UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE CIENCIAS MATEMÁTICAS

Departamento de Estadística e Investigación Operativa

TESIS DOCTORAL

Métodos heurísticos para un problema multicriterio de distribución de ayuda humanitaria

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

José María Ferrer Caja

Directores

María Teresa Ortuño Sánchez
Gregorio Tirado Domínguez

Madrid, 2017

Métodos Heurísticos para un Problema Multicriterio de Distribución de Ayuda Humanitaria

Tesis doctoral realizada por:

D. José María Ferrer Caja

Bajo la dirección de:

Dra. Dª. M. Teresa Ortuño Sánchez

Dr. D. Gregorio Tirado Domínguez

Noviembre 2015

Departamento de Estadística e Investigación Operativa

Facultad de Ciencias Matemáticas
Universidad Complutense de Madrid

D^{a}. M. Teresa Ortuño Sánchez y D. Gregorio Tirado Domínguez, Profesores de la Universidad Complutense de Madrid, AUTORIZAN:

La presentación de la Tesis Doctoral titulada ***Métodos Heurísticos para un Problema Multicriterio de Distribución de Ayuda Humanitaria***, realizada por D. José María Ferrer Caja en el Departamento de Estadística e Investigación Operativa y que presenta para la obtención del grado de Doctor, por la Universidad Complutense de Madrid.

En Madrid, a 10 de noviembre de 2015

M. Teresa Ortuño Sánchez

Gregorio Tirado Domínguez

A mis padres, Emilio y Anita

Agradecimientos

A Teresa Ortuño, principal artífice de que por fin haya podido lograr este objetivo tan importante para mí cuando ya había perdido la esperanza. La confianza que depositó en mí, sin pedir nada a cambio, ha sido un estímulo imprescindible para cumplir el compromiso que contraje con ella al retomar este proyecto. Su talento y su visión global han sido fundamentales para encauzar el trabajo en la dirección correcta.

A Gregorio Tirado, que, tras incorporarse como codirector, se ha volcado completamente en su labor, dedicándome una impagable cantidad de tiempo, animándome en todo momento, aportando excelentes ideas, corrigiendo mis errores una y otra vez y, sobre todo, dirigiéndome de forma perfecta.

Mi más profundo agradecimiento a mis dos directores por todas las molestias que se han tomado y la paciencia que han tenido conmigo, especialmente estos últimos meses.

A Javier Montero y Begoña Vitoriano, responsables de los proyectos de investigación gracias a los cuales he podido asistir a mis primeros congresos y realizar mis primeras publicaciones, dando a conocer el trabajo de investigación cuyo resultado queda plasmado en esta tesis.

A mis compañeros del Departamento de Estadística e Investigación Operativa de la UCM y, en especial, de la Sección Departamental de Estadística e Investigación Operativa de la Facultad de Medicina, que tanto me han apoyado y se han interesado en mis progresos.

A los responsables y financiadores de EOLO, el clúster de computación de Cambio Global y Nuevas Energías del Campus de Excelencia Internacional Moncloa, donde he podido realizar de forma óptima todas las pruebas computacionales que he necesitado para obtener los resultados que se presentan en esta memoria.

A toda mi familia, por confiar en mí, comprenderme y apoyarme en todos los sentidos.

A mis amigos, que siempre me han animado y durante estos años han entendido que no pudiera dedicarles el tiempo que me gustaría y sin duda merecen.

Y sobre todo a Raquel, por su cariño, su apoyo incondicional y sus ánimos cuando más los he necesitado, sin los cuales hubiera sido imposible terminar este trabajo.

Índice general

Prólogo

En la última década hemos sido testigos de grandes desastres, como los causados por el terremoto de Nepal en 2015, el tifón Haiyan, que asoló Filipinas en 2013, el terremoto y tsunami de Japón de 2011, las inundaciones de Pakistán en 2010 o el devastador terremoto que golpeó Haiti en 2010 provocando más de 300000 víctimas mortales. Estos hechos tienen efectos terribles y duraderos sobre las poblaciones afectadas.

Desde el principio de la historia, los seres humanos han tenido que enfrentarse a los impactos y consecuencias de catástrofes de diversa índole. Además, lejos de disminuir, los desastres, naturales o provocados por el ser humano, afectan cada vez a mayor número de personas por todo el mundo.

Proporcionar ayuda de emergencia a las víctimas de estos desastres es un reto para los gobiernos, organizaciones y para la sociedad entera en todo el mundo. Las organizaciones humanitarias se enfrentan a multiples problemas de decisión al tratar de responder (Kovács y Spens, 2007; Tomasini y van Wassenhove, 2009a), puesto que se trata de un proceso muy complejo con un alto nivel de incertidumbre y se cuenta con recursos escasos. Además, la responsabilidad en la toma de decisiones recae habitualmente en organizaciones heterogéneas, que han de actuar de forma coordinada, complicando todavía más la administración de la crisis. Aunque actualmente existe mucha información que podría usarse para mejorar la toma de decisiones en la gestión de desastres, normalmente no está disponible en el momento oportuno o en la forma adecuada, o bien esa información no es suficientemente precisa para poder ser utilizada en la dirección correcta.

Frente a los desastres, además es necesario considerar la situación en la que vive la mayor parte de la población mundial. Según el informe sobre Desarrollo Humano 2014 publicado por el Programa de las Naciones Unidas para el Desarrollo (PNUD), en la actualidad hay más de 2.200 millones de personas viviendo en pobreza multidimensional o muy cerca de esa condición. Si se añade que buena parte de las catástrofes naturales se producen en zonas en vías de desarrollo, las consecuencias de las mismas son devastadoras. Las cifras muestran cómo las consecuencias de un desastre varían enormemente en función del nivel de desarrollo humano de los países afectados. En la última década, la media de muertos por desastre fue de 44 en países de alto desarrollo humano y de 300 en países en vías de desarrollo.

Estas situaciones a menudo sobrepasan la capacidad de respuesta de las administracio-

nes locales, de modo que se requiere la cooperación internacional para atender a las zonas afectadas. Gracias a las comunicaciones actuales, las noticias de estas tragedias llegan a la comunidad internacional de forma casi instantánea, y la ayuda se puede movilizar en cuestión de horas. Este caudal de ayuda inmediata puede beneficiar considerablemente a un país devastado por un desastre, pero el compromiso de atender dichas necesidades requiere de una logística que asegure la eficacia de las actuaciones y de los medios invertidos. Este tipo de logística se conoce como logística humanitaria.

En esta tesis se estudia un problema concreto en logística humanitaria. El problema aborda la organización del reparto de ayuda desde unos puntos donde se encuentra almacenada cierta cantidad de material, que serán denominados depósitos, hasta los lugares donde se necesita la ayuda en última instancia. Se trata de la última fase de la distribución -conocida en inglés como *last mile distribution*-, realizada sobre la zona afectada por el desastre. Previamente, el material de ayuda humanitaria habrá debido ser reunido y transportado a los depósitos, en muchas ocasiones desde otros países.

Concretamente, el problema consiste en diseñar un conjunto de rutas para que los vehículos disponibles para tal fin desplacen la ayuda humanitaria desde los depósitos hasta los lugares que la requieran. En la elaboración de las rutas se han de contemplar diversos factores con los que evaluar el plan de actuación. El reparto ha de ser rápido y económicamente eficiente, pero también equitativo en la medida de lo posible y, eventualmente, prestar atención a los distintos niveles de urgencia que existan en los lugares de destino. Además se deben tener en cuenta el estado de la red de comunicaciones y el posible ambiente de inseguridad que se haya podido crear tras el desastre, que pueden dificultar que el transporte de la ayuda se realice con normalidad. Todos estos factores hacen de este un problema multicriterio, y como tal se deberá enfocar.

El elevado número de elementos que intervienen, y las relaciones específicas que existen entre ellos, provocan que el problema presente una gran dificultad de forma que, si se desea modelar con hipótesis suficientemente realistas, resulta aconsejable y muchas veces necesario el uso de técnicas heurísticas para resolverlo. La compleja estructura de las soluciones factibles del problema determina en gran medida el tipo de métodos heurísticos y metaheurísticos que se pueden aplicar. Concretamente, cualquier pequeña modificación realizada en una solución dada implica con mucha frecuencia la pérdida de la factibilidad de dicha solución. Además, comprobar si una solución es factible no es un proceso trivial, ya que requiere realizar un conjunto de operaciones que suponen cierto tiempo de computación. Por esta razón, las metaheurísticas que mejor se adaptan al problema son las que incluyen un mayor componente constructivo en su metodología, como es el caso de las metaheurísticas GRASP (del inglés *Greedy Randomized Adaptive Search Procedure*) y colonia de hormigas.

La presente monografía se compone de ocho capítulos. Su contenido se resume brevemente a continuación.

En el Capítulo 1 se presenta una revisión bibliográfica de los trabajos dedicados al desa-

rrollo y aplicación de métodos matemáticos en logística humanitaria y gestión de desastres, prestando especial atención a los problemas centrados en la última fase del reparto. Asimismo, se realiza una revisión de las metaheurísticas aplicadas a problemas de rutas de vehículos semejantes al que se trata en esta memoria.

En el Capítulo 2 se describe el problema que es objeto de estudio de este trabajo, definiendo con detalle los elementos que lo conforman y proponiendo medidas para los atributos en consideración. Se presenta una formulación matemática como modelo de programación lineal entera, y se establece cómo se tratará la naturaleza multicriterio del problema.

En el Capítulo 3 se propone un algoritmo heurístico para construir soluciones factibles del problema. La construcción constará de varias fases y debe prestar especial atención a la relación entre los convoyes de vehículos que se forman para recorrer las vías de comunicación entre los depósitos y los puntos de demanda. En algunas fases de la construcción se introduce el azar para decidir qué elementos deben formar parte de la solución. De esta forma, se diversifica el conjunto de soluciones que puedan ser obtenidas mediante el algoritmo.

En el Capítulo 4 se presenta un conjunto de procedimientos heurísticos para depurar las soluciones factibles construidas, que denominaremos Algoritmo de Mejora, con el propósito de mejorarlas en diversos aspectos. En el proceso se debe vigilar que se preserve la factibilidad de la solución, que en este problema es de muy frágil naturaleza.

En el Capítulo 5 se presenta una adaptación de la metaheurística GRASP para resolver el problema. Esta metaheurística permite guiar la construcción de soluciones a través de funciones *greedy* y del conjunto con las mejores soluciones que vayan siendo obtenidas. También se explica cómo se ha modificado el algoritmo constructivo ya propuesto para servir de base a la metaheurística.

En el Capítulo 6 se adapta la metaheurística colonia de hormigas para crear otro método de resolución. En este caso, los procedimientos constructivos son modificados convenientemente para introducir el uso de rastros de feromonas que permitan transmitir la información ofrecida por soluciones ya construidas para guiar la obtención de nuevas soluciones. En este trabajo, debido a la complejidad del problema, las feromonas se han tratado de forma novedosa en algunos aspectos para potenciar su utilidad.

En el Capítulo 7 se presentan los resultados computacionales obtenidos al aplicar los algoritmos propuestos en los capítulos anteriores a distintas instancias del problema y casos de estudio extraídos de situaciones reales. En primer lugar se analiza la eficacia del algoritmo de mejora cuando se aplica a las soluciones obtenidas mediante el algoritmo constructivo. A continuación se explica cómo se han calibrado los parámetros de las metaheurísticas. La parte central del capítulo se dedica a un caso de estudio basado en el terremoto que sucedió en Haití en 2010, en el que se analiza la relación entre los criterios de optimización, se compara el rendimiento de los algoritmos heurísticos desarrollados y se propone un conjunto de soluciones bajo distintos puntos de vista. Finalmente, se estudian brevemente otros casos de estudio, tanto basados en casos reales como generados al azar.

En el Capítulo 8 se destacan las conclusiones más importantes que se derivan de esta tesis, se resumen sus principales aportaciones y se indican algunas líneas futuras de investigación.

Por último, se incluye un resumen en inglés de esta monografía, un apéndice describiendo con todo detalle algunos procedimientos modificados que forman parte de las metaheurísticas y una amplia bibliografía.

Objetivos de la tesis

El propósito de esta tesis es el estudio de un problema de distribución de recursos en logística humanitaria. En particular, se trata de un problema de reparto de ayuda en zonas vulnerables, con redes viarias potencialmente dañadas y donde el transporte se realiza en condiciones de inseguridad. Se busca diseñar procedimientos que permitan planificar dicho reparto de la mejor manera posible en un tiempo razonable, teniendo en cuenta la naturaleza multicriterio del problema.

Para ello, se plantean los siguientes objetivos:

- Formular y describir detalladamente el problema, construyendo un modelo de optimización adecuado.
- Destacar la naturaleza multicriterio de los problemas de optimización que surgen en estos contextos y tratarla de forma conveniente en el problema concreto bajo estudio.
- Desarrollar algoritmos heurísticos capaces de construir soluciones factibles en un tiempo razonable.
- Adaptar metaheurísticas apropiadas que permitan guiar los procesos de construcción con el fin de aumentar la calidad de las soluciones.
- Efectuar una calibración apropiada de los parámetros de las metaheurísticas para aplicarlas con eficacia.
- Analizar en detalle los resultados obtenidos sobre un caso de estudio ya tratado en la literatura, correspondiente a la situación de distribución de ayuda después del terremoto de Haití (enero 2010).
- Realizar una extensa experiencia computacional que incluya diversos casos obtenidos a partir de situaciones reales, así como una batería de instancias de distintos tamaños. Estudiar el comportamiento de los algoritmos utilizados y presentar, relacionar y analizar los resultados obtenidos.

Capítulo 1

Estado del arte

Este capítulo está dedicado a introducir el contexto en el que se sitúa el problema que se tratará en la presente monografía -la gestión de desastres y la logística humanitaria-, así como las técnicas que se utilizarán para resolver el problema -heurísticas y metaheurísticas- y presentar una revisión de las publicaciones más relevantes en estos campos.

En la Sección 1.1 se presentan los principales conceptos relacionados con la gestión de desastres, se señalan las particularidades de la logística humanitaria y se introducen las distintas fases que forman el ciclo de la gestión de desastres. En la Sección 1.2 se presenta una revisión bibliográfica sobre modelos de ayuda a la decisión aplicados a la logística humanitaria, clasificándose la investigación según las distintas fases del ciclo de gestión de desastres. Por último, en la Sección 1.3, se realiza una breve introducción a las heurísticas y metaheurísticas, destacándose aquellas que serán aplicadas en este trabajo.

1.1. Introducción a la gestión de desastres

El número de desastres, naturales o inducidos por el ser humano, y su impacto en la población y en las infraestructuras parecen estar aumentando en las últimas décadas, y es necesario gestionar de la mejor manera posible sus consecuencias. En esta sección se introducen los principales conceptos usados en la gestión de desastres y emergencias así como en logística humanitaria.

1.1.1. Riesgos y desastres

Un riesgo es un evento que supone una amenaza o probabilidad de ocurrencia de un fenómeno potencialmente dañino dentro de un período de tiempo y área determinados. Puede ser:

- Natural: fenómeno natural causado por un evento de aparición rápida o lenta, que puede ser geofísico, hidrológico, climatológico, meteorológico o biológico. Fenómenos de este tipo son terremotos, corrimientos de tierra, tsunamis, volcanes, aludes, inundaciones, temperaturas extremas, sequías, incendios, ciclones, tormentas, epidemias, plagas, etc.

- Tecnológico o provocado por el ser humano: evento causado por humanos y que ocurre en o cerca de asentamientos humanos. Por ejemplo, conflictos, hambrunas, desplazamiento de poblaciones, accidentes industriales (vertidos tóxicos, escapes radioactivos), accidentes de transporte, etc.

Una emergencia es una situación que pone en inmediato riesgo la salud, la vida, las propiedades o el entorno.

Un desastre es una situación de ruptura del funcionamiento normal de un sistema o comunidad, que causa fuerte impacto sobre las personas, sus obras y su ambiente, superando la capacidad local de respuesta. En ocasiones, declarar o no una emergencia como un desastre es una decisión política, porque tiene consecuencias en la participación de otros estados o en los seguros, por ejemplo.

Catástrofe es un término que se usa normalmente para referirse a un desastre extremadamente grande. Este salto de magnitud se refleja en algunas características que introducen diferencias en la logística de la intervención.

1.1.2. Gestión de desastres y logística humanitaria

La respuesta a un desastre es un proceso complejo que lleva consigo una gran presión de tiempo, un elevado grado de incertidumbre y muchas partes involucradas. Normalmente intervienen diversas agencias autónomas cooperando para mitigar, preparar, responder y recuperar la sociedad de un conjunto dinámico de amenazas.

Las entidades involucradas son distintas según el tipo de desastre, sus consecuencias y el lugar donde acontece, debido al diferente grado de vulnerabilidad asociado a unas zonas u otras. Los agentes que intervienen se pueden clasificar en tres niveles:

- Nivel local: Es el primer nivel de respuesta, y lo llevan a cabo normalmente organizaciones civiles, agentes locales (policía, bomberos...) y protección civil. Es el nivel de las emergencias que no son declaradas desastres.
- Nivel nacional: Protección civil nacional y ejército, otras organizaciones gubernamentales, ONG. Intervienen cuando la emergencia se declara desastre. Algunas veces participan también organizaciones internacionales con oficinas locales.
- Nivel internacional: Gobiernos extranjeros, ONG internacionales de intervención en desastres y agencias de la Organización de las Naciones Unidas. La coordinación a este nivel es esencial, y normalmente es realizada por la OCHA (Oficina de la ONU para la Coordinación de Asuntos Humanitarios) y el IASC (*Inter-Agency Standing Committee*), organismo creado para la coordinación entre agencias de asistencia humanitaria. Este nivel se alcanza cuando la nación afectada no tiene capacidad de respuesta suficiente, debido a la magnitud del desastre o a la vulnerabilidad del país, y el gobierno nacional autoriza una operación humanitaria internacional.

El proceso de decisión en gestión de desastres es extremadamente difícil, debido a la multitud de agentes que están involucrados y a la complejidad de las tareas a realizar. Entre

estas tareas se encuentran las que definen la logística humanitaria, según la definición en el Congreso de Logística Humanitaria en 2004 (Fritz Institute): Proceso de planificación, ejecución y control del flujo eficiente y efectivo y el almacenamiento de mercancías y materiales, así como la información relacionada, desde el punto de origen al punto de consumo, con el fin de satisfacer las necesidades del beneficiario final y aliviar el sufrimiento de la gente vulnerable.

La logística humanitaria aparece no solo en el contexto de gestión de desastres; el Programa Mundial de Alimentos y la Organización Mundial de la Salud, por ejemplo, desarrollan muchas operaciones que pueden ser consideradas de logística humanitaria sin estar asociadas a ningún desastre específico. No obstante, es en la gestión de desastres donde la aplicación de la logística humanitaria es más compleja y donde surgen mayores diferencias con la logística empresarial. Por consiguiente, la revisión bibliográfica se centrará en la logística humanitaria aplicada a gestión desastres. Las diferencias principales entre cadenas de suministro humanitarias en el contexto de gestión de desastres y cadenas de suministro empresariales son las siguientes:

- Demanda imprevisible en términos de tiempo, ubicación geográfica, tipo y cantidad de mercancía.
- Plazo de entrega corto y repentino de grandes cantidades de una amplia variedad de productos y servicios.
- Escasez de recursos iniciales en términos de suministro, recursos humanos, tecnología, capacidad y financiación.
- Presencia de múltiples decisores que, a veces, pueden ser difíciles de identificar.
- Los destinatarios finales son las personas que sufren.
- El primer propósito es la eficacia, es decir, llegar a los beneficiarios, a tantos como se pueda, y debe ser el principal criterio a la hora de planificar estrategias.
- El proceso requiere de una total transparencia.

Más información acerca de las particularidades de la logística humanitaria y de sus diferencias con la logística empresarial puede encontrarse en Balcik y Beamon (2008) y Vitoriano *et al.* (2015).

1.1.3. Ciclo de gestión de desastres

En los procesos de toma de decisiones que se necesitan en logística humanitaria para la gestión de desastres, el contexto y la naturaleza de las decisiones a realizar cambian con el tiempo a medida que se avanza desde antes de que se produzca el desastre hasta después de que lo haya hecho. Decidir sobre las acciones preventivas para mitigar los efectos de un posible terremoto futuro no es lo mismo que decidir sobre las acciones precisas para llevar a cabo inmediatamente después de que se haya producido, ni sobre las tareas a realizar un mes después de ocurrido el terremoto. La incertidumbre y la urgencia asociadas al contexto

pueden variar mucho de una situación a otra, así como la naturaleza de las decisiones y los criterios de los agentes involucrados.

Esto ha llevado a distinguir cuatro fases sucesivas en la gestión de emergencias y desastres de acuerdo con la naturaleza principal de las tareas a realizar y su ubicación temporal respecto al momento del desastre:

- Mitigación: todas las acciones y decisiones a medio y largo plazo encaminadas a prevenir y mitigar las consecuencias de un desastre futuro, mientras este no es inminente. Tareas típicas de esta fase son la identificación de grupos de riesgo y niveles de vulnerabilidad y su tratamiento, o el desarrollo de sistemas de predicción y planes de emergencia y la asignación de recursos para ellos.

- Preparación: todas las intervenciones a corto plazo desde que los sistemas de predicción disponibles dan una alarma sobre un fenómeno adverso inminente hasta que finalmente se produce. Esto incluye el establecimiento de los sistemas de emergencia y los planes de evacuación, el seguimiento en tiempo real del riesgo, el análisis de los escenarios más probables, el refuerzo de las infraestructuras críticas, etc. Esta fase también incluye algunas decisiones a largo plazo, como el diseño de la red de ayuda o la localización y dotación de los inventarios.

- Respuesta: esta fase se centra en salvar vidas y se caracteriza por su corta duración, gran urgencia y alta incertidumbre. Habitualmente se divide en una primera fase de respuesta, dedicada al rescate y la asistencia médica urgente de heridos y afectados (dependiendo del tipo de desastre y su magnitud, dura alrededor de una semana desde el momento en que se produce el desastre), y una fase de respuesta a medio plazo, dedicada a estimar y mitigar las primeras necesidades potencialmente desatendidas de la población afectada como consecuencia de posibles daños en las infraestructuras y recursos vitales (refugio, asistencia médica ordinaria, agua y suministro de alimentos, etc.). Esta etapa a medio plazo, por lo general, consiste en el envío de ayuda desde fuera de la zona afectada, y puede durar semanas o incluso meses desde el momento en que se produce el desastre, dependiendo de su naturaleza y su magnitud, así como de las circunstancias económicas y el desarrollo del país afectado.

- Recuperación: esta fase se centra en lograr la eficiencia y se caracteriza por su larga duración, poca o moderada urgencia y baja incertidumbre. Consta de las acciones y decisiones a largo plazo, una vez superado el impacto inmediato del desastre, para restablecer el funcionamiento normal de la comunidad y reconstruir el tejido social, que incluye los recursos vitales, los servicios y la infraestructura, y las mejoras necesarias para no repetir las vulnerabilidades específicas mostradas por las poblaciones y lugares afectados. A veces, después de ciertos desastres, será necesario un flujo periódico de ayuda humanitaria para apoyar a las comunidades especialmente vulnerables, lo que no se corresponde con la gestión de desastres propiamente dicha.

Aunque esta división en fases es clara y bien fundada desde un punto de vista temporal y conceptual, es importante destacar que el proceso de gestión de desastres tiene que ser

entendido como un todo, donde las fases no son independientes entre sí, a pesar de su diferente enfoque. De esta manera, la fase de preparación se basa fundamentalmente en los sistemas de predicción creados en la fase de mitigación, por lo que la toma urgente de decisiones de la fase de preparación sería imposible sin los planes de análisis de la vulnerabilidad y de emergencia desarrollados en la fase previa. Del mismo modo, la asignación de recursos para las operaciones de la primera fase de respuesta debe ser tomada en cuenta a la hora de diseñar la política de mitigación. Además, la fase de respuesta a medio plazo no puede tener éxito si no permite un proceso de recuperación adecuado. En este sentido, el envío de ayuda externa debe ser cuidadosamente examinado y llevado a cabo para no destruir la economía local. Hay que señalar que las tareas de recuperación y mitigación se superponen parcialmente, de tal forma que una vez que una comunidad afectada ha alcanzado un nuevo equilibrio tiene que volver a evaluar la posible aparición de desastres en el futuro, por lo que una nueva fase de mitigación sigue a la fase de recuperación. Por lo tanto, el proceso de gestión de desastres sigue un desarrollo cíclico (ver Figura 1.1, Tomasini y van Wassenhove (2009b)) en el que cada fase se basa en las anteriores, mostrando claramente interdependencia entre todas ellas.

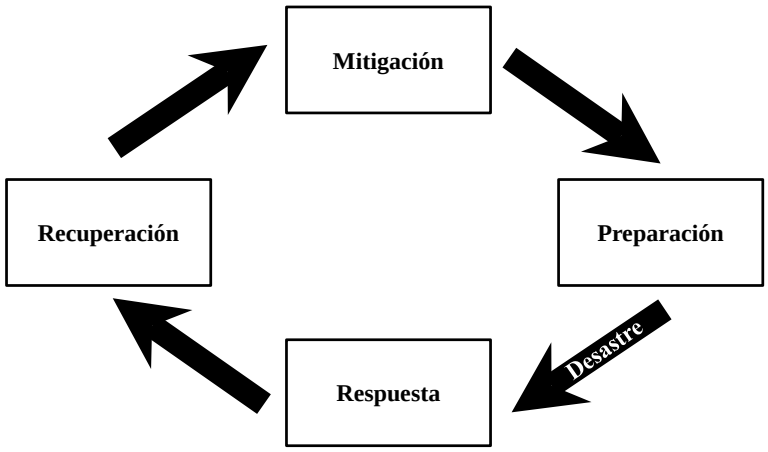


Figura 1.1: Ciclo del proceso de gestión de desastres

1.2. Modelos de decisión en la gestión de desastres

En los últimos años se ha producido una explosión de literatura, científica y de divulgación, sobre la cuestión de la gestión de desastres y emergencias, ya que se trata de un tema vital en el mundo globalizado actual. Entre esa literatura, va aumentando el número de trabajos que presentan modelos matemáticos que pueden ayudar en la toma de decisiones a las que se enfrentan las personas y organizaciones a la hora de dar respuesta a las consecuencias de un desastre.

En esta sección se presenta una revisión bibliográfica con trabajos publicados en la última década sobre este tipo de modelos, clasificados por la fase del ciclo de gestión

de desastres donde se plantean, donde se prestará especial atención a los problemas correspondientes a la última etapa del reparto de la fase de respuesta y a la aplicación de metaheurísticas y técnicas multicriterio.

Una revisión de los modelos de investigación operativa realizados hasta el año 2005 puede encontrarse en Altay y Green (2006), trabajo que se continúa en Galindo y Batta (2013) hasta 2011. En de la Torre *et al.* (2012) se efectúa una revisión de los problemas de rutas y distribución aplicados a la gestión de desastres, analizándose la aplicabilidad de los modelos revisados a situaciones reales. En Ortuño *et al.* (2013) se realiza una extensa revisión bibliográfica de modelos de decisión en logística humanitaria, clasificados por fase del ciclo de desastre y por método de solución, y de sistemas de ayuda a la decisión. Otras revisiones bibliográficas recientes son Liberatore *et al.* (2013), donde se analiza el tratamiento de la incertidumbre en la literatura de logística humanitaria, Zhang *et al.* (2014), acerca de métodos metaheurísticos inspirados en la naturaleza aplicados al transporte de emergencias, Hoyos *et al.* (2015), orientada a los modelos con componentes estocásticos, y Özdamar y Ertem (2015), que se centra en las fases de respuesta y recuperación.

1.2.1. Fases de mitigación y preparación

En esta subsección se comentan los principales trabajos que abordan problemas predesastre, esto es, englobados en las fases de mitigación o de preparación. Se han incluido también aquellos trabajos que combinan tareas de preparación con otras específicas de la fase de respuesta.

En Tovia (2007) se construye un modelo de simulación de respuesta urgente para ser utilizado por las oficinas de la preparación para emergencias. Está concebido para evaluar la capacidad de respuesta y valorar los recursos logísticos necesarios para evacuar, refugiar y proteger a la población, en el momento oportuno en caso de un huracán. En Zobel y Khansa (2014) se presenta un enfoque multicriterio que proporciona una medida cuantitativa de la resistencia ante la presencia de múltiples desastres relacionados, con el fin de poder comparar varias configuraciones y evaluar sus rendimientos y riesgos relativos para la gestión eficaz de la emergencia.

Un problema importante dentro de la fase de preparación es el de localización, de centros de ayuda, de refugios, etc. En este sentido, cabe destacar los siguientes trabajos: En Doerner *et al.* (2009) se proporciona un modelo de programación multicriterio para la localización de instalaciones públicas -como escuelas- en zonas cercanas a la costa, teniendo en cuenta los riesgos de inundación por tsunamis. Se realiza una comparación entre dos enfoques de solución: un algoritmo genético y una técnica de descomposición. En Eiselt y Marianov (2012) se propone un modelo de programación lineal binaria para la localización o el fortalecimiento de torres de telefonía móvil con el fin de maximizar la cobertura de los servicios y reducir al mínimo la pérdida de comunicaciones en caso de que ocurra un desastre natural. Por último, en Bozorgi-Amiri y Asvadi (2015) se aborda un problema de ubicación de centros de ayuda logística, en el que se pretende elaborar una clasificación de los centros de acuerdo a cinco criterios. Para resolverlo se emplean dos métodos basados

en programación por metas, uno lexicográfico y el otro logarítmico en dos pasos.

En Salmerón y Apte (2010) se aborda un problema de preposicionamiento, dotación de recursos y distribución mediante un modelo estocástico en dos etapas tratado de forma lexicográfica para su resolución. En Nolz *et al.* (2010b) se proporciona una metaheurística multicriterio basada en conceptos evolutivos para planificar circuitos de distribución de agua en las operaciones de socorro, determinando la ubicación física de los tanques de agua y seleccionando las vías que se utilizarán para el transporte de agua potable. El problema de localización y distribución de recursos también se trata en Esmaeili y Barzinpour (2014), en este caso en un ámbito urbano. Se propone un algoritmo genético multiobjetivo para su resolución. Un modelo similar se presenta en Barzinpour y Esmaeili (2014), donde se utiliza el enfoque de programación por metas para resolver el problema multicriterio planteado. En Bozorgi-Amiri *et al.* (2013), se desarrolla un modelo de programación compromiso estocástico para un conjunto de operaciones logísticas de socorro que incluyen localización, dotación y distribución de recursos, minimizando la suma del valor esperado y la varianza del coste total, y minimizando la suma de las demandas insatisfechas máximas en las zonas afectadas. Una variante mono-objetivo de este problema se trata en Bozorgi-Amiri *et al.* (2012) mediante un método de resolución basado en la metaheurística optimización de enjambre de partículas. El problema de preposicionamiento y posterior distribución se afronta en Rezaei-Malek y Tavakkoli-Moghaddam (2014) de un modo estocástico mediante programación robusta, y se propone un método interactivo para tratar el carácter multicriterio establecido por los objetivos tiempo medio de respuesta y coste total de la operación.

En cuanto al diseño de la cadena de suministro, en Van Wassenhove (2006) se muestra la complejidad de la gestión de las cadenas de suministro en contextos humanitarios y se esbozan las estrategias para una preparación más adecuada. En Hale y Moberg (2005) se considera un modelo de localización para el establecimiento de una red eficiente de instalaciones seguras de almacenamiento que puedan respaldar con eficacia las labores de la cadena de suministro.

También se ha tratado la evacuación de afectados ante un eventual desastre. En Saadatseresht *et al.* (2009) se presenta un enfoque en tres pasos basado en el uso de algoritmos evolutivos multicriterio y en un sistema de información geográfica para determinar la distribución de los evacuados a zonas seguras en caso de emergencia, mientras que en Bretschneider y Kimms (2012) se propone un método heurístico en dos etapas para un modelo dinámico de flujo basado en patrones con el fin de reestructurar las rutas de tráfico para la evacuación de un área urbana afectada por un desastre.

Respecto a los problemas de inventarios, en Qin *et al.* (2012) se propone un modelo de inventarios integrado intertemporal de un solo periodo para determinar el tamaño óptimo de pedido de los recursos de emergencia en un incidente de inundación. También se describe un modelo de simulación basado en algoritmos genéticos.

1.2.2. Fase de respuesta

La fase de respuesta ha despertado el interés de muchos investigadores, por ser la fase en la que quizá mayor rédito se pueda obtener por realizar una buena gestión. En esta subsección se presenta una selección de los trabajos publicados hasta la fecha en este campo.

A continuación se presentan los trabajos que plantean el problema de la última etapa de la distribución de ayuda humanitaria, en la que se hace llegar la ayuda a los puntos donde se necesita en última instancia, como ocurre con el problema que se tratará en esta monografía. También se incluyen los trabajos que combinan este problema con otros propios de la fase de respuesta.

- En Tzeng *et al.* (2007) se aplican técnicas de programación lineal difusa multiobjetivo para abordar un problema multiperiodo de distribución de ayuda, donde se trata de minimizar el coste total y la duración del reparto y maximizar la equidad a través de la satisfacción mínima.

- En Yi y Özdamar (2007) se describe un modelo de flujo que integra la localización, distribución y evacuación para coordinar las operaciones de apoyo logístico. El modelo determina el número de vehículos, el número de personas heridas y la cantidad de mercancías que se transportan por cada arco mediante programación lineal. A continuación se calculan las cargas que se han de distribuir entre los vehículos resolviendo un sistema de ecuaciones lineales. En Yi y Kumar (2007) se aplica la metaheurística colonia de hormigas para resolver este mismo problema, pero sin tener en cuenta la ubicación de los centros de emergencia.

- En Balcik y Beamon (2008) se presenta un modelo de planificación multiperiodo en dos fases para la distribución de ayuda. En la primera fase, se generan las posibles rutas para los vehículos dentro del horizonte temporal. En la segunda fase, se seleccionan las rutas óptimas y las cantidades de material que se transportan para los siguientes periodos, resolviendo el problema de programación entera subyacente.

- En Campbell *et al.* (2008) se introducen y analizan dos funciones objetivo alternativas para el problema del viajante y el problema de rutas de vehículos, considerando tiempos de llegada a los destinos, con el fin de ajustarse a la naturaleza humanitaria del reparto. Los problemas así planteados son resueltos mediante un enfoque heurístico basado en algoritmos de inserción y de búsqueda local.

- En Gibbons y Samaddar (2009) se desarrolla un modelo de simulación para un problema de distribución de vacunas en una situación de pandemia y un contexto de gran incertidumbre.

- En Shen *et al.* (2009) se plantea un problema de distribución de medicinas para el caso de un ataque terrorista con armas biológicas. El problema se descompone en dos fases: en la primera fase se determinan las rutas y en la segunda se deciden cómo se reparte el material médico y otros aspectos de la distribución. El método de resolución propuesto se basa en la búsqueda tabú.

- En Blecken *et al.* (2010) se aplican técnicas tanto exactas como heurísticas para afrontar un problema de dotación de depósitos y flujo de ayuda humanitaria en el que se minimiza el coste total.

- En Nolz *et al.* (2010a) se propone un método híbrido basado en algoritmos genéticos, búsqueda de entorno variable y encadenamiento de trayectorias (*path relinking*) para encarar un problema multicriterio de distribución de agua a la población afectada por un desastre. En Nolz *et al.* (2011) se introduce un objetivo basado en el riesgo, entendido como posibilidad de que no pueda llevarse a cabo la distribución o parte de ella. Se proponen diferentes alternativas para medir el riesgo basadas en las características de las conexiones.

- En Vitoriano *et al.* (2011) se presenta un modelo multicriterio de flujo doble para abordar el problema de distribución de bienes a la población afectada por un desastre. Se consideran conjuntamente los atributos coste, tiempo de respuesta, equidad en la distribución, seguridad y fiabilidad mediante el enfoque de programación por metas. Algunos resultados preliminares pueden encontrarse en Vitoriano *et al.* (2009) y Ortuño *et al.* (2011). En Tirado *et al.* (2014) se desarrolla un modelo de flujo dinámico considerando cuatro criterios (cantidad total repartida, tiempo de la operación, equidad de la distribución, y coste), que permite obtener una planificación detallada de la operación de distribución de ayuda.

- En Özdamar (2011) se describe un sistema de planificación para coordinar operaciones de ayuda con helicópteros, que consisten en transportar diversos elementos médicos a las localidades afectadas y evacuar a las personas heridas, minimizando la duración total de la misión. En Özdamar y Demir (2012) se propone un procedimiento jerárquico de agrupación de nodos para coordinar rutas de vehículos en las actividades de distribución y evacuación posteriores a los desastres.

- En Berkoune *et al.* (2012) se desarrolla un algoritmo genético para resolver un problema de rutas multi-producto y multi-depósito que tiene como objetivo minimizar la duración total de todos los viajes.

- En Huang *et al.* (2012) se afronta un problema de rutas en el que se consideran como funciones objetivo el coste, la velocidad del reparto y la equidad. Las tres variantes se resuelven mediante metaheurísticas basadas en el GRASP y se comparan los valores de las funciones objetivo.

- En Lin *et al.* (2012) se desarrolla un modelo multiobjetivo de rutas de vehículos para operaciones de logística humanitaria que considera varios productos, varios periodos, entrega dividida y ventanas de tiempo. Se proponen dos métodos heurísticos para su resolución, uno basado en algoritmos genéticos y otro en técnicas de descomposición.

- En Tricoire *et al.* (2012) se propone un modelo estocástico con recurso en el que, en la primera etapa, se deciden la ubicación de centros de distribución y las rutas de los vehículos disponibles desde el depósito hasta los centros de distribución y, en la segunda etapa, se determinan las cantidades transportadas. Los objetivos corresponden al coste de la operación y la demanda satisfecha esperada. Para resolver el

problema multicriterio se aplica el método de las ϵ-restricciones en conjunción con un método de ramificación y corte.

- En Afshar y Haghani (2012) se desarrolla un modelo integrado que describe las operaciones logísticas en respuesta a los desastres naturales. El modelo de flujo propuesto tiene en cuenta tanto las rutas de recogida y reparto como la ubicación óptima de instalaciones temporales en distintos niveles.

- En Ke y Feng (2013) se afronta un problema de rutas de vehículos de capacidades limitadas donde se pretende minimizar la suma de los tiempos de llegada a los destinos, aplicable en el ámbito de la logística humanitaria. El método de resolución propuesto consiste en una metaheurística que combina perturbaciones para modificar soluciones y búsqueda local para mejorarlas. Para resolver este problema, en Rivera *et al.* (2015) se aplica una metaheurística multi-comienzo de búsqueda local iterada (MS-ILS).

- En Najafi *et al.* (2013) se propone un modelo estocástico multiobjetivo y multiperiodo para gestionar la distribución de varios tipos de productos y la evacuación de personas heridas como respuesta a un terremoto. Se consideran tres objetivos a minimizar jerárquicamente: el número de personas no evacuadas, la demanda total insatisfecha y el número de vehículos usados.

- En Zheng y Ling (2013) se desarrolla un modelo de optimización difusa para la planificación global de la distribución de ayuda, considerando varios medios de transporte y tres objetivos principales: el coste de transporte, el retraso en la entrega de la ayuda y el riesgo del plan de transporte. El método de resolución incluye varias técnicas heurísticas, como la búsqueda tabú multiobjetivo y algoritmos genéticos.

- En Chang *et al.* (2014) se propone un algoritmo genético multiobjetivo con búsquedas *greedy* que proporciona conjuntos factibles de rutas para un problema de distribución de ayuda humanitaria, en el que se consideran la demanda insatisfecha, el tiempo y el coste de la operación como criterios de optimización.

- En Abounacer *et al.* (2014) se aborda un problema de localización y distribución, en el que se consideran tres objetivos: duración del reparto, número de personas involucradas en la operación y demanda no satisfecha. El problema multicriterio se resuelve mediante el método de las ϵ-restricciones para obtener la frontera de Pareto, y se aporta una versión heurística para reducir el tiempo de computación. El método de las ϵ-restricciones se aplica también en Rath y Gutjahr (2014) a un problema de localización de depósitos y rutas de distribución, desde los puntos de suministro a los depósitos y desde allí a los destinos, en el que se consideran objetivos relacionados con el coste y la demanda total cubierta. Los subproblemas mono-objetivo correspondientes son resueltos mediante un enfoque heurístico.

- En Liu y Guo (2014) se señala la dificultad existente en diseñar planes de preposicionamiento ante posibles terremotos por la gran impredecibilidad de sus consecuencias y se propone un modelo estocástico para planificar, una vez que el terremoto ya se

ha producido, tareas de localización de centros de distribución, dotación de recursos y distribución por medio de helicópteros considerando objetivos basados en el coste y la demanda satisfecha esperada. El problema multicriterio se aborda con un enfoque lexicográfico no estricto, considerando el segundo objetivo de mayor importancia que el primero, y aplicando algoritmos heurísticos a los subproblemas que resultan de la descomposición en escenarios.

- En Barzinpour *et al.* (2014) se propone un algoritmo genético para resolver un problema dinámico de localización y distribución, en el que se pretende minimizar costes y maximizar la equidad de la distribución, usando como medida la mínima proporción de demanda satisfecha entre las zonas afectadas.

- En Allahviranloo *et al.* (2014) se critica la aplicación de programación estocástica con recurso a problemas de rutas en los que se dispone de medios y bienes limitados, como en el caso de la logística humanitaria. En su lugar, se proponen diversos modelos de rutas selectivos (SVRP) con dos objetivos, siendo el primero de ellos una agregación entre el coste y el tiempo de la operación, dependiendo el segundo objetivo del modelo propuesto. Como método de resolución, se utilizan algoritmos genéticos en paralelo.

- En Sharif y Salari (2015) se aplica la metaheurística GRASP a un problema de rutas de vehículos con fines humanitarios. Los vehículos parten de un único depósito y no están obligados a regresar una vez finalizado el reparto. Se permite que algunos destinos puedan no ser visitados por los vehículos, siempre que se encuentren cerca de otros destinos que sí hayan sido visitados, donde los vehículos habrán dejado el material suficiente para cubrir la demanda de los destinos cercanos no visitados.

- En Huang *et al.* (2015) se proponen medidas para atributos que consideran la efectividad salvando vidas, las demoras en la atención a los damnificados y la equidad en la distribución de ayuda. Los tres atributos se agregan en una función objetivo escalar en un modelo dinámico de programación cuadrática para abordar un problema de asignación y distribución de recursos.

Algunos trabajos se centran en la distribución a una escala mayor, incluyendo transporte internacional desde países colaboradores. Por ejemplo, en Angelis *et al.* (2007) se considera un modelo de programación lineal entera para resolver un problema multiperiodo de planificación semanal de rutas aéreas para transportar comida a Angola, en el contexto del Programa Mundial de Alimentos de la Organización de las Naciones Unidas. En Adıvar y Mert (2010) se propone un modelo de programación lineal difusa para diseñar un plan de transporte de ayuda internacional desde países donantes al país necesitado. En Charles (2010) se desarrolla un modelo global en varios niveles de localización y distribución. Este modelo de programación entera mixta incluye proveedores internacionales, potenciales depósitos, zonas afectadas y dos tipos de medios de transporte: barcos y aviones. En Afsar *et al.* (2014) se aborda un problema de rutas generalizado con un número variable de vehículos, donde los nodos están agrupados en conjuntos y se ha de visitar un nodo de cada conjunto. El problema, aplicable a la logística humanitaria y, particularmente, al transporte de ayuda hasta la zona afectada por un desastre, tiene como objetivo minimizar

el coste total de la operación y es resuelto mediante distintos métodos exactos y heurísticos.

El problema de la evacuación de afectados por un desastre, tratado desde el punto de vista de la fase de respuesta, también ha sido ampliamente estudiado en la literatura. En Kongsomsaksaku *et al.* (2005) se proporciona un modelo de planificación de evacuación de inundaciones en dos niveles: en primer lugar, las autoridades deciden acerca de la ubicación y capacidad de los refugios; en segundo lugar, los evacuados deciden dónde refugiarse y la ruta de evacuación. El problema de programación se resuelve usando un algoritmo genético. En Chiu y Zheng (2007) se presenta un modelo lineal de planificación de evacuación y apoyo sobre una red de transporte para la respuesta urgente en tiempo real ante desastres imprevistos. En Sheu (2007) se presenta un plan de ayuda dinámica con varios propósitos para ser utilizado tras producirse un terremoto: agrupación de zonas, estimación de las personas afectadas en cada zona, estimación de las prioridades y distribución de la ayuda. Se aplican diferentes técnicas matemáticas, tales como agrupamiento difuso (*fuzzy clustering*) y programación dinámica multiobjetivo. En Sheu (2010) se propone un modelo similar centrado en las cantidades demandadas de ayuda (en lugar del número de personas afectadas), que se estiman de forma dinámica. En Jotshi *et al.* (2009) se propone un modelo de simulación que usa fusión de datos (*data fusion*) para un problema de emergencia de recogida y traslado de pacientes a hospitales. En Ben-Tal *et al.* (2011) se aplica optimización robusta para un problema de transporte multiperiodo que consiste en asignar dinámicamente la respuesta de emergencia y el flujo de evacuados con incertidumbre dependiente del tiempo en las demandas. En Lim *et al.* (2012) se aplica un método que combina el algoritmo de Dijkstra con un algoritmo *greedy* a un problema de evacuación urgente sobre una red de transporte.

En el campo de la gestión de inventarios, se puede destacar el trabajo Beamon y Kotleba (2006b), donde se introduce un modelo de inventario con varios proveedores, demandas aleatorias discretas y dos tipos de pedidos: pedidos normales y pedidos de emergencia. En Beamon y Kotleba (2006a) se desarrollan y prueban tres estrategias diferentes de gestión de inventarios para el modelo anterior aplicado en el contexto de la situación de emergencia asociada a la guerra civil de Sudán del Sur.

Por último, se comentan algunos trabajos de temática dispar. En Jin y Ekşioğlu (2010) se presenta un modelo de programación entera para determinar rutas alternativas en una red de carreteras tras producirse un desastre. Se propone un algoritmo para actualizar los parámetros del programa lineal simplificado y se prueba utilizando escenarios simulados. También se ha tratado el problema de la adjudicación de contratos a empresas en competencia para el suministro de recursos en el contexto de la gestión de desastres (Ertem *et al.*, 2010, 2012), proponiéndose enfoques tanto exactos como heurísticos para su resolución. En Zhang *et al.* (2012a) se plantea un problema de dotación de recursos de varios tipos a un conjunto de depósitos bajo la posibilidad de que se produzcan réplicas del desastre ya acontecido. El modelo lineal entero propuesto se resuelve mediante un método heurístico. En Falasca y Zobel (2012) se aborda un problema de asignación de voluntarios a labores humanitarias, teniendo en cuenta la aptitud de los voluntarios para las distintas tareas pero también sus preferencias personales.

1.2.3. Fase de recuperación

Se concluye esta sección comentando algunos trabajos relacionados con la fase de recuperación de la gestión de desastres.

Algunas publicaciones tratan la recuperación de un desastre desde un punto de vista global. Por ejemplo, en Chen *et al.* (2009) se propone una medida cuantitativa para evaluar la capacidad de recuperación, y se establece un modelo de recuperabilidad para minimizar el tiempo de recuperación y optimizar la asignación de recursos. En Fiedrich y Burghardt (2007) se muestra cómo se pueden utilizar las aplicaciones de la tecnología de agentes (*agent technology*) para apoyar muchos procesos a lo largo de las fases del ciclo de gestión de desastres, desde la mitigación y la preparación hasta la respuesta real y la recuperación. En Kaklauskas *et al.* (2009) se describe el desarrollo de un modelo de conocimiento para la gestión postdesastre mediante teoría de la decisión multicriterio. En Cimellaro *et al.* (2010) se presenta una evaluación cuantitativa de la capacidad de recuperación de desastres que se implementa para evaluar la red de centros de salud en una zona proclive a sufrir terremotos.

El mayor número de trabajos acerca de esta fase tratan sobre la recuperación de infraestructuras civiles. En Karlaftis *et al.* (2007) se presenta una metodología para asignar de manera óptima los fondos para la reparación de una red de transporte de infraestructura urbana, dañada por un desastre natural, utilizando un enfoque basado en un algoritmo genético de tres etapas. También se aplican algoritmos genéticos en Permann (2007), en este caso a un problema de modelado y análisis de las redes de infraestructura con el fin de determinar las inversiones óptimas para restaurar o protegerla de ataques u otros desastres. En Orabi *et al.* (2009) se presenta un modelo para asignar recursos limitados a proyectos en competencia de reconstrucción de redes de transporte dañadas, generando equilibrios óptimos entre la duración y el coste de reconstrucción. En Orabi *et al.* (2010) se optimizan los esfuerzos de recuperación de los sistemas de infraestructura civil dañados por un desastre, minimizando tanto la pérdida de rendimiento de la red de transporte dañada como los costes de la reconstrucción. En Matisziw *et al.* (2010) se describe un modelo multiobjetivo para garantizar que se priorice la restauración de la red durante la recuperación de un desastre de forma que se optimice el rendimiento del sistema. En Mehlhorn *et al.* (2011) se presenta un plan de reparación de puentes para restaurar una red vial que permite el acceso a las instalaciones clave de la zona dañada. En Maya y Sörensen (2011) se afronta el problema de asignar los escasos recursos disponibles para reparar una red de caminos rurales dañados por un desastre, natural o de origen humano. Se proponen dos metaheurísticas para resolverlo, GRASP, y búsqueda de entorno variable (VNS). En Liberatore *et al.* (2014) se plantea un problema multicriterio de planificación de la recuperación de los elementos dañados de la red de distribución. Para resolverlo, se propone un método lexicográfico en tres niveles, donde en el primer nivel se maximiza la demanda satisfecha y en los dos siguientes se aplica programación compromiso.

En el campo concreto de la restauración de los sistemas de energía, en Ang (2006) se describe un modelo para planificar la recuperación de una red de energía eléctrica que ha sido dañado por un desastre natural o un ataque terrorista. En Coffrin *et al.* (2011a) se

modela y resuelve el problema de la planificación de la restauración del sistema de energía para la recuperación de desastres, mientras que en Coffrin *et al.* (2011b) se presenta un modelo para decidir cómo almacenar los componentes del sistema de energía a través de una zona poblada afectada por un desastre para maximizar la cantidad de energía dispensada.

En Jung *et al.* (2007) se introducen algunos modelos matemáticos para la planificación de la reconstrucción de las zonas urbanas y el diseño de nuevas estrategias de urbanización que minimicen el riesgo futuro de inundaciones y maximicen el beneficio social neto bajo limitaciones espaciales y económicas. En El-Anwar *et al.* (2010) se estudia cómo cuantificar y maximizar la sostenibilidad de las actividades de recuperación de la vivienda tras producirse desastres naturales.

La gestión de escombros aparece en algunos trabajos, de los que se pueden destacar los siguientes. En Fetter y Rakes (2012) se propone un modelo de programación entera mixta para planificar la retirada de escombros hacia instalaciones temporales de eliminación y almacenamiento de residuos. En Hu y Sheu (2013) se introduce un modelo dinámico de retirada y gestión de los escombros ocasionados por un desastre, con el propósito de minimizar los costes de la operación, los costes medioambientales y los costes psicológicos derivados de los tiempos de espera. En Özdamar *et al.* (2014) se propone un algoritmo heurístico constructivo que genera planes de limpieza de escombros en la red de carreteras para minimizar la duración de la operación y maximizar la accesibilidad de la red.

En Saleem *et al.* (2008) se propone un modelo para la preparación previa a un desastre y la rápida recuperación de la continuidad del negocio después del desastre. El modelo se utiliza para diseñar y desarrollar un prototipo de *web* de su sistema de red de información de la continuidad del negocio, facilitando la colaboración entre el estado, las agencias federales locales y la comunidad empresarial para la rápida recuperación de desastres. En de Mel *et al.* (2008) se cuantifica la recuperación psicológica y económica de los propietarios de pequeñas empresas afectadas por el tsunami ocurrido en Sri Lanka de en 2004.

1.3. Heurísticas y metaheurísticas

1.3.1. Introducción

Los algoritmos heurísticos, o simplemente heurísticas, son métodos con los que se pretende resolver de forma aproximada un problema de optimización en un corto espacio de tiempo. Los algoritmos metaheurísticos, o metaheurísticas, son métodos generales iterativos que, a través de un conjunto de parámetros controlables, guían ciertos procedimientos subordinados con el mismo fin que los algoritmos heurísticos, esto es, proporcionar buenas soluciones factibles de forma rápida. Los procedimientos subordinados suelen ser, a su vez, algoritmos heurísticos. En Blum y Roli (2003) se recogen algunas de las definiciones propuestas para estos términos, que en ocasiones presentan límites difusos, y se establecen las siguientes propiedades que caracterizan a las metaheurísticas:

- Las metaheurísticas son estrategias que guían el proceso de búsqueda.
- El objetivo es explorar de manera eficiente el espacio de búsqueda con el fin de encontrar soluciones óptimas o casi óptimas.
- Las técnicas que constituyen los algoritmos metaheurísticos abarcan desde simples procedimientos de búsqueda local a complejos procesos de aprendizaje.
- Los algoritmos metaheurísticos son aproximados y, por lo general, no deterministas.
- Suelen incorporar mecanismos para evitar quedarse atrapado en áreas confinadas del espacio de búsqueda.
- Los conceptos básicos de las metaheurísticas deben ser simples y claros, y deben permitir una descripción abstracta fácil de entender.
- Las metaheurísticas no son específicas de cada problema, pero pueden hacer uso de las particularidades del problema a través de los procedimientos heurísticos subordinados.
- Las metaheurísticas más avanzadas utilizan el aprendizaje para guiar la búsqueda de forma eficiente.

Las heurísticas y metaheurísticas son apropiadas para aplicar a problemas para los que no se conoce un método exacto de resolución, o bien, el tiempo de computación necesario para encontrar el óptimo mediante los métodos exactos conocidos es demasiado grande. No garantizan la obtención de una solución óptima, aunque aspiran a ello o, al menos, a proporcionar soluciones factibles cercanas al óptimo del problema.

Un grupo importante de metaheurísticas se basa en ideas provenientes de las ciencias naturales, como la biología (colonia de hormigas, optimización de enjambre de partículas), la genética (algoritmos genéticos y evolutivos) o la física (temple simulado), mientras que otras metaheurísticas utilizan básicamente conceptos propios de las matemáticas y las ciencias de la computación (búsqueda tabú, GRASP, búsqueda en entorno variable, etc.). Para una visión global sobre las metaheurísticas, puede consultarse Talbi (2009), donde se ofrece información acerca del diseño e implementación de una amplísima selección de metaheurísticas, clasificadas por familias.

Entre estos métodos heurísticos y metaheurísticos, se pueden distinguir los que están basados en algoritmos constructivos, que, partiendo de una solución vacía, van incorporando elementos hasta completar una solución, de los que están basados en búsquedas locales, que parten de una solución inicial y en cada iteración, a través de un movimiento, se sustituye la solución actual por otra de su proximidad. Las peculiaridades del problema que estudia este trabajo dificultan en gran medida los movimientos que caracterizan a los métodos basados en búsqueda en entornos, por lo que se prestará atención a metaheurísticas que se apoyan en algoritmos constructivos. A continuación se presentan dos de las metaheurísticas con estas características, GRASP y colonia de hormigas, destacándose las variantes que serán aplicadas en este trabajo e indicándose algunas aplicaciones a los problemas de rutas y la logística humanitaria que pueden encontrarse en la literatura.

1.3.2. GRASP

El GRASP (del inglés *Greedy Randomized Adaptive Search Procedure*) es una metaheurística de búsqueda adaptativa aleatorizada descrita explícitamente por primera vez en Feo y Resende (1989), aunque el nombre por el que se la conoce actualmente se introdujo algunos años después (Feo y Resende, 1995). Cada iteración del GRASP consta de dos fases, una constructiva, en la que se genera una solución inicial, y otra de mejora, en la que se busca un mínimo en un entorno de la solución inicial a través de un proceso de búsqueda local.

En la fase constructiva las soluciones se construyen de forma iterativa, añadiendo un elemento nuevo en cada iteración siguiendo una estrategia *greedy* aleatorizada. En cada iteración, el siguiente elemento que va a pasar a formar parte de la solución se elige de entre una lista de candidatos ordenada según el beneficio estimado que produciría cada uno. La heurística es adaptativa porque la estimación del beneficio que produce cada elemento se actualiza en cada iteración a medida que se va construyendo la solución, y es aleatorizada porque no se añade necesariamente el mejor de los candidatos en cada momento, sino que se elige de forma aleatoria dando mayor probabilidad a los elementos con mayor beneficio estimado. En la fase de mejora se realiza una búsqueda local sobre cada solución proporcionada por la fase constructiva para intentar alcanzar un óptimo local.

Un inconveniente del método GRASP básico que se acaba de describir es la falta de aprendizaje de las soluciones obtenidas en las iteraciones previas. Entre las posibilidades para incorporar esta información en el proceso constructivo, se han propuesto varias alternativas. Una de las opciones más interesantes es el uso de un conjunto élite de soluciones para guiar la construcción junto con las funciones *greedy*. Para que una nueva solución forme parte del conjunto élite, debe ser buena en comparación con las del conjunto élite y suficientemente diferente del resto de soluciones del conjunto, con el fin de preservar no solo la calidad de las soluciones del conjunto élite, sino también su diversidad. Así, si $g(i)$ es la función greedy asociada a un elemento i candidato a ser incorporado a la solución e $I(i)$ es la función intensidad del candidato, que mide la frecuencia con la que el elemento i aparece en el conjunto élite, la probabilidad de seleccionar el elemento i puede establecerse como

$$p(i) = \frac{K(i)}{\sum_{i'} K(i')}$$

En esta expresión, $K(i) = F(g(i), I(i))$ es una función que combina las funciones *greedy* e intensidad, mientras que el índice i' recorre el conjunto de todos los elementos candidatos a formar parte de la solución en la iteración actual de la fase constructiva. El uso de un conjunto élite en el GRASP se introdujo en Fleurent y Glover (1999) y ha sido empleado posteriormente en Santos *et al.* (2005) y Ribeiro *et al.* (2006).

Una publicación reciente donde se recopilan los principios básicos y las mejoras introducidas en el GRASP, incluidas otras posibilidades para incorporar la memoria en el proceso constructivo, es Festa y Resende (2011), mientras que en Resende y Ribeiro (2003) y Resende y Ribeiro (2010) se presentan revisiones bibliográficas de esta metaheurística. En Martí *et al.* (2014) se compendian, proponen y comparan diversas variantes del GRASP

y de la metaheurística reencadenamiento de trayectorias (*path relinking*) en el contexto de la optimización con múltiples objetivos.

La metaheurística GRASP, frecuentemente hibridada con otras técnicas, ha sido aplicada ampliamente a problemas de rutas de vehículos (véase, por ejemplo, Kontoravdis y Bard (1995); Villegas *et al.* (2011); Nguyen *et al.* (2012); Allahyari *et al.* (2015)), pero apenas se ha utilizado en el contexto de la gestión de desastres, habiéndose encontrado únicamente dos publicaciones donde se afronta un problema de rutas (Huang *et al.*, 2012; Sharif y Salari, 2015). Al margen de los problemas de rutas y distribución de recursos, el GRASP se ha aplicado a la gestión de desastres en un problema de asignación de recursos para reparar una red de caminos rurales (Maya y Sörensen, 2011).

1.3.3. Colonia de hormigas

La optimización de colonia de hormigas, o ACO (del inglés *Ant Colony Optimization*), es una metaheurística inspirada en la observación de las colonias de hormigas y, particularmente, en cómo las hormigas logran encontrar los caminos más cortos entre su hormiguero y las fuentes de alimentos. Durante la búsqueda de comida, las hormigas inicialmente exploran los alrededores del hormiguero de forma aleatoria y, según se mueven, dejan un rastro de feromonas químicas en el suelo que puede ser percibido por otras hormigas. En su búsqueda, las hormigas tienden a elegir más frecuentemente caminos con mayores concentraciones de feromonas. Tan pronto como una hormiga encuentra una fuente de alimento, evalúa la cantidad y la calidad de la comida, y transporta alguna cantidad de la comida encontrada de vuelta al hormiguero. Durante el viaje de regreso, la cantidad de feromona que una hormiga deja sobre el terreno puede depender de la cantidad y la calidad de la comida. Los rastros de feromonas guiarán a las hormigas que posteriormente busquen comida hacia la fuente de alimento ya encontrada, y a su vez las hormigas depositarán feromonas que harán más intenso el rastro en ese camino.

El primer algoritmo basado en colonias de hormigas fue establecido en Dorigo *et al.* (1996) después de algunos trabajos previos. Este algoritmo recibió el nombre de sistema de hormigas o AS (del inglés *Ant System*), y fue desarrollado para resolver el problema del viajante o TSP (del inglés *Traveling Salesman Problem*). Se trata de un algoritmo iterativo en el que, en cada iteración, una hormiga construye una solución factible del problema. Para decidir el elemento a incorporar a la solución que se está construyendo, se tienen en cuenta tanto el rastro de feromonas correspondiente a ese elemento, $\tau(i)$, como la visibilidad que ofrece dicho elemento, $\eta(i)$, que se basa en la mejora aparente que supondría incorporarlo a la solución. Si i' recorre el conjunto de los elementos candidatos a ser incorporados a la solución, la probabilidad de seleccionar el elemento i se expresa como

$$p(i) = \frac{[\tau(i)]^\alpha [\eta(i)]^\beta}{\sum_{i'} [\tau(i')]^\alpha [\eta(i')]^\beta}$$

Los parámetros α y β controlan la influencia de las feromonas y la visibilidad en la selección de elementos. Una vez finalizada la construcción de una solución, el rastro de

feromonas de cada elemento i se actualiza mediante la fórmula

$$\tau(i) = (1 - \rho)\tau(i) + \Delta\tau(i)$$

El término ρ es el factor de evaporación, que propicia que el rastro de feromonas se vaya perdiendo con el paso de las iteraciones, mientras que $\Delta\tau(i)$ permite aumentar el rastro de las feromonas en los elementos que hayan intervenido en la solución obtenida, de forma que el aumento sea mayor cuanto mejor sea la solución. Las soluciones pueden ser obtenidas una a una o en grupos de varias soluciones a la vez, donde cada solución es construida por una hormiga de la colonia, actualizándose los rastros de feromonas solo cuando todas las hormigas de la colonia han completado sus soluciones.

En el algoritmo sistema de colonia de hormigas o ACS (del inglés *Ant Colony System*), desarrollado en Dorigo y Gambardella (1997), se introdujeron algunas novedades que se comentan a continuación. Cada hormiga de la colonia construye una solución y se realizan dos tipos de actualizaciones de feromonas. La actualización local se efectúa cada vez que una hormiga hace incorporar un elemento a la solución que está construyendo, y modifica únicamente el rastro de feromonas del elemento incorporado, disminuyéndolo con el fin de diversificar la construcción por parte de todas las hormigas de la colonia. Si i es el último elemento incorporado a la solución, la actualización local se realizaría de la siguiente forma:

$$\tau(i) = (1 - \phi)\tau(i) + \phi\tau_0(i)$$

El término ϕ es el factor de evaporación local y $\tau_0(i)$ es una constante positiva de valor pequeño, habitualmente el valor inicial del rastro de feromonas asociado al elemento i. La actualización global se efectúa una vez que todas las hormigas de la colonia han completado sus soluciones, de forma similar a la actualización en el sistema de hormigas, pero solo a través de la mejor solución obtenida por las hormigas de la colonia en la iteración actual o de la mejor solución obtenida desde el comienzo del algoritmo. La selección del elemento a incorporar a la solución se realiza del siguiente modo: con probabilidad q_0, se elige directamente el elemento que maximiza $[\tau(i)]^\alpha[\eta(i)]^\beta$; y con probabilidad $1 - q_0$, se elige aleatoriamente según las probabilidades $p(i)$, como en el algoritmo sistema de hormigas.

Para obtener información detallada acerca de las múltiples variantes de la metaheurística colonia de hormigas pueden consultarse Blum (2005), Dorigo y Blum (2005), Zhao *et al.* (2010) o Chandra Mohan y Baskaran (2012).

Los algoritmos de colonias de hormigas se adaptan de forma natural a los problemas de rutas, y han sido ampliamente aplicados en este campo desde los comienzos de la metaheurística (Bullnheimer *et al.*, 1999), incluyendo variantes donde existen distintos tipos de hormigas con funciones específicas (Gambardella *et al.*, 1999). En Rizzoli *et al.* (2007) se revisan algunas de las técnicas aplicadas en este tipo de problemas. Algunas publicaciones más recientes son Yu *et al.* (2009), Lee *et al.* (2010) y Yu *et al.* (2011). En el ámbito de la gestión de desastres, el único trabajo destacado ha sido Yi y Kumar (2007), donde se usa un conjunto élite con las mejores soluciones obtenidas para guiar la construcción, en tanto que en Zhang *et al.* (2012b) se aplica la optimización de colonia

de hormigas para resolver un problema de transporte de emergencias, afín a la gestión de desastres.

Capítulo 2

Planteamiento del problema

El problema que se afronta en esta tesis es una generalización del problema de reparto de ayuda humanitaria que se estudia en Ortuño *et al.* (2011) y Vitoriano *et al.* (2011). Dentro del ciclo de gestión de desastres, estaría incluido en la fase de respuesta, concretamente en lo que se conoce en la literatura como *last mile distribution*, donde se desea hacer llegar la ayuda humanitaria a los afectados por un desastre desde una serie de depósitos situados cerca de las poblaciones afectadas. La mayoría de las veces, un reparto de este tipo ha de hacerse en una situación de urgencia y, con bastante frecuencia, se ha de afrontar con cierta incertidumbre acerca de la infraestructura de transporte y bajo riesgo de ataques para saquear la mercancía que se transporta. Ante una situación con estas características, puede ser aconsejable que los vehículos viajen juntos formando convoyes, permitiendo además que los convoyes puedan ser escoltados. El problema pretende contemplar estas circunstancias y, para ello, se adopta como hipótesis que todos los vehículos que han de transitar por una determinada vía lo han de hacer juntos.

Si bien el coste suele ser el criterio de optimización habitual en los problemas de logística, la naturaleza humanitaria del problema que se estudia conlleva otros criterios a considerar. Por supuesto, el tiempo empleado en realizar la misión es relevante por el carácter urgente del reparto. Por otro lado, es muy importante realizar un reparto equitativo en la medida de lo posible, y eventualmente atender prioritariamente a las poblaciones que más lo necesiten. Además es posible que las vías de comunicación hayan sufrido daños, de forma que haya vías con diferentes probabilidades de estar disponibles. Se ha de tener en cuenta a su vez el peligro de ataques por dichas vías.

En los artículos mencionados anteriormente se proponen métodos exactos de resolución basados en modelos de flujo. En esta tesis se pretende construir un modelo más realista y en el que se pueda tener un control más preciso sobre la estructura de las soluciones. En primer lugar, se tratará a los vehículos individualmente para conocer en qué punto se encuentra cada uno en cualquier momento del reparto. También se permitirá que un vehículo pueda pasar más de una vez por un nodo, lo que en algunas situaciones puede ser imprescindible o, al menos, ciertamente ventajoso. Hay que señalar que en Tirado *et al.* (2014) se propone un modelo de programación lineal entera dinámico para el problema de distribución de ayuda que presenta estas dos últimas características. Sin embargo, dicho

modelo no contempla la hipótesis de que los vehículos deban viajar juntos formando convoyes. Además, las medidas propuestas para los atributos en los modelos de programación matemática están condicionadas por la linealidad. Se pretende ahora medir de forma más precisa los atributos. Estas nuevas características conducen a un modelo demasiado complejo como para ser resuelto de forma exacta en instancias de tamaño medio-grande, por lo que habitualmente se deberá usar algún método heurístico, como los que se describirán en capítulos posteriores. No obstante, como complemento para describir el problema, en este capítulo se incluye un modelo de programación lineal entera, que podrá usarse para resolver instancias pequeñas.

El resto del capítulo se organiza de la siguiente forma: en la Sección 2.1 se describe el problema, definiéndose los parámetros y variables más importantes que lo determinan; en la Sección 2.2 se introducen los atributos del problema y se proponen diversas funciones para medirlos; en la Sección 2.3 se presenta un modelo matemático de programación lineal entera para el problema; por último, en la Sección 2.4 se discute cómo tratar con la intrínseca naturaleza multicriterio del problema.

2.1. Descripción del problema

El problema consiste en decidir cómo repartir una cantidad predeterminada de ayuda humanitaria desde unos depósitos a unos puntos de demanda por medio de vehículos de diferentes tipos a través de una red de transporte. Para realizar el reparto, habitualmente se dispondrá de un límite presupuestario.

Los nodos pueden ser depósitos, nodos de demanda o puntos intermedios. En los depósitos se dispone de una cantidad conocida de material de ayuda disponible para el reparto, mientras que en los nodos de demanda se requiere una cantidad también de material, que habrá sido estimada según el tamaño de la población, las necesidades de sus habitantes, etc. Entre los nodos de demanda puede haber distintas prioridades en función de la gravedad con que el desastre haya afectado a las poblaciones asociadas. Se supondrá que no es posible que un mismo nodo sea un depósito y un nodo de demanda al mismo tiempo. Todos los nodos pueden ser usados para realizar transbordos de material entre vehículos.

Como se ha mencionado en la introducción de este capítulo, en este trabajo se asume que todos los vehículos han de viajar juntos por los arcos. Esta importante hipótesis implica que el grafo a considerar para modelar el problema debe ser un grafo dirigido, aunque las vías de comunicación con frecuencia sean carreteras o caminos de doble sentido. En estos casos cada vía de doble sentido se modela mediante un par de arcos de sentidos contrarios. Cada arco lleva asociadas una serie de características: longitud, velocidad máxima, probabilidad de que se encuentre disponible y probabilidad de sufrir un asalto un vehículo que viaje solo. Se supone que la probabilidad de asalto para un convoy es menor para convoyes formados por más vehículos, y a partir de cierto límite para el tamaño del convoy, que se denominará tamaño disuasorio (el mismo para todos los arcos), la probabilidad de asalto permanece constante. Por tanto, también se ha de considerar la probabilidad de sufrir un

asalto un convoy cuyo tamaño es al menos el tamaño disuasorio.

Los vehículos con los que se debe realizar el reparto pueden ser de distintos tipos. Cada tipo se caracteriza por su velocidad media y su capacidad. Es posible que los vehículos de ciertos tipos no puedan circular por algunos de los arcos, por ejemplo los vehículos más grandes podrían estar imposibilitados para transitar por caminos estrechos, o atravesar túneles angostos, mientras que los vehículos menos potentes podrían tener dificultades para superar fuertes pendientes o atravesar zonas inundadas. También se conocen el coste fijo por unidad de tiempo y el coste por lote y unidad de tiempo que suponen que un vehículo de cada tipo recorra cada arco. Cada vehículo puede estar situado originalmente en cualquier nodo, si bien suele haber más cantidad de vehículos disponibles en los depósitos que en el resto de nodos. Los vehículos pueden repetir nodos en su recorrido, pero el hecho de que deban viajar juntos formando convoyes impide que puedan repetir arcos, lo que llevaría de forma evidente a una situación de infactibilidad. Este hecho conduce a una relación de precedencia entre los arcos: cada arco recorrido por un vehículo debe preceder a los arcos que recorre después dicho vehículo. Cualquiera que sea el modelo propuesto para representar el problema, debe tener en cuenta la obligación de preservar esta relación de orden estricto. Una vez completada su función en el reparto, cada vehículo ha de regresar a su nodo de origen. El regreso ha de hacerse por el camino más económico y ya no es necesario que se formen convoyes debido a que se considera que un vehículo vacío no tiene valor para los eventuales asaltantes.

