Revista Electrónica de Comunicaciones y Trabajos de ASEPUMA. **Rect@**

Volumen 17 (2016), páginas 57 a 80
.

MODELO REAL DE PLANIFICACIÓN Y RUTAS BI-OBJETIVO. EQUILIBRIO ENTRE COSTES Y PREFERENCIAS DE CLIENTES

AMAYA MARTÍNEZ PURAS
amayamar@ubu.es
Universidad de Burgos/Departamento de Didácticas Específicas
C/ Villadiego s/n, 09001 Burgos, España

JOAQUÍN ANTONIO PACHECO BONROSTRO
jpacheco@ubu.es
Universidad de Burgos/ Departamento de Economía Aplicada
Plaza Infanta Elena s/n, 09001 Burgos, España)

Recibido (16/01/2016)
Revisado (22/03/2016)
Aceptado (03/04/2016)

RESUMEN: Un modelo bi-objetivo para el diseño de rutas diarias de una empresa a lo largo de un período de planificación es analizado. Este modelo viene motivado por un problema real de diseño de las rutas de una empresa de Análisis Químicos a lo largo de un horizonte de planificación y la asignación de los calendarios de visita a sus clientes. Los dos objetivos bajo consideración son: minimizar el coste del transporte y la reducción de las modificaciones sobre los calendarios actuales de los clientes. Para su resolución, se ha desarrollado una metodología ad hoc basada en búsqueda tabú en el contexto del PVRP (Periodic Vehicle Routing Problem). El método de solución fue desarrollado por aplicación de búsqueda tabú combinada con la estrategia MOAMP (MultiObjective Adaptative Memory Procedure) y los resultados son comparados con una implementación de NSGA-II (Non-dominated Sorting Genetic Algorithm), una reconocida buena estrategia de optimización multi-objetivo.

Palabras claves: Problemas bi-objetivo, Rutas, Búsqueda tabú, MOAMP, NSGA II.

ABSTRACT: A bi-objective model for the design of daily routes of a company over a planning period is analyzed. This model is motivated by a real design problem routes Chemical Analysis Company over a planning horizon and allocation schedules visit to its customers. The two objectives under consideration are: minimizing transport costs and reducing modifications on current customer schedules. For resolution, it has developed an ad hoc methodology based on tabu search in the context of PVRP (Periodic Vehicle Routing Problem). The solution method was developed by application of combined tabu search with MOAMP (Multiobjective Adaptive Memory Procedure) strategy and the results are compared with an implementation of NSGA-II (Non-dominated Sorting Genetic Algorithm), a well-known approach to multi-objective optimization.

Keywords: Bi-objective Problems, Routes, Tabu Search, MOAMP, NSGA II.

1. Introducción

1.1. *Evolución del PVRP en la literatura*

El PVRP es una generalización del conocido Vehicle Routing Problem (VRP) en el que las rutas deben diseñarse sobre múltiples días o periodos, es decir, en un horizonte de planificación. Una definición simple del problema la encontramos en Gulczynski *et al* (2011), quienes lo definen como una variante del VRP, en la cual, los clientes pueden requerir los servicios en múltiples ocasiones dentro de un horizonte de planificación.

Este problema es una de las variantes de mayor dificultad de VRP, debido a que conlleva dos niveles de decisión que están directamente conectados entre sí, como se menciona en Alegre *et al.* (2007). Los dos niveles son: la selección de los días en que va a ser visitado cada cliente y el diseño de las rutas correspondiente a cada uno de los días.

Beltrami and Bodin (1974) plantearon por primera vez un problema que consideraba un período de planificación mayor a un día. Aportaron dos procedimientos para la resolución del problema: en el primero, desarrollaron las rutas y después asignaron los días, en el segundo, asignaron aleatoriamente los clientes a los días y posteriormente resolvieron el VRP utilizando el algoritmo de Clarke and Wright (1969). En los años siguientes, se publicaron algunos trabajos que continúan en esta línea. Algunos de ellos son los de Russell and Igo (1979), Christofides and Beasley (1984), Tan and Beasley (1984) y Golden and Wasil (1987). Podemos considerar que estos trabajos forman parte de la primera etapa del PVRP.

En la década de los 90's, Russell and Gribbin (1991) proponen una heurística de cuatro etapas para la solución de un PVRP. Además, surgieron trabajos que representaron una pequeña revolución al generar métodos específicos para el PVRP, así como instancias unificadas que han sido utilizadas posteriormente. Chao *et al* (1995) presentan un método de dos fases en el que se introduce el algoritmo "Record-to-Record" y posteriormente, Cordeau *et al* (1997) utilizan una heurística basada en búsqueda tabú, resultando ser una de sus principales aportaciones al permitir el estudio de soluciones no factibles durante la búsqueda. Ambos algoritmos han sido un punto de referencia para heurísticas y algoritmos posteriores.

Ya en el siglo XXI encontramos un incremento en la literatura dedicada al PVRP, surgiendo un mayor número de enfoques y métodos para su solución. Drummond *et al* (2001) plantea un algoritmo genético en paralelo. Bertazzi *et al* (2004) proponen un algoritmo constructivo con mejora incluida, en el cual, de forma sencilla se pueden encontrar soluciones de buena calidad y además tiene en cuenta el problema de no factibilidad que podría generarse. Blakeley *et al* (2003) presentan un software experto para la resolución de este problema utilizando un método de dos fases: en la primera, efectúan la asignación de días y trabajadores a cada uno de los clientes y posteriormente construyen las rutas y Nuortio *et al* (2006) consideran las dos fases en orden inverso.

En el trabajo de Alegre *et al* (2007) se muestra una heurística basada en búsqueda dispersa diseñada para trabajar con amplios horizontes de planificación. La búsqueda dispersa se aplica al proceso de asignación y como consecuencia a los VRP que surgen, y que son resueltos por una heurística basada en GRASP (Greedy Randomized Adaptative Search Procedure) y búsqueda local. Ronen and Goodhart (2008) presentaron un trabajo con 3 fases: diseño de clusters, asignación de días y patrones de distribución y desarrollo de rutas mediante un software comercial. Pourghaderi *et al* (2008) proponen un nuevo algoritmo constructivo con mejora incluida utilizando búsqueda dispersa. Posteriormente Méndez *et al* (2009) utilizan algoritmos meméticos para la solución del problema en dos etapas.

Otra estrategia de solución que se ha desarrollado en los últimos años es la búsqueda por entornos variables (Variable Neigborhood Search VNS). Una de las primeras aplicaciones en este sentido es la propuesta por Hemmelmayr *et al* (2009). Peixoto (2010) utiliza una metodología de dos fases primero se realiza la asignación de días de visita y posteriormente se diseñan las rutas. En Pirkwieser (2010) se utiliza el denominado VNS Multinivel (Multilevel VNS, MVNS).

La gran variedad de heurísticas para solucionar el PVRP, también incluye la optimización de colonia de hormigas (Ant Colony Optimization ACO). Así, Yu and Yang (2011) presentan una aplicación interesante que denominan Improved Ant Colony Optimization (IACO), que consiste en añadir búsqueda local a la estrategia ACO.

Un texto que resulta interesante es el de Gulczynski *et al* (2011). En este trabajo se presenta una heurística que incluye uso de programación entera para formular partes del problema, así como búsqueda local con amplios vecindarios.

El trabajo de Meyer (2012) presenta un enfoque muy diferente a los anteriormente mencionados y es básicamente constructivo. Utiliza un planteamiento basado en programación por restricciones (programming constraint PC). La heurística para resolver el problema empieza de manera constructiva y continúa con un procedimiento de búsqueda local en un entorno de gran tamaño.

Vidal *et al* (2012) presentan un enfoque genético en su trabajo en el que desarrollan un método que nombran Búsqueda Genética Híbrida con Control de Diversidad Adaptativo (Hybrid Genetic Search with Adaptative Diversity Control, HGSADC). Su objetivo es aprovechar el amplio espectro de búsqueda de los algoritmos genéticos combinados con las agresivas capacidades de búsqueda de las metaheurísticas basadas en entornos.

Rademeyer and Bennetto (2012) utilizan un algoritmo memético para la solución del PVRP donde se consideran restricciones operativas reales tales como uso de instalaciones y capacidad. Vahed *et al* (2012) proponen dar solución a un PVRP mediante una metodología basada en reencadenamiento de trayectorias (Path Relinking PR).

Pacheco *et al* (2012) presentan un modelo real interesante. Este problema puede ser considerado como una generalización del CVRP y también como un caso particular del PVRP. Los autores proponen una heurística adaptada a este modelo que combina GRASP con PR. Posteriormente para este mismo modelo Pacheco *et al* (2014) proponen un método basado en VNS con memoria.

En Hemmelmayr *et al* (2013) se analiza otra variante interesante con facilidades intermedias. Este modelo tiene aplicaciones en recogida de basuras y en logística inversa. Los autores proponen un método basado en VNS. En Yao *et al* (2013) se propone un método que combina ACO con búsqueda local. En el trabajo de Cacchiani *et al* (2014) se presenta un método de solución que parte de considerar el modelo como un problema de cubrimiento de conjuntos (Set Covering Problem SCP) con restricciones adicionales. El modelo combina relajación lineal, generación de columnas y búsqueda tabú.

También, aunque en menor medida, existen métodos exactos que se pueden encontrar en la literatura para la resolución de PVRP. La primera aparece en el trabajo de Francis *et al* (2006) quienes plantean un PVRP donde los posibles patrones de entrega para cada uno de los clientes son considerados como variable para ser elegida por éstos. Baldacci *et al* (2011) proponen diferentes tipos distintos de relajaciones que permiten resolver algunas instancias de más de 100 clientes. Por último Ngueveu *et al* (2013) presentan un método Branch-and-Cut para una variante del PVRP, donde cada cliente es servido una vez.

Finalmente se ha de indicar que una reciente revisión de trabajos sobre el PVRP se puede encontrar en la recopilación de Campbell and Wilson (2014).

1.2. *Enfoque multi-objetivo para el PVRP en la literatura*

Es extensa la literatura de variantes de modelos de rutas con múltiples objetivos. Una recopilación de problemas de rutas multiobjetivo hasta inicios de los 90 se puede encontrar en Current and Marsh (1993). Otra más reciente recopilación de trabajos sobre problemas de rutas bi-objetivo se puede encontrar en Jozefowiez *et al* (2008). En general los trabajos en este campo se relacionan principalmente con transporte escolar, transporte de mercancías peligrosas, recogida de residuos y logística inversa. Además

de la reducción del coste de las operaciones se buscan otros objetivos relacionados con el nivel de servicio, peligrosidad, etc. Recientemente cabe destacar los trabajos de Pacheco *et al* (2013) sobre transporte escolar, o Goméz *et al* (2015) sobre recogida de basuras.

No existen muchas referencias sobre problemas multi-objetivos en el contexto específico del PVRP. No obstante se pueden encontrar algunas aportaciones interesantes. Así, en el trabajo de Smilowitz et al (2013) se presenta un modelo donde además de los costes de las operaciones se tienen en cuenta objetivos referidos a la gestión del personal contratado para realizarlas. Más recientemente en Hsieh *et al* (2015) se presenta una generalización del PVRP con aplicaciones a diseño de rutas de coches patrullas. Además de minimizar el coste de las rutas se busca maximizar la cantidad de carriles cubierto por estas rutas.

