Trabajo de Fin de Grado
Grado en Ingeniería de Tecnología Industrial

IMPLEMENTACIÓN DE UNA HEURÍSTICA PARA EL CÁLCULO DE MILK RUNS

Autor: Haoyu Ji

Tutor: Jesús Muñuzuri Sanz

UNIVERSIDAD DE SEVILLA

Trabajo de Fin de Grado

Grado en Ingeniería de Tecnología Industrial

IMPLEMENTACIÓN DE UNA HEURÍSTICA PARA EL CÁLCULO DE MILK RUNS

Autor:

Haoyu Ji

Tutor:

Jesús Muñuzuri Sanz

Departamento de Organización Industrial y Gestión de Empresas II

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2021

Trabajo de Fin de Grado: IMPLEMENTACIÓN DE UNA HEURÍSTICA PARA EL CÁLCULO DE MILK RUNS

Autor: Haoyu Ji

Tutor: Jesús Muñuzuri Sanz

El tribunal nombrado para juzgar el Trabajo arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

RESUMEN

Just In Time es una filosofía de gestión y de organización que la empresa japonesa Toyota ha introducido en el mundo, en este trabajo, se quiere estudiar la posibilidad de aplicar la herramienta de Lean Manufacturing denominada Milk Runs, que consiste en recoger las mercancías totales de distintas categorías desde distintos proveedores hasta la fábrica en lotes pequeños, y planificar las rutas de recogidas en una planificación semanal para las paradas que haya que hacer en cada proveedor dependiendo de la demanda y el coste de la mercancía.

Para conseguir este objetivo, primero se enumeran las ventajas de la filosofía Just In Time y Lean Manufacturing, la definición de la herramienta Milk Runs. Una vez definida, se explicará el procedimiento que se sigue para la aplicación de la herramienta con el pseudo-algoritmo y por último el algoritmo en softwares de programación que se usan para conseguir la planificación.

Con los algoritmos de Milk Runs y de la heurística de ahorro, se resuelve primero el ejemplo que viene en el artículo publicado en Journal of Operations Management: "Reducing Vendor Delivery Uncertainties in a JIT Environment" para asegurar de la efectividad de los algoritmos y después se resolverá varios problemas de ejemplo de mayor dificultad aumentando el número de proveedores y número de categorías de productos, obtener resultados de ellos aplicando los algoritmos. Y haciendo observaciones a los resultados que se obtienen con los algoritmos, se sacan las conclusiones de este trabajo.

ÍNDICE DE CONTENIDO

ÍNDICE DE CONTENIDO

Índice de figuras

Índice de figuras

Índice de tablas

1.Introducción y objeto del Trabajo

Just in time, conocido también como método Toyota, es una filosofía de gestión y de organización que tiene su origen en la empresa japonesa de fabricante Toyota, para gestionar los suministros de piezas de su producción y planificar la producción de su fábrica con el fin de optimizar el proceso de fabricación.

El método Toyota consiste en que la producción solo se planifica en el momento que hay un pedido real y que los materiales para producir estos pedidos lleguen justo a tiempo, poco antes que la producción y solo con las cantidades necesarias. De esta forma, en la fábrica no habrá ni stock de piezas ni stock de productos finales, reduciendo al máximo el coste de mantenimientos para cubrir las demandas de producciones.

Con estas gestiones en la organización y en la planificación, se reduce el coste de mantenimiento de la materia prima y de producto final para la fábrica ya que solo se almacenan lo justo para la producción y justo antes de la producción, aumenta el margen de la producción reduciendo el coste. El método fue tan exitoso que llevó a la empresa japonesa a ser una de las empresas más valorada en la historia de automoción, actualmente la empresa Toyota sigue siendo sinónimo de coches de calidad, poca avería, alto valor residual, etc., y es la empresa de automoción con el mayor margen de beneficio del mundo.

Después de ver el éxito del método Toyota, las otras empresas, no solo del sector de automoción, sino empresas de todos los sectores empiezan a replicar esta eficiente filosofía para la gestión y organización de sus empresas, con el fin de reducir sus gastos de la empresa, así aumentar la competitividad de su empresa en el sector y en una sociedad cada vez más exigente para las empresas.

Figura 1 Fabricante TOYOTA

Fuente: https://es.slideshare.net/AbelBryanIamaguaPaz/just-in-time-45183126

El método Toyota consiste en eliminar todos los desperdicios o despilfarros que se observan en una fábrica, estandarizar los procesos, mantener los procesos y siempre con la intención de mejora continua a la fábrica. El desperdicio o despilfarro se define como:

"cualquier cosa distinta de la cantidad mínima de equipamiento, materiales, partes, espacio y tiempo, que sea absolutamente esencial para añadir valor al producto".

Por lo tanto, cualquier proceso, acción, objeto, movimiento, gestión, etc., que no añade valor a la producción, son considerados como innecesarios y siempre se intenta eliminarlos.

Hay muchos tipos de desperdicios, por ejemplo, mantener unos niveles de stocks innecesarios de materia prima para asegurar la producción, de productos finales para cumplir los tiempos de entregas, producción con muchos tiempos muertos, defectos en los productos, exceso de personales de gestión, logísticas ineficientes, etc. Todos estos desperdicios no solo que no añade valor a los productos, sino que muchas veces generan costes innecesarios para la empresa, así como reducir nuestros márgenes de beneficio en el producto final, disminuyendo la competitividad de la empresa frente a otras empresas del mismo sector que sí se han eliminado estos despilfarros.

Con el fin de eliminar todos estos despilfarros, el método Toyota se basa en aplicar las herramientas útiles de Just In Time para eliminar todas aquellas actividades o acciones que no son absolutamente esenciales para añadir valor al producto, la herramienta MILK

RUNS en estudio de este trabajo es precisamente una de estas herramientas útiles para eliminar desperdicios en la logística.

Una de las críticas hacia el sistema Just In Time es sobre la eliminación de los stocks de materia prima o piezas para la producción, se creen que esa eliminación en realidad no se ha conseguido para la fábrica que aplica Just In Time, sino que estos stocks solo se han trasladado desde la fábrica a los proveedores de esa pieza en la cadena de suministro de la fábrica, ya que los proveedores, con el fin de satisfacer las demandas en pequeños lotes de la fábrica, tienen que guardar un nivel de stock más alto que si la demanda de la fábrica fuese en grandes cantidades en un plazo más largo. Esto solo es cierto si el proveedor no aplica el sistema de Just In Time, por lo que, para responder a una demanda con variación en corto plazo, lo único que puede hacer es el aprovisionamiento de un nivel alto de materia prima para asegurar la producción en cualquier momento. Si el proveedor también aplicase el sistema Just In Time, no tiene la necesidad de mantener un stock alto de materias primas para producción. Además, puede aprovechar de que el fabricante tiene una demanda estable y segura hacia él, siempre avisan ante cualquier variación en la necesidad de demanda, comunicación con el fabricante si hubiese alguna necesidad de cambio en especificaciones, averías, etc.

Hoy en día, ningún fabricante industrial fabrica todos los componentes de sus productos, y menos todavía para un fabricante de coches, por la complejidad y el alto número de componentes de sus productos, si un una empresa como Toyota o Volkswagen se dedicase a desarrollar y mejorar cada uno de los componentes de sus coches, se gastarían muchos dineros y recursos humanos en investigación y desarrollo, y el resultado sería que su coche no es competitivo contra coches de otras grandes compañía de automoción, la causa es muy simple, su margen de beneficio se reduce muchísimo contra las otras compañías de coche que solo se dedican al diseño del coche y colaboran con una cadena de suministro integral para la producción de sus coches, aumentando así la competitividad de sus productos, cada proveedor se dedican a una sola pieza, con esa especialización se consigue producir con la máxima calidad y mínimo coste de producción por la economía de escala.

Por eso, todas las grandes compañías industriales cuentan con numerosos proveedores de todos los componentes distribuidos en zonas más o menos cercanos a su instalación, la selección de la red de proveedores se convierte en una tarea de vital importancia para las fábricas industriales desarrollar sus actividades de producción y satisfacer las demandas de los pedidos con calidad y en plazos de entrega requeridos por el mercado, sacando el máximo beneficio posible.

La selección de proveedores para las grandes empresas, no solo se basa en la relación de precio-calidad como la gente piensa, sino que hay que valorar muchísimos aspectos más tales como la proximidad del proveedor a la instalación, si se puede incorporar en una ruta razonable de la red de proveedores actuales, si los tamaños de lotes que producen, y

los lotes que tienen para el transporte, etc. Todos estos aspectos, que al final se traduce en coste de la fabricación de los productos, son muy importantes a la hora de seleccionar un proveedor y convertirlo en un proveedor certificado, pero aún más importante es establecer una relación de confianza digna entre el fabricante industrial y los proveedores, y no simplemente una relación de comprador y vendedor.

La confianza con los proveedores es un factor imprescindible para aplicar el sistema de Just In Time con éxito, si no se establece una relación de confianza con determinado proveedor, aunque todos los demás aspectos de este proveedor cumplen con los requisitos de la fábrica, no es recomendable trabajar a largo plazo con él, porque esta desconfianza podría provocar graves problemas para el funcionamiento de la fábrica en determinadas ocasiones y traer muchas pérdidas e incluso la quiebra para la fábrica, hay que tener mucho cuidado a la hora de tomar la decisión de con cuáles proveedores trabajar en base a todos los factores y mantener una relación a largo plazo con el proveedor si resulta que se puede establecer una relación de confianza entre fabricante-proveedor y cumple todas las exigencias requeridas, aumentar tanto la influencia de fabricante como la influencia de todos y cada uno de los proveedores de la cadena de suministro, crecer juntos en el mercado. No es bueno cambiar con frecuencia de proveedores, al contrario, hay que establecer una relación cada vez más profunda, como por ejemplo intercambio de acciones entre el fabricante y los proveedores, una forma muy habitual en el mundo de automoción para establecer una relación que durará muchos años.

Una vez establecida una red de proveedores, la primera cuestión que se enfrentan y requieren una solución, es la logística entre proveedores y el fabricante, ¿quién se encarga de llevar las piezas de los proveedores a la fábrica?

Figura 2 Transporte de mercancías

Fuente: http://laclasedeoscarboluda.blogspot.com/2015/02/los-costes-y-el-calculo-del-transporte.html

Introducción y objeto del Trabajo

Tradicionalmente, el proveedor es quien se encarga de la entrega hacia la fábrica, y luego incurrir este gasto a la fábrica con una factura por el servicio del transporte realizado.

Al tener una cadena de suministro de alto número de proveedores y entre ellos no se conocen, a eso se suma a que queremos aplicar sistema Just In Time de gestión y de organización en la fábrica, la fábrica tiene la necesidad de hacer pedidos con una mayor frecuencia y pedidos de cantidades menores, por los que la empresa tiene que asumir todos los meses un alto coste por el servicio de transporte realizado por los proveedores en cada pedido que se hace.

Además de generar altos costes, genera también una incertidumbre en el seguimiento de estos transportes. Si el proveedor es el responsable de la logística, la fábrica no puede tener un control sobre el transporte directamente, en lugar de eso, tiene que lanzar una consulta de estado a los proveedores, y en muchos casos, esta consulta requiere un gran retardo para obtener la respuesta. Con lo cual, en caso de que se produzca una incidencia, la fábrica no tiene ningún tiempo de reacción para minimizar el efecto que trae esa avería, en un caso extremo, se podría generar una parada en la producción, retraso en la entrega al cliente, perder la confianza del cliente y mantener un stock inútil de otras piezas entregadas por los otros proveedores.

Para evitar estos altos costes y tener un mayor control sobre la logística de suministros, el sistema Just In Time propone para la fábrica, que la logística de la cadena de suministro sea controlada y realizada por la fábrica directamente.

Con esta acción, se quiere reducir el gasto de transporte de la fábrica, tener un mayor control sobre la logística de las mercancías a fábrica y disminuir el gasto de almacenaje haciendo pedidos recurrentes de pequeñas cantidades, aplicando un algoritmo de ahorro para organizar la planificación de recogida de todos los proveedores en una ruta optimizada con criterio del algoritmo de Clarke Wright o también llamado algoritmo de ahorro. La ruta no es una ruta simple con solo dos paradas, salida de la fábrica hacia un proveedor, se carga la mercancía, y se vuelve a la fábrica, sino, es una ruta llamada MILK RUNS, un término inglés que su traducción directa es ruta lechera, la ruta se asimila con la ruta que realiza un lechero para entregar leches a un número alto de casas localizadas en una determinada zona, y que los leches sean entregados en una misma ruta.

En este trabajo, se quiere estudiar la efectividad para una fábrica con un número considerable de proveedores de aplicar el algoritmo de Clarke Wright en una ruta lechera, con una única restricción de la capacidad de camiones que tiene la fábrica para el transporte de las mercancías necesarias recogiéndolas desde las instalaciones de los proveedores, sabiendo las ubicaciones de todos los proveedores y las demandas de mercancías de cada proveedor.

El resultado que se debe obtener será el número de camiones que hacen falta, y la distribución de los distintos proveedores en las distintas rutas minimizando la suma de las distancias totales de todas las rutas, cumpliendo la restricción de la capacidad en cada una de las rutas. En otras palabras, se quiere estudiar la efectividad del algoritmo de ahorro en los problemas de problema de ruteo de vehículos con capacidad limitada de carga.

2. Herramienta Lean: el sistema de MILK RUNS o Rounds

2.1 MILK RUNS o Rounds

Lean Manufacturing es un término inglés, su traducción directa sería manufactura esbelta, es una filosofía de gestión de origen japonés concebido por el director y consultor de Toyota: señor Taiichi Ohno, uno de los gigantes en la industria de automoción y padre creador del método Toyota de producción en los años 70 del siglo XX.

Después de la Segunda Guerra Mundial, para ayudar a aumentar la productividad de las fábricas japonesas, que en aquel periodo era muy inferior a la productividad americana, y con la intención de aprender de las fábricas americanas, Taiichi Ohno visitó las grandes fábricas americana pioneras en productividad de aquella época como la fábrica de coche de Henry Ford.

Figura 3 Padre de método Toyota Taichi Ohno

Fuente: https://qrius.com/remembering-titans-taiichi-ohno-the-father-of-toyotas-revolutionary-production-system/

Taiichi Ohno se quedó muy impresionado por el empeño que las fábricas americanas ponían en la producción masiva, filosofía de producción masiva y en una línea de ensamblaje introducida por Henry Ford para su fábrica de coches Ford en el siglo XX,

con el fin de bajar enormemente el coste de los coches fabricados haciendo lotes masivas de producción, desde entonces, los coches dejaron de ser un objeto de lujo, las personas de clase media también pueden adquirir coches para la familia, pueden ir a viajes largos evitando trenes, barcos o aviones.

En las fábricas americanas fabricaban todos a grande escala sin considerar los pedidos reales (anticipar las demandas para un plazo de entrega en muy corto tiempo), por lo que, en esas fábricas, se limitaban en la variedad del producto y generando muchos stocks en el almacén.

Estos fenómenos que observaba Taiichi Ohno en las fábricas americanas, para él eran desperdicios en la industria de las fábricas, no quería aumentar la producción de esta manera y empezó a investigar y pensar para encontrar una solución que aumentase la productividad evitando estos desperdicios en variedad y alto nivel de stock.

Durante una compra en un supermercado después de la visita a las fabricantes americanas, Taiichi Ohno tuvo una brillante inspiración observando a los supermercados para mejorar los procesos de producción en las fábricas de Toyota en japón, la inspiración consiste en que hay que eliminar todos aquellos procesos innecesarios mediante las herramientas que él inventó basándose en la teoría de Lean manufacturing que desarrolló.

