Universidad de Burgos
Escuela Politécnica Superior
Programa de doctorado: *Ingeniería Civil e Industrial*

Tesis doctoral

DISEÑO DE UN SISTEMA DE RECOGIDA DE RESIDUOS URBANOS:

ENFOQUE MULTIOBJETIVO Y USO DE METAHEURÍSITCOS

Realizada por:

José Rubén Gómez Cámara

Dirigida por:
Dr. Joaquín Antonio Pacheco Bonrostro
Dr. Hernán Gonzalo Orden

Burgos
Febrero 2010

Quiero expresar mi agradecimiento:

A los Doctores Pacheco y Gonzalo, codirectores de este trabajo, sin los cuales no hubiera podido llevar a cabo el mismo.

A la mancomunidad Alfoz de Lara y a la Diputación Provincial de Soria, por su gentileza a la hora de aportar la información necesaria para la elaboración de este trabajo.

A los compañeros de mi departamento y, en general, de la Universidad de Burgos, por el apoyo que me han prestado en todo momento.

A mi familia y amigos, por el apoyo, la paciencia y el ánimo constante sin el cual, no hubiese sido posible este trabajo.

A todos ellos, sinceramente…
muchísimas gracias.

ÍNDICE

CAPÍTULO 3: BÚSQUEDA TABÚ

CAPÍTULO 4: LOS PROBLEMAS DE RUTAS DE VEHÍCULOS

INTRODUCCIÓN

En este trabajo se desarrolla una metodología ad hoc basada en búsqueda tabú, para la resolución de un problema multiobjetivo, consistente en el diseño de un sistema de recogida de la fracción orgánica de los Residuos Urbanos (RU) generados en un conjunto de municipios y su aplicación al sistema de recogida de residuos en la mancomunidad Alfoz de Lara situada al sur-este de la provincia de Burgos.

Es la Ley 10/1998, de 21 de abril, surgida de la adaptación al derecho nacional de la Directiva 91/156/CEE, la que pretende establecer una política de residuos fijando, entre otros aspectos, una política específica para las diferentes categorías de residuos, el ámbito de aplicación de la misma, las competencias de las diferentes administraciones, la forma en que habrá de hacerse la recogida de los residuos urbanos por las entidades locales, etc. Se pretende, asímismo, contribuir a la protección del medio ambiente coordinando la política de residuos con otras políticas, a fin de incentivar la reducción en origen, dando prioridad a la reutilización, reciclado y valorización de residuos.

El problema aquí planteado trata, por tanto, de diseñar un sistema de recogida para un conjunto de núcleos urbanos, teniendo en cuenta dos aspectos que, de forma evidente, influyen en la eficacia del servicio prestado: el coste total del servicio y la calidad del mismo (medida en términos de duración del intervalo entre dos recogidas consecutivas en un núcleo determinado)

El interés por este problema, surge tras la elaboración de un Proyecto Fin de Carrera en el que se aborda ya esta cuestión, recurriendo a herramientas informáticas existentes en el mercado (Sistemas de Información Geográfica con módulos que abordan la optimización de rutas en términos de coste). Tras reconsiderar el problema, se observa que puede ser mejorado incorporando una segunda función objetivo (que cuantifica la calidad del servicio prestado) dando lugar a un conjunto de soluciones no dominadas que conforman una curva de eficiencia.

La existencia de un Grupo de Investigación en el Departamento de Economía Aplicada de la Universidad de Burgos, coodinado por el Dr. Pacheco y especializado en el desarrollo de estrategias heurísticas y metaheurísticas aplicables, entre otros, a este tipo de problemas, y con gran número de publicaciones, permite el abordaje del mismo en condiciones óptimas.

La literatura científica relativa a este problema es muy escasa, siendo bastante más abundante en el caso de recogida de RU en entornos urbanos, como se indica más adelante.

La metodología empleada en su resolución, pasa por el empleo de la estrategia MOAMP (MultiObjective Adaptative Memory Procedure), que resulta ser muy efectiva en problemas multiobjetivo, así como otras estrategias empleadas en otros niveles de decisión.

A fin de validar y contrastar el algoritmo, se generan una serie de instancias ficticias en las que se recogen diferentes horizontes temporales, diferente número de núcleos a visitar, así como diferentes tasas de generación; dichas instancias son resueltas por el algoritmo a fin de ajustar las diferentes partes del mismo. Asimismo, se contrastan los resultados de este procedimiento, con los obtenidos tras la adaptación de un conocido algoritmo genético de propósito general para problemas multiobjetivo.

Por último, se generan las instancias reales (que se corresponden con los datos reales del problema a resolver) y se resuelven aplicando el algoritmo aquí desarrollado; también se resuelven mediante el algoritmo genético anteriomente citado.

El trabajo se estructura de la siguiente forma: en los capítulos uno y dos se aborda el estado actual de las técnicas heurísticas y metaheurísticas tanto en problemas mono-objetivo como en problemas multiobjetivo. En el capítulo tres, se describe, de forma más profunda, la búsqueda tabú. En el siguiente

capítulo, se hace una descripción de Los problemas de rutas y sus variantes, así como los diferentes métodos de resolución. En el resto de capítulos, se realiza la descripción del problema, la definición de variables, desarrollo del algoritmo, pruebas y resolución final; se indican también futuras líneas de investigación. En el último capítulo se aporta las referencias bibliográficas utilizadas.

Las diferentes partes de los algoritmos utilizados, se escriben en PASCAL, siendo posteriormente compilados con Delphi.

1.- HEURÍSTICOS Y METAHEURÍSTICOS: ESTADO DEL ARTE

1.- HEURÍSTICOS Y METAHEURÍSTICOS: ESTADO DEL ARTE

1.1.- INTRODUCCIÓN

Optimizar consiste en encontrar la mejor solución posible para un determinado problema; o como indica Martí coloquialmente, *poco más que mejorar* [1]. Ahora bien, en un contexto más científico, optimizar consiste en *encontrar la mejor solución posible para un determinado problema*; siempre existen varias soluciones y un criterio para discriminar entre ellas.

Se pueden encontrar gran cantidad de problemas de optimización en la industria, empresa, economía, ingeniería, etc. Entre otros: problemas de localización, servicios, centros de tratamiento, de recogida de Residuos Urbanos (RU), problemas de asignación de vehículos a rutas, personal a líneas de trabajo, problemas de partición, problemas de cubrimiento de conjunto, el conocido problema VRP (*Vehicle Routing Problem*) con todas sus variantes [2].

La parte de la matemática que aborda, en conjunto, estos problemas, es la Programación Matemática, la cual se subdivide, a su vez, en varias ramas, según las características de las variables y de las ecuaciones y/o inecuaciones que describen los modelos matemáticos. Si las ecuaciones son lineales, entonces se habla de Programación Lineal; en los restantes casos, se habla de Programación No Lineal.

Si las variables asumen solo valores enteros, se habla de Programación Entera, para distinguirlo del caso en el que asuman variables continuas, que constituye la Programación Continua; si son varios los objetivos a optimizar, se habla de Programación Multiobjetivo. Así mismo, se habla de Programación Determinística, cuando los parámetros que describen el problema tienen valores fijos; en el caso de tener valores aleatorios, se habla de Programación Estocástica.

Otra rama de la Programación matemática, es la Optimización Combinatoria, que trata de resolver problemas de optimización caracterizados por tener un número finito de soluciones.

1.2.- COMPLEJIDAD ALGORÍTMICA

La Teoría de la Complejidad Algorítmica trata de responder a preguntas del tipo: ¿cuán bueno es un algoritmo para resolver un problema? o ¿cuán difícil es intrínsecamente un problema dado? [3]

La idea intuitiva de problema 'difícil de resolver' queda reflejada en el término científico *NP-Hard* [4] (*dificultad no polinómica*), utilizado en el contexto de la complejidad algorítmica; según ella, *un problema de optimización difícil, es aquel para el cual no se puede garantizar encontrar una solución factible en un tiempo razonable* [5]. La existencia de gran cantidad y variedad de problemas difíciles que aparecen en la práctica, ha impulsado, desde hace varias décadas, el desarrollo de procedimientos eficientes para encontrar soluciones aceptables, aunque no sean las óptimas.

Para abordar con mayor profundidad el estudio de la Teoría de la Complejidad Algorítmica, se puede consultar, entre otros, a Salazar [3].

1.3.- MÉTODOS EXACTOS

Dada la complejidad de los problemas, solo las instancias con pocos clientes pueden ser resueltas de forma consistente por métodos exactos. En este tipo de metodologías, suele resolverse alguna relajación del problema y utilizar un esquema de ramificación y acotamiento (*Branch and Bound* [6]).

También se emplean otros métodos como el algoritmo de planos de corte de Gomory, o el de descomposición de Benders.

Además, existen otros tipos de problemas de optimización relativamente fáciles de resolver; éste es el caso de los problemas lineales, en los que, tanto la función objetivo como las restricciones, son expresiones lineales. Estos problemas pueden resolverse mediante un método, elaborado por Dantzig en 1947 y conocido como método simplex.

1.4.- HEURÍSTICOS

El término heurístico deriva del griego *heuriskein* que significa 'encontrar o descubrir'. Una de las acepciones que da la Real Academia Española de la Lengua, es la de *...en algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc.* Se ha adoptado, por tanto, este término en el ámbito de la optimización, para esta forma de resolución de problemas, que no garantiza el óptimo, pero sí una buena solución.

Debe quedar claro que, frente a los heurísticos, el tiempo que los métodos exactos pueden precisar para resolver un problema difícil, en caso de existir solución, es de un orden de magnitud muy superior al de los heurísticos, pudiendo darse el caso de no encontrarla.

De lo anterior se puede deducir que, tanto los métodos exactos como los heurísticos, persiguen encontrar el máximo (o el mínimo) de una función objetivo sobre un conjunto finito de soluciones, denotadas por *X;*

$$Min\ f(x)$$

$$Sujeto\ a:\ x \in X$$

No se exige ninguna condición o propiedad sobre la función objetivo o sobre la definición del conjunto; además, puesto que las variables son discretas y, generalmente, acotadas, *S* es finito, por lo que queda restringido su dominio a una serie finita de valores.

Aparte del concepto de 'problema difícil' citado anteriormente, existen otras razones que justifican el empleo de métodos heurísticos:

- No siempre se conocen los parámetros de la función objetivo.
- El método heurístico aporta una flexibilidad mayor a la hora de modelizar un problema.
- No siempre es posible incorporar todos los aspectos en el modelo matemático.

Hay que destacar, además, que así como en los métodos exactos existe, por lo general, un método claro y conciso, en los heurísticos, las estructuras aplicadas son específicas de cada tipo de problema, por lo que deben particularizarse para el mismo.

Hay un gran número de heurísticos, hasta el punto de editarse una revista *Journal of Heuristics*, editada por *Springer*, donde se recogen los últimos avances en la materia; otras publicaciones más recientes que abundan en la misma temática son *Journal of Memetics* e *International Journal of Metaheuristics.*

Existen, hoy en día, muchos tipos de heurísticos, siendo difícil su clasificación, no obstante, se puede establecer la siguiente clasificación [7].

1. Métodos constructivos.
2. Métodos de descomposición.
3. Métodos de reducción.
4. Métodos de manipulación del modelo.
5. Métodos de búsqueda por entornos.

Aunque todos estos métodos han contribuido a la resolución de diferentes problemas, son los métodos constructivos y los métodos de búsqueda por entornos los que constituyen la base de los métodos metaheurísticos, los cuales serán abordados más adelante.

1. MÉTODOS CONSTRUCTIVOS

Van añadiendo, de forma interactiva, elementos hasta completar una solución. Tradicionalmente suelen ser métodos deterministas y hacen la mejor elección en cada iteración según algún criterio.

2. MÉTODOS DE DESCOMPOSICIÓN

Descomponen el problema en sub-problemas más pequeños, siendo el resultado de uno, el dato de entrada del otro.

3. MÉTODOS DE REDUCCIÓN

Identifican propiedades que se cumplen en la mayoría de las soluciones buenas, que son introducidas como restricciones del problema. La finalidad es restringir el espacio de soluciones y simplificar el problema. Presentan el riesgo de obviar soluciones buenas.

4. MÉTODOS DE MANIPULACIÓN DEL MODELO

Modifican la estructura del modelo a fin de hacerlo más sencillo de resolver, deduciendo, a partir de la solución obtenida, la solución final del problema. Estos métodos pueden reducir el espacio de soluciones o aumentarlo.

5. MÉTODOS DE BÚSQUEDA POR ENTORNOS O BÚSQUEDA LOCAL

En estos procedimientos, se parte de una solución inicial que va siendo mejorada de forma progresiva, mediante la exploración de su entorno, y que se sustituye por la mejor solución de éstas, hasta no encontrar una solución mejor. Se dice, entonces, que existe un mínimo/máximo local. Por tanto, la solución de este algoritmo da lugar a un óptimo local, que puede o no, coincidir con el óptimo global.

Así, en un problema de optimización combinatoria, cada solución presenta un conjunto de soluciones asociadas a ella, denominadas entorno de s, y que se denota como $N(s)$, se parte de una solución inicial s_0, se calcula su entorno $N(s_0)$ y se escoge una nueva solución s_1 mejor que s_0, se sustituye s_0 por s_1 y así sucesivamente; la sustitución de una solución por otra vecina, se denomina 'movimiento'.

El procedimiento de búsqueda local en pseudocódigo se puede escribir como:

Leer una solución inicial s_0
Repetir
- *Seleccionar* $s \in N(s_0) / f(s) < f(s_0)$ *por un método preestablecido*
- *Reemplazar* s_0 *por* s

Hasta que $f(s) \geq f(s_0), \ \forall \ s \in N(s_0)$

Así pues, un procedimiento de búsqueda local queda completamente determinado al definir un entorno y el criterio de selección dentro del entorno.

La definición de entorno depende en gran medida de la estructura del problema, así como de la función objetivo; así mismo, se debe definir el criterio para seleccionar una nueva solución del entorno; según lo anterior, puede buscarse la mejor solución de $N(s)$ (estrategia *best improvement*) o tomar la primera solución de $N(s)$ que mejore la función (estrategia *first improvement*).

En la figura 1 se puede observar cómo el óptimo obtenido por este procedimiento (óptimo local) se aleja bastante del óptimo global ya que, como se puede ver en el pseudocódigo, el algoritmo se detiene en el momento que la solución no mejora (empeora en todas las dirección) incurriendo en el error de detenerse en el anteriormente citado óptimo local.[1]

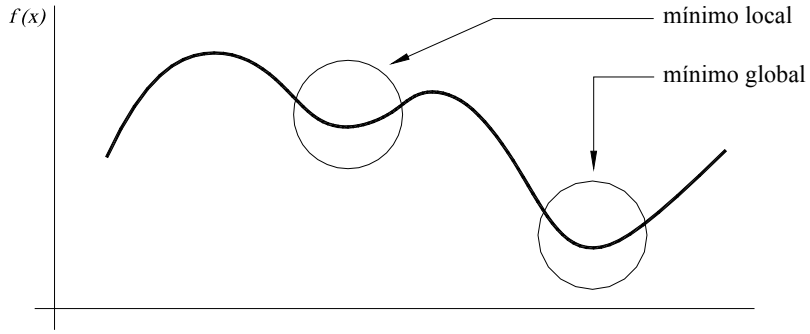


Figura 1. Mínimo local y mínimo global.

Esta limitación de estrategias da pie a la aparición de nuevos algoritmos, denominados 'metaheurísticos', que evitan quedar atrapados en mínimos/máximos locales, permitiendo la realización de ciertos movimientos que empeoran temporalmente la función objetivo.

1.5.- METAHEURÍSTICOS

Suponen un paso más en el desarrollo de las estrategias de optimización, ya que están destinados a la resolución de problemas más complejos que aparecen en la economía, empresa, ingeniería y otras.

El término metaheurístico aparece citado por primera vez en un artículo de Fred Glover [8] en 1986. Algunas definiciones que se pueden encontrar, son las siguientes:

> *...Metaheurística se refiere a una estrategia maestra que guía y modifica otras heurísticas para producir soluciones más allá de aquellas que normalmente se generan en una búsqueda de óptimos locales* [9].

> *Los Procedimientos Metaheurísticos son una clase de métodos aproximados, que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son ni efectivos ni eficientes. Los metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de Inteligencia Artificial, evolución biológica y mecanismos estadísticos* [10].

> *Un Metaheurístico es un procedimiento iterativo, con una estructura y unas reglas generales de funcionamiento que lo caracterizan, que guía un método (normalmente un heurístico) subordinado combinando inteligentemente diversos conceptos para explorar los espacios de búsqueda utilizando estrategias aprendidas para conseguir soluciones quasi-óptimas de manera eficiente* [11].

Los algoritmos metaheurísticos utilizan conceptos derivados de la Inteligencia Artificial, Biología, Matemáticas, Termodinámica… para, simulando aspectos esenciales de estas disciplinas, ser capaces de modelizar un problema y optimizarlo.

[1] A efectos de descripción, se considera un problema de minimización de la función objetivo, sin pérdida de generalidad.

Al igual que los heurísticos, son procedimientos iterativos que disponen de un mecanismo de parada pues, de lo contrario, no se detendrían nunca; dichos mecanismos de parada pueden ser: número de iteraciones, número de iteraciones sin mejorar, acercarse bastante a un óptimo, previa fijación de un rango admisible, etc. En cuanto a la jerarquía, si se establece un nivel, se debe indicar que los metaheurísticos están por encima de los heurísticos, por cuanto guían a estos.

Propiedades deseables, entre otras, podrían ser las siguientes:

- **Sencillez:** principios simples y claros.
- **Coherencia:** todos los pasos aplicados a un problema particular deben derivarse de los principios del metaheurístico.
- **Eficiencia**: en un problema particular deben aportar soluciones óptimas o pseudo-óptimas en la mayoría de las instancias reales.
- **Efectividad**: los tiempos de computación empleados en suministrar las soluciones anteriormente citadas deben ser aceptables.
- **Robustez**: comportamiento adecuado frente a diferentes instancias.
- **Amigables con el usuario**: deben estar bien definidos, comprensibles y fáciles de usar.
- **Innovación**: preferentemente deben llevar nuevas aplicaciones.

Tradicionalmente se han clasificado en dos grandes grupos: metaheurísticos basados en búsqueda por entornos y metaheurísticos basados en evolución de poblaciones, sin embargo, hoy en día, tal y como indican Laguna y Delgado [12] se producen hibridaciones entre las diferentes estrategias lo que hace más difícil la clasificación.

Efectivamente, en la actualidad, la mayoría de los metaheurísticos que se implementan resultan ser combinaciones de diferentes grupos; así, existen métodos constructivos que se complementan con una búsqueda local, como GRASP; otros, evolutivos, que se complementan con búsquedas guiadas, además de otras muchas combinaciones.

Otra apreciación importante, se puede hacer respecto a la forma de explorar el espacio de soluciones factibles; según esto, hay metaheurísticos que utilizan un procedimiento preestablecido y sistemático (basado generalmente en el uso de memoria), frente a otros, basados en estrategias aleatorias. Ahora bien, tampoco en este caso, se puede hablar de estrategias puras.

Más recientemente han aparecido estrategias heurísticas que determinan cuales son los heurísticos más adecuados para la resolución de un determinado problema. Estos métodos se denominan hiperheurísticos. La característica principal de estos algoritmos es la capacidad de diseñar estrategias de programación generales que pueden ser aplicadas a diferentes problemas o, como indican Cowling et al. [13], un hiperheurístico es un heurístico de alto nivel que controla las posibles soluciones a través de otro algoritmo de bajo nivel (otro heurístico). Burke y Kendall [14] lo definen como un algoritmo que elige entre los diferentes heurísticos para solucionar un problema de optimización.

1.6.- TIPOS DE METAHEURÍSTICOS

De entre los diferentes metaheurísticos que han obtenido más éxito en la resolución de problemas combinatorios, se pueden destacar los siguientes, indicando en cada caso los trabajos pioneros:

- Algoritmos Genéticos: Holland (1975).
- Recocido Simulado (Simulated Annealing): Kirkpatrick et al. (1983).

- Búsqueda Tabú (Tabu Search): Glover (1986), (1989), (1990).
- Algoritmos Meméticos: Moscato (1989).
- Path Relinking (Reencadenamiento de trayectorias): Glover (1989), (1994) y Glover y Laguna (1993).
- GRASP: Feo y Resende (1989), (1995).
- Búsqueda Reactiva: Battiti (1996).
- Colonia de Hormigas: Dorigo et al. (1996).
- Concentración Heurística: Rosing (1997) y Rosing y Revelle (1997).
- Scatter Search (Búsqueda Dispersa): Glover (1998) y Laguna (1999).
- Búsqueda Local Guiada: Voudouris y Tsang (1999).
- Búsqueda por Entornos Variables: Mladenovic (1995).
- Algoritmos de Estimación de distribuciones: Mühlenbein y Paaβ (1986).
- Algoritmos Bionómicos: Christofides (1994).
- Búsquedas Multiarranque: Los y Lardinois (1982) y Boender (1983).

Otras estrategias que pueden citarse:

- Cúmulo de Partículas (Particle Swarm Optimization): Kennedy y Eberhart (2001).
- Evolución Diferencial (DE): Store y Price (1997).
- Algoritmos Culturales: Reynolds (1994).
- Sistema Inmune Artificial: Nunes de Castro (2002).
- Búsqueda por ruido: Charon y Hudry (1993).
- Optimización Extrema: Boettcher y Percus (2000).
- Cross Entropy: Rubinstein (1997).

En el siguiente apartado se hace una breve descripción de cada uno de ellos:

1.6.1.- Algoritmos Genéticos

Encuadrados dentro de los algoritmos evolutivos, se inspiran en modelos de evolución biológica, ya que utilizan principios de selección natural para resolver problemas de optimización; fueron desarrollados por Holland [15] en los años 70.

A diferencia de otros métodos, en cada iteración del algoritmo no se tiene una solución, sino un conjunto de soluciones. El algoritmo se sustenta en el ciclo generar, seleccionar, combinar y reemplazar un conjunto de soluciones; suelen presentar mayores tiempos de computación ya que trabajan constantemente con un conjunto de soluciones. Aún así, y como se indica en Caballero et al. [16], *son menos sensibles a características geométricas de la frontera de Pareto como la concavidad, convexidad, continuidad, etc.*

El paradigma básico de los algoritmos genéticos es la utilización de operadores genéticos sobre cromosomas (codificación, normalmente binaria, de una solución, es decir, genotipo), a diferencia de otras estrategias evolutivas: Estrategias de Evolución, Programación Evolutiva y Programación Genética, que enfatizan los cambios a nivel de individuos y especies.

Según lo anterior, a cada solución o fenotipo, le corresponde un cromosoma o genotipo; sobre los diferentes individuos de una población inicial, actúa un operador de cruce que genera un conjunto de soluciones hijo En cada iteración, cada individuo o solución es evaluado mediante una función, teniendo mayor probabilidad de ser elegidos, aquellos individuos que presenten un mayor valor o *fitness*; en caso

de mejorar, sustituyen a las peores soluciones de la iteración actual; se incorpora, además, un operador de mutación, a fin de explorar nuevas zonas del espacio de soluciones.

Hay que resaltar que los algoritmos genéticos tienen una componente aleatoria significativa.

1.6.2.- Recocido Simulado

Este algoritmo fue desarrollado, inicialmente, por Kirkpatrick et al. [17]. Debe su nombre a las similitudes que presenta con el proceso físico conocido como recocido, en el cual, un material es calentado hasta la temperatura de austenización, para ser luego enfriado lentamente, a fin de mejorar algunas de sus propiedades.

Tal y como se indica en Caballero et al. [16] el recocido simulado presenta la virtud de poder escapar con facilidad de óptimos locales, lo que provoca su gran utilización en aplicaciones prácticas de los diferentes ámbitos de la Ciencia e Ingeniería.

Se suele decir que, aunque es muy fácil hacer que recocido simulado funcione, es difícil hacer que funcione bien; Dowsland y Adenso [18] indican que esto es debido a que no es propiamente un algoritmo, sino una estrategia heurística que necesita de varias decisiones para que quede totalmente diseñado.

El fundamento de este método se basa en el trabajo de Metropolis et al. [19] en el campo de la Termodinámica Estadística; básicamente, Metropolis modeló el proceso de recocido simulando los cambios energéticos de partículas, conforme disminuye la temperatura, hasta que converge a un punto.

Las leyes de la Termodinámica dicen que a una temperatura t, la probabilidad de un incremento energético de valor δE se puede aproximar a:

$$P(\delta E) = e^{(-\delta E / kT)}$$

Siendo k una constante física denominada constante de Boltzmann. En el algoritmo de Metropolis, se genera una perturbación aleatoria en el sistema, calculándose los cambios de energía resultantes; si hay una caída energética, el cambio se acepta automáticamente; por el contrario, si se produce un incremento, se acepta según una probabilidad dada por la expresión anterior. El proceso se repite hasta un número predeterminado de iteraciones en sentido decreciente de la temperatura, hasta que 'se enfría'.

Hay que indicar que, a efectos de optimización, la constante k no se considera; en cuanto a t, se puede decir que es un parámetro de control, denominado normalmente 'temperatura', según la analogía con el proceso físico. Según esto, una solución que suponga un empeoramiento en la función objetivo, se aceptará según la probabilidad $exp(-\delta E/t)$. Si t es bajo, será difícil que haya movimientos a peor y la función convergerá a un óptimo local; ahora bien, está demostrado que, si el enfriamiento es muy lento, según una función de reducción α, se converge a un óptimo global, aunque esto pueda conducir a tiempos de computación excesivamente largos. Aún así, diferentes trabajos demuestran que, empleando velocidades de enfriamiento grandes, los resultados obtenidos son bastante eficientes.

Anteriormente, se ha indicado que el recocido simulado es una estrategia heurística, que necesita de varias decisiones para que quede definido. Dichas decisiones se pueden clasificar en genéricas y específicas.

Las decisiones genéricas tienen que ver, básicamente, con los parámetros relacionados con el programa de enfriamiento: valores máximos y mínimos de la temperatura, la velocidad de reducción y

las condiciones de parada; más detalles se pueden encontrar en Aarts [20], en Aarts y Korks [21] y en Hajek [22].

Las decisiones específicas se refieren, principalmente, a la definición del espacio de soluciones, la estructura del entorno, la función objetivo y la solución inicial. Tal y como indica Hajek, la interactuación de estas variables influye de modo notable en la solución final.

Es muy amplio el campo de aplicación de esta metaheurística; así, en Vidal [23], se pueden encontrar aplicaciones en el Problema del Viajante, VRP y diseño de redes de telecomunicaciones.

1.6.3.- Búqueda Tabú

Tiene su origen en los trabajos de Glover [8] y es la mejor representante dentro de la conocida como Programación mediante Memoria Adaptativa [24], caracterizada por buscar en el entorno, utilizando información de los movimientos realizados con anterioridad. Los atributos de las soluciones ya visitadas son guardados en una lista (lista tabú) durante un determinado número de iteraciones, a fin de evitar ciclos. De esta forma, una solución posible es declarada tabú si alguno de sus atributos está en dicha lista; no obstante, existe un 'criterio de aspiración' que permite obviar esta prohibición bajo ciertas condiciones (por ejemplo, mejorar la mejor solución encontrada hasta el momento); asimismo se admiten movimientos que empeoren la solución actual, con el objetivo de escapar de óptimos locales.

Existen estrategias complementarias como son: la intensificación, que concentra la búsqueda dentro de una zona determinada, la diversificación que explora otras zonas no exploradas anteriormente, reencadenamiento de trayectorias que permite combinar soluciones, etc.

Otra estrategia también empleada es la oscilación estratégica. La base de esta estrategia consiste en no detener la búsqueda en un punto donde, según las reglas habituales, se debería hacer como, por ejemplo, en la frontera del conjunto de soluciones factibles. Se modifican las normas para avanzar a partir de ese punto hasta una profundidad determinada y se vuelve a retroceder en sentido contrario hasta alcanzar un nuevo punto de oscilación, desde donde volver a rebotar en sentido contrario, creando un movimiento oscilatorio que da nombre al método. Combina elementos de intensificación, al lado factible de la frontera, con elementos de diversificación, ya que de otra manera, el acceso a las nuevas regiones nos hubiera resultado imposible en un proceso corto de movimientos.

En resumen, sus aspectos más característicos son los siguientes:

- Básicamente es un algoritmo determinista, aunque admite variantes estocásticas.
- Realiza una exploración inteligente, basada en la información recogida durante la exploración.
- Utiliza memoria.
- Almacena los últimos movimientos, así como su frecuencia.

1.6.4.- Algoritmos meméticos

Los algoritmos meméticos son un conjunto de metaheurísticos caracterizados por fusionar ideas y principios derivados de los algoritmos evolutivos y la búsqueda por entornos. El adjetivo 'memético' deriva del inglés *meme* para designar al equivalente del gen en el contexto de la evolución cultural, aunque ello no suponga, bajo ningún concepto, que tenga que adherirse necesariamente a ese tipo de estrategias.

Según lo anterior, los algoritmos meméticos se asocian a una estrategia de búsqueda en la que una población de agentes optimizadores compiten y cooperan de manera sinérgica [25]; aún más, estos agentes hacen uso explícito de conocimiento sobre el problema que pretenden resolver [26].

A diferencia de otras técnicas de optimización, y como indica Cotta, *los algoritmos meméticos fueron concebidos explícitamente como un paradigma ecléctico y pragmático, abierto a la integración de otras técnicas,(metaheurísticas o no)* [27].

Es precisamente este eclecticismo el que permite crear las sinergias necesarias entre las diferentes metaheurísticas, dando a esta herramienta un potencial que es causa del éxito que está teniendo.

1.6.5.- Path Relinking

Ha sido tradicionalmente considerada como una estrategia dentro de la búsqueda tabú, si bien, en los últimos años ha alcanzado identidad propia. Su principio de funcionamiento es la exploración de un conjunto de soluciones en el camino entre pares de soluciones generadas previamente. Se pretende la incorporación de atributos de soluciones de alta calidad para crear una buena composición de atributos en la solución final. Más información se puede encontrar en Glover [28], [29] y Glover y Laguna [30].

1.6.6.- GRASP

Acrónimo de *Greedy Randomize Adaptive Search Procedure* o, en castellano, Procedimientos de Búsqueda basados en funciones Ávidas, Aleatorios y Adaptativos y dados a conocer en el trabajo de Feo y Resende [31].

GRASP es una técnica en la que cada iteración aporta una solución al problema sobre el que se está trabajando; la mejor solución se guarda como resultado final. El algoritmo se desarrolla en dos fases: en la primera, se construye de forma inteligente una solución mediante una función ávida y aleatoria (método multiarranque); en la segunda, de aplica un procedimiento de búsqueda local a la solución construida, a fin de encontrar una mejora; es considerado como un metaheurístico constructivo.

1.6.7.- Búsqueda Reactiva

La búsqueda reactiva [32] propone la integración de procedimientos de autoajuste de parámetro en algoritmos basados en búsqueda por entornos.

1.6.8.- Colonias de Hormigas

Este algoritmo está inspirado en el comportamiento de las colonias de hormigas. Éstas son capaces de determinar la ruta más corta en su camino de ida y vuelta entre la colonia y la fuente de alimento; esto es debido a que son capaces de transmitirse entre ellas la información, a través del rastro dejado por cada una en su trayectoria; dicho rastro o feromona varía de intensidad dependiendo de la cantidad de hormigas que transiten por el mismo. Según lo anterior, una hormiga que sale en busca de alimento se mueve de forma aleatoria, dejando un rastro; otras hormigas seguirán el rastro dejado por la primera y, en caso de encontrar alimento, volverán por el mismo camino a depositarlo, con lo que vuelven a impregnar el mismo con más feromona, incrementando la presencia de la misma (comportamiento autocatalítico); de esta forma, cuantas más hormigas sigan dicho trayecto, más atractivo se vuelve para ellas.

Observando dicho fenómeno, se infiere que se produce un *feedback* o retroalimentación, mediante el cual, la probabilidad de que una hormiga escoja una ruta, depende de la cantidad de hormigas que hayan transitado previamente por la misma.

La feromona depositada en cada ruta se 'evapora' según un parámetro $\rho \in [0,1]$, denominado coeficiente de evaporación, que establece el porcentaje de feromona que permanece de una iteración a otra, lo que provoca que, en el caso de transcurrir un periodo de tiempo sin usar una ruta determinada, ésta pierda interés para las hormigas y surge con el objetivo principal de evitar la convergencia prematura del algoritmo con el paso del tiempo.

El primer algoritmo basado en este comportamiento, aplicado al Problema del Viajante fue desarrollado por Dorigo et ál [33].

Gravel et al. [34] y Guntsch y Middendorf [35] lo adaptaron a la resolución de problemas de Programación Multiobjetivo obteniendo buenos resultados.

1.6.9.- Concetración Heurística

Es una estrategia propuesta por Rosing [36] que consta de dos fases: en la primera, se generan múltiples soluciones; en la segunda, con los elementos de las mejores soluciones, se construye el denominado conjunto de concentración, con lo que se resuelve el problema original, pero restringiendo la sección a los elementos del conjunto de concentración. En el clásico problema VRPTW aporta ligeras mejoras en comparación con otras heurísticas [37]

1.6.10.- Scatter Search

Introducido por Glover [38] y conocido en castellano como búsqueda dispersa*,* se basa en estrategias de combinación de reglas de decisión, especialmente en problemas de secuenciación y combinación de restricciones [39]. Dichas estrategias fueron expuestas en el congreso *Management Science and Engineering Management* celebrado en Austin, Texas, en septiembre de 1967.

Scatter Search (SS) trabaja sobre un conjunto de referencia, formado por diferentes soluciones, combinando éstas entre sí a fin de encontrar otras soluciones que mejoren las iniciales. Bajo este prisma, se podría considerar un método evolutivo, pero no lo es (en el concepto clásico que se tiene de éstos), ya que no se fundamenta en métodos aleatorizados, sino en elecciones sistemáticas sobre un conjunto pequeño.

Hay que indicar que los puntos que forman parte del conjunto inicial deben presentar puntos de alta calidad y puntos dispersos; de este conjunto, se generan sub-conjuntos que se combinarán entre sí para dar nuevas soluciones; por último, se suele aplicar un procedimiento de mejora local, ya sea por búsqueda local o, incluso, por búsqueda tabú.

Sus características principales son las siguientes:

- Usa un conjunto de referencia.
- Combina dos o más soluciones del conjunto de referencia; si son más de dos soluciones busca el objetivo de generar centroides.
- La versión reducida genera soluciones a partir de dos dadas.
- Al realizar la combinación, selecciona pesos apropiados y no al azar.
- Realiza combinaciones convexas y no convexas de las soluciones.
- Se da importancia a la distribución de los puntos, los cuales deben tomarse dispersos.

Habitualmente se suele completar el proceso mediante un procedimiento de búsqueda local a fin de mejorar las soluciones.

1.6.11.- Búsqueda Local Guiada

Este tipo de metaheurísticas se dieron a conocer a través de los trabajos de Voudouris y Tsang [40].

Es un procedimiento iterativo de búsqueda local que se basa en el uso de memoria; al finalizar cada una de las iteraciones, se modifica la función objetivo, penalizando determinados elementos que aparecen en el último óptimo local, estimulando de esta forma diversificación de la búsqueda y evitando, por tanto, incurrir en dichos óptimos locales.

En los trabajos de Kilby et ál [41], se puede encontrar una aplicación particular al VRPTW.

1.6.12.- Búsqueda por Entornos Variables

La búsqueda de entorno variable (*Variable Neighbourhood Search*, VNS) es una metaheurística reciente destinada a resolver problemas de optimización, cuya idea básica es el cambio sistemático de entorno dentro de una búsqueda local [42], [43], [44].

Se basa en el hecho sencillo de cambiar la estructura del entorno cuando el algoritmo se detiene en un óptimo local; existen gran cantidad de extensiones, aunque todas ellas persiguen el principio de mantener su simplicidad.

Se fundamenta en tres principios simples:

- Un mínimo local con una estructura de entornos, no lo es necesariamente con otra.
- Un mínimo global es un mínimo local en todas las estructuras de entornos.
- En muchos problemas los mínimos locales con la misma o distinta estructura de entornos, están relativamente cerca.

1.6.13.- Algoritmos de Estimación de Distribuciones

Son un conjunto de metaheurísticos encuadrados dentro de los procedimientos evolutivos.

Los algoritmos evolutivos, tal y como están configurados, evolucionan dependiendo de una serie de factores a saber: operadores de cruce, operadores de mutación, tamaño de la población, número de generaciones, etc. De no poseer una experiencia previa en la implementación del algoritmo, y tal y como observa Grefenstette [45], la determinación de los mismos se convierte, por sí mismo, en un problema de optimización.

Si a esto se une el hecho, como indican Larrañaga et al [46], de que la predicción de los movimientos de la población de individuos en el espacio de búsqueda sea extremadamente difícil, provoca la aparición de un nuevo tipo de algoritmos denominados algoritmos de estimación de distribuciones (en ingles *Estimation of Distribution Algorithms* (EDA), introducidos en el ámbito de la computación por Mülhlenbein y Paaβ en 1986 [47].

A diferencia de los algoritmos genéticos, los EDAs no precisan de operadores de cruce y mutación; la nueva población de individuos es obtenida mediante la simulación de una distribución de probabilidad estimada a partir de datos que contienen atributos de la población anterior.

Inicialmente y de forma aleatoria, generan una población de individuos; sobre esta población se generan tres tipos de operaciones: en primer lugar se genera un subconjunto de las mejores soluciones; en segundo lugar y, a partir de este sub-conjunto, se realiza un proceso de aprendizaje de un modelo de distribución de probabilidad y, en tercer lugar, se generan nuevos individuos simulando dicha distribución.

Información más exhaustiva sobre EDAs se puede encontrar en los trabajos de Larrañaga et al. [48], y Pelikan et al. [49].

1.6.14.- Algoritmos Bionómicos

Fueron introducidos por Christofides [50] y han sido aplicados a problemas logísticos y financieros. Se trata de una estrategia de optimización global evolutiva de búsqueda probabilística, similar a los algoritmos genéticos y a la búsqueda dispersa, que actualiza la población de soluciones en cada iteración, pero con una forma diferente en la búsqueda del espacio de soluciones.

En un primer momento, se genera una población inicial de soluciones —que pueden ser no factibles—, que puede ser generada de forma aleatoria. Después, se mejora cada solución de la población inicial. Para ello se calcula el valor de cada una de ellas considerada así como un vector que mide la 'no factibilidad' de la solución. Si es 'no factible' se aplica un algoritmo de optimización local buscando la factibilidad; si después de aplicar el algoritmo, la solución obtenida sigue siendo 'no factible' se pasa a otra solución. Si la solución sí 'es factible' lo que se busca es mejorar el valor de la misma con otro algoritmo de optimización local que mantenga la factibilidad. Se aplica una estrategia generacional, donde, a partir de un conjunto de padres (generación previa), se generan hijos con unos pasos específicos de madurez, mejora y crecimiento, usando una heurística específica o procedimientos genéricos de búsqueda local y una aproximación para la selección de padres, basada en la estructura de la población y la construcción de conjuntos de padres.

Dos características importantes comunes con la búsqueda dispersa son la posibilidad de tener un conjunto de soluciones de tamaño variable y el uso de múltiples padres. Para la selección de padres, utiliza un procedimiento de identificación basado en grafos.

Para generar nuevas soluciones, se busca un conjunto de padres de forma que tengan un gran índice de dominancia y que existan distinciones entre ellas. La definición de distinción entre soluciones depende del problema. A continuación, se usa un procedimiento que combina las soluciones padres para producir las soluciones hijos. Se calcula la dominancia de todos los hijos obtenidos, se elige el mejor y se mejora igual que en el caso de las soluciones de la población inicial, mediante dos procedimientos de optimización local: uno para buscar la factibilidad, si la solución no es factible; y otro, para mejorar el valor de la solución. Los empates en las preferencias se rompen de forma aleatoria. Se estudia si este hijo mejora a algunas soluciones existentes y se reemplaza la solución más dominada por este mismo hijo. El nuevo individuo también puede reemplazar alguna solución vieja; la definición de solución vieja también es dependiente del problema. Si no se dan ninguna de estas dos circunstancias se rechaza al hijo. Además debe incorporar algún criterio de finalización.

1.6.15.- Búquedas Multiarranque

En las búsquedas multiarranque se alterna una fase de generación de soluciones iniciales, con una fase de mejora de las mismas [51]. En la fase inicial, los procedimientos empleados pueden ser simples: generación aleatoria de soluciones iniciales o bien, complejos procedimientos para obtener soluciones de calidad. La misma consideración se puede hacer respecto a la segunda fase.

La repetición de los procesos generar solución inicial y búsqueda local constituye el primer método multiarranque descrito en la literatura [52] [53].

El algoritmo genera una nueva solución inicial cuando se llega a un óptimo local en la fase de mejora de la solución anterior.

Los métodos multiarranque han sido utilizados, principalmente, para resolver dos tipos de problemas de optimización: los problemas no lineales y los problemas combinatorios. En el campo de los primeros, se ha desarrollado principalmente la teoría sobre las condiciones de convergencia del método al óptimo global del problema. En el segundo caso se encuentran una gran cantidad de procedimientos heurísticos o aproximados para encontrar buenas soluciones en tiempos de computación relativamente pequeños.

Se puede encontrar más información en Glover [54], Martí [55] y Martí y Moreno Vega [51].

1.6.16.- Evolución Diferencial

Propuesta recientemente por Storn y Price [56], la Evolución Diferencial (ED) ha sido aplicada en problemas de optimización sobre dominios continuos. En cada variable de decisión se representa en el cromosoma como un número real (codificación real); al igual que en otros algoritmos evolutivos, la población inicial se genera de forma aleatoria. Para el proceso de selección, se eligen tres padres (uno de ellos, el principal) que generan un hijo. Este hijo se crea sumando al principal la diferencia de los otros dos padres.

1.6.17.- Cúmulo de partículas

Conocido en inglés como PSO (*Particle Swarm Optimization*), fue desarrollado por el psicólogo-sociólogo Jammes Kennedy y por el ingeniero electrónico Russell Eberhart [57] en 1995, fundamentándose en experimentos con algoritmos que modelaban el comportamiento en vuelo de algunas especies de pájaros, el comportamiento de los bancos de peces e, incluso, las tendencias sociales en el comportamiento humano.

La metáfora social que plantea este tipo de algoritmos se puede resumir de la siguiente forma: los individuos que son parte de una sociedad tienen una opinión influenciada por la creencia global compartida por todos los posibles individuos. Cada individuo, puede modificar su opinión (estado) dependiendo de tres factores: el conocimiento del entorno (su adaptación), los estados en la historia por los que ha pasado el individuo (su memoria) y estados en la historia de los individuos cercanos (memoria del vecindario).

En el algoritmo PSO, cada individuo, llamado partícula, se va moviendo en un espacio multidimensional que representa su espacio social. Cada partícula tiene memoria mediante la que conserva parte de su estado anterior. Cada movimiento de una partícula es la composición de una velocidad, que inicialmente es aleatoria, y dos valores ponderados aleatoriamente: individual (la tendencia de las partículas de preservar su mejor estado anterior) y social (la tendencia a moverse hacia vecinos con mejor posición). Debido a su planteamiento, este tipo de algoritmo se adapta muy bien a problemas matemáticos de carácter continuo, aunque también se ha aplicado con gran éxito en tareas de naturaleza discreta.

El cúmulo de partículas es un sistema multiagente, es decir, las partículas son agentes simples que se mueven por el espacio de búsqueda y que guardan (y posiblemente comunican) la mejor solución que han encontrado; cada partícula tiene un *fitness*, una posición y un vector velocidad que dirige su movimiento. El movimiento de las partículas por el espacio está guiado por las partículas óptimas en el momento actual.

Dentro de la programación multiobjetivo, Santana et al. [58] ha presentado recientemente un híbrido de esta estrategia con búsqueda dispersa que amplía el conjunto de soluciones no dominadas.

1.6.18.- Algoritmos Culturales

Fueron desarrollados por Robert G. Reynolds [59] como un complemento a la metáfora que usan los algoritmos de computación evolutiva, que se habían concentrado en conceptos genéticos y de selección natural.

Los algoritmos culturales están basados en las teorías de algunos sociólogos y arqueólogos, que han tratado de modelar la evolución cultural; tales investigadores indican que ésta puede ser vista como un proceso de herencia en dos niveles: el nivel micro-evolutivo, que consiste en el material genético heredado por los padres a sus descendientes, y el nivel macro-evolutivo, que es el conocimiento adquirido por los individuos a través de las generaciones, y que una vez codificado y almacenado, sirve para guiar el comportamiento de los individuos que pertenecen a una población.

La cultura puede verse como un conjunto de fenómenos ideológicos compartidos por una población [60], pero por medio de los cuales, un individuo puede interpretar sus experiencias y decidir su comportamiento. En estos modelos se aprecia muy claramente la parte del sistema que es compartida por la población: el conocimiento, recabado por miembros de la sociedad, pero codificado de tal forma que sea potencialmente accesible a todos. De igual manera se distingue la parte del sistema que es individual: la interpretación de ese conocimiento codificado en forma de un conjunto de símbolos, y los comportamientos que trae como consecuencia su asimilación; también la parte individual incluye las experiencias vividas, y la forma en que éstas pueden aportar algo al conocimiento compartido.

Reynolds intenta captar ese fenómeno de herencia doble en los algoritmos culturales. El objetivo es incrementar las tasas de aprendizaje o convergencia, y de esta manera, que el sistema responda mejor a un gran número de problemas.

Los algoritmos culturales operan en dos espacios. Primero, el espacio de la población, como en todos los métodos de computación evolutiva, en el que se tiene un conjunto de individuos. Cada individuo tiene una serie de características independientes de los otros, con las que es posible determinar su aptitud. A través del tiempo, tales individuos podrán ser reemplazados por algunos de sus descendientes, obtenidos a partir de un conjunto de operadores aplicados a la población.

El segundo espacio es el de creencias, donde se almacenarán los conocimientos que han adquirido los individuos en generaciones anteriores.

La información contenida en este espacio debe ser accesible a cualquier individuo, que puede utilizarla para modificar su comportamiento. Para unir ambos espacios se establece un protocolo de comunicación que dicta las reglas sobre el tipo de información que se debe intercambiar entre los espacios.

1.6.19.- Sistema Inmune Artificial

Este metaheurístico utiliza como punto de partida el sistema inmune y se apoya en la característica de ser un sistema de aprendizaje distribuido.

Una de las principales tareas del sistema inmune es mantener al organismo sano; algunos microorganismos llamados patógenos, que invaden al organismo, pueden resultar dañinos para éste. Los antígenos son moléculas que se encuentran expresadas en la superficie de los patógenos que pueden ser reconocidos por el sistema inmune y que además son capaces de dar inicio a la respuesta para eliminarlos.

Esta respuesta defensiva del sistema inmune presenta interesantes características desde el punto de vista del procesamiento de información. Es por ello que se ha usado como inspiración para crear soluciones alternativas a problemas complejos del campo de la Ingeniería y la Ciencia. Ésta es una área relativamente nueva a la que se denomina Sistema Inmune Artificial. Las primeras aplicaciones se pueden atribuir a Nunes de Castro y Timmis [61]. Respecto a la Programación Multiobjetivo, los primeros intentos de resolver problemas multiobjetivo se pueden encontrar en Coello y Cruz-Cortés [62].

1.6.20.- Búsqueda por Ruido

Básicamente, este método, diseñado por Charon y Hudry [63], consiste en: definir una vecindad, partir de una solución inicial, generar nuevas soluciones mediante la introducción de algún tipo de perturbación de datos (ruido), tomar la mejor como solución actual y repetir el ciclo. Según aumentan las iteraciones, disminuye la perturbación hasta llegar a cero.

1.6.21.- Optimización Extrema

Este método se inspira en procesos selectivos propios de la naturaleza; la evolución progresa de forma que son las especies mejor adaptadas al medio ambiente las que se reproducen con mayor facilidad y elimina componentes extremos indeseables que son los que conducen a la extinción [64].

La idea principal se sustenta en la eliminación sucesiva de elementos no deseados en las soluciones intermedias [65]; dichos elementos quedan caracterizados por un valor de adaptación $\lambda \in (0,1)$. Según esto, las especies más débiles y las más dependientes de éstas, son seleccionadas para cambios adaptativos actualizando su valor λ. El método actúa sobre una única solución (a diferencia de los algoritmos genéticos que actúan sobre el conjunto).

1.6.22.- Cross Entropy

Cross Entropy (CE) tiene su origen en un algoritmo adaptativo utilizado en la estimación de probabilidades de eventos raros en redes estocásticas complejas, desarrollado por Rubinstein [66]. Fue el mismo autor quien, más adelante [67], observó que podía ser aplicado no solo a estimaciones de probabilidad, sino también a problemas de optimización combinatoria.

Existen problemas en los que no se conoce el valor de la función objetivo debido a factores de ruido o al carácter estocástico del sistema [68]; una forma habitual de estimar dicho valor es la simulación. *Cross Entropy* proporciona un método simple y eficiente para resolver este tipo de problemas.

Cross Entropy es un proceso iterativo en el que cada iteración se compone de dos pasos:

Paso 1. Generación de un conjunto aleatorio de datos según un mecanismo determinado.

Paso 2. Actualización de los parámetros de generación aleatoria basada en datos, a fin de generar un mejor conjunto aleatorio de datos.

Un aspecto interesante es que *Cross Entropy* provee un método unificado en problemas de optimización y simulación.

2.- METAHEURÍSTICOS EN PROBLEMAS MULTIOBJETIVO

2.- METAHEURÍSTICOS EN PROBLEMAS MULTIOBJETIVO

2.1.- INTRODUCCIÓN

Son muchos los problemas en la vida real en los que es necesario decidir teniendo en cuenta dos o más criterios, es decir, optimizar de forma simultanea cada uno de dichos criterios; se debe realizar, por tanto, un análisis para determinar que alternativa o alternativas son las más adecuadas para este problema.

Hay que indicar que, por lo general, estas funciones entran en conflicto, por lo que, la primera decisión a tomar suele ser cuál de los dos criterios es más relevante puesto que, ahora, ya no existe una función objetivo, sino un vector de funciones objetivo. Es poco probable que una solución optimice todos los criterios a la vez, siendo difícil discernir qué solución es mejor que otra por cuanto no existe un orden total.

Según lo anterior, se entiende por Programación Multiobjetivo (PMO) la parte de la programación matemática que se ocupa de este tipo de problemas. Pueden ser formulados bajo la siguiente expresión:

$$Maximizar\ [f_1(x), f_2(x), \ldots f_p(x)]$$

$$Sujeto\ a:\ x \in X$$

donde:

$x = (x_1, x_2 \ldots x_n)$ son las variables de decisión.
X es el conjunto de puntos factibles.
f_i corresponde a cada uno de los criterios u objetivos.
$f = (f_1, f_2, \ldots, f_p)$ es la función vectorial objetivo.

Dado el conflicto existente entre los diferentes criterios que componen el vector, es necesario introducir un nuevo concepto que ayude a resolverlo. Dicho concepto es la optimalidad, desarrollado por Wilfredo Pareto en 1896 [69] y que considera que el óptimo ha de ser aquel para el cual no se pueda mejorar un objetivo sin empeorar los demás.

Matemáticamente, se define el *orden de Pareto* de la siguiente forma:[2]

Dados dos puntos y, y' $\in R^p$, se dice que el punto y es preferido a y' si se verifica que $y_i \geq y_i'$ $\forall$ i=1,... p, y existe al menos un j $\in$ {1,..., p} tal que $y_j > y_j'$ (cuando y se prefiere a y', se denota $y \succ y'$)

Es claro que el orden de Pareto es un orden parcial, por lo que no se puede decir que un punto determinado es la elección óptima porque es preferido a los demás, por lo tanto, es necesario desarrollar otro concepto que, de alguna forma, generalice el concepto de optimalidad y que permita indicar, en un problema multiobjetivo, aquellos puntos que constituyan una elección adecuada.

Este principio deriva del orden de Pareto y es lo que se conoce como 'eficiencia paretiana'.

[2] A efectos de descripción del problema, se considera un problema de maximización del vector objetivo, sin pérdida de generalidad del problema.

Un punto $x^* \in X$ es eficiente si no existe un punto $x \in X$ tal que f(x) es preferido a f(x) según el orden de Pareto, es decir, $x^* \in X$ es eficiente si no existe un punto $x \in X$ tal que $f_i(x) \geq f_i(x^*)$ $\forall$ i=1,2, ...p, con al menos un $j \in \{1, ...p\}$ tal que $f_i(x) > f_i(x^*)$.*

Del párrafo anterior, se deduce lo ya expresado*: un punto x* es un óptimo de Pareto si no existe otro punto factible $x \in X$ que mejore algún objetivo sin empeorar alguno de los demás.* Ahora bien, este concepto, generalmente, proporciona más de una solución; a este conjunto de soluciones se le denomina 'Conjunto de óptimos de Pareto'.

Es frecuente, por otra parte, que este conjunto de puntos sea bastante grande, lo que dificulta al decisor, la elección de una solución por el conflicto existente entre los diferentes criterios; llegados a este punto, existen diferentes posibilidades:

Una primera posibilidad es que el decisor aporte una información mínima y que espere a ver el conjunto eficiente obtenido para acabar de aplicar sus preferencias en una segunda fase, apareciendo así las denominadas 'Técnicas Generadoras del Conjunto Eficiente'.

Otra posibilidad, consiste en no decantarse explícitamente por una u otra solución, intentando encontrar un punto de equilibrio entre los diferentes criterios; se habla entonces de la Programación Compromiso.

Si en una primera actuación, el decisor incorpora información al proceso mediante el establecimiento de los denominados niveles de aspiración para los diferentes objetivos, aparece la Programación por Metas.

En general, se puede afirmar que el común de todas estas técnicas está en la necesidad de encontrar un conjunto de puntos eficientes suficientemente amplio y representativo como para poder encontrar en él una alternativa que se ajuste a las preferencias del decisor. Para más información, se puede recurrir a los trabajos de Steuer [70].

2.2.- ALGORITMOS APLICADOS EN PROGRAMACIÓN MULTIOBJETIVO

Tal y como se indica en Caballero et ál, [71] la mayor parte de las técnicas exactas para la resolución de este tipo de problemas están diseñadas para problemas lineales, tanto continuos como enteros, y su aplicación a problemas con algún tipo de dificultad (número elevado de variables enteras o binarias, carácter no lineal de alguna de las funciones o restricciones, carácter estocástico de alguno de los elementos del problema, etc.) dificultan de forma notable la resolución del problema; por otra parte, estas dificultades son normales en problemas reales de programación multiobjetivo.

El desarrollo de las técnicas metaheurísticas ha supuesto un gran avance en la resolución de dichos problemas; más información se puede encontrar en el trabajo de Jones et al. [72].

A grandes rasgos se pueden hacer dos grandes grupos: algoritmos basados en poblaciones, cuyo representante más destacado son los algoritmos genéticos, y los basados en búsquedas por entornos, o búsquedas vecinales. Ahora bien, recientemente, se han presentando híbridos inspirados en la biología, evolución de las especies que están siendo aplicadas con gran éxito.

Jones, Mirrazavi y Tamiz [72] estiman que, aproximadamente el 70% de las metaheurísticas aplicadas en PMO se basan en algoritmos evolutivos, un 24% en recocido simulado y un 6% en búsqueda tabú.

2.3.- ALGORITMOS BASADOS EN BÚSQUEDAS POR ENTORNOS

Estos métodos tienen en común el uso de una operación básica denominada movimiento, consistente en la modificación de atributos o características de una solución, a fin de crear un conjunto de posibles soluciones alternativas denominadas vecindario, entre las que se elegirá la que que mejore a las demás, según unas reglas definidas previamente.

2.3.1.- Recocido Simulado

Como se ha descrito anteriormente, fue introducido por Kirkpatrick et al. en 1983, y es conocido en inglés como *Simulated Annealing*; este metaheurístico ha demostrado ser una herramienta bastante eficaz en una amplia gama de problemas de optimización combinatoria.

En el campo de la programación multiobjetivo, recocido simulado se empleó por primera vez en el trabajo de Serafíni [73], donde la idea principal para la aplicación de esta metaheurística a la programación multiobjetivo, es la utilización de un criterio de aceptación de soluciones de peor calidad basado en la relación de dominancia entre dos soluciones dadas.

Otro planteamiento consiste en utilizar funciones agregativas basadas en pesos e ir variando de forma adecuada dichos pesos; en Teghem y Ulungu [74] [75] se pueden encontrar ejemplos. Ponce y Matos [76] lo utilizan en la resolución de programación multiobjetivo aplicada a distribución y planificación de redes. Otros trabajos se pueden encontrar en Erhgott y Gandibleux [77] y en Jones et al. [72].

2.3.2.- Búqueda Tabú

En cuanto a su aplicación en problemas multiobjetivo, las diferentes técnicas que se han desarrollado, se centran en algún tipo de agregación de los diferentes criterios que forman parte del problema, a fin de convertirlos en un problema mono-objetivo, resuelto mediante una búsqueda tabú; así, en Dahl y otros [78] se genera una familia de vectores de pesos; para cada uno de ellos se resuelve una función mono-objetivo resultado de incluir de forma ponderada cada uno de los criterios; En Hertz et al. [79] se resuelve un conjunto de funciones mono-objetivo considerando cada vez una de las funciones objetivo junto a una función de penalización.

En Gandibleux et al. [80] se utiliza una función escalarizada de logro, apoyada en las funciones objetivo del problema, como guía para la búsqueda tabú.

Otro método denominado MOAMP (*MultiObjective Adaptative Memory Procedure*), desarrollado por Caballero et al. [81], trata de adaptar una búsqueda tabú a un conjunto eficiente de un problema multiobjetivo; tal y como comenta *...es un hecho conocido que los puntos eficientes de un problema multiobjetivo, se encuentran 'conectados' entre sí...* es decir, en el caso de ser variables continuas y el problema cumple unas mínimas condiciones de continuidad, cualquiera de los puntos eficientes, están conectados por una curva dentro del conjunto eficiente y, en el caso de variables enteras, cualquier punto está 'suficientemente cerca' de otro punto eficiente. Es precisamente este principio de proximalidad, la propiedad que se intenta rentabilizar. Según lo anterior, este método trata de generar, mediante una búsqueda tabú, un conjunto eficiente inicial de puntos y, mediante un proceso de intensificación una aproximación al resto. Otros autores que han trabajado sobre este tema son Hansen [82], Ben Abdelaziz et al. [83], Gandibleux y Freville [84], Alves y Clímaco [85], Al-Yamani et al. [85] y en Ehrgott et al. [86].

2.3.3.- GRASP

Tal y como ya se ha citado anteriormente, es un acrónimo de *Greedy Randomize Adaptive Search Procedure* o, en castellano, procedimientos de búsqueda basados en funciones ávidas, aleatorios y adaptativos.

GRASP es un método interactivo en el que cada iteración aporta una solución al problema sobre el que se está trabajando; la mejor solución se guarda como resultado final. Es considerado como un metaheurístico constructivo, fácil de implementar; en cuanto a problemas multiobjetivo, se pueden encontrar trabajos en Higgins et al. [87] y en Gandibleux et al. [88].

2.4.- ALGORITMOS BASADOS EN POBLACIONES: ALGORITMOS EVOLUTIVOS

En este campo, la mayor parte de las aplicaciones, se engloban dentro de los algoritmos genéticos; los primeros trabajos, tal y como indica Goldberg [89], se centran en el uso de funciones agregativas de criterios mediante una escalarización de los mismos; más adelante Wilson y Macleod [90] plantean un enfoque de programación por metas ponderadas.

A fin de evitar las dificultades inherentes a la agregación de criterios, Schaffer [91], presentó en la Primera Conferencia Internacional de algoritmos genéticos, en 1985, el método VEGA (*Vector Evaluated Genetic Algorithms*) basado en *rankings*, cuya única diferencia con un algoritmo genético usual, es la forma en la que se realiza la selección para la reproducción.

En este caso, en cada iteración, la población se agrupa en cierto número de sub-poblaciones (tantas como criterios) atendiendo en cada una de ellas al valor de una de las funciones objetivos. Dentro de cada una de estas poblaciones, se realiza la selección atendiendo al valor de ese criterio y, a continuación, las poblaciones se mezclan de nuevo para aplicar los operadores de cruce y de mutación, dando lugar a la siguiente generación.

Como se indica en Caballero et al. [16] VEGA presenta varios problemas entre los que cabe destacar su incapacidad para retener buenas soluciones eficientes (aquéllas que, sin ser las mejores en ninguno de sus criterios, representan una buena solución compromiso entre ellos); aunque se han presetado mejoras, sigue siendo incapaz de incorporar una solución que contenga la dominancia de Pareto.

Existe otro tipo de algoritmo, basado en el ordenamiento lexicográfico [92], que ordena las funciones objetivo según la importancia de cada objetivo, antes de iniciar el proceso de búsqueda de soluciones, empezando por el más importante y procediendo según el orden de importancia asignado a cada uno de los objetivos. Este método es adecuado sólo cuando la importancia de cada objetivo es conocida y puede ser determinada, lo cual no suele ser normal.

Más tarde, aparecen otro conjunto de algoritmos basados en el orden de Pareto; según este enfoque, el valor de cada individuo no depende del valor de sus criterios, sino de su dominancia dentro de cada población. Según lo anterior, la idea es encontrar individuos que no estén dominados por ningún otro, extraerlos del conjunto y volver a repetir el proceso con el resto de individuos; al final del proceso, todos los individuos tienen una posición en el ranking, pudiéndose realizar una selección por ranking. Este enfoque resulta superior a otros, tal y como se puede ver en Hilliard [93].

Dentro de la primera generación de estos algoritmos, se pueden citar:

> NSGA: (*Non-Dominated Sorting Genetic Algorithm*) desarrollado por Srinivas y Deb [94] en 1994, se fundamenta en la clasificación de los puntos en función de su 'no dominación', antes de seleccionar a los individuos. De esta forma, a todos los puntos no dominados de la

población, se les asigna un mismo valor de aptitud, proporcional al tamaño de la población. Una vez eliminados estos, se repite el proceso hasta la total clasificación de los puntos. Este paso provoca dos acciones: en primer lugar, aquellos puntos con mayor clasificación, tienen mayor probabilidad de reproducirse en la siguiente generación y, en segundo lugar, la existencia de conjuntos de puntos *(niching)* que comparten el mismo valor de la función de aptitud garantiza la diversidad del conjunto de puntos eficientes.

NPGA: (*Niched Pareto Genetic Algorithm*), propuesto por Horn y Nafpliotis en 1993 [95], usa un esquema de selección por torneo, basado en la dominancia de Pareto. Según este planteamiento, se toman dos individuos de forma aleatoria, del conjunto de la población, que son comparados con un subconjunto de la misma; si uno de ellos es dominado y el otro no, por el sub-conjunto de la población, el no dominado gana; en el resto de casos: ganan los dos o pierden los dos; el resultado del torneo se decide a través de *fitness sharing* o método de proporción (persiguiendo la formación de subconjuntos de elementos vecinos en la población, llamados 'nichos', reduciendo la aptitud de los individuos por la presencia de otros muy parecidos).

MOGA: (*Multiobjective Genetic Algorithm*), en este algoritmo, el ranking de cada individuo se corresponde con el número de cromosomas en la población actual, por los cuales es dominado. Si se considera un individuo x_i en la iteración t que es dominado por $p_i^{(t)}$ individuos de la generación actual, el ranking de x_i viene dado por $rank(x_i, t) = 1 + p_i^{(t)}$. De esta forma, los individuos no dominados tiene un ranking igual a 1, mientras que los dominados son penalizados según el número de individuos en sus correspondientes regiones.

Una segunda generación, introduce el concepto de *elitismo*. Dicho concepto se refiere, dentro del contexto de la optimización multiobjetivo, al uso de una población externa que almacena los individuos no dominados encontrados hasta esta generación y que participa en el proceso de selección. También puede ser introducido a través del uso de (μ+λ)-selección en la cual μ padres compiten con λ hijos y, aquellos que no son dominados, se seleccionan para la siguiente generación.

Según Caballero [16], los más representativos son los siguientes:

SPEA: Acrónimo de *Strength Pareto Evolutionary Algorithm* y desarrollado por Zitle y Thiele en 1999 [96]; difiere de los anteriores en varios aspectos. En primer lugar, utiliza dos poblaciones, incorporando el concepto de elitismo a través del almacenamiento de las soluciones no dominadas en una población externa, la cual participa del proceso de selección. Además, el cálculo de selección se realiza utilizando un procedimiento basado en la asignación de un valor de fuerza (Strength) a todos los elementos de la población externa. Este procedimiento introduce la formación de nichos a partir del concepto de dominancia Pareto, llamado niching por strength. Dado que el conjunto de soluciones en la población externa puede ser grande y ésta interviene en el proceso evolutivo, también se utiliza un procedimiento de clustering para reducir el número de soluciones de dicho conjunto preservando la diversidad en la población.

SPEA 2: Posteriormente, aparece SPEA 2 [97], que presenta tres diferencias respecto al anterior: en primer lugar presenta una estrategia de asignación de aptitud de 'grano fino' (para cada individuo se tiene en cuenta en número de individuos que domina y el número de individuos por el que es dominado); en segundo lugar, utiliza una técnica para la estimación de densidad de su vecindario, que guía la búsqueda más eficientemente y, en tercer lugar,

incorpora un método para truncar el archivo, garantizando el mantenimiento de soluciones extremas.

PAES: Propuesto por Knowles y Corne en 2000 [98], este algoritmo denominado *Pareto Archived Evolution Strategy* incorpora una estrategia del tipo (1+1)-PAES (un solo padre genera un solo hijo, compitiendo entre ellos), junto con una búsqueda local y la utilización de un archivo histórico (elitismo) que almacena soluciones no dominadas encontradas a lo largo de la ejecución del algoritmo.

NSGA-II: Deb et al. [99] solventan parte de los problemas de la versión original con el algoritmo NSGA-II. Es más eficiente, usa elitismo y un operador de comparación (*crowding*) en función de la proximidad de soluciones alrededor de cada uno de los puntos de la población. No usa, como su predecesor, una población externa, pero su mecanismo de selección consiste en la selección de los mejores padres con los mejores hijos, obtenidos por selección del tipo ($\mu+\lambda$).

Controlled Elitist NSGA-II: Difiere del anterior solamente en que el número de individuos pertenecientes al mejor frente actual de no-dominados es seleccionado de forma adaptativa.

NPGA2: Este algoritmo fue introducido por Erickson et al. [100] y es una revisión del algoritmo NPGA que incorpora el ranking de Pareto, pero mantiene la selección por torneo. No utiliza memoria externa y mantiene un mecanismo de elitismo similar al NSGA-II.

2.5.- OTRAS TENDENCIAS

Existen, en la actualidad, nuevos algoritmos hibridando los dos anteriormente citados e intentando adaptar a la programación multiobjetivo, metaheurísticos que están dando buenos resultados en Programación Mono-objetivo. Entre otros, se pueden citar:

2.5.1.- Búqueda Dispersa

Respecto a la programación multiobjetivo, hasta el momento no hay excesivas aplicaciones usando búsqueda dispersa; entre otros, se puede citar los trabajos de Molina et al. [101] que aplican SS a la resolución de problemas de optimización multiobjetivo no lineales; También se pueden encontrar en Beausoleil [102].

En cuanto a los procedimientos de mejora, en trabajos recientes, Pérez et al. [103], reemplazan la búsqueda local aplicada en la mejora de las soluciones, por un mecanismo de mejora guiado por direcciones proporcionadas por los vectores gradiente de las funciones objetivo.

2.5.2.- Evolución Diferencial

Se puede encontrar, dentro de la literatura, varias extensiones de la Evolución Diferencial (ED) en Problemas Multiobjetivo; entre otros, se pueden destacar PDE [104], que maneja una única población; en la reproducción toma únicamente soluciones no dominadas y se usa una métrica de distancia para favorecer la diversidad. PDEA [105] que combina la DE con distintos elementos de la NSGA-II; VEDE [106], inspirado en VEGA que maneja múltiples poblaciones manejadas en paralelo y DEMORS [107] donde se combina la evolución diferencial con *Rough Sets*, herramienta de la Inteligencia Artificial.

3.- BÚSQUEDA TABÚ

3.- BÚSQUEDA TABÚ

3.1.- INTRODUCCIÓN

Tal y como indican Melián y Glover [108], *...la búsqueda tabú es un procedimiento metaheurístico cuya característica distintiva es el uso de memoria Adaptativa y de estrategias especiales de resolución de problemas...* Su planteamiento se apoya en la idea de explotar diversas estrategias inteligentes aplicadas a la resolución de problemas, basadas en procedimientos de aprendizaje.

Los primeros trabajos sobre búsqueda tabú (también conocido como *Tabú Search, TS*) se deben a Glover, [28] [109] aunque los principios básicos ya fueron citados en 1986 por el mismo autor [8]. En dicho trabajo, hace alusión al término *tabú* como referencia a *…un tipo de inhibición a algo debido a connotaciones culturales o históricas y que puede ser superada en ciertas condiciones*…

La búsqueda tabú se puede considerar como una extensión de una búsqueda local, en la que se permiten movimientos que, provisionalmente, empeoran la solución, a fin de extender el espacio de búsqueda a zonas fuera de los óptimos locales, que puedan incluir un óptimo global; además de lo anterior, se establecen los últimos movimientos como movimientos *tabú* a fin de evitar problemas de ciclabilidad, aunque tal y como indican Melián y Glover, el fin último es *...continuar estimulando el descubrimiento de soluciones de alta calidad*. Una descripción de la estrategia en pseudocódigo, puede ser la siguiente [110]:

PROCEDIMIENTO DE BÚSQUEDA TABÚ BÁSICO

Seleccionar una solución inicial s_0, $s^* = s_0$
Inicializar $T = \varnothing$
Repetir
- Determinar $N^*(s_0) \subset N(s_0)$
- Seleccionar $f(s') = min \{f(s) / s \in N^*(s_0)\}$
- *Si* $f(s') < f(s^*)$ entonces $s^* = s'$
- Hacer $s_0 = s'$
- Actualizar T

Hasta alcanzar condición de parada.

---

donde $N(s)$ es el entorno; $N^*(s)$ es el entorno modificado y T es el conjunto de soluciones tabú.

Hay que destacar que, a diferencia de otros métodos, la búsqueda tabú, en la mayoría de sus versiones, elimina el carácter estocástico del procedimiento, al incorporar estrategias de búsqueda inteligentes y sistemáticas, por lo que se considera que es principalmente 'determinista'.

El marco de la memoria adaptativa empleada en búsqueda tabú, no solo explota la historia del proceso de resolución del problema, sino que exige la creación de estructuras para hacer posible tal exploración; según lo anterior, los movimientos prohibidos lo son en virtud de la confianza depositada en una memoria evolutiva, que permite modificar las características de dicho movimiento en función del número de iteraciones, estado u otras circunstancias.

Es necesario, a fin de poder evaluar la calidad de la solución analizada, definir una función objetivo que puede ser función de diferentes parámetros a optimizar, a saber: coste, tiempo, distancia, orden, etc. según esto, en cada iteración, una solución elegida dentro del entorno de la solución actual, es analizada mediante dicha función objetivo y, si mejora la solución y no es tabú o, siendo tabú, cumple un criterio

de aspiración, sustituye a la solución inicial, se actualiza el entorno de las soluciones candidatas y se repite el ciclo; así sucesivamente, hasta que se cumple una condición de parada.

Puede establecerse una analogía con el modo de actuar de un escalador en su búsqueda por encontrar la cumbre [111]; desde una determinada posición debe decidir la dirección a tomar a fin de encontrar una posición, cercana a la actual, que le permita seguir avanzando hacia la cumbre. La decisión tomada, dependerá de su experiencia y de la observación del entorno; para cada una de las posibles rutas que pueda analizar desde su posición actual, deberá evaluar la ventaja obtenida y, en función de este último parámetro, decidir el camino. No es infrecuente que, llegado a un punto, compruebe que no puede continuar por el mismo, hacia el objetivo final, teniendo que volver a descender a lugares ya visitados y tomar otra dirección. Dicho escalador tendrá, además, que recordar las rutas descartadas a fin de no volver a caer en las mismas, incurriendo en trayectorias cíclicas.

El ejemplo anterior ilustra, con bastante precisión, los diferentes aspectos que intervienen dentro una búsqueda tabú: movimientos tabú, que se fijan como tal a fin de evitar problemas de ciclabilidad; intensificación, consistente en explorar soluciones vecinas a una solución dada; diversificación que conduce la búsqueda a un nuevo espacio de soluciones no encontrado anteriormente y criterios de aspiración, mediante los que se pueden realizar movimientos tabú bajo ciertas condiciones.

Se puede indicar, por tanto, que la búsqueda tabú descansa en dos estrategias que se alternan secuencialmente: la intensificación y la diversificación; el uso que se hace de una u otra depende, fundamentalmente, de la información guardada en la memoria.

De todo lo citado anteriormente cabe inferir que, en definitiva, para clasificar un metaheurístico como inteligente, es necesario que incorpore memoria adaptativa así como una exploración sensible. Respecto a la primera cabe resaltar este matiz para distinguirla de otras metaheurísticas que confían en procesos aleatorios, como pueden ser recocido simulado o algoritmos genéticos o también diseños muy rígidos en la memoria, como *Branch and Bound*. En cuanto a la segunda, el énfasis en la exploración sensible se deriva de la suposición de que una mala elección estratégica puede aportar más información que una buena elección al azar.

3.2.- INTENSIFICACIÓN Y DIVERSIFICACIÓN

En muchas aplicaciones de búsqueda tabú, el procedimiento básico puede ser reforzado con estrategias de intensificación y diversificación. La intensificación (figura 2) consiste en explorar con más detalle las regiones donde se encuentran las mejores soluciones descubiertas hasta ese momento mediante la modificación de las reglas de selección. La diversificación (figura 3) se realiza después de la intensificación y consiste en dirigir la búsqueda a regiones no exploradas y que, por tanto, difieren significativamente de las soluciones ya evaluadas.

3.3.- USO DE MEMORIA

Las estructuras de memoria en la búsqueda tabú actúan en cuatro direcciones principales: proximidad en el tiempo, frecuencia, calidad e influencia; las memorias basadas en lo reciente y en lo frecuente se complementan entre sí y se emplean en las estructuras de memoria a corto y a largo plazo respectivamente. La calidad se refiere a la capacidad de diferenciar entre buenas y malas soluciones (en términos de función objetivo), reforzando aquellos movimientos que conducen a buenas soluciones, y penalizando aquellos que llevan a malas soluciones. Por último, la influencia, que considera el alcance de las elecciones realizadas durante la búsqueda, no solo en lo referido a la calidad de las soluciones, sino también en lo que se refiere a su estructura.

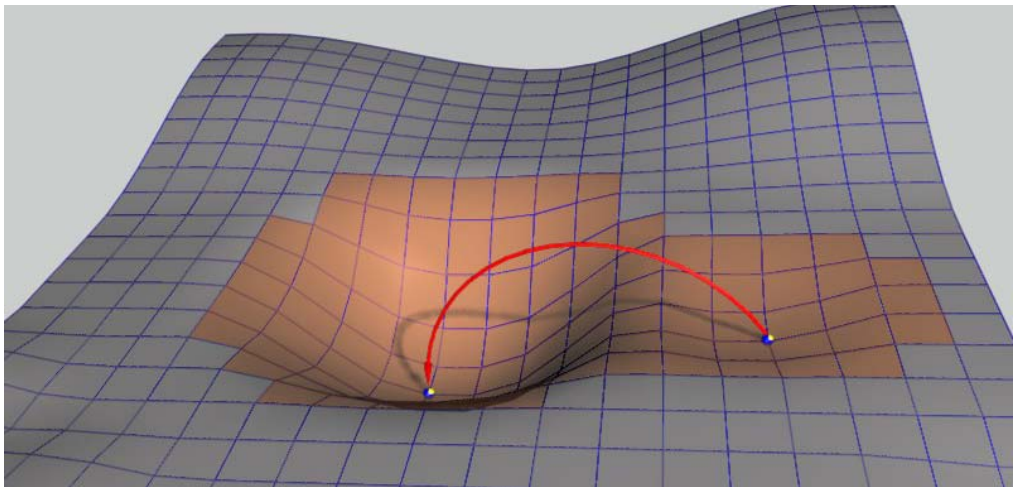


Figura 2. Estrategia de Intensificación.

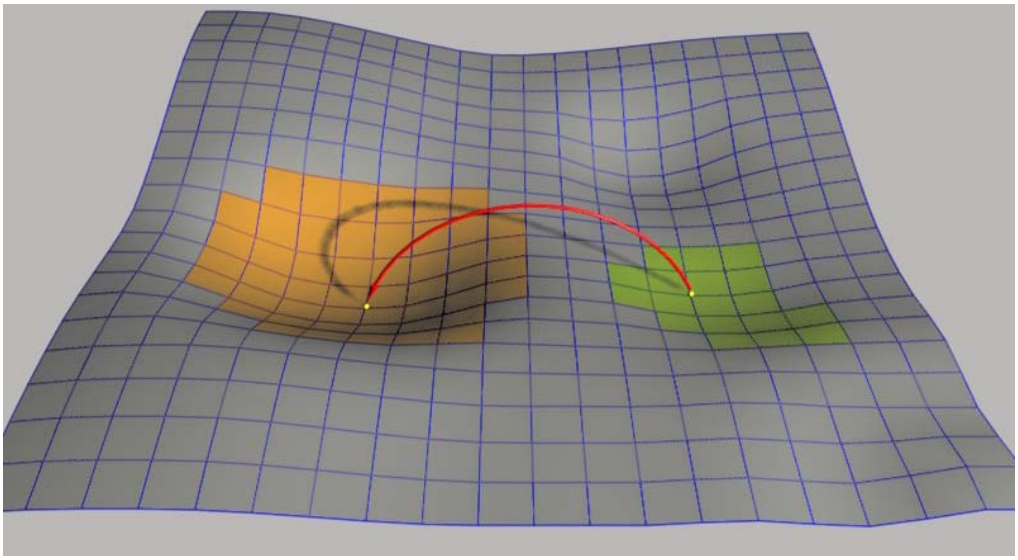


Figura 3. Estrategia de Diversificación.