El modelo que se considera en este trabajo es un modelo, motivado por problema real, que se ajusta al PVRP pero con características (objetivos y restricciones) diferentes a los analizados antes en la literatura y que entendemos interesantes. Concretamente los objetivos buscados son minimizar el coste de las operaciones y maximizar las preferencias de los clientes. Para resolverlo se va a diseñar un método basado en la estrategia MOAMP (MultiObjective Adaptative Memory Procedure) propuesta inicialmente en el Caballero *et al* (2003). Posteriormente, para evaluar la calidad de este método se diseña un segundo método basado en la conocida estrategia evolutiva NSGA-II.

El resto del trabajo se organiza de la forma siguiente: en la sección 2 se describe el problema y su motivación; en la sección 3 se describen algunos problemas similares existentes en la literatura; en la sección 4 se establece el planteamiento del problema así como diferentes definiciones; en las secciones 5, 6 y 7 se describe con detalle el método MOAMP propuesto (en la 5 la descripción genérica, en la 6 los procedimientos principales y en la 7 los procedimientos de rutas); en la sección 8 se describe el método NSGA II para este problema; en las secciones 9 y 10 se muestran los resultados de diferentes pruebas computacionales (en la 9 con instancias ficticias, y en la 10 con datos reales). Finalmente en la sección 11 se establecen las conclusiones.

2. Motivación y descripción del problema

En este trabajo se analiza un modelo bi-objetivo para el diseño de rutas diarias de una empresa a lo largo de un período de planificación. Este modelo está basado en un problema real propuesto a los autores de este trabajo por una empresa de Análisis Químicos de Salamanca, España. La empresa está interesada en mejorar la planificación actual, es decir, el diseño las rutas diarias así como asignación de los calendarios de visita a sus clientes. El objetivo de esta nueva planificación es reducir los costes de las rutas. Pero por otro lado la empresa estaba interesada en evitar excesivos cambios en los calendarios actuales.

Normalmente cuando una empresa pide que se racionalice su planificación de productos/servicios a clientes a lo largo de un horizonte temporal, se requiere la reducción de los costes de las operaciones (coste de las rutas). Sin embargo, se ha observado que esta nueva planificación suele ser significativamente diferente de la empleada hasta ese momento. Es decir, para muchos clientes supone la modificación de sus calendarios o fechas de entrega. Estas modificaciones no suelen ser bien recibidas por los clientes y puede suponer una cierta pérdida de imagen de la empresa.

Por consiguiente, al plantear el problema debemos tener en cuenta dos objetivos: reducción de los costes y minimización del número de cambios realizados. Así planteado, el problema es claramente bi-objetivo: reducción de los costes y reducción de las modificaciones sobre los calendarios actuales de los clientes.

Otros aspectos que debemos considerar en este problema planteado son los siguientes. La empresa suele realizar unos 1200 servicios (visitas) al mes a 100 clientes, la mayoría en el norte de España. Los servicios son efectuados por 10 técnicos, quienes conduciendo su propio vehículo son los encargados de realizar las rutas de servicios. Todas las rutas comienzan y finalizan en la empresa.

Los servicios se realizan de lunes a viernes y son de tres tipos: recogida de muestras (A), control de plagas (B) y control de calidad (C) con una duración aproximada de 15, 30 y 60 minutos respectivamente. El horizonte de planificación es mensual, 20 días (4 semanas) y la jornada laboral es como máximo de 9 horas e incluye el tiempo de conducción y de servicio. Además, debemos tener en cuenta que no todos los técnicos están cualificados para realizar los tres servicios. Concretamente, 6 técnicos están capacitados para realizar el servicio "A" solamente, 2 pueden hacer el "A" y "B" pero no el "C" y sólo 2 técnicos están cualificados para realizar los tres tipos de servicio.

Por otro lado, para cada cliente se considera un conjunto de posibles calendarios en los que pueden ser visitados (calendarios definidos principalmente por la frecuencia). En la figura 1, se muestra el mapa con las localizaciones de los clientes representados en distintos colores según su frecuencia de visitas. El significado de los colores es: rojo (empresa); azul (todos los días); amarillo (3 días a la semana); verde (2 días a la semana); naranja (1 vez a la semana); gris (1 vez cada 15 días) y negro (1 vez al mes).

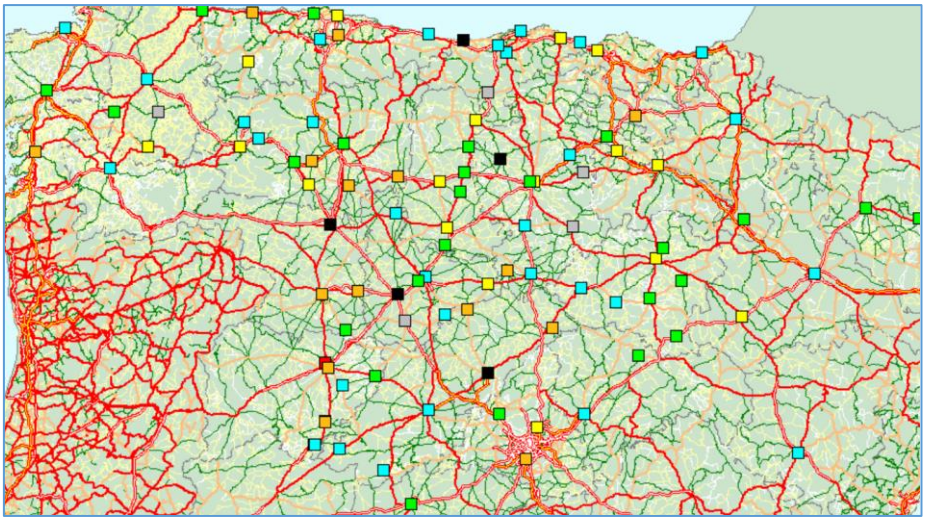

Figura 1. Localización de los clientes

La frecuencia de visitas y el tipo de servicio que requieren los clientes se distribuye como se muestra en la tabla 1.

Tabla 1. Frecuencias y tipos de servicio

FRECUENCIAS	CLIENTES	SERVICIO
Todos los días	18	A
3 días a la semana	31	A
2 días a la semana	27	A o B
1 día a la semana	14	B
1 vez cada dos semanas	5	B o C
1 vez al mes	5	C

Para los clientes que son visitados todas las semanas, en cada calendario, los días de visita deben ser los mismos cada semana. Por ejemplo, si un cliente es visitado dos días a la semana, en todas las semanas de cada calendario, las visitas deben ser los mismos días: o bien Lunes y Miércoles, o bien Martes y Jueves, o bien Miércoles y Viernes, o bien Lunes y Jueves, etc. De igual forma los clientes que son visitados una vez cada 2 semanas el día de la semana debe ser el mismo.

No todos los clientes con la misma frecuencia de visita tienen los mismos conjuntos de calendarios. Así por ejemplo, con dos días a la semana de visita podemos encontrarnos con los siguientes patrones de conjuntos de calendario):

$$P1 = \{(L, X), (M, J) (X, V)\}, \text{ o } P2 = \{(L, J), (M, V)\}.$$

En el patrón P1 el cliente puede ser visitado o bien Lunes y Miércoles, o bien Martes y Jueves o bien Miércoles y Viernes. Mientras que en el patrón P2 el cliente puede ser visitado o bien Lunes y Jueves, o bien Martes y Viernes.

Finalmente hay que indicar que los servicios realizados apenas suponen espacio o peso en el vehículo: se recogen pequeñas muestras de los análisis realizados. Por tanto no se consideran restricciones de capacidad.

3. Primeros indicios en la literatura de problemas similares

El control las diferencias en las asignaciones a calendarios respecto a los de una solución preestablecida en el contexto del PVRP solamente ha sido tratado en el trabajo de Gulczynsky *et al* (2011), desde nuestro conocimiento. No obstante en esta referencia no se plantean estas diferencias como una nueva función objetivo sino como una restricción sino como una restricción o como una penalización que se añade a la función objetivo original de costes. Al modelo resultante lo denominan PVRP con restricciones de reasignaciones (PVRP with Reassignments Constraints PVRP-RC) y consideran 3 variantes:

- PVRP-RCH (con restricciones fuertes, *hard*): se busca minimizar el coste con la restricción adicional de que el número de asignaciones a calendarios diferentes de la solución preestablecida sea menor que una constante W prefijada.
- PVRP-RCS (con restricciones, *soft*): en este caso, se añade a la función objetivo del coste una penalización por asignación diferente. Esta penalización depende del número de veces que cada cliente es visitado a lo largo del periodo del horizonte temporal.
- PVRP-RCR (con reasignaciones limitadas, *restricted*): permite reasignaciones diferentes en clientes que son visitados sólo una vez. Estos clientes son los que generan menos problemas.

Otro trabajo que presenta ciertas similitudes con el que aquí se trata es el de García *et al* (2013). En este trabajo se analiza un problema bi-objetivo de una empresa de repostería que debe planificar la entrega de pedidos a delegaciones situadas en el norte de España. El período de planificación es semanal (de lunes a viernes). Cada pedido viene determinado por una fecha de entrega inicial (día en que se debe efectuar el pedido) y el número de pallets a entregar. Con la finalidad de minimizar el coste de las operaciones, se permite que los pedidos sean entregados en la fecha inicial o el día anterior (excepto los del lunes que no pueden adelantarse al domingo). Estos adelantos producen stocks. El problema, por tanto, consiste en determinar la fecha definitiva de entrega de cada pedido y diseñar las rutas diarias con dos objetivos: minimizar la distancia total y el stock total.

Este problema puede verse como un caso más particular de un PVRP. Se puede considerar que cada cliente dispone de dos posibles calendarios de un día cada uno: la fecha inicial o el día anterior (excepto los clientes con fecha inicial el lunes que sólo tienen un calendario). El stock puede ser considerado como una forma de minimizar o controlar los cambios sobre la fecha inicial (calendario inicial), ponderando estos cambios por el número de pallets a entregar.

No obstante, en nuestro trabajo se considera un modelo más general, ya que consideramos periodos de planificación mayor (1 mes) y los conjuntos de calendarios son más diversos (varios calendarios para cada cliente y frecuencias de visita diferentes).

4. Notación, planteamiento y definiciones

Se utiliza la siguiente notación, adaptada del trabajo Gulczynsky *et al* (2011), para plantear nuestro problema:

T : el número de días del período de planificación

V : conjunto de puntos del problema; $V = \{0,1,\ldots,n\}$ donde 0 representa el depósito, y $\{1,\ldots,n\}$ las localizaciones de los clientes. (Para simplificar de ahora en adelante identificaremos los clientes con sus respectivas localizaciones)

Λ_i: Conjunto de patrones o calendarios de visita posibles para el cliente i, i = 1,...,n

d_{ij} : distancias entre los puntos i y j, $i, j \in V$

t_{ij} : distancias entre los puntos i y j, $i, j \in V$. En este tiempo se incluye el tiempo de servicio del cliente j cuando $j \neq 0$.

H_i : Tipo de servicio requerido por el cliente i, i = 1,..., n. Por lo comentado anteriormente H_i = A, B, o C.

m : Tamaño de la flota de vehículos (o número de técnicos)

K_l : Conjunto de tipos de servicio que es capaz de hacer el conductor del vehículo l, l = 1,...,m. Por lo comentado anteriormente K_l = { A } o K_l = { A, B } o K_l = { A, B, C }.