Con estos principios, Taiichi consiguió implementar un método de producción mucho más eficiente en la fábrica de Toyota, y que este método posteriormente fue denominado método Toyota porque fue la primera fábrica que lo aplicaba y tuvo muchos éxitos globalmente.

Esta filosofía de Lean manufacturing consiste en eliminar mediante las herramientas útiles de Lean a todos los desperdicios o despilfarros en los procesos de fabricación de un producto, mejorando así la calidad de producción y reduciendo costes de las fábricas, por lo tanto, aumenta enormemente el margen de beneficio de todos los productos que se fabrican.

Hay en total 8 tipos diferentes de desperdicios o despilfarros:

- Sobreproducción: queriendo reducir el coste de cada pieza, hacen una producción masiva cuando no hay pedido real de ese componente para cubrir esa producción.
- Retrabajo: hacer trabajo extra provocado por no realizar bien algún proceso de fabricación, falta de stock mínimo o tener demasiado stock, etc. Todos estos defectos pueden provocar duplicar tarea o tareas que, en una producción eficiente, solo se realiza una vez.
- Transporte: movimiento innecesario de los materiales entre operaciones.
- Defectos: acciones correctivas que se realiza para corregir al producirse una pieza defectuosa.

- Inventario: mantener un nivel de stock alto, desperdicia espacio y recurso para gestión de inventario.
- Espera: tiempos muertos que aparecen en los cambios de operaciones durante la fabricación.
- Movimiento: movimientos de materiales o humanos que no son necesarios para la producción.
- Conocimientos desconectados: no hay una comunicación efectiva entre la fábrica con sus clientes y/o sus proveedores.

Todos estos desperdicios son objetivos del método que Taiichi Ohno quería aplicar en la fábrica de Toyota, para eliminar estos desperdicios, se inventó distintas herramientas Lean que permite evitar la aparición de estos fenómenos, entre las herramientas, la más conocida es la técnica de 5S, que consiste en mejoras continuas en el nivel de organización, limpieza y orden, su nombre proviene de 5 palabras japonesas que empieza por la letra S:

- seiri: subordinar, clasificar, descartar
- seiton: sistematizar, ordenar
- seiso: sanear y limpiar
- seiketsu: simplificar, estandarizar y volver coherente
- shitsuke: sostener el proceso, disciplinar

Otra de las herramientas Lean es el denominado Milk Run o Round, ruta lechera inventada para optimizar el transporte de mercancías de una fábrica con sus proveedores cercanos.

La herramienta Lean de Milk Run/round es un término logístico para describir el proceso de entregar botellas de leche y recoger las botellas vacías de una zona en un camino largo, en vez de mandar un transportista cada vez que haya una petición de una botella de leche, es una forma de satisfacer las demandas muchas más razonables y disminuye la distancia total recorrida por el lechero, y la distancia total va directamente vinculada con el coste de transporte o coste del lechero para repartir y satisfacer las demandas de las casas.

Herramienta Lean: el sistema de MILK RUNS o Rounds

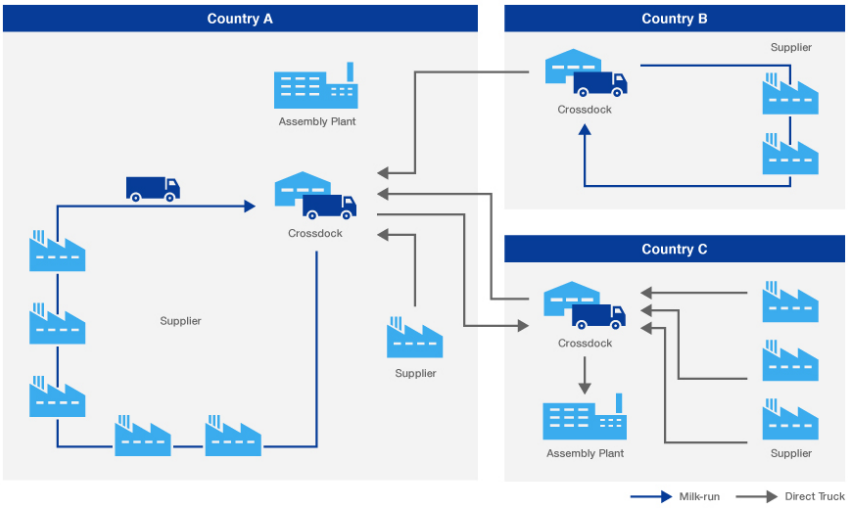

Figura 4 ilustración de Milk Runs

Fuente: https://www.yusen-logistics.com/en/resources/case-studies/milk-run-crossdock

El transporte es un coste que no añade valor directamente al producto en la mayoría de los casos en el mundo de la industria, sin embargo, es uno de los servicios más imprescindibles, porque cuando la diferencia es pequeña entre los productos que tiene características y cualidades similares, el servicio de transporte pasa a ser un factor determinante para que el cliente elige una marca u otra. Por eso, es de vital importancia la optimización en transporte para una empresa si se quiere destacar entre productos similares.

Esta misma filosofía se puede aplicar también en las fábricas industriales, hoy en día, ninguna fábrica industrial produce todas las piezas de un producto en cuanto el número de las piezas supera unas determinadas cantidades, uno de los ejemplos más claros son las fábricas de coches, que cuenta con numerosos proveedores de piezas, se llevan esas piezas a la fábrica, donde se procede al ensamblaje de los coches en una línea de montaje. Hay dos formas de traer las piezas que se necesitan:

1. Que los proveedores se encargan de traer las piezas en cantidades que se ha pedido por la fábrica.
2. Que la fábrica se desplace hasta cada uno de los proveedores y se traen las piezas en cantidades que necesita la fábrica.

¿Cuáles son las diferencias entre estas dos formas de traer las piezas para una fábrica?

Tradicionalmente, la forma que utilizan las fábricas es la primera, se hace un pedido con la cantidad de determinada pieza que se necesitan para la fabricación de productos, esa cantidad suele ser grande para cubrir demandas de un tiempo largo, evitar así tener que pedir en cada lote de producción.

La principal ventaja de esta forma de suministro es que la fábrica tiene menos tareas, no se tiene que preocupar del transporte salvo llevar un seguimiento de fecha de llegada de las piezas de los distintos proveedores. ¿Entonces esta es la mejor forma para la fábrica? Suena bien no tener que hacer gestión.

Sin embargo, también tiene desventajas, las principales son:

1. Los proveedores se encargan de traer las piezas a la fábrica, pero este servicio no es gratis ni mucho menos, en las facturas que traen junto con las piezas, hay un apartado que pone coste de envío, dependiendo de la distancia de cada proveedor a la fábrica, este coste de envío será más alto o más bajo.
2. Para no tener que pedir piezas a los proveedores, se piden piezas para varios lotes de producción, generando costes de almacenamiento y de gestión por guardar esas piezas si al final no se han llevado a la producción.
3. Al no llevar el transporte, la fábrica no tiene un control sobre la entrega de los proveedores si no se lanza una consulta a ellos, y en el mundo de transportes, es muy frecuente de que se retrasa algún proveedor, puede incluso afectar a nuestra producción sin es una pieza que no tenemos en stock.

Estas desventajas van en contra con la filosofía de Lean Manufacturing, todos estos son despilfarros o desperdicios que se intentan eliminar mediante las herramientas de Lean Manufacturing.

Por eso, en las fábricas que se aplican la filosofía Lean, para evitar estas desventajas, se utilizan la segunda forma para traer las piezas a la fábrica, que la fábrica sea quien se desplace a las instalaciones de los proveedores a traer las piezas que necesitan para la producción.

Pero lo que se hacen no es desplazar hasta un solo proveedor, sino que se aplica la herramienta Lean de Milk Run, se aprovecha toda la capacidad de carga que tiene el vehículo para planificar un ruteo de varios proveedores a la vez, de forma que traer todas las piezas que necesitamos haciendo la menor distancia posible.

Las ventajas de transportar mercancías de esta forma es el aprovechamiento máximo de los recursos: en una misma ruta podemos satisfacer la recogida de las demandas de varios proveedores, en un solo vehículo compartido para recoger mercancías desde los almacenes de los proveedores, evitando así recorridos largos desde el almacén a un determinado proveedor con una sola demanda, por lo tanto:

- la ventaja uno es el aprovechamiento del camión con mayor eficiencia;
- la segunda ventaja es la posibilidad de reducir el coste de almacenamiento, si la fábrica se encarga de recoger la mercancía aprovechando toda la capacidad del camión en una misma ruta en vez de que cada proveedor nos suministre, no hay que pedir grandes cantidades para cubrir varios lotes de producción, sino que solo

la cantidad necesaria para producir exactamente la cantidad de pedidos reales que se ha recibido, evitando así tener que guardar las piezas sobrantes en la fábrica.

- La fábrica se conoce en todo el momento durante el transporte cualquier incidencia que surja durante desde el almacén del proveedor hasta la fábrica, solamente hay que asegurar que cuando el camión de la fábrica llega a los almacenes de proveedores, los proveedores tienen el pedido preparado para cargarlo en el camión, para eso, se avisa a los proveedores antes de salir a la ruta, pedir la mercancía con antelación a los proveedores y una vez confirmada por parte de los proveedores, se hace la ruta.

De esta forma, con una buena planificación, la fábrica puede conseguir un buen ahorro en coste para el transporte y para almacenaje, estos ahorros se convertirán en un mayor margen de beneficio para la fábrica. Con un simple cambio de estrategia, se consigue que se aumente la competitividad de nuestra empresa en el sector pudiendo disminuir nuestro coste de producción frente a otros fabricantes que no aplican la herramienta Lean de Milk Runs.

Existe una consideración importante que la fábrica se tiene que tener en cuenta, y es que para aplicar esta herramienta, será imprescindible de conocer de antemano una condición o información: el conocimiento de todas las informaciones relativas al transporte, como por ejemplo la situación geográfica de todos los proveedores de mercancía, las mercancías necesarias para la producción de una semana para cubrir la demanda en números de pedidos reales que se ha producido, la capacidad de carga de cada vehículo en la flota de la fábrica etc.

La otra condición importante es la fiabilidad de los proveedores sobre la fecha de entrega de sus productos. Al tener las rutas hechas antes de salir, si falla algún proveedor, a la fábrica no le da tiempo para corregir rutas durante la ruta, por lo que se puede generar costes extras para la fábrica en algunos proveedores que no han cumplido su plazo de entrega y estropea la factibilidad de aplicar la herramienta Lean de Milk Runs, cada retraso se convierte en una ruta extra para la fábrica.

El problema de Milk Runs en la vida real es muy complejo, la fábrica tiene que tener en conocimiento muchos datos para decidir las rutas de cada día, no solo la geolocalización de los proveedores, sino que se tiene que tener en cuenta todos los factores que influyen en la planificación:

- Demandas de la fábrica de cada producto que se compran a los proveedores
- Plazos de entrega de los proveedores en preparar estos productos
- Ventanas o horarios de los proveedores
- Tiempos de carga y descarga estimados
- Capacidades de los vehículos para el transporte de la fábrica

- Kilometrajes que se permiten conducir a los conductores durante un día
- Etc.

Es un problema que puede llegar a tener una magnitud de complejidad muy alta, y además, cuando mayor sea el número de proveedores, más complejidad tiene para resolverlo. A eso se les suma a los fenómenos no previstos que ocurren en la realidad, que añade otra dificultad extra al problema, todos estos hacen que los problemas de este tipo sean imposibles de resolver con exactitud.

2.2 Descripción del problema

El problema que se enfrentan, es un problema que Arthur V. Hill y Thomas E. Vollmann estudiaron en 1986, quienes publicaron el problema en Journal of Operations Management: "Reducing Vendor Delivery Uncertainties in a JIT Environment".

El problema que se querían resolver es algo que cualquier fábrica con numerosos proveedores que le suministran piezas todos los días para producir sus lotes de producciones: sabiendo que hay que comprar una serie de mercancías (demandas) a un número determinado de proveedores, en un periodo definido, con la flota de uno o varios vehículos que disponen para el transporte de esas mercancías, minimizando el incertidumbres, la solución a este problema consiste en encontrar una planificación de rutas en periodo de una semana, con los recursos disponibles que satisface las demandas de la fábrica para traer todas las mercancía que necesita y que cumplir las limitaciones.

La complejidad de este tipo de problema de planificación de rutas únicamente depende de las limitaciones que se quieren poner y el número de proveedores de la cadena de suministro de la fábrica.

Para este trabajo, se quiere resolver el problema de Milk Runs (rutas lecheras), con la única restricción de capacidad de los vehículos destinados a las recogidas de mercancías de los proveedores usando un algoritmo determinado. Ese algoritmo permite obtener planificaciones de rutas que en la suma recogen todas las mercancías necesarias para la producción de la fábrica, la distancia total no es primer criterios a tener en cuenta, sino se basa en calcular una variable que decide el número de paradas de todos los proveedores y distribuir estas paradas en distintos días de la semana.

Después de esto, se aplica el algoritmo de ahorro para optimizar las rutas de cada día de la semana.

El problema de planificación de rutas con la única restricción de la capacidad de los vehículos destinados se llama Problema de Enrutamiento del Vehículo con Capacidad (Capacitated Vehicle Routing Problem CVRP).

Los problemas de enrutamientos de vehículos son de tipo NP-duro de optimización combinatoria, ya que para encontrar la solución óptima de n proveedores, existe en total factor de N posibles soluciones (N!), por lo tanto, cuando el número de proveedores implicados en el problema excede una determinada cantidad, el tiempo de cálculo aumenta exponencialmente también, si el número de proveedores es muy grande, con la capacidad de cálculo de los ordenadores, no será posible usar este método exhaustivo para obtener la solución óptima.

Las implementaciones que normalmente se utilizan para resolver este tipo de enrutamiento de vehículos son las implementaciones heurísticas, que son procedimientos simples que no hacen una búsqueda exhaustiva, sino que exploran un espacio de búsqueda de una forma limitada. Con las implementaciones heurísticas, no se puede asegurar que la solución encontrada es la óptima, pero con un corto tiempo de ejecución, se encuentra una solución factible aceptable.

Las heurísticas más utilizadas para el problema de enrutamiento de vehículo son:

- El Algoritmo de Ahorros -
- Heurísticas de Inserción -
- Métodos de asignación elemental -
- Métodos Asignar Primero – Rutear Después -
- Método Rutear Primero – Asignar Después -
- Algoritmos de Pétalos -
- Procedimientos de Búsqueda Local -

En este trabajo, se utiliza el algoritmo de ahorros para resolver el problema de enrutamiento de vehículos con capacidad mediante la capacidad de cálculo de los ordenadores.

Para que Matlab, Python o cualquier programa que nos ayuda para resolver problemas, primero hay que definir los problemas en un modelo matemático. En el siguiente capítulo, se explica la forma de definir el problema y la resolución.

3.Descripción del algoritmo de Milk Runs y de la heurística método de ahorro

Las máquinas no entienden nuestro lenguaje, para que calculen la solución del problema de enrutamiento de vehículos, hay que traducir el problema y el algoritmo de heurística al lenguaje de la máquina. Ese lenguaje, dependiendo de cuál programa utilizar, será de una forma u otra.

Pero dentro de cualquier lenguaje de máquina, el razonamiento del algoritmo es el mismo, solo que sus formas de expresar la lógica son distintas entre los distintos programas. Por lo tanto, lo más importante es entender el razonamiento del algoritmo del método de ahorro de Clarke Wright.