El uso de memoria en la búsqueda tabú es tanto explícito como implícito; en el primer caso, se almacenan en memoria soluciones completas, generalmente soluciones élite visitadas durante la búsqueda. En el segundo caso, lo que se almacena es información sobre determinadas características de las mismas: atributos (que se modifican después de cada movimiento). Estos dos tipos de memoria son complementarios ya que el primer tipo permiten expandir los entornos de búsqueda mediante la inclusión de soluciones élite, mientras que el segundo tipo los reduce prohibiendo determinados movimientos.

3.3.1.- Memoria a Corto Plazo

Como ya se ha indicado anteriormente, un problema de optimización se puede expresar de la forma:

Mín f(x)

Sujeto a: $x \in X$

La función *f(x)* puede ser lineal o no lineal y la expresión $x \in X$ resume las restricciones impuestas sobre el vector *x*. Estas restricciones pueden incluir desigualdades lineales o no lineales e incluso forzar a algunas de sus componentes a tomar valores discretos.

3.3.1.1.- Búqueda Tabú Básica

En la búsqueda tabú, una estrategia interesante de búsqueda en la vecindad, identifica para cada solución $x \in X$, un conjunto asociado de vecinos $N(x) \subset X$, llamado *entorno* de *x*. Una descripción en pseudocódigo, puede ser la que a continuación se cita [108]:

MÉTODO DE BÚSQUEDA EN EL ENTORNO

Paso 1. Inicialización

Seleccionar una solución de arranque $xActual \in X$
Almacenar la mejor solución actual conocida haciendo $xMejor = xActual$ y definiendo $MejorCoste = f(xMejor)$

Paso 2. Elección y finalización

Elegir una solución $xSiguiente \in N(xActual)$. Si los criterios de elección empleados no pueden ser satisfechos por ningún miembro de $N(xActual)$, o si se aplican otros criterios de parada, entonces el método para.

Paso 3. Actualización

Rehacer $xActual = xSiguiente$, y si $f(xActual) < MejorCoste$, ejecutar el paso 1(B).
Volver al paso 2.

3.3.1.2.- Clasificaciones Tabú

La diferencia entre la memoria a corto y largo plazo es un aspecto importante en la búsqueda tabú, presentando cada una de ellas sus propias estrategias; ahora bien, en ambos casos, lo que provocan es una modificación de la estructura del entorno de la solución actual. En cualquier caso, es predecible el efecto de dicha memoria prefijando que la búsqueda tabú mantenga un historial selectivo de los estados encontrados durante la búsqueda y reemplazando $N(xActual)$ por un entorno modificado $N(H,xActual)$. Es precisamente dicho historial quien determina qué soluciones pueden ser alcanzadas por un movimiento desde la solución actual.

Respecto a la memoria a corto plazo, $N(H, xActual)$ es un subconjunto de $N(xActual)$ y es precisamente la clasificación tabú quien determina los elementos de $N(xActual)$ excluidos de $N(H, xActual)$; En las estrategias de memoria a largo plazo $N(H, xActual)$ puede contener soluciones que no estén incluidas en $N(xActual)$ (generalmente óptimos locales).

La memoria a corto plazo constituye un método agresivo de exploración [110], que pretende realizar siempre el mejor movimiento, sujeto a las posibles soluciones factibles fruto de la aplicación de las restricciones tabú; su primer objetivo es permitir ir más allá de los óptimos locales, realizando en todo momento, movimientos de alta calidad.

A diferencia de la memoria explícita que guarda la solución entera, esta memoria no almacena soluciones completas, sino que se basa en el almacenamiento de atributos constituyendo lo que se denomina memoria basada en atributos.

3.3.1.3.- Memoria basada en atributos

Un atributo de un movimiento o de una solución visitada, es cualquier característica asociada a dicho movimiento o solución. Los atributos seleccionados que se presentan en soluciones recientemente visitadas (memoria basada en lo reciente de los eventos ocurridos), son designados como tabú-activos, provocando el que posibles soluciones que contengan atributos tabú-activos se conviertan en soluciones-tabú las cuales pasan a estar prohibidas durante un periodo determinado de tiempo.

Con la finalidad de que no crezcan demasiado las listas que contienen los atributos tabú, se actualizan tras cada movimiento, permaneciendo en memoria sólo la información más reciente; de aquí se puede inferir que el estado tabú de cada atributo puede cambiar en cada iteración.

Se considera que, pasado cierto número de iteraciones, la búsqueda se desarrolla en regiones distintas por lo que se libera de la condición tabú a dichos movimientos; se denomina periodo tabú o *tabu tenure* al periodo de tiempo en el que un atributo permanece tabú-activo.

3.3.1.4.- Periodo Tabú

El uso de la memoria basada en lo reciente se gestiona mediante la creación de una o varias listas tabú. Estas listas almacenan los atributos tabú-activos e identifican los estados-tabú actuales. El periodo tabú puede ser diferente según qué tipos de atributos, o cambiar según diferentes estados de proceso de búsqueda. Como se indica en Melián y Glover [108], estas variaciones en el periodo tabú de los atributos hacen posible crear diferentes formas de balance entre las estrategias de memoria a corto y largo plazo.

Es necesario definir funciones de memoria que permitan almacenar y usar eficazmente los estados tabú de los diferentes atributos, a fin de determinar cuándo son aplicables las restricciones tabú; un planteamiento puede consistir en la definición de dos vectores: uno para el comienzo del movimiento tabú y otro para la finalización del mismo. Es indispensable tener un contador de iteraciones, de forma que con sumar al comienzo de la restricción el periodo tabú, se obtiene la iteración que determina el fin del movimiento tabú, desapareciendo dicha restricción.

Independientemente de la estructura de datos usada, una cuestión importante es la referida a la duración del periodo tabú. Abundan los estudios realizados sobre esta cuestión, demostrando que un buen valor de este periodo depende del tamaño del problema que se aborda. La experiencia demuestra que cuando el periodo es excesivamente corto, aparecen problemas de ciclado; así mismo, si el periodo es demasiado largo, se produce un deterioro en la calidad de las soluciones encontradas.

Las normas que permiten determinar la duración del periodo tabú se pueden clasificar en estáticas y dinámicas. Las primeras determinan un valor para el periodo que permanece fijo; las dinámicas, con periodo variable, dependen del tipo y calidad de los atributos, asignando periodos mayores a fin de evitar la reversión de atributos que participan en movimientos de alta calidad y que han demostrado ser muy eficientes en problemas de asignación y conducción [9].

3.3.1.5.- Criterios de Aspiración

La prohibición de visitar soluciones que contengan atributos tabú, no es absoluta; puede haber razones de peso que hagan conveniente relajar dicha restricción. Es por lo que se introducen los criterios de aspiración que pueden determinar en qué casos se puede ignorar una restricción tabú, eliminando así la clasificación tabú asociada a un movimiento. Según esto, cualquier solución de más calidad que las encontradas anteriormente, merece ser analizada, aunque para ello se deba utilizar un movimiento prohibido.

Además del tipo simple de criterio que elimina la clasificación tabú de un movimiento en un momento dado, existen otros criterios efectivos para mejorar la búsqueda. Uno de ellos se deriva del concepto de influencia, que mide el grado de cambio producido en una estructura. Se puede establecer un símil con el problema de distribuir objetos desigualmente pesados entre dos cajas, intentando que las dos mantengan el mismo peso al final; un movimiento de alta influencia —cambiar un objeto pesado de una caja a otra— es un movimiento de alta influencia que cambia de forma significativa la estructura de la solución actual.

Se realizarán movimientos de baja influencia mientras existan probabilidades de mejora; en el momento en que se agote esta posibilidad, se cambian los criterios de aspiración a fin de dar entrada a movimientos más influyentes; cabe pensar que, una vez que se ha dado un salto a otra región (diversificación) dentro del campo de exploración, deberían eliminarse las restricciones tabú establecidas previamente para movimientos menos influyentes.

3.3.1.6.- Elección de Candidatos

Una vez iniciado el algoritmo (figura 4), se parte de una solución inicial obtenida por otros métodos y de buena calidad, se crea una lista de candidatos y se procede a elegir el mejor candidato posible. En la creación de la lista de candidatos será determinante la existencia de listas tabú así como los criterios de aspiración.

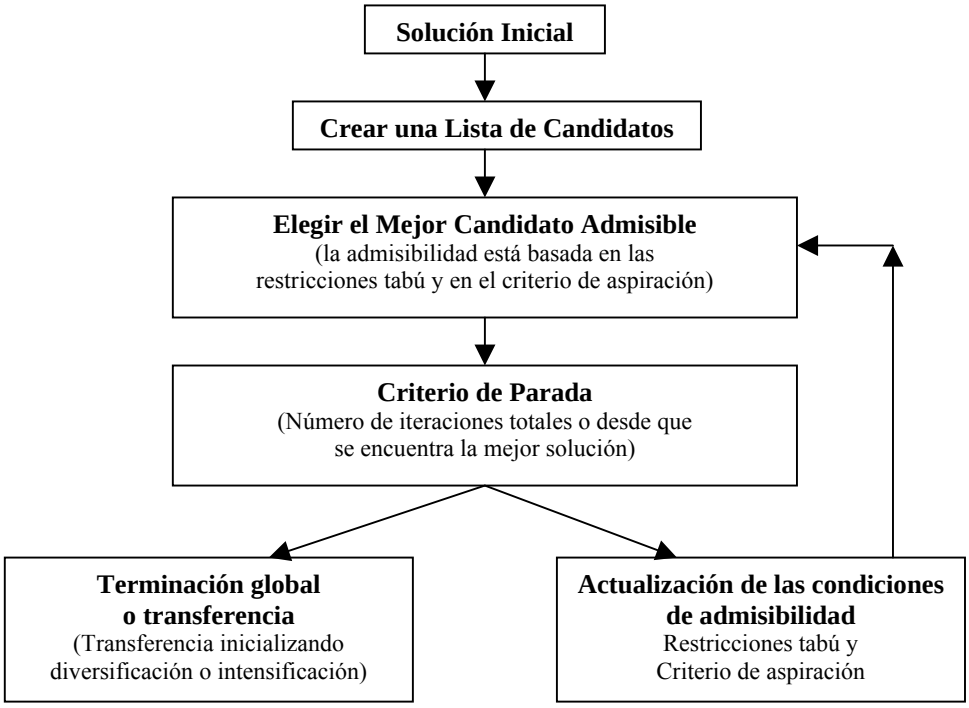


Figura 4. Componentes de memoria a corto plazo (tomado de Glover, 1990b).

A fin de hacer posible la exploración del entorno en un periodo razonable, es normal no estudiar más que un entorno restringido, es decir, aquella parte de las soluciones vecinas que puedan resultar más alentadoras. En búsqueda tabú se han desarrollado diferentes estrategias para confeccionar estas listas de candidatos, entre otras: *Aspiration Plus*, lista de élite, filtros sucesivos…

En el proceso de selección de candidatos, se evalúa cada uno de los movimientos posibles de la lista según una función objetivo o, como en otros casos, mediante medidas de aproximación.

Dado que, generalmente, el número de movimientos tabú es considerablemente inferior en relación al total de movimientos posibles y, puesto que se considera que el coste de evaluar movimientos no es grande, se comprueba el estado tabú del movimiento y, en caso de no ser tabú, se admite de forma inmediata; en caso contrario, el criterio de aspiración puede dar una segunda oportunidad para calificar el movimiento como admisible. Es normal que, en la fase de memoria a corto plazo, el algoritmo se detenga tras un número de iteraciones sin mejora.

Generalmente, la búsqueda tabú se basa en reglas sistemáticas, omitiendo los procesos estocásticos, no obstante, en algunas ocasiones se recomienda aleatorizar algunos procesos a fin de facilitar la elección de candidatos de buena calidad o, también, cuando no esté clara la estrategia a seguir. La selección aleatoria puede ser uniforme o seguir alguna distribución de probabilidad construida empíricamente a partir de la evaluación asociada a cada movimiento.

3.3.2.- Memoria a largo plazo

En muchas ocasiones, las estrategias basadas en memoria a corto plazo son suficientes para obtener soluciones de buena calidad, ahora bien, la inclusión de memoria a largo plazo, así como las estrategias asociadas a la misma, hacen de la búsqueda tabú una estrategia más robusta. La idea subyacente es aplicar memoria a corto plazo y, en caso de necesitar mayor refinamiento, introducir memoria a largo plazo.

3.3.2.1.- Memoria Basada en la Frecuencia

La memoria basada en la frecuencia, proporciona una información que complementa a la información aportada por la memoria basada en lo reciente, ampliando la base para la selección de movimientos preferidos.

La memoria a largo plazo almacena la frecuencia u ocurrencia de atributos en las soluciones visitadas, tratando de descubrir nuevas regiones. Supone el fundamento de las estrategias de intensificación y de diversificación, aunque los elementos fundamentales de estas dos ya se presentan en las estrategias de memoria a corto plazo: las listas tabú en memoria a corto plazo presentan una componente intensificadora, ya que obliga a elegir soluciones atractivas; y una componente diversificadora, ya que obliga a extender la búsqueda a nuevas soluciones cuyos movimientos no estén descartados.

Se concibe la medida de frecuencia como proporciones, cuyos numeradores representan contadores del número de ocurrencias de un evento particular, y cuyos denominadores pueden representar el número total de ocurrencias, la suma de los numeradores, el máximo valor del numerador o la media del valor del numerador.

Dentro de estas medidas, se puede hablar de frecuencias de residencia, es decir, el número de veces que un atributo determinado permanece en una solución, y frecuencias de transición, que indican el número de veces que un atributo cambia de valor.

Ambas frecuencias aportan información adicional sobre la calidad de las soluciones; según lo anterior, una frecuencia de residencia alta puede indicar que un atributo es muy atractivo en una secuencia de soluciones de alta calidad, o puede indicar lo contrario si es una secuencia de soluciones de baja calidad. Si la frecuencia es alta en una secuencia con soluciones de alta y de baja calidad, puede indicar que es un atributo fortalecido (o excluido) que restringe el espacio de búsqueda, y que debe ser desechado (o incorporado) a fin de introducir la diversificación.

En cuanto a la memoria de transición, un valor elevado puede indicar la capacidad de un atributo de incorporarse, en un momento dado, en una solución a fin de realizar un buen ajuste, y volver a salir en la siguiente iteración.

La memoria basada en la frecuencia permite introducir penalizaciones en función de la frecuencia de la ocurrencia de un evento, permitiendo la existencia de listas tabú graduadas, en contraposición al enfoque de memoria basada en lo reciente que implica movimientos de la forma todo-nada. La penalización introducida, por regla general, suele depender de un múltiplo lineal de una medida de frecuencia.

La experiencia indica que el uso de memoria a largo plazo no requiere necesariamente recorrer muchas soluciones antes de que sus beneficios se hagan visibles. En el caso de algunos problemas de rutas y programación de tareas, la inclusión de memoria a largo plazo resulta fundamental para la obtención de buenas soluciones.

3.3.2.2.- Mejoras en el Largo Plazo: Intensificación y Diversificación

Las estrategias de intensificación y diversificación, aunque ya están presentes de forma implícita en memoria a corto plazo, resultan muy interesantes en la memoria a largo plazo; las primeras, incorporando atributos de soluciones de subconjuntos élite seleccionados; las segundas, generando soluciones que incorporan composiciones de atributos significativamente diferentes a los encontrados durante la búsqueda. Como indican Melián y Glover [108] estas estrategias se contrapesan y refuerzan mutuamente.

En la intensificación, se analizan regiones ya exploradas a fin de ser estudiadas más a fondo; las estrategias de intensificación modifican las reglas de elección, a fin de incrementar las combinaciones de movimientos y las características de las soluciones que han demostrado ser buenas. Según lo anteriormente expuesto, se examinan dichas soluciones, extrayendo de ellas las características más relevantes con la finalidad de dirigir la búsqueda hacia soluciones con características parecidas y poder encontrar entre ellas la óptima.

Dos variantes han demostrado ser eficaces al ofrecer soluciones de alta calidad. La primera de ellas introduce una medida de diversificación para asegurar que las soluciones registradas difieran en un grado correcto, borrando a continuación toda memoria de corto plazo antes de reanudar el proceso desde la mejor de las soluciones encontradas [112]. Dicha medida de diversificación puede relacionarse, por ejemplo, con el número de movimientos necesarios para transformar una solución en otra.

La segunda variante [113] conserva una lista de soluciones de longitud limitada y añade al final una nueva solución solo si es mejor que cualquier otra previamente visitada. El último miembro de la lista en cada momento es el escogido y suprimido para reanudar la búsqueda. La memoria a corto plazo, que acompaña a esta solución, se salva; el movimiento previamente tomado de esta solución es inhibido iniciando así un nuevo camino de búsqueda. Este enfoque de recuperar soluciones élite seleccionadas, se denomina *backtracking*.

Un tercera variante consiste en reanudar la búsqueda desde entornos no visitados previamente generados [109]. Esta estrategia explora los entornos vecinos de óptimos locales o vecindarios de soluciones alcanzadas en pasos previos de alcanzar dichos óptimos locales.

Se puede citar, además, la intensificación por descomposición; en este caso se imponen restricciones a partes del problema o a la estructura de solución, a fin de generar una forma de descomposición que permita un enfoque más concentrado en otras partes de la estructura.

En cuanto a la diversificación, que conduce la búsqueda a nuevas regiones aún no exploradas, un enfoque que da buenos resultados en problemas de *scheduling* (programación de tareas) es el que mantiene la memoria basada en la frecuencia sobre todas las soluciones previamente generadas.

Tal y como indican Glover y Laguna [114] se han obtenido mejoras significativas en la aplicación de la memoria a corto plazo mediante este procedimiento.

A continuación (figura 5), se describe un enfoque simple de diversificación en búsqueda tabú.

Aplicar Memoria TS a corto plazo

▼

Mantener la memoria basada en frecuencia de atributos en las soluciones

▼

Cuando el ratio de hallazgo de nuevas soluciones cae por debajo de un umbral, entrar en el bucle siguiente

▼

Aplicar la memoria de TS de corto plazo (manteniendo también la memoria de frecuencia hasta llegar a un óptimo local TS)

▼

Penalizar la inclusión de atributos que ocurran con frecuencia
(multiplicando el factor de penalización por la frecuencia relativa)
Si se alcanza la iteración límite, FIN

▼

Continuar aplicando penalizaciones hasta que se seleccione un movimiento que cree una solución mejor que su inmediato predecesor.
Interrumpir entonces las penalizaciones.

Figura 5. Diversificación en búsqueda tabú (Tomado de Díaz et al., 1996).

En los trabajos de Díaz y Fernández [115] se utiliza la memoria basada en la frecuencia, pero de diferente modo que en la estrategia de intensificación; cuanto mayor sea la frecuencia, más se penaliza una asignación. De esta forma, se producen soluciones con diferentes estructuras a las generadas anteriormente. Los resultados obtenidos presentan una gran calidad y tras un reducido número de iteraciones.

El equilibrio entre estrategias de intensificación y diversificación representan un área de investigación interesante; desde la óptica de TS, el objetivo es el diseño de patrones que permitan obtener mejores compromisos entre ambas estrategias.

3.4.- PATH RELINKING

Uno de los principales objetivos de cualquier método de búsqueda, es crear un buen equilibrio entre las estrategias de intensificación y las estrategias de diversificación. La estrategia *Path Relinking* [28, 30] o re-encadenamiento de trayectorias viene a integrar estas dos estrategias, generando nuevos caminos entre soluciones que ya hayan sido conectadas previamente mediante movimientos durante el proceso de búsqueda.

Inicialmente, *Path Relinking* se usó como un método de intensificación dentro de la búsqueda tabú, aunque en la actualidad, está adquiriendo identidad propia, siendo posible su uso sin estar vinculado a ningún otro método.

Path Relinking se puede interpretar como un método evolutivo por cuanto las soluciones son generadas por combinación de otras soluciones previas, si bien, difiere de éste en que prescinde del carácter aleatorio ya que utiliza reglas sistemáticas y deterministas para combinar soluciones [116].

Esta estrategia se basa en el hecho de que entre dos soluciones élite, se puede trazar un camino que las una, de forma que las soluciones que pertenecen a dicho camino contengan atributos de ellas; según esto, en cada paso, se determina el mejor movimiento como áquel que incorpora la mayor cantidad de atributos, o el más destacado de los atributos de cada solución guía.

Para generar estos movimientos es necesario seleccionar los que cumplan los siguientes objetivos: partiendo de una solución inicial, los movimientos deben introducir, de forma progresiva, los atributos de la función guía (reduciendo, por ejemplo, la distancia entre los atributos de la solución inicial y la función guía).

El diseño básico se puede generalizar de varias formas. La primera consiste en generar caminos que vayan más allá de los extremos, conocida como re-encadenamiento extrapolado; otra posible variante conocida como 'variaciones' empieza desde ambos extremos x' y x'' simultáneamente, produciendo dos secuencias $x' = x'(1), ..., x'(r)$ y $x''= x''(1), ..., x''(s)$. Las elecciones se diseñan para provocar que $x'(r) = x''(s)$ para los valores finales de r y s. Para progresar hacia este resultado cuando $x'(r) \neq x''(s)$, se selecciona $x'(r)$ para crear $x'(r+1)$, mediante el criterio de minimizar el número de movimientos restantes hasta alcanzar $x''(s)$, o se selecciona $x''(s)$ para crear $x''(s+1)$, mediante el criterio de minimizar el número de movimientos restantes hasta alcanzar $x'(r)$. De estas opciones se selecciona la que produzca el menor valor $c(x)$, determinando también si r o s se incrementa en el paso siguiente [108].

También se puede beneficiar de un procedimiento basado en el 'efecto túnel', que permite usar una estructura de entornos diferente a la de la fase de búsqueda estándar, ya que es deseable permitir periódicamente movimientos para el reenlace del camino que, en circunstancias normales, se excluirían por crear infactibilidad, lo que provoca que sea menos probable perderse en una región infactible, ya que la factibilidad se recupera al llegar a x''.

Las estrategias de diversificación y de intensificación se benefician también cuando la búsqueda aporta información no solo de las soluciones visitadas, sino también acerca de las soluciones adicionales evaluadas durante el examen de los movimientos no adoptados; se conoce como soluciones evaluadas pero no visitadas.

Existen otros métodos que indagan sobre la correcta elección de los procedimientos de selección de listas de candidatos, a fin de ahorrar en el coste computacional de tener que evaluar todo el entorno; estas estrategias se conocen con el nombre de estrategias de listas de candidatos. Algunos de ellos, proponen una descomposición del entorno en subconjuntos críticos, así como una regla que asegure que los subconjuntos no evaluados en una iteración, se evalúen en la siguiente; otras estrategias crean listas maestras que contienen un número determinado de las mejores alternativas encontradas.

En general, las estrategias de listas de candidatos provocan una acción diversificante, ya que diferentes partes del entorno son examinadas en diferentes iteraciones. Esto obliga a pensar en la necesidad de coordinar estas estrategias con otras estrategias puras de diversificación, siendo ésta una área abierta a la investigación.

3.5.- OSCILACIÓN ESTRATÉGICA

Considerada en su inicio como un refinamiento especial, resulta ser ya muy familiar entre las diferentes herramientas metaheurísticas.

Proporciona una técnica efectiva entre intensificación y diversificación a medio y largo plazo. La base de este procedimiento consiste en no detener la búsqueda en un punto donde, según las reglas habituales, se debería detener, por ejemplo, la frontera factible. Se modifican las normas, de forma que pueda avanzar desde un punto hasta una profundidad determinada, para volver en sentido contrario, hasta alcanzar un nuevo punto de oscilación, iniciando de nuevo el proceso de rebote; es este movimiento cíclico el que da nombre al proceso.

Este procedimiento combina la intensificación (al lado posible de la frontera) con la diversificación; de otra manera hubiera sido difícil acceder a nuevas regiones en un proceso de movimientos cortos. El control sobre estos movimientos cíclicos se establece generando evaluaciones modificadas y reglas de movimiento, dependiendo de la zona en la que se esté explorando.

4.- LOS PROBLEMAS DE RUTAS DE VEHÍCULOS

4.- LOS PROBLEMAS DE RUTAS DE VEHÍCULOS

4.1.- INTRODUCCIÓN

El problema del diseño de una red de distribución de productos, desde ciertos depósitos, a una serie de clientes, la recogida de productos, o la visita a los mismos, juega un papel importante en la gestión de los sistemas logísticos, suponiendo un ahorro importante en los costes su correcta planificación [117]. Esto, justifica el uso de herramientas de Investigación Operativa como parte importante en la planificación de dichos sistemas pues, como indican Toth y Vigo [118], el coste del transporte se estima entre un 10% y un 20% del coste total del bien.

Los primeros intentos de resolver este tipo de problemas, se atribuyen a Dantzig y Ramser [119], quienes lo aplicaron a la gestión de la distribución de combustible. Posteriormente, Clarke y Wright [120], desarrollaron el primer algoritmo efectivo para su resolución: el algoritmo de los ahorros.

A partir de ese momento, el desarrollo de estas técnicas ha sido imparable, intentando ajustarse, cada vez más, a modelos que incorporen características más propias de la realidad, así como a algoritmos que permitan resolver los problemas de forma más eficiente.

Por otra parte, el espectacular desarrollo de la informática, con su gran poder de computación, permite disminuir el tiempo de ejecución de los algoritmos o, a igualdad de tiempo, resolver modelizaciones de sistemas más complejos; unido esto a otro tipo de herramientas como son los Sistemas de Información Geográfica (SIGs), Sistemas de Posicionamiento Global (en inglés *Global Position System* GPS)… dan lugar a potentes herramientas que resultan ser clave en la adecuada gestión de sistemas logísticos.

Hay que indicar que este tipo de problemas pertenecen al grupo de Problemas de Optimización Combinatoria incluidos, en su mayoría, en la clase NP-Hard (dificultad no polinómica), lo que implica que el esfuerzo por resolver el problema crece de forma exponencial con el tamaño del mismo.

Fisher y Jaikumar [121] y Toth y Vigo [122] resuelven el problema de forma exacta, aunque, como indica Pacheco [123], es mucho más extensa la literatura sobre técnicas heurísticas. Desde el algoritmo de Clarke y Wright [120], el algoritmo *sweep* de Gillet and Miller [124] con su algoritmo de barrido, o el algoritmo de Fisher & Jailkumar [121], hasta la aparición de técnicas metaheurísticas más modernas, como los algoritmos genéticos (Potvin y Bengio [125], Thangiah [126], Baker y Ayechew [127]), recocido simulado (Osman [128]), búsqueda tabú (Gendreau et ál. [129], Rochat y Taillard [130], Taillard et al. [131], Cordeau, Laporte y Mercier [132]), GRASP, (Kontoravdis y Bard [133]), búsqueda local guiada (Vondouris y Tsang [134]), colonias de hórmigas (Gambardella, Taillard, y Agazzi [135]) o *Variable Neigbourhood Search* (Bräysy [136]).

Aplicado al problema VRP Mutiobjetivo (MOVRP), Jozefowiez et al. [137], hacen un repaso a las diferentes estrategias —extensiones de problemas mono-objetivo— entre las que se pueden citar: Park y Koelling [138, 139] que tratan el problema VRP mediante programación por metas, Sutcliffe y Board [140], que abordan el VRP mediante programación lineal; Current y Schilling [141], Lee y Ueng [142], Sessomboon et al. [143], Hansen [144]; Ribeiro y Lourenço [145] estudian el MPVRP; Jozefoiew et al. [146-149] inluyen el balanceo de rutas; Borges y Hansen [150] y Paquete y Stützle [151], con el problema del viajante multiobjetivo (MOTSP); Zhenyu et al. [152], Angel et al. [152], Chitty y Hernandez [153], Li [154] y Murata e Itai [155, 156], que estudian el MOVRP en sus diferentes variantes.

Otros autores tratan el problema MOVRP como generalización del VRP mono-objetivo; se pueden citar a Baràn y Schaerer [157], que tratan el VRPTW mediante colonias de hormigas; Tan et al. [158, 159], que lo resuelven mediante algoritmos genéticos; Jozefowiez et al. [146, 160] tratan el *Covering Tour Problem*; Riera-Ledesma y Salazar-González [161] y Ombuki et al. [162], con variantes del MOVRP.

También se pueden encontrar en la literatura científica autores que desarrollan aplicaciones a problemas reales, como Bowerman et al. [163] quien estudia el diseño de rutas de transporte escolar; Giannikos [164], que analiza el emplazamiento de centros de tratamiento de residuos urbanos; El-Sherbeny [165] que particulariza un VRP para el transporte de una compañía de Bélgica y Corberan et at [166] que lo aplica al transporte escolar.

Trabajos más recientes en el mismo ámbito, se encuentran en Pacheco y Marti [167] con sus trabajos sobre transporte escolar en el entorno rural; Zografros y Androustsopoulos [168] que inciden en el VRPTW con mercancías peligrosas; Tan et al. [158], Mourgaya [169], Doerner et al. [170] que lo aplican al transporte sanitario y Caballero et al. [171], que lo aplican a un caso de localización de centros de incineración de residuos peligrosos en Andalucía.

4.2.- DESCRIPCIÓN DEL PROBLEMA

En el modelo general del problema de rutas de vehículos (conocido en inglés como *Vehicle Routing Problem* o VRP), se dispone de un conjunto de vehículos, un conjunto de clientes a visitar y uno o varios depósitos desde donde se inician las rutas; los vehículos deben satisfacer la demanda de los clientes. Las diferentes configuraciones entre clientes, vehículos y depósitos, así como las características respecto a entregas y recogidas, dan pie a las diferentes variantes del problema.

A continuación se describe el modelo más clásico, conocido como CVRP (en la literatura tradicional, a menudo se ha considerado el *Capacited Vehicle Routing Problem* como VRP clásico) así como sus principales variantes.

4.2.1.- El VRP con capacidades (CVRP)

4.2.1.1.- Descripción

El CVRP básico trata de determinar los recorridos de m vehículos de capacidad C que, partiendo de un depósito, deben visitar un conjunto de clientes para recoger o distribuir mercancías según una demanda d_i y volver de nuevo al depósito de manera que se minimice el valor de la función objetivo considerada (en términos de distancia, coste…).

A partir de este problema básico, aparece todo un conjunto de extensiones o particularizaciones; entre otras: minimizar el número total de vehículos requeridos para dar servicio a todos los clientes, minimizar los costes fijos asociados con el uso de los vehículos (o los conductores), minimizar el coste total de transporte (coste fijo más coste variable de la ruta), etc.

4.2.1.2.- Elementos

CLIENTES

A cada cliente se le asocia una determinada demanda q_i que deberá ser satisfecha por algún vehículo. A partir de aquí, se puede dar el caso de que un vehículo no pueda satisfacer la demanda de todos los clientes, dentro de una misma ruta; de que, en vez de tener que entregar un producto, tenga que recogerlo de los clientes (caso de los sistemas de recogida de RU); de que la mercancía a entregar se

encuentre en determinados centros proveedores, etc. Los clientes pueden ser puntos de entrega o de recogida. En el caso del CVRP, todos los clientes son del mismo tipo. Sin pérdida de generalidad, se considerarán puntos de entrega.

DEPÓSITOS

Las mercancías y los vehículos se encuentran alojados en un depósito. Normalmente, cada ruta empieza y termina en un mismo punto. Pueden tener ventanas de tiempo asociadas así como tiempos asignados a labores de mantenimiento; por otra parte, pueden aparecer problemas de capacidad del depósito, provocando que los vehículos tengan que finalizar la ruta en depósitos diferentes al origen de la ruta. En el caso del CVRP, solo se considera un depósito.

VEHÍCULOS

Pueden existir diferentes tipos de vehículos en cuanto a capacidad y volumen. Si se trata de transportar mercancías con diferentes pesos y volúmenes, la existencia de diferentes vehículos condiciona de forma importante la solución final.

Cada vehículo tendrá asociado, en función de sus características, un coste fijo inherente al hecho de tenerlo a disposición, más un coste variable proporcional a la distancia y/o material transportado. No es infrecuente, por otra parte, que las distancias y/o tiempos recorridos por los vehículos puedan estar sujetos a restricciones legales asociadas al tiempo de permanencia al volante por parte de los conductores.

Al final, uno de los objetivos que siempre se suele perseguir es la minimización del coste (caso de problemas mono-objetivo), altamente dependiente de la distancia recorrida por los vehículos. Por tanto, suele ser este criterio uno de los más utilizados para minimizar. En el caso del CVRP, la flota es homogénea y limitada.

RUTAS

Normalmente, los problemas de rutas se formulan en términos de grafos; de forma que se tiene un grafo $G=(V,E)$, donde V es el conjunto de nodos, que coincide con el conjunto de puntos del problema (depósitos y clientes), y E, que representa el conjunto de arcos entre todos los pares de puntos. Cada arco (i, j) lleva asociado un coste, un tiempo y una distancia de ir del punto i al punto j.

4.2.1.3.- Notación

Existen varias formas de formular este tipo de problemas; en este caso, se va a emplear el modelo de dos índices de Tucker et al. [172]. Respecto a las variables del problema, se pueden citar:

c_{ij} : *coste de ir del nodo i al nodo j.*
t_{ij} : *tiempo de viaje desde el nodo i al nodo j.*
d_{ij}: *distancia entre el nodo i y el nodo j.*

$x_{ij} = 1$ *el vehículo visita al nodo j, después de visitar al nodo i.*
$x_{ij} = 0$ *el vehículo NO visita al nodo j, después de visitar al nodo i.*

El grafo no tiene por qué ser completo, pero se puede completar, a efectos de resolución del problema, asociando $c_{ij} = \infty$ a aquellos arcos no posibles, es decir, que se hayan tenido que añadir; además, para evitar lazos, también se considera $c_{ii} = \infty$; análoga consideración se puede hacer respecto a los tiempos de viaje [173].

4.2.1.4.- Función Objetivo

La función objetivo constituye el eje sobre el cual se estructura la modelización del problema; el objetivo final puede ser maximizar un beneficio o, caso habitual en problemas VRP, minimizar un coste, ya sea de transporte, de uso de recursos, etc. eso sí, sujeto a unas restricciones: entrega de mercancía en unos plazos determinados, número mínimo de visitas, recogida de RU, etc.

Minimizar (o maximizar) f(x)

Sujeto a: $x \in X$.

En el caso de existir más de un objetivo a satisfacer, caso más real, pues se considera que la mayoría de los problemas reales son multiobjetivo, habrá que optimizar dichas funciones obteniendo un conjunto de soluciones no dominadas, que constituyen lo que se denomina conjunto de puntos eficientes.

Minimizar (o maximizar) $(f_1(x), f_2(x), \ldots f_p(x))$

Sujeto a: $x \in X$

En el caso concreto del CVRP el objetivo es minimizar el coste del transporte que se define como la suma de los costes de los arcos que componen las rutas.

4.2.1.5.- Formulación

A continuación se describe la formulación del CVRP.

Sea $V=\{1,2,\ldots,n\}$ el conjunto de nodos donde *1* representa el depósito y *{2, 3, ...,n}* los clientes. Sea $S \subset \{2,3,..,n\}$ un subconjunto de clientes; Q la demanda total y m el tamaño de la flota. Toth y Vigo [118] lo formulan de la siguiente manera:

$$\min \quad \sum_{i=1}^{n}\sum_{J=1}^{n} c_{ij} \sum_{k=1}^{m} x_{ij} \qquad (1.1)$$

$$\text{s. a.} \quad \sum_{j=2}^{n} x_{1j} \leq m \qquad (1.2)$$

$$\sum_{i=2}^{n} x_{i1} = \sum_{j=2}^{n} x_{1j} \qquad (1.3)$$

$$u_i - u_j + Qx_{ij} \leq Q - d_i \qquad \forall i \neq j \;\; i,j \in \{2,\ldots,n\} \qquad (1.4)$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E$$

$$u_j \in R^{+} \qquad \forall i \{2,..,n\}$$

$$u_i > 0$$

La función objetivo 1.1 es el coste total de la solución. Las restricciones 1.2 y 1.3 indican que m es la cantidad máxima de vehículos utilizados en la solución y que todos los vehículos que parten del depósito deben regresar. La restricción 1.4 asegura que se respeten las restricciones de capacidad máxima, además de evitar que se formen subciclos.

4.2.2.- El problema del Agente Viajero (TSP)

En este problema, se dispone de un solo depósito y un solo vehículo que debe visitar a todos los clientes en una única ruta. No existe demanda asociada a cada cliente, se prescinde del coste fijo del vehículo; el problema puede formularse como:

$$\min \quad \sum_{i=1}^{n}\sum_{J=1}^{n} c_{ij}x_{ij} \tag{1.5}$$

$$\text{s. a.} \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad \forall i \in \{1,...,n\} \tag{1.6}$$

$$\sum_{J=1}^{n} x_{ij} = 1 \qquad \forall j \in \{1,...,n\} \tag{1.7}$$

$$y_i - y_j + nx_{ij} \leq n-1 \tag{1.8}$$

$$x_{ij}, y_{ij} \in \{0,1\} \qquad \forall (i,j) \in E$$

La función objetivo 1.5 obtiene el coste total de la solución como el sumatorio de todos los costes de cada arco que intervienen en la solución (es decir $x_{ij} = 1$). Las restricciones 1.6 y 1.7 indican que a cada nodo debe llegar un único arco y partir otro (en caso de formar parte de la solución). Por último, la restricción 1.8 evita la formación de subciclos.

4.2.3.- El problema de los m Agentes Viajeros (m-TSP)

Este caso resulta ser una generalización del TSP en el que hay un depósito y *m* vehículos; el objetivo es la obtención de *m* rutas, una para cada vehículo, de modo que cada cliente sea visitado solamente por un vehículo. Cada ruta comienza y termina en el depósito y puede contener como máximo *p* clientes. Una formulación del problema se puede encontrar en Miller [172] .

4.2.4.- Variantes del VRP

4.2.4.1.- VRP con múltiples depósitos (MDVRP)

De planteamiento similar al VRP, el MDVRP (*Multi Depot Vehicle Routing Problem*) difiere del primero en la existencia de múltiples depósitos de los que parten los vehículos y a los que deben regresar tras visitar a los clientes.

Los trabajos previos se deben a Golden et al. [174], si bien son Kulkarni y Bhave [175] quienes, en 1985, presentan las primeras formulaciones para VRP, m-TSP y MDVRP.

VARIANTES DEL MDVRP

- Flotas fijas asociadas a cada depósito

Puede darse el caso de que las flotas de vehículos estén fijadas previamente e incluso asociadas a los depósitos, caso que se puede considerar frecuente; así no cabe decidir cuántos vehículos se asignan a cada depósito, puesto que está fijado de antemano. Si se considera que *(n+1,...,n+m)* representan los

depósitos y si se denota v_i el número de vehículos de la flota asignada a un determinado depósito i, donde $i \in \{n+1, \ldots n+m\}$, la suma de los vehículos de todas las flotas debe ser v, o lo que es lo mismo $\sum_{i=n+1}^{n+m} v_i = v$.

Considerar esta nueva situación implica añadir nuevas restricciones que impongan que cada depósito haga uso de sus vehículos y no de otros.

- Flotas fijas que parten de un depósito y terminan en otro.

Otra posibilidad puede ser el que los vehículos puedan partir de un depósito y regresar a otro; introduciendo esta relajación, es posible, tal y como indica Tirado [173], que se consiga reducir la longitud de las rutas, a costa de desconocer el número final de vehículos en cada depósito.

La diferencia respecto al caso anterior radica en la eliminación de la restricción que impone el regreso de los vehículos asociados inicialmente a un depósito, al mismo. En caso de no existir asociación de vehículos a depósitos, se elimina esta restricción y la que impone la restricción del número máximo de vehículos que salen de un depósito.

4.2.4.2.- VRP durante varios días (PVRP)

Este problema PVRP (*Periodic Vehicle Routing Problem*) difiere del problema VRP clásico en que, en este caso, el horizonte temporal es de T días, de forma que los vehículos recorren sus rutas diariamente, pudiendo visitar a los clientes en cualquiera de los días de ese periodo. Cada cliente lleva asociada una demanda y, a veces, un calendario de recogida o de entrega.

El objetivo del problema es determinar la correcta planificación de las visitas a los clientes así como los días en que deben visitarlos, minimizando el coste total.

Los primeros trabajos con este problema se deben a Beltrami y Bodin [176], en 1974, y a Russell e Igo [177] en 1979; Más adelante, en 1984, aparecen los trabajos de Christofides y Beasley [178], donde lo que se intenta minimizar es el tamaño de la flota. Buenos resultados se obtienen en Chao et al. [179], donde se implementa una heurística en dos fases. Cordeau et al. [180], en 1997, plantean un algoritmo basado en búsqueda tabú; más recientemente, en 2001, Drummond et al. [181], usan algoritmos genéticos en paralelo. En Ball se puede encontrar una recopilación de modelos que combinan diseños de rutas con asignación de días en que se deben realizar. En Alegre et al. [182] se hacen interesantes aportaciones en cuanto a las metaheurísticas utilizadas: búsqueda tabú, algoritmos meméticos, *Ejection Chains*, con buenos resultados.

Otras trabajos se pueden encontrar en Alegre et al. [183], donde trabajan con horizontes más largos, Francis y Smilowitz [184], que introducen una variante: el PVRP-SC, es decir, con elección de la frecuencia de visitas a los clientes, y en Mourgaya y Vanderbeck [185].

4.2.4.3.- VRP con carga divisible (SDVRP)

En este caso se permite que la carga a suministrar a cada cliente, sea distribuida por varios vehículos, consiguiendo una reducción total en los costes.

El problema SDVRP (*Split Delivery Vehicle Routing Problem*) fue abordado inicialmente por Dror y Trudeau en los años 1989 y 1990, [186, 187] que lo definieron de forma bastante precisa; además, propusieron un algoritmo para su resolución basado en una búsqueda local. Fue en el año 1994 cuando, estos mismos investigadores [188] propusieron una nueva modelización basada en programación lineal. Más adelante, en 1995, se formuló una variante del problema por Frizzell y Griffin [189], resuelta por un

método heurístico. Mullaseril et al. [190], en 1997, y Sierksma y Tijssen [191], en 1998, desarrollaron más variantes de este problema.

Un caso particular del SDVRP, conocido como SDP (*Skip Delivery Problem*), es aquel en cual la capacidad de los vehículos es pequeña (habitualmente dos) y la carga solo puede dividirse entre dos clientes. Estos problemas pueden ser resueltos en tiempo polinomial bajo una serie de supuestos (ver Archetti et al. [192]). En otro trabajo de los mismos autores [193] en 2006, se obtiene una cota bastante ajustada del ahorro, lo que permite la división de la carga entre varios vehículos; por otra parte, Berenguer et al. [194] propusieron una cota inferior.

Gendreau et al. aporta en 2003 [195] un método exacto aplicado a este caso y Archetti et al. [196], en 2006, un algoritmo basado en búsqueda tabú.

4.2.4.4.- VRP con variables aleatorias (SVRP)

Este problema aparece cuando parte de los componentes del problema tienen un carácter estocástico, es decir, se comportan según una determinada función de probabilidad que, en muchos casos, se desconoce.

Dentro de la componente aleatoria del problema se pueden incluir:

- Clientes: un cliente determinado i puede estar presente en el problema con una probabilidad P_i y no formar parte, con una probabilidad $1\text{-}P_i$.
- Demandas: las demandas de los diferentes clientes d_i se ajustan a una distribución de probabilidad en vez de permanecer constantes.
- Tiempos: el tiempo t_{ij} que tarda un vehículo en realizar el trayecto de i a j o, incluso, el tiempo de descarga tienen comportamiento aleatorio.

Por lo general, todos los procedimientos empleados para resolver este problema, se suelen descomponer en dos fases:

1. Obtención de una solución inicial sin tener en cuenta la parte aleatoria de los diferentes componentes o, lo que es lo mismo, resolviendo con los valores esperados de dichas distribuciones, mediante un heurístico determinado o, incluso, mediante métodos exactos.
2. Se va corrigiendo el problema, a medida que se conocen aspectos relacionados con las variables aleatorias, de forma que no violen las restricciones del problema.

Al igual que en el resto de casos, el objetivo final es minimizar el coste total, a base de reducir la longitud total recorrida y el coste fijo asociado a cada vehículo.

Los primeros trabajos sobre SVRP se atribuyen a Tillman [197] quien, en 1969, y partiendo del conocido algoritmo de los ahorros de Clarke y Wright [120], presenta una variante que considera demandas distribuidas según una distribución de Poisson. En 1983, Stewart y Golden [198] profundizan en el problema SVRP, introduciendo una penalización y utilizando restricciones dependientes de la realización de las diferentes variables aleatorias. Dror y Trudeau [199] presentan un modelo alternativo que mejora los datos de Stewart y Golden.

4.2.4.5.- VRP con entrega y recogida de mercancías (VRPPD)

En este caso, algunos de los clientes, además de recibir mercancías, entregan otras a los vehículos para su recogida. También conocido como VRPPD (*Vehicle Routing Problem with Pick-up and Delivering*). Para que la recogida sea efectiva, es necesario disponer de espacio dentro del vehículo, que hace que la solución del problema sea mucho más compleja.

Se suelen hacer varias simplificaciones, aunque no todas necesariamente:

a. Cada cliente recibe o entrega mercancía, pero no las dos cosas al mismo tiempo.

b. No hay intercambio de mercancías entre clientes.

c. Se puede imponer que los vehículos repartan primero toda la mercancía y, posteriormente, recojan la que tienen pendiente.

El principal campo de aplicación de esta variante del VRP ha sido el transporte aéreo y naval de personal, personas discapacitadas, transporte de estudiantes. Estos problemas llevan aparejado, a veces, limitaciones temporales (es decir, ventanas de tiempo) e, incluso, limitaciones por tiempo máximo de permanencia en el vehículo (caso del transporte escolar) [123].

Dentro de la literatura científica, los primeros trabajos se atribuyen a Wilson et al., en el año 1971 [200], quienes trataron de resolver problemas prácticos aplicados al transporte de personas. Ampliaron posteriormente sus publicaciones [201, 202], en el intento de optimización de redes tanto en Haddonfield como en Rochester (USA); en estos trabajos, se introdujo el concepto de creación de rutas a partir de inserción secuencial de vértices.

Otros trabajos, como Toth y Vigo [203], introducen una implementación de búsqueda tabú en el procedimiento de inserción de vértices en la solución, y otro de mejora.

Respecto a procedimientos exactos, se pueden encontrar algunas propuestas en los trabajos de Kohl et al. [204], donde introducen las desigualdades válidas, y los trabajos de Du Merle et al. [205], donde utiliza variables de perturbación acotadas.

4.2.4.6.- VRP con ventanas de tiempo (VRPTW)

En esta variante, conocida como VRPTW (*Vehicle Routing Problem with Time Windows*), cada cliente tiene asociada, además de una capacidad, un espacio temporal en el que está permitida la entrega o recogida de mercancía.

Se suelen considerar dos casos: en el primero, los vehículos no pueden prestar un servicio después de una hora determinada, aunque, en caso de llegar antes de tiempo, sí pueden esperar; en el segundo caso, se permite violar las ventanas de tiempo, aunque pagando por ello un determinado coste. Esta segunda opción ha sido menos estudiada.

Esta variante del VRP presenta dificultades inherentes al propio problema, lo que hace que los primeros estudios se centrasen en casos concretos; esto se puede ver, entre otros, en Pullen y Webb en 1967, Knight y Hofer en 1968 y Madsen, en 1976 [206-208]. Más adelante, se realizaron estudios centrados en el desarrollo de nuevos algoritmos, aplicables a casos reales e, incluso, a variantes más sencillas con ventanas de tiempo, como en Desoiers et al. [209].

Unos trabajos de Kolen et al. [210] y Desrochers et al. [211] fueron el origen de unos algoritmos exactos, basados en técnicas de *Branch and Cut* y que funcionaron inicialmente mejor de lo esperado

[212] [204], además de algoritmos metaheurísticos que presentan buenos resultados [24, 132, 213]. En los trabajos de Kallehauge et al., se pueden ver las investigaciones realizadas en los últimos años, sobre todo en la hibridación de métodos. [214, 215].

Existe unas librerias, denominadas Instancias de Solomon [216] que recogen una serie de casos, divididos en seis grupos y denotados como R1, R2, C1, C2, RC1 y RC2. Éstas, se utilizan como ejemplo en diferentes implementaciones a fin de comprobar la bondad de las mismas.

4.3.- ALGORITMOS APLICADOS A LA RESOLUCIÓN DEL VRP

4.3.1.- Métodos Exactos

Este tipo de problemas son tan complejos que, solo aquellos que tienen pocos puntos, pueden ser resueltos por métodos exactos. Se suelen aplicar esquemas del tipo *Branch and Bound*, así como algoritmos que se apoyan en programación dinámica. Más información sobre métodos exactos se puede encontrar en los trabajos de Laporte y Norbert [217], y en Laporte [218].

De todo lo anterior, se puede deducir que son necesarios algoritmos heurísticos que aporten soluciones de calidad aceptable, en tiempos razonables de computación. Así mismo, no es menos importante que dichos algoritmos se puedan implementar a través de lenguajes de programación como pueden ser PASCAL, C, C++, etc.

4.3.2.- Heurísticos aplicados al VRP

Se pueden clasificar en tres categorías:

1. Algoritmos constructivos: construyen una solución de forma gradual, mejorando la solución objetivo, aunque no presentan, posteriormente, fases de mejora.
2. Algoritmos por fases: proceden, en dos etapas, a agrupar los vértices, formando *clusters* y, a continuación, calcular la mejor ruta para cada uno de estos *clusters*. Se puede ejecutar en este orden o a la inversa.
3. Algoritmos de inserción: constituyen, en realidad, un caso particular de los algoritmos constructivos. Construyen una solución mediante sucesivas iteraciones; en cada una de ellas se inserta un nuevo cliente según diferentes criterios; el algoritmo se detiene cuando todos los clientes están insertados.

Existen, además, otros procedimientos como búsqueda local, en los que, previa definición de un vecindario, se parte de una solución y se reemplaza por otra, en la medida en que mejora la función objetivo; se repite el proceso hasta que la solución no puede ser mejorada. El vecindario se obtiene mediante la definición de movimientos que pueden ser intra-rutas o entre-rutas. Estos movimientos se implementan mediante operadores de intercambio, como pueden ser λ-intercambio, *Or-opt*, Van Breedam, etc.

4.3.2.1.- Algoritmos Constructivos

Dentro de este grupo, el algoritmo más conocido, es el algoritmo de los ahorros, de Clarke y Wright [120]; en ese caso, si dos rutas definidas *(1, … , i,1)* y *(1, j, …,1)* son recombinadas (figura 6) formando la nueva ruta *(1, … , i, j, …,1)*, el ahorro obtenido es $s_{ij} = c_{i1} + c_{1j} - c_{ij}$.

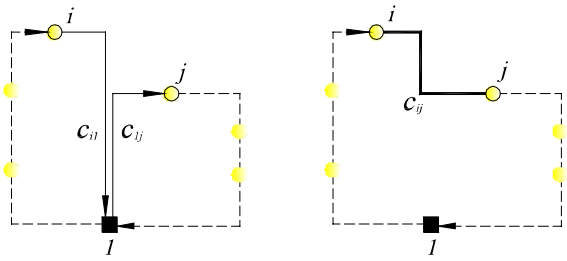


Figura 6.

En este algoritmo, se parte de una solución inicial y se realizan las uniones que den mayores ahorros siempre y cuando no se violen las restricciones del problema.

Existen dos versiones del algoritmo: la versión en paralelo y la versión secuencial; en la primera, se trabaja de forma simultánea sobre todas las rutas, mientras que en la versión secuencial, se recorren de forma ordenada.

Se observa que, con la implementación general, a medida que se avanza en la configuración de las rutas, van quedando los clientes más lejanos, dando lugar a rutas de muy baja calidad. Gaskell [219] y Yellow [220] resuelven esta cuestión introduciendo un parámetro λ que penaliza la unión de clientes lejanos, según la expresión $s_{ij} = c_{i0} + c_{0j} - \lambda\, c_{ij}$. Golden et al. [174] muestran que para valores de λ entre 0.4 y 1, se obtienen buenos resultados.

4.3.2.1.1.- Algoritmos de ahorros basado en Matching

La elección del máximo ahorro suele ser una estrategia demasiado voraz, ya que cuando se unen dos rutas, se elimina la posibilidad de que éstas participen en otras posibles rutas, dado que dejan de ser comienzo o final; en este algoritmo se decide la unión a realizar, según afecta ésta a las nuevas iteraciones. Para esto, se considera un grafo que tiene a todas las rutas como nodos y un arco entre dos nodos r_i, r_j cuyo peso es el ahorro obtenido si ambas rutas se combinan (siempre y cuando sea factible).

Más propuestas sobre este algoritmo, se pueden encontrar en Desrochers y Verhoog [221], en Altinkemer y Gavish [222], y en Wark y Holt [223].

4.3.2.2.- Algoritmos por Fases

En este tipo de algoritmos, se procede en dos fases, pudiendo variar el orden, según esto pueden ser:

- Asignar primero – rutar después *(cluster first – route second)*: en la primera fase se generan los grupos de clientes denominados *clusters* que están dentro de la misma ruta en la solución final y, en la segunda, se genera la ruta; las restricciones de capacidad se consideran en la primera fase, asegurando que la demanda total no exceda de la capacidad del vehículo; de esta forma, construir la ruta para un *cluster*, se convierte en un TSP que puede ser resuelto de forma exacta o aproximada.
- Rutar primero – asignar después *(route first – cluster second)*: en la primera fase se calcula un TSP que visite a todos los clientes; como esta ruta viola las restricciones del problema (capacidad de los vehículos) se fragmenta en varias subrrutas factibles.

4.3.2.2.1.- Métodos Clusters primero – Rutas después

Se pueden destacar los siguientes métodos:

a. heurístico de barrido: en este heurístico [124, 224], los *clusters* se forman girando una semirrecta con origen en el depósito e incorporando los clientes barridos hasta violar la restricción de capacidad (figura 7). Cada cluster es resuelto posteriormente mediante un TSP.

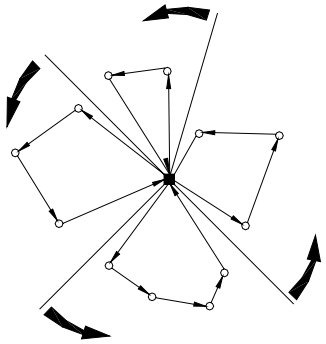


Figura 7.

Este algoritmo se puede aplicar en el caso de grafos planos, de forma que cada nodo *i* queda representado mediante sus coordenadas polares es decir, (θ_i, ρ_i), siendo el primero, el ángulo y el segundo, la longitud a dicho nodo. El procedimiento se repite *n* veces, tantas como clientes existen.

b. Heurístico de asignación generalizada: propuesto por Fisher y Jaikumar [121], genera los *clusters* resolviendo un problema de asignación generalizada (en inglés *Generalized Assignment Problem* GAP); en un primer paso, se fijan *k* clientes semilla s_k con $k = (1, 2, ..., K)$, sobre los que se construyen los *clusters*. En una segunda fase, se eligen los clientes que van a pertenecer a cada *cluster*, mediante la resolución de un GAP. Finalmente se resuelve el TSP correspondiente a cada *cluster*.

Puede utilizarse en los casos del VRP simétricos o asimétricos y el número de vehículos a utilizar, que coincide con el número total de rutas, es conocido de antemano.

c. Heurístico de localización: original de Bramel y Simchi-Levi [225], presenta similitudes con el algoritmo anterior, ya que en ambos existe un conjunto de clientes semilla (también denominados 'concentradores'); en una primera fase, se trata de encontrar *v* concentradores tal que se minimice la distancia de los vértices a cada concentrador y siempre y cuando no se viole la restricción de capacidad del vehículo asociado a cada concentrador. En una segunda fase, se resuelve la construcción de la ruta. En definitiva, lo que se hace es resolver un problema de localización con restricción de capacidad.

4.3.2.2.2.- Métodos Rutas primero – Clusters después

Como ya se ha citado anteriormente, en este método (figura 8) se calcula en primer lugar un TSP que visite a todos los clientes y, en una segunda fase, se fragmenta en subrrutas de forma que no se violen las restricciones de capacidad. Se puede formular también como el de hallar un camino mínimo en un grafo dirigido y sin ciclo. Un ejemplo de este método se puede encontrar en Beasley [226].

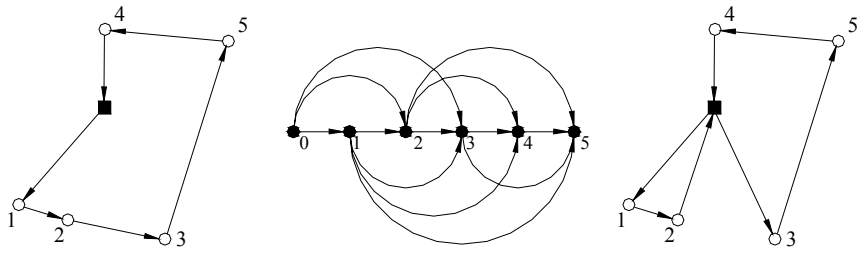


Figura 8. (Tomado de Olivera, 2004).

4.3.2.2.3.- Algoritmo de Pétalo

Este algoritmo fue propuesto por Balinski y Quandt en 1964 [227] y supone una generalización del algoritmo de barrido.

En este caso, se dispone de un conjunto de rutas R, de modo que cada ruta $r \in R$ es factible, pero cada cliente es visitado por varias de las rutas. El problema de seleccionar un subconjunto de R de coste mínimo que visite exactamente una vez a cada cliente puede formularse como un problema de partición de conjuntos (*Set Partitioning Problem* o SPP):

Este problema para elegir las rutas adecuadas, se podría formular de la siguiente forma:

$$\min \quad \sum_{k \in S} d_k x_k$$

$$\text{s.a.} \quad \sum_{k \in S} a_{ik} x_k = 1 \qquad \forall i = 2,...,n$$

$$x_k \in \{0,1\} \qquad \forall k \in S$$

Siendo S el conjunto de rutas generado en la primera fase del algoritmo; a_{ik} vale 1, si el vértice i pertenece a la ruta k y 0 en otro caso; d_k representa el coste (en forma de distancia) de la ruta k, y la variable x_k con valor 1, si forma parte de la ruta y 0 en otro caso.

El algoritmo de pétalos presenta dos fases: en la primera y a partir de un conjunto de reglas heurísticas, se obtiene un conjunto de rutas S que formen soluciones factibles que se van a considerar; en la segunda fase, a partir del conjunto S, un conjunto de rutas que conformen una solución factible mediante un SPP.

En Foster y Ryan [228] y en Ryan et al. [229], se pueden encontrar reglas heurísticas para determinar cómo elegir rutas simples (*1*-pétalos) que formen parte del conjunto S y, una posterior ampliación de éstas (*2*-pétalos), en Renaud et al. [230].

4.3.2.3.- Algoritmos de Inserción

Aun siendo un caso particular de algoritmos constructivos, presentan la suficiente entidad como para ser considerados en un grupo aparte. Crean una solución a partir de inserciones sucesivas de clientes en las rutas. En cada iteración, se obtiene una solución parcial de la cual forman parte un subconjunto de clientes.

Existen dos procedimientos: la inserción secuencial en la que un nuevo cliente es añadido a la última ruta creada; esto presenta el inconveniente de que los últimos clientes tienden a estar muy dispersos con

lo que empeora la calidad de la solución. Para solventar esta deficiencia, aparece la inserción en paralelo en la que se permite insertar un cliente en cualquier ruta.

Todos los heurísticos de inserción para el TSP pueden ser aplicados al VRP, siempre y cuando no se violen las restricciones. Más información sobre este heurístico se puede encontrar en Bodin et al. [231].

4.3.2.3.1.- Inserción secuencial de Mole y Jameson

En este heurístico [232] se utilizan dos parámetros para decidir el próximo cliente a insertar en la solución parcial. En un primer paso, para cada cliente no visitado, se calcula la mejor posición de inserción teniendo en cuenta solamente las distancias y sin reordenar los nodos ya presentes en la ruta.

Si solo se tuviese en cuenta este parámetro, los clientes finales solo serían tenidos en cuenta en las iteraciones finales, es decir, en el único momento en que podrían ser factibles; para evitar esta situación, se utiliza un incentivo adicional. Así, en cada iteración se busca el cliente que maximiza el segundo parámetro en la posición indicada por el primero. Todo esto siempre y cuando sean factibles las inserciones.

4.3.2.3.2.- Inserción en Paralelo de Christofides, Mingozzi y Toth

Este algoritmo, propuesto por Christofides et al. [233], trabaja en dos fases; en la primera, se determina la cantidad de rutas a utilizar y el cliente inicio de cada uno y, en la senda fase, se crean dichas rutas insertándose el resto de clientes.

Hay que resaltar que este algoritmo, más sofisticado que el anterior, aplica una inserción secuencial en la primera fase y una inserción en paralelo, en la segunda. Diferentes propuestas en pseudocódigo de estos algoritmos se pueden encontrar entre otros, en Olivera [117] y en Tirado [173].

4.3.2.4.- Procedimiento de Búsqueda Local

Aparte de la clasificación anteriormente citada, existe otro tipo de heurísticos cuya finalidad es la de intentar mejorar una solución ya propuesta; según esto, se puede definir un conjunto de soluciones vecinas *N(s)* asociado a cada una de las soluciones ya encontradas *s*; un procedimiento de búsqueda local parte de una solución *s*, tantea dentro del vecindario de dicha solución y, si la nueva solución $s^* \in N(s)$ es mejor, es decir, mejora la función objetivo, se sustituye por la actual y así de forma repetida hasta que no mejore la función objetivo.

El resultado de este proceso es un óptimo local; por lo general, este tipo de algoritmos detienen la búsqueda en el momento en el que no pueden mejorar.

El vecindario de una solución s se suele obtener aplicando reglas o procedimientos sencillos denominados movimientos. Éstos pueden ser intra-rutas o entre-rutas dependiendo de si definen el vecindario mediante combinaciones de movimientos en una ruta determinada, o bien combinaciones entre dos o más rutas, para dar lugar a otra nueva ruta.

Se describen a continuación algunos de los operadores que dan lugar a la generación de vecindarios:

4.3.2.4.1.- El operador λ-Intercambio

Constituye uno de los operadores intrarrutas más conocido. Desarrollado por Lin en 1965 [234], consiste en eliminar λ arcos de una solución ($\lambda > 1$) y reconectar los λ segmentos restantes de todas las formas posibles; se evalúan cada una de las nuevas rutas y, si alguna de ellas mejora la solución, se

adopta como nueva ruta. El proceso se detiene cuando no encuentra ninguna combinación que mejore la solución actual.

Se debe especificar si se adopta una estrategia *first improvement* frente a otra estrategia *best improvement*, es decir, la primera solución buena frente a la mejor de todas las soluciones evaluadas.

Dada una ruta de *n* clientes, el número de formas de eliminar λ arcos, es $\binom{n+1}{\lambda}$ y, determinada la elección de arcos a eliminar, existen $2^{\lambda-1}(\lambda-1)!$ formas de recombinar dichos arcos, por lo que, al final, existen $2^{\lambda-1}(\lambda-1)!\binom{n+1}{\lambda}$ λ-intercambios. Comprobar si una solución es óptima puede hacerse en $O(n^{\lambda})$ en el peor de los casos.

Los valores de λ más adoptados suelen ser 2-intercambio y 3-intercambio. En la siguiente figura (9) se pueden observan los movimientos para un 2-intercambio.

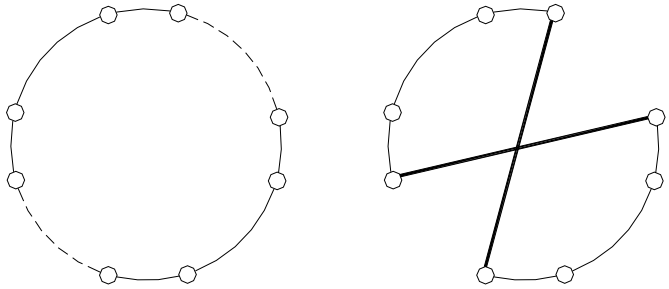


Figura 9. Movimientos para un 2-intercambio (Tomado de Olivera, 2004).

En Johnson y McGeoch [235] se aportan movimientos que mejoran el coste de una ruta sin necesidad de explorar todas las posibilidades. Renaud et al. [236] proponen una versión reducida denominada 4-opt en la cual se parten de dos secuencias disjuntas de nodos, donde *w* es un parámetro del algoritmo y que explora un conjunto de reconexiones, con un tiempo de chequeo $O(wn^2)$.

En el caso anterior, λ es un valor prefijado de antemano. Un nuevo algoritmo, propuesto por Lin y Kernighan [237], propone un intercambio de un conjunto de arcos por otro, pero determinando el número de dichos arcos de forma dinámica. En la figura 10 se muestra una posible aplicación del método para encontrar un intercambio. El proceso se repite hasta encontrar un óptimo local, es decir, no se pueden encontrar nuevos intercambios que mejoren la solución.

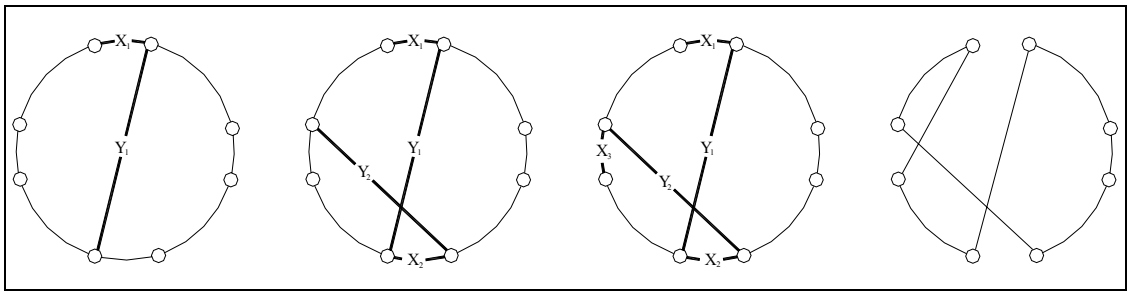


Figura 10. (Tomado de Olivera, 2004).

Otra propuesta, es la realizada por Or [238], partiendo del 3-opt y que se denomina Or-Opt; consiste en eliminar *k* clientes de una ruta y colocarlos en otra posición de la misma ruta, de forma que se

conserve el orden. Se comienza con $k = 3$, luego con $k = 2$ y, finalmente, con $k = 1$. En la figura 11, se indican las formas de poder ordenar los 3 primeros clientes.

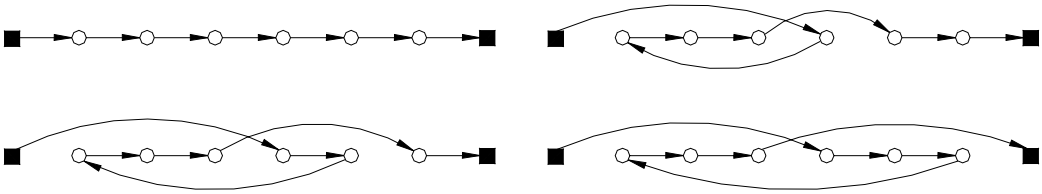


Figura 11. (Tomado de Olivera, 2004).

4.3.2.4.2.- Operadores de Van Breedam

Estos dos operadores, propuestos por Van Breedam [239], intecambian clientes entre un par de rutas; en el primero de ellos, denominado *String Relocation* una secuencia de *m* nodos se transfiere de una ruta a otra manteniendo el orden en la ruta original. El operador *String Exchange* envía una secuencia de *m* clientes de una ruta a la segunda, y ésta, devuelve una secuencia de *n* clientes a la primera.

En la siguiente figura (12) se puede apreciar el resultado del intercambio:

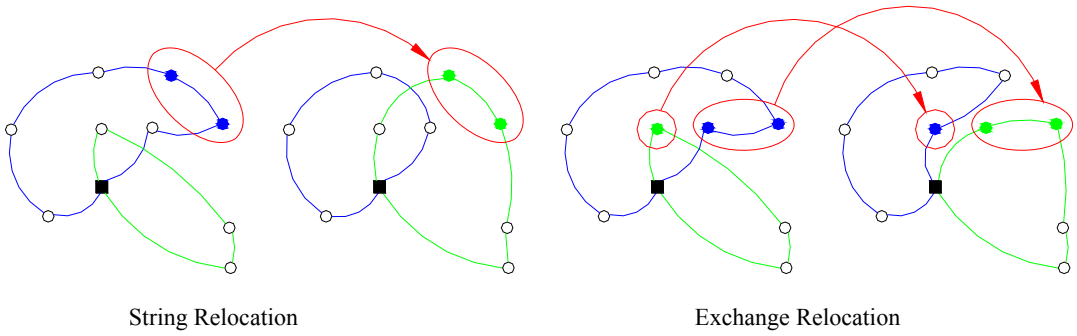


Figura 12. Intercambio de clientes.

4.3.2.4.3.- Geni y Genius

Acrónimos de *GENeralizated Insertions* (inserciones generalizadas), surgen en el entorno del TPS y son presentadas por Gendreau et al. [240]; su principal característica es que la inserción de un nuevo cliente no tiene por qué ocurrir necesariamente entre dos nodos consecutivos. Existen dos tipos de inserciones: tipo I y tipo II.

En una inserción de Tipo I (figura 13), se considera un nodo v_k en el camino de v_j a v_i ($v_k \neq v_i$ y $v_k \neq v_j$). La inserción consiste en eliminar los arcos (v_i, v_{i+1}), (v_j, v_{j+1}) y (v_k, vk_{+1}) y agregar los arcos (v_i, v), (v, v_j), (v_{i+1}, v_k) y (v_{j+1}, v_{k+1}). Es necesario, además, invertir el sentido de los caminos $(v_{i+1},. . ., v_j)$ y $(v_{j+1},. . ., v_k)$.

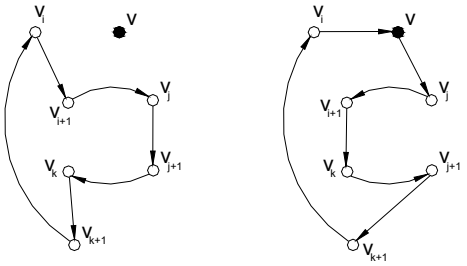


Figura 13. Inserción GENI del tipo I

En la inserción de Tipo II (figura 14), debe seleccionarse v_k en el camino de v_j a v_i de modo que $v_k \neq v_j$ y $v_k \neq v_{j+1}$ y, además, v_l en el camino de v_i a v_j tal que $v_l \neq v_i$ *y* $v_l \neq v_{i+1}$. La inserción consiste en eliminar los arcos (v_i, v_{i+1}), (v_{l-1}, v_l), (v_j, v_{j+1}) y (v_{k-1}, v_k), y agregar los arcos (v_i, v), (v, v_j), (v_l, v_{j+1}), (v_{k-1}, v_{l-1}) y (v_{i+1}, v_k). Además, debe revertirse el orden de los caminos $(v_{i+1},. . ., v_{l-1})$ y $(_{vl},. . ., v_j)$.

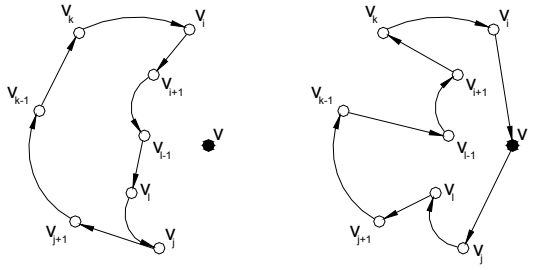


Figura 14. Inserción GENI del tipo II.

El algoritmo GENI, para construir una ruta, inicializa la misma con 3 nodos al azar. Luego selecciona un nodo *v* que no esté en la ruta y realiza la mejor de todas las inserciones de Tipo I y Tipo II restringidas a las p-vecindades. El proceso se repite hasta que todos los nodos hayan sido visitados.

4.3.2.4.4.- Transferencias Cíclicas

Son movimientos multi-ruta que eliminan clientes de una ruta y los incorporan a otra ruta de forma cíclica. Fueron introducidos por Thompson y Psaraftis [241] y describen un mecanismo general ***b-ruta k-vértice***, según el cual se considera una permutación cíclica de *b* rutas y se mueven *k* vértices de una ruta a la siguiente, en la permutación elegida.

4.3.3.- Metaheurísticos aplicados al VRP

Se ha producido un importante avance en los últimos años en el desarrollo de metaheurísticos aplicados al VRP. Una gran variedad de estrategias se pueden citar, aunque se pueden agrupar en tres grandes categorías [242]: búsqueda tabú, algoritmos basados en poblaciones y algoritmos basados en mecanismos de aprendizaje. Esta información se puede encontrar en Cordeau et al. [243] y en Gendreau et al. [244] desarrollada con mayor profundidad.

4.3.3.1.- Búsqueda Tabú

Entre los múltiples algoritmos desarrollados, se pueden citar los siguientes:

Algoritmo de Osman [128]: el vecindario de una solución se obtiene mediante intercambios de clientes entre pares de rutas mediante un λ-intercambio. El coste del movimiento se evalúa de forma heurística. Si se realiza un movimiento de clientes de una ruta a otra, el movimiento contrario, es declarado tabú; el periodo tabú es fijo y se determina en función de las características del problema. Se utiliza también un criterio de aspiración. Para la elección de la mejor solución dentro del vecindario, se puede aplicar la estrategia *best Admissible*, que elige la mejor solución factible, o la estrategia *First Best Admissible* que elige la primera solución admisible.

Algoritmo Taburoute: propuesto por Gendreau et al. [129], acepta soluciones no factibles durante la búsqueda, considerando conjuntos de rutas que visitan exactamente una vez a cada cliente y permitiendo la violación de las restricciones de capacidad y de longitud máxima; se define una función objetivo que depende, entre otros, de los excesos de capacidad y de longitud, de tal forma que se penalizan estos dos excesos, debidamente ponderados, obligando a las soluciones a pertenecer a la región factible. Se introducen estrategias de diversificación mediante penalización de la concentración de clientes sobre un cliente determinado.

Algoritmo de Taillard [245]: realiza una partición del conjunto de clientes, resolviendo cada partición como un VRP independiente de los demás mediante búsqueda tabú. Si el depósito está centrado, se puede realizar una partición según consideraciones geométricas; en el resto de casos, se puede recurrir a árboles de cubrimiento formados por los caminos mínimos desde el depósito a cada cliente. El vecindario se define mediante el algoritmo de Osman con λ=1.

Otras estrategias que se pueden citar son el algoritmo de Xu y Kelly [246], memorias adaptativas de Rochat y Taillard [130], *Ejection Chains* de Rego y Roucariol [247], *Granular Tabu Search* [248].

Dentro del VRPTW Cordeau et al. [249] proponen un método de búsqueda tabú denominado *Unified Tabu Search*. En este caso, el tamaño de la flota es conocido. Se permiten soluciones no factibles, así como violaciones de la restricción de capacidad, ventanas de tiempo, que son penalizadas como en el algoritmo de Taburoute. Las ponderaciones que afectan a los excesos se actualizan en cada iteración. Se eliminan clientes de una ruta y se insertan en otras, siendo declarados tabú los movimientos contrarios; así mismo, se utilizan estrategias de diversificación.

4.3.3.2.- Algoritmos basados en poblaciones

En el contexto del VRP la codificación de soluciones como cadenas de bits no es el proceso más habitual, siendo usados procedimientos más naturales; así, Prins [250] transforma una solución VRP en una solución TSP eliminando los delimitadores de rutas y reconstruyendo la solución al final del proceso. Las mutaciones, de carácter aleatorio, también son sustituidas, a menudo, por otras estrategias de tipo heurístico en la descendencia, conocidas como algoritmos meméticos. Un método similar, denominado procedimiento de memoria adaptativa, ha sido desarrollado por Rochat y Taillard [130] para el VRP; en primer lugar ejecutan una búsqueda tabú y guardan las mejores soluciones; en un segundo paso, recombinan las mismas, a través de procedimientos de cruce y búsqueda local, con la esperanza de conseguir mejores soluciones. Una implementación de este método se puede encontrar en Tarantilis y Kiranoudis [251].

4.3.3.3.- Algoritmos basados en mecanismos de aprendizaje

Estos algoritmos incluyen, entre otros, redes neuronales y colonias de hormigas. Ghaziri [252] y Scumann y Retzko [253] aplican por primera vez las redes en la resolución del VRP aunque, posteriormente, se abandona esta línea de trabajo.