$Tmax$: Tiempo máximo de cada ruta (incluyendo los tiempos de servicios).

Se define $nc_i = |\Lambda_i|$ y denotamos por λ_{ij}, j = 1,..., nc_i, a cada uno de los posibles calendarios del cliente i, es decir $\Lambda_i = \{\lambda_{ij},\ j = 1, \ldots, nc_i\}$.

Cada solución S va a estar definida por el calendario asignado a cada cliente y el conjunto de rutas diarias. Para cada solución S denotemos por por $\lambda_{iC(i)}$ el calendario asignado al cliente i, i = 1,…,n. Así mismo denotamos por R_t el conjunto de sus rutas para cada día $t \in \{1, \ldots, T\}$; y por $V(r)$ y $E(r)$ respectivamente el conjunto de puntos y arcos para cada ruta $r \in R_t$. Se debe cumplir las siguientes restricciones para que S sea factible:

- Cada ruta r empieza y finaliza en el depósito 0, $\forall\ r \in R_t, t \in \{1, \ldots, T\}$ (1)

- $\forall i \in \{1, \ldots, n\}$ y $t \in \lambda_{i\,C(i)}$ existe una $r \in R_t$ verificando que $i \in V(r)$ (2)

- Sea $l(r)$ el vehículo asignado a la ruta r, entonces $H_i \in K_{l(r)}$

$$\forall\ i \in V(r), \forall\ r \in R_t, t \in \{1, \ldots, T\} \quad (3)$$

$$-\ \sum_{(i,j)\in E(r)} t_{ij} \leq Tmax \qquad \forall\ r \in R_t, t \in \{1, \ldots, T\} \quad (4)$$

Obsérvese que (2) obliga que cada cliente sea servido por una ruta de cada día del calendario que tiene asignado; (3) asegura que el técnico-conductor de cada ruta pueda realizar todos los servicios de esa ruta; finalmente (4) hace referencia al tiempo máximo de cada ruta (que incluye los tiempos de servicio), es decir a la duración máxima de la jornada laboral. Los objetivos que se plantean son dos: minimizar el coste total de las rutas, que es equivalente a minimizar la distancia total recorrida, y el número de clientes a calendarios con asignaciones a diferentes de las establecidas por una solución predefinida S_0 (en nuestro caso la solución actual). Más formalmente ambos objetivos se formulan de la forma siguiente

$$\text{minimizar} \sum_{t=1}^{T} \sum_{r \in R_t} \sum_{(i,j)\in E(r)} d_{ij} \quad (5)$$

$$\text{minimizar}\ dif(S, S_0) \quad (6)$$

donde $dif(S, S')$ se puede definir como el número de clientes con calendarios diferentes en S y S'.

Dada una solución cualquiera S denotemos por $f_1(S)$ al valor de la función objetivo "económica", es decir, la definida por (5) y $f_2(S)$ la función objetivo definida por (6). Sean S y S' dos soluciones, se dice que S domina a S' si: a) $f_1(S) \leq f_1(S')$, b) $f_2(S) \leq f_2(S')$ y c) o bien $f_1(S) < f_1(S')$ o bien $f_2(S) < f_2(S')$. Una solución es eficiente si no existe otra que la domina. El problema que aquí se plantea consiste en encontrar soluciones lo más cercanas posibles al conjunto de soluciones eficientes del problema que se denomina 'curva de eficiencia'. Se definen:

- f_1^{min} y f_1^{max} como el mínimo y máximo valor encontrado para f_1 en el conjunto de soluciones no dominadas.

- f_2^{min} y f_2^{max} como el mínimo y máximo valor encontrado para f_2 en el conjunto de soluciones no dominadas.

$$F_\lambda(S) = \max\left\{\lambda \cdot \frac{f_1(S) - f_1^{\min}}{f_1^{\max} - f_1^{\min}}, (1-\lambda)\frac{f_2(S) - f_2^{\min}}{f_2^{\max} - f_2^{\min}}\right\} \qquad \text{donde } \lambda \in (0,1).$$

5. Método de solución

La estrategia que se propone en este trabajo para resolver nuestro nuevo modelo, es una adaptación del procedimiento MOAMP para problemas multiobjetivo. Este procedimiento crea, básicamente, una aproximación a la curva de eficiencia, enlazando la ejecución de una serie de procedimientos de búsqueda tabú, es decir, procedimientos que utilizan como solución inicial, la solución final obtenida en la ejecución anterior. Posteriormente, se realiza una exploración en torno a las soluciones obtenidas hasta ese momento.

Las ideas en las que se basa MOAMP son: 1) El principio de proximidad de puntos eficientes, en un entorno o vecindario de una solución eficiente, se puede encontrar otra solución eficiente y 2) la solución que minimiza la distancia L_∞ normalizada y/o ponderada al punto ideal es también eficiente.

Obsérvese que, precisamente la función F_λ representa esta distancia ponderada por λ y 1 – λ. Se ha observado que las soluciones que minimizan F_λ son también eficientes. Se va a denotar por Set_ND el conjunto de soluciones no dominadas encontradas durante la búsqueda. El conjunto Set_ND final será la aproximación a la curva de eficiencia obtenida. La estrategia de solución tiene las tres fases que se describen en seudocódigo 1.

FASE I

1. Hacer $Set\_ND = \emptyset$.
2. Ejecutar **Generador_Preferencias**(S) para obtener una solución S teniendo en cuenta las preferencias de los clientes (f_2).
3. Ejecutar **Tabu_Search**(f_1, S), tomando f_1 como función objetivo ($f_2 \rightarrow f_1$)
4. Ejecutar **Generador_Costes**(S) para obtener una solución S minimizando la distancia (f_1).
5. Ejecutar **Tabu_Search**(f_2, S) ($f_2 \rightarrow f_1$)
6. Actualizar Set_ND con todas las soluciones no-dominadas encontradas

FASE II

Repetir

7. Generar $\lambda \in (0,1)$ aleatoriamente.
8. Ejecutar **Tabu_Search**(F_λ, S) usando F_λ como función objetivo.
9. Actualizar Set_ND

hasta que trascurran max_iter_faseii iteraciones sin cambios en Set_ND

FASE III

Repetir

10. Identificar las soluciones S de Set_ND cuyos vecindarios no han sido explorados (todas en la primera iteración)
11. Para cada una de estas soluciones S explorar sus vecindarios y actualizar Set_ND.
12. Identificar las nuevas soluciones de Set_ND y mejorarlas

hasta Set_ND se estabilice (no cambie)

Seudocódigo 1. Descripción de la adaptación de MOAMP a nuestro problema

Con la primera fase se quieren obtener las mejores soluciones con respecto a cada una de las funciones objetivos originales. En otras palabras, se obtienen los extremos de la curva de eficiencia y por tanto los valores de f_1^{min}, f_1^{max}, f_2^{min} y f_2^{max} que son usados en el cálculo de las funciones mixtas F_λ en la fase siguiente. No obstante, como se ha comentado, en la fase I en realidad se obtienen aproximaciones a estos valores que no siempre tienen por qué coincidir con el real. Incluso a veces estas aproximaciones son mejoradas en fases posteriores. Por tanto, en la descripción del método general, f_1^{min} y f_2^{max} se re-definen respectivamente como los valores de la mejor solución con respecto al coste de Set_ND, y por su parte f_2^{min} y f_1^{max} se re-definen respectivamente como los valores de la mejor solución en Set_ND con respecto a las preferencias de los clientes. En la segunda fase, se obtiene una muestra diversa de soluciones 'de compromiso'; es decir, no las mejores soluciones con respecto a algún objetivo concreto, pero sí buenas soluciones en conjunto. Esta fase finaliza cuando transcurre un cierto número de iteraciones (max_iter_faseii) sin cambios en Set_ND. Hay que indicar que, dentro de cada procedimiento de búsqueda tabú, se comprueba si cada solución visitada se incorpora a Set_ND. En otras palabras, el conjunto de soluciones no dominadas se actualiza con todas las soluciones visitadas durante cada ejecución de la búsqueda tabú en estas dos primeras fases. Finalmente, en la tercera fase se completa el conjunto de soluciones no dominadas explorando el vecindario de las encontradas hasta ese momento.

La figura 2 muestra la jerarquía de procedimientos de la estrategia de solución (MOAMP):

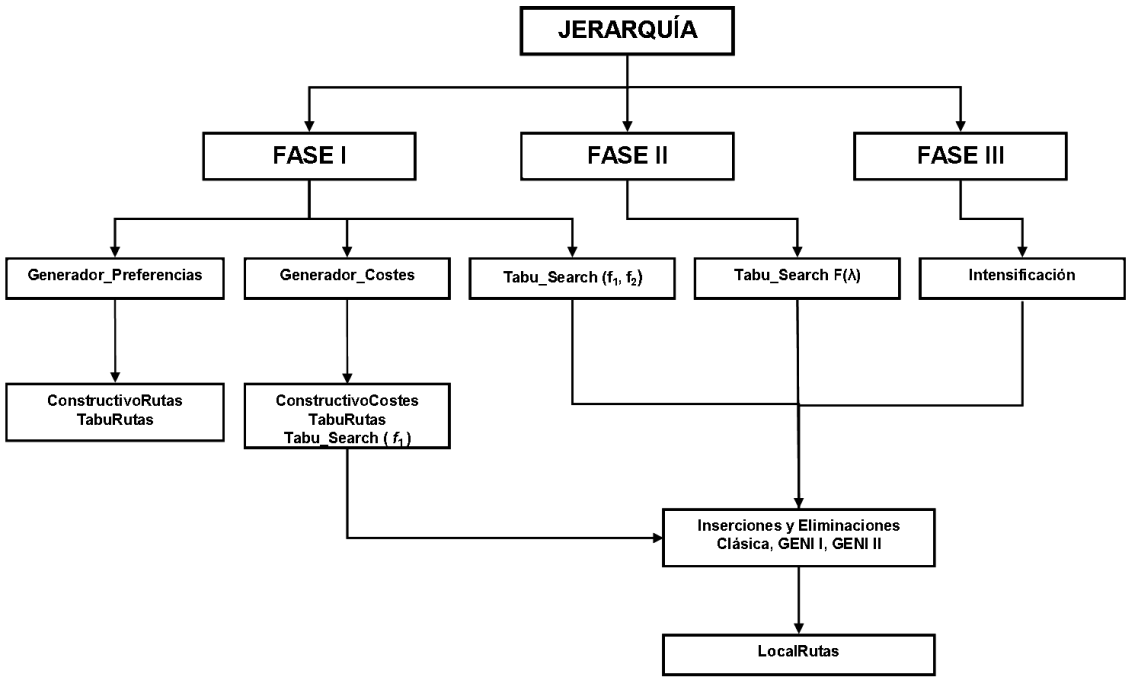

Figura 2. Jerarquía de procedimientos

A continuación se describen, con más detalle, los procedimientos **Generador_Preferencias**, **Generador_Costes** y **Tabu_Search**. Así mimo se describirá cómo se realiza la exploración del vecindario de cada solución S en el paso 11 de la fase III y las mejoras de las nuevas soluciones que entran en Set_ND en el paso 12.

