LOCALIZACIÓN DE CENTROS DE SERVICIO. APLICACIÓN A LA PROVINCIA DE BURGOS

ALEGRE MARTINEZ, JESÚS FRANCISCO
Departamento de Economía Aplicada
Universidad de Burgos
correo-e: jfalegre@ubu.es

CASADO YUSTA, SILVIA
Departamento de Economía Aplicada
Universidad de Burgos
correo-e: scasado@ubu.es

PACHECO BONROSTRO, JOAQUÍN
Departamento de Economía Aplicada
Universidad de Burgos
correo-e: jpacheco@ubu.es

RESUMEN

En este trabajo se resuelven dos problemas de localización de centros de servicio. Estos son el conocido problema del *p*-centro y el Maximun Set Covering Problem (MSCP), muy relacionado con problemas de cubrimiento de conjuntos. Para ello se propone un algoritmo basado en la estrategia metaheurística búsqueda dispersa (scatter search). El algoritmo scatter search propuesto incorpora diferentes estrategias, como búsqueda local, GRASP y path relinking. El objetivo es obtener soluciones de forma eficaz para un número bajo de centros de servicio. Se realizan una serie experiencias computacionales y se muestran aplicaciones con datos reales a la localización de recursos sanitarios en la provincia de Burgos.

Palabras clave: *localización, p-centro, tiempo crítico, Scatter Search*

1.- Introducción

El problema del *p*-centro es un problema de localización bien conocido. En Kariv y Hakimi (1979) se demostró que es NP-Hard. Consiste en colocar *p* facilidades y asignar clientes a dichas facilidades (cada facilidad puede servir a diferente número de clientes) de forma que se minimice la máxima distancia entre un cliente y su facilidad.

Sea $U = \{u_1, u_2, \ldots, u_m\}$ un conjunto de usuarios y $V = \{v_1, v_2, \ldots, v_n\}$ un conjunto de localizaciones donde colocar facilidades. Considérese conocida la distancia d_{ij} entre cada cliente u_i y la localización v_j , el problema consiste en encontrar un subconjunto X de p localizaciones de forma que se minimice

$$\max_{i=1..m}\left\{\min_{v_j \in X} d_{ij}\right\}$$

El problema se puede formular de forma lineal como sigue

Minimizar z

sujeto a:

$$\sum_{j=1..n} x_{ij} = 1, \qquad i = 1..m; \qquad (1)$$

$$x_{ij} \le y_j, \qquad i = 1..m; j = 1..n; \qquad (2)$$

$$\sum_{j=1..n} y_j = p; \qquad (3)$$

$$\sum_{j=1..n} d_{ij} x_{ij} \le z \qquad i = 1..m; \qquad (4)$$

$$x_{ij}, y_j \in \{0,1\} \qquad i = 1..m; j = 1..n; \qquad (5)$$

donde $y_j = 1$ indica que se ha colocado una facilidad en v_j (0 en caso contrario); $x_{ij} = 1$ indica que al usuario u_i se le ha asignado la facilidad v_j (0 en caso contrario).

Este modelo es usado por ejemplo en localización de estaciones de bomberos, policía o ambulancias, unidades de urgencias, etc. Centrándonos en los recursos sanitarios hay ocasiones en donde más importante que minimizar la distancia del usuario más alejado a su facilidad más cercana, es maximizar el número de usuarios potenciales que serían atendidos en un tiempo aceptable (‘tiempo crítico’). Supóngase más concretamente que se quieren poner en una determinada zona o región unas unidades especiales de atención

a diabéticos; ante un ataque por bajada en el nivel de insulina se sabe que el enfermo no debe permanecer más de 20' sin ser atendido, de lo contrario se corre el riesgo de que los daños puedan ser irreversibles. En este caso interesa que la mayor parte de los diabéticos estén a menos de 20' de su unidad más cercana.

Esto sugiere plantear un nuevo modelo que puede ser expresado como el problema del Maximum Set Covering (ver Love, Morris y Wesolowsky, 1988). Considérese que $U = \{u_1, u_2, \ldots, u_m\}$ es ahora un conjunto de localidades de una determinada región o zona, cada una de ellas con una población q_i, i=1..m, de demandantes potenciales de un determinado servicio o facilidad (por ejemplo unidad de atención a diabéticos); V sigue siendo el conjunto de localizaciones donde ubicar p facilidades; considérese que la matriz d_{ij} es una matriz de tiempos y el tiempo máximo en atender a los clientes no conviene que supere una cantidad *t_crítico*. Se trata de encontrar el subconjunto $X \subset V$ de tamaño p de forma que se minimice

$$\sum_i q_i \cdot r_i$$

sujeto a: (3)

$$\sum_{j/d_{ij} \le t\_critico} y_j + r_i \ge 1 \qquad i = 1..m;$$

$$y_j \in \{0,1\} \qquad j = 1..n;$$

$$r_i \in \{0,1\} \qquad i = 1..m; \qquad (8)$$

donde r_i toma el valor de 1 si la facilidad que le corresponde a u_i dista más de *t_crítico*, y 0 en caso contrario. Obsérvese que (8) se podría sustituir por

$$r_i \in (0,1) \qquad i = 1..m. \qquad (8')$$

Obsérvese que aunque el MSCP es un problema de maximización se minimiza la función objetivo. Así se consigue minimizar el número de pacientes que no son atendidos en un tiempo igual o inferior al *t_critico*.

Este trabajo se inserta en un proyecto en el que se quiere desarrollar un sistema para analizar las localizaciones donde se deben colocar determinados recursos sanitarios

especiales (unidades de quemados, geriátricos, diabetes ...) en las provincias de Castilla y León. Se consideran 2 criterios, el económico (determinado por el número de unidades que se abren), y el 'social' determinado por diferentes funciones objetivos según el tipo de unidad o servicio. Por ejemplo, si se trata de una unidad de urgencia, el objetivo es minimizar la máxima distancia de cada cliente a su unidad más cercana; si es una unidad de diabetes interesa, según se ha comentado, maximizar el número de pacientes que podrían ser atendidos en un determinado tiempo.