2.1.1. Datos del problema

A continuación se introducen los datos del problema. Por una parte, se definen los conjuntos e índices que se emplearán a lo largo de esta memoria y, por otra, se presentan los datos propiamente dichos del problema. Salvo algunas excepciones que se irán indicando en su momento, esta es la notación que se usará en todos los modelos propuestos.

Conjuntos e índices

$c \in C$	Clase genérica de vehículos. Dos vehículos pertenecen a la misma clase si comparten el tipo t y el origen i
e, e'	Etapas genéricas para las rutas de los vehículos
$g \in G$	Objetivo genérico
$i, i' \in I$	Nodos genéricos
$ID \subseteq I$	Subconjunto de nodos de demanda
$IO \subseteq I$	Subconjunto de depósitos
$j, j' \in J$	Vehículos genéricos
$k, k' \in K$	Arcos genéricos
$t \in T$	Tipo genérico de vehículos

Datos

b	Presupuesto disponible (u. m.)
cap_t	Capacidad máxima de los vehículos del tipo t (lotes)

$comp_{t,k}$	Parámetro binario que indica si un vehículo del tipo t puede recorrer el arco k (1 sí; 0 no)
$cosf_{t,k}$	Coste por cada km recorrido por un vehículo del tipo t en el arco k (u. m.)
$cosv_{t,k}$	Coste por cada lote y cada km recorrido por un vehículo del tipo t en el arco k (u. m.)
dem_i	Demanda de material en el nodo i (lotes)
$dist_k$	Longitud del arco k (km)
fin_k	Nodo final del arco k
ini_k	Nodo inicial del arco k
$nivp_i$	Nivel de prioridad del nodo i. Es un valor entre 0 y 1 que indica la urgencia de atención en ese nodo (a mayor valor, mayor prioridad)
ori_j	Nodo origen del vehículo j
p_k	Probabilidad de que un vehículo viajando solo sufra un asalto en el arco k
pm_k	Mínima probabilidad de sufrir un asalto en el arco k
qav_i	Cantidad de material disponible en el nodo i (lotes)
$qglobal$	Cantidad total de material que se desea repartir (lotes)
r_k	Probabilidad de que el arco k esté disponible
$tdis$	Tamaño disuasorio: número de vehículos de que debe constar un convoy como mínimo para que la probabilidad de asalto sea mínima
$tipv_j$	Tipo del vehículo j
$vela_k$	Velocidad máxima en el arco k (km/h)
$velv_t$	Velocidad media de un vehículo del tipo t (km/h)

Hay que tener en cuenta que no siempre es fácil obtener algunos de los valores de estos parámetros. Concretamente, las probabilidades de disponibilidad y de sufrir asaltos en las vías presentan dificultades de cara a ser estimadas con precisión. Habitualmente, dicha información se obtiene de forma cualitativa, mediante preguntas del tipo, ¿el desastre ha afectado mucho o poco a la zona en que se encuentra cierta vía?, ¿cuáles son las vías más peligrosas?, ¿qué volumen de tráfico tienen en condiciones normales?, etc. De este modo se puede realizar una estimación inicial para planificar el primer reparto. Posteriormente, como el modelo está diseñado para usarse repetidas veces en misiones sucesivas, la información que se obtiene en el transcurso de cada misión se utiliza para actualizar los valores de los parámetros de cara a las siguientes misiones.

Además de los datos básicos del problema, se consideran los siguientes parámetros derivados que han de calcularse antes de aplicar cualquier modelo:

Parámetros derivados

$creg_{j,i}$	Coste de regreso del vehículo j si finaliza el reparto en el nodo i (u. m.)
$treg_{j,i}$	Tiempo de regreso del vehículo j si finaliza el reparto en el nodo i (u. m.)

Puesto que los costes de transporte son todos positivos, ambos parámetros se calculan aplicando el algoritmo de Dijkstra para obtener los caminos de regreso más económicos. Esto se hará mediante el procedimiento CAMINO MÍNIMO descrito en el Algoritmo Cons-

tructivo, en el Capítulo 3.

2.1.2. Variables del problema

El conjunto de variables descrito a continuación determina completamente una solución. A partir de sus valores puede calcularse el valor de cualquier otra variable que pueda ser necesaria en un momento dado.

Variables de decisión

$load_j$	Secuencia de cantidades transportadas por el vehículo j hasta que termina su reparto
$secv_j$	Secuencia de arcos que recorre el vehículo j hasta que termina su reparto
$tmin_k$	Instante en el que el convoy que recorre el arco k finaliza su trayecto

Siguiendo la notación anterior, $secv_{j,e}$ y $load_{j,e}$ serían, respectivamente, el arco y la cantidad de material transportada por el vehículo j en la etapa e de su ruta.

Se definen a continuación algunas variables auxiliares importantes que serán de ayuda para proporcionar información adicional sobre la solución y establecer las medidas de los atributos del problema.

Variables auxiliares

$conv_k$	Conjunto de vehículos (convoy) que recorre el arco k
$lonv_j$	Número de arcos recorridos por el vehículo j
$ltar_k$	Cantidad total de material que transporta el convoy correspondiente al arco k
sf_i	Cantidad de material existente en el nodo i al terminar el reparto
$tamc_k$	Tamaño del convoy que recorre el arco k

2.2. Atributos

En logística humanitaria con mucha frecuencia se deben considerar diferentes atributos a la hora de tomar decisiones. En Campbell *et al.* (2008) se proponen objetivos basados en el tiempo y en la equidad para problemas de rutas de reparto de ayuda, en Huang *et al.* (2012) se subraya la dificultad de tratar atributos de naturalezas distintas en este tipo de problemas de rutas, y en Holguín-Veras *et al.* (2013) se incide en la necesidad de incorporar el sufrimiento de los afectados a la función objetivo en el contexto de la logística humanitaria. En la Sección 1.2 se ha citado una buena cantidad de trabajos que abordan problemas multicriterio en el ámbito de la logística humanitaria.

En esta sección se introducen los atributos relevantes en el problema, así como una serie de medidas cuantitativas de los mismos. Aunque en algunos casos se propondrán varias medidas, se indicará en cada caso cuáles se usarán en los modelos descritos posteriormente,

indicando los motivos de las elecciones. Particularmente, la medida principal propuesta para cada atributo será la empleada en los algoritmos Constructivo y de Mejora, base de las metaheurísticas que se describirán en capítulos posteriores.

2.2.1. Tiempo

La duración del reparto es uno de los atributos que suelen tener mayor importancia. La gravedad de la situación implica que se deba actuar urgentemente, intentando hacer llegar la ayuda requerida a las poblaciones necesitadas lo antes posible.

El tiempo de regreso no se considera especialmente relevante, por lo que no se incluye en la medida principal, que consiste, simplemente, en tomar el instante en el que finaliza su trayecto el convoy que llega a su destino más tarde.

$$f_1 = \max_k \{tmin_k\}$$

Esta medida para el tiempo es la que se usará en todos los modelos propuestos, incluido el modelo de programación lineal. Como alternativa, se propone otra medida en la que sí se considera el tiempo de regreso. Se ha de tener en cuenta que cada vehículo inicia su regreso nada más completar su ruta, sin esperar a que todos los vehículos completen el reparto.

$$f_{1,1} = \max_j \{tmin_k + treg_{j,i}\} \quad \text{siendo } k = secv_{j,lonv_j}, i = fin_k$$

2.2.2. Coste

El coste suele ser el atributo a optimizar en la gran mayoría de los problemas de rutas y logística empresarial. En logística humanitaria, aunque no sea el factor principal, ha de considerarse como complemento a otros atributos usualmente más importantes. Incluso en el caso de que no se considere como criterio de optimización, normalmente se dispondrá de un presupuesto limitado que no se podrá exceder.

En primer lugar se propone una medida alternativa en la que no se tienen en cuenta los costes de regreso, solo los costes de transporte derivados de las rutas de reparto de los vehículos. Esta medida se usaría en el caso de que los vehículos no tuviesen que regresar a sus respectivos orígenes una vez completada su misión en el reparto.

$$f_{2,1} = \sum_k \sum_{j \in conv_k} \left(cosf_{tipv_j,k} + cosv_{tipv_j,k} \cdot load_{j,e}\right) dist_k \quad \text{siendo } k = secv_{j,e}$$

En la medida principal para el coste total del reparto, que será la empleada en esta tesis, se han de añadir los costes de regreso de los vehículos desde sus destinos a sus orígenes.

$$f_2 = f_{2,1} + \sum_j creg_{j,i} \quad \text{siendo } i = fin_k \text{ y } k = secv_{j,lonv_j}$$

2.2.3. Equidad

La equidad en el reparto es claramente un atributo crucial en problemas de reparto en logística humanitaria. La escasez de recursos impide normalmente que puedan satisfacerse en su totalidad las necesidades de las personas afectadas. En este contexto, es muy importante que la ayuda disponible se reparta de la manera más justa posible entre las poblaciones necesitadas.

En primer lugar se define una medida fácilmente linealizable, que será usada en el modelo de programación matemática. Consiste en tomar la máxima proporción de demanda no satisfecha entre todos los nodos de demanda.

$$f_{3,1} = \max_{i \in ID} \left\{ 1 - \frac{sf_i}{dem_i} \right\}$$

Sin embargo, esta medida presenta un inconveniente importante: se deja libertad para repartir la ayuda entre los nodos de demanda donde no se presenta la mayor proporción de demanda no satisfecha, lo que podría llevar a repartos poco equitativos entre dichos nodos. Esto sería especialmente grave en el caso de que no puedan alcanzarse todos los nodos de demanda. Para paliar este defecto, se debe proponer alguna medida que tenga en cuenta todas las proporciones de demanda no satisfechas. Para ello se define una nueva variable auxiliar, $pmds$, que representa la proporción media de demanda no satisfecha.

$$pmds = \frac{1}{|ID|} \sum_{i \in ID} \frac{sf_i}{dem_i}$$

Como medida principal de la equidad, se propone la desviación típica de las proporciones de demandas no satisfechas.

$$f_3 = \sqrt{\frac{1}{|ID|} \sum_{i \in ID} \left(\frac{sf_i}{dem_i} - pmds \right)^2}$$

Para terminar se propone otra medida lineal, que podría ser útil en determinadas situaciones: el número de nodos de demanda que no reciben nada de material.

$$f_{3,2} = |i \in ID / sf_i = 0|$$

2.2.4. Prioridad

En ocasiones es posible que unas poblaciones requieran una atención más prioritaria que otras, al margen de la cantidad de ayuda que se necesite. Esto puede ser debido a que el desastre haya afectado de forma desigual a las poblaciones, o a que los habitantes de algunas de ellas sean más vulnerables.

Para la prioridad se presenta una sola medida: la suma de proporciones de demandas no satisfechas ponderadas con los niveles de prioridad respectivos.

$$f_4 = \sum_{i \in ID} nivp_i \left(1 - \frac{sf_i}{dem_i} \right)$$

Los atributos equidad y prioridad habitualmente presentan un gran conflicto: conseguir un reparto equitativo probablemente implica que no se pueda satisfacer la demanda de los nodos de mayor prioridad, y viceversa, para aumentar el valor de la medida de la prioridad se ha de satisfacer en gran medida la demanda de los nodos con más prioridad, lo que produce un reparto poco equitativo. En todo caso, en la práctica no suelen considerarse juntos al resolver un problema: o bien se requiere un reparto equitativo, o bien que se satisfaga la demanda en mayor medida en los nodos prioritarios, pero no ambas cosas a la vez. Este hecho es importante y repercutirá en algún aspecto del desarrollo del Algoritmo de Mejora, como se verá en el Capítulo 4.

2.2.5. Seguridad

En un contexto de escasez y necesidad, como el que habitualmente se genera tras ocurrir un desastre, es posible que los convoyes sufran ataques para saquear la ayuda transportada. Por lo tanto, es conveniente incluir un atributo que tenga en cuenta, tanto el grado de peligro de las distintas zonas en las que se debe actuar, como los tamaños de los convoyes que se formen, ya que se asume que atacar a convoyes con más vehículos es más difícil que atacar a convoyes con pocos vehículos.

Antes de proponer las medidas para la seguridad, se definirá una función que evalúe la probabilidad de sufrir un asalto en un arco. Esta función depende de la probabilidad de que un vehículo que circule solo sufra un asalto, p, la probabilidad mínima de asalto, pm, y el tamaño del convoy, tc. En la expresión también aparece el tamaño disuasorio, $tdis$, que es un dato del problema. Se propone una función principal y dos alternativas.

- Función principal (cuadrática).

$$FP(p, pm, tc) = \begin{cases} 0 & \text{si } tc = 0 \\ p + \dfrac{p - pm}{(tdis-1)^2}\left(tc^2 - 2\,tdis \cdot tc + 2\,tdis - 1\right) & \text{si } 1 \leq tc \leq tdis \\ pm & \text{si } tc > tdis \end{cases}$$

- Función alternativa (lineal).

$$FP(p, pm, tc) = \begin{cases} 0 & \text{si } tc = 0 \\ p - \dfrac{(tc-1)(p-pm)}{tdis-1} & \text{si } 1 \leq tc \leq tdis \\ pm & \text{si } tc > tdis \end{cases}$$

- Función alternativa (exponencial).

$$FP(p, pm, tc) = \begin{cases} 0 & \text{si } tc = 0 \\ p \cdot e^{\frac{(tc-1)(\ln pm - \ln p)}{tdis-1}} & \text{si } 1 \leq tc \leq tdis \\ pm & \text{si } tc > tdis \end{cases}$$

Las funciones anteriores han sido diseñadas y ajustadas con el propósito de que verifiquen las siguientes propiedades:

- La probabilidad de sufrir un asalto es p cuando el convoy lo forma un único vehículo ($tc = 1$).
- La probabilidad de sufrir un asalto es pm cuando el tamaño del convoy es mayor o igual que el tamaño disuasorio ($tc = tdis$).
- Es una función decreciente y convexa respecto del tamaño del convoy en el intervalo $[1, tdis]$.

Se ha optado por usar la función cuadrática por entender que, debido a su suave convexidad, modela de forma más adecuada la probabilidad de sufrir un asalto.

Una vez definida la función anterior, se propondrán varias medidas para la seguridad.

En primer lugar, la probabilidad de sufrir algún asalto, definida como el complemento a uno de la probabilidad de que ningún convoy sea asaltado.

$$f_{5,1} = 1 - \prod_{k/conv_k \neq \varnothing} (1 - FP(p_k, pm_k, tamc_k))$$

Esta medida es relativamente sencilla, pero presenta dos incovenientes. En primer lugar, en instancias grandes y situaciones de gran riesgo, es muy posible que su valor se aproxime siempre a su máximo, 1, independientemente de las rutas empleadas. En segundo lugar, no tiene en cuenta la cantidad de ayuda transportada por los arcos, cuando parece lógico pensar que un ataque es más grave cuánto más material transporte el convoy asaltado. De hecho, en este trabajo, se ha adoptado la hipótesis de que un convoy que viaja sin material no interesa a los posibles asaltantes, hipótesis incompatible con esta medida.

La siguiente propuesta, también alternativa, mide la máxima pérdida esperada entre todos los arcos.

$$f_{5,2} = \max_{k/conv_k \neq \varnothing} \{FP(p_k, pm_k, tamc_k) ltar_k\}$$

Como medida principal, se presenta la suma de pérdidas esperadas de material, que respecto a la medida anterior, tiene la ventaja de tener en cuenta todos los arcos, no solo el que presente mayor gravedad esperada.

$$f_5 = \sum_{k/conv_k \neq \varnothing} FP(p_k, pm_k, tamc_k) ltar_k$$

Como ya se ha indicado, una buena medida para la seguridad debe tener en cuenta los siguientes dos factores:

- La probabilidad de asalto en un arco es decreciente en el tamaño del convoy.
- Un asalto es más grave cuanto más material se transporte por el arco.

Sin embargo, no es posible definir una medida lineal razonable que tenga en cuenta ambos factores. Se ha decidido dar más importancia al primero, llegando a la siguiente expresión, que representa el logaritmo de la probabilidad de sufrir algún asalto.

$$f_{5,3} = \sum_k \left(2 - \frac{tamc_k}{|J|}\right) \ln pm_k$$

Para poder usar esta medida, se ha de suponer que la probabilidad mínima de asalto en un arco se alcanza cuando el convoy está formado por todos los vehículos, es decir, $tdis = |J|$.

2.2.6. Fiabilidad

Con frecuencia, los desastres dañan en mayor o menor medida la infraestructura de transporte de la zona afectada. Pueden bloquearse tramos de carreteras por desprendimientos, derrumbarse puentes, hundirse túneles, etc., modificándose de este modo la red de transporte habitual. Además, no siempre se conoce con seguridad el estado de las vías en el momento de planificar el reparto, no siendo posible esperar para recabar información en ese sentido. Por tanto, se deberá usar únicamente la información parcial que se disponga para estimar las probabilidades de disponibilidad de los arcos.

La medida más sencilla que puede definirse para la fiabilidad es la probabilidad de que algún arco no esté disponible.

$$f_{6,1} = 1 - \prod_{k/conv_k \neq \varnothing} r_k$$

Aunque no es lineal, esta medida puede linealizarse fácilmente tomando logaritmos. Sin embargo, no tiene en cuenta el tamaño del convoy ni la carga que se transporta por cada arco.

La siguiente medida, que se usará en el modelo lineal, asume que la gravedad de encontrarse un arco no disponible durante su misión depende del tamaño del convoy que recorre el arco, definiéndose como la suma de las probabilidades de que los arcos no estén disponibles multiplicadas por los tamaños de los convoyes.

$$f_{6,2} = \sum_k (1 - r_k) tamc_k$$

Para establecer medidas más ajustadas a la realidad, ha de tenerse en cuenta que la repercusión de que un vehículo se encuentre un arco no disponible vendrá dada por lo que ese imprevisto le impide realizar de su ruta. En este sentido, se define la importancia de un vehículo j sobre un arco k de su ruta como la suma de las descargas que hará a partir de ese arco.

$$imv_{j,k} = \sum_{e \leq e' < lonv_j} \text{máx}\{0, load_{j,e'} - load_{j,e'+1}\} + load_{j,lonv_j} \quad \text{siendo } secv_{j,e} = k$$

Se define ahora la importancia del convoy que transita por el arco k como la suma de las importancias de los vehículos que forman el convoy sobre ese arco. Cuanta más importancia tenga el convoy asociado a un arco, más grave resultará encontrarse el arco no disponible.

$$imc_k = \sum_{j \in conv_k} imv_{j,k}$$

Así, de forma análoga a lo que se ha realizado para el atributo seguridad, se definen una nueva medida alternativa como la máxima gravedad esperada entre todos los arcos y la medida principal como la suma de las gravedades esperadas.

$$f_{6,3} = \max_{k/conv_k \neq \varnothing} \{(1 - r_k) imc_k\}$$

$$f_6 = \sum_{k/conv_k \neq \varnothing} (1 - r_k) imc_k$$

2.3. Modelo de programación matemática

En esta sección se presenta un modelo matemático de programación lineal entera multiobjetivo para el problema planteado. Se aporta este modelo principalmente por su interés teórico, ya que su complejidad, especialmente el gran número de variables enteras y binarias, hace inviable su aplicación a problemas reales, quedando restringido su uso a pequeños ejemplos.

El modelo impone todas las hipótesis del problema, si bien, como se ha indicado en la sección anterior, algunas variables objetivo son simplificaciones de las medidas principales propuestas para los atributos, imposibles de usar aquí por ser no lineales.

2.3.1. Conjuntos, parámetros y variables

Se ha mantenido la notación para los índices y parámetros definidos en la Sección 2.1 de este capítulo. No obstante, se declaran de nuevo todos los índices por presentar ciertos matices diferentes. En este modelo, para dar mayor claridad a las expresiones, pero sin merma en la generalidad del problema, los vehículos no se han clasificado por tipos, sino que se consideran individualmente, por lo que algunos de los parámetros se han debido ajustar a esta situación. Se han introducido varias variables nuevas necesarias para imponer las restricciones del problema a través de ecuaciones e inecuaciones lineales. En cuanto a las variables definidas previamente, se ha cambiado la nomenclatura para que se ajuste a la notación habitual de este tipo de modelos.

Conjuntos e índices

$i, i' \in I$	Nodos genéricos
$j \in J$	Vehículo genérico
$k, k', k'' \in K$	Arcos genéricos
$e, e' \in E = \{1, 2, \ldots, f = \lvert K \rvert\}$	Etapas genéricas de un vehículo
$n(i), n'(i) \in N(i) = \{0, 1, \ldots, g(i) = gra_i\}$	Eventos genéricos de un nodo

Parámetros adicionales

$comp_{j,k}$	Parámetro binario que indica si el vehículo j puede recorrer el arco k (1 sí; 0 no)
$cosf_{j,k}$	Coste por cada km recorrido por el vehículo j en el arco k
$cosv_{j,k}$	Coste por cada lote de material y cada km recorrido por el vehículo j en el arco k
gra_i	Número de arcos que llegan o salen del nodo i (grado)
M	Cota superior para el tiempo total del reparto. Puede tomarse $M = b_1$, definida en la siguiente sección
$velv_j$	Velocidad media del vehículo j

Variables de decisión

D_k	Duración del trayecto del convoy que circula por el arco k
$L_{j,k}$	Cantidad de material que el vehículo j transporta por el arco k
$P_{k,k'}$	Indicador de que el arco k precede al arco k' (1 sí; 0 no)
$X_{j,k,e}$	Indicador de que el vehículo j recorre el arco k en la etapa e (1 sí; 0 no)
$S_{i,n(i)}$	Cantidad de material que queda en el nodo i tras el evento $n(i)$
$T_{i,n(i)}$	Instante en el que se produce el evento $n(i)$ del nodo i
$Z_{i,k,n(i)}$	Indicador de que el evento $n(i)$ del nodo i corresponde al arco k (1 sí; 0 no)

Variables objetivo

$COST$	Coste total del reparto, incluyendo el regreso de los vehículos a sus orígenes
$EQUI$	Índice de equidad, máxima proporción de demanda no satisfecha
$PRIO$	Índice de prioridad, suma de las proporciones de demandas no satisfechas ponderadas con los niveles de prioridad
$RELI$	Índice de fiabilidad, suma de los tamaños de los convoyes multiplicados por las probabilidades de que los arcos no estén disponibles
$SECU$	Índice de seguridad, logaritmo de la probabilidad de sufrir algún asalto
$TIME$	Duración total del reparto

2.3.2. Restricciones

Las restricciones han sido agrupadas en distintos tipos para facilitar la comprensión del modelo. Aunque algunas de las restricciones no son imprescindibles, se incluyen con el propósito de reforzar el modelo.

Restricciones relativas a las asociaciones vehículos-arcos

- Cada vehículo sólo puede recorrer los arcos permitidos

$$X_{j,k,e} \leq comp_{j,k} \quad \forall j,\ \forall k,\ \forall e \tag{2.1}$$

- En la etapa 1 cada vehículo sólo puede recorrer arcos que salen de su nodo inicial

$$X_{j,k,1} = 0 \quad \forall j,\ \forall k \ /\ ini_k \neq ori_j \tag{2.2}$$

- En cada etapa cada vehículo recorre a lo sumo un arco

$$\sum_{k} X_{j,k,e} \leq 1 \quad \forall j,\ \forall e \tag{2.3}$$

- Cada vehículo recorre cada arco a lo sumo una vez. No es imprescindible por incluir el modelo la restricción (2.38) o las restricciones (2.37) y (2.39)

$$\sum_{e} X_{j,k,e} \leq 1 \quad \forall j,\ \forall k \tag{2.4}$$

- Si en una etapa no se realiza ningún trayecto, en la siguiente tampoco

$$\sum_{k} X_{j,k,e+1} \leq \sum_{k} X_{j,k,e} \quad \forall j,\ \forall e < f \tag{2.5}$$

- En cada etapa cada vehículo sólo puede recorrer arcos que salgan del nodo al que haya llegado en la etapa anterior

$$X_{j,k',e+1} \leq 1 - X_{j,k,e} \quad \forall j,\ \forall e < f,\ \forall k, k' \ /\ ini_{k'} \neq fin_k \tag{2.6}$$

Restricciones relativas a los eventos en los nodos

- El evento 0 es artificial. No corresponde a ningún arco

$$Z_{i,k,0} = 0 \quad \forall i,\ \forall k \tag{2.7}$$

- Cada evento de un nodo corresponde a lo sumo a un arco

$$\sum_{k} Z_{i,k,n(i)} \leq 1 \quad \forall i,\ \forall n(i) > 0 \tag{2.8}$$

- A cada arco no le puede corresponder más de un evento de un mismo nodo

$$\sum_{n(i)>0} Z_{i,k,n(i)} \leq 1 \quad \forall i,\ \forall k \tag{2.9}$$

- Si un evento no corresponde a ningún arco, el siguiente evento tampoco

$$\sum_{k} Z_{i,k,n(i)+1} \leq \sum_{k} Z_{i,k,n(i)} \quad \forall i,\ \forall n(i)\ /\ 0 < n(i) < g(i) \tag{2.10}$$

- Cada evento de un nodo no puede corresponder a arcos no incidentes en el nodo

$$Z_{i,k,n(i)} = 0 \quad \forall i,\ \forall n(i) > 0,\ \forall k\ /\ ini_k \neq i, fin_k \neq i \tag{2.11}$$

- A un arco por el que no circulen vehículos no le puede corresponder ningún evento

$$\sum_{j,e} X_{j,k,e} \geq Z_{i,k,n(i)} \quad \forall i,\ \forall n(i) > 0,\ \forall k\ /\ ini_k = i \text{ ó } fin_k = i \tag{2.12}$$

- A un arco por el que circulen vehículos le debe corresponder algún evento de los nodos en los que incide

$$\sum_{j,e} X_{j,k,e} \leq |J| \sum_{n(i)>0} Z_{i,k,n(i)} \quad \forall i,\ \forall k \ /\ ini_k = i \text{ ó } fin_k = i \tag{2.13}$$

Restricciones relativas a las cantidades de material transportadas por los vehículos

- La carga de un vehículo por un arco no puede exceder la capacidad del vehículo. Si el vehículo no recorre el arco, la carga es 0.

$$L_{j,k} \leq cap_j \sum_{e} X_{j,k,e} \quad \forall j,\ \forall k \tag{2.14}$$

- En su último trayecto cada vehículo ha de transportar material. (Si no se tiene en cuenta el coste como criterio de optimización, es posible que aparezcan movimientos absurdos de algunos vehículos. Si un vehículo en su último trayecto no transporta ayuda, puede eliminarse dicho trayecto por no aportar nada. Un movimiento de un vehículo vacío por un arco solo tiene sentido si más adelante ha de transportar material en algún arco posterior de su ruta)

$$L_{j,k} \geq X_{j,k,e} - \sum_{k'} X_{j,k',e+1} \quad \forall j,\ \forall k,\ \forall e < f \tag{2.15}$$

$$L_{j,k} \geq X_{j,k,f} \quad \forall j,\ \forall k \tag{2.16}$$

Restricciones relativas a las cantidades de material en los nodos

- Cálculo de la cantidad inicial de material en cada nodo

$$S_{i,0} = qav_i \quad \forall i \tag{2.17}$$

- Si un evento de un nodo corresponde a un arco saliente, el *stock* disminuye en la cantidad de material que transporte el convoy por el arco

$$\begin{gathered} S_{i,n(i)+1} \leq S_{i,n(i)} - \textstyle\sum_j L_{j,k} + 2qglobal(1 - Z_{i,k,n(i)+1}) \\ \forall i,\ \forall n(i) < g(i),\ \forall k \ /\ ini_k = i \end{gathered} \tag{2.18}$$

$$\begin{gathered} S_{i,n(i)+1} \geq S_{i,n(i)} - \textstyle\sum_j L_{j,k} - qglobal(1 - Z_{i,k,n(i)+1}) \\ \forall i,\ \forall n(i) < g(i),\ \forall k \ /\ ini_k = i \end{gathered} \tag{2.19}$$

- Si un evento de un nodo corresponde a un arco entrante, el *stock* aumenta en la cantidad de material que transporte el convoy por el arco

$$\begin{gathered} S_{i,n(i)+1} \leq S_{i,n(i)} + \textstyle\sum_j L_{j,k} + qglobal(1 - Z_{i,k,n(i)+1}) \\ \forall i,\ \forall n(i) < g(i),\ \forall k \ /\ fin_k = i \end{gathered} \tag{2.20}$$

$$\begin{gathered} S_{i,n(i)+1} \geq S_{i,n(i)} + \textstyle\sum_j L_{j,k} - 2qglobal(1 - Z_{i,k,n(i)+1}) \\ \forall i,\ \forall n(i) < g(i),\ \forall k \ /\ fin_k = i \end{gathered} \tag{2.21}$$

- Si un evento de un nodo no corresponde a ningún arco, el *stock* se mantiene

$$S_{i,n(i)+1} \leq S_{i,n(i)} + qglobal \sum_k Z_{i,k,n(i)+1} \quad \forall i,\ \forall n(i) < g(i) \tag{2.22}$$

$$S_{i,n(i)+1} \geq S_{i,n(i)} - qglobal \sum_k Z_{i,k,n(i)+1} \quad \forall i,\ \forall n(i) < g(i) \tag{2.23}$$

- Se debe repartir la cantidad planeada

$$\sum_{i/dem_i>0} S_{i,g(i)} = qglobal \tag{2.24}$$

- En cada nodo que no sea depósito el *stock* final no puede ser mayor que la demanda

$$S_{i,g(i)} \leq dem_i \quad \forall i \ /\ qav_i = 0 \tag{2.25}$$

Restricciones relativas a la duración de los trayectos

- La duración de un trayecto por un arco es 0 si no circula ningún vehículo por el arco. No es imprescindible

$$D_k \leq \frac{dist_k}{\min\{vela_k, \min_j\{velv_j\}\}} \sum_{j,e} X_{j,k,e} \quad \forall k \tag{2.26}$$

- La duración de un trayecto por un arco viene dada por el mínimo entre la velocidad del arco y la velocidad del vehículo más lento

$$D_k \geq \frac{dist_k}{vela_k} X_{j,k,e} \quad \forall k,\ \forall j,\ \forall e \tag{2.27}$$

$$D_k \geq \frac{dist_k}{velv_j} X_{j,k,e} \quad \forall k,\ \forall j,\ \forall e \tag{2.28}$$

Restricciones relativas a los instantes en los que se producen eventos en los nodos

- Cálculo del instante inicial en cada nodo

$$T_{i,0} = 0 \quad \forall i \tag{2.29}$$

- La sucesión de instantes en cada nodo es no decreciente

$$T_{i,n(i)+1} \geq T_{i,n(i)} \quad \forall i,\ \forall n(i) < g(i) \tag{2.30}$$

- Si un evento de un nodo corresponde a un arco saliente, el tiempo no transcurre desde el evento anterior. No es imprescindible si se minimiza el tiempo

$$T_{i,n(i)+1} \leq T_{i,n(i)} + M(1 - Z_{i,k,n(i)+1}) \quad \forall i,\ \forall n(i) < g(i),\ \forall k \ /\ ini_k = i \tag{2.31}$$

- Si un evento de un nodo no corresponde a ningún arco, el tiempo no transcurre desde el evento anterior

$$T_{i,n(i)+1} \leq T_{i,n(i)} + M \sum_{k} Z_{i,k,n(i)+1} \quad \forall i,\ \forall n(i) < g(i) \tag{2.32}$$

- El instante en el que se produce un evento correspondiente a un arco entrante en un nodo coincide con el instante en el que partió el convoy hacia el nodo más la duración del trayecto. La primera restricción no es imprescindible si se minimiza el tiempo

$$\begin{gathered} T_{i',n(i')} \leq T_{i,n(i)} + D_k + M(2 - Z_{i,k,n(i)} - Z_{i',k,n(i')}) \\ \forall i, i', \forall n(i) > 0, n(i') > 0,\ \forall k \ /\ ini_k = i, fin_k = i' \end{gathered} \tag{2.33}$$

$$\begin{gathered} T_{i',n(i')} \geq T_{i,n(i)} + D_k - M(2 - Z_{i,k,n(i)} - Z_{i',k,n(i')}) \\ \forall i, i', \forall n(i) > 0, n(i') > 0,\ \forall k \ /\ ini_k = i, fin_k = i' \end{gathered} \tag{2.34}$$

Restricciones relativas a las precedencias entre los arcos

- Un arco no puede precederse a sí mismo, la relación de precedencia es antirreflexiva

$$P_{k,k} = 0 \quad \forall k \tag{2.35}$$

- La relación de precedencia es asimétrica

$$P_{k,k'} + P_{k',k} \leq 1 \quad \forall k, k' \tag{2.36}$$

- La relación de precedencia es transitiva

$$P_{k,k''} \geq P_{k,k'} + P_{k',k''} - 1 \quad \forall k, k', k'' \tag{2.37}$$

- Cada arco recorrido por un vehículo precede a los arcos que este recorre después

$$P_{k,k'} \geq X_{j,k,e} + X_{j,k',e'} - 1 \quad \forall j,\ \forall k, k',\ \forall e, e' \ /\ e < e' \tag{2.38}$$

- Alternativa a (2.38). Cada arco recorrido por un vehículo precede al que recorre inmediatamente después

$$P_{k,k'} \geq X_{j,k,e} + X_{j,k',e+1} - 1 \quad \forall j,\ \forall k, k',\ \forall e < f \tag{2.39}$$

- Cada arco entrante a un nodo precede a los arcos salientes que corresponden a eventos posteriores de ese nodo

$$\begin{gathered} P_{k,k'} \geq Z_{i,k,n(i)} + Z_{i,k',n'(i)} - 1 \\ \forall i,\ \forall n(i), n'(i) \ /\ 0 < n(i) < n'(i),\ \forall k, k' \ /\ fin_k = i, ini_{k'} = i \end{gathered} \tag{2.40}$$

- Cada arco entrante a un nodo no puede ser precedido por los arcos entrantes que corresponden a eventos posteriores de ese nodo

$$\begin{gathered} P_{k',k} \leq 2 - Z_{i,k,n(i)} - Z_{i,k',n'(i)} \\ \forall i,\ \forall n(i), n'(i) \ /\ 0 < n(i) < n'(i),\ \forall k, k' \ /\ fin_k = i, fin_{k'} = i \end{gathered} \tag{2.41}$$

- Cada arco saliente de un nodo no puede ser precedido por los arcos que corresponden a eventos posteriores de ese nodo

$$\begin{gathered} P_{k',k} \leq 2 - Z_{i,k,n(i)} - Z_{i,k',n'(i)} \\ \forall i,\ \forall n(i), n'(i)\ /\ 0 < n(i) < n'(i),\ \forall k, k'\ /\ ini_k = i, ini_{k'} = i \text{ ó } fin_{k'} = i \end{gathered} \tag{2.42}$$

Restricciones relativas al cálculo de las medidas de los atributos

- Cálculo de la duración del reparto

$$TIME \geq T_{i,g(i)} \quad \forall i \tag{2.43}$$

- Cálculo del coste total del reparto

$$\begin{aligned} COST = & \sum_{j,k} dist_k \left(cosv_{j,k} L_{j,k} + cosf_{j,k} \sum_{e} X_{j,k,e} \right) \\ & + \sum_{i,j} creg_{j,i} \left(\sum_{e,k/fin_k = i} X_{j,k,e} - \sum_{e,k/ini_k = i} X_{j,k,e} \right) \end{aligned} \tag{2.44}$$

- Limitación del presupuesto

$$COST \leq b \tag{2.45}$$

- Cálculo del índice de equidad

$$EQUI \geq 1 - \frac{S_{i,g(i)}}{dem_i} \quad \forall i\ /\ dem_i > 0 \tag{2.46}$$

- Cálculo del índice de prioridad

$$PRIO = \sum_{i/nivp_i > 0} nivp_i \left(1 - \frac{S_{i,g(i)}}{dem_i} \right) \tag{2.47}$$

- Cálculo del índice de seguridad

$$SECU = \sum_{k} \left(2 - \frac{\sum_{j,e} X_{j,k,e}}{|J|} \right) \ln pm_k \tag{2.48}$$

- Cálculo del índice de fiabilidad

$$RELI = \sum_{k} (1 - r_{k)} \sum_{j,e} X_{j,k,e} \tag{2.49}$$

Restricciones relativas a la naturaleza de las variables

- Condiciones sobre las variables de decisión

$$X_{j,k,e}, Z_{i,k,n(i)}, P_{k,k'} \in \{0, 1\} \tag{2.50}$$

$$L_{j,k}, S_{i,n(i)} \in \mathbb{Z}^+ \tag{2.51}$$

$$T_{i,n(i)}, D_k \geq 0 \tag{2.52}$$

- Condiciones sobre las variables objetivo

$$TIME, COST, EQUI, PRIO, SECU, RELI \geq 0 \tag{2.53}$$

2.3.3. Función objetivo

El modelo propuesto es multiobjetivo, si bien, como se verá en la siguiente sección, puede sustituirse la función objetivo vectorial propuesta por una función escalar agregando los objetivos, o aplicarse algunos enfoques de decisión multicriterio como la programación por metas o la programación compromiso.

- Minimizar el vector formado por todas las variables objetivo.

$$\text{mín}(TIME, COST, EQUI, PRIO, SECU, RELI)$$

2.4. Tratamiento del carácter multicriterio

Un posible enfoque para resolver un problema multicriterio, como el que nos ocupa, consiste en obtener la frontera de Pareto del conjunto factible, esto es, el conjunto de soluciones no dominadas. Sin embargo, el elevado número de atributos y la urgencia con la que habitualmente se han de planificar las rutas, impiden en la práctica enfocar el problema a la obtención de la frontera de Pareto, incluso en el caso de utilizar técnicas heurísticas para su resolución.

En los artículos que se han tomado como base para desarrollar el presente trabajo, el enfoque utilizado fue el de programación por metas, introducido en Charnes y Cooper (1961). En programación por metas se fija un nivel de aspiración para la medida de cada atributo y se calcula la diferencia entre el valor obtenido para la medida del atributo y el nivel de aspiración. La función objetivo recoge estas diferencias, penalizando aquellas en las que el nivel de aspiración no es alcanzado. Por ejemplo, mediante una expresión lexicográfica, en la que los atributos son clasificados en distintos niveles de importancia (Ortuño *et al.*, 2011) o, simplemente, como una suma ponderada y normalizada (Vitoriano *et al.*, 2011).

También es posible transformar el problema en uno mono-objetivo agregando todos los objetivos en una sola función, definida como la suma de las medidas de los atributos, donde la medida de cada objetivo g se ha de ponderar con un peso comprendido entre 0 y 1, $peso_g$, y se ha de normalizar dividiendo entre una cota superior adecuada, $cota_g$.

$$fob = \sum_g \frac{peso_g \cdot f_g}{cota_g}$$

El problema puede resolverse sucesivas veces variando paramétricamente los pesos para obtener un conjunto amplio de soluciones no dominadas. Para una representación gráfica de la frontera de Pareto o, al menos, de una parte de ella, pueden combinarse los atributos de dos en dos, o incluso de tres en tres, y variar los pesos correspondientes en sus rangos posibles. De esta manera se pueden obtener (o estimar en el caso de que se use un método heurístico) proyecciones de la frontera de Pareto sobre los espacios bidimensionales o tridimensionales formados por los atributos considerados. Nuevamente, esta técnica puede requerir demasiado esfuerzo computacional.

También es habitual trabajar con el problema mono-objetivo que resulta al fijar los pesos de acuerdo a las preferencias del decisor. Una vez obtenida la solución para los valores fijados de los pesos, si esta no satisface al decisor, los pesos van siendo modificados hasta que se logre obtener una solución que se ajuste a sus preferencias.

En el Capítulo 3 se propondrán cotas superiores para las medidas elegidas de los atributos. A modo de ejemplo, se introducen dos de las cotas. La cota para la medida del tiempo (tiempo total del reparto) se calcula asumiendo que el vehículo más lento ha de transitar por todos los arcos.

$$cota_1 = \sum_k \frac{dist_k}{\text{mín}\{vela_k, \text{mín}_t\{velv_t\}\}}$$

La cota para la medida de la equidad (desviación típica de las proporciones de demandas no satisfechas), es simplemente 0.5, por tratarse de la desviación típica de una variable cuyos valores están comprendidos entre 0 y 1.

$$cota_3 = \frac{1}{2}$$

Sin pérdida de generalidad, se supondrá que los pesos suman 1.

$$\sum_g peso_g = 1$$

Para que las preferencias del decisor queden convenientemente plasmadas en los pesos, las cotas deberían ser lo suficientemente ajustadas para que los cocientes $\frac{f_g}{cota_g}$ sean del mismo orden de magnitud. Pero esto no será posible en muchos casos: las cotas que dependen fuertemente de los datos, como las cotas para el coste o para el tiempo, habitualmente presentan valores muy grandes en comparación con los valores de las medidas correspondientes en soluciones no dominadas, mientras que la cota para la prioridad o sobre todo la cota para la equidad (esta última además no depende de los datos) suelen estar mucho más ajustadas.

Otro posible enfoque para tratar la naturaleza multicriterio del problema es la programación compromiso (Zeleny, 1973). En la programación compromiso (abreviada habitualmente como CP en la literatura) el objetivo es minimizar la distancia, según la norma p, al punto ideal en el espacio de resultados. El punto ideal tiene como componentes los valores óptimos de las medidas de los atributos. Cada valor óptimo es obtenido minimizando cada una de las medidas por separado. Para normalizar se ha de calcular también el punto anti-ideal, formado por los peores valores posibles de las medidas de los atributos sobre el conjunto de soluciones no dominadas. Sean $ideal_g$ y $anti_g$ las componentes del ideal y el anti-ideal correspondientes al atributo g. La función compromiso para la norma p presenta la siguiente expresión:

$$fco_p = \left[\sum_g peso_g \left(\frac{f_g - ideal_g}{anti_g - ideal_g}\right)^p\right]^{\frac{1}{p}}$$

Esta función representa la proximidad entre la solución que se esté considerando y el punto ideal, que no corresponde a ninguna solución factible (en otro caso el punto factible asociado sería óptimo para todos los atributos a la vez y por lo tanto el óptimo del problema multiobjetivo), por lo que no es alcanzable. Nuevamente, los pesos, $peso_g$, representan las preferencias del decisor. En la práctica, las normas más habituales son $p = 1, p = \infty$ -con ambas la función compromiso preserva la linealidad- y $p = 2$.

Capítulo 3

Algoritmo Constructivo

En este capítulo se presenta un algoritmo que permite construir soluciones factibles en un tiempo computacional aceptable para el problema planteado. El proceso es sofisticado, pues el hecho de que los vehículos deban viajar juntos implica una sincronización que se debe preservar en todo momento para asegurar la factibilidad de la solución. Esta sincronización se consigue introduciendo una relación de precedencia entre los arcos: en el itinerario de cada vehículo los arcos que se recorren antes en el tiempo deben preceder a los arcos que se recorren después. Esta relación de precedencia debe ser actualizada tras cada cambio en los itinerarios de los vehículos, y a su vez los cambios en los itinerarios de los vehículos deben respetar la relación. Se han distinguido dos tipos de precedencia, precedencia inmediata y precedencia total. La precedencia inmediata se establece entre los pares de arcos que son recorridos consecutivamente en la ruta de algún vehículo, mientras que la precedencia total, o precedencia propiamente dicha, es la extensión de la precedencia inmediata en el conjunto de todos los arcos para que la relación sea de orden estricto. Para diversificar el conjunto de soluciones potencialmente construidas se introduce el azar en algunas fases de la construcción.

El presente algoritmo puede dividirse en varias fases generales que se explicarán en esta introducción, indicándose entre paréntesis y en mayúsculas los procedimientos correspondientes a las descripciones. En el resto de secciones se introduce la notación y se explica con todo detalle, procedimiento a procedimiento, el funcionamiento del algoritmo.

En primer lugar se efectúa una fase de preproceso (procedimiento PREPROCESO) que consta de varias tareas: simplificar la red eliminando aquellos arcos que no tendría sentido usar (INUTILIZAR ARCOS); encontrar los caminos de retorno más baratos para los vehículos desde todos sus posibles nodos finales (CAMINO MÍNIMO); obtener cotas superiores para las funciones objetivo consideradas (OBTENER COTAS). Nótese que cuando el algoritmo se vaya a repetir para obtener distintas soluciones (como se hará en capítulos sucesivos cuando se integre en metaheurísticas) esta fase preproceso se realizará solo una vez.

A continuación se determinan los itinerarios de los vehículos (CONSTRUIR RUTAS), sin tener en cuenta todavía la cantidad de ayuda que han de transportar. Se realizan su-

cesivos sorteos entre todos los posibles pares vehículo-arco disponibles. Tras cada sorteo, el vehículo seleccionado es situado en el nodo final del arco seleccionado y se actualiza la relación de precedencia entre los arcos (AGREGAR PRECEDENCIAS). El proceso termina cuando no es posible realizar ningún movimiento más. Una vez determinadas las rutas, se calculan las duraciones de los trayectos de los convoyes por los arcos (DURACIÓN), teniendo en cuenta la velocidad del vehículo más lento de cada convoy, así como los instantes de llegada de los convoyes a sus nodos de destino (FIN ARCOS) e implícitamente los instantes de salida de sus nodos de origen.

La siguiente fase consiste en decidir la cantidad de ayuda que debe ser transportada por cada convoy (ENVIAR FLUJO). Para ello se emplea una adaptación aleatorizada del algoritmo de Ford-Fulkerson (Ford y Fulkerson, 1956) sobre una red de transporte en la que la capacidad de cada arco viene dada por la suma de las capacidades de los vehículos de los convoyes que se han determinado en la fase anterior. Además, es necesario introducir dos nodos ficticios: la fuente, conectada mediante arcos ficticios a los depósitos; y el sumidero, al que llegan arcos ficticios que parten de los nodos de demanda. El objetivo en este caso es hacer llegar a los depósitos la cantidad de ayuda planificada a priori. Por la naturaleza de este problema y la adaptación del algoritmo empleada, no está garantizado que pueda lograrse dicho objetivo. Para aumentar la probabilidad de conseguirlo, se permiten varios intentos tanto en la obtención de un camino de aumento como en la aplicación completa del algoritmo. En caso de agotar todos los intentos, las rutas obtenidas son eliminadas, repitiéndose la fase anterior. En caso de conseguir repartir lo planeado, se ordenan los eventos (llegadas y salidas de convoyes) en los nodos de acuerdo al instante en el que se producen y se calculan los *stocks* de material disponibles en los nodos tras cada evento (ORDENAR EVENTOS).

La solución construida hasta este punto es coherente en muchos aspectos: respeta la relación de precedencia entre los arcos, cumple la ley de equilibrio de flujo en los nodos en cuanto a los vehículos y el material de ayuda, los instantes en los que se producen los eventos están sincronizados, etc. Pero aún así no está garantizada su factibilidad, ya que podría ocurrir que tras un evento el *stock* en un nodo sea negativo. Esto es debido a que algún convoy parte del nodo con una cantidad de material que todavía no está disponible, pero que lo estará más adelante. Así, en esta fase (POSITIVAR) se detectan los posibles *stocks* negativos (BUSCAR NEGATIVOS) y se introducen nuevas precedencias entre los arcos para impedir que se produzcan. En caso de que sea imposible eliminar todos los *stocks* negativos, la solución no sería válida y habría que comenzar de nuevo su construcción.

Al final de algunos de los procedimientos más relevantes, incluyendo el programa principal, se incluirá, como resumen, un diagrama de flujo. Además, para ilustrar el funcionamiento del algoritmo se utilizará el pequeño ejemplo de la Figura 3.1. Hay dos depósitos (a y b) con 15 toneladas de ayuda cada uno, tres nodos de demanda (c, f, y g) donde se requieren 10, 15 y 15 toneladas respectivamente, así como dos nodos que pueden usarse para transbordos (d y e). Hay doce vehículos disponibles, seis de cada uno de los dos tipos (que se denominarán grande y pequeño) con las características (capacidad y velocidad) que se describen en la parte superior derecha de la figura. Los vehículos inicialmente están en

los nodos indicados. La cantidad total de material que se desea repartir es de 25 toneladas.

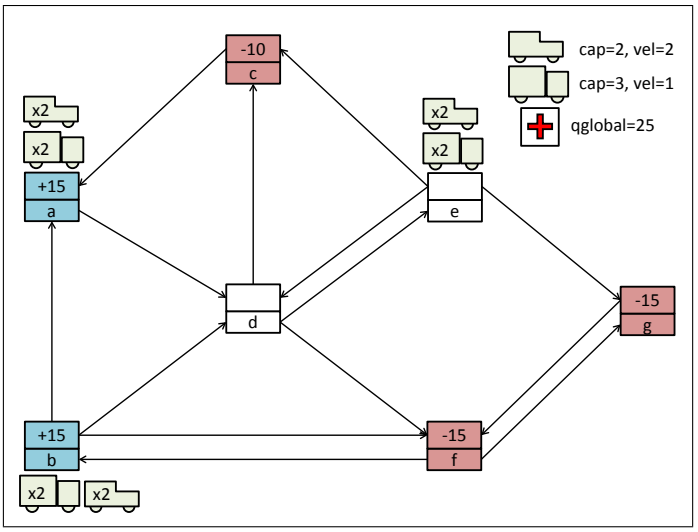


Figura 3.1: Situación inicial

El resto del capítulo se organiza de la siguiente forma: En la Sección 3.1 se introducen, definiéndose si no se ha hecho con anterioridad, los parámetros y variables del algoritmo; en la Sección 3.2 se presentan los distintos procedimientos internos de que consta el Algoritmo Constructivo; para terminar, el programa principal es detallado en la Sección 3.3. Esta misma estructura se usará en los capítulos 4, 5 y 6, dedicados también a otros algoritmos heurísticos.

3.1. Parámetros y variables

En esta sección se introducen los elementos que intervienen en el Algoritmo Constructivo, definiéndose aquellos que aparecen por primera vez en esta tesis y, simplemente, enumerándose los que han sido definidos con anterioridad. Están clasificados en cinco grupos: parámetros de lectura directa, parámetros calculados a partir de los datos del problema, variables de decisión, variables objetivo y variables auxiliares. Se consideran parámetros aquellos elementos cuyo valor inicial se fija, bien por lectura, bien en algún procedimiento previo a la construcción propiamente dicha de la solución, y no es modificado en el transcurso del algoritmo. Las variables pueden ver modificado su valor durante la construcción de la solución.

Se ha supuesto que la función objetivo que se debe evaluar es la suma ponderada, fob, con los pesos que se hayan establecido. En el caso de que se emplee la función compromiso, fco_p, se omite el cálculo de las cotas de los objetivos (procedimiento OBTENER COTAS) que se realiza en la fase de preproceso.

Parámetros de lectura directa

b, cap, $comp$, $cosf$, $cosv$, dem, $dist$, fin, ini, $nivp$, ori, p, $peso$, pm, qav, $qglobal$, r, $tdis$, $tipv$, $vela$, $velv$

Parámetros derivados

$clas_{t,i}$	Clase que corresponde al tipo de vehículos t y nodo de origen i
$cota_g$	Cota superior para el objetivo g. Se obtiene considerando los datos del problema general
$ctip_t$	Número de vehículos de tipo t
$dreg_{j,i}$	Distancia que recorre el vehículo j desde el nodo i para regresar
$grae_i$	Número de arcos que llegan al nodo i (grado de entrada)
$gras_i$	Número de arcos que salen del nodo i (grado de salida)
$ince_i$	Lista con los arcos que llegan al nodo i (incidencia de entrada)
$incs_i$	Lista con los arcos que salen del nodo i (incidencia de salida)
$sig_{c,i}$	Siguiente nodo al nodo i en el camino de regreso de un vehículo de la clase c

$creg$, $treg$

Variables de decisión

$load$, $secv$, $tmin$

Variables objetivo

f, fob

Variables auxiliares

$cont$	Variable binaria que indica si se está en proceso de construcción de rutas (1 sí; 0 no)
dur_k	Duración del trayecto del convoy que recorre el arco k
$inst_i$	Secuencia ordenada de instantes en los que se producen eventos en el nodo i
$into_i$	Secuencia ordenada de los arcos incidentes en el nodo i de acuerdo al instante en el que llegan o en el que salen
$negat$	Variable binaria que indica si hay algún *stock* negativo, en cuyo caso habría que arreglarlo si es posible (1 sí; 0 no)
$nuan_i$	Número de arcos recorridos incidentes en el nodo i
$nump_k$	Número de arcos que preceden inmediatamente al arco k
$prec_{k,k'}$	Variable binaria que indica si el arco k precede al arco k' (1 sí; 0 no)
$pred_k$	Lista con los arcos predecesores inmediatos del arco k
$prei_{k,k'}$	Variable binaria que indica si el arco k precede inmediatamente al arco k' (1 sí; 0 no)

$pret_k$	Lista con los arcos predecesores del arco k
$repc$	Variable binaria que indica que se ha conseguido repartir lo planeado (1 sí; 0 no)
$stck_i$	Secuencia ordenada de *stocks* disponibles en el nodo i de acuerdo al instante en el que se producen los eventos correspondientes
$suci_k$	Lista con los arcos sucesores inmediatos del arco k
$suct_k$	Lista con los arcos sucesores del arco k
$tamc_k$	Tamaño del convoy que recorre el arco k
$util_{j,k}$	Variable binaria que indica si puede ser útil que el vehículo j recorra el arco k (1 sí; 0 no)

$conv$, $lonv$, $ltar$, sf

3.2. Procedimientos

A continuación se presentan los procedimientos en que se divide el Algoritmo Constructivo. En cada uno de los procedimientos donde se requieren, se incluyen listas de variables de entrada, variables de salida, variables internas y procedimientos llamados. Las variables de entrada alimentan al procedimiento y proceden del programa principal o de otros procedimientos de mayor jerarquía. Se consideran variables de salida las que han podido ver modificado su valor en el transcurso del procedimiento. Las variables internas son de uso exclusivo del propio procedimiento. Los procedimientos llamados son los que pueden ser ejecutados en el transcurso del procedimiento descrito. Además de los valores de las variables de entrada, en cada procedimiento se tiene acceso a los valores de todos los parámetros que ya hayan sido fijados en procedimientos ejecutados previamente.

Nótese que las expresiones entre paréntesis que aparecen en el desarrollo de los algoritmos son meramente aclaratorias del estado actual en el paso correspondiente, no son condiciones que se hayan de comprobar.

Procedimiento 1. - LEER

Establecimiento de los valores de los parámetros del problema.

Algoritmo

- Leer cardinales de los conjuntos I, J, K y T
- Leer valores de los parámetros de lectura directa
- Obtener los valores de los parámetros derivados $clas_{t,i}$, $ctip_t$, $grae_i$, $gras_i$, $ince_i$ y $incs_i$

Procedimiento 2. - INUTILIZAR ARCOS

Búsqueda de arcos por los que en ningún caso se debería pasar por empeorar la solución de forma obvia. Cuando un nodo está conectado con el resto del grafo únicamente por un arco o un par de arcos de sentidos contrarios, no tendría sentido que determinados vehículos alcancen dicho nodo. En este procedimiento se impide que estos vehículos puedan desplazarse por los arcos correspondientes.

Variables de salida

util

Algoritmo

- Hacer útiles todos los arcos compatibles a los vehículos

$$util_{j,k} = comp_{tipv_j,k}$$

- Cuando algún nodo que no es de demanda ni de oferta está conectado exclusivamente con otro nodo mediante un par de arcos de sentidos contrarios, se prohíbe a todos los vehículos que no tengan origen en dicho nodo transitar por los arcos incidentes en él

$$\forall i \in I - (ID \cup IO) \ / \ gras_i = 1,\ grae_i = 1,\ ini_{ince_{i,1}} = fin_{incs_{i,1}} \quad \Rightarrow$$
$$\Rightarrow \quad util_{j,ince_{i,1}} = 0,\ util_{j,incs_{i,1}} = 0 \quad \forall j \ / \ ori_j \neq i$$

- Cuando algún nodo que no es de demanda está conectado exclusivamente con otro nodo mediante un par de arcos de sentidos contrarios, se prohíbe a todos los vehículos que tengan origen en dicho nodo regresar a él durante el reparto (sí podría regresar una vez finalizado el reparto)

$$\forall i \in I - (ID) \ / \ gras_i = 1,\ grae_i = 1,\ ini_{ince_{i,1}} = fin_{incs_{i,1}} \quad \Rightarrow$$
$$\Rightarrow \quad util_{j,ince_{i,1}} = 0 \quad \forall j \ / \ ori_j = i$$

- Cuando no existen arcos de salida en algún nodo que no es de demanda, se prohíbe a todos los vehículos llegar a dicho nodo, ya que de otro modo no podrían continuar su ruta (si es obligatorio que todos los vehículos regresen a su origen tras el reparto se incluirían aquí también los nodos de demanda)

$$\forall i \in I - (ID) \ / \ gras_i = 0 \quad \Rightarrow \quad util_{j,ince_{i,e}} = 0 \quad \forall j,\ \forall e \in \{1, \ldots, grae_i\}$$

Procedimiento 3. - CAMINO MÍNIMO

Obtención de los caminos de mínimo coste (que se denominarán caminos mínimos) desde todos los posibles nodos finales a los orígenes de los vehículos. El procedimiento calcula los costes, $creg_{j,i}$, los tiempos, $treg_{j,i}$, y las longitudes, $dreg_{j,i}$, de los caminos mínimos para cada vehículo j y cada posible nodo final i, así como los nodos subsiguientes en los

caminos de regreso obtenidos para los vehículos de cada clase, $sig_{c,i}$. Para ello se aplica el algoritmo de Dijkstra (Dijkstra, 1959) para cada clase de vehículos tomando como origen el que corresponda a la clase y considerando los arcos en sentido contrario. Este procedimiento se realiza previamente a la construcción de las soluciones para, de esta forma, poder evaluar directamente la función objetivo en cualquier solución que se obtenga; se omite si no se requiere que los vehículos regresen a sus respectivos orígenes.

Se ha considerado que el coste es el criterio más importante a la hora de decidir los caminos de regreso. Con mínimos cambios se podrían obtener los caminos de regreso más cortos o los más rápidos.

Cada vehículo regresa a su nodo de origen una vez completada su tarea en el reparto. Como los vehículos regresan vacíos, sin material de ayuda, se ha asumido que no existe riesgo de ataques y no es necesario que formen convoyes.

Variables internas

$c_{i,i'}$	Coste por recorrer el arco que une los nodos i e i' con un vehículo vacío de la clase que se esté considerando
$cr_{c,i}$	Coste de regreso de un vehículo de la clase c que finaliza su reparto en el nodo i
$d_{i,i'}$	Distancia del arco que une los nodos i e i'
$dr_{c,i}$	Distancia que recorre un vehículo de la clase c desde el nodo i para regresar
$tp_{i,i'}$	Tiempo que tarda en recorrer el arco que une los nodos i e i' un vehículo de la clase que se esté considerando
$tr_{c,i}$	Tiempo que tarda un vehículo de la clase c en regresar desde el nodo i

Algoritmo

PASO 0

- Tomar el primer tipo de vehículos, $t = 1$
- Si no quedan más tipos, ir al PASO 2
- Calcular los costes, los tiempos y las distancias entre cada par de nodos para el tipo t

$$c_{i,i'} = \begin{cases} cosf_{t,k} \cdot dist_k & \text{si } \exists k \ / \ ini_k = i',\ fin_k = i,\ comp_{t,k} = 1 \\ \infty & \text{en otro caso} \end{cases}$$

$$tp_{i,i'} = \begin{cases} \dfrac{dist_k}{\text{mín}\{velv_t, vela_k\}} & \text{si } \exists k \ / \ ini_k = i',\ fin_k = i,\ comp_{t,k} = 1 \\ \infty & \text{en otro caso} \end{cases}$$

$$d_{i,i'} = \begin{cases} dist_k & \text{si } \exists k \ / \ ini_k = i',\ fin_k = i,\ comp_{t,k} = 1 \\ \infty & \text{en otro caso} \end{cases}$$

$$\forall i, i'$$

- Tomar el primer nodo, $i' = 1$

PASO 1

- Si no quedan más nodos, pasar al siguiente tipo t (si queda alguno) y volver al PASO 0
- Si no existe ningún vehículo que parta de i' y sea de tipo t, pasar al siguiente nodo i' (si queda alguno) y volver al PASO 1
- (Existen vehículos que parten de i' y son de tipo t). Por medio del algoritmo de Dijkstra, calcular el camino de coste mínimo desde cualquier nodo i al nodo i' para la clase $c = clas_{t,i'}$. Sean $cr_{c,i}$, $tr_{c,i}$ y $dr_{c,i}$ el coste, la distancia y el tiempo del camino de mínimo coste, respectivamente.
- Pasar al siguiente nodo i' (si queda alguno) y volver al PASO 1

PASO 2

- Para cada vehículo j y cada nodo i (sea $c = clas_{tipv_j,ori_j}$ la clase correspondiente al vehículo j), hacer:
  - Si el camino de mínimo coste desde i hasta el nodo origen de j tiene coste infinito, $cr_{c,i} = \infty$, prohibir que el vehículo j recorra los arcos que llegan al nodo i
  $$util_{j,k} = 0 \quad \forall k \in ince_i$$
  - En otro caso, guardar el coste, tiempo y distancia de regreso del vehículo j desde el nodo i
  $$creg_{j,i} = cr_{c,i}$$
  $$treg_{j,i} = tr_{c,i}$$
  $$dreg_{j,i} = dr_{c,i}$$

Procedimiento 4. - ACOTAR COSTE

Obtención de la cota superior para el coste total del reparto (sin regresos) cuando la suma de las capacidades de todos los vehículos supera la cantidad total de material a repartir. En este caso, se tiene en cuenta que se deben llenar primero los vehículos con menor coste variable de transporte, ya que este es el criterio que se sigue en el procedimiento REPARTIR CARGAS.

Variables de salida

csin Cota para el coste total sin tener en cuenta el regreso

Variables internas

lisv Lista con los vehículos compatibles con un arco, ordenados de menor a mayor coste variable de transporte

lres Material que todavía no ha sido repartido entre los vehículos que recorren el arco actual

Algoritmo

- Inicializar la cota para el coste, suponiendo que todos los vehículos recorren todos los arcos compatibles

$$csin = \sum_{t/comp_{t,k}=1} cosf_{t,k}$$

- Tomar el primer arco, $k = 1$

PASO 0

- Si no quedan más arcos, FIN del procedimiento
- Inicializar la cantidad de material que queda por repartir en el arco k

$$lres = qglobal$$

- Crear la lista, $lisv$, con los vehículos compatibles con dicho arco

$$j \in lisv \quad \Leftrightarrow \quad comp_{tipv_j,k} = 1$$

- Ordenar la lista $lisv$ de menor a mayor coste variable de transporte. Los empates se deshacen al azar (o arbitrariamente)

PASO 1

- Elegir el primer vehículo j de la lista $lisv$. Si no quedan vehículos, $lisv = \varnothing$, pasar al siguiente arco k (si queda alguno) y volver al PASO 0
- Si en el vehículo actual cabe todo el material restante, $cap_{tipv_j} \geq lres$:
  - Suponer que se llena el vehículo j con el material restante

$$csin = csin + lres \cdot cosv_{tipv_j,k} \cdot dist_k$$

  - Pasar al siguiente arco k (si queda alguno)
  - Volver al PASO 0
- (En el vehículo actual no cabe todo el material restante, $cap_{tipv_j} < lres$):
  - Suponer que se llena del todo el vehículo j

$$csin = csin + cap_{tipv_j} \cdot cosv_{tipv_j,k} \cdot dist_k$$

  - Actualizar la cantidad de material restante

$$lres = lres - cap_{tipv_j}$$

  - Eliminar el vehículo j de la lista de vehículos que todavía no han recibido carga, $lisv$
  - Volver al PASO 1

Procedimiento 5. - PÉRDIDA ESPERADA

Cálculo de las máximas pérdidas esperadas en los arcos ($pema_k$). Se requieren para obtener una cota superior para la función objetivo basada en la seguridad. Se ha considerado que la gravedad de un asalto viene dada por la cantidad de material que transporte el convoy, objetivo usual de los asaltantes. La máxima pérdida esperada en un arco para un convoy de tamaño fijo es el producto entre la probabilidad de que un convoy de ese tamaño sufra un asalto en el arco y la cantidad total de material que puede caber en los vehículos de un convoy de ese tamaño. La máxima pérdida esperada en un arco se obtiene como el máximo entre las máximas pérdidas esperadas para los convoyes de todos los tamaños posibles. Los vehículos son ordenados de mayor a menor capacidad con el objeto de considerar siempre convoyes de máxima capacidad total para cada tamaño posible.

Variables internas

ch	Tamaño hipotético para un convoy. Durante la ejecución del algoritmo esta variable tomará todos los valores enteros desde 1 hasta el tamaño mínimo de un convoy en el que se podría cargar todo el material que se quiere repartir
$lisv$	Lista con los vehículos ordenados de mayor a menor capacidad
$maxc_{ch}$	Máxima capacidad para un convoy con ch vehículos

Variables de salida

$pema_k$	Máxima pérdida esperada en caso de sufrir un asalto en el arco k

Algoritmo

PASO 0

- Obtener la máxima capacidad para un convoy con $j = 0$ vehículos

$$maxc_0 = 0$$

- Crear la lista $lisv$ con todos los vehículos ordenados de mayor a menor capacidad. Los empates se deshacen al azar (o arbitrariamente)
- Inicializar la cantidad de material que quedaría por repartir entre los vehículos

$$lres = qglobal$$

- Inicializar el tamaño hipotético del convoy

$$ch = 1$$

- Tomar el primer vehículo j de la lista $lisv$

PASO 1

- Si no quedan vehículos en la lista, $lisv = \varnothing$:
  - Fijar el tamaño mínimo de un convoy en el que se puede cargar todo el material como el número total de vehículos

$$ch = |J|$$

  - Ir al PASO 2
- Si el material restante es menor o igual que la capacidad del vehículo actual, $lres \leq cap_{tipv_j}$,
  - Calcular la máxima capacidad para un convoy con ch vehículos

$$maxc_{ch} = qglobal$$

  - Ir al PASO 2
- (El material restante es mayor que la capacidad del vehículo actual, $lres > cap_{tipv_j}$):
  - Calcular la máxima capacidad para un convoy con ch vehículos

$$maxc_{ch} = maxc_{ch-1} + cap_{tipv_j}$$

  - Actualizar la cantidad de material restante

$$lres = lres - cap_{tipv_j}$$

  - Eliminar el vehículo j de la lista de vehículos $lisv$
  - Tomar el primer vehículo j de la lista de vehículos $lisv$ (si queda alguno)
  - Actualizar el tamaño hipotético del convoy

$$ch = ch + 1$$

  - Volver al PASO 1

PASO 2

- Calcular la máxima pérdida esperada para cada arco k, $pema_k$, considerando todos los posibles tamaños para un convoy

$$pema_k = \text{máx}\left\{\underset{j \leq tdis}{\text{máx}}\{FP(p_k, pm_k, j) \cdot maxc_j\}, pm_k \cdot maxc_{ch}\right\}$$

Procedimiento 6. - OBTENER COTAS

Obtención de las cotas superiores para los objetivos, $cota_g$, necesarias para evaluar la solución que se obtenga.

Variables internas

pema, *csin*

Procedimientos llamados

ACOTAR COSTE, PÉRDIDA ESPERADA

Algoritmo

- Cota para el tiempo total del reparto. Su expresión viene dada por lo que tardaría el vehículo más lento en recorrer todos los arcos

$$cota_1 = \sum_k \frac{dist_k}{\min\{vela_k, \min_t\{velv_t\}\}}$$

- Cota para el coste total del reparto:

  - Si el máximo (entre los arcos) de las sumas de las capacidades de todos los vehículos compatibles no supera la cantidad total de material a repartir, *qglobal*, la cota para el coste sin regresos se obtiene suponiendo que todos los vehículos compatibles recorren llenos todos los arcos

$$\max_k \left\{ \sum_{t/comp_{t,k}=1} cap_t \cdot ctip_t \right\} \leq qglobal \quad \Rightarrow$$

$$\Rightarrow \quad csin = \sum_k \sum_{t/comp_{t,k}=1} (cosf_{t,k} + cosv_{t,k} \cdot cap_t \cdot ctip_t)\, dist_k$$

  - Si el máximo (entre los arcos) de las sumas de las capacidades de todos los vehículos compatibles supera la cantidad total de material a repartir, *qglobal*, la cota se obtiene suponiendo que se llenan primero los vehículos de menor coste variable. Se utiliza para ello el procedimiento ACOTAR COSTE

$$\max_k \left\{ \sum_{t/comp_{t,k}=1} cap_t \cdot ctip_t \right\} > qglobal \quad \Rightarrow \quad \text{ACOTAR COSTE}$$

  - Añadir la cota para los regresos de los vehículos

$$cota_2 = csin + \sum_j \max_i \{creg_{clas_j,i}\}$$

- Cota para la medida de la equidad en el reparto. La desviación típica de un conjunto de valores comprendidos entre 0 y 1 es a lo sumo 0.5

$$cota_3 = \frac{1}{2}$$

- Cota para la medida de la prioridad. La cota se obtiene suponiendo que no se satisface demanda alguna en los nodos con prioridad positiva

$$cota_4 = \sum_{i \in ID} nivp_i$$

- Cota para la medida de la seguridad:
  - Ejecutar el procedimiento PÉRDIDA ESPERADA, para calcular la máxima pérdida esperada en cada arco k, $pema_k$
  - La cota para la suma de máximas pérdidas esperadas de material se obtiene sumando las máximas pérdidas esperadas en los arcos

$$cota_5 = \sum_{k} pema_k$$

- Cota para la medida de la fiabilidad. La cota para la suma de gravedades esperadas se alcanza cuando por todos los arcos pasan convoyes de importancia máxima, es decir, de forma que los vehículos que lo forman deberán descargar a partir de cada arco todo el material que se desea repartir

$$cota_6 = \sum_{k} (1 - r_k) qglobal$$

Procedimiento 7. - PREPROCESO

Realización de lecturas y cálculos previos a la construcción de la solución.

Variables de salida

util

Procedimientos llamados

CAMINO MÍNIMO, INUTILIZAR ARCOS, LEER, OBTENER COTAS

Algoritmo

- Ejecutar el procedimiento LEER, para almacenar los datos del problema
- Ejecutar el procedimiento INUTILIZAR ARCOS, para descartar arcos y nodos que no sean necesarios en el reparto

- Ejecutar el procedimiento CAMINO MÍNIMO, para obtener los caminos de mínimo coste desde los potenciales nodos finales a los orígenes de los vehículos
- Ejecutar el procedimiento OBTENER COTAS, para calcular cotas superiores de las funciones objetivo

Procedimiento 8. - AGREGAR PRECEDENCIAS

Actualización de las variables que almacenan las relaciones de precedencia (precedencia inmediata y precedencia total). El procedimiento recibe un par de arcos, denominados arco predecesor principal, *apre*, y arco sucesor principal, *apos*. Estos dos arcos deben relacionarse mediante una precedencia inmediata y, si no estuviesen relacionados previamente, también mediante la correspondiente precedencia total. Además, la relación de precedencia total debe extenderse para que se preserve su estructura de orden estricto. Cada vez que se introduce una precedencia total entre dos arcos, se debe impedir que los vehículos que recorren el arco sucesor recorran posteriormente el arco predecesor. Esto se consigue mediante las variables $aper_{j,k}$, que indican qué arcos puede recorrer en el futuro cada vehículo.

Las precedencias se van agregando a medida que se van construyendo las rutas de los vehículos, con el fin de conseguir la sincronización necesaria para que los vehículos viajen en convoyes por todos los arcos. Asimismo, también se introducirán ciertas precedencias entre los arcos cuando en algún nodo se produzca un *stock* negativo, indicador de que un convoy está saliendo del nodo antes de tiempo.