6. Procedimientos principales: constructivos y de búsqueda tabú

Para facilitar la descripción de estos procedimientos y de los siguientes vamos a formalizar la representación de las soluciones. Cada solución S viene definida por la asignación de clientes a calendarios y el diseño de las correspondientes rutas diarias. Por tanto, cada solución S viene representada

por el par $(\boldsymbol{C}, \boldsymbol{R})$. $\boldsymbol{C}$ representa el vector de asignaciones de calendarios a clientes y es de la forma:

$$\boldsymbol{C} = (C(1), C(2), \ldots, C(i), \ldots\ C(n))$$

donde $C(i)$ indica el segundo subíndice del calendario que tiene asignado el cliente i, es decir $\lambda_{i\,C(i)} \in \Lambda_i$. (A partir de ahora, con el fin de simplificar las notaciones y expresiones en ocasiones identificaremos cada calendario con su segundo subíndice: Por ejemplo, decir que el cliente *i* tiene asignado el calendario *j* equivale a decir que el cliente *i* tiene asignado el calendario λ_{ij}).

Por su parte, $\boldsymbol{R}$ representa al conjunto de todas las rutas diarias, es decir, $\boldsymbol{R} = \{R_t : t = 1, \ldots, T\}$ y R_t es el conjunto de rutas del día t. Cada ruta viene representada por una secuencia de puntos que comienzan y finalizan en el depósito 0, y los puntos intermedios son los clientes que se visitan y su orden. Para referirnos a los elementos de una solución genérica S sencillamente nos referiremos a ellos como $\boldsymbol{C}$, $\boldsymbol{R}$ o R_t. En ocasiones, especialmente si estamos refiriéndonos a más de una solución, por ejemplo S y S', para evitar confusiones a sus correspondientes elementos los denotaremos por $S.\boldsymbol{R}$, $S.\boldsymbol{C}$ o $S.R_t$ (o bien $S'.\boldsymbol{R}$, $S'.\boldsymbol{C}$ o $S'.R_t$).

Como se ha comentado antes, S_0 representa la solución actual. Uno de los objetivos, como ya se ha dicho es respetar al máximo estos calendarios actuales, o separarnos lo mínimo posible de ellos. La medida de separación viene dada por f_2 y obviamente el óptimo de este objetivo lo da cualquier solución S, verificando que $S.\boldsymbol{C} = S_0.\boldsymbol{C}$ y por tanto $f_2(S) = 0$. Luego, el procedimiento **Generador_Preferencias** (descrito en el seudocódigo 2) obtiene la solución S definiendo $S.\boldsymbol{C} = S_0.\boldsymbol{C}$ y tratando de optimizar el coste de las correspondientes rutas diarias

---

Procedimiento **Generador_Preferencias** (var *S*)

1. Hacer $S.\boldsymbol{C} = S_0.\boldsymbol{C}$
2. $\forall\ t = 1, \ldots, T$
   a. Definir U como el conjunto de puntos a visitar el día t (definido por $S.C$)
   b. $S.R_t$ = ConstructivoRutas(U)
   c. $S.R_t$ = TabuRutas($S.R_t$)

Seudocódigo 2. Descripción del procedimiento **Generador_Preferencias**

Los procedimientos **ConstructivoRutas** y **TabuRutas** son explicados en una sección posterior dedicada a los procedimientos de rutas.

El procedimiento **Generador_Costes** busca la solución que minimice la distancia total. Este problema es equivalente al PVRP clásico con algunos matices, ya que en este trabajo consideramos las restricciones (3) y (4) que no se consideran en el modelo original. En nuestro caso, el procedimiento **Generador_Costes** sigue una estrategia iterativa de tipo multi-arranque. En cada iteración se construye una solución que es mejorada posteriormente por el procedimiento **Tabu_Search**. Cada vez que se construye una solución se tienen en cuenta las asignaciones de clientes a calendarios de las soluciones visitadas anteriormente, de forma que se penalizan las asignaciones más frecuentes y se favorecen las menos. El objetivo es favorecer la diversidad del proceso. La matriz auxiliar $freq\_cal(i,j)$ registra el número de veces que el pedido i ha tenido asignado el calendario j (es decir λ_{ij}) en las soluciones visitadas en las ejecuciones del procedimiento **Tabu_Search**.

Procedimiento **Generador_Costes**(var S)

1. Hacer $f\_best = \infty$ y $freq\_cal(i,j) = 0$

Repetir

2. S = ConstructivoCostes($freq\_cal$, S)
3. $\forall\, t = 1, \ldots, T : S.R_t$ = TabuRutas($S.R_t$)
4. S = Tabu_Search(f_1, S)
5. Actualizar $freq\_cal$ con las soluciones visitadas en el paso 4
6. Si $f_1(S) < f\_best$ entonces hacer: $S\_best = S$ y $f\_best = f_1(S)$

hasta que trascurran max_iter_costes iteraciones sin cambios en f_best

7. Hacer $S = S\_best$

Seudocódigo 3. Descripción del procedimiento **Generador_Costes**

Como se observa en el seudocódigo 3, en cada iteración se construye una solución S mediante el procedimiento **ConstructivoCostes** que tiene en cuenta las frecuencias de asignaciones de calendarios a clientes anteriores (registradas en $freq\_cal$). Después, sin cambiar los calendarios, se mejora el conjunto de rutas de cada uno de los días con el procedimiento **TabuRutas**. Finalmente, en el paso 3 la solución S con **Tabu_Search**. El procedimiento **ConstructivoCostes** se describe en el seudocódigo 4.

Procedimiento **ConstructivoCostes**($freq\_cal$; var S)

1. $\forall t = 1, \ldots, T$ iniciar R_t, es decir, hacer 0 – 0 las m posibles rutas
2. Ordenar los clientes por el ángulo que forma la horizontal y la bisectriz que une el origen con su localización

Desde $l = 1$ hasta n

3. Tomar el l-ésimo cliente según ese orden y denotarlo por i
4. $\forall\, j = 1, \ldots, nc(i)$: Examinar si la asignación del calendario j al cliente i es factible. En su caso determinar la mejor inserción (ruta, posición) de i en $R_t\ \forall\, t \in \lambda_{ij}$ y calcular $Incdist(i,j)$ el incremento en la distancia total de estas inserciones
5. Determinar $LF = \{\, j : j = 1, \ldots, nc(i)$, la asignación del calendario j al cliente i es factible $\}$
6. Determinar:
$$fmax = max\{\, freq\_cal(i,j) : j \in LF \,\}$$
$$fmin = min\{\, freq\_cal(i,j) : j \in LF \,\}$$
$$Imax = max\{\, Incdist\,(i,j) : j \in LF \,\}$$
$$Imin = min\{\, Incdist\,(i,j) : j \in LF \,\}$$
7. $\forall j = 1, \ldots, nc(i)$ calcular
$$fg(j) = \beta \cdot \left(\frac{Incdist(i,j) - Imin}{Imax - Imin} \right) + (1-\beta) \cdot \left(\frac{freq\_cal(i,j) - fmin}{fmax - fmin} \right)$$
8. Determinar
$$fgmax = max\{\, fg(j) : j \in LF \,\}$$
$$fgmin = min\{\, fg(j) : j \in LF \,\}$$
9. Determinar $LF = \{\, j : j \in LF,\ fg(j) \leq \alpha \cdot fgmin + (1-\alpha) \cdot fgmax \,\}$
10. Elegir $j^* \in LF$ aleatoriamente.
11. Asignar el calendario λ_{ij^*} al cliente i. Ejecutar las correspondientes inserciones de i en $R_t\ \forall t \in \lambda_{ij^*}$

Seudocódigo 4. Descripción del procedimiento **Generador_Costes**

El procedimiento **ConstructivoCostes** ordena inicialmente los clientes. A continuación, en cada iteración toma un cliente según ese orden y va asignándole un calendario y actualizando las rutas. Los calendarios de cada cliente son valorados teniendo en cuenta el coste y las frecuencias de asignaciones anteriores. El peso de cada una de estas componentes viene regulado por el parámetro β. A continuación, se forma una lista (LC) con los mejores calendarios y de esta lista se elige uno al azar (j^*) que es el que se asigna al cliente. El tamaño de la lista viene regulado por el parámetro α. Obsérvese que el uso de la lista LC ("Lista de Candidatos") es una idea tomada del estrategia metaheurística GRASP (Feo and Resende 1989, 1995). Como se explica en el paso 4 para cada día de cada calendario se determina la mejor inserción. El conjunto de inserciones que se consideran se describe más adelante.

El procedimiento **Tabu_Search** es una búsqueda vecinal, que se basa en la estrategia metaheurística búsqueda tabú (Glover 1989, 1990; Glover and Laguna 1997). En cada iteración, se mueve de la solución actual a la mejor solución vecina. A diferencia de los procedimientos de búsqueda local, el proceso no finaliza en el primer óptimo local sino que permite empeoramientos. Para que el proceso no cicle (volver a soluciones anteriores) determinados movimientos son declarados *tabú*. A su vez, este criterio tabú puede ser ignorado (y por tanto el movimiento aceptado) en determinadas condiciones ("Criterio de Aspiración"). El proceso finaliza cuando se alcanza un criterio de parada. El procedimiento **Tabu_Search** se describe con detalle en el seudocódigo 5.

Procedimiento **Tabu_Search**(f_r o F_λ ; var S)

1. Hacer $S\_best = S$
2. Iniciar estructuras de memoria ($tabu\_cal$)

Repetir

3. Examinar el sub-conjunto de soluciones vecinas de S, $N^*(S) \subset N(S)$ que no son tabú o que cumplen el criterio de aspiración y determinar la mejor, S'
4. Hacer $S = S'$
5. Para todos los días *t* afectados por el cambio hacer $S.R_t$= **LocalRutas**($S.R_t$)
6. Actualizar $tabu\_cal$
7. Actualizar en su caso S_best y Set_ND

hasta que trascurran max_iter_tabu sin cambiar S_best

8. Hacer $S = S\_best$
9. $\forall\, t = 1, \ldots, T$: $S.R_t$ = **TabuRutas**($S.R_t$)
10. Si $f_1(S) < f_1(S\_best)$ entonces actualizar Set_ND e ir al paso 1

Seudocódigo 5. Descripción del procedimiento **Tabu_Search**

A continuación, se describen y definen con más detalle algunos de los puntos del procedimiento anterior. Como el procedimiento permite cambios a soluciones peores, la solución actual S no tiene por qué coincidir con la mejor solución encontrada durante la búsqueda. A esta la denotamos por S_best. El conjunto de 'soluciones vecinas' de S, $N(S)$, es el conjunto de soluciones factibles a las que se llega desde S por movimientos vecinales (de esta forma se identifica los movimientos vecinales con las soluciones a las que dan lugar). En este caso cada movimiento viene definido por el cambio de calendario asignado a un cliente. Cuando un cliente cambia del calendario actual $C(i)$ a un nuevo calendario j, $j \neq C(i)$ se realizan 2 tipos de operaciones: eliminaciones de i de $R_t\ \forall\, t \in \lambda_{iC(i)} \setminus \lambda_{ij}$ e inserciones de i en $R_t\ \forall\, t \in \lambda_{iC(i)} \setminus \lambda_{ij}$. El conjunto de eliminaciones e inserciones que se examinan se explicará con más detalle posteriormente. En cada caso, siempre se considera la mejor inserción o eliminación (la más "barata").