Por tanto, para cada tipo de unidad, nos enfrentamos a un problema bi-criterio: económico (número de unidades) y social. Se trata de desarrollar un sistema que aporte soluciones que se aproximen lo más posible a la curva de eficiencia. De esta forma el responsable o decisor podrá elegir la que estime más conveniente. Por otra parte el número de unidades a añadir de cada tipo va a ser pequeño: nunca superior a 10 en cada provincia. Por tanto se ha creído conveniente que el sistema resuelva los problemas mono-objetivo correspondientes a diferentes números de unidades añadir (desde $p = 1$ hasta 10). Esta es la razón por la que en este trabajo desarrollamos sendos algoritmos para estos 2 modelos de localización, *p*-centro y MSC, correspondientes a los 2 objetivos sociales considerados. Nuestro objetivo es que estos algoritmos sean especialmente eficaces para bajos valores de *p*.

En un trabajo reciente, Mladenović y otros (2001), adaptan al problema del *p*-centro varios de los heurísticos más clásicos para el problema de las *p*-medianas, (ver Hansen y Mladenović (1997), Whitaker (1983)), así como diferentes tipos de movimientos y estructuras vecinales (ver Mladenović y otros (1995) y (1996)). También proponen 2 algoritmos basados en las estrategias *búsqueda por entornos variables* y *búsqueda tabú*. En la literatura se ha encontrado una referencia importante del problema del Maximum Set Covering con el planteamiento realizado en este trabajo (Resende, 1998).

En este trabajo se va a proponer un algoritmo de tipo *búsqueda dispersa* (scatter search, SS) (Glover (1998), Glover, Laguna y Martí (2000), Laguna (2002) y Laguna y Martí (2003)) para el problema del *p*-centro. Este algoritmo incorpora diferentes elementos basados en estrategias GRASP (Feo and Resende (1989), Feo y Resende (1995), Pitsoulis y Resende (2002)), búsqueda local y path relinking (Glover y Laguna (1997) y Laguna y Martí (2003)). Se va a comprobar la eficacia de este algoritmo para instancias con bajos valores de *p* comparándolo con otras estrategias recientes.

Concretamente se van a usar las instancias de la conocida librería OR-Lib (Beasley, 1990). Así mismo, este algoritmo scatter search y sus elementos van a ser adaptados para dar lugar a un algoritmo para el MSCP. Para evaluar la eficacia de esta nueva estrategia, también para pequeños valores de p, se van a hacer diferentes pruebas con adaptaciones de estas instancias. Se van a comparar los resultados obtenidos por nuestra estrategia scatter search con los obtenidos por implementaciones propias de las adaptaciones a este modelo de otros algoritmos para el problema del p-centro. Así mismo se van a analizar una serie de instancias basadas en datos reales considerando ambos modelos. Estos datos reales son referidos a la provincia de Burgos, (Norte de España), concretamente a estimaciones de casos de diabetes.

El trabajo se estructura de la forma siguiente: en el siguiente apartado se describen los heurísticos más clásicos para el problema del p-centro de los que haremos uso más adelante, así como metaheurísticos más recientes; en el apartado 3 se describe el algoritmo scatter search propuesto para este modelo, así como sus elementos; en el apartado 4 se describen las adaptaciones realizadas para obtener un algoritmo para el problema del Maximum Set Covering; en el apartado 5 se muestran los resultados computacionales para ambos modelos considerando las instancias ficticias; en el apartado 6 se describen los datos reales, y se muestran las soluciones para estas instancias reales; finalmente en el apartado 7 se muestran las conclusiones.

2.- Principales Heurísticos para el problema del p-centro

En Mladenović y otros (2001) se hacen adaptaciones al problema del p-centro de los 3 heurísticos clásicos que desarrollaron Mulvey y Beck (1984) para el problema de las p-medianas. Estos 3 heurísticos son un método constructivo (*Greedy)* y 2 procedimientos de mejora, *Alternate* e *Interchange*. Para describirlos brevemente considérese un problema definido por m, n, d y p. (En adelante para abreviar identificaremos U y V directamente con los índices, es decir $U = \{1, 2, ..., m\}$ y $V = \{1, 2, ..., n\}$). Así mismo denotamos por X la solución, total o parcial (solución parcial si X tiene menos de p elementos y solución total cuando tiene p elementos), en cada momento, es decir, las localizaciones (índices) donde se han colocado facilidades, y f el

valor de la función objetivo correspondiente a X. Los heurísticos se pueden describir como sigue:

Procedimiento Greedy

Hacer $X = \varnothing$

Mientras $|X| < p$ *hacer*

- *Determinar la localización j* que más baje el valor de f si se le añade una facilidad*
- *Hacer* $X = X \cup \{j^*\}$

Procedimiento Alternate

Repetir

- *Para cada facilidad j de X, determinar el subconjunto de los puntos de U que tienen a j como facilidad más cercana*
- *Para cada uno de estos subconjuntos de usuarios resolver el problema del 1-centro*
- *Hacer X' el conjunto de soluciones de estos p problemas, y f' su valor*
- *Si* $f' < f$ *hacer* $X = X'$ *y* $f = f'$

hasta que no haya cambios en X

Procedimiento Interchange

Repetir

- *Para cada* $j \in V\text{-}X$ *y* $k \in X$ *determinar el valor de la función objetivo* v_{jk} *si se cambiara la facilidad de k a j*
- *Determinar* $v_{j^*k^*} = \min \{ v_{jk} / j \in V\text{-}X \text{ y } k \in X \}$
- *Si* $v_{j^*k^*} < f$ *entonces hacer* $X = X\text{-}\{k^*\}$, $X = X \cup \{j^*\}$ *y* $f = v_{j^*k^*}$

hasta que no haya mejora

En estos algoritmos, al igual que en los que se describen a continuación, se usa una variable auxiliar fundamental, $c_1(i)$, que indica para cada usuario i, i=1..m, la

localización de la facilidad más cercana en la solución actual. Además en Mladenović y otros (2001), en este último procedimiento se hace uso de la siguiente propiedad: sea i^* el usuario que define el valor de f, es decir, $d_{i^* c1(i^*)} = f$, ('usuario crítico') cualquier movimiento, definido por $j \in V\text{-}X$ y $k \in X$, que reduzca el valor de función objetivo ($v_{jk} < f$) debe verificar que $d_{i^*j} < d_{i^* c1(i^*)}$. Esto hace reducir la búsqueda en cada iteración y acelerar el procedimiento.

Además en Mladenović y otros (2001) se definen 2 heurísticos basados en las estrategias *búsqueda tabú* (Tabu Search, TS) y *búsqueda por entornos variables* (Variable Neigborhood Search, VNS).

En el procedimiento de búsqueda tabú se usan dos listas tabú correspondientes a los elementos que acaban de entrar en la solución (*L_in*) y a los que acaban de salir (*L_out*). Obsérvese que se aprovecha la propiedad del cliente crítico cuando se encuentra un buen camino (mejora de la mejor solución). El procedimiento TS se describe brevemente como sigue.

