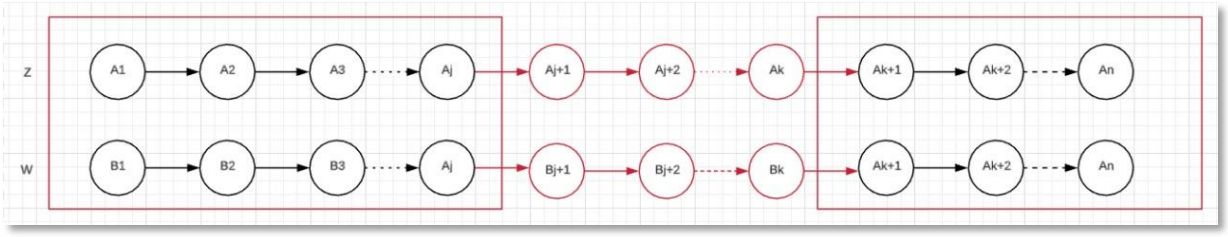
1) $a_1,...,a_j = b_1,...,b_j$



2) $a_j,...,a_n = b_j,...,b_n$



3) $a_1,...,a_j = b_1,...,b_j \ \wedge \ a_k,...,a_n = b_k,...,b_n$



El procedimiento de búsqueda en el espacio local se realiza de la siguiente manera:

Se toma como punto de partida, en forma aleatoria, el camino de uno de los camiones de la solución inicial y se procede a modificar su recorrido, dejando intactos el resto de los caminos. Esto es importante, ya que al no alterar los recorridos nos aseguramos que el resto de los camiones van a seguir cumpliendo con la restricción de límite temporal, y siempre habrá una combinación de arcos a servir tal que la suma de la demanda será menor o igual que la capacidad de dicho camión (en principio sirviendo a los mismos arcos se respetará también con la restricción de capacidad).

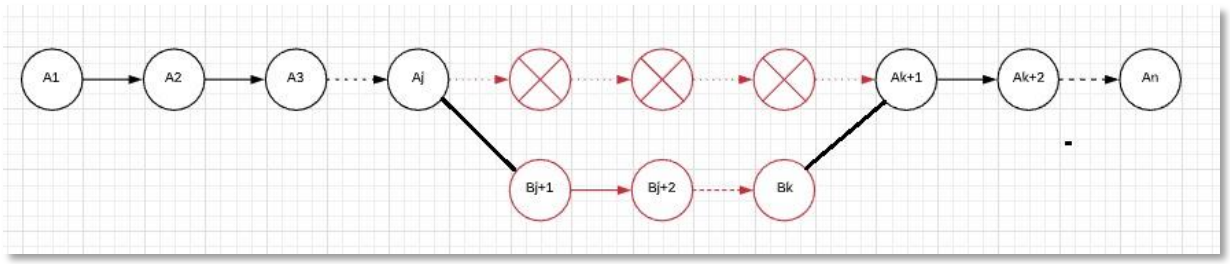
De esta forma, modificando solo un camino basta para que ese recorrido pueda cumplir con las restricciones de forma que la nueva solución sea también factible.

Construcción del nuevo camino

La construcción del camino alternativo consta de dos etapas:

La primera consiste en elegir dos nodos al azar y eliminar los arcos entre los dos.

La segunda etapa se trata de encontrar un camino entre los dos nodos elegidos. Recordemos que inicialmente calculamos Floyd para todo el grafo, por lo cual cada nodo tiene el tiempo mínimo para llegar a cualquier otro nodo. Se puede chequear los tiempos entre cada uno de sus nodos adyacentes y el destino, y así reconstruir el camino de un punto a otro. De esa forma se obtiene la secuencia de arcos nuevos a recorrer y se inserta en reemplazo de los arcos eliminados.



Así queda constituido un nuevo camino que se usará como candidato para una posible solución.

Para que sea realmente una solución alternativa se comprueba si cumple con la restricción de límite temporal, sumando los costos de recorrer esas aristas y contrastándola con la capacidad temporal original del vehículo. De ser satisfactoria, se simula el recorrido de los vehículos por los arcos para obtener el beneficio resultante (en este caso no se le agrega componente de aleatoriedad para satisfacer la demanda y tomar la recompensa, simplemente se van satisfaciendo hasta alcanzar el límite de capacidad). De no serlo se descarta automáticamente.

Finalmente si la validación arroja un resultado mejor que la solución original, esta es reemplazada por la nueva solución obtenida y se sigue iterando con búsquedas locales, hasta que no se logre una mejora dentro de una cota de iteraciones definida.

3.3. Variantes de la heurística propuesta

A lo largo del desarrollo se fueron incorporando una serie de variantes que al no lograr resultados positivos, fueron descartadas.

Las pruebas fueron realizadas sobre instancias propuestas por Archetti et al. [Arc10]. Agradecemos a Claudia Archetti por enviarnos las instancias que utilizaron en su trabajo. Estas instancias detallan los ejes requeridos con los costos, beneficios y las demandas correspondientes.

Todas las variantes fueron probadas utilizando las instancias de test pequeñas (las que poseen 34, 35 y 39 arcos) del conjunto de prueba, para poder analizar los resultados obtenidos y ver su comportamiento con mayor claridad. En todos los casos, se realizaron pruebas con dos, tres y cuatro vehículos.

3.3.1. Local Search con un solo arco

Se probó utilizar otro concepto de vecindad, reemplazando sólo una arista de uno de los viajes de los camiones de la solución inicial. Es decir, tomar alguno de los vehículos al azar de la misma forma que en la heurística definitiva, y seleccionar también aleatoriamente una de sus aristas. Una vez que se cortaba el camino se buscaba un camino alternativo.

El problema de esta solución propuesta es que en general, el único camino posible entre esos dos puntos es pasando por la misma arista que fue removida, por lo que escasamente se conseguían mejoras.

3.3.2. Búsqueda de caminos con Dijkstra

También en pos de encontrar nuevas variantes a la confección de soluciones vecinas, se intentó cortar un tramo de la misma forma descripta, pero reconstruyendo el camino con el algoritmo de camino mínimo de Dijkstra. Se comprobó que la mayoría de las soluciones cumplían con la restricción del costo temporal, y por ende eran factibles, pero aportaba muy pocas variantes (el camino mínimo entre dos nodos en gral era único), por lo que tampoco se lograban soluciones mejores.

3.3.3. Creación de solución inicial de forma semialeatoria

Así como originalmente explicamos la generación de una solución inicial basada en una heurística golosa con un grado de aleatoriedad, también se confeccionó una variante de esta primera fase, para intentar lograr una mayor dispersión en el espacio inicial.

El método consistía en seleccionar un nodo al azar y buscar el camino mínimo con el algoritmo de Floyd entre ese nodo y el inicial. De este modo el trayecto final consistía en un camino entre el depósito y algún nodo y la vuelta.

Dado que el camino producido era completamente arbitrario, era necesario finalmente probar la factibilidad de la solución. En caso de ser positiva se tomaba como válida y luego se le realizaba la búsqueda local pertinente.

La idea de fondo de esto era poder llegar a visitar y recorrer aristas inexploradas por el método anterior, ya sea por lejanía o porque los caminos que conducían hasta ella tenían inicialmente valores de beneficio relativamente bajos.

Después de algunas pruebas notamos que el algoritmo goloso inicialmente descripto arrojaba mejores resultados y esta alternativa fue descartada.

4. Implementación

4.1. Lenguaje y metodología

El trabajo fue desarrollado en .Net Framework v 4.6.1 con C# como lenguaje y diseñamos la arquitectura de la aplicación dividiendo los componentes según la responsabilidad inherente a cada uno.

Parametrizamos una serie de variables del algoritmo que utiliza la aplicación para ajustarlos luego de realizar las pruebas con cada uno.

Esas variables son las siguientes:

RandomPercentage: Determina el tamaño de la lista RCL respecto de la lista de candidatos a tomar en cada iteración de la primera fase.

ImprovementIterationPercentage: Determina cuál es el mínimo porcentaje de mejora que se pide para que el algoritmo continúe iterando.