Para evitar que el procedimiento cicle y se vuelva a soluciones ya visitadas, algunos movimientos se declaran tabú y, en principio, no son admitidos. Concretamente, el procedimiento declara tabú la reasignación a clientes de calendarios que recientemente han dejado de tener asignados. El número de iteraciones que una reasignación permanece tabú es el parámetro $Tenure$ definido previamente. Para

chequear cuando un movimiento es tabú hace uso de una estructura de memoria, la matriz $tabu\_cal$. El valor de $tabu\_cal(i,j)$ es el número de la iteración en la que el cliente i dejó de tener asignado el calendario j. Comparando los valores de esta matriz y el contador de iteraciones se determina cuando un movimiento es tabú.

No obstante, el status tabú de un movimiento puede ser obviado en determinadas circunstancias: a) el movimiento da lugar a una solución mejor S_best; b) en alguno de los días afectados se reduce el mínimo valor de la distancia recorrida en ese día en las soluciones visitadas hasta ese momento. Si se da alguna de estas condiciones se dice que se alcanza el criterio de aspiración. En este caso, se ignora el status tabú. Por tanto en la exploración de $N(S)$ del paso 3 se consideran sólo el sub-conjunto $N^*(S)$ de los movimientos que o no son tabú o que cumplen el criterio de aspiración.

Cuando se ejecuta un movimiento, las rutas de los días afectados (por inserción o eliminación) son mejoradas por un procedimiento de búsqueda local (paso 5) denominado **LocalRutas**, que se explicará con más detalle en la sección posterior dedicada a los procedimientos de rutas. Con la ejecución de este procedimiento de búsqueda local, se pretende que las rutas diarias de las soluciones visitadas sean óptimos locales.

El paso 9 intenta mejorar aún más las rutas diarias de la solución que devuelve el método. Para ello aplica el procedimiento **TabuRutas** a cada uno de los días. Si hay mejora en al menos uno de los días (paso 10), la solución se considera interesante y el proceso se reinicia en dicha solución. Se re-inicia también la matriz *tabú_cal* y por tanto en ese momento no hay movimientos tabú, con lo que se favorece la intensificación de la búsqueda en esa región.

Para terminar con la descripción de los procedimientos generales hemos de indicar que los vecindarios de cada solución $S \in Set\_ND$ que se exploran en el paso 11 de la Fase III, son precisamente los mimos vecindarios $N(S)$ usados en el procedimiento **Tabu_Search**. Cuando una nueva solución S entra en Set_ND se mejoran las rutas aplicando **LocalRutas** a todos los conjuntos $S.R_t$ (paso 12 de la Fase III).

7. Procedimientos de rutas

En esta sección se describen los procedimientos de construcción y mejora de las rutas diarias usadas en los procedimientos generales. El objetivo, en este caso, siempre es minimizar el coste. Concretamente, se desarrollan **ConstructivoRutas**, **LocalRutas** y **TabuRutas**.

ConstructivoRutas toma como parámetro de entrada el subconjunto de puntos a visitar ese día y como salida se obtiene el conjunto de las rutas, para ello utiliza inserciones.

LocalRutas es una heurística de búsqueda local que intenta mejorar el conjunto de rutas de un determinado día. Los datos de entrada y salida son la propia variable $Rutas$. Se define el conjunto $NR(Rutas)$ como el vecindario de Rutas, es decir, todos los conjuntos de rutas que se pueden obtener realizando en *Rutas* cambios o movimientos. Estos cambios consisten en diferentes intercambios de cadenas que se ilustran en la figura 3. Obsérvese que los intercambio Or (Or 1976) supone intercambiar 2 cadenas consecutivas dentro de una misma ruta, el intercambio Or-Generalizado supone intercambiar dos cadenas no consecutivas dentro de una misma ruta, el CROSS intercambio supone intercambiar dos cadenas de dos rutas diferentes (Taillard *et al* 1997) y finalmente Inserción supone cambiar una cadena de una ruta a otra.

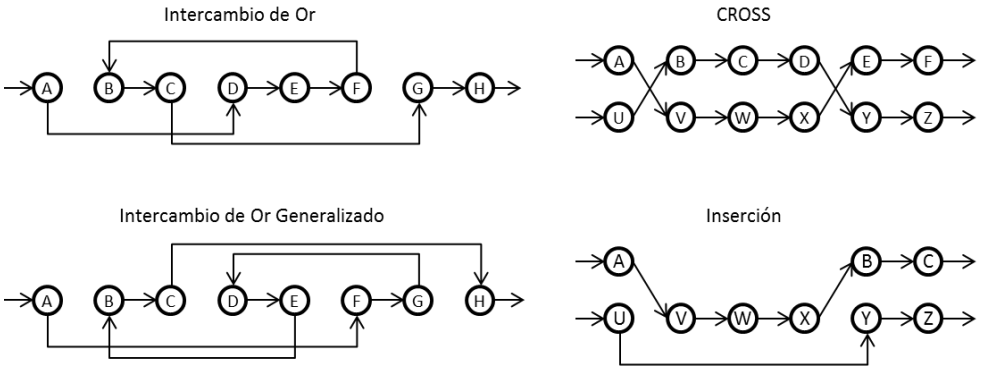

Figura 3. Diferentes tipos de movimientos e intercambios para las rutas diarias.

LocalRutas mejora cada conjunto de rutas inicial hasta que no hay mejoras posibles. Por su parte **TabuRutas** permite mejorar los resultados obtenidos por **LocalRutas** ya que realiza exploraciones más allá del primer óptimo local encontrado. Para ello se permiten movimientos a conjuntos de rutas peores. Los movimientos consisten en la eliminación e incorporación de una serie de arcos y para evitar que el algoritmo cicle, algunos movimientos se declaran tabú aunque pueden ser admitidos si dan lugar a un conjunto de rutas mejor.

8. NSGA II

Con el fin de poder examinar la 'bondad' de los resultados obtenidos por el algoritmo MOAMP descrito en las secciones anteriores, estos se van a comparar con los resultados obtenidos por una adaptación del algoritmo NSGA II. Este método, propuesto por Deb *et al* (2002), está encuadrado dentro de los algoritmos evolutivos, usa elitismo y un operador de comparación ("crowding") en función de la proximidad entre sí de las diferentes soluciones.

Se dan ciertas definiciones para describir la adaptación del algoritmo NSGA II. Sea P un conjunto de soluciones (población); P puede ser dividido en subconjuntos o frentes no dominados como sigue:

F_1 : Conjunto de soluciones no dominadas de P
F_2 : Conjunto de soluciones no dominadas de $P - F_1$
F_3 : Conjunto de soluciones no dominadas de $P - \{ F_1 \cup F_2 \}$,

y así sucesivamente. Para cada solución S, definimos:

$ind(S)$: índice de la capa de no-dominancia a la que pertenece S.
$cwd\_dst(S)$, *crowding distance*, que se define como:

$$cwd\_dst(S) = \frac{f_1(S_a) - f_1(S_b)}{f_1^{\max} - f_1^{\min}} + \frac{f_2(S_c) - f_2(S_d)}{f_2^{\max} - f_2^{\min}}$$

donde S_a y S_b son las soluciones justo anterior y posterior a S al ordenar su capa de no-dominancia ($F_{ind(S)}$ según los valores de f_1 en orden descendente; de igual forma S_c y S_d son las soluciones justo anterior y posterior a S al ordenar $F_{ind(S)}$ según los valores de f_2 en orden descendente. En caso de que S tome valores extremos, dentro de su capa de no-dominancia, o bien para f_1 o bien para f_2 entonces $cwd\_dst(S) = \infty$.

Con estos valores, se define un orden parcial $\prec$, (operador *Crowded Comparison*) de la forma siguiente. Sean dos soluciones S y S'

$$S \prec S' \text{ si } [ind(S) < ind(S')] \text{ o } [(ind(S) = ind(S')) \text{ y } (cwd\_dst(S) > cwd\_dst(S'))].$$

Este operador se va a utilizar en la selección de padres y en determinar qué soluciones pasan a la siguiente generación. Con esto, se trata de favorecer, en primer lugar, la selección de soluciones menos dominadas

y, en segundo lugar, soluciones más 'aisladas' dentro de su capa de dominancia para tratar de rellenar esa región.

Para describir la adaptación del algoritmo NSGA-II a este problema se ha de señalar lo siguiente: Como se ha comentado anteriormente, cada solución S tiene dos componentes $\boldsymbol{C}$, el vector que indica que calendario tiene asignado cada cliente y $\boldsymbol{R}$, el conjunto de rutas para cada día. Ahora bien, los operadores de cruce y mutación de NSGA-II solo van a actuar sobre $\boldsymbol{C}$. Por tanto, con el fin de explicar más sencillamente estos operadores identificaremos cada solución S con su componente $\boldsymbol{C}$.

El operador de cruce es el clásico *one-point crossover*. Se muestra a continuación un ejemplo del operador de cruce para un problema con 10 clientes. Cada solución viene representada como un vector de 10 posiciones en el que cada posición indica que calendario tiene asignado cada cliente. Una vez seleccionas las soluciones padre S y S' se selecciona aleatoriamente un "punto de corte" y se construyen las soluciones hijas S^* y S^{**} según se ilustra la figura 4.

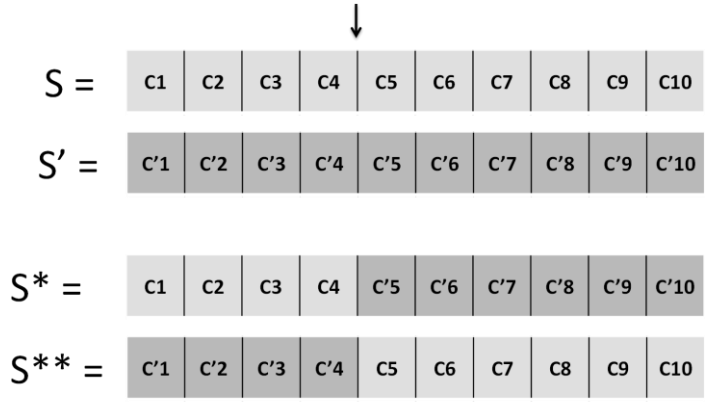

Figura 4. Ejemplo de operador de cruce. Una vez seleccionadas las soluciones $\boldsymbol{S}$ y $\boldsymbol{S'}$ se generan las soluciones hijas $\boldsymbol{S^*}$ y $\boldsymbol{S^{**}}$.

Una vez generadas las nuevas soluciones hijas $\boldsymbol{S^*}$ **y** $\boldsymbol{S^{**}}$ cada componente de ellas cambia con una pequeña probabilidad. En este caso la mutación de la componente i consiste en asignarle un valor aleatoriamente entre 1 y $nc(i)$, (en otras palabras, asignar al correspondiente cliente i un calendario de Λ_i). De igual forma la generación de soluciones aleatorias iniciales consiste en asignar a cada posición i del vector $\boldsymbol{C}$ un valor aleatoriamente entre 1 y $nc(i)$.