Procedimiento Búsqueda_Tabú

Leer Solución inicial X, f, y hacer X_best = X, f_best = f, mejora = TRUE
Iniciar Listas Tabú, L_in = ∅, L_out = ∅
Repetir

- *Determinar i* el cliente crítico correspondiente a X*
- *Hacer* $J = \{ j \in V - X - L\_out / \ d_{i^*j} < f \}$
- *Si (J = ∅) o (not mejora) entonces* $J = \{ j \in V - X - L\_out \}$
- $\forall k \in X\text{-}L\_in$ *y* $j \in J$ *determinar* v_{jk}
- *Determinar* $v_{j^*k^*} = min \{ v_{jk} / j \in V\text{-}X \text{ y } k \in X \}$
- *Hacer* $X = X\text{-}\{k^*\}$, $X = X \cup \{j^*\}$ *y* $f = v_{j^*k^*}$
- *Si* $f < f\_best$ *entonces hacer f_best = f, X_best = X, mejora = TRUE; sino mejora = FALSE*
- *Actualizar Listas Tabú: L_in y L_out*

hasta criterio de parada

El procedimiento VNS se puede describir como sigue

Procedimiento Busqueda_Entorno_Variable

Leer solución inicial X_best, f_best

Repetir

k = 1

Repetir

- *X = X_best, f = f_best*
- *$\forall j = 1..k$*

  Determinar i cliente crítico correspondiente a X*

  Tomar in $\in$ V-X / d(i,in) < f aleatoriamente*

  Tomar out $\in$ X aleatoriamente

  Hacer X = X-{out}, X = X $\cup$ {in} y actualizar f
- *Aplicar el procedimiento Interchange a X y f*
- *Si f < f_best entonces X_best = X, f_best = f y k = 1;*

  sino k = k+1

hasta k = kmax

hasta alcanzar criterio de parada

En ambos algoritmos el criterio de parada es un número de iteraciones sin mejora en *f_best* o un tiempo máximo de computación.

3.- Algoritmo Scatter Search

El método de solución que se ha desarrollado consiste en una adaptación de scatter search (SS). Scatter search es un ejemplo de lo que se conoce como métodos evolutivos. Pero a diferencia de otros métodos evolutivos no utiliza la aleatoriedad como principal mecanismo para buscar soluciones. Recientes tutoriales sobre scatter search se pueden encontrar en Glover (1998), Glover, Laguna y Martí (2000), Laguna (2002) y Laguna y Martí (2003).

SS está caracterizado básicamente por el uso de un *Conjunto de Referencia* (*RefSet*) de soluciones. Dicho conjunto está formado por dos subconjuntos: Subconjunto de Calidad (*RefSet1*) formado por las mejores soluciones y Subconjunto de Diversidad (*RefSet2*) formado por las soluciones más diferentes con el resto de *RefSet*. En cada

paso o ciclo se generan nuevas soluciones a partir de las del *Conjunto de Referencia*, que actualizan éste.

Se ha diseñado una versión que podríamos denominar 'estática' de esta estrategia para este problema. La descripción en seudocódigo de forma muy general es la siguiente

Procedimiento Busqueda_Dispersa_Estática

1. *Generar un conjunto inicial de soluciones con un método Generador-*
1. *Diversificador*
2. *Mejorar estas soluciones con un método de mejora*
3. *Con estas soluciones construir un RefSet inicial*
4. *Repetir*
   4.1. Obtener todos los subconjuntos de pares de soluciones del RefSet
   4.2. Combinar estos subconjuntos para obtener nuevas soluciones
   4.3. Mejorar estas nuevas soluciones con el método de mejora
   4.4. Actualizar RefSet con estas nuevas soluciones

   hasta que RefSet se estabilice (i.e. no se incluyan nuevas soluciones) {fin 4.}
5. *Si han trascurrido max_iter iteraciones (pasos 1-4) sin mejora finalizar; sino volver al paso 1 (Reinicializar)*

Se va a denotar por *n_pob* el tamaño del conjunto inicial (paso 1). Así mismo se denota por *Tam_Ref1* y *Tam_Ref2* los tamaños de *RefSet1* y *RefSet2* respectivamente.

Para formar el conjunto de Referencia, (paso 3) se comienza por *RefSet1*, i.e, por los elementos de mayor calidad según la función objetivo. Posteriormente para añadir los elementos de *RefSet2*, se usa la siguiente función o criterio que mide la 'diversidad' de una solución candidata *X* a entrar con respecto a los que ya están en *RefSet*

$$Difmin(X, RefSet) = \min \{dif(X, X') / X' \in RefSet\};$$

donde $dif(X, X') = |X\text{-}X'|$

La actualización de *RefSet* (paso 4.4.) se realiza sólo considerando la calidad de las soluciones. Es decir, se incorporan aquellas nuevas soluciones que mejoren la función objetivo de alguna de las soluciones existentes en *RefSet*. Como método de mejora se

usa la combinación de los procedimientos Alternate e Interchange aplicados en este orden (el orden se ha determinado tras la implementación de diferentes pruebas). A continuación se describen los métodos de diversificación y combinación.

3.1 Método Generador-Diversificador

Nuestro método diversificador está basado en constructivos tipo GRASP. GRASP (Greedy Randomized Adaptive Search Procedure), es una estrategia heurística que construye soluciones usando una función voraz y aleatoriedad controlada. La mayoría de las implementaciones GRASP incluyen un procedimiento de búsqueda local para mejorar las soluciones obtenidas por el método ávido-aleatorio. GRASP fue originalmente propuesto en el contexto de problemas de cubrimientos de conjuntos (Feo y Resende, 1989). Un clásico tutorial se puede encontrar en Feo y Resende (1995) y más recientemente en Pitsoulis y Resende (2002).