CostRelation: Es el parámetro que le da peso a la relación Costo/Beneficio que se explicó en la sección de algoritmo.

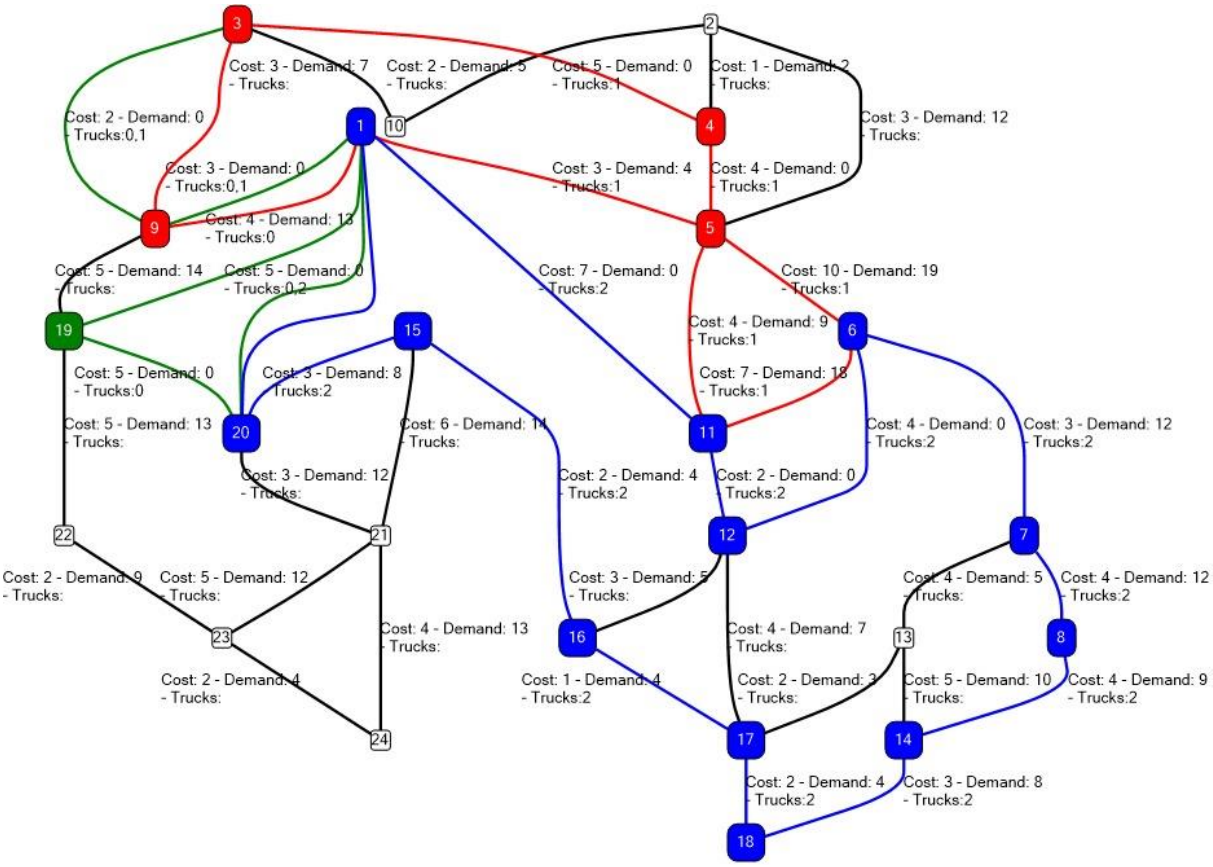
MaxSeedIteration: Determina el máximo de iteraciones de ejecución en el cual se espera que el algoritmo presente alguna mejora. Si no hay mejora en esa cantidad de iteraciones el algoritmo termina.

MaxIterations: Determina el máximo de iteraciones en la búsqueda local en el cual se espera que el algoritmo presente alguna mejora. Si no hay mejora en esa cantidad de iteraciones el algoritmo de búsqueda local termina.

TakeArcPercentage: Es la probabilidad de tomar o no el beneficio del arco que se está recorriendo.

También fue desarrollada una interface gráfica para facilitar el análisis y comprensión de los resultados obtenidos.

En ella se muestra el grafo de una instancia, con sus tiempos y beneficios, así como también cada uno de los recorridos de los vehículos en colores diferentes:



Cada eje posee una etiqueta que detalla el tiempo de viaje, su demanda y que camiones lo cruzaron.

Los ejes en negro no fueron cruzados por ningún vehículo y la etiqueta Trucks, queda vacía.

4.2. Hardware y Sistema Operativo

Las pruebas fueron realizadas en un equipo Intel Core I5 3.2GHz, 16GB RAM con un sistema operativo Windows 10 de 64bits.

El promedio de tiempo de ejecución para cada caso de test es de 3 segundos.

4.3. Ajustes de Parámetros

Los ajustes fueron realizados utilizando las instancias de test val1, val2 y val3 del conjunto de prueba definido en Archetti et al. [Arc10]. En todos los casos, se realizaron pruebas con dos, tres y cuatro vehículos. Para poder analizar los resultados obtenidos y ver su comportamiento con mayor claridad se realizó el promedio de todos ellos.

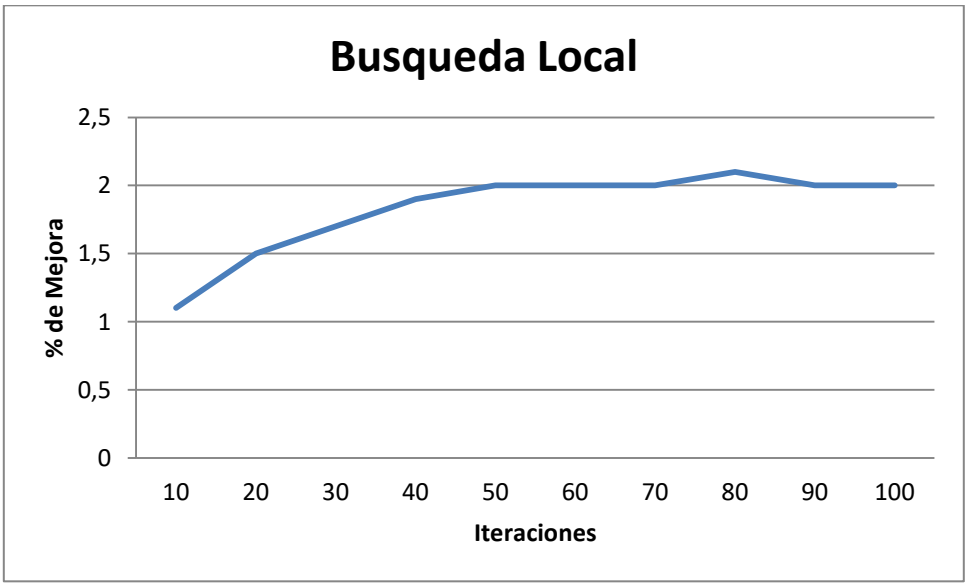
4.3.1. RandomPercentage

Para obtener la mejor combinación de los parámetros de ejecución se comenzó determinado el mejor valor del RandomPercentage. Se utilizaron valores arbitrariamente altos (1000) para MaxSeedIteration y MaxIterations de modo de poder dar foco solo a este parámetro.
Se realizaron pruebas con valores en el rango 10% al 90%, obteniéndose los mejores resultados cuando el parámetro era del 60%.

4.3.2. MaxIterations

Con el parámetro anterior definido, se procedió con MaxIterations, realizando pruebas con valores en el rango de 10 a 100, verificando que no había mejoras a partir de las 50 iteraciones, como puede observarse en el grafico siguiente.