4.3.4.- Valoración General

Todos los metaheurísticos anteriormente citados, así como algunas implementaciones con híbridos de los anteriores, han sido utilizados en la solución del VRP. En el área de búsqueda local, Li et al. [254] han desarrollado un sencillo, aunque eficiente, heurístico *record-to-record*. Implementaciones de búsqueda tabú, altamente satisfactorias, se pueden encontrar en el algoritmo de Taillard [245], así como una evolución consistente en el uso de una memoria adaptativa (Rochat y Taillard [130]). Como indica Laporte [242] estos dos algoritmos han proporcionado algunas de las mejores soluciones obtenidas en las instancias de prueba.

Otra implementación de búsqueda tabú, conocida como UTSA (*Unified Tabu Search Algorithm*) de Cordeau et al. [132] muy flexible y aplicada a gran variedad de problemas de rutas; los heurísticos de propósito general, de Pisinger y Ropke [255] y Derigs y Kaiser [256] con su heurístico basado en la estrategia *hill climber*. Recientemente Kytöjoki et al. [257] han desarrollado un heurístico basado en búsqueda por entornos variables (VNS) capaz de resolver instancias de gran tamaño.

Dentro de los algoritmos genéticos, cabe destacar los procedimientos de memoria adaptativa (AMP), el algoritmo de Prins [250] y el algoritmo AGES, de Mester y Bräysy [258, 259] que combina algoritmos genéticos, búsqueda por entornos variables y búsqueda tabú granular (el concepto ‘granular’, atribuido a Toth y Vigo [248], consiste en eliminar varias caras o arcos poco prometedores del grafo). Otro algoritmo memético, con resultados muy interesantes, es el desarrollado por Nagata [260], donde se produce una relajación de las restricciones de capacidad que maneja a través de una función penalizadora, mientras explora los vecindarios.

Los algoritmos de hormigas, uno muy prometedor es el D-Ants, de Reimann et al. [261] que, será probablemente uno de los mejores algoritmos aplicados al VRP. Más información sobre estos avances, se puede encontrar en Laporte [242].

5.- LA GESTIÓN DE LOS SISTEMAS DE RECOGIDA DE RU

5.- LA GESTIÓN DE LOS SISTEMAS DE RECOGIDA DE RU

5.1.- INTRODUCCIÓN

Se entiende por residuo cualquier producto en estado sólido, líquido o gaseoso procedente de un proceso de extracción, transformación o utilización, que carente de valor para su propietario, éste decide abandonar; dentro de este grupo, se puede hablar de residuos urbanos (RU), que son los que se originan en la actividad doméstica y comercial de ciudades y pueblos.

La gestión de los residuos afecta, en general, a todas las actividades, personas y espacios, convirtiéndose en problema no solo por lo que representa en términos de recursos abandonados, sino por la creciente dificultad para encontrar lugares que permitan su acomodo correcto, es decir, sin interferir con las estrictas regulaciones existentes en el marco medioambiental y en los diferentes niveles de administración: europeo, estatal, autonómico, local, etc.

Una segunda cuestión, no menos importante, afecta al aspecto logístico, pues se generan unos residuos que deben ser recogidos, de forma adecuada, y transportados a centros de transferencia (en el caso de entornos rurales), de donde parten hacia centros de clasificación y/o tratamiento. Esto implica, necesariamente, la existencia de una red o sistema de recogida de los mismos, compuesta, generalmente, por vehículos adaptados para dicho cometido.

En la actualidad, la gestión de estos sistemas corresponde a los municipios, como así lo indica la *Ley 7/85 de Bases de Régimen Local*, de 2 de abril [262], en su *artículo 25, apartado 1*, donde dice literalmente:

> *1. El Municipio, para la gestión de sus intereses y en el ámbito de sus competencias, puede promover toda clase de actividades y prestar cuantos servicios públicos contribuyan a satisfacer las necesidades y aspiraciones de la comunidad vecinal.*
>
> ***2. El Municipio ejercerá en todo caso, competencias, en los términos de la legislación del Estado y de las Comunidades Autónomas, en las siguientes materias:***
>
> *a) Seguridad en lugares públicos.*
>
> *b) Ordenación del tráfico de vehículos y personas en las vías urbanas.*
>
> *c) Protección civil, prevención y extinción de incendios.*
>
> *d) Ordenación, gestión, ejecución y disciplina urbanística; promoción y gestión de viviendas; parques y jardines, pavimentación de vías públicas y conservación de caminos y vías rurales.*
>
> *e) Patrimonio histórico-artístico.*
>
> *f) Protección del medio ambiente.*
>
> *g) Abastos, mataderos, ferias, mercados y defensa de usuarios y consumidores.*
>
> *h) Protección de la salubridad pública.*
>
> *i) Participación en la gestión de la atención primaria de la salud.*
>
> *j) Cementerios y servicios funerarios.*
>
> *k) Prestación de los servicios sociales y de promoción y reinserción social.*

> ***l) Suministro de agua y alumbrado público; servicios de limpieza viaria, de recogida y tratamiento de residuos, alcantarillado y tratamiento de aguas residuales.***
>
> *ll) Transporte público de viajeros.*
>
> *m) Actividades o instalaciones culturales y deportivas: ocupación del tiempo libre; turismo.*
>
> *n) Participar en la programación de la enseñanza y cooperar con la Administración educativa en la creación, construcción y sostenimiento de los centros docentes públicos, intervenir en sus órganos de gestión y participar en la vigilancia del cumplimiento de la escolaridad obligatoria.*
>
> *3. Sólo la ley determina las competencias municipales en las materias enunciadas en este artículo, de conformidad con los principios establecidos en el artículo 2.*

Bhat [263] estima en torno al 75-80% el coste de la recogida y traslado de residuos a sus centros, respecto al total del presupuesto destinado a la gestión global de RU; de aquí, se puede observar que un programa de trabajo debidamente diseñado e implementado puede aportar unos beneficios importantes desde el punto de vista del coste del servicio.

Otro aspecto importante a destacar, como indica Bautista et al. [264], es el diseño adecuado del sistema de recogida en función del tipo de residuo generado pues, como se indica en el *artículo 26* de la Ley anteriormente citada, *se debe proceder a la clasificación de residuos en municipios mayores de 5000 habitantes*. Las diferentes fracciones (vidrio, metal, orgánicos, etc.) sufren tratamientos distintos según sus características. En el caso particular de un entorno rural, el diseño del sistema de recogida se refiere a la fracción orgánica, ya que es la que más afecta, desde el punto de vista medioambiental (olores, roedores, etc.) a la normal convivencia de los vecinos.

Es muy extensa la legislación existente en materia de residuos sobre la que se sustentan jurídicamente todas las actuaciones; comenzando por la legislación europea, con más de 25 normas, entre Directivas y Reglamentos; más de 10 normas a nivel estatal, entre Leyes y Reales Decretos y, por último, más de 7 normas a nivel autonómico —caso de Castilla y León— entre Leyes, Decretos y Decretos Legislativos. Todas ellas se pueden encontrar citadas, en un Suplemento al nº 37 del Boletín Oficial de Castilla y León de 23 de febrero de 2005.

La norma sobre la que se sustenta el modelo previsto de gestión a nivel de Castilla y león es el *Plan Regional de Ámbito Sectorial de Residuos Urbanos y Residuos de Envases de Castilla y León 2004-2010* donde se fijan los principios generales bajo los cuales se desarrolla el plan: *principio de prevención, principio de cooperación y responsabilidad compartida, implicación de las Administraciones Locales, en particular Diputaciones y grandes Ayuntamientos y aplicación del lema "quien contamina, paga".*

En el documento anteriormente citado, se recogen los principios rectores y directrices que han de marcar las pautas a seguir para implantar el plan, recogiendo, entre otras, las actuaciones a desarrollar en cuanto a infraestructas de tratamiento de residuos urbanos, de envases y especiales, en el ámbito geográfico de Castilla y León, durante el perido 2004-2010.

Los datos de partida utilizados, se pueden encontrar en publicaciones periódicas *on-line* del Ministerio de Medio Ambiente, Rural y Marítimo, del Gobierno de España [265], así como las metas que se fijan a nivel de la Unión Europea para el año horizonte 2010; de ahí se deducen las estrategias a seguir con el fin de alcanzar los objetivos fijados.

5.2.- SISTEMAS DE GESTIÓN DE RECOGIDA DE RU

5.2.1.- Modelos Iniciales

La idea de modelizar un sistemas de recogida de RU no es nueva. Un resumen de diferentes modelos desarrollados en los años 70, años 80 y primeros de los 90, se puede encontrar en Gottinger [266], en MacDonald [267], en Berger et al. [268] y en Tanskanen [269]. En ellos se describen, entre otros, un modelo de programación dinámica entera, propuesto por Baetz y Neebe [270], un modelo multiperiodo y multiregional, desarrollado por Everett y Modak [271] y un modelo de programación no lineal (MIMES/WASTE), de Sundberg et al. [272]. Todos estos artículos están esparcidos en un periodo de 12 años y recogen la evolución de los diferentes sistemas de gestión de sistemas de recogida a lo largo de los años inicialmente indicados; en Morrissey et al. [273] se detallan con profundidad las diferentes metodologías empleadas (modelos basados en el análisis coste-beneficio, modelos basados en el análisis del ciclo de vida y modelos basados en análisis de decisión multicriterio) así como sus beneficios y limitaciones.

5.2.2.- Técnicas Heurísticas

La introducción de técnicas heurísticas en la modelización de un sistema de recogida de RU, se puede hacer de dos formas: la primera, en entornos urbanos (grandes municipios) donde la propia entidad local es responsable de la gestión de sus propios residuos, siendo posible modelizar el problema como un *Capacited Arc Routing Problem* (CARP) o problema por arcos; en este caso, la generación de residuos se produce a lo largo de diferentes puntos de cada calle (arcos), donde los nodos representan los cruces de las mismas; la segunda, aparece en la recogida de RU en entornos rurales, donde cada vez es más frecuente la gestión mancomunada, es decir, mediante una *entidad legalmente constituida por agrupación de municipios*,[3] que decide gestionar de forma conjunta un determinado servicio; en este caso, se debe hablar de un *Capacited Vehicle Routing Problem* (CVRP) o problema por nodos, ya que se considera que la generación de residuos se produce en los diferentes núcleos urbanos (nodos), unidos por vías de comunicación (arcos) que, por lo general, se pueden recorrer en ambos sentidos, de forma que de lo que se trata es de determinar los ciclos que recorran los diferentes nodos sin violar las diferentes restricciones que se puedan imponer.

Dentro de los sistemas modelizados como un CARP la literatura es extensa; una primera modelización es aportada por Gelders y Cattrysse [274]. Este problema no puede ser resuelto de forma exacta, por lo que, en la práctica, se recurre al uso de diferentes estrategias heurísticas para resolverlo. Beltrami y Bodin [176], y también Ulusoy [275] desarrollan una metodología consistente en calcular una única ruta y agrupar posteriormente (*route first – cluster second*) a fin de no violar la restricción de capacidad. Otro procedimiento, basado en la exploración del camino y desarrollado por Golden et al. [276], parte de la construcción de un ciclo, simultáneamente con la aplicación de un criterio miope de optimización; en la formación de un ciclo, el arco más prometedor es añadido hasta agotar la capacidad del vehículo, entonces se completa con el regreso al depósito para completar el ciclo. Más trabajos en esta línea se pueden encontrar en Pearn [277] y Kulcar [278]. Chapleau [279] usa un algoritmo basado en inserciones en paralelo.

Más recientes en el tiempo, se pueden encotrar entre otros, los trabajos de Mourao [280], quien desarrolla una metodología para la resolución de problemas de recogida de residuos aplicando el método *lower-bounding*, Benavent y Soler [281] con su trabajo sobre el RPP (*Rural Postman Problem*), Amberg

[3] Definición de la *Real Academia Española de la Lengua.*

et al. [282] con el CARP Multi-depósito y Ghiani et al. [283] que trabaja sobre el CARP con facilidades intermedias.

La primera aplicación con recocido simulado se atribuye a Eglese [284]; aplicaciones de búsqueda tabú aplicadas al RPP no dirigido, se pueden encontrar en Hertz et al. [285], y en Corberán et al. [286], el RPP mixto. Lacomme y Prins [287, 288] aplican estrategias evolutivas al CARP.

Respecto a sistemas modelizados mediante un CVRP, la literatura es escasa, si bien en Kim et al. [289] se puede encontrar un caso de resolución real de recogida de RU con ventanas de tiempo; además se incluye una colección de problemas VRPTW; otros autores que desarrollan el tema son Sahoo et al. [290] y Amponsah [291].

6.- CONTEXTO: EL ALFOZ DE LARA

6.- CONTEXTO: EL ALFOZ DE LARA

6.1.- INTRODUCCIÓN

La mancomunidad Alfoz de Lara se encuadra dentro del sur-este de la provincia de Burgos, lindando con las provincias de Soria y La Rioja, y estando vertebrada por la carretera Nacional N-234, así como las carreteras regionales BU-825 y BU-925; se completa la comunicación con carreteras provinciales.

Presenta dicha mancomunidad, según datos aportados por el INE y la Diputación Provincial de Burgos, una población de derecho de 7.740 habitantes y una población estival de 24.055 habitantes[4], por lo que se produce un incremento, en los momentos punta, de un 210%, fenómeno por otra parte, frecuente en estas poblaciones, dadas las características geográficas, sociales y turísticas de la zona.

Está formada por 45 núcleos urbanos algunos de ellos, encuadrados dentro de un mismo municipio; los municipios de mayor población son Salas de los Infantes, Huerta del Rey y Hontoria del Pinar. En el siguiente cuadro (tabla 1) se adjunta la relación de núcleos, junto a la población de derecho y estival.

N.º	N.º REG.	NÚCLEO	MUNICIPIO	KM^2	HAB/INV	HAB/VER.
1	1090200	Arauzo de Miel	Arauzo de Miel	57.10	387	450
2	1090217	Arauzo de Salce	Arauzo de Salce	18.50	70	250
3	1090222	Arauzo de Torre	Arauzo de Torre	13.59	107	450
		Arroyo de Salas	Salas de los Infantes		26	85
		Aldea del pinar	Hontoria del Pinar		52	112
4	1090373	Barbadillo de Herreros	Barbadillo de Herreros	64.16	145	1000
5	1090389	Barbadillo del Mercado	Barbadillo del Mercado	13.42	175	750
6	1090392	Barbadillo del Pez	Barbadillo del Pez	20.76	101	990
7	1090623	Cabezón de la Sierra	Cabezón de la Sierra	19.83	75	300
8	1090709	Carazo	Carazo	24.03	50	230
9	1090780	Cascajares de la Sierra	Cascajares de la Sierra	6.81	29	190
		Castrovido	Salas de los Infantes		45	139
10	1091101	Contreras	Contreras	38.12	110	230
		Doñasantos	Doñasantos		63	156
11	1091448	Gallega, La	Gallega, La	17.35	87	350
		Gete	Pinilla de los Barruelos		36	96
12	1091545	Hacinas	Hacinas	7.90	208	640
		Hinojar del Rey	Huerta del Rey		49	102
13	1091638	Hontoria del Pinar	Hontoria del Pinar	80.84	820	3172
		Hoyuelos	Salas de los Infantes		22	67
14	1091736	Huerta de Arriba	Huerta de Arriba	33.18	176	500
15	1091741	Huerta de Rey	Huerta de Rey	97.81	1200	2630
		Iglesiapinta	Iglesiapinta		31	69
16	1091834	Jaramillo de la Fuente	Jaramillo de la Fuente	21.56	31	355
17	1091849	Jaramillo Quemado	Jaramillo Quemado	17.45	17	205

[4] Población estimada en el año 2004, en el que se realiza la toma de datos.

N.º	N.º REG.	NÚCLEO	MUNICIPIO	KM^2	HAB/INV	HAB/VER.
18	1092019	Mamolar	Mamolar	18.32	67	400
19	1092236	Monasterio de la Sierra	Monasterio de la Sierra	5.89	39	330
20	1092267	Monterrubio de la Demanda	Monterrubio de la Demanda	15.11	96	200
		Navas del Pinar	Navas del Pinar		137	280
		Peñalba de Castro	Huerta del Rey		99	312
		Piedrahita de Muño	Piedrahita de Muño		21	45
21	1092680	Pinilla de los Barruecos	Pinilla de los Barruecos	32.52	142	450
22	1092693	Pinilla de los Moros	Pinilla de los Moros	11.01	50	305
		Quintanarraya	Huerta de Rey		81	159
		Quintanilla Urrilla	Quintanilla Urrilla		45	130
23	1093023	Rabanera del Pinar	Rabanera del Pinar	33.24	135	500
24	1093121	Revilla y Ahedo, La	Revilla, La	15.29	130	423
25	1093180	Riocavado de la Sierra	Riocavado de la Sierra	43.32	72	595
26	1093305	Salas de los Infantes	Salas de los Infantes	31.32	2064	4360
27	1093403	San Millán de Lara	San Millán de Lara	33.62	75	351
		Terrazas	Salas de los Infantes		15	45
		Vallejimeno	Vallejimeno		51	110
28	1094141	Valle de Valdelaguna	Huerta de Abajo	92.66	214	1152
29	1094509	Villanueva de Carazo	Villanueva de Carazo	7.38	29	90
30	1094786	Vizcaínos	Vizcaínos	11.48	66	300
					7740	24055

Tabla 1. Relación de núcleos urbanos que componen la mancomunidad Alfoz de Lara.

Hay un aspecto importante a destacar en cuanto a distribución geográfica de la población, que es el siguiente: los tres núcleos urbanos con mayor población, Salas de los Infantes, Huerta del Rey y Hontoria del Pinar, representan el 52% de la población de derecho, siendo de un 42% en el periodo estival (tabla 2).

	INVIERNO	VERANO
Salas de los Infantes	2.064	4.360
Hontoria del Pinar	820	3.172
Huerta de Rey	1.200	2.630
Suma	4.084	10.162
Alfoz de Lara	7.740	24.055
Porcentaje	52.76%	42.24%

Tabla 2. Municipios con mayor población.

En la figura 15 se muestra la ubicación de la mancomunidad Alfoz de Lara dentro de la provincia de Burgos.

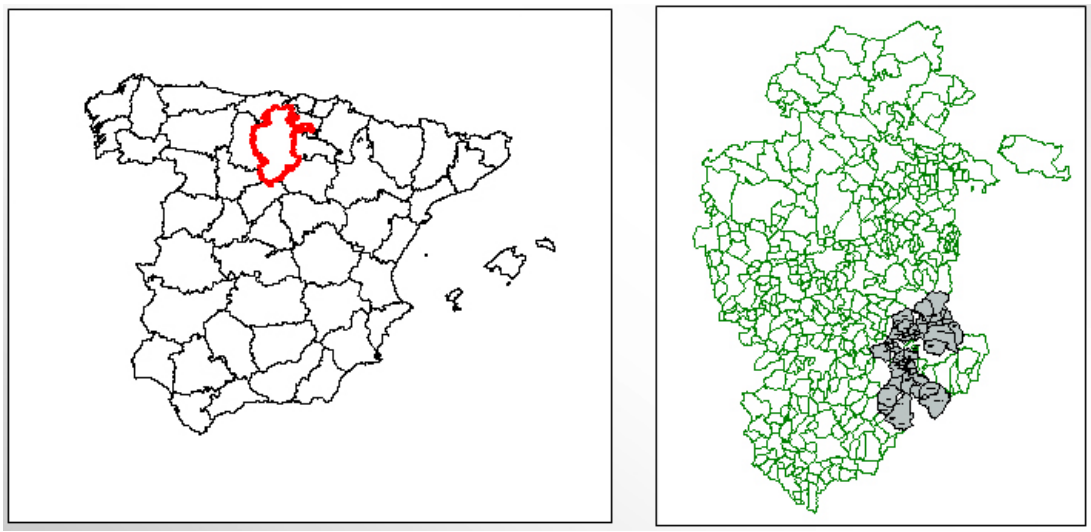


Figura 15. Ubicación de la mancomunidad Alfoz de Lara.

6.2.- ENTORNO GEOGRÁFICO

La zona de enclave de la mancomunidad Alfoz de Lara presenta características fisiográficas propias de la conocida como Sierra de la Demanda, dentro de la cual se enclava. Representa dicha zona la segunda área montañosa de la provincia y forma parte de las estribaciones del sistema Ibérico [292].

Dentro de esta zona se pueden hacer varias sub-unidades:

Sierra de Neila: situada al sur de la Sierra de la Demanda, constituye una alineación montañosa con una morfología propia de grandes cuestas, con presencia de restos de morfología glaciar, como la de la Laguna Negra. Es una zona donde predomina el uso forestal y de aprovechamiento del pino.

Corredor Soria-Burgos: Situado entre las sierras de Neila y Cervera, está formado por materiales terrígenos, que han dado lugar a una zona deprimida, en la que el uso predominante es el de tierras de cultivo y pastizal. Ha sido vía tradicional de comunicación entre Burgos y Soria.

Sierra de Cervera: constituida por relieves cretácicos en cuesta; está situada al sur de la unidad anterior y se encuentra delimitada por materiales terciarios de la cuenca. En esta zona se destaca la presencia de sabinares.

En la siguiente figura (16), se pueden apreciar las distintas unidades fisiográficas dentro de la zona sobre la cual se enclava la mancomunidad Alfoz de Lara.

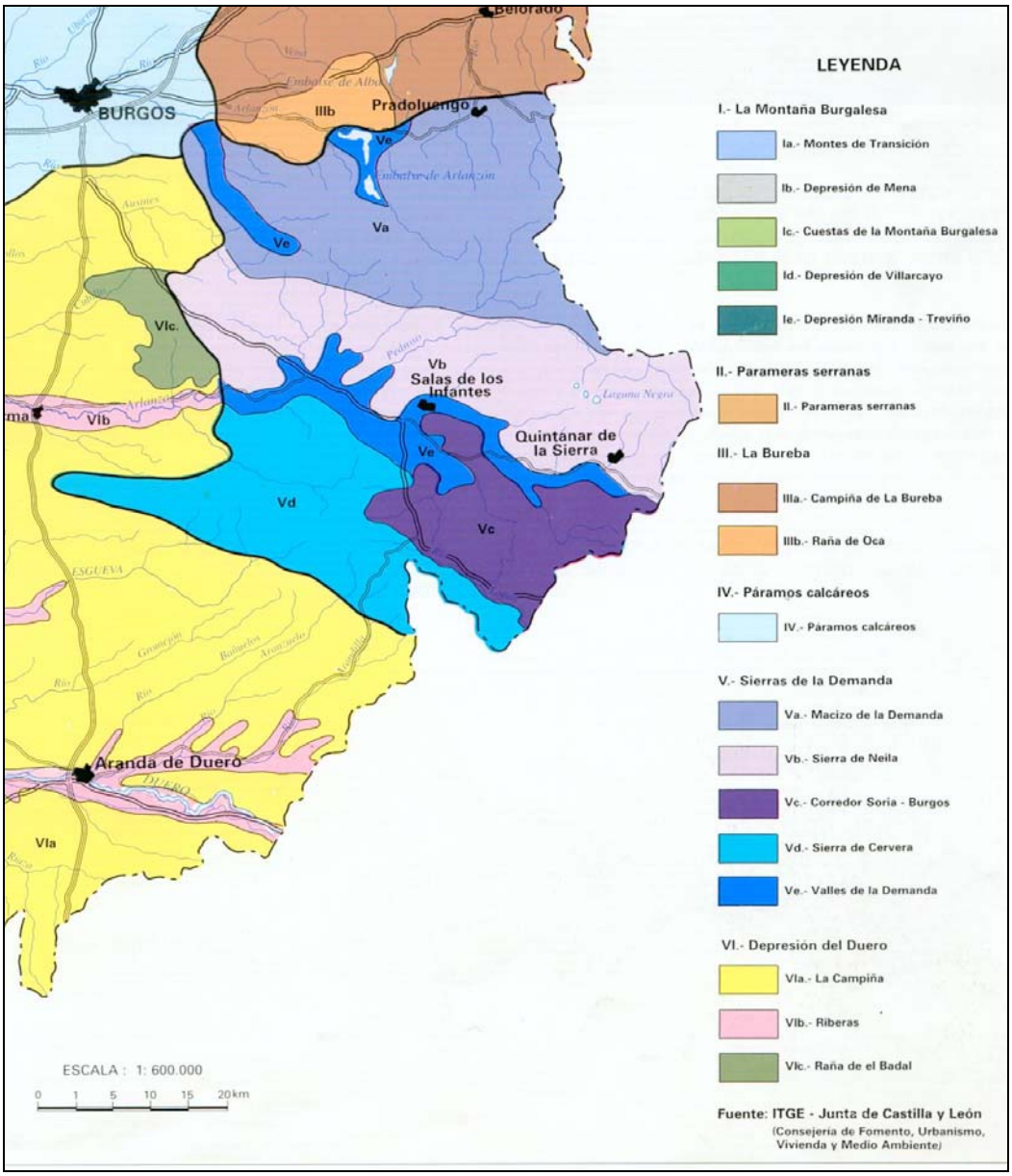


Figura 16. Unidades fisiográficas.
(Fuente: *Atlas del medio hídrico de la provincia de Burgos*)

6.3.- CLIMA

El clima en la provincia de Burgos presenta un marcado carácter continental, con inviernos largos y rigurosos, veranos cortos y escasa humedad durante todo el año en gran parte del territorio. Por su situación sobre la sub-meseta septentrional, presenta una altura media superior a los 800 metros, siendo éstos, factores que determinan su clima y, por tanto, condiciones hidrológicas, vegetación, temperatura, precipitaciones, vientos, orografía, etc.

Presentan los parámetros de la precipitación anual media y la precipitación máxima acumulada, valores mayores que la media de la provincia, pudiendo establecerse un límite entre el Burgos húmedo y el Burgos seco, siguiendo la isoyeta de 600 mm.

La zona de estudio es atravesada por dicha isoyeta, por lo que presenta dos zonas claramente diferenciadas, una zona húmeda: zona nor-este, y otra zona seca: zona sur-oeste (figura 17).

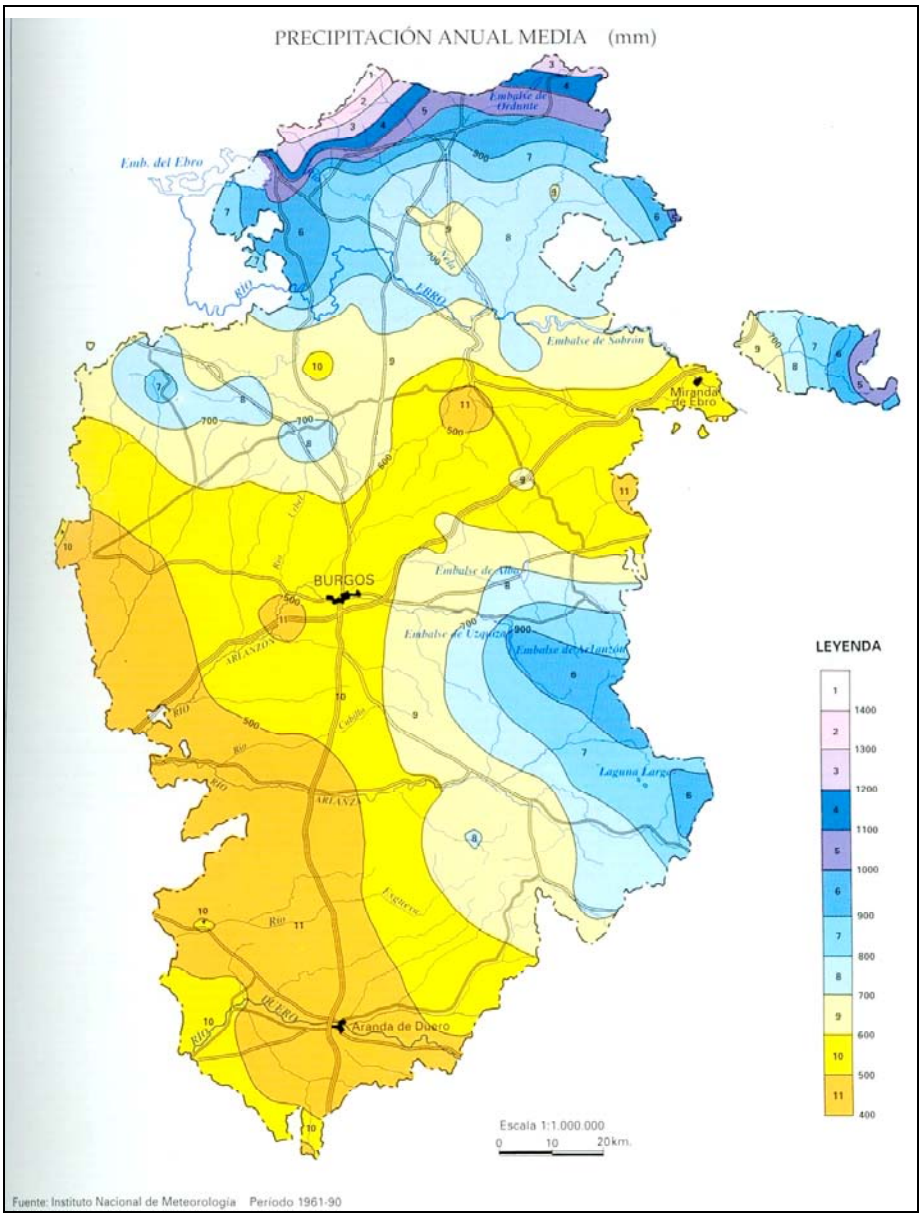


Figura 17. Isoyetas.
(Fuente: *Atlas del medio hídrico de la provincia de Burgos*)

En la zona de sierra, el efecto orográfico produce mayores precipitaciones y temperaturas medias de 8-10º C. No existiendo grandes variaciones anuales. Según disminuye la latitud, se producen incrementos de temperatura, así como una disminución apreciable de las precipitaciones, algo característico del Burgos seco (figura 18).

6.4.- POBLACIÓN

Al igual que en el resto de la provincia, la despoblación ha sido la característica fundamental de esta zona, pudiendo hablar de diferentes periodos.

Hasta los años 50, se produce un crecimiento demográfico, a excepción de los años 20, a raíz de la epidemia de gripe; a partir de los años 50, se produce una despoblación continuada, en el que sus efectivos demográficos se ven reducidos en un 10,4%.

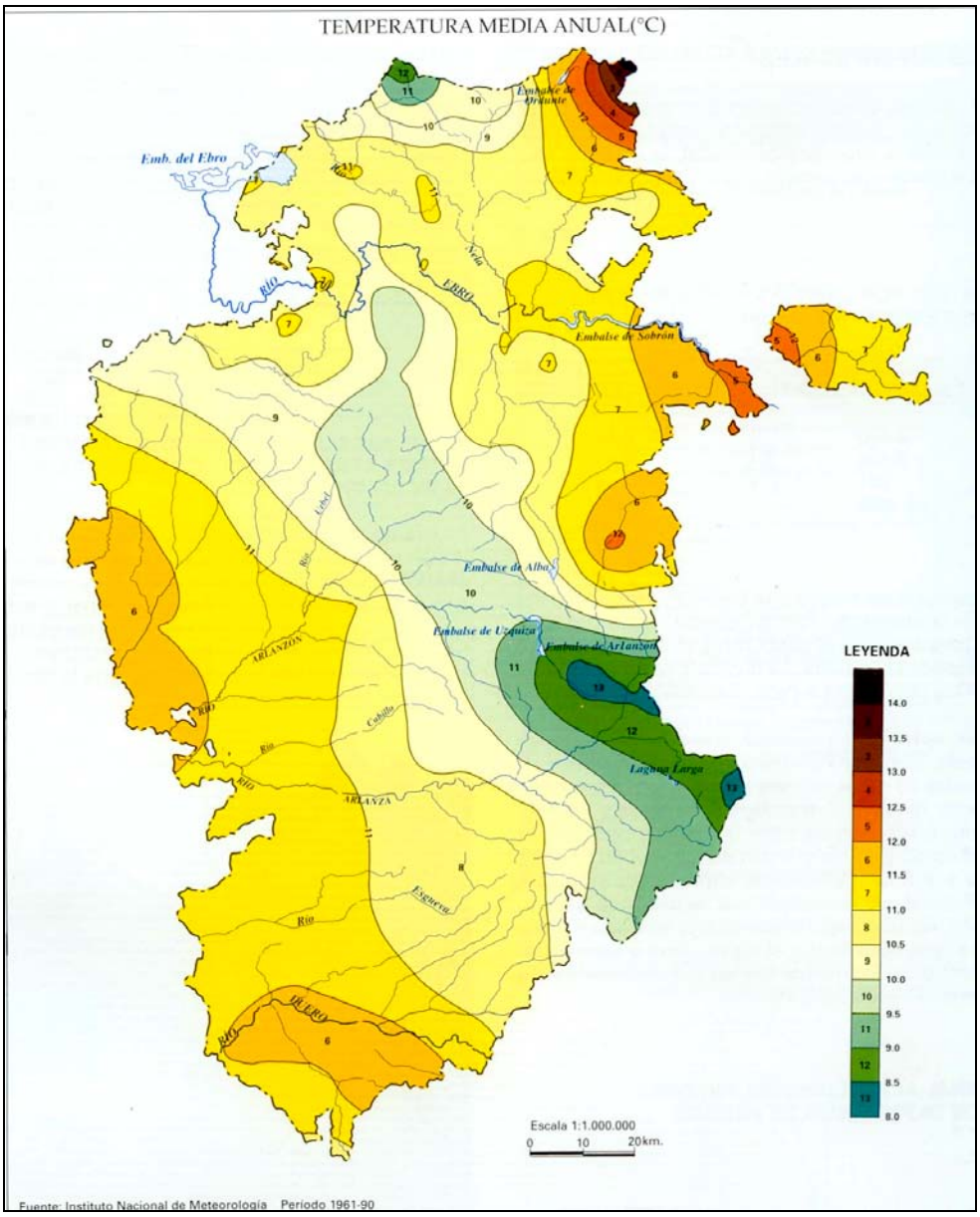


Figura 18. Isotermas.
(Fuente: *Atlas del medio hídrico de la provincia de Burgos*)

El periodo comprendido entre los años 60 y 75 destaca por el éxodo masivo de población hacia Burgos, Aranda de Duero y Miranda de Ebro, debido al nacimiento de industrias, al amparo del Polo de Desarrollo Industrial de Burgos, de 1964. En la siguiente figura (19) se puede observar la evolución de la población en la provincia de Burgos, entre los años 1983 y 1994 siendo visible el crecimiento natural negativo.

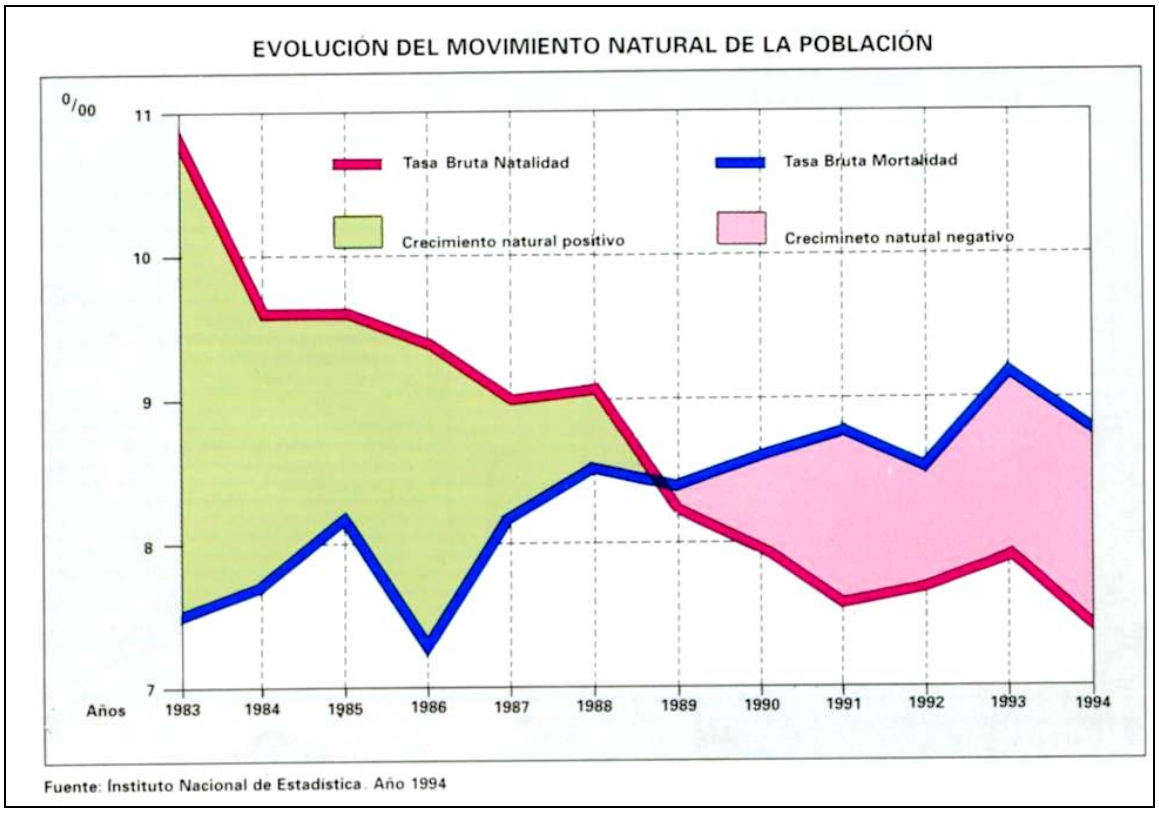


Figura 19. Evolución de la población.
(Fuente: *Atlas del medio hídrico de la provincia de Burgos*)

En la figura 20 se aprecia la densidad de población de los diferentes municipios, observándose una fuerte despoblación en los términos alejados de las cabezas de partido: Burgos, Aranda de Duero, Salas de los Infantes, etc. Se observa así mismo, una mayor actividad en toda la zona de la sierra de Neila, debido ello a una apreciable actividad industrial maderera y de la piedra.

El resto de la zona de estudio, (zonas de Lara y sierra de Cervera) presenta una densidad ciertamente preocupante, siendo claros candidatos a un abandono de pueblos por el escaso interés que las diversas actividades económicas suscitan entre la posible población a ejercerla.

Como línea de tendencia, cabe destacar un decremento de la población, causado por la elevada edad media de la población, y por un sector productivo apoyado en la construcción, en la agricultura y en la ganadería; el cambio de los sistemas de explotación en el sector primario provocan que, para ser medianamente competitivos, haya que proceder a la explotación intensiva en ganadería, así como el trabajo de grandes superficies en agricultura, siendo estos, nichos productivos con grandes barreras de entrada.

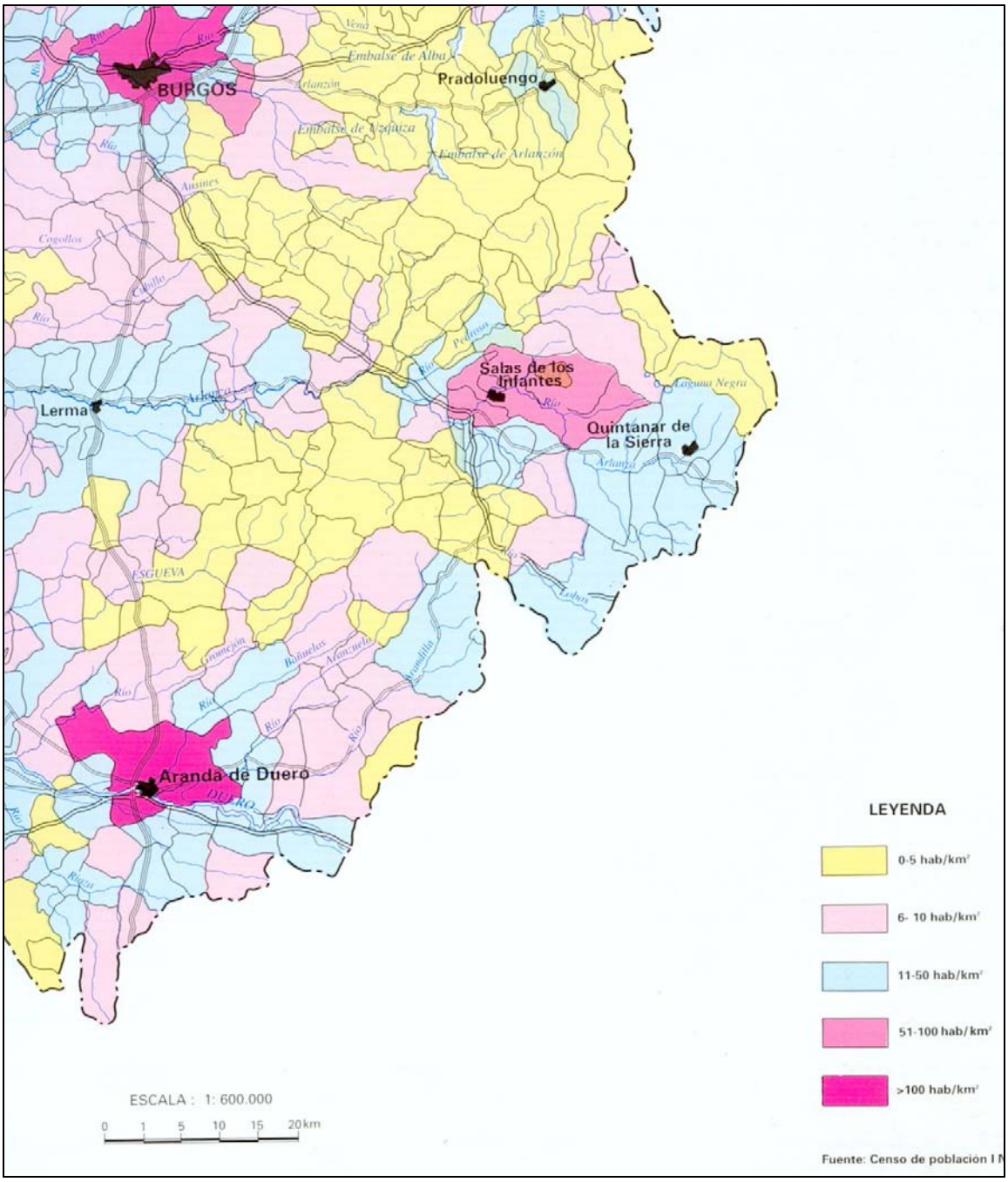


Figura 20. Densidad de población.
(Fuente: *Atlas del medio hídrico de la provincia de Burgos*)

7.- DATOS INICIALES

7.- DATOS INICIALES

7.1.- POBLACIÓN DE CÁLCULO

A efectos de cálculo de la población, y tal y como se puede apreciar más adelante, se consideran dos periodos con dos planificaciones distintas. El periodo INVIERNO, que transcurre entre el 1 de septiembre al 30 de junio del año siguiente, y el periodo de VERANO, que transcurre entre el 1 de julio y 31 de agosto.

En el siguiente cuadro (tabla 3) se adjuntan los datos de población de núcleos que se van a emplear en la estimación de generación de RU.

	Población	Hab/Inv	Hab/Ver		Población	Hab/Inv	Hab/Ver
1	Ahedo de la Sierra	**20**	**65**	25	Iglesiapinta	**31**	**69**
2	Arauzo de Miel	**387**	**450**	26	Jaramillo de la Fuente	**31**	**355**
3	Arauzo de Salce	**70**	**250**	27	Jaramillo Quemado	**17**	**205**
4	Arauzo de Torre	**107**	**450**	28	Mamolar	**67**	**400**
5	Arroyo de Salas	**26**	**85**	29	Monasterio de la Sierra	**39**	**330**
6	Aldea del Pinar	**52**	**112**	30	Monterrubio de la Demanda	**96**	**200**
7	Barbadillo de Herrera	**145**	**1000**	31	Navas del Pinar	**137**	**280**
8	Barbadillo del Mercado	**175**	**750**	32	Peñalba de Castro	**99**	**312**
9	Barbadillo del Pez	**101**	**990**	33	Piedrahita de Muño	**21**	**45**
10	Cabezón de la Sierra	**75**	**300**	34	Pinilla de los Barruecos	**142**	**450**
11	Carazo	**50**	**230**	35	Pinilla de los Moros	**50**	**305**
12	Cascajares de la Sierra	**29**	**190**	36	Quintanarraya	**81**	**159**
13	Castrovido	**45**	**139**	37	Quintanilla Urrilla	**45**	**130**
14	Contreras	**110**	**230**	38	Rabanera del Pinar	**135**	**500**
15	Doñasantos	**63**	**156**	39	Revilla, La	**110**	**358**
16	Gallega, La	**87**	**350**	40	Riocavado de la Sierra	**72**	**595**
17	Gete	**36**	**96**	41	Salas de los Infantes	**2064**	**4360**
18	Hacinas	**208**	**640**	42	San Millán de Lara	**75**	**351**
19	Hinojar del Rey	**49**	**102**	43	Terrazas	**15**	**45**
20	Hontoria del Pinar	**820**	**3172**	44	Tolbaños de abajo	**54**	**288**
21	Hoyuelos	**22**	**67**	45	Tolbaños de arriba	**53**	**288**
22	Huerta de Abajo	**107**	**576**	46	Vallejimeno	**51**	**110**
23	Huerta de Arriba	**176**	**500**	47	Villanueva de Carazo	**29**	**90**
24	Huerta de Rey	**1200**	**2630**	48	Vizcaínos	**66**	**300**

Tabla 3. Datos de población usados en el estudio.

En esta tabla se adjuntan datos de todos los núcleos urbanos, estando algunos de ellos agrupados dentro de un mismo municipio; los municipios que más entidades agrupan, son Salas de los Infantes, Huerta del Rey y Hontoria del Pinar. Los municipios de la zona de la sierra de Neila, Huerta de Abajo y Los Tolbaños, se agrupan dentro de otra entidad supramunicipal denominada Valle de Valdelaguna.

Se observa un fuerte incremento estacional del 210% en periodo estival y que obliga a una planificación distinta para dicho periodo, siempre y cuando se pretenda mantener los parámetros de calidad, dentro de unos límites razonables. Respecto a la densidad de población, en la siguiente figura se puede apreciar las densidades de los diferentes municipios (figura 21).

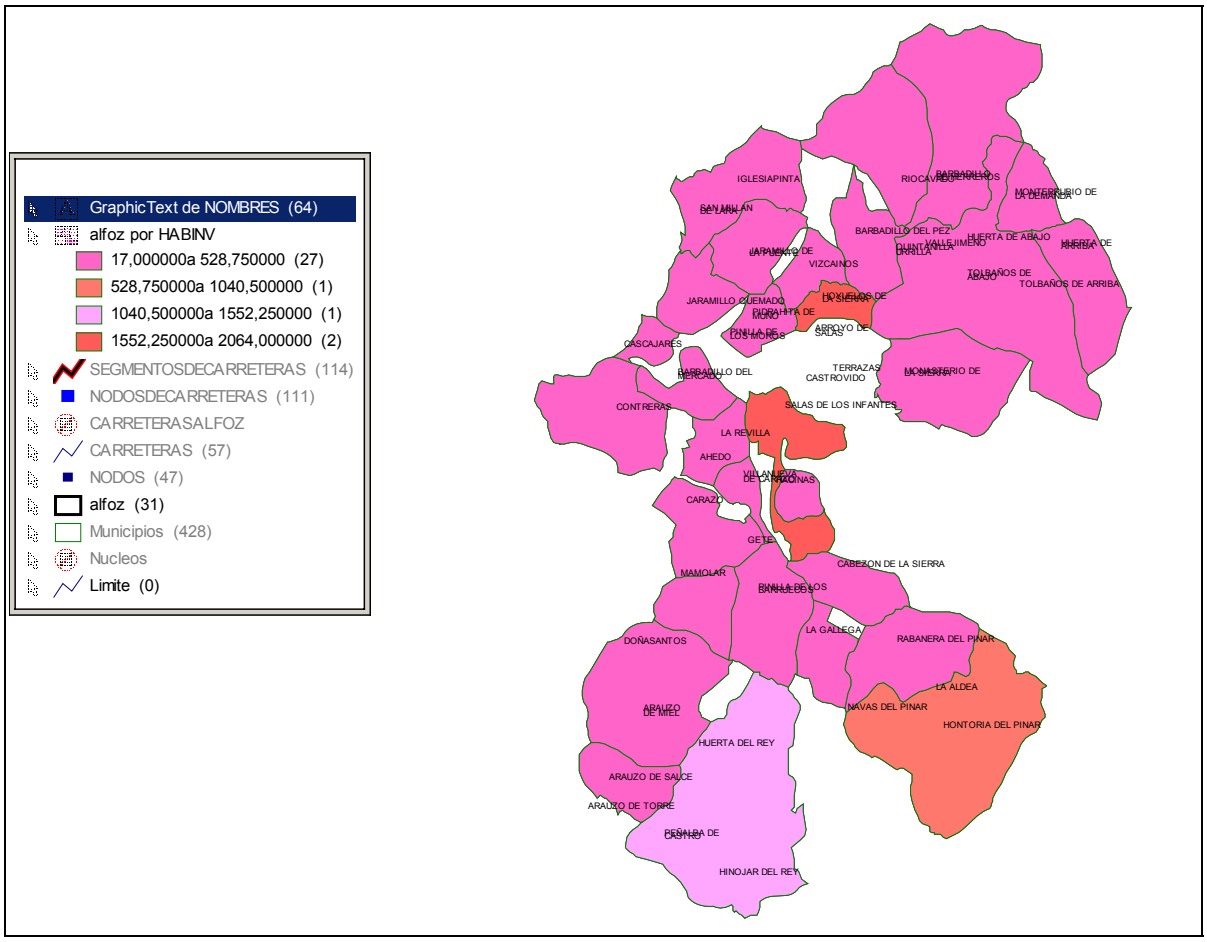


Figura 21. Densidad de población.

En cuanto a la previsión en un año horizonte determinado, no se prevé un cambio en la tendencia de variación de población, siendo el descenso de población la tónica prevista en los próximos años para Castilla y León. Es por lo que se toman dichos valores de población como valores máximos para el cálculo de las diferentes opciones (tabla 4).

PERIODO	FECHA	POBLACIÓN
INVIERNO	1/9 – 30/6	7.740
VERANO	1/7 – 31/8	24.055

Tabla 4. Datos de población por periodo utilizados en el estudio.

7.2.- RESIDUOS

Del análisis de los partes de recogida y de los albaranes de entrega en el Centro de Tratamiento de Residuos de Burgos, se deduce lo siguiente:

- La producción de residuos en el año 2002 fue de 1.998.480 kg.
- Se observa un incremento estacional medio de un 176% aproximadamente en el periodo estival, debido, en parte, al atractivo turísitico de la sierra (figura 22).

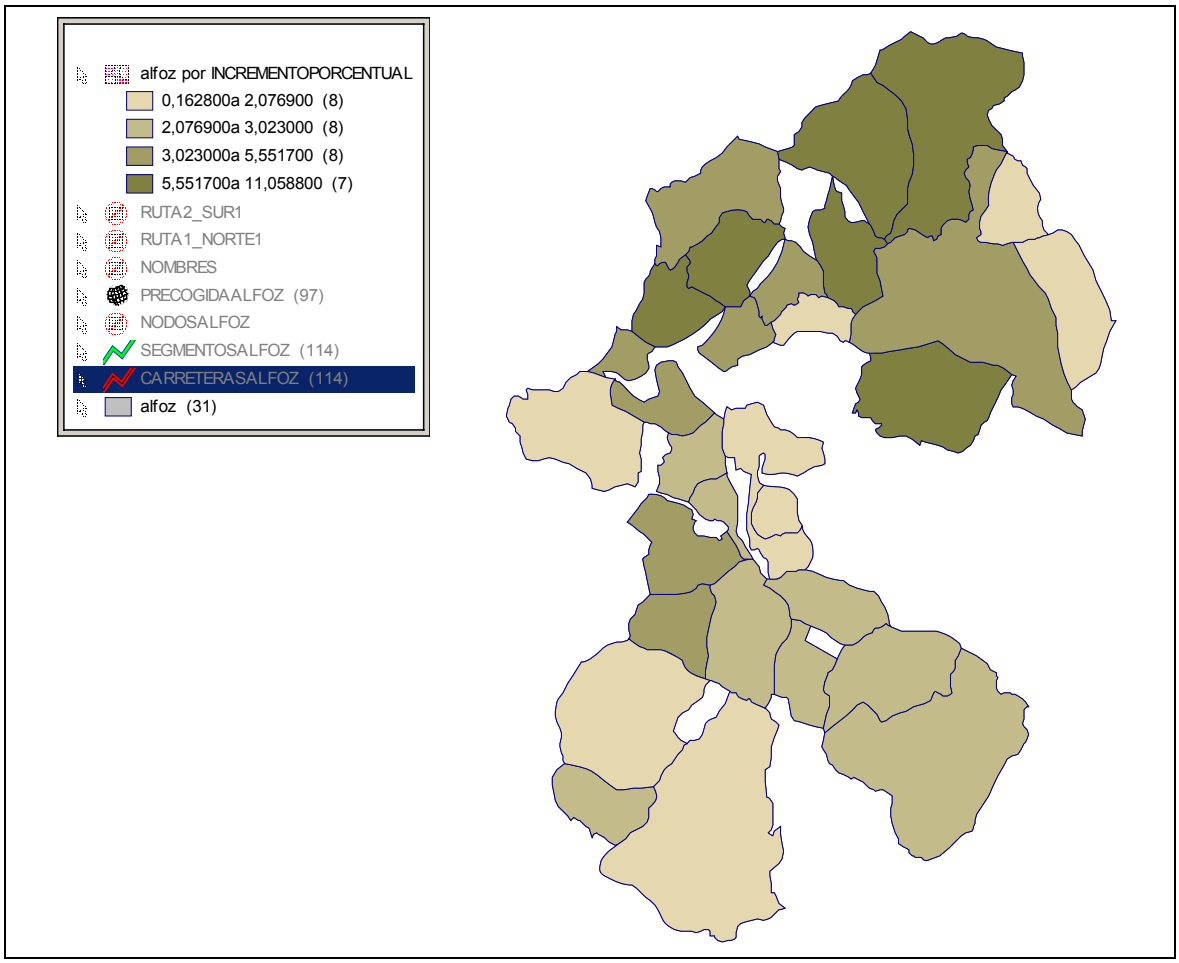


Figura 22. Incremento estival per capita de generación de RU.

- Se observa además, como queda reflejado en el siguiente cuadro (tabla 5), una variación importante en la producción per cápita de RU; hay que indicar que esta variación puede ser debida a que, aunque se considera un incremento de un 210% global, no necesariamente tienen que coincidir todos los máximos en cada población, mientras que la producción de RU sí que es real.

MES	Kg/hab x día
enero	1,77
febrero	1,65
marzo	1,55
abril	1,91
mayo	1,92
junio	2,04
julio	2,20
agosto	1,47
septiembre	1,71
octubre	2,44
noviembre	2,32
diciembre	2,41
MEDIA	**1,95**
INVIERNO	2,00
VERANO	1,69

Tabla 5. Datos de producción per cápita de residuos.

A efectos de cálculo, se adopta el valor abajo indicado (tabla 6). Dicho valor es claramente inferior a los obtenidos de los datos de la mancomunidad, resultando ser más acorde con los ratios aportados por las diferentes administraciones, obtenidos de muestreos más extensos. Debe tenerse en cuenta la cada vez mayor conciencia medioambiental que provoca, entre otras acciones, la clasificación de residuos domésticos que siguen procesos diferentes, siendo menor, por tanto, la fracción orgánica objeto de este estudio.

PERIODO	KG/PERSONA x DÍA
INVIERNO	**1.00**
VERANO	**1.00**

Tabla 6. Datos de producción per cápita para cada periodo utilizados en estudio.

La mancomunidad dispone, en la actualidad, de tres vehículos de recogida, si bien en el periodo de toma de datos, disponía de dos, diseñados para transportar los RU y que, mediante un sistema hidráulico, permiten la carga automática de los contenedores. En las siguientes figuras (23 y 24), se observa la variabilidad bruta de producción de RU; se observa que uno de los vehículos (BU-2022-Z) atiende la producción de RU en zona valle, mientras que el vehículo (BU-3433-S) actúa en las puntas de producción (correspondiente principalmente al periodo estival) y con apoyos puntuales en Semana Santa y Navidad.

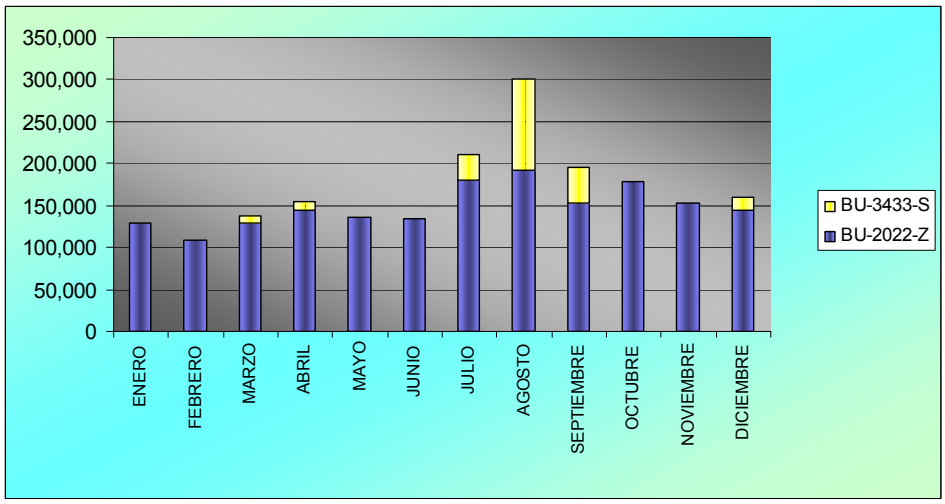


Figura 23. Carga transportada por los vehículos (en kg.)

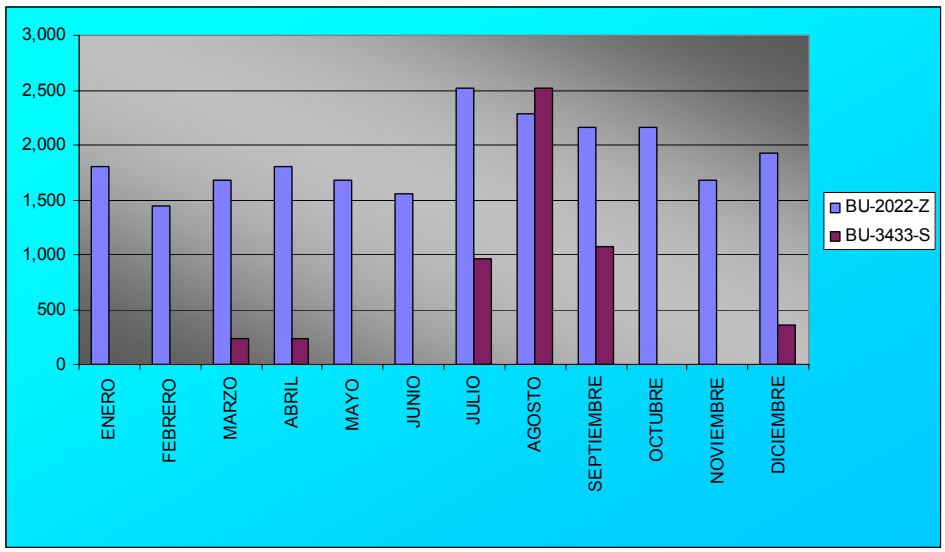


Figura 24. Kilómetros recorridos por cada vehículo.

7.3.- CONFIGURACIÓN ACTUAL DE LAS RUTAS

El sistema actual de recogida en la mancomunidad Alfoz de Lara implica la ejecución de un ciclo de cinco días de duración durante el periodo invernal con un camión recogedor, y que varía durante el periodo estival a otro ciclo con dos camiones que ejecutan a su vez dos ciclos.

Para analizar la eficacia del servicio, se han tenido en cuanta, a partir de los partes de recogida, los días transcurridos entre recogida y recogida de contenedor por meses y en cada núcleo. El resultado se adjunta en las dos primeras columnas, en los campos MEDIA y DESV.STDR (desviación típica). Al analizar de forma cualitativa dichos datos, se observa que aparecen dos tipos de valores predominantes entorno a 3, y en torno a 6; curiosamente, corresponden todos excepto uno, a núcleos de mayor población y con mayor poder dentro de los órganos de decisión.

Se observa además que, en dichos núcleos, la desviación es menor, lo que se puede interpretar como que, en caso de no llegar a cumplir la ruta, se quedan sin recoger núcleos de menor entidad.

MUNICIPIO	MEDIA	DESV. STDR
Arauzo de Miel	3,36	0,56
Arauzo de Salce	6,63	1,55
Arauzo de Torre	6,63	1,55
Arroyo de Salas	6,09	1,77
Aldea del pinar	6,48	1,48
Barbadillo de Herreros	6,09	1,77
Barbadillo del Mercado	3,53	0,25
Barbadillo del Pez	6,09	1,77
Cabezón de la Sierra	6,54	1,40
Carazo	6,77	1,42
Cascajares de la Sierra	6,43	1,82
Castrovido	6,09	1,77
Contreras	6,83	1,32
Doñasantos	6,50	1,51
Gallega, La	6,54	1,40
Gete	6,54	1,40
Hacinas	6,77	1,42
Hinojar del Rey	6,63	1,55
Hontoria del Pinar	2,41	0,18
Hoyuelos	6,07	1,52
Huerta de Arriba	6,09	1,77
Huerta de Rey	2,38	0,13
Iglesiapinta	6,85	1,12
Jaramillo de la Fuente	6,85	1,12
Jaramillo Quemado	6,43	1,82
Mamolar	6,63	1,55
Monasterio de la Sierra	6,77	1,42
Monterrubio de la Demanda	6,09	1,77
Navas del Pinar	6,54	1,40
Peñalba de Castro	6,59	1,62
Piedrahita de Muño	6,09	1,77
Pinilla de los Barruecos	6,50	1,48
Pinilla de los Moros	6,09	1,77
Quintanarraya	6,63	1,55

MUNICIPIO	MEDIA	DESV. STDR
Quintanilla Urrilla	6,09	1,77
Rabanera del Pinar	6,50	1,49
Revilla y Ahedo, La	6,69	1,49
Riocavado de la Sierra	6,12	1,72
Salas de los Infantes	2,38	0,13
San Millán de Lara	6,73	1,12
Terrazas	6,77	1,42
Vallejimeno	6,09	1,77
Valle de Valdelaguna	6,09	1,77
Villanueva de Carazo	7,04	1,20
Vizcaínos	6,09	1,77

Tabla 7. Indicadores de calidad del servicio actualmente prestado.

Analizando los datos anteriormente expuestos, en función del número de núcleos con valores medios estables, y el número de habitantes, se obtiene el siguiente cuadro (figura 25).

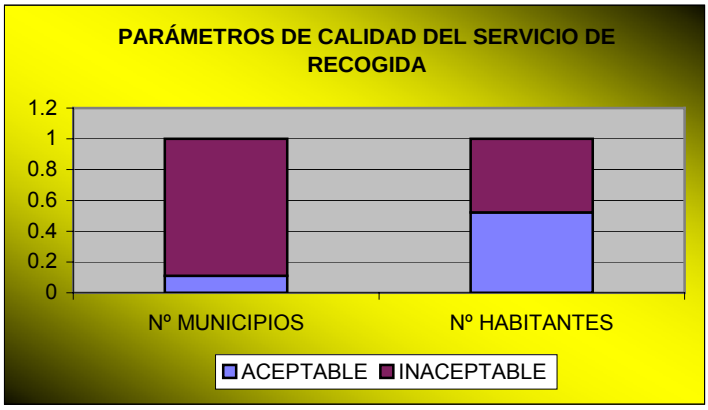


Figura 25. Parámetros de calidad actuales.

En en siguiente cuadro (tabla 8) se indican el total de kilómetros mensuales para cada vehículo, así como número de viajes y carga transportada.

MES	KILOGRAMOS		TOTAL KG	N.º TOTAL DE VIAJES		TOTAL KM		TOTAL KM
	BU-2022-Z	BU-3433-S		BU-2022-Z	BU-3433-S	BU-2022-Z	BU-3433-S	
ENERO	128.340	0	128.340	15	0	1.800	0	1.800
FEBRERO	108.800	0	108.800	12	0	1.440	0	1.440
MARZO	129.460	8.320	137.780	14	2	1.680	240	1.920
ABRIL	143.840	10.420	154.260	15	2	1.800	240	2.040
MAYO	136.460	0	136.460	14	0	1.680	0	1.680
JUNIO	134.100	0	134.100	13	0	1.560	0	1.560
JULIO	179.400	31.940	211.340	21	8	2.520	960	3.480
AGOSTO	192.720	107.860	300.580	19	21	2.280	2.520	4.800
SEPTIEMBRE	153.560	42.660	196.220	18	9	2.160	1.080	3.240
OCTUBRE	178.300	0	178.300	18	0	2.160	0	2.160
NOVIEMBRE	152.680	0	152.680	14	0	1.680	0	1.680
DICIEMBRE	144.040	15.580	159.620	16	3	1.920	360	2.280
TOTAL	**1.781.700**	**216.780**	**1.998.480**	**189**	**45**	**22.680**	**5.400**	**28.080**

Tabla 8. Kilómetros, número de viajes y carga mensual de cada vehículo.

7.4.- RUTAS EXISTENTES EN LA ACTUALIDAD

La descripción de las rutas, por día, con expresión de los núcleos recogidos, kilómetros recorridos y carga realizada, se adjunta a continuación. No obstante, previamente se adjunta en la siguiente figura el grafo descriptivo de las vías de comunicación existentes en la zona, vertebrada por la N-234 (Sagunto-Burgos), de la que parten sendas carreteras regionales: BU-825 y BU-925, y queda complementado con las carreteras de orden provincial que dan acceso a núcleos de menor entidad (figura 26).

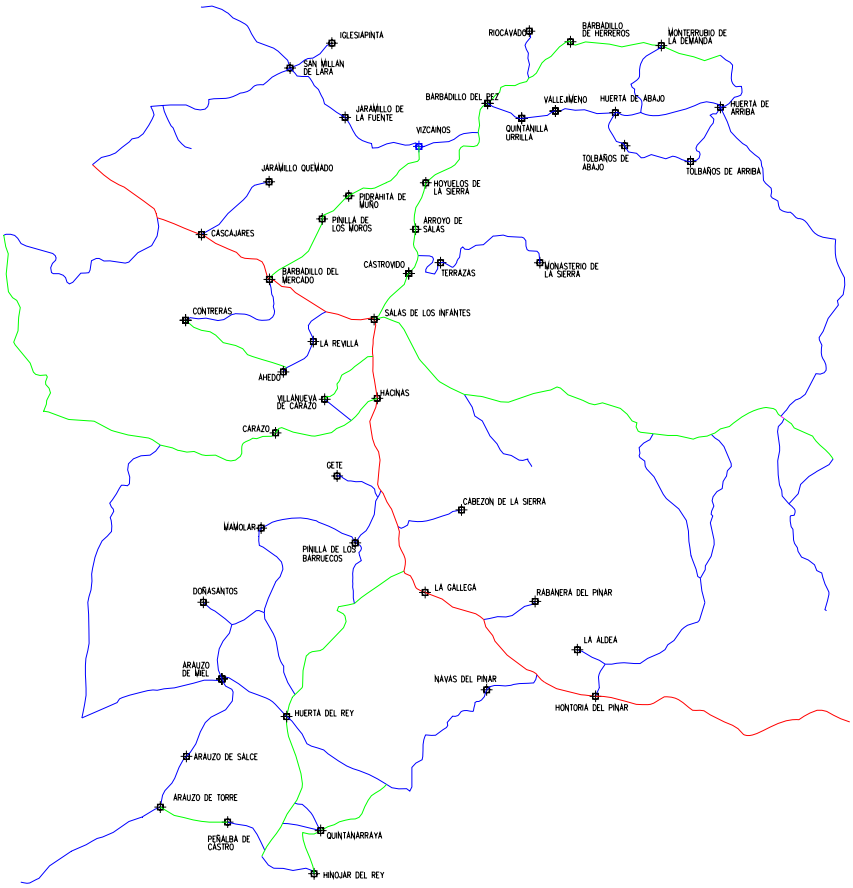


Figura 26. Vías de comunicación existentes.

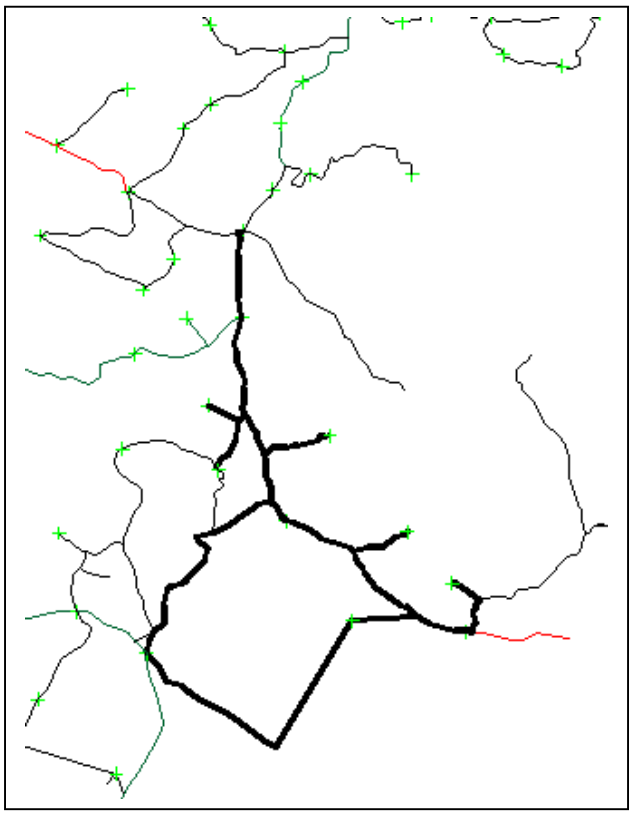
En rojo se aprecia la N-234, que partiendo de Sagunto, pasa, entre otras, por las localidades de Hontoria del Pinar, La Gallega, Hacinas, Salas de los Infantes, Barbadillo del Mercado, etc.

En verde intenso, se encuentran las regionales BU-825 y BU-925, que pasan entre otras, por Carazo, Huerta del Rey, Arroyo de Salas, Hoyuelos, Barbadillo del Pez, Riocavado, etc.

En marrón-verde, se aprecian el resto de carreteras, que pertenecen a categoría provincial, y que unen el resto de localidades.

A continuación se incluyen las rutas que componen el ciclo de recogida en periodo invernal. Para cada una de ellas se incluye kilometraje, etc.

CICLO INVIERNO

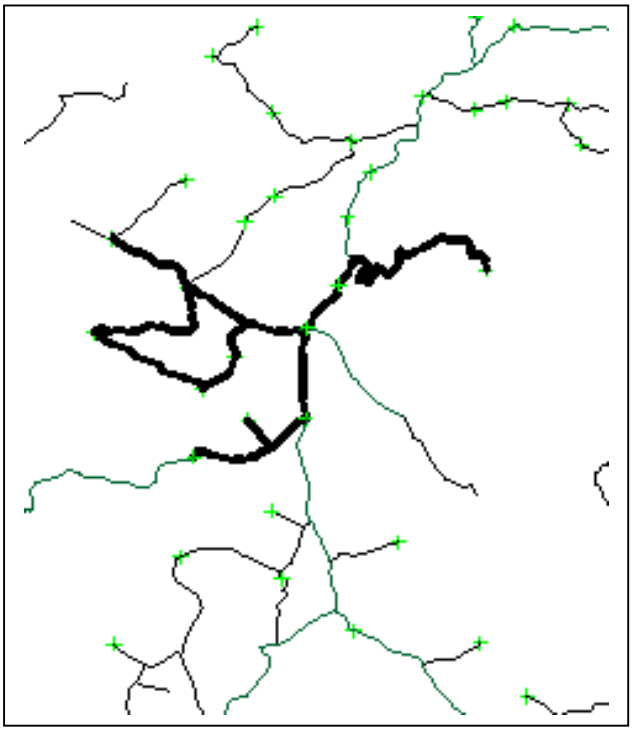


CICLO INVIERNO
RUTA 1

KILÓMETRO EN RUTA: 59 KM
POBLACIÓN RECOGIDA: 4.748

SECUENCIA:

SALAS DE LOS INFANTES - GETE-PINILLA DE LOS BARRUECOS - LA GALLEGA - RABANERA DEL PINAR - LA ALDEA - HONTORIA DEL PINAR - HUERTA DEL REY - CABEZÓN DE LA SIERRA - SALAS DE LOS INFANTES.

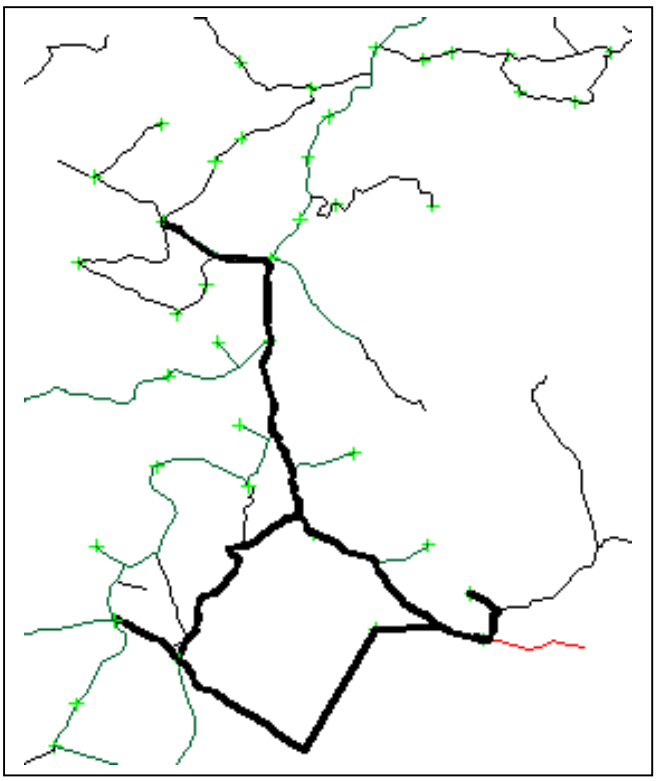


CICLO INVIERNO
RUTA 2

KILÓMETRO EN RUTA: 85 KM
POBLACIÓN RECOGIDA: 627

SECUENCIA:

SALAS DE LOS INFANTES – CASTROVIDO – TERRAZAS – MONASTERIO DE LA SIERRA – HACINAS – VILLANUEVA DE CARAZO – BARBADILLO DEL MERCADO – CONTRERAS – AHEDO – LA REVILLA – CASCAJARES DE LA SIERRA – JARAMILLO QUEMADO.

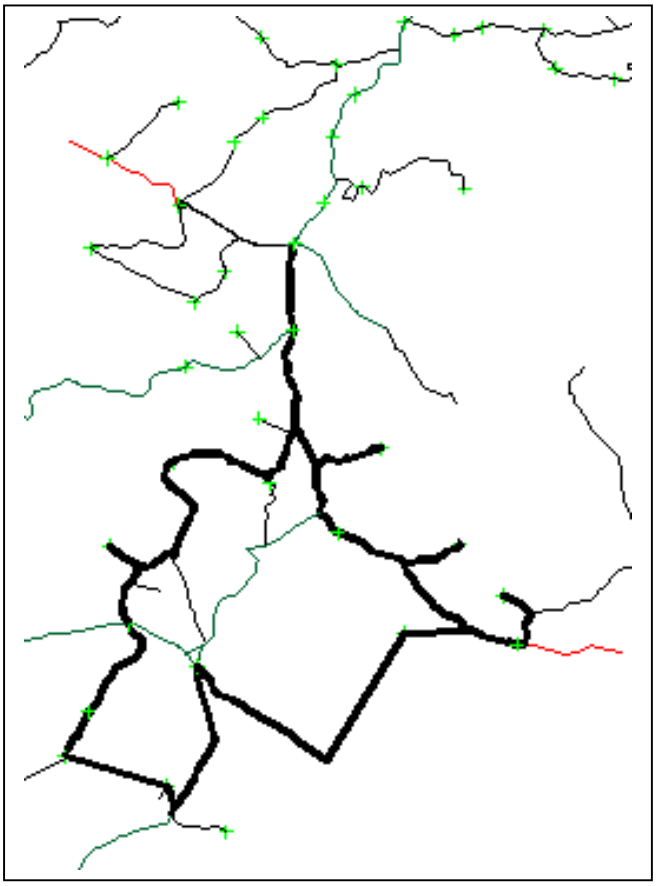


CICLO INVIERNO
RUTA 3

KILÓMETRO EN RUTA: 108 KM
POBLACIÓN RECOGIDA: 4.646

SECUENCIA

SALAS DE LOS INFANTES – HONTORIA DEL PINAR – HUERTA DEL REY ARAUZO DE MIEL – BARBADILLO DEL MERCADO.