3.1 Razonamiento

En cualquier heurística, los primeros pasos son iguales, consiste en sacar una solución factible que cumpla todos los requisitos, y luego se utiliza los procedimientos de una heurística para mejorar la solución.

El algoritmo genérico que se utiliza normalmente para resolver problemas de enrutamiento se basa en calcular el número óptimo de recogida mediante una fórmula basada en la fórmula de costes siguiente:

TIC= coste incremental total= coste de almacenaje + coste de transporte + coste de carga y descarga =

$$\sum_{k}[0{,}5 * h * \frac{\sum_{i \in P(k)} d(i) * c(i)}{n(k)} + VVC * dt(k) * n(k) + \sum_{i \in P(k)} s(i) * n(k)]$$

Calculando el diferencial de TIC respecto a *n(k)*, se obtiene que la solución óptima de *n(k)* sea de la siguiente forma:

$$n(k)^* = fn[dt(k)] = [\frac{0{,}5*h*\sum_{i\in P(k)} d(i)*c(i)}{\sum_{i\in P(k)} s(i)+2*VVC*dt(k)}]^{\frac{1}{2}}$$

Donde:

n(k) = número de paradas de cada proveedor

dt(k)=es la función diferencial del tiempo de viaje requerido para hacer una parada adicional en el proveedor k

d(i)= demanda del producto i

c(i)= coste del producto i

h= recargo de transporte por unidad por semana

s(i)= coste de preparación para mover producto i

VVC= coste variable del vehículo

P(i, k)= Los productos que el proveedor k suministra. Si el producto i es suministrado por el proveedor k, *P(i, k)*=1, en caso contrario, *P(i, k)*=0

Una vez calculada el número de recogida en cada proveedor, se asigna estas recogidas en los días de semana y se obtiene una solución inicial, con el método de CLARKE WRIGHT HEURISTIC, denominado también el método de ahorro o algoritmo de ahorro, se mejora esta solución inicial.

3.2 Pseudocódigo

3.2.1 Milk Runs

Este razonamiento se lleva a un pseudo-algoritmo que la siguiente tabla de flujo de algoritmo muestra:

Paso 1: Leer los datos

En este paso, se recogen todos los datos iniciales, las localizaciones de los proveedores, las demandas que se necesitan de los proveedores, también los datos de restricciones como capacidad de los camiones, tiempo de descarga, las ventanas de tiempos de los proveedores, limitación de kilometraje por el tacógrafo, etc.

Paso 2: Inicialización

Para este razonamiento, es crítico calcular la función de tiempo de viaje extra por hacer una parada en un determinado proveedor, pero como no se puede saber con exactitud el tiempo de viaje diferencial requerido porque no se sabe la ruta planificada antes de la planificación, hay que estimar este variable de alguna manera. Se sabe que el valor del tiempo de viaje diferencial por hacer una parada en un proveedor oscila entre la máxima, que es una ruta solo para ese proveedor, distancia desde la fábrica a la instalación del proveedor multiplicado por dos, y la

mínima, que es la distancia del proveedor hasta el otro proveedor más cercano más la distancia del proveedor hasta el proveedor segundo más cercano. Luego:

$$dtmx(k) = t(0,k) + t(k,0)$$

Siendo *t(i,j)* igual a la distancia desde punto i hasta el punto j, y 0 representa el punto donde está la fábrica.

$$dtmn(k) = \min\,[t(j,k) + t(k,l)]$$

Para todos los j y l donde j ≠ l.

Con estos dos datos, se saca el valor umbral de *n(k)* con la fórmula de *n(k)**:

$$mnn(k) = fn[dtmx(k)]$$

$$mxn(k) = fn[dtmn(k)]$$

Con este rango de valor máximo y mínimo de *n(k)*, se crea una variable paramétrica alpha para controlar la oscilación de tiempo de viaje requerido en el siguiente paso.

Paso 3 Calcular *n(k)* de cada proveedor en base a los tiempos de viajes

$$n(k) = mnn(k) + alpha * (mxn(k) - mnn(k))$$

Con esta ecuación paramétrica con variable Alpha, conseguimos probar todos los valores de *n(k)* entre el umbral máximo y mínimo, cuando alpha vale 0, *n(k) = mnn(k)*, cuando alpha vale 1, *n(k)= mxn(k).*

Paso 4: asignar las recogidas de cada proveedor a los días de la semana

Si *n(k)* ≥ 5, asignar recogidas cada día de la semana y poner *n(k) = n(k)* – 5

Si *n(k)* =4, asignar recogidas lunes, martes, jueves y viernes

Si *n(k)* =3, asignar recogidas lunes, miércoles y viernes

Si *n(k)* =2, asignar recogidas martes y jueves

Si *n(k)* =1, asignar recogidas miércoles

A priori, esta repartición se puede parecer irrazonable, pero si se analiza las veces que aparecen cada día de la semana, se da cuenta de que cada día de la semana aparece tres veces, con eso se quiere conseguir que el reparto sea lo más homogéneo posible.

Hasta este paso, se obtiene las primeras rutas de cada día de la semana, rutas que aplicamos el método de ahorro para optimizarlas. El último paso es la aplicación de método de ahorro.

3.2.2 Método de ahorro

El método de ahorro es uno de los métodos heurísticas más frecuentes para resolver el problema de enrutamiento de vehículos.

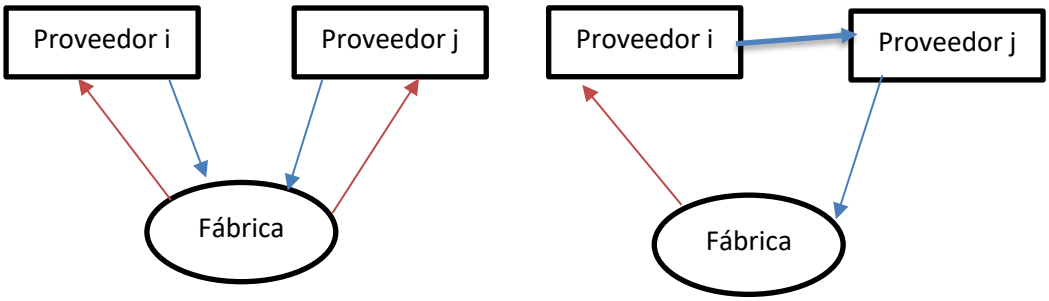

Figura 5 Agregar varios proveedores en una ruta

Se divide en tres pasos:

Paso 1: Encontrar una solución factible

La hemos encontrado con los pasos que hemos seguido en el anterior punto.

Paso 2: Calcular los ahorros de la siguiente manera:

$$\Delta C_{ij} = C_{i0} + C_{0j} - C_{ij}$$

Donde $i \neq j$

Este ahorro se calcula agregando las rutas iniciales, desde la fábrica hasta cada proveedor. Por ejemplo, la ruta 1-2 con distancia de 200 y la ruta 1-3 con distancia de 300, sabiendo que 2-3 tiene una distancia 100, el ahorro de agregar estas dos rutas en 1-2-3 es igual a 200 + 300 -100= 400. Con esta fórmula, se puede calcular el ahorro que supone agregando cualquier de las rutas iniciales.

Ordenar el resultado en orden descendente.

Paso 3: Construir las rutas agregadas y eliminar rutas iniciales:

Una vez obtenida la lista de ahorros ordenados, se procede a la agregación de rutas iniciales en rutas con esos ahorros, atendiendo a estas reglas:

1. Un vértice no se puede enlazar más de dos veces en una ruta.
2. La ruta siga cumpliendo las restricciones de los recursos disponibles con el proveedor agregado, por ejemplo, que la carga de la ruta sea menor que la capacidad del vehículo.
3. Si no satisface las restricciones, esa agregación no es válida y se crea otra ruta nueva.

Siguiendo a estos pasos, se obtiene una solución aplicando la heurística de Clarke Wright o algoritmo de ahorro.

3.3 Algoritmo de Milk Runs y algoritmo de ahorro

3.3.1 Milk Runs

El código de Matlab creado de Milk Runs para asignar los proveedores a cada día de la semana, está dividido en tres secciones:

1. La primera sección es para introducir los datos iniciales:

```
% -----------------Datos iniciales--------------------
nvehicle=1;
h=10;
VVC=1;
s=0;
nvendor=21;
nitem=21;
d=[110 70 80 140 210 40 80 10 50 60 120 130 130 30 90 210 100 90 250 180 70];

c=[1 2 3 4 5 6 7 8 9 1 1 2 3 4 5 6 7 8 9 2 2];

pk=zeros(21,21);
for i=1:nvendor
    for j=1:nvendor
        if i==j
            pk(i,j)=1;
        end
    end
end
```

Figura 6 Primera sección de código de Matlab

2. La segunda sección es la sección donde se calculan todos los datos necesarios para la asignación de los proveedores, incluyen cálculo de 5 variables distintas pero relacionados entre ellos: *dtmx*, *dtmn*, *mxn*, *mnn* y *n*.

```
%-----------------------calculo de dtmx----------------------------
for k=1:nvendor
    dtmx(k)=time(1,k+1)+time(k+1,1);
end
dtmx;

%-----------------------calculo de dtmn--------------------------
MatDist=time;
MatDist(:,[1])=[];
MatDist([1],:)=[];
for k=1:nvendor

    VectDist=MatDist(k,:);
    OrdDist=sort(VectDist);
    dtmn(k)=OrdDist(2)+OrdDist(3);
end
```

Figura 7 Segunda sección del código de Matlab dtmx y dtmn

```
% ----------------------calculo de mnn--------------------------
for k=1:nvendor
    sum=0; setup=0;
    for i=1:nitem
        sum=sum+d(i)*c(i)*pk(k,i);
        setup=setup+s*pk(k,i);
    end
    mnn(k)=sqrt((h*sum)/(setup+2*VVC*dtmx(k)));


end
mnn;

% ----------------------------calculo de mxn---------------------
for k=1:nvendor
    sum=0; setup=0;
    for i=1:nitem
        sum=sum+d(i)*c(i)*pk(k,i);
        setup=setup+s*pk(k,i);
    end
    mxn(k)=sqrt((h*sum)/(setup+2*VVC*dtmn(k)));


end
mxn;
```

Figura 8 Segunda sección del código de Matlab mnn y mxn

```
% ----------------------------Calculo de n-----------------------
for k=1:nvendor
    n(k)=mnn(k)+alpha*(mxn(k)-mnn(k));
    n(k)=round(n(k));
    if n(k)>5
        n(k)=5;
    end
end
n;
```

Figura 9 Segunda sección del código de Matlab n

3. La tercera sección es donde se asignan los proveedores según el valor de *n* de cada proveedor a los 5 días laborales de la semana, la asignación se guarda en la variable *MatDays* en formato de matriz, la variable *MatDays* es una matriz de dimensión 5x número de proveedores del problema, si el proveedor i no está asignado a la ruta de lunes, el valor de la posición *MatDays* (1,i) es igual a cero, en caso contrario, el valor será i :

```
MatDays=zeros(5,6);
for k=1:nvendor

    if n(k)==1
        MatDays(3,k)=k;
    else
        if n(k)==2
            MatDays(2,k)=k;
            MatDays(4,k)=k;
        else
            if n(k)==3
                MatDays(1,k)=k;
                MatDays(3,k)=k;
                MatDays(5,k)=k;
            else
                if n(k)==4
                MatDays(1,k)=k;
                MatDays(2,k)=k;
                MatDays(4,k)=k;
                MatDays(5,k)=k;
                else
                    if n(k)==5
                        MatDays(1,k)=k;
                        MatDays(2,k)=k;
                        MatDays(3,k)=k;
```

Figura 10 Tercera sección del código de Matlab

Para aplicar el algoritmo de Milk Runs, previamente se tiene que preparar los datos de los problemas.

Los datos que trae el fichero descargado de internet, están juntos en una sola columna, no se puede trabajar con ellos de esta forma, hay que separarlos. En la pestaña superior de Excel, hay una opción de DATOS que permite tratar con estas coordenadas, con la función de Texto en Columna, se puede separar datos de una misma columna.

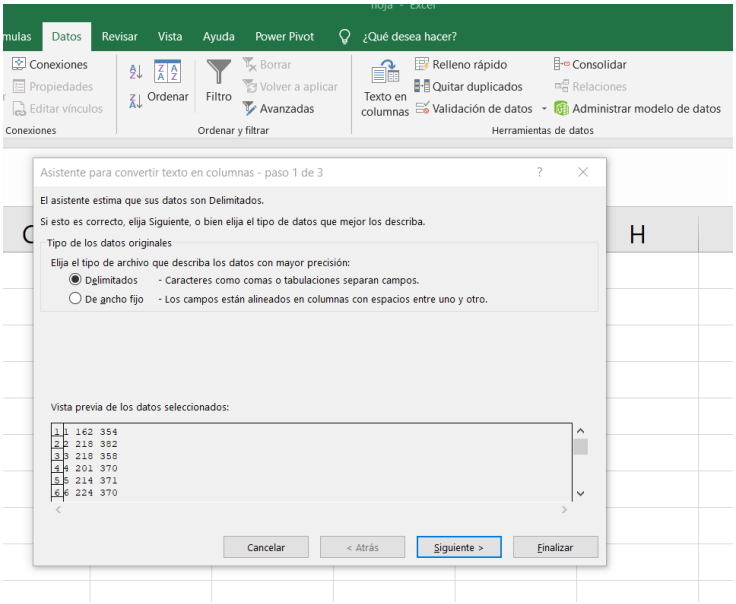

Figura 11 Función separador de textos de Excel

Los datos que se han obtenido del Bloc de Nota son textos separados por un espacio en blanco, elegimos la opción Delimitados.

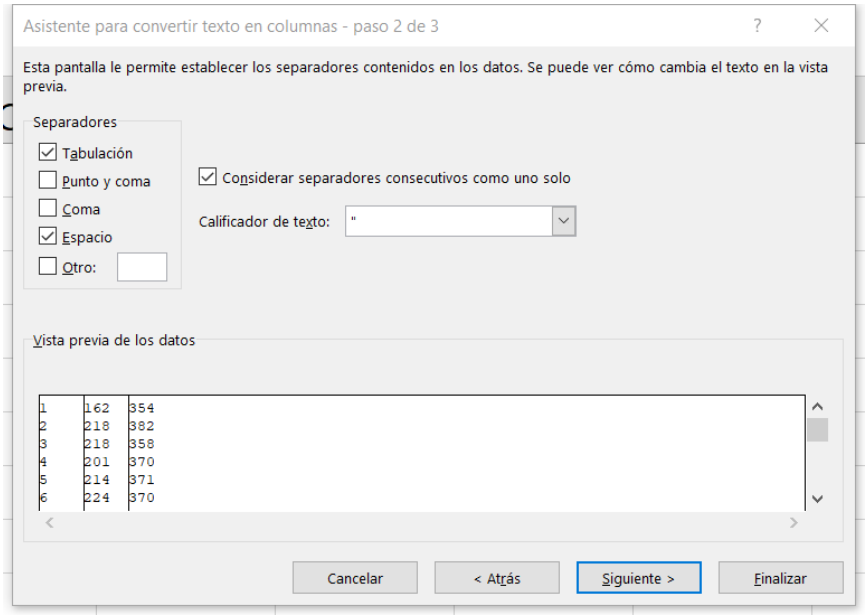

Figura 12 Función separador de textos con Espacio

Se marca en los separadores el Espacio como separador, automáticamente se transforman en tres columnas.