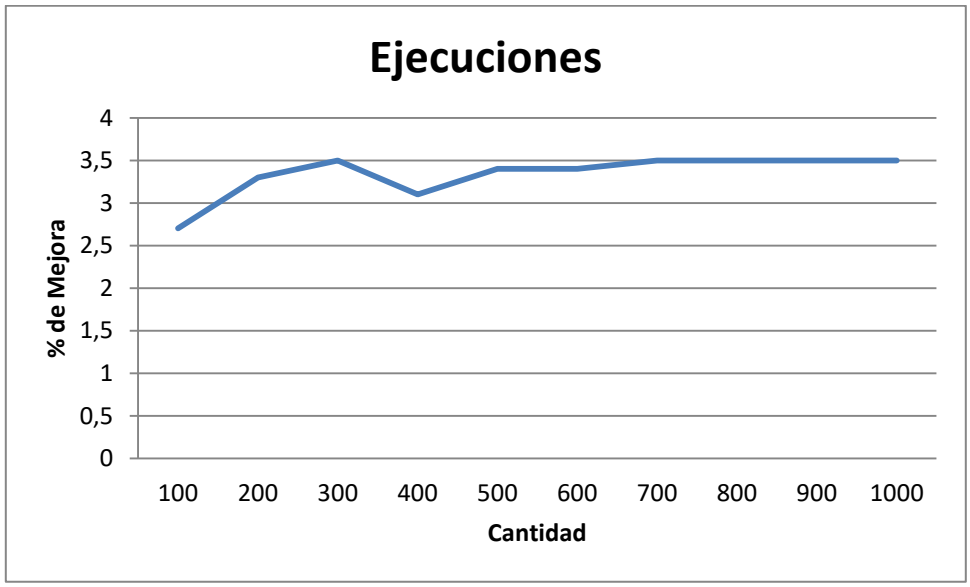
Comportamiento de la solución en función de la cantidad de búsquedas realizadas:



4.3.3. MaxSeedIteration

Finalmente se procedió con MaxSeedIteration, realizando pruebas con valores en el rango de 100 a 10.000, verificando que no había mejoras a partir de las 500 iteraciones, como puede observarse en el grafico siguiente.

Comportamiento de la solución en función de la cantidad de ejecuciones:



5. Resultados

5.1. Archivos de Prueba

Las pruebas fueron realizadas sobre instancias propuestas por Archetti et al. [Arc10], para el problema de ruteo sobre arcos con beneficios.

El conjunto de las instancias originales se compone de 34 casos. Para cada uno de estos casos, considera un número de vehículos m = 2,3,4 obteniéndose 102 instancias de prueba en total.

Las instancias se caracterizaron por el número de ejes con beneficios. Este número es 39 en las instancias val1, 34 en val2, 35 en val3, 69 en val4, 65 en val5, 50 en val6, 66 en val7, 63 en val8, 92 en val9 y 97 en val10.

También consideramos el otro conjunto de 102 casos de test que difieren de los detallados anteriormente solo para los valores de capacidad del camión (Q) y el tiempo máximo que estos pueden recorrer (T_{max}).

Cada instancia del primer conjunto tiene Q = 30 y T_{max} = 40, mientras que en las instancias originales, cada una tiene definido, un valor de Q que varía entre 20 y 250 y T_{max} entre 27 y 133.

Los valores más pequeños de Q y T_{max} hacen que las instancias sean más fáciles de resolver, ya que el número de Los clientes atendidos son más pequeños y las rutas más cortas.

5.2. Tabla 1

Resultados de las instancias con Q=30 y T_{max}=40.

La mejor solución para los casos de test fue obtenida de los datos proporcionados en el trabajo realizado por Archetti et al. [Arc10]. Estos resultados provienen en su mayoría de un algoritmo exacto. En los casos que el algoritmo exacto no finalizó, se tomó la mejor solución que provino de las heurísticas implementadas por los mismos autores.

Instancia	Camiones	Mejor Solución	GRASP Valor	GRASP % Error
val1A.dat	2	175	170	2,86%
val1A.dat	3	251	236	5,98%
val1A.dat	4	323	288	10,84%
val1B.dat	2	175	170	2,86%
val1B.dat	3	252	236	6,35%
val1B.dat	4	324	289	10,80%
val1C.dat	2	122	121	0,82%
val1C.dat	3	179	172	3,91%
val1C.dat	4	233	211	9,44%
val2A.dat	2	125	99	20,80%
val2A.dat	3	168	120	28,57%
val2A.dat	4	183	162	11,48%
val2B.dat	2	125	108	13,60%
val2B.dat	3	168	156	7,14%
val2B.dat	4	183	121	33,88%
val2C.dat	2	85	69	18,82%
val2C.dat	3	115	107	6,96%
val2C.dat	4	125	111	11,20%
val3A.dat	2	143	130	9,09%
val3A.dat	3	196	180	8,16%
val3A.dat	4	236	224	5,08%
val3B.dat	2	133	122	8,27%
val3B.dat	3	182	168	7,69%
val3B.dat	4	222	212	4,50%
val3C.dat	2	92	84	8,70%
val3C.dat	3	127	120	5,51%
val3C.dat	4	154	148	3,90%
val4A.dat	2	165	142	13,94%
val4A.dat	3	240	209	12,92%
val4A.dat	4	308	269	12,66%
val4B.dat	2	166	143	13,86%
val4B.dat	3	241	210	12,86%

val4B.dat	4	309	272	11,97%
val4C.dat	2	155	134	13,55%
val4C.dat	3	226	198	12,39%
val4C.dat	4	291	255	12,37%
val4D.dat	2	125	110	12,00%
val4D.dat	3	182	161	11,54%
val4D.dat	4	235	204	13,19%
val5A.dat	2	166	143	13,86%
val5A.dat	3	227	198	12,78%
val5A.dat	4	285	251	11,93%
val5B.dat	2	166	143	13,86%
val5B.dat	3	227	198	12,78%
val5B.dat	4	285	251	11,93%
val5C.dat	2	166	143	13,86%
val5C.dat	3	227	198	12,78%
val5C.dat	4	285	251	11,93%
val5D.dat	2	133	116	12,78%
val5D.dat	3	183	159	13,11%
val5D.dat	4	229	204	10,92%
val6A.dat	2	208	165	20,67%
val6A.dat	3	286	235	17,83%
val6A.dat	4	355	296	16,62%
val6B.dat	2	208	169	18,75%
val6B.dat	3	286	235	17,83%
val6B.dat	4	355	313	11,83%
val6C.dat	2	137	118	13,87%
val6C.dat	3	192	162	15,63%
val6C.dat	4	244	229	6,15%
val7A.dat	2	181	164	9,39%
val7A.dat	3	263	237	9,89%
val7A.dat	4	342	297	13,16%
val7B.dat	2	181	165	8,84%
val7B.dat	3	263	235	10,65%
val7B.dat	4	344	290	15,70%
val7C.dat	2	154	141	8,44%
val7C.dat	3	223	201	9,87%
val7C.dat	4	291	251	13,75%
val8A.dat	2	163	150	7,98%
val8A.dat	3	233	212	9,01%
val8A.dat	4	300	269	10,33%
val8B.dat	2	164	151	7,93%

val8B.dat	3	234	210	10,26%
val8B.dat	4	301	270	10,30%
val8C.dat	2	129	119	7,75%
val8C.dat	3	186	166	10,75%
val8C.dat	4	240	209	12,92%
val9A.dat	2	176	162	7,95%
val9A.dat	3	253	230	9,09%
val9A.dat	4	330	295	10,61%
val9B.dat	2	175	162	7,43%
val9B.dat	3	253	229	9,49%
val9B.dat	4	329	293	10,94%
val9C.dat	2	176	162	7,95%
val9C.dat	3	253	232	8,30%
val9C.dat	4	330	293	11,21%
val9D.dat	2	144	134	6,94%
val9D.dat	3	208	192	7,69%
val9D.dat	4	273	243	10,99%
val10A.dat	2	170	143	15,88%
val10A.dat	3	244	186	23,77%
val10A.dat	4	317	246	22,40%
val10B.dat	2	171	147	14,04%
val10B.dat	3	245	190	22,45%
val10B.dat	4	318	242	23,90%
val10C.dat	2	169	146	13,61%
val10C.dat	3	243	192	20,99%
val10C.dat	4	316	243	23,10%
val10D.dat	2	154	129	16,23%
val10D.dat	3	221	171	22,62%
val10D.dat	4	288	224	22,22%

5.3. Tabla 2

Resultados de las instancias con los valores originales de Q y T_{max} proporcionados en el trabajo realizado por Archetti et al. [Arc10].