En nuestro caso la función voraz Δ_j en cada paso es el valor de la función objetivo que se obtendría si se añade una facilidad a *j*. El método diversificador consta de los siguientes pasos

Procedimiento Avido-Aleatorio

Hacer $X = \varnothing$

Mientras $|X| < p$ *hacer*

- *Determinar* $\forall j \in V\text{-}X\ \Delta_j$ *el valor de f si se añadiera j a X*
- *Determinar* $\Delta max = max\ \{\Delta_j / j \in V\text{-}L\}$ *y* $\Delta min = min\ \{\Delta_j / j \in V\text{-}L\}$
- *Construir* $L = \{j \in V\text{-}L / \Delta_j \leq \alpha \cdot \Delta min + (1\text{-}\alpha) \cdot \Delta max\}$
- *Elegir* $j^* \in L$ *aleatoriamente*
- *Hacer* $X = X \cup \{j^*\}$

El parámetro α ($0 \leq \alpha \leq 1$) controla el nivel de aleatoriedad. A mayor valor de α menor nivel de aleatoriedad. Con esta uso de aleatoriedad controlada se consigue una muestra de soluciones en la que normalmente la mejor de ellas supera a la encontrada con una elección totalmente determinística, (con $\alpha = 1$). Una selección adecuada de α permite un equilibrio entre diversificación y calidad de las soluciones.

La primera vez que se emplea el método generador-diversificador (paso 1) no hay 'historia' acerca de cuantas veces un elemento ha formado parte de las soluciones del conjunto de referencia. Sin embargo, esta información puede ser utilizable cuando el método se usa para reinicializar el proceso. La información se registra en el siguiente vector

$freq(j)$ = Número de veces que cada localidad j de V ha pertenecido a las soluciones del conjunto de referencia

La información registrada en $freq(j)$ se usa para modificar los valores Δ_j en el método diversificador de la siguiente manera

$$\Delta'_j = \Delta_j - \beta\Delta_{\max}\frac{freq(j)}{freq_{\max}}$$

donde $freq_{\max} = \max\{freq(j) : \forall j\}$. Con los valores modificados Δ'_j se calculan Δ'_{min} y Δ'_{max} y se ejecuta el método diversificador con estos valores para construir la lista de candidatos L. Obsérvese que con $\beta = 0$, el método diversificador modificado coincide con el original. Altos valores de β fuerzan a la selección de elementos que menos han aparecido. El uso de información de dada por la frecuencia en el método diversificador esta inspirada en Campos y otros (2001) y Ribeiro y otros (2002).

3.2 Método de Combinación

Se obtienen nuevas soluciones combinando pares de soluciones del conjunto de referencia, (paso 4.2). El número de soluciones generadas de cada par depende de la relativa calidad de las soluciones que son combinadas. Considerese x^p y x^q las soluciones del conjunto de referencia que son combinadas, donde $p < q$. Se asume que el conjunto de referencia está ordenado de forma que x^1 es la mejor solución y x^{Tam_Ref} la peor; entonces el número de soluciones generadas de cada combinación es:

3 si $p \le Tam\_Ref1$ y $q \le Tam\_Ref1$

2 si $p \le Tam\_Ref1$ y $q > Tam\_Ref1$

1 si $p > Tam\_Ref1$ y $q > Tam\_Ref1$.

Cada par de soluciones del *RefSet* se usa para generar nuevas soluciones. Para ello se usa la estrategia denominada path relinking. Path relinking es una estrategia, tradicionalmente asociada a la fase de intensificación de la búsqueda tabú (Glover y Laguna, 1997). También se ha usado en procedimientos de scatter search (Laguna y Martí, 2002) y GRASP (Laguna y Martí (1999) y Resende y Ribeiro (2003)). La idea que subyace es que en el camino entre dos buenas soluciones se espera que haya soluciones de parecida calidad (incluso en algún caso mejor). Para una mayor ilustración ver Glover y otros (2000).

El path relinking consiste en construir un camino que una las dos soluciones. Un número de pasos (soluciones) intermedios en ese camino o cadena son seleccionados como nuevas soluciones. Estas soluciones intermedias han de ser lo más equidistantes entre si. A esas soluciones intermedias se las aplica el procedimiento de mejora. La Figura 1 describe esta idea.

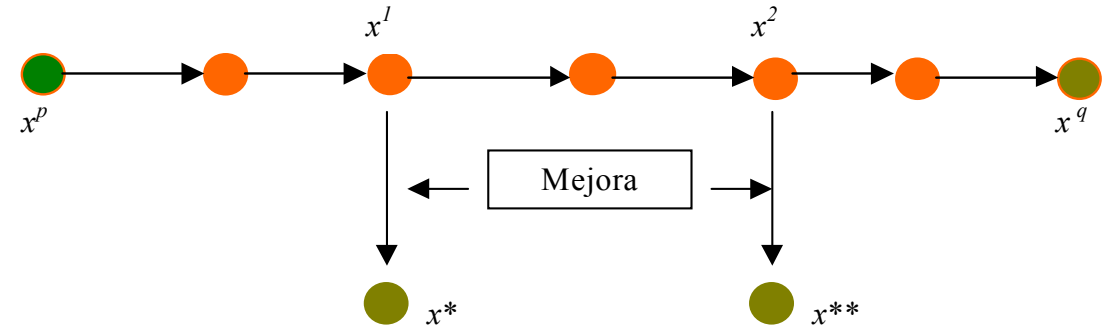

Figura 1.- *Generación de nuevas soluciones usando Path Relinking*

De cada par de soluciones en el conjunto de referencia, en la figura x^p y x^q, se construye un camino que las une. Soluciones en los pasos intermedios preseleccionados, en la figura x^1 y x^2 , son mejoradas. Así se generan nuevas soluciones (en la figura x^* y x^{**}).

El camino que une x^p y x^q se construye como sigue. Inicialmente se hace $x = x^p$. En los siguientes pasos se añade a x un elemento de x^q que no esté en x y se elimina un elemento que no esté en x^q. De esta forma la solución intermedia x en cada paso tiene un elemento más en común con x^q. En cada paso se elige el mejor entre estos posibles cambios.

4.- Adaptaciones para el problema del tiempo crítico

El algoritmo Scatter Search para el problema del MSC sigue el mismo esquema básico y estructura del expuesto para el problema del *p*-centro. Solamente se han realizado modificaciones en los procedimientos de los que hace uso. Y estas modificaciones en realidad sólo afectan a la forma de calcular la función objetivo en cada caso.