Variables de entrada

$aper_{j,k}$	Variable binaria que indica si el vehículo j podrá transitar en el futuro por el arco k (1 sí; 0 no)
apos	Arco sucesor principal para agregar precedencias
apre	Arco predecesor principal para agregar precedencias

cont, *conv*, *nump*, *prec*, *pred*, *prei*, *pret*, *suci*, *suct*

Variables de salida

aper, *nump*, *prec*, *pred*, *prei*, *pret*, *suci*, *suct*

Algoritmo

- Actualizar la matriz de precedencias inmediatas

$$prei_{apre,apos} = 1$$

- Actualizar las listas de predecesores y sucesores inmediatos

$$pred_{apos} = pred_{apos} \cup \{apre\}$$
$$suci_{apre} = suci_{apre} \cup \{apos\}$$

- Si ya existe precedencia total entre $apre$ y $apos$, $prec(apre, apos) = 1$, FIN del procedimiento
- (No existe precedencia total entre $apre$ y $apos$, $prec(apre, apos) = 0$). Actualizar la matriz de precedencias totales

$$prec_{apre,apos} = 1$$

- Actualizar las listas de predecesores y sucesores totales

$$pret_{apos} = pret_{apos} \cup \{apre\}$$
$$suct_{apre} = suct_{apre} \cup \{apos\}$$

- Si se están construyendo las rutas, $cont = 1$, impedir que los vehículos del convoy del arco sucesor principal, $apos$ (arco seleccionado en el procedimiento CONSTRUIR RUTAS), recorran en el futuro el arco predecesor principal, $apre$

$$aper_{j,apre} = 0 \quad \forall j \in conv_{apos}$$

- Para cada arco k predecesor del arco predecesor principal, $apre$, sin relación de precedencia con el arco sucesor principal, $apos$, $\forall k \in pret_{apre}$ / $prec(k, apos) = 0$, hacer:
  - Actualizar la matriz de precedencias inmediatas

$$prec_{k,apos} = 1$$

  - Actualizar las listas de predecesores y sucesores totales

$$pret_{apos} = pret_{apos} \cup \{k\}$$
$$suct_{k} = suct_{k} \cup \{apos\}$$

  - Si se están construyendo las rutas, $cont = 1$, impedir que los vehículos del convoy del arco sucesor principal, $apos$ (arco seleccionado en el procedimiento CONSTRUIR RUTAS), recorran en el futuro el arco k

$$aper_{j,k} = 0 \quad \forall j \in conv_{apos}$$

  - Para cada arco k' sucesor del arco sucesor principal, $apos$, sin relación de precedencia con el arco k, $\forall k' \in suct_{apos}$ / $prec(k, k') = 0$, hacer:
    - Actualizar la matriz de precedencias inmediatas

$$prec_{k,k'} = 1$$

    - Actualizar las listas de predecesores y sucesores totales

$$pret_{k'} = pret_{k'} \cup \{k\}$$
$$suct_{k} = suct_{k} \cup \{k'\}$$

- Si se están construyendo las rutas, $cont = 1$, impedir que los vehículos del convoy del arco k' recorran en el futuro el arco k

$$aper_{j,k} = 0 \quad \forall j \in conv_{k'}$$

- Para cada arco k' sucesor del arco sucesor principal, *apos*, sin relación de precedencia con el arco predecesor principal, *apre*, $\forall k' \in suct_{apos} \ / \ prec(apre, k') = 0$, hacer:
  - Actualizar la matriz de precedencias inmediatas

$$prec_{apre,k'} = 1$$

  - Actualizar las listas de predecesores y sucesores totales

$$pret_{k'} = pret_{k'} \cup \{apre\}$$
$$suct_{apre} = suct_{apre} \cup \{k'\}$$

  - Si se están construyendo las rutas, $cont = 1$, impedir que los vehículos del convoy del arco k' recorran en el futuro el arco predecesor principal, *apre*

$$aper_{j,apre} = 0 \quad \forall j \in conv_{k'}$$

Procedimiento 9. - CONSTRUIR RUTAS

Construcción de las rutas de los vehículos. Inicialmente todos los vehículos están situados en sus nodos de origen. En cada etapa del algoritmo se identifican todos los posibles movimientos (cada movimiento representa un trayecto de un vehículo por un arco) y se selecciona uno al azar. Una vez seleccionado un movimiento, se añade el arco a la ruta del vehículo, se sitúa el vehículo en el nodo final del arco y se actualizan las relaciones de precedencia entre los arcos de forma que no se produzcan incompatibilidades en las rutas. El algoritmo termina cuando no es posible realizar ningún movimiento, es decir, cuando no pueden prolongarse las rutas.

Como variante, se puede incluir un criterio de parada basado en una estimación de la demanda satisfecha. En este caso, se computarían el material estimado en movimiento y la demanda satisfecha estimada suponiendo que cada vez que un vehículo pasa por un depósito toma todo el material que puede y cada vez que pasa por un nodo de demanda deja todo el material que puede. Cuando la demanda satisfecha estimada alcanza la cantidad a repartir, se para la construcción de rutas con determinada probabilidad. Si tras el procedimiento CONSTRUIR RUTAS no es posible repartir lo planeado, las rutas se prolongarían hasta que el criterio de parada lo impida. Se continuaría este proceso hasta que se reparta lo planeado o hasta que no puedan prolongarse las rutas. La versión que se describe en el siguiente algoritmo no contempla la posibilidad de este criterio de parada, al no haberse encontrado indicios de mejora en las pruebas realizadas. Sin embargo, en las versiones modificadas de este procedimiento incluidas en las metaheurísticas que se describirán en los capítulos 5 y 6, sí se introducirán criterios de parada basados en elementos de las propias metaheurísticas.

Variables de entrada

util

Variables de salida

cont, *conv*, *lonv*, *nump*, *prec*, *pred*, *prei*, *pret*, *secv*, *suci*, *suct*, *tamc*

Variables internas

act_j	Nodo en el que se encuentra el vehículo j en una fase de la construcción
agan	Arco seleccionado para el siguiente movimiento
asor	Lista con los arcos que corresponden a los distintos movimientos posibles
vgan	Vehículo seleccionado para el siguiente movimiento
vsor	Lista con los vehículos que corresponden a los distintos movimientos posibles

aper, *apos*, *apre*

Procedimientos llamados

AGREGAR PRECEDENCIAS

Algoritmo

PASO 0

- Inicializar las variables de almacenamiento de la solución y de las relaciones de precedencia asignando el valor 0 a las siguientes variables, vectores y matrices

$$lonv, nump, prec, prei, tamc$$

- Indicar que se está en proceso de construcción de rutas

$$cont = 1$$

- Situar los vehículos en sus nodos de origen

$$act_j = ori_j \quad \forall j$$

- Cualquier trayecto útil de un vehículo j por un arco k está permitido

$$aper_{j,k} = util_{j,k} \quad \forall j, \forall k$$

PASO 1

- Identificar todos los movimientos posibles. Cada movimiento corresponde a un vehículo j y un arco transitable k desde su ubicación actual act_j

- Si no hay ningún movimiento posible:
  - Indicar que no se está en proceso de construcción de rutas

$$cont = 0$$

  - FIN del procedimiento
- Elegir al azar un movimiento. Sean *vgan* y *agan* el vehículo y el arco seleccionados (los correspondientes al movimiento elegido)

PASO 2

- Aumentar en una unidad el tamaño de la ruta del vehículo seleccionado *vgan* y añadir el arco *agan* a dicha ruta

$$lonv_{vgan} = lonv_{vgan} + 1$$
$$secv_{vgan,lonv_{vgan}} = agan$$

- Aumentar en una unidad el tamaño del convoy del arco seleccionado y añadir el vehículo a dicho convoy

$$tamc_{agan} = tamc_{agan} + 1$$
$$conv_{agan,tamc_{agan}} = vgan$$

- Ubicar el vehículo seleccionado en el nodo final del arco seleccionado

$$act_{vgan} = fin_{agan}$$

- Si el arco seleccionado es el primero en el itinerario del vehículo seleccionado, $lonv_{vgan} = 1$, ir al PASO 3 (prohibición de movimientos futuros por violar las precedencias)
- (El arco seleccionado no es el primero de la ruta del vehículo seleccionado, $lonv_{vgan} > 1$). Guardar el arco anterior al arco seleccionado en la ruta del vehículo seleccionado como candidato a preceder inmediatamente al arco seleccionado

$$apre = secv_{vgan,lonv_{vgan}-1}$$

- Si el arco seleccionado no estaba ya precedido inmediatamente por el arco anterior en la ruta del vehículo seleccionado, $prei_{apre,agan} = 0$, hacer:
  - Guardar el arco seleccionado como precedido

$$apos = agan$$

  - Ejecutar el procedimiento AGREGAR PRECEDENCIAS, para obligar a que el arco *apre* preceda inmediatamente al arco *apos* y actualizar las precedencias correspondientes

PASO 3

- Impedir que el vehículo seleccionado recorra en el futuro el arco seleccionado

$$aper_{vgan,agan} = 0$$

- Impedir que el vehículo seleccionado recorra en el futuro los arcos predecesores del arco seleccionado

$$aper_{vgan,k} = 0 \quad \forall k \in pret_{agan}$$

- Volver al PASO 1

En la Figura 3.2 se ha resumido el procedimiento mediante un diagrama de flujo.

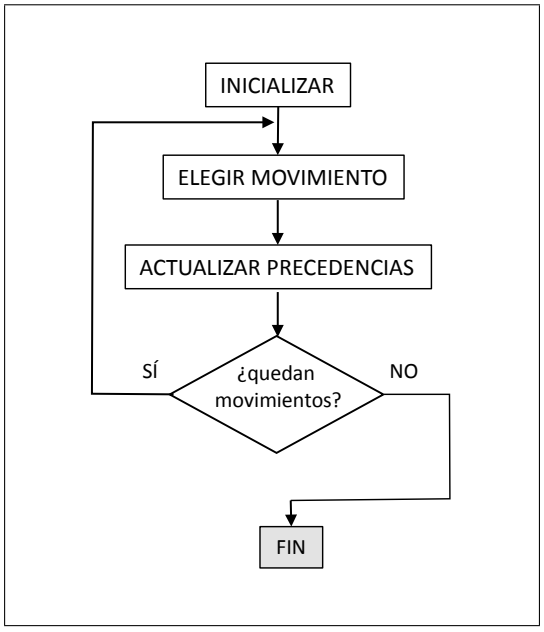
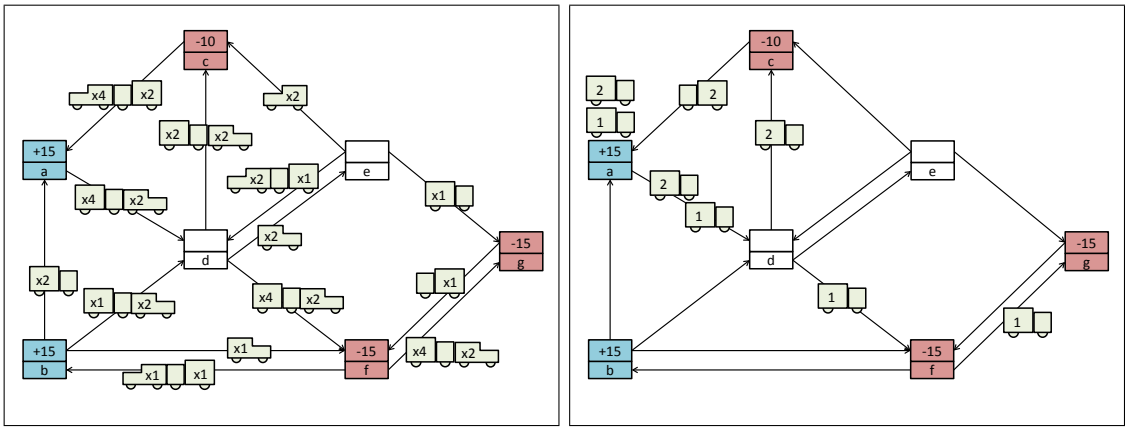


Figura 3.2: Diagrama de flujo de CONSTRUIR RUTAS

La Figura 3.3a muestra los convoyes formados tras ejecutar este procedimiento una vez tomando como datos los del ejemplo de la Figura 3.1. En esta figura no pueden apreciarse explícitamente las rutas de los vehículos, aunque en algún caso pueden intuirse, solo la composición de los convoyes. A modo de ejemplo, en la Figura 3.3b se muestran en este punto las rutas de los dos vehículos grandes que originalmente estaban situados en el nodo a. El primero de ellos sigue el itinerario $a - d - f - g$ mientras que el segundo realiza la ruta $a - d - c - a$. Nótese que la naturaleza aleatoria del procedimiento propiciaría que se obtuviesen soluciones distintas en distintas ejecuciones.

Procedimiento 10. - DURACIÓN

Realización de dos tareas independientes. Por una parte, se crean las listas de incidencia asociadas a los itinerarios creados y, por otra parte, se calculan las duraciones de los



(a) Convoyes (b) Rutas de los vehículos 1 y 2

Figura 3.3: Solución parcial tras CONSTRUIR RUTAS

trayectos de los convoyes por los arcos. Se tiene en cuenta que el convoy viaja a la velocidad del vehículo más lento, o a la velocidad máxima permitida en el arco, si esta es menor que las velocidades de los vehículos.

Variables de entrada

conv, *tamc*

Variables de salida

dur, *into*, *nuan*

Algoritmo

- Construir, para cada nodo i, la lista de incidencia, $into_i$, y su grado, $nuan_i$. Para que un arco esté incluido en una de estas listas ha de ser recorrido por algún vehículo
- Obtener la duración del trayecto de cada convoy

$$dur_k = \frac{dist_k}{\text{mín}\{vela_k, \{\text{mín}_{j \in conv_k}\{velv_{tipv_j}\}\}} \quad \forall k \ / \ tamc_k > 0$$

Procedimiento 11. - FIN ARCOS

Cálculo de los instantes mínimos de finalización de los trayectos por los arcos usados. Se comienza por los arcos sin predecesores inmediatos. Para que se pueda calcular el instante mínimo de finalización de un trayecto de un arco, previamente han debido haber sido calculados los respectivos instantes de sus arcos predecesores inmediatos.

Variables de entrada

dur, *nump*, *pred*, *suci*, *tamc*

Variables de salida

tmin

Variables internas

lisr	Lista de arcos sin evaluar
$numprest_k$	Número de predecesores inmediatos del arco k sin considerar

Algoritmo

PASO 0

- Formar la lista inicial de arcos sin evaluar, *lisr*, incluyendo todos los arcos usados en las rutas
- Inicializar el número de predecesores inmediatos sin considerar, $numprest_k$, de cada arco usado k, haciéndolo coincidir con el número de predecesores inmediatos original, $nump_k$

$$numprest_k = nump_k \quad \forall k \ / \ tamc_k > 0$$

PASO 1

- Elegir un arco k sin predecesores inmediatos que esté sin evaluar, es decir, tal que $numprest_k = 0$
- Calcular el tiempo mínimo de finalización del arco elegido teniendo en cuenta que no puede comenzar hasta que hayan finalizado su recorrido los convoyes de sus arcos predecesores

$$tmin_k = \begin{cases} \text{máx}_{k' \in pred_k}\{tmin_{k'}\} + dur_k & \text{si } pred_k \neq \varnothing \\ dur_k & \text{si } pred_k = \varnothing \end{cases}$$

PASO 2

- Eliminar el arco k de la lista de arcos sin evaluar, *lisr*
- Si no quedan arcos sin evaluar, $lisr = \varnothing$, FIN del procedimiento
- Decrementar en una unidad el número de predecesores inmediatos sin considerar, $numprest_{k'}$ de cada arco k' que sea sucesor inmediato de k

$$numprest_{k'} = numprest_{k'} - 1 \quad \forall k' \in suci_{k'}$$

- Volver al PASO 1

Procedimiento 12. - ENVIAR FLUJO

Cálculo de las cantidades de material transportadas por los arcos. Para ello, se crea una red de transporte a partir del grafo dirigido original formada por todos los nodos, pero solo los arcos transitados por algún vehículo. La capacidad de cada arco es la suma de las capacidades de los vehículos que lo recorren. Se introducen dos nodos artificiales: la fuente, f, del que parten arcos ficticios a los depósitos, y el sumidero, s, al que llegan arcos ficticios desde los nodos de demanda. Las capacidades de los arcos ficticios vienen dadas por las ofertas o demandas de sus respectivos nodos. Además, la cantidad a repartir supone una cota superior para las capacidades de todos los arcos. Sobre la red así construida se aplica una adaptación del algoritmo de Ford-Fulkerson con las siguientes características:

- La elección de los nodos a etiquetar se realiza al azar.
- No se permiten movimientos de retroceso (por originar con frecuencia incompatibilidades entre precedencias).
- Solo se permiten movimientos de avance que no violen las precedencias existentes.
- Se permiten varios intentos para obtener un camino de aumento en cada iteración.
- Se permiten varios intentos en la aplicación completa del algoritmo.
- El algoritmo finaliza cuando el flujo coincide con la cantidad a repartir, *qglobal*, o cuando se acaban todos los intentos. En este último caso, se desechan las rutas construidas y se construye una nueva solución desde el principio a partir del procedimiento CONSTRUIR RUTAS.

Parámetros internos

$maxet$	Número máximo de intentos para obtener un camino de aumento en cada iteración
$maxff$	Número máximo de intentos para aplicar el algoritmo completo

Variables de entrada

$conv$, $prec$, $repc$

Variables de salida

$ltar$, $repc$

Variables internas

$aant_i$	Arco anterior al nodo i en el camino de aumento
$aumf$	Cantidad en la que se ha de aumentar el flujo en una iteración
$capr_{i,i'}$	Capacidad residual del arco que va del nodo i al nodo i'
$capt_{i,i'}$	Capacidad total del arco que va del nodo i al nodo i'

$delta_i$	Etiqueta asignada al nodo i, máxima cantidad en la que se puede aumentar el flujo teniendo en cuenta el camino de aumento hasta ese nodo
$etiq_i$	Variable binaria que indica si el nodo i está etiquetado (1 sí; 0 no)
$incp$	Variable binaria que indica para una iteración concreta que algún nodo no puede ser etiquetado por incompatibilidad de precedencias (1 sí; 0 no)
$intff$	Número de intentos realizados en la aplicación completa del algoritmo
$inut_i$	Variable binaria que indica si el nodo i es inútil (1 sí; 0 no). Cuando un nodo es inútil, no puede ser etiquetado
$itet$	Número de intentos realizados para obtener un camino de aumento en cada iteración
$flujo_{i,i'}$	Flujo a través del arco que va del nodo i al nodo i'
$nant_i$	Nodo anterior al nodo i en el camino de aumento
$ngan$	Nodo seleccionado para etiquetar
$npar$	Nodo de referencia (desde el que se van a buscar candidatos para etiquetar o desde el que se va a actualizar el flujo)
$valf$	Valor del flujo

Algoritmo

PASO 0

- Indicar que no se ha repartido lo planeado

$$repc = 0$$

- Inicializar las cantidades de material que transportan los convoyes

$$ltar_k = 0 \quad \forall k$$

- Calcular las capacidades totales de los arcos que unen la fuente, f, con los depósitos

$$capt_{f,i} = \text{mín}\{qglobal, qav_i\} \quad \forall i \in IO$$

- Calcular las capacidades totales de los arcos que unen los nodos de demanda con el sumidero, s

$$capt_{i,s} = \text{mín}\{qglobal, dem_i\} \quad \forall i \in IO$$

- Calcular las capacidades totales de los arcos transitados

$$capt_{i,i'} = \text{mín}\left\{qglobal, \sum_{j \in conv_k} cap_{tipv_j}\right\} \quad \forall k \ / \ conv_k \neq \varnothing$$
$$\text{siendo } i = ini_k, i' = fin_k$$

- Hacer inútiles los nodos (no de demanda) desde los que no puede salir flujo. No es necesario, pero puede ahorrar operaciones

$$\forall i \in I - (ID) \ / \ \sum_{i'} capt_{i,i'} = 0 \quad \Rightarrow \quad inut_i = 1$$

- Inicializar el número de intentos realizados de aplicación completa del algoritmo

$$intff = 1$$

PASO 1

- Inicializar el flujo entre cada par de nodos i e i' (incluyendo la fuente y el sumidero) y el valor del flujo

$$flujo_{i,i'} = 0 \quad \forall i, i' \in I \cup \{f, s\}$$
$$valf = 0$$

- Inicializar las capacidades residuales. Deben coincidir con las capacidades totales

$$capr_{i,i'} = capt_{i,i'} \quad \forall i, i' \in I \cup \{f, s\}$$

- Inicializar el número de intentos realizados en la búsqueda de un camino de aumento

$$itet = 1$$

- Indicar que no hay problemas de precedencias

$$incp = 0$$

PASO 2

- Borrar etiquetas, reiniciar variables auxiliares

$$etiq_i = 0, delta_i = qglobal, nant_i = f \quad \forall i \in I \cup \{f, s\}$$

- Etiquetar la fuente y tomarla como nodo de referencia

$$etiq_f = 1, npar = f$$

PASO 3

- Identificar todos los nodos candidatos a etiquetar desde el nodo de referencia, $npar$. Cada nodo candidato i lleva aparejado un arco asociado, $k = (npar, i)$, que lo une con el nodo de referencia. Para que un nodo i sea candidato debe cumplir:
  a. No está etiquetado, $etiq_i = 0$
  b. Su arco asociado tiene capacidad residual positiva, $capr_{npar,i} > 0$
  c. No es un nodo inútil, $inut_i = 0$
  d. Su arco asociado no precede al arco por el que se accedió al nodo de referencia, $prec_{k,aant_{npar}} = 0$. Esta condición no se comprueba cuando el nodo de referencia es la fuente, $npar = f$, o cuando el presumible candidato es el sumidero, $i = s$

Si se cumplen las tres primeras condiciones pero no la cuarta, indicar que hay problemas de precedencia

$$incp = 1$$

- Si hay candidatos:
  - Seleccionar al azar un nodo candidato, $ngan$, etiquetarlo, calcular su máximo aumento permitido de flujo y fijar su nodo anterior y su arco anterior

$$\begin{gathered} etiq_{ngan} = 1 \\ delta_{ngan} = \text{mín}\{delta_{npar}, capr_{npar,ngan}\} \\ nant_{ngan} = npar \\ aant_{ngan} = k \quad \text{siendo } npar = ini_k, ngan = fin_k \end{gathered}$$

  - Tomar el nodo seleccionado como nodo de referencia

$$npar = ngan$$

  - Si el nodo seleccionado es el sumidero, $ngan = s$, se ha encontrado un camino de aumento. Ir al PASO 4 (actualizar flujo)
  - (El nodo seleccionado no es el sumidero, $ngan \neq s$). Ir al PASO 3 (búsqueda de candidatos para etiquetar)
- (No hay candidatos)
  - Si el nodo de referencia no es la fuente, $npar \neq f$:
    - Tomar como nuevo nodo de referencia el nodo anterior al actual nodo de referencia

$$npar = npar_{ngan}$$

    - Ir al PASO 3 (búsqueda de candidatos para etiquetar)
  - (El nodo de referencia es la fuente, $npar = f$). Si no hay problemas de precedencia, $prec = 0$, FIN del procedimiento
  - (Hay problemas de precedencia). Si se han agotado los intentos para encontrar un camino de aumento, $itet = maxet$:
    - Si se han agotado los intentos para aplicar el algoritmo completo, $intff = maxff$, FIN del procedimiento
    - Actualizar el número de intentos de aplicación del algoritmo completo

$$intff = intff + 1$$

    - Volver al PASO 1 (aplicación del algoritmo desde el principio)
  - (No se han agotado los intentos para encontrar un camino de aumento, $itet \neq maxet$). Actualizar el número de intentos para encontrar un camino de aumento

$$itet = itet + 1$$

  - Volver al PASO 2 (búsqueda de un nuevo camino de aumento)

PASO 4

- Calcular la cantidad en la que se va a aumentar el flujo, $aumf$. Se tiene en cuenta que el valor del flujo, $valf$, no debe superar la cantidad a repartir, $qglobal$

$$aumf = \text{mín}\{delta_s, qglobal - valf\}$$

- Aumentar el flujo y disminuir la capacidad residual en los arcos del camino de aumento. Hacer hasta que el nodo de referencia sea la fuente, $npar = f$:

$$\begin{gathered} i = nant_{npar}, i' = npar \\ flujo_{i,i'} = flujo_{i,i'} + aumf \\ capr_{i,i'} = capr_{i,i'} - aumf \\ npar = nant_{npar} \end{gathered}$$

- Actualizar el valor del flujo añadiéndole la cantidad de aumento de flujo

$$valf = valf + aumf$$

- Si se ha repartido lo planeado, $valf = qglobal$:
  - Calcular las cargas de los convoyes por los arcos

$$ltar_k = flujo_{i,i'} \quad \text{siendo } i = ini_k, i' = fin_k \quad \forall k$$

  - Indicar que se ha repartido lo planeado

$$repc = 1$$

  - FIN del procedimiento
- (No se ha repartido lo planeado, $valf \neq qglobal$). Volver al PASO 2 (búsqueda de un nuevo camino de aumento)

El diagrama de flujo que aparece en la Figura 3.4 resume el procedimiento ENVIAR FLUJO.

En la Figura 3.5 aparece el flujo de material sobre la red del ejemplo representado en la Figura 3.1. Sobre cada convoy, formado en el procedimiento CONSTRUIR RUTAS y reflejado en la Figura 3.3, se puede ver el número total de toneladas que son transportadas por cada arco, aunque todavía no se ha decidido cómo repartir la carga entre los vehículos que forman los convoyes. Asimismo puede apreciarse el *stock* que terminaría en los nodos. En este caso, han sido transportadas 14 de las 15 toneladas disponibles en el depósito a y 11 de las 15 toneladas disponibles en el depósito b. Este material ha servido para satisfacer plenamente las 10 toneladas que demandaba el nodo c, 13 de las 15 que demandaba el nodo f y 2 de las 15 que demandaba el nodo g.

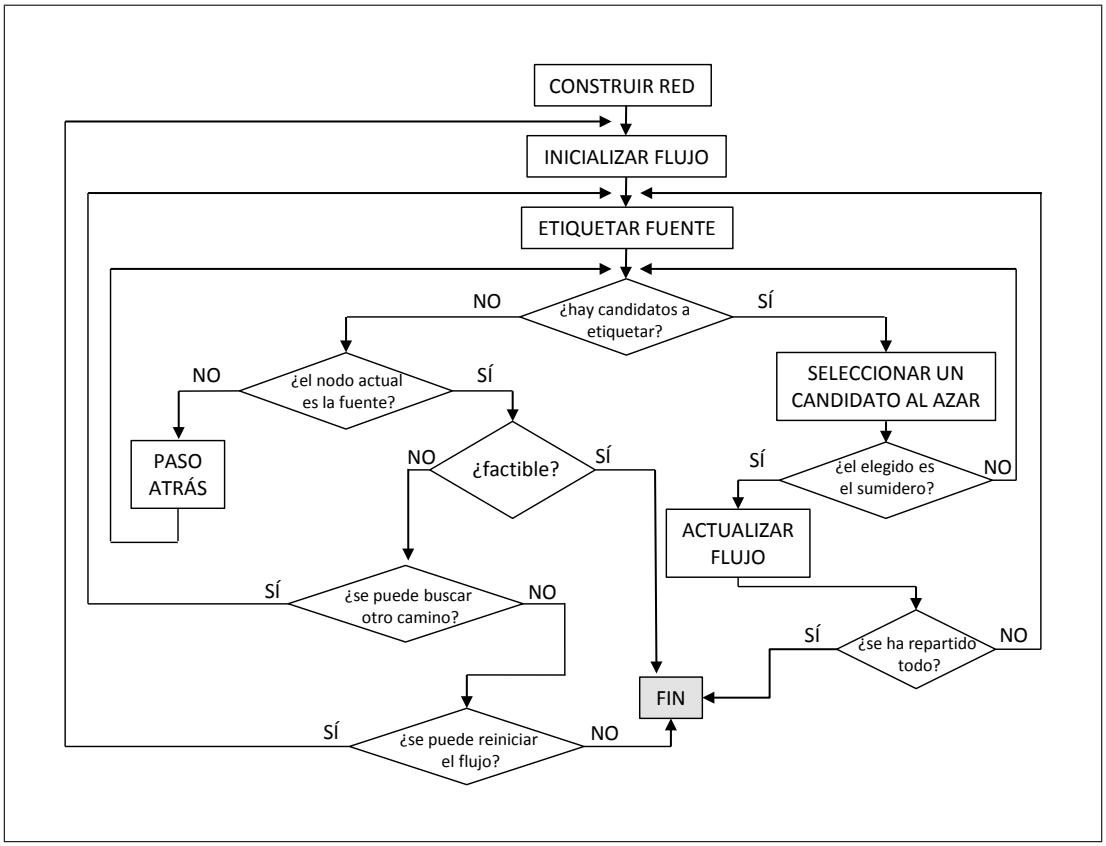


Figura 3.4: Diagrama de flujo de ENVIAR FLUJO

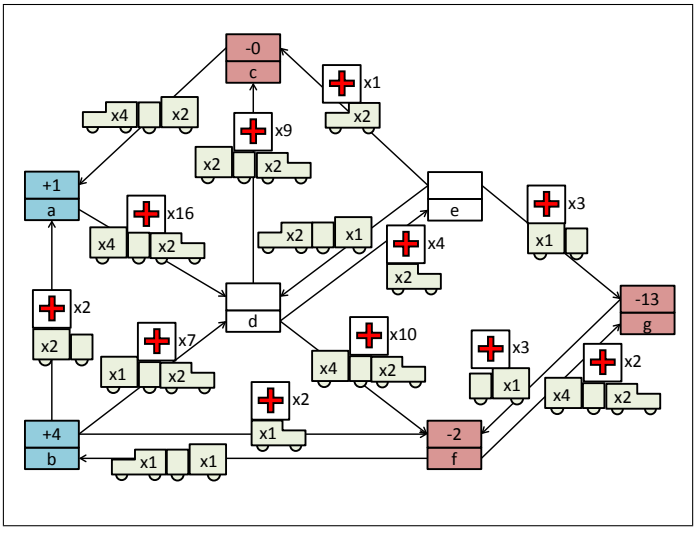


Figura 3.5: Solución parcial tras ENVIAR FLUJO

Procedimiento 13. - ORDENAR EVENTOS

Ordenación de las listas de eventos en los nodos según los instantes en los que se producen y cálculo de los *stocks* disponibles tras los eventos. Un evento en un nodo corresponde a la llegada o salida de un convoy. Este procedimiento es necesario, además de para controlar la cantidad de material existente en cada momento en los nodos, para sincronizar correctamente los convoyes.

Este procedimiento también será ejecutado desde el Algoritmo de MEJORA descrito en el siguiente capítulo, ya que será necesario aplicarlo cada vez que se modifique algún elemento de la solución.

Variables de entrada

$into$, $ltar$, $nuan$, $tmin$

Variables de salida

$inst$, $stck$

Algoritmo

- Calcular los instantes en los que se producen eventos en cada nodo. Si el convoy llega al nodo, es su tiempo de finalización; si el convoy sale del nodo, es su tiempo de finalización menos la duración del trayecto

$$inst_{i,e} = \begin{cases} tmin_k & \text{si } k = into_{i,e}, i = fin_k \\ tmin_k - dur_k & \text{si } k = into_{i,e}, i = ini_k \end{cases} \quad \forall i,\ \forall e \in \{1, \ldots, nuan_i\}$$

- Ordenar de menor a mayor la lista de instantes de cada nodo, $inst_i$. Cuando varios eventos se producen al mismo tiempo, los eventos correspondientes a convoyes que llegan al nodo se colocan antes en la lista que los eventos correspondientes a convoyes que salen del nodo. Adaptar el orden de la lista de incidencia usada de cada nodo, $into_i$ al orden obtenido en la lista de instantes

$$\left.\begin{array}{r} inst_{i,e} < inst_{i,e'} \\ inst_{i,e} = inst_{i,e'} \\ k = into_{i,e}, i = fin_k \\ k' = into_{i,e'}, i = ini_{k'} \end{array}\right\} \begin{array}{ll} \Rightarrow & e < e' \\ \Rightarrow & e < e' \end{array} \quad \forall i,\ \forall e, e' \in \{1, \ldots, nuan_i\}$$

- Calcular los *stocks* disponibles en los nodos tras los eventos. En el evento 0, el *stock* es la cantidad disponible inicialmente. En el resto de eventos, si el convoy llega se suma el material que lleva al *stock* disponible tras el evento anterior, si el convoy sale se resta el material que lleva al *stock* disponible tras el evento anterior

$$stck_{i,e} = \begin{cases} qav_i & \text{si } e = 0 \\ stck_{i,e-1} + ltar_k & \text{si } e \geq 1, k = into_{i,e}, i = fin_k \\ stck_{i,e-1} - ltar_k & \text{si } e \geq 1, k = into_{i,e}, i = ini_k \end{cases}$$
$$\forall i,\ \forall e \in \{0, \ldots, nuan_i\}$$

Procedimiento 14. - BUSCAR NEGATIVOS

Detección de eventuales *stocks* negativos e introducción de precedencias para intentar solventar la incoherencia producida. Tal y como se realiza la construcción de las rutas y el transporte de ayuda, es posible que un convoy parta de un nodo con material que todavía no haya llegado a dicho nodo. Una situación así produciría la aparición de *stocks* negativos en la secuencia temporal de *stocks* del nodo. La búsqueda de *stocks* negativos se realiza ordenadamente por nodos y en cada nodo por etapas. El primer evento que produce un *stock* negativo en un nodo corresponde obligatoriamente a un convoy que sale del nodo. Para intentar paliar el problema se busca en dicho nodo un evento posterior correspondiente a un convoy que llega al nodo transportando material y se obliga a que el arco entrante preceda al arco saliente o, si no es posible por existir ya relación de precedencia entre ambos arcos, a otro arco saliente anterior al evento en el que se produce el *stock* negativo.

Variables de entrada

cont, *conv*, *into*, *ltar*, *nuan*, *nump*, *prec*, *pred*, *prei*, *pret*, *stck*, *suci*, *suct*

Variables de salida

camb Variable binaria que indica si se puede hacer algún cambio en las secuencias de eventos de los nodos con el fin de evitar *stocks* negativos (1 sí; 0 no)

cont, *conv*, *negat*, *nump*, *prec*, *pred*, *prei*, *pret*, *suci*, *suct*

Variables internas

apos, *apre*

Procedimientos llamados

AGREGAR PRECEDENCIAS

Algoritmo

PASO 0

- Inicializar los indicadores de *stock* negativo y de posible arreglo

$$negat = 0$$
$$camb = 0$$

- Buscar el primer nodo i y el primer evento $e \in \{1, \ldots, nuan_i\}$ que produzcan un *stock* negativo, $stck_{i,e} < 0$
- Si no existe tal *stock* negativo, FIN del Procedimiento

- El primer *stock* negativo encontrado, $stck_{i,e} < 0$, con seguridad corresponde a un convoy que sale del nodo por un arco $k = into_{i,e}$, $i = ini_k$
- Indicar que hay algún *stock* negativo

$$negat = 1$$

PASO 1

- Buscar un evento posterior $e' > e$ del nodo i correspondiente a un convoy que llega al nodo transportando material

$$k' = into_{i,e'}, i = fin_{k'}, ltar_{k'} > 0$$

- Si no quedan eventos con estas características, FIN del procedimiento
- Si no existe relación de precedencia entre los arcos k y k', $prec_{k,k'} = 0$:
  - Guardar los arcos k' y k como predecesor y precedido

$$apre = k'$$
$$apos = k$$

  - Ejecutar el procedimiento AGREGAR PRECEDENCIAS, para obligar a que el arco *apre* preceda inmediatamente al arco *apos* y actualizar las precedencias correspondientes
  - Indicar que hay un posible arreglo para el *stock* negativo

$$camb = 1$$

  - FIN del procedimiento
- (Existe relación de precedencia entre los arcos k y k', $prec_{k,k'} = 1$):

PASO 2

- Buscar un evento anterior $e'' < e$ del nodo i correspondiente a un convoy que sale del nodo transportando material

$$k'' = into_{i,e''}, i = ini_{k''}, ltar_{k''} > 0$$

- Si no quedan eventos e'' con estas características, desechar el evento e' y volver al PASO 1
- Si no existe relación de precedencia entre los arcos k'' y k', $prec_{k'',k'} = 0$:
  - Guardar los arcos k' y k'' como predecesor y precedido

$$apre = k'$$
$$apos = k''$$

- Ejecutar el procedimiento AGREGAR PRECEDENCIAS, para obligar a que el arco *apre* preceda inmediatamente al arco *apos* y actualizar las precedencias correspondientes
- Indicar que hay un posible arreglo para el *stock* negativo

$$camb = 1$$

- FIN del procedimiento

- (Existe relación de precedencia entre los arcos k'' y k', $prec_{k'',k'} = 1$). Desechar el evento e'' y volver al PASO 2

Procedimiento 15. - POSITIVAR

Determinación de *stocks* negativos y arreglo de los mismos. Un *stock* negativo se produce cuando en un nodo queda una cantidad negativa de material tras la salida de un convoy. Esta situación es claramente infactible y ha de impedirse reorganizando los instantes en que los convoyes parten del nodo conflictivo o llegan a él. El procedimiento consiste en repetir la secuencia BUSCAR NEGATIVOS - FIN ARCOS - ORDENAR EVENTOS hasta que no existan *stocks* negativos. No siempre es posible evitar la presencia de *stocks* negativos para una secuenciación de rutas concreta. En ese caso, desde el programa principal, se desecharán las rutas actuales y se construirán nuevas rutas.

Variables de entrada

cont, *conv*, *dur*, *into*, *ltar*, *nuan*, *nump*, *prec*, *pred*, *prei*, *pret*, *stck*, *suci*, *suct*, *tamc*, *tmin*

Variables de salida

cont, *conv*, *inst*, *into*, *negat*, *nump*, *prec*, *pred*, *prei*, *pret*, *stck*, *suci*, *suct*, *tmin*

Variables internas

camb

Procedimientos llamados

BUSCAR NEGATIVOS, FIN ARCOS, ORDENAR EVENTOS

Algoritmo

PASO 0

- Ejecutar el procedimiento BUSCAR NEGATIVOS, para buscar *stocks* negativos y determinar si existe arreglo
- Si no hay arreglo (posiblemente porque no hay *stocks* negativos), $camb = 0$, FIN del procedimiento

- Ejecutar el procedimiento FIN ARCOS, para calcular los instantes mínimos de finalización de los trayectos
- Ejecutar el procedimiento para ORDENAR EVENTOS, ordenar los eventos en los nodos según sus instantes de ocurrencia y calcular los *stocks* correspondientes
- Volver al PASO 0

Después de aplicar ORDENAR EVENTOS al ejemplo que se está analizando, los primeros eventos asociados al nodo e correspondían a la salida de los convoyes hacia los nodos c y g, con 1 y 3 lotes de material respectivamente (ver Figura 3.5). Sin embargo, aunque cuatro vehículos tenían su origen en el nodo e (ver Figura 3.1), inicialmente no existía material en ese nodo, por lo que los convoyes mencionados dejan un saldo de material de -4 lotes tras su partida. Con el procedimiento POSITIVAR, se logra eliminar este *stock* negativo impidiendo que dichos convoyes puedan iniciar su marcha hasta que llegue el convoy procedente del nodo d, que transporta los 4 lotes necesarios para surtir a los convoyes que parten hacia los nodos c y g.

Se debe recalcar que este procedimiento no altera la composición de los convoyes ni las cantidades transportadas, solo los instantes en los que se alcanzan los nodos, por lo que los convoyes y cargas que aparecen en la Figura 3.5 son definitivos. En el capítulo siguiente se verá cómo se ve afectada esta solución tras aplicarle distintas técnicas de mejora.

Procedimiento 16. - REPARTIR CARGAS

Reparto del material transportado por cada arco entre los vehículos del convoy correspondiente. Hasta el momento sólo se había determinado la cantidad total de material que transporta cada convoy, pues no había sido necesaria información más precisa acerca de cómo se distribuía el material entre los vehículos que lo forman. Nótese que se deben cargar primero los vehículos de menor coste variable de transporte.

Variables de entrada

conv, *ltar*, *secv*

Variables de salida

load

Variables internas

lisv	Lista con los vehículos del convoy del arco actual que todavía no han recibido carga, ordenados de menor a mayor coste variable de transporte
lres	Material que todavía no ha sido repartido entre los vehículos que recorren el arco actual

Algoritmo

- Tomar el primer arco k por el que se transporte material, $ltar_k > 0$

  PASO 0

  - Si no quedan más arcos, FIN del procedimiento
  - Inicializar la cantidad de material que queda por repartir en el arco k

    $$lres = ltar_k$$

  - Inicializar la lista con los vehículos del convoy del arco k que todavía no han recibido carga

    $$lisv = conv_k$$

  - Ordenar la lista $lisv$ de menor a mayor coste variable de transporte. Los empates se deshacen al azar (o arbitrariamente)

  PASO 1

  - Elegir el primer vehículo j de la lista $lisv$. Si no quedan vehículos, $lisv = \varnothing$, pasar al siguiente arco k (si queda alguno) y volver al PASO 0
  - Si en el vehículo actual cabe todo el material restante, $cap_{tipv_j} \geq lres$:
    - Llenar el vehículo j con el material restante

      $$load_{j,e} = lres \quad \text{siendo } k = secv_{j,e}$$

    - Pasar al siguiente arco k (si queda alguno)
    - Volver al PASO 0
  - (En el vehículo actual no cabe todo el material restante, $cap_{tipv_j} < lres$):
    - Llenar del todo el vehículo j

      $$load_{j,e} = cap_{tipv_j} \quad \text{siendo } k = secv_{j,e}$$

    - Actualizar la cantidad de material restante

      $$lres = lres - cap_{tipv_j}$$

    - Eliminar el vehículo j de la lista de vehículos que todavía no han recibido carga, $lisv$
    - Volver al PASO 1

3.3. Programa principal

En esta sección se presenta el procedimiento principal del algoritmo, que se denominará CONSTRUCTIVO. Sirve para organizar el resto de procedimientos y, a su vez, será utilizado, tras realizarse algunas adaptaciones necesarias, como parte de los algoritmos basados en metaheurísticas que se describirán en posteriores capítulos.

Procedimiento 17. - CONSTRUCTIVO

Procedimientos llamados

CONSTRUIR RUTAS, DURACIÓN, ENVIAR FLUJO, FIN ARCOS, ORDENAR EVENTOS, POSITIVAR, PREPROCESO, REPARTIR CARGAS

Algoritmo

PASO 0

- Ejecutar el procedimiento PREPROCESO, para realizar las operaciones previas

PASO 1

- Ejecutar el procedimiento CONSTRUIR RUTAS, para diseñar los itinerarios de los vehículos
- Ejecutar el procedimiento DURACIÓN, para calcular las duraciones de los trayectos de los convoyes
- Ejecutar el procedimiento FIN ARCOS, para calcular los instantes mínimos de finalización de los trayectos
- Ejecutar el procedimiento ENVIAR FLUJO, para obtener las cantidades de material transportadas por los convoyes
- Si no se ha repartido lo planeado, $repc = 0$, volver al PASO 1 (construir nuevas rutas)
- Ejecutar el procedimiento ORDENAR EVENTOS, para ordenar los eventos en los nodos según sus instantes de ocurrencia y calcular los *stocks* correspondientes

PASO 2

- Ejecutar el procedimiento POSITIVAR, para detectar y eliminar los posibles *stocks* negativos
- Si hay algún *stock* negativo, $negat = 1$, volver al PASO 1 (construir nuevas rutas)
- (No hay ningún *stock* negativo, $negat = 0$). Ejecutar el procedimiento REPARTIR CARGAS, para repartir el material que se transporta por los arcos entre los vehículos de los convoyes correspondientes
- Evaluar la función objetivo de acuerdo a los pesos establecidos para los atributos

$$fob = \sum_{g} \frac{peso_g \cdot f_g}{cota_g}$$

El Algoritmo Constructivo completo puede sintetizarse en el diagrama de flujo que se muestra en la Figura 3.6.

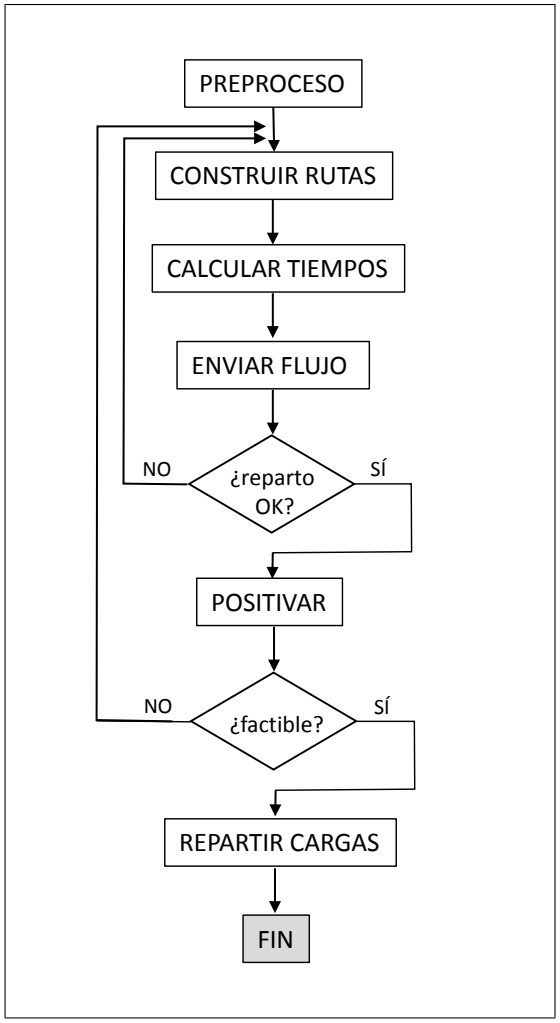


Figura 3.6: Diagrama de flujo de CONSTRUCTIVO

Capítulo 4

Algoritmo de Mejora

Una vez desarrollado un método que proporciona soluciones factibles para el problema de reparto que se estudia en este trabajo, como es el Algoritmo Constructivo presentado en el capítulo anterior, es conveniente manipular esas soluciones factibles para mejorarlas en la medida de lo posible. En este capítulo se describe un algoritmo con ese propósito. El algoritmo toma como entrada una solución factible, a través de los valores de las variables que la conforman, e intenta proporcionar otra solución con menor valor en la función objetivo. Como normalmente se aplicará a las soluciones obtenidas mediante el Algoritmo Constructivo (en cualquiera de sus versiones, pues más adelante se adaptará para servir de base de metaheurísticas), se supondrá que se conoce el valor de todos los parámetros derivados y todas las variables empleadas en dicho algoritmo. En el caso de que no se disponga de algunos de esos valores, sería necesario calcularlos previamente.

El procedimiento REFINAR SOLUCIÓN es el más importante del algoritmo, y consta de varias fases centradas en mejorar en diversos aspectos la solución. Primeramente se impide que cierta cantidad de material abandone nodos de demanda donde no se haya repartido la cantidad idealmente justa (RETENER MATERIAL), mejorando de esta manera la medida de la equidad y también el coste. A continuación, se elimina flujo de material que pasa por depósitos donde exista material sobrante tras el reparto (APROVECHAR DEPÓSITOS), mejorando de esta forma, al menos, el coste de la operación. Ambos procedimientos requieren de sendas adaptaciones del algoritmo de Ford-Fulkerson, al igual que ocurría en el procedimiento ENVIAR FLUJO del Algoritmo Constructivo. Por otro lado, como en dicho algoritmo las rutas de los vehículos son construidas alargándolas todo lo posible para facilitar el posterior reparto, una vez conocido el flujo de material sobre la red, parte de los trayectos pueden eliminarse (ACORTAR RUTAS), incluyendo los circuitos innecesarios de los vehículos. De esta manera se mejora, con certeza, el coste total del reparto y, en muchos casos, también su duración y las medidas de seguridad y fiabilidad. Por último, se realizan intercambios entre vehículos del mismo tipo con el fin de seguir eliminando trayectos (SUSTITUIR VEHÍCULOS) o de disminuir la suma de distancias de regreso y por tanto el coste total del reparto (INTERCAMBIAR VEHÍCULOS). Tras el procedimiento REFINAR SOLUCIÓN, se deben actualizar las relaciones de precedencia, reorganizar los eventos (ACTUALIZAR EVENTOS) y eliminar de nuevo los *stocks* negativos que eventualmente hayan podido aparecer (POSITIVAR).

A continuación, se realiza otra posible mejora de la solución, en este caso evitando que se transporte material por arcos que conectan los mismos dos nodos pero en sentidos opuestos (EVITAR CRUCES). No está asegurado que la solución vaya a mejorar con la realización de esta operación, incluso es posible que deje de ser factible. Por ello, solo se implementarán las modificaciones en caso de que la solución mejore y se mantenga su factibilidad. Si mediante el procedimiento EVITAR CRUCES se modifica la solución, se repite REFINAR SOLUCIÓN con el objeto de seguir mejorando la solución si esto es posible.

El diseño de los procedimientos que componen el Algoritmo de Mejora se ha realizado con la intención de mejorar al menos uno de los objetivos sin empeorar los demás, aunque pueden presentarse algunas pequeñas excepciones que se detallarán cuando se expongan los procedimientos a los que afectan. Se seguirá usando el ejemplo de la Figura 3.1, introducido en el capítulo anterior, para explicar cómo se modifica la solución allí obtenida (ver Figura 3.5) tras algunos de los procedimientos del Algoritmo de Mejora.

4.1. Parámetros y variables

En esta sección se definen los nuevos elementos que intervienen en el Algoritmo de Mejora y se enumeran los elementos necesarios ya definidos con anterioridad.

Parámetros de lectura directa

cap, $comp$, $cosf$, $cosv$, dem, $dist$, fin, ini, $nivp$, ori, p, pm, $peso$, qav, $qglobal$, r, $tdis$, $tipv$, $vela$, $velv$

Parámetros derivados

$pjus$	Proporción de demanda satisfecha óptima

$clas$, $cota$, $creg$, $ctip$, $dreg$, $grae$, $gras$, $ince$, $incs$, sig, $treg$

Variables de decisión

$load$, $secv$, $tmin$

Variables objetivo

f, fob

Variables auxiliares

$actpr$	Variable binaria que indica si hay que actualizar las precedencias (1 sí; 0 no)

$evcr$	Variable binaria que indica si se ha evitado algún cruce de material (1 sí; 0 no)
pf_g	Valor del objetivo g en la solución original
$pfob$	Valor de la función objetivo global en la solución original
$pload_j$	Secuencia de cantidades transportadas por el vehículo j hasta que termina su reparto en la solución original
$psecv_j$	Secuencia de arcos (ruta) que recorre el vehículo j hasta que termina su reparto en la solución original
$ptmin_k$	Instante mínimo en el que el convoy que recorre el arco k finaliza su trayecto en la solución original

$cont$, $conv$, dur, $inst$, $into$, $lonv$, $ltar$, $negat$, $nuan$, $nump$, $prec$, $pred$, $prei$, $pret$, sf, $stck$, $suci$, $suct$, $tamc$, $util$

4.2. Procedimientos

En esta sección se presentan los procedimientos que forman parte del Algoritmo de Mejora. En algunos de estos procedimientos se ejecutan procedimientos expuestos anteriormente, en este caso, en el Algoritmo Constructivo desarrollado en el Capítulo 3.

Procedimiento 18. - RETENER MATERIAL

Retención de material de ayuda en los nodos de demanda necesitados. En un reparto equitativamente óptimo todos los nodos de demanda verían satisfecha su demanda en la misma proporción, que en esta memoria se denomina proporción de demanda satisfecha óptima, $pjus$. Se llaman nodos necesitados a aquellos nodos de demanda donde la proporción de demanda satisfecha es inferior a la proporción de demanda satisfecha óptima. El resto de nodos de demanda se llaman no necesitados. Se desea conseguir que parte del material que pasa en su recorrido por algún nodo de demanda necesitado y termina su recorrido en un nodo de demanda no necesitado, no abandone el nodo necesitado, contribuyendo así a mejorar la equidad en el reparto sin empeorar otros objetivos. El único objetivo que podría empeorar sería el objetivo basado en la prioridad, pero los atributos equidad y prioridad no suelen considerarse juntos al resolver un problema. En caso de que se considere la prioridad como criterio de optimización, puede omitirse este procedimiento o bien modificarlo de forma que los nodos prioritarios no puedan considerarse como no necesitados, impidiendo de esta forma que disminuya la cantidad de material que llega a dichos nodos prioritarios.

Para conseguir esta redistribución del material de ayuda, se procede de forma similar al procedimiento ENVIAR FLUJO, adaptando el algoritmo de Ford-Fulkerson. Se crea una red de transporte a partir del grafo dirigido original formada por todos los nodos y los arcos transitados por algún vehículo. La capacidad de cada arco es la cantidad de material que se transporta por él. Se crean dos nodos artificiales: la fuente, f, del que parten arcos ficticios a los nodos de demanda necesitados de los que sale material, y el sumidero, s, al que llegan arcos ficticios desde los nodos de demanda no necesitados. La capacidad de cada arco ficticio que parte de la fuente es el mínimo entre la cantidad de material que

sale del nodo necesitado y la diferencia entre su demanda y su *stock* final, mientras que la capacidad de cada arco ficticio que llega al sumidero es la cantidad de material sobrante respecto de la cantidad equitativamente óptima en el nodo no necesitado. En este caso, la adaptación del algoritmo de Ford-Fulkerson presenta las siguientes características:

- El flujo entre dos nodos se interpreta como la cantidad de material a eliminar del arco correspondiente.
- La elección de los nodos a etiquetar se realiza al azar.
- No se permiten movimientos de retroceso.
- Se deben respetar y actualizar las secuencias de *stocks* en los nodos.
- Se permiten varios intentos para obtener un camino de aumento en cada iteración.
- El algoritmo finaliza cuando se acaban los intentos para obtener un camino de aumento en una iteración.

Variables de entrada

$ltar$, $nuan$, sf, $stck$

Variables de salida

$ltar$

Variables internas

$aumf$	Cantidad en la que se ha de aumentar el flujo en una iteración
$capr_{i,i'}$	Capacidad residual del arco que va del nodo i al nodo i'
$capt_{i,i'}$	Capacidad total del arco que va del nodo i al nodo i'
$delmx$	Cota superior para las eventuales etiquetas de los nodos alcanzables desde el nodo actual en el proceso de etiquetado
$delta_i$	Etiqueta asignada al nodo i, máxima cantidad en la que se puede aumentar el flujo teniendo en cuenta el camino de aumento hasta ese nodo
$etiq_i$	Variable binaria que indica si el nodo i está etiquetado (1 sí; 0 no)
$flujo_{i,i'}$	Flujo a través del arco que va del nodo i al nodo i'
$flumx_{i,i'}$	Cota superior para el aumento de flujo a través del arco que va del nodo i al nodo i' en la iteración actual
$itet$	Número de intentos realizados para obtener un camino de aumento en cada iteración
$lcan$	Lista de nodos candidatos para etiquetar
$lnec$	Lista de nodos necesitados en los que se puede retener material
$lsob$	Lista de nodos no necesitados de los que se puede quitar material
$nant_i$	Nodo anterior al nodo i en el camino de aumento
$nnec$	Número de nodos necesitados en los que se puede retener material

$npar$	Nodo de referencia (desde el que se van a buscar candidatos para etiquetar o desde el que se va a actualizar el flujo)
$nuevo$	Variable binaria que indica si el nodo de referencia para etiquetar es nuevo en la iteración actual (1 sí; 0 no)
$pos1_i$	Posición que ocupa el último arco utilizado para etiquetar en la lista de eventos de su nodo final i
$pos2_i$	Posición que ocupa el último arco utilizado para etiquetar en la lista de eventos de su nodo inicial i
ult	Último nodo (sin contar el sumidero) en el camino de aumento
$valf$	Valor del flujo

Parámetros internos

$maxet$	Número máximo de intentos para obtener un camino de aumento en cada iteración

Algoritmo

PASO 0

- Crear la lista de nodos necesitados en los que se puede retener material, $lnec$

$$i \in lnec \quad \Leftrightarrow \quad \frac{sf_i}{dem_i} < pjus, \sum_{k/ini_k=i} ltar_k > 0$$

- Si no hay ningún nodo con estas características, FIN del procedimiento
- Calcular las capacidades totales de los arcos que unen la fuente, f, con los nodos necesitados

$$capt_{f,i} = \text{mín} \left\{ \sum_{k/ini_k=i} ltar_k, dem_i - sf_i \right\} \quad \forall i \in lnec$$

- Crear la lista de nodos no necesitados de los que se puede quitar material, $lsob$

$$i \in lsob \quad \Leftrightarrow \quad \frac{sf_i}{dem_i} > pjus, \sum_{k/fin_k=i} ltar_k > 0$$

- Calcular las capacidades totales de los arcos que unen los nodos no necesitados con el sumidero, s

$$capt_{i,s} = sf_i - dem_i \cdot pjus \quad \forall i \in lsob$$

- Calcular las capacidades totales de los arcos transitados

$$capt_{i,i'} = ltar_k \quad \forall k \ / \ conv_k \neq \varnothing \quad \text{siendo } i = ini_k, i' = fin_k$$

PASO 1

- Inicializar el flujo entre cada par de nodos i e i' (incluyendo la fuente y el sumidero) y el valor del flujo

$$flujo_{i,i'} = 0 \quad \forall i, i' \in I \cup \{f, s\}$$
$$valf = 0$$

- Inicializar las capacidades residuales. Deben coincidir con las capacidades totales

$$capr_{i,i'} = capt_{i,i'} \quad \forall i, i' \in I \cup \{f, s\}$$

- Inicializar el número de intentos realizados en la búsqueda de un camino de aumento

$$itet = 1$$

PASO 2

- Borrar etiquetas, reiniciar variables auxiliares

$$etiq_i = 0, delta_i = qglobal, nant_i = f \quad \forall i \in I \cup \{f, s\}$$

- Etiquetar la fuente y tomarla como nodo de referencia

$$etiq_f = 1, npar = f$$

- Indicar que el nodo de referencia es nuevo en la presente iteración

$$nuevo = 1$$

- Inicializar las cotas superiores para el aumento de flujo en la iteración actual

$$flumx_{i,i'} = capr_{i,i'} \quad \forall i, i' \in I \cup \{f, s\}$$

PASO 3

- Si el nodo de referencia es la fuente, $npar = f$, o el nodo anterior al de referencia es la fuente, $nant_{npar} = f$:
  - Crear la lista de nodos candidatos a etiquetar, *lcan*, revisando los nodos alcanzables desde el nodo de referencia. Para que un nodo i sea candidato debe cumplir:
    a. No está etiquetado, $etiq_i = 0$
    b. Su arco asociado tiene capacidad residual positiva, $capr_{npar,i} > 0$
  - Ir al PASO 4 (conteo de candidatos)

- (Ni el nodo de referencia ni su anterior son la fuente, $npar \neq f$, $nant_{npar} \neq f$). Si el nodo de referencia es nuevo para esta iteración, $nuevo = 1$, fijar la cota superior para las eventuales etiquetas de los nodos alcanzables desde el nodo de referencia como la etiqueta del nodo de referencia

$$delmx = del_{npar}$$

- Revisar todos los arcos usados, k, que salen del nodo de referencia, $npar$, a partir del evento correspondiente al arco desde el que se etiquetó el nodo de referencia, $pos1_{npar}$, y por orden de ocurrencia. Cada arco k lleva asociado un nodo $i = fin_k$ y un evento e tal que $k = into_{npar,e}$:
  - Añadir el nodo i a la lista de candidatos, $lcan$, si cumple:
    a. No está etiquetado, $etiq_i = 0$
    b. Su arco asociado tiene capacidad residual positiva, $capr_{npar,i} > 0$
  - Si el nodo de referencia es nuevo para esta iteración, $nuevo = 1$, y el nodo i ha sido incluido en la lista de candidatos, $i \in lcan$, actualizar el aumento máximo de flujo entre los nodos $npar$ e i

$$flumx_{npar,i} = \text{mín}\{delmx, flumx_{npar,i}\}$$

  - Si no queda *stock* tras el evento, $stck_{npar,e} = 0$, ir al PASO 4 (conteo de candidatos)
  - (Queda *stock* tras el evento, $stck_{npar,e} > 0$). Si el nodo de referencia es nuevo para esta iteración, $nuevo = 1$, actualizar la cota para las etiquetas

$$delmx = \text{mín}\{delmx, stck_{npar,e}\}$$

- Si la capacidad residual del arco ficticio correspondiente al sumidero es positiva, $capr_{npar,s} > 0$, añadir el sumidero a la lista de candidatos a etiquetar
- Si el nodo de referencia es nuevo para esta iteración, $nuevo = 1$, y el sumidero ha sido incluido en la lista de candidatos, $s \in lcan$, actualizar el aumento máximo de flujo entre los nodos $npar$ y s

$$flumx_{npar,s} = \text{mín}\{delmx, flumx_{npar,s}\}$$

PASO 4

- Si hay candidatos, $lcan \neq \varnothing$:
  - Seleccionar al azar un nodo, $ngan$, de la lista de candidatos, $lcan$, etiquetarlo, calcular su máximo aumento permitido de flujo y fijar su nodo anterior

$$\begin{gathered} etiq_{ngan} = 1 \\ delta_{ngan} = \text{mín}\{delta_{npar}, flumx_{npar,ngan}\} \\ nant_{ngan} = npar \end{gathered}$$

- Si el nodo seleccionado es el sumidero, $ngan = s$, se ha encontrado un camino de aumento. Ir al PASO 5 (preparación para actualizar flujo)
- (El nodo seleccionado no es el sumidero, $ngan \neq s$). Si el nodo de referencia no es la fuente, $npar \neq f$, guardar las posiciones que ocupa el arco $k = (npar, ngan)$ en las listas de eventos de los nodos que une dicho arco

$$pos2_{npar} = e \quad \text{siendo } k = into_{npar,e}$$
$$pos1_{ngan} = e' \quad \text{siendo } k = into_{ngan,e'}$$

- Tomar el nodo seleccionado como nodo de referencia

$$npar = ngan$$

- Ir al PASO 3 (búsqueda de candidatos para etiquetar)

▪ (No hay candidatos, $lcan = \varnothing$):

- Si el nodo de referencia no es la fuente, $npar \neq f$:
  - Tomar como nuevo nodo de referencia el nodo anterior al actual nodo de referencia

$$npar = npar_{ngan}$$

  - Indicar que el nuevo nodo de referencia ya ha sido anteriormente nodo de referencia en esta iteración

$$nuevo = 0$$

  - Ir al PASO 3 (búsqueda de candidatos para etiquetar)
- (El nodo de referencia es la fuente, $npar = f$). Si se han agotado los intentos para encontrar un camino de aumento, $itet = maxet$:
  - Actualizar las cargas de los convoyes por los arcos, restando los flujos obtenidos correspondientes

$$ltar_k = ltar_k - flujo_{i,i'} \quad \text{siendo } i = ini_k, i' = fin_k \quad \forall k$$

  - FIN del procedimiento
- (No se han agotado los intentos para encontrar un camino de aumento, $itet \neq maxet$):
  - Actualizar el número de intentos para encontrar un camino de aumento

$$itet = itet + 1$$

  - Volver al PASO 2 (búsqueda de un nuevo camino de aumento)

PASO 5

▪ Indicar que el nodo de referencia actual es el último del camino de aumento (exceptuando el sumidero)

$$ult = npar$$

- Tomar el sumidero como nodo de referencia

$$npar = s$$

- Calcular la cantidad en la que se va a aumentar el flujo, $aumf$

$$aumf = delta_s$$

PASO 6

- Actualizar el flujo y la capacidad residual en el arco $(nant_{npar}, npar)$

$$\begin{array}{l} flujo_{i,i'} = flujo_{i,i'} + aumf \\ capr_{i,i'} = capr_{i,i'} - aumf \end{array} \quad \text{siendo } i = nant_{npar}, i' = npar$$

- Si el nodo de referencia es el último nodo del camino de aumento (sin contar el sumidero), $npar = ult$, actualizar los *stocks* correspondientes a los eventos posteriores a la llegada del arco utilizado para etiquetar

$$stck_{npar,e} = stck_{npar,e} - aumf \quad \forall e \geq pos1_{npar}$$

- Si el nodo de referencia es un nodo intermedio del camino de aumento, $npar \neq ult$, $npar \neq s$, $nant_{npar} \neq f$:
  - Actualizar los *stocks* correspondientes a los eventos comprendidos entre la llegada y salida de los arcos utilizados para etiquetar

$$stck_{npar,e} = stck_{npar,e} - aumf \quad \forall e \in \{pos1_{npar}, \ldots, pos2_{npar} - 1\}$$

  - Si el nodo de referencia es un nodo necesitado, $npar \in lnec$, actualizar la capacidad residual entre la fuente y dicho nodo

$$capr_{f,npar} = \text{máx}\{0, capr_{f,npar} - aumf\}$$

- Si el nodo de referencia es el primer nodo en el camino de aumento (exceptuando la fuente), $nant_{npar} = f$:
  - Actualizar los *stocks* correspondientes a los eventos posteriores a la salida del arco utilizado para etiquetar

$$stck_{npar,e} = stck_{npar,e} + aumf \quad \forall e \geq pos2_{npar}$$

  - Actualizar el valor del flujo añadiéndole la cantidad de aumento de flujo

$$valf = valf + aumf$$

  - Volver al PASO 2 (búsqueda de un nuevo camino de aumento)
- (El nodo de referencia no es el primer nodo del camino de aumento, $nant_{npar} \neq f$). Tomar el nodo anterior en el camino de aumento como nodo de referencia

$$npar = nant_{npar}$$

- Volver al PASO 6 (actualización del flujo)

El diagrama de flujo de este procedimiento, así como el del siguiente procedimiento, APROVECHAR DEPÓSITOS, sería muy similar al correspondiente al procedimiento ENVIAR FLUJO, que se muestra en la Figura 3.4.

En la Figura 3.5 se podía observar que se transportaban 3 toneladas de ayuda por el arco $g - f$. Sin embargo, como en el nodo f se satisfacía la demanda en mucha mayor proporción que en el nodo g, tras aplicar este procedimiento se consigue que esas 3 toneladas permanezcan en el nodo g, equilibrándose el reparto en cierta medida. La nueva situación queda contemplada en la Figura 4.1. Además, dado que ahora el convoy que recorre el arco $g-f$ transporta menos material, el coste total del reparto también disminuye.

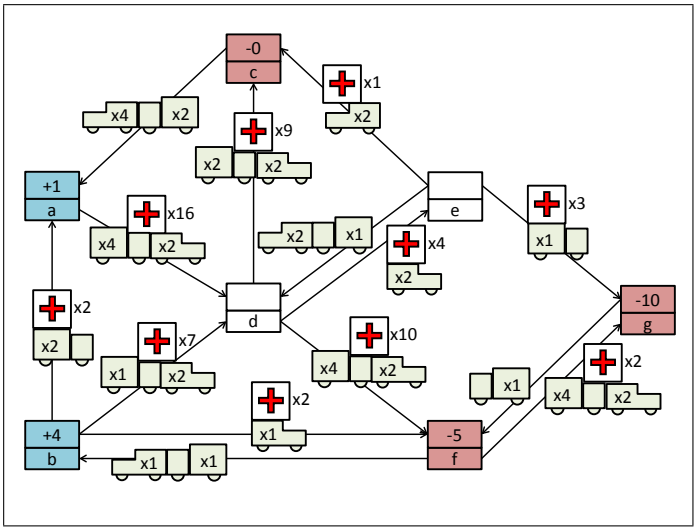


Figura 4.1: Solución parcial tras RETENER MATERIAL

Procedimiento 19. - APROVECHAR DEPÓSITOS

Eliminación de flujo de material superfluo que pasa por los depósitos. Si a algún depósito llega material procedente de otros depósitos y con destino a los nodos de demanda, parte de este material (o todo) puede sustituirse por el que originalmente hubiese disponible en el depósito y que no se haya repartido. Se eliminaría de forma consistente el flujo correspondiente de material en su camino hasta el depósito considerado. De esta forma se mejorarían algunos objetivos sin empeorar ninguno de ellos.

Este procedimiento es muy similar a RETENER MATERIAL. Se crea una red de transporte a partir del grafo dirigido original formada por todos los nodos y los arcos transitados por algún vehículo, pero ahora invirtiendo su sentido. La capacidad de cada arco es la cantidad de material que se transporta por él (en sentido contrario). Se crean dos nodos artificiales: la fuente, f, del que parten arcos ficticios a los depósitos, y el sumidero,

s, al que llegan arcos ficticios desde los depósitos. La capacidad de cada arco ficticio que parte de la fuente es el mínimo entre la cantidad de material que recibe el depósito y su *stock* final, mientras que la capacidad de cada arco ficticio que llega al sumidero es la cantidad de material disponible originalmente en el depósito. En este caso, la adaptación del algoritmo de Ford-Fulkerson presenta las siguientes características:

- El flujo entre dos nodos se interpreta como la cantidad de material a eliminar del arco que une los nodos en sentido contrario.
- La elección de los nodos a etiquetar se realiza al azar.
- No se permiten movimientos de retroceso.
- Se deben respetar y actualizar las secuencias de *stocks* en los nodos.
- Se permiten varios intentos para obtener un camino de aumento en cada iteración.
- El algoritmo finaliza cuando se acaban los intentos para obtener un camino de aumento en una iteración.