La mejor solución para los casos de test fue obtenida de los datos proporcionados en el trabajo realizado por Archetti et al. [Arc10]. Estos resultados provienen en su mayoría de un algoritmo exacto. En los casos que el algoritmo exacto no finalizó, se tomó la mejor solución que provino de las heurísticas implementadas por los mismos autores.

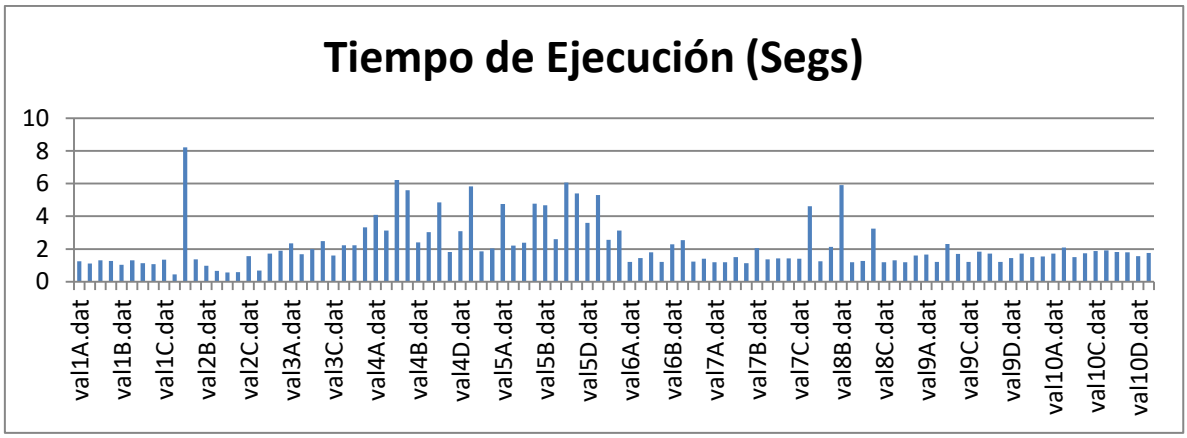
Instancia	Camiones	Mejor Solución	GRASP Valor	GRASP % Error
val1A.dat	2	713	685	3,93%
val1A.dat	3	728	728	0,00%
val1A.dat	4	728	728	0,00%
val1B.dat	2	516	489	5,23%
val1B.dat	3	682	641	6,01%
val1B.dat	4	731	696	4,79%
val1C.dat	2	179	160	10,61%
val1C.dat	3	256	230	10,16%
val1C.dat	4	328	294	10,37%
val2A.dat	2	585	555	5,13%
val2A.dat	3	605	605	0,00%
val2A.dat	4	605	605	0,00%
val2B.dat	2	409	399	2,44%
val2B.dat	3	481	481	0,00%
val2B.dat	4	541	484	10,54%
val2C.dat	2	146	124	15,07%
val2C.dat	3	209	182	12,92%
val2C.dat	4	261	236	9,58%
val3A.dat	2	245	228	6,94%
val3A.dat	3	253	245	3,16%
val3A.dat	4	253	253	0,00%
val3B.dat	2	186	180	3,23%
val3B.dat	3	227	206	9,25%
val3B.dat	4	237	223	5,91%
val3C.dat	2	65	58	10,77%
val3C.dat	3	92	82	10,87%
val3C.dat	4	116	106	8,62%
val4A.dat	2	923	854	7,48%
val4A.dat	3	1211	1037	14,37%
val4A.dat	4	1242	1230	0,97%
val4B.dat	2	718	689	4,04%
val4B.dat	3	978	885	9,51%

val4B.dat	4	1195	1001	16,23%
val4C.dat	2	566	523	7,60%
val4C.dat	3	798	717	10,15%
val4C.dat	4	964	859	10,89%
val4D.dat	2	300	272	9,33%
val4D.dat	3	426	378	11,27%
val4D.dat	4	547	484	11,52%
val5A.dat	2	948	897	5,38%
val5A.dat	3	1191	1043	12,43%
val5A.dat	4	1221	1162	4,83%
val5B.dat	2	725	685	5,52%
val5B.dat	3	978	892	8,79%
val5B.dat	4	1137	1040	8,53%
val5C.dat	2	592	552	6,76%
val5C.dat	3	773	718	7,12%
val5C.dat	4	925	860	7,03%
val5D.dat	2	326	300	7,98%
val5D.dat	3	464	411	11,42%
val5D.dat	4	588	519	11,73%
val6A.dat	2	700	688	1,71%
val6A.dat	3	871	819	5,97%
val6A.dat	4	891	867	2,69%
val6B.dat	2	504	498	1,19%
val6B.dat	3	684	636	7,02%
val6B.dat	4	804	753	6,34%
val6C.dat	2	207	191	7,73%
val6C.dat	3	291	272	6,53%
val6C.dat	4	371	348	6,20%
val7A.dat	2	848	809	4,60%
val7A.dat	3	1081	1046	3,24%
val7A.dat	4	1110	1095	1,35%
val7B.dat	2	633	607	4,11%
val7B.dat	3	911	860	5,60%
val7B.dat	4	1082	998	7,76%
val7C.dat	2	286	255	10,84%
val7C.dat	3	418	367	12,20%
val7C.dat	4	541	479	11,46%
val8A.dat	2	856	780	8,88%
val8A.dat	3	1074	943	12,20%
val8A.dat	4	1099	1036	5,73%
val8B.dat	2	665	603	9,32%

val8B.dat	3	893	825	7,61%
val8B.dat	4	1058	916	13,42%
val8C.dat	2	270	240	11,11%
val8C.dat	3	389	337	13,37%
val8C.dat	4	496	433	12,70%
val9A.dat	2	996	915	8,13%
val9A.dat	3	1229	1126	8,38%
val9A.dat	4	1265	1201	5,06%
val9B.dat	2	774	733	5,30%
val9B.dat	3	1033	927	10,26%
val9B.dat	4	1216	1071	11,92%
val9C.dat	2	627	599	4,47%
val9C.dat	3	876	808	7,76%
val9C.dat	4	1066	958	10,13%
val9D.dat	2	296	275	7,09%
val9D.dat	3	426	400	6,10%
val9D.dat	4	555	505	9,01%
val10A.dat	2	1038	939	9,54%
val10A.dat	3	1302	1162	10,75%
val10A.dat	4	1348	1315	2,45%
val10B.dat	2	829	781	5,79%
val10B.dat	3	1104	1014	8,15%
val10B.dat	4	1307	1126	13,85%
val10C.dat	2	658	616	6,38%
val10C.dat	3	912	800	12,28%
val10C.dat	4	1129	938	16,92%
val10D.dat	2	322	285	11,49%
val10D.dat	3	458	388	15,28%
val10D.dat	4	591	462	21,83%