Obsérvese que en todos los procedimientos a los que se llama (Altenate, Interchange, Avido-Aleario o Path Relinking) es recomendable, según se indicó en el apartado 2, el uso de la variable auxiliar $c_1(i)$ que indican para cada usuario i, i=1..*m*, la localización de la facilidad más cercana en la solución actual (sea parcial o total). De esta forma el cálculo de la función objetivo f en el problema del *p*-centro es

$$f = \max\left\{ d_{i\,c_1(i)} / i = 1..m \right\}$$

y en el caso del problema del MSC

$$f = \sum_{i / d_{i\,c_1(i)} > t\_umbral} q_i \quad .$$

Así, por ejemplo, en el procedimiento Ávido-Aleatorio el cálculo de los valores Δ_j, el valor de *f* si se añadiera *j* a *X*, se podría realizar fácilmente con las siguientes sentencias para el problema del *p*-centro:

- $\Delta_j = 0$
- *Para i = 1..m hacer:*

  $dist = \min\left\{ d_{ij}, d_{i\,c_1(i)} \right\}$

  $\Delta_j = max\ \{\Delta_j\ ,\ dist\};$ *(*)*

en el caso del problema del MSC valdría con cambiar la sentencia (*) por

si dist > t_crítico entonces $\Delta_j = \Delta_j + q_i$.

En definitiva se requiere cambiar muy pocas líneas de código para hacer la adaptación.

5.- Experiencias Computacionales

Para contrastar la eficacia de la estrategia propuesta para ambos problemas se han realizado una serie de pruebas que se describen en los dos apartados siguientes. Todas estas pruebas han sido realizadas en un Pentium 3, 600 MHz. Las implementaciones de los algoritmos (los propuestos en Mladenović y otros (2001) y nuestro scatter search) han sido realizadas en lenguaje Pascal, con el compilador Delphi 5.0. En este apartado se usan instancias ficticias, mientras que en el siguiente se analizan casos reales.

5.1.- Pruebas para el *p*-center problem

En este caso se comparan los resultados de nuestro algoritmo scatter search (SS) con los obtenidos por los siguientes algoritmos propuestos en Mladenović y otros (2001): VNS y TS-II (tabu search con las dos listas tabú, L_in y L_out). Para ello se han usado las instancias de la librería OR-Lib para el problema de las *p*-medianas correspondientes a valores de $p \leq 10$. En estas instancias $U = V$, es decir las localizaciones donde colocar las facilidades coinciden con los usuarios. Los valores de los parámetros que usa el procedimiento scatter search son los siguientes: *n_pob* = 12, *TamRef1* = *TamRef2* = 3; $\alpha = \beta = 0{,}8$ y *max_iter* = 5 (estos valores se han determinado tras la realización de una serie de pruebas previas). A continuación se muestran los resultados

Fichero	*n*	*P*	*VNS*	*T.M.VNS*	*TS-II*	*T.M.TS*	*SS*	*T.Mejor*	*T.Total*
Pmed1	100	5	127	*0,48*	**127**	***0,08***	127	*0.11*	*6.20*
Pmed2	100	10	98	*197,40*	98	*1,48*	**98**	***1.26***	*7.96*
Pmed3	100	10	93	*8,72*	94	*27,28*	**93**	***0.49***	*7.91*
Pmed6	200	5	84	*2,92*	84	*0,12*	**84**	***0.06***	*17.91*
Pmed7	200	10	65	*5,16*	64	*45,80*	**64**	***16.31***	*37.73*
Pmed11	300	5	59	*2,36*	**59**	***0,76***	59	*7.96*	*41.85*
Pmed12	300	10	51	*114,4*	51	*28,28*	**51**	***4.29***	*46.03*
Pmed16	400	5	**47**	***0,72***	47	*0,76*	47	*2.42*	*59.21*
Pmed17	400	10	39	*203,08*	40	*6,44*	**39**	***10.49***	*81.95*
Pmed21	500	5	**40**	***1,56***	40	*34,68*	40	*2.91*	*84.42*
Pmed22	500	10	39	*94,76*	39	*9,24*	**38**	***81.12***	*192.95*
Pmed26	600	5	**38**	***3,88***	38	*6,04*	38	*4.50*	*131.11*
Pmed27	600	10	33	*102,48*	33	*81,32*	**32**	***78.60***	*256.83*
Pmed31	700	5	30	*63,32*	**30**	***1,80***	30	*2.25*	*158.51*
Pmed32	700	10	29	*67,04*	29	*374,2*	**29**	***21.42***	*224.21*
Pmed35	800	5	**30**	***3,0***	30	*387,6*	30	*4.88*	*212.50*
Pmed36	800	10	28	*3,08*	28	*30,84*	**27**	***26.47***	*302.74*
Pmed38	900	5	**29**	***7,40***	29	*13,24*	29	*11.32*	*284.24*
Pmed39	900	10	23	*142,64*	24	*38,68*	**23**	***35.54***	*352.51*

Tabla 1.- *Resultados en las instancias de OR-Lib, con $p \leq 10$*

Las columnas (*T.M.VNS*) y (*T.M.TS*) indican el tiempo de computación (en segundos) hasta que encuentran la mejor solución VNS y TS respectivamente. El tiempo de computación total máximo para estas estrategias es de 400 segundos. La última columna (*T.Total*) muestra el tiempo de computación total (en segundos) usado por SS. La penúltima columna (*T.Mejor*) muestra el tiempo de computación (en segundos) empleado por SS hasta obtener la mejor solución encontrada. Para cada instancia, se ha señalado en negrita la estrategia que alcanza la mejor solución, o la más rápida en caso de empate.

En la tabla 1 se observa como nuestra estrategia scatter search en todos los casos alcanza la mejor solución o iguala la mejor solución obtenida por Mladenović y otros (2001). SS es la mejor estrategia al estar señalada en negrita (tabla 1) en 12 de los 19 casos frente a VNS que está en 5 y TS en 3. Por tanto, podemos concluir que SS encuentra soluciones de la misma calidad que las otras estrategias pero de una forma más rápida.

5.2.- Pruebas para el problema del Maximum Set Covering

En este caso, para medir la calidad de las soluciones obtenidas por nuestro algoritmo SS, se obtiene una cota inferior ejecutando el paquete de programación CPLEX MIP 8.0 durante dos horas para cada instancia. Dado que no existen librerías de instancias para este modelo se han tomado las mismas de la subsección 5.1., definiendo en todos los casos *t_crítico* de la siguiente manera: *t_crítico* = *Round*(*Solucion p-centro* /1.5); donde *Solución p-centro* es la mejor solución conocida previamente para el problema del *p*-centro, y *Round* es la función que redondea al entero más próximo. Por su parte q_i se genera aleatoria y uniformemente entre 1 y 100 (Estas instancias están disponibles para los lectores interesados).