Cuando se obtiene un nuevo vector $\boldsymbol{C}$ (ya sea por la generación inicial o tras las operaciones de cruce y mutación), las correspondientes rutas $\boldsymbol{R}$, se obtienen ejecutando los algoritmos **ConstructivoRutas** y **LocalRutas** a cada día y para cada conjunto de puntos a visitar. Obsérvese que estos dos procedimientos son determinísticos. Por tanto, en este algoritmo, todas las soluciones S vienen unívocamente determinadas por el vector $\boldsymbol{C}$.

Sea n_pob (N en la descripción original de NSGA-II) el tamaño de la población inicial, p_mut, la probabilidad de mutación o cambio de cada componente de cada solución S, la adaptación del algoritmo NSGA-II se describe en el seudocódigo 6.

---

1. Generar la población de soluciones iniciales P_0 de tamaño n_pob.
2. Determinar $f_1(S)$ y $f_2(S)$ para las soluciones $S \in P_0$.
3. Dividir P_0 en capas de no dominancia;
4. hacer $t := 0$

Repetir:

5. Calcular la *crowding distance* para los elementos de P_t.
6. Generar poblacion Q_t de soluciones hijas de tamaño n_pob a partir de P_t.

7. Aplicar el operador mutación a los elementos de Q_t.
8. Determinar $f_1(S)$ y $f_2(S)$ para las soluciones $S \in Q_t$.
9. Hacer $H_t = P_t \cup Q_t$ y dividir H_t en capas de no dominancia;
10. Formar P_{t+1} con los n_pob primeros elementos de H_t según el orden parcial $\prec$
11. Hacer $t := t + 1$;

hasta alcanzar criterio de parada

Seudocódigo 6. Descripción de la adaptación de NSGA II

Finalmente, como criterio de parada de NSGA II se puede usar un máximo número de iteraciones total o un número de iteraciones sin cambios en P_t, un tiempo máximo de computación, etc.

9. Experimentos computacionales

Como se ha comentado anteriormente para comprobar la calidad del algoritmo MOAMP propuesto se realizan una serie de pruebas para comparar sus resultados con los obtenidos por el algoritmo NSGA-II. Estas pruebas consisten en ejecuta ambos algoritmos en un conjunto de instancias seudo-reales, así como con la instancia real. Las instancias seudoreales se generan a partir de los datos reales proporcionados por la empresa y eligiendo aleatoriamente y con reemplazamiento clientes con sus correspondientes datos (localización, conjunto de calendarios, tipo de servicio). Se generan instancias de 20, 40, 60, 80 y 100 clientes, concretamente 5 instancias de cada tamaño (número de clientes). Para abreviar denominamos *Ins_20_1* a la primera instancia de tamaño 20, *Ins_20_2* a la segunda instancia de tamaño 20, *Ins_40_1* a la primera instancia de tamaño 40 etc.

Los parámetros usados han sido los siguientes:

$max\_iter\_faseii = 10$
$Tenure = n$
$max\_iter\_tabu = 50 \cdot n$ en los pasos 3 y 5 (Fase I);
$max\_iter\_tabu = 10 \cdot n$ en el paso 8 (Fase II)
$tenure\_rutas = 5 \cdot n$
$max\_iter\_rutas = 20 \cdot n$
$max\_iter\_costes = 10$

Inicialmente se ejecuta el procedimiento MOAMP y se registra el número de soluciones no dominadas ($|Set\_ND|$) y el tiempo de computación empleado (en segundos). Entonces se ejecuta NSGA- II tomando $n\_pob = 2 \cdot |Set\_ND|$ y $p\_mut = 0.02$. El criterio de parada de NSGA-II se alcanza cuando tras finalizar una iteración se supera el tiempo de computación empleado por MOAMP. El ajuste de estos parámetros se ha obtenido mediante diferentes experimentos basándonos además en los resultados obtenidos en Gómez *et al* (2015) y Pacheco *et al* (2013) en otros problemas bi-objetivos de rutas.

En la tabla 2 se muestra el número de soluciones obtenido por cada fase de MOAMP. Concretamente se indica el número medio por tipo de instancia.

Tabla 2. Número medio de soluciones obtenidas en cada fase por tamaño de instancia

	TAMAÑO DE INSTANCIAS				
FASES MOAMP	20	40	60	80	100
FASE I	6	9,8	14,4	17	20,2
FASE II	6,8	15,2	19,8	27,8	28,2
FASE III	9,8	20,4	28,8	41,4	52,2

Para ilustrar estos resultados, se muestran gráficamente (figuras 5 a 9) las aproximaciones a la curva de eficiencia obtenidas en cada fase para la primera instancia de cada tamaño.

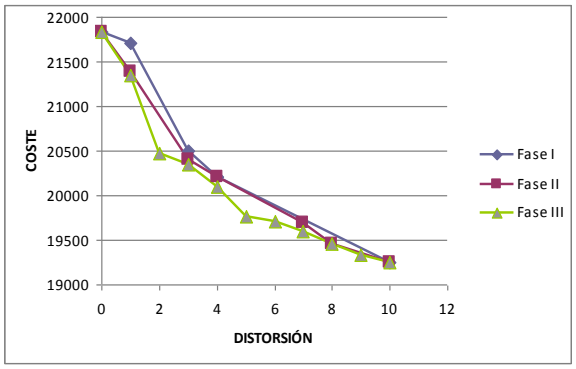

Figura 6. Resultados de MOAMP en *Ins_20_1*

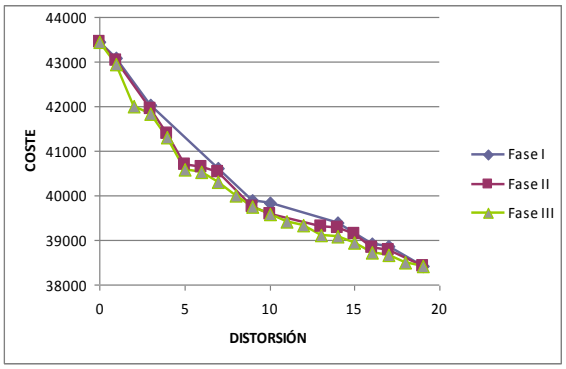

Figura 5. Resultados de MOAMP en *Ins_40_1*

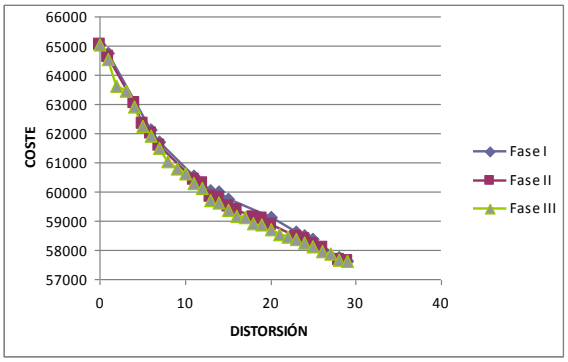

Figura 7. Resultados de MOAMP en *Ins _60_1*

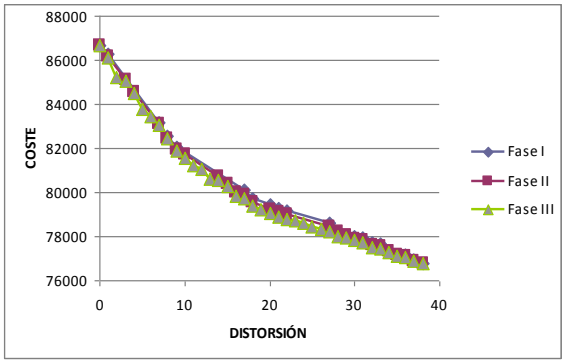

Figura 8. Resultados de MOAMP en *Ins_80_1*

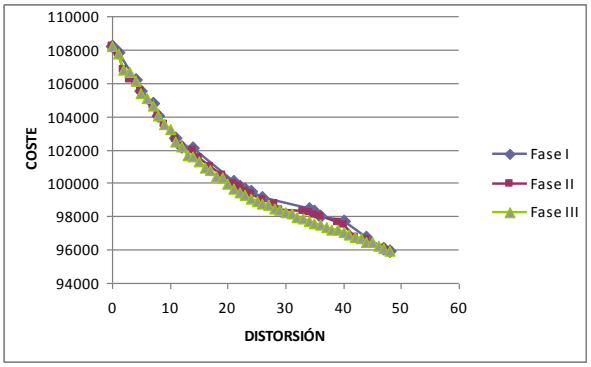

Figura 9. Resultados de MOAMP en *Ins_100_1*

Se puede apreciar como en cada fase se mejora el conjunto de soluciones obtenidas en la fase anterior. Además, a medida que aumenta el tamaño de las instancias se obtienen conjuntos más densos y con un mayor nº de soluciones.

Las figuras 10-14 muestran la comparación de los resultados obtenidos por ambos métodos, MOAMP y NSGA-II (concretamente se muestra una figura con la primera instancia de cada tamaño).