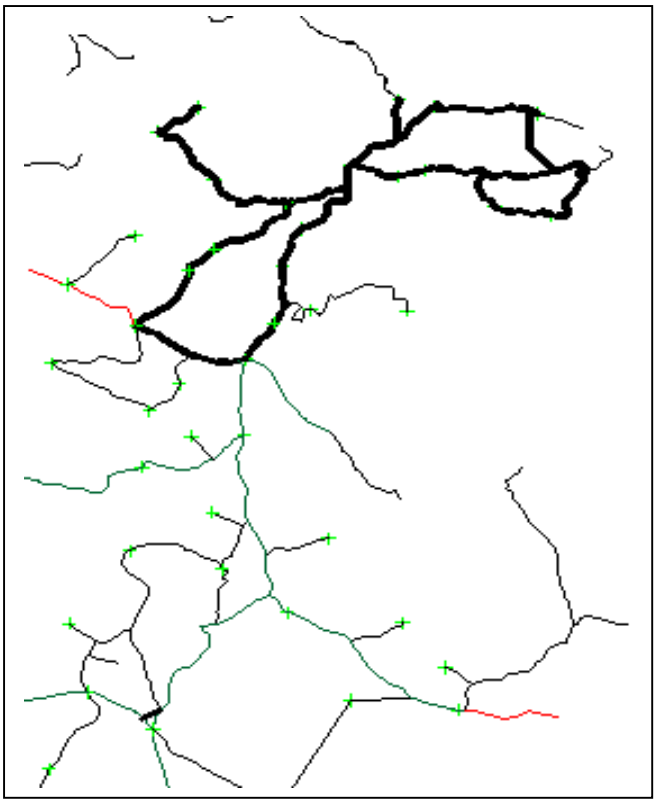


CICLO INVIERNO
RUTA 4

KILÓMETROS EN RUTA: 109 KM
POBLACIÓN RECOGIDA: 5.007

SECUENCIA

SALAS DE LOS INFANTES – HONTORIA DEL PINAR – HUERTA DEL REY – QUINTANARAYA – HINOJAR DEL REY – ARAUZO DE TORRE – ARAUZO DE SALCE – ARAUZO DE SALCE – ARAUZO DE MIEL – DOÑASANTOS – MAMOLAR – SALAS DE LOS INFANTES.



CICLO INVIERNO
RUTA 5

KILÓMETRO EN RUTA: 109 KM
POBLACIÓN RECOGIDA: 1.442

SECUENCIA

SALAS DE LOS INFANTES – CASTROVIDO – ARROYO DE SALAS – HOYUELOS DE LA SIERRA – BARBADILLO DEL PEZ – QUINTANILLA URRILLA – VALLEJIMENO – HUERTA DE ABAJO – TOLBAÑOS DE ABAJO – TOLBAÑOS DE ARRIBA – HUERTA DE ARRIBA-MONTERRUBIO DE LA DEMANDA – BARBADILLO DE LOS HERREROS – RIOCAVADO DE LA SIERRA – VIZCAÍNOS – JARAMILLO DE LA FUENTE – SAN MILLÁN DE LARA – IGLESIAPINTA – PIEDRAHITA DE MUÑO – PINILLA DE LOS MOROS – BARBADILLO DEL MERCADO

CICLO DE VERANO

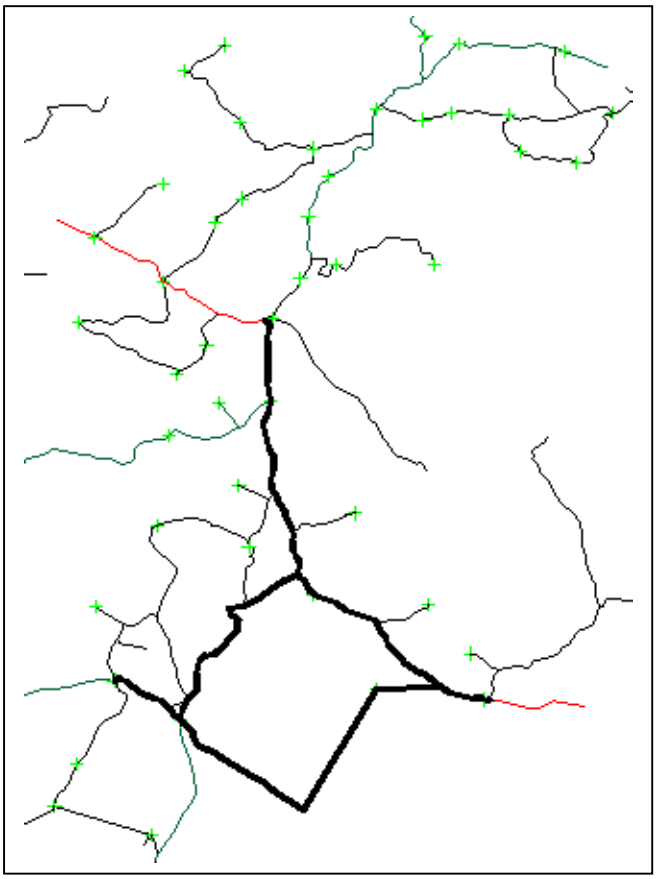
Se inicia el ciclo de verano en la segunda semana de julio (15 de julio) y finaliza sobre el 15 de septiembre; los RU son recogidos por dos camiones, que ejecutan simultáneamente un ciclo cada dos días el primero y cada tres o cuatro, el segundo.

El primer camión (BU-2022-Z) ejecuta un ciclo de dos días; el primer día (RUTA 1A) recoge a los municipios: Salas de los Infantes, Huerta del Rey- Arauzo de Miel Hontoria del Pinar y Salas de los Infantes. El segundo día (RUTA 2A) recoge al resto de Arauzos, Aldea del Pinar, Doñasantos, La Gallega, Gete, Hinojar, Mamolar, Navas ,Peñalba de Castro, Pinilla de los Barruecos, Quintanarraya y Rabanera del Pinar.

El segundo camión ejecuta un ciclo de tres días (variable) de los que uno, no recoge; el primer día (RUTA 1B) recoge Arroyo de Salas, Hoyuelos de la Sierra, Barbadillo del Pez, Los Tolbaños, Huerta de Arriba, Barbadillo del Mercado, Vizcaínos, Piedrahita de Muño, Pinilla de los Moros. El segundo día (RUTA 2B) recoge: Carazo, Contreras, Hacinas, Iglesiapinta, Jaramillo de la Fuente, Monasterio de la Sierra, La Revilla y Ahedo, San Millán de Lara, Terrazas.

A continuación se detalla de forma gráfica cada una de las rutas.

CICLO VERANO

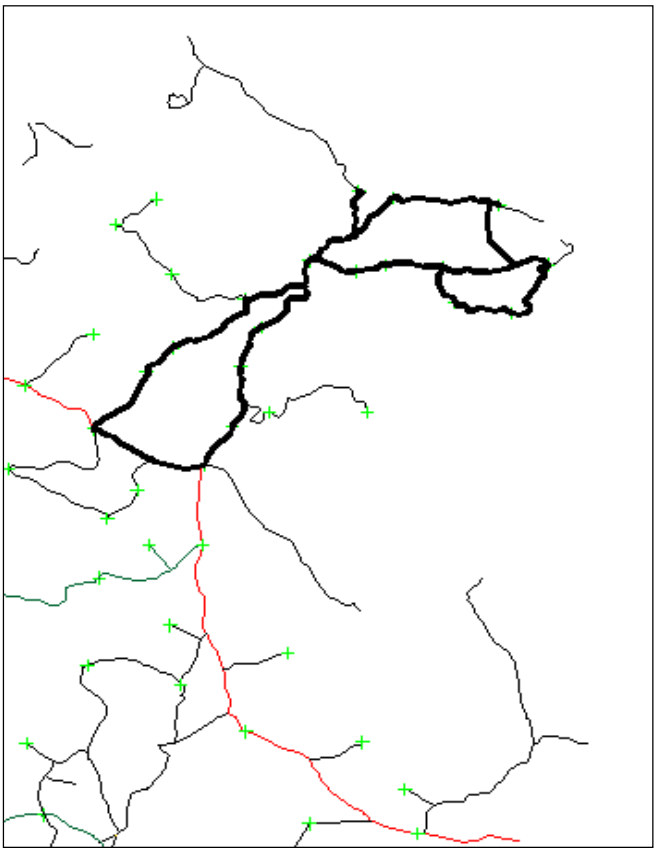


RUTA 1-A

VEHÍCULO BU-2022-Z
DURACIÓN CICLO: DOS DÍAS

SECUENCIA

SALAS DE LOS INFANTES – HUERTA DEL REY – ARAUZO DE MIEL – HONTORIA DEL PINAR – SALAS DE LOS INFANTES.

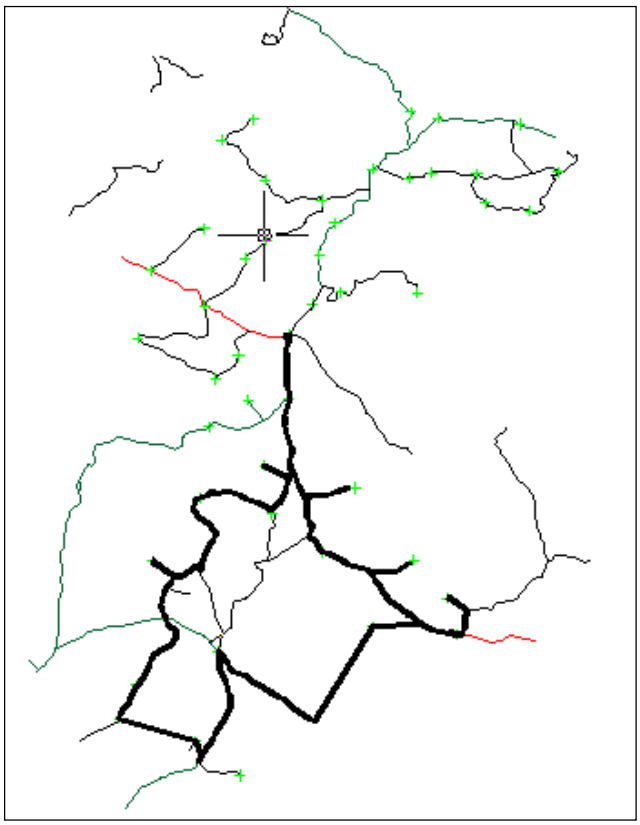


RUTA 1-B

VEHÍCULO BU-3433-S
DURACIÓN CICLO: TRES DÍAS

SECUENCIA

SALAS DE LOS INFANTES – CASTROVIDO – ARROYO DE SALAS – HOYUELOS DE LA SIERRA – BARBADILLO DEL PEZ – VALLEJIMENO-QUINTANILLA URRILLA- HUERTA DE ABAJO-TOLBAÑOS DE ABAJO–TOLBAÑOS DE ARRIBA-HUERTA DE ARRIBA – MONTERRUBIO DE LA DEMANDA – BARBADILLO DE HERREROS – RIOCAVADO- VIZCAÍNOS- PIEDRAHITA DE MUÑO – PINILLA DE LOS MOROS -. BARBADILLO DEL MERCADO – SALAS DE LOS INFANTES.

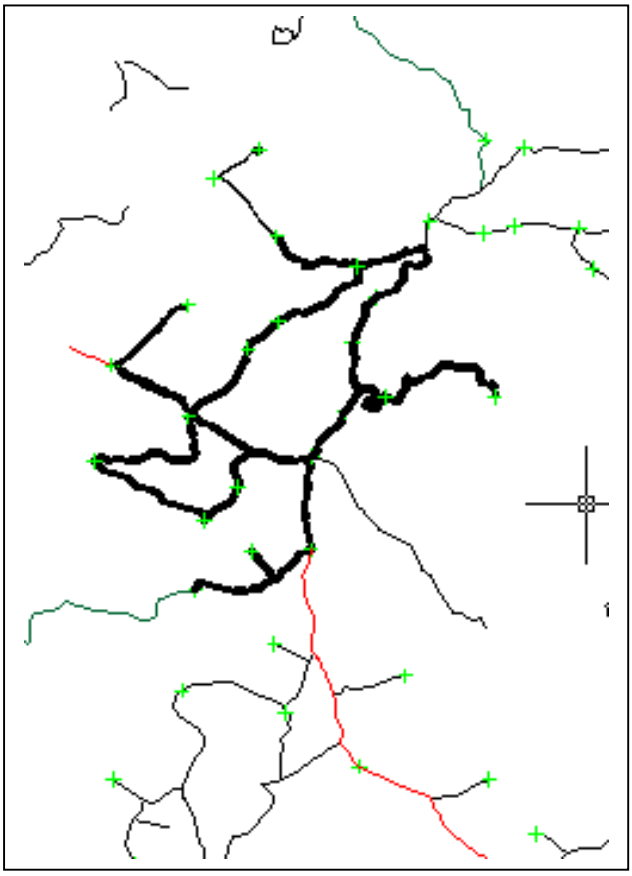


RUTA 2 A

Vehículo BU-2022-Z
Duración ciclo: dos días

SECUENCIA

SALAS DE LOS INFANTES – GETE – PINILLA DE LOS BARRUECOS – MAMOLAR – DOÑASANTOS – ARUZO DE TORRE – ARAUZO DE SALCE – PEÑALBA DE CASTRO – HINOJAR DEL REY – QUINTANARRAYA – NAVAS DEL PINAR – ALDEA DEL PINAR – RABANERA DEL PINAR – CABEZÓN DE LA SIERRA – SALAS DE LOS INFANTES.



RUTA 2 B

VEHÍCULO BU-3433-S
DURACIÓN CICLO: TRES DÍAS

SECUENCIA

SALAS DE LOS INFANTES – CARAZO – VILLANUEVA DE CARAZO – HACINAS – TERRAZAS-MONASTERIO DE LA SIERRA – JARAMILLO DE LA FUENTE – IGLESIAPINTA – LA REVILLA – AHEDO – SALAS DE LOS INFANTES.

8.- MODELIZACIÓN DEL PROBLEMA

8.- MODELIZACIÓN DEL PROBLEMA

El problema que aquí se aborda consiste en el diseño de un sistema de recogida de la fracción orgánica de RU en una mancomunidad de municipios del sur-este de la provincia de Burgos, así como su traslado a un centro de transferencia (situado en el término municipal de Salas de los Infantes), donde son cargados a un vehículo de gran tonelaje, que los transporta a un centro de tratamiento de residuos en Aranda de Duero, también en la provincia de Burgos.

Este esquema de procesado de residuos urbanos, con traslado de los mismos a centros de tratamiento, lo es así en virtud de lo recogido en el *SUPLEMENTO AL N.º 37 del BOCYL*, que recoge el *DECRETO 18/2005, de 17 de febrero, por el que se aprueba el Plan Regional de Ámbito Sectorial de Residuos Urbanos y Residuos de Envases de Castilla y León 2004-2010* [293].

En dicho suplemento se recoge, de forma muy extensa, la estrategia a seguir respecto a la gestión de todos los residuos con origen en suelo Castellano-Leones, y que se sustenta en una serie de políticas como son: la aplicación del principio de prevención como primera prioridad, la aplicación del principio de cooperación y responsabilidad compartida, la implicación de la Administración Local, especialmente las Diputaciones Provinciales y los grandes Ayuntamientos, la aplicación de las mejores técnicas disponibles en la gestión de residuos urbanos y residuos de envases, el desarrollo de una política que afectará a todos los ciudadanos, puesto que hará recaer los costes de la gestión sobre el productor del residuo o, en su caso, sobre los responsables de la puesta en el mercado de productos que con su uso se transforman en residuos, en aplicación del principio *quien contamina, paga*.

En el párrafo anterior, se puede leer *...aplicación de las mejores técnicas disponibles en la gestión de residuos urbanos...*; es pues, objeto de esta tesis, **la optimización del sistema de recogida de R.U. a dichos municipios a fin de minimizar el coste total del servicio y mejorar la calidad del mismo**. Se plantea, por tanto, un problema de optimización biobjetivo, intentando la aproximación a la curva de eficiencia.

8.1.- PLANTEAMIENTO GENERAL

La mancomunidad Alfoz de Lara presenta una población de derecho de 7740 habitantes, repartidos en 48 núcleos urbanos, pertenecientes a 30 municipios; dicha población se eleva hasta los 24055 habitantes en verano debido, en parte, al atractivo turístico de la zona así como a los vínculos familiares en la misma.

8.1.1.- Estado actual

En la actualidad, la población deposita los residuos en unos contenedores de 250 l que están distribuidos en todos los núcleos, en número suficiente para evitar su colmatación. Los vehículos de la mancomunidad (dos en el periodo de toma de datos, tres en la actualidad) recogen, de forma periódica y según la ruta establecida, dichos contenedores mediante un mecanismo elevador especialmente diseñado para ello; tras recoger todos los contenedores del núcleo, continúan hasta el siguiente núcleo para repetir la operación. Cuando los camiones alcanzan su máximo de capacidad, deben finalizar la ruta a fin de transportar dichos residuos al centro de transferencia y retomar la ruta. Si termina la ruta sin completar la carga, el camión descarga los residuos y queda listo para la siguiente ruta.

8.1.2.- Aspectos a tener en cuenta en el nuevo diseño

Si se tienen en cuenta las características de la población (variabilidad estacional entre periodo invernal y periodo estival) y, deducido de lo anterior, que el volumen (y por tanto el peso) de RU ha de ser muy diferente en ambos periodos, se concluye la necesidad de considerar dos periodos, con dos programaciones distintas según las siguientes fechas.

PERIODO	FECHA	POBLACIÓN	CICLO
I	1/9 – 30/6	7740 Habitantes	Programación INVIERNO
II	1/7 – 31/8	24055 Habitantes	Programación VERANO

Otro aspecto, no menos importante, tiene que ver con el parámetro *días transcurridos entre dos recogidas consecutivas*. En este punto, hay que tener en cuenta criterios de salubridad, pues la influencia de la temperatura ambiental en la velocidad de degradación de los residuos es grande, dando lugar, sobre todo en el periodo estival, a la aceleración en el tiempo de aparición de olores y otras molestias, lo que incide negativamente en la calidad de vida de los ciudadanos afectados. Si a esto se le añade la presencia de roedores y otros animales, más frecuentes en este periodo, hace aconsejable establecer un periodo máximo entre dos recogidas simultáneas (max_esp_i), inferior respecto al invierno.

Por tanto, se tienen dos objetivos: coste y calidad. El coste total se obtiene como sumatorio del coste de recorrer cada una de las subrrutas. Como se explicará más adelante, el coste fijo del transporte es independiente del trayecto recorrido, dependiendo solo del día en que se efectúa la recogida, y del carácter del vehículo; el coste variable depende, mayormente, de la distancia recorrida, por lo que el coste total se puede considerar como la suma de un elemento fijo más un coste variable, dependiente linealmente de la distancia recorrida. Así pues, la solución del problema consiste, en determinar, para un horizonte temporal determinado, la programación diaria de los vehículos, es decir, número de rutas, vehículos asignados, núcleos a visitar y orden de visita, buscando minimizar la distancia total recorrida y maximizando la calidad.

Llegados a este punto, se deben hacer las siguientes consideraciones:

A cada población se le asocia una producción de residuos, dependiente de la población estimada para el mismo (según el periodo considerado), así como de la tasa de generación.

La producción de residuos en cada núcleo se considera linealmente dependiente del tiempo transcurrido (la tasa de generación se indica en *kg/hab x dia.*).

Respecto a restricciones de tipo laboral, se considera que la duración de la ruta más larga es inferior a la duración de la jornada laboral del conductor.

No se consideran ventanas de tiempo, ya que al ser depositados los residuos dentro de los contenedores, pueden ser recogidos en cualquier instante.

Existe un único depósito (centro de transferencia) del que parten los vehículos que se utilizan en la recogida y al que regresan tras terminar la ruta programada o tras completar la carga del mismo.

Las restricciones utilizadas tienen que ver, con la capacidad de los vehículos, así como las clásicas restricciones propias del CVRP (a cada nodo llega y parte un vehículo, todos vuelven al depósito, evitar sub-ciclos, etc).

Otras restricciones a tener en cuenta son causadas por la normativa legal, tanto en materia laboral como en cuanto a tiempos máximos de conducción; en cualquier caso, estas restricciones no se saturan.

8.2.- FUNCIONES OBJETIVO

8.2.1.- Función de Calidad

Considérese un periodo de análisis *p* (invierno o verano) y sea T_i la tasa de generación de residuos (medida en kg. por habitante y día) en dicho periodo. Para cada núcleo *i*, se denota por H_i su población y esp_i los días transcurridos entre dos recogidas consecutivas; a partir de la expresión $q_i = T_i \, x \, H_i$ se obtiene la generación total diaria (en kilogramos/día); con q_i y esp_i se puede calcular el total de residuos (en kilogramos) generados en el núcleo *i*, entre dos recogidas consecutivas, mediante la expresión $m_i = q_i \, x \, esp_i$. Si la producción es lineal respecto al tiempo transcurrido, el comportamiento de esta variable se puede representar gráficamente[5] (figura 27).

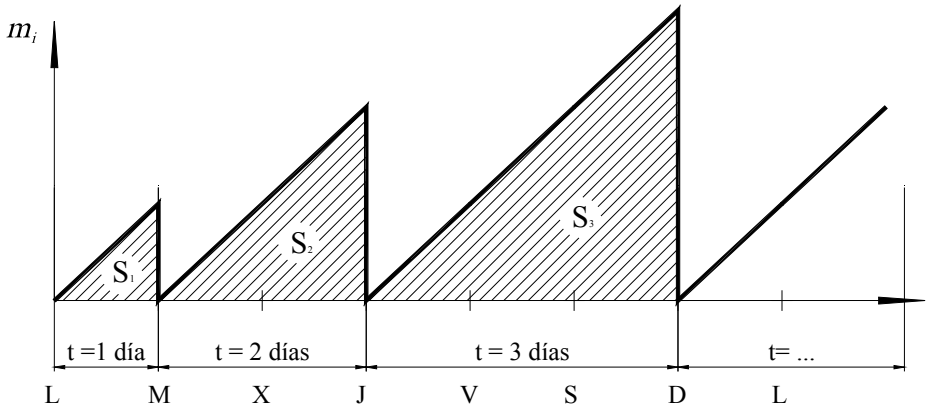


Figura 27. Representación gráfica de la variable q_i.

Como valor de la tasa de generación T_i, se adopta el valor *1.00 kg/hab*día*, el cual es coherente con los datos y estimaciones realizadas por el Ministerio de Medio Ambiente y Medio Rural y Marino (op. cit. pág. 74)

Una vez definidas las variables que intervienen en la generación de residuos, la optimización del servicio pasa por tratar de minimizar el total de la superficie comprendida entre la función y el eje de abscisas $(S_1+S_2+S_3+....S_n)$ (figura 27), entendida ésta como indicador de acumulación de residuos y, por tanto, de falta de calidad del servicio. Dicha superficie, se puede expresar como:

$$Superficie\_total = \sum_{i=1}^{nr} S_i \text{ Donde } S_i \text{ tiene por valor } S_i = \frac{q_i \times esp_i^2}{2}$$

Siendo *nr* = número de recogidas, para un horizonte temporal determinado.

Por tanto, la función de calidad es la suma de los valores de las superficies totales para el conjunto de las poblaciones y, por tanto, la función a optimizar es:

$$\min \quad \sum_{2}^{n} Superficies\_totales$$

[5] Ejemplo con fines ilustrativos.

8.2.2.- Función de Coste

El coste diario total de la recogida de RU es la suma del coste de las rutas recorridas ese día (combustible, neumáticos, etc.), que resulta ser proporcional a la distancia total recorrida, y el coste fijo asociado a la disponibilidad del vehículo (coste del conductor, seguros, etc). Éste, (cte_{vd}) depende del tipo de vehículo *v* (flota propia o alquilada) y del carácter del día *d* (laboral, sábado o festivo). En función de lo anteriormente descrito, se definen en el siguiente cuadro los siguientes costes, que quedan fijados en términos de *€/día* y que son conocidos.

cte_{vd}	Laboral	Sábado	Festivo
Flota propia	C_{pl}	C_{ps}	C_{pf}
Flota ajena	C_{al}	C_{as}	C_{af}

Con lo que el coste total se compone de dos sumandos: el primero (cte_{vd}), de carácter fijo, dependiente del tipo de vehículo *v* y del carácter del día *d*, y un segundo sumando, de carácter variable y que es proporcional a la distancia recorrida.

Según ello, el coste de cada ruta se puede expresar como:

cte_{vd} + *distancia_recorrida* * *cte_km* donde *cte_km* viene dado en €/km

Y la función a minimizar será la suma del coste de las rutas de cada día, durante el horizonte temporal considerado.

8.2.3.- Análisis de las Funciones Objetivo

Ambas funciones entran en conflicto; la minimización del coste de prestación del servicio, implica una disminución en la calidad del mismo, por lo que habrá que buscar soluciones que equilibren ambos aspectos. Aparece aquí, y como ya se ha citado anteriormente, un ejemplo claro de problema biobjetivo, en el que existen dos criterios a optimizar, requiriendo de un análisis a fin de determinar qué alternativa o alternativas son las más adecuadas. Ello pasa por obtener un conjunto eficiente de puntos que recoja aquellas soluciones no dominadas, es decir, no existen otras soluciones que sean mejores que éstas, en ninguno de los objetivos que se deben optimizar.

En la siguiente figura (28) se puede observar el resultado, tras la ejecución de una serie de búsquedas enlazadas, representados en una gráfica con el valor de la función CALIDAD en el eje de abscisas y el valor de la función COSTE en el eje de ordenadas. Cada uno de los puntos de la curva representa una solución no dominada para ambas funciones según un valor de ponderación determinado.

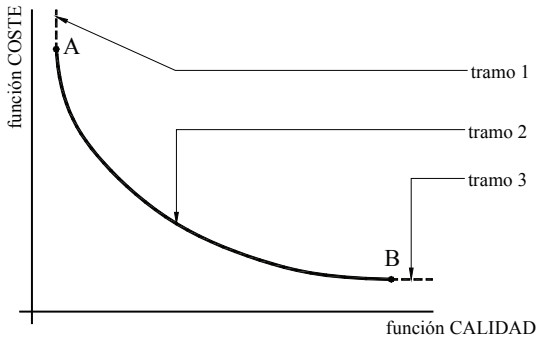


Figura 28. Curva de eficiencia para una función biobjetivo.

Habitualmente, si la aproximación a la curva de eficiencia es buena, se puede apreciar la existencia de tres tramos claramente diferenciados; en el primer tramo vertical con extremo en A, se observa que, al descender, se mejora la función COSTE, sin empeorar la función CALIDAD, hasta llegar al punto A; a partir de este punto, es imposible mejorar la función COSTE, sin empeorar la función CALIDAD. El punto A es una buena solución inicial para la función CALIDAD.

El tramo horizontal con extremo en B representa la situación opuesta al tramo vertical; aquí, según se avanza por la derecha y en horizontal hacia el punto B, se produce una mejora en la función calidad, sin mejorar la función COSTE. Esto es así hasta el punto B; a partir de aquí, no se puede mejorar más la función CALIDAD, sin empeorar la función COSTE. El punto B es una buena solución inicial para la función COSTE.

En el tramo AB, se obtienen puntos que son buenas soluciones (no dominadas) para ambos objetivos. Dichas soluciones se acercan más al punto A o al B, según se dé más peso, en cada búsqueda, a una función o a otra (factor λ). El desplazamiento a lo largo de este tramo provoca una variación tanto en la función COSTE como en la función CALIDAD, tal y como se puede apreciar en la siguiente figura (29), donde se representa, de forma gráfica, la evolución de ambas funciones (en términos de mejoría de la misma), según el diagrama de recogida en un punto determinado.

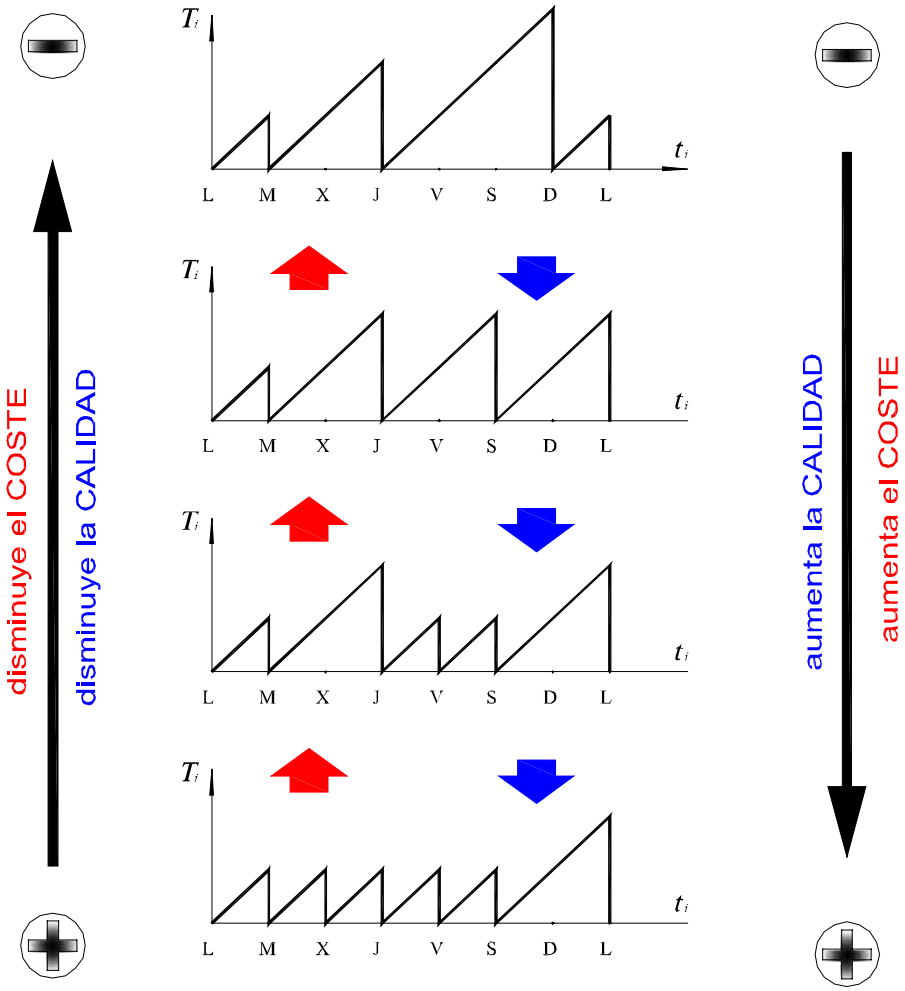


Figura 29. Evolución de las funciones objetivo.

9.- METODOLOGÍA EMPLEADA

9.- METODOLOGÍA EMPLEADA

Analizado el problema anteriormente descrito, se observa la existencia de dos niveles de decisión; un primer nivel o nivel de asignación, en el cual, se debe decidir **qué** núcleos son visitados cada día del horizonte temporal, y un segundo nivel o nivel de rutas, en el que se debe decidir **cómo** se visitan dichos núcleos.

En el primer nivel, o nivel de asignación, se debe decidir qué núcleos se visitan el día *d* del horizonte temporal, es decir, hay que definir una matriz *(i, d)* de valor:

$$matriz\ (i,\ d)\ \langle \begin{matrix} 1 & Si\ i\ es\ visitado\ el\ día\ d \\ 0 & en\ otro\ caso \end{matrix} \qquad d\ \left\{ 1,... \ HT \right\} \forall i\ \left\{ 1,..., \ n \right\}$$

Este conjunto de valores puede quedar representado mediante una matriz[6] como la que se adjunta a continuación (tabla 9), en la que se indica, para cada día del HT, qué núcleos son visitados.

		HORIZONTE TEMPORAL																						
		DIA DE RECOGIDA																						
		1	2	3	4	5	6	7	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	HT
NÚCLEO A VISITAR	núcleo 1				1	0	0																	
	núcleo 2				0	1	0																	
	núcleo 3				1	0	1																	
	.				1	0	1																	
	.				1	0	1																	
	.				0	1	0																	
	.				1	1	1																	
	.				0	0	0																	
	.				0	1	1																	
	.				1	1	1																	
	.				1	1	1																	
	.				0	0	0																	
	.				1	0	0																	
	núcleo n				0	1	1																	

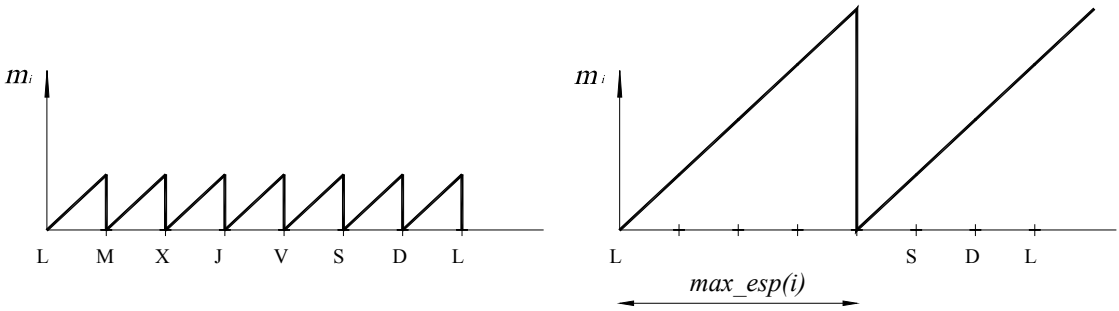
Tabla 9. Asignación de núcleos a una ruta determinada.

En el segundo nivel, o nivel de rutas, se debe decidir, una vez determinados los núcleos a visitar en un día *d*, de qué forma se visitan. Esto pasa por resolver un VRP con un único objetivo (problema mono-objetivo) que es la minimización del coste; esto para cada uno de los días del HT.

9.1.- ASPECTOS GENERALES

Al ser éste un problema biobjetivo (coste y calidad del servicio), el proceso de resolución comienza con la obtención de una buena solucion inicial para la función CALIDAD mediante un procedimiento constructivo; dicha solución, será aquella en la que se visitan todos los núcleos, todos los días del horizonte temporal. Respecto a la función COSTE, la mejor solución es aquella en la que se visita a todos los núcleos según el máximo espacio permitido entre dos recogidas consecutivas (max_esp_i); en la siguiente figura (30) se ilustran gráficamente estas consideraciones.

[6] Ejemplo a título ilustrativo.



Solución inicial para función calidad Buena solución para la función coste

Figura 30. Constructivos iniciales para las dos funciones objetivo.

El valor *max_esp,* debe ser fijado previamente por el decisor. Dicho valor no tiene por qué ser igual para todos los núcleos; por tradición, así como razones de índole técnica, se toman como buenos los valores abajo citados, y se adoptan en función de la población de los núcleos a visitar:

Tipo Municipio	max_esp_i _inv	max_esp_i _ver
G	2	1
P	5	2

9.2.- ESTRATEGIA GENERAL

Los movimientos en el nivel de asignación siguen la estrategia MOAMP[7] (*op cit. pág. 28*). Este procedimiento intenta rentabilizar el principio de proximalidad de los puntos eficientes, según el cual, puntos eficientes de un problema multiobjetivo, se encuentran 'conectados' entre sí, es decir, se encuentran dentro de una curva, conformando un conjunto eficiente de puntos. Según esto, se intenta generar un conjunto eficiente de puntos mediante una serie de búsquedas enlazadas, cada una ellas considerando diferentes funciones objetivo; inicialmente se parte de las funciones objetivos (COSTE y CALIDAD) y, a partir de ahí, se continúa con funciones mixtas (ponderan cada una de las funciones objetivo mediante un valor aleatorio), durante estas búsquedas se va actualizando el conjunto de soluciones o dominadas. Finalmente, se busca una aproximación al resto de puntos mediante un proceso de intensificación.

A continuación se describen con más detalle estos pasos.

9.2.1.- Pasos

Una solución *S* se compone de:

1. Conjunto de puntos que se deben visitar el día *d*. (nivel de asignación) *d {1, 2, … , HT}*
2. Para cada día del HT, determinar la ruta correspondiente. (nivel de rutas)

Con lo que los pasos a dar son los siguientes:

[7] Aunque en el trabajo inicial, las diferentes búsquedas se realizaban con búsqueda tabú, se pueden adoptar otras Metaheurísticas.

FASE 1:

En esta fase se parte de una buena solución inicial para la función CALIDAD; a partir de aquí, mediante una búsqueda, se intenta obtener una solución pseudo-óptima para la función COSTE y, de nuevo, desde aquí, mediante una nueva búsqueda, obtener otra solución buena para la función inicial (figura 31).

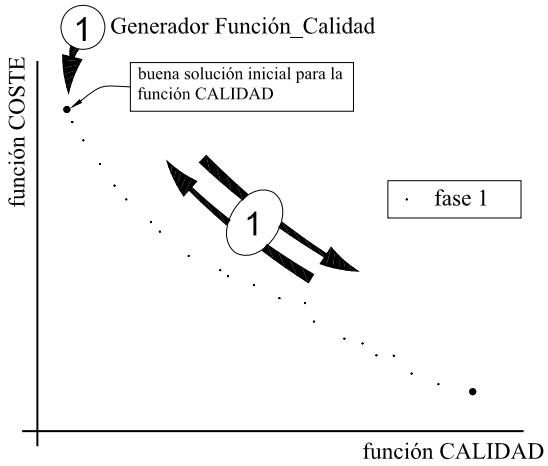


Figura 31. Fase 1.

Las búsquedas que se realizan se puede apoyar en diferentes metaheurísticas: VNS, GRASP, *Tabu Search*, recocido simulado, etc.

FASE 2:

En la fase 2 se ejecutan una serie de búsquedas hasta generar un conjunto eficiente según los siguientes pasos:

- Ejecutar una serie de búsquedas enlazadas, siendo la función a optimizar la siguiente (y que se describe más adelante):

$$F_d(S) = \max\left(\lambda \frac{f^1_{(S)} - f^1_{\min}}{f^1_{\max} - f^1_{\min}}, (1-\lambda) \frac{f^2_{(S)} - f^2_{\min}}{f^2_{\max} - f^2_{\min}} \right) \quad \forall \lambda \in \{ 0, 1 \}$$

  El resultado de estos pasos es un conjunto de puntos o conjunto compromiso (figura 32) con la característica común de representar un buen equilibrio entre los diferentes objetivos, aportando buenos valores para el conjunto, pero sin ser el mejor en alguno de ellos.

  Cada uno de los puntos corresponde a una solución para la *matriz (i, d)*; la elección de uno u otro punto dependerá del peso relativo que se quiera dar a cada una de las funciones, respecto al total.

- En cada iteración de cada una de las búsquedas se comprueba si cada punto está dominado (según el orden de Pareto) por algún punto de la lista existente hasta el momento de puntos eficientes; si no está dominado se envía a dicha lista y la actualiza. A continuación, se eliminan los puntos a los que domina. De esta forma, se obtiene una lista de los puntos visitados por la búsqueda tabú.

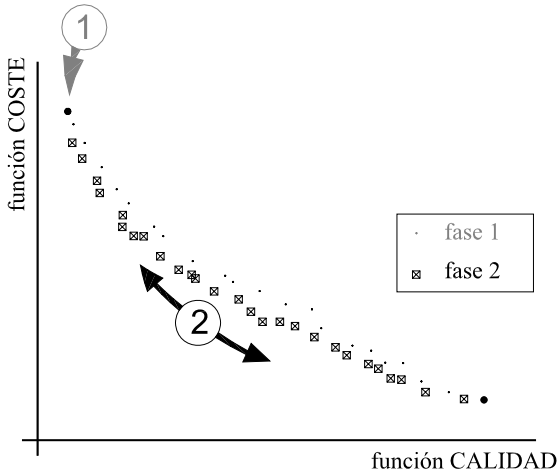


Figura 32. Lista de puntos eficientes.

FASE 3:

Por último, una vez creada una lista de puntos eficientes, se intensifica la búsqueda sobre ellos (figura 33); para ello, cada vez que se incluye un punto en la lista de puntos eficientes, se marca como 'no explorado', obteniendo al final de la búsqueda un conjunto de puntos 'no explorados' sobre los que se realizan una serie de búsquedas locales.

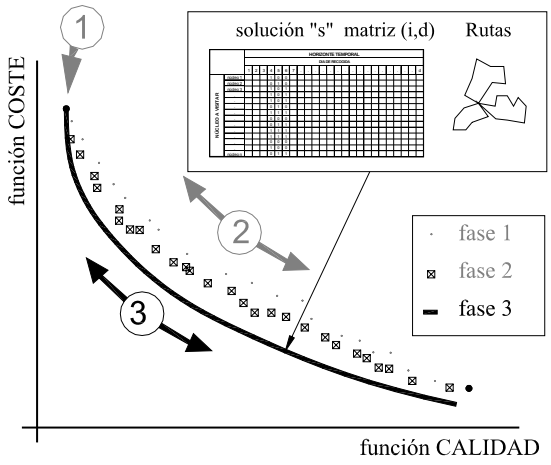


Figura 33. Proceso de Intensificación.

9.3.- DEFINICIÓN Y CUANTIFICACIÓN DE LOS MOVIMIENTOS DE ENTRADA Y SALIDA

Para la valoración de un movimiento de entrada o de salida en una ruta (inserción o eliminación de un punto), en términos de incremento o decremento de las funciones objetivo, es preciso cuantificar de alguna forma el coste[8] de la entrada y/o salida de un núcleo respecto a dicha ruta. La determinación de dichos parámetros, precisa del conocimiento previo de los factores que intervienen en dicha estimación, a saber: distancias entre núcleos, tiempos de trayecto, tasa de generación de residuos por núcleo, etc.

[8] En este caso se entiende por 'coste' la variación del valor tanto en la función CALIDAD como en la función COSTE, sin tener necesariamente relación con el concepto de coste asociado a dinero.

Se define por tanto el coste del movimiento entrada y salida, tanto en la función COSTE como en la función CALIDAD, como la variación del valor de la función, correspondiente al hecho de entrar o salir un núcleo de la ruta considerada.

VALORACIÓN DE LOS MOVIMIENTOS EN LA FUNCIÓN COSTE

El movimiento inserción (figura 34) consiste en introducir un núcleo en una ruta determinada; se observan los arcos que entran (*i-1, i*) y (*i, i+1*), y el arco que sale (*i-1, i+1*), por tanto, el coste asociado a la inserción de dicho punto será la diferencia entre la suma de los costes de los arcos que entran y el coste del arco que sale.

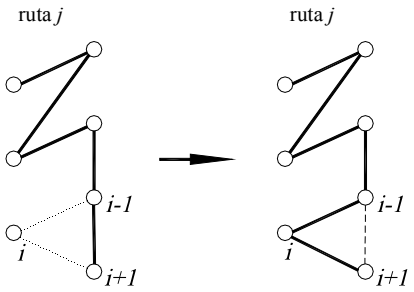


Figura 34. Inserción clásica de un punto.

Este movimiento responde a una inserción clásica, aportando un coste aproximado. Una mejor inserción es posible, recurriendo a inserciones del tipo GENI I y II. Estas inserciones (GENI, por *GENeralized Insertions*), que se decriben más adelante, surgen dentro de un método de solución del TSP y tienen como principal característica que la inserción de un nuevo núcleo en una ruta no necesariamente ocurre entre dos núcleos adyacentes.

El movimiento eliminación se considera igual y de signo opuesto al coste anterior. Respecto a la forma en que un núcleo abandona una ruta, ésta es la forma clásica, obteniendose mayor precisión con movimientos GENI I y II.

Se debe indicar que si la ruta está vacía, al incorporar el primer punto, al coste derivado de la inserción se debe añadir el coste fijo, según el tipo de vehículo que se considere. Asimismo, tras un movimiento, ya sea de entrada o de salida, se deben renombrar todos los puntos.

VALORACIÓN DE LOS MOVIMIENTOS EN LA FUNCIÓN CALIDAD

En la función CALIDAD un movimiento de inserción consiste en la inclusión de un núcleo en la ruta de recogida; esto se traduce en la mejora de la función CALIDAD, pues la diferencia entre la superficie para un núcleo determinado, en el estado final, frente al estado inicial, es negativa como se puede aprecir en la siguiente figura (35).

El movimiento contrario al anteriormente descrito, es decir, un movimiento de eliminación (figura 36), provoca un aumento en la superficie y, por tanto, un empeoramiento en la función CALIDAD. La forma de valorarlo es conceptualmente idéntica a la del movimiento anterior.

En el nivel de rutas se debe decidir **cómo** se visitan los núcleos que forman parte de la ruta para un determinado día *d* del HT. Para ello, se debe resolver un VRP. La resolución del mismo, pasa por obtener grupos de clientes denominados clusters, que estén dentro de una misma ruta, de forma que no se exceda la capacidad del vehículo y, de estas forma, resolver un TSP.

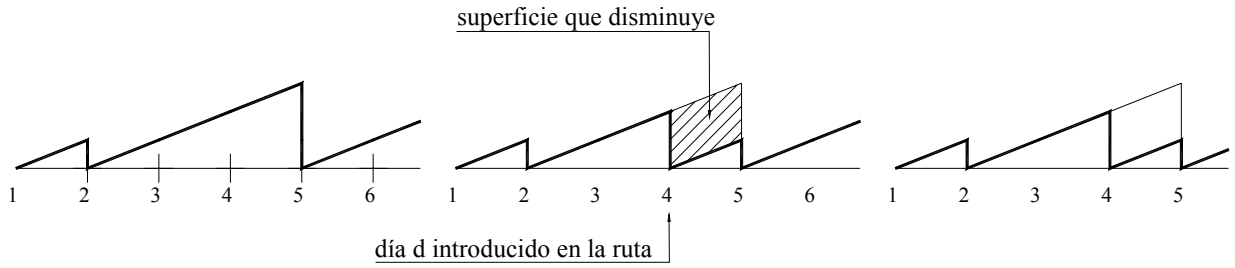


Figura 35. Inserción de un núcleo en una ruta: incremento de la Función CALIDAD.

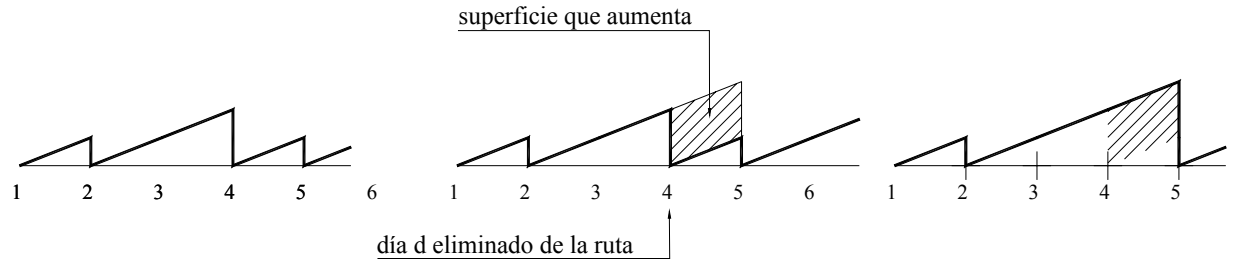


Figura 36. Eliminación de un núcleo en una ruta: decremento de la Función CALIDAD.

9.4.- OTROS TIPOS DE MOVIMIENTOS

Existen otros tipos de movimientos, intra-rutas, entre-rutas, que se describen más adelante, consistentes en el intercambio de cadenas de núcleos dentro de una misma ruta o entre dos rutas diferentes. Los intercambios posibles, derivados de estos movimientos conforman lo que se conoce como vecindario, y que constituye la base de los procedimientos de mejora como búsqueda local y búsqueda tabú.

10.- DESCRIPCIÓN DEL ALGORITMO

10.- DESCRIPCIÓN DEL ALGORITMO

10.1.- NOTACIÓN Y DEFINICIÓN DE PARÁMETROS. PLANTEAMIENTO

Se adjunta la notación que se va a utilizar en este capítulo a fin de ilustrar con mayor facilidad los métodos de solución propuestos.

HT: número de días que componen el periodo de planificación.

n: número de poblaciones o puntos de recogida que intervienen en el problema; por tanto $\{1,.. n\}$ representa el conjunto de puntos de recogida, y 0 representa el centro o depósito (origen y final de las rutas).

q_i: cantidad de RU (en kg) por día que se genera en el punto *i*. $i = 1,..., n$.

esp_i: espacio entre dos recogidas consecutivas en el núcleo *i*.

max_esp_i: máximo espacio permitido entre dos recogidas consecutivas, en el punto *i*. $i = 1,..., n$.

t_{ij}, d_{ij} : respectivamente, matrices de tiempo y distancia entre cada par de puntos *i, j,* $\forall i, j \in \{0,1,..,n\}$.

n_{td}: número de tipo de días; en este caso *ntp* = 3. Laborales (L), Sábados (S) y Festivos (F).

n_{tv}: número de clases de vehículos; en este caso *ntv* = 2. (flota propia y flota ajena).

cte_{vd} : coste fijo de cada clase de vehículo *v* según el tipo de día *d*.

m_v: número de vehículos de cada clase . $v = 1... ntv$.

m: número total de de vehículos. Obviamente, suma de los anteriores m_v.

Q_r: capacidad del vehículo *r*. $r = 1...m$.

tv_r: clase del vehículo *r*. $r = 1...m$.

cte_km: coste por kilómetro recorrido.

Cada solución *S* va a estar compuesta por dos elementos: *Asig* y *Rut*. *Asig* es una matriz de valores booleanos, que indican qué puntos se visitan cada día; más concretamente $\forall x \in \{1...n\}$ y $\forall d \in \{1.. HT\}$

$$Asig(x,d) = \begin{cases} true; & \text{si el punto x es visitado el día d} \\ false; & \text{en caso contrario} \end{cases}$$

Por otra parte *Rut* representa el conjunto de rutas de cada día, es decir:

$$Rut = \{Rutas(d) / d = 1... HT\};$$

donde *Rutas(d)* son las rutas del día *d*. Los puntos de visita y las cantidades a recoger en cada día vienen determinados por *Asig*. Como se ha comentado anteriormente cada vehículo hace a lo máximo una ruta por día, por tanto se consideran que *Rutas(d)* va a estar compuesto por un máximo de *m* rutas cada una correspondiente a un vehículo.

Sea *S* una solución; se define:

$f_1(S)$ = Función de coste total de las rutas de esa solución ('objetivo coste').

$f_2(S)$ = Función de 'no-calidad' ('objetivo calidad') según se ha definido en anteriores capítulos.

Como se ha definido antes, sean 2 soluciones S y S'; se dice que S domina a S' si:

a) $f_1(S) \leq f_1(S')$,

b) $f_2(S) \leq f_2(S')$ y

c) $f_1(S) < f_1(S')$ o bien $f_2(S) < f_2(S')$

Una solución es eficiente si no existe otra que la domina. El problema que aquí se plantea consiste en encontrar soluciones lo más cercanas posibles al conjunto de soluciones eficientes del problema que se denomina 'curva de eficiencia'.

Finalmente se definen:

f_{1min} y f_{1max}: respectivamente mínimo y máximo valores encontrados para f_1 en el conjunto de soluciones no dominadas encontradas hasta ese momento.

f_{2min} y f_{2max}: respectivamente mínimo y máximo valores encontrados para f_2 en el conjunto de soluciones no dominadas encontradas.

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

10.2.- ESTRATEGIA DE SOLUCIÓN

Como ya se ha citado anteriormente, la estrategia que se propone en este trabajo es una adaptación del procedimiento MOAMP (*MultiObjective Adaptative Memory Procedure*) para problemas multiobjetivos. Este procedimiento crea, básicamente, una aproximación a la curva de eficiencia, enlazando la ejecución de una serie de procedimientos de búsqueda tabú (es decir, que usan, como solución inicial, la solución final obtenida en la ejecución anterior) y realizando, posteriormente, una exploración en torno a las soluciones obtenidas hasta ese momento. Como se comenta en este apartado, MOAMP se basa en las siguientes ideas:

1) El principio de proximidad de puntos eficientes según el cual, en un entorno o vecindario de una solución eficiente, se puede encontrar otra solución eficiente.

2) Las solución que minimiza la distancia L_∞ normalizada al punto ideal es también eficiente.

Obsérvese que, precisamente la función F_λ representa esta distancia ponderada por λ y 1-λ. Se ha observado que las soluciones que minimizan F_λ son también eficientes.

Se va a denotar por *Set_ND* el conjunto de soluciones no dominadas encontradas durante la búsqueda. El conjunto *Set_ND* final será la aproximación a la curva de eficiencia obtenida. La estrategia de solución tiene tres fases que se describen a continuación:

FASE I

Hacer *Set_ND* = ∅.
Ejecutar **Generador_Calidad** para obtener una solución *S* a la función calidad.
Hacer *Set_ND* = {S}.
Ejecutar **Tabu_Search** *(S)* considerado f_1 como función objetivo.
Ejecutar **Tabu_Search** *(S)* considerado f_2 como función objetivo.

FASE II

Hacer niter := 0, iter_last_change := 0.
Repetir:
  niter := niter+1;
  Generar $\lambda \in (0,1)$ aleatoriamente.
  Ejecutar **Tabu_Search** *(S)* usando F_λ como función objetivo.
Si en esta ejecución *Set_ND* ha cambiado hacer iter_last_change := niter.
Hasta niter > iter_last_change + max_iter

FASE III

$\forall$ *S* $\in$ *Set_ND* marcar *S* como "*no-explorado*".
Repetir:
  $\forall$ *S* $\in$ *Set_ND* / *S* está marcado como "*no-explorado*" hacer:
    - Marcar *S* como "*explorado*".
    - Ejecutar **Exploración_vecinos** *(S)*.
    - Marcar como "*no-explorado*" a todas las soluciones que hayan entrado en *Set_ND* tras la última ejecución.
Hasta *Set_ND* se estabilice (no cambie).

La primera fase obtiene, por tanto, las mejores soluciones con respecto a cada una de las funciones objetivo. En la segunda fase, se obtiene una muestra diversa de soluciones 'de compromiso'; es decir, no las mejores soluciones con respecto a algún objetivo concreto, pero sí buenas soluciones en conjunto. Esta fase finaliza cuando trascurre un cierto número de iteraciones (max_iter) sin cambio en *Set_ND*. Hay que indicar que, dentro de cada procedimiento de búsqueda tabú, se comprueba si cada solución visitada se incorpora a *Set_ND*. En otras palabras, el conjunto de soluciones no dominadas se actualiza con todas las soluciones visitadas durante cada ejecución de la búsqueda tabú en estas dos primeras fases. Finalmente, en la tercera fase, se completa el conjunto de soluciones no dominadas explorando el vecindario de las encontradas hasta ese momento.

A continuación se describen, con más detalle, los procedimientos **Generador_Calidad**, **Tabu_Search** y **Exploración_vecinos**.

10.3.- DESCRIPCIÓN DETALLADA DE PROCEDIMIENTOS GENERALES

El procedimiento **Generador_Calidad** construye la mejor solución *S* considerando el objetivo 'calidad'. Es claro que esta solución consistirá en visitar todos los puntos, todos los días del periodo de planificación, es decir, $Asig(x,d)=true,\ \forall x \in \{1...n\}\ y\ \forall d \in \{1...\ HT\}$. No es posible encontrar una solución mejor en cuanto a calidad. Por tanto, el procedimiento debe diseñar las rutas de cada día, de forma que se minimice el coste tanto como sea posible. De esta manera, se pretende obtener una solución inicial aproximada a la curva de eficiencia. Por otra parte, como el conjunto de puntos de visita es el mismo para todos los días; las rutas solo podrían cambiar según el tipo de días (L, S o F) que podría hacer más o menos conveniente usar un vehículo más o no, según sea flota propia o ajena, etc. Por tanto, solo es necesario diseñar las rutas para cada tipo de día, considerando la visita de todos los puntos y, posteriormente, asignar a cada día las rutas de su correspondiente tipo.

El procedimiento **Generador_Calidad** por tanto puede ser descrito de la forma siguiente:

- Procedimiento **Generador_Calidad** (S)
  - Hacer $Asig(x, d) = true,\ \forall x \in \{1...\ n\}\ y\ \forall d \in \{1...\ HT\}$.
  - Para cada tipo de día (L, S, F):

    Diseñar las rutas que minimicen el coste, considerando todos los puntos y con los RU acumulados en un día. Estas rutas se obtienen inicialmente mediante la ejecución de **Constructivo_Rutas** y mejoradas con la ejecución de **Búsqueda_Tabú_Rutas.**
  - (obtención de *Rut*) $\forall$ d $\in$ {1... HT}: Asignar a cada día *d* las rutas correspondientes a su tipo.

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

El procedimiento **Tabu_Search** de forma general puede ser descrito, genéricamente, de la siguiente forma:

- Procedimiento **Tabu_Search**(S)
  - Hacer *S_best* = *S*, niter = 0 y iter_best = 0;
  - Iniciar variables auxiliares.
  - Repetir:
    1. niter = niter + 1.
    2. determinar S_1 = argmin { $F_\lambda(S') / S' \in N(S)$, *S'* no es tabú o $F_\lambda(S') < F_\lambda(S\_best)$}.
    3. Hacer $S = S_1$, mejorar las rutas de los días afectados en S con **Búsqueda_Local_Rutas** y actualizar *Set_ND* con *S.*
    4. Actualizar *Set_ND* con *S* y actualizar variables auxiliares.
    5. Si $F_\lambda(S) < F_\lambda(S\_best)$ entonces hacer *S_best* = *S* y iter_best = niter hasta niter > iter_best + max_iter_tabu.
  - Hacer *S* = *S_best.*

A continuación, se describen y definen, con más detalle, algunos de los puntos del procedimiento anterior. *N(S)* o conjunto de 'soluciones vecinas' de *S* es el conjunto de soluciones factibles a las que se llega desde *S* o bien eliminando la recogida de RU de un punto *x* en un día *d*, si *Asig(x,d)=true*, o bien insertándola, si *Asig(x,d)=false*. En el caso de las eliminaciones de un punto de una ruta, se consideran tres tipos que se denominan: Normal, GENI-I y GENI-2. De igual forma, se consideran tres tipos de inserciones análogas en cada una de las rutas de ese día: Normal, GENI-I y GENI-2. Tanto los tipos de inserciones como de eliminaciones se describen con más detalle en la sección 10.5. Estos movimientos (eliminaciones e inserciones) producen modificaciones en las funciones objetivos que pueden ser calculadas rápidamente. En general, se pueden identificar movimientos con las soluciones vecinas a las que dan lugar.

En cuanto a la factibilidad de las eliminaciones, hay que comprobar que se respete la máxima separación entre dos recogidas consecutivas en un punto, y que viene determinado por el valor de max_esp_x en cada punto *x*. Por ejemplo, si en un punto *x* se visita los días 2, 4 y 6, la eliminación de la visita de *x* el día 4 será infactible si max_esp_x es menor que 4. Otro punto que hay que tener en cuenta, en la factibilidad de una eliminación, es el incremento de RU a recoger en este punto el siguiente día de recogida. Para ilustrar esta situación considérese el ejemplo anterior con recogidas los días 2, 4 y 6: si la producción diaria en el punto *x* es de 10 unidades, antes de la eliminación en ese punto se han de recoger 20 unidades los días 4 y 6 ($20 = 10\cdot(6-4) = 10\cdot(4-2)$); ahora bien, la eliminación de la recogida en el día 4 aumenta la carga a recoger el día 6 de 20 a 40 ($40 = 10\cdot(6-2)$). Por tanto, habría que comprobar si esa eliminación respeta la factibilidad de capacidad del vehículo que hace dicha ruta.

Respecto a las inserciones, solo es necesario comprobar la factibilidad de la capacidad del vehículo correspondiente en el día de la inserción. Hay que señalar que una inserción produce una disminución en la cantidad a recoger el día de la siguiente recogida, es decir, justo el efecto contrario de una eliminación. Por ejemplo, si en un punto *x* se recogen los días 2 y 6, la inserción de la recogida del punto *x* en el día 4 hace que disminuya la cantidad a recoger el día 6, pero obviamente, esto no produce infactibilidad, antes bien, da más 'holgura' de capacidad a los vehículos.

Para evitar que el procedimiento cicle y se vuelvan a soluciones ya visitadas, algunos movimientos se declaran tabú y, en principio, no son admitidos. Por ejemplo, si el último movimiento ha supuesto la eliminación de la recogida en el punto *x* el día *d* (es decir, pasar de *Asig(x,d)=true* a *Asig(x,d)=false*), el procedimiento declarará tabú la reinserción de *x* en el día *d* durante cierto número de iteraciones siguientes, es decir, no se podrá realizar. De la misma forma, si ha sido insertado un punto x en la recogida de un día *d*, (es decir se ha pasdo de *Asig(x,d)=false a Asig(x,d)=true*) en las siguientes iteraciones se declara tabú eliminar la recogida de *x* en el día *d*. Se definen *Tabu_tenure_ins* y *Tabu_tenure_eli* respectivamente como el número de iteraciones en las que una inserción y una eliminación permanecen tabú.

El status tabú de un movimiento se puede comprobar con el uso de las siguientes *variables auxiliares*:

Matriz_eli (x, d)= última iteración en la que la recogida en el punto *x* ha sido eliminada del día *d*;

Matriz_ins (x, d)= última iteración en la que la recogida en el punto *x* ha sido insertada en el día *d*.

de forma que la inserción del punto *x* el día *d* es tabú si:

niter <= *Matriz_eli(x, d)* + *Tabu_tenure_ins* (es decir, ha sido eliminado recientemente).

de la misma manera la eliminación del punto *x* el día *d* es tabú si:

niter <= *Matriz_ins(x, d)* + *Tabu_tenure_eli.*

Estas variables auxiliares se actualizan tras cada iteración (paso 4 del bucle principal) y se inicializan con los valores *Matriz_ins(x, d) = -Tabu_tenure_eli y Matriz_eli(x, d) = -Tabu_tenure_ins*. Como se indica en el paso 2 del bucle principal, el 'status tabú' de un movimiento se puede ignorar si la solución a la que da lugar mejora la mejor encontrada hasta ese momento (respecto a la función objetivo F_λ).

Cuando se ejecuta un movimiento, las rutas de los días afectados (por inserción o eliminación) son mejoradas por un procedimiento de búsqueda local (paso 3 del bucle principal) denominado **Búsqueda_Local_Rutas**, que se explicará con más detalle en la sección posterior, dedicada a los procedimientos de rutas. Como se ha explicado anteriormente, una eliminación (o una inserción) no solo afecta al día en que ésta se produce, sino también al día de la siguiente recogida en ese punto. Con la ejecución de este procedimiento de búsqueda local, se pretende que las rutas diarias de las soluciones visitadas sean óptimos locales.

La actualización de *Set_ND* con la nueva solución *S* consiste en lo siguiente: comprobar si *S* no está dominada por ninguna solución de *Set_ND*; en caso de que no estuviera dominada se introduce *S* en *Set_ND* y se eliminan de este conjunto aquellas soluciones dominadas por *S*.

Finalmente, la función F_λ, tal como se define al principio del capítulo, generaliza las dos funciones originales f_1 y f_2 considerando $\lambda = 1$ y $\lambda = 0$ respectivamente. Sin embargo, hay que advertir que antes de la primera ejecución de **Tabu_Search** (en la Fase I, tras la obtención de la primera solución de calidad) *Set_ND* solo está compuesto por una solución, por tanto $f_{1min} = f_{1max}$ *y* $f_{2min} = f_{2max}$. En este caso, ficticiamente se toma $f_{1min} = f_{2min} = 0$, *y* $f_{1max} = f_{2max} = 1$. Posteriormente los valores de f_{1min}, f_{2min}, f_{1max} y f_{2max} se actualizan tras cada ejecución de *Tabu_Search* según la definición dada en la sección 1.

El procedimiento **Exploración_vecinos** que se usa en la Fase III puede ser descrito de la forma siguiente:

- Procedimiento **Exploración_vecinos(*S*)**

  $\forall$ *S'* $\in$ N(*S*):

  - Si *S'* no está dominada por ninguna solución en *Set_ND* entonces:
    1) Mejorar los días afectados por el cambio con **Búsqueda_Local_Rutas.**
    2) Introducir *S*' en *Set_ND.*
    3) Eliminar las soluciones en *Set_ND* dominadas por *S*'.

10-4.- PROCEDIMIENTOS DE RUTAS

En esta sección se describen los procedimientos de construcción y mejora de las rutas diarias usadas en los procedimientos generales explicados en la sección anterior. El objetivo en este caso siempre es minimizar el coste (por vehículos y por distancia recorrida), ya estos procedimientos no alteran el conjunto de puntos visitados. Concretamente se explican **Constructivo_Rutas**, **Búsqueda_Local_Rutas** y **Búsqueda_Tabu_Rutas**.

El procedimiento **Constructivo_Rutas** considera los siguientes parámetros de entrada (*inputs*):

- *tipo_dia*: tipo de día en el que se hace la recogida (L, S, F).
- *Set_ptos*: subconjunto de puntos a visitar,
- *q_día*: cantidad de RU a recoger en los puntos anteriores;

Como salida (*output*), se define *Rutas* como el conjunto de rutas obtenidas (de tamaño menor o igual que *m* según se explica en 10.1). Se denota por *Ruta_d (j), j=1...m*, a cada una de las rutas que componen *Rutas*. Este procedimiento puede ser descrito de la forma siguiente:

- Procedimiento **Constructivo_Rutas** (*tipo_dia, Set_ptos, q_dia, Rutas*)
  - Iniciar solución: hacer *Ruta_d(j) = 0 – 0, j = 1..m*; (rutas solo compuestas por el origen)
  - Ordenar los puntos de *Set_ptos* según el ángulo que forma la horizontal con la bisectriz que une el origen con cada punto
  - Desde *i* = 1 hasta |*Set_ptos*| hacer:
    1. Tomar *x* el i-ésimo punto de *Set_ptos* según el orden anterior.
    2. Calcular la mejor (menos costosa) inserción factible de *x* en alguna ruta de *Rutas.*
    3. Realizar dicha inserción.

Se consideran los tres tipos de inserciones usadas anteriormente (en la definición de *N(S)*) Normal, GENI-I y GENI-2 y que se explicarán más adelante. Obviamente, como se ha comentado en la sección 10.3, cuando **Constructivo_Rutas** se ejecuta dentro de **Generador_Calidad**, *Set_ptos={1...n}* y *q_dia(x)* es la cantidad de R.U. generados diariamente en cada punto.

El procedimiento **Búsqueda_Local_Rutas** intenta mejorar el conjunto de rutas, *Rutas*, de un determinado día. Usa los mismos datos de entrada del constructivo anterior, *tipo_dia* y *q_dia*, ademas de la variable *Rutas* que es también salida. (El conjunto de puntos *Set_ptos* viene definido por los puntos que se visitan en *Rutas*). Puede ser descrito en pseudocódigo de la forma siguiente.

- Procedimiento **Búsqueda_Local_Rutas** (*tipo_día, Set_ptos, q_día, Rutas*)

  Repetir:

  Buscar *Rutas'* el mejor conjunto de rutas en *NR(Rutas*).

  Si *Rutas*' mejora *Rutas* entonces hacer *Rutas* := *Rutas*'.

  Hasta *Rutas*' no mejora *Rutas*.

Se define *NR(Rutas)* como 'vecindario' de Rutas, es decir, todos los conjuntos de rutas que se pueden obtener realizando en las rutas de *Rutas* los cambios o 'movimientos' que se describen a continuación.

Básicamente, estos cambios consisten en intercambios de cadenas. Concretamente se consideran dos tipos: intercambios intra-rutas (i.e. dentro de una ruta) y cambios entre-rutas. Dentro de los intercambios intra-rutas se consideran los intercambios tipo Or y los intercambios Or generalizados. Los primeros

consisten en intercambiar cadenas consecutivas; los segundos intercambian cadenas no consecutivas (ambos dentro de una ruta). Las figuras 37 y 38 ilustran ambos tipos de intercambios intra-rutas.

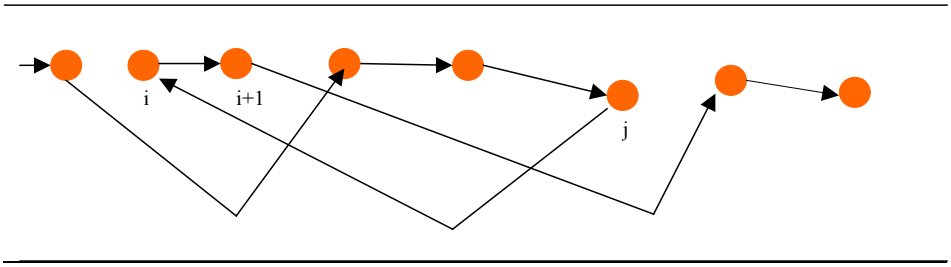


Figura 37. Intercambio de dos cadenas consecutivas en una ruta (Or).

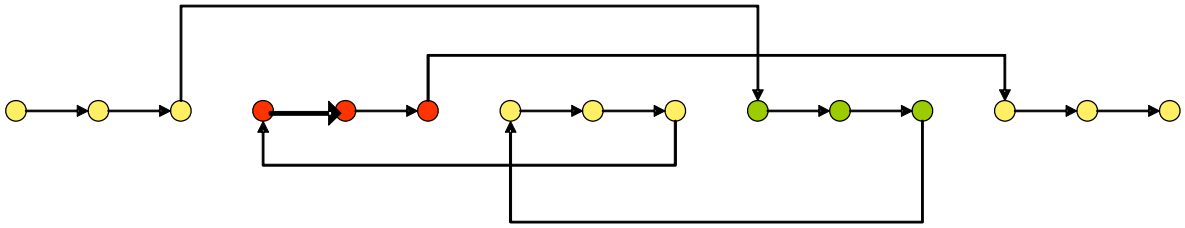


Figura 38. Intercambios de dos cadenas no consecutivas (Or generalizados).

Los cambios entre-rutas que se usan en este trabajo son también de dos tipos: el primer tipo son intercambios de 2 cadenas de rutas diferentes, también denominados CROSS intercambios, y el segundo tipo es el cambio de una cadena de una ruta a otra diferente. Las figuras 39 y 40 ilustran ambos tipos de cambios entre-rutas.

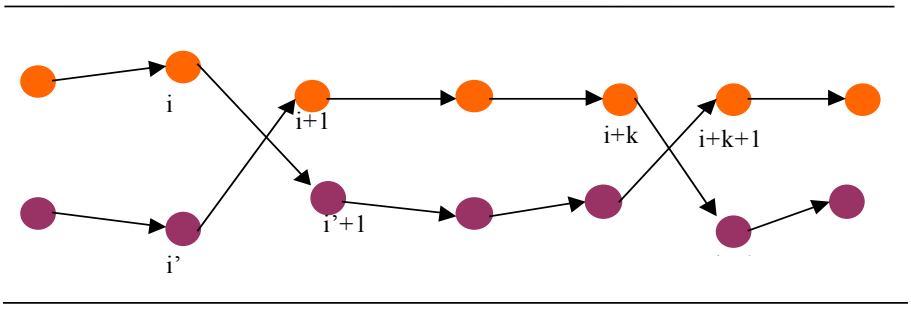


Figura 39. Ejemplo CROSS intercambio.

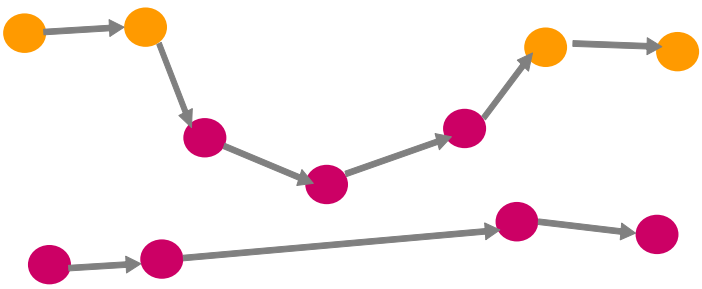


Figura 40. Cambio de una cadena de una ruta a otra.

El procedimiento **Búsqueda_Local_Rutas** mejora cada conjunto de rutas inicial hasta que no haya mejoras posibles con los movimientos así definidos, (es decir, hasta llegar a un óptimo local). El procedimiento **Búsqueda_Tabu_Rutas** permite mejorar los resultados obtenidos por **Búsqueda_Local_Rutas** ya que permite exploraciones mas allá del primer óptimo local encontrado. Para ello se permiten movimientos a conjuntos de rutas peores y, para evitar que el algoritmo cicle (visitar soluciones anteriores), algunos movimientos se declaran tabú.

El procedimiento **Búsqueda_Tabu_Rutas** tiene las mismas entradas y salidas que **Búsqueda_Local_Rutas;** además usa dos parámetros importantes *tabu_tenure* y *max_iter_rutas*. Puede ser descrito de la forma siguiente.

- Procedimiento **Búsqueda_Tabu_Rutas** (*tipo_día, Set_ptos, q_dia, Rutas*)

  Hacer *Rutas_best = Rutas, niter:=0, iter_best:=0,*

  Iniciar variables auxiliares: *matriz_tabu_ruta(i, j) := - tabu_tenure;* $\forall i, j \in \{0,1,..,n\}$

  Repetir:

  - niter := niter+1;
  - Buscar *Rutas*' el mejor conjunto de rutas en *NR(Rutas) / Rutas'* no es tabú o mejora *Rutas_best*.
  - Hacer *Rutas:= Rutas'*.
  - Actualizar variables auxiliares (*matriz_tabu_ruta*).
  - Si *Rutas* es mejor *Rutas_best* hacer *Rutas_best:= Rutas y iter_best := niter* hasta n*iter > iter_best + max_iter_rutas*.

Más concretamente: obsérvese que los movimientos o intercambios anteriores consisten en la eliminación de una serie de arcos y la incorporación de otros. Para que el algoritmo no cicle, se impide que los arcos que han sido eliminados, vuelvan a ser incorporados (al menos en una serie de iteraciones).

Para ello se hace uso de *matriz_tabu_ruta* que se define como:

matriz_tabu_ruta(i, j) = Iteración en la que el arco (*i, j*) ha sido eliminado

$i, j = 0, 1,.., n; i \neq j;$

un arco (i, j) es tabú si

$$niter \leq matriz\_tabu\_ruta(i, j) + tabu\_tenure\_rutas;$$

donde *tabu_tenure* indica durante cuántas iteraciones un arco es tabú. Un movimiento, y por tanto el correspondiente elemento en *NR(Rutas),* es tabú si supone incorporar un arco tabú. Un movimiento tabú puede ser admitido si da lugar a un coniunto de Rutas mejor que Rutas (esto es conocido en la literatura como criterio de aspiración). El procedimiento finaliza cuando trascurren max_iter_rutas iteraciones sin mejora.

10.5.- ELIMINACIONES E INSERCIONES

Finalmente, para terminar la descripción de todos los procedimientos y métodos que componen la estrategia de solución, en esta sección se describen los tipos de eliminaciones e inserciones usadas por los procedimientos anteriores. Como se explicó anteriormente, las eliminaciones son usadas por el

procedimiento **Tabu_Search**, y las inserciones por **Tabu_Search** y por el procedimiento **Constructivo_Rutas**.

Concretamente, se contemplan tres tipos de eliminaciones:

i) Eliminación Normal o clásica

En las tres siguientes figuras, sin perdida de generalidad y para simplificar la notación, se identifican los puntos de visita con el orden que ocupan en la ruta. Los arcos que se eliminan se muestran con líneas de puntos, los que se añaden con guiones, y en los tramos que cambian de sentido se añade la nueva orientación debajo de la original. La figura 41 ilustra la eliminación normal: La eliminación de la ruta del punto i supone la eliminación de los arcos *(i-1, i)* y *(i, i+1)*, y la incorporación del arco *(i-1, i+1)*.

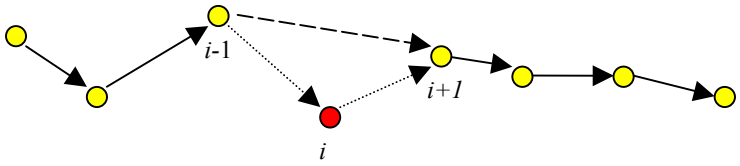


Figura 41. Eliminación normal.

ii) Eliminaciones tipo **GENI-I**

La figura 42 ilustra dicha eliminación. La eliminación de la ruta del punto *i* supone la eliminación de los arcos *(i-1, i)*, *(i, i+1)*, *(k, k+1)* y *(j, j+1)*; la incorporación de los arcos *(i-1, k)*, *(i+1, j)*, *(k+1, j+1)*; y el cambio de sentido de los tramos de *i+1* a *k*, y de *k+1* a *j*. Por tanto, además del punto i a eliminar, este tipo de eliminaciones dependen de otros dos parámetros *k* y *j* *(k ≥ i+1, j ≥ k+1)*.

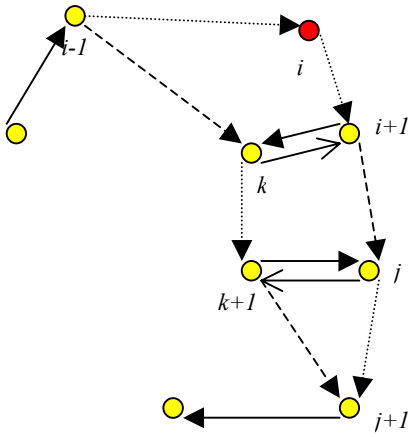


Figura 42. Eliminación tipo GENI_I.

iii) Eliminaciones tipo **GENI-II**

La figura 43 ilustra dicha eliminación. La eliminación de la ruta del punto *i* supone la eliminación de los arcos *(i-1, i)*, *(i, i+1)*, *(j-1, j)*, *(l, l+1)* y *(k, k+1)*; la incorporación de los arcos *(i-1, k)*, *(l+1, j-1)*, *(i+1, j)*, *(l, k+1)*; y el cambio de sentido de los tramos de *l+1* a *k*, y de *i+1* a *j-1*. Por tanto, además del punto i a eliminar, este tipo de eliminaciones depende de tres parámetros *j, l* y *k (j* ≥ *i+2, l* ≥ *j, k* ≥ *l+1)*.