Los valores de los parámetros que ha usado SS son los mismos que se han usado para el problema del *p*-centro: *n_pob* = 12, *TamRef1* = *TamRef2* = 3, $\alpha = \beta = 0{,}8$ y *max_iter* = 5. A continuación se muestra una tabla con los siguientes resultados: La cota inferior obtenida por CPLEX MIP 8.0 (*CPLEX CI*), el valor de la solución encontrada por SS (*Valor*), el porcentaje de población que no es atendida en un tiempo igual o inferior al *t_critico (Pob.Out)*, el porcentaje de desviación respecto de la mejor cota inferior

(*CIDev*), el tiempo de cálculo hasta encontrar la mejor solución, (*T.Mejor*), y tiempo total de cálculo (*T.Total)* para el algoritmo SS (ambos tiempos en segundos).

Fichero Or Lib	*n*	*p*	*CPLEX CI*	*SS*				
				Valor	*Pop. Out*	*CIdev*	*T.Mejor*	*T.Total*
Pmed1	100	5	965	967	19.62	*0.2*	*0.03*	*0.88*
Pmed2	100	10	845	845	17.14	*0*	*0.13*	*1.75*
Pmed3	100	10	878	880	17.86	*0.22*	*0.01*	*1.69*
Pmed6	200	5	1399	1406	14.28	*0.49*	*0.33*	*6.58*
Pmed7	200	10	1274	1283	13.03	*0.70*	*0.22*	*11.8*
Pmed11	300	5	1780	1780	12.3	*0*	*0.11*	*87.84*
Pmed12	300	10	1474	1474	10.18	*0*	*0.16*	*41.36*
Pmed16	400	5	2251	2263	11.33	*0.53*	*0.21*	*65.52*
Pmed17	400	10	2599	2609	13.07	*0.38*	*5.63*	*29.82*
Pmed21	500	5	3289	3318	13.36	*0.88*	*5.70*	*43.13*
Pmed22	500	10	3316	3322	13.38	*0.18*	*3.24*	*34.61*
Pmed26	600	5	2982	2987	9.95	*0.17*	*0.83*	*389.15*
Pmed27	600	10	3919	3941	13.13	*0.56*	*0.85*	*207.14*
Pmed31	700	5	5536	5576	15.78	*0.72*	*1.05*	*195.7*
Pmed32	700	10	4901	4943	13.99	*0.86*	*5.80*	*111.12*
Pmed35	800	5	2897	2910	7.28	*0.45*	*19.62*	*178.13*
Pmed36	800	10	4657	4702	11.76	*0.96*	*26.22*	*222.48*
Pmed38	900	5	3046	3075	6.85	*0.95*	*3.24*	*399.84*
Pmed39	900	10	5739	5780	12.88	*0.71*	*2.14*	*315.01*

Tabla 2.- *Resultados para el problema del MSC.*

Se observa como la calidad de las soluciones que obtiene SS es aceptable, dado que la desviación media con respecto a la cota inferior es menor al 1%. Además el tiempo de computación empleado por SS en encontrar la mejor solución solo supera los 20 segundos en uno de los casos (26.22 para $n = 800$, $p = 10$).

6.- Experiencias con datos reales

En este apartado se muestran los resultados de las pruebas realizadas con problemas reales considerando las funciones objetivos de ambos modelos. Los datos de los problemas reales se refieren a la provincia de Burgos (Norte de España). Con estas experiencias se quieren analizar donde situar una serie de unidades de diabetes entre las diferentes localidades que pueden acoger las mismas.

En principio se consideran 452 poblaciones dentro de la provincia con al menos algún caso declarado de diabetes, (en realidad estos datos son estimaciones, ya que datos exactos por población no se nos ha documentado, en cualquier caso entendemos

que estas estimaciones son bastantes aproximadas a la realidad porque nos las ha suministrado el personal de la Junta de Castilla y León, www.jcyl.es). Dentro de estas poblaciones se considera un subconjunto de 152 localidades capaces de acoger una unidad de este tipo (por tener algún tipo de instalación que pueda considerarse adecuada).

Se considera la matriz de tiempos (en minutos) entre las 452 poblaciones origen, y las 152 poblaciones que, potencialmente, pueden ser destinos. Para hallar estos tiempos de recorrido se ha usado la información sobre carreteras suministrada por el CNIG (Centro Nacional de Información Geográfica), considerando diferentes velocidades según el tipo de tramo (Nacionales, Autonómicas, Provinciales etc…). Con esta información sobre la red de carreteras se ha calculado la matriz de tiempos usando el conocido algoritmo de Djikstra. Por supuesto, quedan a disposición del lector interesado los ficheros con todos los datos, (poblaciones origen y destino, matrices de tiempos, casos en cada población).

En la figura siguiente se muestra un mapa con las poblaciones de la provincia de Burgos con casos de diabetes (en azul claro), así como entre estas aquellas que podrían acoger unidades especiales, (en verde).