Se introducen las demandas también, el resultado es:

	A	B	C	D
1	Proveedor	Coordenada X	Coordenada Y	Demanda
2	1	162	354	0
3	2	218	382	300
4	3	218	358	3100
5	4	201	370	125
6	5	214	371	100
7	6	224	370	200
8	7	210	382	150
9	8	104	354	150
10	9	126	338	450
11	10	119	340	300
12	11	129	349	100
13	12	126	347	950

Figura 13 Textos separados en distintas columnas

Con los datos en hoja de Excel, se aplica la fórmula de calcular distancia euclidiana entre los dos puntos a estas coordenadas y se saca la tabla de distancias

que es una matriz de dimensión *NxN*, *N* es el número total de puntos incluido la fábrica.

$$D(i,j) = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

Se ponen en columnas los datos de las coordenadas de los proveedores, la primera pareja de coordenadas es la posición de la fábrica. Mediante la fórmula de Excel:

RAIZ((B1-B$1)^2+(C1-C$1)^2)

se calcula la distancia desde la fábrica (B1, C1) hasta la fábrica, obviamente esta distancia es cero.

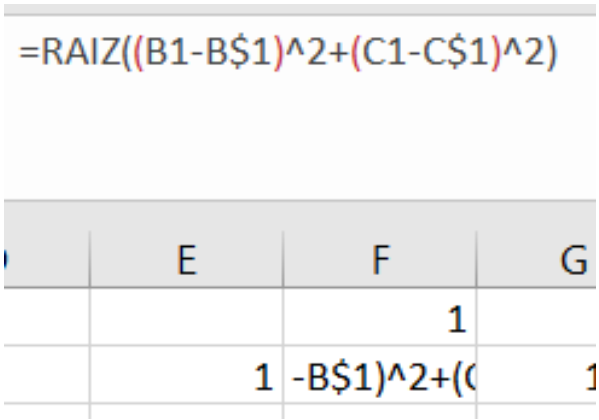

Figura 14 Fórmula de Excel para calcular distancia euclídea entre dos puntos

Con esta fórmula y bloqueando el segundo elemento dentro de cada paréntesis con el símbolo $, el primer elemento de cada paréntesis se va modificando mientras se arrastran hacia abajo en la primera columna de la tabla de distancias.

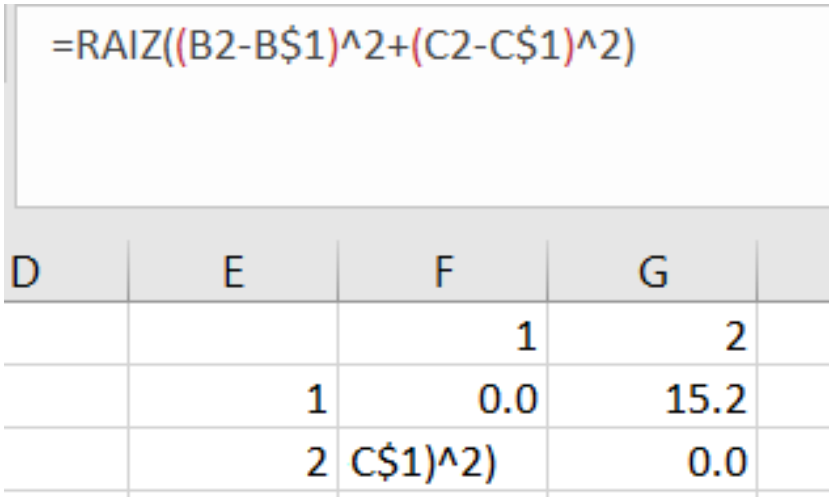

Figura 15 Fórmula de Excel para calcular tabla de distancia de B2 a B1 del problema E-n22-k4

Descripción del algoritmo de Milk Runs y de la heurística método de ahorro

Se obtiene la tabla de distancias:

E	F	G	H	I	J	K	L	M	N	O	P	Q	R
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.0	15.2	18.0	22.4	25.0	20.6	11.2	21.2	26.2	32.0	25.5	33.5	15.0
2	15.2	0.0	32.6	14.6	32.2	32.2	24.8	21.0	31.6	17.8	15.6	26.4	16.6
3	18.0	32.6	0.0	34.4	20.2	23.9	16.4	36.2	36.1	47.4	43.3	50.3	23.4
4	22.4	14.6	34.4	0.0	25.0	42.7	33.5	35.4	45.0	15.0	29.2	40.3	11.2
5	25.0	32.2	20.2	25.0	0.0	41.2	31.6	46.1	50.5	40.0	47.2	57.0	15.8
6	20.6	32.2	23.9	42.7	41.2	0.0	10.0	20.6	13.9	50.0	33.5	35.4	35.4
7	11.2	24.8	16.4	33.5	31.6	10.0	0.0	20.6	19.8	42.4	30.4	35.4	25.5
8	21.2	21.0	36.2	35.4	46.1	20.6	20.6	0.0	12.2	36.4	14.1	15.0	33.5
9	26.2	31.6	36.1	45.0	50.5	13.9	19.8	12.2	0.0	48.1	26.2	24.2	40.8
10	32.0	17.8	47.4	15.0	40.0	50.0	42.4	36.4	48.1	0.0	25.0	35.4	25.5
11	25.5	15.6	43.3	29.2	47.2	33.5	30.4	14.1	26.2	25.0	0.0	11.2	32.0
12	33.5	26.4	50.3	40.3	57.0	35.4	35.4	15.0	24.2	35.4	11.2	0.0	42.4
13	15.0	16.6	23.4	11.2	15.8	35.4	25.5	33.5	40.8	25.5	32.0	42.4	0.0

Figura 16 Tabla de distancias euclidianas calculadas en Excel

Los datos de esta tabla de distancia obtenida, se almacena en formato CSV (acrónimo en inglés de datos separado por coma) para usarlo luego en el programa como muestra en la figura siguiente:

	A	B	C	D	E	F	G	H
1	0;49.4;48.1;41.8;40.7;36.7;31;31.4;24.2;27.7;17.3;23.3;11.2;16;7.1;20.2;9.8;22.1;29.1;30.5;31.6;33.5							
2	49.4;0;8.5;23.3;25.9;20.8;18.7;24.2;26.6;30.5;32.1;40.2;47.3;54.6;56.2;57.5;58.9;71.1;72.2;78.2;79.1;82.9							
3	48.1;8.5;0;29.8;32.3;14.6;19.8;19.1;27.8;25.3;31;43.1;44.1;55.8;54.6;53.2;57.9;69.1;68.2;78;76.1;81.5							
4	41.8;23.3;29.8;0;2.8;33.7;17.9;33.2;19.2;37.6;28.4;23.1;45.2;40;48.7;57.2;49.2;63.3;69.8;65;73.4;72.6							
5	40.7;25.9;32.3;2.8;0;35.4;19;34.5;19.1;38.5;28.3;21;44.8;38;47.5;56.9;47.8;62;69.1;63;72.2;70.9							
6	36.7;20.8;14.6;33.7;35.4;0;17;5.4;22.5;11;21.2;38.5;30.8;47.4;42.5;39;46.5;56.3;54;67.2;62.5;69.3							
7	31;18.7;19.8;17.9;19;17;0;15.5;8.1;19.7;14.1;23.4;30.7;36.2;38;42;40.3;53;56;59.5;61.7;64.4							
8	31.4;24.2;19.1;33.2;34.5;5.4;15.5;0;19.2;6.3;16.4;34.8;25.5;42.5;37.2;34.1;41.2;51;49.1;61.9;57.3;63.9							
9	24.2;26.6;27.8;19.2;19.1;22.5;8.1;19.2;0;21.2;9.2;16.1;26.1;28.2;31.3;38;33;46.3;51;51.7;55.5;57.1							
10	27.7;30.5;25.3;37.6;38.5;11;19.7;6.3;21.2;0;15.5;35.4;20.2;40.5;32.8;28;37.2;45.9;43;58;51.6;59.1							
11	17.3;32.1;31;28.4;28.3;21.2;14.1;16.4;9.2;15.5;0;20;17;26.2;24.1;28.8;26.9;39;42.2;47;47.5;50.8							
12	23.3;40.2;43.1;23.1;21;38.5;23.4;34.8;16.1;35.4;20;0;31.3;17;29.2;42.7;28.2;42.5;52.3;42;53.3;50.2							
13	11.2;47.3;44.1;45.2;44.8;30.8;30.7;25.5;26.1;20.2;17;31.3;0;27.2;13.5;12;18.6;25.6;25.3;38.9;32;38.9							
14	16;54.6;55.8;40;38;47.4;36.2;42.5;28.2;40.5;26.2;17;27.2;0;18;35.5;14.4;27.7;40.8;25;38.9;33.5							
15	7.1;56.2;54.6;48.7;47.5;42.5;38;37.2;31.3;32.8;24.1;29.2;13.5;18;0;18;5.4;15;23.4;25.5;24.7;26.9							
16	20.2;57.5;53.2;57.2;56.9;39;42;34.1;38;28;28.8;42.7;12;35.5;18;0;23.1;22.7;15;39.8;24.7;36.1							
17	9.8;58.9;57.9;49.2;47.8;46.5;40.3;41.2;33;37.2;26.9;28.2;18.6;14.4;5.4;23.1;0;14.3;26.4;20.8;25.2;24.1							
18	22.1;71.1;69.1;63.3;62;56.3;53;51;46.3;45.9;39;42.5;25.6;27.7;15;22.7;14.3;0;17;18.4;11.3;13.6							
19	29.1;72.2;68.2;69.8;69.1;54;56;49.1;51;43;42.2;52.3;25.3;40.8;23.4;15;26.4;17;0;35.2;12;27.3							
20	30.5;78.2;78;65;63;67.2;59.5;61.9;51.7;58;47;42;38.9;25;25.5;39.8;20.8;18.4;35.2;0;26.3;12.2							
21	31.6;79.1;76.1;73.4;72.2;62.5;61.7;57.3;55.5;51.6;47.5;53.3;32;38.9;24.7;24.7;25.2;11.3;12;26.3;0;16.3							
22	33.5;82.9;81.5;72.6;70.9;69.3;64.4;63.9;57.1;59.1;50.8;50.2;38.9;33.5;26.9;36.1;24.1;13.6;27.3;12.2;16.3;0							

Figura 17 Tabla de distancias de E-n22-k4 guardada en CSV

Con los datos que se han dado y calculado en Excel, se calcula la planificación de rutas mediante el código de Matlab siguiendo el procedimiento siguiente:

Descripción del algoritmo de Milk Runs y de la heurística método de ahorro

Paso 1: Introducción de los datos iniciales en Matlab

Milkway.m

```
% -----------------Datos iniciales--------------------
nvehicle=1;
h=10;
VVC=1;
s=0;
nvendor=21;
nitem=21;
d=[110 70 80 140 210 40 80 10 50 60 120 130 130 30 90 210 100 90 250 180 70];

c=[1 2 3 4 5 6 7 8 9 1 1 2 3 4 5 6 7 8 9 2 2];

pk=zeros(21,21);
for i=1:nvendor
    for j=1:nvendor
        if i==j
            pk(i,j)=1;
        end
```

Figura 18 Introducción de datos ejemplo n22 en código de Matlab

Se crea la matriz *Pk* que será una matriz de 21x21, en este problema, cada proveedor tiene un solo producto, por lo tanto, solamente los elementos diagonales como *Pk*(1,1), *Pk*(2,2), etc. tienen valor de 1, los demás elementos son ceros.

Paso 2: cálculo de *dtmx* y *dtmn* de cada proveedor mediante las fórmulas descritas, el resultado es lo siguiente:

```
dtmx =

  Columns 1 through 10

   98.8000   96.2000   83.6000   81.4000   73.4000   62.0000   62.8000   48.4000   55.4000   34.6000

  Columns 11 through 20

   46.6000   22.4000   32.0000   14.2000   40.4000   19.6000   44.2000   58.2000   61.0000   63.2000

  Column 21

   67.0000


dtmn =

  Columns 1 through 10

   27.2000   23.1000   20.7000   21.8000   16.4000   22.2000   11.7000   17.3000   17.3000   23.3000

  Columns 11 through 20

   33.1000   25.5000   31.4000   18.9000   27.0000   19.7000   24.9000   27.0000   30.6000   23.3000

  Column 21

   25.8000
```

Figura 19 ejemplo dtmn y dtmx calculado de n22

Paso 3: cálculo de *mxn* y *mnn* de cada proveedor según las fórmulas descritas:

```
mnn =

  Columns 1 through 10

    2.3594    2.6975    3.7887    5.8650    8.4573    4.3994    6.6773    2.8748    6.3729    2.9446

  Columns 11 through 20

    3.5882    7.6181    7.8062    6.5003    7.4628   17.9284    8.8986    7.8648   13.5804    5.3368

  Column 21

    3.2323
```

```
mxn =

  Columns 1 through 10

    4.4967    5.5048    7.6139   11.3332   17.8920    7.3521   15.4698    4.8085   11.4043    3.5882

  Columns 11 through 20

    4.2576    7.1401    7.8805    5.6344    9.1287   17.8829   11.8559   11.5470   19.1741    8.7894

  Column 21

    5.2088
```

Figura 20 ejemplo mxn y mnn calculados de n22

Paso 4: cálculo de *n* de cada proveedor según fórmula descrita:

```
n =

  Columns 1 through 17

     2     3     4     5     5     4     5     3     5     3     4     5     5     5     5     5     5

  Columns 18 through 21

     5     5     5     3
```

Figura 21 ejemplo n calculado de n22

Paso 5: Asignar proveedores a cada día de la semana según el *n* de cada proveedor, las cinco rutas resultantes son:

```
Columns 1 through 17

     0     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17
     1     0     3     4     5     6     7     0     9     0    11    12    13    14    15    16    17
     0     2     0     4     5     0     7     8     9    10     0    12    13    14    15    16    17
     1     0     3     4     5     6     7     0     9     0    11    12    13    14    15    16    17
     0     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17
```

```
Columns 18 through 21

   18     19     20     21
   18     19     20      0
   18     19     20     21
   18     19     20      0
   18     19     20     21
```

Figura 22 ejemplo Planificación de rutas de n22

Se observan que las rutas de lunes y viernes son iguales, las rutas de martes y jueves son iguales, por lo tanto, hay que aplicar el algoritmo de ahorro de Python 3 veces. Hay que preparar la tabla de distancias de estas rutas eliminando los proveedores que no aparecen en las rutas, hay que tener en cuenta que la fila y columna que se elimina es el índice del número de proveedor que no aparece en la ruta más uno, ya que, en la tabla de distancias, el primero es la fábrica:

La primera tabla de distancias de las rutas de lunes y viernes se consigue eliminando el proveedor 1 de la tabla completa de distancias, con lo cual, se elimina la fila y la columna del proveedor 2:

	1	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0.00	48.08	41.79	40.72	36.72	31.02	31.38	24.19	27.66	17.26	23.35	11.18	16.03	7.07	20.25	9.85	22.09	29.07	30.53	31.62	33.54
3	48.08	0.00	29.83	32.28	14.56	19.85	19.10	27.80	25.32	31.02	43.14	44.10	55.76	54.57	53.24	57.87	69.05	68.18	78.00	76.11	81.49
4	41.79	29.83	0.00	2.83	33.73	17.89	33.24	19.21	37.59	28.43	23.09	45.22	40.01	48.70	57.20	49.24	63.32	69.84	65.01	73.39	72.56
5	40.72	32.28	2.83	0.00	35.36	18.97	34.48	19.10	38.48	28.28	21.00	44.82	38.01	47.54	56.85	47.80	61.98	69.12	63.01	72.24	70.86
6	36.72	14.56	33.73	35.36	0.00	17.03	5.39	22.47	11.00	21.21	38.48	30.81	47.38	42.54	39.01	46.53	56.32	54.01	67.23	62.51	69.29
7	31.02	19.85	17.89	18.97	17.03	0.00	15.52	8.06	19.72	14.14	23.43	30.68	36.24	38.00	42.05	40.31	53.01	55.97	59.48	61.66	64.38
8	31.38	19.10	33.24	34.48	5.39	15.52	0.00	19.24	6.32	16.40	34.79	25.50	42.52	37.16	34.13	41.18	50.96	49.09	61.91	57.31	63.91
9	24.19	27.80	19.21	19.10	22.47	8.06	19.24	0.00	21.21	9.22	16.12	26.08	28.18	31.26	38.01	33.02	46.27	50.99	51.66	55.54	57.08
10	27.66	25.32	37.59	38.48	11.00	19.72	6.32	21.21	0.00	15.52	35.36	20.25	40.50	32.76	28.02	37.20	45.88	43.01	58.01	51.62	59.09
11	17.26	31.02	28.43	28.28	21.21	14.14	16.40	9.22	15.52	0.00	20.02	17.00	26.17	24.08	28.84	26.93	39.01	42.15	47.01	47.52	50.80
12	23.35	43.14	23.09	21.00	38.48	23.43	34.79	16.12	35.36	20.02	0.00	31.30	17.03	29.21	42.72	28.18	42.49	52.35	42.01	53.34	50.22
13	11.18	44.10	45.22	44.82	30.81	30.68	25.50	26.08	20.25	17.00	31.30	0.00	27.17	13.45	12.04	18.60	25.63	25.30	38.90	32.02	38.91
14	16.03	55.76	40.01	38.01	47.38	36.24	42.52	28.18	40.50	26.17	17.03	27.17	0.00	18.03	35.51	14.42	27.66	40.82	25.00	38.95	33.53
15	7.07	54.57	48.70	47.54	42.54	38.00	37.16	31.26	32.76	24.08	29.21	13.45	18.03	0.00	18.00	5.39	15.03	23.43	25.50	24.70	26.93
16	20.25	53.24	57.20	56.85	39.01	42.05	34.13	38.01	28.02	28.84	42.72	12.04	35.51	18.00	0.00	23.09	22.67	15.00	39.82	24.70	36.07
17	9.85	57.87	49.24	47.80	46.53	40.31	41.18	33.02	37.20	26.93	28.18	18.60	14.42	5.39	23.09	0.00	14.32	26.42	20.81	25.24	24.08
18	22.09	69.05	63.32	61.98	56.32	53.01	50.96	46.27	45.88	39.01	42.49	25.63	27.66	15.03	22.67	14.32	0.00	17.00	18.44	11.31	13.60
19	29.07	68.18	69.84	69.12	54.01	55.97	49.09	50.99	43.01	42.15	52.35	25.30	40.82	23.43	15.00	26.42	17.00	0.00	35.23	12.04	27.31
20	30.53	78.00	65.01	63.01	67.23	59.48	61.91	51.66	58.01	47.01	42.01	38.90	25.00	25.50	39.82	20.81	18.44	35.23	0.00	26.31	12.21
21	31.62	76.11	73.39	72.24	62.51	61.66	57.31	55.54	51.62	47.52	53.34	32.02	38.95	24.70	24.70	25.24	11.31	12.04	26.31	0.00	16.28
22	33.54	81.49	72.56	70.86	69.29	64.38	63.91	57.08	59.09	50.80	50.22	38.91	33.53	26.93	36.07	24.08	13.60	27.31	12.21	16.28	0.00

Figura 23 Ejemplo Tabla de distancias de lunes y viernes de n22

La segunda tabla de distancias de las rutas de martes y jueves se consigue eliminando los proveedores 2, 8, 10 y 21 de la tabla completa de distancias, con lo cual, se elimina la fila y la columna del proveedor 3, 9 ,11 y 22:

	1	2	4	5	6	7	8	10	12	13	14	15	16	17	18	19	20	21
1	0.00	49.37	41.79	40.72	36.72	31.02	31.38	27.66	23.35	11.18	16.03	7.07	20.25	9.85	22.09	29.07	30.53	31.62
2	49.37	0.00	23.26	25.94	20.81	18.68	24.17	30.46	40.22	47.27	54.63	56.22	57.49	58.86	71.11	72.18	78.16	79.10
4	41.79	23.26	0.00	2.83	33.73	17.89	33.24	37.59	23.09	45.22	40.01	48.70	57.20	49.24	63.32	69.84	65.01	73.39
5	40.72	25.94	2.83	0.00	35.36	18.97	34.48	38.48	21.00	44.82	38.01	47.54	56.85	47.80	61.98	69.12	63.01	72.24
6	36.72	20.81	33.73	35.36	0.00	17.03	5.39	11.00	38.48	30.81	47.38	42.54	39.01	46.53	56.32	54.01	67.23	62.51
7	31.02	18.68	17.89	18.97	17.03	0.00	15.52	19.72	23.43	30.68	36.24	38.00	42.05	40.31	53.01	55.97	59.48	61.66
8	31.38	24.17	33.24	34.48	5.39	15.52	0.00	6.32	34.79	25.50	42.52	37.16	34.13	41.18	50.96	49.09	61.91	57.31
10	27.66	30.46	37.59	38.48	11.00	19.72	6.32	0.00	35.36	20.25	40.50	32.76	28.02	37.20	45.88	43.01	58.01	51.62
12	23.35	40.22	23.09	21.00	38.48	23.43	34.79	35.36	0.00	31.30	17.03	29.21	42.72	28.18	42.49	52.35	42.01	53.34
13	11.18	47.27	45.22	44.82	30.81	30.68	25.50	20.25	31.30	0.00	27.17	13.45	12.04	18.60	25.63	25.30	38.90	32.02
14	16.03	54.63	40.01	38.01	47.38	36.24	42.52	40.50	17.03	27.17	0.00	18.03	35.51	14.42	27.66	40.82	25.00	38.95
15	7.07	56.22	48.70	47.54	42.54	38.00	37.16	32.76	29.21	13.45	18.03	0.00	18.00	5.39	15.03	23.43	25.50	24.70
16	20.25	57.49	57.20	56.85	39.01	42.05	34.13	28.02	42.72	12.04	35.51	18.00	0.00	23.09	22.67	15.00	39.82	24.70
17	9.85	58.86	49.24	47.80	46.53	40.31	41.18	37.20	28.18	18.60	14.42	5.39	23.09	0.00	14.32	26.42	20.81	25.24
18	22.09	71.11	63.32	61.98	56.32	53.01	50.96	45.88	42.49	25.63	27.66	15.03	22.67	14.32	0.00	17.00	18.44	11.31
19	29.07	72.18	69.84	69.12	54.01	55.97	49.09	43.01	52.35	25.30	40.82	23.43	15.00	26.42	17.00	0.00	35.23	12.04
20	30.53	78.16	65.01	63.01	67.23	59.48	61.91	58.01	42.01	38.90	25.00	25.50	39.82	20.81	18.44	35.23	0.00	26.31
21	31.62	79.10	73.39	72.24	62.51	61.66	57.31	51.62	53.34	32.02	38.95	24.70	24.70	25.24	11.31	12.04	26.31	0.00

Figura 24 Ejemplo Tabla de distancias de martes y jueves de n22

La tercera tabla de distancias de las rutas de miércoles se consigue eliminando los proveedores 1, 3, 6 y 11 de la tabla completa de distancias, con lo cual, se elimina la fila y la columna del proveedor 2, 4, 7 y 12:

	1	2	4	6	7	9	10	11	12	14	15	16	17	18	19	20	21	22
1	0.00	49.37	41.79	36.72	31.02	24.19	27.66	17.26	23.35	16.03	7.07	20.25	9.85	22.09	29.07	30.53	31.62	33.54
2	49.37	0.00	23.26	20.81	18.68	26.57	30.46	32.14	40.22	54.63	56.22	57.49	58.86	71.11	72.18	78.16	79.10	82.87
4	41.79	23.26	0.00	33.73	17.89	19.21	37.59	28.43	23.09	40.01	48.70	57.20	49.24	63.32	69.84	65.01	73.39	72.56
6	36.72	20.81	33.73	0.00	17.03	22.47	11.00	21.21	38.48	47.38	42.54	39.01	46.53	56.32	54.01	67.23	62.51	69.29
7	31.02	18.68	17.89	17.03	0.00	8.06	19.72	14.14	23.43	36.24	38.00	42.05	40.31	53.01	55.97	59.48	61.66	64.38
9	24.19	26.57	19.21	22.47	8.06	0.00	21.21	9.22	16.12	28.18	31.26	38.01	33.02	46.27	50.99	51.66	55.54	57.08
10	27.66	30.46	37.59	11.00	19.72	21.21	0.00	15.52	35.36	40.50	32.76	28.02	37.20	45.88	43.01	58.01	51.62	59.09
11	17.26	32.14	28.43	21.21	14.14	9.22	15.52	0.00	20.02	26.17	24.08	28.84	26.93	39.01	42.15	47.01	47.52	50.80
12	23.35	40.22	23.09	38.48	23.43	16.12	35.36	20.02	0.00	17.03	29.21	42.72	28.18	42.49	52.35	42.01	53.34	50.22
13	11.18	47.27	45.22	30.81	30.68	26.08	20.25	17.00	31.30	27.17	13.45	12.04	18.60	25.63	25.30	38.90	32.02	38.91
14	16.03	54.63	40.01	47.38	36.24	28.18	40.50	26.17	17.03	0.00	18.03	35.51	14.42	27.66	40.82	25.00	38.95	33.53
16	20.25	57.49	57.20	39.01	42.05	38.01	28.02	28.84	42.72	35.51	18.00	0.00	23.09	22.67	15.00	39.82	24.70	36.07
17	9.85	58.86	49.24	46.53	40.31	33.02	37.20	26.93	28.18	14.42	5.39	23.09	0.00	14.32	26.42	20.81	25.24	24.08
18	22.09	71.11	63.32	56.32	53.01	46.27	45.88	39.01	42.49	27.66	15.03	22.67	14.32	0.00	17.00	18.44	11.31	13.60
19	29.07	72.18	69.84	54.01	55.97	50.99	43.01	42.15	52.35	40.82	23.43	15.00	26.42	17.00	0.00	35.23	12.04	27.31
20	30.53	78.16	65.01	67.23	59.48	51.66	58.01	47.01	42.01	25.00	25.50	39.82	20.81	18.44	35.23	0.00	26.31	12.21
21	31.62	79.10	73.39	62.51	61.66	55.54	51.62	47.52	53.34	38.95	24.70	24.70	25.24	11.31	12.04	26.31	0.00	16.28
22	33.54	82.87	72.56	69.29	64.38	57.08	59.09	50.80	50.22	33.53	26.93	36.07	24.08	13.60	27.31	12.21	16.28	0.00

Figura 25 Ejemplo Tabla de distancias de miércoles de n22

Una vez aplicado el código de Matlab, se obtiene una planificación para el problema, se elabora tabla de distancias según los proveedores que aparecen en cada ruta de la semana mediante fórmula de calcular distancia euclidiana:

0	48.08326	40.71855	36.71512	31.38471	24.18677	27.65863	17.26268	11.18034	16.03122	7.071068	20.24846	9.848858	22.09072	29.06888	30.52868	31.62278	33.54102
48.08326	0	32.28002	14.56022	19.10497	27.80288	25.31798	31.01612	44.10215	55.75841	54.57105	53.23533	57.87055	69.05071	68.18358	78	76.10519	81.49233
40.71855	32.28002	0	35.35534	34.48188	19.10497	38.48376	28.28427	44.82187	38.01316	47.53946	56.85068	47.80167	61.98387	69.11584	63.00794	72.23573	70.85901
36.71512	14.56022	35.35534	0	5.385165	22.47221	11	21.2132	30.80584	47.38143	42.54409	39.01282	46.52956	56.32051	54.00926	67.23095	62.514	69.28925
31.01612	19.84943	18.97367	17.02939	15.52417	8.062258	19.72308	14.14214	30.67572	36.23534	38	42.04759	40.31129	53.00943	55.97321	59.48109	61.66036	64.38167
24.18677	27.80288	19.10497	22.47221	19.23538	0	21.2132	9.219544	26.07681	28.17801	31.257	38.01316	33.01515	46.27094	50.9902	51.66237	55.54278	57.07889
27.65863	25.31798	38.48376	11	6.324555	21.2132	0	15.52417	20.24846	40.49691	32.75668	28.01785	37.20215	45.88028	43.01163	58.00862	51.62364	59.09315
17.26268	31.01612	28.28427	21.2132	16.40122	9.219544	15.52417	0	17	26.1725	24.08319	28.84441	26.92582	39.01282	42.15448	47.01064	47.51842	50.80354
23.34524	43.13931	21	38.48376	34.78505	16.12452	35.35534	20.02498	31.30495	17.02939	29.20616	42.72002	28.17801	42.48529	52.34501	42.0119	53.33854	50.21952
11.18034	44.10215	44.82187	30.80584	25.4951	26.07681	20.24846	17	0	27.16616	13.45362	12.04159	18.60108	25.63201	25.29822	38.8973	32.01562	38.91015
7.071068	54.57105	47.53946	42.54409	37.16181	31.257	32.75668	24.08319	13.45362	18.02776	0	18	5.385165	15.0333	23.43075	25.4951	24.69818	26.92582
20.24846	53.23533	56.85068	39.01282	34.1321	38.01316	28.01785	28.84441	12.04159	35.51056	18	0	23.08679	22.67157	15	39.82462	24.69818	36.06938
9.848858	57.87055	47.80167	46.52956	41.18252	33.01515	37.20215	26.92582	18.60108	14.42221	5.385165	23.08679	0	14.31782	26.41969	20.80865	25.23886	24.08319
22.09072	69.05071	61.98387	56.32051	50.96077	46.27094	45.88028	39.01282	25.63201	27.65863	15.0333	22.67157	14.31782	0	17	18.43909	11.31371	13.60147
29.06888	68.18358	69.11584	54.00926	49.09175	50.9902	43.01163	42.15448	25.29822	40.81666	23.43075	15	26.41969	17	0	35.22783	12.04159	27.313
30.52868	78	63.00794	67.23095	61.91123	51.66237	58.00862	47.01064	38.8973	25	25.4951	39.82462	20.80865	18.43909	35.22783	0	26.30589	12.20656
31.62278	76.10519	72.23573	62.514	57.31492	55.54278	51.62364	47.51842	32.01562	38.94868	24.69818	24.69818	25.23886	11.31371	12.04159	26.30589	0	16.27882
33.54102	81.49233	70.85901	69.28925	63.90618	57.07889	59.09315	50.80354	38.91015	33.52611	26.92582	36.06938	24.08319	13.60147	27.313	12.20656	16.27882	0

Figura 26 Ejemplo de una tabla de distancia n22

Esta tabla de distancias se almacena en formato CSV para ser usadas en el código de Python de algoritmo de ahorro.