Variables de entrada

$into$, $ltar$, $nuan$, sf, $stck$

Variables de salida

$ltar$

Variables internas

$aumf$	Cantidad en la que se ha de aumentar el flujo en una iteración
$capr_{i,i'}$	Capacidad residual del arco que va del nodo i al nodo i'
$capt_{i,i'}$	Capacidad total del arco que va del nodo i al nodo i'
$delmx$	Cota superior para las eventuales etiquetas de los nodos alcanzables desde el nodo actual en el proceso de etiquetado
$delta_i$	Etiqueta asignada al nodo i, máxima cantidad en la que se puede aumentar el flujo teniendo en cuenta el camino de aumento hasta ese nodo
$entra_i$	Cantidad de material que entra al nodo i si este es un depósito
$etiq_i$	Variable binaria que indica si el nodo i está etiquetado (1 sí; 0 no)
$flujo_{i,i'}$	Flujo a través del arco que va del nodo i al nodo i'
$flumx_{i,i'}$	Cota superior para el aumento de flujo a través del arco que va del nodo i al nodo i' en la iteración actual
$itet$	Número de intentos realizados para obtener un camino de aumento en cada iteración
$lcan$	Lista de nodos candidatos para etiquetar
$nant_i$	Nodo anterior al nodo i en el camino de aumento
$npar$	Nodo de referencia (desde el que se van a buscar candidatos para etiquetar o desde el que se va a actualizar el flujo)

nuevo	Variable binaria que indica si el nodo de referencia para etiquetar es nuevo en la iteración actual (1 sí; 0 no)
$pos1_i$	Posición que ocupa el último arco utilizado para etiquetar en la lista de eventos de su nodo final i
$pos2_i$	Posición que ocupa el último arco utilizado para etiquetar en la lista de eventos de su nodo inicial i
ult	Último nodo (sin contar el sumidero) en el camino de aumento
$valf$	Valor del flujo

Parámetros internos

maxet	Número máximo de intentos para obtener un camino de aumento en cada iteración

Algoritmo

PASO 0

- Calcular las cantidades de material que entran en los depósitos

$$entra_i = \sum_{k/fin_k=i} ltar_k \quad \forall i \ / \ qav_i > 0$$

- Calcular las capacidades totales de los arcos que unen la fuente, f, con los depósitos

$$capt_{f,i} = \text{mín}\{entra_i, sf_i\} \quad \forall i \ / \ qav_i > 0$$

- Calcular las capacidades totales de los arcos que unen los depósitos con el sumidero, s

$$capt_{i,s} = qav_i \quad \forall i \ / \ qav_i > 0$$

- Calcular las capacidades totales de los arcos transitados

$$capt_{i,i'} = ltar_k \quad \forall k \ / \ conv_k \neq \varnothing \quad \text{siendo } i = fin_k, i' = ini_k$$

PASO 1

- Inicializar el flujo entre cada par de nodos i e i' (incluyendo la fuente y el sumidero) y el valor del flujo

$$flujo_{i,i'} = 0 \quad \forall i, i' \in I \cup \{f, s\}$$
$$valf = 0$$

- Inicializar las capacidades residuales. Deben coincidir con las capacidades totales

$$capr_{i,i'} = capt_{i,i'} \quad \forall i, i' \in I \cup \{f, s\}$$

- Inicializar el número de intentos realizados en la búsqueda de un camino de aumento

$$itet = 1$$

PASO 2

- Borrar etiquetas, reiniciar variables auxiliares

$$etiq_i = 0, delta_i = qglobal, nant_i = f \quad \forall i \in I \cup \{f, s\}$$

- Etiquetar la fuente y tomarla como nodo de referencia

$$etiq_f = 1, npar = f$$

- Indicar que el nodo de referencia es nuevo en la presente iteración

$$nuevo = 1$$

- Inicializar las cotas superiores para el aumento de flujo en la iteración actual

$$flumx_{i,i'} = capr_{i,i'} \quad \forall i, i' \in I \cup \{f, s\}$$

PASO 3

- Si el nodo de referencia es la fuente, $npar = f$, o el nodo anterior al de referencia es la fuente, $nant_{npar} = f$:
  - Crear la lista de nodos candidatos a etiquetar, $lcan$, revisando los nodos alcanzables desde el nodo de referencia. Para que un nodo i sea candidato debe cumplir:
    a. No está etiquetado, $etiq_i = 0$
    b. Su arco asociado tiene capacidad residual positiva, $capr_{npar,i} > 0$
  - Ir al PASO 4 (conteo de candidatos)
- (Ni el nodo de referencia ni su anterior son la fuente, $npar \neq f$, $nant_{npar} \neq f$). Si el nodo de referencia es nuevo para esta iteración, $nuevo = 1$, fijar la cota superior para las eventuales etiquetas de los nodos alcanzables desde el nodo de referencia como la etiqueta del nodo de referencia

$$delmx = del_{npar}$$

- Revisar todos los arcos usados, k, incidentes en el nodo de referencia, $npar$, desde el evento inmediatamente anterior al correspondiente al arco desde el que se etiquetó el nodo de referencia, $pos1_{npar}$, hasta el primer evento, en orden inverso de ocurrencia. Cada arco k lleva asociado un evento e tal que $k = into_{npar,e}$:
  - Si el arco k sale del nodo, $npar = ini_k$:
    - Si no queda $stock$ tras el evento, $stck_{npar,e} = 0$, ir al PASO 4 (conteo de candidatos)

- (Queda *stock* tras el evento, $stck_{npar,e} > 0$). Si el nodo de referencia es nuevo para esta iteración, $nuevo = 1$, actualizar la cota para las etiquetas

$$delmx = \text{mín}\{delmx, stck_{npar,e}\}$$

- (El arco k llega al nodo, $npar = fin_k$):
  - Añadir el nodo i a la lista de candidatos, $lcan$, si cumple:
    a. No está etiquetado: $etiq_i = 0$
    b. El arco tiene capacidad residual positiva : $capr_{npar,i} > 0$
  - Si el nodo de referencia es nuevo para esta iteración, $nuevo = 1$, y el nodo i ha sido incluido en la lista de candidatos, $i \in lcan$, actualizar el aumento máximo de flujo entre los nodos $npar$ e i

$$flumx_{npar,i} = \text{mín}\{delmx, flumx_{npar,i}\}$$

- Si la capacidad residual del arco ficticio correspondiente al sumidero es positiva, $capr_{npar,s} > 0$, y no hay otros candidatos, $lcan = \varnothing$, incluir el sumidero en la lista de candidatos a etiquetar

PASO 4

- Si hay candidatos, $lcan \neq \varnothing$:
  - Seleccionar al azar un nodo, $ngan$, de la lista de candidatos, $lcan$, etiquetarlo, calcular su máximo aumento permitido de flujo y fijar su nodo anterior

$$\begin{gathered} etiq_{ngan} = 1 \\ delta_{ngan} = \text{mín}\{delta_{npar}, flumx_{npar,ngan}\} \\ nant_{ngan} = npar \end{gathered}$$

  - Si el nodo seleccionado es el sumidero, $ngan = s$, se ha encontrado un camino de aumento. Ir al PASO 5 (preparación para actualizar flujo)
  - (El nodo seleccionado no es el sumidero, $ngan \neq s$). Si el nodo de referencia no es la fuente, $npar \neq f$, guardar las posiciones que ocupa el arco $k = (npar, ngan)$ en las listas de eventos de los nodos que une dicho arco

$$\begin{aligned} pos2_{npar} &= e \quad \text{siendo } k = into_{npar,e} \\ pos1_{ngan} &= e' \quad \text{siendo } k = into_{ngan,e'} \end{aligned}$$

  - Tomar el nodo seleccionado como nodo de referencia

$$npar = ngan$$

  - Ir al PASO 3 (búsqueda de candidatos para etiquetar)
- (No hay candidatos, $lcan = \varnothing$):
  - Si el nodo de referencia no es la fuente, $npar \neq f$:

- Tomar como nuevo nodo de referencia el nodo anterior al actual nodo de referencia

$$npar = npar_{ngan}$$

- Indicar que el nuevo nodo de referencia ya ha sido anteriormente nodo de referencia en esta iteración

$$nuevo = 0$$

- Ir al PASO 3 (búsqueda de candidatos para etiquetar)

- (El nodo de referencia es la fuente, $npar = f$). Si se han agotado los intentos para encontrar un camino de aumento, $itet = maxet$:
  - Actualizar las cargas de los convoyes por los arcos, restando los flujos obtenidos correspondientes

$$ltar_k = ltar_k - flujo_{i,i'} \quad \text{siendo } i = fin_k, i' = ini_k \quad \forall k$$

  - FIN del procedimiento
- (No se han agotado los intentos para encontrar un camino de aumento, $itet \neq maxet$):
  - Actualizar el número de intentos para encontrar un camino de aumento

$$itet = itet + 1$$

  - Volver al PASO 2 (búsqueda de un nuevo camino de aumento)

PASO 5

- Indicar que el nodo de referencia actual es el último del camino de aumento (exceptuando el sumidero)

$$ult = npar$$

- Tomar el sumidero como nodo de referencia

$$npar = s$$

- Calcular la cantidad en la que se va a aumentar el flujo, $aumf$

$$aumf = delta_s$$

PASO 6

- Actualizar el flujo y la capacidad residual en el arco $(nant_{npar}, npar)$

$$\begin{array}{l} flujo_{i,i'} = flujo_{i,i'} + aumf \\ capr_{i,i'} = capr_{i,i'} - aumf \end{array} \quad \text{siendo } i = nant_{npar}, i' = npar$$

- Si el nodo de referencia es el último nodo del camino de aumento (sin contar el sumidero), $npar = ult$:

- Actualizar los *stocks* correspondientes a los eventos posteriores a la llegada del arco utilizado para etiquetar

$$stck_{npar,e} = stck_{npar,e} + aumf \quad \forall e \geq pos1_{npar}$$

- Actualizar la capacidad residual desde la fuente al nodo de referencia

$$capr_{f,npar} = \text{mín}\{entra_{npar}, sf_{npar}\}$$

- Si el nodo de referencia es un nodo intermedio del camino de aumento, $npar \neq ult$, $npar \neq s$, $nant_{npar} \neq f$:
  - Actualizar los *stocks* correspondientes a los eventos comprendidos entre la llegada y salida de los arcos utilizados para etiquetar

$$stck_{npar,e} = stck_{npar,e} - aumf \quad \forall e \in \{pos2_{npar}, \ldots, pos1_{npar} - 1\}$$

  - Si el nodo de referencia es un depósito, $npar \in ldep$, actualizar la cantidad de material que entra y la capacidad residual entre la fuente y dicho nodo

$$entra_{npar} = entra_{npar} - aumf$$
$$capr_{f,npar} = \text{mín}\{entra_{npar}, sf_{npar}\}$$

- Si el nodo de referencia es el primer nodo en el camino de aumento (exceptuando la fuente), $nant_{npar} = f$:
  - Actualizar los *stocks* correspondientes a los eventos posteriores a la salida del arco utilizado para etiquetar

$$stck_{npar,e} = stck_{npar,e} - aumf \quad \forall e \geq pos2_{npar}$$

  - Actualizar el valor del flujo añadiéndole la cantidad de aumento de flujo

$$valf = valf + aumf$$

  - Volver al PASO 2 (búsqueda de un nuevo camino de aumento)
- (El nodo de referencia no es el primer nodo del camino de aumento, $nant_{npar} \neq f$). Tomar el nodo anterior en el camino de aumento como nodo de referencia

$$npar = nant_{npar}$$

- Volver al PASO 6 (actualización del flujo)

Con este procedimiento se consigue que 1 de las 2 toneladas transportadas por el arco $b - a$ (ver Figura 4.1) sea eliminada ya que, al finalizar el reparto, quedaba 1 tonelada sin usar en el depósito a que puede sustituir a la tonelada eliminada. La situación resultante aparece en la Figura 4.2.

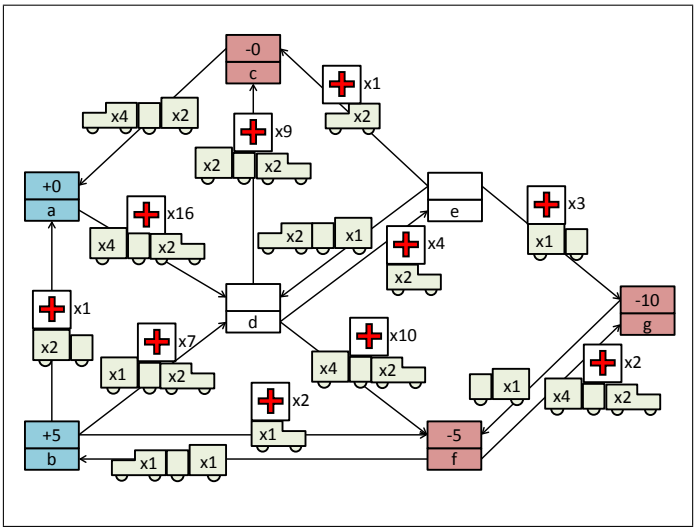


Figura 4.2: Solución parcial tras APROVECHAR DEPÓSITOS

Procedimiento 20. - ACORTAR RUTAS

Eliminación de trayectos superfluos de los vehículos. Según el Algoritmo Constructivo, los itinerarios de los vehículos se alargan tanto como sea posible, por lo que puede ocurrir que una vez establecidas las cargas de material de los convoyes, algunos de los trayectos no sean necesarios. Mediante este procedimiento se eliminan, de cada vehículo, los últimos trayectos y los circuitos que no sean imprescindibles.

La solución final depende del orden en el que se tomen los vehículos. Por ejemplo, en un convoy formado por dos vehículos en el que cualquiera de los dos es suficiente para transportar la carga correspondiente, eliminar uno u otro vehículo da lugar a dos soluciones diferentes. A continuación, se proponen distintos criterios de ordenación de los vehículos, cada uno de los cuales se corresponde con un valor del parámetro *crit*. Los empates se deshacen al azar en todos los casos:

a. Al azar.

b. De mayor a menor número de trayectos realizados.

c. De menor a mayor velocidad.

d. De menor a mayor capacidad.

e. De mayor a menor tiempo empleado en el reparto.

f. De mayor a menor distancia recorrida en el reparto.

g. De mayor a menor longitud del camino de regreso.

h. De mayor a menor distancia total recorrida.

Por ejemplo, con el criterio "a" se pretende que los vehículos no tengan que hacer demasiados trayectos. Con el criterio "b" se quiere preservar a los vehículos más rápidos, mientras que con el criterio "c" se preserva a los de mayor capacidad, etc. La elección del criterio a adoptar no es trivial, ya que, entre otros factores, habría que tener en cuenta los pesos de los objetivos. En los análisis efectuados para este trabajo, el criterio "g" ha funcionado de forma convincente y es el que se ha tomado para la elaboración de la experiencia computacional que se presenta en el Capítulo 7.

El procedimiento ACORTAR RUTAS puede empeorar la medida de la seguridad. Según se estableció como hipótesis, la probabilidad de que un convoy sea asaltado decrece con el tamaño del convoy, hasta que el convoy alcanza el tamaño disuasorio, $tdis$, en cuyo caso permanece constante. Por lo tanto, esta probabilidad puede aumentar al eliminar un vehículo de un convoy, como sucede al ejecutar este procedimiento. Sin embargo, si el convoy del que es eliminado el vehículo no transporta material, la medida de la seguridad no varía en absoluto, pues dicha medida también depende de la cantidad de material que se transporte.

Si no se desea, de ningún modo, empeorar la medida de la seguridad, puede modificarse fácilmente el procedimiento para no permitir ciertas eliminaciones de trayectos. No obstante, desde un punto de vista práctico, no parece muy razonable mantener vehículos transportando carga que podrían repartirse otros vehículos del mismo convoy solo para no empeorar la seguridad. Por esta razón, se ha decidido establecer el procedimiento ACORTAR RUTAS como se presenta a continuación.

Variables de entrada

$conv$, $lonv$, $ltar$, $secv$, $tamc$

Variables de salida

$conv$, $lonv$, $ltar$, $secv$, $tamc$

Parámetros internos

$crit$	Criterio para ordenar los vehículos

Algoritmo

- Ordenar los vehículos de acuerdo al criterio establecido en el parámetro $crit$
- Para cada vehículo, j, hacer:
  - Eliminar todos los últimos trayectos en los que el vehículo j no sea imprescindible, empezando por el último, $k = secv_{j,lonv_j}$. Parar cuando el vehículo sea imprescindible en el trayecto considerado

$$ltar_k \leq \sum_{j' \in conv_k} cap_{tipv_{j'}} - cap_{tipv_j} \quad \Rightarrow$$

$$\begin{cases} \text{eliminar el arco } k \text{ de la ruta del vehículo } j,\ secv_j \\ \text{reducir en una unidad la longitud de la ruta del vehículo } j,\ lonv_j \\ \text{eliminar el vehículo } j \text{ del convoy del arco } k,\ conv_k \\ \text{reducir en una unidad el tamaño del convoy del arco } k,\ tamc_k \end{cases}$$

- Eliminar todos los arcos de cada circuito en el que el vehículo j no sea imprescindible. Asimismo, eliminar el vehículo de los convoyes de los arcos del circuito. El vehículo j no es imprescindible en un circuito $(k_1 = secv_{j,e}, k_2 = secv_{j,e+1}, \ldots, k_n = secv_{j,e+n-1})$ con $ini_{k_1} = fin_{k_n}$ si

$$ltar_k \leq \sum_{j' \in conv_k} cap_{tipv_{j'}} - cap_{tipv_j} \quad \forall k \in \{k_1, k_2, \ldots, k_n\}$$

Como era de esperar, muchos de los trayectos de las rutas originalmente obtenidas para el ejemplo han resultado ser innecesarios. Con la aplicación de RETENER MATERIAL y APROVECHAR DEPÓSITOS, situación reflejada en la Figura 4.2, se ha reducido el flujo de material, por lo que el número de trayectos imprescindibles es todavía menor. Después de aplicar ACORTAR RUTAS se obtiene la solución que puede verse en la Figura 4.3. El número total de trayectos (suma del número de trayectos de todos los vehículos) ha disminuido de 45 a 29. Si se observan los vehículos 1 y 2, cuyas rutas originales aparecían en la Figura 3.5, puede verse cómo el primer vehículo pasa de realizar la ruta $a-d-f-g$ a la nueva ruta $a-d$, eliminándose así dos trayectos, mientras que el vehículo 2, que originalmente seguía el itinerario $a-d-c-a$, ve acortado su último trayecto, resultando la nueva ruta $a-d-c$, como puede observarse en la Figura 4.4.

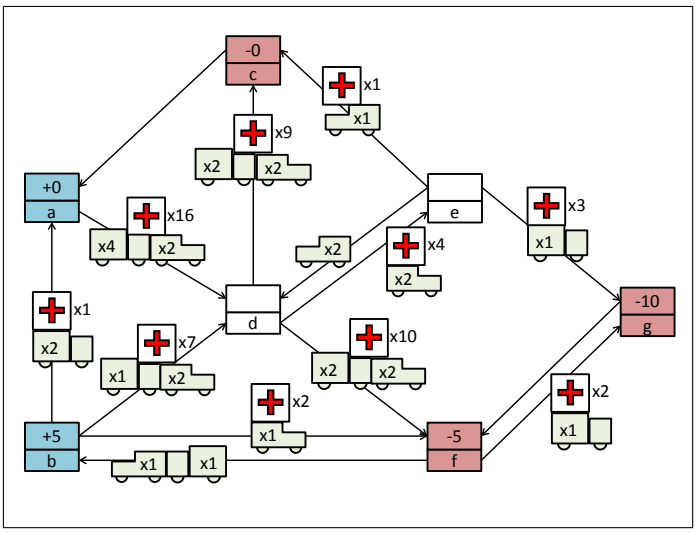


Figura 4.3: Solución parcial tras ACORTAR RUTAS

Procedimiento 21. - SUSTITUIR VEHÍCULOS

Sustituciones entre pares de vehículos del mismo tipo a partir de ciertos nodos para poder eliminar algunos trayectos de alguno de los vehículos del par. En cada iteración se

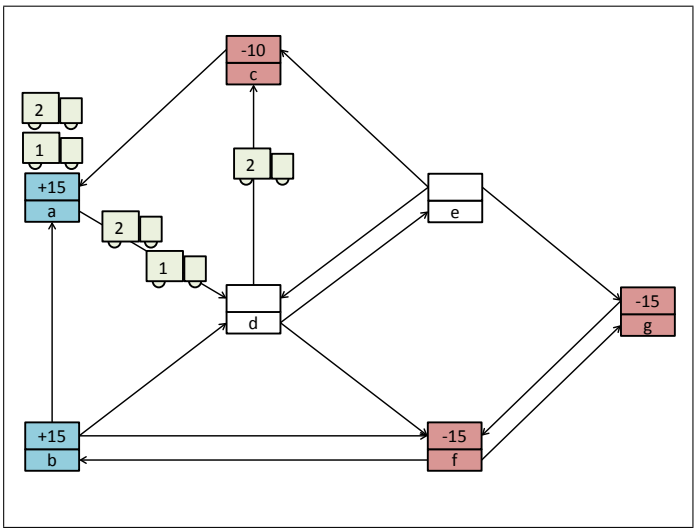


Figura 4.4: Rutas de los vehículos 1 y 2 tras ACORTAR RUTAS

comprueba si cada vehículo puede ser sustituido por algún otro a partir de cierto nodo, de forma que la sustitución permita eliminar algunas etapas consecutivas donde el vehículo no sea imprescindible. El procedimiento termina cuando en una iteración no se pueden realizar sustituciones de esta forma. Al igual que en el procedimiento ACORTAR RUTAS, la solución final puede depender del orden en el que se tomen los vehículos. Aunque se ha optado por ordenarlos aleatoriamente, como alternativa puede usarse alguno de los criterios mencionados en dicho procedimiento.

Como ya se ha señalado al presentar el procedimiento ACORTAR RUTAS, eliminar trayectos de los vehículos puede empeorar la medida de la seguridad. Además, el intercambio de rutas de vehículos a partir de un nodo podría suponer, en determinadas situaciones, un empeoramiento en la medida de la fiabilidad, pues esta medida depende de las descargas que realicen los vehículos en el futuro. Sin embargo, estos posibles empeoramientos son habitualmente poco significativos en comparación con las mejoras producidas en los demás atributos. En cualquier caso, si el atributo que potencialmente podría empeorar fuera el único a considerar o tuviera un peso predominante en la función objetivo, siempre puede recuperarse la solución original o simplemente omitirse el procedimiento.

Variables de entrada

conv, *dur*, *lonv*, *ltar*, *secv*, *tamc*, *tmin*

Variables de salida

conv, *lonv*, *secv*, *tamc*

Variables internas

$c1$	Clase del vehículo $v1$
$c2$	Clase del vehículo $v2$
$cveh$	Variable binaria que indica si ha habido cambios en la iteración actual
e, e'	Etapas genéricas en la ruta de un vehículo
$e1$	Primera etapa de una secuencia de etapas en las que un vehículo no es imprescindible
$e2$	Primera etapa de una secuencia de etapas en las que un vehículo es imprescindible
$e3$	Etapa del vehículo $v1$ a partir de la cual será sustituido
l, l'	Índices que recorren la lista $lisv$
$lisv$	Lista con todos los vehículos
$nend_j$	Nodo de finalización de la ruta del vehículo j
$tend_j$	Instante de finalización de la ruta del vehículo j
$v1$	Vehículo susceptible de ser sustituido
$v2$	Vehículo elegido para sustituir al vehículo $v1$

Algoritmo

PASO 0

- Calcular el nodo y el instante de finalización de cada vehículo j
  - Si el vehículo j no ha sido usado, $lonv_j = 0$

$$\begin{array}{c} nend_j = ori_j \\ tend_j = 0 \end{array}$$

  - Si el vehículo j ha sido usado, $lonv_j \geq 1$

$$\begin{array}{c} nend_j = fin_k \\ tend_j = tmin_k \end{array} \quad \text{siendo } k = secv_{j,lonv_j}$$

- Crear una lista con los vehículos ordenados según algún criterio (por ejemplo al azar o simplemente el orden original), $lisv$

PASO 1

- Indicar que no se ha producido ninguna sustitución en la presente iteración

$$cveh = 0$$

- Tomar el primer índice, que corresponde al primer vehículo de la lista

$$l = 1$$

PASO 2

- Sea $v1$ el primer vehículo de la lista a partir del índice l que cumple:

a. Recorre al menos 2 arcos: $lonv_{v1} \geq 2$
b. No es imprescindible en alguno de los arcos que recorre

$$ltar_k \leq \sum_{j \in conv_k} cap_{tipv_j} - cap_{tipv_{v1}} \quad \text{para algún arco } k \ / \ v1 \in conv_k$$

- Si no queda ningún vehículo con estas características, ir al PASO 4 (fin de la presente iteración)
- Actualizar el índice l adjudicándole el índice correspondiente al vehículo actual

$$l \ / \ lisv_l = v1$$

- Tomar la primera etapa del vehículo $v1$

$$e = 1$$

PASO 3

- Si no existe, a partir de la etapa e, ningún arco de la ruta del vehículo $v1$ donde este no sea imprescindible:
  - Pasar al siguiente vehículo de la lista

$$l = l + 1$$

  - Volver al PASO 2
- (Existe algún arco a partir de la etapa e donde el vehículo $v1$ no es imprescindible). Sea $e1$ la primera etapa a partir de la etapa e en la que el vehículo $v1$ no es imprescindible
- Encontrar, a partir de la etapa $e1$, el primer arco de la ruta del vehículo $v1$ donde este sea imprescindible. Sea $e2$ la etapa correspondiente a este arco
- Encontrar, si existe, la última etapa de la secuencia $(e1, \ldots, e2-1)$ en la que exista un vehículo candidato a sustituir al vehículo $v1$ desde el nodo inicial del arco correspondiente a esa etapa. Un vehículo j es candidato a sustituir al vehículo $v1$ desde el inicio de un arco k si cumple:
  a. Es del mismo tipo que $v1$, $tipv_j = tipv_{v1}$
  b. Finaliza su ruta en el nodo inicial del arco k, $nend_j = ini_k$
  c. Finaliza su ruta como muy tarde en el instante en el que el convoy del arco k empieza a recorrerlo, $tend_j \leq tmin_k - dur_k$
- Si no existe ningún vehículo candidato a sustituir al vehículo $v1$ en ninguna de las etapas de la secuencia $(e1, \ldots, e2-1)$:
  - Pasar a la primera etapa del vehículo $v1$ posterior a la última etapa encontrada donde el vehículo era imprescindible

$$e = e2 + 1$$

- Volver al PASO 3

- (Existe algún vehículo candidato a sustituir al vehículo $v1$ en alguna de las etapas de la secuencia $(e1, \ldots, e2-1)$). Sea $e3$ la última etapa de la secuencia donde existen vehículos candidatos. Sea k el arco recorrido en esta etapa por el vehículo $v1$ e i el nodo inicial del arco k

$$k = secv_{v1,e3}$$
$$i = ini_k$$

- Obtener el vehículo $v2$ que minimiza la suma de distancias de regreso entre todos los candidatos. Es decir, el vehículo candidato $v2$ cumple que, para cualquier otro vehículo candidato j, se tiene que:

$$dreg_{c1,i} + dreg_{c2,i'} - dreg_{c2,i} \geq dreg_{c1,i} + dreg_{c,i'} - dreg_{c,i}$$
$$\text{siendo} \begin{cases} c1 = clas_{tipv_{v1},ori_{v1}} \\ c2 = clas_{tipv_{v2},ori_{v2}} \\ c = clas_{tipv_j,ori_j} \\ i' = nend_{v1} \end{cases}$$

- Indicar que ha habido cambios en esta iteración

$$cveh = 1$$

- Sustituir el vehículo $v1$ por el vehículo $v2$ en los convoyes de los arcos correspondientes a la secuencia de etapas $(e3, \ldots, lonv_{v1})$ del vehículo $v1$

$$conv_{secv_{v1,e'}} = conv_{secv_{v1,e'}} \cup \{v2\} \setminus \{v1\} \quad \forall e' \ / \ e3 \leq e' \leq lonv_{v1}$$

- Prolongar la secuencia del vehículo $v2$

$$secv_{v2,lonv_{v2}+e'-e3+1} = secv_{v1,e'} \quad \forall e' \ / \ e3 \leq e' \leq lonv_{v1}$$
$$lonv_{v2} = lonv_{v2} + lonv_{v1} - e3 + 1$$

- Eliminar el vehículo $v1$ de los convoyes de los arcos correspondientes a la secuencia de etapas $(e1, \ldots, e3-1)$

$$\begin{matrix} conv_{secv_{v1,e'}} = conv_{secv_{v1,e'}} \setminus \{v1\} \\ tamc_{secv_{v1,e'}} = tamc_{secv_{v1,e'}} - 1 \end{matrix} \quad \forall e' \ / \ e1 \leq e' \leq e3-1$$

- Acortar la ruta del vehículo $v1$

$$lonv_{v1} = e1 - 1$$

- Actualizar el nodo y el instante de finalización del vehículo $v2$

$$nend_{v2} = nend_{v1}$$
$$tend_{v2} = tend_{v1}$$

- Actualizar el nodo y el instante de finalización del vehículo $v1$
  - Si el vehículo $v1$ ya no se usa, $lonv_{v1} = 0$

$$nend_{v1} = ori_{v1}$$
$$tend_{v1} = 0$$

  - Si el vehículo $v1$ se sigue usando, $lonv_{v1} \geq 1$

$$nend_{v1} = fin_{k'} \quad \text{siendo } k' = secv_{v1,lonv_{v1}}$$
$$tend_j = tmin_{k'}$$

- Si el vehículo $v2$ ya ha sido estudiado en la presente iteración, $l' < l$, siendo $v2 = lisv_{l'}$:
  - Intercambiar las posiciones de $v1$ y $v2$ en la lista de vehículos $lisv$
  - Volver al PASO 2
- (El vehículo $v2$ no ha sido estudiado todavía en la presente iteración, $l' > l$, siendo $v2 = lisv_{l'}$):
  - Pasar al siguiente vehículo de la lista

$$l = l + 1$$

  - Volver al PASO 2

PASO 4

- Si se ha producido alguna sustitución en la presente iteración, $cveh = 1$, volver al PASO 1 (comienzo de una nueva iteración)
- (No se ha producido ninguna sustitución en la presente iteración, $cveh = 0$). FIN del procedimiento

Un esquema del funcionamiento de este procedimiento puede observarse en la Figura 4.5. Muy similar sería el diagrama de flujo del siguiente procedimiento, INTERCAMBIAR VEHÍCULOS, que se omite por esta razón.

En el ejemplo que se está estudiando no se realiza ninguna de estas sustituciones, ya que no existe forma, a partir de la solución que aparece en la Figura 4.3, de que una sustitución de vehículos dé como resultado la eliminación de ningún trayecto.

Procedimiento 22. - INTERCAMBIAR VEHÍCULOS

Intercambios entre pares de vehículos utilizados en el reparto, del mismo tipo pero distinto origen, para disminuir la suma de distancias de regreso. Los vehículos son analizados de dos en dos, buscando un nodo común en sus rutas, a partir del cual puedan intercambiarse de forma que se mejore la solución. Como criterio alternativo, podrían considerarse también el coste total de regreso o las sumas de los tiempos de regreso. Al igual que en los

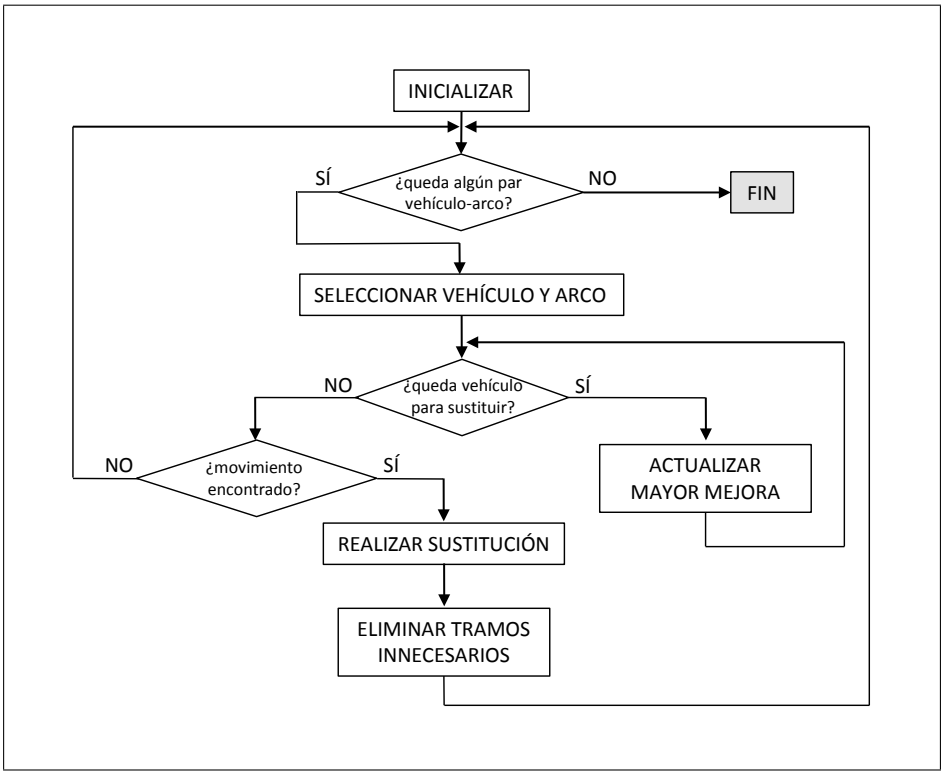


Figura 4.5: Diagrama de flujo de SUSTITUIR VEHÍCULOS

procedimientos anteriores, la solución final puede depender del orden en el que se tomen los vehículos. Nuevamente, se ha optado por ordenarlos aleatoriamente.

Variables de entrada

$conv$, dur, $lonv$, $secv$, $tmin$

Variables de salida

$conv$, $lonv$, $secv$

Variables internas

$antl$	Longitud de la ruta del vehículo $v1$ antes de un intercambio
$ants$	Ruta del vehículo $v1$ antes de un intercambio
$c1$	Clase del vehículo $v1$
e, e'	Etapas genéricas en la ruta de un vehículo
$e1$	Etapa a partir de la cual el vehículo $v1$ será intercambiado. $e1 = 0$ indica que $v1$ será intercambiado desde el final de su ruta
$e2$	Etapa a partir de la cual el vehículo $v2$ será intercambiado. $e2 = 0$ indica que $v2$ será intercambiado desde el final de su ruta

l, l'	Índices que recorren la lista $lisv$
$lisv$	Lista con todos los vehículos usados en el reparto
$maxm$	Máxima mejora en la suma de distancias de regreso obtenida hasta cada momento
$nact$	Nodo desde el que se comprueba si se puede hacer algún intercambio
$nend_j$	Nodo de finalización de la ruta del vehículo j
nvu	Número de vehículos usados en el reparto
$vard$	Disminución en la suma de distancias de regreso para un posible intercambio
$v1$	Vehículo propuesto para intercambiar
$v2$	Vehículo elegido para intercambiarse con el vehículo $v1$. Toma el valor 0 mientras no se haya elegido ninguno
yav_j	Variable binaria que indica si el vehículo j ya está analizado para intercambiarse con el vehículo $v1$

Algoritmo

PASO 0

- Crear una lista con los vehículos usados en el reparto ordenados según algún criterio (por ejemplo al azar o simplemente atendiendo al orden original), $lisv$. Sea nvu el número de vehículos usados
- Calcular el nodo de finalización de cada vehículo usado j

$$nend_j = fin_k \quad \text{siendo } k = secv_{j,lonv_j}$$

- Tomar el primer índice, que corresponde al primer vehículo de la lista

$$l = 1$$

PASO 1

- Considerar el vehículo a intercambiar, $v1$

$$v1 = lisv_l$$

- Reiniciar la máxima mejora a su valor nulo

$$maxm = 0$$

- Indicar que, por el momento, no existe ningún vehículo candidato a intercambiarse con el vehículo $v1$

$$v2 = 0$$

- Indicar que no se ha analizado ningún vehículo para sustituir a $v1$

$$yav_j = 0 \quad \forall j \in lisv$$

- Tomar la primera etapa del vehículo $v1$

$$e = 1$$

PASO 2

- Considerar el arco actual de la ruta del vehículo $v1$, k

$$k = secv_{v1,e}$$

- Considerar el nodo inicial del arco actual k, $nact$

$$nact = ini_k$$

- Considerar, para cada arco k' entrante al nodo $nact$, cada vehículo que recorra dicho arco, $j \in conv_{k'}$, que cumpla:
  a. Es del mismo tipo que $v1$, $tipv_j = tipv_{v1}$
  b. Comienza en un nodo distinto a $v1$, $ori_j \neq ori_{v1}$
  c. Termina en un nodo distinto a $v1$, $nend_j \neq nend_{v1}$
  d. Todavía no se ha analizado para intercambiarse por $v1$, $yav_j = 0$
  e. Llega al nodo $nact$ como muy tarde en el instante en el que $v1$ sale de ese nodo, $tmin_{k'} \leq tmin_k - dur_k$
  f. Sale del nodo $nact$ como muy pronto en el instante en el que $v1$ llega a ese nodo, $tmin_{secv_{j,e'+1}} - dur_{secv_{j,e'+1}} \geq tmin_{secv_{v1,e-1}}$, siendo $k' = secv_{j,e'}$. Esta condición no se comprueba si e es la primera etapa de $v1$ o si k' es el último arco que recorre j
- Para cada vehículo j que cumpla las condiciones anteriores:
  - Indicar que ya está analizado

$$yav_j = 1$$

  - Calcular la disminución que produciría el intercambio de $v1$ y j en la suma de distancias de regreso

$$vard = dreg_{c',i'} + dreg_{c,i} - dreg_{c',i} - dreg_{c,i'}$$

$$\text{siendo} \begin{cases} c = clas_{tipv_{v1},ori_{v1}} \\ c' = clas_{tipv_j,ori_j} \\ i = nend_{v1} \\ i' = nend_j \end{cases}$$

  - Si j es mejor (presenta una mayor disminución en la suma de distancias de regreso) que el mejor candidato obtenido hasta el momento, $vard > maxm$:
    - Guardar el vehículo j como el mejor candidato obtenido para sustituir al vehículo $v1$

$$v2 = j$$

- Actualizar la máxima mejora

$$maxm = vard$$

- Guardar la etapa del vehículo $v1$ a partir de la cual se haría el intercambio

$$e1 = e$$

- Guardar la etapa del vehículo j a partir de la cual se haría el intercambio

$$e2 = \begin{cases} 0 & \text{si } k' = secv_{j,lonv_j} \\ e'+1 \ / \ k' = secv_{j,e'} & \text{si } k' \neq secv_{j,lonv_j} \end{cases}$$

- Considerar, para cada arco k' saliente del nodo $nact$, cada vehículo que recorra dicho arco, $j \in conv_{k'}$, que cumpla:
  a. Es del mismo tipo que $v1$, $tipv_j = tipv_{v1}$
  b. Comienza en un nodo distinto a $v1$, $ori_j \neq ori_{v1}$
  c. Termina en un nodo distinto a $v1$, $nend_j \neq nend_{v1}$
  d. Todavía no se ha analizado para intercambiarse por $v1$, $yav_j = 0$
  e. Llega al nodo $nact$ como muy tarde en el instante en el que $v1$ sale de ese nodo, $tmin_{secv_{j,e'-1}} \leq tmin_k - dur_k$, siendo $k' = secv_{j,e'}$. Esta condición no se comprueba si e' es la última etapa de j
  f. Sale del nodo $nact$ como muy pronto en el instante en el que $v1$ llega a ese nodo, $tmin_{k'} - dur_{k'} \geq tmin_{secv_{v1,e-1}}$. Esta condición no se comprueba si e es la primera etapa de $v1$
- Para cada vehículo j que cumpla las condiciones anteriores:
  - Indicar que ya está analizado

$$yav_j = 1$$

  - Calcular la disminución que produciría el intercambio de $v1$ y j en la suma de distancias de regreso

$$vard = dreg_{c',i'} + dreg_{c,i} - dreg_{c',i} - dreg_{c,i'}$$
$$\text{siendo} \begin{cases} c = clas_{tipv_{v1},ori_{v1}} \\ c' = clas_{tipv_j,ori_j} \\ i = nend_{v1} \\ i' = nend_j \end{cases}$$

  - Si j es mejor (presenta una mayor disminución en la suma de distancias de regreso) que el mejor candidato obtenido hasta el momento, $vard > maxm$:
    - Guardar el vehículo j como el mejor candidato obtenido para sustituir al vehículo $v1$

$$v2 = j$$

    - Actualizar la máxima mejora

$$maxm = vard$$

- Guardar la etapa del vehículo $v1$ a partir de la cual se haría el intercambio

$$e1 = e$$

- Guardar la etapa del vehículo j a partir de la cual se haría el intercambio

$$e2 = e' \ / \ k' = secv_{j,e'}$$

- Si la etapa actual es la última etapa del vehículo $v1$, $e = lonv_{v1}$, ir al PASO 3
- (La etapa actual no es la última etapa del vehículo $v1$, $e < lonv_{v1}$)
  - Pasar a la siguiente etapa del vehículo $v1$

$$e = e + 1$$

  - Volver al PASO 2

PASO 3

- Considerar el arco actual de la ruta del vehículo $v1$, k

$$k = secv_{v1,e}$$

- Considerar el nodo final del arco actual k, $nact$

$$nact = fin_k$$

- Considerar, para cada arco k' saliente del nodo $nact$, cada vehículo que recorra dicho arco, $j \in conv_{k'}$, que cumpla:
  a. Es del mismo tipo que $v1$, $tipv_j = tipv_{v1}$
  b. Comienza en un nodo distinto que $v1$, $ori_j \neq ori_{v1}$
  c. Termina en un nodo distinto que $v1$, $nend_j \neq nend_{v1}$
  d. Todavía no se ha analizado para intercambiarse por $v1$, $yav_j = 0$
  e. Sale del nodo $nact$ como muy pronto en el instante en el que $v1$ llega a ese nodo, $tmin_{k'} - dur_{k'} \geq tmin_k$
- Para cada vehículo j que cumpla las condiciones anteriores:
  - Indicar que ya está analizado

$$yav_j = 1$$

  - Calcular la disminución que produciría el intercambio de $v1$ y j en la suma de distancias de regreso

$$vard = dreg_{c',i'} + dreg_{c,nact} - dreg_{c',nact} - dreg_{c,i'}$$

$$\text{siendo} \begin{cases} c = clas_{tipv_{v1},ori_{v1}} \\ c' = clas_{tipv_j,ori_j} \\ i' = nend_j \end{cases}$$

- Si j es mejor (presenta una mayor disminución en la suma de distancias de regreso) que el mejor candidato obtenido hasta el momento, $vard > maxm$:
  - Guardar el vehículo j como el mejor candidato obtenido para sustituir al vehículo $v1$
  $$v2 = j$$
  - Actualizar la máxima mejora
  $$maxm = vard$$
  - Guardar la etapa del vehículo $v1$ a partir de la cual se haría el intercambio
  $$e1 = 0$$
  - Guardar la etapa del vehículo j a partir de la cual se haría el intercambio
  $$e2 = e' \ / \ k' = secv_{j,e'}$$

PASO 4

- Si no se ha obtenido ningún vehículo candidato, $v2 = 0$:
  - Si no quedan vehículos para intentar intercambiar, $l = nvu$, FIN del procedimiento
  - (Quedan vehículos para intentar intercambiar, $l \neq nvu$):
    - Pasar al siguiente vehículo de la lista
    $$l = l + 1$$
    - Volver al PASO 1 (propuesta de vehículo para intercambiar)
- (Se ha obtenido algún vehículo candidato, $v2 > 0$). Ir al PASO 5 (ejecución del intercambio)

PASO 5

- Si el vehículo $v2$ se intercambia desde su nodo final, $e2 = 0$:
  - Sustituir el vehículo $v1$ por el vehículo $v2$ en los convoyes de los arcos correspondientes a la secuencia de etapas $(e1, \ldots, lonv_{v1})$ del vehículo $v1$
  $$conv_{secv_{v1,e}} = conv_{secv_{v1,e}} \cup \{v2\} \setminus \{v1\} \quad \forall e \ / \ e1 \leq e \leq lonv_{v1}$$
  - Prolongar la secuencia del vehículo $v2$
  $$secv_{v2,lonv_{v2}+e-e1+1} = secv_{v1,e} \quad \forall e \ / \ e1 \leq e \leq lonv_{v1}$$
  $$lonv_{v2} = lonv_{v2} + lonv_{v1} - e1 + 1$$
  - Actualizar el nodo de finalización del vehículo $v2$
  $$nend_{v2} = nend_{v1}$$

- Acortar la ruta del vehículo $v1$

$$lonv_{v1} = e1 - 1$$

- Actualizar el nodo de finalización del vehículo $v1$

$$nend_{v1} = fin_k \quad \text{siendo } k = secv_{v1,lonv_{v1}}$$

- Si el vehículo $v1$ se intercambia desde su nodo final, $e1 = 0$:
  - Sustituir el vehículo $v2$ por el vehículo $v1$ en los convoyes de los arcos correspondientes a la secuencia de etapas $(e2, \ldots, lonv_{v2})$ del vehículo $v2$

$$conv_{secv_{v2,e'}} = conv_{secv_{v2,e'}} \cup \{v1\} \setminus \{v2\} \quad \forall e \ / \ e2 \leq e' \leq lonv_{v2}$$

  - Prolongar la secuencia del vehículo $v1$

$$secv_{v1,lonv_{v1}+e'-e2+1} = secv_{v2,e'} \quad \forall e' \ / \ e2 \leq e' \leq lonv_{v2}$$
$$lonv_{v1} = lonv_{v1} + lonv_{v2} - e2 + 1$$

  - Actualizar el nodo de finalización del vehículo $v1$

$$nend_{v1} = nend_{v2}$$

  - Acortar la ruta del vehículo $v2$

$$lonv_{v2} = e2 - 1$$

  - Actualizar el nodo de finalización del vehículo $v2$

$$nend_{v2} = fin_{k'} \quad \text{siendo } k' = secv_{v2,lonv_{v2}}$$

- (Ninguno de los dos vehículos se intercambian desde sus nodos finales, $e1 \neq 0$ y $e2 \neq 0$):
  - Sustituir el vehículo $v1$ por el vehículo $v2$ en los convoyes de los arcos correspondientes a la secuencia de etapas $(e1, \ldots, lonv_{v1})$ del vehículo $v1$

$$conv_{secv_{v1,e}} = conv_{secv_{v1,e}} \cup \{v2\} \setminus \{v1\} \quad \forall e \ / \ e1 \leq e \leq lonv_{v1}$$

  - Sustituir el vehículo $v2$ por el vehículo $v1$ en los convoyes de los arcos correspondientes a la secuencia de etapas $(e2, \ldots, lonv_{v2})$ del vehículo $v2$

$$conv_{secv_{v2,e'}} = conv_{secv_{v2,e'}} \cup \{v1\} \setminus \{v2\} \quad \forall e \ / \ e2 \leq e' \leq lonv_{v2}$$

  - Actualizar la secuencia del vehículo $v1$, guardando la secuencia antigua

$$ants_e = secv_{v1,e} \quad \forall e \ / \ e1 \leq e \leq lonv_{v1}$$
$$secv_{v1,e1+e'-e2} = secv_{v2,e'} \quad \forall e' \ / \ e2 \leq e' \leq lonv_{v2}$$
$$antl = lonv_{v1}$$
$$lonv_{v1} = e1 + lonv_{v2} - e2$$

- Actualizar el nodo de finalización del vehículo $v1$

$$nend_{v1} = nend_{v2}$$

- Actualizar la secuencia del vehículo $v2$

$$secv_{v2,e2+e-e1} = ants_e \quad \forall e \ / \ e1 \leq e \leq antl$$
$$lonv_{v2} = e2 + antl - e1$$

- Actualizar el nodo de finalización del vehículo $v2$

$$nend_{v2} = fin_{k'} \quad \text{siendo } k' = secv_{v2,lonv_{v2}}$$

- Si el vehículo $v2$ ya había sido propuesto para intercambiarse, $l' < l$, siendo $v2 = lisv_{l'}$:
  - Situar el vehículo $v2$ al final de la lista de vehículos $lisv$ (los vehículos a partir de $v2$ corren un puesto a la izquierda)
  - Actualizar el índice l para volver a estudiar el vehículo $v1$ con su nueva secuencia
$$l = l - 1$$
  - Volver al PASO 1 (propuesta de vehículo para intercambiar)
- (El vehículo $v2$ no había sido propuesto para intercambiarse, $l' > l$, siendo $v2 = lisv_{l'}$). Volver al PASO 1 (propuesta de vehículo para intercambiar)

Este procedimiento provoca varios cambios en los itinerarios de los vehículos en el ejemplo propuesto. Por ejemplo, el vehículo 1, que realizaba un solo trayecto (itinerario $a - d$), como podía observarse en la Figura 4.4, realiza ahora el itinerario $a - d - f - b - d - c$ (ver Figura 4.6). Esto es debido a que uno de los vehículos (vehículo 6) de su mismo tipo, ubicado originalmente en el nodo b, realizaba el itinerario $b - a - d - f - b - d - c$. Traspasando la secuencia $d - f - b - d - c$ del vehículo 6 al vehículo 1 se consigue que ambos vehículos terminen su misión más cerca de sus nodos de origen (a los que han de volver una vez completado el reparto), acortando así el coste de regreso. También puede verse en la Figura 4.6 que el vehículo 2 no ha visto modificada su ruta.

Procedimiento 23. - REFINAR SOLUCIÓN

Refinamiento de la solución construida con el objeto de mejorarla en algunos aspectos. Consta de cinco fases, que corresponden a sendos procedimientos descritos con anterioridad. En la primera fase (RETENER MATERIAL) se impide que cierta cantidad de material abandone nodos de demanda donde no se haya repartido la cantidad idealmente justa. En la segunda fase (APROVECHAR DEPÓSITOS) se elimina flujo de material que pase por depósitos donde exista material sobrante tras el reparto. En la tercera fase (ACORTAR RUTAS) se eliminan trayectos para cada uno de los vehículos donde estos no sean imprescindibles. La cuarta (SUSTITUIR VEHÍCULOS) y la quinta fase (INTERCAMBIAR VEHÍCULOS) consisten en realizar intercambios entre vehículos con el fin seguir eliminando trayectos o disminuir la suma distancias de regreso.

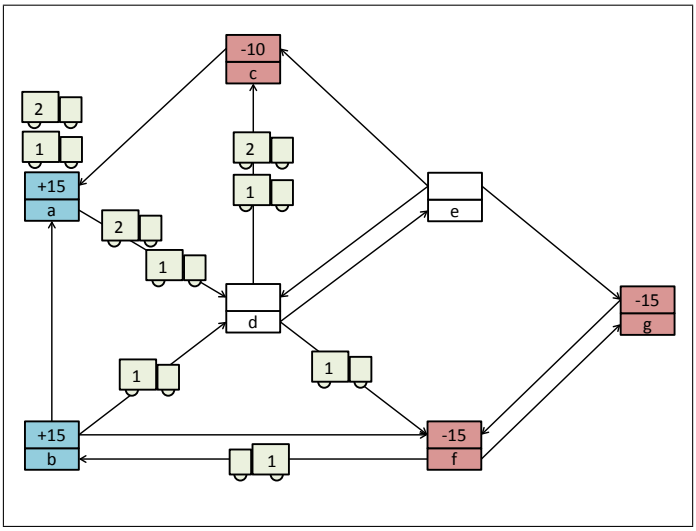


Figura 4.6: Rutas de los vehículos 1 y 2 tras INTERCAMBIAR VEHÍCULOS

Variables de entrada

$conv$, dur, $evcr$, $into$, $lonv$, $ltar$, $nuan$, $secv$, sf, $stck$, $tamc$, $tmin$

Variables de salida

$actpr$, $conv$, $lonv$, $ltar$, $secv$, $tamc$

Procedimientos llamados

ACORTAR RUTAS, APROVECHAR DEPÓSITOS, INTERCAMBIAR VEHÍCULOS, RETENER MATERIAL, SUSTITUIR VEHÍCULOS

Algoritmo

PASO 0

- Indicar que no hay que actualizar las precedencias

$$actpr = 0$$

PASO 1

- Si ya se ha aplicado el procedimiento EVITAR CRUCES, $evcr = 1$, ir al PASO 2
- (Todavía no se han evitado cruces, $evcr = 0$). Ejecutar el procedimiento RETENER MATERIAL, para retener parte del material que sale de nodos con demanda satisfecha baja

- Ejecutar el procedimiento APROVECHAR DEPÓSITOS, para eliminar flujo superfluo de material que pasa por los depósitos

PASO 2

- Ejecutar el procedimiento ACORTAR RUTAS, para eliminar trayectos innecesarios de los vehículos
- Ejecutar el procedimiento SUSTITUIR VEHÍCULOS, para sustituir vehículos para seguir eliminando trayectos innecesarios
- Ejecutar el procedimiento INTERCAMBIAR VEHÍCULOS, para intercambiar vehículos para disminuir la suma de distancias de regreso
- Si se ha producido algún cambio en alguno de los elementos de la solución, indicar que hay que actualizar las precedencias

$$actpr = 1$$

Procedimiento 24. - RECALCULAR PRECEDENCIAS

Obtención de las relaciones de precedencia y precedencia inmediata entre los arcos por haberse producido cambios en la solución tras la fase de refinamiento.

Variables de entrada

lonv, *secv*

Variables de salida

nump, *prec*, *pred*, *prei*, *pret*, *suci*, *suct*

Variables internas

apos	Arco sucesor principal para agregar precedencias
apre	Arco predecesor principal para agregar precedencias
ea	Etapa genérica en la ruta de un vehículo
ja	Vehículo genérico

Procedimientos llamados

AGREGAR PRECEDENCIAS

Algoritmo

- Inicializar las variables de almacenamiento de las relaciones de precedencia asignando el valor 0 a las siguientes variables, vectores y matrices

$$nump, prec, prei$$

- Para cada vehículo, ja, y cada par de etapas consecutivas, ea y $ea+1$, del vehículo:
  - Si el arco $k = secv_{ja,ea}$ no precede inmediatamente al arco $k' = secv_{ja,ea+1}$, $prei_{k,k'} = 0$, hacer:
    - Guardar los arcos k y k' como predecesor y precedido

$$apre = k$$
$$apos = k'$$

    - Ejecutar el procedimiento AGREGAR PRECEDENCIAS, para obligar a que el arco *apre* preceda inmediatamente al arco *apos* y actualizar las precedencias correspondientes

Procedimiento 25. - ACTUALIZAR EVENTOS

Actualización de los eventos de los nodos por haberse producido alguna mejora en la solución. Consta de cuatro procedimientos, tres de ellos introducidos en el Algoritmo Constructivo.

Variables de entrada

conv, *lonv*, *ltar*, *nuan*, *nump*, *pred*, *secv*, *suci*, *tamc*

Variables de salida

dur, *inst*, *into*, *nump*, *prec*, *pred*, *prei*, *pret*, *stck*, *suci*, *suct*, *tmin*

Procedimientos llamados

DURACIÓN, FIN ARCOS, ORDENAR EVENTOS, RECALCULAR PRECEDENCIAS

Algoritmo

- Ejecutar el procedimiento RECALCULAR PRECEDENCIAS, para obtener de nuevo las relaciones de precedencia
- Ejecutar el procedimiento DURACIÓN, para calcular las duraciones de los trayectos de los convoyes
- Ejecutar el procedimiento FIN ARCOS, para calcular los instantes mínimos de finalización de los trayectos
- Ejecutar el procedimiento para ORDENAR EVENTOS, ordenar los eventos en los nodos según sus instantes de ocurrencia y calcular los *stocks* correspondientes

Procedimiento 26. - RESERVAR SOLUCIÓN

Almacenamiento de las variables de decisión y variables objetivo de la solución actual. Esta solución se recuperará en caso de que después del procedimiento EVITAR CRUCES se obtenga otra solución peor.

Variables de entrada

f, fob, $load$, $secv$, $tmin$

Variables de salida

pf, $pfob$, $pload$, $psecv$, $ptmin$

Algoritmo

- Guardar los valores de las variables de decisión

$$\begin{aligned} pload_j &= load_j \quad \forall j \\ psecv_j &= secv_j \quad \forall j \\ ptmin_k &= tmin_k \quad \forall k \end{aligned}$$

- Guardar los valores de las funciones objetivo

$$\begin{aligned} pf_g &= f_g \quad \forall g \\ pfob &= fob \end{aligned}$$

Procedimiento 27. - EVITAR CRUCES

Detección y eliminación de material que es transportado entre dos nodos en sentidos inversos. La finalidad es tratar de evitar todos los cruces de material posibles. Esto no garantiza que la solución vaya a mejorar, incluso puede resultar infactible, por lo que previamente se guarda la solución original para recuperarla si es preciso.

Variables de entrada

$ltar$

Variables de salida

$evcr$, $ltar$

Algoritmo

- Para cada pareja de arcos, k, k', en los que se transporta material, $ltark > 0$ y $ltark' > 0$, y que unen dos nodos en sentidos opuestos, $ini_k = fin_{k'}$ y $ini_{k'} = fin_k$,

reducir el material transportado en ambos arcos hasta que en uno de ellos no se transporte nada

$$ltar_k = ltar_k - \text{mín}\{ltar_k, ltar_{k'}\}$$
$$ltar_{k'} = ltar_{k'} - \text{mín}\{ltar_k, ltar_{k'}\}$$

- Si se ha evitado algún cruce de material, actualizar el indicador correspondiente

$$evcr = 1$$

Siguiendo con el ejemplo que se está considerando, en el flujo original reflejado en la Figura 3.5 existía un cruce de material: 2 toneladas de material eran transportadas del nodo f al nodo g y 3 toneladas eran transportadas del nodo g al nodo f. Sin embargo, al aplicar el procedimiento RETENER MATERIAL, este cruce ha desaparecido eliminándose las 3 toneladas del arco $g - f$, como puede apreciarse en la Figura 4.1. Por cosiguiente, el procedimiento EVITAR CRUCES no modifica en nada la solución obtenida hasta el momento.

Procedimiento 28. - RECUPERAR SOLUCIÓN

Recuperación de la solución obtenida antes de aplicar el procedimiento EVITAR CRUCES por haber proporcionado este otra solución peor o infactible.

Variables de entrada

pf, $pfob$, $pload$, $psecv$, $ptmin$

Variables de salida

f, fob, $load$, $secv$, $tmin$

Algoritmo

- Recuperar los valores de las variables de decisión

$$load_j = pload_j \quad \forall j$$
$$secv_j = psecv_j \quad \forall j$$
$$tmin_k = ptmin_k \quad \forall k$$

- Recuperar los valores de las funciones objetivo

$$f_g = pf_g \quad \forall g$$
$$fob = pfob$$

4.3. Programa principal

En esta sección se presenta el procedimiento MEJORA, que actúa como programa principal del algoritmo.

Procedimiento 29. - MEJORA

Procedimientos llamados

ACTUALIZAR EVENTOS, EVITAR CRUCES, POSITIVAR, RESERVAR SOLUCIÓN, RECUPERAR SOLUCIÓN, REFINAR SOLUCIÓN, REPARTIR CARGAS

Algoritmo

PASO 0

- Calcular la proporción de demanda satisfecha óptima

$$pjus = \frac{qglobal}{\sum_{i \in ID} dem_i}$$

- Indicar que todavía no se han evitado cruces de material

$$evcr = 0$$

PASO 1

- Ejecutar el procedimiento REFINAR SOLUCIÓN, para eliminar elementos innecesarios de la solución
- Si no hay que actualizar las precedencias, $actpr = 0$, ir al PASO 3 (solución válida)
- (Hay que actualizar las precedencias, $actpr = 1$). Ejecutar el procedimiento ACTUALIZAR EVENTOS, para reorganizar los eventos y *stocks* en los nodos.
- Ejecutar el procedimiento POSITIVAR, para detectar y eliminar los posibles *stocks* negativos

PASO 2

- Ejecutar el procedimiento REPARTIR CARGAS, para repartir el material que se transporta por los arcos entre los vehículos de los convoyes correspondientes
- Evaluar la función objetivo, fob, de acuerdo a los pesos establecidos para los atributos

PASO 3

- Ejecutar el procedimiento RESERVAR SOLUCIÓN, para guardar la solución obtenida
- Ejecutar el procedimiento EVITAR CRUCES, para impedir cruces de material entre dos nodos
- Si la solución no ha cambiado, $evcr = 0$, FIN del procedimiento

- (La solución ha cambiado, $evcr = 1$). Ejecutar el procedimiento REFINAR SOLUCIÓN, para eliminar elementos innecesarios de la solución
- Si no hay que actualizar las precedencias, $actpr = 0$, ir al PASO 4 (nueva solución válida)
- (Hay que actualizar las precedencias, $actpr = 1$). Ejecutar el procedimiento ACTUALIZAR EVENTOS, para reorganizar los eventos y *stocks* en los nodos.
- Ejecutar el procedimiento POSITIVAR, para detectar y eliminar los posibles *stocks* negativos
- Si hay algún *stock* negativo, $negat = 1$:
  - Ejecutar el procedimiento RECUPERAR SOLUCIÓN, para recuperar la última solución válida
  - FIN del procedimiento

PASO 4

- Ejecutar el procedimiento REPARTIR CARGAS, para repartir el material que se transporta por los arcos entre los vehículos de los convoyes correspondientes
- Evaluar la función objetivo, fob, de acuerdo a los pesos establecidos para los atributos
- Si la nueva solución obtenida es mejor que la solución original, $fob \leq pfob$, FIN del procedimiento
- (La nueva solución obtenida es peor que la solución original, $fob > pfob$). Ejecutar el procedimiento RECUPERAR SOLUCIÓN, para recuperar la mejor solución obtenida

El diagrama de flujo que se muestra en la Figura 4.7 resume el funcionamiento del Algoritmo de Mejora.

En el ejemplo considerado, el flujo de vehículos y material reflejado en la Figura 4.3 ha resultado ser definitivo, aunque algunos vehículos han intercambiado parte de sus rutas con posterioridad. En la Figura 4.8 puede apreciarse la carga que transportan los vehículos en sus trayectos. Por ejemplo, el vehículo 1 (Figura 4.8a) inicia su ruta transportando 3 toneladas por los arcos $a-d$ y $d-f$ y depositándolas en el nodo f. A continuación, viaja vacío al nodo b, donde carga 3 nuevas toneladas que lleva al nodo c pasando por el nodo d. Los vehículos 10 y 12 no han sido utilizados en el reparto. El progreso que ha experimentado la solución tras aplicársele el algoritmo de Mejora ha sido sustancial. El tiempo del reparto ha disminuido un 18 %, el coste un 34 %, la medida de la equidad un 29 %, la medida de la prioridad un 11 % y la medida de la seguridad un 7 %. Sin embargo, la medida de la fiabilidad ha empeorado, aunque aumentando solo un 2 %.

Todos los procedimientos que componen el Algoritmo de Mejora pretenden mejorar al menos uno de los objetivos sin empeorar los demás. Sin embargo, como ya se comentó anteriormente, y como se puede observar en el ejemplo en el caso de la fiabilidad, existen

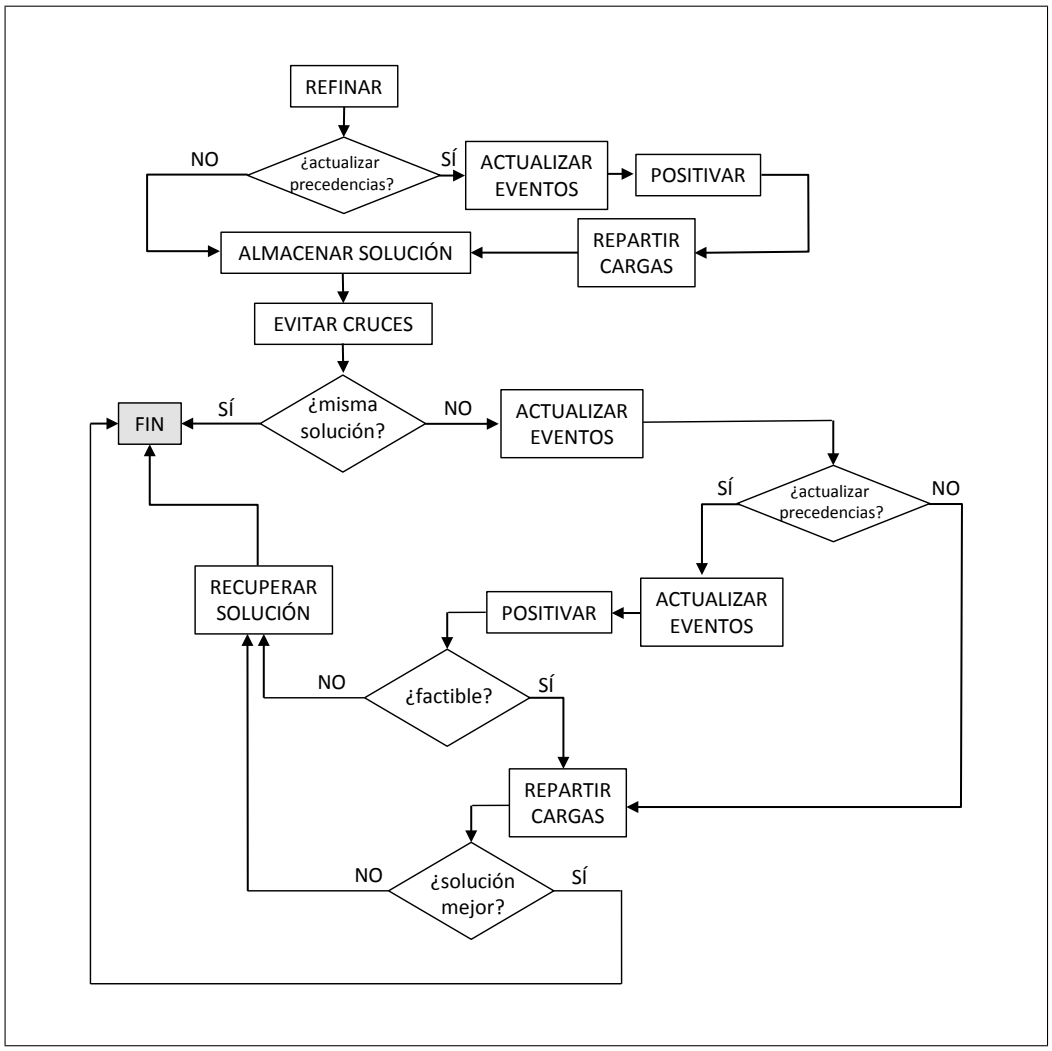
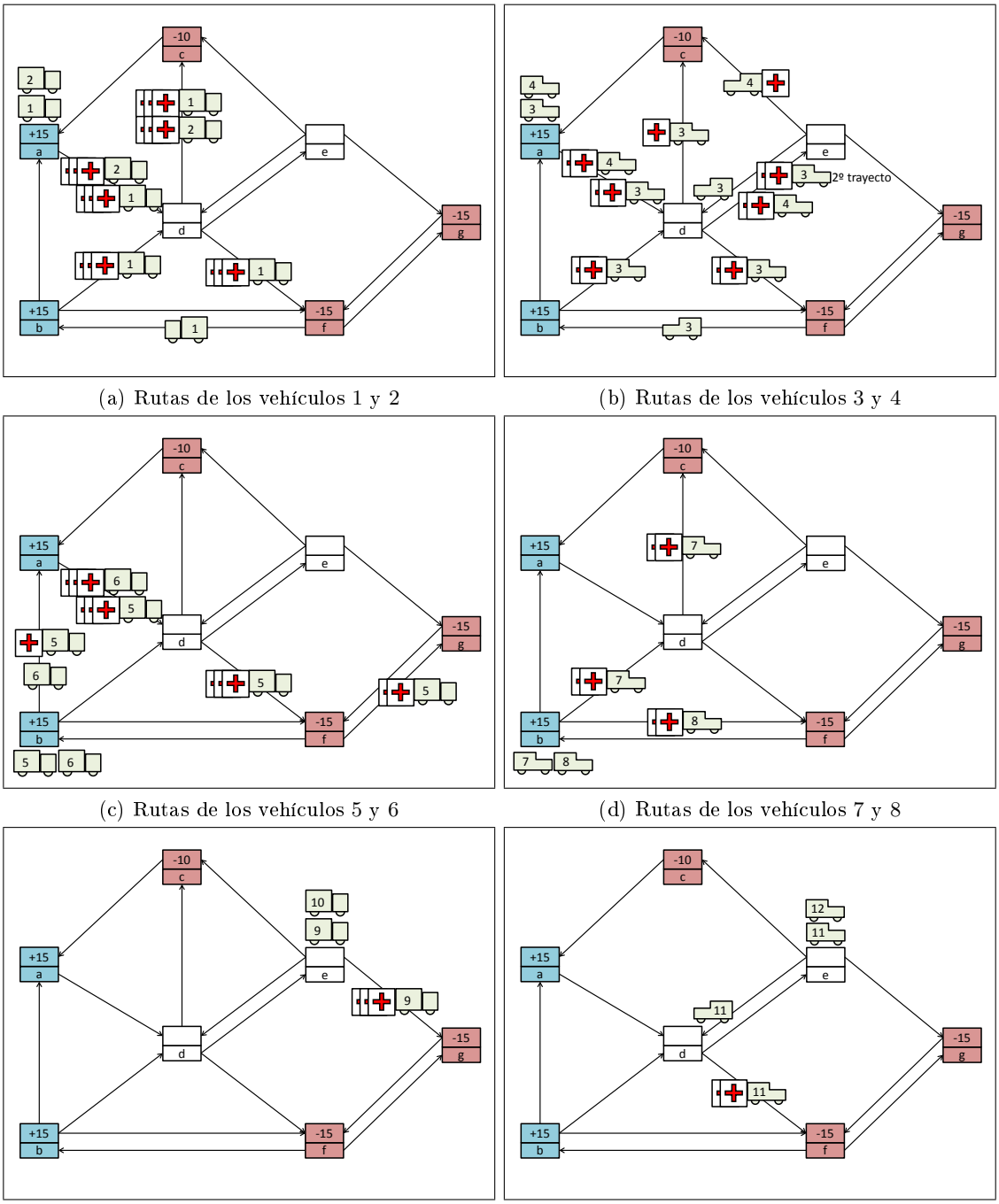


Figura 4.7: Diagrama de flujo de MEJORA

algunas excepciones. RETENER MATERIAL puede hacer empeorar la prioridad pero, como ya se indicó al establecer los atributos del problema en la Sección 2.2, la equidad y la prioridad no suelen presentarse combinados en un mismo problema por ser claramente antagónicos. ACORTAR RUTAS puede empeorar la medida de la seguridad, SUSTITUIR VEHÍCULOS puede empeorar las medidas de la seguridad y de la fiabilidad, e INTERCAMBIAR VEHÍCULOS, la medida de la fiabilidad. En cualquier caso, estos posibles empeoramientos son normalmente poco frecuentes e insignificantes en comparación con las mejoras producidas en los demás atributos. Además, aunque un procedimiento pueda perjudicar inicialmente a algún atributo, la aplicación completa del algoritmo habitualmente termina por compensar el posible empeoramiento del atributo. En todo caso, si se desea evitar este posible empeoramiento, por pequeño que sea, existe la opción de guardar la solución antes de que se le aplique cualquiera de estos procedimientos, para recuperarla en caso de que así se desee. Esta idea se ha utilizado al ejecutar EVITAR CRUCES, procedi-



(a) Rutas de los vehículos 1 y 2

(b) Rutas de los vehículos 3 y 4

(c) Rutas de los vehículos 5 y 6

(d) Rutas de los vehículos 7 y 8

(e) Rutas de los vehículos 9 y 10

(f) Rutas de los vehículos 11 y 12

Figura 4.8: Rutas definitivas

miento que, en algún caso, podría aumentar la duración del reparto.

Capítulo 5

Algoritmo GRASP

En el Capítulo 3 se ha descrito un algoritmo que permite obtener soluciones factibles para el problema objeto de estudio de este trabajo, mientras que el Capítulo 4 detalla un algoritmo para mejorar las soluciones que se hayan obtenido con anterioridad. Ambos procesos están fuertemente aleatorizados para que el conjunto de soluciones alcanzables sea lo más amplio posible. El siguiente paso, que se desarrollará en este capítulo y en el siguiente, es guiar la construcción de las soluciones de forma que se puedan obtener soluciones mejores (con valores más pequeños en la función objetivo) que mediante la mera repetición de los algoritmos Constructivo y de Mejora. Para ello se desarrollarán dos algoritmos basados en sendas metaheurísticas. En este capítulo se presenta un algoritmo basado en la metaheurística GRASP, mientras que en el Capítulo 6, el algoritmo que se presentará se basa en la metaheurística colonia de hormigas.

La mecánica general del GRASP, introducida originalmente en Feo y Resende (1989), consiste en construir iterativamente soluciones de forma aleatorizada usando una filosofía *greedy*, a través de funciones que se actualizan a lo largo del proceso constructivo. Cada solución obtenida es sometida a un proceso de búsqueda local, con el objeto de mejorarla en la medida de lo posible. En el algoritmo que aquí se presenta, la información de las soluciones obtenidas en iteraciones anteriores se transmite mediante un conjunto élite, formado por soluciones diferentes entre sí y a la vez con valores pequeños en la función objetivo. Los conjuntos élite han sido utilizados en diferentes metaheurísticas (por ejemplo, en Greistorfer (2003) se usa en la búsqueda tabú), incluida la metaheurística GRASP (Fleurent y Glover, 1999; Ribeiro *et al.*, 2006). Además, a cada solución obtenida se le aplicará el Algoritmo de Mejora, cuyos procedimientos SUSTITUIR VEHÍCULOS (con una ligera modificación) e INTERCAMBIAR VEHÍCULOS conforman la fase de búsqueda local propia de la metaheurística.