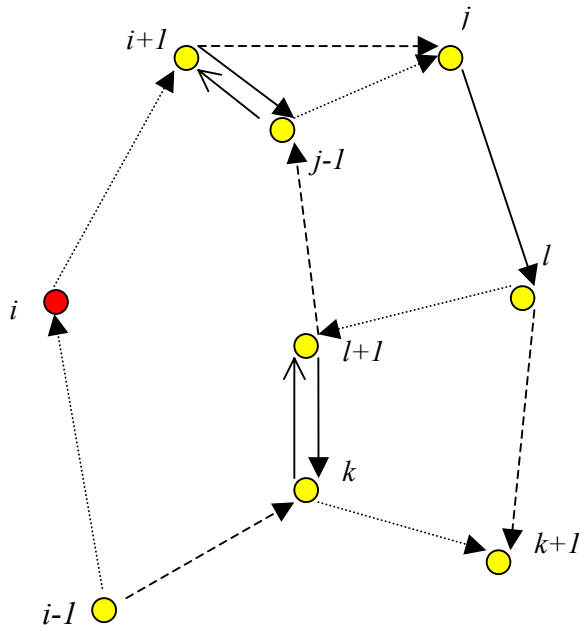


Figura 43. Eliminación tipo GENI_II.

Análogamente, se contemplan tres tipos de inserciones:

i) Inserción Normal o clásica

En las tres siguientes figuras, igual que antes, se identifica los puntos de la ruta con el orden que ocupan. El punto a insertar se denota por *x*. Los arcos que se eliminan se muestran con líneas de puntos, los que se añaden con guiones, y en los tramos que cambian de sentido se añade la nueva orientación debajo de la original. La figura 44 ilustra la inserción normal: la inserción entre los puntos *i* e *i+1 (i* ≥ *1)* de la cadena supone la eliminación del arco *(i, i+1)*, y la incorporación de los arcos *(i, x)* y *(x, i+1)*.

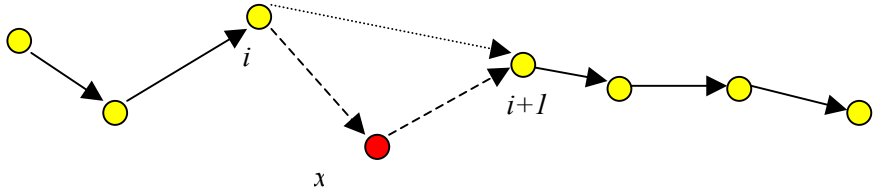


Figura 44. Inserción normal.

ii) Inserciones tipo **GENI-I**

La figura 45 ilustra dicha inserción. La inserción en la ruta del punto *x* supone la eliminación de los arcos *(i, i+1)*, *(j, j+1)* y *(k, k+1)*; la incorporación de los arcos *(i, x)*, *(x, j)*, *(i+1, k)* y *(j+1, k+1)*; y el cambio de sentido de los tramos de *i+1* a *j*, y de *j+1* a *k*. Por tanto, (además del punto a insertar) este tipo de inserciones dependen de tres parámetros *i, j y k (i ≥ 1, j ≥ i+1, k ≥ j)*.

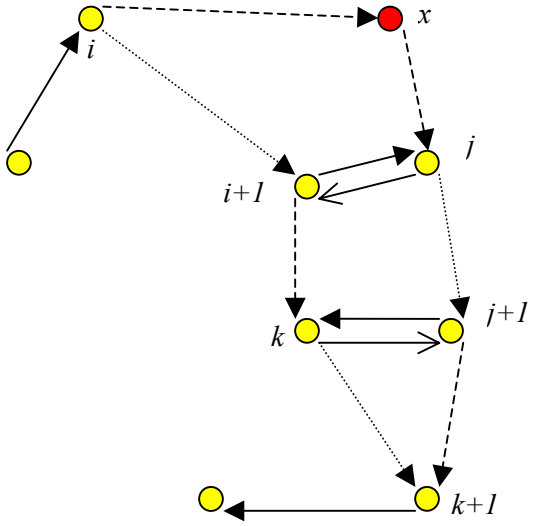


Figura 45. Inserción tipo GENI_I.

iii) Inserciones tipo **GENI-II**

La figura 46 ilustra dicha inserción que supone la eliminación de los arcos *(i, i+1)*, *(l-1, l)*, *(j, j+1)* y *(k-1, k)*; la incorporación de los arcos *(i, x)*, *(y, j)*, *(l, j+1)*, *(k-1,l-1)* y *(i+1, k)*; y el cambio de sentido de los tramos de *l* a *j*, y de *i+1* a *l-1*. Por tanto, (además de la cadena a insertar) este tipo de inserciones dependen de cuatro parámetros *i, l, j* y *k (i ≥ 1, l ≥ i+2, j ≥ l, k ≥ j+2)*.

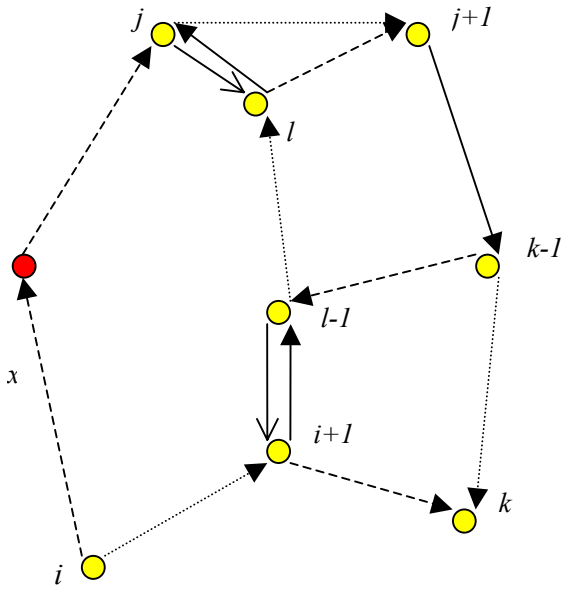


Figura 46. Inserción tipo GENI_II.

Hay que indicar que, tanto para las eliminaciones como para las inserciones, se contemplan ambas orientaciones de las rutas implicadas.

10.6.- EJEMPLO ILUSTRATIVO

Se detalla, a continuación, un ejemplo que ilustra el funcionamiento del algoritmo propuesto. Concretamente, los valores de los parámetros son los siguientes: *HT= 20*, *n=40*; los valores de q_i se generan aleatoriamente entre *1* y *50*, para *i =1…20*, y entre *51* y *100* para *i =21… 40*; max_esp_i*= 2*, *i=1..20*; max_esp_i*= 3*, *i =21..41*; a cada punto de visita se le asigna un par de coordenadas *(x, y)* generadas en el rango [-100, +100]; las matrices de distancia y tiempo se obtienen redondeando las distancias euclídeas. Los 20 días son: Laborales del 1 al 5, del 8 al 12 y del 15 al 19; sábado el 6, 13 y 20; domingos el 7 y 14. La matriz de costes fijo (según tipo de vehículo y día) es *cteFL=0*, *cteFS=100*, *cteFD=200*, *cteAL=100*, *cteAS=225*, *cteAD=375*, Se consideran cuatro vehículos, dos de flota propia y dos de ajena; finalmente *cte_km = 1*.

A continuación se muestra un gráfico (figura 47) con las diferentes aproximaciones a la curva de eficiencia obtenidas en cada una de las 3 fases del algoritmo: en negro se represente *Set_ND* tras la primera fase, en rosa tras la segunda, y en azul tras la tercera.

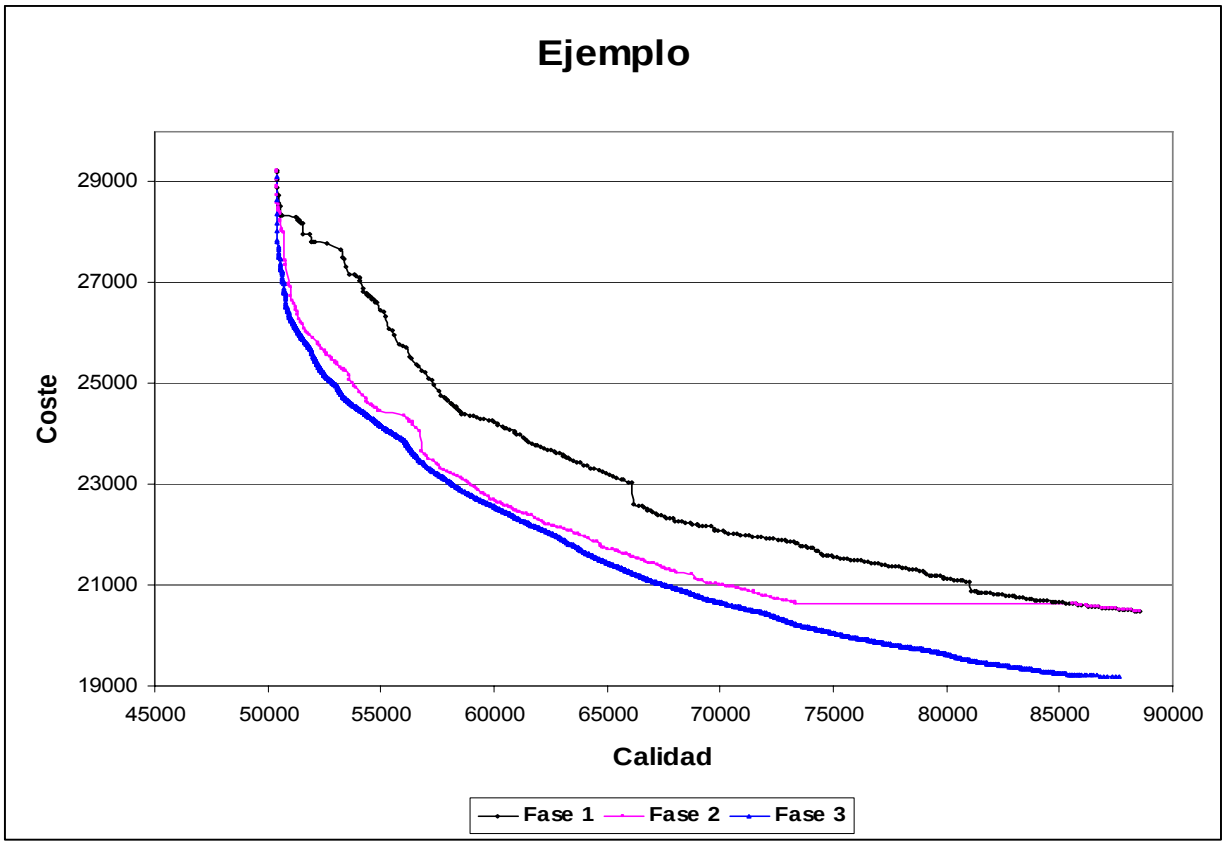


Figura 47. Evolución del conjunto de soluciones no dominadas.

El número de puntos que componen *Set_ND* es de 313 en la Fase I, 243 en Fase II y 2538 en la Fase III. Es claro que esta fase ha resultado ser más efectiva que las otras dos. Por otra parte, se entiende que el resultado final (2538 soluciones no dominadas) es interesante para el tamaño del problema considerado.

11.- MÉTODOS DE ACELERACIÓN

11.- MÉTODOS DE ACELERACIÓN

Los procedimientos **Búsqueda_Local_Rutas** y **Tabu_Search** pueden ser acelerados de manera significativa, respectivamente con tres métodos; el primero se basa en el la aplicación de la estrategia denominada búsqueda local rápida, el segundo, que consiste en el uso de árboles binarios y el tercero, que consiste en la modificación del conjunto de soluciones no dominadas mediante movimiento por índices; a continuación se explican dichos métodos.

11.1.- BÚSQUEDA LOCAL RÁPIDA

Se ha acelerado la ejecución del procedimiento **Búsqueda_Local_Rutas** con un mecanismo basado en la versión geométrica del heurístico 2-óptimo sugerido por Bentley [294] en el contexto del Problema del Viajante (*Traveling Salesman Problem*, TSP). Este mecanismo consiste en dividir el vecindario en subconjunto de movimientos. Se asigna una variable binaria a cada subvecindario para indicar si esta activo o no; en cada iteración solo se exploran los vecindarios activos. Inicialmente todos los vecindarios están activos. Si tras explorar un subvecindario no se ha encontrado en éste ningún movimiento que mejore la solución actual, pasa a ser inactivo. Cuando un movimiento es ejecutado, todos los sub-vecindarios afectados pasan a ser activos.

Concretamente, se definen los siguientes sub-vecindarios:

A (r) = Intercambios de 2 cadenas consecutivas en la ruta *r, r = 1...m*;
B (r) = Intercambios de 2 cadenas no consecutivas en la ruta *r, r = 1...m*;
C (r, r') = Intercambios de 2 cadenas pertenecientes respectivamente a las rutas *r* y *r'*; *r = 1... m-1* y *r' = r+1...m;*
D (r, r') = Cambios de una cadena de la ruta *r* a la ruta *r'*; *r, r' = 1...m y r' ≠ r*;

Es decir, se han definido cuatro tipos de sub-vecindarios correspondientes a los cuatro tipos de movimientos usados en **Búsqueda_Local_Rutas**.

11.2.- ÁRBOLES BINARIOS EN TABU_SEARCH

Para explicar el uso de árboles binarios en el procedimiento **Tabu_Search** se han de hacer las siguientes consideraciones:

- Inicialmente un movimiento, de eliminación o inserción, viene definido por el punto *i* a eliminar (o insertar), el día *d* y la forma de eliminación (o inserción), es decir: tipo y parámetros. Sin embargo, como la forma de eliminación (o inserción) no afecta a la calidad de la solución resultante, parece claro que siempre se debe elegir a la mejor solución, según el coste. Además, por la definición del procedimiento, el carácter tabú tampoco depende de la forma de eliminación, solo del par (*i, d*). Por tanto, un movimiento está definido por el par (*i, d*) (de eliminación si *Asig(i, d) = true* y de inserción si *Asig(i,d) = false*), ya que la forma de eliminación (o inserción) es la mejor según la función coste.
- Cada movimiento (*i, d*) lleva, además, asociado el carácter factible o no factible. También lleva asociadas dos variables: *Inc_Coste* e *Inc_no_cal* que son respectivamente los incrementos (o decrementos si son negativos) de la función objetivos f_1 y f_2 en caso de ejecutarse dicho movimiento. Con estos valores, se calcula fácilmente $F_\lambda(S')$, donde *S'* es la solución a la que se llega si se ejecuta el cambio definido por (*i, d*), (paso 2 del bucle principal).

- Cuanto se ejecuta un movimiento, para cada par (*i, d*) hay que actualizar el tipo y los parámetros que definen la eliminación (o inserción) correspondiente, su carácter factible, y los valores *Inc_Coste* e *Inc_no_cal*. Esto forma parte del paso 4 del bucle principal, junto a la actualización de las matrices que definen las listas tabú.
- En realidad pocos movimientos se ven afectados tras cada ejecución (paso 3 del bucle principal), es decir, en pocos movimientos cambian estos valores.
- Sin embargo, aunque un movimiento no se vea afectado (no cambien los valores de *Inc_Coste* e *Inc_no_cal* o el resto de parámetros), los valores de los correspondientes $F_\lambda(S')$ sí deben ser calculados en cada iteración. Por tanto, el número de operaciones en cada iteración es de O(n·HT).

Para mejorar el tiempo de cálculo se proponen las siguientes modificaciones:

- A cada movimiento (*i, d*) se le asocia un valor V_{id} que se define de la siguiente manera:

  $v_{id} = \lambda \cdot Inc\_Coste + (1-\lambda) \cdot Inc\_no\_cal$ si (*i, d*) es factible o es tabú; ∞ en caso contrario.

  Obsérvese que este valor solo cambia si el movimiento está afectado; este nuevo valor será usado para determinar el mejor movimiento en cada iteración.

- Para ordenar los movimientos y elegir el mejor, según este nuevo criterio, se van a usar árboles binarios ([295]), como se describe en el siguiente párrafo.

En cada nodo *k* del árbol binario está localizado un movimiento (*i, d*) de forma que en los nodos descendientes *2k* y *2k+1*, se localizan 2 movimientos (*i', d'*) y (*i'', d''*) iguales o peores que (*i, d*) (es decir $v_{id} \leq v_{i'd'}$ y $v_{id} \leq v_{i''d''}$). Lógicamente, en cada iteración, el mejor movimiento está situado en el nodo raíz. Un árbol binario puede ser programado usando un vector HP y una matriz QP:

- HP (*k*): indica que movimiento (*i, d*) está localizado en el nodo k del árbol;
- Q (*i, d*): indica el número del nodo donde el movimiento (*i, d*) está localizado.

La figura 48 ilustra un árbol binario. En este caso HP es igual a *((1,2), (2,4), (3,5), (2,1), (4,6))* y *QP(1,2) = 1, QP(2,4) = 2, QP(3,5) = 3, QP(2,1) = 4, y QP(4,6) = 5.*

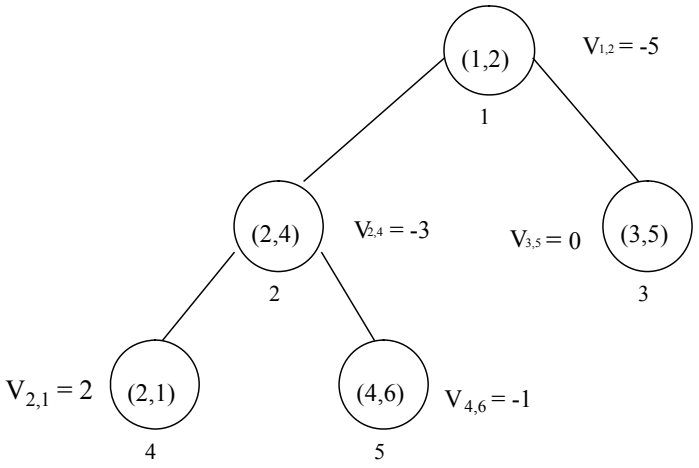


Figura 48. Ejemplo de árbol binario para ordenar movimientos.

Entonces, en cada iteración, para determinar el movimiento mejor se compara el situado en el nodo raíz del árbol binario y los movimientos tabú que cumplen el criterio de aspiración, (a lo sumo *Tabu_tenure_ins+Tabu_tenure_eli*). Cuando un movimiento (*i*, d**) es ejecutado, se deben actualizar las localizaciones de los movimientos cuyos valores V_{id} cambian. Como el número de nodos en el árbol binario es n·HT, cada una de estas actualizaciones emplea O ($\log_2$(n·HT)) operaciones. Se puede probar fácilmente que el número de movimientos afectados es del orden de 3·n. Por tanto, el número de operaciones para actualizar el árbol binario es O (n·$\log_2$ (n·HT)).

11.3.- MOVIMIENTOS POR ÍNDICES

Hasta el momento, la inclusión de una solución determinada, en el conjunto de soluciones no dominadas, obliga a 'mover' una posición de memoria todo el resto de soluciones que la siguen; esta situación también se da en el caso de que, por incluir una solución, aparezcan soluciones que antes eran no dominadas, pero que ahora sí lo son y que, por tanto, deben desaparecer de dicho conjunto. Sabiendo que cada solución se compone de la matriz de asignación (*Asig*) y del conjunto de rutas que le acompañan (*Rut*), se intuye que este proceso presenta un tiempo de cómputo muy alto (figura 49).

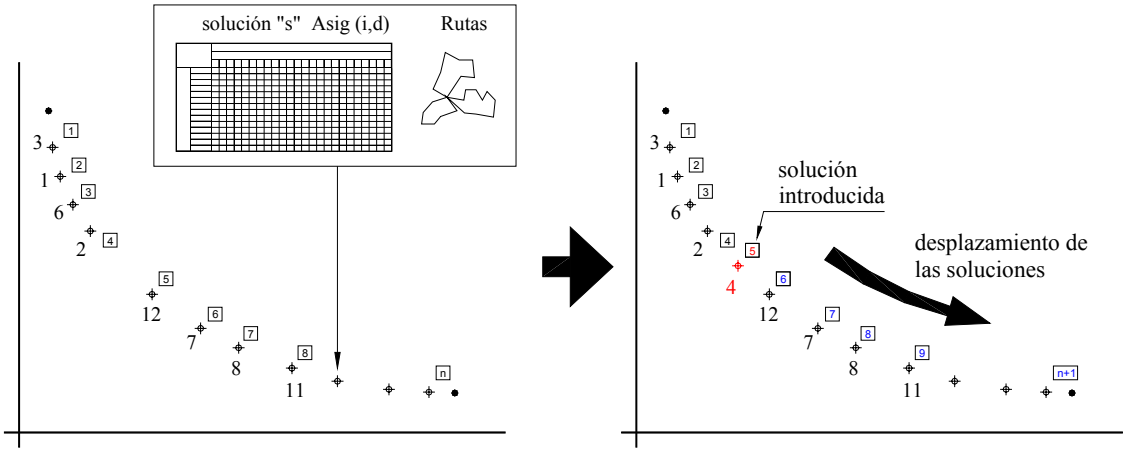


Figura 49. Ejemplo de inserción de una solución no dominada.

Como se puede observar en la figura anterior, al insertar la solución 4, se provoca un desplazamiento de las soluciones siguientes —en azul— de la celda de memoria que ocupaba (identificada mediante un índice), a otra celda de memoria. Este desplazamiento consume gran cantidad de tiempo como más adelante se podrá comprobar. Así mismo, la introducción de una solución puede provocar la salida del conjunto de soluciones no dominadas, de uno o más puntos, por pasar a estrar dominados; en este caso, también se produce un desplazamiento de soluciones al ocupar éstas, los espacios de memoria, ahora vacíos.

A fin de tratar de aumentar la rapidez de este proceso, lo que se hace es 'asociar' de forma unívoca e inequívoca, un índice a cada solución, de forma que, al explorar una nueva solución, si es no dominada, se incluye en el conjunto, pero ahora, lo que se desplaza, no son las soluciones, sino los índices a los cuales han sido asociadas. En la siguiente figura (50) se ilustra un ejemplo de este movimiento.

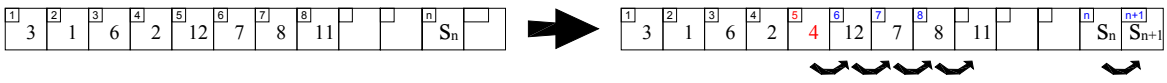


Figura 50. Ejemplo de inserción de una solución no dominada.

La expulsión de soluciones, ahora dominadas, también provoca un desplazamiento de los índices a los nuevos espacios de memoria, ahora libres (figura 51).

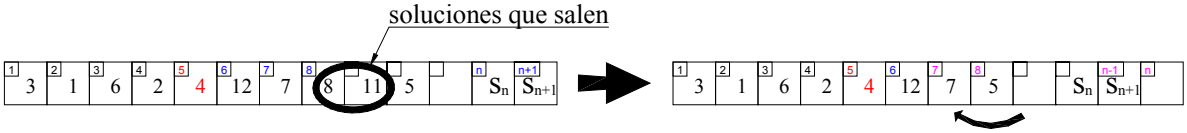


Figura 51. Ejemplo de eliminación de dos soluciónes no dominadas

La estrategia anteriormente descrita precisa de una gestión adecuada de los índices que se asocian a cada solución; en este caso lo que se hace es generar una lista (figura 52) con todos los índices disponibles *(1, 2, ..., n)* que se asocian a todas las soluciones.

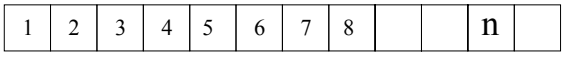


Figura 52. Lista inicial de índices.

Esta estrategia de gestión debe ocuparse tanto del conjunto de índices libres (no asociados a ninguna solución, como del conjunto de índices ocupados, (por estar vinculados a soluciones que forman parte del *conjunto _ND*).

A cada índice se le asocian dos atributos que contienen el índice anterior y el índice posterior respectivamente (figura 53). Además, es necesario indicar dentro de esta secuencia, cuál es el primer índice; esto se hace mediante un apuntador que apunta, inicialmente, al índice *1*.

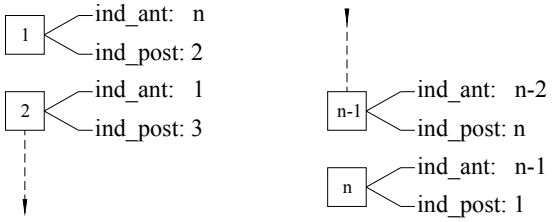


Figura 53. Valores de los atributos de cada índice.

Esta estructura de relaciones de un índice con su anterior y posterior se puede asemejar a una estructura circular en la que cada nodo es un índice, de forma que de ella salen los índices que pasan a estar ocupados (por asociarse a una solución no dominada) y se incorporan los índices libres (por abandonar el conjunto de soluciones no dominadas, otras que ya no lo son).

Al entrar una solución en el conjunto de soluciones no dominadas, lo primero que se hace es asociarla el primer índice que esté libre (indicado por el apuntador); inicialmente, se asocia el índice *a*, con lo cual éste deja de estar disponible; a continuación, lo que se hace es actualizar los atributos del índice anterior *n* y del índice posterior *b* pasando el punteador al índice *b* (figura 54):

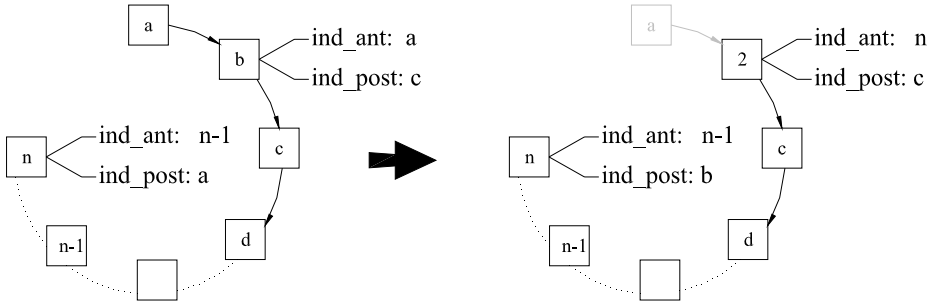


Figura 54. Actualización de atributos tras la ocupación de un índice.

Las soluciones que entran en el *conjunto_ND* lo hacen de una en una, por lo que la actualización de los atributos se hará de uno en uno; ahora bién, puede ser una sola solución o una cadena de soluciones, la que abandone el *conjunto_ND* al pasar a ser dominadas, por la solución que acaba de entrar. Cuando entra una solución, se comprueba su dominancia frente al resto, pudiendo eliminar a una o más soluciones. El índice asociado a las soluciones expulsadas del *Set_ND*, se incorporan al conjunto de índices libres añadiendose en la posición según su orden de dominancia. Si lo que se incorpora es una cadena de índices, se incorpora al conjunto, manteniendo la misma secuencia (figura 55).

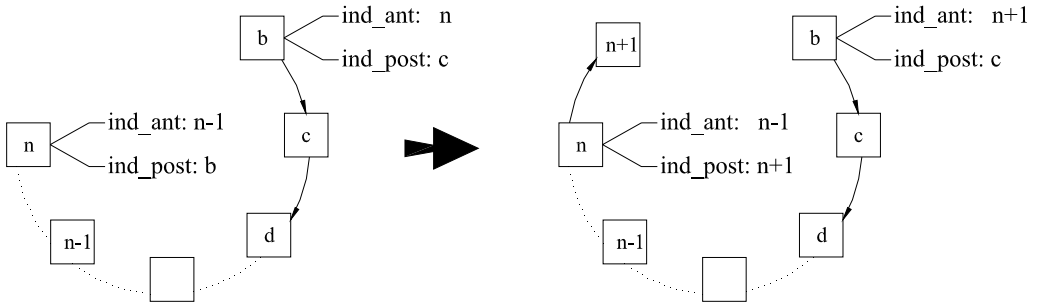


Figura 55. Actualización de atributos tras la liberación de un índice.

En la siguiente figura (56) se muestra la interrelación existente entre ambos grupos de índices, entre los que existe una comunicación bidireccional. Los niveles 1 y 5 muestran, respectivamente, los estados iniciales —en un momento dado— de los índices, así como sus atributos, de forma que el uso de un índice libre (por asociarse a una solución que entra en el conjunto de soluciones no dominadas provoca:

- La actualización de los atributos de los índices restantes del conjunto de índices libres, por desaparecer el índice arriba citado (punto 2).
- La actualización de los atributos del conjunto de índices ocupados, por incorporar al nuevo índice (punto 6).

Si la entrada de una solución al conjunto de soluciones no dominadas, provoca el abandono de una o más soluciones (una solución o una cadena de soluciones), por pasar a ser dominadas por la nueva solución, entoces se produce:

- La incorporación al conjunto de índices libres, del índice o cadena de índices (asociados a la o las soluciones que abandonan el conjunto de soluciones no dominadas) y, por tanto, la actualización de los atributos de los índices último (a partir del cual se incorporan los nuevos índices) y el que, en ese momento, sea primero (pasos 4 y 7).

Los puntos 4 y 8 reflejan el estado final tras estos intercambios.

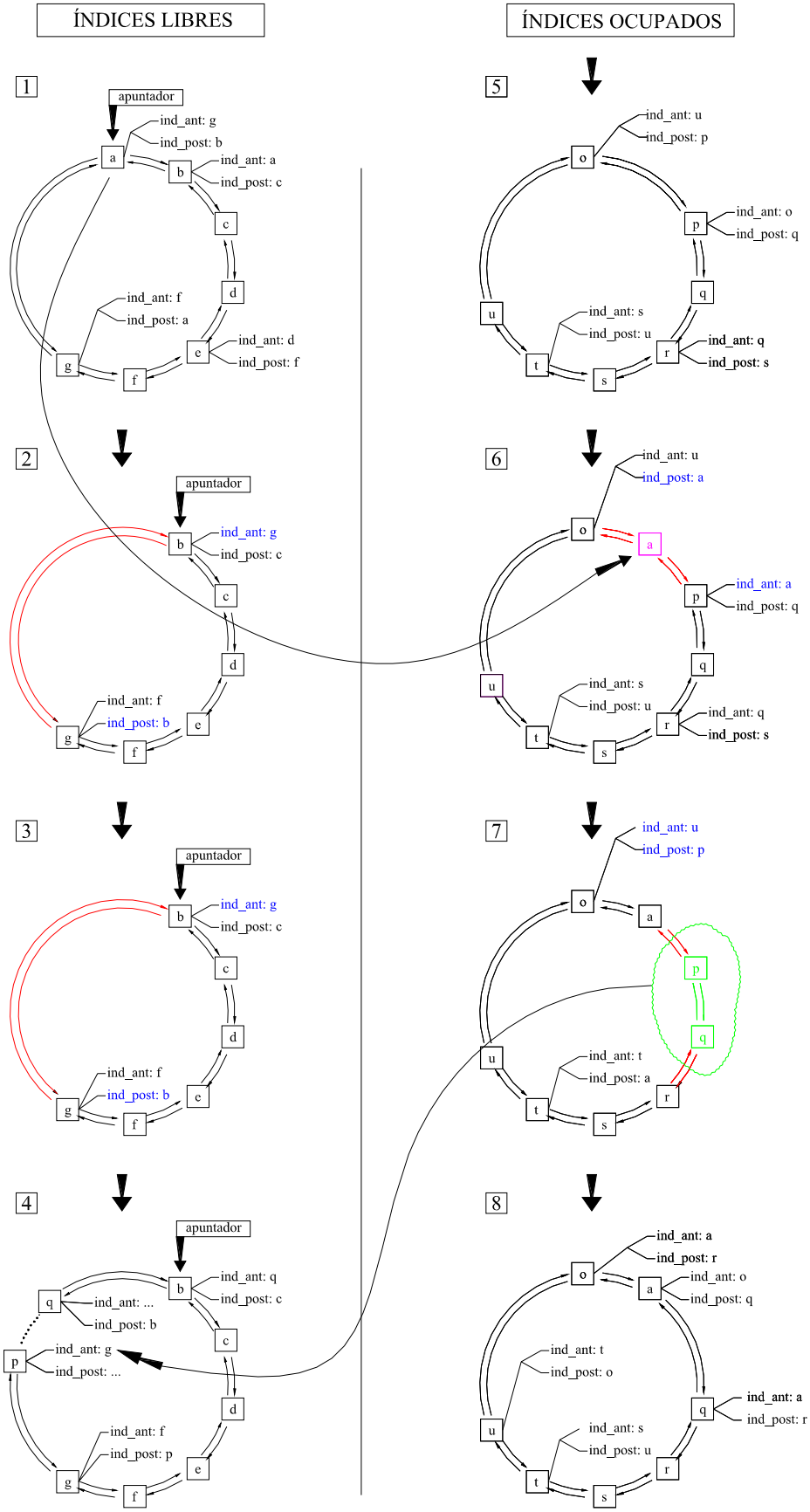


Figura 56. Interrelaciones entre conjuntos de índices.

12.- NSGA II

12.- NSGA II

A fin de poder examinar la 'bondad' de los resultados obtenidos por el algoritmo diseñado para la resolución de este problema, se van a comparar los mismos con los resultados obtenidos por una adaptación del conocido, y ya citado anteriormente, como NSGA II.

Éste es un algoritmo genético que ha demostrado obtener buenos resultados en gran cantidad de problemas multiobjetivo en estos últimos años, por lo que se considera que la comparación con éste puede dar una buena medida sobre la calidad del algoritmo aquí propuesto.

12.1.- CONCEPTOS GENERALES

Encuadrado dentro de los algoritmos evolutivos, se inspira en modelos evolutivos propios de la naturaleza. (Holland *op. cit.* pág. 18).

Desciende del algoritmo NSGA que se basa en la clasificación de puntos en función de su no-dominación, antes de seleccionar a los individuos. Según esto, a todos los puntos no dominados, se les asigna un mismo valor de aptitud, dependiente del tamaño de la población. Tras la conclusión del proceso, todos los puntos quedan clasificados, teniendo mayor posibilidad de reproducirse aquellos con mejor clasificación.

Como se indica en Deb et al. (*op.cit.* pág. 36) este algorimo presenta los siguientes inconvenientes: 1) Elevada complejidad computacioneal 2) Falta de elitismo y 3) la falta de un operador de selección.

El algoritmo NSGA II alivia estas limitaciones, principalmente la relacionada con el orden de complejidad, puesto que pasa a tener un orden de complejidad inferior; también, la presencia de un operador de comparación que crea un conjunto cruzado por combinación de poblaciones de padres y descendientes y que selecciona a los mejores (según su *fitness*).

Un concepto importante, como indican Correa et al. [296], es la 'dominancia', derivado del orden de Pareto, ya citado anteriormente, y que ayuda a la hora de clasificar las diferentes soluciones, teniendo en cuenta la presencia y cuantificación de los diferentes objetivos.

El concepto de 'dominancia' se puede utilizar para encontrar un conjunto de soluciones no dominadas, dentro de una población; el conjunto de todas estas soluciones no dominadas, conforman lo que se denomina un *Frente de Pareto*, es decir, soluciones no dominadas entre sí, pero que dominan al resto de soluciones (figura 57).

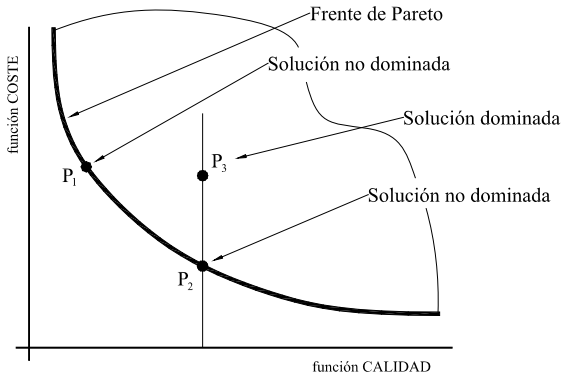


Figura 57. Frente óptimo de Pareto (min-min).

Pueden existir diferentes niveles de no dominancia. Al ejecutar el algorimo anterior por primera vez, se obtiene un conjunto de soluciones no dominadas de mejor nivel, para obetener segundos niveles de no dominancia, se pueden omitir temporalmente las soluciones inicialmente obtenidas (las de mejor nivel) y así sucesivamente, hasta clasificar todas las soluciones dentro de un nivel o frente de Pareto, es decir:

$\mathcal{F}_1$: Conjunto de soluciones no dominadas de P.

$\mathcal{F}_2$: Conjunto de soluciones no dominadas de P-$\mathcal{F}_1$.

$\mathcal{F}_3$: Conjunto de soluciones no dominadas de P-$\{\mathcal{F}_1 \cup \mathcal{F}_2\}$.

y así sucesivamente.

A fin de mejorar el proceso de búsqueda de soluciones no dominadas, se desarrolla el algoritmo NSGA II.

12.2.- DESCRIPCIÓN DEL ALGORITMO NSGA II.

Propuesto por Deb y sus alumnos (Deb et al. *op.cit.* pág. 36), es más eficiente, usa elitismo y un operador de comparación (*crowding*) en función de la la proximidad entre sí de la diferentes soluciones.

En este algoritmo, la población descendiente Q (tamaño N) es creada a partir de la población de los padres P (tamaño N). A continuación, ambas poblaciones se combinan, creando una nueva población R de tamaño 2N; después, se procede a un ordenamiento no dominado, clasificando la población R en diferentes frentes de Pareto. Terminado este proceso, la nueva población se configura a partir de las soluciones del mejor frente no dominado y así sucesivamente. La población inicial puede ser generada por métodos aleatorios o mediante alguna técnica de inicialización.

Al ser N el tamaño de la población y, dado que en el caso del primer cruce, se genera una población R de tamaño 2N, no todas las soluciones de los diferentes frentes de las poblaciones descendientes podrán tener encaje en la nueva población, por lo que tendrán que desaparecer (figura 58).

Puede ocurrir que, cuando se consideren soluciones del último frente de Pareto, las soluciones, que pertenezcan a éste, excedan a las restantes para completar el conjunto de la población descendiente. En este caso, es interesante definir alguna estrategia que permita elegir soluciones que estén en zonas poco pobladas a fin de presentar un frente más amplio, frente a una elección aleatoria de soluciones descendientes. Para favorecer esta idea, se introduce el concepto de ordenamiento no dominado que ordena las soluciones según un parámetro —definido más adelante— prefiriendo, dentro de un mismo frente de Pareto, soluciones ‘más aisladas’ frente a soluciones ‘menos aisladas’ a fin de incentivar la diversidad de las mismas.

Lo anteriormente citado suele ser poco relevante en las primeras iteraciones, ya que hay muchos frentes que sobreviven; ahora bien, según avanza el proceso, hay soluciones que pasan al primer frente, con lo que pueden exceder el número máximo de individuos. Esta cuestión es de vital importancia, por cuanto hay que procurar que las soluciones no rechazadas sean de buena calidad y, a la vez, presentar una adecuada diversidad. Tal y como se configura el algoritmo, cuando todas las soluciones convergen a un frente de Pareto óptimo, se garantiza que las soluciones están distanciadas unas de otras.

Como se puede observar en la figura, inicialmente se crea una población (aleatoria o mediante alguna técnica constructiva). Dicha población se ordena según niveles de no dominancia y se le asigna un *fitness* según dicho nivel. El cruce, la mutación y la selección por torneo se emplean en la creación de la población descendiente. Los principales pasos del algoritmo se describen a continuación.

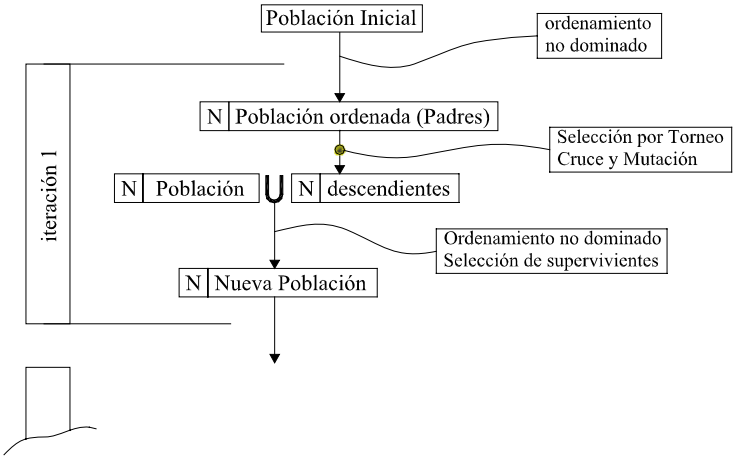


Figura 58. NSGA II. Esquema de funcionamiento.

12.3.- ADAPTACIÓN AL PROBLEMA REAL

A efectos de adaptación, se aportan las siguientes definiciones:

ind(S) : índice de la capa de no-dominancia a la que pertenece *S*.

cwd_dst(S) crowding distance que se define como:

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

donde S_a y S_b son las soluciones justo anterior y posterior a *S* al ordenar su capa de no-dominancia *(F ind(S))* según los valores de *f1* en orden descendente; de igual forma S_c y S_d son las soluciones justo anterior y posterior a *S* al ordenar *F ind(S)* según los valores de f_2 en orden descendente (figura 59). En caso de que *S* tome valores extremos, dentro de su capa de no-dominancia, o bien para f_1 o bien para f_2 entonces *cwd_dst(S)* = ∞.

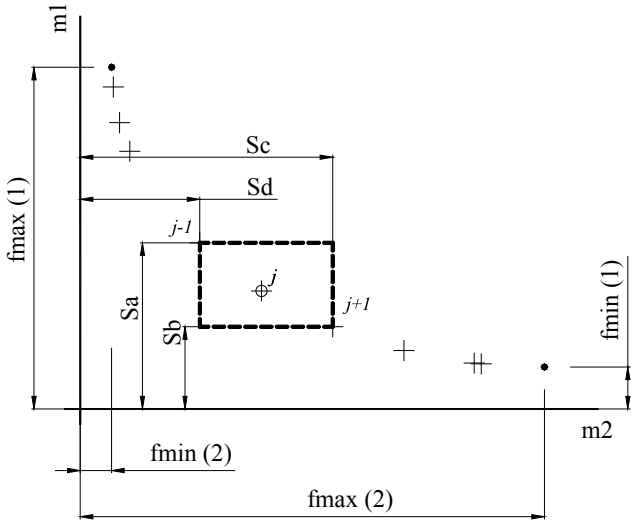


Figura 59. Crowding distance

Con estos valores, se define un orden parcial $\prec$, (operador *Crowded Comparison*) de la forma siguiente:

Sean dos soluciones S y S'

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

Este operador se va a utilizar en la selección de padres y en determinar que soluciones pasan a la siguiente generación. Con esto se trata de favorecer, en primer lugar, la selección de soluciones menos dominadas y, en segundo lugar, soluciones más 'aisladas' dentro de su capa de dominancia para tratar de rellenar esa región.

Para describir la adaptación del algoritmo NSGA-II a este problema se ha de señalar lo siguiente:

Como se ha comentado anteriormente, cada solución tiene dos componentes *Asig*, matriz que índica que puntos se visitan cada día y *Rut*, el conjunto de rutas para cada día. Ahora bien, los operadores de cruce y mutación de NSGA-II solo van a actuar sobre *Asig*.

La matriz booleana *Asig* de *n* filas y HT columnas puede ser representada como un vector de $n \cdot HT$ componentes; esto favorece las operaciones clásicas de cruce mutación, etc.

Tras las operaciones de cruce y mutación, las matrices resultantes pueden no ser factibles; concretamente puede haber puntos *x* con intervalos entre visitas consecutivas superiores a *max_espx*. Por tanto se ha dotado al algoritmo de un procedimiento que lleve a estas matrices a la factibilidad.

Cuando se obtiene una matriz *Asig* (ya sea por la generación inicial o tras las operaciones de cruce y mutación), y una vez transformada en factible, las correspondientes rutas, *Rut,* se obtienen ejecutando los algoritmos **Constructivo_Rutas** y **Búsqueda_Local_Rutas** a cada día y para cada conjunto de puntos a visitar. Obsérvese que estos dos procedimientos son determinísticos. Por tanto, en este algoritmo, todas las soluciones vienen unívocamente determinadas por la matriz *Asig*.

Sea *n_pob* el tamaño de la población inicial, *p_mut*, la probabilidad de mutación o cambio de cada componente de cada matriz *Asig*, la adaptación del algoritmo NSGA-II puede ser descrita de la forma siguiente:

Generar población de soluciones iniciales P_0 de tamaño *n_pob.*
Aplicar el procedimiento *Llevar_a_factibilidad* a aquellas soluciones de P_0 que no sean factibles.
Determinar $f_1(S)$ y f_2 (S) para las soluciones $S \in P_0$.
Dividir P_0 en capas de no dominancia;
hacer $t:=0$
Repetir:
    Calcular la *crowding distance* para los elementos de P_t.
    Generar poblacion Q_t de soluciones hijas de tamaño *n_pob* a partir de P_t.
    Aplicar el operador mutación a los elementos de Q_t.
    Aplicar el procedimiento **Llevar_a_factibilidad** a aquellas soluciones de *Qt* que no sean factibles.
    Determinar $f_1(S)$ y $f_2(S)$ para las soluciones $S \in Q_t$.
    Hacer $R_t = P_t \cup Q_t$ y dividir R_t en capas de no dominancia;
    Formar P_t+1 con los *n_pob* primeros elementos de R_t según el orden parcial $\prec$
    Hacer $t:=t+1$;
Hasta alcanzar criterio de parada;

La generación de soluciones iniciales se hace generando aleatoriamente los valores (TRUE o FALSE) de la correspondiente matriz *Asig*. Como se ha comentado anteriormente, las matrices resultantes de este paso —como las del paso 6.d— pueden no ser factibles. Concretamente puede haber puntos *x* con intervalos entre visitas consecutivas superiores a *max_espx*. Por tanto se ha dotado al algoritmo de un sencillo procedimiento denominado **Llevar_a_factibilidad** que realiza pequeñas modificaciones en estas matrices para que sean factibles. Básicamente actúa de la forma siguiente:

Dada una matriz *Asig* para cada punto *x* se define

$$Suma\_\inf = \sum_{r=1}^{n\,\mathrm{int}} \max\{0, Inter_r - \max\_sep_x\}$$

donde

n_{int} : número de intervalos entre 2 recogidas consecutivas.
$Inter_r$: longitud del r-ésimo intervalo.

Por tanto *Suma_inf* es una medida de infactibilidad para cada fila *x* de la matriz. La figura 60 muestra el ejemplo de una fila. En este caso *HT = 9*, el numero de recogidas es de 5 (además de las recogidas en *0*, y *HT+1*), *nint = 6*. Supóngase *max_espx = 3*, en el correspondiente punto *x*, entonces *Suma_inf =1*.

d	0	1	2	3	4	5	6	7	8	9	HT+1
	X	X				X		X	X	X	X

Figura 60.- Ejemplo de recogidas. Para una mejor visualización Las celdas con X indican recogida en ese día (TRUE) y las vacías no-recogida (FALSE).

El procedimiento va realizando cambios en la fila hasta conseguir la factibilidad, es decir, *Suma_inf = 0*. Para ello se realizan dos tipos de cambios: a) cambiar una recogida de un día a otro; y b) añadir una recogida (sin cambiar el resto). Las figuras 61 y 62 ilustran estos cambios en el ejemplo de la figura 13.

d	0	1	2	3	4	5	6	7	8	9	HT+1
	X	X		X		X		X		X	X

Figura 61.- Ejemplo de cambio tipo a. La recogida del día 8 se traslada al día 3.

d	0	1	2	3	4	5	6	7	8	9	HT+1
	X	X		X		X		X	X	X	X

Figura 62.- Ejemplo de cambio tipo b. Se añade la recogida en el día 3.

En cada paso se realiza el cambio que mas reduzca el valor de *Suma_inf*. En caso de 'empate', es decir, dos cambios con la misma reducción, se preferirá el que dé lugar a nuevos intervalos entre visitas más equilibrados. La medida de 'equilibrio' viene dada por la máxima longitud de los nuevos intervalos. La figura 63 ilustra un cambio (sobre el ejemplo de la figura 60) que da lugar a dos intervalos menos equilibrados que el de las figuras 61 y 62.

d	0	1	2	3	4	5	6	7	8	9	HT+1
	X	X	X			X		X	X	X	X

Figura 63.- Ejemplo de cambio no equilibrado. Se añade la recogida en el día 3.

Obsérvese como el cambio de la figura 53, añadir una recogida el día 2, produce 2 nuevos intervalos: del día 1 al 2, y del día 2 al 5. Por tanto 2 intervalos de longitudes 1 y 3. Sin embargo el cambio de las figuras 52 produce 2 intervalos (del día 1 al 3, y del 3 al 5) de longitud 2. Por tanto, aunque ambos cambios supongan la misma reducción en la infactibilidad (si $max\_esp_x = 3$), se preferirá el cambio de la figura 52. Este criterio de equilibro entre intervalos pretende favorecer la función de calidad en las soluciones resultantes. Por tanto, sea una matriz *Asig* el procedimiento *Llevar_a_factibilidad* actúa de la forma siguiente.

Procedimiento **Llevar_a_factibilidad** (var *Asig*)

Para cada $x = 1 \ldots n$ hacer:

- Determinar el *Suma_inf* correspondiente a la fila *x*
- Mientras *Suma_inf* > 0 hacer
  Determinar el cambio que más reduzca *Suma_inf* en la fila x; en caso de empate usar el criterio de mayor equilibro, según se ha explicado anteriormente
  Ejecutar dicho cambio

Una vez obtenida la correspondiente matriz *Asig* factible de una solución *S*, se obtiene fácilmente el valor $f_2(S)$ (función de 'no-calidad'). La componente *Rut* se obtiene con los algoritmos **Constructivo_Rutas** *y* **Búsqueda_Local_Rutas** para cada día y para los puntos a visitar determinados por *Asig*. De esta manera se obtiene el valore de $f_1(S)$.

La división de una determinado conjunto de soluciones (P_0 en el paso 4, o R_t en el 6.*f*) se realiza mediante el procedimiento *fast-not-dominated-sort* propuesto en el propio trabajo de Deb et al. El proceso de *renovación* (paso 6.g), es decir, determinar los elementos de R_t que pasan a la siguiente generación ("sobreviven") se realiza de la forma siguiente:

- Denótese por $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3 \ldots$ las capas de no-dominancia en que se divide R_t.
- Hacer *i:=i* y $P_{t+1} = \varnothing$.
- Mientras $|P_{t+1}| + |\mathcal{F}_i| \leq n\_pob$ hacer $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$ *e i=i+1*.
- Calcular la *crowding distance* para los elementos de $\mathcal{F}_i$.
- Incluir en P_{t+1} los *n_pob* - $|P_{t+1}|$ elementos de $\mathcal{F}_i$ con mayor *crowding distance.*

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

13.- PRUEBAS COMPUTACIONALES

13.- PRUEBAS COMPUTACIONALES

13.1.- DESCRIPCIÓN DE LAS INSTANCIAS GENERADAS

A fin de validar la robustez del algoritmo diseñado, así como poder analizar su comportamiento frente a la variación de diferentes parámetros del mismo y frente a NSGA II, se van a generar, de forma aleatoria, una serie de instancias ficticias, las cuales servirán como punto de partida para las diferentes pruebas. Acontinuación se detalla la estructura de las mismas (tabla 10).

Semilla	Número de puntos	Horizonte temporal
1-2-3-4-5	20	10
1-2-3-4-5	30	15
1-2-3-4-5	40	20
1-2-3-4-5	50	30

Tabla 10.- Descripción de las instancias generadas.

Para cada semilla se genera una instancia, por lo que se generan un total de veinte instancias, presentadas a través de un fichero de tipo ASCII, con la siguiente estructura:

1. Coord X, coord. Y, tasa de generación, *max_sep.*
2. Matriz de distancias.
3. Número de tipos de día (1, 2, 3).
4. Número de tipos de vehículos (2).
5. Número de vehículos de cada tipo (2).
6. Coste de cada tipo de vehículo, según tipo de día.

Los ficheros generados son los siguientes:

- Instancia_20-10-s.dat.
- Instancia_30-15-s.dat.
- Instancia_40-20-s.dat.
- Instancia_50-30-s.dat. donde s $\in \{1\ldots5\}$

13.2.- REPRESENTACIÓN GRÁFICA DE LAS INSTANCIAS GENERADAS

A fin de poder visualizar gráficamente el aspecto de las instancias generadas, se adjuntan para cada tamaño, las dos primeras instancias, en las que cada núcleo aparece representado mediante un símbolo que consta de varios datos, los cuales se explican en la siguiente figura (64).

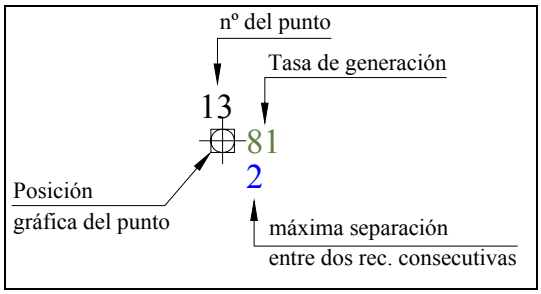


Figura 64. Desripción del símbolo.

Instancia 20-10-1.dat

Coste fijo de los vehículos, según tipo de vehículo (flota propia o ajena) y tipo de día (Laboral = 1, Sábado = 2, Festivo = 3):

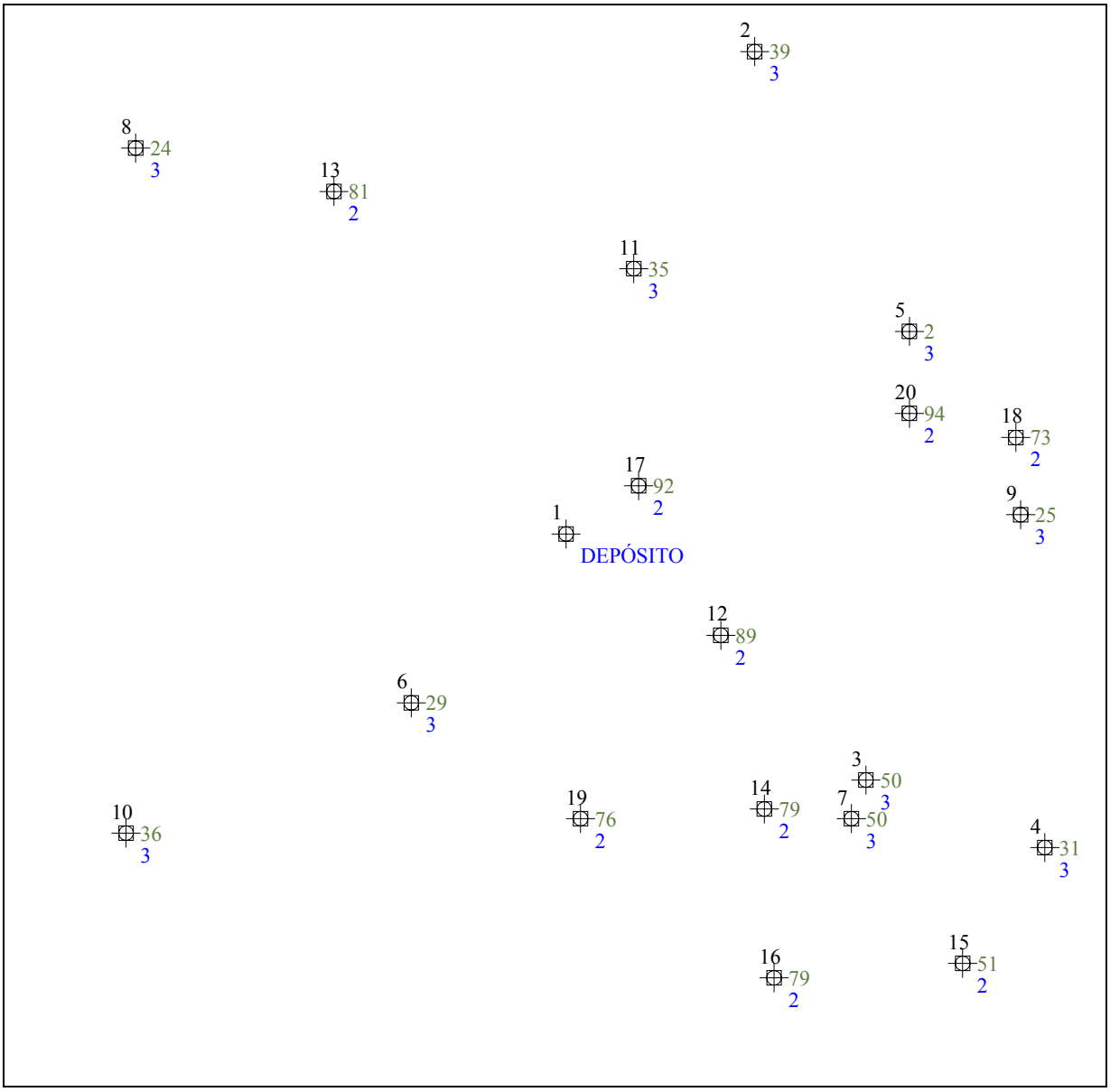
0 100 200
100 225 375

Coste variable de los vehículos:

Cv = 1 Ud / km.

Capacidad de los vehículos:

Cap = 1200 Ud.



Instancia 20-10-2.dat

Coste fijo de los vehículos, según tipo de vehículo (flota propia o ajena) y tipo de día (Laboral = 1, Sábado = 2, Festivo = 3):

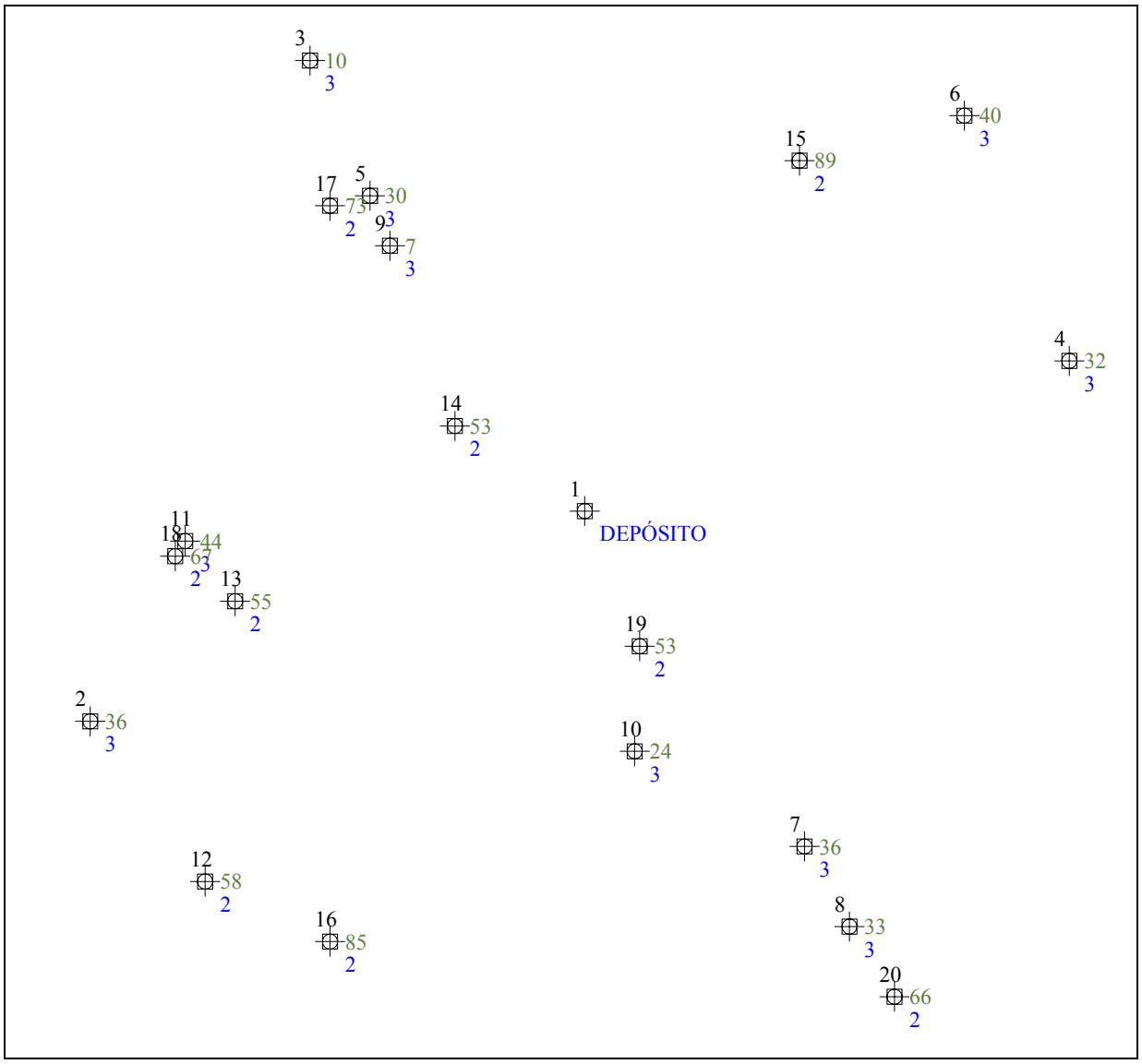
0	100	200
100	225	375

Coste variable de los vehículos:

Cv = 1 Ud / km.

Capacidad de los vehículos:

Cap = 1200 Ud.



Instancia 30-15-1.dat

Coste fijo de los vehículos, según tipo de vehículo (flota propia o ajena) y tipo de día (Laboral = 1, Sábado = 2, Festivo = 3):

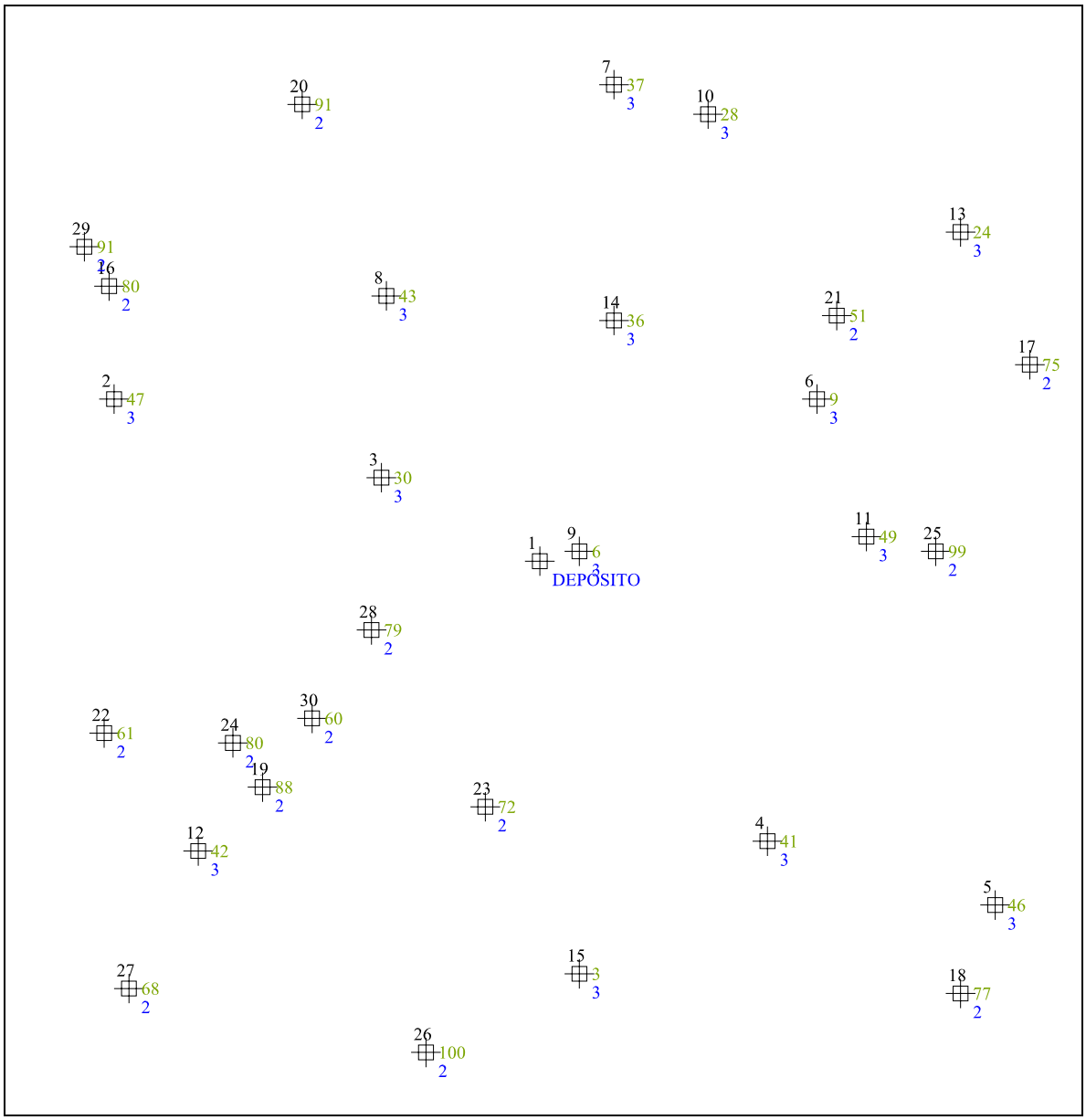
0	100	200
100	225	375

Coste variable de los vehículos:

Cv = 1 Ud / km.

Capacidad de los vehículos:

Cap = 1200 Ud.



Instancia 30-15-2.dat

Coste fijo de los vehículos, según tipo de vehículo (flota propia o ajena) y tipo de día (Laboral = 1, Sábado = 2, Festivo = 3):

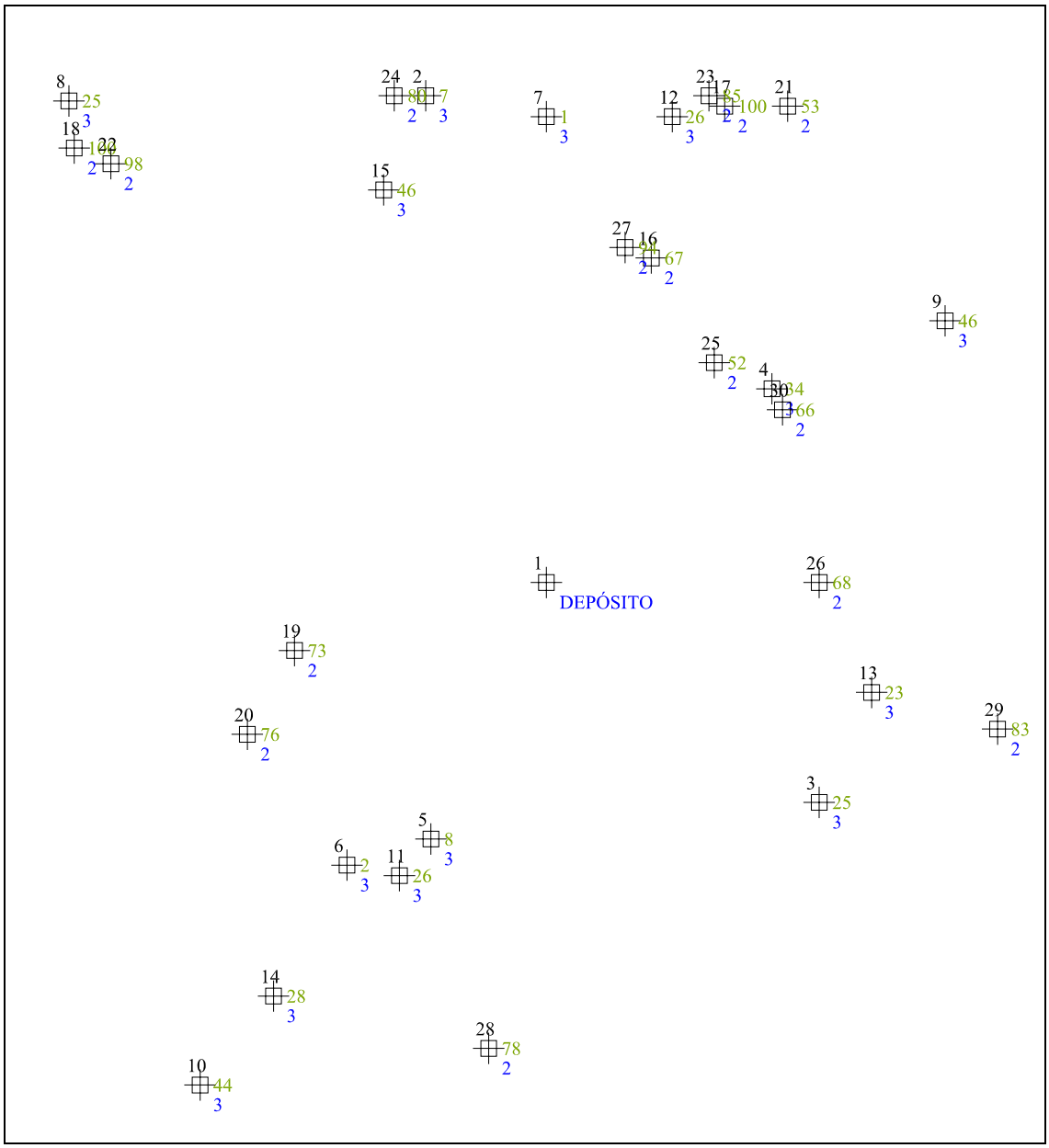
0	100	200
100	225	375

Coste variable de los vehículos

Cv = 1 Ud / km.

Capacidad de los vehículos

Cap = 1200 Ud.



Instancia 40-20-1.dat

Coste fijo de los vehículos, según tipo de vehículo (flota propia o ajena) y tipo de día (Laboral = 1, Sábado = 2, Festivo = 3):

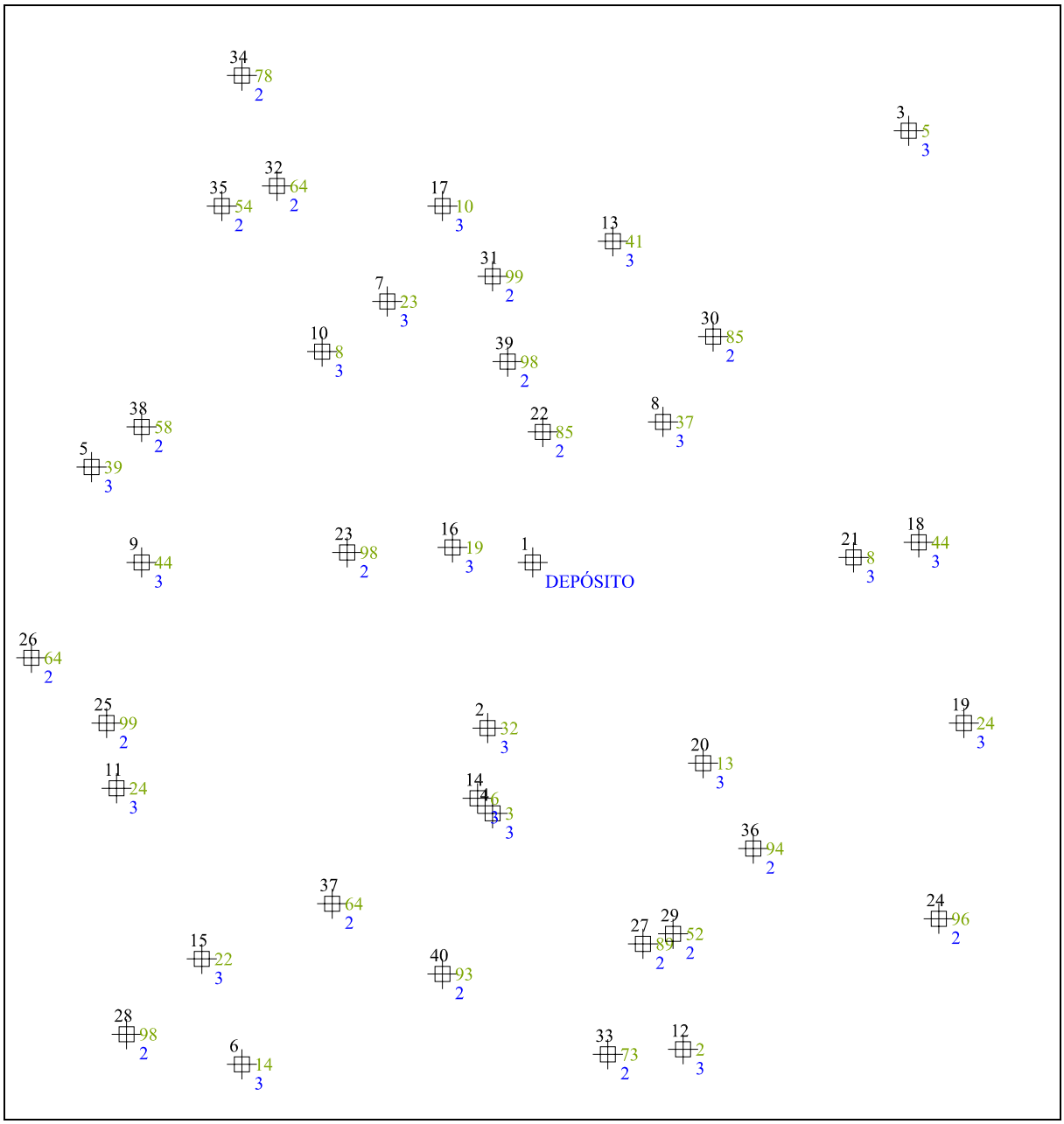
0	100	200
100	225	375

Coste variable de los vehículos:

Cv = 1 Ud / km.

Capacidad de los vehículos:

Cap = 1200 Ud.



Instancia 40-20-2.dat

Coste fijo de los vehículos, según tipo de vehículo (flota propia o ajena) y tipo de día (Laboral = 1, Sábado = 2, Festivo = 3)

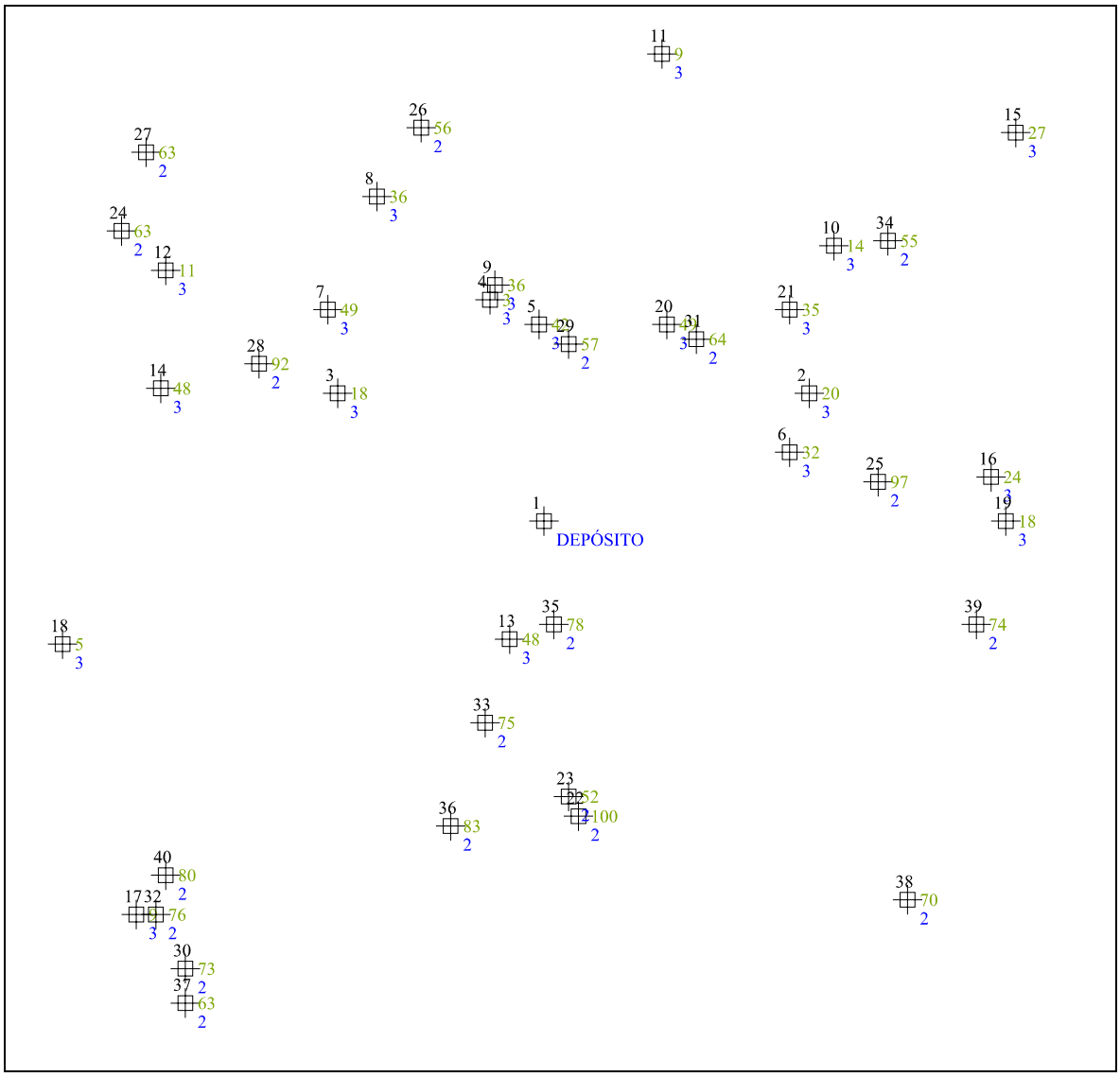
0	100	200
100	225	375

Coste variable de los vehículos

Cv = 1 Ud / km.