3.3.2 Algoritmo de ahorro

El código Python que se ha encontrado en Github para resolver el problema de enrutamiento con la restricción de la capacidad de un camión, está dividido también en varias secciones:

1. La primera sección es para introducir los datos iniciales del problema: número de cliente, capacidad de la carga y las demandas de los proveedores. En esta sección, se guardan todos estos datos:

```
# -----------Datos iniciales---------------------

def __init__(valor):
    valor.prov = 100
    valor.tons = 220
    valor.Limitekilom = 10000
    valor.distance = []
    valor.q = [0,10,7,13,19,26,3,5,9,16,
               16,12,19,23,20,8,19,2,12,17,
               9,11,18,29,3,6,17,16,16,9,
               21,27,23,11,14,8,5,8,16,31,
               9,5,5,7,18,16,1,27,36,30,
               13,10,9,14,18,2,6,7,18,28,
               3,13,19,10,9,20,25,25,36,6,
               5,15,25,9,8,18,13,14,3,23,
               6,26,16,11,7,41,35,26,9,15,
               3,1,2,22,27,20,11,12,10,9,17 ]
```

Figura 27 Sección de datos iniciales del código

2. La segunda sección es para importar la tabla de distancia calculada en Excel y guardada en formato CSV, se lee el documento CSV con la función csv.reader y se guarda en la matriz valor.distance:

```
# -----------Importar tabla de distancia-----------

def datainput(valor):
    with open("data2.csv", "r") as csvfile:
        reader = csv.reader(csvfile)
        for line in reader:
            line = [float(x) for x in line]
            valor.distance.append(line)
```

Figura 28 Sección de importar tabla de distancia del código

3. La tercera sección es la parte de la función de la heurística de ahorro llamado función savingsAlgorithms:

```
# ----------Algoritmo de ahorro---------------------

def savingsAlgorithms(valor):
    saving = 0
    for i in range(1, len(valor.q)):
        valor.Routes.append([i])

    for i in range(1, len(valor.Routes) + 1):
        for j in range(1, len(valor.Routes) + 1):
            if i == j:
                pass
            else:
                saving = (valor.distance[i][0] + valor.distance[0][j]) - valor.distance[i][j]
                valor.savings.append([i, j, saving])

    valor.savings = sorted(valor.savings, key=itemgetter(2), reverse=True)
    for i in range(len(valor.savings)):
        print(valor.savings[i][0],'--',valor.savings[i][1], "  ",valor.savings[i][2])

    for i in range(len(valor.savings)):
        startRoute = []
        endRoute = []
        routeDemand = 0
        for j in range(len(valor.Routes)):
            if (valor.savings[i][0] == valor.Routes[j][-1]):
                endRoute = valor.Routes[j]
            elif (valor.savings[i][1] == valor.Routes[j][0]):
                startRoute = valor.Routes[j]

            if ((len(startRoute) != 0) and (len(endRoute) != 0)):
                for k in range(len(startRoute)):
                    routeDemand += valor.q[startRoute[k]]
                for k in range(len(endRoute)):
                    routeDemand += valor.q[endRoute[k]]
                routeDistance = 0
                routestore = [0]+endRoute+startRoute+[0]
```

Figura 29 Sección de algoritmo de ahorro del código

4. Por último, la sección donde se utiliza el algoritmo de ahorro para sacar los resultados del cálculo en la pantalla:

```
# ----------Sacar los resultados---------------------

def printRoutes(valor):
    for i in valor.Routes:
        costs = 0
        for j in range(len(i)-1):
            costs += valor.distance[i[j]][i[j+1]]
        print("Ruta:  ",i,"  Distancia de la ruta:  ",costs)

def calcCosts(valor):
    for i in range(len(valor.Routes)):
        for j in range(len(valor.Routes[i]) - 1):
            valor.Cost += valor.distance[valor.Routes[i][j]][valor.Routes[i][j + 1]]

    print("\nTotal Distance: ", round(valor.Cost, 3))
```

Figura 30 Sección de sacar resultados del código

El procedimiento que se sigue para obtener las soluciones al aplicar el método de ahorro es de la siguiente manera:

1. Se coge la tabla de distancias completas de los problemas y se elimina aquellos proveedores que no aparecen en las rutas obtenidas con el código de Milk Runs, tanto en la fila como en la columna.

2. Esta tabla se guarda en un fichero de Excel de formato CSV, preparado para ser usado en el código de Python. El código abrirá el documento correspondiente al problema que se está resolviendo en CSV y guarda la tabla de distancia en la variable valor.distance del código.

	A	B	C	D	E	F	G	H	I	J	K	L
1	0	62.6	56.1	42.2	54.7	64	55.6	58	39.4	45.2	33.4	36.7
2	62.6	0	24	20.8	11.7	13.4	8	117.4	102	107.5	94.9	98.4
3	56.1	24	0	20.8	13.6	13.4	25.3	114.1	94.1	100.6	89.5	92.7
4	42.2	20.8	20.8	0	13	23	15	98.3	81.5	87.3	75	78.4
5	54.7	11.7	13.6	13	0	10	11.7	111.3	94	99.9	87.8	91.2
6	64	13.4	13.4	23	10	0	18.4	121.1	103.1	109.2	97.3	100.7
7	55.6	8	25.3	15	11.7	18.4	0	109.6	94.8	100.2	87.5	91
8	58	117.4	114.1	98.3	111.3	121.1	109.6	0	27.2	20.5	25.5	23.1
9	39.4	102	94.1	81.5	94	103.1	94.8	27.2	0	7.3	11.4	9
10	45.2	107.5	100.6	87.3	99.9	109.2	100.2	20.5	7.3	0	13.5	9.9
11	33.4	94.9	89.5	75	87.8	97.3	87.5	25.5	11.4	13.5	0	3.6
12	36.7	98.4	92.7	78.4	91.2	100.7	91	23.1	9	9.9	3.6	0
13	37.9	99.7	93.8	79.7	92.4	101.9	92.3	22.5	8.1	8.5	5	1.4
14	46	105.5	102	86.3	99.3	109	97.8	12	19.7	15.3	14.3	12.8
15	40.7	103.3	94.8	82.8	95.1	104.1	96.3	29.1	3	8.6	14.3	12
16	37	96.8	93	77.5	90.4	100.1	89.2	21	17	16.2	7.2	8.1
17	43.1	102.1	99	83	96	105.8	94.4	15.3	20.2	17	12.8	12.2
18	48.8	110.9	104.4	90.8	103.4	112.8	103.5	17	11.4	4.1	16.1	12.5
19	9.5	72	65.4	51.6	64.2	73.5	64.9	49.1	30	35.7	24.1	27.3
20	15.8	47	43.3	26.9	39.8	49.5	39.8	71.6	55	60.5	48.1	51.5
21	19	43.9	38.1	23.3	35.7	45.1	37.2	76.2	58.3	64.2	52.2	55.5

Figura 31 Tabla de distancias euclidianas guardadas en CSV

3. Se coge los datos de las demandas de los puntos del problema, y se introduce en el código de Python en la sección de datos iniciales, se guarda en la variable valor.q del código.

4. Se coge el dato de la capacidad de los vehículos de transporte y se introduce en el código de Python en la sección de datos iniciales y se guarda en la variable valor.tons.

```
import csv
from operator import itemgetter

class Vrp():

    # ----------Datos iniciales----------------------

    def __init__(valor):
        valor.prov = 100                                                    # Número de clientes
        valor.tons = 220                                                    # Capacidad de carga del
        valor.Limitekilom = 10000                                           # Limitación de kilómet
        valor.distance = []                                                 # Tabla de distancia
        valor.q = [0, 10, 7, 13, 19, 26, 3, 5, 9, 16,
                   16, 12, 19, 23, 20, 8, 19, 2, 12, 17,
                   9, 11, 18, 29, 3, 6, 17, 16, 16, 9,
                   21, 27, 23, 11, 14, 8, 5, 8, 16, 31,
                   9, 5, 5, 7, 18, 16, 1, 27, 36, 30,
                   13, 10, 9, 14, 18, 2, 6, 7, 18, 28,
                   3, 13, 19, 10, 9, 20, 25, 25, 36, 6,
                   5, 15, 25, 9, 8, 18, 13, 14, 3, 23,
                   6, 26, 16, 11, 7, 41, 35, 26, 9, 15,
                   3, 1, 2, 22, 27, 20, 11, 12, 10, 9, 17 ]
                                                    # Cantidades de los proveedores, el primero es de
        valor.savings = []                                                  # Ahorros
        valor.Routes = []                                                   # Rutas
        valor.Cost = 0                                                      # Total distancia de las
    # ----------Importar tabla de distancia--------------

    def datainput(valor):
        with open("data2.csv", "r") as csvfile:
            reader = csv.reader(csvfile)
            for line in reader:
                line = [float(x) for x in line]
                valor.distance.append(line)



    # ----------Algoritmo de ahorro----------------------
```

Figura 32 Datos iniciales introducidos en el código de Python

Con todos estos datos iniciales preparados, se ejecuta el problema y se obtiene el resultado de planificación de rutas para el problema aplicando el algoritmo de ahorro.

La capacidad deja de ser una limitación ya que las cargas se han dividido en 5 rutas distintas de la semana. Se pone un valor imposible de alcanzar en el código de Python. Los resultados que se obtienen son tres planificaciones distintas para cada tabla de distancia, pero será una sola ruta.

```
== == == == == == == == == == == == == == == Resultados == == == == == == == == = == == == == == == == == =
Ruta:   [0, 13, 15, 16, 18, 20, 19, 17, 14, 11, 9, 7, 2, 3, 5, 1, 4, 6, 8, 10, 12, 0]   Distancia de la ruta:   294.548
036338

Total Distance:  294.548
\\\
```

Figura 33 Resultado de rutas de lunes y viernes de n22

4.Realización de pruebas y resultados

Para comprobar la eficacia del algoritmo, se ha creado un código en Matlab para asignar los proveedores a cada día de la semana, un código de Python encontrado en internet para aplicar el algoritmo de ahorro y una hoja de Excel creado con fórmulas para calcular las tablas de distancias, siguiendo siempre a los razonamientos anteriormente explicados. Se adjuntan el código de Matlab y el código de Python en los anexos al final del documento.

4.1 Aplicar el algoritmo de Milk Runs y el algoritmo de ahorro

Primero se prueba la eficiencia de los algoritmos con el problema que viene en el documento publicado en Journal of Operations Management: "Reducing Vendor Delivery Uncertainties in a JIT Environment".

- **Problema de ejemplo:**

El problema consiste en 6 proveedores que suministra 7 productos a la fábrica que tienen estos datos iniciales:

Número de vehículo =1;
Recargo de transporte =0.01;
Coste variable de vehículo VVC=12;
Coste de carga y descarga s=0;
Número de proveedor =6;
Número de productos=7;
Vector de demandas=[1000 200 300 400 50 600 700];
Vector de coste de los productos =[10 60 30 400 4000 200 40];

La tabla de distancia es:

```
distancias=[0.0 1.6 0.6 2.0 1.7 1.6 1.5;
            1.6 0.0 1.9 3.5 0.7 1.3 3.1;
            0.6 1.9 0.0 1.6 2.2 2.2 1.6;
            2.0 3.5 1.6 0.0 3.7 3.6 1.4;
            1.7 0.7 2.2 3.7 0.0 0.7 3.1;
            1.6 1.3 2.2 3.6 0.7 0.0 2.7;
            1.5 3.1 1.6 1.4 3.1 2.7 0.0];
```

Se introduce todos estos datos en la sección de datos iniciales en Matlab.

Se crea un parámetro *Pk (6,7)* que representa en qué proveedor se suministra cuál productos, por lo tanto, si el producto 1 lo suministra el proveedor 1, *Pk(1,1)* = 1, en el caso contrario, *Pk(1,1)*=0;

Pk (6x7):

1	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	1	1

Este parámetro sirve para calcular los umbrales de tiempos de viaje *mxn (k)* y *mnn(k)*.

Una vez que se tienen los datos iniciales, se proceden a calcular los diferenciales de los umbrales de tiempos de viajes de cada proveedor k: *dtmx(k)* y *dtmn(k)* con la variable paramétrica desde *alpha* = 0 hasta *alpha*=1 en intervalo de incremento de 0.1:

alpha =0						
dtmx =	3.2000	1.2000	4.0000	3.4000	3.2000	3.0000
dtmn =	2.0000	3.2000	3.0000	1.4000	2.0000	3.0000
alpha =	0.1000					
dtmx =	3.2000	1.2000	4.0000	3.4000	3.2000	3.0000
dtmn =	2.0000	3.2000	3.0000	1.4000	2.0000	3.0000
alpha =	0.2000					
dtmx =	3.2000	1.2000	4.0000	3.4000	3.2000	3.0000
dtmn =	2.0000	3.2000	3.0000	1.4000	2.0000	3.0000
alpha =	0.3000					
dtmx =	3.2000	1.2000	4.0000	3.4000	3.2000	3.0000
dtmn =	2.0000	3.2000	3.0000	1.4000	2.0000	3.0000
alpha =	0.4000					
dtmx =	3.2000	1.2000	4.0000	3.4000	3.2000	3.0000
dtmn =	2.0000	3.2000	3.0000	1.4000	2.0000	3.0000
alpha =	0.5000					
dtmx =	3.2000	1.2000	4.0000	3.4000	3.2000	3.0000
dtmn =	2.0000	3.2000	3.0000	1.4000	2.0000	3.0000
alpha =	0.6000					

dtmx =	3.2000	1.2000	4.0000	3.4000	3.2000	3.0000
dtmn =	2.0000	3.2000	3.0000	1.4000	2.0000	3.0000
alpha =	0.7000					
dtmx =	3.2000	1.2000	4.0000	3.4000	3.2000	3.0000
dtmn =	2.0000	3.2000	3.0000	1.4000	2.0000	3.0000
alpha =	0.8000					
dtmx =	3.2000	1.2000	4.0000	3.4000	3.2000	3.0000
dtmn =	2.0000	3.2000	3.0000	1.4000	2.0000	3.0000
alpha =	0.9000					
dtmx =	3.2000	1.2000	4.0000	3.4000	3.2000	3.0000
dtmn =	2.0000	3.2000	3.0000	1.4000	2.0000	3.0000
alpha =	1					
dtmx =	3.2000	1.2000	4.0000	3.4000	3.2000	3.0000
dtmn =	2.0000	3.2000	3.0000	1.4000	2.0000	3.0000

Con los *dtmx(k)* y los *dtmn(k)*, se calcula los *mxn (k)*, *mnn(k)* y por lo consiguiente el *n(k)*:

mnn =	1.1411	2.0412	0.9682	4.4281	5.1031	4.5338
mxn =	1.4434	1.2500	1.1180	6.9007	6.4550	4.5338
n =	1	2	1	4	5	5
mnn =	1.1411	2.0412	0.9682	4.4281	5.1031	4.5338
mxn =	1.4434	1.2500	1.1180	6.9007	6.4550	4.5338
n =	1	2	1	5	5	5
mnn =	1.1411	2.0412	0.9682	4.4281	5.1031	4.5338
mxn =	1.4434	1.2500	1.1180	6.9007	6.4550	4.5338
n =	1	2	1	5	5	5
mnn =	1.1411	2.0412	0.9682	4.4281	5.1031	4.5338
mxn =	1.4434	1.2500	1.1180	6.9007	6.4550	4.5338
n =	1	2	1	5	5	5
mnn =	1.1411	2.0412	0.9682	4.4281	5.1031	4.5338
mxn =	1.4434	1.2500	1.1180	6.9007	6.4550	4.5338
n =	1	2	1	5	5	5
mnn =	1.1411	2.0412	0.9682	4.4281	5.1031	4.5338
mxn =	1.4434	1.2500	1.1180	6.9007	6.4550	4.5338
n =	1	2	1	5	5	5
mnn =	1.1411	2.0412	0.9682	4.4281	5.1031	4.5338
mxn =	1.4434	1.2500	1.1180	6.9007	6.4550	4.5338
n =	1	2	1	5	5	5
mnn =	1.1411	2.0412	0.9682	4.4281	5.1031	4.5338
mxn =	1.4434	1.2500	1.1180	6.9007	6.4550	4.5338
n =	1	1	1	5	5	5
mnn =	1.1411	2.0412	0.9682	4.4281	5.1031	4.5338
mxn =	1.4434	1.2500	1.1180	6.9007	6.4550	4.5338
n =	1	1	1	5	5	5
mnn =	1.1411	2.0412	0.9682	4.4281	5.1031	4.5338

mxn =	1.4434	1.2500	1.1180	6.9007	6.4550	4.5338
n =	1	1	1	5	5	5
mnn =	1.1411	2.0412	0.9682	4.4281	5.1031	4.5338
mxn =	1.4434	1.2500	1.1180	6.9007	6.4550	4.5338
n =	1	1	1	5	5	5