El algoritmo GRASP que se expone en este capítulo consta de tres fases. En primer lugar, se efectúa la fase de preproceso (procedimiento PREPROCESO) del Algoritmo Constructivo. Esta fase se realiza una sola vez, independientemente del número total de soluciones que se desee obtener. Otra tarea previa que se debe realizar antes de comenzar a construir soluciones es obtener cotas superiores para las funciones *greedy* que se usarán en la última fase.

En la segunda fase se construye el conjunto élite (procedimiento OBTENER ÉLITE), cuyo cardinal se denominará *nelite* (parámetro que habrá sido fijado de antemano). Este conjunto élite servirá para guiar la construcción de nuevas soluciones en la última fase. Las primeras *nelite* soluciones obtenidas forman directamente el conjunto élite inicial. A continuación, se construyen nuevas soluciones hasta tener *telite* en total, actualizando el conjunto élite tras cada nueva solución (procedimiento ACTUALIZAR ÉLITE) de forma que las soluciones del conjunto élite guarden siempre el mejor equilibrio posible entre función objetivo y diversidad. Existen muchos aspectos para medir lo diferentes que son dos soluciones entre sí. En este trabajo se han propuesto, por considerarse especialmente relevantes, cuatro tipos de diversidad, que serán agregados para definir una medida global de la diversidad. El primer tipo está basado en los vehículos de cada clase que circulan por los arcos, el segundo en la cantidad de material que circula por los arcos, el tercero en la cantidad de material que reciben los nodos de demanda, y el cuarto en las relaciones de precedencia entre los arcos. En esta fase, todas las soluciones se obtienen aplicando los algoritmos Constructivo y de Mejora descritos en los capítulos anteriores.

En la última y más importante fase (procedimiento GUIAR CONSTRUCCIÓN), se usa el conjunto élite obtenido en la fase anterior para guiar la construcción de soluciones, conjuntamente con funciones *greedy*. Para ello se han modificado los algoritmos Constructivo y de Mejora introduciendo en algunos de sus procedimientos funciones *greedy* y funciones intensidad. Con el fin de distinguirlos, los procedimientos modificados presentes en el Algoritmo GRASP serán nombrados añadiendo un "2" a los nombres de los procedimientos originales. Por un lado, las funciones *greedy* priman los elementos candidatos a incorporar a la solución que aportan una mayor mejora inmediata. Las funciones *greedy* serán utilizadas en el diseño de itinerarios (CONSTRUIR RUTAS 2) y en la determinación del flujo (ENVIAR FLUJO 2). Sus expresiones concretas dependerán del atributo o atributos que se estén considerando y de la etapa de la construcción en la que se requieran. Por otro lado, las funciones intensidad priman los elementos más semejantes a sus recíprocos en las soluciones del conjunto élite. Concretamente, se denominará intensidad de un elemento a la proporción de veces que dicho elemento aparece en las soluciones del conjunto élite. Cuanta más intensidad presente un elemento, más probable será su inclusión en la solución que se esté construyendo. Se usarán funciones intensidad en los procedimientos importantes de la construcción (CONSTRUIR RUTAS 2 y ENVIAR FLUJO 2) pero también en la mejora (RETENER MATERIAL 2, APROVECHAR DEPÓSITOS 2 y ACORTAR RUTAS 2). No obstante, siempre se mantiene de forma parcial el carácter aleatorio de los algoritmos. Además, con el objetivo de permitir al algoritmo escapar de mínimos locales no deseados y para aprovechar la cada vez mayor potencia del conjunto élite, que para este fin se sigue actualizando cada vez que se construye una nueva solución, la influencia de las funciones *greedy* va decreciendo con el transcurso de las iteraciones a la vez que aumenta la influencia de las funciones intensidad. Una vez finalizada cada ejecución completa, el algoritmo GRASP proporciona la mejor solución encontrada durante todo el proceso, que para tal fin se irá almacenando y actualizando como parte del conjunto élite.

En este capítulo se incluyen resúmenes de los procedimientos modificados con los elementos novedosos respecto de sus respectivos procedimientos originales. Los procedimientos

modificados completos pueden verse en sendos apéndices.

En las secciones correspondientes a los procedimientos pertinentes, se mostrará la evolución del conjunto élite cuando se aplica el algoritmo GRASP al ejemplo de la Figura 3.1 introducido en el Capítulo 3, especificando cuál es la mejor solución obtenida. Con el fin de facilitar la comprensión del proceso, se ha tomado como función objetivo únicamente el coste total del reparto.

5.1. Parámetros y variables

En esta sección se presentan los elementos que intervienen en el algoritmo GRASP, si bien se omiten los parámetros y variables definidos en los algoritmos Constructivo y de Mejora, descritos en los capítulos anteriores.

Conjuntos e índices adicionales

$d, d' \in D$	Tipos de diversidad
$o, o' \in O$	Soluciones del conjunto élite

Parámetros de la metaheurística

β	Peso de la diversidad, respecto de la función objetivo, para la ponderación de la calidad de una solución del conjunto élite
γ	Peso de la función *greedy*, respecto del azar, para la ponderación de un candidato a movimiento en cualquier fase de la construcción de una solución
σ_1	Complemento a uno del peso de la intensidad al comienzo de la última fase del GRASP
σ_2	Velocidad de crecimiento del peso de la intensidad
$nelite$	Número de soluciones de que consta el conjunto élite
$telite$	Número total de soluciones obtenidas en la fase de construcción del conjunto élite
$tsol$	Número total de soluciones obtenidas en la aplicación del método GRASP

Otros parámetros

$cclas_c$	Número de vehículos de clase c
cfl_g	Cota para la función *greedy* del objetivo g en ENVIAR FLUJO 2
cru_g	Cota para la función *greedy* del objetivo g en CONSTRUIR RUTAS 2
$pesotot$	Peso total de las funciones *greedy* en CONSTRUIR RUTAS 2

Variables de decisión

$eload_{j,mej}$	Secuencia de cantidades transportadas por el vehículo j hasta que termina su reparto en la mejor solución del conjunto élite
$esecv_{j,mej}$	Secuencia de arcos (ruta) que recorre el vehículo j hasta que termina su reparto en la mejor solución del conjunto élite

$etmin_{k,mej}$	Instante mínimo en el que el convoy que recorre el arco k finaliza su trayecto en la mejor solución del conjunto élite

Variables objetivo

$ef_{g,o}$	Valor del objetivo g en la solución élite o
$efob_o$	Valor de la función objetivo global en la solución élite o
$fobmin$	Mejor valor obtenido para la función objetivo global

Variables auxiliares

α_d	Peso para cada tipo de diversidad d
$div_{d,o,o'}$	Diversidad de tipo d entre las soluciones élite o y o'
$diver_{o,o'}$	Diversidad total entre las soluciones élite o y o'
$divert$	Diversidad total del conjunto élite
$eload_{j,o}$	Secuencia de cantidades transportadas por el vehículo j hasta que termina su reparto en la solución élite o
$elonv_{j,o}$	Número de arcos recorridos por el vehículo j en la solución élite o
$eltar_{k,o}$	Cantidad total de material que transporta el convoy correspondiente al arco k en la solución élite o
$fobt$	Suma de los valores de la función objetivo en las soluciones del conjunto élite
$eprec_{k,k',o}$	Variable binaria que indica si el arco k precede al arco k' en la solución élite o
$esecv_{j,o}$	Secuencia de arcos (ruta) que recorre el vehículo j hasta que termina su reparto en la solución élite o
$esf_{i,o}$	Cantidad de material que queda en el nodo i al final del reparto en la solución élite o
$etmin_{k,o}$	Instante mínimo en el que el convoy que recorre el arco k finaliza su trayecto en la solución élite o
mej	Mejor solución del conjunto élite
$tamclas_{c,k,o}$	Número de vehículos de la clase c que recorren el arco k en la solución élite o

5.2. Procedimientos

Procedimiento 30. - ACOTAR GREEDY

Obtención de las cotas superiores para las funciones *greedy* que se usarán tanto en la construcción de las rutas como en la determinación de las cantidades de material que se transportarán. En el procedimiento CONSTRUIR RUTAS 2 no se consideran funciones *greedy* para los atributos equidad y prioridad, ya que ambos dependen de las cantidades de material recibidas en los nodos de demanda, mientras que en CONSTRUIR RUTAS 2 sólo se determinan los itinerarios de los vehículos. Por lo tanto, en el procedimiento que se describe a continuación, se calcula la suma de los pesos de los atributos considerados en CONSTRUIR RUTAS 2 para poder normalizar la expresión de la función *greedy* agregada

correspondiente a dicho procedimiento.

Algoritmo

- Cotas para CONSTRUIR RUTAS 2:
  - Cota para el tiempo. Máximo tiempo que requiere un vehículo para recorrer un arco
$$cru_1 = \max_k \left\{ \frac{dist_k}{\min\{vela_k, \min_t\{velv_t\}\}} \right\}$$
  - Cota para el coste. Máximo coste fijo que requiere un vehículo para recorrer un arco. Se multiplica por 2 puesto que se requiere regresar, y en el peor de los casos habrá que hacerlo por el mismo camino
$$cru_2 = \max_k \left\{ 2 \cdot dist_k \cdot \max_t\{cosf_{t,k}\} \right\}$$
  - Cota para la seguridad. Máxima probabilidad de asalto en un arco
$$cru_5 = \max_k \{p_k\}$$
  - Cota para la fiabilidad. Máxima probabilidad de que un arco no esté transitable
$$cru_6 = \max_k \{1 - r_k\}$$
- Calcular el peso total para CONSTRUIR RUTAS 2:
$$pesotot = peso_1 + peso_2 + peso_5 + peso_6$$
- Cotas para ENVIAR FLUJO 2:
  - La cota para el tiempo se obtiene en el propio procedimiento ENVIAR FLUJO 2, una vez conocidas las rutas de los vehículos que se han construido
  - Cota para el coste. Máximo coste de transporte por unidad de carga que puede suponer a un vehículo recorrer un arco
$$cfl_2 = \max_{k,t} \{dist_k \cdot cosv_{t,k}\}$$
  - Cota para la equidad. Máximo entre la proporción de demanda satisfecha óptima y su complemento a uno
$$cfl_3 = \max\{pjus, 1 - pjus\}$$
  - Cota para la prioridad
$$cfl_4 = 1$$
  - Cota para la seguridad. Máxima probabilidad de asalto en un arco
$$cfl_5 = \max_k \{p_k\}$$
  - Cota para la fiabilidad. Máxima probabilidad de que un arco no esté transitable
$$cfl_6 = \max_k \{1 - r_k\}$$

Procedimiento 31. - GUARDAR SOLUCIÓN

Almacenamiento de los elementos importantes de las soluciones del conjunto élite e indicación de la mejor solución obtenida. Las primeras *nelite* soluciones que se obtienen se guardan directamente como soluciones élite. A partir de entonces, cada nueva solución obtenida se guarda en la posición $nelite + 1$ como candidata a entrar en el conjunto élite.

Variables de entrada

guar Posición del conjunto élite en la que se debe guardar la nueva solución

$conv$, f, fob, $fobmin$, $load$, $lonv$, $ltar$, mej, $prec$, $secv$, sf, $tmin$

Variables de salida

ef, $efob$, $eload$, $elonv$, $eltar$, $eprec$, $esecv$, esf, $etmin$, $fobmin$, mej, $tamclas$

Algoritmo

- Guardar los valores de las funciones objetivo

$$ef_{g,guar} = f_g \quad \forall g$$
$$efob_{guar} = fob$$

- Guardar las características de la solución

$$elonv_{j,guar} = lonv_j \quad \forall j$$
$$eltar_{k,guar} = ltar_k \quad \forall k$$
$$esecv_{j,guar,e} = secv_{j,e} \quad \forall j,\ \forall e \ /\ 1 \leq e \leq lonv_j$$
$$eload_{j,guar,e} = load_{j,e} \quad \forall j,\ \forall e \ /\ 1 \leq e \leq lonv_j$$
$$esf_{i,guar} = sf_i \quad \forall i$$
$$etmin_{k,guar} = tmin_k \quad \forall k$$
$$eprec_{k,k',guar} = prec_{k,k'} \quad \forall k, k'$$

- Calcular el número de vehículos de cada clase que recorren cada arco

$$tamclas_{c,k,guar} = \left| j \in conv_k / clas_{tipv_j,ori_j} = c \right| \quad \forall k,\ \forall c$$

- Si la nueva solución es la mejor que se ha obtenido hasta el momento, $fob < fobmin$:
  - Actualizar el mejor valor obtenido para la función objetivo global

$$fobmin = fob$$

  - Actualizar el ordinal de la mejor solución

$$mej = guar$$

Procedimiento 32. - OBTENER DIVERSIDAD

Cálculo de la diversidad entre dos soluciones. Consta de cuatro sumandos, uno para cada uno de los tipos de diversidad, ponderados mediante los parámetros α_d. El primero de ellos mide la diferencia entre el número de vehículos de cada clase que transitan por los arcos. El segundo sumando mide la diferencia entre el flujo de material que es transportado por los arcos. El tercer sumando mide la diferencia entre las cantidades de material que llegan a los nodos de demanda. El cuarto sumando mide la diferencia entre las precedencias creadas entre los arcos.

Variables de entrada

$o1, o2$ Soluciones élite de las que calcular su diversidad

$eltar$, $eprec$, esf, $tamclas$

Variables de salida

div, $diver$

Algoritmo

- Calcular la diversidad debida al número de vehículos de cada clase que circulan por los arcos

$$div_{1,o1,o2} = \frac{1}{|K||C|} \sum_{k \in K, c \in C} \frac{|tamclas_{c,k,od1} - tamclas_{c,k,o2}|}{cclas_c}$$

- Calcular la diversidad debida a la cantidad de material que circula por los arcos

$$div_{2,o1,o2} = \frac{1}{|K| qglobal} \sum_{k \in K} |eltar_{k,o1} - eltar_{k,o2}|$$

- Calcular la diversidad debida a la cantidad de material que se reparte a los nodos de demanda al final del reparto

$$div_{3,o1,o2} = \frac{1}{|ID|} \sum_{i \in ID} \frac{|esf_{i,o1} - esf_{i,o2}|}{dem_i}$$

- Calcular la diversidad debida a las precedencias entre los arcos

$$div_{4,o1,o2} = \frac{1}{\binom{|K|}{2}} \sum_{k,k' \in K/k<k'} |eprec_{k,k',o1} - eprec_{k,k',o2}|$$

- Calcular la diversidad global de la pareja de soluciones

$$diver_{o1,o2} = \sum_{d \in D} \alpha_d div_{d,o1,o2}$$
$$diver_{o2,o1} = diver_{o1,o2}$$

Procedimiento 33. - ACTUALIZAR ÉLITE

Actualización del conjunto élite. Cada vez que se obtiene una nueva solución, se decide si esta debe sustituir a alguna de las que conforman el conjunto élite, atendiendo tanto a la función objetivo de las soluciones como a la diversidad entre ellas. El parámetro β indica el peso que tiene cada uno de estos dos factores. El equilibrio entre función objetivo y diversidad determina la calidad del conjunto élite. Se denominará conjunto élite ampliado al conjunto que resulta de añadir la nueva solución al conjunto élite, antes de eliminar ninguna.

Variables de entrada

$diver$, $divert$, $eact$, ef, $efob$, $eload$, $elonv$, $eltar$, $eprec$, $esecv$, esf, $etmin$, $fobmin$, $fobt$, mej, $tamclas$

Variables de salida

$divert$, ef, $efob$, $eload$, $elonv$, $eltar$, $eprec$, $esecv$, esf, $etmin$, $fobmin$, $fobt$, mej, $tamclas$

Variables internas

$calsin_o$ Calidad del conjunto élite resultante al quitar del conjunto élite ampliado la solución o

$guar$

Algoritmo

- Para cada solución del conjunto élite ampliado (conjunto élite junto con la nueva solución), excepto para la solución con mejor función objetivo, obtener la calidad del conjunto resultante de eliminar dicha solución

$$calsin_o = \beta \frac{\dfrac{divert - \sum_{o' \in O} diver_{o,o'}}{\binom{nelite}{2}}}{\dfrac{divert}{\binom{nelite+1}{2}}} - (1-\beta)\frac{\dfrac{fobt - efob_o}{nelite}}{\dfrac{fobt}{nelite+1}} \quad \forall o \neq mej$$

- Elegir la solución que, al eliminarla, dé lugar al mejor conjunto élite

$$guar = \arg\max\{calsin_o\}$$

- Actualizar la diversidad total y la suma de funciones objetivo globales

$$divert = divert - \sum_{o \in O} diver_{o,guar}$$

$$fobt = fobt - efob_{guar}$$

- Si la peor es la última obtenida, $guar = nelite + 1$, FIN del procedimiento
- Ejecutar el procedimiento GUARDAR SOLUCIÓN, para guardar la solución $nelite+1$ como solución $guar$ del conjunto élite

Procedimiento 34. - CONSTRUIR RUTAS 2

Modificación del procedimiento CONSTRUIR RUTAS en el que se introducen las funciones *greedy* e intensidad para guiar la construcción de los itinerarios. Las funciones *greedy* se establecen teniendo en cuenta cuatro de los atributos. La equidad y la prioridad no son tenidas en cuenta en las funciones *greedy*, ya que, durante la ejecución de este procedimiento, no se ha determinado todavía el material de ayuda que llega a los nodos de demanda. Al igual que se hará en el resto de procedimientos modificados, en este capítulo se presenta un resumen indicando las novedades respecto del procedimiento original, en este caso CONSTRUIR RUTAS. El procedimiento completo se incluye en el Apéndice A.1.

Es necesario definir dos variables de entrada del procedimiento:

λ	Peso de la función intensidad para la ponderación de un candidato a movimiento en cualquier fase de la construcción de una solución
$tcpar_{c,k}$	Número de vehículos de la clase c que circulan por el arco k en la solución actual

También se definen las siguientes variables internas:

$gr_{g,mov}$	Función *greedy* para el objetivo g del movimiento mov
$greedy_{mov}$	Función *greedy* global del movimiento mov
int_{mov}	Función intensidad del movimiento mov

En cada iteración, para cada movimiento mov candidato a añadirse a la solución en construcción, que en este caso es un par vehículo-arco, se calculan la función intensidad y la función *greedy*. La intensidad del movimiento mov, asociado al vehículo j y al arco k, es el número de soluciones élite para las que el número de vehículos de la clase del vehículo j que circulan por el arco k es mayor que en la solución que se está construyendo.

$$int_{mov} = |o \in O / tamclas_{c,k,o} > tcpar_{c,k}| \quad \text{siendo } c = clas_{tipv_j, ori_j}$$

Las funciones *greedy* del movimiento mov son las siguientes:

- Función *greedy* para el tiempo. Tiempo que tarda el vehículo j en recorrer el arco k.

$$gr_{1,mov} = \frac{dist_k}{\text{mín}\{vela_k, velv_{tipv_j}\}}$$

- Función *greedy* para el coste. Coste que le supone al vehículo j recorrer el arco k más la diferencia entre el coste de regreso desde el final del arco y el coste de regreso

desde el inicio del arco.

$$gr_{2,mov} = dist_k \cdot cosf_{tipv_j,k} + creg_{c,fin_k} - creg_{c,ini_k}$$
$$\text{siendo } c = clas_{tipv_j,ori_j}$$

- Función *greedy* para la seguridad. Probabilidad de que el convoy que recorre el arco k, incluyendo al vehículo j, sufra un asalto.

$$gr_{5,mov} = FP(p_k, pm_k, tamc_k + 1)$$

- Función *greedy* para la fiabilidad. Probabilidad de que el arco k no esté transitable.

$$gr_{6,mov} = 1 - r_k$$

- La función *greedy* global del movimiento es la media ponderada de las distancias normalizadas de las funciones *greedy* a sus cotas en los criterios considerados.

$$greedy_{mov} = \frac{1}{pesotot} \sum_{g \in \{1,2,5,6\}} \left(\frac{peso_g(cru_g - gr_{g,mov})}{cru_g} \right)$$

La ponderación de cada movimiento mov se calcula mediante la siguiente fórmula:

$$kpon_{mov} = \lambda \frac{int_{mov}}{nelite} + (1 - \lambda)(\gamma \cdot greedy_{mov} + (1 - \gamma))$$

La variable λ representa el peso que recibe la función intensidad respecto de la función *greedy* y el azar, mientras que el parámetro γ representa el peso que recibe la función *greedy* respecto del azar. La probabilidad de seleccionar cada movimiento mov siempre será proporcional a la ponderación del movimiento. El índice mov' recorre todos los movimientos posibles.

$$prob_{mov} = \frac{kpon_{mov}}{\sum_{mov'} kpon_{mov'}}$$

La ponderación está así formulada para evitar que haya movimientos con probabilidad nula. Esto se consigue mediante el sumando $(1 - \gamma) \cdot 1$, donde la unidad representa la parte del azar de la expresión. De este modo, cualquier movimiento posible tendrá una probabilidad positiva y por lo tanto podrá ser seleccionado aunque sea con una mínima probabilidad. Se pretende diversificar así el conjunto de soluciones que pueden obtenerse. El valor de λ irá creciendo paulatinamente a medida que se ejecuta el Algoritmo GRASP, con el fin de otorgar más importancia a las funciones intensidad respecto al resto de términos. En algunos procedimientos posteriores se describirán situaciones donde se utilicen funciones intensidad, pero no funciones *greedy*, para determinar las probabilidades con que seleccionar los posibles movimientos. En esos casos, la ponderación de cada movimiento se expresará de la siguiente forma:

$$kpon_{mov} = \lambda \frac{int_{mov}}{nelite} + (1 - \lambda)$$

Nuevamente, el sumando $(1 - \lambda)$ permitirá que todos los candidatos tengan una probabilidad positiva de ser elegidos.

Además de para la construcción de las rutas, la información proporcionada por las soluciones del conjunto élite se usa también para introducir un criterio de parada más restrictivo: cuando el máximo entre las intensidades de los movimientos candidatos no alcance un determinado umbral, la construcción terminará. Si con las rutas creadas no pudiese repartirse el material deseado, se disminuirá el valor del umbral para intentar prolongar las rutas. La variable *cont*, ya presente en el Algoritmo Constructivo, indicará, cada vez que se ejecute el procedimiento CONSTRUIR RUTAS 2, si las rutas de la solución actual se deben prolongar o si debe comenzarse la construcción desde el principio.

Procedimiento 35. - ENVIAR FLUJO 2

Modificación del procedimiento ENVIAR FLUJO en el que se introducen las funciones *greedy* e intensidad para guiar el envío de material. Se define un parámetro interno:

$cosvm_k$ Coste variable medio de los vehículos que recorren el arco k

Este parámetro, que se utilizará en una de las funciones *greedy*, se calcula, para cada arco k recorrido por algún vehículo, mediante la siguiente expresión:

$$cosvm_k = \frac{1}{tamc_k} \sum_{j \in conv_k} cosv_{tipv_j,k}$$

Se debe calcular también la cota para la función *greedy* para el tiempo: instante más tardío en el que un convoy termina de recorrer un arco.

$$cfl_1 = \max_k \{tmin_k\}$$

En este procedimiento existen dos situaciones en las que tomar decisiones. Por una parte, se debe elegir el depósito a etiquetar desde la fuente, para lo que no se utilizarán funciones *greedy* aunque sí función intensidad. Esta elección debe realizarse cuando el nodo de referencia es la fuente, $npar = f$. En este caso, la intensidad de cada depósito i se calcula como el número de soluciones élite para las que el material neto que sale del depósito es mayor que el flujo desde la fuente al depósito en la solución que se está construyendo.

$$int_i = |o \in O / qav_i - esf_{i,o} > flujo_{f,i}|$$

Por otra parte, se debe elegir el nodo a etiquetar desde cualquier nodo distinto de la fuente, para lo que se tienen en cuenta tanto las funciones *greedy* como la función intensidad. Cada nodo candidato i corresponde al final de un arco k que parte del nodo $npar$, último nodo etiquetado. La intensidad de cada nodo candidato i es el número de soluciones élite para las que el material que circula por el arco es mayor que el flujo por el arco en la solución que se está construyendo.

$$int_i = |o \in O / eltar_{k,o} > flujo_{npar,fin_k}|$$

Las funciones *greedy* del nodo candidato i son las siguientes:

- Función *greedy* para el tiempo. Instante en el que el convoy termina de recorrer el arco k.

$$gr_{1,i} = tmin_k$$

- Función *greedy* para el coste. Coste medio que supone transportar una unidad de material por el arco k.

$$gr_{2,i} = dist_k \cdot cosvm_k$$

- Funciones *greedy* para la equidad y la prioridad.

$$gr_{3,i} = 0$$
$$gr_{4,i} = 1$$

- Función *greedy* para la seguridad. Probabilidad de que el convoy que recorre el arco k sufra un asalto.

$$gr_{5,i} = FP(p_k, pm_k, tamc_k)$$

- Función *greedy* para la fiabilidad. Probabilidad de que el arco k no esté transitable.

$$gr_{6,i} = 1 - r_k$$

- La función *greedy* global del candidato i es la media ponderada de las distancias normalizadas de las funciones *greedy* a sus cotas.

$$greedy_i = \sum_g \left(\frac{peso_g(cfl_g - gr_{g,i})}{cfl_g} \right)$$

Etiquetar el sumidero supone implícitamente decidir el nodo de demanda al que llega el flujo en la iteración actual del algoritmo de Ford-Fulkerson, por lo que el sumidero requiere un tratamiento diferente respecto al resto de nodos, especialmente en cuanto a los atributos equidad y prioridad, que dependen del material que llega a los nodos de demanda. Así, si el nodo de referencia es un nodo de demanda, $dem_{npar} > 0$, además del resto de nodos candidatos, el sumidero se considera también como candidato. La intensidad del sumidero, s, es el número de soluciones élite para las que el material que termina en el nodo de demanda es mayor que el flujo desde el nodo de demanda al sumidero en la solución que se está construyendo.

$$int_s = |o \in O / esf_{npar,o} > flujo_{npar,s}|$$

Las funciones *greedy* del sumidero son las siguientes:

- Funciones *greedy* para el tiempo y el coste.

$$gr_{1,s} = 0$$
$$gr_{2,s} = 0$$

- Función *greedy* para la equidad. Diferencia entre la distancia a la proporción de demanda satisfecha óptima eligiendo este movimiento y sin elegirlo.

$$gr_{3,s} = \left| pjus - \frac{flujo_{npar,s} + \min\{delta_{npar}, capr_{npar,s}\}}{dem_i} \right| - \left| pjus - \frac{flujo_{npar,s}}{dem_{npar}} \right|$$

- Función *greedy* para la prioridad. Complemento a uno de la contribución del movimiento a la proporción de demanda satisfecha del nodo multiplicada por su nivel de prioridad.

$$gr_{4,s} = 1 - \frac{\text{mín}\{delta_{npar}, capr_{npar,s}\} \cdot nivp_{npar}}{dem_{npar}}$$

- Funciones *greedy* para la seguridad y la fiabilidad.

$$gr_{5,s} = 0$$
$$gr_{6,s} = 0$$

- La función *greedy* global asociada al sumidero es, como en el resto de los casos, la media ponderada de las distancias normalizadas de las funciones *greedy* a sus cotas.

$$greedy_s = \sum_g \left(\frac{peso_g(cfl_g - gr_{g,s})}{cfl_g} \right)$$

Además del uso de funciones *greedy* e intensidad, existe otra diferencia respecto al procedimiento original ENVIAR FLUJO: en caso de que no logre enviarse la cantidad de material establecida, la solución en construcción se redirigirá a CONSTRUIR RUTAS 2 para prolongar, si es posible, los itinerarios de los vehículos.

En el Apéndice A.2 se describe el procedimiento completo.

Procedimiento 36. - CONSTRUCTIVO 2

Modificación del procedimiento CONSTRUCTIVO, programa principal del Algoritmo Constructivo descrito en el Capítulo 3, en el que se introducen las funciones *greedy* e intensidad en algunos de los procedimientos empleados en la construcción de la solución, concretamente, en los procedimientos CONSTRUIR RUTAS y ENVIAR FLUJO.

Las diferencias respecto al procedimiento CONSTRUCTIVO son las siguientes:

- Se omite el PASO 0 (ejecutar PREPROCESO).
- Se usa la variable binaria *cont* para posibilitar que se prolonguen las rutas ya construidas cuando no haya sido posible repartir la cantidad planeada.
- Se ejecutan los procedimientos modificados CONSTRUIR RUTAS 2 y ENVIAR FLUJO 2 en lugar de sus correspondientes procedimientos originales.
- Se omite la evaluación de la función objetivo y la ejecución del procedimiento REPARTIR CARGAS.

El procedimiento CONSTRUCTIVO 2, completamente detallado, puede encontrarse en el Apéndice A.3.

Procedimiento 37. - RETENER MATERIAL 2

Modificación del procedimiento RETENER MATERIAL en el que se introducen las funciones intensidad. Tanto en este como en el resto de procedimientos del Algoritmo de Mejora no se usan funciones *greedy* por considerarse, por un lado, que no existe una forma clara de estimar a priori cómo influye un movimiento en las medidas de los atributos, y por otro lado, que el uso de las funciones *greedy* en la fase constructiva es suficiente para ayudar a obtener soluciones factibles razonablemente buenas. A continuación, se describe la forma de introducir estas funciones intensidad y su uso para la elección de nodos candidatos a etiquetar. Este procedimiento, completamente detallado, puede encontrarse en el Apéndice A.4.

Si el nodo de referencia es la fuente, $npar = f$, se consideran como candidatos los nodos de demanda necesitados, esto es, los que reciben una cantidad de material inferior a la que les correspondería en un reparto perfectamente equitativo. La intensidad de cada uno de ellos, i, es el número de soluciones élite para las que el material que termina en el nodo de demanda necesitado es mayor que en la solución que se está construyendo.

$$int_i = |o \in O / esf_{i,o} > sf_i|$$

Si el nodo de referencia no es la fuente, cada nodo candidato i corresponde al final de un arco k que parte del nodo $npar$, último nodo etiquetado. La intensidad es el número de soluciones élite para las que el material que circula por el arco es menor que la diferencia entre el material que circula por el arco y el flujo correspondiente en la solución que se está construyendo.

$$int_i = |o \in O / eltar_{k,o} < ltar_k - flujo_{npar,i}|$$

Si el nodo de referencia es un nodo de demanda no necesitado, es decir, un nodo de demanda que recibe más material del que le correspondería en un reparto perfectamente equitativo, hay que considerar la posibilidad de llegar al sumidero. La intensidad en este caso es el número de soluciones élite para las que el material que termina en el nodo de demanda no necesitado, $npar$, es menor que en la solución que se está construyendo.

$$int_s = |o \in O / esf_{npar,o} < sf_{npar}|$$

Tras cada iteración del algoritmo de Ford-Fulkerson, se deben actualizar las variables sf_i, que representan las cantidades de material que quedarían en los nodos al final del reparto.

Procedimiento 38. - APROVECHAR DEPÓSITOS 2

Modificación del procedimiento APROVECHAR DEPÓSITOS en el que se introducen las funciones intensidad. A continuación, se describe la forma de introducir estas funciones intensidad y su uso para la elección de nodos candidatos a etiquetar. Este procedimiento, completamente detallado, puede encontrarse en el Apéndice A.5.

Si el nodo de referencia es la fuente, $npar = f$, se consideran como candidatos los depósitos. La intensidad de cada uno de ellos, i, es el número de soluciones élite para las que el material que queda en el depósito es menor que en la solución que se está construyendo.

$$int_i = |o \in O / esf_{i,o} < sf_i|$$

Si el nodo de referencia no es la fuente, cada nodo candidato i corresponde al comienzo de un arco k que llega al nodo $npar$, último nodo etiquetado. La intensidad es el número de soluciones élite para las que el material que circula por el arco es menor que la diferencia entre el material que circula por el arco y el flujo correspondiente en la solución que se está construyendo. La diferencia indicada, $ltar_k - flujo_{npar,i}$, expresa el material que circularía por el arco si el procedimiento hubiese finalizado justo antes de comenzar la presente iteración del mismo.

$$int_i = |o \in O / eltar_{k,o} < ltar_k - flujo_{npar,i}|$$

Si el nodo de referencia es un depósito, el sumidero se considera candidato si la capacidad residual del arco ficticio correspondiente es positiva, $capr_{npar,s} > 0$, y no hay otros candidatos. Obviamente, en este caso se deberá etiquetar el sumidero por ser el único candidato.

$$kpon_s = 1$$

Procedimiento 39. - ACORTAR RUTAS 2

Modificación del procedimiento ACORTAR RUTAS en el que se introducen las funciones intensidad. A continuación se describen los cambios respecto del procedimiento original. El procedimiento completo puede encontrarse en el Apéndice A.6.

En primer lugar, para cada vehículo, se eliminan los últimos trayectos y los circuitos donde no haya transporte de material. A continuación, se eliminan, iterativamente, trayectos de vehículos donde estos no sean imprescindibles. Para ello, en cada iteración, se considera una lista, $lcan$, con todos los vehículos que no sean imprescindibles en su último trayecto. La función intensidad correspondiente a cada vehículo j de la lista es el número de soluciones élite para las que el número de vehículos de la clase del vehículo j que circulan por el arco k, último arco de la ruta del vehículo, es menor que en la solución actual.

$$int_j = |o \in O / tamclas_{c,k,o} < tcpar_{c,k}| \quad \text{siendo } c = clas_{tipv_j,ori_j}$$

Para terminar, para cada vehículo, se eliminan todos los circuitos en los que el vehículo no sea imprescindible.

Cada vez que se elimina un trayecto de un vehículo j, además de las actualizaciones que se realizan en ACORTAR RUTAS, también es necesario actualizar el número de vehículos de la clase del vehículo j que circulan por el arco eliminado k.

$$tcpar_{c,k} = tcpar_{c,k} - 1 \quad \text{siendo } c = clas_{tipv_j,ori_j}$$

Procedimiento 40. - SUSTITUIR VEHÍCULOS 2

Modificación del procedimiento SUSTITUIR VEHÍCULOS 2 en el que se deber realizar una actualización adicional.

Como en ACORTAR RUTAS 2, cada vez que se elimina un trayecto de un vehículo j, es necesario actualizar el número de vehículos de la clase del vehículo j que circulan por el arco eliminado k.

$$tcpar_{c,k} = tcpar_{c,k} - 1 \quad \text{siendo } c = clas_{tipv_j, ori_j}$$

Procedimiento 41. - REFINAR SOLUCIÓN 2

Modificación del procedimiento REFINAR SOLUCIÓN en el que se se emplean las versiones modificadas de algunos procedimientos. Se ejecutan los procedimientos modificados RETENER MATERIAL 2, APROVECHAR DEPÓSITOS 2, ACORTAR RUTAS 2 y SUSTITUIR VEHÍCULOS 2, en lugar de sus correspondientes procedimientos originales. Los procedimientos SUSTITUIR VEHÍCULOS 2 e INTERCAMBIAR VEHÍCULOS, este último sin modificar, constituyen la fase de búsqueda local del Algoritmo GRASP. El procedimiento REFINAR SOLUCIÓN 2, completamente detallado, puede encontrarse en el Apéndice A.7.

Procedimiento 42. - MEJORA 2

Modificación del procedimiento MEJORA, descrito en el Capítulo 4, en el que se hace uso de funciones intensidad en los procedimientos RETENER MATERIAL, APROVECHAR DEPÓSITOS Y ACORTAR RUTAS, englobados en el procedimiento REFINAR SOLUCIÓN. Además, en el procedimiento SUSTITUIR VEHÍCULOS se realiza una pequeña modificación. En el PASO 0 no se calcula $pjus$, parámetro cuyo valor ya ha sido calculado. Este procedimiento, completamente detallado, puede encontrarse en el Apéndice A.8.

Procedimiento 43. - OBTENER ÉLITE

Construcción del conjunto élite con el que comenzar la aplicación del algoritmo GRASP. En primer lugar, se forma el conjunto élite inicial a partir de las primeras *nelite* soluciones obtenidas, tras lo que se actualizan los parámetros α_d que regulan la diversidad entre dos soluciones. A continuación, con el fin de mejorar el conjunto élite inicial, se siguen obteniendo soluciones hasta completar el número estipulado. Cada vez que se obtiene una solución se actualiza el conjunto élite. En la construcción de las soluciones en este procedimiento no se tienen en cuenta todavía las funciones *greedy* e intensidad que caracterizan al método GRASP, sino que se obtienen aplicando el Algoritmo Constructivo seguido del Algoritmo de Mejora. El cálculo de $pjus$ se realiza solo la primera vez que se ejecuta el Algoritmo de Mejora.

Variables de salida

α, $csol$, div, $diver$, $divert$, ef, $efob$, $eload$, $elonv$, $eltar$, $eprec$, $esecv$, esf, $etmin$, $fobmin$, $fobt$, mej, $tamclas$

Variables internas

$divm_d$	Diversidad media de tipo d en el conjunto élite
$eact$	Indicador de la solución élite actual

$conv$, f, fob, $guar$, $load$, $lonv$, $ltar$, $o1$, $o2$, $prec$, $secv$, sf, $tmin$

Procedimientos llamados

ACTUALIZAR ÉLITE, CONSTRUCTIVO, GUARDAR SOLUCIÓN, MEJORA, OBTENER DIVERSIDAD

Algoritmo

PASO 0

- Inicializar los pesos para la diversidad

$$\alpha_d = \frac{1}{|D|} \quad \forall d \in D$$

- Inicializar el contador de soluciones obtenidas

$$csol = 1$$

- Inicializar la variable con la solución élite actual

$$eact = 1$$

- Inicializar la suma de funciones objetivo globales

$$fobt = 0$$

- Inicializar la diversidad total global

$$divert = 0$$

- Inicializar el mejor valor obtenido para la función objetivo global

$$fobmin = \infty$$

PASO 1

- Ejecutar el procedimiento CONSTRUCTIVO, omitiendo el PASO 0, para obtener una solución

- Ejecutar el procedimiento MEJORA, para mejorar la solución obtenida
- Ejecutar el procedimiento GUARDAR SOLUCIÓN, para guardar la solución mejorada como solución $guar = eact$ del conjunto élite
- Actualizar la suma de funciones objetivo globales

$$fobt = fobt + fob$$

- Actualizar el contador de soluciones obtenidas

$$csol = csol + 1$$

- Actualizar el indicador de la solución élite actual

$$eact = eact + 1$$

- Si ya se han obtenido las soluciones deseadas, $csol = nelite + 1$, ir al PASO 2
- Volver al PASO 1

PASO 2

- Para cada par de soluciones élite o y o', ejecutar el procedimiento OBTENER DIVERSIDAD, para obtener la diversidad entre las soluciones $o1 = o$ y $o2 = o'$
- Obtener las diversidades medias de cada tipo d

$$divm_d = \frac{1}{\binom{nelite}{2}} \sum_{o,o' \in O/o<o'} div_{d,o,o'}$$

- Actualizar los pesos para la diversidad (en función de las medias de cada tipo de diversidad) de cada tipo d

$$\alpha_d = \begin{cases} 0 & \text{si } divm_d = 0 \\ \dfrac{\dfrac{1}{divm_d}}{\displaystyle\sum_{d' \in D} \dfrac{1}{divm_{d'}}} & \text{si } divm_d > 0 \end{cases}$$

- Actualizar la diversidad global de cada pareja de soluciones: para cada par de soluciones élite o y o'

$$diver_{o,o'} = \sum_{d \in D} \alpha_d div_{d,o,o'}$$

$$diver_{o',o} = diver_{o,o'}$$

- Calcular la diversidad global total

$$divert = \sum_{o,o' \in O/o<o'} diver_{o,o'}$$

PASO 3

- Si ya se han obtenido las soluciones deseadas, $csol > telite$, FIN del procedimiento
- Ejecutar el procedimiento CONSTRUCTIVO, omitiendo el PASO 0, para obtener una solución
- Ejecutar el procedimiento MEJORA, para mejorar la solución obtenida
- Ejecutar el procedimiento GUARDAR SOLUCIÓN, para guardar la solución mejorada como solución $guar = nelite + 1$ del conjunto élite ampliado
- Para cada solución élite o, ejecutar el procedimiento OBTENER DIVERSIDAD, para calcular la diversidad entre la solución $o1 = o$ y la nueva solución $o2 = nelite + 1$
- Actualizar la diversidad global total

$$divert = divert + \sum_{o \in O} diver_{nelite+1,o}$$

- Actualizar la suma de funciones objetivo globales

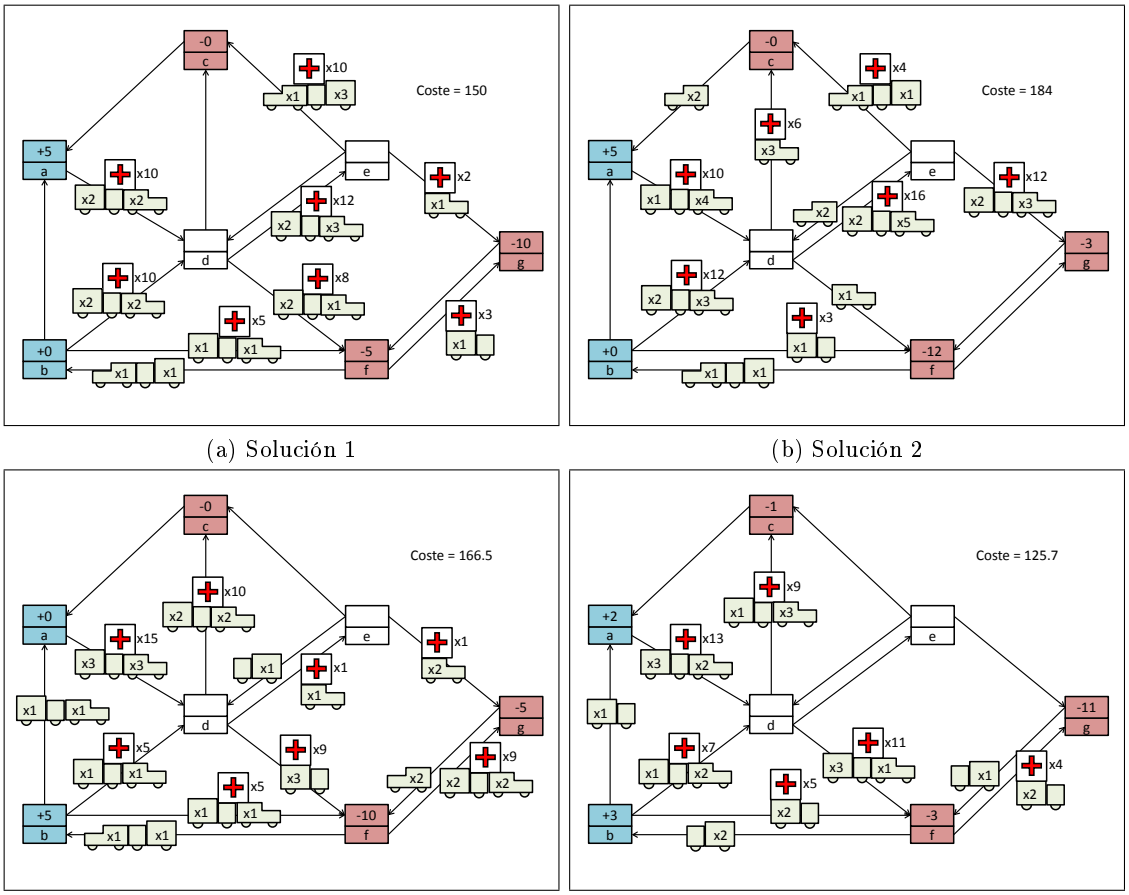
$$fobt = fobt + fob$$

- Ejecutar el procedimiento ACTUALIZAR ÉLITE, para actualizar el conjunto élite teniendo en cuenta la nueva solución
- Actualizar el contador de soluciones obtenidas

$$csol = csol + 1$$

- Volver al PASO 3

La Figura 5.1 muestra las $nelite = 4$ primeras soluciones obtenidas durante una ejecución de este procedimiento para el ejemplo de la Figura 3.1, y forman el conjunto élite inicial. Puede apreciarse que las soluciones son bastante diferentes entre sí, como cabría esperar al obtenerlas de forma no guiada, pero no demasiado buenas en cuanto al coste, como se verá más adelante. La mejor de estas soluciones es la 4 (Figura 5.1d), con un coste de 125.7 u. m. El coste medio del conjunto élite es de 156.55 u. m. La diversidad total es de 0.19. Tras completar este procedimiento, obteniéndose en total $telite = 20$ soluciones, el conjunto élite ha mejorado en cuanto a los valores de la función objetivo, presentando un coste medio de 129.075 u. m., y se ha aumentado ligeramente la diversidad total a 0.21. Las nuevas soluciones pueden verse en la Figura 5.2. Se observa que la solución 4 de este nuevo conjunto élite (Figura 5.2d) ya aparecía en el conjunto élite inicial (Figura 5.1d), que de hecho era la mejor solución en ese momento. Ahora, la mejor solución es la 3 (Figura 5.2c), con un coste de 117.3 u. m.



(a) Solución 1

(b) Solución 2

(c) Solución 3

(d) Solución 4

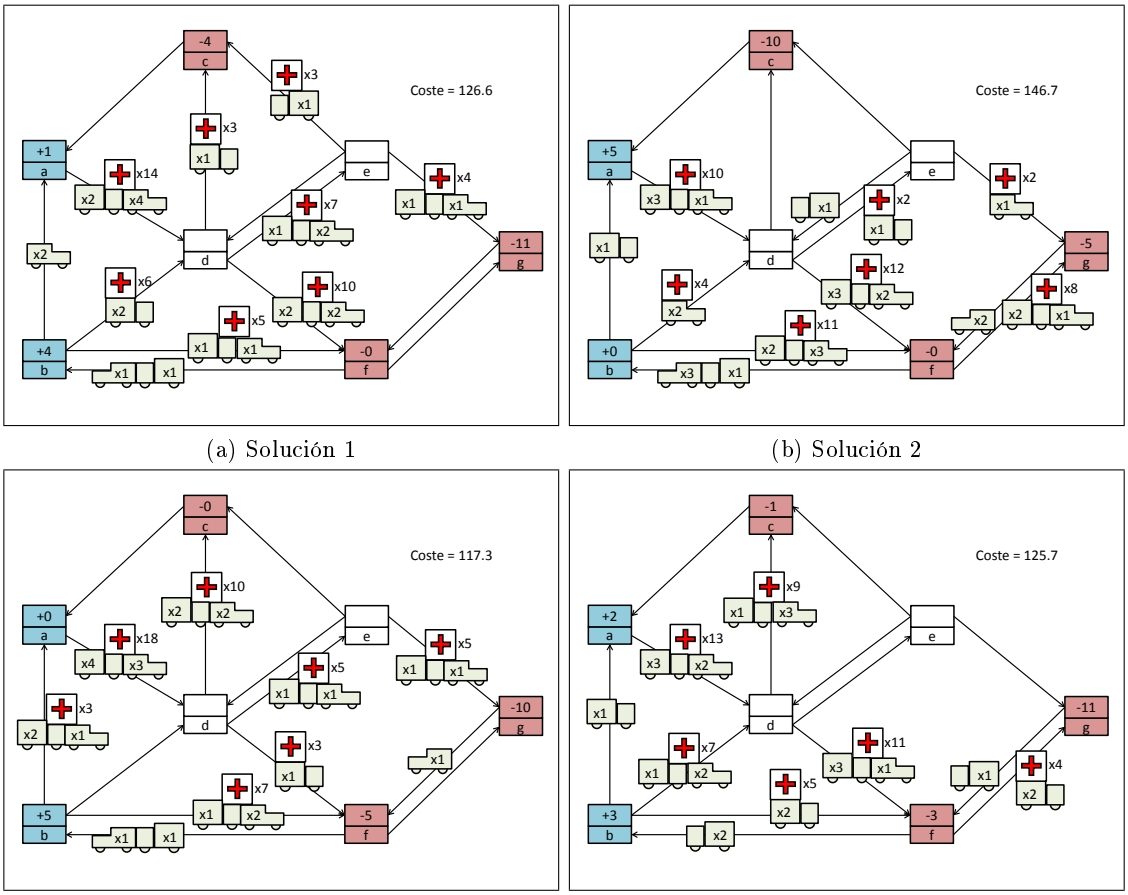
Figura 5.1: Conjunto élite inicial

Procedimiento 44. - GUIAR CONSTRUCCIÓN

Obtención del resto de soluciones mediante la versión modificada del procedimiento CONSTRUCTIVO para tener en cuenta las funciones *greedy* e intensidad. La variable λ controla el peso que tienen la intensidad, el azar, y eventualmente la función *greedy*, y su valor en cada iteración *csol* se calcula mediante la siguiente expresión:

$$\lambda = 1 - \sigma_1 e^{-\sigma_2 \frac{csol - telite}{tsol - telite}}$$

De este modo, el valor de λ va aumentando a medida que crece el número de soluciones obtenidas, con el objeto de que la función intensidad tenga una incidencia creciente. El parámetro σ_1 determina el peso que reciben inicialmente las funciones intensidad (a mayor σ_1 menor peso recibe la intensidad), mientras que el parámetro σ_2 indica la velocidad con la que crece λ. Mediante la función exponencial propuesta, se pretende que al principio λ crezca rápidamente hacia 1, amortiguándose el crecimiento según el número de iteraciones se aproxima al número de soluciones que se desea obtener en total, *tsol*. Cada vez que se obtiene una nueva solución, se actualiza el conjunto élite. Además, siempre se guarda la



(a) Solución 1

(b) Solución 2

(c) Solución 3

(d) Solución 4

Figura 5.2: Conjunto élite tras OBTENER ÉLITE

mejor solución obtenida hasta el momento (la de menor valor en la función objetivo).

Variables de entrada

α, $csol$, $diver$, $divert$, ef, $efob$, $eload$, $elonv$, $eltar$, $eprec$, $esecv$, esf, $etmin$, $fobmin$, $fobt$, mej, $tamclas$

Variables de salida

ef, $efob$, $eload$, $elonv$, $eltar$, $eprec$, $esecv$, esf, $etmin$, mej

Variables internas

λ, $conv$, f, fob, $guar$, $load$, $lonv$, $ltar$, $o1$, $o2$, $prec$, $secv$, sf, $tmin$

Procedimientos llamados

ACTUALIZAR ÉLITE, CONSTRUCTIVO 2, GUARDAR SOLUCIÓN, MEJORA 2, OBTENER DIVERSIDAD

Algoritmo

PASO 0

- Inicializar el peso de las funciones intensidad respecto al azar y las funciones *greedy*

$$\lambda = 1 - \sigma_1 e^{-\sigma_2 \frac{csol - telite}{tsol - telite}}$$

PASO 1

- Si ya se han obtenido las soluciones deseadas, $csol > tsol$, FIN del procedimiento
- Ejecutar el procedimiento CONSTRUCTIVO 2, para obtener una solución guiada
- Ejecutar el procedimiento MEJORA 2, para mejorar de forma guiada la solución obtenida
- Ejecutar el procedimiento GUARDAR SOLUCIÓN, para guardar la solución mejorada como solución $guar = nelite + 1$ del conjunto élite ampliado
- Para cada solución élite o, ejecutar el procedimiento OBTENER DIVERSIDAD, para calcular la diversidad entre la solución $o1 = o$ y la nueva solución $o2 = nelite + 1$
- Actualizar la diversidad global total

$$divert = divert + \sum_{o \in O} diver_{nelite+1,o}$$

- Actualizar la suma de funciones objetivo globales

$$fobt = fobt + fob$$

- Ejecutar el procedimiento ACTUALIZAR ÉLITE, para actualizar el conjunto élite teniendo en cuenta la nueva solución
- Actualizar el contador de soluciones obtenidas

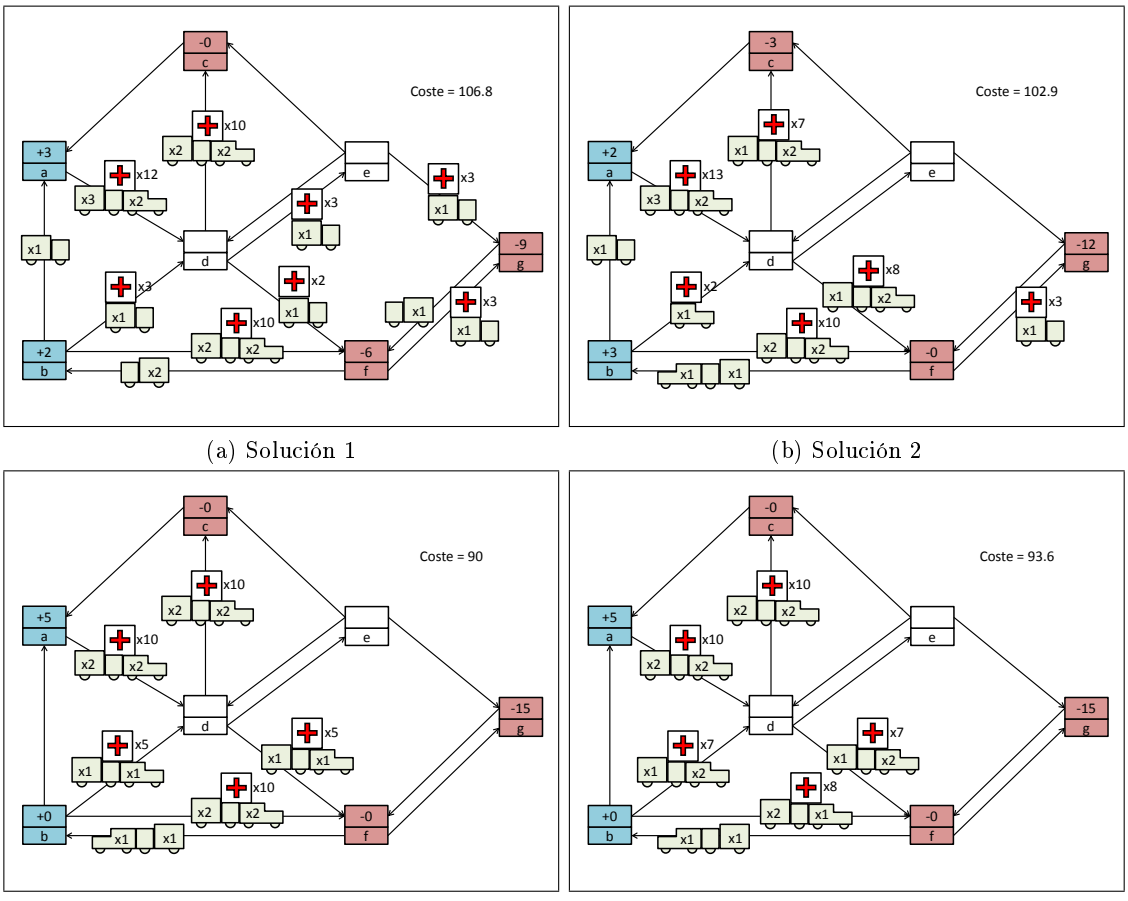
$$csol = csol + 1$$

- Actualizar el peso de las funciones intensidad respecto al azar y las funciones *greedy*

$$\lambda = 1 - \sigma_1 e^{-\sigma_2 \frac{csol - telite}{tsol - telite}}$$

- Volver al PASO 1

Tras aplicar este procedimiento al ejemplo de la Figura 3.1, después de obtener $tsol = 400$ soluciones en total, se obtiene el conjunto élite final formado por las soluciones que se muestran en la Figura 5.3. Las soluciones han mejorado sustancialmente. Al finalizar el procedimiento OBTENER ÉLITE, el coste medio era de 129.075 u. m., mientras que en el conjunto élite final es de 98.325 u. m. La mejor solución es la 3 (Figura 5.3c), con un coste de 90 u. m. Puede apreciarse en esta solución que no se satisface nada de la demanda del nodo g, por resultar más alejado de los depósitos que los otros nodos de demanda y por tanto suponer mayor coste de transporte. La diversidad total, en cambio, ha disminuido de 0.21 a 0.14: para mejorar la función objetivo de las soluciones, estas han debido tener cada vez más elementos parecidos entre sí. Esta disminución de la diversidad puede apreciarse en las figuras: las soluciones presentan bastantes elementos en común, resultando casi idénticas las dos últimas.



(a) Solución 1 (b) Solución 2

(c) Solución 3 (d) Solución 4

Figura 5.3: Conjunto élite final

5.3. Programa principal

En esta última sección del capítulo, se presenta el procedimiento principal del algoritmo.

Procedimiento 45. - GRASP

Procedimientos llamados

ACOTAR GREEDY, GUIAR CONSTRUCCIÓN, OBTENER ÉLITE, PREPROCESO

Algoritmo

- Ejecutar el procedimiento PREPROCESO, para realizar las operaciones previas
- Ejecutar el procedimiento ACOTAR GREEDY, para obtener las cotas que se emplearán en las funciones *greedy*
- Ejecutar el procedimiento OBTENER ÉLITE, para obtener el conjunto élite
- Ejecutar el procedimiento GUIAR CONSTRUCCIÓN, para obtener la muestra completa de soluciones

El Algoritmo GRASP en su conjunto puede sintetizarse en el diagrama de flujo que se muestra en la Figura 5.4.

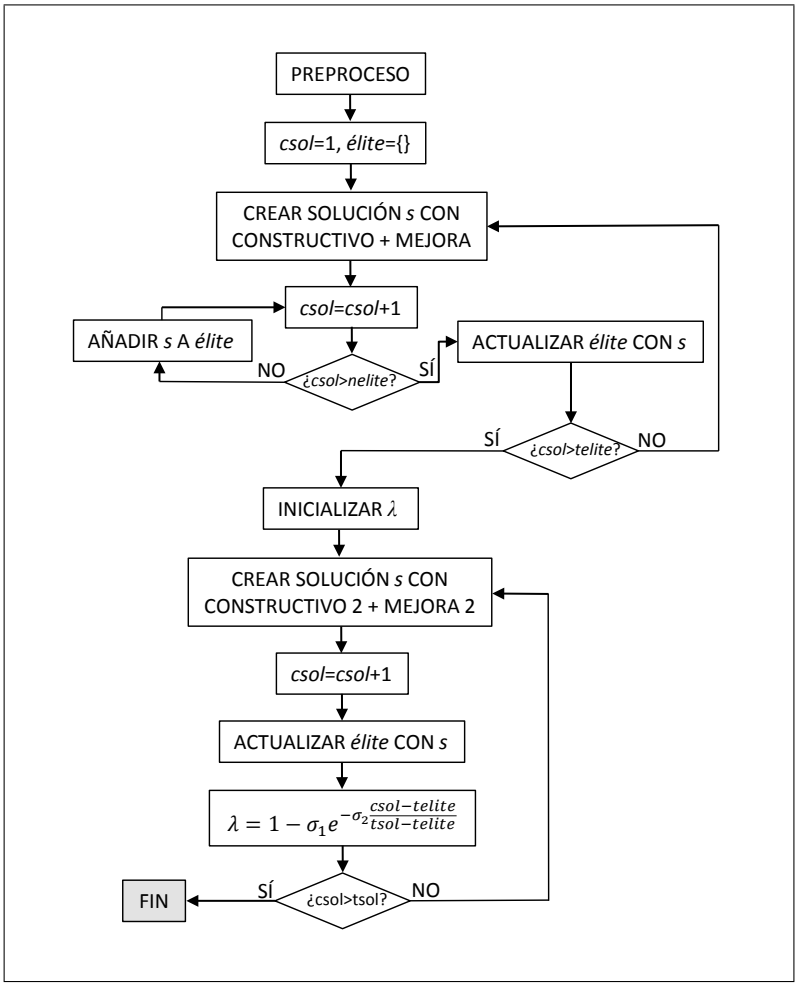


Figura 5.4: Diagrama de flujo de GRASP

Capítulo 6

Algoritmo Colonia de Hormigas

Como complemento del capítulo anterior, dedicado a la metaheurística GRASP aplicada al problema que se está tratando, en este capítulo se presenta un algoritmo basado en la metaheurística colonia de hormigas. En esta metaheurística, como en el caso del GRASP, la construcción de soluciones desempeña un papel muy importante, característica deseable para explotar los algoritmos Constructivo y de Mejora previamente descritos en los capítulos 3 y 4.

Se conoce con el nombre de colonia de hormigas (ACO) a un conjunto de algoritmos metaheurísticos que tratan de emular el modo en que las hormigas de una colonia consiguen encontrar el camino más corto desde el hormiguero hasta una fuente de comida. Según buscan la fuente de alimento, las hormigas van dejando un rastro de feromonas, y van construyendo su camino moviéndose al azar cuando no encuentran feromonas y priorizando el paso por las zonas con más feromonas cuando estas existen. Como un camino más corto se recorre más rápido que un camino más largo y, dado que después de llegar a su destino las hormigas han de regresar al hormiguero, más hormigas tendrán tiempo de realizar el recorrido completo por el camino más corto que por el más largo, incrementándose así cada vez más la diferencia entre la cantidad de feromonas depositadas en cada camino, y por lo tanto aumentando a su vez el número de hormigas que eligen el camino más corto.

El primer algoritmo basado en colonias de hormigas recibió el nombre de *Ant System* (Dorigo *et al.* (1996)). En el algoritmo *Ant Colony System*, o ACS (Dorigo y Gambardella (1997)), se introdujeron algunas novedades, destacando el uso de actualizaciones locales de feromonas, en las que, para diversificar el rango de soluciones obtenidas, se disminuye el rastro de feromonas por donde pasan las hormigas. En el Capítulo 1 puede encontrarse una selección de publicaciones donde se aplica la metaheurística colonia de hormigas a problemas de rutas y de logística humanitaria.

El Algoritmo Colonia de Hormigas que se presenta en este capítulo, y que en lo sucesivo se abreviará como Algoritmo ACO, es un procedimiento iterativo en el que, en cada iteración, se construye una solución completa para el problema. El funcionamiento del algoritmo es similar al del algoritmo GRASP, si bien, en vez de funciones intensidad, se emplean rastros de feromonas para guiar la construcción de las soluciones. En la construcción de cada

solución se utilizan varios tipos de hormigas artificiales -*Multiple Ant Colony*- que actúan de forma coordinada en función de lo que parece mejor a priori, a través de funciones *greedy*, y del rastro de feromonas que se encuentran. Una vez completada la solución, se actualizan las feromonas de cara a la siguiente iteración. Tal y como está establecido en el método ACS, se realizan dos tipos de actualizaciones. En la actualización global, que se efectúa cada m iteraciones, se aumenta la cantidad de feromonas de los elementos que forman parte de la mejor solución obtenida desde el comienzo del algoritmo, con la finalidad de priorizar esos elementos en la construcción de futuras soluciones. La actualización local se realiza después de cada iteración; en este caso, se disminuye la cantidad de feromonas de los elementos que aparecen en la última solución obtenida. El propósito de esta actualización local es diversificar la búsqueda, disminuyendo de esta forma la posibilidad de estancarse en mínimos locales indeseados.

Como en el Algoritmo GRASP, la construcción de las soluciones se efectúa mediante los algoritmos Constructivo y de Mejora, siendo necesario en varios de los procedimientos que conforman ambos algoritmos ciertas modificaciones para introducir el uso de hormigas artificiales, en algunos casos de distintos tipos. En la fase de diseño de los itinerarios de los vehículos (CONSTRUIR RUTAS 3), cada vehículo es representado mediante una hormiga, considerándose del mismo tipo las hormigas que representan vehículos iguales y con el mismo nodo de origen, mientras que en la fase de envío de la ayuda (ENVIAR FLUJO), las hormigas representan los lotes de ayuda que se han de repartir. En cada uno de estos dos procedimientos, el rastro de feromonas se va reajustando a medida que las hormigas van construyendo la solución, disminuyendo paulatinamente las feromonas asociadas a los elementos que van siendo incorporandos ya que, de otro modo, se favorecería en exceso los elementos con mayor feromona a priori, en perjuicio de otros con menor feromona pero potencialmente deseables. La misma técnica se aplica en los procedimientos RETENER MATERIAL 3, APROVECHAR DEPÓSITOS 3 y ACORTAR RUTAS 3 para desechar elementos superfluos de las soluciones.

Se distinguen cuatro tipos de feromonas, cada uno de ellos con una misión específica en la construcción y mejora de las soluciones. Las feromonas de primer tipo son depositadas por los vehículos al desplazarse por los arcos, mientras que las feromonas de segundo tipo son depositadas por los lotes de material también en los arcos por los que circulan. Las feromonas de tercer tipo están asociadas a las cantidades de material que salen de los depósitos y, por último, las feromonas de cuarto tipo corresponden a las cantidades de material que quedan en los nodos al final del reparto. Nuevamente, la influencia de las feromonas irá aumentando con el transcurso de las iteraciones a la par que disminuye la influencia de las funciones *greedy*. Para facilitar este cambio progresivo en la influencia de los dos términos -feromonas y funciones *greedy*-, se ha modificado la función que combina ambos términos usada habitualmente en la metodología ACO, cuya expresión utiliza potencias, sustituyéndose estas por productos.

Al ejecutar el algoritmo, cada uno de los procedimientos donde se usan hormigas artificiales recibe los valores de las feromonas que hayan sido establecidos en la última actualización efectuada, ya sea esta local o global. En el transcurso del procedimiento, según

los elementos vayan siendo introducidos en la solución, los valores de las feromonas asociadas a estos elementos deben ser disminuidos para impedir un exceso de concentración de dichos elementos en la solución. Por esta razón, en este trabajo se ha introducido el término "feromonas efectivas", que son las encargadas de guiar realmente la construcción de las soluciones, y las que ven variar su valor durante la ejecución de los procedimientos donde se usan hormigas, para distinguirlas de las feromonas propiamente dichas, de las que solo se modifican sus valores en las actualizaciones locales y/o globales que se realizan una vez completada cada solución.

6.1. Parámetros y variables

En esta sección se presentan los elementos que intervienen en el algoritmo ACO, definiéndose los que aparecen por primera vez en esta memoria.

Parámetros de la metaheurística

ρ_l	Factor de evaporación de las feromonas en las actualizaciones locales
ρ_g	Factor de evaporación de las feromonas en las actualizaciones globales
σ_1	Complemento a 1 del peso de las feromonas al comienzo del algoritmo
σ_2	Velocidad de crecimiento del peso de las feromonas
$\hat{\tau}^{ru}_{c,k}$	Feromona inicial de primer tipo (rutas) en el arco k para los vehículos de clase c
$\hat{\tau}^{fl}_{k}$	Feromona inicial de segundo tipo (flujos) en el arco k
$\hat{\tau}^{dep}_{i}$	Feromona inicial de tercer tipo (depósitos) en el depósito $i \in IO$
$\hat{\tau}^{st}_{i}$	Feromona inicial de cuarto tipo (*stocks*) en el nodo $i \in ID \cup IO$
m	Número de soluciones obtenidas entre cada par de actualizaciones globales consecutivas.
q_0	Probabilidad de tomar el elemento de mayor ponderación en cualquiera de las situaciones de decisión
$tsol$	Número total de soluciones obtenidas en la aplicación del algoritmo ACO

Otros parámetros

$cclas$, cfl, cru, $pesotot$

Variables de decisión

$mload_j$	Secuencia de cantidades transportadas por el vehículo j hasta que termina su reparto en la mejor solución obtenida
$msecv_j$	Secuencia de arcos (ruta) que recorre el vehículo j hasta que termina su reparto en la mejor solución obtenida
$mtmin_k$	Instante mínimo en el que el convoy que recorre el arco k finaliza su trayecto en la mejor solución obtenida

Variables objetivo

mf_g	Valor del objetivo g en la mejor solución obtenida
$fobmin$	Mejor valor obtenido para la función objetivo global

Variables auxiliares

λ	Peso de las feromonas para la ponderación de un candidato a movimiento en cualquier fase de la construcción de una solución
$\tau_{c,k}^{ru}$	Feromona de primer tipo en el arco k para los vehículos de clase c
τ_k^{fl}	Feromonas de segundo tipo en el arco k
τ_i^{dep}	Feromona de tercer tipo en el depósito $i \in IO$
τ_i^{st}	Feromona de cuarto tipo en el nodo $i \in ID \cup IO$
$cglobal$	Contador global de soluciones
$clocal$	Contador local de soluciones
$mlonv_j$	Número de arcos recorridos por el vehículo j en la mejor solución obtenida
$mltar_k$	Cantidad total de material que transporta el convoy correspondiente al arco k en la mejor solución obtenida
$msf_{i,o}$	Cantidad de material que queda en el nodo i al final del reparto en la mejor solución obtenida
$mtcpar_{c,k}$	Número de vehículos de la clase c que recorren el arco k en la mejor solución obtenida

$conv$, f, fob, $fobmin$, $load$, $lonv$, $ltar$, $secv$, sf, $tcpar$, $tmin$

6.2. Procedimientos

Procedimiento 46. - MEJOR SOLUCIÓN

Almacenamiento de los elementos importantes de la mejor solución obtenida.

Variables de entrada

$conv$, f, fob, $fobmin$, $load$, $lonv$, $ltar$, $secv$, sf, $tcpar$, $tmin$

Variables de salida

$fobmin$, mf, $mload$, $mlonv$, $mltar$, $msecv$, msf, $mtcpar$, $mtmin$

Algoritmo

- Guardar los valores de las funciones objetivo

$$mf_g = f_g \quad \forall g$$
$$fobmin = fob$$

- Guardar las características de la solución

$$\begin{gathered} mlonv_j = lonv_j \quad \forall j \\ mltar_k = ltar_k \quad \forall k \\ msecv_{j,e} = secv_{j,e} \quad \forall j,\ \forall e \ / \ 1 \leq e \leq lonv_j \\ mload_{j,e} = load_{j,e} \quad \forall j,\ \forall e \ / \ 1 \leq e \leq lonv_j \\ msf_i = sf_i \quad \forall i \\ mtmin_k = tmin_k \quad \forall k \\ mtcpar_{c,k} = tcpar_{c,k} \quad \forall c,\ \forall k \end{gathered}$$

Procedimiento 47. - ACTUALIZAR LOCALMENTE

Actualización local de las feromonas. Este tipo de actualización se realiza cada vez que se obtiene una solución. El propósito es disminuir las feromonas de los elementos que aparecen en la última solución obtenida con idea de diversificar la construcción.

Variables de entrada

τ, $ltar$, sf, $tcpar$

Variables de salida

τ

Algoritmo

- Actualizar las feromonas de primer tipo, favoreciendo a los pares clase-arco con menos presencia en la última solución obtenida

$$\tau^{ru}_{c,k} = (1-\rho_l)\tau^{ru}_{c,k} + \frac{\rho_l}{1+tcpar_{c,k}} \quad \forall c,\ \forall k$$

- Actualizar las feromonas de segundo tipo, favoreciendo a los arcos con menos transporte de material en la última solución obtenida

$$\tau^{fl}_{k} = (1-\rho_l)\tau^{fl}_{k} + \frac{\rho_l}{1+ltar_{k}} \quad \forall k$$

- Actualizar las feromonas de tercer tipo, favoreciendo a los depósitos de los que menos material ha salido en la última solución obtenida

$$\tau^{dep}_{i} = (1-\rho_l)\tau^{dep}_{i} + \frac{\rho_l}{1+qav_i-sf_i} \quad \forall i \in IO$$

- Actualizar las feromonas de cuarto tipo, favoreciendo a los depósitos y nodos de demanda donde menos material queda al final del reparto en la última solución obtenida

$$\tau^{st}_{i} = (1-\rho_l)\tau^{st}_{i} + \frac{\rho_l}{1+sf_i} \quad \forall i \in ID \cup IO$$

Procedimiento 48. - ACTUALIZAR GLOBALMENTE

Actualización global de las feromonas al completar cada ciclo de m soluciones. La intención es favorecer a los elementos que más aparecen en la mejor solución obtenida desde el comienzo del algoritmo.