Capacidad de los vehículos

Cap = 1200 Ud.



Instancia 50-30-1.dat

Coste fijo de los vehículos, según tipo de vehículo (flota propia o ajena) y tipo de día (Laboral = 1, Sábado = 2, Festivo = 3)

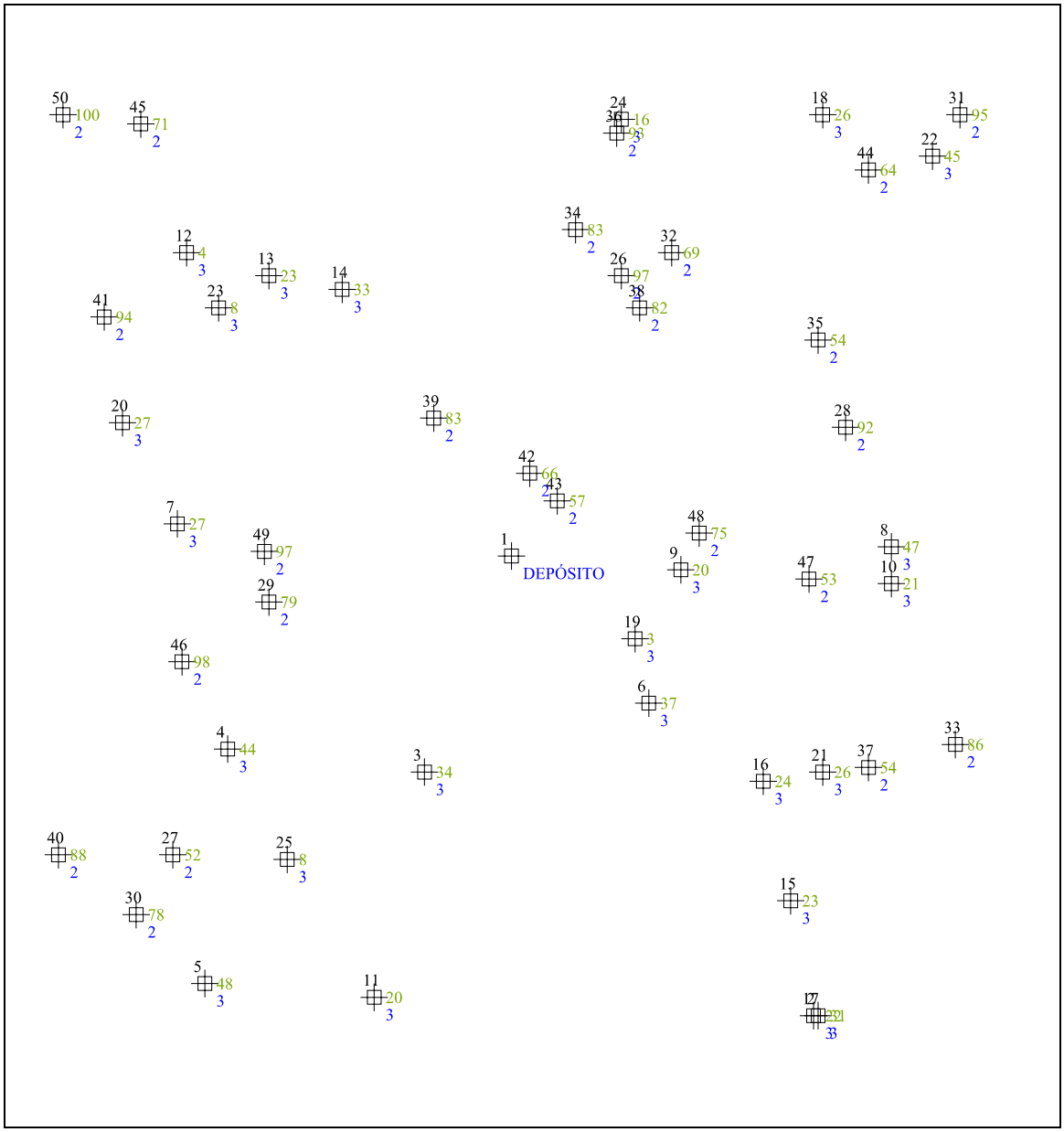
0	100	200
100	225	375

Coste variable de los vehículos

Cv = 1 Ud / km.

Capacidad de los vehículos

Cap = 1200 Ud.



Instancia 50-30-2.dat

Coste fijo de los vehículos, según tipo de vehículo (flota propia o ajena) y tipo de día (Laboral = 1, Sábado = 2, Festivo = 3)

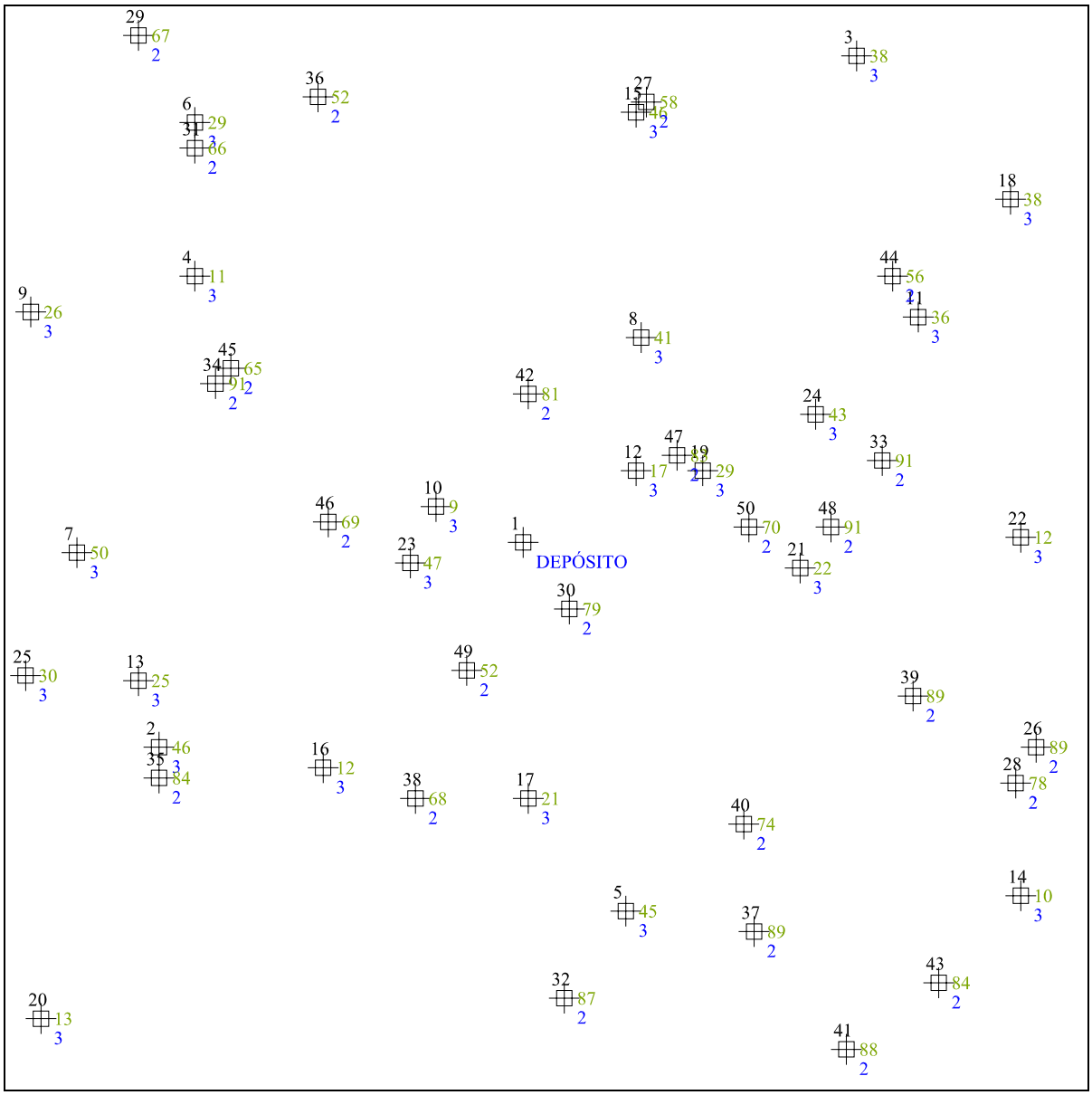
0	100	200
100	225	375

Coste variable de los vehículos

Cv = 1 Ud / km.

Capacidad de los vehículos

Cap = 1200 Ud.

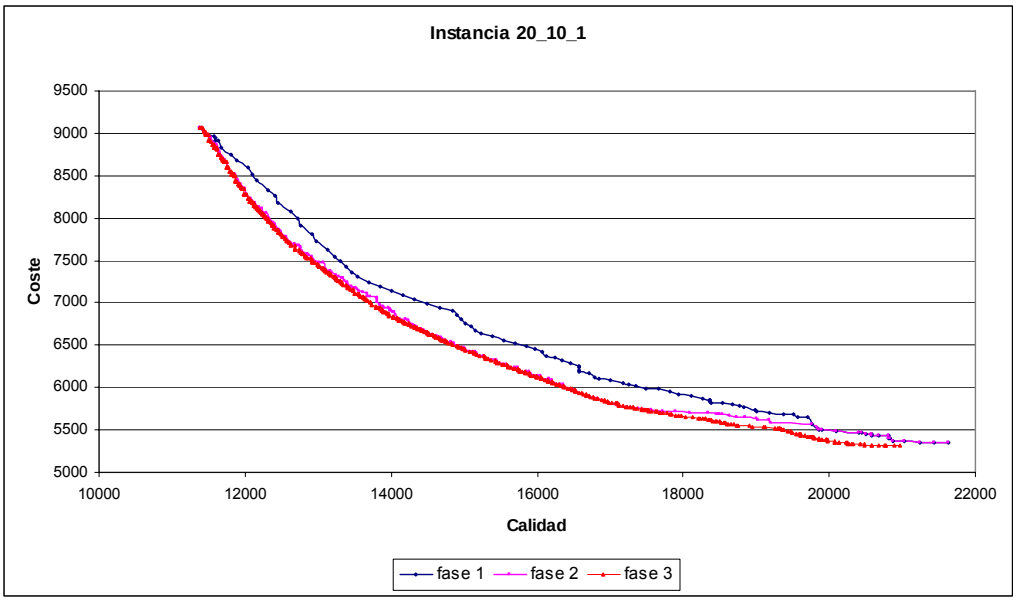


13.3.- RESULTADOS COMPUTACIONALES CON LAS INSTANCIAS FICTICIAS

Tras ejecutar el algoritmo, sobre las instancias de prueba anteriormente citadas, se obtiene un conjunto de soluciones; a efectos ilustrativos, se representa gráficamente la primera instancia (semilla 1) para cada tamaño de instancia.

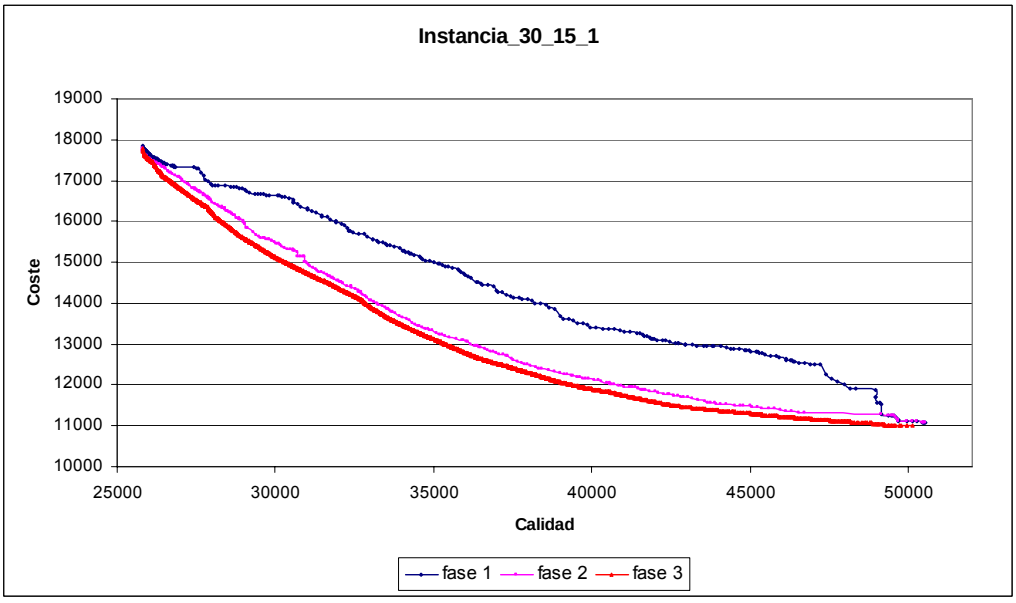
Instancia 20-10-1.dat

FASE 1 = 101 soluciones.
FASE 2 = 215 soluciones.
FASE 3 = 671 soluciones.



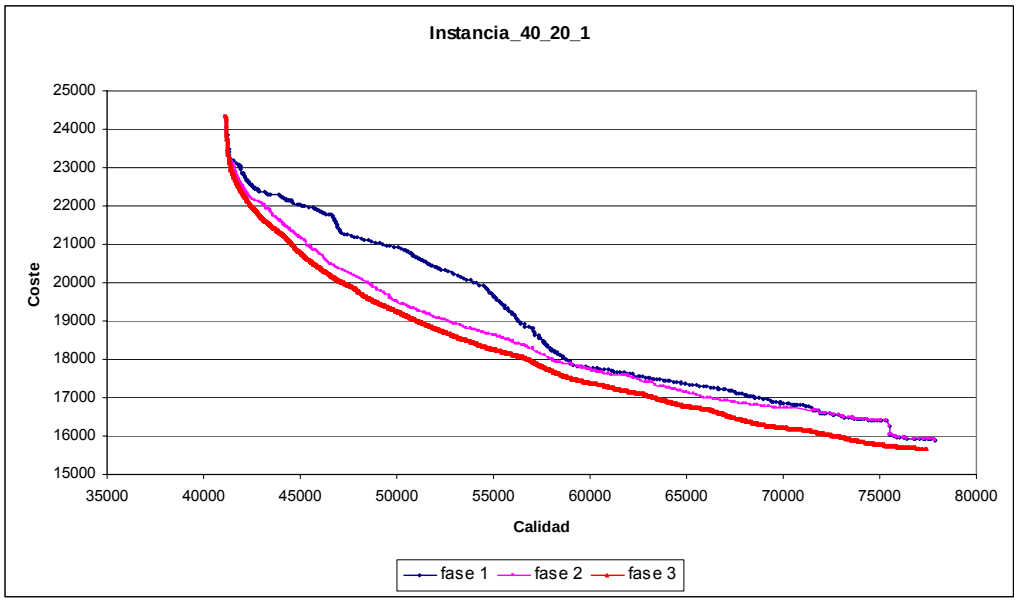
Instancia 30-15-1.dat

FASE 1 = 265 soluciones
FASE 2 = 317 soluciones
FASE 3 = 2170 soluciones



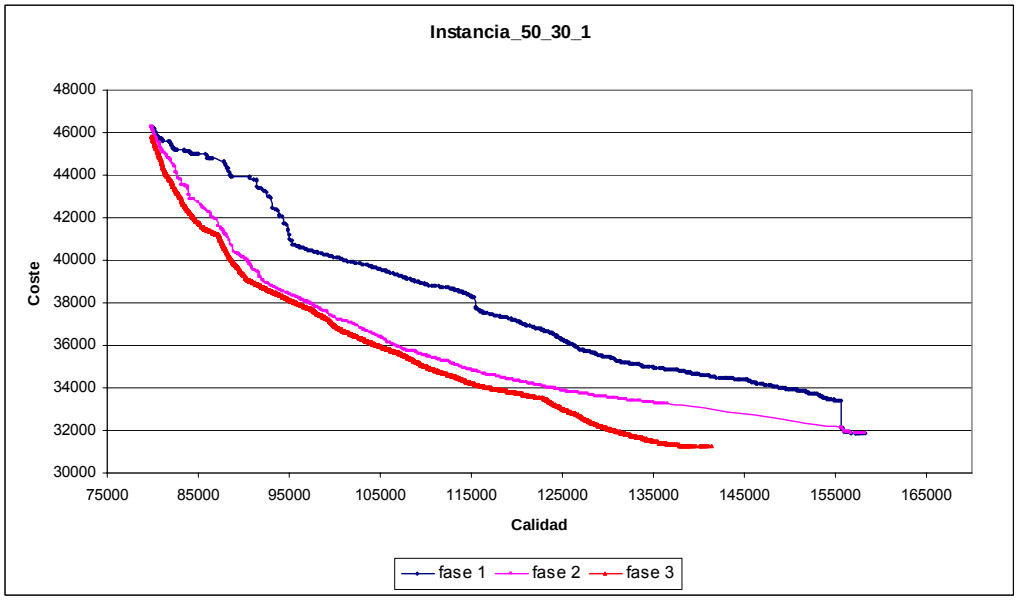
Instancia 40-20-1.dat

FASE 1 = 431 soluciones
FASE 2 = 391 soluciones
FASE 3 = 4405 soluciones



Instancia 50-30-1.dat

FASE 1 = 849 soluciones
FASE 2 = 592 soluciones
FASE 3 = 8071 soluciones



Se puede obervar en estas gráficas, claramente diferenciadas las tres fases (fase 1-azul, fase 2-magenta, fase 3- rojo). La fase 1 comienza con una buena solución de calidad, localiza una solución aceptable para el coste y vuelve a la solución inicial. En la fase 2 se ejecuta una búsqueda mediante una función mixta y, por último, la fase 3, ejecuta una intensificación sobre las soluciones obtenidas en la fase 2. A medida que se ejecutan las fases, se produce un desplazamiento de soluciónes hacia el punto

ideal, lo que da una idea del buen comportamiento del algoritmo. Se observa también la diferencia entre el número de soluciones en las fases 1 y 2, y la fase 3.

13.4.- MÉTODOS DE ACELERACIÓN: RESULTADOS

13.4.1.- Búsqueda Local Rápida

Para contrastar el efecto de la estrategia Búsqueda_Local_Rápida dentro del procedimiento **Búsqueda_Local_Rutas** se han realizado una serie de pruebas computacionales con ambas variantes de este procedimiento: sin Búsqueda_Local_Rápida, y con Búsqueda_Local_Rápida.

Se ha de recordar que el procedimiento **Búsqueda_Local_Rutas**, a partir de una solución inicial, propone otra solución, igual o mejor, al problema del diseño de las rutas diarias. Se han considerado dos grupos de instancias para este problema. Las instancias del primer grupo se definen tomando cada una de las instancias definidas en la sección 13.1 y considerando la recogida en todos los puntos, de las cantidades acumuladas de RU en un día. El tipo de día se genera aleatoriamente. Las instancias del segundo grupo son exactamente igual que las del primero, pero las cantidades a recoger se han multiplicado por 1,8.

Se han considerado 10 soluciones iniciales aleatorias para cada instancia. Para cada una de ellas y con cada solución inicial se han ejecutado ambas variantes del procedimiento **Búsqueda_Local_Rutas**.

A continuación (tabla 11) se muestran los estadísticos de los tiempos de computación empleados por cada variante (sin Búsqueda Local Rápida, No BLR, y con Búsqueda Local Rápida, BLR) por grupo y tamaño. Concretamente se muestran el mínimo, el máximo, la desviación típica y la media.

INSTANCIAS		MÍNIMO		MÁXIMO		DESVIACIÓN TÍPICA		MEDIA	
Grp	Tamaño	No BLR	BLR	No BLR	BLR	No BLR	BLR	No BLR	BLR
I	20 – 10	0	0	0,016	0,016	0,00763	0,00737	0,0054	0,0047
	30 – 15	0,016	0,016	0,062	0,063	0,01259	0,01469	0,0368	0,0374
	40 – 20	0,062	0,062	0,14	0,125	0,02255	0,02149	0,0974	0,0939
	50 – 30	0,156	0,125	0,375	0,359	0,06519	0,06321	0,2322	0,1928
II	20 – 10	0	0	0,016	0,016	0,00786	0,00763	0,0062	0,0054
	30 – 15	0	0	0,031	0,032	0,00505	0,00724	0,0156	0,0156
	40 – 20	0,015	0,015	0,047	0,032	0,01049	0,00707	0,0258	0,0195
	50 – 30	0,015	0	0,11	0,047	0,02343	0,01280	0,0424	0,0232

Tabla 11.- Resumen de los resultados de cada variante según el grupo y tamaño.

Para contrastar si los tiempos empleados con búsqueda local rápida son significativamente menores estadísticamente hablando, se ha realizado un test para cada grupo y tamaño de instancia en el que se comprueba si el valor medio de la diferencia de tiempos (No BLR - BLR) es significativamente menor que 0. La tabla 12 muestra los resultados, concretamente, la media de la diferencia de tiempos, su error, el valor del estadístico *t* y la cola de probabilidad unilateral.

De la tabla 12 se puede concluir que el uso de la estrategia búsqueda local rápida no produce una reducción significativa en las instancias más pequeñas (20-10 y 30-15 en ambos grupos). Incluso en las instancias de tamaño 30-15 produce un pequeño aumento en los tiempos de cálculo. Sin embargo, en las

instancias de mayor tamaño, 40-20 y 50-30, las diferencias sí que son claramente significativas en ambos grupos.

Instancias		media	Error de la media	Estadístico t	Cola de probabilidad
Grupo	Tamaño				
I	20 – 10	0,00075	0,0028845	0,260	0,3988
	30 – 15	-0,00065	0,0017893	-0,363	0,8602
	40 – 20	0,00355	0,0022740	1,561	0,0018
	50 – 30	0,03940	0,0040049	9,838	0,0000
II	20 – 10	0,00080	0,0030955	0,258	0,3994
	30 – 15	-0,00005	0,0016262	-0,031	0,5121
	40 – 20	0,00630	0,0017337	3,634	0,0009
	50 – 30	0,01920	0,0035419	5,421	0,0000

Tabla 12.- Resultados de los test estadísticos sobre valor medio de las diferencias.

Las conclusiones, por tanto, son prácticamente iguales en ambos grupos. Como se ha comentado antes, las instancias que componen ambos grupos se diferencian en que las cantidades a recoger se han multiplicado por una constante, para forzar al uso de mayor número de vehículos. Sin embargo, es el tamaño de la instancia y no el número de vehículos usados, el factor determinante en la significación de la reducción del tiempo de computación por el uso de búsqueda local rápida. A continuación, en la tabla 13, se muestra los vehículos medios usados en las soluciones por cada grupo y tamaño de instancia.

Grupo	I				II			
Tamaño	20-10	30-15	40-20	50-30	20-10	30-15	40-20	50-30
N.º vehículos	1	2	2	3	2	3	3	4

Tabla 13.- Número medio de vehículos de las soluciones obtenidas por grupo y tamaño.

13.4.2.- Árboles Binarios en Tabu_Search

Al igual que antes y, para contrastar el efecto del uso de la estructura de árboles binarios en el procedimiento **Tabu_Search**, se han realizado una serie de pruebas. Concretamente se han ejecutado dichos procedimientos considerando ambas variantes, sin árbol y con árbol, en una serie de instancias y para diferentes funciones objetivos F_λ (es decir, diferentes valores de λ).

Se han considerado cuatro tamaños de las instancias descritas con anterioridad: 20-10, 30-15, 40-20 y 50-30; para cada tamaño, se han tomado 2 instancias y generado aleatoriamente 10 valores de λ para cada una de ellas, (por tanto, 20 instancias por tamaño). La solución inicial es, en todos los casos, la obtenida por el procedimiento **Generador_Calidad** para cada instancia.

A continuación se muestran los resultados de estas pruebas. En la tabla 14 se muestran una serie de estadísticos de los tiempos de computación para cada tamaño y cada variante usada. Concretamente se muestran el mínimo, el máximo, la desviación típica y la media.

	Mínimo		Máximo		Desviación típica		Media	
Tamaño	No árbol	Arbol	No árbol	Arbol	No árbol	Arbol	No árbol	Arbol
20 – 10	0	0	0,109	0,063	0,03333	0,02407	0,0407	0,02415
30 – 15	0	0	0,734	0,532	0,26910	0,20453	0,3446	0,25465
40 – 20	0	0	2,594	1,625	0,93101	0,64742	1,1203	0,8555
50 – 30	0	0	7,938	5,188	2,90274	2,00684	3,06015	2,2961

Tabla 14.- Resumen de los resultados de cada variante según el tamaño.

Como se puede comprobar los tiempos medios en la variante que usa árbol binario son menores que cuando no se usan. Para contrastar si estos tiempos son significativamente menores, estadísticamente hablando, se ha realizado un test para cada tamaño de instancia en el que se comprueba si el valor medio de la diferencia de tiempos (arbol - No árbol) es significativamente menor que 0. La tabla 15 muestra los resultados, concretamente la media de la diferencia de tiempos, su error, el valor del estadístico *t* y la cola de probabilidad unilateral.

Tamaño	media	Error de la media	Estadístico *t*	Cola de probabilidad
20 – 10	-0,01655	0,0030869	5,361	0,0000
30 – 15	-0,08995	0,0167337	5,376	0,0000
40 – 20	-0,26480	0,0784830	3,374	0,0016
50 – 30	-0,76405	0,2096827	3,644	0,0009

Tabla 15.- Resultados de los test estadísticos sobre valor medio de las diferencias.

De los resultados de la tabla 15 que resumen el test de contraste de medias, se puede concluir que en todos los tamaños analizados, el empleo del árbol binario consigue una reducción significativa en los tiempos de cálculo. En la figura 65 muestra la evolución de los tiempos medios de computación.

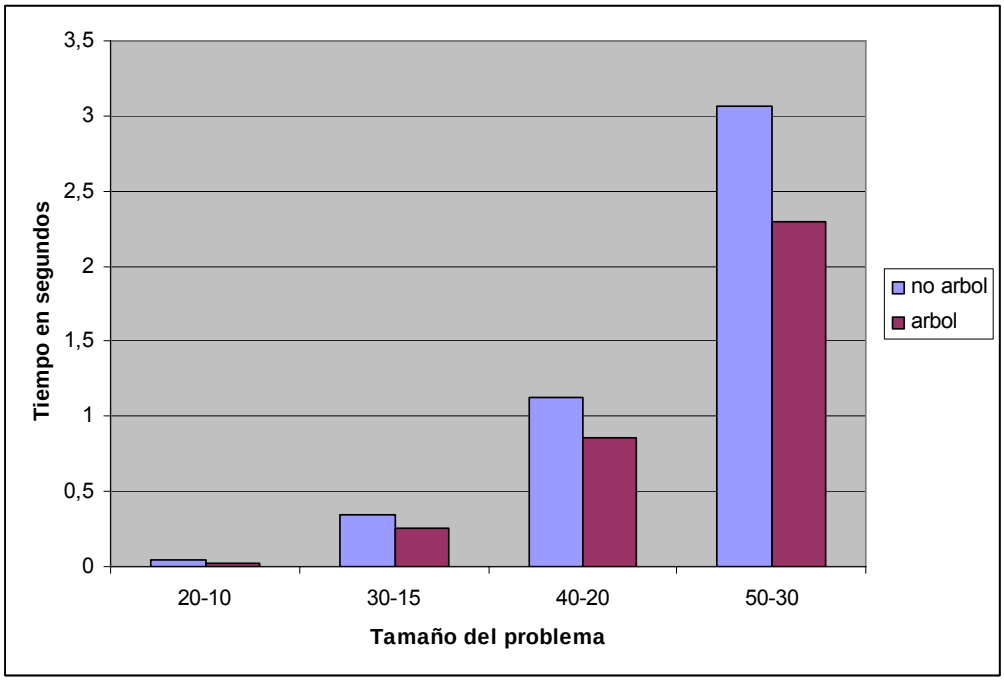


Figura 65.- Tiempo medio de computación de Tabu_Search según tamaño de instancias y la variante usada.

Aunque la reducción en el tiempo de cálculo no es considerable, sí se entiende que es más que apreciable.

13.4.3.- Movimientos por índices

Finalmente, para contrastar el efecto del uso de la gestión de índices se han realizado una serie de pruebas. Concretamente, se ha ejecutado dicho procedimiento considerando ambas variantes —sin gestión de índices y con gestión de índices— en las instancias generadas anteriormente.

Se han considerado 3 tamaños de instancias: 20-10, 30-15 y 40-20. No se han considerado la instancia de tamaño 50-30 por resultar excesivo el tiempo de computación. Para cada tamaño, se han considerado 5 instancias. En la tabla 16 se muestran los tiempos de computación empleados por cada variante y cada instancia, así como la reducción obtenida.

Tamaño Instancia	Instancia	Sin Gestión Indices	Con Gestión Indices	Reducción (%)
20-10	1	18.80	5.67	69.84
	2	19.18	4.44	76.85
	3	124.11	6.63	94.66
	4	31.14	4.32	86.13
	5	19.68	5.62	71.44
30-15	1	1270.33	57.36	95.48
	2	1364.18	81.34	94.04
	3	313.92	38.08	87.87
	4	738.04	50.30	93.18
	5	639.16	48.73	92.38
40-20	1	9421.16	361.40	96.16
	2	7137.31	477.32	93.31
	3	4739.77	293.21	93.81
	4	20592.78	771.38	96.25
	5	6071.36	341.64	94.37

Tabla 16.- Resultados de los tiempos de computación empleados por cada variante.

Como se puede observar de la tabla 16, con la gestión de índices se consiguen reducciones en los tiempos de computación realmente importantes, especialmente en las instancias de mayor tamaño donde este ahorro está en torno al 95%.

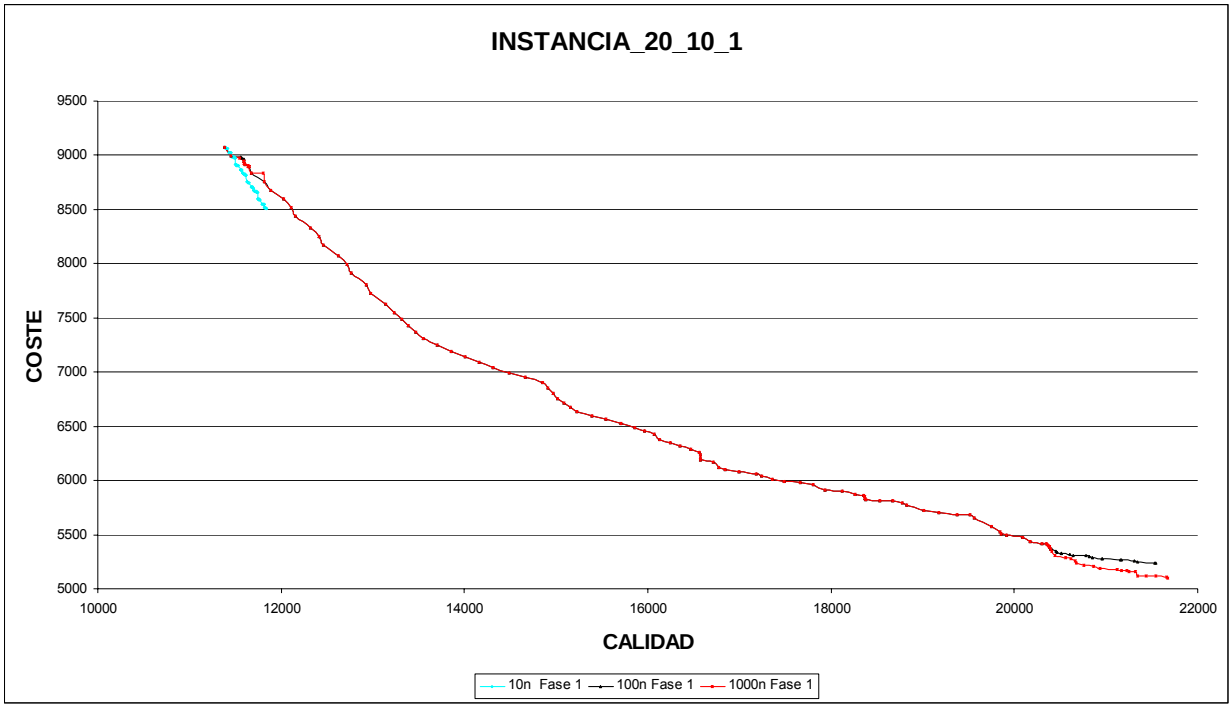
13.5.- PARÁMETRO MAX_ITER.

A continuación, se procede a ejecutar el algoritmo con diferentes valores de max_iter. Este parámetro limita el número de iteraciones que ejecuta el algoritmo sin mejora, antes de detenerse. Un valor mayor permitira explorar mayor número de soluciones dentro del vecindario antes de detenerse y, así, poder escapar de un óptimo local a costa, probablemente, de un mayor tiempo de computación.

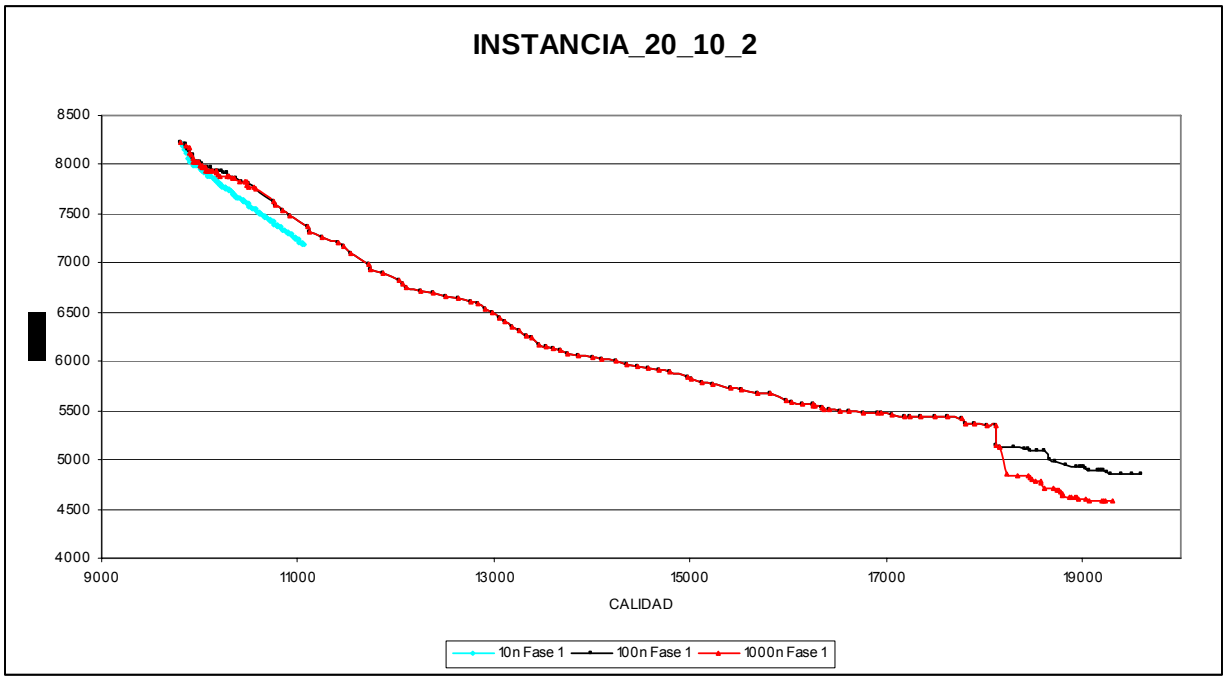
Concretamente, se va a comprobar la bondad de las soluciones (en términos de función calidad y función coste) a partir de los valores de *max_iter* 10**n*, 100**n* y 1000**n*, siendo *n* el número de iteraciones sin cambio; este análisis se va a realizar sobre un conjunto de instancias generadas previamente.

El análisis, se realiza sobre la primera fase de MOAMP que es totalmente determinista, pues la segunda fase, tiene componente aleatoria y al ser la tercera fase, una intensificación de la segunda, le dota también del carácter aleatorio.

Instancia_20_10_1

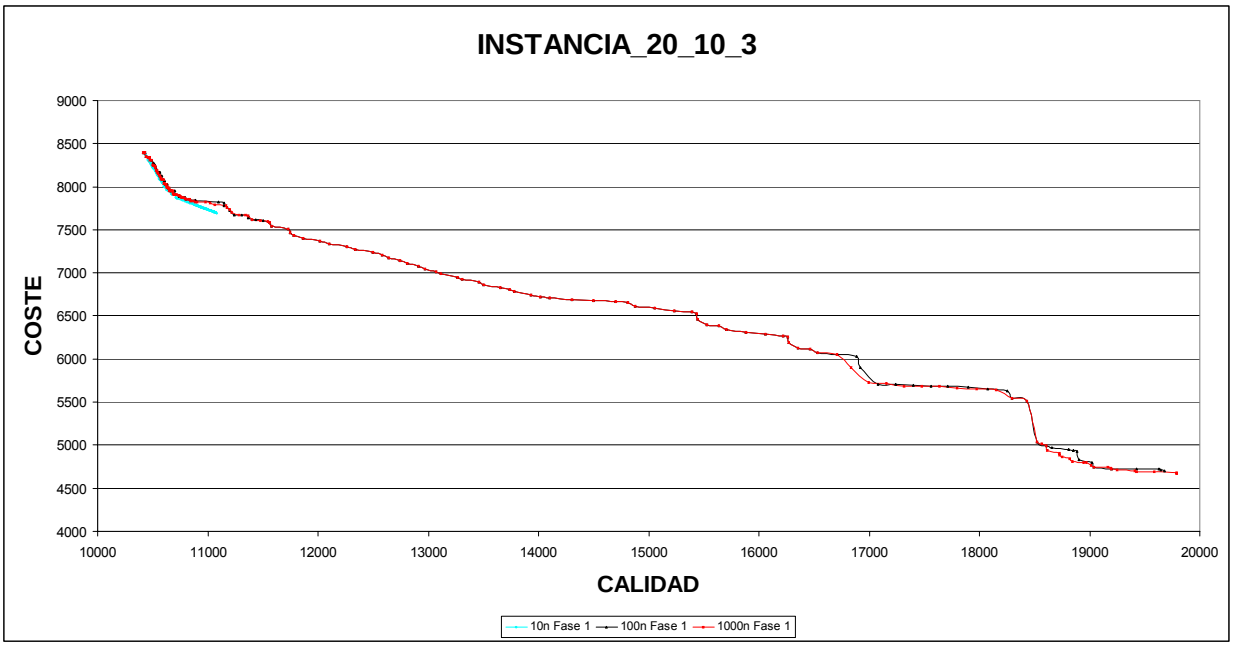


Instancia_20_10_2

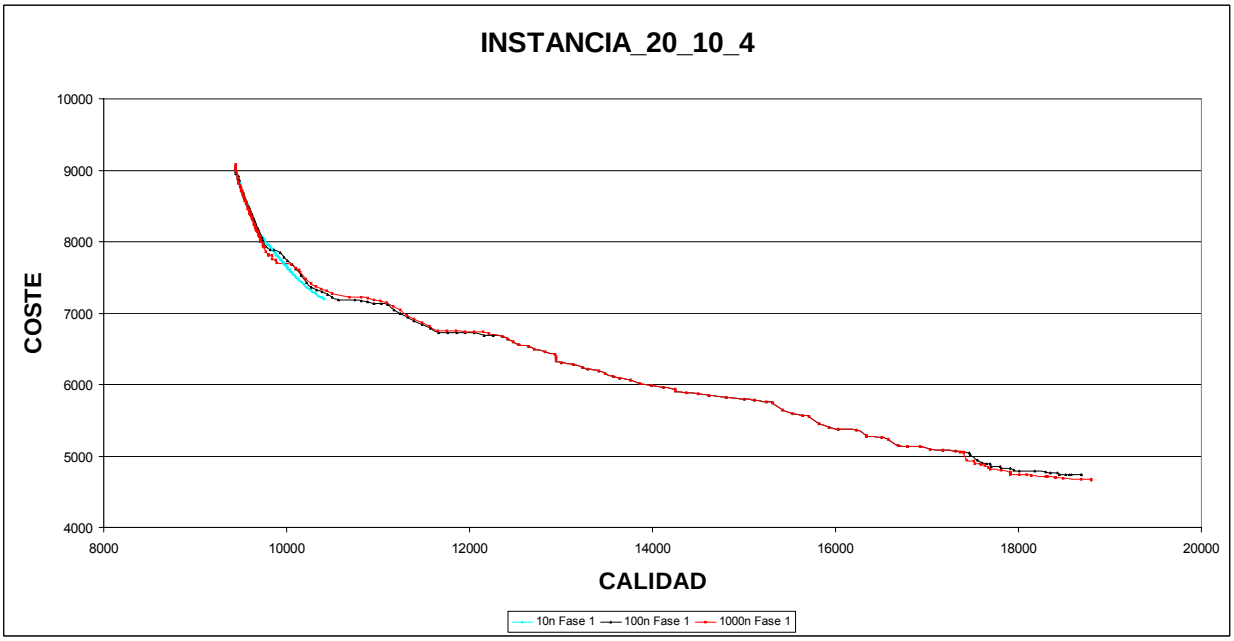


En estas dos instancias, se observa el paro del algoritmo cuando *max_iter* tiene por valor 10n, por lo que no es capaz de abandonar un óptimo local de escaso valor en cuanto a funciones objetivo.

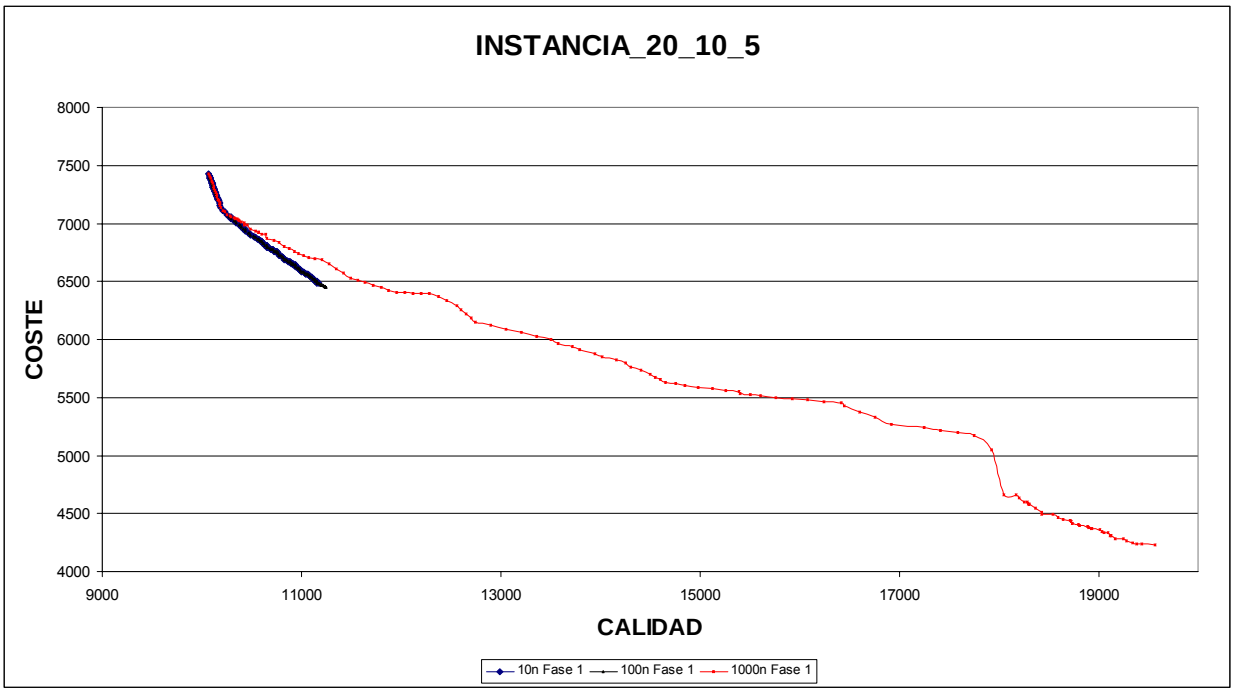
Instancia_20_10_3



Instancia_20_10_4

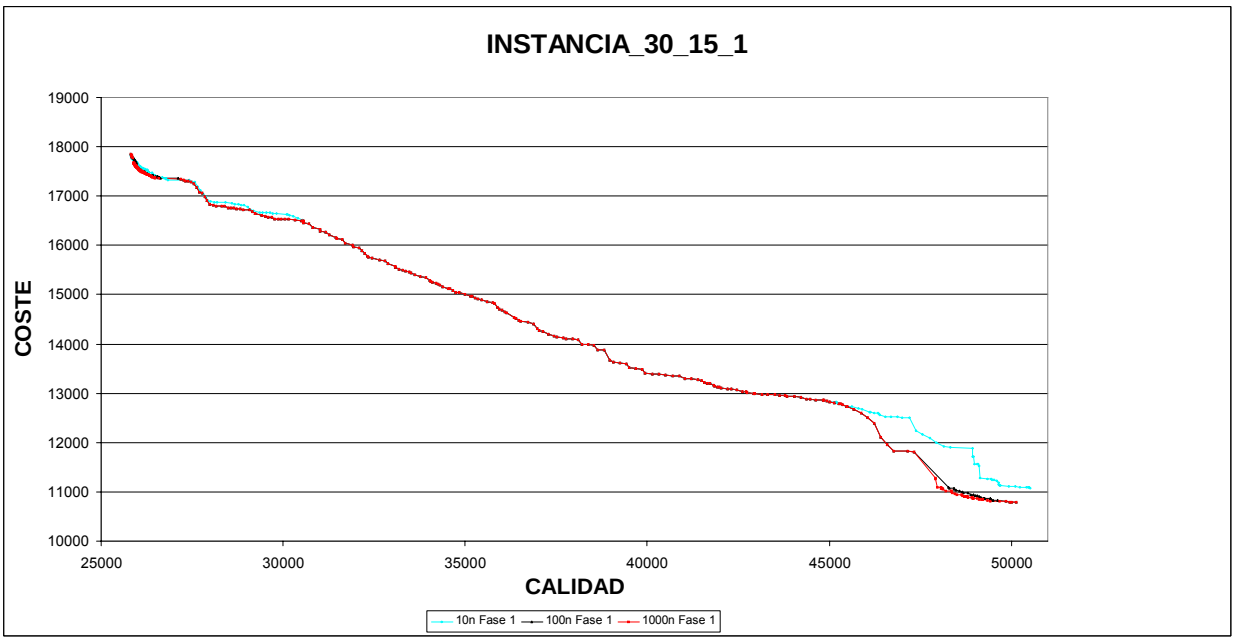


Instancia_20_10_5

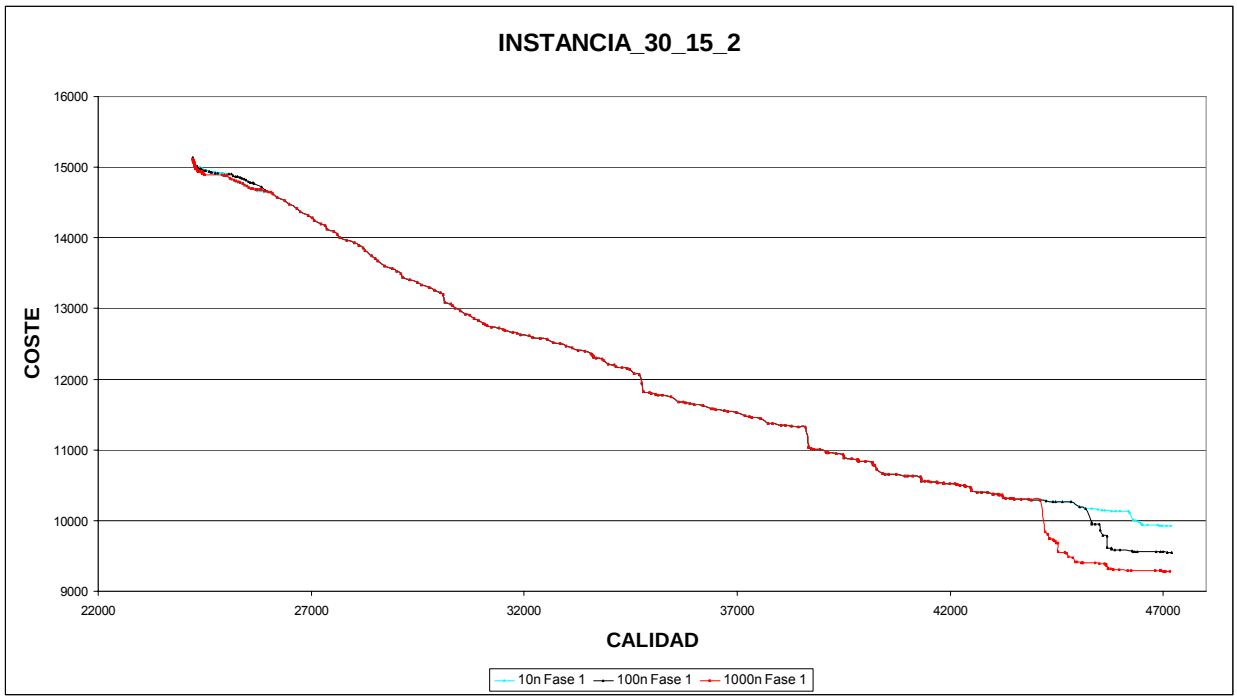


La misma consideración se puede realizar en estos casos; la parada del algoritmo prematuramente (para valores de max_iter de 10n y 100n), de lo que cabe deducir que no ha sido capaz de salir de un óptimo local de poco valor.

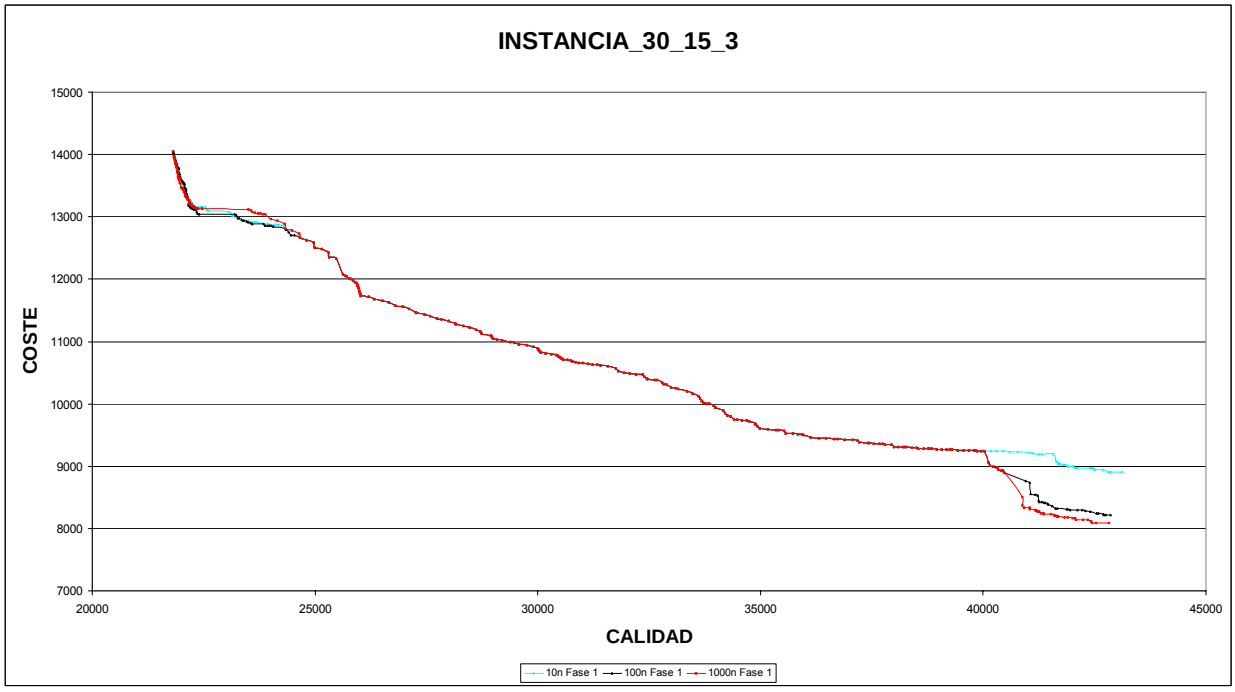
Instancia_30_15_1



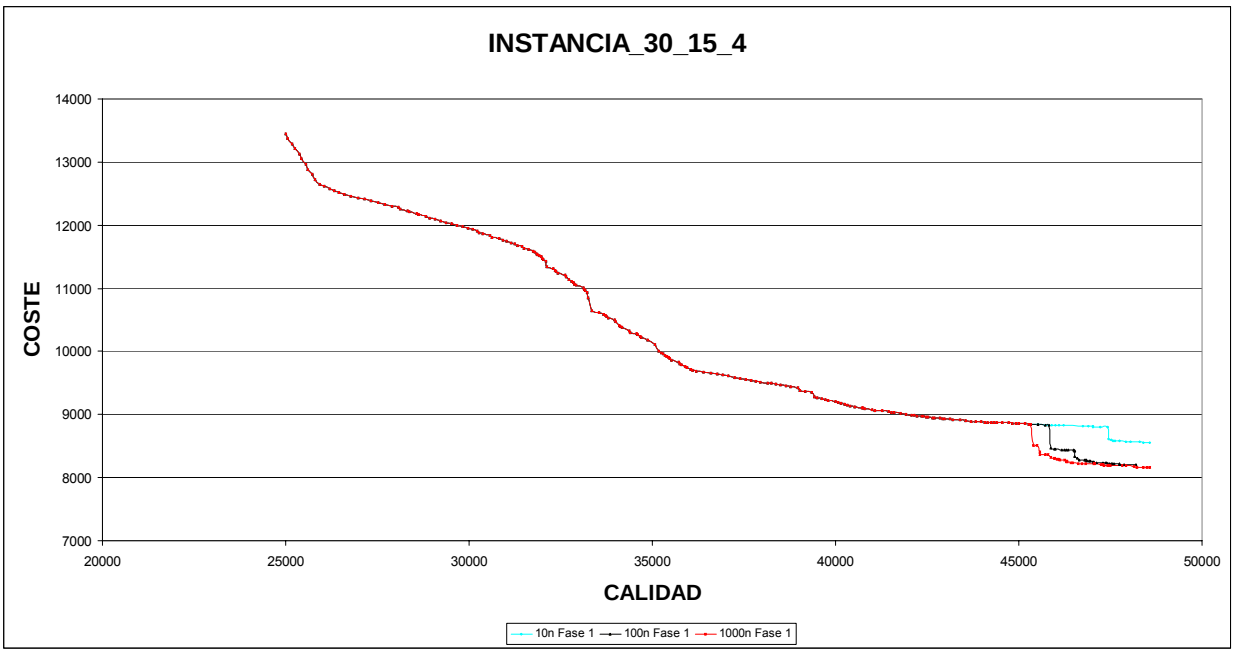
Instancia_30_15_2



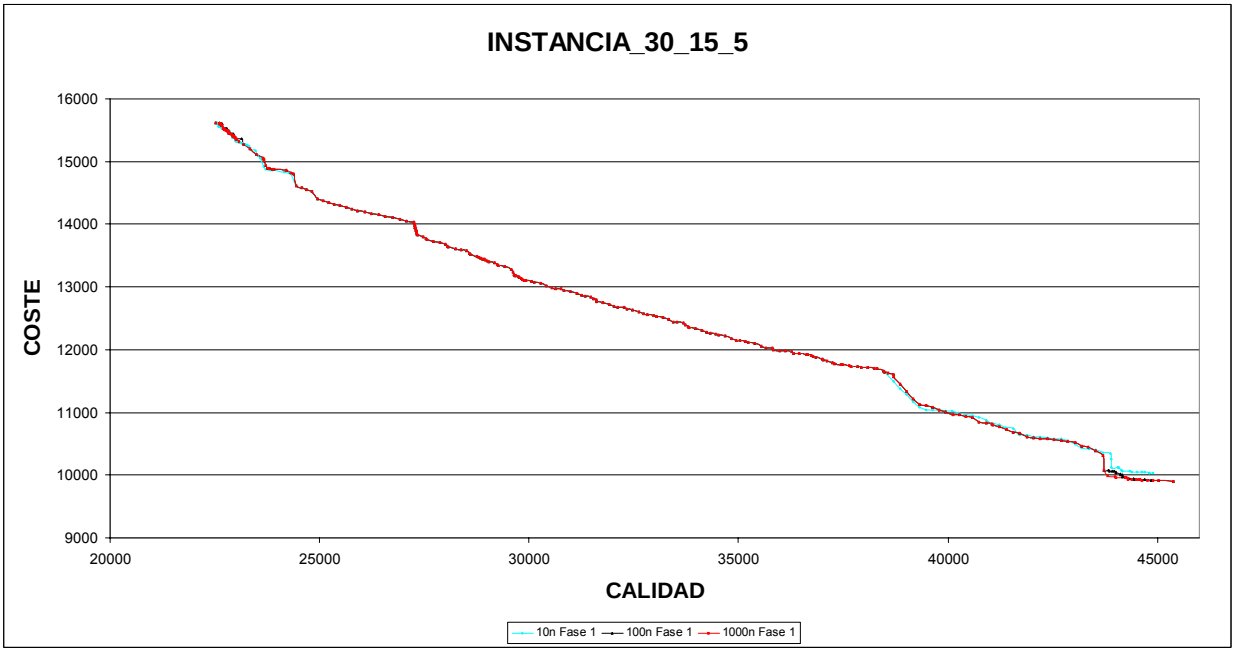
Instancia_30_15_3



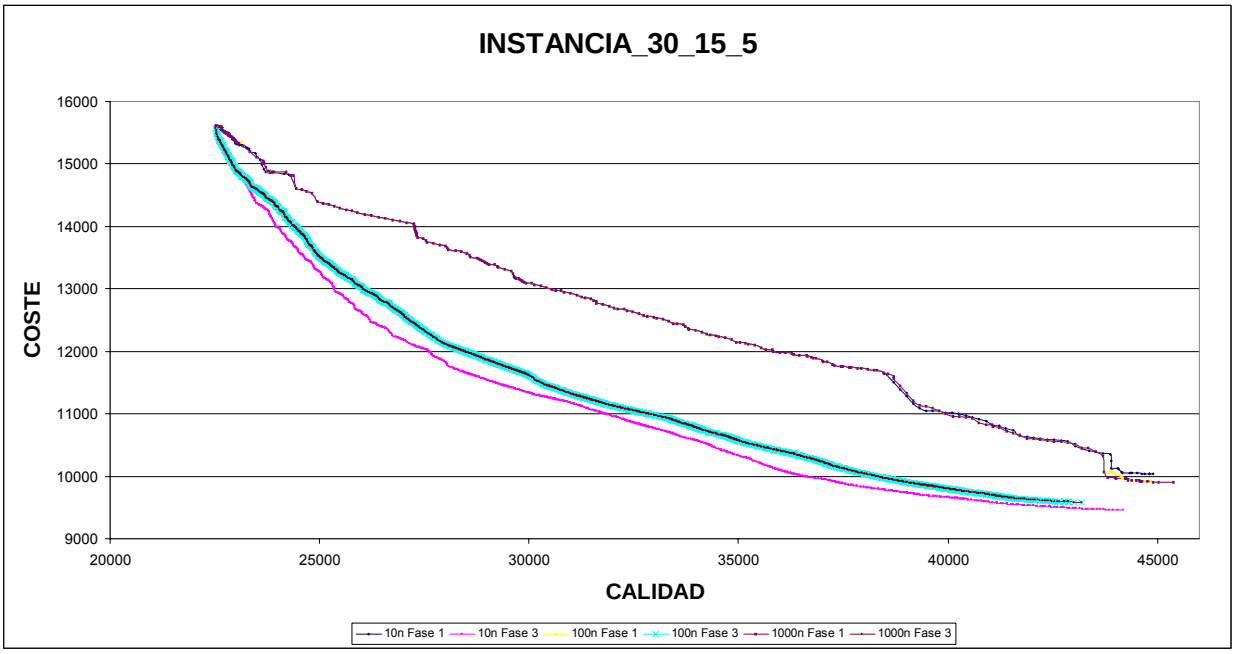
Instancia_30_15_4



Instancia_30_15_5



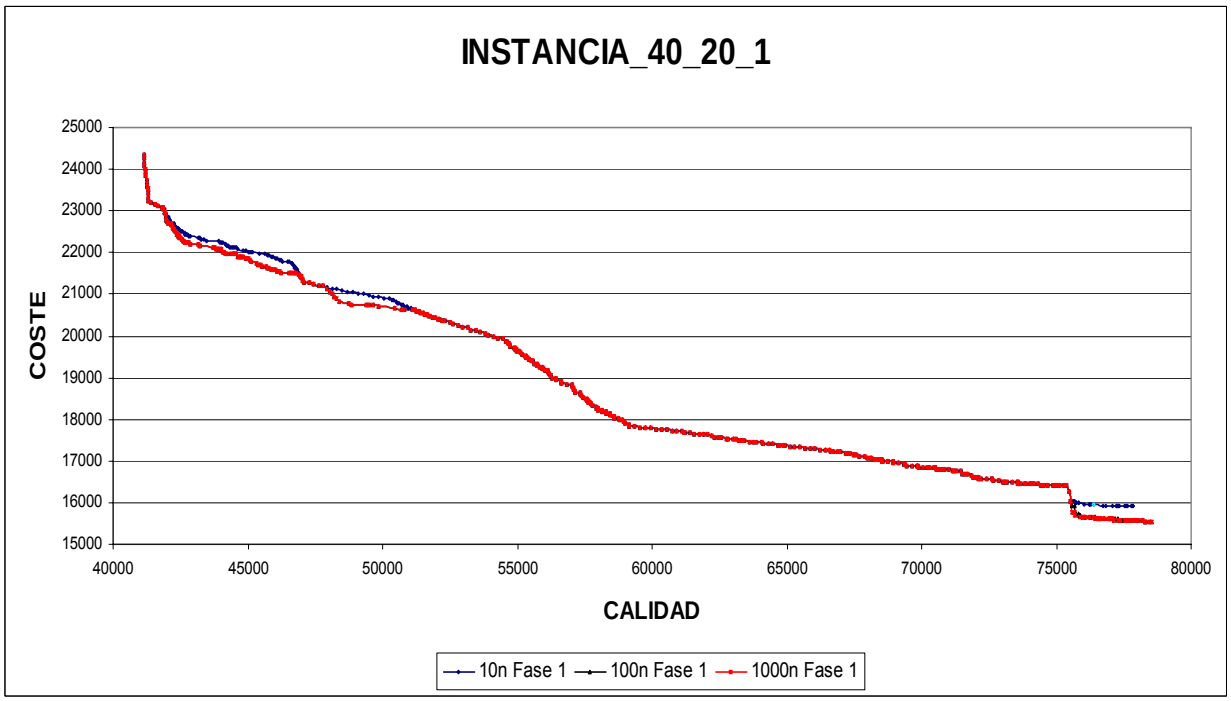
Instancia_30_15_5



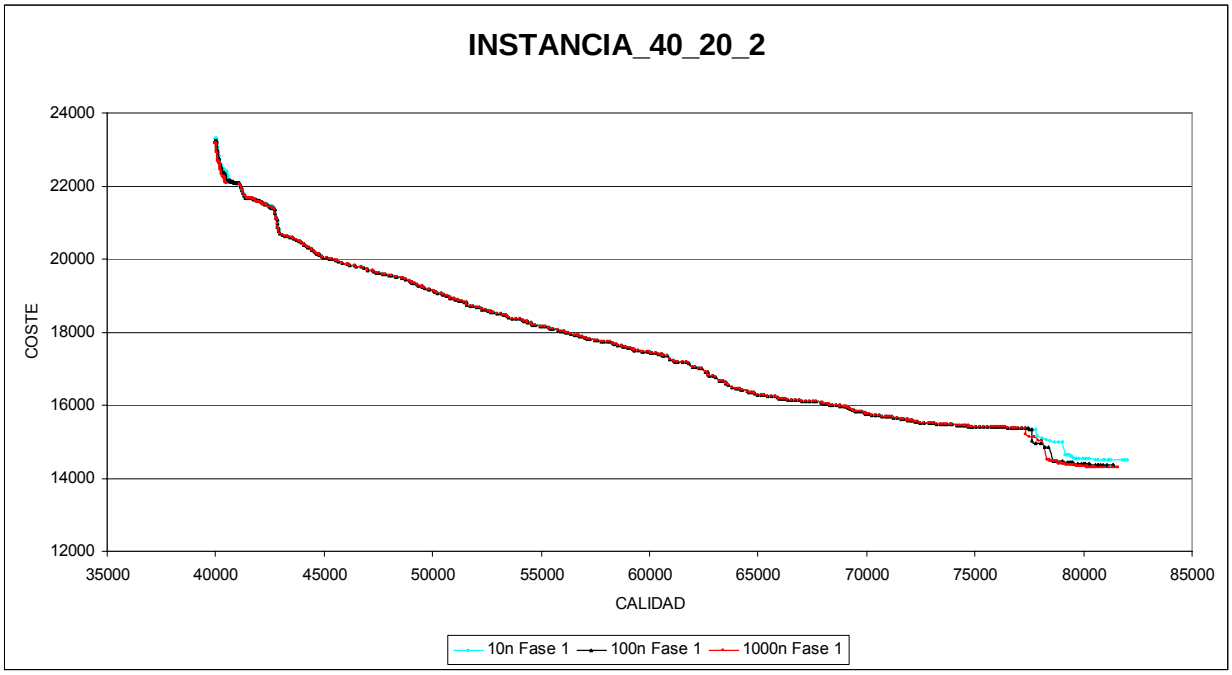
En este gráfico se representa, a modo ilustrativo, las soluciones de las fases 1 y 3 y se puede apreciar la mejoría en la aproximación a la curva de eficiencia, de la fase 3, con valor de *max_iter* = 1000n frente a otros valores de *max_iter*.

Las soluciones pueden variar en caso de repetir la ejecución del algoritmo, pues la fase 2 no es determinista, sino que tiene una componente aleatoria, aún así se considera que globalmente mejora la calidad de las soluciones.

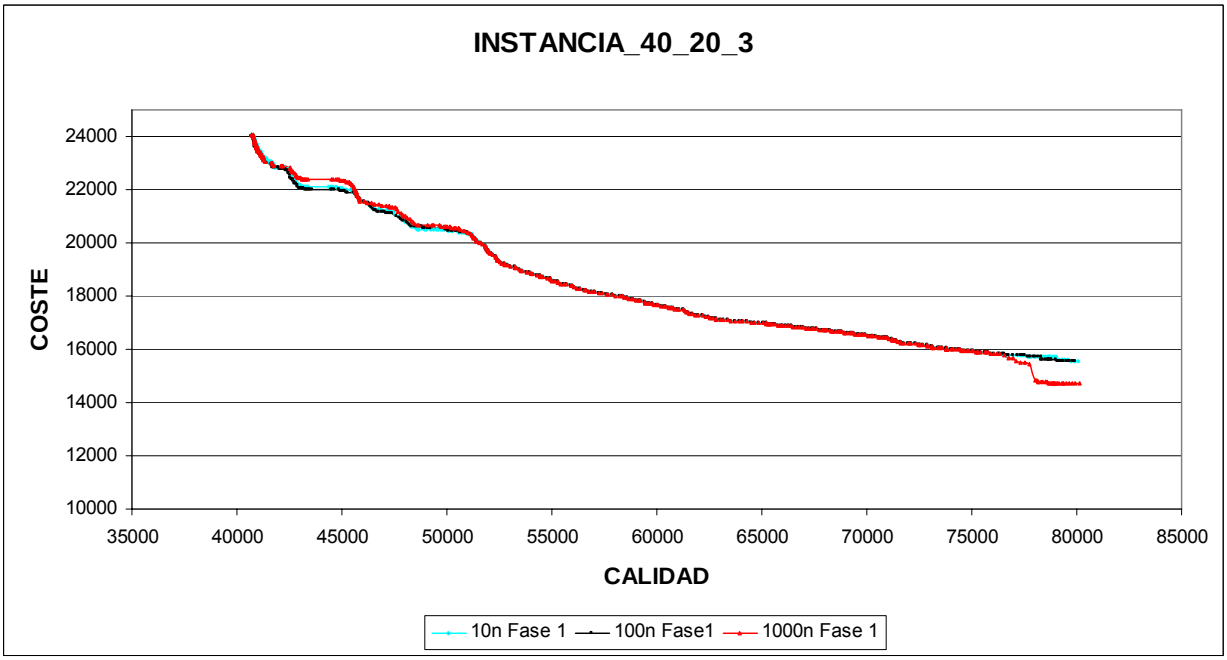
Instancia_40_20_1



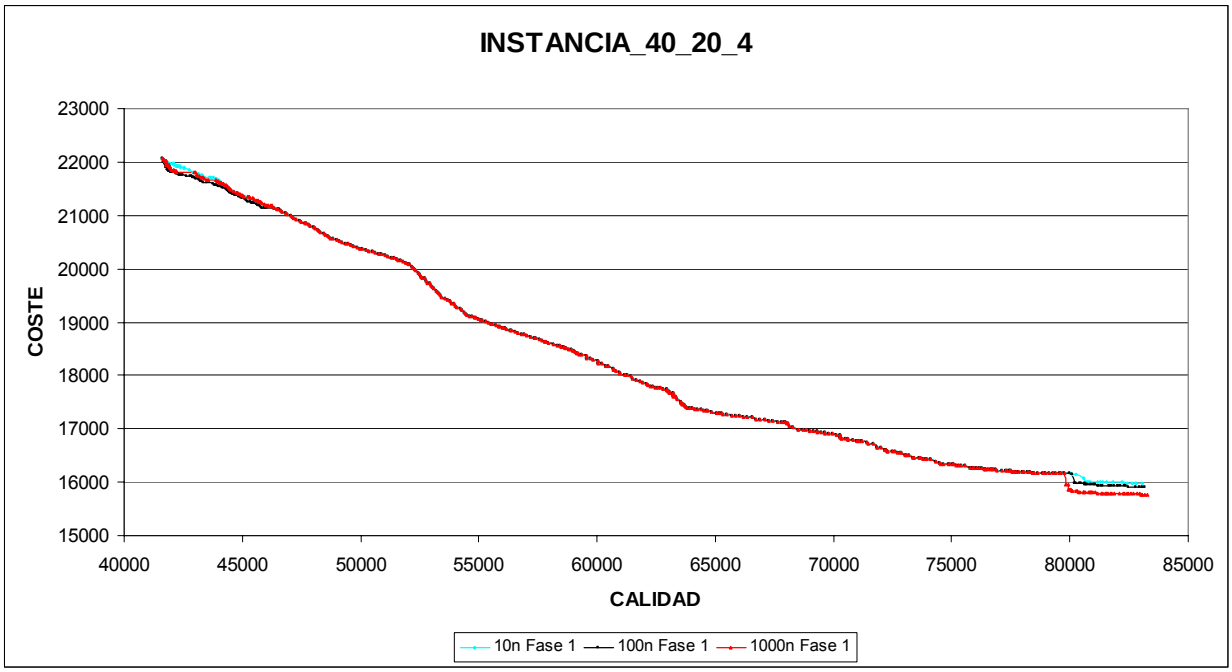
Instancia_40_20_2



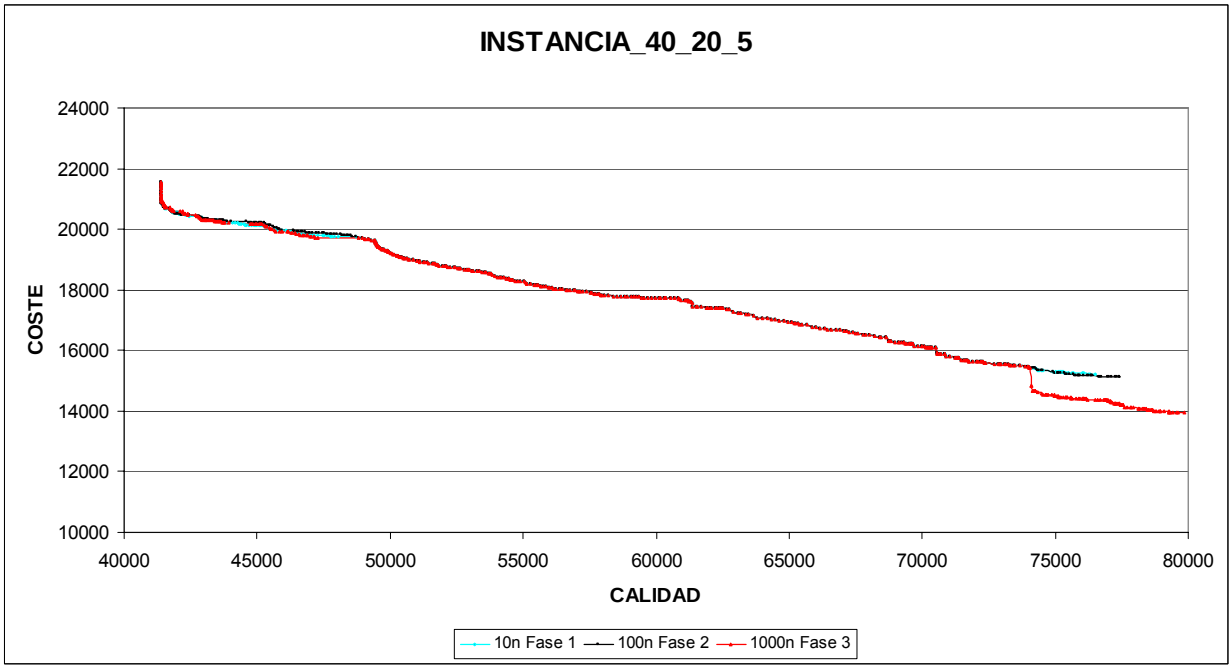
Instancia_40_20_3



Instancia_40_20_4

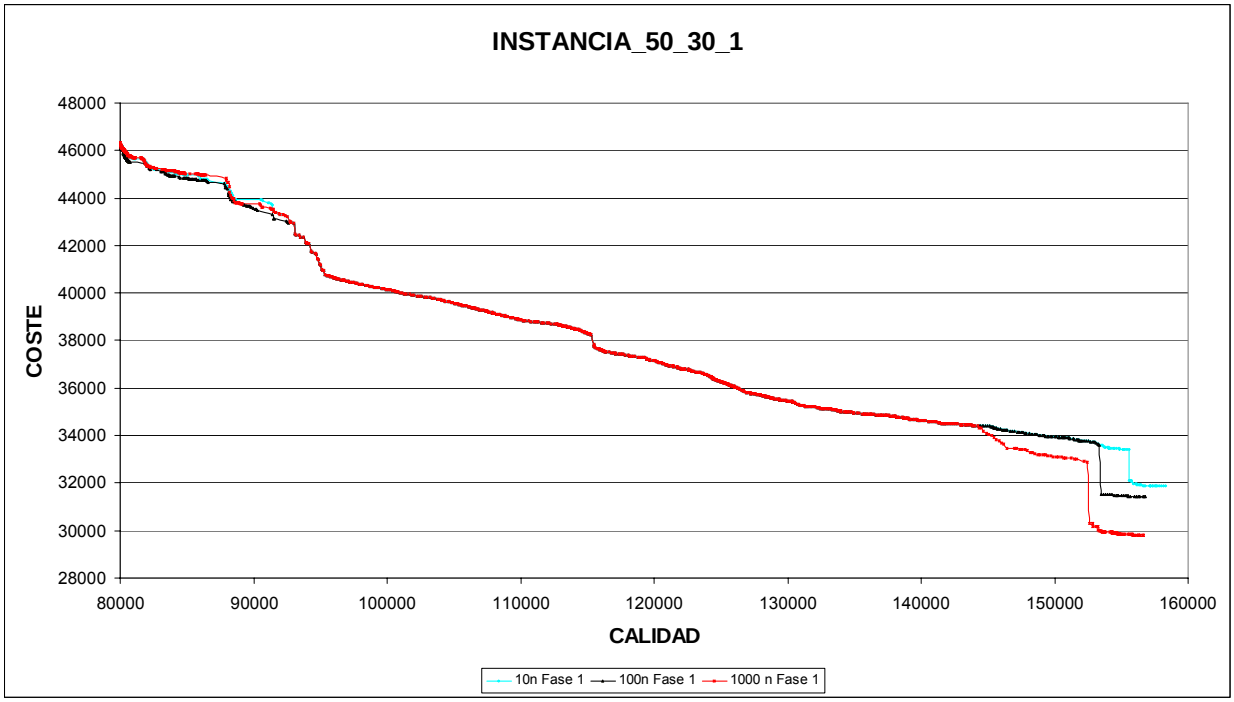


Instancia_40_20_5

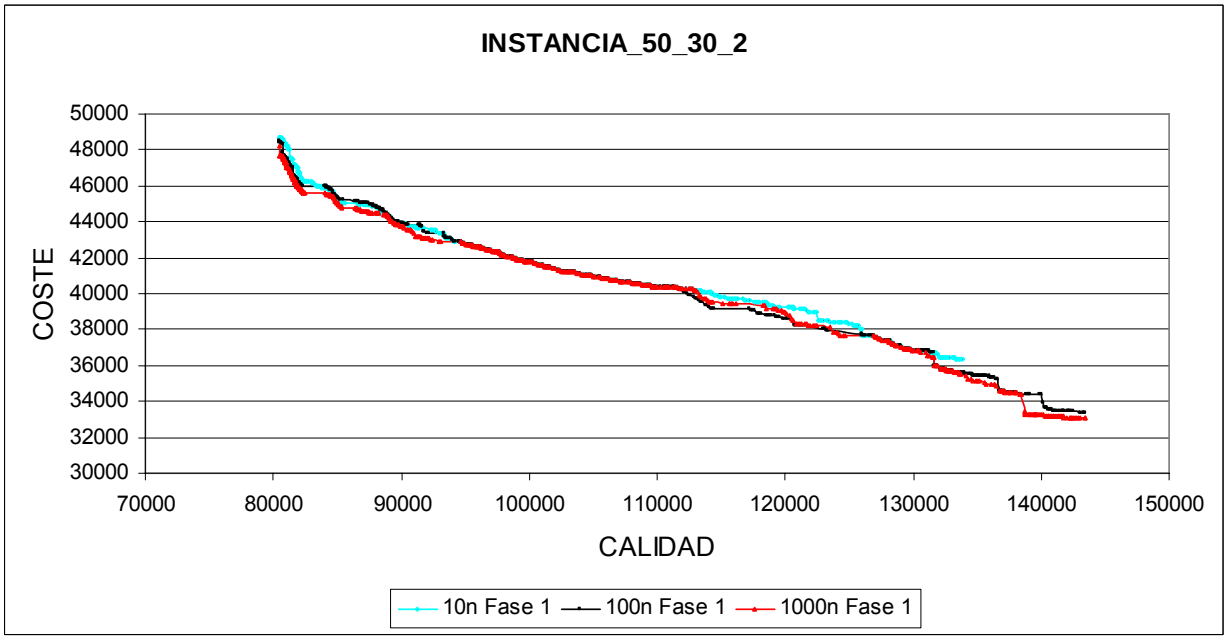


En este caso particular, a diferencia de los anteriores, se puede observar que consigue, para un valor de *max_iter* más elevado, encontrar una solución con un valor de calidad apreciablemente peor que para los otros valores de *max_iter*, a costa de disminuir apreciablemente el coste.