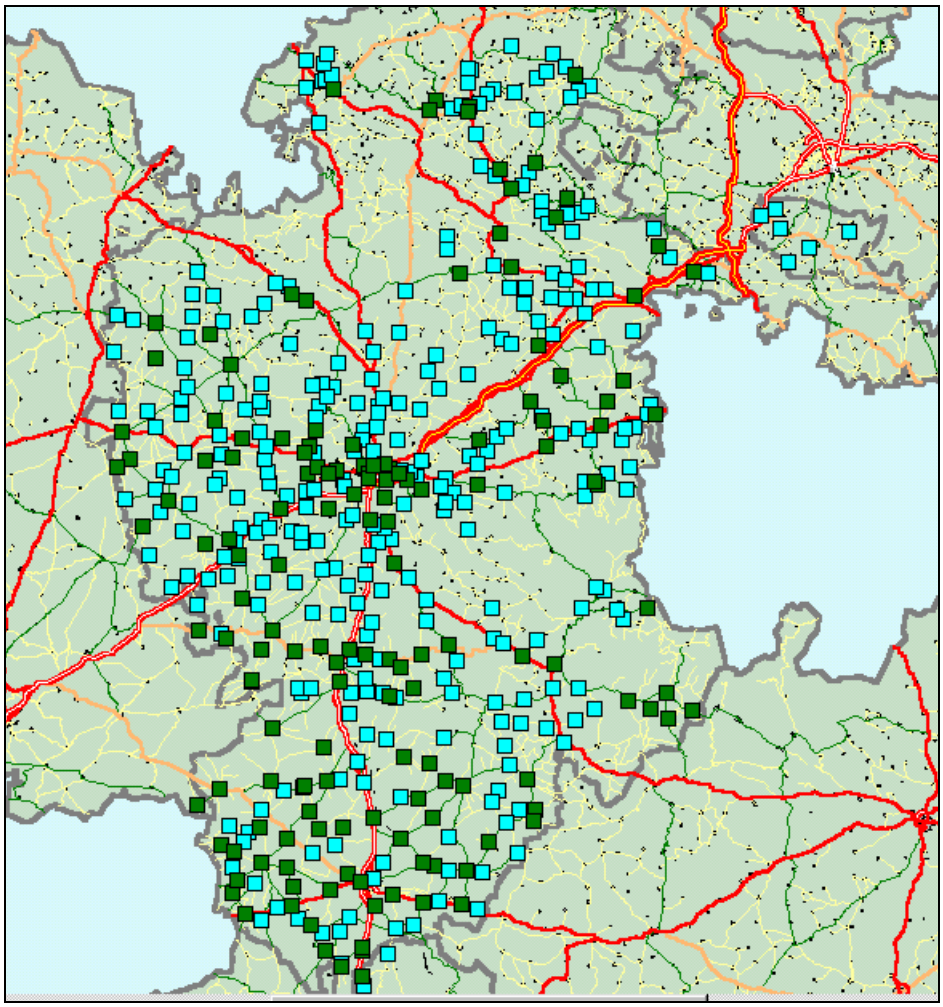

Figura 2.- *Poblaciones de la provincia de Burgos con casos de diabetes (azul claro), y aquellas que pueden acoger unidades especiales (verde).*

Para ambos modelos se han considerado 3 valores de p, p = 5, 7 y 10. Además para el modelo del MSC se ha considerado dos valores de *t_crítico*,

$$t\_crítico = Round(Solucion\ p\text{-}centro/1.5) \text{ y}$$
$$t\_crítico = Round(Solucion\ p\text{-}centro/2)$$

donde *Solucion p-centro* es el valor de la mejor solución del problema del p-centro correspondiente. Por tanto se tienen 3 instancias para el problema del p-centro y 6 para el problema del MSC. Con estas instancias se ha ejecutado nuestro algoritmo SS y algoritmo VNS (Mladenović y otros, 2001) para el problema del p-centro, y nuestro algoritmo de SS y CPLEX para el problema del MSC. En todos los casos como criterio de parada para SS se considera un tiempo de computación de n (152) segundos y en el caso de CPLEX de dos horas. A continuación se muestran dos tablas con los resultados (*Valor*) para ambos modelos, así como el tiempo de cálculo hasta obtener la mejor

solución (*T.mejor*) usado por cada algoritmo. Además en la tabla 4 también se muestra el porcentaje de desviación respecto de la mejor cota inferior encontrada (*CI Dev*).

p	*VNS*		*SS*	
	Valor	*T.Mejor*	*Valor*	*T.Mejor*
5	61	0.22	61	0.05
7	46	10.86	46	0.48
10	41	7.20	41	0.05

Tabla 3.- *Resultados con datos reales para el problema del p-centro*

p	*T_critico*	*CPLEX*	*SS*		
			Valor	*CI Dev*	*T.Mejor*
5	41	*119*	121	1.68	*0.22*
	31	*595*	595	0	*0.77*
7	31	*260*	262	0.77	*4.28*
	23	*768*	768	0	*0.17*
10	27	*204*	206	0.98	*0.6*
	21	*621*	622	0.16	*1.05*

Tabla 4.- *Resultados con datos reales para el problema del MSC*

Como ocurría en el caso de las instancias de la librería Or, SS obtiene mejores resultados que VNS para el problema del *p*-centro (es mejor en dos de los tres casos e iguala los resultados en uno de ellos). En el problema del MSC, SS obtiene soluciones de una calidad aceptable porque la desviación media de las soluciones respecto de la cota inferior es menor del 1%.

A continuación en la figura 3 se muestra un plano con la solución obtenidas por SS, para el caso de p= 10, para el problema del *p*-centro.

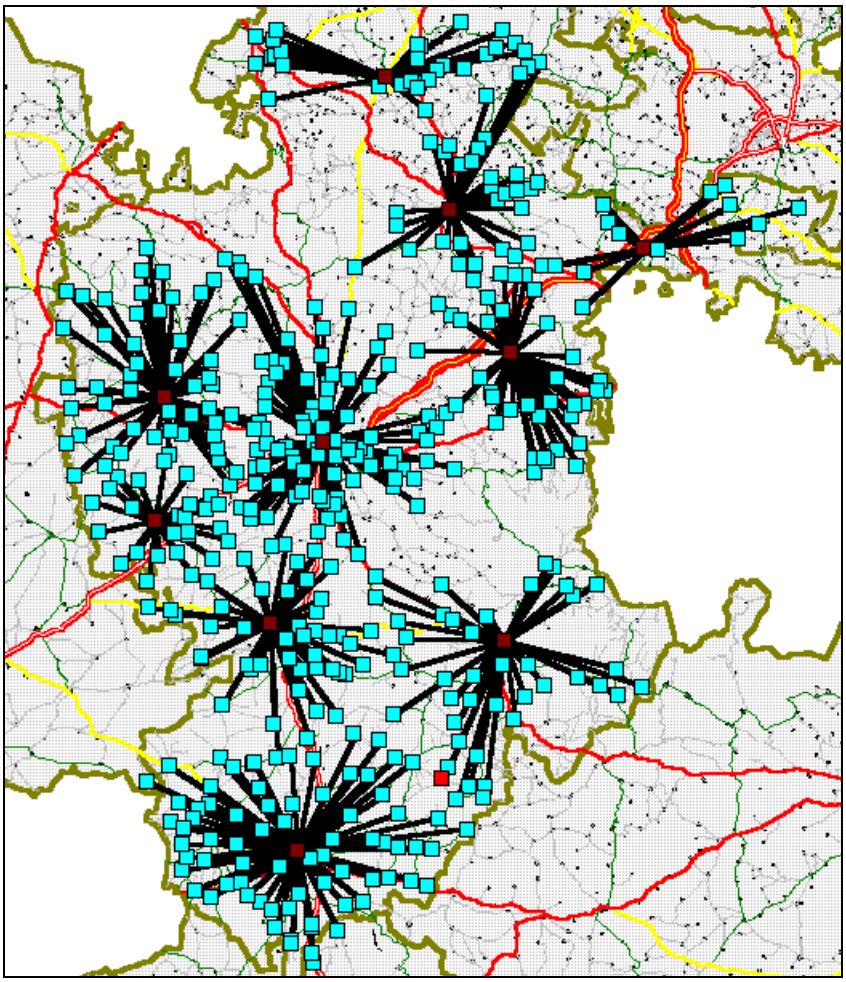

Figura 3.- *Mapa que indica en morado las localizaciones donde se colocan las facilidades correspondientes al problema de los p_centros, con la solución dada por SS. Así mismo cada población va unida mediante líneas a la facilidad asignada. En rojo la población 'crítica' (máximo tiempo a su facilidad más cercana).*

Finalmente de la misma forma se muestra en la siguiente figura un mapa con la solución del problema del tiempo crítico, para $p = 10$ y $t\_crítico = 21$.