Con los *n* (k) calculados, se planifica las rutas de cada día de la semana expresada en la matriz *MatDays* siendo las dimensiones de *MatDays* 5x6 (5 días de la semana y 6 clientes):

MatDays =

0 0 0 4 5 6

0 2 0 4 5 6

1 0 3 0 5 6

0 2 0 4 5 6

0 0 0 4 5 6

MatDays =

0 0 0 4 5 6

0 2 0 4 5 6

1 0 3 4 5 6

0 2 0 4 5 6

0 0 0 4 5 6

MatDays =

0 0 0 4 5 6

0 2 0 4 5 6

1 0 3 4 5 6

0 2 0 4 5 6

0 0 0 4 5 6

MatDays =

0 0 0 4 5 6

0 2 0 4 5 6

1 0 3 4 5 6

0 2 0 4 5 6

0 0 0 4 5 6

MatDays =

0 0 0 4 5 6

0 2 0 4 5 6

1 0 3 4 5 6

0 2 0 4 5 6

0 0 0 4 5 6

MatDays =

0 0 0 4 5 6

0 2 0 4 5 6

1 0 3 4 5 6

0 2 0 4 5 6

0 0 0 4 5 6

MatDays =

0 0 0 4 5 6

0 2 0 4 5 6

1 0 3 4 5 6

0 2 0 4 5 6

0 0 0 4 5 6

MatDays =

0 0 0 4 5 6

0 0 0 4 5 6

1 2 3 4 5 6

0 0 0 4 5 6

0 0 0 4 5 6

MatDays =

0 0 0 4 5 6

0 0 0 4 5 6

1 2 3 4 5 6

0 0 0 4 5 6

0 0 0 4 5 6

MatDays =

0 0 0 4 5 6

0 0 0 4 5 6

1 2 3 4 5 6

0 0 0 4 5 6

0 0 0 4 5 6

MatDays =

0 0 0 4 5 6

0 0 0 4 5 6

1 2 3 4 5 6

0 0 0 4 5 6

0 0 0 4 5 6

A estas 10 planificaciones de rutas, se le aplica el algoritmo de ahorro para mejorar el resultado de cada ruta. Como se puede apreciar, sola hay dos planificaciones distintas, se calculan las distancias totales y se queda con la mejor solución como solución de Milk Runs:

Como no se especifican las coordenadas en los datos del problema, procedemos a copiar trozos de distancias en la tabla de distancia B1 Travel Time Matriz en el documento CSV.

En la primera ruta (ruta de lunes), se tienen que recoger mercancías de tres proveedores que son proveedores 4, 5 y 6, junto con la fábrica, la tabla de distancia en CSV queda como una matriz 4x4:

	A	B	C	D	E	F	G
1	0	1.7	1.6	1.5			
2	1.7	0	0.7	3.1			
3	1.6	0.7	0	2.7			
4	1.5	3.1	2.7	0			
5							
6							
7							
8							
9							
10							

Figura 34 Tabla de distancias de rutas de lunes del ejemplo

Se aplica el algoritmo de ahorro a esta tabla y se obtiene:

```
3 -- 1    0.10000000000000009
== == == == == == == == == == == == == == Resultados == == == == == =
= == == == == == == =
Ruta:   [0, 1, 2, 3, 0]   Distancia de la ruta:   6.6

Total Distance:  6.6
```

Figura 35 Resultado de rutas de lunes del ejemplo

La combinación 0-4-5-6-0 de la planificación de lunes y de viernes ya es la mejor combinación según el algoritmo de ahorro (en la ruta aparece 0-1-2-3-0 es por la denominación en el código).

Se prueba ahora la ruta de martes y de jueves, son 4 proveedores (2,4,5,6), la tabla de distancia queda una matriz de 5x5:

	A	B	C	D	E	F
1	0	0.6	1.7	1.6	1.5	
2	0.6	0	2.2	2.2	1.6	
3	1.7	2.2	0	0.7	3.1	
4	1.6	2.2	0.7	0	2.7	
5	1.5	1.6	3.1	2.7	0	
6						
7						

Figura 36 Tabla de distancias de rutas de martes y jueves del ejemplo

El resultado queda:

```
== == == == == == == == == == == == == == Resultados == == == == == =
= == == == == == == =
Ruta:   [0, 1, 4, 2, 3, 0]   Distancia de la ruta:   7.600000000000001

Total Distance:  7.6
```

Figura 37 Resultado de rutas de martes del ejemplo

La mejor combinación queda en 0-2-6-4-5-0 en la planificación de martes y jueves.

Se prueba la ruta de miércoles ahora, la ruta inicial es 0-1-3-5-6-0.

	A	B	C	D	E	F
1	0	1.6	2	1.6	1.5	
2	1.6	0	3.5	1.3	3.1	
3	2	3.5	0	3.6	1.4	
4	1.6	1.3	3.6	0	2.7	
5	1.5	3.1	1.4	2.7	0	
6						

Figura 38 Tabla de distancias de rutas de miércoles del ejemplo

El resultado queda 0-1-6-3-5-0:

```
== == == == == == == == == == == == == == Resultados == == == == ==
= == == == == == == =
Ruta:   [0, 1, 3, 2, 4, 0]   Distancia de la ruta:   9.4

Total Distance:  9.4
```

Figura 39 Resultado de rutas de miércoles del ejemplo

Queda comprobado la efectividad del procedimiento. Solo se aproxima mucho a la solución perfecta porque el algoritmo de ahorro no analiza todas las soluciones posibles, solo una parte.

Una vez comprobada la efectividad del procedimiento, ahora se prueba un problema con mayor número de proveedores:

4.2 Otros ejemplos de problemas

El resultado que se ha obtenido son combinaciones de números que representan a los proveedores y la fábrica, esta combinación está dividida en varias rutas distintas, cada ruta empieza y termina por el número 0 que

representa la fábrica, el camión sale desde la fábrica, visita cada proveedor de la ruta y vuelve a la fábrica, así completa su ruta.

El problema, en definitiva, se resume en una planificación de enrutamientos de vehículos en el último paso, por lo que su efectividad depende también de la efectividad del algoritmo de ahorro, al aumentar el número de proveedores, aumenta también la complejidad del problema, ahora se hace problemas con mayor número de proveedores y comprobar la efectividad del algoritmo de ahorro comparándose con las soluciones mejor conocida de cada problema en una tabla.

Se ha comprobado de que el algoritmo de ahorro es útil, ahora se calcula y se hace una comprobación con distintos problemas de mayor número de proveedores:

Ahora se procede a calcular soluciones de algunos problemas de ejemplo.

- **Ejemplo 1: 22 proveedores (E-n23)**

  Número de vehículo =1;

  Recargo de transporte =0.1;

  Coste variable de vehículo VVC=15;

  Coste de carga y descarga s=0;

  Número de proveedor =22;

  Número de productos=22;

  Vector de demandas=[100 200 300 400 500 60 70 80 90 100
  110 120 130 140 150 160 170 180 190 200
  210 220];

  Vector de coste de los productos =[220 210 200 190 180 170 160 150
  140 130 120 110 100 90 80 70 60
  50 40 30 20 10];

Las coordenadas de la fábrica y de los proveedores son:

Proveedor	Coordenada X	Coordenada Y
1	26	23
2	29	27
3	30	25
4	30	26
5	21	27
6	21	28
7	28	26
8	24	24

9	23	26
10	24	26
11	25	26
12	26	25
13	26	24
14	25	26
15	31	23
16	32	25
17	31	25
18	32	22
19	26	21
20	27	19
21	30	20
22	20	21
23	32	18

Tabla 1 Coordenadas y demandas del problema E-n23

El resultado obtenido al aplicar el algoritmo de Milk Runs es:

```
MatDays =

  Columns 1 through 17

     1     2     3     4     5     0     7     0     0    10    11    12    13     0     0     0     0
     0     2     3     4     5     6     0     8     9     0     0    12     0    14    15    16    17
     1     0     0     0     5     0     7     0     0    10    11    12    13     0     0     0     0
     0     2     3     4     5     6     0     8     9     0     0    12     0    14    15    16    17
     1     2     3     4     5     0     7     0     0    10    11    12    13     0     0     0     0

  Columns 18 through 22

    18     0    20    21     0
     0    19    20     0    22
    18     0     0    21     0
     0    19    20     0    22
    18     0    20    21     0
```

Figura 40 Resultado de Milk Runs n23

A este resultado de Milk Runs, se aplica el algoritmo de ahorro tres veces, el resultado de rutas de lunes/viernes después de la aplicación es:

```
== == == == == == == == == == == == == == == Resultados == =
= == == == == == = == == == == == == == =
Ruta:    [0, 9, 8, 6, 13, 4, 5, 7, 10, 1, 3, 2, 12, 11, 0]
Distancia de la ruta:    38.4

Total Distance:  38.4
```

Figura 41 Resultado de ahorro n23 lunes viernes

En índices originales, la ruta de lunes/viernes es:

0-12-11-7-21-4-5-10-13-1-3-2-20-18-0

El resultado de rutas de martes/jueves después de la aplicación es:

```
== == == == == == == == == == == == == == Resultados == =
= == == == == == = == == == == == == == =
Ruta:   [0, 13, 14, 15, 12, 9, 2, 10, 11, 1, 5, 3, 4, 6, 7,
8, 0]   Distancia de la ruta:   40.799999999999999

Total Distance:  40.8
```

Figura 42 Resultado de ahorro n23 martes jueves

En índices originales, la ruta de martes/jueves es:

0-19-20-22-17-14-3-15-16-2-6-4-5-8-9-12-0

El resultado de rutas de miércoles después de la aplicación es:

```
== == == == == == == == == == == == == == Resultados == == == == == ==
= == == == == == = =
Ruta:   [0, 6, 5, 1, 4, 7, 2, 9, 3, 8, 0]   Distancia de la ruta:   31.9

Total Distance:  31.9
```

Figura 43 Resultado de ahorro n23 miercoles

En índices originales, la ruta de miércoles es:

0-12-11-1-10-13-5-21-7-18-0

- **Ejemplo 2: 50 proveedores (E-n51)**

Número de vehículo =1;
Recargo de transporte =0.1;
Coste variable de vehículo VVC=15;
Coste de carga y descarga s=0;
Número de proveedor =50;
Número de productos=50;
Vector de demandas=[100 200 300 400 500 60 70 80 90 100
110 120 130 140 150 160 170 180 190 200
210 220 230 240 250 260 270 280 290 300
310 320 330 340 350 360 370 380 390 400
410 420 430 440 450 460 470 480 490 500];

Realización de pruebas y resultados

Vector de coste de los productos =[50 400 480 470 460 450 440 430 42
410 40 390 380 310 360 350 340
330 320 310 300 29 280 270 260
250 240 230 22 210 200 190 180
170 160 150 140 130 120 110 10 90
80 70 60 50 40 30 20 10];

Las coordenadas de la fábrica y de los proveedores son:

Proveedor	Coordenada X	Coordenada Y
1	30	40
2	37	52
3	49	49
4	52	64
5	20	26
6	40	30
7	21	47
8	17	63
9	31	62
10	52	33
11	51	21
12	42	41
13	31	32
14	5	25
15	12	42
16	36	16
17	52	41
18	27	23
19	17	33
20	13	13
21	57	58
22	62	42
23	42	57
24	16	57
25	8	52
26	7	38
27	27	68
28	30	48
29	43	67
30	58	48
31	58	27
32	37	69

33	38	46
34	46	10
35	61	33
36	62	63
37	63	69
38	32	22
39	45	35
40	59	15
41	5	6
42	10	17
43	21	10
44	5	64
45	30	15
46	39	10
47	32	39
48	25	32
49	25	55
50	48	28
51	56	37

Tabla 2 Coordenadas del problema E-n51

El resultado que se ha obtenido con el algoritmo Milk Runs es:

```
MatDays =

  Columns 1 through 18

     0     2     3     4     5     0     0     0     0     0     0    12     0     0     0     0     0    18
     0     0     0     4     5     6     0     8     0    10     0     0    13    14    15    16    17     0
     1     2     3     0     5     0     7     0     9     0    11    12     0     0     0     0     0    18
     0     0     0     4     5     6     0     8     0    10     0     0    13    14    15    16    17     0
     0     2     3     4     5     0     0     0     0     0     0    12     0     0     0     0     0    18

  Columns 19 through 36

     0     0     0     0     0     0     0     0    27     0     0     0     0    32     0     0     0     0
    19    20    21     0    23    24    25    26    27    28     0    30    31     0    33    34    35     0
     0     0     0    22     0     0     0     0     0     0    29     0     0    32     0     0     0    36
    19    20    21     0    23    24    25    26    27    28     0    30    31     0    33    34    35     0
     0     0     0     0     0     0     0     0    27     0     0     0     0    32     0     0     0     0

  Columns 37 through 50

     0     0     0     0     0     0     0     0     0    46     0     0     0     0
    37    38     0     0     0     0     0     0     0    46    47     0     0     0
     0     0    39    40     0    42    43    44    45     0     0    48    49    50
    37    38     0     0     0     0     0     0     0    46    47     0     0     0
     0     0     0     0     0     0     0     0     0    46     0     0     0     0
```

Figura 44 Resultado de Milk Runs n51

A este resultado de Milk Runs, se aplica el algoritmo de ahorro tres veces, el resultado de rutas de lunes/viernes después de la aplicación es:

```
== == == == == == == == == == == == == == == Resultados == == == == == == == = =
= == == == == == = =
Ruta:   [0, 9, 3, 6, 5, 4, 1, 2, 8, 7, 0]   Distancia de la ruta:   126.0

Total Distance:  126.0
```

Figura 45 Resultado de ahorro n51 lunes/viernes

En índices originales, la ruta de lunes/viernes es:

0-46-4-18-12-5-2-3-32-27-0

El resultado de rutas de martes/jueves después de la aplicación es:

```
== == == == == == == == == == == == == == == Resultados == == == == ==
== == = == == == == == == = 
Ruta:   [0, 27, 2, 26, 5, 22, 8, 25, 10, 9, 20, 23, 13, 12, 24, 19, 21,
17, 4, 14, 15, 7, 16, 6, 11, 1, 28, 3, 18, 0]   Distancia de la ruta:
326.40000000000001

Total Distance:  326.4
```

Figura 46 Resultado de ahorro n51 martes/jueves

En índices originales, la ruta de martes/jueves es:

0-46-5-38-10-31-15-37-18-17-30-34-21-20-35-28-31-26-8-23-24-14-25-13-19-4-47-6-27-0

El resultado de rutas de miércoles después de la aplicación es:

```
== == == == == == == == == == == == == == == Resultados == == == == ==
== == = == == == == == == = 
Ruta:   [0, 20, 5, 17, 9, 7, 12, 1, 2, 10, 3, 13, 11, 22, 6, 21, 14, 19
, 15, 16, 18, 4, 8, 0]   Distancia de la ruta:   337.1

Total Distance:  337.1
```

Figura 47 Resultado de ahorro n51 miércoles

En índices originales, la ruta de miércoles es:

0-48-7-43-18-11-32-1-2-22-3-36-29-50-9-49-39-45-40-42-44-5-12-0

5. Conclusiones

A vista del estudio de investigación, los resultados obtenidos con el algoritmo y las comparaciones que se han hecho de estos resultados con las mejores soluciones conocidas, se puede sacar varias conclusiones:

1. La aplicación de Milk Runs para el transporte de mercancías es una condición imprescindible para poder recoger las mercancías de los proveedores en pequeños lotes.