Variables de entrada

τ, $mltar$, msf, $mtcpar$

Variables de salida

τ

Algoritmo

- Actualizar las feromonas de primer tipo, favoreciendo a los pares clase-arco con más presencia en la mejor solución obtenida

$$\tau_{c,k}^{ru} = (1-\rho_g)\tau_{c,k}^{ru} + \frac{mtcpar_{c,k}}{cclas_c} \quad \forall c,\ \forall k$$

- Actualizar las feromonas de segundo tipo, favoreciendo a los arcos con más transporte de material en la mejor solución obtenida

$$\tau_k^{fl} = (1-\rho_g)\tau_k^{fl} + \frac{mltar_k}{qglobal} \quad \forall k$$

- Actualizar las feromonas de tercer tipo, favoreciendo a los depósitos de los que más material ha salido en la mejor solución obtenida

$$\tau_i^{dep} = (1-\rho_g)\tau_i^{dep} + \frac{qav_i - msf_i}{qav_i} \quad \forall i \in IO$$

- Actualizar las feromonas de cuarto tipo, favoreciendo a los depósitos y nodos de demanda donde más material queda al final del reparto en la mejor solución obtenida

$$\tau_i^{st} = (1-\rho_g)\tau_i^{st} + \frac{msf_i}{qav_i} \quad \forall i \in IO$$

$$\tau_i^{st} = (1-\rho_g)\tau_i^{st} + \frac{msf_i}{dem_i} \quad \forall i \in ID$$

Procedimiento 49. - CONSTRUIR RUTAS 3

Modificación del procedimiento CONSTRUIR RUTAS en el que se introducen las funciones *greedy* y las feromonas para guiar la construcción de los itinerarios. Los cambios respecto del procedimiento CONSTRUIR RUTAS 2 consisten en el uso de feromonas en lugar de las funciones intensidad presentes en el Algoritmo GRASP.

Se define la siguiente variable interna:

fer_{mov} Feromona efectiva del movimiento mov

En cada iteración, para cada movimiento mov candidato a añadirse a la solución en construcción, que corresponde a un vehículo j y un arco k, se calculan la feromona efectiva y la función *greedy*. Las funciones *greedy* tienen las mismas expresiones que en los procedimientos respectivos del Algoritmo GRASP. En este caso, dichas expresiones son las mismas que en CONSTRUIR RUTAS 2. La feromona efectiva, que es la que realmente se utiliza para guiar la construcción, se obtiene a partir de la feromona de primer tipo. La feromona de primer tipo favorece, gracias a la actualización global, a los pares clase-arco con más presencia en la mejor solución obtenida, pero también, gracias a la actualización local, a los que tienen menos presencia en todas las soluciones obtenidas, especialmente las obtenidas más recientemente. Conforme el número de vehículos de cada clase que recorre cada arco vaya aumentando, la feromona efectiva asociada al par clase-arco correspondiente irá disminuyendo. El propósito de esta disminución es que no se favorezca en exceso a los pares con mayor feromona, posibilitando, de esta forma, que otros elementos presentes en la mejor solución, pero con menor feromona, también aparezcan en la solución en construcción. Esta idea se repetirá en todos los procedimientos donde se emplean feromonas.

$$fer_{mov} = \left(1 - \frac{tcpar_{c,k}}{cclas_c}\right) \tau^{ru}_{c,k} \quad \text{siendo } c = clas_{tipv_j, ori_j}$$

La ponderación de cada movimiento mov se calcula mediante la siguiente fórmula, donde la variable λ representa el peso que recibe la feromona respecto de la función *greedy*.

$$kpon_{mov} = \lambda \cdot fer_{mov} + (1 - \lambda) greedy_{mov}$$

Una vez establecidas las ponderaciones correspondientes a todos los movimientos candidatos, con probabilidad q_0 se elige directamente el movimiento con mayor ponderación, y con probabilidad $1 - q_0$ se efectúa un sorteo entre los movimientos de acuerdo a sus probabilidades, que se calculan de forma que sean proporcionales a las ponderaciones. Si el índice mov' recorre todos los movimientos posibles, la probabilidad de elegir el movimiento mov sería:

$$prob_{mov} = \frac{kpon_{mov}}{\sum_{mov'} kpon_{mov'}}$$

A diferencia de la función intensidad, la feromona efectiva nunca llega a alcanzar el valor 0, por lo que, en este caso, no es necesario incorporar un término que represente al azar en la ponderación, ya que esta siempre tomará valores estrictamente positivos. En este procedimiento, por ejemplo, la feromona efectiva valdría 0 si $tcpar_{c,k} = cclas_c$, es decir, si todos los vehículos de la clase c recorren el arco pero, en este caso, un vehículo j de esa clase no podría recorrer el arco k porque ya lo habría recorrido con anterioridad, por lo que el par (j, k) no podría ser ya candidato a movimiento.

En los procedimientos donde no se utilicen funciones *greedy* podrá omitirse el término λ, por lo que la ponderación de cada movimiento se expresará de una de las dos siguientes formas, dependiendo de la situación donde se presente la decisión.

$$kpon_{mov} = fer_{mov} \quad \text{ó} \quad kpon_{mov} = \frac{1}{fer_{mov}}$$

En las decisiones constructivas, presentes en su mayoría en el procedimiento CONSTRUCTIVO 3, se decide qué elementos incorporar a la solución, por lo que se emplea la primera expresión, ya que interesa incorporar elementos con mayor nivel de feromona. En las decisiones destructivas, asociadas al procedimiento MEJORA 3, se decide qué elementos eliminar de la solución y, por tanto, se empleará habitualmente la segunda expresión, ya que en este caso se pretende eliminar elementos con menor nivel de feromona.

El criterio de parada está basado en la feromona efectiva. Cuando el máximo entre las feromonas efectivas de los movimientos candidatos no alcance un determinado umbral, la construcción terminará. Si con las rutas creadas no pudiese repartirse el material deseado, se disminuirá el valor del umbral a su mitad, para intentar prolongar las rutas.

Procedimiento 50. - ENVIAR FLUJO 3

Modificación del procedimiento ENVIAR FLUJO en el que se introducen las funciones *greedy* y las feromonas para guiar el envío de material. A continuación se detallan los cambios respecto del procedimiento ENVIAR FLUJO 2.

La primera situación en la que tomar una decisión es la elección del depósito a etiquetar desde la fuente, situación que se presenta cuando el nodo de referencia es la fuente, $npar = f$. En este caso, la feromona efectiva de cada depósito i se calcula a partir de la feromona de tercer tipo, que favorece a los depósitos de los que más material sale en la mejor solución obtenida y a los depósitos de los que menos material sale en las últimas soluciones obtenidas.

$$fer_i = \left(1 - \frac{flujo_{f,i}}{qav_i}\right) \tau_i^{dep}$$

Para elegir el nodo a etiquetar desde cualquier nodo distinto de la fuente, cada nodo candidato i corresponde al final de un arco k que parte del nodo $npar$, último nodo etiquetado. La feromona efectiva del nodo candidato i se calcula a partir de la feromona de segundo tipo, que favorece a los arcos por los que más material circula en la mejor solución obtenida y a los arcos por lo que menos material circula en las últimas soluciones obtenidas. En este caso se deben calcular también funciones *greedy* según las fórmulas que aparecen en ENVIAR FLUJO 2.

$$fer_i = \left(1 - \frac{flujo_{npar,i}}{qglobal}\right) \tau_k^{fl}$$

Si el nodo de referencia es un nodo de demanda, $dem_{npar} > 0$, además del resto de nodos candidatos a etiquetar, el sumidero se considera también como candidato. La feromona efectiva del sumidero, s, se calcula a partir de la feromona de cuarto tipo, que favorece a los nodos donde más material queda al final del reparto en la mejor solución obtenida y a los nodos donde menos material queda en las últimas soluciones obtenidas. Las funciones *greedy* se calculan según se estableció en ENVIAR FLUJO 2.

$$fer_s = \left(1 - \frac{flujo_{npar,s}}{dem_{npar}}\right) \tau_{npar}^{st}$$

Procedimiento 51. - CONSTRUCTIVO 3

Modificación del procedimiento CONSTRUCTIVO, en el que se introducen las funciones *greedy* y las feromonas en algunos de los procedimientos empleados en la construcción de la solución, concretamente, en los procedimientos CONSTRUIR RUTAS y ENVIAR FLUJO. Respecto al procedimiento CONSTRUCTIVO 2, la única diferencia es que se ejecutan los procedimientos modificados CONSTRUIR RUTAS 3 y ENVIAR FLUJO 3 en lugar de los procedimientos CONSTRUIR RUTAS 2 y ENVIAR FLUJO 2.

Procedimiento 52. - RETENER MATERIAL 3

Modificación del procedimiento RETENER MATERIAL en el que se introducen las feromonas. Como ocurría en el Algoritmo GRASP, en los procedimientos propios del Algoritmo de Mejora no se usan funciones *greedy*. A continuación, se describe la forma de introducir las feromonas y su uso para la elección de nodos candidatos a etiquetar.

Si el nodo de referencia es la fuente, $npar = f$, se consideran como candidatos los nodos de demanda necesitados, esto es, los que reciben una cantidad de material inferior a la que les correspondería en un reparto perfectamente equitativo. La feromona efectiva de cada uno de ellos, i, se calcula a partir de la feromona de cuarto tipo, que favorece a los nodos donde más material queda al final del reparto en la mejor solución obtenida y a los nodos donde menos material queda en las últimas soluciones obtenidas.

$$fer_i = \left(1 - \frac{sf_i}{dem_i}\right) \tau_i^{st}$$

Obsérvese que en la expresión anterior la feromona efectiva no puede ser 0. Esto ocurriría si $sf_i = dem_i$, es decir, si al nodo llegase todo el material demandado, pero, en ese caso, el nodo no sería necesitado y no podría ser candidato a etiquetarse.

Si el nodo de referencia no es la fuente, cada nodo candidato i corresponde al final de un arco k que parte del nodo $npar$, último nodo etiquetado. La feromona efectiva de cada nodo candidato se calcula a partir de la feromona de segundo tipo, que favorece a los arcos por los que más material se transporta en la mejor solución obtenida y a los arcos por lo que menos material se transporta en las últimas soluciones obtenidas.

$$fer_i = \left(1 - \frac{ltar_k - flujo_{npar,i}}{1 + qglobal}\right) \tau_k^{fl}$$

El numerador de la fracción indica la cantidad de material que se transporta por el arco, habiéndose descontado lo que de momento se ha decidido eliminar hasta la última iteración realizada del procedimiento, $flujo_{npar,i}$. Como en este caso se desea eliminar flujo de material, la ponderación se toma inversamente proporcional a la feromona efectiva. La unidad que aparece sumada en el denominador de la fracción evita que la feromona efectiva pueda ser 0, lo que podría ocurrir en el caso de que por un arco se transporte todo el material que se desee repartir, $ltar_k = qglobal$, si no se añadiese esa unidad.

$$kpon_{mov} = \frac{1}{fer_k}$$

La misma idea, sumar 1 al denominador, se aplicará en aquellas situaciones donde la ponderación se tome inversamente proporcional a la feromona efectiva.

Si el nodo de referencia es un nodo de demanda no necesitado, es decir, un nodo de demanda que recibe más material del que le correspondería en un reparto perfectamente equitativo, hay que considerar la posibilidad de llegar al sumidero. La feromona efectiva, en este caso, también se calcula a partir de la feromona de cuarto tipo. Pero, nuevamente, como se desea evitar que parte del material llegue a los nodos de demanda no necesitados, la ponderación se ha de tomar inversamente proporcional a la feromona efectiva.

$$fer_s = \left(1 - \frac{sf_{npar}}{1 + dem_{npar}}\right) \tau^{st}_{npar}$$

Procedimiento 53. - APROVECHAR DEPÓSITOS 3

Modificación del procedimiento APROVECHAR DEPÓSITOS en el que se introducen las feromonas. A continuación, se describe la forma de introducir las feromonas.

Si el nodo de referencia es la fuente, $npar = f$, se consideran como candidatos los depósitos. La feromona efectiva de cada uno de ellos, i, se calcula a partir de la feromona de cuarto tipo, que favorece a los nodos donde más material queda al final del reparto en la mejor solución obtenida y a los nodos donde menos material queda en las últimas soluciones obtenidas. Se desea que parte del material que no se ha utilizado en el reparto salga del nodo y, por lo tanto, quede menos material al final. Entonces, la ponderación se ha de tomar inversamente proporcional a la feromona efectiva.

$$fer_i = \left(1 - \frac{sf_i}{1 + qav_i}\right) \tau^{st}_i$$

Si el nodo de referencia no es la fuente, cada nodo candidato i corresponde al comienzo de un arco k que llega al nodo $npar$, último nodo etiquetado. La feromona efectiva de cada nodo candidato se calcula a partir de la feromona de segundo tipo, que favorece a los arcos por los que más material se transporta en la mejor solución obtenida y a los arcos por lo que menos material se transporta en las últimas soluciones obtenidas. Como, en este caso, se desea eliminar flujo de material, la ponderación se toma inversamente proporcional a la feromona efectiva.

$$fer_i = \left(1 - \frac{ltar_k - flujo_{npar,i}}{1 + qglobal}\right) \tau^{fl}_k$$

Si el nodo de referencia es un depósito, el sumidero se considera candidato si la capacidad residual del arco ficticio correspondiente es positiva, $capr_{npar,s} > 0$, y no hay otros candidatos. En este caso, el sumidero se etiquetaría directamente, finalizando el camino de aumento.

Procedimiento 54. - ACORTAR RUTAS 3

Modificación del procedimiento ACORTAR RUTAS en el que se introducen las feromonas.

El único cambio respecto del procedimiento ACORTAR RUTAS 2 consiste en sustituir la función intensidad por el uso de feromonas en la selección de los vehículos. La feromona efectiva de cada vehículo j, siendo k el último arco que recorre, se obtiene a partir de la feromona de primer tipo, que favorece a los pares clase-arco con más presencia en la mejor solución obtenida. Como interesa eliminar trayectos, la ponderación se toma inversamente proporcional a la feromona efectiva.

$$fer_j = \left(1 - \frac{tcpar_{c,k}}{1 + cclas_c}\right) \tau^{ru}_{c,k} \quad \text{siendo } c = clas_{tipv_j, ori_j}$$

Procedimiento 55. - REFINAR SOLUCIÓN 3

Modificación del procedimiento REFINAR SOLUCIÓN en el que se se emplean las versiones modificadas de algunos procedimientos para introducir las feromonas.

Se ejecutan los procedimientos modificados RETENER MATERIAL 3, APROVECHAR DEPÓSITOS 3, ACORTAR RUTAS 3, en los que se hace uso de feromonas, y SUSTITUIR VEHÍCULOS 2, tal como se presentó en el Algoritmo GRASP, en lugar de sus correspondientes procedimientos originales. El procedimiento INTERCAMBIAR VEHÍCULOS se aplica sin modificar, según se estableció en el Algoritmo de Mejora.

Procedimiento 56. - MEJORA 3

Modificación del procedimiento MEJORA, descrito en el Capítulo 4, en el que se hace uso de feromonas en los procedimientos RETENER MATERIAL, APROVECHAR DEPÓSITOS Y ACORTAR RUTAS, englobados en el procedimiento REFINAR SOLUCIÓN, y se ejecuta la versión del procedimiento SUSTITUIR VEHÍCULOS utilizada en el Algoritmo GRASP.

6.3. Programa principal

En esta última sección del capítulo, se presenta el procedimiento principal del algoritmo.

Procedimiento 57. - ACO

Procedimientos llamados

ACOTAR GREEDY, ACTUALIZAR GLOBALMENTE, ACTUALIZAR LOCALMENTE, CONSTRUCTIVO 3, MEJOR SOLUCIÓN, MEJORA 3, PREPROCESO

Algoritmo

PASO 0

- Ejecutar el procedimiento PREPROCESO, para realizar las operaciones previas
- Ejecutar el procedimiento ACOTAR GREEDY, para obtener las cotas que se emplearán en las funciones *greedy*
- Inicializar las feromonas

$$\tau_{c,k}^{ru} = \hat{\tau}_{c,k}^{ru} \quad \forall c,\ \forall k$$

$$\tau_{k}^{fl} = \hat{\tau}_{k}^{fl} \quad \forall k$$

$$\tau_{i}^{dep} = \hat{\tau}_{i}^{dep} \quad \forall i \in IO$$

$$\tau_{i}^{ru} = \hat{\tau}_{i}^{st} \quad \forall i \in ID \cup IO$$

- Inicializar el peso de las feromonas respecto de las funciones *greedy*

$$\lambda = 1 - \sigma_1 e^{-\sigma_2 \frac{cglobal}{tsol}}$$

- Inicializar los contadores de soluciones

$$cglobal = 1$$
$$cglocal = 1$$

- Inicializar el mejor valor obtenido para la función objetivo global

$$fobmin = \infty$$

PASO 1

- Si ya se han obtenido las soluciones deseadas, $cglobal > tsol$, FIN del procedimiento
- Ejecutar el procedimiento CONSTRUCTIVO 3, para obtener una solución guiada
- Ejecutar el procedimiento MEJORA 3, para mejorar de forma guiada la solución obtenida
- Si la solución actual es la mejor que se ha obtenido hasta el momento, $fob < fobmin$, ejecutar el procedimiento MEJOR SOLUCIÓN, para guardar la solución actual como la mejor solución obtenida
- Ejecutar el procedimiento ACTUALIZAR LOCALMENTE, para realizar la actualización local de las feromonas
- Si se ha completado el ciclo, $clocal = m$:
  - Ejecutar el procedimiento ACTUALIZAR GLOBALMENTE, para realizar la actualización global de las feromonas
  - Reiniciar el contador local

$$clocal = 1$$

- Si no se ha completado el ciclo, $clocal < m$, actualizar el contador local

$$clocal = clocal + 1$$

- Actualizar el contador global

$$cglobal = cglobal + 1$$

- Actualizar el peso de las feromonas respecto de las funciones *greedy*

$$\lambda = 1 - \sigma_1 e^{-\sigma_2 \frac{cglobal}{tsol}}$$

- Volver al PASO 1

La Figura 6.1 muestra el diagrama de flujo del Algoritmo ACO.

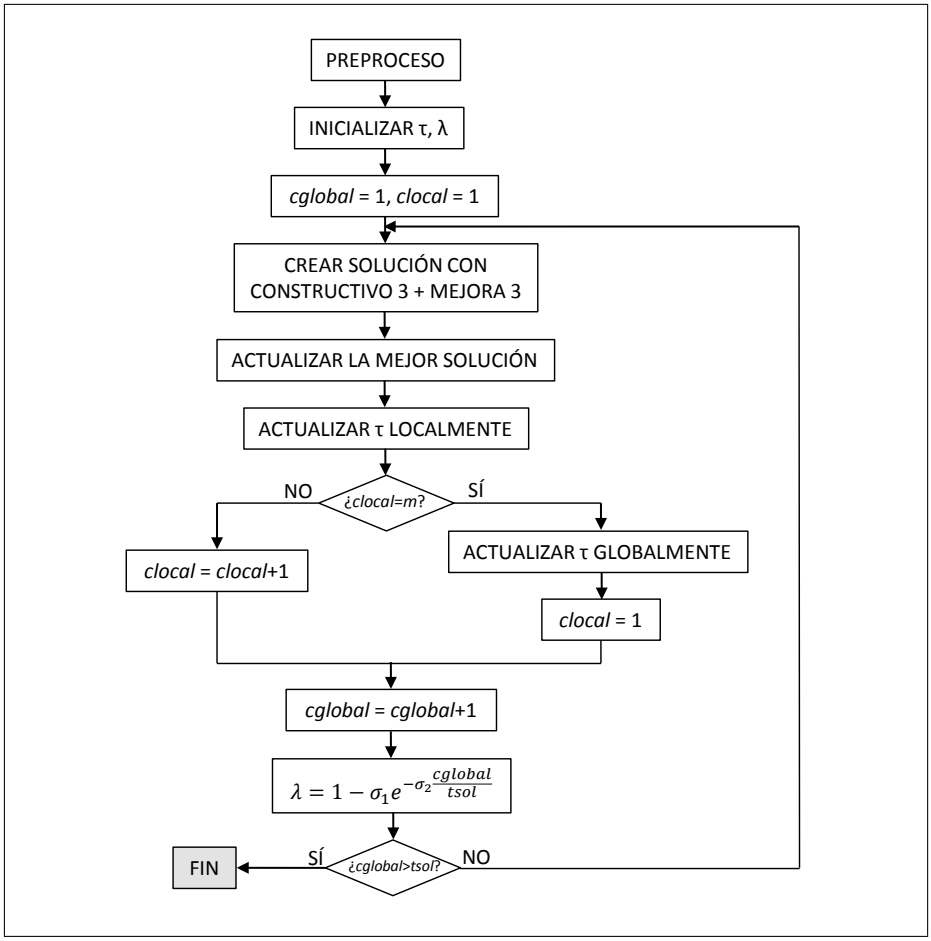


Figura 6.1: Diagrama de flujo de ACO

Capítulo 7

Experiencia computacional

En este capítulo se presentan y analizan los resultados obtenidos al aplicar los algoritmos desarrollados en este trabajo al problema que se está estudiando. Todos los algoritmos han sido implementados en Fortran 95 y ejecutados en un procesador Intel Xeon E5645 hexacore a una frecuencia de reloj de 2.4 Ghz y 4Gb RAM, operando bajo Linux 5.7.

El capítulo se organiza de la siguiente forma. En la Sección 7.1 se analiza el impacto del Algoritmo de Mejora cuando se aplica sobre las soluciones obtenidas con el Algoritmo Constructivo. En la Sección 7.2 se detalla cómo se han calibrado los distintos parámetros que intervienen en los algoritmos GRASP y ACO. En la Sección 7.3 se describe un caso de estudio real, al que se han aplicado los algoritmos propuestos para estudiar la relación entre los atributos y proponer distintas soluciones. Por último, en la Sección 7.4, se analiza brevemente el comportamiento de los algoritmos sobre otros casos de estudio, dos de ellos reales y el resto generados al azar.

7.1. Impacto del Algoritmo de Mejora

Una de las partes más complejas del presente trabajo ha sido la elaboración de un método aleatorizado para obtener soluciones factibles del problema. El Algoritmo Constructivo, presentado en el Capítulo 3, parece lograr ese propósito de forma satisfactoria, y requiere de un esfuerzo computacional razonable, como se verá a lo largo de este capítulo. Sin embargo, por la propia naturaleza del método de construcción, las soluciones obtenidas pueden presentar elementos superfluos y/o claramente mejorables. El Algoritmo de Mejora, descrito en el Capítulo 4, aprovecha la estructura de las soluciones para introducir ciertos cambios con la finalidad de aumentar su calidad. En esta sección se analiza el beneficio que ofrece aplicar el Algoritmo de Mejora a las soluciones obtenidas mediante el Algoritmo Constructivo.

Se han diseñado cinco instancias test para analizar el efecto que produce la aplicación combinada de los algoritmos Constructivo y de Mejora frente a la aplicación únicamente del Algoritmo Constructivo. La Tabla 7.1 muestra un resumen de las características de cada una de estas instancias. El número total de nodos oscila entre 10 y 17, el de arcos

entre 29 y 41, y el de vehículos entre 20 y 30. En todos los casos, los vehículos se clasifican en dos tipos diferentes. Los vehículos del primer tipo tienen una capacidad de 20 lotes y los del segundo tipo una capacidad de 10 lotes. La cantidad de material que se debe repartir, *qglobal*, está comprendida entre 180 lotes, en la instancia Test3, y 400 lotes, en la instancia Test5. El resto de características -estructura de la red, cantidades disponibles de material, cantidades demandadas de material, probabilidades de disponibilidad de los arcos, probabilidades de sufrir asaltos en los arcos, etc.- presenta diferencias sustanciales entre las instancias test. Estas instancias se utilizan, además, para la calibración de los parámetros de las metaheurísticas, cuestión que se tratará en la Sección 7.2.

	# Depósitos	# Destinos	# Nodos	# Arcos	# Vehículos	*qglobal*
Test1	2	4	13	30	20	200
Test2	1	6	13	29	22	200
Test3	3	5	12	33	20	180
Test4	4	4	17	41	30	200
Test5	3	4	10	29	30	400

Tabla 7.1: Resumen de las instancias test

Se han obtenido 1000 soluciones en cada una de las instancias test, aplicando el Algoritmo Constructivo seguido del Algoritmo de Mejora y tomando como función objetivo, en todos los casos, la suma ponderada (fob), dando pesos iguales a todos los atributos. En la Tabla 7.2 aparece un resumen de los resultados obtenidos. Además del valor medio obtenido para la función objetivo en cada instancia (columnas encabezadas con fob), la tabla muestra el porcentaje de tiempo (columnas encabezadas con $\%CPU$) dedicado a cada proceso. La aplicación del Algoritmo de Mejora supone una disminución en el valor medio de la función objetivo en todos las soluciones obtenidas, siendo la disminución media en el conjunto de todas las instancias test de un 29.2 %. Además, la fase de mejora supone, en media, un 41.8 % sobre el tiempo de computación total necesario para obtener una solución, frente a un 58.2 % que conlleva la fase constructiva. La experimentación completa ha requerido 4.5 segundos de computación. Estos resultados indican que el Algoritmo de Mejora aumenta significativamente la calidad de las soluciones, sin suponer un incremento desproporcionado en el tiempo de computación. En consecuencia, en el resto del presente capítulo, los algoritmos Constructivo y de Mejora se aplicarán siempre de forma conjunta para aumentar la calidad de las soluciones que se obtengan.

7.2. Calibración

Las metaheurísticas GRASP y ACO dependen de un conjunto de parámetros para los que es necesario fijar unos valores concretos. En esta sección se presentan los resultados obtenidos en los experimentos realizados con esa finalidad. En ambos casos, GRASP y ACO, se han efectuado las pruebas sobre las cinco instancias test introducidas en la Sección 7.1, tomando como función objetivo la suma ponderada dando pesos iguales a todos

	Constructivo		Mejora		
	fob	% CPU	fob	% CPU	% reducción fob
Test1	0.408	75.9	0.285	24.1	29.8
Test2	0.389	36.4	0.303	63.6	22.4
Test3	0.373	53.8	0.242	46.2	35.0
Test4	0.376	66.7	0.278	33.3	26.0
Test5	0.331	58.1	0.220	41.9	32.7
Media		58.2		41.8	29.2

Tabla 7.2: Algoritmo Constructivo frente a Constructivo + Mejora

los atributos.

Es preciso señalar que el objetivo de la calibración no es obtener una combinación óptima de valores para los parámetros, sino una combinación que se comporte de forma satisfactoria en el mayor número posible de potenciales instancias. Existen dos razones para actuar de este modo. Por un lado, el elevado número de parámetros que intervienen -seis en cada algoritmo- imposibilita un diseño de experimentos aplicable en un tiempo razonable que tenga en cuenta las interacciones entre todos los parámetros. Por otro lado, incluso en el caso de que se pudiese conseguir una combinación óptima para la muestra (en este caso de cinco instancias), difícilmente esa optimalidad se podría extrapolar a la población de todas las posibles instancias del problema. De hecho, las pruebas efectuadas para esta calibración sugieren conclusiones significativamente distintas en cuanto a los valores de los parámetros dependiendo de la instancia test que se examine.

7.2.1. Calibración del Algoritmo GRASP

El Algoritmo GRASP requiere la calibración conjunta de seis parámetros: el tamaño del conjunto élite, *nelite*; la proporción de soluciones construidas en la obtención del conjunto élite inicial, *telite*/*tsol*, que en lo sucesivo será denominada ϕ; el peso de la diversidad, respecto del valor de la función objetivo, para la ponderación de la calidad de una solución candidata a formar parte del conjunto élite, β; el peso de las funciones *greedy*, respecto del azar, para la ponderación de un candidato a movimiento en cualquier fase de la construcción de una solución, γ; el complemento a uno del peso de la intensidad al comienzo de la última fase del Algoritmo GRASP (GUIAR CONSTRUCCIÓN), σ_1; y la velocidad de crecimiento del peso de la intensidad a lo largo de esa última fase, σ_2.

En una primera etapa de la calibración se asignaron arbitrariamente valores a los seis parámetros. Se tomaron los parámetros de dos en dos y se ejecutó el Algoritmo GRASP sobre cada una de las instancias test, variando los valores de los dos parámetros elegidos en un rango de entre diez y veinte valores, dependiendo del parámetro, y manteniendo fijos los valores de los otros cuatro parámetros según las asignaciones establecidas a priori. Para cada variación de los parámetros de cada pareja, se obtuvieron $tsol = 2000$ soluciones en la ejecución del algoritmo. Los parámetros se emparejaron por analogías de esta forma:

$(nelite, \phi)$, (β, γ) y (σ_1, σ_2). Con los resultados obtenidos en esta primera etapa se estableció una nueva asignación de valores, concretamente, $nelite = 8$, $\phi = 0.02$, $\beta = 0.05$, $\gamma = 0.6$, $\sigma_1 = 0.5$ y $\sigma_2 = 4$.

En la segunda etapa de la calibración se tomaron los parámetros de uno en uno y se ejecutó el Algoritmo GRASP cien veces, obteniéndose $tsol = 2000$ soluciones en cada una de las ejecuciones, variando en cada caso el valor del parámetro elegido como se detallará a continuación y manteniendo fijos los valores de los otros cinco parámetros según las asignaciones establecidas tras la primera etapa de la calibración. Cada ejecución del Algoritmo GRASP, que requirió de media 1.2 segundos de computación, proporcionó un valor óptimo (mínimo) para la función objetivo, $fobmin$, y finalmente se calculó la media sobre las cien ejecuciones y sobre las cinco instancias test.

El parámetro $nelite$, correspondiente al número de soluciones del conjunto élite, se varió en el rango $\{2, 4, \ldots, 20\}$, obteniéndose los resultados que se muestran en la Figura 7.1. En el eje de abscisas se representan los valores del parámetro a estudiar, en este caso $nelite$, y en el eje de ordenadas la media de los valores mínimos obtenidos para la función objetivo en las quinientas ejecuciones del algoritmo (cien en cada una de las cinco instancias test). Como se aprecia, el algoritmo mejora rápidamente según aumenta el tamaño del conjunto élite, hasta llegar a 6 y, a partir de esa cifra, empeora sostenidamente. Como, además, el tiempo de computación es mayor para conjuntos élite con más soluciones, se ha determinado finalmente $nelite = 6$.

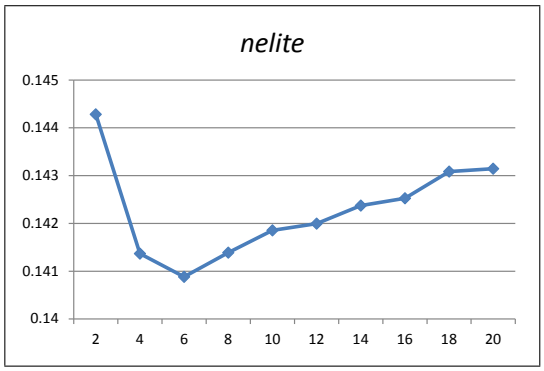


Figura 7.1: Calibración del número de soluciones del conjunto élite

Para calibrar el parámetro que controla la cantidad la cantidad de soluciones que se obtienen hasta que se determina el conjunto élite inicial, ϕ, se varió este en el rango $\{0, 0.02, \ldots, 0.2\}$. Los resultados, que se pueden observar en la Figura 7.2, señalan un crecimiento muy suave en la media de la función objetivo según aumenta ϕ. No obstante, la escala del eje de ordenadas indica una variación muy pequeña en los resultados para los distintos valores del parámetro. Por ese motivo, se ha determinado tomar un valor pequeño pero no nulo para ese parámetro, concretamente $\phi = 0.02$, que significa que el 2 % de las soluciones se deben obtener con la versión original de los algoritmos Constructivo y

de Mejora, mientras que el 98 % restante se obtendrán de forma guiada usando funciones *greedy* e intensidad.

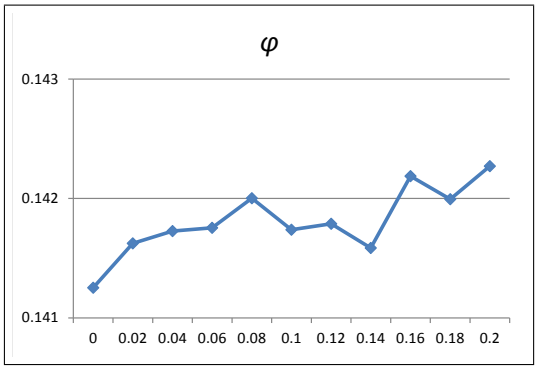


Figura 7.2: Calibración de la proporción de soluciones obtenidas sin guiar

A continuación se calibró el peso de la diversidad en la actualización del conjunto élite. Para β se tomaron los valores del rango $\{0, 0.01, \ldots, 0.1\}$. La Figura 7.3 muestra una tendencia suavemente descendente en la media de los valores mínimos de la función objetivo para, a partir de 0.04, crecer con cierta velocidad. Se ha resuelto tomar el mínimo que aparece en esta gráfica, $\beta = 0.04$, que refleja, como se podía suponer, que resulta beneficioso considerar la información dada por la diversidad para actualizar el conjunto élite.

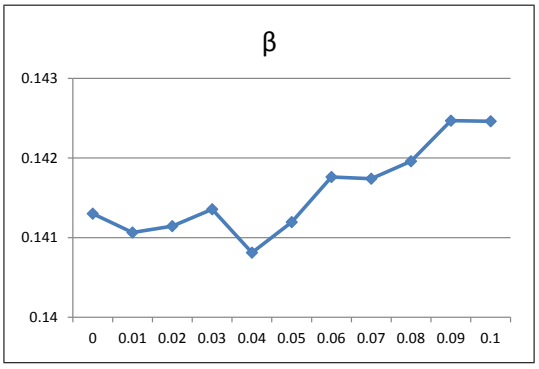


Figura 7.3: Calibración del peso de la diversidad frente a la función objetivo

Para calibrar el peso de la función *greedy*, γ, se consideraron los valores en el conjunto $\{0, 0.1, \ldots, 1\}$. Los valores medios obtenidos para la función objetivo aparecen en la Figura 7.4. El algoritmo muestra un funcionamiento levemente mejor -todos los valores medios están en el rango de una milésima- para los valores grandes de γ, si bien existe cierta discrepancia entre los resultados obtenidos en las distintas instancias. Por ese motivo, se ha preferido tomar $\gamma = 0.9$, con el que se mantiene una pequeña probabilidad en todos los elementos candidatos a ser seleccionados para formar parte de la solución que se esté construyendo.

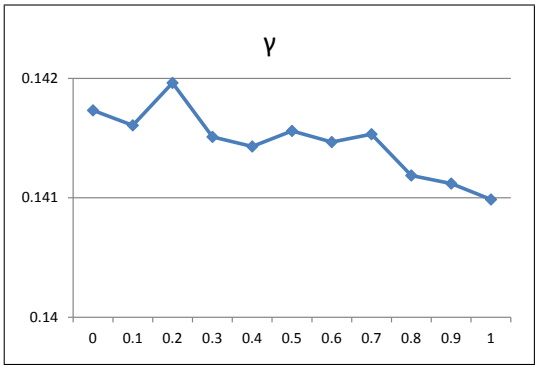
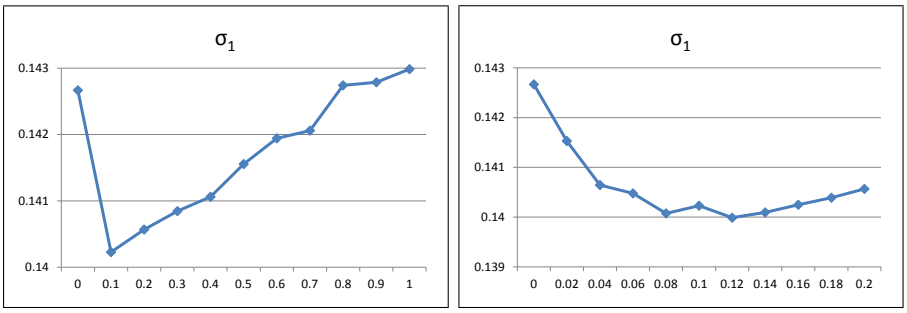


Figura 7.4: Calibración del peso de la función *greedy* frente al azar

Finalmente, se calibraron los parámetros que controlan la evolución del peso de la función intensidad, λ, cuya expresión es la siguiente:

$$\lambda = 1 - \sigma_1 e^{-\sigma_2 \frac{csol-telite}{tsol-telite}}$$

Para σ_1 se consideró inicialmente el rango de variación $\{0, 0.1, \ldots, 1\}$. El mínimo sobre este conjunto, como puede observarse en la Figura 7.5a, se alcanza claramente en 0.1, habiendo un fuerte descenso entre 0 y 0.1. Por esta razón, se decidió refinar la partición, considerándose un nuevo rango de valores, $\{0, 0.02, \ldots, 0.2\}$. En la Figura 7.5b puede apreciarse el comportamiento de la función objetivo sobre este nuevo conjunto. Se observa un descenso hasta aproximadamente $\sigma_1 = 0.08$, comenzando la función objetivo a crecer lentamente a partir de 0.12. Así, se ha determinado tomar $\sigma_1 = 0.1$, que significa que, al comienzo de la construcción guiada de soluciones, el peso de la intensidad es 0.9.



(a) (b)

Figura 7.5: Calibración del peso de la intensidad al comenzar la construcción guiada

Para σ_2, que indica la velocidad con la que crece la importancia de las funciones intensidad, se consideraron los valores del conjunto $\{0, 1, \ldots, 10\}$. Según se observa en la Figura 7.6, que recoge los resultados sobre este rango, inicialmente hay una disminución muy rápida en la función objetivo, atenuándose el decrecimiento a partir de $\sigma_2 = 5$, y

comenzando un ligerísimo crecimiento a partir de 7, por lo que se ha decidido fijar este parámetro a ese valor, $\sigma_2 = 7$.

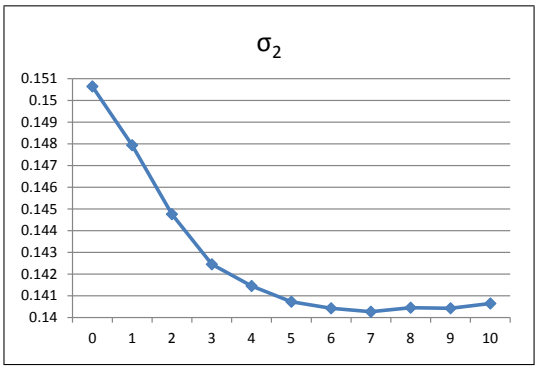


Figura 7.6: Calibración de la velocidad de crecimiento del peso de la intensidad

7.2.2. Calibración del Algoritmo ACO

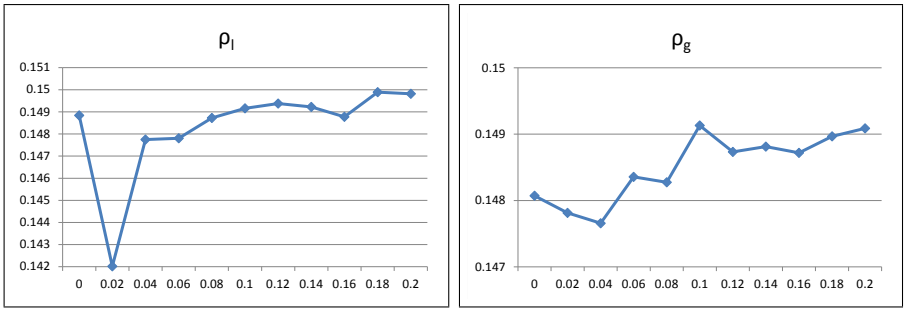
Los parámetros que intervienen en el Algoritmo ACO son también seis: el número de soluciones obtenidas en cada ciclo del algoritmo, m; la probabilidad de tomar directamente el elemento con mayor ponderación en las situaciones de decisión, q_0; los factores de evaporación de las feromonas en las actualizaciones local y global, respectivamente ρ_l y ρ_g; y, como en el Algoritmo GRASP, los parámetros que controlan la evolución del peso de la función intensidad, σ_1 y σ_2. Solo se ilustrarán las calibraciones que guardan mayor interés.

La metodología fue idéntica a la empleada en la calibración de los parámetros de la metaheurística GRASP. Inicialmente, se fijaron arbitrariamente los valores de los parámetros, usando, cuando fue posible, los valores que con mayor frecuencia se encuentran en la literatura. En la primera etapa se formaron las parejas (m, q_0), (ρ_l, ρ_g) y (σ_1, σ_2) y la experimentación con el Algoritmo ACO sobre las variaciones correspondientes de los valores de los parámetros arrojó una segunda asignación. En la segunda etapa de la calibración, se tomaron los parámetros de uno en uno y se varió cada parámetro considerado en un rango adecuado. Se ejecutó el Algoritmo ACO cien veces sobre cada instancia test y sobre cada valor del rango, tomando $tsol = 2000$ en cada caso. Los parámetros no considerados se mantuvieron fijos en los valores de la asignación establecida tras la primera etapa de la calibración. Cada ejecución del Algoritmo ACO requirió de media 1.5 segundos de computación. De cada ejecución se obtuvo el valor mínimo para la función objetivo, $fobmin$, y se calculó la media sobre las cien ejecuciones y sobre las cinco instancias test, que se representará, nuevamente, en el eje de ordenadas de las correspondientes gráficas.

Para el tamaño del ciclo, habitualmente en la literatura se han tomado los valores $m = 5$ o $m = 10$. Los resultados obtenidos para esta calibración no rectifican esta elección, existiendo pocas diferencias entre los resultados para los valores del rango analizado. Por tanto, se ha determinado tomar $m = 5$. Razonamientos análogos a los efectuados en la

calibración descrita en la subsección anterior han conducido a tomar los siguientes valores en los parámetros que controlan la evolución del peso de la función intensidad: $\sigma_1 = 0.2$ y $\sigma_2 = 5$.

En la calibración del resto de parámetros sí se han obtenido valores sustancialmente distintos a los estándares que pueden encontrarse en la literatura. Concretamente, para el factor de evaporación local, se ha tomado $\rho_l = 0.02$, según el mínimo que puede apreciarse en la Figura 7.7a, y que se ha visto refrendado en una partición más fina. Para el factor de evaporación global, se ha tomado $\rho_g = 0.04$, donde se alcanza el mínimo de la función objetivo media que se muestra en la Figura 7.7b. Ambos factores de evaporación toman, por lo tanto, valores pequeños, lo que indica que el rastro de las feromonas será persistente, especialmente en el caso de las actualizaciones locales, que se realizan con mayor frecuencia que las actualizaciones globales.



(a) Actualización local (b) Actualización global

Figura 7.7: Calibración de los factores de evaporación de las feromonas

Para terminar, la Figura 7.8 muestra la función objetivo media sobre los valores estudiados para q_0. Si bien el recorrido para la función objetivo es pequeño, existe una tendencia creciente a partir de $q_0 = 0.3$ que, por lo tanto, se ha tomado como valor definitivo para q_0. Esto significa que, en las situaciones de decisión del Algoritmo ACO, el 30 % de las veces se elegirá directamente el candidato con mayor ponderación, mientras que el 70 % restante se realizará un sorteo de acuerdo a las probabilidades de los candidatos, que son proporcionales a las ponderaciones correspondientes.

7.3. Caso de estudio principal. Haití 2010

Los algoritmos heurísticos presentados en esta memoria se han aplicado a diversos casos de estudio. Uno de ellos, basado en el terremoto ocurrido en Haití en enero del año 2010, es el objeto de esta sección, mientras que en la Sección 7.4 se presentan brevemente los resultados obtenidos sobre otros casos.

El seísmo que motiva este caso de estudio alcanzó una magnitud de 7.0 en la escala de Richter y su epicentro se localizó a escasos kilómetros de la capital del país, Puerto

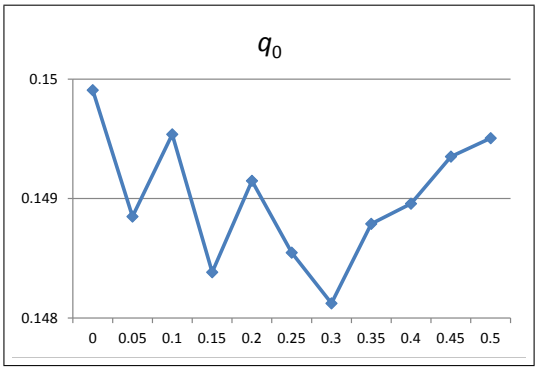


Figura 7.8: Calibración de la probabilidad de elegir directamente el movimiento con mayor ponderación

Príncipe, con efectos devastadores sobre la población, provocando más de trescientas mil víctimas mortales y alrededor de un millón y medio de damnificados. Las infraestructuras de la zona afectada fueron severamente dañadas, incluyendo el aeropuerto y el puerto, dificultando enormemente las labores de socorro a la población. Un análisis de las operaciones de respuesta gestionadas por las autoridades locales y diversas organizaciones internacionales puede consultarse en Pedraza *et al.* (2010), donde se incide en el papel que debe desempeñar la investigación operativa en la gestión de desastres.

En Vitoriano *et al.* (2011) se afronta el problema de distribución de ayuda humanitaria a diversos puntos de Puerto Príncipe y alrededores. En dicho artículo se aplicó un modelo de programación matemática que fue resuelto mediante un método exacto, cuyos resultados servirán como referencia para analizar el rendimiento de los algoritmos heurísticos desarrollados en este trabajo.

Se ha elegido este caso de estudio, además de por disponer de ciertos resultados con los que comparar los métodos heurísticos propuestos, por considerarse muy apropiado para ilustrar el problema. A pesar de corresponder a un entorno mayoritariamente urbano, el nivel de destrucción era tal que muchas calles y carreteras no estaban, o no se podía asegurar que estuvieran, en condiciones de ser transitadas. Asimismo, las operaciones de ayuda se realizaron en un ambiente de gran inseguridad durante los primeros días tras el terremoto, siendo frecuentes los asaltos a los convoyes, llegando a tener que suspenderse en algunos momentos el reparto de ayuda. Por último, las misiones de reparto debieron repetirse a lo largo de varios días, lo que permite reajustar los datos del problema a medida que se va reuniendo información acerca del estado de las vías, tiempos requeridos, costes, etc.

7.3.1. Descripción

El mapa logístico de la zona puede observarse en la Figura 7.9, y corresponde a la situación en que se encontraba el entorno de Puerto Príncipe transcurridos quince días desde que se produjera el seísmo principal. Los datos del caso de estudio que se presenta

a continuación, introducido en Vitoriano *et al.* (2011), fueron extraídos de la información ofrecida en OCHA (2015) y Redhum.org (2015).

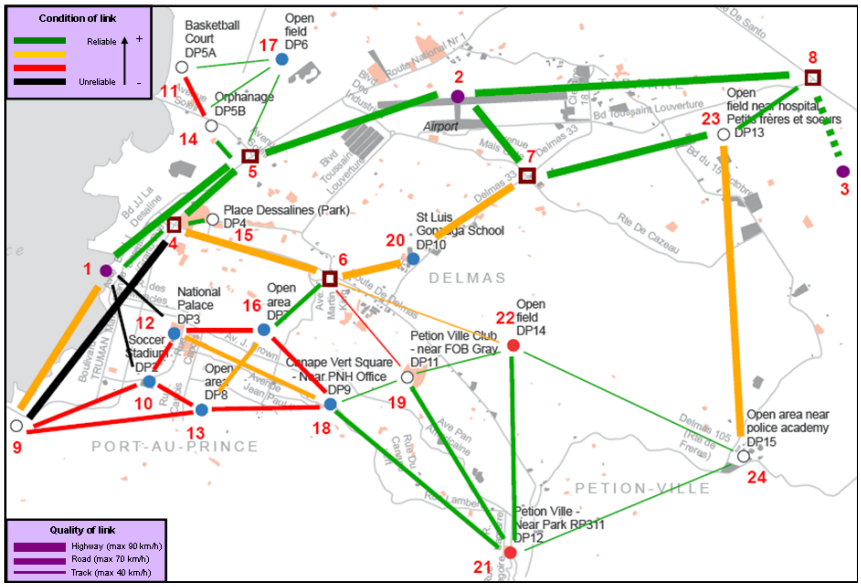


Figura 7.9: Red de transporte de Puerto Príncipe (Haití) tras el terremoto de 2010

La red consta de 24 nodos, que representan sendos puntos de interés. Los nodos 1, 2 y 3 son depósitos, donde se encuentran disponibles 60, 80 y 140 toneladas de ayuda humanitaria, respectivamente. Los nodos corresponden al puerto, al aeropuerto y a la ciudad dominicana de Jimaní, fronteriza con Haití. Los nodos 10, 12, 13, 16, 17, 18, 20, 21 y 22 demandan 30, 40, 30, 30, 10, 30, 40, 20 y 20 toneladas de ayuda, respectivamente. De estos nodos de demanda, el nodo 13 tiene especial prioridad, concretamente se le ha asignado prioridad 1, mientras que al resto se les ha asignado prioridad 0. El resto de nodos, 12 en total, son nodos intermedios, donde puede efectuarse transbordo de material.

Existen 84 arcos, correspondientes a 42 vías de doble sentido. Los dos arcos correspondientes a los dos sentidos de una misma vía se supone que presentan las mismas características. Las vías han sido clasificadas según su calidad -las vías más rápidas han sido representadas con líneas de trazo más grueso-, y según su fiabilidad -mediante colores que van del negro, menos fiable, al verde, más fiable, según la escala que se muestra en la esquina superior izquierda de la figura-. Toda la zona se considera insegura, por lo que las probabilidades de asalto a vehículos solos en los arcos se han establecido en el rango $[0.9, 0.96]$, mientras que la probabilidad de asalto a un convoy formado por 20 vehículos o más se ha estimado en 0.1, independientemente de la vía que se considere.

Para realizar el reparto se dispone en total de 300 vehículos, clasificados en 3 tipos distintos según sus características. Hay 70 vehículos grandes, con 3 toneladas de capacidad, 95 de tamaño medio, con una capacidad de 2 toneladas y 135 pequeños, que pueden trans-

portar hasta 1 tonelada de material. Al comienzo del reparto, los vehículos se encuentran situados en los depósitos y en los nodos intermedios 4, 6 y 18. Los vehículos más pequeños son los más rápidos, seguidos de los de tamaño medio, mientras que los vehículos grandes son los más lentos. El propósito de la misión es repartir 150 toneladas de ayuda desde los depósitos a los nodos de demanda.

7.3.2. Comparación de algoritmos

Para comparar los algoritmos propuestos se han realizado las siguientes pruebas. En primer lugar, se ha ejecutado 25000 veces la combinación de los algoritmos Constructivo y de Mejora. En la segunda columna de la Tabla 7.3 se muestra el mejor valor alcanzado para la medida de cada uno de los atributos entre todas las soluciones obtenidas, así como el valor mínimo para la suma ponderada, fob, con pesos iguales en todos los atributos, que puede encontrarse en la última fila con el nombre de "Agregada". El atributo tiempo se mide, en este caso, en minutos, mientras que el coste está expresado en dólares. El tiempo total de computación (que se abreviará como CPU en todas las tablas del presente capítulo donde se requiera, y se expresará en segundos) ha sido de 402 segundos.

	Constructivo + Mejora		GRASP 1 ejecución		ACO 1 ejecución	
	Valor	CPU	Valor	CPU	Valor	CPU
Tiempo	31.79	-	31.29	1003	27.95	2029
Coste	43226.00	-	36587.00	595	37055.00	2026
Equidad	0.07	-	0.05	507	0.08	982
Prioridad	0.00	-	0.00	1	0.00	1
Seguridad	42.96	-	37.07	515	37.00	2096
Fiabilidad	66.10	-	53.00	880	53.80	2114
Agregada	0.121	402	0.110	1274	0.117	2529

Tabla 7.3: Mejores soluciones obtenidas con una ejecución para el caso Haití 2010

Los algoritmos GRASP y ACO han sido ejecutados siete veces, seis tomando como función objetivo la medida de los correspondientes atributos y una para minimizar la suma ponderada con pesos iguales para todos los atributos. En cada una de las ejecuciones se han obtenido $tsol = 25000$ soluciones. En ambos algoritmos se han utilizado los valores de los parámetros establecidos en la calibración expuesta en la Sección 7.2. La Tabla 7.3 muestra el valor mínimo obtenido para la medida de cada atributo así como el tiempo de computación (en segundos) que ha requerido cada ejecución, para cada una de las metaheurísticas. En el caso de la prioridad, se ha parado la ejecución cuando se ha alcanzado el valor óptimo de ese atributo, 0, que corresponde a cualquier solución donde el nodo 13 recibe las 30 toneladas de material demandadas.

El Algoritmo GRASP ha proporcionado el valor más pequeño para los atributos coste, equidad y fiabilidad, así como para la suma ponderada. El Algoritmo ACO ha aportado

el mínimo para el tiempo y la seguridad. Todos los algoritmos han llegado rápidamente al valor óptimo para la prioridad. Puede observarse que el tiempo de computación varía bastante dependiendo del atributo que se considere, siendo, prioridad aparte, la equidad el que menos tiempo ha necesitado y la suma ponderada la función objetivo que más tiempo ha requerido. Esta relación entre los tiempos de computación es similar en otros casos de estudio. Se puede observar que el Algoritmo ACO es más lento que el Algoritmo GRASP en este caso, habiendo requerido entre el doble y el cuádruple del tiempo según el atributo. La combinación Constructivo + Mejora es bastante más rápida que el resto de los algoritmos analizados, proporcionando con 25000 soluciones valores razonablemente bajos en las medidas de todos los atributos, pero sí sustancialmente peores que los obtenidos con GRASP y ACO para algunos atributos, especialmente para el coste, seguridad y fiabilidad.

Además de las ejecuciones realizadas para obtener los resultados que se recogen en la Tabla 7.3, se han efectuado otra serie de pruebas con los algoritmos metaheurísticos, obteniéndose, en algunos casos, valores más pequeños para las medidas de los atributos que los que figuran en dicha tabla. La Tabla 7.4 reúne los mejores valores obtenidos para cada una de las medidas de los atributos. Además, en la segunda columna, aparecen los valores óptimos para los atributos cuando se dispone de ellos.

	Óptimo	GRASP	ACO
Tiempo	* 27.95	(+6.7 %) 29.83	27.95
Coste	* 35835.00	(+1.6 %) 36408.00	(+0.3 %) 35945.00
Equidad	0.00	0.00	0.05
Prioridad	0.00	0.00	0.00
Seguridad	-	35.07	37.00
Fiabilidad	-	53.00	53.00
Agregada	-	0.098	0.113

Tabla 7.4: Mejores soluciones obtenidas con varias ejecuciones para el caso Haití 2010

Los valores óptimos que se proporcionan para el tiempo y el coste han sido obtenidos mediante el modelo de flujo descrito en Vitoriano *et al.* (2011). Sin embargo, dicho modelo parte de hipótesis más restrictivas, convenientes para simplificar la formulación matemática lineal. Fundamentalmente, los vehículos no pueden visitar un mismo nodo varias veces. Además, la manera de medir los atributos tiempo y coste no es exactamente la misma que en el presente trabajo. En el modelo de flujo se impone que los regresos de los vehículos se realicen por el mismo camino por el que hayan efectuado el reparto, y, además, se supone que todos los convoyes que parten de un nodo lo hacen al mismo tiempo. De la estructura de las soluciones óptimas que se proporcionan en dicho artículo, se intuye que los valores correspondientes para tiempo y coste son también óptimos bajo las hipótesis y medidas que en este trabajo se han utilizado, aunque no puede asegurarse por carecer de un modelo exacto que pueda resolver el problema en un tiempo razonable. Por esta razón, los valores óptimos para tiempo y coste, que se aportan para servir de referencia, están precedidos por un asterisco en la tabla. Tampoco se dispone de los valores óptimos para los

atributos seguridad y fiabilidad -que en el modelo de flujo se miden de forma sensiblemente diferente- ni para la suma ponderada. Los valores óptimos para la equidad y la prioridad, en este caso, corresponden a los valores mínimos que admite la definición de ambas medidas.

El Algoritmo GRASP ha permitido encontrar el valor óptimo para la medida de la equidad, 0, que corresponde a una solución en la que la demanda de todos los nodos se satisface en la misma proporción. El supuesto óptimo para el tiempo total del reparto, 27.95 minutos, se ha podido conseguir aplicando el Algoritmo ACO, de hecho, con la primera ejecución de dicho algoritmo, como puede comprobarse en la Tabla 7.3.

En los casos en los que no se ha alcanzado el óptimo, se ha añadido, entre paréntesis, el porcentaje en el que se ha superado dicho óptimo, excepto en el caso de que el óptimo sea 0. La mejor solución obtenida para el tiempo del reparto con el Algoritmo GRASP presenta un valor un 6.7 % mayor que el tiempo óptimo. Para el coste del reparto los mejores valores obtenidos con los algoritmos GRASP y ACO se quedan muy cerca del óptimo, son un 1.6 % y un 0.3 % mayores que el coste óptimo, respectivamente. Ambos algoritmos proporcionan el mismo valor para la medida de la fiabilidad, mientras que, para la medida de la seguridad y la suma ponderada, el Algoritmo GRASP ha proporcionado mejores valores que el Algoritmo ACO. Este mejor rendimiento del GRASP respecto al ACO cuando se toma como función objetivo la suma ponderada se debe a que, por disponer de una cota más ajustada que otros atributos, la medida de la equidad, en la que el Algoritmo GRASP logra mejores resultados, tiene mayor influencia en dicha función objetivo.

Para terminar, la Tabla 7.5 presenta la matriz de pagos utilizando las mejores soluciones obtenidas con cualquiera de los algoritmos heurísticos propuestos. La matriz de pagos ayuda a entender la relación existente entre los distintos atributos y el grado de conflicto entre ellos. En cada fila, figuran los valores para todos los atributos en la mejor solución encontrada para el atributo asociado a dicha fila. Las columnas de la tabla corresponden también a los atributos: T (tiempo), C (coste), E (equidad), P (prioridad), S (seguridad) y F (fiabilidad). Los valores mínimos obtenidos se resaltan en negrita y están dispuestos sobre la diagonal, mientras que los peores valores se muestran subrayados.

	T	C	E	P	S	F
Tiempo	**27.95**	57292.50	0.45	0.17	79. 28	197. 35
Coste	47.25	**35945.00**	0.44	<u>1.00</u>	86.43	116.00
Equidad	<u>72.18</u>	72805.50	**0.00**	0.40	<u>153.96</u>	<u>203.50</u>
Prioridad	47.25	37295.00	0.46	**0.00**	79.38	105.50
Seguridad	36.91	59340.00	<u>0.47</u>	0.33	**35.08**	115.10
Fiabilidad	52.75	<u>85245.00</u>	0.47	1.00	74.83	**53.00**

Tabla 7.5: Matriz de pagos para el caso Haití 2010

La matriz de pagos revela el conflicto existente entre los atributos, indicando cómo debe deteriorarse el valor de algunos atributos cuando se pretende optimizar otro atributo

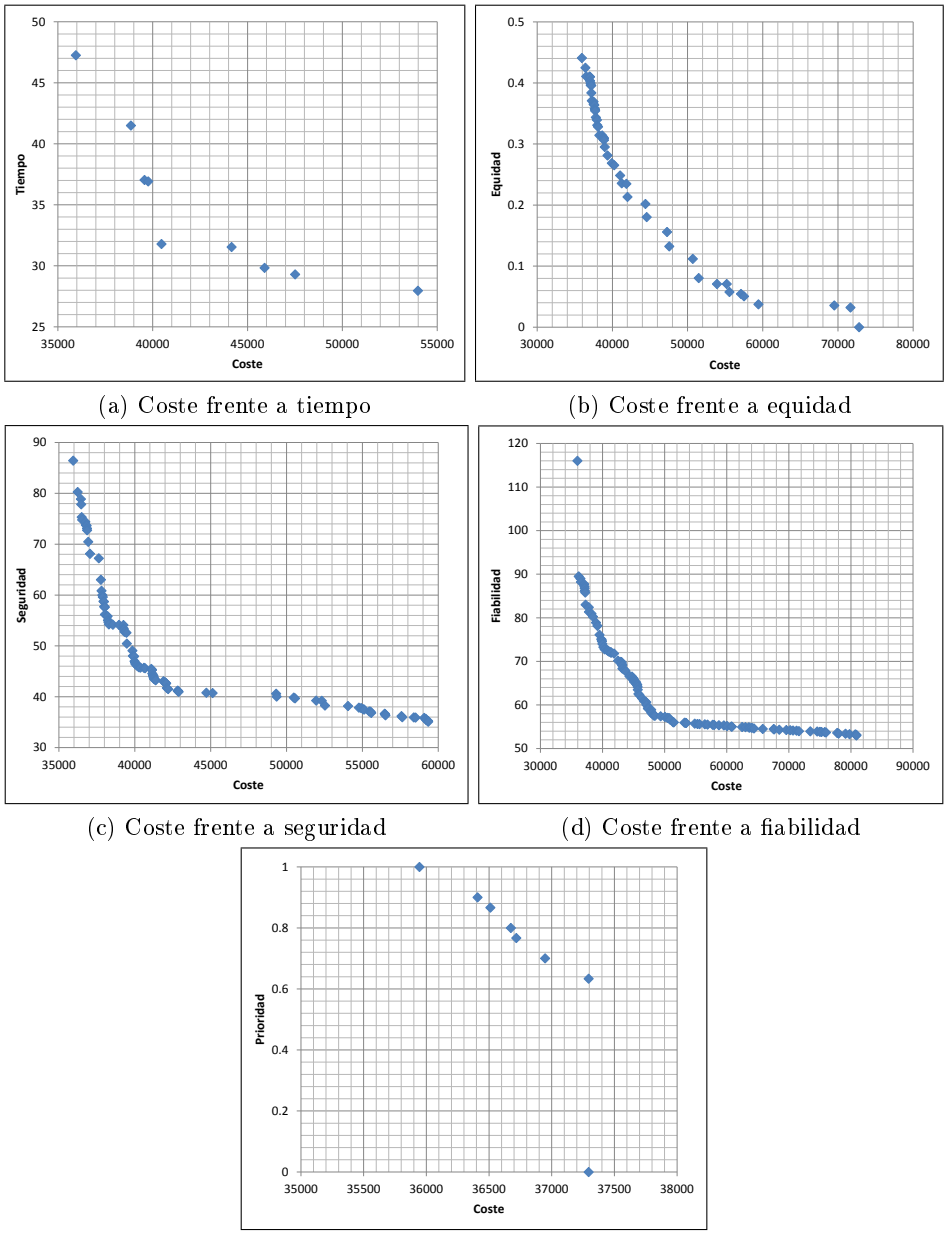
distinto. Particularmente, obtener el valor óptimo para la equidad ha supuesto perjudicar en gran medida al tiempo, coste, seguridad y fiabilidad. Además, de la matriz de pagos se pueden extraer los ideales (en este caso, en negrita) y los anti-ideales (subrayados) para constituir la función compromiso, fco_p, que se utilizará más adelante.

7.3.3. Frontera de Pareto

Dado el elevado número de atributos considerados, es especialmente difícil obtener la frontera de Pareto y, desde luego, imposible representarla gráficamente en su conjunto. No obstante, con el propósito de seguir explorando las relaciones entre los atributos del problema para este caso de estudio, se ha realizado una aproximación de la frontera de Pareto mediante proyecciones en dos y tres dimensiones. Para el caso bidimensional se ha ejecutado el Algoritmo GRASP sucesivas veces minimizando la suma ponderada, fob, y variando los pesos de dos de los atributos, g y g'. Para uno de los dos atributos elegidos se han considerado los pesos $peso_g \in \{0, 0.05, 0.1, \cdots, 0.95, 1\}$ y para el otro atributo elegido se ha establecido como $peso_{g'} = 1 - peso_g$, en tanto que el resto de atributos ha sido omitido. De esta forma se han obtenido estimaciones de las proyecciones de la frontera de Pareto sobre planos correspondientes a los dos atributos considerados.

La Figura 7.10 muestra las proyecciones en las que uno de los atributos considerados ha sido el coste, que se ha representado en el eje horizontal. En cada una de las gráficas, los puntos corresponden a las soluciones no dominadas obtenidas respecto de los atributos considerados. El rango de valores para el coste es un indicador del grado de conflicto entre el coste y el otro atributo considerado. Este rango es mayor para los atributos equidad y fiabilidad que, en este caso, son los atributos más confrontados al coste. Es decir, obtener una solución muy equitativa o que utilice mayoritariamente vías de las más fiables, solo puede hacerse a costa de aumentar significativamente el coste total del reparto. Recíprocamente, las soluciones con menor coste presentan valores cercanos a 0.5 en la medida de la equidad, máximo posible para ese atributo. El rango mínimo en el coste corresponde al atributo prioridad, ya que su valor óptimo, 0, se consigue para una solución con un coste bajo, concretamente de 37295 dólares. El mayor o menor número de puntos en la gráfica no solo está relacionado con el grado de conflicto, sino, sobre todo, con la cantidad de valores posibles que pueden tomar los atributos, siendo esta claramente menor para los atributos tiempo y prioridad.

En la Figura 7.11 están representadas las proyecciones sobre otros pares de atributos. Destaca la forma prácticamente lineal de los puntos no dominados correspondientes a los atributos fiabilidad y prioridad, como puede observarse en la Figura 7.11e. Esta forma se debe a las medidas definidas para ambos atributos y, especialmente, a que todos los arcos incidentes en el nodo prioritario (nodo número 13) tienen la misma fiabilidad, siendo además una fiabilidad baja (dichos arcos están representados en rojo en la Figura 7.9). La medida de la prioridad es menor cuanto más material reciba el nodo prioritario pero, para llegar al nodo prioritario, los vehículos que transporten el material han de transitar forzosamente por alguno de los arcos que desembocan en dicho nodo. Como todos esos arcos tienen la misma fiabilidad, una disminución en la medida de la prioridad supone un



(a) Coste frente a tiempo

(b) Coste frente a equidad

(c) Coste frente a seguridad

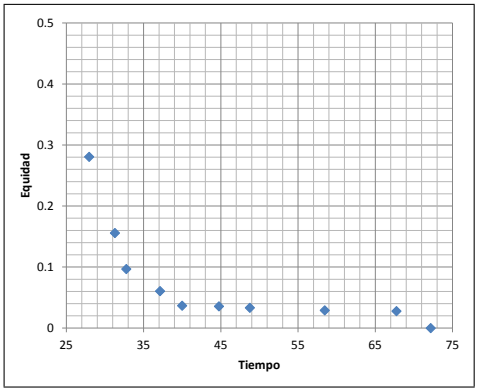
(d) Coste frente a fiabilidad

(e) Coste frente a prioridad

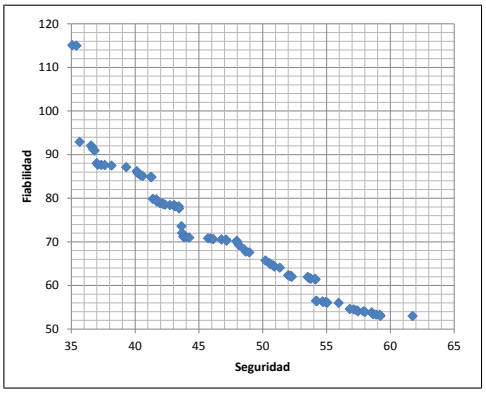
Figura 7.10: Proyecciones bidimensionales de la frontera de Pareto (coste frente al resto)

aumento proporcional en la medida de la fiabilidad.

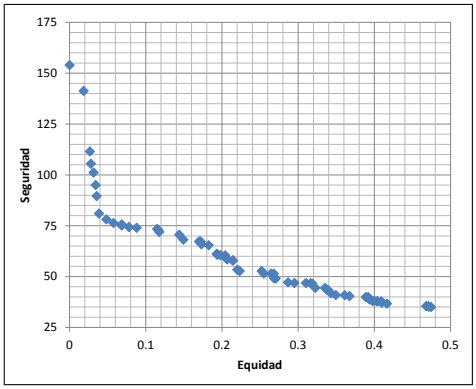
También puede destacarse el gran conflicto que existe entre los atributos equidad y seguridad (Figura 7.11c), como se desprende del amplio rango de valores que se cubre en las medidas de ambos atributos. Esto es debido a que, para conseguir una solución equitativa, hay que llegar a todos los nodos de demanda, siendo para ello necesario que muchos arcos



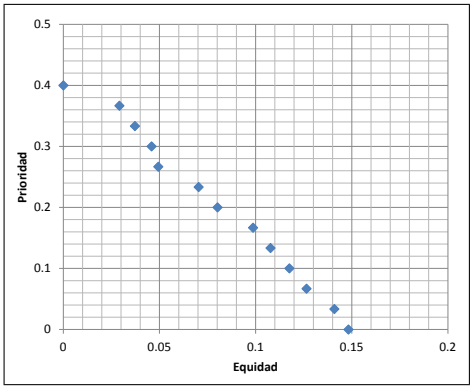
(a) Tiempo frente a equidad



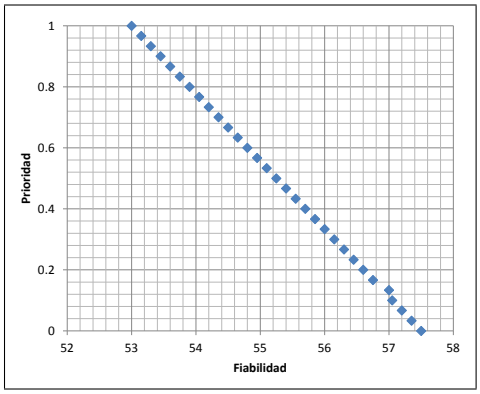
(b) seguridad frente a fiabilidad



(c) Equidad frente a seguridad



(d) Equidad frente a prioridad



(e) Fiabilidad frente a prioridad

Figura 7.11: Proyecciones bidimensionales de la frontera de Pareto (otros casos)

sean transitados, pero como, en este caso, las probabilidades de asalto son similares en todos los arcos, la medida de la seguridad depende fundamentalmente del número de arcos transitados -interesa que sea pequeño- y de los tamaños de los convoyes correspondientes -interesa que sean grandes, que consten de muchos vehículos-.

Como complemento de las gráficas bidimensionales, y para ahondar en las relaciones

entre atributos, se han realizado estimaciones de las proyecciones de la frontera de Pareto sobre espacios de tres dimensiones. Para ello se han considerado algunas ternas de atributos y se ha ejecutado el Algoritmo GRASP minimizando fob para cada combinación de pesos de esos tres atributos, donde cada peso pertenece al conjunto $\{0, 0.1, \cdots, 0.9, 1\}$. Las figuras 7.12, 7.13 y 7.14 muestran las gráficas con un mayor número de puntos no dominados, que corresponden a determinadas combinaciones de los atributos coste, equidad, seguridad y fiabilidad.