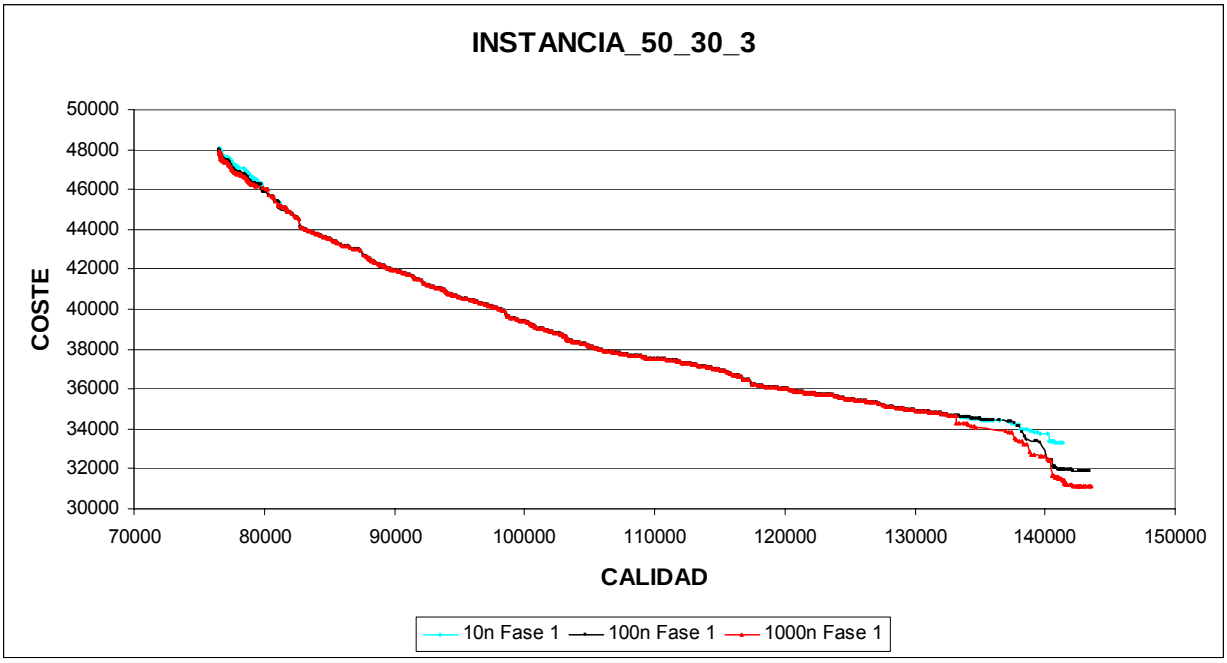
Instancia_50_30_1



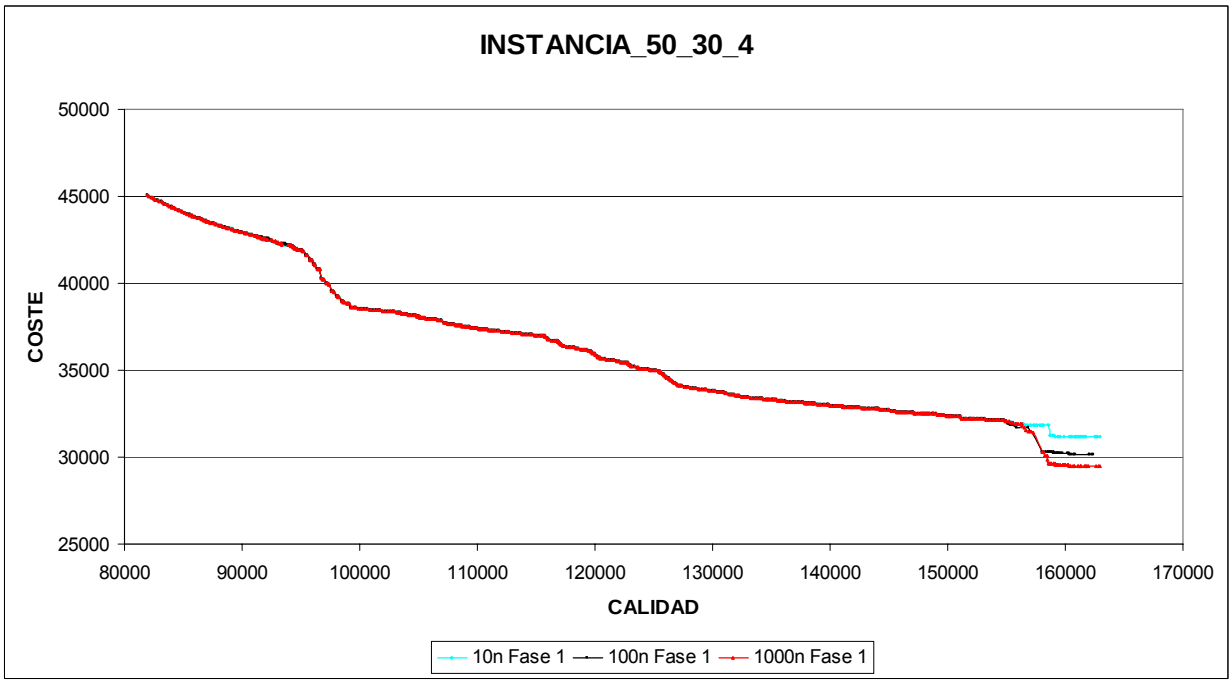
Instancia_50_30_2



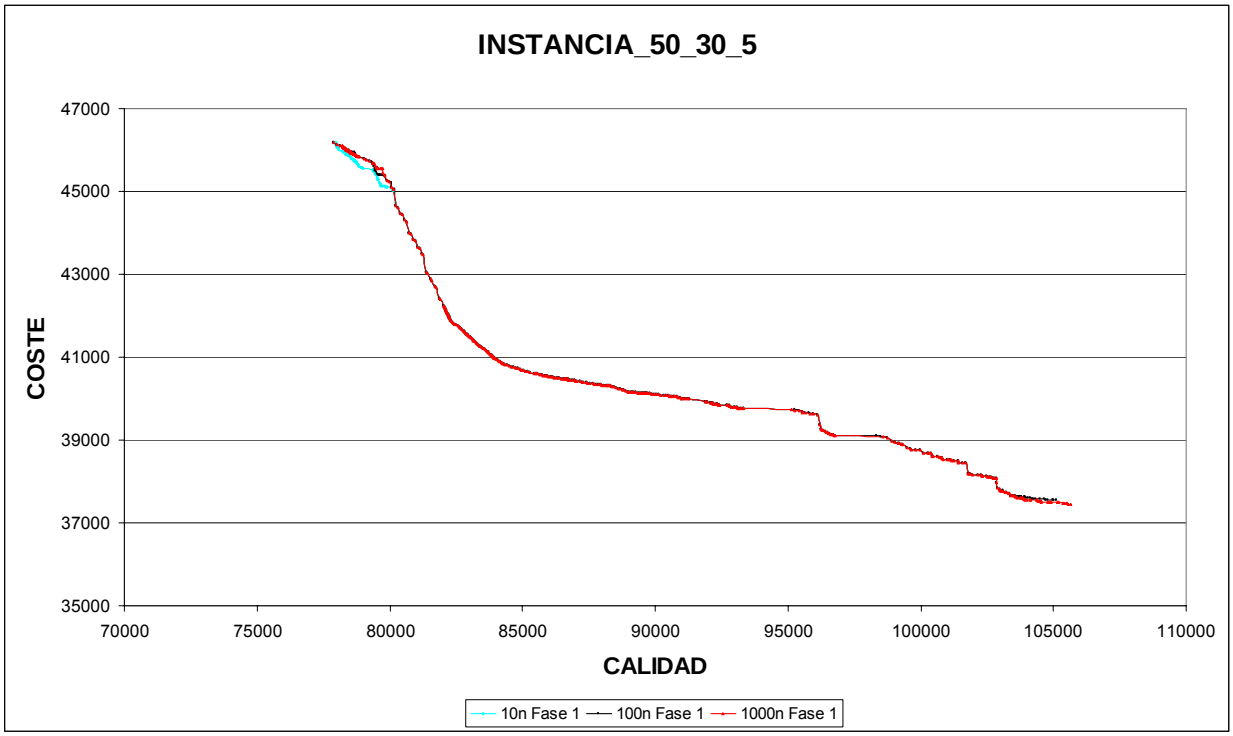
Instancia_50_30_3



Instancia_50_30_4



Instancia_50_30_5



Del análisis de las figuras anteriores, puede observarse que con valor de *max_iter* = 1000 se consiguen mejores soluciones que con valores inferiores; se puede ver también que, en las instancias con tamaños más pequeños, incluso se queda detenido en óptimos locales de poco valor (respecto a ambas funciones objetivo).

Según se acerca a la buena solución de calidad, se observa que converge con los tres valores; ahora bien, cuando obtiene buenas soluciones en la función coste, es cuando se observa la obtención de mejores soluciones, según *max_iter* aumenta por ser capaz de escapara de dichos óptimos locales.

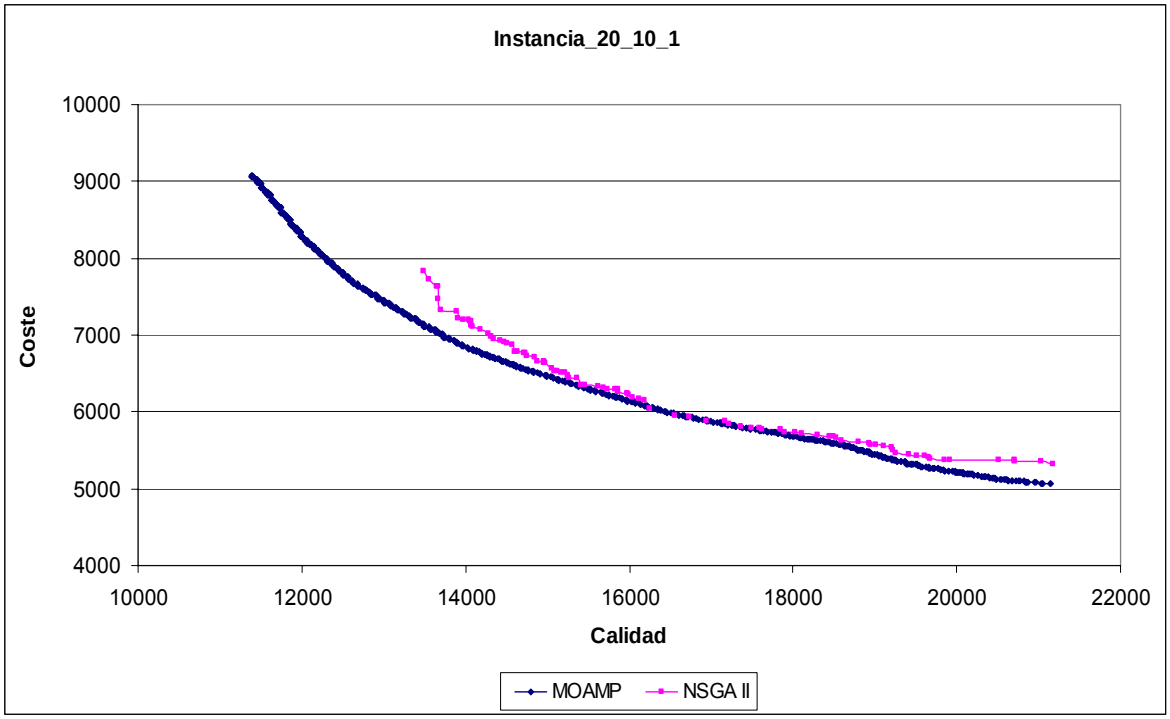
13.6.- COMPARACIÓN CON NSGA-II.

En este apartado, se compara el conjunto de puntos no dominado (tercera fase) obtenido con MOAMP, con el conjunto de puntos no dominados obtenidos con NSGA II. La comparación se realiza sobre las instancias 20-10 hasta 40-20; no se representa la instancia 50-30 por resultar muy elevado el tiempo de computación.

Respecto a los resultados obtenidos, en los casos en que NSGA II aporte soluciones que dominen a MOAMP, se indica el número de soluciones de NSGA II dominantes así como su porcentaje respecto a MOAMP.

Instancia 20-10-1.dat

MOAMP = 743 soluciones
NSGA II = 93 soluciones

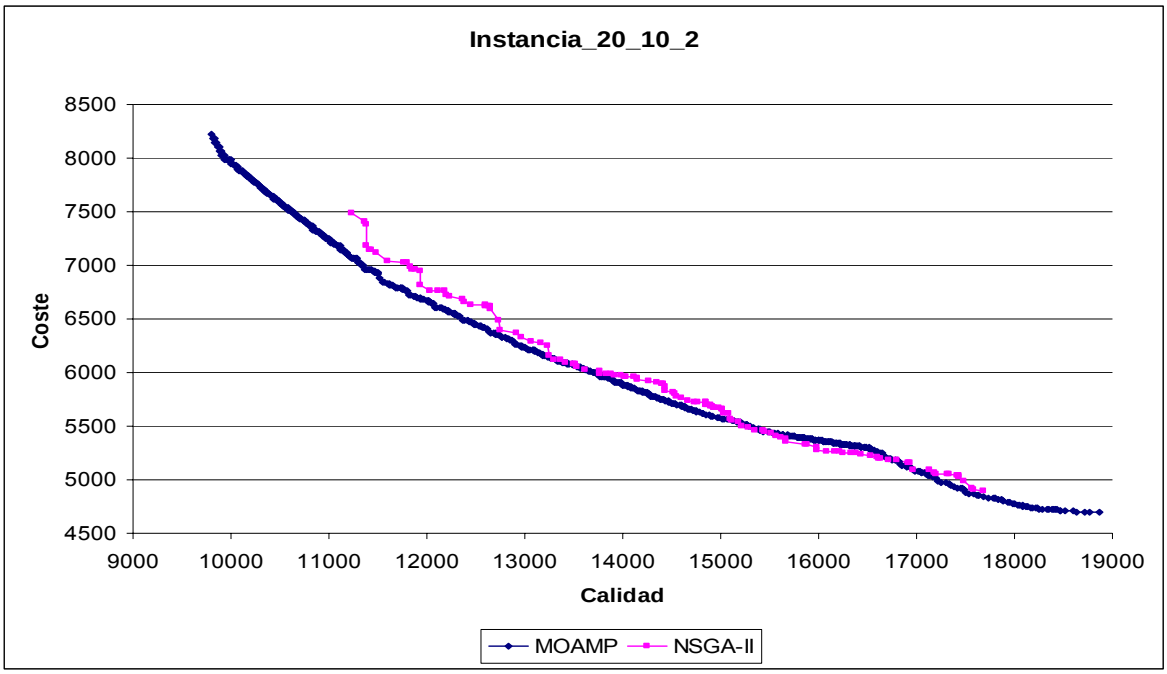


Soluciones NSGA II dominadas por MOAMP:	90	% de soluciones NSGA dominadas:	96.8 %
Soluciones MOAMP dominadas por NSGA II:	11	% de soluciones MOAMP dominadas:	1.5 %

Instancia 20-10-2.dat

MOAMP = 608 soluciones

NSGA II = 126 soluciones

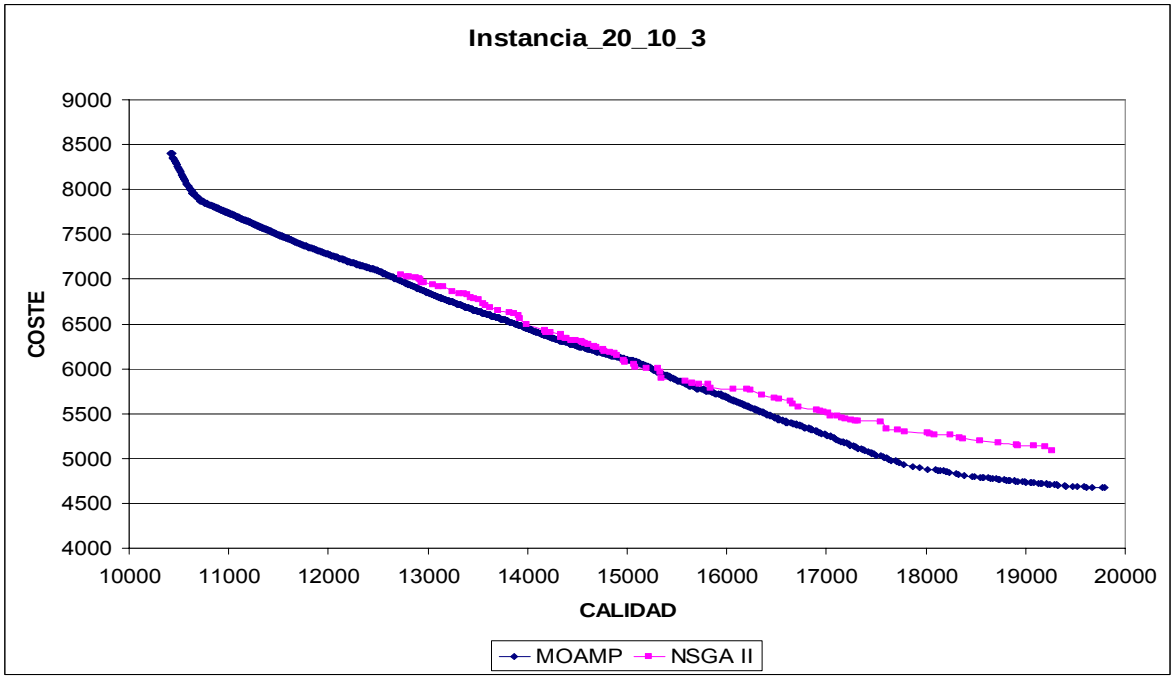


Soluciones NSGA II dominadas por MOAMP:	100	% de soluciones NSGA dominadas:	79.4 %
Soluciones MOAMP dominadas por NSGA II:	119	% de soluciones MOAMP dominadas:	19.5 %

Instancia 20-10-3.dat

MOAMP = 1034 soluciones

NSGA II = 98 soluciones

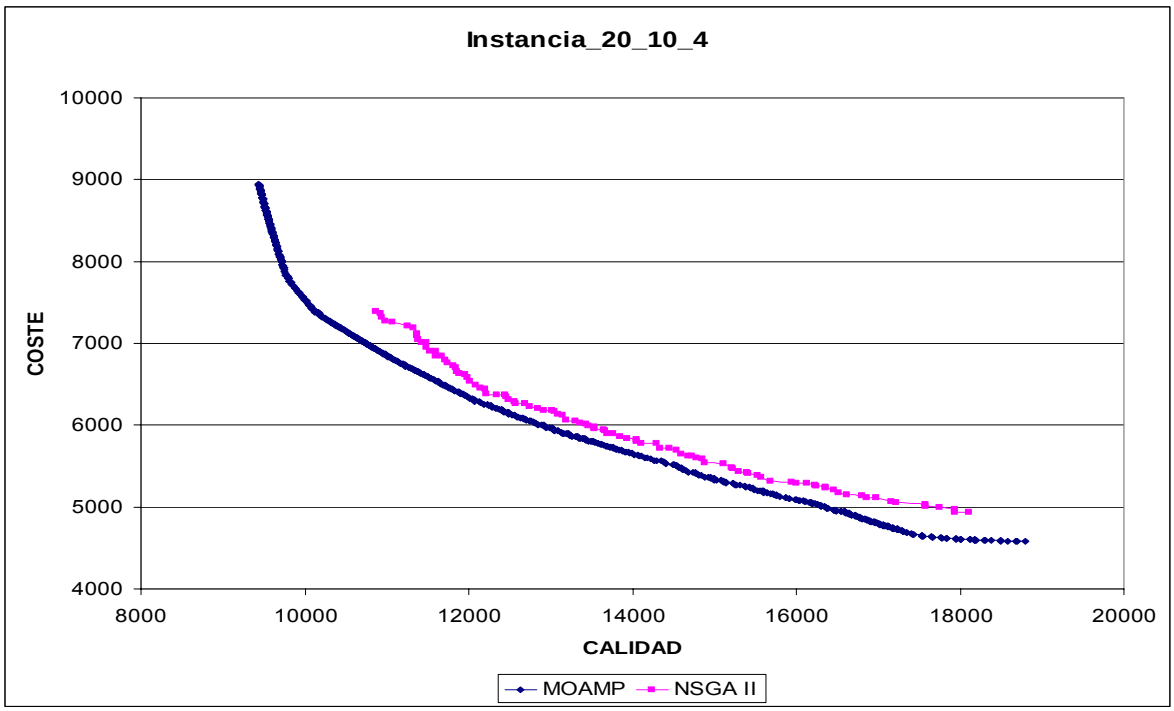


Soluciones NSGA II dominadas por MOAMP:	92	% de soluciones NSGA dominadas:	93.9 %
Soluciones MOAMP dominadas por NSGA II:	62	% de soluciones MOAMP dominadas:	6 %

Instancia 20-10-4.dat

MOAMP = 664 soluciones

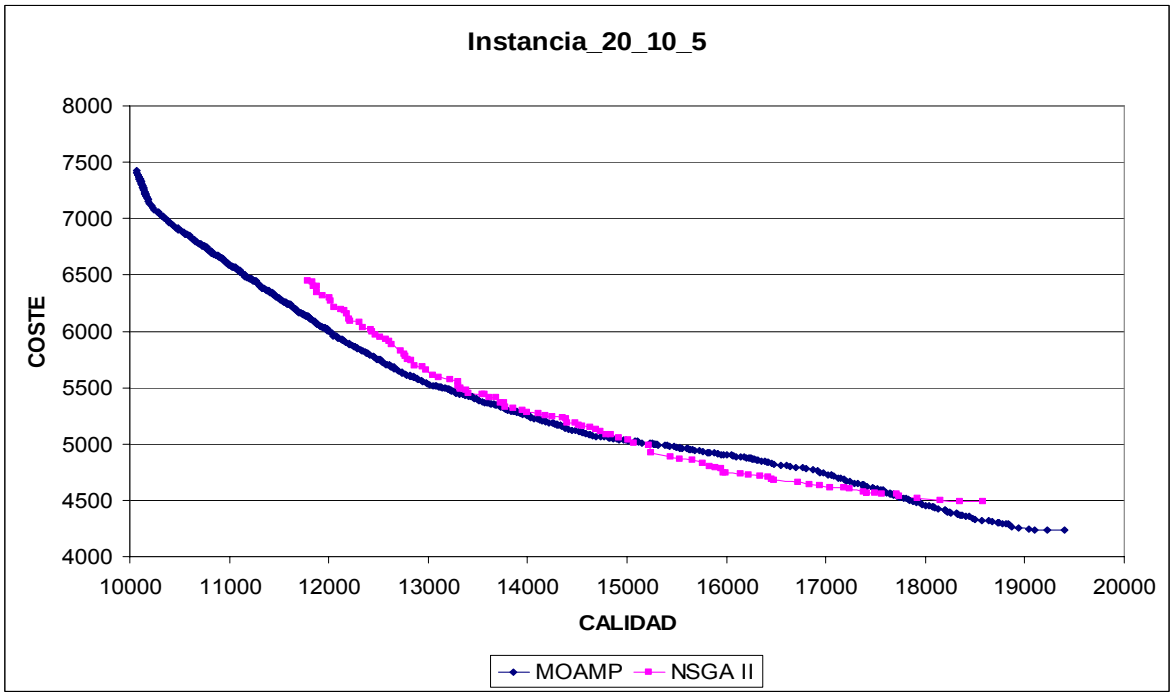
NSGA II = 100 soluciones



Instancia 20-10-5.dat

MOAMP = 581 soluciones

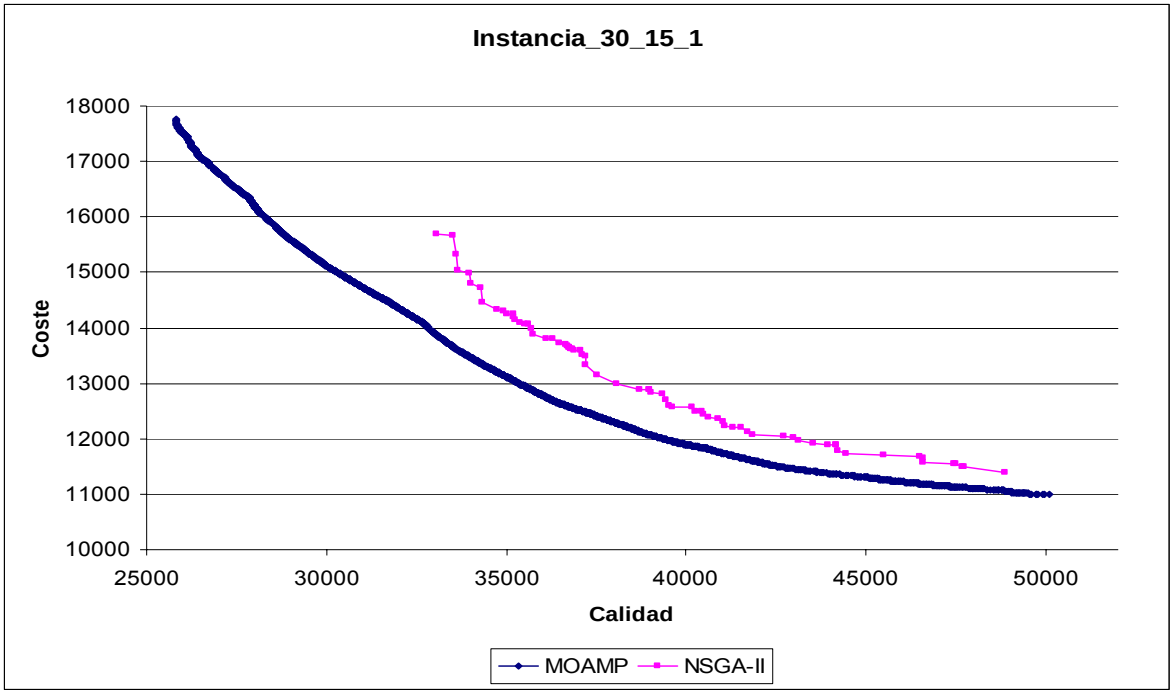
NSGA II = 100 soluciones



Soluciones NSGA II dominadas por MOAMP:	72	% de soluciones NSGA dominadas:	72 %
Soluciones MOAMP dominadas por NSGA II:	86	% de soluciones MOAMP dominadas:	14.8 %

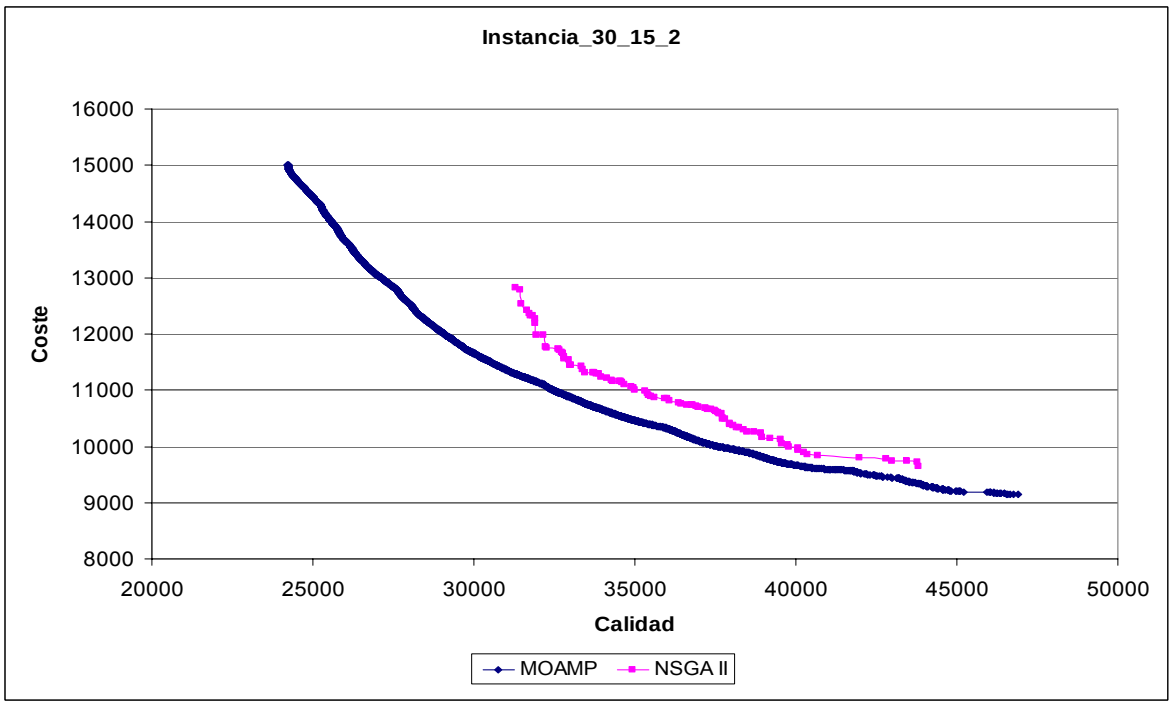
Instancia 30-15-1.dat

MOAMP = 2170 soluciones
NSGA II = 74 soluciones



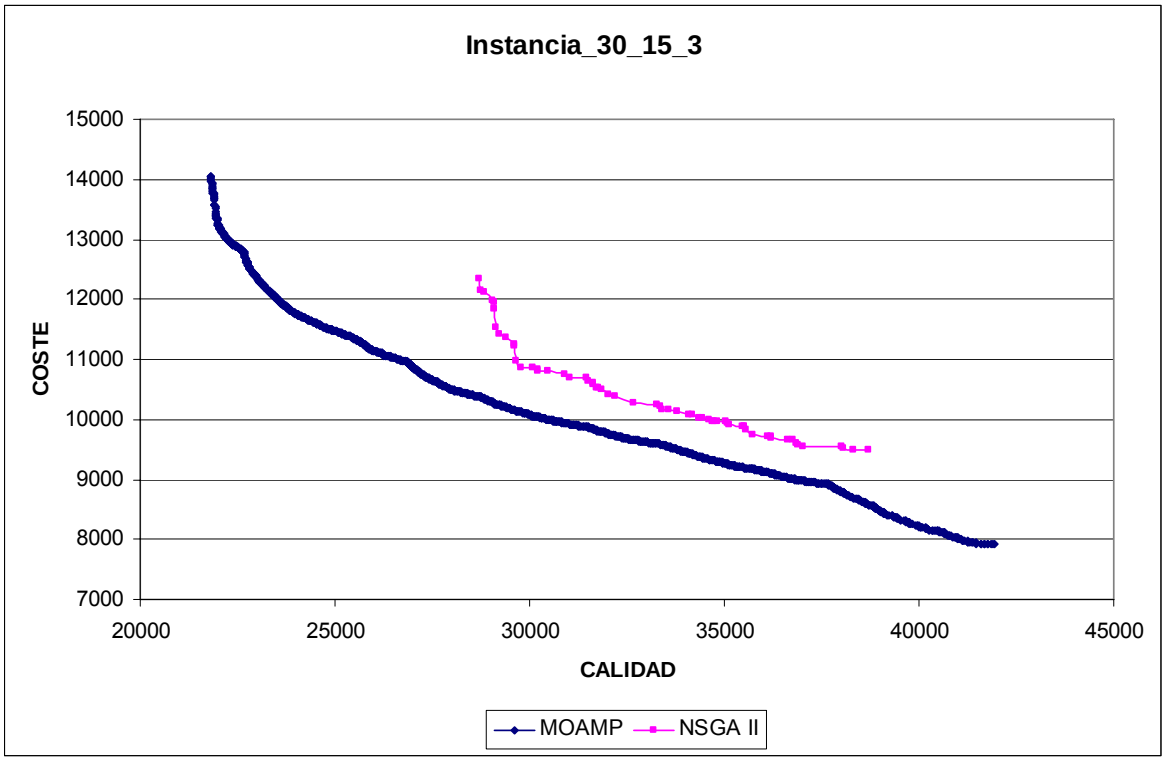
Instancia 30-15-2.dat

MOAMP = 2975 soluciones
NSGA II = 94 soluciones



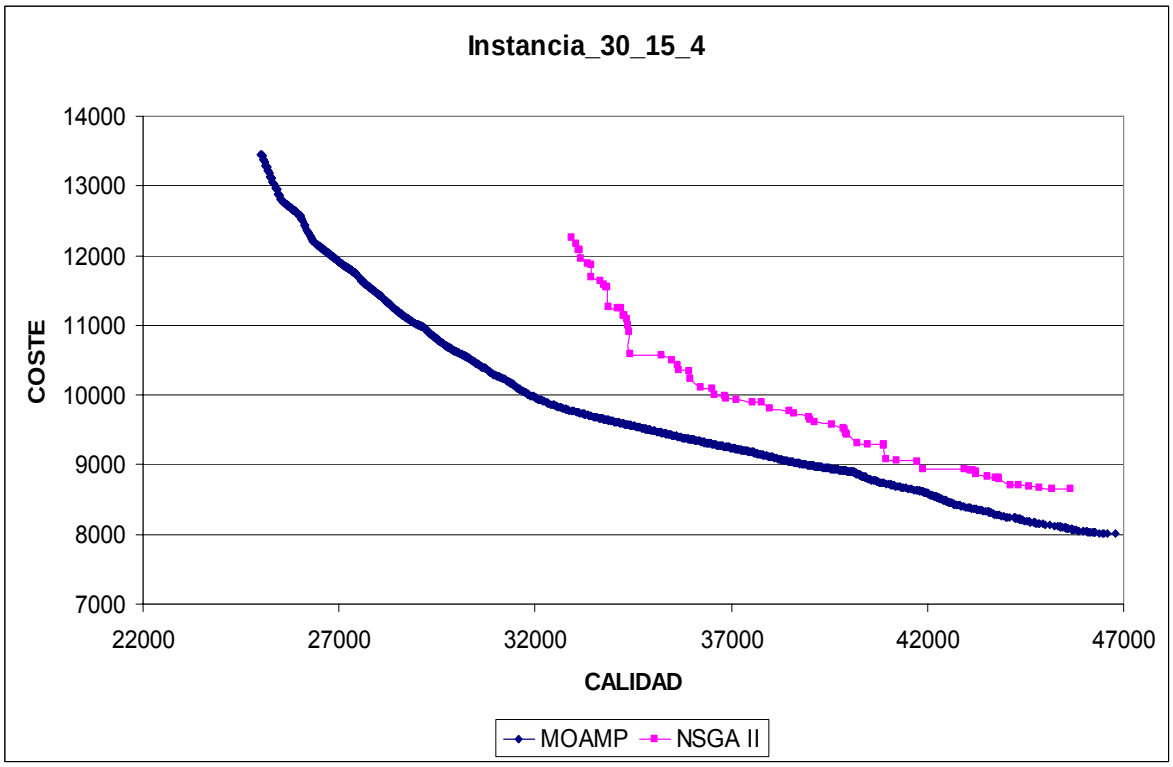
Instancia 30-15-3.dat

MOAMP = 1459 soluciones
NSGA II = 62 soluciones



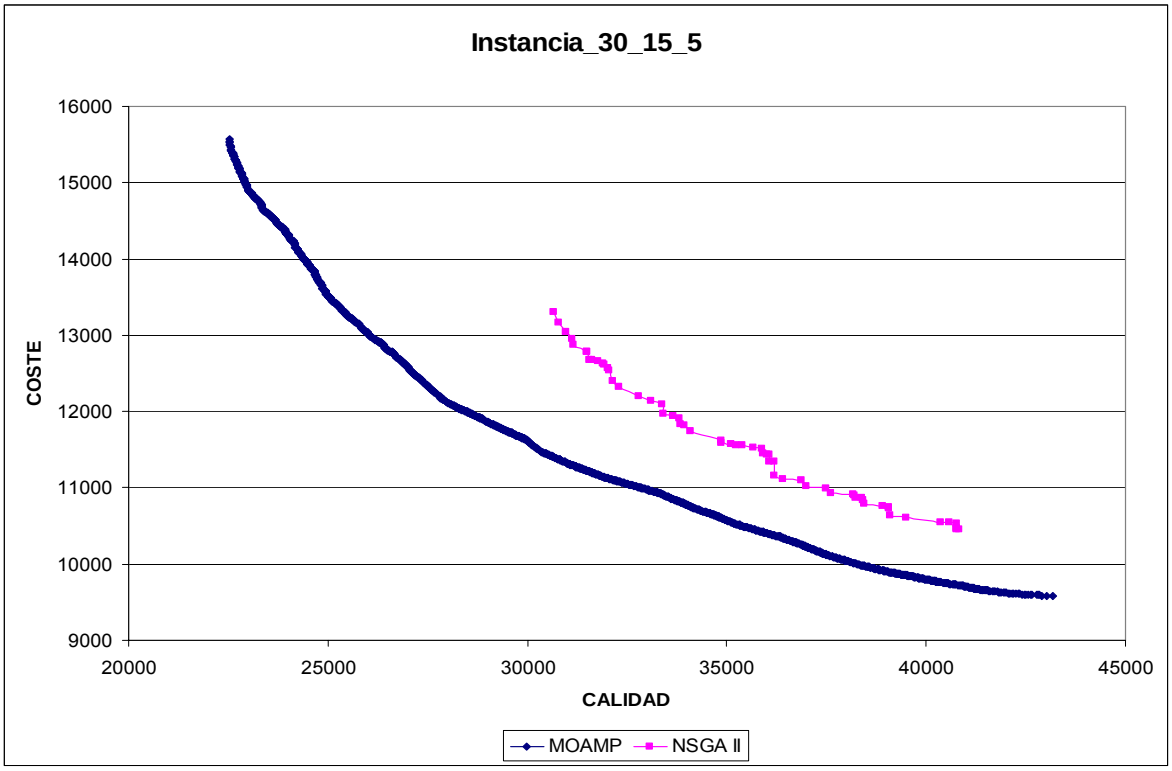
Instancia 30-15-4.dat

MOAMP = 1345 soluciones
NSGA II = 69 soluciones



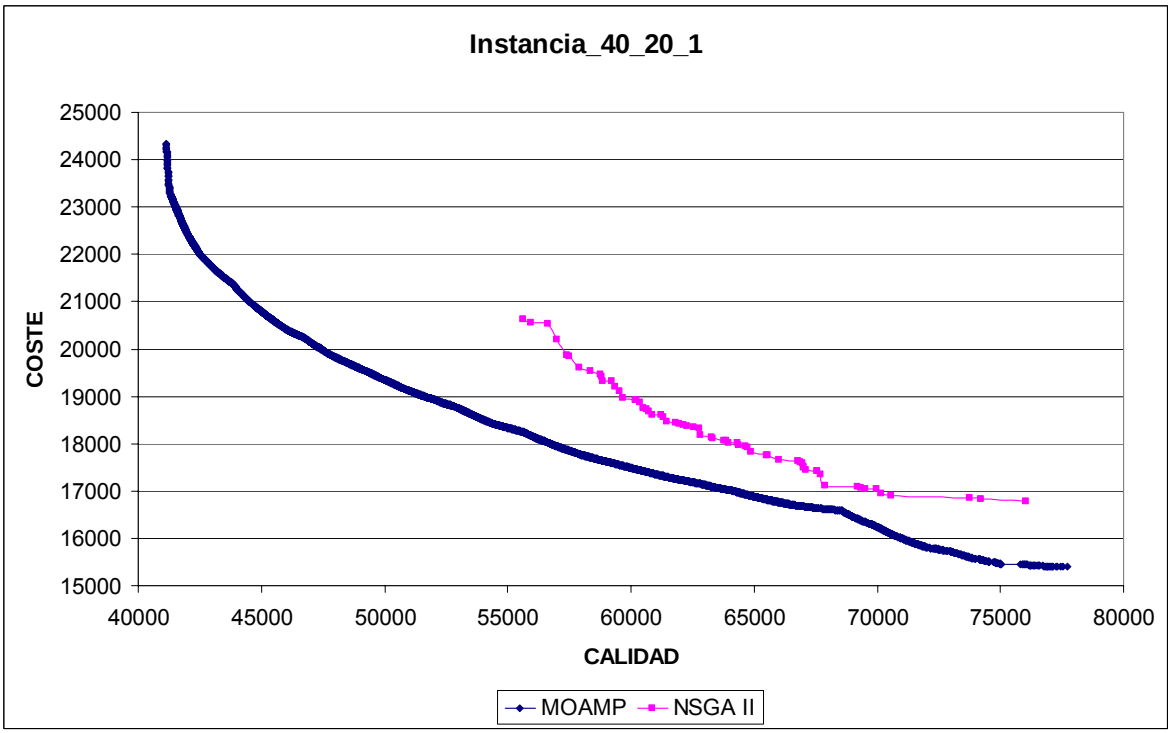
Instancia 30-15-5.dat

MOAMP = 1499 soluciones
NSGA II = 60 soluciones



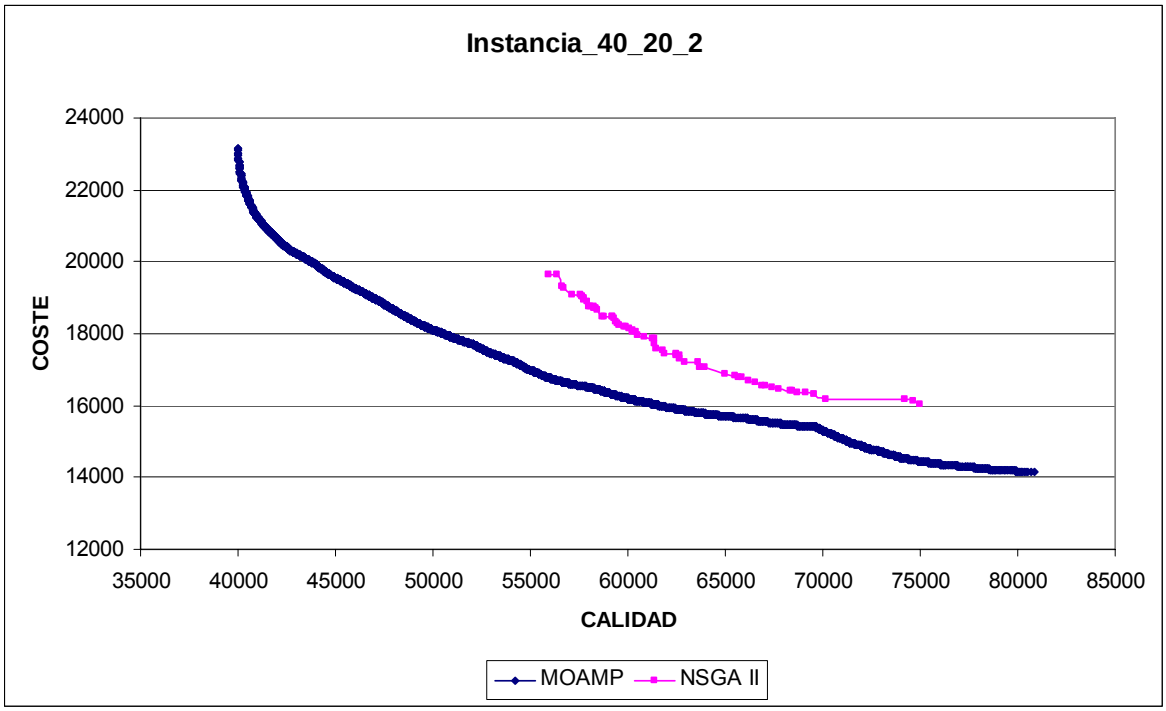
Instancia 40-20-1.dat

MOAMP = 4327 soluciones
NSGA II = 64 soluciones



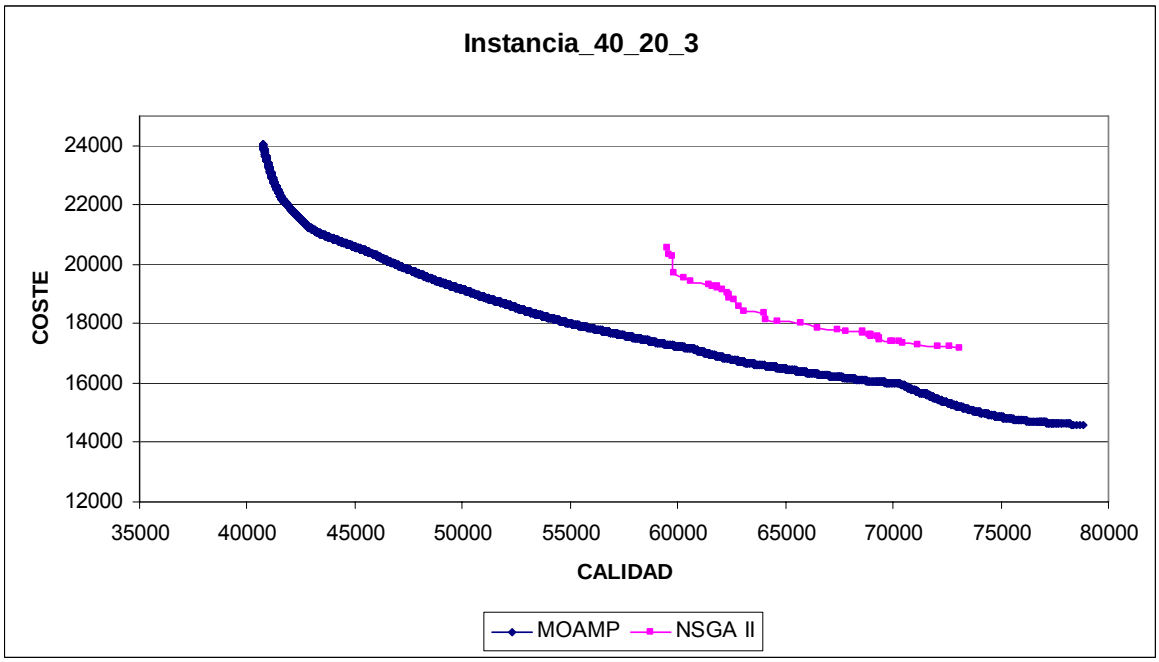
Instancia 40-20-2.dat

MOAMP = 4437 soluciones
NSGA II = 68 soluciones



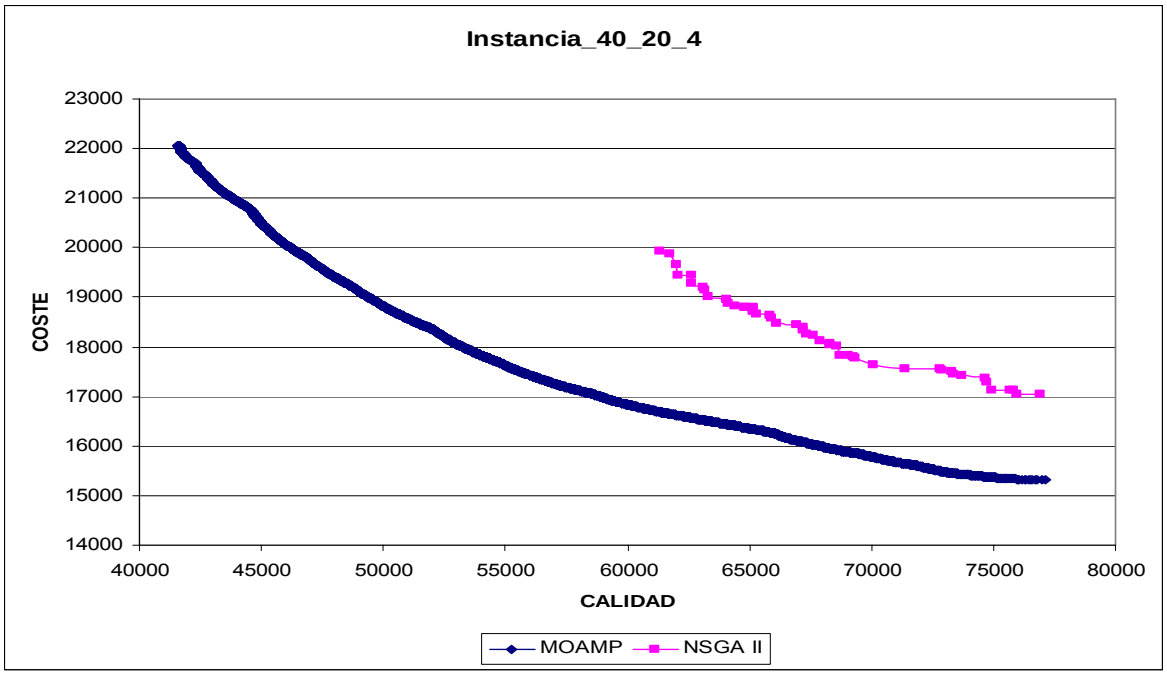
Instancia 40-20-3.dat

MOAMP = 4021 soluciones
NSGA II = 40 soluciones



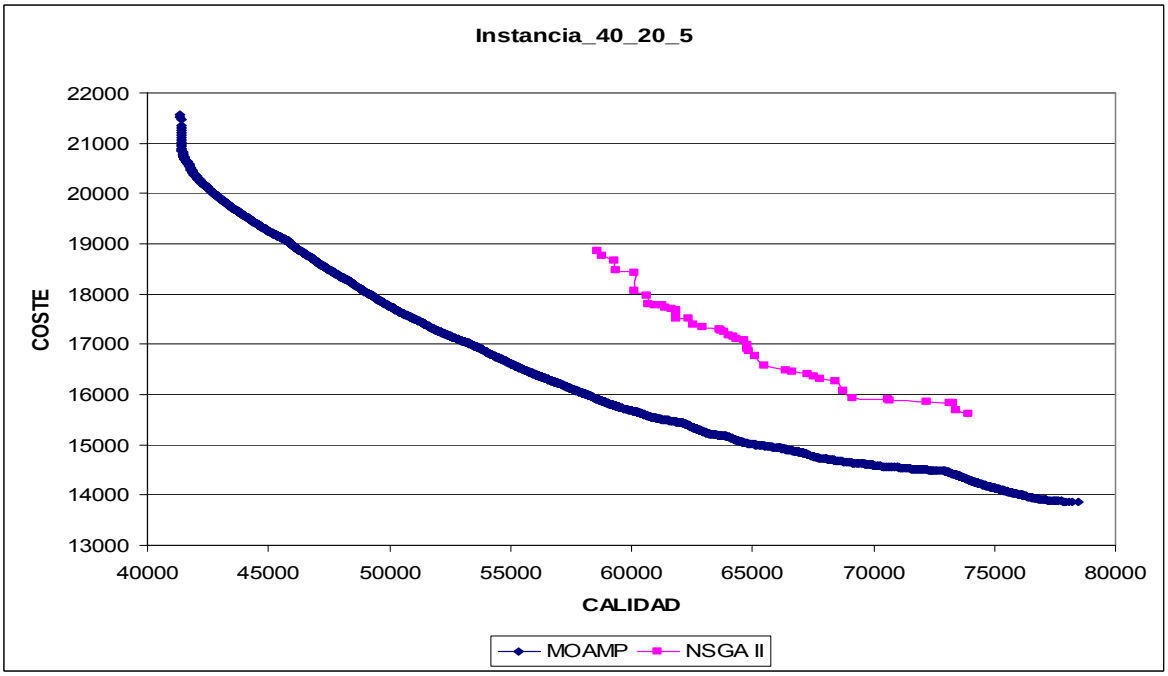
Instancia 40-20-4.dat

MOAMP = 6043 soluciones
NSGA II = 48 soluciones



Instancia 40-20-5.dat

MOAMP = 3459 soluciones
NSGA II = 45 soluciones



Como se puede observar en las gráficas anteriores, a medida que aumenta el tamaño de las instancias, es más apreciable la diferencia entre el conjunto de soluciones MOAMP y NSGA II. En las primeras instancias, hay soluciones de MOAMP dominadas por NSGA II; ahora bien, según aumenta el tamaño, el algoritmo diseñado ad hoc aporta mejores soluciones que el algoritmo genético de propósito general.

Además, MOAMP consigue curvas de eficiencia más pobladas y de mejor calidad; respecto al número de soluciones, llega a encontrar más de 6000 soluciones en alguna instancia (40-30-3), sin embargo, NSGA II apenas supera las 125 soluciones (instancia 20-10-2); respecto a la dominancia, las curvas de eficiencia obtenidas con MOAMP dominan mayoritariamente a las obtenidas con NSGA II. En el caso de menor dominancia, supera el 79% (instancia 20-10-2); a medida que aumenta el tamaño, la dominancia es del 100%.

14.- DEFINICIÓN DE LAS INSTANCIAS REALES

14.- DEFINICIÓN DE LAS INSTANCIAS REALES

14.1.- ESTRUCTURA

Una vez testeado el algoritmo, se debe proceder a su ejecución sobre el problema real anteriormente citado. Esto obliga a la definición de las instancias que van a contener toda la información necesaria para que el algoritmo se pueda ejecutar.

En el caso que aquí se aborda, se van a considerar cuatro horizontes temporales de 31 días de duración: dos en invierno (diciembre y enero) y dos en verano (julio y agosto). Debe tenerse en cuenta que los parámetros *q(i)* y *max_sep* son distintos según sea periodo invernal o estival. Dentro de cada periodo, se considera un mes con más festivos (diciembre y agosto) y otro con menos festivos (enero y julio)

A continuación se adjuntan los núcleos (Tabla 16), con los datos de generación de RU, tanto en verano como en invierno (en kg) así como los valores del parámetro *max_sep* (en días).

	POBLACIÓN	HAB/INV	HAB/VER	*q(i) invierno*	*q(i) verano*	*Tipo_Mun. (G/P)*	*max_sep_inv*	*max_sep_ver*
1	Hacinas	208	640	208	640	P	5	2
2	Barbadillo del Mercado	175	750	175	750	P	5	2
3	Hontoria del Pinar	820	3172	820	3172	G	2	1
4	Terrazas	15	45	15	45	P	5	2
5	Jaramillo Quemado	17	205	17	205	P	5	2
6	Arauzo de Miel	387	450	387	450	P	5	2
7	Arauzo de Salce	70	250	70	250	P	5	2
8	Arauzo de Torre	107	450	107	450	P	5	2
9	Arroyo de Salas	26	85	26	85	P	5	2
10	Aldea del pinar	52	112	52	112	P	5	2
11	Barbadillo de Herreros	145	1000	145	1000	P	5	2
12	Barbadillo del Pez	101	990	101	990	P	5	2
13	Cabezón de la Sierra	75	300	75	300	P	5	2
14	Carazo	50	230	50	230	P	5	2
15	Cascajares de la Sierra	29	190	29	190	P	5	2
16	Castrovido	45	139	45	139	P	5	2
17	Contreras	110	230	110	230	P	5	2
18	Doñasantos	63	156	63	156	P	5	2
19	Gallega, La	87	350	87	350	P	5	2
20	Gete	36	96	36	96	P	5	2
21	Hinojar del Rey	49	102	49	102	P	5	2
22	Hoyuelos	22	67	22	67	P	5	2
23	Huerta de Arriba	176	500	176	500	P	5	2
24	Huerta de Rey	1200	2630	1200	2630	G	2	1
25	Iglesiapinta	31	69	31	69	P	5	2
26	Jaramillo de la Fuente	31	355	31	355	P	5	2
27	Monasterio de la Sierra	39	330	39	330	P	5	2
28	Monterrubio de la Demanda	96	200	96	200	P	5	2
29	Navas del Pinar	137	280	137	280	P	5	2
30	Peñalba de Castro	99	312	99	312	P	5	2
31	Piedrahita de Muño	21	45	21	45	P	5	2
32	Pinilla de los Barruecos	142	450	142	450	P	5	2

	POBLACIÓN	HAB/INV	HAB/VER	q(i) invierno	q(i) verano	Tipo_Mun. (G/P)	max_sep_inv	max_sep_ver
33	Pinilla de los Moros	50	305	50	305	P	5	2
34	Quintanarraya	81	159	81	159	P	5	2
35	Quintanilla Urrilla	45	130	45	130	P	5	2
36	Rabanera del Pinar	135	500	135	500	P	5	2
37	Revilla, La	110	358	110	358	P	5	2
38	Ahedo de la Sierra	20	65	20	65	P	5	2
39	Riocavado de la Sierra	72	595	72	595	P	5	2
40	San Millán de Lara	75	351	75	351	P	5	2
41	Vallejimeno	51	110	51	110	P	5	2
42	Huerta de Abajo	107	576	107	576	P	5	2
43	Tolbaños de Abajo	54	288	54	288	P	5	2
44	Tolbaños de Arriba	53	288	53	288	P	5	2
45	Villanueva de Carazo	29	90	29	90	P	5	2
46	Vizcaínos	66	300	66	300	P	5	2
47	Mamolar	67	400	67	400	P	5	2
48	Salas de los Infantes	2064	4360	2064	4360	G	2	1

Tabla 16. Relación de núcleos, con valores de *q(i)* y de *max_sep*.

A continuación se describe la estructura de una instancia tipo.

Es preciso determinar la situación para cada núcleo (en coordenadas UTM) mediante un Sistema de Información Geográfica. Así mismo, se necesita calcular la matriz de distancias (en hm) para cada par de núcleos *i, j* en la cual queden determinadas las distancias mínimas entre ellos; para ello se hace uso del algoritmo de Dijkstra. Una vez determinada la matriz de distancias, se puede obtener la matriz de tiempos (función lineal de la distancia, así como de otras características: pendientes, tipo de carretera, etc.).

Siguiendo el orden anterior, se debe indicar para cada núcleo la producción diaria de RU (en kg) así como el valor adoptado de *max_sep* (según sea periodo invernal o estival). A modo ilustrativo, se adjunta la lista para el mes de agosto.

640 2	230 2	330 2	351 2
750 2	190 2	200 2	110 2
3172 1	139 2	280 2	576 2
45 2	230 2	312 2	288 2
205 2	156 2	45 2	288 2
450 2	350 2	450 2	90 2
250 2	96 2	305 2	300 2
450 2	102 2	159 2	400 2
85 2	67 2	130 2	4360 1
112 2	500 2	500 2	
1000 2	2630 1	358 2	
990 2	69 2	65 2	
300 2	355 2	595 2	

Así mismo, en cada instancia debe quedar reflejado el número de tipos de día —en este caso, tres— (1=laboral, 2=sábado, 3=festivo) así como la secuencia de tipos de día para cada mes considerado. Las secuencias que se adoptan para cada instancia son:

- enero 1 1 1 1 2 3 1 1 1 1 1 2 3 1 1 1 1 1 2 3 1 1 1 1 1 2 3 1 1 1 1
- febrero 1 1 1 1 1 2 3 1 1 1 1 1 2 3 1 1 1 1 1 2 3 1 1 1 1 1 2 3 1 1 1
- agosto 1 1 2 3 1 1 1 1 1 2 3 1 1 1 3 1 2 3 1 1 1 1 1 2 3 1 1 1 1 1 2

- diciembre 3 1 1 1 1 3 2 3 1 1 1 1 1 2 3 1 1 1 1 1 2 3 1 1 3 1 1 2 3 1 3

A continuación, se debe recoger el número de tipos de vehículos, el número total de vehículos y el número de vehículos para cada tipo; este valor puede ser distinto en cada mes.

Por último, se adjunta la información relativa a costes y capacidades de los vehículos de recogida. Los valores adoptados son los que a continuación se citan:

COSTES DE EXPLOTACIÓN

- COSTES FIJOS (€ \ día)

DÍA / TIPO DE FLOTA	**LABORABLE**	**SÁBADO**	**FESTIVO**
FLOTA PROPIA	470	470	543
FLOTA AJENA	658	658	760

- COSTES VARIABLES (€ \ km)

0.35 € \ km

CAPACIDAD DE LOS VEHÍCULOS

- 14000 kg.

Las instancias generadas se reflejan en los ficheros de estructura ASCII que a continuación se citan:

- Datos_reales_Alfoz_Enero.dat.
- Datos_reales_Alfoz_Julio.dat.
- Datos_reales_Alfoz_Agosto.dat.
- Datos_reales_Alfoz_Diciembre.dat.

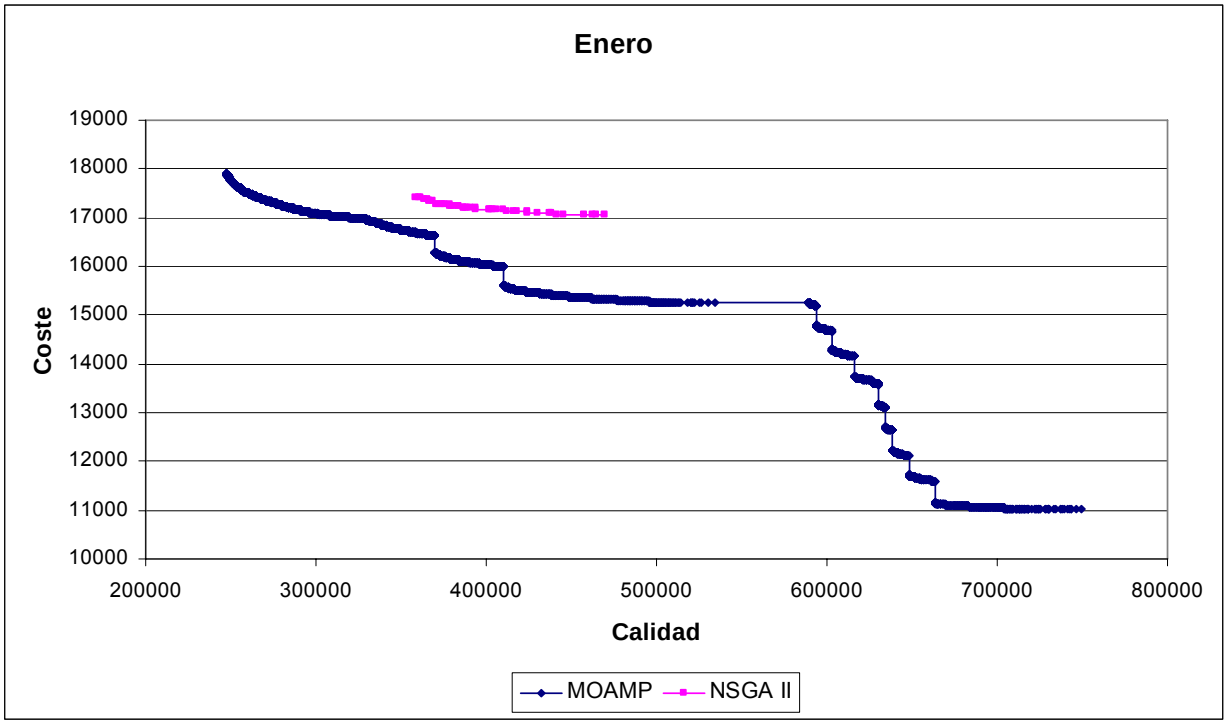
14.2.- PLANTEAMIENTO INICIAL

Las instancias anteriormente definidas se van a resolver con MOAMP y con NSGA-II. La resolución, pasa por obtener el conjunto de soluciones no dominadas con ambas estrategias. Dichos conjuntos se reflejarán sobre un mismo gráfico a fin de observar las características de los mismos.

14.3. - RESULTADOS

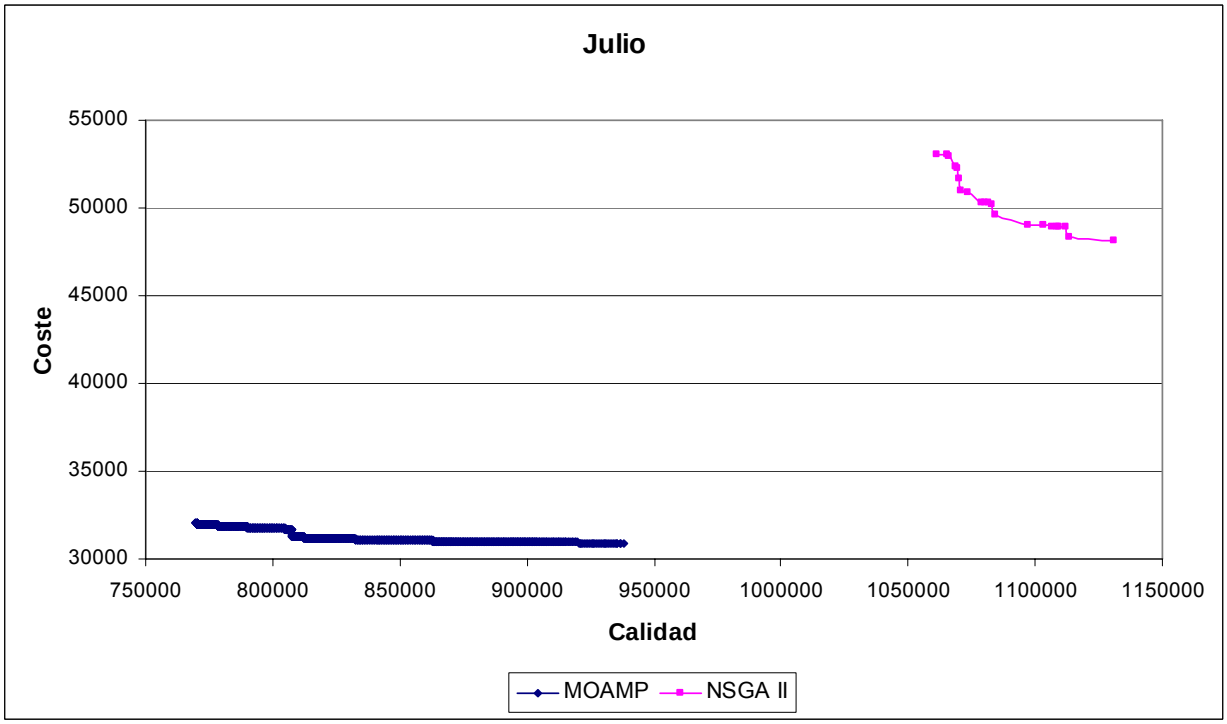
ENERO

MOAMP	6474	Soluciones
NSGA II	46	Soluciones



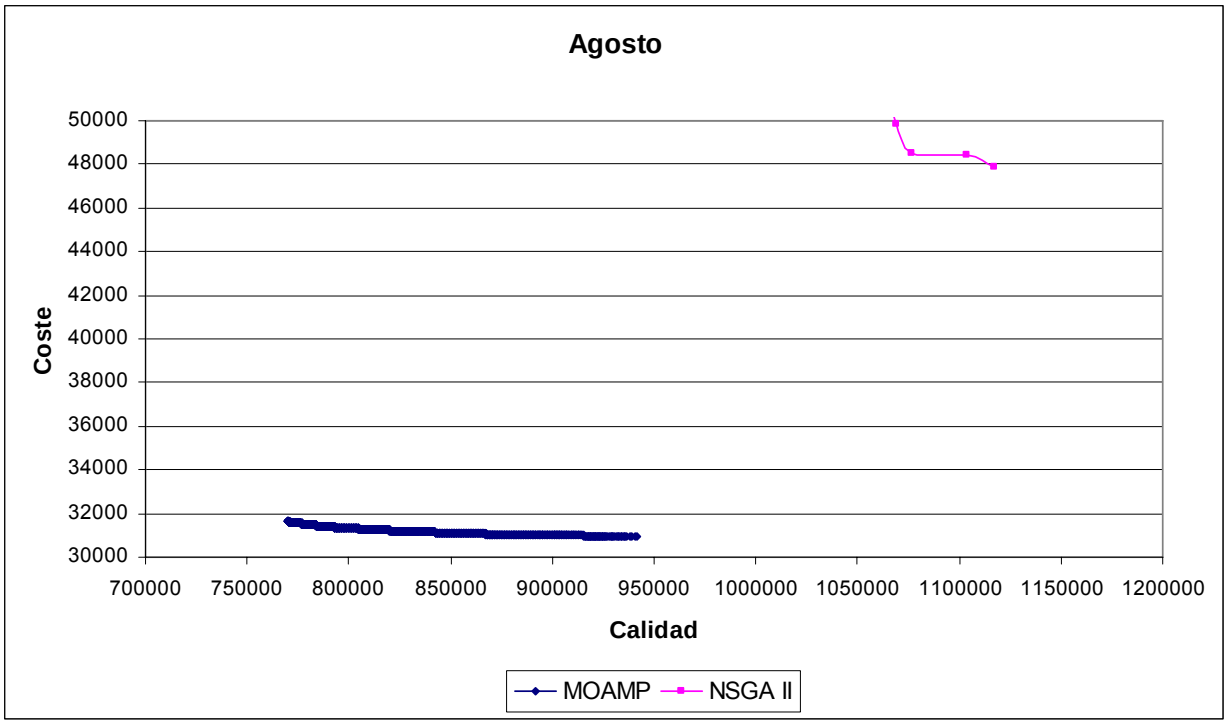
JULIO

MOAMP	1478	Soluciones
NSGA II	21	Soluciones



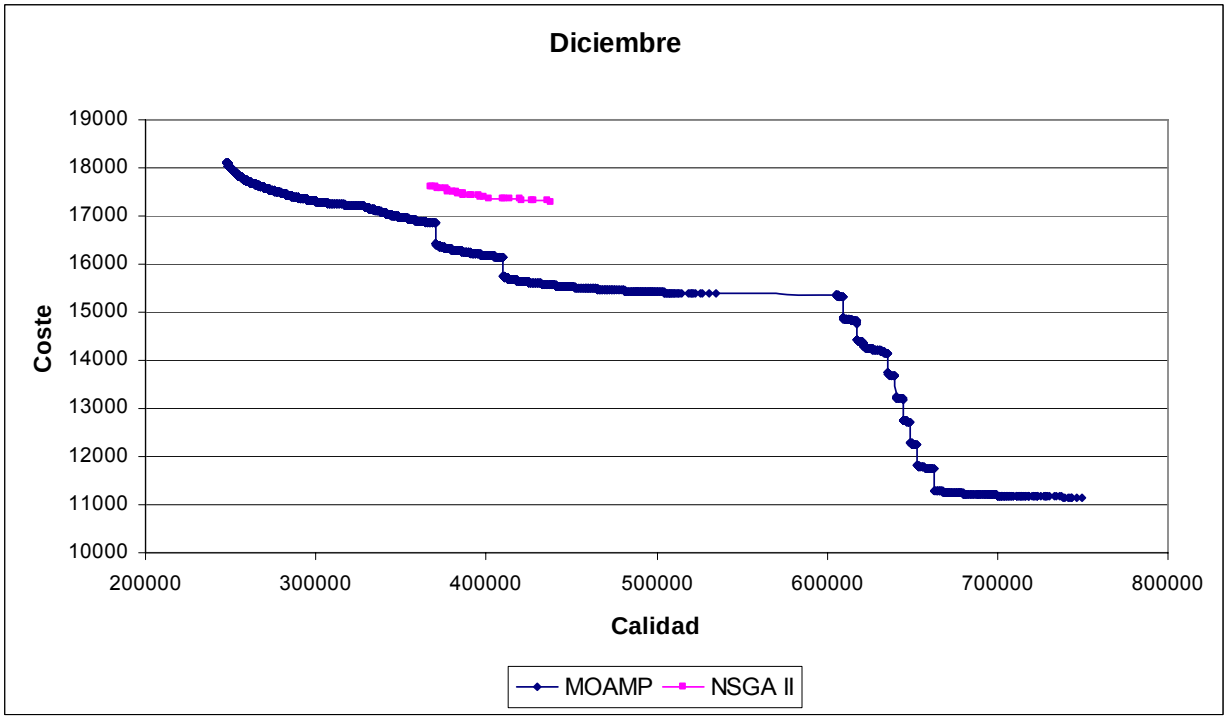
AGOSTO

MOAMP	1549	Soluciones
NSGA II	9	Soluciones



DICIEMBRE

MOAMP	5737	Soluciones
NSGA II	32	Soluciones



Se aprecia, al igual que en las instancias ficticias, la mejor calidad del conjunto de soluciones obtenido con MOAMP, frente al obtenido con NSGA II.

Esta diferencia es más acentuada en los meses de julio y agosto, en los cuales hay menor número de soluciones por tener restricciones más severas, concluyendo, por tanto, un mejor comportamiento del heurístico ad hoc implementado para tal fin frente al algoritmo genético de propósito general.

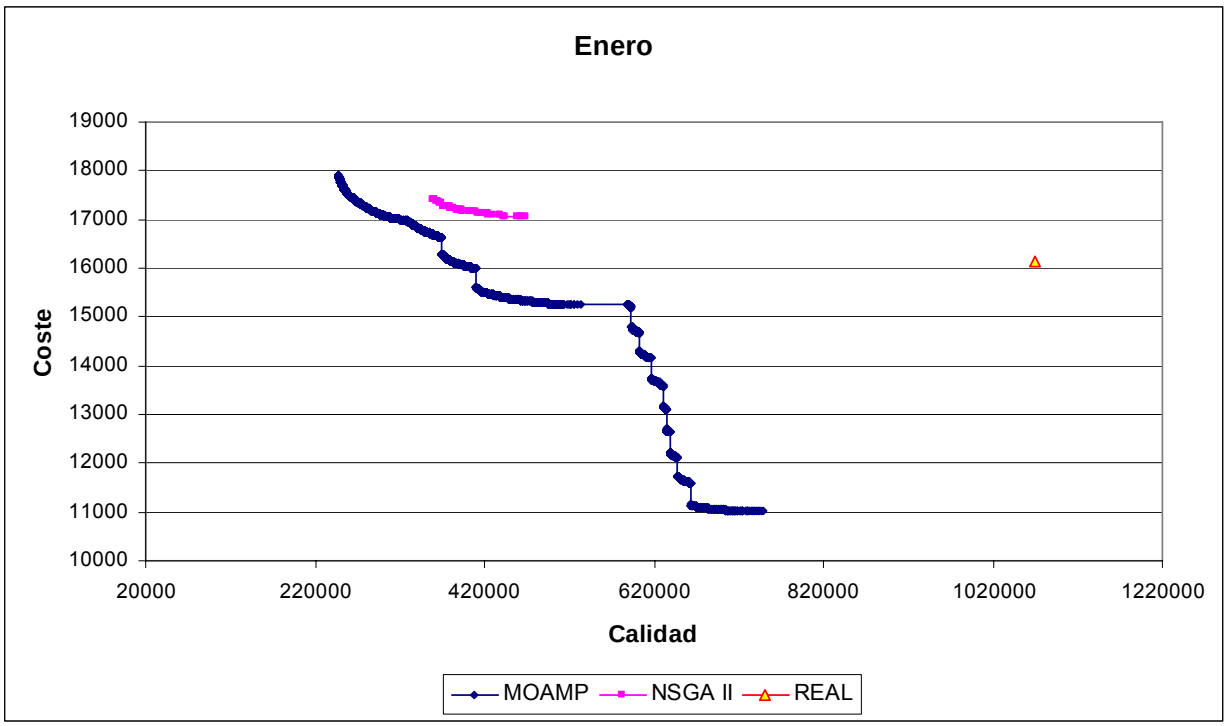
En los meses de enero y diciembre se observan saltos importantes (en sentido vertical) en el conjunto de soluciones no dominadas, lo cual puede ser atribuible, entre otras, a la incorporación-eliminación de un nuevo vehículo —en el caso de soluciones con valores de calidad muy parecidos y gran salto en el coste— o bien en sentido horizontal, con configuraciones de rutas de coste muy parecido pero con gran variación en calidad.

Respecto a los meses centrales, las soluciones no dominadas conforman una curva con un aspecto más continuo, lo que puede ser debido a la falta de movimientos que causen variaciones importantes en las funciones de calidad y coste, por no disponer de tanta elasticidad como en los meses de enero y diciembre.