2. Si se analiza solo la distancia total de la planificación, el resultado de Milk Runs es mucho mayor que una simple planificación de enrutamiento de vehículos, pero tiene la ventaja de poder visitar varias veces a un proveedor si la cantidad es grande, por lo tanto, el lote de mercancía que se trae es mucho menor, cumple con el principio de Lean Manufacturing sobre pedir mercancías justas para la producción, de esta forma, reduce el despilfarro de la producción.

3. El último paso de Milk Runs consiste en optimizar una planificación obtenida con las condiciones que se dan en el problema, es igual que un problema clásico de enrutamientos de vehículos, por lo tanto, podemos usar los algoritmos metaheurísticos que se usan en un problema VRP para el último paso de Milk Runs también, con estos algoritmos metaheurísticos, optimizamos la planificación.

4. El problema del enrutamiento de vehículos es un problema que las fábricas del sector automóvil se enfrentan en el día a día durante su funcionamiento, es muy importante que se lleve un control sobre el transporte de las mercancías desde los proveedores hasta la fábrica.

5. La filosofía de Just In Time es un método útil tanto para la gestión como para la organización de una empresa, y las herramientas de JIT ayudan a las empresas para eliminar los despilfarros o desperdicios de la empresa aumentando de este modo la rentabilidad de la empresa consiguiendo una producción “perfecta”, evita en la medida de lo posible las producciones con fallos, reproducciones, tiempo muerto durante cambios de procesos, tener que guardar un nivel alto de stock para asegurar los lotes de producciones.

6. Para recoger las mercancías de distintos proveedores, lo más lógico es aplicar el razonamiento de Milk Runs o Ruta lechera, de esta forma, aprovechar en mayor proporción la capacidad de los camiones en un uso

compartido. Un mismo camión recoge varias mercancías visitando varios proveedores en una misma ruta.

7. Para resolver los problemas de enrutamiento de vehículos, no es posible hacer una búsqueda exhaustiva de posibles soluciones, ya que cuando el número de puntos a visitar supera un límite, el número de combinaciones posibles es sumamente grande. Por ejemplo, si son 20 puntos, las posibles soluciones que existen son factorial de 20 que son 2.4×10^{18}, es imposible de hacer estos cálculos, aunque en el ordenador más potente del planeta Tierra. Lo más habitual es utilizar una heurística tal como la que se ha utilizado en este trabajo, heurística de ahorro.

8. Las heurísticas que se usan para resolver problemas de enrutamiento de vehículos, consisten en hacer una búsqueda orientada, hace un barrido en una zona de espacio de las posibles combinaciones de la solución y acercarse hacia la mejor solución ya que estas heurísticas no garantizan encontrar la solución óptima.

9. Los problemas de enrutamiento son muy complicados de resolver en la vida real, la complicación aumenta conforme a las restricciones que se conocen para las planificaciones de las rutas.

10. La heurística de ahorro consiste en un algoritmo simple que se intentan agrupar aquellos puntos que tienen mayor ahorro en distancia enlazando estos dos puntos en una misma ruta cumpliendo todas las restricciones que se dan.

11. La heurística de ahorro es muy eficaz cuando el número de proveedores del problema de enrutamiento de vehículos con capacidades son relativamente bajo, como unos 30 proveedores, se aproxima bastante bien a la mejor solución del problema. Sin embargo, cuando el número supera 50 proveedores, las soluciones que se obtienen son relativamente “malas”.

Para una empresa que tiene una cadena de suministros de piezas, es muy importante que sea la empresa quien se encargue de los transportes, ya que además de reducir los costes en transporte, la empresa puede hacer pedidos en lotes pequeños sin tener en cuenta las restricciones de cantidad mínima para el envío de los proveedores, reduciendo de esta forma el nivel de stock innecesario de ese componente; otra ventaja de hacerlo así es el control absoluto por la parte de la empresa sobre el seguimiento de los envíos, cuándo se produce una incidencia, la fábrica lo puede saber a primera hora de la incidencia. Una incidencia puede ocurrir en cualquier empresa y en cualquier hora, cuanto antes se sepa, más tiempo tiene la fábrica para tratar de minorizar el efecto o consecuencia negativa que provoca esa incidencia, tener más tiempo de reacción ante las incidencias es una manera de optimización también en el proceso de la producción.

Conclusiones

En resumen, que las empresas se encarguen del transporte de los suministros implica tener que dedicar recursos a un servicio que se puede externalizar, pero para aplicar la filosofía de JIT, convienen hacerlo. El resultado será que se aumenta la rentabilidad de la empresa disminuyendo gastos en el transporte.

6. Bibliografías

1. Monden, Yasuhiro ; traducción: Gloria Hillers ; [prólogo de Pedro (1996). El "just in time" hoy en Toyota (2 edición). Bilbao: Deusto. p. 22. ISBN 84-234-1442-6.

2. Colón Parra, Nicolás Bartolomé (febrero de 2012). «Implantación de metodologías Lean Manufacturing en el almacén de logística de una empresa aeronáutica». Escuela Superior de Ingenieros de Sevilla. Consultado el 2 de octubre de 2019.

3. Aguilar, Carlos (2016). «¿Qué herramientas utilizo? Kaizen, 5s, Seis Sigma, TPM». Causa & Efecto: 10.

4. Ronald H. Ballou y otros: "Logística, administración de la cadena de suministro" Edición: 5 - Pearson Educación 2004 - ISBN 970-26-0540-7, ISBN 978-970-26-0540-9

5. Hill, A. V., & Vollmann, T. E. (1986). Reducing vendor delivery uncertainties in a JIT environment. Journal of Operations Management, 6(3-4), 381-392.

6. Beck, J. C., Prosser, P., & Selensky, E. (2002, October). Graph transformations for the vehicle routing and job shop scheduling problems. In International Conference on Graph Transformation (pp. 60-74). Springer, Berlin, Heidelberg

7. VRP-CW: ishelo. https://github.com/ishelo/VRP-CW.

7. ANEXO I CÓDIGO DEL ALGORITMO DE PYTHON

```
import csv
from operator import itemgetter
class Vrp():
   # ----------Datos iniciales----------
   def __init__(valor):
      valor.prov = 100                          # Número de clientes
      valor.tons = 220                          # Capacidad de carga del vehículo
      valor.Limitekilom = 10000                 # Limitación de kilómetros de una ruta
      valor.distance = []                       # Tabla de distancia
      valor.q = [0,10,7,13,19,26,3,5,9,16,
            16,12,19,23,20,8,19,2,12,17,
            9,11,18,29,3,6,17,16,16,9,
            21,27,23,11,14,8,5,8,16,31,
            9,5,5,7,18,16,1,27,36,30,
            13,10,9,14,18,2,6,7,18,28,
            3,13,19,10,9,20,25,25,36,6,
            5,15,25,9,8,18,13,14,3,23,
            6,26,16,11,7,41,35,26,9,15,
            3,1,2,22,27,20,11,12,10,9,17]
       # Cantidades de los proveedores, el primero es de la fábrica
      valor.savings = []                        # Ahorros
```

```
    valor.Routes = []                          # Rutas
    valor.Cost = 0                             # Total distancia de las rutas


# ----------Importar tabla de distancia----------
def datainput(valor):
    with open("data2.csv", "r") as csvfile:
        reader = csv.reader(csvfile)
        for line in reader:
            line = [float(x) for x in line]
            valor.distance.append(line)
# ----------Algoritmo de ahorro----------
def savingsAlgorithms(valor):
    saving = 0
    for i in range(1, len(valor.q)):
        valor.Routes.append([i])
         for i in range(1, len(valor.Routes) + 1):
          # Calcular ahorros usando 〖ΔC〗_ij=C_i0+C_0j-C_ij
        for j in range(1, len(valor.Routes) + 1):
            if i == j:
                pass
            else:
                saving = (valor.distance[i][0] + valor.distance[0][j]) - valor.distance[i][j]
                valor.savings.append([i, j, saving])          # Almacenar ahorros en vector
```

```
valor.savings = sorted(valor.savings, key=itemgetter(2), reverse=True)                #
Ordenar los ahorros en orden descendente

for i in range(len(valor.savings)):

print(valor.savings[i][0],'--',valor.savings[i][1], "  ",valor.savings[i][2])

    # Imprimir en pantalla los ahorro

for i in range(len(valor.savings)):

   startRoute = []

   endRoute = []

   routeDemand = 0

   for j in range(len(valor.Routes)):

      if (valor.savings[i][0] == valor.Routes[j][-1]):

         endRoute = valor.Routes[j]

      elif (valor.savings[i][1] == valor.Routes[j][0]):

         startRoute = valor.Routes[j]


      if ((len(startRoute) != 0) and (len(endRoute) != 0)):

         for k in range(len(startRoute)):

            routeDemand += valor.q[startRoute[k]]

         for k in range(len(endRoute)):

            routeDemand += valor.q[endRoute[k]]

         routeDistance = 0

         routestore = [0]+endRoute+startRoute+[0]

         for i in range(len(routestore)-1):

            # print(routestore[i],routestore[i+1])

            # print(valor.distance[routestore[i]][routestore[i+1]])
```

```
                routeDistance += valor.distance[routestore[i]][routestore[i+1]]


            #print(routestore,"== ==:",routeDistance)


            if (routeDemand <= valor.tons) and (routeDistance <= valor.Limitekilom):    # Modificar rutas según los recursos
                valor.Routes.remove(startRoute)
                valor.Routes.remove(endRoute)
                valor.Routes.append(endRoute + startRoute)
            break


    for i in range(len(valor.Routes)):
        valor.Routes[i].insert(0, 0)
        valor.Routes[i].insert(len(valor.Routes[i]), 0)


# ----------Sacar los resultados----------


def printRoutes(valor):
    for i in valor.Routes:
        costs = 0
        for j in range(len(i)-1):
            costs += valor.distance[i[j]][i[j+1]]
        print("Ruta:  ",i,"  Distancia de la ruta:  ",costs)


def calcCosts(valor):
```

```
    for i in range(len(valor.Routes)):
        for j in range(len(valor.Routes[i]) - 1):
            valor.Cost += valor.distance[valor.Routes[i][j]][valor.Routes[i][j + 1]]

    print("\nTotal Distance: ", round(valor.Cost, 3))

# -----------Funcion Master----------------------

def start(valor):                   # Usar las otras funciones
    print("== == == == == == == == == == == == == == == == Importando Datos == == == == == == == = == == == == == == == =")
    valor.datainput()
    print("== == == Tabla de distancias == == ==")
    for i in valor.distance:
        print(i)
    print("== == == Cantidades == == ==")
    print(valor.q)
    print("== == == Restricciones == == ==")
    print("Máxima capacidad de carga del vehículo：",valor.tons)
    print("Máximo kilometro de una ruta：", valor.Limitekilom)
    print("== == == == == == == == == == == == == == == == Ahorros == == == == == == == = == == == == == == == =")
    valor.savingsAlgorithms()       # Calcular las rutas según ahorros
    print("== == == == == == == == == == == == == == == == Resultados == == == == == == = == == == == == == == =")
    valor.printRoutes()
```

```
valor.calcCosts()

valor.datainput()
```

8. ANEXO II CÓDIGO DEL ALGORITMO DE MATLAB

```
% -----------------Datos iniciales--------------------

nvehicle=1;

h=10;

VVC=1;

s=0;

nvendor=21;

nitem=21;

d=[110 70 80 140 210 40 80 10 50 60 120 130 130 30 90 210 100 90 250 180 70];


c=[1 2 3 4 5 6 7 8 9 1 1 2 3 4 5 6 7 8 9 2 2];


pk=zeros(21,21);

for i=1:nvendor

   for j=1:nvendor

      if i==j

         pk(i,j)=1;

      end

   end

end


%------------------Matriz de distancias----------------------
```

```
distancia=[        ];
% for alpha=0:0.1:1;
 alpha=0;

%-----------------------cálculo de dtmx---------------------------
for k=1:nvendor
   dtmx(k)=time(1,k+1)+time(k+1,1);
end
dtmx;

%-----------------------cálculo de dtmn-------------------------
MatDist=time;
MatDist(:,[1])=[];
MatDist([1],:)=[];
for k=1:nvendor

   VectDist=MatDist(k,:);
   OrdDist=sort(VectDist);
   dtmn(k)=OrdDist(2)+OrdDist(3);
end
dtmn;

% -----------------------cálculo de mnn---------------------------
for k=1:nvendor
  sum=0; setup=0;
  for i=1:nitem
```

```
    sum=sum+d(i)*c(i)*pk(k,i);
    setup=setup+s*pk(k,i);
  end
  mn*n(k)*=sqrt((h*sum)/(setup+2*VVC*dtmx(k)));



end
mnn;

% ----------------------------cálculo de *mxn*---------------------
for k=1:nvendor
  sum=0; setup=0;
  for i=1:nitem
    sum=sum+d(i)*c(i)*pk(k,i);
    setup=setup+s*pk(k,i);
  end
  *mxn(k)*=sqrt((h*sum)/(setup+2*VVC*dtm*n(k)*));



end
*mxn*;

% ---------------------------Calculo de *n*----------------------
for k=1:nvendor
   *n(k)*=mn*n(k)*+alpha*(*mxn(k)*-mn*n(k)*);
   *n(k)*=round(*n(k)*);
```

```
   if n(k)>5
      n(k)=5;
   end
end
n


% -----------Asignación de proveedores en rutas de semana----------
MatDays=zeros(5,6);
for k=1:nvendor


   if n(k)==1
      MatDays(3,k)=k;
   else
      if n(k)==2
         MatDays(2,k)=k;
         MatDays(4,k)=k;
      else
         if n(k)==3
            MatDays(1,k)=k;
            MatDays(3,k)=k;
            MatDays(5,k)=k;
         else
            if n(k)==4
            MatDays(1,k)=k;
            MatDays(2,k)=k;
            MatDays(4,k)=k;
```

```
            MatDays(5,k)=k;
            else
               if n(k)==5
                  MatDays(1,k)=k;
                  MatDays(2,k)=k;
                  MatDays(3,k)=k;
                  MatDays(4,k)=k;
                  MatDays(5,k)=k;
               end
            end
         end
      end
   end
end
MatDays;
% end

VectMon=MatDays([1],:);
VectTue=MatDays([2],:);
VectWed=MatDays([3],:);
VectThu=MatDays([4],:);
VectFri=MatDays([5],:);
```