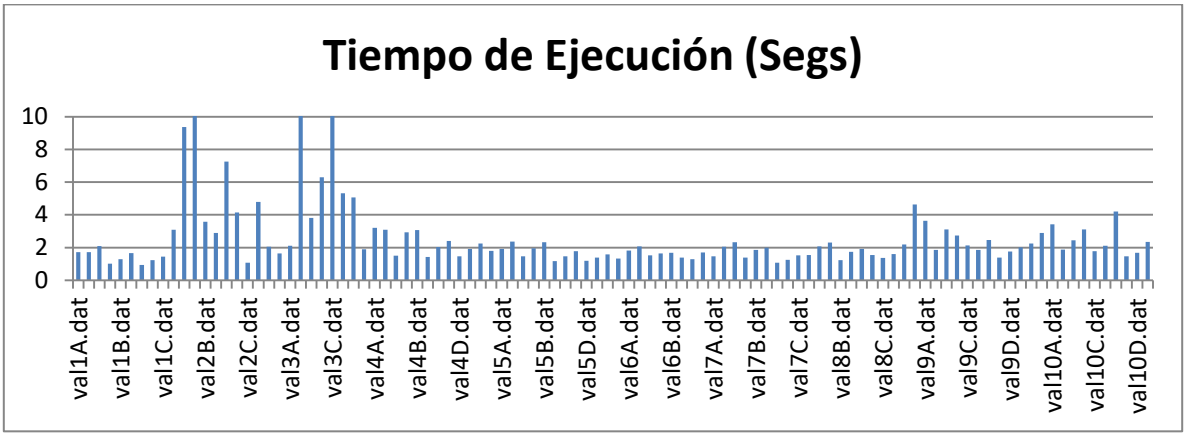
5.4. Tiempo de ejecución

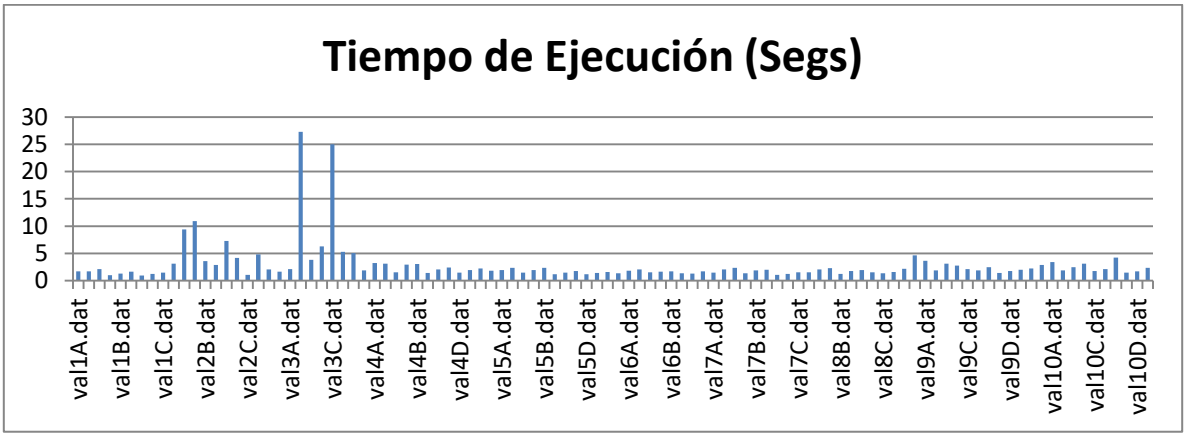
Resultados de las instancias con Q=30 y T_{max}=40.



Promedio: 2,22 segundos

Resultados de las instancias con los valores originales de Q y T_{max}.





Promedio: 2,85 segundos

5.5. Comparación con otros trabajos

Nuestro trabajo es comparado con el desarrollado por Archetti et al. [Arc10] y Zachariadis et al. [Zac11].

No tuvimos oportunidad de acceder a los papers desarrollados por Tunchan Cura et al. [Cur13] y Qian-Qian et al. [Qui12]

Comparación con el desarrollo de Archetti et al. [Arc10]:

Los resultados de nuestro algoritmo GRASP son comparados con la mejor solución y con los seis algoritmos propuestos en dicho trabajo, a saber:

- VNS S = VNS Slow
- VNS F = VNS Fast
- Tabu fea S = Tabu feasible Slow
- Tabu fea F = Tabu feasible Fast
- Tabu adm S = Tabu admissible Slow
- Tabu adm F = Tabu admissible Fast

Resultados de las instancias con Q=30 y T_{max}=40.

Instancia	Camiones	GRASP % Error	VNS S	VNS F	Tabu fea S	Tabu fea F	Tabu adm S	Tabu adm F
val1A.dat	2	2,86%	0,57%	0,57%	0,00%	0,57%	0,00%	0,00%
val1A.dat	3	5,98%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val1A.dat	4	10,84%	0,31%	0,00%	0,31%	0,00%	0,94%	0,00%
val1B.dat	2	2,86%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val1B.dat	3	6,35%	0,00%	0,00%	1,61%	0,00%	0,00%	0,00%
val1B.dat	4	10,80%	0,31%	0,00%	0,31%	0,31%	0,00%	0,00%
val1C.dat	2	0,82%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val1C.dat	3	3,91%	0,56%	0,00%	0,00%	0,00%	0,56%	0,00%
val1C.dat	4	9,44%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val2A.dat	2	20,80%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val2A.dat	3	28,57%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val2A.dat	4	11,48%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val2B.dat	2	13,60%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val2B.dat	3	7,14%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val2B.dat	4	33,88%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val2C.dat	2	18,82%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val2C.dat	3	6,96%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val2C.dat	4	11,20%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val3A.dat	2	9,09%	0,00%	0,00%	2,88%	0,00%	0,00%	0,00%
val3A.dat	3	8,16%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%

val3A.dat	4	5,08%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val3B.dat	2	8,27%	0,00%	0,00%	2,31%	0,00%	0,00%	0,00%
val3B.dat	3	7,69%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val3B.dat	4	4,50%	0,00%	0,00%	0,45%	0,00%	0,00%	0,00%
val3C.dat	2	8,70%	0,00%	0,00%	6,98%	0,00%	0,00%	0,00%
val3C.dat	3	5,51%	0,00%	0,00%	3,25%	0,00%	0,00%	0,00%
val3C.dat	4	3,90%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val4A.dat	2	13,94%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val4A.dat	3	12,92%	0,42%	0,42%	0,00%	0,84%	1,27%	0,42%
val4A.dat	4	12,66%	0,00%	0,00%	0,00%	1,99%	1,99%	0,00%
val4B.dat	2	13,86%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val4B.dat	3	12,86%	0,00%	0,42%	0,42%	0,42%	0,00%	0,42%
val4B.dat	4	11,97%	0,32%	0,00%	0,98%	1,64%	1,31%	0,32%
val4C.dat	2	13,55%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val4C.dat	3	12,39%	0,44%	0,44%	0,44%	0,44%	0,44%	0,44%
val4C.dat	4	12,37%	1,04%	0,00%	0,00%	0,69%	0,69%	0,69%
val4D.dat	2	12,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val4D.dat	3	11,54%	0,00%	0,00%	1,11%	0,00%	0,00%	0,00%
val4D.dat	4	13,19%	0,00%	0,86%	1,29%	0,86%	0,86%	0,86%
val5A.dat	2	13,86%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val5A.dat	3	12,78%	0,00%	0,00%	0,00%	4,13%	0,00%	0,00%
val5A.dat	4	11,93%	0,00%	0,00%	0,00%	1,42%	0,00%	0,00%
val5B.dat	2	13,86%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val5B.dat	3	12,78%	0,00%	0,00%	0,00%	4,13%	0,00%	0,00%
val5B.dat	4	11,93%	0,00%	0,00%	0,00%	1,42%	0,00%	0,00%
val5C.dat	2	13,86%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val5C.dat	3	12,78%	0,00%	0,00%	0,00%	4,13%	0,00%	0,00%
val5C.dat	4	11,93%	0,00%	0,00%	0,00%	1,42%	0,00%	0,00%
val5D.dat	2	12,78%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val5D.dat	3	13,11%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val5D.dat	4	10,92%	0,00%	0,00%	0,00%	3,15%	0,00%	0,00%
val6A.dat	2	20,67%	0,00%	0,00%	0,00%	0,00%	3,48%	0,00%
val6A.dat	3	17,83%	0,70%	0,00%	1,06%	0,00%	2,51%	0,00%
val6A.dat	4	16,62%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val6B.dat	2	18,75%	0,00%	0,00%	0,00%	0,00%	3,48%	0,00%
val6B.dat	3	17,83%	0,70%	0,00%	1,06%	0,00%	2,51%	0,00%
val6B.dat	4	11,83%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val6C.dat	2	13,87%	1,48%	0,00%	0,00%	0,00%	0,00%	0,00%
val6C.dat	3	15,63%	0,00%	0,00%	0,52%	0,00%	1,59%	0,00%
val6C.dat	4	6,15%	1,67%	0,00%	0,83%	2,09%	1,67%	0,00%
val7A.dat	2	9,39%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%