14.4.- COMPARACIÓN CON LA SITUACIÓN ACTUAL

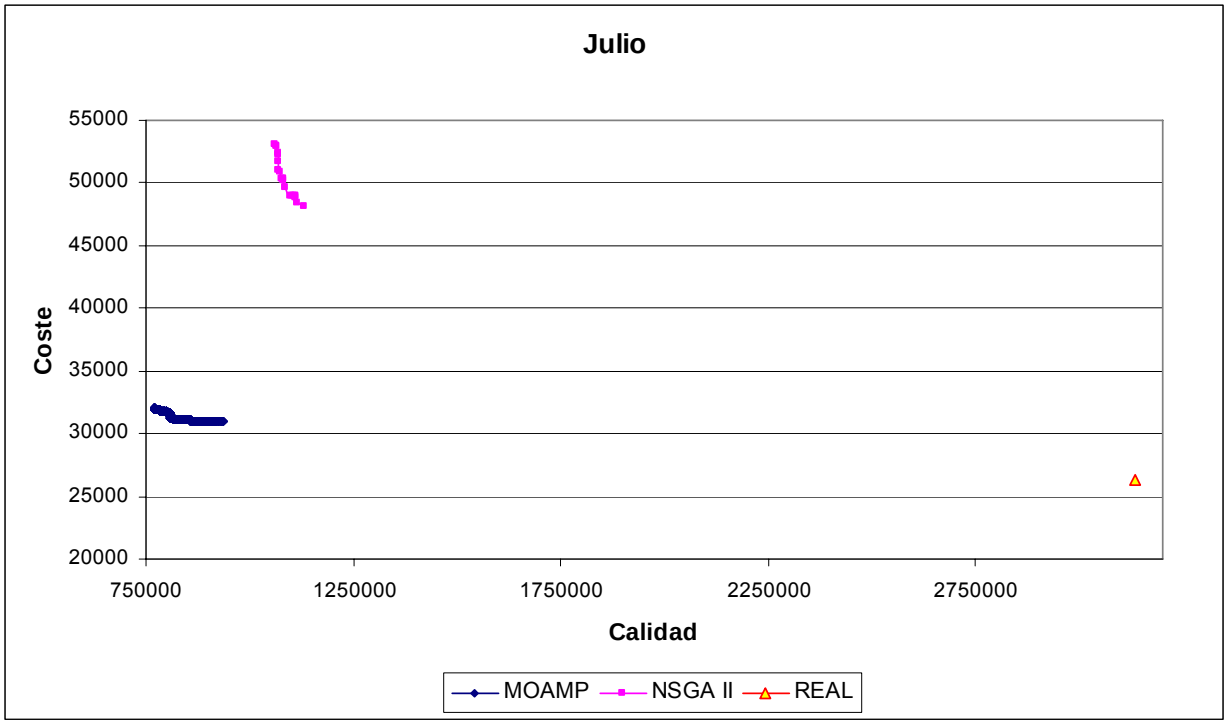
A continuación, se adjuntan los gráficos de la sección anterior, en los que se añade la posición del punto que representa al sistema de recogida actual, a fin de poder comparar dichas posiciones. Hay que destacar que el sistema diseñado no es el mismo que el sistema existente, por contemplar diferentes valores de max_sep_i (máximo intervalo, en días, entre dos recogidas simultáneas en el núcleo i) de los diferentes núcleos, no obstante, se considera interesante adjuntarlo.

ENERO



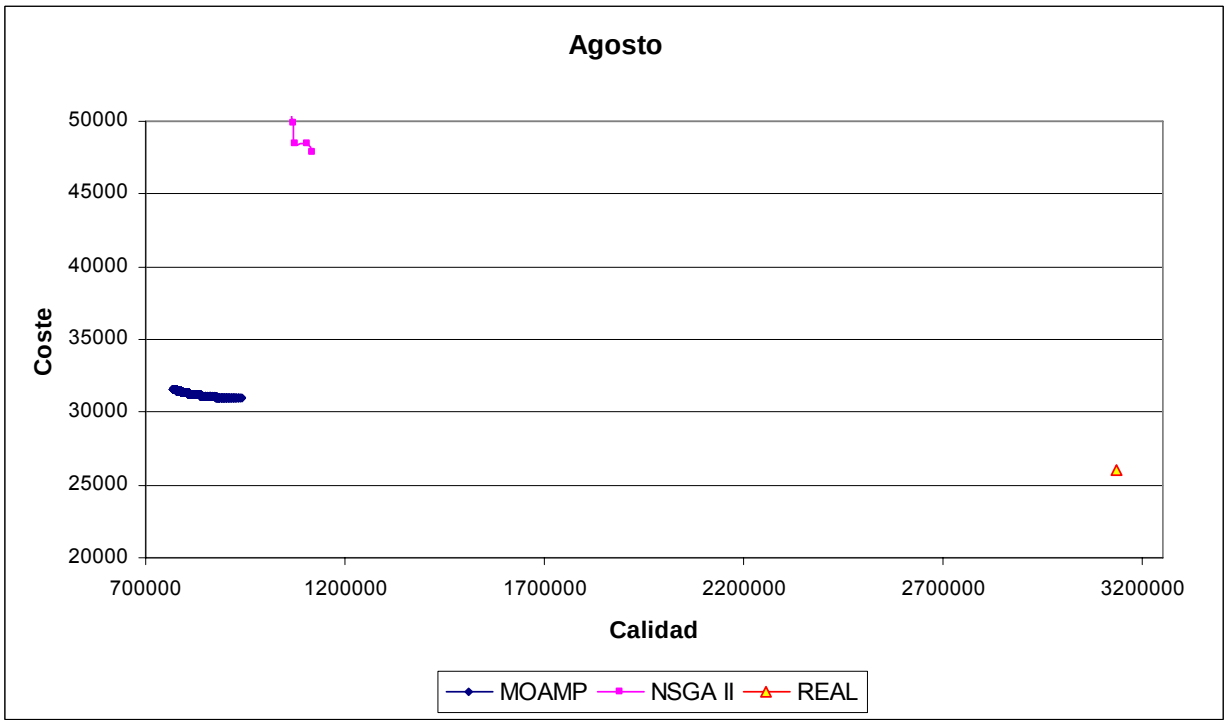
Valores situación actual (Calidad [1068972], Coste [16154.20]).

JULIO



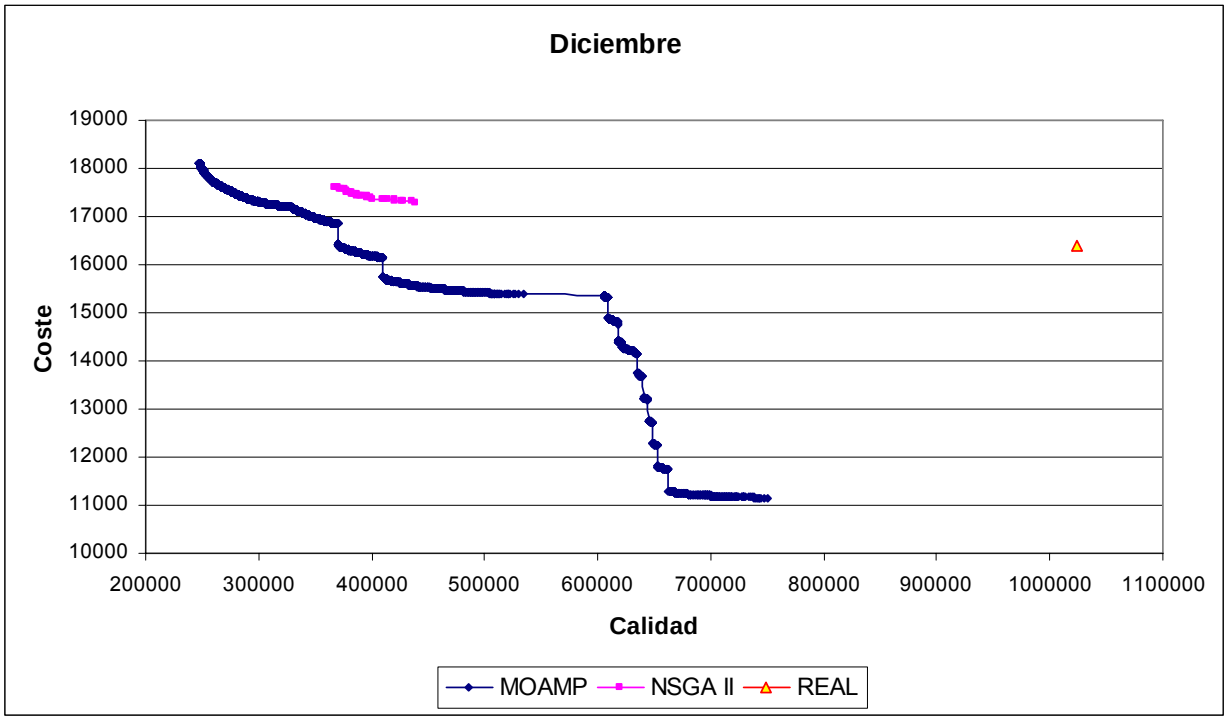
Valores situación actual (Calidad [3132576], Coste [26351.20]).

AGOSTO



Valores situación actual (Calidad [3134054], Coste [25994.65]).

DICIEMBRE



Valores situación actual (Calidad [1024692], Coste [16389.30]).

En la comparación con la situación actual, se aprecia la diferencia del conjunto de soluciones, con dicha solución.

En julio y agosto, se observa un mayor coste de las soluciones no dominadas, frente a la solución actual, a costa de una gran mejora en la calidad; esto se debe a que se imponen valores del parámetro max_sep_i más restrictivos que los actuales, lo que provoca una gran mejora en la calidad, a costa de un incremento no excesivo en el coste.

Se debe indicar que valores de max_sep_i pequeños (1 día en núcleos grandes y 2 días en núcleos pequeños) como es este caso, impone una gran rigidez en la frecuencia de recogida, lo que provoca no solo un menor número de soluciones —en torno a 1500, frente a las aproximadamente 6000 en invierno— sino también un mayor coste; ahora bien, gráficamente, y complementando el párrafo anterior, el problema aquí tratado no se puede comparar al caso real, por partir de restricciones distintas.

En invierno, en los núcleos grandes también se da esta circunstancia, si bien las soluciones obtenidas son claramente mejores a la situación real, tanto en calidad como en coste, lo cual refleja la bondad del conjunto de soluciones. Se debe hacer mención al coste pues, en este caso, si se consigue mejorar sustancialmente el coste, logrando, además, una mejora notable en la calidad.

15.- REFLEXIONES, CONCLUSIONES Y LÍNEAS FUTURAS DE INVESTIGACIÓN

15.- APORTACIONES, REFLEXIONES, CONCLUSIONES Y LÍNEAS FUTURAS DE INVESTIGACIÓN

15.1.- APORTACIONES

En este trabajo se ha desarrollado un método para resolver el problema de diseño de rutas diarias, a lo largo de un horizonte de planificación, para la recogida de residuos en diferentes núcleos de población de un área rural. El objetivo en este problema es doble: racionalizar el coste de las operaciones de rutas (recogida de residuos y su transporte al depósito central) y mejorar el nivel de servicio (es decir, conseguir frecuencias de recogidas adecuadas en los diferentes núcleos); se trata, por tanto, de un problema bi-objetivo. El método desarrollado aporta un conjunto denso de soluciones no dominadas que pueden ayudar a encontrar un equilibrio adecuado entre ambos objetivos, a menudo en conflicto. El problema está inspirado en la problemática existente en un área geográfica concreta (Mancomunidad de "El Alfoz de Lara", en el sureste de la provincia de Burgos); sin embargo, es susceptible de ser adaptado a otros entornos geográficos con las correspondientes modificaciones.

El problema en si es nuevo; de hecho, no existen muchas referencias sobre problemas de recogidas de residuos urbanos en áreas rurales. No obstante, existen parecidos con situaciones similares, lógicamente, pero este problema real presenta características diferentes a los anteriores. Por ejemplo, respecto a las funciones objetivos, como indicador de calidad (o "no-calidad") se usa una función diferente a las de anteriores referencias, y que se relaciona con la integral de la curva de residuos acumulados (al considerar como medidor de no-calidad "la suma" de estos en todos los instantes del periodo de planificación). Evidentemente este indicador está estrechamente relacionado con la frecuencia como otros usados anteriormente, pero se entiende que está más acorde con la idea de calidad y nivel de servicio en este contexto.

Desde un punto de vista metodológico se ha optado por considerar este problema como un modelo con 2 niveles de decisión: a) Para cada núcleo de población determinar en qué fechas este es visitado y b) diseñar las rutas correspondientes para cada día. Obviamente, esta idea no excluye otras posibilidades.

Más concretamente, se ha diseñado un método ad hoc para este problema. Este método sigue las ideas de la estrategia MOAMP diseñada para problemas multi-objetivo. Esta estrategia ha sido propuesta recientemente y se basa en dos principios: 1) la proximidad entre puntos eficientes; y 2) los puntos que minimizan la distancia $L\infty$ (ponderada y/o escalada) al punto ideal son puntos eficientes. Explotando estos dos principios, la estrategia MOAMP se compone de 3 fases: 1) Obtención de buenas soluciones al problema considerando cada una de las funciones objetivos originales; 2) obtención de buenas soluciones considerando funciones objetivos mixtas, que son diferentes ponderaciones usadas en la distancia $L\infty$ y 3) Exploración de las soluciones vecinas de las soluciones no-dominadas. En las dos primeras fases se enlaza la ejecución de un metaheurístico varias veces, considerando en cada una de ellas, las funciones objetivo antes mencionadas (originales y mixtas). Este metaheurístico suele ser una estrategia basada en movimientos vecinales (como búsqueda tabú, VNS, recocido simulado, etc.) Estos movimientos vecinales son los mismos que se usan en la tercera fase.

En este caso concreto, las herramientas y procedimientos que se han diseñado para componer nuestro método son los siguientes:

- Tres procedimientos para resolver el problema de diseños de rutas diarios: Un método constructivo, un método de búsqueda local y un método de búsqueda tabú.

- Un método de búsqueda tabú para el problema completo que pude usar como función guía tanto las funciones objetivos originales, como las funciones mixtas antes mencionadas.

- Un método de búsqueda vecinal o por entornos, basado en los vecindarios usados por el anterior procedimiento de búsqueda tabú. El conjunto de soluciones vecinas de cada una dada se obtiene de todas las posibles eliminaciones o inserciones factibles de puntos en las diferentes rutas.

Así mismo, se han considerado varias formas de inserción y eliminación (basadas en la heurística GENI para problemas de rutas) que van más allá de las inserciones y eliminaciones clásicas, dando mayor potencia al método propuesto.

Finalmente, desde el punto de vista metodológico, se han desarrollado estrategias de aceleración para algunos de los diferentes procedimientos del método propuesto: a) Una estrategia de búsqueda local rápida para acelerar el procedimiento de búsqueda local para el diseño de rutas diarias; b) una estructura de árboles binarios para ordenar los mejores movimientos en el procedimiento de búsqueda tabú en el problema general; y c) una actualización del conjunto de soluciones no dominadas basados en gestionar (eliminar, insertar, mover) los índices asociados a cada solución y no soluciones enteras; además esta gestión de índices no va a depender del numero de soluciones no dominadas en cada momento. Especialmente esta última idea permite reducciones considerables en el tiempo de cálculo.

El método propuesto se ha comparado, tanto en instancias reales como ficticias, con una adaptación a este problema de la conocida estrategia NSGA-II. Esta estrategia está reconocida actualmente como quizás el mejor “standard” para problemas multi-objetivo. Este método, basado en la estrategia MOAMP, consigue curvas de eficiencia más pobladas y de mejor calidad; en pocos casos soluciones obtenidas por NSGA-II dominan a las obtenidas por MOAMP, y si al revés. No obstante, hay que indicar, que este método está hecho ad-hoc para este problema concreto, y la estrategia NSGA-II es de propósito general, con un patrón muy claro y fácilmente adaptable e implementable en la mayoría de los casos.

En general el uso de estrategias poblacionales como NSGA-II, suele ser la primera opción para problemas multiobjetivo: la facilidad de adaptación, y el hecho de trabajar con poblaciones de soluciones hace preferible su uso (en los problemas multiobjetivo se buscan conjunto de soluciones no dominadas). Sin embargo también se ha de resaltar que en este trabajo ha quedado claro que los métodos poblacionales no son la única alternativa a problemas multi-objetivos y que estrategias como MOAMP, basadas en movimientos vecinales sobretodo en problemas de la complejidad de este, pueden ser una alternativa razonable.

Además, y relacionado con este último punto, creemos que el diseño de técnicas de solución ad hoc para cada problema, frente a adaptaciones de patrones estándar, supone un esfuerzo extra que entendemos, merece la pena ya que puede repercutir en conseguir resultados claramente mejores, al menos en problemas de gran complejidad.

15.2.- REFLEXIONES

En el contexto socioeconómico actual es necesaria la optimización de los servicios, en especial los públicos y, dentro de ellos, los derivados del transporte, pues, como se indica en este trabajo, sus costes pueden suponer entre un 10% y un 20% del coste total del servicio.

Este tipo de problemas, como el que aquí se trata, presentan una elevada complejidad, derivada, entre otros, del número de objetivos que se abordan —en este caso, calidad y coste— generalmente en conflicto: un amplio horizonte temporal, elevado número de poblaciones a visitar y una serie de restricciones a cumplir —capacidad de los vehículos, niveles de servicio, etc.—.

Respecto a la experiencia de los gestores en el diseño de la planificación de este tipo de servicios, si bien logran soluciones que han sido válidas, al menos hasta ahora, no llegan a alcanzar el nivel al que se puede llegar con herramientas aquí descritas o similares. Llegados a este punto, entendemos que la incorporación y desarrollo de herramientas científico-técnicas (con mayor o menor grado de rigor) puede ser crítica en la mejora del nivel de las soluciones obtenidas y, por consiguiente, de la mejora en la eficacia de este tipo de servicios.

En estos sistemas de gestión surgen de forma frecuente reflexiones acerca del papel restricción-función objetivo (como puede ser calidad-coste) y no está claro, desde el punto de vista del decisor, si es mejor mejorar al máximo la calidad considerando un presupuesto fijo u optimizar el coste considerando un nivel de servicio mínimo. Para ilustrar esta dualidad, supongamos el siguiente ejemplo: en una determinada actividad se dispone de un presupuesto de 1000 unidades monetarias (u.m.) para gestionar un sistema con un determinado nivel de servicio. ¿Qué ocurriría si el decisor supiera que, con un incremento de 10 u.m., puede aumentar el nivel de su servicio en un 50%?; quizás fuese más racional flexibilizar el presupuesto inicial. En definitiva, planteamientos multiobjetivo como el aquí aportado, facilitan enormemente este tipo de análisis.

Desde un punto de vista científico-técnico, las diferentes metaheurísticas y sus aplicaciones a problemas multiobjetivo, resultan ser un campo muy estudiado en los últimos años y, por ello, se justifica su aplicación a este problema, que aunque presentan aspectos muy similares con otros estudiados en la literatura científica, posee algunas características que lo hacen diferente. Como además, se trata de un problema real, entendemos que es extrapolable a otros entornos geográficos (con sus propias especificidades) con las adaptaciones correspondientes.

En un plano más técnico, el desarrollo de heurísticas aplicadas a problemas biobjetivo, como el que aquí se desarrolla, ha estado mayoritariamente dominado por métodos poblacionales-evolutivos. Esto es entendible, ya que en problemas multiobjetivo, se buscan conjuntos de soluciones, más que soluciones de tipo individual. De hecho, existen en la literatura científica aplicaciones exitosas resueltas mediante este tipo de algoritmos. Hay que destacar, además, a favor de este tipo de técnicas, que siguen patrones o esquemas fácilmente reproducibles.

Ahora bien, entendemos que el esfuerzo de diseñar un método ad hoc puede dar lugar, como en este caso, a conjuntos de soluciones significativamente mejores. Concretamente, la búsqueda tabú ha demostrado su buen desempeño en problemas de rutas; es por lo que resulta entendible su uso como elemento fundamental del algoritmo MOAMP aquí utilizado.

Al margen de los elementos que componen MOAMP o estrategias similares, la búsqueda tabú resulta ser una alternativa razonable a los procedimientos poblacionales-evolutivos, al hacer uso de dos principios fundamentales: la proximidad entre puntos eficientes y el uso de funciones mixtas basadas en la distancia al infinito.

15.3.- CONCLUSIONES

El proceso de toma de decisiones, enfocado a la mejora en la gestión de sistemas de recogida de RU como el que aquí se propone, precisa del uso de herramientas científico-técnicas como las que aquí se proponen.

El enfoque multiobjetivo de este problema, facilita un análisis claro del papel y del valor de cada elemento y del grado de flexibilidad que se puede dar a cada objetivo/función.

La complejidad de este problema: horizonte temporal amplio, elevado número de núcleos, restricciones de capacidad y frecuencia de visita, aumentados con el enfoque biobjetivo anteriormente citado, hace que sea necesario el uso o adaptación de métodos heurísticos.

El modelo que aquí se presenta, basado en un problema real, presenta especificidades que le hacen singular a la par que interesante, si bien, existen en la literatura científica modelos con ciertas similitudes.

Como se demuestra en este caso, el diseño de un método ad hoc basado en la estrategia MOAMP ha demostrado aportar un conjunto de soluciones significativamente mejor que los tradicionales métodos poblacionales-evolutivos. Si bien, estos últimos suelen ser de más fácil implementación.

En cuanto a la obtención de curvas de eficiencia, esta estrategia resulta ser más eficaz, si bien el desarrollo ad hoc resulta ser más sofisticado.

15.4.- LÍNEAS FUTURAS DE INVESTIGACIÓN

Si bien en este caso se han considerado dos objetivos, en el futuro se pretende incorporar otros objetivos que, aunque a priori puedan parecer menos relevantes, pueden resultar ser críticos; entre otros: aspectos laborales, transportes peligrosos, etc.

Otro aspecto importante que se debe tener en cuenta, con ciertas connotaciones sociales, es que afecta a la imagen del gestor, de cara a sus administrados, en lo referente a la falta de equilibrio de frecuencia entre núcleos vecinos, y la percepción social que de esta falta puedan tener los afectados.

Por último, considerar patrones de recogida más uniformes, mediante otro tipo de estrategias de planificación.

16.- BIBLIOGRAFÍA

16.- BIBLIOGRAFÍA

1. Martí, R., *Algoritmos Heurísticos en Optimización Combinatoria.* Cursos de Doctorado. Departamento de Estadística e Investigación Operativa. Universidad de Valencia.
2. Laguna, M. y C. Delgado, *Introducción a los metaheurísticos.* Procedimientos metaheurísticos en economía y empresa.
3. Salazar, J.J., *Programación matemática*. Díaz de Santos, S.A. ed. 2.001.
4. Garey, M.R., *Computers and Intractability: A Guide to the Teory of NP-Completeness*. 1978: Freeman, New York.
5. Martí, R., *Procedimientos Metaheurísticos en Optimización Combinatoria.*
6. Nemhauser, G. y L. Wolsey, *Integer and Combinatorial Optimization.* John Wiley&Sons, 1988.
7. Silver, E., R.V. Vidal y D. Werra, *A tutorial on Heuristic Methods.* European Journal of Operational Research, 1980. 5: p. 153-162.
8. Glover, F., *Future Paths for integer programming and links to artificial intelligence.* Computer and Operational Research, 1986. 5: p. 533-549.
9. Glover, F. y M. Laguna, *Tabú search.* Kluwer Academic Publishers, Boston, 1997.
10. Osman, I. y J.P. Kelly, *Meta-heuristics: Theory & Applications.* Kluwer Academic Publisher. Boston, 1996.
11. Voss, S., *Solving Quadratic Assingnment Problems Using the Reverse Elimination Method.* Technische Hochschule Darmstadt. Germany, 1993.
12. Laguna, M. y C. Delgado, *Introducción a los metaheurísticos*, en *Procedimientos metaheurísticos en economía y empresa*, T.l. blanch, Editor. 2007. p. 3-28.
13. Cowling, P., G. Kendall y E. Soubeiga, *Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation*, en *Applications of Evolutionary Computing, Proceedings*. 2002. p. 1-10.
14. Burke, E.K., G. Kendall y E. Soubeiga, *A tabu-search hyperheuristic for timetabling and rostering.* Journal of Heuristics, 2003. 9(6): p. 451-470.
15. Holland, J., *Adaptation in Natural and Artificial Systems.* University of Michigan.Press, Ann Abor MI, 1975.
16. Caballero, R., J. Molina y A.G. Hernández-Díaz, *Metaheurísticos en Programación Multiobjetivo*, en *Procedimientos Metaheurísticos en Economía y Empresa*, T.L. Blanch, Editor. 2007. p. 117-138.
17. Kirkpatrick, S., J.R. Gelatt y M.P. Vecchi, *Optimization by Simulated Anneaing* Science, 1983. 220: p. 671-680.
18. Dowsland, K.A. y B. Adenso, *Diseño de Heurísticas y Fundamentos del Recocido Simulado.* Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial, 2001. 20: p. 34-52.
19. Metropolis, N., et al., *Equation of state calculation by fast computing machines.* Journal of Chemistry Physics, 1953. 21: p. 1087-1091.
20. Aarts, E.H.L. y J. Lenstra, *Local Search in Combinatorial Optimization.* 1997: Wiley, Chinchester.
21. Aarts, E.H.L. y J.H.M. Korst, *Simulated Annealing and Boltzmann Machines*. 1989: Wiley, Chichester.
22. Hajek, B., *Cooling schedules for optimal annealing.* Mathematics of Operations Research., 1988. 13: p. 311-329.
23. Vidal, R.V., *Applied Simulated Annealing.* 1993: Springer-Verlag. Berlin.
24. Taillard, E.D., et al., *Adaptive memory programming: A unified view of metaheuristics.* European Journal of Operational Research, 2001. 135(1): p. 1-16.
25. Moscato, P., *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Toward Memetics Algorithms.* Technical Report Caltech Concurrent Computation Program, C3P. California Institute of Technology, Pasadena. California USA, 1989. Report 826.
26. Culberson, J., *On the futility of blind search: An algorithmic view of "No Free Lunch".* Evolutionary Computation, 1998. 6: p. 109-127.

27. Cotta, C., *Una Visión General de los Algoritmos Meméticos*. Procedimientos Metaheurísticos en Economía y Empresa, ed. ASEPUMA. 2007.
28. Glover, F., *Tabu Search. Part 1.* ORSA Journal of Computing, 1989. 1: p. 190-206.
29. Glover, F., *Genetic Algorithms and Scatter Search: Unsuspected Potencials.* Statistics and Computing, 1994a. 4: p. 131-140.
30. Glover, F., *Tabu Thresholding: Improved Search by Nommonotonic Trajectories.* ORSA Journal of Computing, 1993.
31. Feo, T.A. y M.G.C. Resende, *A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem.* Operations Research Letters, 1989. 8: p. 67-71.
32. Battiti, R., *Reactive Search: Toward Self-Tuning Heuristics.* In V.J. Rayward-Smith, editor, Modern Heuristics Search Methods, chapter 4., 1996: p. 61-83.
33. Dorigo, M., V. Maniezzo y A. Colorni, *The Ant system: Optimization by a Colony of Cooperating Agents.* IEEE Transactions on Systems, Man and Cybernetics, 1996. Part B 26(1): p. 29-41.
34. Gravel, M., W.L. Price y C. Gagne, *Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic.* European Journal of Operational Research, 2002. 143(1): p. 218-229.
35. Guntsch, M. y M. Middendorf, *Solving multi-criteria optimization problems with population-based ACO.* Evolutionary Multi-Criterion Optimization, Proceedings, 2003. 2632: p. 464-478.
36. Rosing, K.E., C.S. Revelle y D.A. Schilling, *A gamma heuristic for the p-median problem.* European Journal of Operational Research, 1999. 117(3): p. 522-532.
37. Pacheco, J. y C. Delgado, *Diseño de Metaheurísticos para Problemas de Rutas con Flota Heterogénea: Concentración Heurística.* Estudios de Economía Aplicada, 2000. 14: p. 137-151.
38. Glover, F., *Heuristics for integer programming using surrogate constraints.* Decision Sciences, 1977. 8: p. 156-166.
39. Martí, R. y M. Laguna, *Scatter Search: Diseño Básico y Estrategias Avanzadas.* Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial, 2003. 19: p. 123-130.
40. Voudouris, C. y E. Tsang, *Guided Local Search*, en *Department of Computer Science*. 1995, University of Essex: Colchester, C04 3SQ, UK.
41. Kilby, P., P. Prosser y P. Shaw, *Guided Local Search for the Vehicle Routing Problem.*, en *2nd Metaheuristics International Conference (MIC 97)*. 1997: Sophie Antipolis, France.
42. Hansen, P. y N. Mladenovic, *Variable neighborhood search for the p-median.* Location Science, 1997. 5: p. 207-226.
43. Hansen, P. y N. Mladenovic, *A tutorial on variable neighborhood search.*, en *Les Cahiers du GERAD*. 2002.
44. Mladenovic, N., *A Variable Neighborhood Algorithm - A New Metaheuristic for Combinatorial Optimization*, en *Abstract of papers presented at Optimization Days*. 1995: Montreal. p. 112.
45. Grefenstette, J.J., *Optimization of control parameters for genetic algorithms.* IEEE Transactions on Systems, Man, and Cybernetics, 1986. 16(1): p. 122-128.
46. Larrañaga, P., J.A. Lozano y H. Muhlenbein, *Estimation of Distribution Algorithms Applied To Combinatorial Optimization Problems.* Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial, 2003. 19: p. 149-168.
47. Mühlenbein, H. y G. Paab, *From recombination of genes to the estimation of distributions I. Binary parameters.* Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature, 1986: p. 178-187.
48. Larrañaga, P., et al., *Optimization by learning and simulation of Bayesian and Gaussian networks. Technical Report KZZA-IK-4-99, Department of Computer Science and Artificial Intelligence, University of the Basque Country, 1999.*
49. Pelikan, M., D.E. Godberg y F. Lobo, *A survey of optimization by building and usin probabilistic models.Technical Report IlliGAL Report 99018. 1999. University of Illinois at Urbana-Champaing.*
50. Christofides, N., *The bionomic algorithm.*, en *AIRO*. 1994: Savona. Italy.
51. Martí, R. y J.A. Moreno, *Métodos Multiarranque.* Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial, 2003. 19: p. 49-60.

52. Boender, C.G.E. y A.H.G. Rinnooy, *A Bayesian Analysis of the Number of Cells of a Multinomial Distribution.* The Statician, 1983. 32: p. 240-248.
53. Los, M. y C. Lardinois, *Combinatorial Programming, Statistical Optimization and the Optimal Transportation Network Problem.* Transportation Research, 1982. 2: p. 89-124.
54. Glover, F., *Multi-start and strategic oscillation methods - principles to explit adaptive memory*, en *Computing Tools for modeling, Optimization and Simulation.*, M. Laguna y J.L. González Velarde, Editors. 2000. p. 1-24.
55. Martí, R., *Multistart methods*, en *Handbook of Metaheuristics*, F. Glover y A. Kochenberger, Editors. 2003. p. 355-368.
56. Storn, R. y K. Price, *Differential Evolution- A Fast and Efficient Heuristic for Global Optimization over Continuous Spaces.* Journal of Global Optimization, 1997. 11: p. 341-359.
57. Kennedy, J. y R.C. Eberhart, *Swarm Intelligence*, M.K. Publishers, Editor. 2001: California, USA.
58. Santana-Quintero, L.V., et al., *A new proposal for multiobjective optimization using particle swarm optimization and rough sets theory.* Parallel Problem Solving from Nature - Ppsn Ix, Proceedings, 2006. 4193: p. 483-492.
59. Reynolds, R.G., *An Introduction to Cultural Algorithms.*, en *Third Annual Conference on Evolutionary Programming*, A.V. Sebald y L.J. Fogel, Editors. 1994, World Scientific, River Edge, New Jersey. p. 131-139.
60. Landa, R., *Algoritmos Culturales Aplicados a Optimización con Restricciones y Optimización Multiobjetivo*, en *DEPARTAMENTO DE INGENIERÍA ELÉCTRICA. SECCIÓN DE COMPUTACIÓN*. 2002, CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL: México.
61. Nunes de Castro, L. y J. Timmis, *An Introduction to Artificial Inmune Systems: A New Computational Intelligence Paradigm.* Springer-Verlag, 2002.
62. Coello, C.A. y N. Cruz-Cortés. *An Approach to Solve Multiobjective Optimization Problems Based on an ArtificiL Inmune Systems*. en *First International Conference on Artificial Inmune Systems (ICARIS`2002)*. 2002. University of Kent at Canterbury, Inglaterra.
63. Charon, I. y O. Hudry, *THE NOISING METHOD - A NEW METHOD FOR COMBINATORIAL OPTIMIZATION.* Operations Research Letters, 1993. 14(3): p. 133-137.
64. Darwin, C., *The Origin of Species by Means of Natural Selection* 1859: Murray. London.
65. Boettcher, S. y A. Percus, *Nature's way of optimizing.* Artificial Intelligence, 2000. 119(1-2): p. 275-286.
66. Rubinstein, R.Y., *Optimization of computer simulation models with rare events.* European Journal of Operational Research, 1997. 99(1): p. 89-112.
67. Rubinstein, R.Y., *Combinatorial optimization, cross-entropy, ants and rare events.* Stochastic Optimization: Algorithms and Applications, 2001. 54: p. 303-363.
68. González, P.L., et al., *Optimización sw sistemas a medida mediante el algoritmo Cross-Entropy*, en *X Congreso de Ingeniería de Organización*. 2006: Valencia. (Spain).
69. Pareto, V., *Cours D'Economie Politique.* F.Rouge Lausanne, 1896. I y II.
70. Steuer, R.E., *Multiple Criteria Optimization: Theory, computation and Aplication,*. 1986: Wiley, New York.
71. R.Caballero, J. Molina y M.V.R. Uría, *MOAMP. Programación multiobjetivo mediante un procedimiento de Búsqueda Tabú.*
72. Jones, D.F., S.K. Mirrazavi y M. Tamiz, *Multi-objective meta-heuristics: An overview of the current state-of-the-art.* European Journal of Operational Research, 2002. 137(1): p. 1-9.
73. Serafini, P., *Simulated Annealing for multiobjective optimization problems.* Proceedings of the 10th International Conference on Multiple Criteria Decision Making. Taipei, Taiwan, 1992. 1: p. 87-96.
74. Teghem, J., D. Tuyttens y E.L. Ulungu, *An interactive heuristic method for multi-objective combinatorial optimization.* Computers & Operations Research, 2000. 27(7-8): p. 621-634.
75. Ulungu, E.L., J. Teghem y C. Ost, *Efficiency of interactive multi-objective simulated annealing through a case study.* Journal of the Operational Research Society, 1998. 49(10): p. 1044-1050.
76. Ponce, M.T. y M. Matos, *Multicriteria distribution network planning using simulated annealing.* International Transactions in Operational Research, 1999. 6(4): p. 377-391.

77. Ehrgott, M. y X. Gandibleux, *A survey and annotated bibliography on Multiobjective Combinatorial Optimization.* OR Spektrum, 2000. 22: p. 425-460.
78. Dahl, G., K. Jörnsten y A. Lokketangen, *A Tabu Search approach to the channel minimizaion problem.*, en *ICOTA`95*. 1995: Chengdu, China.
79. Hertz, A., et al., *A multicriteria tabu search approach to cell formation problems in group technology with multiple objectives.* Recherche Operationelle / Operational Research., 1994. 28(3): p. 303-328.
80. Gandibleux, X., N. Mezdaoui y A. Freville, *A tabú search procedure to solve multiobjective combinatorial optimization problems.*, en *Advances in ultiple objective and goal programming, Lecture Notes in Economics and Mathematical Systems, pags 291-300*, R. Caballero, F. Ruiz, y R. Steuer, Editors, Springer: Berlin.
81. Caballero, R., J. Molina y M.V. Rodríguez Uría, *MOAMP. Programación multiobjetivo mediante un procedimiento de Búsqueda Tabú*, en *Actas del II Congreso Español de Metaheurísticas y Algoritmos Evolutivos y Bioinspirados MAEB*. 2003: Gijón.
82. Hansen, M.P., *Tabu Search for Multiobjective Optimization MOTS*, en *13th International Conference on Multiple Criteria DEcision Making*. 1997: Cape Town, South Africa.
83. Ben Abdelaziz, F., S.Krichen y J. Chaouachi, *An Hybrid Metaheuristic for the Multiobjective Knapsack Problem.*, en *Metaheuristics-Advances and Trends in Local Search Paradigms for Optimization.*, S. Voss, et ál., Editors. 1999, Kluwer. p. 205-212.
84. Gandibleux, X. y A. Freville, *Tabu search based procedure for solving the 0-1 multiobjective knapsack problem: The two objectives case.* Journal of Heuristics, 2000. 6(3): p. 361-383.
85. Alves, M.J. y J. Climaco, *An interactive method for 0-1 multiobjective problems using Simulated Annealing and Tabu Search.* Journal of Heuristics, 2000. 6(3): p. 385-403.
86. Ehrgott, M., K. Klamroth y C. Schwehm, *An MCDM approach to portfolio optimization.* European Journal of Operational Research, 2004. 155(3): p. 752-770.
87. Higgins, A.J., S. Hajkowicz y E. Bui, *A multi-objective model for environmental investment decision making.* Computers and Operations Research, 2008. 35(1): p. 253-266.
88. Gandibleux, X., D. Vancoppenolle y D. Tuyttens, *A first making use of GRASP for solving MOCO problems.* Proceeding of the 14th International Conference on Multiple. Criteria Decision-Making, 1998: p. 8-12.
89. Goldberg, D.E., *Genetic Algorithms.*, en *Search, Optimization and Machine Learning.* 1989, Addison-Wesley Publishing Co.
90. Wilson, P.B. y M.D. Macleod, *Low Implementations cost IIR digital filter design using genetic allgorithms.*, en *IEE/IEEE Workshop on Natural Algorithms in Signal Processing.* 1993: Chelmsford. p. 4/1 - 4/8.
91. Schaffer, J.D., *Multiple Objective optimization with vector evaluated genetic algorithms.*, en *Proceedings of the First International Conference on Genetic Algorithms and Their Applications. pags 93-100*. 1985: Lawrence Erlbaum.
92. Fourman, M.P. *Compactation of symbolyc layout using genetic algorithms*. en *Genetic Algorithms and their Applications: Proceeding of the First International Conference on Genetics Algorithms and Their applications*. 1985. Lawrence Erlbaum p. 141-153.
93. Hilliard, M.R., et al., *The computer as a partner in algorithmic design: Automatec discovery of parameters for a multiobjective scheduling heuristic*, en *Impact of Recent Computer Advances on Operations Research.*, B.L.G. R.Sharda, E. Wasil, O. Balci and W. Steward, Editor. 1989, North Holland Publishing Company.
94. Srinivas, N. y K. Deb, *Multiobjective Optimization using Nondominated Sorting in Genetic Algorithm.* Evolutionary computation., 1994. 2: p. 221-248.
95. Horn, J. y N. Nafpliotis, *Multiobjective Optimization using the Niched Pareto Genetic Algorithm. Technical Report IlliGA1 Report 93005*, en *University of Illinois at Urbana-Champaign, Illinois, USA*. 1993.
96. Zitzler, E. y L. Thiele, *Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto approach.* Ieee Transactions on Evolutionary Computation, 1999. 3(4): p. 257-271.

97. Zitzler, E., M. Laumanns y L. Thiele, *SPEA2: Improving the Strength Pareto Evolutionary Algorithm.* 2001, Computer Engineering and Networks Laboratory (TIK). Swiss Federal Institute of Technology (ETH): Zurich, Switzerland.
98. Knowles, J.D. y D.W. Corne, *Approximating the nondominated front using the Pareto Archived Evolution Strategy.* Evolutionary computation, 2000. 8(2): p. 149-172.
99. Deb, K., et al., *A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA II*. 2000, KanGall Report 200001. Indian Institute of Tecnology: Kampur, India.
100. Erickson, M., A. Mayer y J. Horn, *The Niched Pareto genetic algorithm 2 applied to the design of groundwater remediation systems*, en *Evolutionary Multi-Criterion Optimization, Proceedings*, E. Zitzlet, et ál., Editors. 2001. p. 681-695.
101. Molina, J., et al., *SSPMO: A Scatter Tabu Search Procedure for Non-Linear Multiobjective Optimization.* 2005.
102. Beausoleil, R.P., *"MOSS" multiobjective scatter search applied to non-linear multiple criteria optimization.* European Journal of Operational Research, 2006. 169(2): p. 426-449.
103. Perez, F., et al., *Hibridación de métodos exactos y heurísticos para el problema multiobjetivo. XV Jornadas de ASEPUMA y III Encuentro Internacional. Palma de Mallorca, (Spain). 2007*
104. Coello, C.A. y G.B. Lamont, *Applications of multi-objective evolutionary algorithms.* Advances in Natural Computation, 2005. 1.
105. Madavan, N.K., *Multiobjective Optimization Using a Pareto Differential Evolution Approach.*, en *CEC2002*. 2002. p. 1145-1150.
106. Parsopoulos, K.E., et al., *Vector evaluated differential evolution for multiobjective optimization*, en *Cec2004: Proceedings of the 2004 Congress on Evolutionary Computation, Vols 1 and 2*. 2004. p. 204-211.
107. Hernandez-Diaz, A.G., et al., *A new proposal for multi-objective optimization using differential evolution and rough sets theory*, en *Gecco 2006: Genetic and Evolutionary Computation Conference, Vol 1 and 2*, M. Keijzer, Editor. 2006. p. 675-682.
108. Melián, B. y F. Glover, *Introducción a la búsqueda tabú.* 2004.
109. Glover, F., *Tabu Search. Part 2.* ORSA Journal of Computing, 1990a. 2: p. 4-32.
110. Casado, S., *Planificación de turnos en un aeropuerto: uso de simulación y metaheurísticos.*, en *Departamento de Economía Aplicada*. 2005, Universidad de Burgos: Burgos.
111. Diego-Más, J.A., *Optimización de la distribución en planta de instalaciones industriales mediante algoritmos genéticos. Aportación al control de la geometría de actividades.*, en *Departamento de Proyectos de Ingeniería*. 2006, Universidad Politécnica de Valencia: Valencia.
112. Voss, S., *Solving Quadratic Assignment Problems Using the Reverse Elimination Method.* Technische Hochschule Darmstadt. Germany, 1993.
113. Nowicki, E. y C. Smutnicki, *A fast taboo search algorithm for the job shop problem.* Management Science, 1996. 42(6): p. 797-813.
114. Glover, F. y M. Laguna, *Tabu Search*, en *Modern Heuristic Techniques for Combinatorial Problems*, C. Reeves, Editor. 1993, Blackwell Scientific Publishing. p. 70-141.
115. Díaz, J.A. y E. Fernández, *A Tabu Search Heuristic for the Generalized Assigment Problem.* 1998, Document de Recerca, Departament D´EIO. Secció Informàtica. Universitat Politècnica de Catalunya. DR 98/08.
116. Ho, S.C. y M. Gendreau, *Path relinking for the vehicle routing problem.* Journal of Heuristics, 2006. 12(1-2): p. 55-72.
117. Olivera, A., *Heurísticas para Problemas de Ruteo de Vehículos*. 2004, Instituto de Computación, Facultad de Ingeniería, Universidad de la República: Montevideo. Uruguay.
118. Toth, P. y D. Vigo, *An Overview of Vehicle Routing Problems*, en *Discrete Mathematics and Applications. The Vehicle Routing Problem*. 2000, SIAM. p. 1-26.
119. Dantzig, G. y J. Ramser, *The truck dispatching problem.* Management Science, 1959. 6: p. 80-91.
120. Clarke, G. y W. Wright, *Scheduling of vehicles from a central depot to a numbre of delivery points.* Operations Research, 1964. 12: p. 568-581.
121. Fisher, M.L. y R. Jaikumar, *A GENERALIZED ASSIGNMENT HEURISTIC FOR VEHICLE-ROUTING.* Networks, 1981. 11(2): p. 109-124.

122. Toth, P. y D. Vigo, *Exact solution of the vehicle routing problem.* Fleet Management and Logistics, 1998: p. 1-31.
123. Pacheco, J.A., *Modelo de rutas bi-objetivo. Aplicación al transporte escolar en áreas urbanas.*
124. Gillett, B.E. y L.R. Miller, *HEURISTIC ALGORITHM FOR VEHICLE-DISPATCH PROBLEM.* Operations Research, 1974. 22(2): p. 340-349.
125. Potvin, J.Y. y S. Bengio, *The vehicle routing problem with time windows- Part II: Genetic Search* INFORMS Journal on Computing, 1996. 8: p. 165-172.
126. Thangiah, S., *Vehicle Routing with Time Windows using Genetic Algorithms*, en *Application Handbook of Genetic Algorithms: New Frontiers, Volume II*, C. Press, Editor. 1995. p. 253-277.
127. Baker, B.M. y M.A. Ayechew, *A genetic algorithm for the vehicle routing problem.* Computers & Operations Research, 2003. 30(5): p. 787-800.
128. Osman, I., *Metastrategy Simulated Annealing and Tabu Search for the Vehicle Routing Problem.* Annals of Operations Research, 1993. 41: p. 421-451.
129. Gendreau, M., A. Hertz y G. Laporte, *A TABU SEARCH HEURISTIC FOR THE VEHICLE-ROUTING PROBLEM.* Management Science, 1994. 40(10): p. 1276-1290.
130. Rochat, Y. y E.D. Taillard, *Probabilistic diversification and intensification in local earch for vehicle routing.* Journal of Heuristics, 1995. 1: p. 147-167.
131. Taillard, E.D., et al., *A tabu search heuristic for the vehicle routing problem with soft time windows.* Transportation Science, 1997. 31(2): p. 170-186.
132. Cordeau, J.F., G. Laporte y A. Mercier, *A unified tabu search heuristic for vehicle routing problem with time windows. Technical Report CRT-00-04*. 2000, Centre for Research on Transportation: Montreal, Canada.
133. Kontoravdis, G. y J.F. Bard, *A GRASP for the Vehicle Routing Problem with Time Windows.* ORSA Journal of Computing, 1995. 7: p. 10-23.
134. Voudouris, C. y E. Tsang, *Guided local search and its application to the traveling salesman problem.* European Journal of Operational Research, 1999. 113(2): p. 469-499.
135. Gambardella, L.M., E.D. Taillard y M. Dorigo, *Ant colonies for the quadratic assignment problem.* Journal of the Operational Research Society, 1999. 50(2): p. 167-176.
136. Braysy, O., *A reactive variable neighborhood search for the vehicle-routing problem with time windows.* Informs Journal on Computing, 2003. 15(4): p. 347-368.
137. Jozefowiez, N., F. Semet y E.-G. Talbi, *Multi-objective vehicle routing problems.* European Journal of Operational Research, 2008. 189(2): p. 293-309.
138. Park, Y.B. y C.P. Koelling, *A SOLUTION OF VEHICLE-ROUTING PROBLEMS IN A MULTIPLE OBJECTIVE ENVIRONMENT.* Engineering Costs and Production Economics, 1986. 10(2): p. 121-132.
139. Park, Y.B. y C.P. Koelling, *AN INTERACTIVE COMPUTERIZED ALGORITHM FOR MULTICRITERIA VEHICLE-ROUTING PROBLEMS.* Computers & Industrial Engineering, 1989. 16(4): p. 477-490.
140. Sutcliffe, C. y J. Board, *OPTIMAL SOLUTION OF A VEHICLE-ROUTEING PROBLEM - TRANSPORTING MENTALLY-HANDICAPPED ADULTS TO AN ADULT TRAINING-CENTER.* Journal of the Operational Research Society, 1990. 41(1): p. 61-67.
141. Current, J.R. y D.A. Schilling, *THE MEDIAN TOUR AND MAXIMAL COVERING TOUR PROBLEMS - FORMULATIONS AND HEURISTICS.* European Journal of Operational Research, 1994. 73(1): p. 114-126.
142. Lee, T.R. y J.-H. Ueng, *A study of vehicle routing problem with load balancing.* International Journal of Physical Distribution and Logistics Management, 1998. 29: p. 646-648.
143. Sessomboon, W., et al., *A study on multi-objective vehicle routing problem considering custoer satisfaction with due-time (the creation of Pareto optimal solutions by hybrid genetic algorithm)* Transaction of the Japan Society of Mechanical Engineering, 1998.
144. Hansen, M.P., *Use of Substitute Scalarizing Functions to Guide a Local Search Based Heuristic: The Case of moTSP.* Journal of Heuristics, 2000. 6(3): p. 419-431.
145. Ribeiro, R. y H. Lourenco, *A multi-objective model for a multi-period distribution management problem*, en *Metaheuristic International Conference 2001 (MIC`2001)*. 2001.

146. Jozefowiez, N., *Modélisation et résolution approchées de problèmes de tournées multi-objectif.*, en *Laboratoire d`Informatique Fondamentale de Lille*. 2004, Université des Sciences et Technologies de Lille: Villenueve d`Ascq, France.
147. Jozefowiez, N., F. Semet y E.-G. Talbi, *Parallel and hybrid models for multi-objective optimization: Application to the vehicle routing problem*, en *Parallel Problem Solving from Nature VII, Lecture Notes in Computer Science*, J.J. Merelo Guervos, Editor. 2002, Springer-Verlag.
148. Jozefowiez, N., F. Semet y E.G. Talbi, *Enhancements of NSGA II and its application to the vehicle routing problem with route balancing.* Artificial Evolution, 2006. 3871: p. 131-142.
149. Jozefowiez, N., F. Semet y E.G. Talbi, *Target aiming Pareto search and its application to the vehicle routing problem with route balancing.* Journal of Heuristics, 2007. 13: p. 455-469.
150. Borges, P.C. y M.P. Hansen, *A study of global convexity for a multiple objective travelling salesman problem.* Essays and Surveys in Metaheuristics, 2002. 15: p. 129-150.
151. Paquete, L. y T. Stutzle, *A two-phase local search for the biobjective traveling salesman problem.* Evolutionary Multi-Criterion Optimization, Proceedings, 2003. 2632: p. 479-493.
152. Zhenyu, Y., et al., *A new moea for multiobjective tsp and its convergence property analysis.* LNCS, 2003. 2632: p. 342-354.
153. Chitty, D.M. y M.L. Hernandez, *A hybrid ant colony optimisation technique for dynamic vehicle routing.* Genetic and Evolutionary Computation - Gecco 2004, Pt 1, Proceedings, 2004. 3102: p. 48-59.
154. Li, W., *Finding pareto-optimal set by merging attractors for a bi-objective traveling salesman problem*, en *Third International Conference, EMO*. 2005: Guanajanto, Mexico.
155. Murata, T. y R. Itai, *Multi-objective vehicle routing problems using two-fold EMO algorithms to enhance solution similarity on non-dominated solutions*, en *Evolutionary Multi-Criterion Optimization*, C.A.C. Coello, A.H. Aguirre, y E. Zitzler, Editors. 2005. p. 885-896.
156. Murata, T. y R. Itai, *Local search in two-fold EMO algorithm to enhance solution similarity for multi-objective vehicle routing problems*, en *Evolutionary Multi-Criterion Optimization, Proceedings*, S. Obayashi, et ál., Editors. 2007. p. 201-215.
157. Baràn, B. y M. Schaerer. *A multiobjective ant colony system for vehicle routing problem with time windows* en *Proceedings of the Twenty-first IASTED International Conference on Applied Informatics* 2003.
158. Tan, K.C., Y.H. Chew y L.H. Lee, *A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems.* European Journal of Operational Research, 2006. 172(3): p. 855-885.
159. Tan, K.C., et al., *A hybrid multiobjective evolutionary algorithm for solving truck and trailer vehicle routing problems.* Cec: 2003 Congress on Evolutionary Computation, Vols 1-4, Proceedings, 2003: p. 2134-2141.
160. Jozefowiez, N., F. Semet y E.G. Talbi, *The bi-objective covering tour problem.* Computers & Operations Research, 2007. 34(7): p. 1929-1942.
161. Riera-Ledesma, J. y J.J. Salazar-Gonzalez, *The biobjective travelling purchaser problem.* European Journal of Operational Research, 2005. 160(3): p. 599-613.
162. Ombuki, B., B.J. Ross y F. Hanshar, *Multi-objective genetic algorithms for vehicle routing problem with time windows.* Applied Intelligence, 2006. 24(1): p. 17-30.
163. Bowerman, R., B. Hall y P. Calamai, *A MULTIOBJECTIVE OPTIMIZATION APPROACH TO URBAN SCHOOL BUS ROUTING - FORMULATION AND SOLUTION METHOD.* Transportation Research Part a-Policy and Practice, 1995. 29(2): p. 107-123.
164. Giannikos, I., *A multiobjective programming model for locating treatment sites and routing hazardous wastes.* European Journal of Operational Research, 1998. 104(2): p. 333-342.
165. El-Sherbeny, N., *Resolution of a vehicle routing problem with a multi-objective simulated annealing method*, en *Faculté Polytechnique de Mons*. 2001: Belgique.
166. Corberán, A., et al., *Heuristic solutions to the problem of routing school buses with multiple objectives.* Journal of the Operational Research Society, 2002. 53(4): p. 427-435.
167. Pacheco, J. y R. Marti, *Tabu search for a multi-objective routing problem.* Journal of the Operational Research Society, 2006. 57(1): p. 29-37.

168. Zografos, K.G. y K.N. Androutsopoulos, *A heuristic algorithm for solving hazardous materials distribution problems.* European Journal of Operational Research, 2004. 152(2): p. 507-519.
169. Mourgaya, M., *The periodic vehicle routing problem: planning before routing.*, en *Laboratoire de Mathématiques Appliquées de Bourdeaux* 2004, Université de Bourdeaux 1: Bourdeaux, France.
170. Doerner, K., A. Focke y W.J. Gutjahr, *Multicriteria tour planning for mobile healthcare facilities in a developing country.* European Journal of Operational Research, 2007. 179(3): p. 1078-1096.
171. Caballero, R., et al., *Solving a multiobjective location routing problem with a metaheuristic based on tabu search. Application to a real case in Andalusia.* European Journal of Operational Research, 2007. 177(3): p. 1751-1763.
172. Miller, C., A. Tucker y R. Zemlin, *Integer programming formulation of traveling salesman problems.* Journal of the ACM, 1960. 7: p. 326-329.
173. Tirado Dominguez, G., *El problema de rutas de vehículos VRP*, en *Estadística e Investigación Operativa* 2007, Universidad Complutense de Madrid: Madrid. Spain.
174. Golden, B.L., T.L. Magnanti y H.Q. Nguyen, *Implementing Vehicle Routing Algorithms.* Networks, 1977. 7(2): p. 113-148.
175. Kulkarni, R.V. y P.R. Bhave, *Integer programming formulations of vehicle routing problems.* European Journal of Operational Research, 1985. 20(1): p. 58-67.
176. Beltrami, E. y L.D. Bodin, *Networks and Vehicle Routing for Municipal Waste Collection.* Networks, 1974. 4: p. 65-94.
177. Russell, R. y W. Igo, *ASSIGNMENT ROUTING PROBLEM.* Networks, 1979. 9(1): p. 1-17.
178. Christofides, N. y J.E. Beasley, *The Period Routing Problem.* Networks, 1984. 14(2): p. 237-256.
179. Chao, I.M., B.L. Golden y E. Wasil, *AN IMPROVED HEURISTIC FOR THE PERIOD VEHICLE-ROUTING PROBLEM.* Networks, 1995. 26(1): p. 25-44.
180. Cordeau, J.F., M. Gendreau y G. Laporte, *A tabu search heuristic for periodic and multi-depot vehicle routing problems.* Networks, 1997. 30(2): p. 105-119.
181. Drummond, L.M.A., L.S. Ochi y D.S. Vianna, *An asynchronous parallel metaheuristic for the period vehicle routing problem.* Future Generation Computer Systems, 2001. 17(4): p. 379-386.
182. Alegre, J.F., S. Casado y C. Delgado, *Diseño de calendarios para transporte de componentes de automóviles.Soluciones heurísticas.* Estudios de Economía Aplicada, 2002. 20(2): p. 301-316.
183. Alegre, J., M. Laguna y J. Pacheco, *Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts.* European Journal of Operational Research, 2007. 179(3): p. 736-746.
184. Francis, P. y K. Smilowitz, *Modeling techniques for periodic vehicle routing problems.* Transportation Research Part B-Methodological, 2006. 40(10): p. 872-884.
185. Mourgaya, M. y F. Vanderbeck, *Column generation based heuristic for tactical planning in multi-period vehicle routing.* European Journal of Operational Research, 2007. 183(3): p. 1028-1041.
186. Dror, M., G. Laporte y P. Trudeau, *VEHICLE-ROUTING WITH STOCHASTIC DEMANDS - PROPERTIES AND SOLUTION FRAMEWORKS.* Transportation Science, 1989. 23(3): p. 166-176.
187. Dror, M. y P. Trudeau, *SPLIT DELIVERY ROUTING.* Naval Research Logistics, 1990. 37(3): p. 383-402.
188. Dror, M., G. Laporte y P. Trudeau, *VEHICLE-ROUTING WITH SPLIT DELIVERIES.* Discrete Applied Mathematics, 1994. 50(3): p. 239-254.
189. Frizzell, P.W. y J.W. Griffin, *The split delivery vehicle scheduling problem with time windows and grid network distances.* Computer and Operations Research, 1995. 22: p. 655-667.
190. Mullaseril, P.A., M. Dror y J. Leung, *Split-delivery routing heuristics in livestock feed distribution.* Journal of the Operational Research Society, 1997. 48(2): p. 107-116.
191. Sierksma, G. y G.A. Tijssen, *Routing helicopters for crew exchanges on off-shore locations.* Annals of Operations Research, 1998. 76: p. 261-286.
192. Archetti, C., R. Mansini y M.G. Speranza, *Complexity and reducibility of the skip delivery problem.* Transportation Science, 2005. 39(2): p. 182-187.

193. Archetti, C., M. Savelsbergh y M. Speranza, *Worst-case analysis for split delivery vehicle routing problems.* Transportation Science, 2006. 40(2): p. 226-234.
194. Belenguer, J.M., M.C. Martinez y E. Mota, *A lower bound for the split delivery vehicle routing problem.* Operations Research, 2000. 48(5): p. 801-810.
195. Gendreau, M. y X. Louveaux, *The Integer L-shaped method for stochastic integer programs with complete recourse.* Operations Research Letters, 2003. 13: p. 133-142.
196. Archetti, C., M.G. Speranza y A. Hertz, *A tabu search algorithm for the split delivery vehicle routing problem.* Transportation Science, 2006. 40(1): p. 64-73.
197. Tillman, F., *The multiple terminal delivery problem with probabilistic demands.* Transportation Science, 1969. 3: p. 192-204.
198. Stewart, W.R. y B.L. Golden, *STOCHASTIC VEHICLE-ROUTING - A COMPREHENSIVE APPROACH.* European Journal of Operational Research, 1983. 14(4): p. 371-385.
199. Dror, M. y P. Trudeau, *STOCHASTIC VEHICLE-ROUTING WITH MODIFIED SAVINGS ALGORITHM.* European Journal of Operational Research, 1986. 23(2): p. 228-235.
200. Wilson, H., et al., *Scheduling algorithms for dial-a-ride systems.* . 1971, Urban system Laboratory, MIT. Technical Report USL TR-70-13: Cambridge, MA.
201. Wilson, H. y N. Colvin, *Computer control of the Rochester dial-a-ride system.* 1977, Departament of Civil Engineering, MIT. Technical Report R-77-31: Cambridge, MA.
202. Wilson, H. y H. Weissberg, *Advanced dial-a-ride algorithms research project: Final Report.* 1977, Departament of Civil Engineering. MIT. Technical Report R-76-20: Cambridge, MA.
203. Toth, P. y D. Vigo, *Heuristic algorithms for the handicapped persons trasnportation problem.* Transportation Science, 1997. 31: p. 60-71.
204. Kohl, N., et al., *2-path cuts for the vehicle routing problem with time windows.* Transportation Science, 1999. 33(1): p. 101-116.
205. du Merle, O., et al., *Stabilized column generation.* Discrete Mathematics, 1999. 194(1-3): p. 229-237.
206. Knight, K.W. y J.P. Hofer, *VEHICLE SCHEDULING WITH TIMED AND CONNECTED CALLS - A CASE STUDY.* Operational Research Quarterly, 1968. 19(3): p. 299-&.
207. Madsen, O.B.G., *Optimal scheduling of trucks- A routing problem with tight due times for delivery*, en *Optimization applied to transportation systems*, H. Strobel, R. Genser, y M.Etschmaier, Editors. 1976, IIASA, International Institute for Applied System analysis: Laxenburgh, Austria. p. 126-136.
208. Pullen, H.G.M. y M.H.J. Webb, *A COMPUTER APPLICATION TO A TRANSPORT SCHEDULING PROBLEM.* Computer Journal, 1967. 10(1): p. 10-&.
209. Desrosiers, J., et al., *ROUTING WITH TIME WINDOWS BY COLUMN GENERATION.* Networks, 1984. 14(4): p. 545-565.
210. Kolen, A.W.J., A. Kan y H. Trienekens, *VEHICLE-ROUTING WITH TIME WINDOWS.* Operations Research, 1987. 35(2): p. 266-273.
211. Desrochers, M., J. Desrosiers y M. Solomon, *A NEW OPTIMIZATION ALGORITHM FOR THE VEHICLE-ROUTING PROBLEM WITH TIME WINDOWS.* Operations Research, 1992. 40(2): p. 342-354.
212. Cook, W. y J.L. Rich, *A parallel cutting plane algorithm for the vehicle routing problem with time windows. Technical Report* 1999, Computational Applied Mathematics: Rice University, Houston TX.
213. Homberger, J. y H. Gehring, *Two evolutionary metaheuristics for the vehicle routing problem with time windows.* Infor, 1999. 37(3): p. 297-318.
214. Kallehauge, B., *Formulations and exact algorithms for the vehicle routing problem with time windows.* Computers and Operations Research, 2008. 35(7): p. 2307-2330.
215. Kallehauge, B., N. Boland y O.B.G. Madsen, *Path inequalities for the vehicle routing problem with time windows.* Networks, 2007. 49(4): p. 273-293.
216. Solomon, M.M., *ALGORITHMS FOR THE VEHICLE-ROUTING AND SCHEDULING PROBLEMS WITH TIME WINDOW CONSTRAINTS.* Operations Research, 1987. 35(2): p. 254-265.
217. Laporte, G. y Y. Norbert, *Exact algorithms for the vehicle routing problem.* Annals of Discrete Mathematics, 1987. 31: p. 147-184.

218. Laporte, G., *The vehicle routing problem: an overview of exact and approximate agorithms.* European Journal of Operational Research, 1992. 59: p. 345-358.
219. Gaskell, T.J., *BASES FOR VEHICLE FLEET SCHEDULING.* Operational Research Quarterly, 1967. 18(3): p. 281-&.
220. Yellow, P.C., *A COMPUTATIONAL MODIFICATION TO SAVINGS METHOD OF VEHICLE SCHEDULING.* Operational Research Quarterly, 1970. 21(2): p. 281-&.
221. Desrochers, M. y T. Verhoog, *A matching based saving algorithm for the vehicle routing problem. Technical Reports Cahiers du GERAD G-89-04*. 1989, École des Hautes Études Commerciales de Montreal.
222. Altinkemer, K. y B. Gavish, *PARALLEL SAVINGS BASED HEURISTICS FOR THE DELIVERY PROBLEM.* Operations Research, 1991. 39(3): p. 456-469.
223. Wark, P. y J. Holt, *A REPEATED MATCHING HEURISTIC FOR THE VEHICLE ROUTING PROBLEM.* Journal of the Operational Research Society, 1994. 45(10): p. 1156-1167.
224. Wren, A. y A. Holliday, *COMPUTER SCHEDULING OF VEHICLES FROM ONE OR MORE DEPOTS TO A NUMBER OF DELIVERY POINTS.* Operational Research Quarterly, 1972. 23(3): p. 333-&.
225. Bramel, J. y D. Simchi-Levi, *A LOCATION BASED HEURISTIC FOR GENERAL ROUTING-PROBLEMS.* Operations Research, 1995. 43(4): p. 649-660.
226. Beasley, J.E., *ROUTE 1ST - CLUSTER 2ND METHODS FOR VEHICLE-ROUTING.* Omega-International Journal of Management Science, 1983. 11(4): p. 403-408.
227. Balinski, M. y R. Quandt, *On an Integer program for a delivery problem.* Operations Research, 1964. 12: p. 300-304.
228. Foster, B.A. y D.M. Ryan, *INTEGER PROGRAMMING APPROACH TO VEHICLE SCHEDULING PROBLEM.* Operational Research Quarterly, 1976. 27(2): p. 367-384.
229. Ryan, D.M., C. Hjorring y F. Glover, *EXTENSIONS OF THE PETAL METHOD FOR VEHICLE ROUTING.* Journal of the Operational Research Society, 1993. 44(3): p. 289-296.
230. Renaud, J., F.F. Boctor y G. Laporte, *An improved petal heuristic for the vehicle routing problem.* Journal of the Operational Research Society, 1996. 47(2): p. 329-336.
231. Bodin, L., et al., *SPECIAL ISSUE - ROUTING AND SCHEDULING OF VEHICLES AND CREWS - THE STATE OF THE ART.* Computers & Operations Research, 1983. 10(2): p. 1-&.
232. Mole, R.H. y S.R. Jameson, *SEQUENTIAL ROUTE-BUILDING ALGORITHM EMPLOYING A GENERALIZED SAVINGS CRITERION.* Operational Research Quarterly, 1976. 27(2): p. 503-511.
233. Christofides, N., A. Mingozzi y P. Toth, *The vehicle Routing Problem*, en *Combinatorial Optimization*, Wiley, Editor. 1979: Chichester. p. 315-338.
234. Lin, S., *COMPUTER SOLUTIONS OF TRAVELING SALESMAN PROBLEM.* Bell System Technical Journal, 1965. 44(10): p. 2245-+.
235. Johnson, D. y L. McGeoch, *The Traveling Salesman Problem: a case study in local optimization*, en *Local Search in Combinatorial Optimization*. 1997, John Wiley&Sons. p. 215-310.
236. Renaud, J., F.F. Boctor y G. Laporte, *A fast composite heuristic for the symmetric travelling salesman problem.* INFORMS Journal on Computing, 1996. 8: p. 134-143.
237. Lin, S. y Kernigha.Bw, *EFFECTIVE HEURISTIC ALGORITHM FOR TRAVELING-SALESMAN PROBLEM.* Operations Research, 1973. 21(2): p. 498-516.
238. Or, I., *Traveling salesman-type combinatorial problems and their relation to the logistic of regional blood banking*, en *Ph.D. dissertation*. 1976: Departament of Industrial Engineering and Management Sciences. Northwestern University, Evanston, IL.
239. Van Breedam, A., *Improvement heuristics for the Vehicle Routing Problem based on simulated annealing.* European Journal of Operational Research, 1995. 86(3): p. 480-490.
240. Gendreau, M., A. Hertz y G. Laporte, *NEW INSERTION AND POSTOPTIMIZATION PROCEDURES FOR THE TRAVELING SALESMAN PROBLEM.* Operations Research, 1992. 40(6): p. 1086-1094.
241. Thompson, P.M. y H.N. Psaraftis, *CYCLIC TRANSFER ALGORITHMS FOR MULTIVEHICLE ROUTING AND SCHEDULING PROBLEMS.* Operations Research, 1993. 41(5): p. 935-946.

242. Laporte, G., *What you should know about the vehicle routing problem.* Naval Research Logistics, 2007. 54(8): p. 811-819.
243. Cordeau, J.F., et al., *New heuristics for the vehicle routing problem*, en *Logistics systems and optimization*, A. Langevin y D.Riopel, Editors. 2005, Springer: New York. p. 279-297.
244. Gendreau, M., et al., *Meta-heuristics for the vehicle routing problem and its extensions: A categorized bibliography*, en *The vehicle routing problem: Latest advances and challenges*, S. Raghavan y E.A. Wasil, Editors, Springer: Boston. p. In press.
245. Taillard, E.D., *Parallel iterative search methods for vehicle routing problems.* Networks, 1993. 23: p. 661-673.
246. Xu, J. y J.P. Kelly, *A network flow-based Tabu Search heuristic for the vehicle routing problem.* Transportation Science, 1996. 30(4): p. 379-393.
247. Glover, C. y R. Mukkamala. *Multilevel secure databases: A new approach*. en *Conference Proceedings - IEEE SOUTHEASTCON*. 1991.
248. Toth, P. y D. Vigo, *The granular tabu search and its application to the vehicle-routing problem.* INFORMS Journal on Computing, 2003. 15(4): p. 333-346.
249. Cordeau, J.F., G. Laporte y A. Mercier, *A unified tabu search heuristic for vehicle routing problems with time windows.* Journal of the Operational Research Society, 2001. 52(8): p. 928-936.
250. Prins, C., *A simple and effective evolutionary algorithm for the vehicle routing problem.* Computers & Operations Research, 2004. 31(12): p. 1985-2002.
251. Tarantilis, C.D. y C.T. Kiranoudis, *BoneRoute: An adaptive memory-based method for effective fleet management.* Annals of Operations Research, 2002. 115(1-4): p. 227-241.
252. Ghaziri, H., *Solving routing problem by a self-organizing map*, en *Artificial Neural networks*, T. Kohonen, et ál., Editors. 1991: North-Holland, Amsterdam. p. 829-834.
253. Schumann, M. y R. Retzko, *Self-organizing maps for vehicle routing problems-minimizing an explicit cost function*, en *Proceedings of the International Conference on Artificial Neural Networks*, F. Fogelman-Soulie, Editor. 1995: Paris. p. 401-406.
254. Li, F.Y., B. Golden y E. Wasil, *Very large-scale vehicle routing: new test problems, algorithms, and results.* Computers & Operations Research, 2005. 32(5): p. 1165-1179.
255. Pisinger, D. y S. Ropke, *A general heuristic for vehicle routing problems.* Computers & Operations Research, 2007. 34(8): p. 2403-2435.
256. Derigs, U. y R. Kaiser, *Applying the attribute based hill climber heuristic to the vehicle routing problem.* European Journal of Operational Research, 2007. 177(2): p. 719-732.
257. Kytojoki, J., et al., *An efficient variable neighborhood search heuristic for very large scale vehicle routing problems.* Computers & Operations Research, 2007. 34(9): p. 2743-2757.
258. Mester, D. y O. Braysy, *Active guided evolution strategies for large-scale vehicle routing problems with time windows.* Computers & Operations Research, 2005. 32(6): p. 1593-1614.
259. Mester, D. y O. Braysy, *Active-guided evolution strategies for large-scale capacitated vehicle routing problems.* Computers & Operations Research, 2007. 34(10): p. 2964-2975.
260. Nagata, Y., *Edge assembly crossover for the capacitated vehicle routing problem.* Evolutionary Computation in Combinatorial Optimization, Proceedings, 2007. 4446: p. 142-153.
261. Reimann, M., K. Doerner y R.F. Hartl, *D-Ants: Savings Based Ants divide and conquer the vehicle routing problem.* Computers & Operations Research, 2004. 31(4): p. 563-591.
262. España, G.d., *Ley 7/85 de 2 de Abril, Reguladora de las Bases de Régimen Local.* 1985.
263. Bhat, V.N., *A model for the optimal allocation of trucks for solid waste management.* Waste Management & Research, 1996. 14(1): p. 87-96.
264. Bautista, J., E. Fernandez y J. Pereira, *Solving an urban waste collection problem using ants heuristics.* Computers & Operations Research, 2008. 35(9): p. 3020-3033.
265. Rural_y_Marítimo, M.d.M.A.y. *Perfil Ambiental de España*. 2006 [cited; Available from: http://www.mma.es/portal/secciones/calidad_contaminacion/indicadores_ambientales/. (última consulta 29 de enero de 2010).
266. Gottinger, H.W., *A computational model for solid waste management with application.* European Journal of Operational Research, 1988. 35(3): p. 350-364.
267. MacDonald, M.L., *Solid waste management models: a state of the art review.* The Journal of resource management and technology, 1996. 23(2): p. [d]73-83.

268. Berger, C., G. Savard y A. Wizere, *EUGENE: an optimization model for integrated regional solid waste management planning.* International Journal of Environment and Pollution, 1999. 12(2-3): p. 280-307.
269. Tanskanen, J.H., *Strategic planning of municipal solid waste management.* Resources, Conservation and Recycling, 2000. 30(2): p. 111-133.
270. Baetz, B.W. y A.W. Neebe, *A PLANNING-MODEL FOR THE DEVELOPMENT OF WASTE MATERIAL RECYCLING PROGRAMS.* Journal of the Operational Research Society, 1994. 45(12): p. 1374-1384.
271. Everett, J.W. y A.R. Modak, *Optimal regional scheduling of solid waste systems .1. Model development.* Journal of Environmental Engineering-Asce, 1996. 122(9): p. 785-792.
272. Sundberg, J., P. Gipperth y C.O. Wene, *A systems approach to municipal solid waste management: A pilot study of Goteborg.* Waste Management and Research, 1994. 12(1): p. 73-91.
273. Morrissey, A.J. y J. Browne, *Waste management models and their application to sustainable waste management.* Waste Management, 2004. 24(3): p. 297-308.
274. Gelders, L. y D. Cattrysse, *Public wasrw collection: a case study.* Belgian Journal of Operations Research Statistics and Computer Science, 1991. 31: p. 5-15.
275. Ulusoy, G., *THE FLEET SIZE AND MIX PROBLEM FOR CAPACITATED ARC ROUTING.* European Journal of Operational Research, 1985. 22(3): p. 329-337.
276. Golden, B.L., J.S. Dearmon y E.K. Baker, *COMPUTATIONAL EXPERIMENTS WITH ALGORITHMS FOR A CLASS OF ROUTING-PROBLEMS.* Computers & Operations Research, 1983. 10(1): p. 47-59.
277. Pearn, W.L., *APPROXIMATE SOLUTIONS FOR THE CAPACITATED ARC ROUTING PROBLEM.* Computers & Operations Research, 1989. 16(6): p. 589-600.
278. Kulcar, T., *Optimizing solid waste collection in Brussels.* European Journal of Operational Research, 1996. 90(1): p. 71-77.
279. Chapleau, L., et al., *A PARALLEL INSERT METHOD FOR THE CAPACITATED ARC ROUTING PROBLEM.* Operations Research Letters, 1984. 3(2): p. 95-99.
280. Mourao, M.C. y M.T. Almeida, *Lower-bounding and heuristic methods for a refuse collection vehicle routing problem.* European Journal of Operational Research, 2000. 121(2): p. 420-434.
281. Benavent, E. y D. Soler, *The directed rural postman problem with turn penalties.* Transportation Science, 1999. 33(4): p. 408-418.
282. Amberg, A., W. Domschke y S. Voss, *Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees.* European Journal of Operational Research, 2000. 124(2): p. 360-376.
283. Ghiani, G., G. Improta y G. Laporte, *The capacitated arc routing problem with intermediate facilities.* Networks, 2001. 37(3): p. 134-143.
284. Egese, R.W., *Routing Winter Gritting Vehicles.* Discrete Applied Mathematics, 1994. 48(3): p. 231-244.
285. Hertz, A., G. Laporte y P.N. Hugo, *Improvement procedures for the undirected rural postman problem.* INFORMS Journal on Computing, 1999. 11(1): p. 53-62.
286. Corberán, A., R. Marti y A. Romero, *Heuristics for the Mixed Rural Postman Problem.* Computers and Operations Research, 2000. 27(2): p. 183-203.
287. Lacomme, P., C. Prins y W. Ramdane-Cherif, *Competitive memetic algorithms for arc routing problems.* Annals of Operations Research, 2004. 131(1-4): p. 159-185.
288. Lacomme, P., C. Prins y M. Sevaux, *A genetic algorithm for a bi-objective capacitated arc routing problem.* Computers & Operations Research, 2006. 33(12): p. 3473-3493.
289. Kim, B.I., S. Kim y S. Sahoo, *Waste collection vehicle routing problem with time windows.* Computers and Operations Research, 2006. 33(12): p. 3624-3642.
290. Sahoo, S., et al., *Routing optimization for Waste Management.* Interfaces, 2005. 35(1): p. 24-36.
291. Amponsah, S.K. y S. Salhi, *The investigation of a class of capacitated arc routing problems: The collection of garbage in developing countries.* Waste Management, 2004. 24(7): p. 711-721.
292. España, I.T.G.d., *Atlas del medio hídrico de la provincia de Burgos*. 1998, Madrid.
293. AMBIENTE, C.D.M., *PLAN REGIONAL DE ÁMBITO SECTORIAL DE RESIDUOS URBANOS Y RESIDUOS DE ENVASES DE CASTILLA Y LEÓN 2004-2010*, J.D.C.Y. LEÓN, Editor. 2005.

294. Bentley, J.L., *Fast Algorithms for Geometric Salesman Problems.* ORSA Journal of Computing, 1992. 4: p. 387-411.
295. Williams, J.W.J., *ALGORITHM-232 - HEAPSORT.* Communications of the Acm, 1964. 7(6): p. 347-348.
296. Correa, C., R. Bolaños y A. Molina, *ALGORITMO MULTIOBJETIVO NSGA-II APLICADO AL PROBLEMA DE LA MOCHILA.* Scientia et Technica, 2008. 39: p. 206-211.