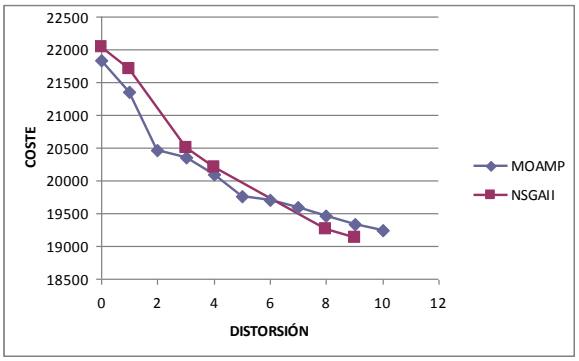

Figura 10. Curvas obtenidas para *Ins_20_1*

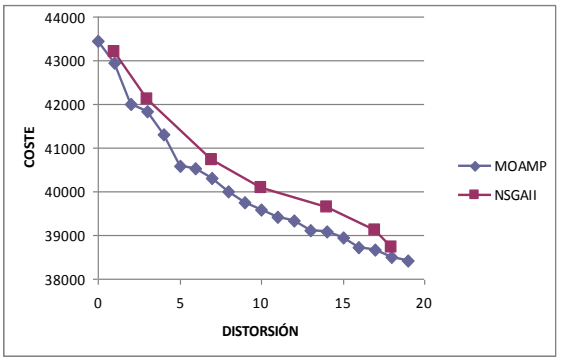

Figura 11. Curvas obtenidas para *Ins_40_1*

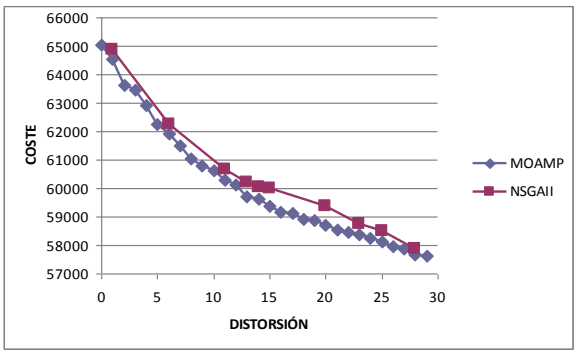

Figura 12. Curvas obtenidas para *Ins_60_1*

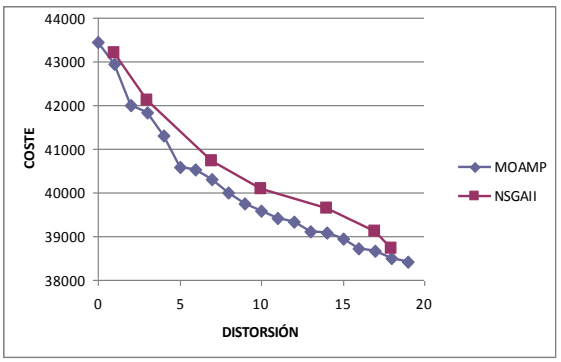

Figura 13. Curvas obtenidas para *Ins_80_1*

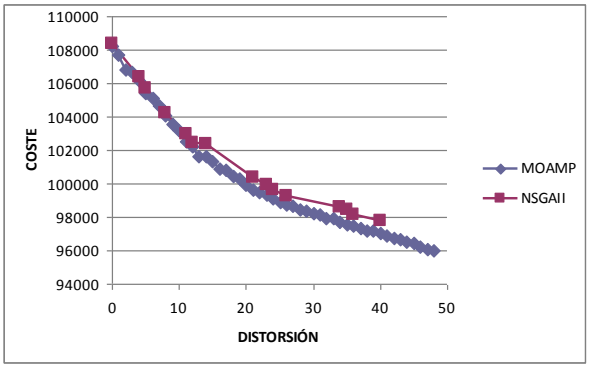

Figura 14. Curvas obtenidas para *Ins_100_1*

En las figuras 10-14, se puede apreciar como a medida que aumenta el tamaño de las instancias aumenta la diferencia entre el número de soluciones MOAMP y NSGA-II. Si nos fijamos, en las instancias de menor tamaño (20) hay algunas soluciones MOAMP que son dominadas por soluciones NSGA-II; ahora bien, para el resto de instancias, todas las soluciones NSGA-II son dominadas por MOAMP. Además, observamos como con MOAMP se consiguen curvas de eficiencia con un mayor número de soluciones, menos dominadas, más densas y con un comportamiento mejor en los extremos.

En la tabla 3 y en la figura 15, se recogen los resultados de la media del número de soluciones obtenidas por MOAMP y por NSGA-II. También aparece el número medio de soluciones MOAMP no dominadas por soluciones NSGA-II así como el número medio de soluciones NSGA-II no dominadas por soluciones MOAMP. Si nos fijamos, vemos que a partir de las instancias de tamaño 60 no hay ninguna solución NSGA-II que no esté dominada por alguna MOAMP.

Tabla 3. Resumen (media) de soluciones dominadas por cada estrategia

	TAMAÑO DE INSTANCIAS				
MEDIA DE	20	40	60	80	100
Soluciones MOAMP	9,8	20,4	28,8	41,4	52,2
Soluciones MOAMP no dominadas por NSGA II	8,8	20,4	28,8	41,4	52,2
Soluciones NSGA II	6,2	11,4	15,8	21,6	24,8
Soluciones NSGA II no dominadas por MOAMP	1,6	0,2	0	0	0

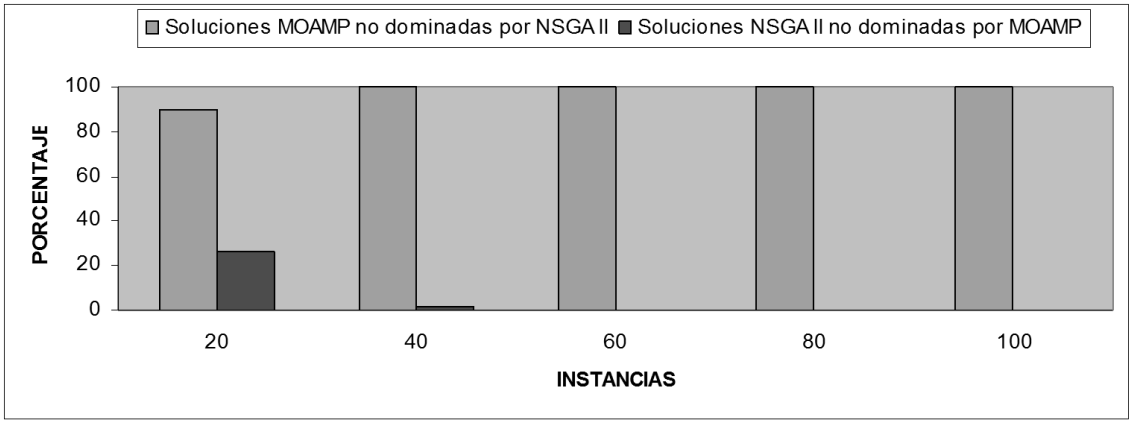

Figura 15. Dominancia entre soluciones MOAMP y NSGA-II

Por tanto, por las figuras 10-15 y por la tabla 3 es clara la mayor calidad del conjunto de soluciones obtenidas por MOAMP: mayor número de soluciones, mayor densidad, mayor calidad en las soluciones (dominancia de las soluciones MOAMP sobre las soluciones NSGA-II). Entendemos que estos resultados hacen innecesario el cálculo de otras medidas (dominancia, volumen, cubrimiento, etc.) para comparar ambas estrategias.

Todas las pruebas se realizaron en un ordenador personal con procesador Corel i7 4790 a 3.6 GHz. Ambos algoritmos fueron implementados en lenguaje Object Pascal mediante el entono de trabajo Rad Studio XE7 (que incluye el compilador Delphi). La tabla 4 muestra los tiempos de computación (media por tamaño) usados por ambos algoritmos.

Tabla 4. Tiempos de computación por tamaño de instancia

Tamaño	**min**	**media**	**max**
20	3,08	4,244	5,95
40	25,08	32,922	46,58
60	89,82	162,696	202,01
80	400,15	677,004	902,65
100	1199,28	2109,688	3000,82

10. Análisis de los resultados obtenidos con datos reales

Finalmente se han ejecutado ambos algoritmos en la instancia real, es decir, con los datos reales descritos en la sección 2. En esta sección se describen los resultados obtenidos. En la figura 16 las funciones objetivos de las soluciones obtenidas con MOAMP, NSGA II y la solución actual, (es decir la empleada por la empresa).

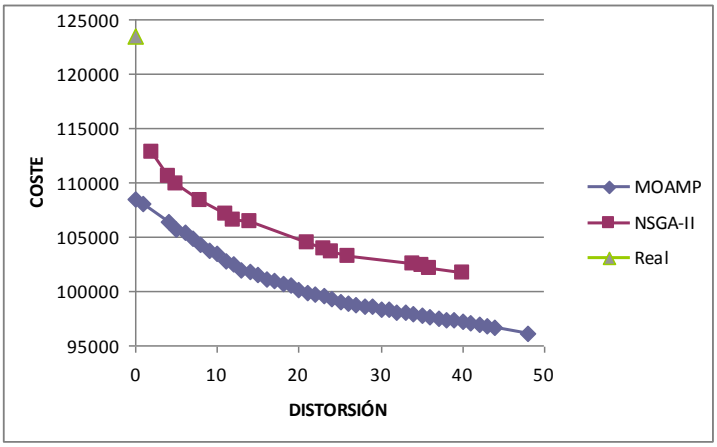

Figura 12. Soluciones MOAMP Y NSGA II y solución actual

Como se observa, la solución actual tiene obviamente distorsión 0 ya que son los calendarios que actualmente tienen asignados los clientes y por los cuales tienen preferencia. Sin embargo, la estrategia MOAMP obtiene una solución con menor coste que respeta dichos calendarios. Esto es debido a que la empresa no tiene una buena planificación de las rutas diarias. También, podemos apreciar como al permitir modificar la asignación de calendarios a los clientes, tanto MOAMP como NSGA II obtienen soluciones que reducen considerablemente el coste. Como ocurría en las instancia pseudoreales, se aprecia que todas las soluciones MOAMP dominan a las soluciones NSGA II y que la 'curva de eficiencia' obtenida por MOAMP es más densa y está más poblada que la obtenida por NSGA II.

Finalmente creemos conveniente hacer un análisis más detallado de la curva de soluciones obtenidas por MOAMP.

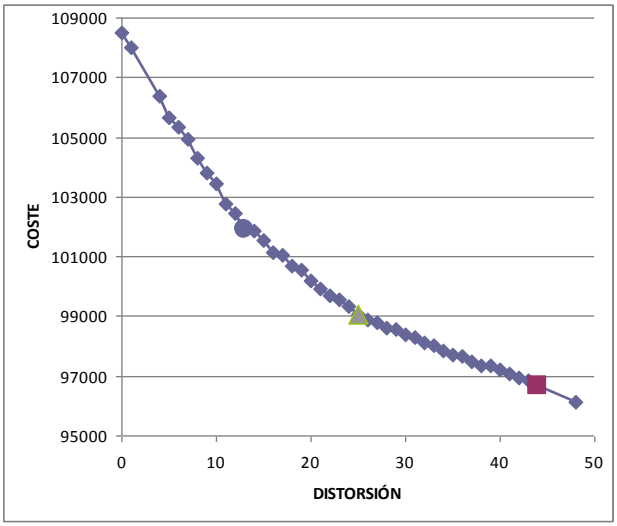

Figura 17. Análisis de la curva de soluciones

Como se muestra en la figura 17, la solución de distorsión 0 de MOAMP ("esquina superior izquierda") produce un ahorro del 12% sobre la solución actual de la empresa. En el otro extremo, la solución de mínimo coste ("esquina inferior derecha") tiene una distorsión de 48 y produce un ahorro de un 11,4% sobre la solución MOAMP de distorsión 0. Entre estas dos soluciones nos encontramos con un amplio abanico de soluciones intermedias. Un análisis que puede resultar muy interesante es estudiarla desde distintas situaciones económicas.

Así, en época de crisis o de recesión, la empresa puede estar más interesada en minimizar el coste aunque esto le suponga aumentar la distorsión. La solución de mínimo coste tiene distorsión 48. Ahora bien, partiendo de esta solución nos encontramos con que la siguiente solución con menos coste (resaltada con un cuadro rojo) reduce la distorsión en un 8% sin apenas aumentar el coste. Incluso, podríamos llegar hasta la solución resaltada en verde en la que se reduce la distorsión hasta 25 produciéndose un aumento del coste de sólo el 2,4%. A partir de aquí, resulta algo más costoso reducir la distorsión.

Por el lado contrario, analicemos las soluciones de menor distorsión, quizás más interesantes en tiempos de expansión (ya que aunque más costosas permiten captar a más clientes). En este tramo, la pendiente de la curva es más pronunciada por lo que un pequeño ahorro supone un aumento considerable de la distorsión. No obstante, podríamos llegar hasta la solución resaltada en amarillo que con distorsión 13 reduce en un 5,3% el coste.

Por consiguiente, se puede concluir que este enfoque multiobjetivo y con las herramientas adecuadas (como el algoritmo MOAMP) facilita un conjunto de soluciones que ayuda a los responsables de la empresa a tomar la decisión final según sus intereses y prioridades en cada momento.

11. Conclusiones

En este trabajo se ha desarrollado un método para resolver una variante bi-objetivo del PVRP. Los dos objetivos analizados son: reducción de los costes y reducción de los cambios de calendario a los clientes. Estos objetivos no son fácilmente agregables ni comparables y además entran en conflicto. Es más la percepción que se tiene de la importancia de cada uno de ellos puede cambiar según las circunstancias económicas (época de crisis o época de expansión). Hay que destacar que este problema es nuevo en sí y es abordado desde una perspectiva que considera las costumbres adquiridas por los clientes como un nuevo objetivo y no como una restricción.