val7A.dat	3	9,89%	0,38%	0,38%	0,00%	1,94%	0,38%	0,38%
val7A.dat	4	13,16%	0,59%	0,29%	0,59%	2,40%	0,29%	0,88%
val7B.dat	2	8,84%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val7B.dat	3	10,65%	0,38%	0,38%	0,38%	0,38%	0,38%	0,38%
val7B.dat	4	15,70%	0,58%	0,29%	0,58%	3,30%	1,78%	0,88%
val7C.dat	2	8,44%	0,00%	0,00%	0,00%	0,00%	0,65%	0,00%
val7C.dat	3	9,87%	0,00%	0,00%	0,90%	0,00%	0,90%	0,45%
val7C.dat	4	13,75%	0,34%	0,34%	0,00%	1,39%	0,34%	0,34%
val8A.dat	2	7,98%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val8A.dat	3	9,01%	0,00%	0,00%	0,00%	0,43%	0,00%	0,00%
val8A.dat	4	10,33%	0,00%	0,00%	0,00%	0,00%	0,33%	0,00%
val8B.dat	2	7,93%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val8B.dat	3	10,26%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val8B.dat	4	10,30%	0,00%	0,00%	0,00%	0,33%	0,67%	0,00%
val8C.dat	2	7,75%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val8C.dat	3	10,75%	0,00%	0,00%	0,00%	1,09%	0,00%	0,00%
val8C.dat	4	12,92%	0,00%	0,00%	0,00%	2,13%	0,00%	0,00%
val9A.dat	2	7,95%	0,00%	1,15%	0,00%	0,00%	7,32%	0,00%
val9A.dat	3	9,09%	0,00%	0,00%	0,40%	2,43%	1,20%	0,00%
val9A.dat	4	10,61%	1,54%	0,92%	1,85%	10,00%	4,76%	0,00%
val9B.dat	2	7,43%	0,00%	0,57%	1,74%	0,00%	2,34%	0,00%
val9B.dat	3	9,49%	1,20%	0,00%	2,02%	3,27%	3,27%	0,00%
val9B.dat	4	10,94%	0,92%	0,92%	2,81%	3,13%	1,23%	0,00%
val9C.dat	2	7,95%	0,00%	1,15%	0,00%	0,00%	7,32%	0,00%
val9C.dat	3	8,30%	0,00%	0,00%	0,40%	2,43%	1,20%	0,00%
val9C.dat	4	11,21%	1,54%	0,92%	1,85%	10,00%	4,76%	0,00%
val9D.dat	2	6,94%	1,41%	0,70%	0,00%	0,70%	1,41%	0,00%
val9D.dat	3	7,69%	0,48%	0,48%	0,48%	0,00%	1,46%	0,48%
val9D.dat	4	10,99%	1,49%	1,49%	1,87%	4,20%	3,80%	1,11%
val10A.dat	2	15,88%	0,59%	0,00%	1,19%	5,59%	5,59%	3,66%
val10A.dat	3	23,77%	2,52%	4,27%	4,72%	7,49%	4,72%	1,67%
val10A.dat	4	22,40%	4,62%	5,67%	2,59%	6,38%	5,67%	3,59%
val10B.dat	2	14,04%	4,27%	1,18%	3,64%	4,27%	1,18%	3,01%
val10B.dat	3	22,45%	2,94%	3,38%	2,08%	7,93%	6,06%	3,81%
val10B.dat	4	23,90%	2,91%	3,25%	3,25%	5,65%	5,30%	4,61%
val10C.dat	2	13,61%	3,68%	0,60%	2,42%	4,97%	3,68%	0,60%
val10C.dat	3	20,99%	1,67%	3,40%	3,40%	7,05%	3,85%	1,67%
val10C.dat	4	23,10%	0,96%	3,95%	2,27%	6,04%	5,33%	3,61%
val10D.dat	2	16,23%	2,67%	1,99%	3,36%	7,69%	0,00%	3,36%
val10D.dat	3	22,62%	1,84%	3,27%	2,79%	5,24%	3,27%	3,27%
val10D.dat	4	22,22%	1,41%	5,49%	2,86%	5,49%	5,49%	2,86%

Resultados de las instancias con los valores originales de Q y $T_{max.}$

Instancia	Camiones	GRASP % Error	VNS S	VNS F	Tabu fea S	Tabu fea F	Tabu adm S	Tabu adm F
val1A.dat	2	3,93%	0,00%	0,00%	0,28%	0,42%	0,00%	0,00%
val1A.dat	3	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val1A.dat	4	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val1B.dat	2	5,23%	0,58%	0,00%	0,58%	1,36%	0,58%	0,58%
val1B.dat	3	6,01%	0,00%	0,44%	0,44%	1,03%	0,44%	0,88%
val1B.dat	4	4,79%	0,00%	0,00%	37,48%	0,00%	0,00%	0,00%
val1C.dat	2	10,61%	0,00%	0,00%	0,00%	0,00%	0,00%	0,56%
val1C.dat	3	10,16%	0,00%	0,00%	0,00%	0,00%	0,78%	0,39%
val1C.dat	4	10,37%	0,00%	0,61%	0,00%	0,61%	1,22%	1,22%
val2A.dat	2	5,13%	0,00%	0,00%	0,00%	1,03%	0,00%	0,00%
val2A.dat	3	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val2A.dat	4	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val2B.dat	2	2,44%	0,00%	0,00%	2,44%	2,44%	0,00%	0,00%
val2B.dat	3	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val2B.dat	4	10,54%	0,00%	0,00%	0,74%	1,48%	0,00%	0,00%
val2C.dat	2	15,07%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val2C.dat	3	12,92%	0,00%	0,00%	0,00%	0,00%	0,48%	0,48%
val2C.dat	4	9,58%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val3A.dat	2	6,94%	0,00%	0,82%	0,00%	0,82%	0,00%	0,00%
val3A.dat	3	3,16%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val3A.dat	4	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val3B.dat	2	3,23%	0,00%	0,00%	0,00%	1,08%	1,08%	0,00%
val3B.dat	3	9,25%	0,00%	0,00%	0,00%	0,88%	0,00%	0,00%
val3B.dat	4	5,91%	0,00%	0,00%	0,00%	0,84%	0,00%	0,00%
val3C.dat	2	10,77%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val3C.dat	3	10,87%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val3C.dat	4	8,62%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val4A.dat	2	7,48%	1,19%	1,73%	6,50%	3,90%	1,30%	0,00%
val4A.dat	3	14,37%	0,00%	0,83%	2,23%	5,70%	0,66%	0,00%
val4A.dat	4	0,97%	0,00%	0,00%	0,00%	91,38%	0,00%	0,00%
val4B.dat	2	4,04%	2,79%	0,00%	4,18%	4,04%	0,70%	0,42%
val4B.dat	3	9,51%	0,00%	1,43%	7,46%	7,46%	2,04%	1,12%
val4B.dat	4	16,23%	0,00%	0,67%	10,38%	6,69%	0,25%	0,08%
val4C.dat	2	7,60%	2,12%	2,30%	4,77%	3,00%	2,30%	0,71%
val4C.dat	3	10,15%	1,63%	2,13%	2,38%	3,38%	0,00%	2,63%
val4C.dat	4	10,89%	0,73%	1,24%	2,07%	3,63%	0,52%	0,00%
val4D.dat	2	9,33%	0,67%	0,33%	1,00%	0,00%	0,67%	0,00%
val4D.dat	3	11,27%	0,23%	0,23%	0,00%	0,70%	0,47%	0,23%