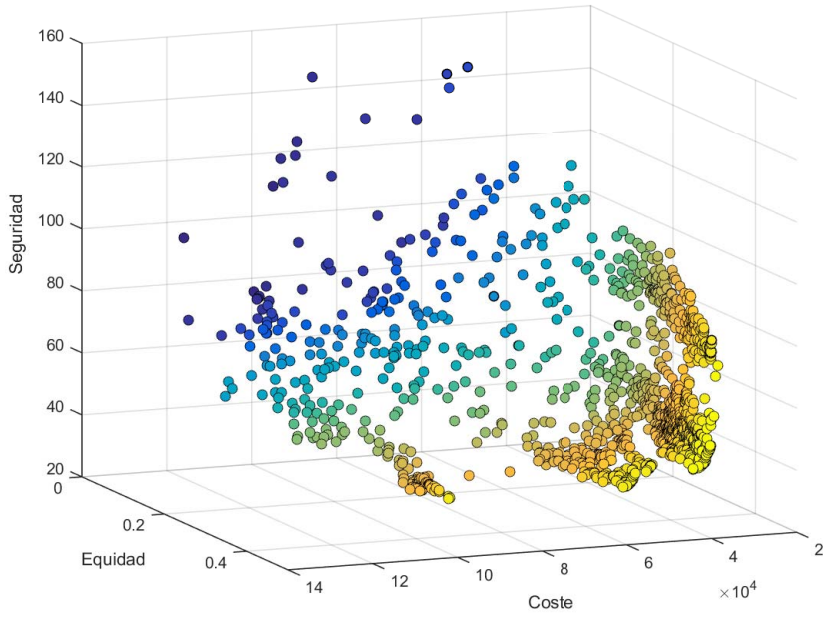


Figura 7.12: Proyección tridimensional de la frontera de Pareto (coste, equidad, seguridad)

La Figura 7.12 muestra los valores de los atributos coste, equidad y seguridad para todas las soluciones no dominadas obtenidas respecto de estos tres atributos. Para ayudar a percibir la tridimensionalidad, los puntos están coloreados según el valor de la medida del atributo cuyo eje representa la profundidad, en este caso, según el valor asociado a la equidad. El área con mayor densidad de puntos se encuentra en la parte derecha de la imagen, con valores razonablemente bajos para el coste y para la seguridad, pero altos para la equidad, lo que incide en el conflicto existente entre la equidad y los otros dos atributos considerados, conflicto ya señalado en los comentarios sobre las figuras 7.10b y 7.11c.

La gráfica que muestra la Figura 7.13 corresponde a los atributos coste, fiabilidad y seguridad, con los puntos coloreados según el coste del reparto. En la parte superior de la gráfica aparece un grupo de buenas soluciones respecto al coste y a la fiabilidad, pero malas respecto a la seguridad. Las soluciones de esa agrupación muy probablemente se han construido guiadas por un mismo conjunto élite, en una sola ejecución del algoritmo, con un peso bajo para el atributo seguridad. En la parte inferior de la imagen existe una zona llana, asociada a soluciones muy seguras y valores diversos en cuanto al coste y la fiabilidad.

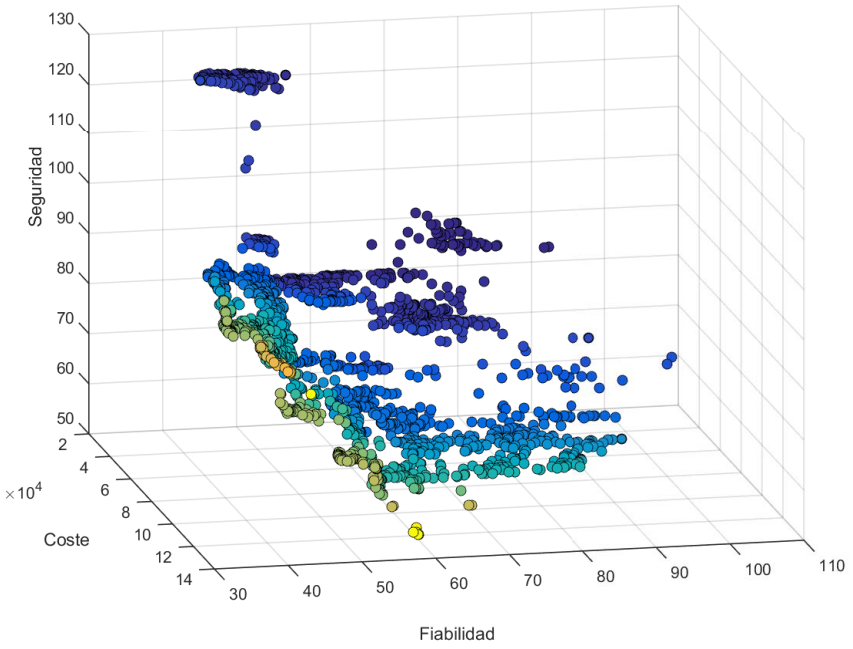


Figura 7.13: Proyección tridimensional de la frontera de Pareto (coste, fiabilidad, seguridad)

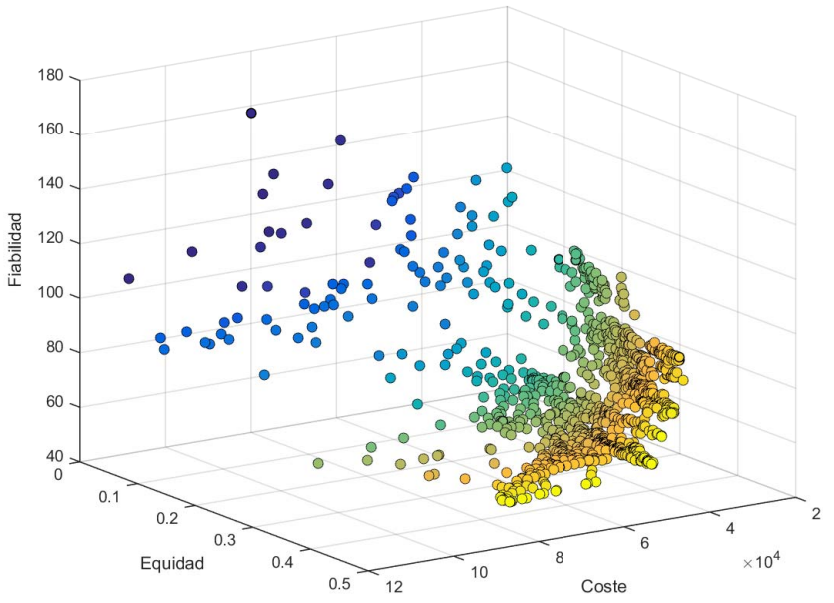


Figura 7.14: Proyección tridimensional de la frontera de Pareto (coste, equidad, fiabilidad)

Para terminar, la Figura 7.14 corresponde a la proyección sobre el espacio formado por los atributos coste, equidad y fiabilidad. Esta gráfica presenta ciertas similitudes con la

mostrada en la Figura 7.12 y, como en aquella, el coloreado de las puntos se ha realizado según los valores de la equidad. Nuevamente, la mayor concentración de puntos no dominados se presenta para valores bajos de coste y, en este caso, fiabilidad, y valores altos de equidad, pero de forma más acusada aún que en la figura mencionada. Esta estructura pone de manifiesto el conflicto existente entre la combinación coste-fiabilidad y la equidad, que se revela como el atributo más enfrentado al resto.

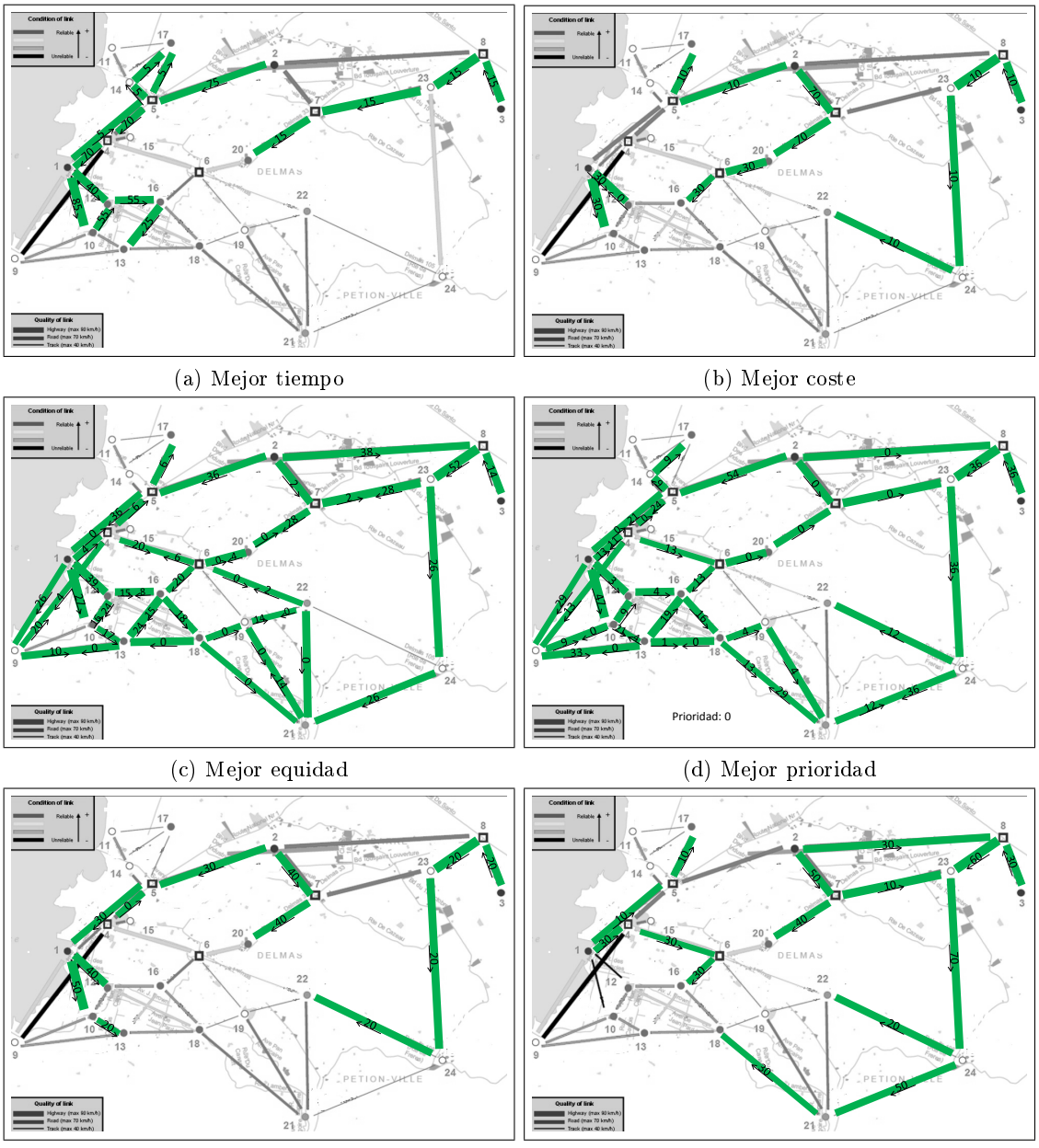
7.3.4. Soluciones propuestas

A través de la matriz de pagos (Tabla 7.5) y de la aproximación a la frontera de Pareto realizada en la Subsección 7.3.3 se ha pretendido ilustrar el conflicto que existe entre los atributos del problema, tomando como apoyo un caso real de especial relevancia. Sin embargo, en la práctica, el objetivo principal de este trabajo es desarrollar un método o conjunto de métodos que proporcionen un plan de reparto acorde a las preferencias del decisor. En esta subsección se exponen una serie de soluciones para este caso de estudio, obtenidas con alguno de los algoritmos presentados en esta memoria, ejecutados bajo distintas combinaciones de pesos para los atributos.

La Figura 7.15 reúne esquemas de las mejores soluciones obtenidas para cada uno de los atributos de forma individual. Las soluciones correspondientes a tiempo y coste han sido obtenidas en sendas ejecuciones del Algoritmo ACO tomando como funciones objetivo las medidas de dichos atributos. El resto de soluciones se han obtenido ejecutando el Algoritmo GRASP, minimizando en cada caso la medida del atributo correspondiente, excepto la mejor solución para la seguridad, que se ha obtenido tomando como pesos $peso_5 = 0.85$, $peso_2 = 0.15$ y el resto 0, es decir, dando un pequeño peso al coste. Los arcos coloreados de verde indican la presencia de convoyes, y las flechas el sentido en el que los convoyes recorren los arcos. Sobre cada flecha figura la cantidad de material que es transportada por cada arco. Nótese que en ciertos arcos hay convoyes que no transportan material (por ejemplo en el arco (12,1) de la Figura 7.15b), que corresponden a vehículos que deberán ser usados más adelante, y que puede haber convoyes recorriendo los dos sentidos de una misma vía (por ejemplo en el mencionado arco (12,1) y su opuesto de la Figura 7.15b). Los valores en las medidas de todos los atributos para cada una de estas soluciones pueden encontrarse en la matriz de pagos de la Tabla 7.5.

La solución propuesta para el tiempo está representada en la Figura 7.15a. Predomina el uso de vehículos medianos y pequeños, por ser más veloces, sobre todo estos últimos. Concretamente, todos los vehículos que parten del depósito 3 son pequeños, pues el arco (3,18) es, con diferencia, el de mayor longitud de la red, y el tiempo empleado en recorrerlo determina, en buena medida, la duración total del reparto. Los nodos de demanda más alejados de los depósitos, numerados como 18, 21 y 22, no reciben ayuda alguna en esta solución.

La mejor solución obtenida para el coste, representada en la Figura 7.15b, se caracteriza por el pequeño número de arcos recorridos, con predominio de los de menor longitud, y por el uso mayoritario de vehículos grandes, que ofrecen la mejor relación coste/capacidad.



(a) Mejor tiempo

(b) Mejor coste

(c) Mejor equidad

(d) Mejor prioridad

(e) Mejor seguridad

(f) Mejor fiabilidad

Figura 7.15: Mejores soluciones obtenidas respecto de cada uno de los atributos

Del nodo 3, correspondiente a la alejada ciudad de Jimaní, sale la mínima cantidad imprescindible de material para poder realizar el reparto, 10 toneladas, transportadas por tres vehículos grandes, de capacidad 3 toneladas, y un vehículo pequeño, con 1 tonelada de capacidad. Los nodos de demanda 13, 18 y 21 no reciben material.

En la solución correspondiente al atributo equidad (Figura 7.15c) todos los nodos de

demanda reciben el mismo porcentaje de ayuda sobre la cantidad demandada, el 60 %, por lo que se trata de una solución óptima para la equidad. Como se puede apreciar, ha sido necesario recorrer un gran número de arcos, lo que propicia malos resultados en los demás atributos.

El único requisito que debe tener una solución para ser óptima respecto de la prioridad, en este caso de estudio, es que el nodo 13 reciba las 30 toneladas de ayuda que demanda. Este requisito se cumple, por ejemplo, en la solución representada en la Figura 7.15d, a costa, nuevamente, de perjudicar los demás atributos. La solución representada es la primera solución óptima obtenida en una ejecución del Algoritmo GRASP, y ha requerido una cantidad ínfima de tiempo de computación. No obstante, no es complicado encontrar soluciones óptimas para la prioridad y con mejores valores para otros atributos. Por ejemplo, con el Algoritmo ACO, se ha obtenido una solución óptima, además de para la prioridad, para el tiempo del reparto.

En la Figura 7.15e se representa la mejor solución obtenida para la seguridad. Esta solución se caracteriza por el pequeño número de arcos recorridos, tan solo 12, y por el gran número de vehículos que forman cada convoy, siempre superando o estando cerca del tamaño disuasorio, que ha sido fijado en 20 en la instancia que se está estudiando. Cuatro son los nodos de demanda que no reciben material, los números 16, 17, 18 y 21.

Por último, el esquema de la mejor solución obtenida para la fiabilidad aparece en la Figura 7.15f. El buen rendimiento para el atributo fiabilidad se ha conseguido utilizando vías fiables para efectuar el reparto. Todos los arcos transitados son del tipo más fiable, que tienen una probabilidad de estar disponibles de 0.95 (vías representadas en verde en la Figura 7.9), excepto tres, los arcos (4,6), (7,20) y (23,24), que, con una probabilidad de estar disponibles de 0.75, pertenecen al segundo tipo más fiable (vías representadas en amarillo en la Figura 7.9). Los nodos de demanda 10, 12 y 13, pertenecientes a la zona más afectada por el terremoto, no reciben material de ayuda.

La información ofrecida por las mejores soluciones para cada uno de los atributos permite fijar unos valores ideales y anti-ideales necesarios para construir la función compromiso, fco_p. Estos valores ideales y anti-ideales pueden observarse en la matriz de pagos que muestra la Tabla 7.5. Se ha ejecutado el Algoritmo GRASP minimizando la función compromiso fco_1 y dando los mismos pesos a todos los atributos, realizándose de nuevo $tsol = 25000$ iteraciones. La solución obtenida está representada en la Figura 7.16. En esta solución, a todos los nodos de demanda les llega ayuda, el nodo prioritario recibe todo el material demandado y se obtienen valores razonablemente buenos para todos los atributos. Estos valores se muestran en la Tabla 7.6.

Es ciertamente deseable usar la función compromiso, aun teniendo en cuenta que requiere haber obtenido con anterioridad los ideales y anti-ideales, lo cual precisa de bastante tiempo de computación. Concretamente, en este caso de estudio, la aplicación completa de programación compromiso hubiese supuesto entre el triple y el cuádruple de tiempo que la ejecución del Algoritmo GRASP utilizando una función mono-objetivo.

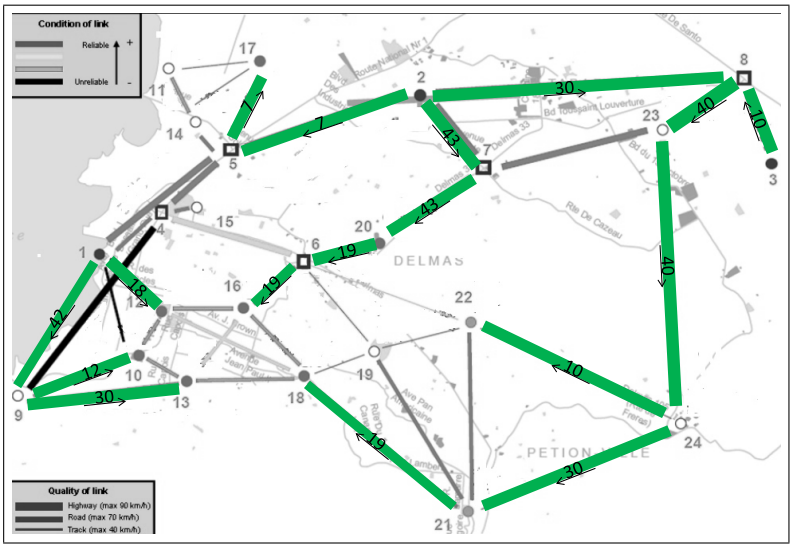


Figura 7.16: Solución obtenida minimizando fco_1 con pesos iguales

T	C	E	P	S	F
44.00	53954.50	0.17	0.00	82.18	86.05

Tabla 7.6: Valores para los atributos en la solución obtenida minimizando fco_1 con pesos iguales

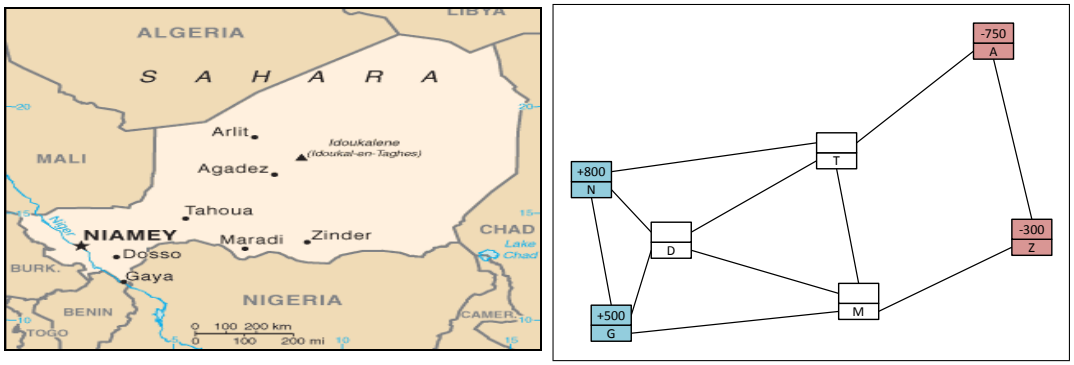
7.4. Otros casos de estudio

En esta sección se presenta un resumen de los resultados obtenidos al aplicar a otras instancias los algoritmos desarrollados en este trabajo. En este caso, el propósito no es realizar un análisis detallado de las relaciones entre los atributos ni estudiar la estructura de las soluciones, como se ha realizado en la sección anterior con el caso de estudio basado en el terremoto de Haití, sino, simplemente, mostrar el comportamiento de los algoritmos y comparar los resultados alcanzados entre sí. En los casos en los que se conocen los valores óptimos se aportarán como referencia.

7.4.1. Níger 2005

El segundo caso de estudio corresponde a la hambruna que se produjo en algunas regiones de Níger en los años 2005 y 2006. La Figura 7.17a muestra el mapa de Níger con las ciudades involucradas en la gestión de la crisis. En las ciudades de Niamey y Gaya, que actuaron como depósitos, se disponía de 800 y 500 toneladas de material de ayuda, respectivamente. La demanda en las ciudades de Agadez y Zinder se estableció en 750 y 300 toneladas respectivamente. Las ciudades de Dosso, Tahoua y Maradi se consideraron nodos intermedios para poder realizar transbordos de material. Un esquema de la red de transporte está representado en la Figura 7.17b. En cada nodo del esquema aparece la letra inicial de la ciudad a la que representa. Se tienen 22 arcos en total, con distintas características,

y que corresponden a los dos sentidos de las 11 conexiones que aparecen en la Figura 7.17b.



(a) Mapa

(b) Esquema

Figura 7.17: Red de transporte de Níger para paliar la hambruna de 2005

El objetivo de la operación es transportar 500 toneladas en total desde los depósitos a los nodos de demanda. Se cuenta con 376 vehículos, clasificados en tres tipos distintos, de capacidades 3, 2 y 1.5 toneladas respectivamente. Los vehículos están dispuestos en los depósitos y en los nodos intermedios. Pueden encontrarse más detalles de este caso de estudio en Vitoriano *et al.* (2009) y Ortuño *et al.* (2011).

La combinación Constructivo + Mejora se ha ejecutado 25000 veces para aplicarla a esta instancia, mientras que los algoritmos GRASP y ACO se han ejecutado una vez para optimizar cada atributo, tomando $tsol = 25000$. Aunque la red es pequeña, su estructura, el gran número de vehículos y la ubicación inicial de los mismos complican la tarea de encontrar soluciones factibles. Por esa razón, los tiempos de computación han sido mayores que en el caso Haití 2010. Concretamente, la ejecución de Constructivo + Mejora ha requerido 1235 segundos, el triple de tiempo que para el caso Haití 2010. El Algoritmo GRASP ha requerido aproximadamente 1500 segundos de media para cada atributo, mientras que para el Algoritmo ACO el tiempo medio de computación ha sido de 2000 segundos. La Tabla 7.7 recoge los valores obtenidos con cada uno de los algoritmos en los seis atributos. En la segunda columna aparecen los valores óptimos para los atributos cuando se dispone de ellos. Cuando un valor óptimo va precedido de asterisco -en este caso el coste óptimo- significa que el valor se ha obtenido con el modelo exacto de flujo y, por lo tanto, su optimalidad para el presente problema no está garantizada.

Con todos los métodos se ha alcanzado el óptimo para los atributos equidad y prioridad, tarea en este caso sencilla por existir únicamente dos nodos de demanda. La combinación de los algoritmos Constructivo y de Mejora ofrece el mejor valor con diferencia para el tiempo, pero proporciona valores superiores para el coste y la fiabilidad. El Algoritmo ACO ha permitido obtener el mejor valor para el coste, que resulta tan solo un 1.5 % superior al coste de referencia obtenido mediante el modelo de flujo que, en este caso, muy posiblemente sea también el óptimo para el problema actual. El valor obtenido en la

	Óptimo	Const. + Mejora	GRASP	ACO
Tiempo	-	34.44	46.25	46.25
Coste	* 63240.00	(+2.6 %) 64868.75	(+2.0 %) 64505.00	(+1.5 %) 64210.00
Equidad	0.00	0.00	0.00	0.00
Prioridad	0.37	(+0.0 %) 0.37	(+0.0 %) 0.37	(+0.0 %) 0.37
Seguridad	-	400.00	400.00	400.00
Fiabilidad	-	2.71	2.70	2.61

Tabla 7.7: Mejores soluciones obtenidas con una ejecución para el caso Níger 2005

medida de la fiabilidad ha sido el mismo con todos los algoritmos. Es de destacar que, en este caso de estudio, el Algoritmo ACO ha proporcionado valores iguales o mejores que el Algoritmo GRASP en todos los atributos, sin haber necesitado mucho más tiempo de computación.

7.4.2. Pakistán 2010

El último caso de estudio que se presenta está basado en las inundaciones que sufrió la región paquistaní de Punyab en el año 2010. El mapa logístico, que está representado en la Figura 7.18, consta de 41 nodos. Los nodos 1, 2 y 3 son depósitos, donde hay almacenados 1500 toneladas de ayuda en total, 300 toneladas en el nodo 1, 750 en el nodo 2 y 450 en el nodo 3. Los 26 nodos numerados del 4 al 29 demandan ayuda en diversas cantidades, y los nodos numerados del 30 al 41 son nodos intermedios. Existen 53 conexiones, que dan lugar a 106 arcos al contar los dos sentidos. La fiabilidad y velocidad de las vías está representada mediante la anchura y color de las conexiones, según las escalas que aparecen en la parte izquierda de la figura.

Para transportar la ayuda humanitaria se dispone de 53 camiones grandes y más lentos, con 25 toneladas de capacidad, 65 camiones de tamaño medio, con 15 toneladas de capacidad y 149 camiones pequeños pero más rápidos, con 3 toneladas de capacidad. La operación consiste en repartir 1000 toneladas de ayuda entre los nodos de demanda. Los datos para conformar este caso de estudio se han extraído de Álvarez (2011).

Las pruebas efectuadas han sido las mismas que en el caso Níger 2005. La combinación Constructivo + Mejora se ha ejecutado 25000 veces para aplicarla a esta instancia, mientras que los algoritmos GRASP y ACO se han ejecutado una vez para optimizar cada atributo, tomando $tsol = 25000$. La ejecución de los algoritmos Constructivo y de Mejora ha requerido 244 segundos. El Algoritmo GRASP ha requerido 250 segundos de media para cada atributo, mientras que para el Algoritmo ACO el tiempo medio de computación ha sido de 1000 segundos aproximadamente. A pesar del mayor número de nodos y de arcos de que consta esta instancia respecto de los casos de estudio anteriores, el tiempo de computación es bastante menor. Especialmente destacable es la diferencia con los tiempos de computación para el caso Níger 2005, cuya red logística es mucho más simple y sin embargo ha precisado de mucho más tiempo de computación. Este hecho indica claramente

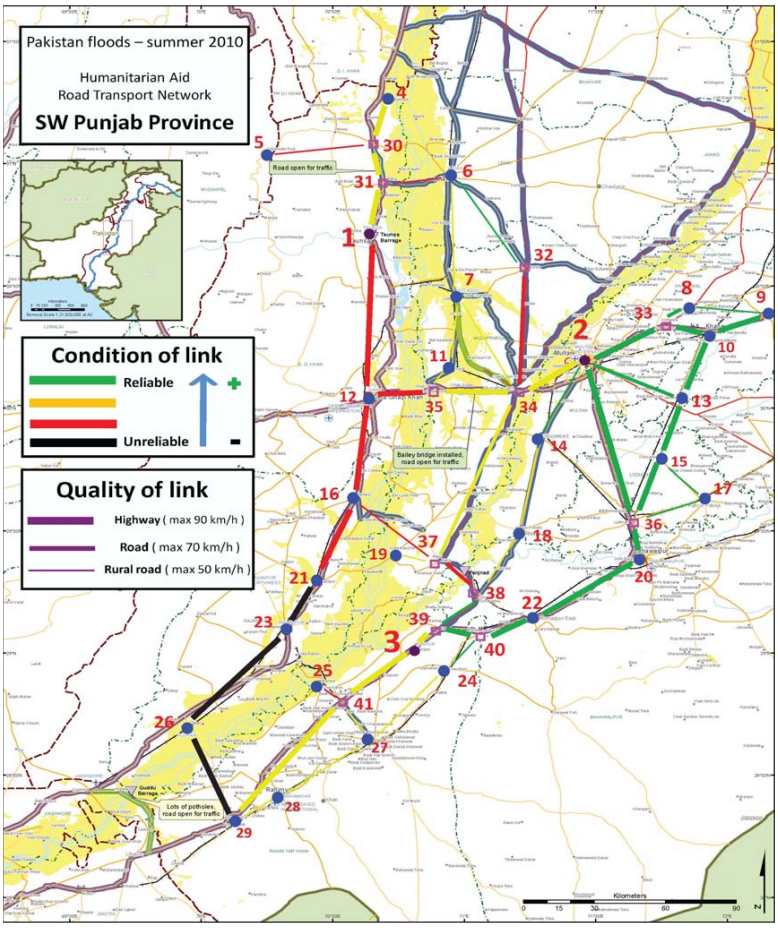


Figura 7.18: Red de transporte de la región de Punyab (Pakistán) tras las inundaciones de 2010

que, además del tamaño de la red, existen otros factores que afectan al rendimiento de los algoritmos propuestos, debido a la metodología empleada en el algoritmo constructivo en el que se basan. Entre estos factores se encuentran la estructura de la red, la ubicación de los vehículos, la relación entre la cantidad a repartir y la capacidad total de los vehículos, etc.

La Tabla 7.8 muestra los valores obtenidos con cada uno de los algoritmos considerados en los seis atributos. En la segunda columna aparecen los valores óptimos para los atributos, cuando se dispone de ellos, o los valores óptimos obtenidos con el modelo de flujo, en este caso, los valores para el tiempo y para el coste.

En cuanto al tiempo del reparto, todos los algoritmos mejoran con holgura el valor de referencia, 3.72 horas. Esto es debido a la diferente forma de medir el tiempo en el modelo de flujo, donde se impone que todos los convoyes que parten de un nodo lo hagan al mismo tiempo. Así, queda patente la ventaja de poder utilizar un modelo más general, donde las

	Óptimo	Const. + Mejora	GRASP	ACO
Tiempo	* 3.72	(-25 %) 2.79	(-54.8 %) 1.68	(-46.2 %) 2.00
Coste	* 692.83	(+29.6 %) 897.78	(-5.1 %) 657.32	(+4.5 %) 723.84
Equidad	-	0.15	0.21	0.32
Prioridad	0.00	0.00	0.00	0.00
Seguridad	-	209.40	37.76	34.41
Fiabilidad	-	344.50	232.25	236.25

Tabla 7.8: Mejores soluciones obtenidas con una ejecución para el caso Pakistán 2010

medidas de los atributos se ajustan más a la realidad. El Algoritmo GRASP también ha mejorado, en un 5.1 %, el coste del reparto respecto al valor de referencia, condicionado por la hipótesis simplificadora de que los vehículos regresan a sus ubicaciones originales por los mismos caminos por los que han transitado en el reparto, pero recorridos a la inversa.

La combinación Constructivo + Mejora ha proporcionado el mejor valor para la equidad, pero se ha comportado peor que los otros dos algoritmos en el resto de atributos, si se exceptúa la prioridad, donde todos los métodos aplicados han conseguido el valor óptimo. El Algoritmo GRASP ha ofrecido un excelente comportamiento, proporcionando los mejores valores en todos los atributos excepto en la seguridad, donde el Algoritmo ACO ha proporcionado un valor ligeramente menor, y mejorando los valores de referencia obtenidos con el modelo de flujo.

7.4.3. Instancias generadas al azar

Finalmente, se ha comparado el rendimiento de los algoritmos heurísticos propuestos sobre un conjunto de instancias generadas al azar. Los criterios para la generación de las instancias se basan en las características de los casos de estudio reales que han sido presentados en este capítulo. Se han considerado tres posibles tipos de vehículos, de capacidades 6, 4 y 2 toneladas, respectivamente, y con velocidades medias y costes también asignados de antemano.

En primer lugar se decide, arbitrariamente, el número de nodos. En las instancias que se van a estudiar, el número de nodos se ha fijado en cada uno de los valores del conjunto $\{10, 20, \ldots, 100\}$. Los nodos se reparten aleatoriamente sobre una cuadrícula y se crean arcos en una cantidad que, al menos, asegure la conexión de la red. La longitud de cada arco se asigna según las posiciones en la cuadrícula de los nodos que conecta el arco. Los nodos se clasifican según la función que deben desempeñar. Al menos uno y como máximo el 10 % del total de nodos se marcan aleatoriamente como depósitos y entre el 30 % y el 70 % como nodos de demanda. El resto serán nodos intermedios.

A continuación se determina el número de vehículos, que debe pertenecer al intervalo $[250, 350]$, y se fija el tamaño disuasorio en el 25 % del número de vehículos. La cantidad a repartir, *qglobal*, se establece aleatoriamente entre el 75 % y el 100 % de la capacidad

global del conjunto de vehículos, bajo el supuesto de que todos los vehículos son del tipo más pequeño. La oferta total y la demanda total se establecen, aleatoriamente y de forma independiente, entre un 125 % y un 150 % de la cantidad a repartir. Las cantidades disponibles de material son iguales en todos los depósitos y las cantidades requeridas de material son iguales en todos los nodos de demanda. A cada nodo de demanda se le asigna una prioridad aleatoria entre 0 y 1.

Los vehículos son ubicados aleatoriamente entre los nodos de forma que la probabilidad de que cada uno de ellos se sitúe en un depósito es 0.8. Para terminar, se determinan la velocidad máxima, la fiabilidad y las probabilidades de asalto de los arcos de forma que pertenezcan a ciertos intervalos estipulados. Todas las distribuciones de probabilidad que intervienen en el proceso son discretas uniformes o continuas uniformes en los intervalos que se indican.

Para establecer unos pesos que no desequilibren excesivamente la influencia de los atributos, se ha utilizado la información proporcionada al ejecutar 25000 veces los algoritmos Constructivo y de Mejora en el caso Haití 2010. De este modo, los pesos para los atributos se han tomado de forma proporcional a los cocientes entre las cotas y los valores medios obtenidos en los atributos. Una vez establecidos los pesos de los atributos, se ha ejecutado cada algoritmo durante media hora sobre cada una de las instancias generadas. En el caso de los algoritmos GRASP y ACO, se ha minimizado la suma ponderada. El número de iteraciones en la fase de construcción no guiada para el algoritmo GRASP se ha establecido en $telite = 100$, y la expresión para calcular la variable λ se ha modificado, en ambas metaheurísticas, para que dependa del tiempo de computación transcurrido y no del número de iteraciones efectuadas, ya que no se conoce a priori cuántas iteraciones se van a realizar en total.

El método de generación propuesto no garantiza que existan soluciones factibles para las instancias creadas, y no se dispone de ninguna herramienta que lo determine, más allá del propio Algoritmo Constructivo. Por esta razón, en cada caso en el que no se han obtenido soluciones factibles con ninguno de los algoritmos se ha desechado la instancia, sustituyéndose por otra instancia generada mediante el mismo procedimiento y con el mismo número de nodos. A partir de instancias con 100 nodos, que es el máximo para el que se proporcionan los resultados, esta situación se presenta con mayor frecuencia, bien por ser más probable que el problema no tenga solución, bien porque las soluciones pueden ser demasiado complejas para ser obtenidas mediante el Algoritmo Constructivo en un periodo corto de tiempo. En todo caso, con configuraciones diferentes en la generación -menos vehículos, menos *qglobal*, etc.-, que simplifican la estructura de las posibles soluciones, se han podido resolver satisfactoriamente instancias con hasta 150 nodos.

En la Tabla 7.9 se muestran el número de soluciones factibles construidas y los valores mínimos obtenidos para la suma ponderada, fob, con cada uno de los algoritmos, ejecutados durante 30 minutos sobre cada instancia. Para presentarlos en la tabla, los valores de fob se han multiplicado por 100 para favorecer su visualización.

	Const + Mejora		GRASP		ACS	
# Nodos	# Soluciones	fob	# Soluciones	fob	# Soluciones	fob
10	65434	15.94	56833	15.69	22460	15.37
20	249916	4.93	88952	4.93	36807	4.76
30	49092	6.54	39462	6.23	21033	6.26
40	64663	2.23	48539	2.13	12822	2.13
50	6907	4.85	66363	4.27	4304	4.36
60	52149	3.24	48667	2.90	21740	3.17
70	615	3.96	39564	3.45	2	4.40
80	80708	2.70	33569	2.58	16685	2.84
90	9227	3.43	28818	3.10	17204	3.29
100	3005	3.23	31562	2.85	12855	3.07
media	58172	5.11	48233	4.81	16591	4.97

Tabla 7.9: Número de soluciones obtenidas y valor de la función objetivo en las instancias aleatorias

La combinación Constructivo + Mejora permite obtener más soluciones, de media, que el resto de los algoritmos pero, sin embargo, proporciona peores valores en la función objetivo. Además, su velocidad construyendo soluciones, de la que dependen en gran medida los resultados que ofrece, varía mucho de unas instancias a otras. Por ejemplo, en las instancias con 20 y 80 nodos logra construir muchas más soluciones que el resto de los algoritmos pero, en otras instancias, especialmente en la que consta de 70 nodos, proporciona tan solo un 1.5 % del número de soluciones que consigue construir el Algoritmo GRASP.

El Algoritmo ACO necesita construir menos soluciones que el resto para ser competitivo, sobre todo en las instancias con pocos nodos. En las instancias más grandes presenta algunos problemas para encontrar las primeras soluciones factibles debido a la ausencia de una fase de construcción no guiada. Este hecho se manifiesta claramente en las dos soluciones que únicamente ha logrado obtener en la instancia con 70 nodos. Una posible explicación es la siguiente. Hasta que se obtienen $m = 5$ soluciones, las feromonas se recalculan mediante la actualización local, que penaliza a los elementos constitutivos de la última solución obtenida de cara a las siguientes iteraciones. En una instancia donde sea especialmente compleja la obtención de una solución factible, como parece ocurrir en la instancia con 70 nodos, esta estrategia diversificadora, que se ha mostrado exitosa en otras instancias, puede suponer un problema. Ejecutando el Algoritmo ACO con $m = 1$, lo que significa que tras cada iteración se realizan de forma consecutiva los dos tipos de actualización, local y global, se han obtenido para esa instancia 26470 soluciones y $fob = 0.035$, mismo valor que con el Algoritmo GRASP.

Otra circunstancia que puede incidir negativamente en algunos casos es el uso de las funciones *greedy* desde el comienzo del algoritmo, cuando todavía no existe un historial suficiente de feromonas que puedan guiar correctamente la construcción. En algunas instancias se ha disminuido a 0 el valor de σ_1, que indica que no se utilizan funciones *greedy*, para que

el Algoritmo ACO pueda obtener las primeras soluciones y, a partir de ahí, funcionar con normalidad usando las feromonas para guiar la construcción. Los resultados que se muestran en la Tabla 7.9, para las instancias a partir de 60 nodos, han sido obtenidos con $\sigma_1 = 0$.

El Algoritmo GRASP es el más estable y el que menos variación presenta en el número de soluciones obtenidas. Su metodología, que se basa primero en una breve fase de construcción aleatorizada para después guiar la construcción a través de soluciones élite, es la clave en este hecho. Además de la estabilidad mostrada, a partir de la instancia con 30 nodos, ha proporcionado los mejores valores en la función objetivo.

Capítulo 8

Conclusiones, aportaciones e investigación futura

8.1. Conclusiones

Destacamos las siguientes conclusiones extraídas de esta monografía:

- Se ha descrito un problema multicriterio correspondiente a la última fase de reparto en logística humanitaria, resultado de generalizar un problema ya estudiado en la literatura. El problema tiene en cuenta algunos atributos propios de la logística humanitaria, aunque poco estudiados, como es el caso de la atención prioritaria a los más necesitados, la seguridad, que contempla la posibilidad de asaltos a los convoyes de ayuda, y la fiabilidad, que incorpora la información existente acerca del estado de las vías de comunicación. Se ha introducido nueva terminología y se han propuesto medidas para todos los atributos de que consta el problema.
- Se ha presentado un modelo de programación matemática para el problema, que integra todas sus hipótesis y elementos que lo conforman. El modelo, de programación lineal entera, incluye variables y restricciones que representan las relaciones de precedencia entre los arcos de la red, impuestas por la hipótesis de que los vehículos deben trasladarse formando convoyes.
- Se ha analizado la naturaleza multicriterio del problema y se han expuesto enfoques que pueden ser aplicados en la práctica, teniendo en cuenta que las decisiones en este contexto deben tomarse, habitualmente, de forma urgente.
- Se ha presentado un algoritmo novedoso, que se ha denominado Algoritmo Constructivo, que permite construir soluciones factibles en un periodo razonable de tiempo. Algunos de sus procedimientos incluyen un considerable componente aleatorio con el propósito de que el método pueda proporcionar una gran diversidad de soluciones factibles. En la estructura del algoritmo cobra gran importancia la sincronización de los vehículos a través de las relaciones de precedencia entre los arcos.
- Dadas las características del problema, en el que encontrar soluciones factibles no es un proceso sencillo, y pequeños movimientos en la solución hacen perder esta

factibilidad, concluimos que los algoritmos basados en búsqueda local en entornos no son adecuados para este problema. Sin embargo, se ha desarrollado un algoritmo heurístico, que se ha denominado Algoritmo de Mejora, que permite depurar cada solución construida del problema, respetando la compleja estructura de la solución, preservando su factibilidad, y mejorándola en diversos aspectos.

- Se han desarrollado algoritmos adaptando dos metaheurísticas centradas en el uso de información de iteraciones previas para guiar el proceso de construcción, con el propósito de conseguir mejores soluciones desde el punto de vista de la función objetivo que se considere.

- En el algoritmo basado en la metaheurística GRASP, que se ha denominado Algoritmo GRASP, se ha empleado un conjunto élite de soluciones para guiar la construcción a través de funciones intensidad, labor que realizan junto con las funciones *greedy* que se han definido en diversas fases de la construcción.

- En el algoritmo basado en la metaheurística colonia de hormigas, que se ha denominado Algoritmo ACO, se han definido feromonas de distintos tipos, que son depositadas por hormigas especializadas asociadas a los vehículos y a los lotes de material en su desplazamiento por la red. Se ha introducido el concepto de feromonas efectivas, que son las encargadas de guiar realmente la construcción de las soluciones junto con las funciones *greedy*. Estas feromonas efectivas son actualizadas durante la ejecución de los procedimientos donde se hace uso de hormigas artificiales, a diferencia de las feromonas propiamente dichas, que son actualizadas tras cada solución obtenida, con el propósito de diversificar la construcción, y tras cada ciclo de soluciones, para potenciar los elementos de la mejor solución obtenida.

- Se ha analizado el impacto que supone aplicar el Algoritmo de Mejora a las soluciones obtenidas mediante el Algoritmo Constructivo, y se ha determinado la conveniencia de ser aplicados combinadamente para aumentar su eficacia.

- Se han calibrado los parámetros de los algoritmos basados en las metaheurísticas a través de una extensa experimentación realizada sobre un conjunto de instancias diseñadas para tal efecto. Asimismo, se ha justificado la estrategia que se ha adoptado para realizar dicha calibración. Se ha comprobado que algunos de los valores de los parámetros comúnmente utilizados en la literatura no son los que mejor se comportan en nuestro problema. En otros casos, se han constatado diferencias notables de unas instancias a otras respecto de los mejores valores posibles para ciertos parámetros. Finalmente, los valores que se han establecido para los parámetros han demostrado funcionar bien en un conjunto amplio de instancias, en especial en las instancias basadas en casos reales, que era el objetivo prioritario de la calibración.

- Se han aplicado los algoritmos propuestos sobre un caso de estudio principal basado en el terremoto ocurrido en Haití en enero del 2010, cuya importancia en el ámbito de la logística humanitaria ha sido puesta de manifiesto en la literatura, y para el que ya existían resultados basados en modelos de flujo. Se ha analizado el comportamiento de los algoritmos, comparando los resultados obtenidos con cada uno de ellos y con los

resultados existentes. Se ha comprobado que, en este caso de estudio, la combinación de los algoritmos Constructivo y de Mejora permite obtener soluciones satisfactorias para todos los atributos en menor tiempo que el resto, y que los algoritmos basados en metaheurísticas proporcionan mejores soluciones, alcanzando el óptimo o quedándose muy cerca en los atributos en los que se dispone de referencias.

- Sobre el caso de estudio principal, se ha realizado un análisis detallado de las relaciones entre los atributos, incidiendo en el conflicto que existe entre ellos. Con este propósito, se ha obtenido la matriz de pagos del problema y se han determinado, representado gráficamente y comentado diversas proyecciones de la frontera de Pareto aproximada sobre espacios de dos y tres dimensiones.

- Se han analizado y representado gráficamente las mejores soluciones obtenidas respecto de cada atributo para el caso de estudio principal. Se ha recomendado el uso de programación compromiso, cuando sea posible, y se ha presentado e ilustrado una solución equilibrada obtenida por medio de dicha técnica multicriterio.

- Se ha estudiado el rendimiento de los algoritmos propuestos sobre otras instancias, bien basadas en casos reales, bien generadas al azar. Se ha comprobado el buen funcionamiento de los algoritmos, especialmente de los basados en metaheurísticas, y se han puesto de relieve las ventajas que ofrece el uso de un modelo realista y de medidas precisas de los atributos. Se ha observado que el Algoritmo GRASP presenta un mejor rendimiento y una mayor estabilidad sobre instancias de gran tamaño, mientras que sobre dichas instancias el Algoritmo ACO puede precisar de ajustes en los valores de los parámetros. En instancias de tamaño medio, el Algoritmo ACO, aun con menor velocidad en la construcción que los demás algoritmos analizados, logra proporcionar con frecuencia mejores soluciones que el resto. Se ha puesto de manifiesto que el tamaño de la instancia no es el único factor que determina el rendimiento de los algoritmos, y se han sugerido otros factores que influyen en este aspecto.

8.2. Principales aportaciones

Los resultados principales de la tesis han sido presentados en los congresos que se enumeran a continuación:

- FuzzyMAD. Madrid (España). Diciembre 2011.
- SEIO XXXIII. Madrid (España). Abril 2012.
- IFORS XX. Barcelona (España). Julio 2014.
- II WORLO. Burgos (España). Noviembre 2014.
- Evostar 2015. Copenhague (Dinamarca). Abril 2015
- POMS 2015. Washington (Estados Unidos). Mayo 2015

- XV EU/ME. Móstoles (España). Septiembre 2015

Además, se han publicado los siguientes trabajos basados en la investigación realizada para la consecución de esta tesis:

- M.T. Ortuño, P. Cristóbal, J.M. Ferrer, F.J. Martín-Campo, S. Muñoz, G. Tirado y B. Vitoriano (2012). Decision Aid Models and Systems for Humanitarian Logistics. A Survey. En B. Vitoriano, J. Montero, y D. Ruan, editores, *Decision Aid Models for Disaster Management and Emergencies*, pp. 17-44. Atlantis Computational Intelligence Systems.

- J. M. Ferrer, M. T. Ortuño, G. Tirado y B. Vitoriano (2015). Heuristics for the design of safe humanitarian aid distribution itineraries. En A. Mora y G. Squillero, editores, *EvoApplications 2015*, volumen 9028 de *Lecture Notes in Computer Science*, pp. 721-731. Springer.

- J. M. Ferrer, M. T. Ortuño y G. Tirado (2015). A GRASP metaheuristic for humanitarian aid distribution. *Journal of Heuristics*, DOI 10.1007/s10732-015-9302-5

8.3. Líneas futuras de investigación

A continuación se proponen algunas ideas prometedoras para establecer líneas futuras de investigación:

- Probar nuevos procedimientos en el Algoritmo de Mejora. Por ejemplo, algunos fragmentos de las rutas de ciertos vehículos pueden ser sustituidos por otros fragmentos que unan los mismos nodos, siempre que se respeten las relaciones de precedencia, para mejorar determinados atributos.

- Explorar las posibilidades de fortalecer la búsqueda local en las algoritmos basados en metaheurísticas. Una idea a priori interesante consiste en, al obtener una solución, especialmente si esta es buena, destruir parte de la solución y aplicar el Algoritmo Constructivo sucesivas veces tomando como punto de partida la porción de solución salvada.

- Desarrollar un algoritmo que combine las metaheurísticas GRASP y colonia de hormigas, potenciando los aspectos positivos que presentan ambos métodos en la resolución del problema.

- Desarrollar un modelo estocástico que, mediante un enfoque exacto, permita obtener soluciones óptimas en algunas versiones simplificadas del problema.

- Crear un modelo de simulación que incorpore los elementos aleatorios del problema. Ensayar, sobre el modelo de simulación, el comportamiento de las soluciones obtenidas en casos de estudio reales mediante los métodos heurísticos propuestos.

- Adaptar los algoritmos desarrollados en este trabajo para aplicarlos a otros problemas de rutas de reparto, no necesariamente correspondientes a logística humanitaria, en los que tenga sentido la hipótesis de que los vehículos formen convoyes para sus desplazamientos por las vías de comunicación.
- Desarrollar un sistema de ayuda a la decisión basado en los métodos heurísticos descritos en esta monografía.

Summary

Introduction

Large scale disasters, such as the one caused by the Typhoon Haiyan, which devastated portions of the Philippines in 2013, or the catastrophic 2010 Haiti earthquake, which caused major damage in Port-au-Prince and other settlements in the region, have massive and lasting effects on populations. Nowadays, disasters can be considered as a consequence of inappropriately managed risk. These risks are the product of hazards and vulnerability, which refers to the extent to which a community can be affected by the impact of a hazard. In this way, developing countries, due to their greater vulnerability, suffer the highest costs when a disaster occurs.

Disaster relief is a challenge for politics, economies, and societies worldwide. Humanitarian organizations face multiple decision problems when responding to disasters. In particular, once a disaster strikes, the distribution of humanitarian aid to the population affected is one of the most fundamental operations in what is called humanitarian logistics. This term is defined as the process of planning, implementing and controlling the efficient, cost-effective flow and storage of goods and materials as well as related information, from the point of origin to the point of consumption, for the purpose of meeting the end beneficiaries' requirements and alleviate the suffering of vulnerable people, [the Humanitarian Logistics Conference, 2004 (Fritz Institute)]. During the last decade there has been an increasing interest in the OR/MS community in studying this topic, pointing out the similarities and differences between humanitarian and business logistics, and developing models suited to handle the special characteristics of these problems.

Several authors have pointed out that traditional logistic objectives, such as minimizing operation cost, are not the most relevant goals in humanitarian operations. Other factors, such as the time of operation, or the design of safe and equitable distribution plans, come to the front, and new models and algorithms are needed to cope with these special features. Up to six attributes related to the distribution plan are considered in our multi-criteria approach. Even though there are usually simple ways to measure the cost of an operation, the evaluation of some other attributes such as security or equity is not easy. As a result, several attribute measures are proposed and developed, focusing on different aspects of the solutions. Furthermore, when metaheuristic solution methods are used, considering non linear objective functions does not increase the complexity of the algorithms significantly, and thus more accurate measures can be utilized.

In particular, the problem addressed focuses on the development of a *mission*, understood as the distribution of a fixed amount of goods (typically in a short unit of time, such as a day or the time needed for a single shipment). The problem then consists of designing a set of routes for the vehicles available for transportation that connects the depots with a stock of goods to the locations where the population in need demands those goods. In addition to the itineraries followed by the vehicles, which may have different types and costs, the aid transported by each vehicle through each link connecting two nodes must also be determined. We will consider distribution problems that arise normally in highly vulnerable areas, with a sparse and probably damaged road network, and under potentially insecure conditions.

The main elements of the problem are the following:

- Transportation Network. It consists of nodes, which represent the locations of pick-up, delivery or connection, and arcs, characterized by distance and average velocity.
- Humanitarian Aid. Information about the amount of aid available at each depot and required at each demand node must be given.
- Vehicle Fleet. Several vehicle types are considered, characterized by capacity, average velocity, variable and fixed costs and availability at each node of the network. Note that, for each vehicle type, fixed costs are related to the cost of moving the vehicle one unit of length, and thus are independent of the amount of load transported by the vehicle; however, variable costs represent the cost of transporting one unit of aid through one unit of length, and thus they depend on both magnitudes.
- Operation elements. They comprise the amount of aid that is planned to be distributed in the actual operation and the available budget, together with the available information regarding the state of the infrastructure.
    - Itinerary reliability. Due to the possible effects of the disaster, at the moment of designing the distribution plan, the future availability of the network connections may be not known with certainty. Some roads may be blocked by debris, bridges may have fallen down or tunnels may be destroyed. However, that information may be not fully available when planning. Furthermore, some new events related to the disaster may likely happen after the plan is designed, resulting in the unavailability of some of the links. Then, the reliability of the links is part of the problem data.
    - Itinerary security. Under extreme conditions, as the ones arising after a disaster in vulnerable places, the vehicles transporting the aid could be assaulted during their mission. In fact, the use of security to escort the vehicles is imposed in some situations, in order to avoid assaults or terrorist attacks. Also, larger convoys with many vehicles traveling together are usually less likely to be attacked. For these reasons, in some extreme situations (such as the Haiti earthquake (2010), or transports traversing war-zones or under terrorist threat) it is required that

all vehicles traveling through a link (both loaded and unloaded) do it together forming a convoy. This requirement will be assumed in our problem.

As a result, the objective of the problem is to determine the itineraries of vehicles and load through the network so that a certain criterion (or set of criteria, as it will be detailed later in the paper) is optimized and the following set of conditions are satisfied:

- The vehicles start their itineraries from the node where they are parked, which will usually be a depot but it may be a connection or a demand node as well.
- Any node (depot, connection or demand nodes) can be used for transshipment of goods among vehicles. Furthermore, vehicles can travel empty to arrive at certain nodes where aid carried by other vehicles is loaded.
- As described earlier, all vehicles traversing one link must do it together forming a convoy for security reasons.
- Vehicles are allowed to visit one node several times. However, they are not allowed to traverse a link more than once. Note that the requirement of all vehicles traveling together through each link reduces significantly the number of feasible distribution plans. However, the possibility to visit each node several times allows the vehicles to load and unload the humanitarian aid at different moments, facilitating the move of the aid through the network even though the vehicles cannot traverse one link at different times.
- Minimizing the cost of the distribution operation is not the main concern, and as it will be shown later, other attributes will be considered as part of the objective function. However, a certain budget cannot be exceeded.
- Once a vehicle arrives at the final node of its route, it will be scheduled to return to its original location following the least expensive possible path. Besides, no risk of being assaulted will be considered for vehicle returns.

The problem considered in this thesis extends a simplified model already considered in the literature, in order to make it more realistic. The simplified version of the problem is based on a linear goal programming static flow model. Such model is able to provide the optimal flow of vehicles and load through the network, but it does not consider each vehicle's individual itinerary. This means that the model cannot provide decision makers with enough information to easily create implementable plans for the distribution of the aid. This thesis extends the model to allow for a more detailed planning, which explicitly considers the movement of vehicles and load in time and thus is able to provide information that can be easily translated into implementable plans. However, this model that cannot be solved to optimality in reasonable times, and thus, a heuristic approach is needed. Furthermore, due to the high complexity of the resulting problem, even feasible solutions are difficult to obtain, and the development and application of local search procedures able to systematically prevent the violation of the constraints of the problem is highly complex. For these reasons, an elaborate constructive algorithm is developed first, together with

an improving algorithm, and they are later modified and included as the core of two metaheuristics based on a Greedy Randomized Adaptive Search Procedure (GRASP) and an Ant Colony Optimization (ACO) algorithm. These metaheuristics are chosen mainly due to their constructive nature, which facilitates the obtaining of feasible solutions which are iteratively improved by learning from the information gathered in previous iterations.

The thesis is composed by eight chapters, summarized in what follows.

Chapter 1 presents a literature review on mathematical models and methods in humanitarian logistics and metaheuristics for vehicle routing problems similar to the ones used in the thesis.

Chapter 2 describes the problem approached in the thesis, detailing its main elements and providing measures for the considered attributes. A mixed integer linear programming model is proposed and the multicriteria nature of the problem discussed.

Chapter 3 develops a novel constructive heuristic aimed at building feasible solutions of the problem within reduced computational times. In some of the phases of the algorithm randomization is used in order to diversify the construction of the solutions.

Chapter 4 introduces several improving algorithms aimed at improving in different ways the initial solutions built by the constructive algorithm. Dealing properly with the feasibility of the solutions is crucial in these heuristics.

Chapter 5 presents a GRASP metaheuristic based on improving the construction process by guiding it by some greedy functions and an elite set of solutions built during previous iterations.

Chapter 6 is devoted to the development of an Ant Colony Optimization algorithm which introduces specialized ants and pheromone trails to guide the constructions of new solutions.

Chapter 7 is focused on evaluating and analyzing the performance of the proposed algorithms through an extensive computational study performed on several case studies and a variety of randomly generated instances.

Finally, the main conclusions that can be drawn from this work, its main contributions and some interesting future research lines are presented in Chapter 8.

Objectives of the thesis

The main objectives of the thesis are summarized next:

- Formulate and describe a humanitarian aid distribution problem in humanitarian logistics. In particular, we are dealing with a distribution problem in a vulnerable

area, with potentially damaged infrastructures and insecure conditions.

- Study the multicriteria nature of the optimization problems that arise in this context, and in particular in the concrete distribution problem considered.
- Develop heuristic algorithms to provide feasible solutions for real sized instances using little running time.
- Develop appropriate metaheuristics able to guide the construction process in order to obtain improved solutions.
- Calibrate the parameters of the proposed metaheuristics so that they can be applied effectively.
- Analyze in detail the results obtained on a case study already proposed in the literature based on aid distribution operations performed in Haiti after the 2010 earthquake.
- Make an extensive computational study by solving other realistic instances and a set of randomly generated instances with different sizes. Evaluate the performance of the proposed algorithms and comment with detail the results obtained.

Results

The main results of the thesis include the algorithms proposed to solve the considered problem: the Constructive Algorithm, the Improving Algorithm, GRASP and ACO. Brief pseudocodes showing the main structure of these algorithms are provided in what follows.

Algorithm 1 Constructive Algorithm

```
1: Perform preprocess.
2: Design itineraries, form convoys and calculate arriving times at nodes.
3: Deliver aid through the network.
4: if all planned aid is delivered then
5:    Determine arriving and departure events.
6:    Calculate stocks between consecutive events.
7: else
8:    Go back to line 2.
9: end if
10: Try to eliminate all negative stocks at nodes.
11: if there exists any negative stock then
12:    Go back to line 2.
13: end if
14: Output the built solution.
```

The performance of these algorithms has been evaluated in a case study based on the distribution operations performed in Port au Prince after the 2010 Haiti earthquake and in other realistic and randomly generated instances. Some of the most important results derived from this analysis are given next.

Algorithm 2 Improving Algorithm

1: Improve equity.
2: Adjust depots.
3: Trim routes.
4: Replace vehicles.
5: Interchange vehicles.
6: Eliminate aid crossings.
7: Output the improved solution.

Algorithm 3 GRASP algorithm

1: Perform preprocess.
2: **for** $i = 1, \cdots, nelite$ **do**
3: Create from scratch a new solution S with the Constructive+Improving Algorithm.
4: Evaluate the quality of S and its diversity with respect to the current elite set.
5: Update the elite set according to the quality and diversity of S.
6: **end for**
7: Initialize λ and select the best solution S^* from the elite set.
8: **for** $i = 1, \cdots, (tsol - nelite)$ **do**
9: Create a new solution S with the modified version of the Constructive+Improving Algorithm by using the information given by the elite set and the greedy functions.
10: Update the elite set according to the quality and diversity of S.
11: **if** S is better than S^* **then**
12: Do $S^* = S$.
13: **end if**
14: Increase λ: do $\lambda = 1 - \sigma_1 e^{\frac{-i\sigma_2}{tsol - nelite}}$
15: **end for**
16: Output S^*.

Algorithm 4 ACO algorithm

1: Perform preprocess.
2: Initialize τ and λ.
3: Do $cglobal = 1$.
4: **repeat**
5: **for** $clocal = 1, \cdots, m$ **do**
6: Create a new solution S with the modified Constructive+Improving Algorithm by using the information given by the pheromones and the greedy functions.
7: **if** (S is better than S^*) **or** ($cglobal = 1$) **then**
8: Do $S^* = S$.
9: **end if**
10: Perform local update of τ.
11: Do $cglobal = cglobal + 1$.
12: **if** ($cglobal > tsol$) **then**
13: Exit for (go to line 18).
14: **end if**
15: **end for**
16: Perform global update of τ
17: Increase λ: do $\lambda = 1 - \sigma_1 e^{-\sigma_2 \frac{cglobal}{tsol}}$
18: **until** $cglobal > tsol$
19: Output S^*.

Conclusions

The main conclusions that can be drawn from this thesis are summarized in what follows:

- A last mile distribution problem in humanitarian logistics which extends a simplified version from the literature has been formulated and described. Some more realistic conditions have been added to the problem, together with some new improved measures for the considered attributes.
- A mixed integer linear programming model for the approached problem has been proposed.
- Several methodologies to deal with the multicriteria nature of the considered problem have been studied and discussed.
- A novel constructive algorithm able to provide feasible solutions for real sized instances using little running time has been proposed and described in detail.
- An improving algorithm that can be applied to any initial feasible of the problem has been developed. It allows to improve the given initial solution in several ways, while maintaining its feasibility.
- Due to the characteristics of the problem, the development of local search algorithms is highly complex. As a result, two metaheuristics focused on using information from

previous iterations to guide better the construction of the solutions have been developed: a Greedy Randomized Adaptive Search Procedure and an Ant Colony Optimization algorithm.

- The main parameters of the proposed metaheuristics have been calibrated, so that they can be applied effectively. We have observed that some of the parameter values usually proposed in the literature are not appropriate for the problem approached in the thesis, and thus they have been replaced for other values provided by the calibration experiments. The final parameter set that was chosen showed a good performance in a wide variety of instances.

- A deep computational experience has been performed on a case study based on the aid distribution operations performed in Haiti after the 2010 earthquake. The proposed algorithms were shown to be able to provide good solutions for this case study and adapted to the preferences of the decision maker. Furthermore, the degree of conflict among the considered attributes has been analyzed through the pay-off matrix and the representation of several Pareto frontier projections.

- The performance of the proposed algorithms have been evaluated through a computational study performed on other realistic instances and a set of randomly generated instances with different sizes. The results showed that the proposed algorithms are able to solve instances with a realistic size and brought into light the advantages of considering the extended version of the problem and using the improved non-linear attribute measures. The GRASP algorithm showed the best performance on large instances. However, the ACO algorithm, even though is usually slower than the others, provided the best solutions in several of the considered instances.

Apéndices

Apéndice A

Procedimientos modificados

En este apéndice se presentan los procedimientos detallados del Algoritmo GRASP correspondientes a modificaciones de procedimientos ya descritos.

A.1. CONSTRUIR RUTAS 2

Modificación del procedimiento CONSTRUIR RUTAS en el que se introducen las funciones *greedy* e intensidad para guiar la construcción de los itinerarios.

Variables de entrada

λ, *cont*, *conv*, *lonv*, *nump*, *prec*, *pred*, *prei*, *pret*, *secv*, *suci*, *suct*, *tamc*, *tamclas*, *tcpar*, *util*

Variables de salida

cont, *conv*, *lonv*, *nump*, *prec*, *pred*, *prei*, *pret*, *secv*, *suci*, *suct*, *tamc*, *tcpar*

Variables internas

coti	Valor mínimo que debe tener la intensidad máxima entre los movimientos para poder seguir prolongando las rutas
$gr_{g,mov}$	Valor de la función *greedy* para el objetivo g aplicada al movimiento *mov*
$greedy_{mov}$	Valor de la función *greedy* agregada aplicada al movimiento *mov*
int_{mov}	Valor de la función intensidad aplicada al movimiento *mov*
$kpon_{mov}$	Ponderación dada al movimiento *mov*
mov, mov'	Índices para los movimientos posibles
$prob_{mov}$	Probabilidad de elegir el movimiento *mov* para incorporarlo a la solución en construcción

act, *agan*, *aper*, *apos*, *apre*, *asor*, *vgan*, *vsor*

Procedimientos llamados

AGREGAR PRECEDENCIAS

Algoritmo

PASO 0

- Si se continúan prolongando rutas ya construidas, $cont = 1$, ir al PASO 1 (identificar movimientos)
- (Se empiezan a construir nuevas rutas, $cont = 0$). Inicializar las variables de almacenamiento de la solución y de las relaciones de precedencia asignando el valor 0 a las siguientes variables, vectores y matrices

$$lonv, nump, prec, prei, tamc$$

- Indicar que se está en proceso de construcción de rutas

$$cont = 1$$

- Situar los vehículos en sus nodos de origen

$$act_j = ori_j \quad \forall j$$

- Cualquier trayecto útil de un vehículo j por un arco k está permitido

$$aper_{j,k} = util_{j,k} \quad \forall j, \forall k$$

- Inicializar el número de vehículos de cada clase que circulan por los arcos

$$tcpar_{c,k} = 0$$

- Inicializar la cota para la intensidad máxima

$$coti = \frac{nelite}{2} - 1$$

PASO 1

- Identificar todos los movimientos posibles. Cada movimiento mov corresponde a un vehículo j y un arco transitable k desde su ubicación actual act_j
- Si no hay ningún movimiento posible:
  - Indicar que no se está en proceso de construcción de rutas

$$cont = 0$$

  - FIN del procedimiento

- Obtener la intensidad de cada movimiento: número de soluciones élite para las que el número de vehículos de la clase del actual que circulan por el arco es mayor que en la solución que se está construyendo

$$int_{mov} = |o \in O / tamclas_{c,k,o} > tcpar_{c,k}|$$
$$\text{siendo } c = clas_{tipv_j, ori_j} \quad \forall mov$$

- Si la intensidad máxima es menor que la cota, $\max_{mov}\{int_{mov}\} < coti$,
  - Disminuir la cota para que, en caso de que sea necesario prolongar las rutas, existan más opciones para hacerlo

$$coti = coti - 1$$

  - FIN del procedimiento
- (La intensidad máxima no es menor que la cota, $\max_{mov}\{int_{mov}\} \geq coti$). Obtener la función *greedy* de cada movimiento (solo si el peso total es positivo, $pesotot > 0$):
  - Función *greedy* para el tiempo. Tiempo que tarda el vehículo j en recorrer el arco k

$$gr_{1,mov} = \frac{dist_k}{\min\{vela_k, velv_{tipv_j}\}} \quad \forall mov$$

  - Función *greedy* para el coste. Coste que le supone al vehículo j recorrer el arco k más la diferencia entre el coste de regreso desde el final del arco y el coste de regreso desde el inicio del arco

$$gr_{2,mov} = dist_k \cdot cosf_{tipv_j,k} + creg_{c,fin_k} - creg_{c,ini_k}$$
$$\text{siendo } c = clas_{tipv_j, ori_j} \quad \forall mov$$

  - Función *greedy* para la seguridad. Probabilidad de que el convoy que recorre el arco k, incluyendo al vehículo j, sufra un asalto

$$gr_{5,mov} = FP(p_k, pm_k, tamc_k + 1) \quad \forall mov$$

  - Función *greedy* para la fiabilidad. Probabilidad de que el arco k no esté transitable

$$gr_{6,mov} = 1 - r_k \quad \forall mov$$

  - La función *greedy* global del movimiento es la media ponderada de las distancias normalizadas de las funciones *greedy* a sus cotas en los criterios considerados

$$greedy_{mov} = \frac{1}{pesotot} \sum_{g \in \{1,2,5,6\}} \left(\frac{peso_g(cru_g - gr_{g,mov})}{cru_g} \right) \quad \forall mov$$

- Calcular la ponderación de cada movimiento

$$kpon_{mov} = \lambda \frac{int_{mov}}{nelite} + (1 - \lambda)(\gamma \cdot greedy_{mov} + 1 - \gamma) \quad \forall mov$$

- Calcular la probabilidad de elegir cada movimiento

$$prob_{mov} = \frac{kpon_{mov}}{\sum_{mov'} kpon_{mov'}} \quad \forall mov$$

- Elegir al azar un movimiento considerando las probabilidades obtenidas. Sean $vgan$ y $agan$ el vehículo y el arco seleccionados, respectivamente (los correspondientes al movimiento elegido)

PASO 2

- Aumentar en una unidad el tamaño de la ruta del vehículo seleccionado $vgan$ y añadir el arco $agan$ a dicha ruta

$$lonv_{vgan} = lonv_{vgan} + 1$$
$$secv_{vgan,lonv_{vgan}} = agan$$

- Aumentar en una unidad el tamaño del convoy del arco seleccionado y añadir el vehículo a dicho convoy

$$tamc_{agan} = tamc_{agan} + 1$$
$$conv_{agan,tamc_{agan}} = vgan$$

- Ubicar el vehículo seleccionado en el nodo final del arco seleccionado

$$act_{vgan} = fin_{agan}$$

- Actualizar el número de vehículos de la clase del vehículo seleccionado que circulan por el arco seleccionado

$$tcpar_{c,agan} = tcpar_{c,agan} + 1 \quad \text{siendo } c = clas_{tipv_{vgan},ori_{vgan}}$$

- Si el arco seleccionado es el primero de la ruta del vehículo seleccionado, $lonv_{vgan} = 1$, ir al PASO 3 (prohibición de movimientos futuros por violar las precedencias)
- (El arco seleccionado no es el primero de la ruta del vehículo seleccionado, $lonv_{vgan} > 1$). Guardar el arco anterior al arco seleccionado en la ruta del vehículo seleccionado como candidato a preceder inmediatamente al arco seleccionado

$$apre = secv_{vgan,lonv_{vgan}-1}$$

- Si el arco seleccionado no estaba ya precedido inmediatamente por el arco anterior en la ruta del vehículo seleccionado, $prei_{apre,agan} = 0$, hacer:
  - Guardar el arco seleccionado como precedido

$$apos = agan$$

  - Ejecutar el procedimiento AGREGAR PRECEDENCIAS, para obligar a que el arco $apre$ preceda inmediatamente al arco $apos$ y actualizar las precedencias correspondientes

PASO 3

- Impedir que el vehículo seleccionado recorra en el futuro el arco seleccionado

$$aper_{vgan,agan} = 0$$

- Impedir que el vehículo seleccionado recorra en el futuro los arcos predecesores del arco seleccionado

$$aper_{vgan,k} = 0 \quad \forall k \in pret_{agan}$$

- Volver al PASO 1

A.2. ENVIAR FLUJO 2

Modificación del procedimiento ENVIAR FLUJO en el que se introducen las funciones *greedy* e intensidad para guiar el envío de material.