El método desarrollado está basado en la estrategia MOAMP (estrategia basada a su vez en movimientos vecinales) y aporta un conjunto denso de soluciones con las que se puede encontrar un equilibrio entre los objetivos. Para contrastar los resultados de este método MOAMP, éstos se comparan con los obtenidos por una adaptación de NSGA II. Los resultados muestran que con MOAMP se una curva de eficiencia más densa, poblada y de mayor calidad por la obtenida con NSGA II (en cuanto a dominancia de una sobre otra). No obstante el diseño de métodos basados en MOAMP, en este problema u otros similares, conlleva un gran esfuerzo de diseño e implementación ya que exige una adaptación "ad-hoc" para cada problema (por ejemplo, en la definición de movimientos vecinales).

Finalmente, el enfoque multi-objetivo, con las herramientas adecuadas que nos suministren buenas aproximaciones a la curva de eficiencia (como las que aquí se desarrollan) permiten tener una más amplia visión de las soluciones disponibles y la evolución de los diferentes objetivos en las mismas. Por tanto, teniendo en cuanta la importancia relativa que tiene cada objetivo, ayuda a la elección de la solución más adecuada.

Agradecimientos

Este trabajo ha sido realizado con la ayuda de Fondos FEDER y el Ministerio de Economía y Competitividad (a través de los proyecto ECO2013-47129-C4-3-R y ECO2016-76567-C4-2-R) la Junta de Castilla y León (a través del proyecto BU329U14) y la Junta de Castilla y León y Fondos FEDER (a través del proyecto BU062U16). Estas ayudas son reconocidas y agradecidas.

Referencias bibliográficas

1. Gulczynski D., B. Golden, and E. Wasil. (2011) "The period vehicle routing problem: New heuristics and real-world variants". Transportation Research Part E 47 (5), pp. 648-668.
2. Alegre J., M. Laguna, and J. Pacheco. (2007). "Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts". European Journal of Operational Research 179 (3) pp. 736-746.
3. Beltrami, E., and L. D. Bodin. (1974). "Networks and Vehicle Routing for Municipal Waste Collection". Networks 4 (1) pp. 65-94.
4. Clarke G., and J. W. Wright. (1964). "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". Operations Research 12 (4) pp. 568-581.
5. Russell R. and W. Igo. (1979). "An assignmente routing problem". Networks 9 pp. 1-17.
6. Christofides N. and J. Beasley. (1984). "The periodic routing problem". Networks 14 (2) pp. 237-256.
7. Tan C., and J. Beasley. (1984). "A heuristic algorithm for the periodic vehicle routing problem". Omega 12 (5) pp. 497-504.
8. Golden B. L., and E. A. Wasil. (1987). "Computerized Vehicle Routing in the Soft Drink Industry". Operations Research 35 (1) pp. 6-17.
9. Russell R. and D. Gribbin. (1991). "A multiphase approach to the period routing problem". Networks 21 (7) pp. 747-465.
10. Chao I-M., B. Golden, and E. Wasil. (1995). "An improved heuristic for the period vehicle routing problem", Networks 26 (1) pp. 25-44.
11. Cordeau J., M. Gendreau, and G. Laporte. (1997). "A tabu search heuristic for periodic and multi-depot vehicle routing problems". Networks 30 (2) pp. 105-119.
12. Drummond L., L. Ochl, and D. Vianna. (2001) "An asynchronous parallel metaheuristic for the periodic vehicle routing problem". Future Generation Computer Systems 17 (4) pp. 379-386.
13. Bertazzi L., G. Paletta, and M.G. Speranza. (2004). "An improved heuristic for the period traveling salesman problem". Computers & Operations Research 31 (8) pp. 1215-1222.
14. Blakeley F., B. Bozkaya, B. Cao, W. Hall, and J. Knolmajer. (2003). "Optimizing Periodic Maintenance Operations for Schindler Elevator Corporation". Interfaces 33 (1) pp. 67-79.
15. Nuortio T., J. Kytöjoki, H. Niska, and O. Bräysy. (2006). "Improved route planning and scheduling of waste collection and transport". Expert Systems with Applications 30 (2) pp. 223–232.
16. Ronen D., and C. A. Goodhart. (2008). "Tactical store delivery planning". Journal of the Operational Research Society 59 (8) pp. 1047–1054.
17. Pourghaderi A.R., R. Tavakkoli-Moghaddam, M. Alinaghian, and B. Beheshti-Pour. (2008). "A Simple and Effective heuristic for Periodic Vehicle Routing Problem". Procedings of the IEEE International Conference on Industrial Engineering and Engineering Management IEEM, Singapore, pp. 133-137.
18. Méndez A., D. Palumbo, M. Carnero, y J. Hernández. (2009). "Algoritmos meméticos aplicados a la resolución de un problema de ruteo de vehículos periódico". Mecánica Computacional 28 pp. 2675-2685.
19. Hemmelmayr, V. C., K. F. Doerner, and R. F. Hartl. (2009). "A Variable Neighborhood Search Heuristic for Periodic Routing Problems". European Journal of Operational Research 195 (3) pp. 791-802.

20. Peixoto T. M. (2010). “Optimization Techniques for the Mixed Urban Rural Solid Waste Collection Problem”. Tesis de Maestría. Faculdade de Engenharia da Universidade do Porto, Portugal.
21. Pirkwieser S., and G. R. Raidl. (2010). “Multilevel Variable Neighborhood Search for Periodic Routing Problems”. Lecture Notes in Computer Science 6022, pp. 226-238.
22. Yu B. and Z. Z.Yang. (2011). “An ant colony optimization model: The period vehicle routing problem with time windows”. Transportation Research Part E 47 (2) pp. 166-181.
23. Meyer A. (2012) “A constraint programming based approach for planning Milk Runs”. 18 International Conference on Principles and Practice of Constraint, Quebec.
24. Vidal T., T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei. (2012). “A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems”. Operations Research 60 (3) pp. 611-624.
25. Rademeyer A. L., and R. Bennetto. (2012). “A rich multi-period delivery vehicle master routing problem. A case study”. Proceedings of the 42 CIE, CIE & SAIIE, Cape Town.
26. Vidal T., T.G. Crainic, M. Gendreau, N. Lahrichi and W. Rei. (2012). “A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems”. Operations Research 60 pp. 611-624.
27. Pacheco J., A. Alvarez, I. García, and F. Angel-Bello. (2012). “Optimizing vehicle routes in a bakery company allowing flexibility in delivery dates”. Journal of Operational Research Society 63 (5) pp. 569–581.
28. Pacheco J., I. García, and A. Alvarez. (2014). “Enhancing Variable Neighborhood Search by Adding Memory: Application to a Real Logistic Problem”. Knowledge-Based Systems 62 pp. 28-37.
29. Hemmelmayr, V. C., K. F. Doerner, R. F. Hartl, and S. Rath. (2013). “A heuristic solution method for node routing based solid waste collection problems”. Journal of Heuristics 19 (2) pp. 129-156.
30. Yao, B., P. Hu, M. Zhang, and S. Wang. (2013). “Artificial bee colony algorithm with scanning strategy for the periodic vehicle routing problem”. Simulation-Transactions of the Society for Modeling and Simulation International 89 (6) pp. 762-770.
31. Cacchiani, V., V. C. Hemmelmayr, and F. Tricoire. (2014). “A set-covering based heuristic algorithm for the periodic vehicle routing problem”. Discrete Applied Mathematics 163 (1) pp. 53-64.
32. Francis P., K. Smilowitz, and M. Tzur. (2006). “The Periodic Vehicle Routing Problem with Service Choice”. Transportation Science 40 (4) pp. 439-454.
33. Baldacci R., E. Bartolini, A. Mingozzi, and A. Valletta. (2011). “An Exact Algorithm for the Period Routing Problem”. Operations Research 59 (1) pp. 228-241.
34. Ngueveu, S. U., C. Prins, and R. W. Calvo. (2013). “New Lower Bounds and Exact Method for the m-PVRP”. Transportation Science 47 (1) pp. 38-52.
35. Campbell, A.M., and J. H. Wilson. (2014). “Forty Years of Periodic Vehicle Routing”. Networks 63 (1) pp. 2-15.
36. Current, J., and M. Marsh. (1993). "Multiobjective Transportation Network Design and Routing Problems: Taxonomy and Annotation". European Journal of Operational Research 65 (1) pp. 4-19.
37. Jozefowiez, N., F. Semet, and E.-G. Talbi. (2008). "Multi-objective vehicle routing problems". European Journal of Operational Research 189 (2) pp. 293-309.
38. Pacheco, J., R. Caballero, M. Laguna, and J. Molina. (2013). "Bi-objective Bus Routing: An Application to School Buses in Rural Areas". Transportation Science 47 (3) pp. 397-411.
39. Gómez, J. R., J. Pacheco, and H. Gonzalo-Orden. (2015). “A Tabu Search method for a Bi-objective Urban Waste Collection Problem”. Computer-Aided Civil and Infrastructure Engineering 30 (1) pp. 36-53.
40. Smilowitz, K., M. Nowak, and T. Jiang. (2013). “Workforce Management in Periodic Delivery Operations”. Transportation Science 47 (2) pp. 214-230.
41. Hsieh, Y. C., P. S. You, P. J. Lee, and Y. C. Lee. (2015). “A novel encoding scheme based evolutionary approach for the bi-objective grid patrol routing problem with multiple vehicles”. Scientia Iranica 22 (4) pp. 1576-1585.
42. Caballero, R., J. Molina, and V. Rodríguez-Uría. (2003). “MOAMP: Programación Multiobjetivo Mediante un Procedimiento de Búsqueda Tabú”. Paper read at II Congreso Español de Metaheurísticas y Algoritmos Evolutivos y Bioinspirados (MAEB), at Gijón.

43. García, I., J. Pacheco, and A. Alvarez. (2013). "Optimizing Routes and Stock". Journal of Heuristics 19 (2) pp. 157-177.
44. Feo T.A., and M.G.C. Resende. (1989). “A probabilistic heuristic for a computationally difficult set covering problem”. Operations Research Letters 8 (2) pp. 67–71.
45. Feo T.A., and M.G.C. Resende. (1995). “Greedy randomized adaptive search procedures”. Journal of Global Optimization 6 (2) pp. 109–133.
46. Glover F. (1989). "Tabu Search — Part I”. ORSA Journal on Computing 1 (3) pp. 190-206.
47. Glover F. (1990). "Tabu Search — Part II”. ORSA Journal on Computing 2 (1) pp. 4-32
48. Glover, M., and M. Laguna. (1997). Tabu Search. Boston: Kluwer Academic Publishers.
49. Or, I. (1976). “Traveling salesman-type combinatorial problems and their relation to the logistic of regional blood banking. In Ph.D. dissertation”. Departament of Industrial Engineering and Management Sciences. Northwestern University, Evanston, IL.
50. Taillard, E., P. Badeau, M. Gendreau, F. Guertain, and J. Y. Potvin. (1997). "A Tabu Search heuristic for the Vehicle Routing Problem with Soft Time Windows". Transportation Science 31 (2) pp. 170-186.
51. Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan. (2002). "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II". IEEE Transactions on Evolutionary Computation 6 (2) pp. 182–197.