val4D.dat	4	11,52%	0,55%	0,37%	0,00%	0,91%	0,91%	0,37%
val5A.dat	2	5,38%	1,79%	2,53%	2,00%	3,16%	0,00%	0,95%
val5A.dat	3	12,43%	0,34%	1,18%	1,26%	2,10%	0,00%	0,92%
val5A.dat	4	4,83%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val5B.dat	2	5,52%	1,79%	2,21%	0,83%	0,97%	0,00%	0,55%
val5B.dat	3	8,79%	1,43%	0,00%	9,41%	6,03%	1,33%	0,31%
val5B.dat	4	8,53%	0,53%	1,32%	8,00%	2,55%	0,00%	0,79%
val5C.dat	2	6,76%	0,00%	3,55%	9,97%	1,18%	2,03%	1,18%
val5C.dat	3	7,12%	1,29%	2,20%	9,96%	4,27%	0,00%	2,85%
val5C.dat	4	7,03%	0,00%	0,76%	8,43%	6,16%	2,70%	3,46%
val5D.dat	2	7,98%	1,23%	0,31%	0,00%	0,31%	0,31%	0,61%
val5D.dat	3	11,42%	1,72%	1,29%	0,00%	1,08%	1,72%	1,72%
val5D.dat	4	11,73%	0,00%	0,51%	0,00%	4,59%	1,70%	0,34%
val6A.dat	2	1,71%	0,00%	0,57%	5,57%	3,00%	3,71%	2,14%
val6A.dat	3	5,97%	0,00%	0,00%	1,38%	1,84%	0,00%	1,38%
val6A.dat	4	2,69%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val6B.dat	2	1,19%	0,00%	0,00%	0,79%	0,00%	0,99%	0,60%
val6B.dat	3	7,02%	0,15%	2,92%	8,04%	2,78%	0,00%	1,90%
val6B.dat	4	6,34%	0,00%	1,24%	2,61%	2,36%	1,00%	1,00%
val6C.dat	2	7,73%	0,00%	0,00%	0,00%	0,00%	0,48%	0,00%
val6C.dat	3	6,53%	0,00%	0,34%	0,00%	0,69%	1,37%	0,00%
val6C.dat	4	6,20%	0,00%	0,27%	0,27%	1,35%	0,81%	1,89%
val7A.dat	2	4,60%	0,00%	1,65%	9,20%	3,07%	1,42%	1,89%
val7A.dat	3	3,24%	0,28%	0,19%	4,07%	0,74%	0,00%	0,28%
val7A.dat	4	1,35%	0,00%	0,00%	0,00%	80,54%	0,00%	0,00%
val7B.dat	2	4,11%	0,00%	0,16%	0,95%	0,79%	2,37%	2,37%
val7B.dat	3	5,60%	0,00%	3,73%	0,22%	3,62%	2,96%	2,96%
val7B.dat	4	7,76%	0,00%	0,92%	6,93%	1,94%	0,74%	1,29%
val7C.dat	2	10,84%	0,00%	0,00%	1,40%	0,00%	0,00%	3,15%
val7C.dat	3	12,20%	0,00%	0,96%	0,00%	4,07%	2,39%	5,50%
val7C.dat	4	11,46%	0,18%	1,66%	0,55%	3,33%	1,29%	4,81%
val8A.dat	2	8,88%	0,82%	0,70%	2,92%	1,99%	0,00%	0,82%
val8A.dat	3	12,20%	0,19%	0,65%	0,00%	1,12%	0,28%	0,84%
val8A.dat	4	5,73%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val8B.dat	2	9,32%	1,65%	2,11%	0,00%	1,35%	1,80%	2,26%
val8B.dat	3	7,61%	0,00%	0,78%	0,34%	1,46%	1,01%	1,01%
val8B.dat	4	13,42%	0,38%	0,85%	0,85%	5,29%	0,00%	1,61%
val8C.dat	2	11,11%	1,11%	0,00%	0,37%	0,00%	0,00%	0,00%
val8C.dat	3	13,37%	1,54%	1,80%	0,51%	1,29%	2,31%	1,80%
val8C.dat	4	12,70%	0,00%	2,02%	0,20%	3,43%	1,41%	2,62%
val9A.dat	2	8,13%	0,00%	0,70%	1,20%	3,51%	2,91%	0,70%

val9A.dat	3	8,38%	0,73%	0,73%	0,00%	1,30%	0,24%	0,65%
val9A.dat	4	5,06%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val9B.dat	2	5,30%	0,00%	5,94%	2,33%	3,23%	3,75%	2,84%
val9B.dat	3	10,26%	0,19%	3,58%	3,39%	0,19%	0,10%	0,00%
val9B.dat	4	11,92%	1,81%	1,32%	8,55%	8,39%	0,00%	1,07%
val9C.dat	2	4,47%	3,03%	3,35%	3,19%	2,07%	0,00%	1,75%
val9C.dat	3	7,76%	0,23%	6,96%	0,00%	4,79%	1,48%	4,34%
val9C.dat	4	10,13%	0,66%	5,91%	0,28%	5,07%	0,00%	3,94%
val9D.dat	2	7,09%	0,00%	0,68%	0,34%	0,34%	1,35%	0,34%
val9D.dat	3	6,10%	0,00%	1,17%	0,70%	0,94%	0,23%	1,17%
val9D.dat	4	9,01%	0,54%	1,08%	0,00%	1,44%	1,08%	1,08%
val10A.dat	2	9,54%	1,93%	2,70%	0,00%	4,43%	3,37%	1,06%
val10A.dat	3	10,75%	0,00%	0,84%	0,31%	2,30%	0,23%	0,77%
val10A.dat	4	2,45%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
val10B.dat	2	5,79%	0,00%	2,29%	1,45%	6,03%	2,90%	0,84%
val10B.dat	3	8,15%	0,00%	3,53%	0,63%	9,96%	1,27%	1,72%
val10B.dat	4	13,85%	0,00%	0,77%	3,52%	5,36%	1,45%	1,30%
val10C.dat	2	6,38%	2,13%	0,00%	1,06%	11,40%	0,61%	1,37%
val10C.dat	3	12,28%	0,33%	0,44%	6,80%	13,82%	1,54%	0,00%
val10C.dat	4	16,92%	0,00%	6,11%	4,07%	13,29%	4,52%	2,83%
val10D.dat	2	11,49%	0,00%	1,55%	0,00%	2,80%	4,04%	2,17%
val10D.dat	3	15,28%	2,18%	2,18%	0,66%	2,18%	0,00%	2,40%
val10D.dat	4	21,83%	0,00%	0,51%	1,35%	10,49%	3,55%	2,54%

Comparación con el desarrollo de Zachariadis et al. [Zac11]:

Los resultados de nuestro algoritmo GRASP son comparados con la mejor solución y con el algoritmo propuesto en dicho trabajo, a saber:

- LSM = Local Search Metaheuristic

Resultados de las instancias con Q=30 y T_{max}=40.

Instancia	Camiones	GRASP % Error	LSM
val1A.dat	2	2,86%	0,00%
val1A.dat	3	5,98%	0,00%
val1A.dat	4	10,84%	0,00%
val1B.dat	2	2,86%	0,00%
val1B.dat	3	6,35%	0,00%
val1B.dat	4	10,80%	0,00%
val1C.dat	2	0,82%	0,00%
val1C.dat	3	3,91%	0,00%
val1C.dat	4	9,44%	0,00%
val2A.dat	2	20,80%	0,00%
val2A.dat	3	28,57%	0,00%
val2A.dat	4	11,48%	0,00%
val2B.dat	2	13,60%	0,00%
val2B.dat	3	7,14%	0,00%
val2B.dat	4	33,88%	0,00%
val2C.dat	2	18,82%	0,00%
val2C.dat	3	6,96%	0,00%
val2C.dat	4	11,20%	0,00%
val3A.dat	2	9,09%	0,00%
val3A.dat	3	8,16%	0,00%
val3A.dat	4	5,08%	0,00%
val3B.dat	2	8,27%	0,00%
val3B.dat	3	7,69%	0,00%
val3B.dat	4	4,50%	0,00%
val3C.dat	2	8,70%	0,00%
val3C.dat	3	5,51%	0,00%
val3C.dat	4	3,90%	0,00%
val4A.dat	2	13,94%	0,00%
val4A.dat	3	12,92%	0,00%
val4A.dat	4	12,66%	0,00%
val4B.dat	2	13,86%	0,00%
val4B.dat	3	12,86%	0,00%
val4B.dat	4	11,97%	0,00%