Variables de entrada

λ, $cont$, $conv$, $eltar$, esf, $prec$, $repc$, $tamc$, $tmin$

Variables de salida

$cont$, $ltar$, $repc$

Variables internas

cfl_1	Cota para la función *greedy* del tiempo
$gr_{g,i}$	Valor de la función *greedy* para el objetivo g aplicada al nodo i
$greedy_i$	Valor de la función *greedy* agregada aplicada al nodo i
int_i	Valor de la función intensidad aplicada al nodo i
$kpon_i$	Ponderación dada al nodo i
$lcan$	Lista con los nodos candidatos a etiquetar
$prob_i$	Probabilidad de elegir el nodo i para etiquetar

$aant$, $aumf$, $capr$, $capt$, $delta$, $etiq$, $incp$, $intff$, $inut$, $itet$, $flujo$, $nant$, $ngan$, $npar$, $valf$

Algoritmo

PASO 0

- Indicar que no se ha repartido lo planeado

$$repc = 0$$

- Inicializar las cantidades de material que transportan los convoyes

$$ltar_k = 0 \quad \forall k$$

- Calcular las capacidades totales de los arcos que unen la fuente, f, con los depósitos

$$capt_{f,i} = \text{mín}\{qglobal, qav_i\} \quad \forall i \in IO$$

- Calcular las capacidades totales de los arcos que unen los nodos de demanda con el sumidero, s

$$capt_{i,s} = \text{mín}\{qglobal, dem_i\} \quad \forall i \in IO$$

- Calcular las capacidades totales de los arcos transitados

$$capt_{i,i'} = \text{mín}\left\{qglobal, \sum_{j \in conv_k} cap_{tipv_j}\right\} \quad \forall k \ / \ conv_k \neq \varnothing$$
$$\text{siendo } i = ini_k, i' = fin_k$$

- Hacer inútiles los nodos (no de demanda) desde los que no puede salir flujo. No es necesario, pero puede ahorrar operaciones

$$\forall i \in I - (ID) \ / \ \sum_{i'} capt_{i,i'} = 0 \quad \Rightarrow \quad inut_i = 1$$

- Inicializar el número de intentos realizados de aplicación completa del algoritmo

$$intff = 1$$

- Calcular la cota para la función *greedy* del tiempo: instante más tardío en el que un convoy termina de recorrer un arco

$$cfl_1 = \underset{k}{\text{máx}}\{tmin_k\}$$

- Calcular, para cada arco k recorrido, el coste variable medio considerando los vehículos del convoy correspondiente

$$cosvm_k = \frac{1}{tamc_k} \sum_{j \in conv_k} cosv_{tipv_j,k} \quad \forall k \ / \ tamc_k > 0$$

PASO 1

- Inicializar el flujo entre cada par de nodos i e i' (incluyendo la fuente y el sumidero) y el valor del flujo

$$flujo_{i,i'} = 0 \quad \forall i, i' \in I \cup \{f, s\}$$
$$valf = 0$$

- Inicializar las capacidades residuales. Deben coincidir con las capacidades totales

$$capr_{i,i'} = capt_{i,i'} \quad \forall i, i' \in I \cup \{f, s\}$$

- Inicializar el número de intentos realizados en la búsqueda de un camino de aumento

$$itet = 1$$

- Indicar que no hay problemas de precedencias

$$incp = 0$$

PASO 2

- Borrar etiquetas, reiniciar variables auxiliares

$$etiq_i = 0, delta_i = qglobal, nant_i = f \quad \forall i \in I \cup \{f, s\}$$

- Etiquetar la fuente y tomarla como nodo de referencia

$$etiq_f = 1, npar = f$$

PASO 3

- Identificar todos los nodos candidatos a etiquetar desde el nodo de referencia, $npar$, e incluirlos en la lista $lcan$. Cada nodo candidato i lleva aparejado un arco asociado, $k = (npar, i)$, que lo une con el nodo de referencia. Para que un nodo i sea candidato debe cumplir:
  a. No está etiquetado, $etiq_i = 0$
  b. Su arco asociado tiene capacidad residual positiva, $capr_{npar,i} > 0$
  c. No es un nodo inútil, $inut_i = 0$
  d. Su arco asociado no precede al arco por el que se accedió al nodo de referencia, $prec_{k,aant_{npar}} = 0$. Esta condición no se comprueba cuando el nodo de referencia es la fuente, $npar = f$, o cuando el presumible candidato es el sumidero, $i = s$

  Si se cumplen las condiciones a, b c, pero no la condición d, indicar que hay problemas de precedencia

$$incp = 1$$

- Para cada nodo candidato hacer:
  - Si el nodo de referencia es la fuente, $npar = f$, cada nodo candidato a etiquetar corresponde a un depósito i:
    - Obtener la intensidad de cada candidato i: número de soluciones élite para las que el material neto que sale del depósito es mayor que el flujo desde la fuente al depósito en la solución que se está construyendo

$$int_i = |o \in O / qav_i - esf_{i,o} > flujo_{f,i}|$$

- Obtener la ponderación de cada candidato i. En este caso no se utilizan funciones *greedy*:

$$kpon_i = \lambda \frac{int_i}{nelite} + (1 - \lambda)$$

- (El nodo de referencia no es la fuente, $npar \neq f$). Cada nodo i candidato a etiquetar corresponde a un arco k que sale del nodo $npar$, último nodo etiquetado
  - Obtener la intensidad de cada candidato i: número de soluciones élite para las que el material que circula por el arco es mayor que el flujo por el arco en la solución que se está construyendo

$$int_i = |o \in O / eltar_{k,o} > flujo_{npar,fin_k}|$$

  - Obtener la función *greedy* de cada cada candidato i:
    - Función *greedy* para el tiempo. Instante en el que el convoy termina de recorrer el arco k

$$gr_{1,i} = tmin_k$$

    - Función *greedy* para el coste. Coste medio que supone transportar una unidad de material por el arco k

$$gr_{2,i} = dist_k \cdot cosvm_k$$

    - Funciones *greedy* para la equidad y la prioridad

$$gr_{3,i} = 0$$
$$gr_{4,i} = 1$$

    - Función *greedy* para la seguridad. Probabilidad de que el convoy que recorre el arco k sufra un asalto

$$gr_{5,i} = FP(p_k, pm_k, tamc_k)$$

    - Función *greedy* para la fiabilidad. Probabilidad de que el arco k no esté transitable

$$gr_{6,i} = 1 - r_k$$

    - La función *greedy* global del candidato i es la media ponderada de las distancias normalizadas de las funciones *greedy* a sus cotas

$$greedy_i = \sum_g \left(\frac{peso_g(cfl_g - gr_{g,i})}{cfl_g} \right)$$

  - Obtener la ponderación de cada cada candidato i

$$kpon_i = \lambda \frac{int_i}{nelite} + (1 - \lambda)(\gamma \cdot greedy_i + 1 - \gamma)$$

- Si el nodo de referencia es un nodo de demanda, $dem_{npar} > 0$, además se considera el movimiento correspondiente al sumidero, s

- Obtener la intensidad del sumidero: número de soluciones élite para las que el material que termina en el nodo de demanda es mayor que el flujo desde el nodo de demanda al sumidero en la solución que se está construyendo

$$int_s = |o \in O / esf_{npar,o} > flujo_{npar,s}|$$

- Obtener la función *greedy* de cada movimiento
  - Funciones *greedy* para el tiempo y el coste

$$gr_{1,s} = 0$$
$$gr_{2,s} = 0$$

  - Función *greedy* para la equidad. Diferencia entre la distancia a la proporción de demanda satisfecha óptima eligiendo este movimiento y sin elegirlo

$$gr_{3,s} = \left| pjus - \frac{flujo_{npar,s} + \text{mín}\{delta_{npar}, capr_{npar,s}\}}{dem_i} \right| - \left| pjus - \frac{flujo_{npar,s}}{dem_{npar}} \right|$$

  - Función *greedy* para la prioridad. Complementario a 1 de la contribución del movimiento a la proporción de demanda satisfecha del nodo multiplicada por su nivel de prioridad

$$gr_{4,s} = 1 - \frac{\text{mín}\{delta_{npar}, capr_{npar,s}\} \cdot nivp_{npar}}{dem_{npar}}$$

  - Funciones *greedy* para la seguridad y la fiabilidad

$$gr_{5,s} = 0$$
$$gr_{6,s} = 0$$

  - La función *greedy* global asociada al sumidero es la media ponderada de las distancias normalizadas de las funciones *greedy* a sus cotas

$$greedy_s = \sum_g \left(\frac{peso_g(cfl_g - gr_{g,s})}{cfl_g} \right)$$

- Obtener la ponderación del sumidero

$$kpon_s = \lambda \frac{int_s}{nelite} + (1 - \lambda)(\gamma \cdot greedy_s + 1 - \gamma)$$

- Obtener la probabilidad de elegir cada nodo i candidato

$$prob_i = \frac{kpon_i}{\sum_{i' \in lcan} kpon_{i'}} \quad \forall i \in lcan$$

PASO 4

- Si hay candidatos, $lcan \neq \varnothing$
  - Elegir al azar un nodo considerando las probabilidades obtenidas. Sea $ngan$ el nodo elegido. Etiquetarlo, calcular su máximo aumento permitido de flujo y fijar su nodo anterior y su arco anterior

$$\begin{gathered} etiq_{ngan} = 1 \\ delta_{ngan} = \text{mín}\{delta_{npar}, capr_{npar,ngan}\} \\ nant_{ngan} = npar \\ aant_{ngan} = k \quad \text{siendo } npar = ini_k, ngan = fin_k \end{gathered}$$

  - Tomar el nodo seleccionado como nodo de referencia

$$npar = ngan$$

  - Si el nodo seleccionado es el sumidero, $ngan = s$, se ha encontrado un camino de aumento. Ir al PASO 5 (actualizar flujo)
  - (El nodo seleccionado no es el sumidero, $ngan \neq s$). Ir al PASO 3 (búsqueda de candidatos para etiquetar)
- (No hay candidatos)
  - Si el nodo de referencia no es la fuente, $npar \neq f$:
    - Tomar como nuevo nodo de referencia el nodo anterior al actual nodo de referencia

$$npar = npar_{ngan}$$

    - Ir al PASO 3 (búsqueda de candidatos para etiquetar)
  - (El nodo de referencia es la fuente, $npar = f$). Si no hay problemas de precedencia, $prec = 0$, FIN del procedimiento
  - (Hay problemas de precedencia). Si se han agotado los intentos para encontrar un camino de aumento, $itet = maxet$:
    - Si se han agotado los intentos para aplicar el algoritmo completo, $intff = maxff$, FIN del procedimiento
    - Actualizar el número de intentos de aplicación del algoritmo completo

$$intff = intff + 1$$

    - Volver al PASO 1 (aplicación del algoritmo desde el principio)
  - (No se han agotado los intentos para encontrar un camino de aumento, $itet \neq maxet$). Actualizar el número de intentos para encontrar un camino de aumento

$$itet = itet + 1$$

  - Volver al PASO 2 (búsqueda de un nuevo camino de aumento)

PASO 5

- Calcular la cantidad en la que se va a aumentar el flujo, $aumf$. Se tiene en cuenta que el valor del flujo, $valf$, no debe superar la cantidad a repartir, $qglobal$

$$aumf = \text{mín}\{delta_s, qglobal - valf\}$$

- Aumentar el flujo y disminuir la capacidad residual en los arcos del camino de aumento. Hacer hasta que el nodo de referencia sea la fuente, $npar = f$:

$$\begin{gathered} i = nant_{npar}, i' = npar \\ flujo_{i,i'} = flujo_{i,i'} + aumf \\ capr_{i,i'} = capr_{i,i'} - aumf \\ npar = nant_{npar} \end{gathered}$$

- Actualizar el valor del flujo añadiéndole la cantidad de aumento de flujo

$$valf = valf + aumf$$

- Si se ha repartido lo planeado, $valf = qglobal$:
  - Calcular las cargas de los convoyes por los arcos

$$ltar_k = flujo_{i,i'} \quad \text{siendo } i = ini_k, i' = fin_k \quad \forall k$$

  - Indicar que se ha repartido lo planeado

$$repc = 1$$

  - FIN del procedimiento
- (No se ha repartido lo planeado, $valf \neq qglobal$). Volver al PASO 2 (búsqueda de un nuevo camino de aumento)

A.3. CONSTRUCTIVO 2

Modificación del procedimiento CONSTRUCTIVO, programa principal del Algoritmo Constructivo descrito en el Capítulo 3, en el que se introducen las funciones *greedy* e intensidad en algunos de los procedimientos empleados en la construcción de la solución, concretamente, en los procedimientos CONSTRUIR RUTAS y ENVIAR FLUJO.

Variables de entrada

$eltar$, esf, $tamclas$, $util$

Variables de salida

λ, $cont$, $conv$, dur, $into$, $lonv$, $ltar$, $nuan$, $nump$, $prec$, $pred$, $prei$, $pret$, $secv$, sf, $stck$, $suci$, $suct$, $tamc$, $tcpar$, $tmin$

Variables internas
inst, *load*, *negat*, *repc*

Procedimientos llamados

CONSTRUIR RUTAS 2, DURACIÓN, ENVIAR FLUJO 2, FIN ARCOS, ORDENAR EVENTOS, POSITIVAR, REPARTIR CARGAS

Algoritmo

PASO 0

- Indicar que todavía no se han evitado cruces de material

$$evcr = 0$$

PASO 1

- Indicar que todavía no se ha empezado a construir rutas

$$cont = 0$$

- Ejecutar el procedimiento CONSTRUIR RUTAS 2, para construir las rutas de los vehículos teniendo en cuenta las funciones *greedy* e intensidad
- Ejecutar el procedimiento DURACIÓN, para calcular las duraciones de los trayectos de los convoyes
- Ejecutar el procedimiento FIN ARCOS, para calcular los instantes mínimos de finalización de los trayectos
- Ejecutar el procedimiento ENVIAR FLUJO 2, para obtener las cantidades de material transportadas por los convoyes teniendo en cuenta las funciones *greedy* e intensidad
- Si no se ha repartido lo planeado, $repc = 0$, volver al PASO 1 (construir nuevas rutas o prolongar las existentes)
- (Se ha repartido lo planeado, $repc = 1$). Indicar que ya no se están construyendo rutas

$$cont = 0$$

- Ejecutar el procedimiento ORDENAR EVENTOS, para ordenar los eventos en los nodos según sus instantes de ocurrencia y calcular los *stocks* correspondientes

PASO 2

- Ejecutar el procedimiento POSITIVAR, para detectar y eliminar los posibles *stocks* negativos
- Si hay algún *stock* negativo, $negat = 1$, volver al PASO 1 (construir nuevas rutas)

A.4. RETENER MATERIAL 2

Modificación del procedimiento RETENER MATERIAL en el que se introducen las funciones intensidad.

Variables de entrada

λ, $eltar$, esf, $into$, $ltar$, $nuan$, sf, $stock$

Variables de salida

$ltar$

Variables internas

int_i	Valor de la función intensidad correspondiente al nodo i
$kpon_i$	Ponderación dada al nodo i
$prob_i$	Probabilidad de elegir el nodo i para etiquetar

$aumf$, $capr$, $capt$, $delmx$, $delta$, $etiq$, $flujo$, $flumx$, $itet$, $lcan$, $lnec$, $lsob$, $nant$, $nnec$, $npar$, $nuevo$, $pos1$, $pos2$, ult, $valf$

Algoritmo

PASO 0

- Crear la lista de nodos necesitados en los que se puede retener material, $lnec$

$$i \in lnec \quad \Leftrightarrow \quad \frac{sf_i}{dem_i} < pjus, \sum_{k/ini_k=i} ltar_k > 0$$

- Si no hay ningún nodo con estas características, FIN del procedimiento
- Calcular las capacidades totales de los arcos que unen la fuente, f, con los nodos necesitados

$$capt_{f,i} = \text{mín} \left\{ \sum_{k/ini_k=i} ltar_k, dem_i - sf_i \right\} \quad \forall i \in lnec$$

- Crear la lista de nodos no necesitados de los que se puede quitar material, $lsob$

$$i \in lsob \quad \Leftrightarrow \quad \frac{sf_i}{dem_i} > pjus, \sum_{k/fin_k=i} ltar_k > 0$$

- Calcular las capacidades totales de los arcos que unen los nodos no necesitados con el sumidero, s

$$capt_{i,s} = sf_i - dem_i \cdot pjus \quad \forall i \in lsob$$

- Calcular las capacidades totales de los arcos transitados

$$capt_{i,i'} = ltar_k \quad \forall k \ / \ conv_k \neq \varnothing \quad \text{siendo } i = ini_k, i' = fin_k$$

PASO 1

- Inicializar el flujo entre cada par de nodos i e i' (incluyendo la fuente y el sumidero) y el valor del flujo

$$flujo_{i,i'} = 0 \quad \forall i, i' \in I \cup \{f, s\}$$
$$valf = 0$$

- Inicializar las capacidades residuales. Deben coincidir con las capacidades totales

$$capr_{i,i'} = capt_{i,i'} \quad \forall i, i' \in I \cup \{f, s\}$$

- Inicializar el número de intentos realizados en la búsqueda de un camino de aumento

$$itet = 1$$

PASO 2

- Borrar etiquetas, reiniciar variables auxiliares

$$etiq_i = 0, delta_i = qglobal, nant_i = f \quad \forall i \in I \cup \{f, s\}$$

- Etiquetar la fuente y tomarla como nodo de referencia

$$etiq_f = 1, npar = f$$

- Indicar que el nodo de referencia es nuevo en la presente iteración

$$nuevo = 1$$

- Inicializar las cotas superiores para el aumento de flujo en la iteración actual

$$flumx_{i,i'} = capr_{i,i'} \quad \forall i, i' \in I \cup \{f, s\}$$

PASO 3

- Si el nodo de referencia es la fuente, $npar = f$, o el nodo anterior al de referencia es la fuente, $nant_{npar} = f$:
  - Crear la lista de nodos candidatos a etiquetar, $lcan$, revisando los nodos alcanzables desde el nodo de referencia. Para que un nodo i sea candidato debe cumplir:
    a. No está etiquetado: $etiq_i = 0$
    b. Su arco asociado tiene capacidad residual positiva, $capr_{npar,i} > 0$
  - Ir al PASO 4 (ponderación de candidatos)

- (Ni el nodo de referencia ni su anterior son la fuente, $npar \neq f$, $nant_{npar} \neq f$). Si el nodo de referencia es nuevo para esta iteración, $nuevo = 1$, fijar la cota superior para las eventuales etiquetas de los nodos alcanzables desde el nodo de referencia como la etiqueta del nodo de referencia

$$delmx = del_{npar}$$

- Revisar todos los arcos usados, k, que salen del nodo de referencia, $npar$, a partir del evento correspondiente al arco desde el que se etiquetó el nodo de referencia, $pos1_{npar}$, y por orden de ocurrencia. Cada arco k lleva asociado un nodo $i = fin_k$ y un evento e tal que $k = into_{npar,e}$:
  - Añadir el nodo i a la lista de candidatos, $lcan$, si cumple:
    a. No está etiquetado: $etiq_i = 0$
    b. El arco tiene capacidad residual positiva : $capr_{npar,i} > 0$
  - Si el nodo de referencia es nuevo para esta iteración, $nuevo = 1$, y el nodo i ha sido incluido en la lista de candidatos, $i \in lcan$, actualizar el aumento máximo de flujo entre los nodos $npar$ e i

$$flumx_{npar,i} = \text{mín}\{delmx, flumx_{npar,i}\}$$

  - Si no queda *stock* tras el evento, $stck_{npar,e} = 0$, ir al PASO 4 (ponderación de candidatos)
  - (Queda *stock* tras el evento, $stck_{npar,e} > 0$). Si el nodo de referencia es nuevo para esta iteración, $nuevo = 1$, actualizar la cota para las etiquetas

$$delmx = \text{mín}\{delmx, stck_{npar,e}\}$$

- Añadir el sumidero, s, a la lista de candidatos a etiquetar si la capacidad residual del arco ficticio correspondiente es positiva, $capr_{npar,s} > 0$.
- Si el nodo de referencia es nuevo para esta iteración, $nuevo = 1$, y el sumidero, s, ha sido incluido en la lista de candidatos, $s \in lcan$, actualizar el aumento máximo de flujo entre los nodos $npar$ y s

$$flumx_{npar,s} = \text{mín}\{delmx, flumx_{npar,s}\}$$

PASO 4

- Si el nodo de referencia es la fuente, $npar = f$, cada candidato i es un nodo de demanda necesitado. Obtener su intensidad como el número de soluciones élite para las que el material que termina en el nodo de demanda necesitado es mayor que en la solución que se está construyendo

$$int_i = |o \in O / esf_{i,o} > sf_i|$$

- Si el nodo de referencia no es la fuente, $npar \neq f$, cada nodo candidato i corresponde al final de un arco k que parte del nodo $npar$. Obtener su intensidad como el número de soluciones élite para las que el material que circula por el arco es menor que la diferencia entre el material que circula por el arco y el flujo correspondiente en la solución que se está construyendo

$$int_i = |o \in O / eltar_{k,o} < ltar_k - flujo_{npar,i}|$$

- Si el sumidero es candidato, su intensidad es el número de soluciones élite para las que el material que termina en el nodo de demanda no necesitado de referencia $npar$ es menor que en la solución que se está construyendo

$$int_s = |o \in O / esf_{npar,o} < sf_{npar}|$$

- Obtener la ponderación de cada candidato i

$$kpon_i = \lambda \frac{int_i}{nelite} + (1 - \lambda)$$

- Obtener la probabilidad de elegir cada nodo i candidato

$$prob_i = \frac{kpon_i}{\sum_{i' \in lcan} kpon_{i'}} \quad \forall i \in lcan$$

PASO 5

- Si hay candidatos, $lcan \neq \varnothing$:
  - Seleccionar al azar un nodo, $ngan$, de la lista de candidatos, $lcan$ de acuerdo a las probabilidades correspondientes, etiquetarlo, calcular su máximo aumento permitido de flujo y fijar su nodo anterior

$$\begin{gathered} etiq_{ngan} = 1 \\ delta_{ngan} = \text{mín}\{delta_{npar}, flumx_{npar,ngan}\} \\ nant_{ngan} = npar \end{gathered}$$

  - Si el nodo seleccionado es el sumidero, $ngan = s$, se ha encontrado un camino de aumento. Ir al PASO 6 (preparación para actualizar flujo)
  - (El nodo seleccionado no es el sumidero, $ngan \neq s$). Si el nodo de referencia no es la fuente, $npar \neq f$, guardar las posiciones que ocupa el arco $k = (npar, ngan)$ en las listas de eventos de los nodos que une dicho arco

$$\begin{gathered} pos2_{npar} = e, \text{ siendo } k = into_{npar,e} \\ pos1_{ngan} = e', \text{ siendo } k = into_{ngan,e'} \end{gathered}$$

  - Tomar el nodo seleccionado como nodo de referencia

$$npar = ngan$$

  - Ir al PASO 3 (búsqueda de candidatos para etiquetar)

- (No hay candidatos, $lcan = \varnothing$):
  - Si el nodo de referencia no es la fuente, $npar \neq f$:
    - Tomar como nuevo nodo de referencia el nodo anterior al actual nodo de referencia
    $$npar = npar_{ngan}$$
    - Indicar que el nuevo nodo de referencia ya ha sido anteriormente nodo de referencia en esta iteración
    $$nuevo = 0$$
    - Ir al PASO 3 (búsqueda de candidatos para etiquetar)
  - (El nodo de referencia es la fuente, $npar = f$). Si se han agotado los intentos para encontrar un camino de aumento, $itet = maxet$:
    - Actualizar las cargas de los convoyes por los arcos, restando los flujos obtenidos correspondientes
    $$ltar_k = ltar_k - flujo_{i,i'} \quad \text{siendo } i = ini_k, i' = fin_k \quad \forall k$$
    - FIN del procedimiento
  - (No se han agotado los intentos para encontrar un camino de aumento, $itet \neq maxet$):
    - Actualizar el número de intentos para encontrar un camino de aumento
    $$itet = itet + 1$$
    - Volver al PASO 2 (búsqueda de un nuevo camino de aumento)

PASO 6

- Indicar que el nodo de referencia actual es el último del camino de aumento (exceptuando el sumidero)
$$ult = npar$$
- Tomar el sumidero como nodo de referencia
$$npar = s$$
- Calcular la cantidad en la que se va a aumentar el flujo, $aumf$
$$aumf = delta_s$$

PASO 7

- Actualizar el flujo y la capacidad residual en el arco $(nant_{npar}, npar)$
$$\begin{array}{l} flujo_{i,i'} = flujo_{i,i'} + aumf \\ capr_{i,i'} = capr_{i,i'} - aumf \end{array} \quad \text{siendo } i = nant_{npar}, i' = npar$$

- Si el nodo de referencia es el último nodo del camino de aumento (sin contar el sumidero), $npar = ult$, actualizar los *stocks* correspondientes a los eventos posteriores a la llegada del arco utilizado para etiquetar

$$stck_{npar,e} = stck_{npar,e} - aumf \quad \forall e \geq pos1_{npar}$$

- Si el nodo de referencia es un nodo intermedio del camino de aumento, $npar \neq ult$, $npar \neq s$, $nant_{npar} \neq f$:
  - Actualizar los *stocks* correspondientes a los eventos comprendidos entre la llegada y salida de los arcos utilizados para etiquetar

$$stck_{npar,e} = stck_{npar,e} - aumf \quad \forall e \in \{pos1_{npar}, \ldots, pos2_{npar} - 1\}$$

  - Si el nodo de referencia es un nodo necesitado, $npar \in lnec$, actualizar la capacidad residual entre la fuente y dicho nodo

$$capr_{f,npar} = \text{máx}\{0, capr_{f,npar} - aumf\}$$

- Si el nodo de referencia es el primer nodo en el camino de aumento (exceptuando la fuente), $nant_{npar} = f$:
  - Actualizar los *stocks* correspondientes a los eventos posteriores a la salida del arco utilizado para etiquetar

$$stck_{npar,e} = stck_{npar,e} + aumf \quad \forall e \geq pos2_{npar}$$

  - Actualizar el valor del flujo añadiéndole la cantidad de aumento de flujo

$$valf = valf + aumf$$

  - Volver al PASO 2 (búsqueda de un nuevo camino de aumento)
- (El nodo de referencia no es el primer nodo del camino de aumento, $nant_{npar} \neq f$). Tomar el nodo anterior en el camino de aumento como nodo de referencia

$$npar = nant_{npar}$$

- Volver al PASO 7 (actualización del flujo)

A.5. APROVECHAR DEPÓSITOS 2

Modificación del procedimiento APROVECHAR DEPÓSITOS en el que se introducen las funciones intensidad.

Variables de entrada

λ, $eltar$, esf, $into$, $ltar$, $nuan$, sf, $stock$

Variables de salida

$ltar$

Variables internas

int_i	Valor de la función intensidad correspondiente al nodo i
$kpon_i$	Ponderación dada al nodo i
$prob_i$	Probabilidad de elegir el nodo i para etiquetar

$aumf$, $capr$, $capt$, $delmx$, $delta$, $entra$, $etiq$, $flujo$, $flumx$, $itet$, $lcan$, $nant$, $npar$, $nuevo$, $pos1$, $pos2$, ult, $valf$

Algoritmo

PASO 0

- Calcular las cantidades de material que entran en los depósitos

$$entra_i = \sum_{k/fin_k=i} ltar_k \quad \forall i \ / \ qav_i > 0$$

- Calcular las capacidades totales de los arcos que unen la fuente, f, con los depósitos

$$capt_{f,i} = \text{mín}\{entra_i, sf_i\} \quad \forall i \ / \ qav_i > 0$$

- Calcular las capacidades totales de los arcos que unen los depósitos con el sumidero, s

$$capt_{i,s} = qav_i \quad \forall i \ / \ qav_i > 0$$

- Calcular las capacidades totales de los arcos transitados

$$capt_{i,i'} = ltar_k \quad \forall k \ / \ conv_k \neq \varnothing \quad \text{siendo } i = fin_k, i' = ini_k$$

PASO 1

- Inicializar el flujo entre cada par de nodos i e i' (incluyendo la fuente y el sumidero) y el valor del flujo

$$flujo_{i,i'} = 0 \quad \forall i, i' \in I \cup \{f, s\}$$
$$valf = 0$$

- Inicializar las capacidades residuales. Deben coincidir con las capacidades totales

$$capr_{i,i'} = capt_{i,i'} \quad \forall i, i' \in I \cup \{f, s\}$$

- Inicializar el número de intentos realizados en la búsqueda de un camino de aumento

$$itet = 1$$

PASO 2

- Borrar etiquetas, reiniciar variables auxiliares

$$etiq_i = 0, delta_i = qglobal, nant_i = f \quad \forall i \in I \cup \{f, s\}$$

- Etiquetar la fuente y tomarla como nodo de referencia

$$etiq_f = 1, npar = f$$

- Indicar que el nodo de referencia es nuevo en la presente iteración

$$nuevo = 1$$

- Inicializar las cotas superiores para el aumento de flujo en la iteración actual

$$flumx_{i,i'} = capr_{i,i'} \quad \forall i, i' \in I \cup \{f, s\}$$

PASO 3

- Si el nodo de referencia es la fuente, $npar = f$, o el nodo anterior al de referencia es la fuente, $nant_{npar} = f$:
  - Crear la lista de nodos candidatos a etiquetar, $lcan$, revisando los nodos alcanzables desde el nodo de referencia. Para que un nodo i sea candidato debe cumplir:
    a. No está etiquetado, $etiq_i = 0$
    b. Su arco asociado tiene capacidad residual positiva, $capr_{npar,i} > 0$
  - Ir al PASO 4 (ponderación de candidatos)
- (Ni el nodo de referencia ni su anterior son la fuente, $npar \neq f$, $nant_{npar} \neq f$). Si el nodo de referencia es nuevo para esta iteración, $nuevo = 1$, fijar la cota superior para las eventuales etiquetas de los nodos alcanzables desde el nodo de referencia como la etiqueta del nodo de referencia

$$delmx = del_{npar}$$

- Revisar todos los arcos usados, k, incidentes en el nodo de referencia, $npar$, desde el evento inmediatamente anterior al correspondiente al arco desde el que se etiquetó el nodo de referencia, $pos1_{npar}$, hasta el primer evento, en orden inverso de ocurrencia. Cada arco k lleva asociado un evento e tal que $k = into_{npar,e}$:
  - Si el arco k sale del nodo, $npar = ini_k$:
    - Si no queda *stock* tras el evento, $stck_{npar,e} = 0$, ir al PASO 4 (ponderación de candidatos)
    - (Queda *stock* tras el evento, $stck_{npar,e} > 0$). Si el nodo de referencia es nuevo para esta iteración, $nuevo = 1$, actualizar la cota para las etiquetas
$$delmx = \text{mín}\{delmx, stck_{npar,e}\}$$
  - (El arco k llega al nodo, $npar = fin_k$):
    - Añadir el nodo i a la lista de candidatos, $lcan$, si cumple:
      a. No está etiquetado: $etiq_i = 0$
      b. El arco tiene capacidad residual positiva : $capr_{npar,i} > 0$

- Si el nodo de referencia es nuevo para esta iteración, $nuevo = 1$, y el nodo i ha sido incluido en la lista de candidatos, $i \in lcan$, actualizar el aumento máximo de flujo entre los nodos $npar$ e i

$$flumx_{npar,i} = \text{mín}\{delmx, flumx_{npar,i}\}$$

- Incluir el sumidero, s, a la lista de candidatos a etiquetar si la capacidad residual del arco ficticio correspondiente es positiva, $capr_{npar,s} > 0$, y no hay otros candidatos, $lcan = \varnothing$. Ir al PASO 5 (conteo de candidatos)

PASO 4

- Si el nodo de referencia es la fuente, $npar = f$, cada candidato i es un depósito. Obtener su intensidad como el número de soluciones élite para las que el material que queda en el depósito es menor que en la solución que se está construyendo

$$int_i = |o \in O / esf_{i,o} < sf_i|$$

- Si el nodo de referencia no es la fuente, $npar \neq f$, cada nodo candidato i corresponde al comienzo de un arco k que llega al nodo $npar$. Obtener su intensidad como el número de soluciones élite para las que el material que circula por el arco es menor que la diferencia entre el material que circula por el arco y el flujo correspondiente en la solución que se está construyendo

$$int_i = |o \in O / eltar_{k,o} < ltar_k - flujo_{npar,i}|$$

- Obtener la ponderación de cada candidato i

$$kpon_i = \lambda \frac{int_i}{nelite} + (1 - \lambda)$$

- Obtener la probabilidad de elegir cada nodo i candidato

$$prob_i = \frac{kpon_i}{\sum_{i' \in lcan} kpon_{i'}} \quad \forall i \in lcan$$

PASO 5

- Si hay candidatos, $lcan \neq \varnothing$:
  - Seleccionar al azar un nodo, $ngan$, de la lista de candidatos, $lcan$, de acuerdo a las probabilidades asociadas, etiquetarlo, calcular su máximo aumento permitido de flujo y fijar su nodo anterior

$$\begin{gathered} etiq_{ngan} = 1 \\ delta_{ngan} = \text{mín}\{delta_{npar}, flumx_{npar,ngan}\} \\ nant_{ngan} = npar \end{gathered}$$

  - Si el nodo seleccionado es el sumidero, $ngan = s$, se ha encontrado un camino de aumento. Ir al PASO 6 (preparación para actualizar flujo)

- (El nodo seleccionado no es el sumidero, $ngan \neq s$). Si el nodo de referencia no es la fuente, $npar \neq f$, guardar las posiciones que ocupa el arco $k = (npar, ngan)$ en las listas de eventos de los nodos que une dicho arco

$$pos2_{npar} = e, \text{ siendo } k = into_{npar,e}$$
$$pos1_{ngan} = e', \text{ siendo } k = into_{ngan,e'}$$

- Tomar el nodo seleccionado como nodo de referencia

$$npar = ngan$$

- Ir al PASO 3 (búsqueda de candidatos para etiquetar)

▪ (No hay candidatos, $lcan = \varnothing$):

- Si el nodo de referencia no es la fuente, $npar \neq f$:
  - Tomar como nuevo nodo de referencia el nodo anterior al actual nodo de referencia

$$npar = npar_{ngan}$$

  - Indicar que el nuevo nodo de referencia ya ha sido anteriormente nodo de referencia en esta iteración

$$nuevo = 0$$

  - Ir al PASO 3 (búsqueda de candidatos para etiquetar)
- (El nodo de referencia es la fuente, $npar = f$). Si se han agotado los intentos para encontrar un camino de aumento, $itet = maxet$:
  - Actualizar las cargas de los convoyes por los arcos, restando los flujos obtenidos correspondientes

$$ltar_k = ltar_k - flujo_{i,i'} \quad \text{siendo } i = fin_k, i' = ini_k \quad \forall k$$

  - FIN del procedimiento
- (No se han agotado los intentos para encontrar un camino de aumento, $itet \neq maxet$):
  - Actualizar el número de intentos para encontrar un camino de aumento

$$itet = itet + 1$$

  - Volver al PASO 2 (búsqueda de un nuevo camino de aumento)

PASO 6

▪ Indicar que el nodo de referencia actual es el último del camino de aumento (exceptuando el sumidero)

$$ult = npar$$

- Tomar el sumidero como nodo de referencia

$$npar = s$$

- Calcular la cantidad en la que se va a aumentar el flujo

$$aumf = delta_s$$

PASO 7

- Actualizar el flujo y la capacidad residual en el arco $(nant_{npar}, npar)$

$$\begin{array}{l} flujo_{i,i'} = flujo_{i,i'} + aumf \\ capr_{i,i'} = capr_{i,i'} - aumf \end{array} \quad \text{siendo } i = nant_{npar}, i' = npar$$

- Si el nodo de referencia es el último nodo del camino de aumento (sin contar el sumidero), $npar = ult$:
  - Actualizar los *stocks* correspondientes a los eventos posteriores a la llegada del arco utilizado para etiquetar

$$stck_{npar,e} = stck_{npar,e} + aumf \quad \forall e \geq pos1_{npar}$$

  - Actualizar la capacidad residual desde la fuente al nodo de referencia

$$capr_{f,npar} = \text{mín}\{entra_{npar}, sf_{npar}\}$$

- Si el nodo de referencia es un nodo intermedio del camino de aumento, $npar \neq ult$, $npar \neq s$, $nant_{npar} \neq f$:
  - Actualizar los *stocks* correspondientes a los eventos comprendidos entre la llegada y salida de los arcos utilizados para etiquetar

$$stck_{npar,e} = stck_{npar,e} - aumf \quad \forall e \in \{pos2_{npar}, \ldots, pos1_{npar} - 1\}$$

  - Si el nodo de referencia es un depósito, $npar \in ldep$, actualizar la cantidad de material que entra y la capacidad residual entre la fuente y dicho nodo

$$\begin{array}{c} entra_{npar} = entra_{npar} - aumf \\ capr_{f,npar} = \text{mín}\{entra_{npar}, sf_{npar}\} \end{array}$$

- Si el nodo de referencia es el primer nodo en el camino de aumento (exceptuando la fuente), $nant_{npar} = f$:
  - Actualizar los *stocks* correspondientes a los eventos posteriores a la salida del arco utilizado para etiquetar

$$stck_{npar,e} = stck_{npar,e} - aumf \quad \forall e \geq pos2_{npar}$$

  - Actualizar el valor del flujo añadiéndole la cantidad de aumento de flujo

$$valf = valf + aumf$$

- Volver al PASO 2 (búsqueda de un nuevo camino de aumento)

- (El nodo de referencia no es el primer nodo del camino de aumento, $nant_{npar} \neq f$). Tomar el nodo anterior en el camino de aumento como nodo de referencia

$$npar = nant_{npar}$$

- Volver al PASO 7 (actualización del flujo)

A.6. ACORTAR RUTAS 2

Modificación del procedimiento ACORTAR RUTAS en el que se introducen las funciones intensidad.

Variables de entrada

λ, $conv$, $lonv$, $ltar$, $secv$, $tamc$, $tamclas$, $tcpar$

Variables de salida

$conv$, $lonv$, $ltar$, $secv$, $tamc$, $tcpar$

Variables internas

int_j	Valor de la función intensidad correspondiente al vehículo j
$kpon_j$	Ponderación dada al vehículo j
$lcan$	Lista con los vehículos candidatos para eliminar su último trayecto
$prob_j$	Probabilidad de elegir el vehículo j

Algoritmo

PASO 1

- Eliminar todos los últimos trayectos de cada vehículo j en los que no haya transporte de material, empezando por el último, $k = secv_{j,lonv_j}$

$$ltar_k = 0 \quad \Rightarrow$$

$$\begin{cases} \text{eliminar el arco } k \text{ de la ruta del vehículo } j,\ secv_j \\ \text{reducir en una unidad la longitud de la ruta del vehículo } j,\ lonv_j \\ \text{eliminar el vehículo } j \text{ del convoy del arco } k,\ conv_k \\ \text{reducir en una unidad el tamaño del convoy del arco } k,\ tamc_k \\ \text{reducir en una unidad } tcpar_{c,k}, \text{ siendo } c = clas_{tipv_j,ori_j} \end{cases}$$

- Eliminar todos los arcos de cada circuito de cada vehículo j en los que no haya transporte de material. Asimismo eliminar el vehículo j de los convoyes

de los arcos del circuito y actualizar $tcpar_{c,k}$. El circuito $(k_1 = secv_{j,e}, k_2 = secv_{j,e+1}, \ldots, k_n = secv_{j,e+n-1})$, con $ini_{k_1} = fin_{k_n}$, se puede eliminar si

$$ltar_k = 0 \quad \forall k \in \{k_1, k_2, \ldots, k_n\}$$

PASO 2

- Construir una lista, $lcan$, con los vehículos cuyo último trayecto no es imprescindible: si $k = secv_{j,lonv_j}$

$$ltar_k \leq \sum_{j' \in conv_k} cap_{tipv_{j'}} - cap_{tipv_j} \quad \Longleftrightarrow \quad j \in lcan$$

- Si no hay candidatos, $lcan = \varnothing$, ir al PASO 3
- Obtener la intensidad de cada vehículo j de la lista $lcan$: número de soluciones élite para las que el número de vehículos de la clase del vehículo j que circulan por el arco $k = secv_{j,lonv_j}$ es menor que en la solución actual

$$int_j = |o \in O / tamclas_{c,k,o} < tcpar_{c,k}|$$
$$\text{siendo } c = clas_{tipv_j, ori_j}$$

- Obtener la ponderación de cada vehículo candidato j

$$kpon_j = \lambda \frac{int_j}{nelite} + (1 - \lambda)$$

- Obtener la probabilidad de elegir cada vehículo candidato j

$$prob_j = \frac{kpon_j}{\sum_{j' \in lcan} kpon_{j'}} \quad \forall j \in lcan$$

- Elegir al azar un vehículo $j \in lcan$ considerando las probabilidades obtenidas. Actualizar la ruta del vehículo elegido, el convoy del arco eliminado de su ruta y el número de vehículos de la clase del vehículo j que recorren el arco k según se describe en el PASO 1
- Volver al PASO 2

PASO 3

- Ordenar los vehículos aleatoriamente (o de acuerdo a cualquier otro criterio descrito en la introducción de ACORTAR RUTAS)
- Para cada vehículo j, eliminar todos los arcos de cada circuito en los que el vehículo no sea imprescindible y realizar las actualizaciones precisas. El vehículo j no es imprescindible en un circuito $(k_1 = secv_{j,e}, k_2 = secv_{j,e+1}, \ldots, k_n = secv_{j,e+n-1})$ con $ini_{k_1} = fin_{k_n}$ si

$$ltar_k \leq \sum_{j' \in conv_k} cap_{tipv_{j'}} - cap_{tipv_j} \quad \forall k \in \{k_1, k_2, \ldots, k_n\}$$

A.7. REFINAR SOLUCIÓN 2

Modificación del procedimiento REFINAR SOLUCIÓN en el que se se emplean las versiones modificadas de algunos procedimientos.

Variables de entrada

λ, $conv$, dur, $eltar$, esf, $evcr$, $into$, $lonv$, $ltar$, $nuan$, $secv$, sf, $stock$, $tamc$, $tamclas$, $tcpar$, $tmin$

Variables de salida

$actpr$, $conv$, $lonv$, $ltar$, $secv$, $tamc$

Procedimientos llamados

ACORTAR RUTAS 2, APROVECHAR DEPÓSITOS 2, INTERCAMBIAR VEHÍCULOS, RETENER MATERIAL 2, SUSTITUIR VEHÍCULOS 2

Algoritmo

PASO 0

- Indicar que no hay que actualizar las precedencias

$$actpr = 0$$

PASO 1

- Si ya se ha aplicado el procedimiento EVITAR CRUCES, $evcr = 1$, ir al PASO 2
- (Todavía no se han evitado cruces, $evcr = 0$). Ejecutar el procedimiento RETENER MATERIAL 2, para retener parte del material que sale de nodos con demanda satisfecha baja teniendo en cuenta las funciones intensidad
- Ejecutar el procedimiento APROVECHAR DEPÓSITOS 2, para eliminar flujo superfluo de material que pasa por los depósitos teniendo en cuenta las funciones intensidad

PASO 2

- Ejecutar el procedimiento ACORTAR RUTAS 2, para eliminar trayectos innecesarios de los vehículos teniendo en cuenta las funciones intensidad
- Ejecutar el procedimiento SUSTITUIR VEHÍCULOS 2, para sustituir vehículos para seguir eliminando trayectos innecesarios

- Ejecutar el procedimiento INTERCAMBIAR VEHÍCULOS, para intercambiar vehículos para disminuir la suma de distancias de regreso
- Si se ha producido algún cambio en alguno de los elementos de la solución, indicar que hay que actualizar las precedencias

$$actpr = 1$$

A.8. MEJORA 2

Modificación del procedimiento MEJORA, descrito en el Capítulo 4, en el que se hace uso de funciones intensidad en los procedimientos RETENER MATERIAL, APROVECHAR DEPÓSITOS Y ACORTAR RUTAS, englobados en el procedimiento REFINAR SOLUCIÓN. Además, en el procedimiento SUSTITUIR VEHÍCULOS se realiza una pequeña modificación.

Variables de entrada

λ, $cont$, $conv$, dur, $eltar$, esf, $into$, $lonv$, $ltar$, $nuan$, $nump$, $prec$, $pred$, $prei$, $pret$, $secv$, sf, $stck$, $suci$, $suct$, $tamc$, $tamclas$, $tcpar$, $tmin$, $util$

Variables de salida

$conv$, f, fob, $load$, $lonv$, $ltar$, $prec$, $secv$, sf, $tmin$

Variables internas
$actpr$, $evcr$, $inst$, $negat$, pf, $pfob$, $pload$, $prei$, $pret$, $psecv$, $ptmin$

Procedimientos llamados

ACTUALIZAR EVENTOS, EVITAR CRUCES, POSITIVAR, RESERVAR SOLUCIÓN, RECUPERAR SOLUCIÓN, REFINAR SOLUCIÓN 2, REPARTIR CARGAS

Algoritmo

PASO 1

- Ejecutar el procedimiento REFINAR SOLUCIÓN 2, para eliminar elementos innecesarios de la solución teniendo en cuenta las funciones intensidad
- Si no hay que actualizar las precedencias, $actpr = 0$, ir al PASO 3 (solución válida)
- (Hay que actualizar las precedencias, $actpr = 1$). Ejecutar el procedimiento ACTUALIZAR EVENTOS, para reorganizar los eventos y *stocks* en los nodos.
- Ejecutar el procedimiento POSITIVAR, para detectar y eliminar los posibles *stocks* negativos

PASO 2

- Ejecutar el procedimiento REPARTIR CARGAS, para repartir el material que se transporta por los arcos entre los vehículos de los convoyes correspondientes
- Evaluar la función objetivo, fob, de acuerdo a los pesos establecidos para los atributos

PASO 3

- Ejecutar el procedimiento RESERVAR SOLUCIÓN, para guardar la solución obtenida
- Ejecutar el procedimiento EVITAR CRUCES, para impedir cruces de material entre dos nodos
- Si la solución no ha cambiado, $evcr = 0$, FIN del procedimiento
- (La solución ha cambiado, $evcr = 1$). Ejecutar el procedimiento REFINAR SOLUCIÓN 2, para eliminar elementos innecesarios de la solución teniendo en cuenta las funciones intensidad
- Si no hay que actualizar las precedencias, $actpr = 0$, ir al PASO 4 (nueva solución válida)
- (Hay que actualizar las precedencias, $actpr = 1$). Ejecutar el procedimiento ACTUALIZAR EVENTOS, para reorganizar los eventos y *stocks* en los nodos.
- Ejecutar el procedimiento POSITIVAR, para detectar y eliminar los posibles *stocks* negativos
- Si hay algún *stock* negativo, $negat = 1$:
  - Ejecutar el procedimiento RECUPERAR SOLUCIÓN, para recuperar la última solución válida
  - FIN del procedimiento

PASO 4

- Ejecutar el procedimiento REPARTIR CARGAS, para repartir el material que se transporta por los arcos entre los vehículos de los convoyes correspondientes
- Evaluar la función objetivo, fob, de acuerdo a los pesos establecidos para los atributos
- Si la nueva solución obtenida es mejor que la solución original, $fob \leq pfob$, FIN del procedimiento
- (La nueva solución obtenida es peor que la solución original, $fob > pfob$). Ejecutar el procedimiento RECUPERAR SOLUCIÓN, para recuperar la mejor solución obtenida

Bibliografía

Abounacer, R., Rekik, M., y Renaud, J. (2014). An exact solution approach for multi-objective location-transportation problem for disaster response. *Computers & Operations Research*, 41:83–93.

Adıvar, B. y Mert, A. (2010). International disaster relief planning with fuzzy credibility. *Fuzzy Optimization and Decision Making*, 9(4):413–433.

Afsar, H. M., Prins, C., y Santos, A. C. (2014). Exact and heuristic algorithms for solving the generalized vehicle routing problem with flexible fleet size. *International Transactions in Operational Research*, 21(1):153–175.

Afshar, A. y Haghani, A. (2012). Modeling integrated supply chain logistics in real-time large-scale disaster relief operations. *Socio-Economic Planning Sciences*, 46(4):327–338.

Allahviranloo, M., Chow, J. Y. J., y Recker, W. W. (2014). Selective vehicle routing problems under uncertainty without recourse. *Transportation Research Part E*, 62:68–88.

Allahyari, S., Salari, M., y Vigo, D. (2015). A hybrid metaheuristic algorithm for the multi-depot covering tour vehicle routing problem. *European Journal of Operational Research*, 242(3):756–768.

Altay, N. y Green, W. G. (2006). OR/MS research in disaster operations management. *European Journal of Operational Research*, 175(1):475–493.

Álvarez, M. (2011). Sistemas de ayuda a la decisión en logística humanitaria. Trabajo académicamente dirigido, Universidad Complutense de Madrid.

Ang, C. C. (2006). Optimized recovery of damaged electrical power grids. Tesis de máster, Naval Postgraduate School, Monterey.

Angelis, V. D., Mecoli, M., Nikoi, C., y Storchi, G. (2007). Multiperiod integrated routing and scheduling of World Food Programme cargo planes in Angola. *Computers & Operations Research*, 34(6):1601–1615.

Balcik, B. y Beamon, B. M. (2008). Facility location in humanitarian relief. *International Journal of Logistics Research and Applications*, 11(2):101–121.

Barzinpour, F. y Esmaeili, V. (2014). A multi-objective relief chain location distribution model for urban disaster management. *The International Journal of Advanced Manufacturing Technology*, 70(5):1291–1302.

Barzinpour, F., Saffarian, M., Makoui, A., y Teimoury, E. (2014). Metaheuristic algorithm for solving biobjective possibility planning model of location-allocation in disaster relief logistics. *Journal of Applied Mathematics*. DOI 10.1155/2014/239868.

Beamon, B. M. y Kotleba, S. A. (2006a). Inventory management support systems for emergency humanitarian relief operations in South Sudan. *International Journal of Logistics Management*, 17(2):187–212.

Beamon, B. M. y Kotleba, S. A. (2006b). Inventory modelling for complex emergencies in humanitarian relief operations. *International Journal of Logistics Research and Applications*, 9(1):1–18.

Ben-Tal, A., Chung, B. D., Mandala, S. R., y Yao, T. (2011). Robust optimization for emergency logistics planning: risk mitigation in humanitarian relief supply chains. *Transportation Research Part B*, 45(8):1177–1189.

Berkoune, D., Renaud, J., Rekik, M., y Ruiz, A. (2012). Transportation in disaster response operations. *Socio-Economic Planning Sciences*, 46(1):23–32.

Blecken, A., Danne, C., Dangelmaier, W., Rottkemper, B., y Hellingrath, B. (2010). Optimal stock relocation under uncertainty in post-disaster humanitarian operations. En *Proceedings of the 43rd Hawaii International Conference on System Sciences*, pp. 1–10, Koloa, Hawaii.

Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2(4):353–373.

Blum, C. y Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308.

Bozorgi-Amiri, A. y Asvadi, S. (2015). A prioritization model for locating relief logistic centers using analytic hierarchy process with interval comparison matrix. *Knowledge-Based Systems*, 86:173–181.

Bozorgi-Amiri, A., Jabalameli, M. S., y Al-e Hashem, S. M. J. M. (2013). A multi-objective robust stochastic programming model for disaster relief logistics under uncertainty. *OR Spectrum*, 35:905–933.

Bozorgi-Amiri, A., Jabalameli, M. S., Alinaghian, y Heydari, M. (2012). A modified particle swarm optimization for disaster relief logistics under uncertain environment. *The International Journal of Advanced Manufacturing Technology*, 60(1):357–371.

Bretschneider, S. y Kimms, A. (2012). Pattern-based evacuation planning for urban areas. *European Journal of Operational Research*, 216(1):57–69.

Bullnheimer, B., Hartl, R. F., y Strauss, C. (1999). An improved Ant System algorithm for the Vehicle Routing Problem. *Annals of Operations Research*, 89(0):319–328.

Campbell, A. M., Vandenbussche, D., y Hermann, W. (2008). Routing for relief efforts. *Transportation Science*, 42(2):127–145.

Chandra Mohan, B. y Baskaran, R. (2012). A survey: Ant Colony Optimization based recent research and implementation on several engineering domain. *Expert Systems with Applications*, 39(4):4618–4627.

Chang, F. S., Wu, J. S., Lee, C. N., y Shen, H. C. (2014). Greedy-search-based multi-objective genetic algorithm for emergency logistics scheduling. *Expert Syst. Appl.*, 41(6):2947–2956.

Charles, A. (2010). *Improving the Design and Management of Agile Supply Chains: Feedback and Application in the Context of Humanitarian Aid*. Tesis doctoral, Université de Toulouse, Toulouse, France.

Charnes, A. y Cooper, W. W. (1961). *Management models and industrial applications of linear programming*. John Wiley and Sons, New York.

Chen, A., Zhao, J., y Chen, N. (2009). A recoverability assessment model in emergency management. Technical Report RTO-MP-IST-086-09, NATO Research and Technology Organisation.

Chiu, Y. C. y Zheng, H. (2007). Real-time mobilization decisions for multi-priority emergency response resources and evacuation groups: model formulation and solution. *Transportation Research Part E*, 43(6):710–736.

Cimellaro, G. P., Reinhorn, A. M., y Bruneau, M. (2010). Framework for analytical quantification of disaster resilience. *Engineering Structures*, 32(11):3639–3649.

Coffrin, C., Van Hentenryck, P., y Bent, R. (2011a). Strategic planning for power system restoration. En *Proceedings of the International Conference on Vulnerability and Risk Analysis and Management*, pp. 1–8, Hyattsville, Maryland.

Coffrin, C., Van Hentenryck, P., y Bent, R. (2011b). Strategic stockpiling of power system supplies for disaster recovery. En *Power and Energy Society General Meeting, 2011 IEEE*, pp. 1–8, San Diego, California.

de la Torre, L. E., Dolinskaya, I. S., y Smilowitz, K. R. (2012). Disaster relief routing: integrating research and practice. *Socio-Economic Plan. Sci.*, 46:88–97.

de Mel, S., McKenzie, D., y Woodruff, C. (2008). Mental health recovery and economic recovery after the tsunami: high-frequency longitudinal evidence from Sri Lankan small business owners. *Social Science & Medicine*, 66(3):582–595.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Doerner, K. F., Gutjahr, W. J., y Nolz, P. C. (2009). Multi-criteria location planning for public facilities in tsunami-prone coastal areas. *OR Spectrum*, 31(3):651–678.

Dorigo, M. y Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243–278.

Dorigo, M. y Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, 1(1):53–66.

Dorigo, M., Maniezzo, V., y Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):29–41.

Eiselt, H. A. y Marianov, V. (2012). Mobile phone tower location for survival after natural disasters. *European Journal of Operational Research*, 216(3):563–572.

El-Anwar, O., El-Rayes, K., y Elnashai, A. S. (2010). Maximizing the sustainability of integrated housing recovery efforts. *Journal of Construction Engineering Management*, 136(7):794–802.

Ertem, M. A., Buyurgan, N., y Pohl, E. A. (2012). Using announcement options in the bid construction phase for disaster relief procurement. *Socio-Economic Planning Sciences*, 46(4):306–314.

Ertem, M. A., Buyurgan, N., y Rossetti, M. D. (2010). Multiple-buyer procurement auctions framework for humanitarian supply chain management. *International Journal of Physical Distribution & Logistics Management*, 40(3):202–227.

Esmaeili, V. y Barzinpour, F. (2014). Integrated decision making model for urban disaster management: A multi-objective genetic algorithm approach. *International Journal of Industrial Engineering Computations*, 5(1):55–70.

Falasca, M. y Zobel, C. (2012). An optimization model for volunteer assignments in humanitarian organizations. *Socio-Economic Planning Sciences*, 46(4):250–260.

Feo, T. y Resende, M. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8(2):67–71.

Feo, T. y Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–113.

Festa, P. y Resende, M. G. C. (2011). GRASP: basic components and enhancements. *Telecommunication Systems*, 46(3):253–271.

Fetter, G. y Rakes, T. (2012). Coordinating debris cleanup operations in post disaster road networks. *Socio-Economic Planning Sciences*, 46(1):14–22.

Fiedrich, F. y Burghardt, P. (2007). Agent-based systems for disaster management. *Communications of the ACM - Emergency Response Information Systems: Emerging Trends and Technologies*, 50(3):41–42.

Fleurent, C. y Glover, F. (1999). Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, 11(2):198–204.

Ford, L. R. y Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404.

Galindo, G. y Batta, R. (2013). Review of recent developments in OR/MS research in disaster operations management. *European Journal of Operational Research*, 230(2):201–211.

Gambardella, L. M., Taillard, E., y Agazzi, G. (1999). MACS-VRPTW: a multiple colony system for vehicle routing problems with time windows. En Corne, D., Dorigo, M., Glover, F., Dasgupta, D., Moscato, P., Poli, R., y Price, K. V., editores, *New ideas in optimization*, pp. 63–76. McGraw-Hill.

Gibbons, D. E. y Samaddar, S. (2009). The maximal expected coverage relocation problem for emergency vehicles. *Decision Sciences*, 40(2):351–371.

Greistorfer, P. (2003). A tabu scatter search metaheuristic for the arc routing problem. *Computers & Industrial Engineering*, 44(2):249–266.

Hale, T. y Moberg, C. R. (2005). Improving supply chain disaster preparedness: a decision process for secure site location. *International Journal of Physical Distribution & Logistics Management*, 35(3):195–207.

Holguín-Veras, J., Pérez, N., Jaller, M., Van Wassenhove, L., y Aros-Vera, F. (2013). On the appropriate objective function for post-disaster humanitarian logistics models. *Journal of Operations Management*, 31(5):262–280.

Hoyos, M. C., Morales, R. S., y Akhavan-Tabatabaei, R. (2015). OR models with stochastic components in disaster operations management: A literature survey. *Computers & Industrial Engineering*, 82:183–197.

Hu, Z. H. y Sheu, J. B. (2013). Post-disaster debris reverse logistics management under psychological cost minimization. *Transportation Research Part B*, 55:118–141.

Huang, K., Jiang, Y., Yuan, Y., y Zhao, L. (2015). Modeling multiple humanitarian objectives in emergency response to large-scale disasters. *Transportation Research Part E: Logistics and Transportation Review*, 75:1–17.

Huang, M., Smilowitz, K., y Balcik, B. (2012). Models for relief routing: equity, efficiency and efficacy. *Transportation Research Part E*, 48(1):2–18.

Jin, M. y Ekşioğlu, B. (2010). Optimal routing of vehicles with communication capabilities in disasters. *Computational Management Science*, 7(2):121–137.

Jotshi, A., Gong, Q., y Batta, R. (2009). Dispatching and routing of emergency vehicles in disaster mitigation using data fusion. *Socio-Economic Planning Sciences*, 43(1):1–24.

Jung, C., Johnson, M., Trick, M., y Williams, J. (2007). Mathematical models for reconstruction planning in urban areas. Technical report, Heinz College, Carnegie Mellon University.

Kaklauskas, A., Amaratunga, D., y Haigh, R. (2009). Knowledge model for post-disaster management. *International Journal of Strategic Property Management*, 13(2):117–128.

Karlaftis, M. G., Kepaptsoglou, K. L., y Lambropoulos, S. (2007). Fund allocation for transportation network recovery following natural disasters. *Journal of Urban Planning and Development*, 133(1):82–89.

Ke, L. y Feng, Z. (2013). A two-phase metaheuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 40(2):633–638.

Kongsomsaksaku, S., Yang, C., y Chen, A. (2005). Shelter location-allocation model for flood evacuation planning. *Journal of the Eastern Asia Society for Transportation Studies*, 6:4237–4252.

Kontoravdis, G. y Bard, J. F. (1995). A GRASP for the vehicle routing problem with time windows. *ORSA Journal on Computing*, 7(1):10–23.

Kovács, G. y Spens, K. M. (2007). Humanitarian logistics in disaster relief operations. *International Journal of Physical Distribution & Logistics Management*, 37(2):99–114.

Lee, C. Y., Lee, Z. J., Lin, S. W., y Ying, K. C. (2010). An enhanced ant colony optimization (EACO) applied to capacitated vehicle routing problem. *Applied Intelligence*, 32(1):88–95.

Liberatore, F., Ortuño, M. T., Tirado, G., Vitoriano, B., y Scaparra, M. P. (2014). A hierarchical compromise model for the joint optimization of recovery operations and distribution of emergency goods in Humanitarian Logistics. *Computers & Operations Research*, 42:3–13.

Liberatore, F., Pizarro, C., Simón de Blas, C., Ortuño, M. T., y Vitoriano, B. (2013). Decision aid models and systems for humanitarian logistics. A survey. En Vitoriano, B., Montero, J., y Ruan, D., editores, *Decision Aid Models for Disaster Management and Emergencies*, pp. 17–44. Atlantis Computational Intelligence Systems.

Lim, G. J., Zangeneh, S., Baharnemati, M. R., y Assavapokee, T. (2012). A capacitated network flow optimization approach for short notice evacuation planning. *European Journal of Operational Research*, 223(1):234–245.

Lin, Y. H., Batta, R., Rogerson, P. A., Blatt, A., y Flanigan, M. (2012). A logistics model for emergency supply of critical items in the aftermath of a disaster. *Socio-Economic Planning Sciences*, 45(4):132–145.

Liu, Y. y Guo, B. (2014). A lexicographic approach to postdisaster relief logistics planning considering fill rates and costs under uncertainty. *Mathematical Problems in Engineering*. DOI:10.1155/2014/939853.

Martí, R., Campos, V., Resende, M. G. C., y Duarte, A. (2014). Multiobjective GRASP with path relinking. *European Journal of Operational Research*, 240(1):54–71.

Matisziw, T. C., Murray, A. T., y Grubesic, T. H. (2010). Strategic network restoration. *Networks and Spatial Economics*, 10:345–361.

Maya, P. y Sörensen, K. (2011). A GRASP metaheuristic to improve accessibility after a disaster. *Disaster Relief*, 33(3):525–542.

Mehlhorn, S., Racer, M., Ivey, S., y Lipinski, M. (2011). A single-objective recovery phase model. *International Journal of Information Technology Project Management*, 2(3):53–71.

Najafi, N., Eshghi, K., y Dullaert, W. (2013). A multi-objective robust optimization model for logistics planning in the earthquake response phase. *Transportation Research Part E*, 49(1):217–249.

Nguyen, V. P., Prins, C., y Prodhon, C. (2012). Solving the two-echelon location routing problem by a GRASP reinforced by a learning process and path relinking. *European Journal of Operational Research*, 216(1):113–126.

Nolz, P. C., Doerner, K. F., Gutjahr, W. J., y Hartl, R. F. (2010a). A bi-objective metaheuristic for disaster relief operation planning. En Coello, C. A., Dhaenens, C., y Jourdan, L., editores, *Advances in Multi-Objective Nature Inspired Computing*, volumen 272 de *Studies in Computational Intelligence*, pp. 167–187. Springer, Berlin Heidelberg.

Nolz, P. C., Doerner, K. F., y Hartl, R. F. (2010b). Water distribution in disaster relief. *International Journal of Physical Distribution & Logistics Management*, 40(8-9):693–708.

Nolz, P. C., Semet, F., y Doerner, K. F. (2011). Risk approaches for delivering disaster relief supplies. *OR Spectrum*, 33(3):543–569.

OCHA (2015). Ocha: Office for the coordination of humanitarian affairs. http://ochaonline.un.org/ [último acceso octubre 2015].

Orabi, W., El-Rayes, K., Senouci, A. B., y Al-Derham, H. (2009). Optimizing postdisaster reconstruction planning for damaged transportation networks. *Journal of Construction Engineering and Management*, 135(10):1039–1048.

Orabi, W., Senouci, A. B., El-Rayes, K., y Al-Derham, H. (2010). Optimizing resource utilization during the recovery of civil infrastructure systems. *Journal of Management in Engineering*, 26(4):237–246.

Ortuño, M., Tirado, G., y Vitoriano, B. (2011). A lexicographical goal programming based decision support system for logistics of Humanitarian Aid. *TOP*, 19:464–479.

Ortuño, M. T., Cristóbal, P., Ferrer, J. M., Martín-Campo, F. J., Muñoz, S., Tirado, G., y Vitoriano, B. (2013). Decision aid models and systems for humanitarian logistics. A survey. En Vitoriano, B., Montero, J., y Ruan, D., editores, *Decision Aid Models for Disaster Management and Emergencies*, pp. 17–44. Atlantis Computational Intelligence Systems.

Özdamar, L. (2011). Planning helicopter logistics in disaster relief. *OR Spectrum*, 33(3):655–672.

Özdamar, L., Aksu, D. T., y Ergünes, B. (2014). Coordinating debris cleanup operations in post disaster road networks. *Socio-Economic Planning Sciences*, 48(4):249–262.

Özdamar, L. y Demir, O. (2012). A hierarchical clustering and routing procedure for large scale disaster relief logistics planning. *Transportation Research Part E*, 48(3):591–602.

Özdamar, L. y Ertem, M. A. (2015). Models, solutions and enabling technologies in humanitarian logistics. *European Journal of Operational Research*, 244(1):55–65.

Pedraza, A. J., Stapleton, O., y Van Wassenhove, L. N. (2010). Using or to support humanitarian operations: Learning from the haiti earthquake. Technical report, INSEAD Humanitarian Research Group.

Permann, M. R. (2007). Genetic algorithms for agent-based infrastructure interdependency modeling and analysis. En *Proceedings of the 2007 Spring Simulation Multiconference*, volumen 2, pp. 169–177, Norfolk, Virginia.

Qin, J., Xing, Y., Wang, S., Wang, K., y Chaudhry, S. S. (2012). An inter-temporal resource emergency management model. *Computers & Operations Research*, 39(8):1909–1918.

Rath, S. y Gutjahr, W. J. (2014). A math-heuristic for the warehouse location-routing problem in disaster relief. *Computers & Operations Research*, 42:25–39.

Redhum.org (2015). Red de información humanitaria para américa latina y el caribe. http://www.redhum.org/mapas [último acceso octubre 2015].

Resende, M. y Ribeiro, C. (2003). Greedy randomized adaptive search procedures. En Glover, F. y Kochenberger, G., editores, *Handbook of Metaheuristics*, pp. 219–249. Kluwer Academic Publishers.

Resende, M. G. C. y Ribeiro, C. C. (2010). Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. En Gendreau, M. y Potvin, J. Y., editores, *Handbook of Metaheuristics, second edition*, pp. 283–320. Kluwer Academic Publishers.

Rezaei-Malek, M. y Tavakkoli-Moghaddam, R. (2014). Robust humanitarian relief logistics network planning. *Uncertain Supply Chain Management*, 2(2):73–96.

Ribeiro, M. H., Plastino, A., y Martins, S. L. (2006). Hybridization of GRASP metaheuristic with data mining techniques. *Journal of Mathematical Modelling and Algorithms*, 5(1):23–41.

Rivera, J. C., Afsar, H. M., y Prins, C. (2015). A multistart iterated local search for the multitrip cumulative capacitated vehicle routing problem. *Computational Optimization and Applications*, 61(1):159–187.

Rizzoli, A. E., Montemanni, R., Lucibello, E., y Gambardella, L. M. (2007). Ant colony optimization for real-world vehicle routing problems. *Annals of Operations Research*, 1(2):135–151.

Saadatseresht, M., Mansourian, A., y Taleai, M. (2009). Evacuation planning using multiobjective evolutionary optimization approach. *European Journal of Operational Research*, 198(1):305–314.

Saleem, K., Luis, S., Deng, Y., Chen, S.-C., Hristidis, V., y Li, T. (2008). Towards a business continuity information network for rapid disaster recovery. En *Proceedings of the 2008 International Conference on Digital Government Research*, pp. 107–116, Montreal, Canada.

Salmerón, A. y Apte, A. (2010). Stochastic optimization for natural disaster asset prepositioning. *Production and Operations Management*, 19(5):561–574.

Santos, L. F., Ribeiro, M. H., Plastino, A., y Martins, S. L. (2005). A hybrid GRASP with data mining for the maximum diversity problem. En Blesa, M. J., Blum, C., Roli, A., y Sampels, M., editores, *Hybrid Metaheuristics*, volumen 3636 de *Lecture Notes in Computer Science*, pp. 116–127. Springer, Berlin Heidelberg.

Sharif, M. T. y Salari, M. (2015). A GRASP algorithm for a humanitarian relief transportation problem. *International Journal of Computational Intelligence Systems*, 41:259–269.

Shen, Z., Dessouky, M. M., y Ordóñez, F. (2009). A two-stage vehicle routing model for large-scale bioterrorism emergencies. *Networks*, 54(4):255–269.

Sheu, J. B. (2007). An emergency logistics distribution approach for quick response to urgent relief demand in disasters. *Transportation Research Part E*, 43(6):687–709.

Sheu, J. B. (2010). Dynamic relief-demand management for emergency logistics operations under large-scale disasters. *Transportation Research Part E*, 46(6):1–17.

Talbi, E. G. (2009). *Metaheuristics: From Design to Implementation*. John Wiley & Sons, Hoboken.

Tirado, G., Martín-Campo, F. J., Vitoriano, B., y Ortuño, M. T. (2014). A lexicographical dynamic flow model for relief operations. *International Journal of Computational Intelligence Systems*, 7(sup1):45–57.

Tomasini, R. y van Wassenhove, L. (2009a). From preparedness to partnerships: case study research on humanitarian logistics. *International Transactions in Operational Research*, 16(5):549–559.

Tomasini, R. y van Wassenhove, L. (2009b). *Humanitarian Logistics*. Palgrave Macmillan.

Tovia, F. (2007). An emergency logistics response system for natural disasters. *International Journal of Logistics: Research and Applications*, 10(3):173–186.

Tricoire, F., Graf, A., y Gutjahr, W. J. (2012). The bi-objective stochastic covering tour problem. *Computers & Operations Research*, 39(7):1582–1592.

Tzeng, G. H., Cheng, H. J., y Huang, T. D. (2007). Multi-objective optimal planning for designing relief delivery systems. *Transportation Research Part E*, 43(6):673–686.

Van Wassenhove, L. N. (2006). Humanitarian aid logistics: supply chain management in high gear. *Journal of the Operational Research Society*, 57(5):475–489.

Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., y Velasco, N. (2011). A GRASP with evolutionary path relinking for the truck and trailer routing problem. *Computers & Operations Research*, 38(9):1319–1334.

Vitoriano, B., Ortuño, M. T., y Tirado, G. (2009). HADS, a goal programming-based humanitarian aid distribution system. *Journal of Multi-Criteria Decision Analysis*, 16(1-2):55–64.

Vitoriano, B., Ortuño, M. T., Tirado, G., y Montero, J. (2011). A multi-criteria optimization model for humanitarian aid distribution. *Journal of Global Optimization*, 51(2):189–208.

Vitoriano, B., Rodríguez, J. T., Tirado, G., Martín-Campo, F. J., Ortuño, M. T., y Montero, J. (2015). Intelligent decision-making models for disaster management. *Human and Ecological Risk Assessment: An International Journal*, 21:1341–1360.

Yi, W. y Kumar, A. (2007). Ant colony optimization for disaster relief operations. *Transportation Research Part E*, 43(6):660–672.

Yi, W. y Özdamar, L. (2007). A dynamic logistics coordination model for evacuation and support in disaster response activities. *European Journal of Operational Research*, 179(3):1177–1193.

Yu, B., Yang, Z. Z., y Xie, J. X. (2011). A parallel improved ant colony optimization for multi-depot vehicle routing problem. *Journal of the Operational Research Society*, 62:183–188.

Yu, B., Yang, Z. Z., y Yao, B. (2009). An improved ant colony optimization for vehicle routing problem. *European Journal of Operational Research*, 196(1):171–176.

Zeleny, M. (1973). *Multiple criteria decision making*. University of South Carolina Press.

Zhang, J. H., Li, J., y Liu, Z. P. (2012a). Multiple-resource and multiple-depot emergency response problem considering secondary disasters. *Expert Systems with Applications*, 39(12):11066–11071.

Zhang, L. Y., Fei, T., Zhang, X., Li, Y., y Yi, J. (2012b). Research about immune ant colony optimization in emergency logistics transportation route choice. En Zhao, M. y Sha, J., editores, *Communications and Information Processing*, volumen 288 de *Communications in Computer and Information Science*, pp. 430–437. Springer, Berlin Heidelberg.

Zhang, M. X., Zhang, B., y Zheng, Y. J. (2014). Bio-inspired meta-heuristics for emergency transportation problems. *Algorithms*, 7(1):15–31.

Zhao, N., Wu, Z., Zhao, Y., y Quan, T. (2010). Ant colony optimization algorithm with mutation mechanism and its applications. *Expert Systems with Applications*, 37(7):4805–4810.

Zheng, Y. J. y Ling, H. F. (2013). Emergency transportation planning in disaster relief supply chain management: a cooperative fuzzy optimization approach. *Soft Computing*, 17(1):1301–1314.

Zobel, C. W. y Khansa, L. (2014). Characterizing multi-event disaster resilience. *Computers & Operations Research*, 42:83–94.

Índice de figuras

Índice de tablas