val4C.dat	2	13,55%	0,00%
val4C.dat	3	12,39%	0,00%
val4C.dat	4	12,37%	0,00%
val4D.dat	2	12,00%	0,00%
val4D.dat	3	11,54%	0,00%
val4D.dat	4	13,19%	0,00%
val5A.dat	2	13,86%	0,00%
val5A.dat	3	12,78%	0,00%
val5A.dat	4	11,93%	0,00%
val5B.dat	2	13,86%	0,00%
val5B.dat	3	12,78%	0,00%
val5B.dat	4	11,93%	0,00%
val5C.dat	2	13,86%	0,00%
val5C.dat	3	12,78%	0,00%
val5C.dat	4	11,93%	0,00%
val5D.dat	2	12,78%	0,00%
val5D.dat	3	13,11%	0,00%
val5D.dat	4	10,92%	0,00%
val6A.dat	2	20,67%	0,00%
val6A.dat	3	17,83%	0,00%
val6A.dat	4	16,62%	0,00%
val6B.dat	2	18,75%	0,00%
val6B.dat	3	17,83%	0,00%
val6B.dat	4	11,83%	0,00%
val6C.dat	2	13,87%	0,00%
val6C.dat	3	15,63%	0,00%
val6C.dat	4	6,15%	0,00%
val7A.dat	2	9,39%	0,00%
val7A.dat	3	9,89%	0,00%
val7A.dat	4	13,16%	0,00%
val7B.dat	2	8,84%	0,00%
val7B.dat	3	10,65%	0,00%
val7B.dat	4	15,70%	0,00%
val7C.dat	2	8,44%	0,00%
val7C.dat	3	9,87%	0,00%
val7C.dat	4	13,75%	0,00%
val8A.dat	2	7,98%	0,00%
val8A.dat	3	9,01%	0,00%
val8A.dat	4	10,33%	0,00%
val8B.dat	2	7,93%	0,00%
val8B.dat	3	10,26%	0,00%

val8B.dat	4	10,30%	0,00%
val8C.dat	2	7,75%	0,00%
val8C.dat	3	10,75%	0,00%
val8C.dat	4	12,92%	0,00%
val9A.dat	2	7,95%	0,00%
val9A.dat	3	9,09%	1,17%
val9A.dat	4	10,61%	0,30%
val9B.dat	2	7,43%	0,00%
val9B.dat	3	9,49%	0,78%
val9B.dat	4	10,94%	0,30%
val9C.dat	2	7,95%	0,00%
val9C.dat	3	8,30%	1,17%
val9C.dat	4	11,21%	0,61%
val9D.dat	2	6,94%	0,00%
val9D.dat	3	7,69%	0,00%
val9D.dat	4	10,99%	0,74%
val10A.dat	2	15,88%	0,00%
val10A.dat	3	23,77%	0,00%
val10A.dat	4	22,40%	0,00%
val10B.dat	2	14,04%	0,00%
val10B.dat	3	22,45%	0,00%
val10B.dat	4	23,90%	0,63%
val10C.dat	2	13,61%	0,00%
val10C.dat	3	20,99%	0,00%
val10C.dat	4	23,10%	0,32%
val10D.dat	2	16,23%	0,00%
val10D.dat	3	22,62%	0,00%
val10D.dat	4	22,22%	0,00%

6. Conclusiones

Nuestro trabajo brinda valores de solución aceptables con una velocidad de mínima de ejecución.

Los aportes de otras investigaciones arrojaron mejores resultados en la mayoría de los casos, pero a costa de un tiempo de procesamiento más elevado.

Nuestra solución así definida podría ser utilizada como buena heurística en sistemas "near real time" para definir las rutas de camiones que estén en pleno recorrido y que ponderen la velocidad de respuesta a costa de resignar un margen de precisión en la optimización.

6.1. Efectividad de la estrategia elegida

La mediana del error promedio entre el resultado óptimo y el devuelto por el algoritmo es de 9.9%.

También como se mencionó anteriormente, el tiempo promedio de ejecución es de 2.53 segundos.

Cabe destacar que en general los resultados óptimos se alcanzan gracias a que la primer fase del algoritmo es muy efectiva. Por el contrario, la fase de búsqueda local no aporta beneficios considerables, probablemente por el concepto de vecindad elegido.

6.2. Posibles mejoras y trabajos futuros

La heurística propuesta aporta soluciones aceptables para la mayoría de los problemas dados, incluso llegando al óptimo y en tiempos prácticamente desestimables. Aún así todavía existe un margen de mejora para gran cantidad de casos, por lo que sería interesante buscar alternativas en la fase de búsqueda local, para llegar al resultado exacto sin detrimento de la efectividad temporal obtenida.

7. Referencias

[Lap92] G. Laporte. "The vehicle routing problem: An overview of exact and approximate algorithms". European Journal of Operational Research. 59:345-358 (1992).

[Nob87] G. Laporte, Y.Nobert. "Exact algorithms for the vehicle routing problem". Annals of Discrete Mathematics, 31:147-184 (1987).

[Tot00] P. Toth, D. Vigo. "The Vehicle Routing Problem". SIAM monographs on discrete mathematics and applications (2000).

[Vig98] P. Toth, D. Vigo. "Exact algoritms for the vehicle routing". Fleet Managament and Logistic, T. Grainic and G. Laporte, editors, Kluwer, Boston, MA, pp. 1-31 (1998).

[TotVi] P. Toth, D. Vigo. "Models, relaxations and exact approches for the capacitated vehicle routing problem". Discrete Applied Mathematics, Pages 487-512 (2002)

[Bod83] L.D. Bodin, B. L. Golden, A A Assad, and M. Ball. "Routing and scheduling of vehicles and crews, the state of art". Computers and Operations Research ,10:63-212, (1983).

[Chr85] N. Christofides. "Vehicle routing". In "The Traveling Salesman Problem", E.L. Lawer, J.K. Lenstra, AH.G. Rinnooy Kan and D.B. Shmoys, editors, Wiley, Chichester, UK, pp 431- 448, (1985).

[Min79] N. Christofides, A Mingozzi and P. Toth. "The vehicle routing problem". In "Combinatorial Optimizations", N. Christofides, A Mingozzi, P. Toth and C. Sandi, editors, Wiley, Chichester, UK, pp 315-338, (1979).

[Hel00] K. Helsgaun. "An effective Implementation ofthe Lin-Kernighan Traveling Salesman Heuristic". European Journal of Operational Research, Pages 106-130 (2000).

[Joh95] D. S. Johnson, L. A. Mc Geoch. "The Traveling Salesman Problem: A Case Study in Local Optirnization", (1995).

[Arc10] C. Archetti, D. Feillet, A. Hertz, M. Grazia Speranza, "The Undirected Capacitated Arc Routing Problem with Profits". Computers & Operations Research 37, 1860—1869, (2010).

[Zac11] E.E. Zachariadis, C.T. Kiranoudis, "Local Search For The Undirected Capacitated Arc Routing Problem With Profits", European Journal of Operational Research 210, 358-367 (2011)

[Cur13] Tunchan Cura, "An artificial bee colony approach for the undirected capacitated arc routing problem with profits," International Journal of Operational Research, Inderscience Enterprises Ltd, vol. 17(4), pages 483-508 (2013).

[Qui12] JIN Qian-Qian, LIN Dan, "Variable Neighborhood Search Algorithm for Solving UCARPP Problem", Computer Engineering, 38(21): 290-292 (2012).

[Feo95] T. A. Feo, M. G. C. Resende, "Greedy Randomized Adaptive Search Procedures". Journal of Global Optimization, March, Volume 6, Issue 2, pp 109–133 (1995).

[Res16] M.G.C. Resende, C. Ribeiro. "Optimization by GRASP". Greedy Randomized Adaptive Search Procedures, Springer (2016).

[Flo62] R. W. Floyd, "Algorithm 97: Shortest Path". Communications of the ACM 5 (6): 345 (1962).