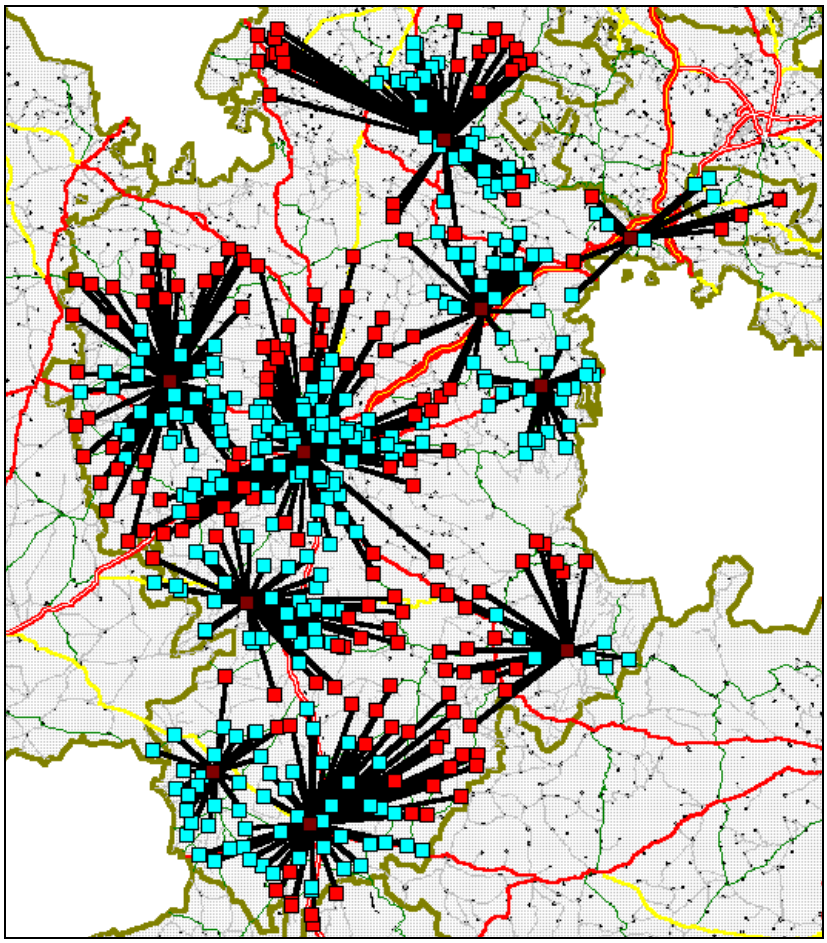

Figura 4.- *Mapa que indica en morado las localizaciones donde se colocan las facilidades correspondientes al problema del tiempo crítico. En rojo las poblaciones que están más alejadas que el t_crítico (t_crítico = 21) de su facilidad más cercana.*

7.- Conclusiones

La investigación descrita en este trabajo ha sido motivada por la necesidad de encontrar soluciones al problema consistente en determinar las localizaciones adecuadas de recursos sanitarios en las provincias de Castilla y León, región al norte de España, concretamente en la provincia de Burgos. Esta es una zona, en general, muy rural con poblaciones muy dispersas. De ahí la necesidad de desarrollar un método que sea capaz de dar buenas soluciones, para el caso de un número bajo de recursos a añadir. En este trabajo se han considerado 2 problemas: minimizar la máxima distancia entre usuarios y facilidades (*p*-centro), y minimizar la población que dista más de su correspondiente facilidad una determinada distancia o tiempo (Maximum Set Covering). Para ambos

casos se ha desarrollado un procedimiento basado en la estrategia búsqueda dispersa (scatter search). Se han analizado tanto instancias ficticias, como reales. Estos datos reales son precisamente, las estimaciones de casos de diabetes en la provincia de Burgos. Se ha comprobado que para el problema del *p*-centro, SS obtiene soluciones de calidad similar a las obtenidas por otros procedimientos desarrollados recientemente, en tiempos de computación mejores. En el caso del problema del MSC se obtienen soluciones de muy buena calidad dado que el porcentaje de desviación media respecto a la cota inferior es menor al 1%.

Bibliografía

Beasley, J. E., (1990): "*OR-Library: Distributing Test Problems by Electronic Mail" in Journal of the Operational Research Society*, 41, 1069-1072.

Campos, V., Glover, F., Laguna, M. and Martí, R. (2001): "*An Experimental Evaluation of a Scatter Search for the Linear Ordering Problem*" *in Journal of Global Optimization*, vol. 21, pp. 397-414.

Dongarra, J.J. (2003): *"Performance of various computers using standard linear equations software" . Technical Report CS-89-85, University of Tennessee Computer Science Department.*

Feo, T.A. and Resende, M.G.C., (1989): "*A Probabilistic heuristic for a computationally difficult Set Covering Problem*" *in Operations Research Letters*, 8, 67-71.

Feo, T.A. and Resende, M.G.C., (1995): "*Greedy Randomized Adaptive Search Procedures*" *in Journal of Global Optimization*, vol. 2, pp 1-27.

Glover, F. (1998): "*A Template for Scatter Search and Path Relinking," in Artificial Evolution, Lecture Notes in Computer Science, 1363, J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer and D. Snyers (Eds.) Springer*, pp. 13-54.

Glover,F. and Laguna,M. (1997): *Tabu Search. Kluwer Academic Publishers.*

Glover, F., Laguna, M. and Martí, R. (2000): "*Fundamentals of Scatter Search and Path Relinking," in Control and Cybernetics*, vol. 39, no. 3, pp. 653-684.

Hansen, P. and Mladenović, N. (1997): "*Variable neigborhhod search for the p-Median*" *in Location Science*, 5, pp. 207-226.

Insalud (2001): *Memoria de Castilla y León. Insalud.* CD-ROM. D.L. VA-532/2001.

Kariv, O. and Hakami, S.L. (1979): "*An algorithmic approach to network location problems. Part I: The P-center Problem*". *SIAM J. Appl. Math.*, 37, pp. 513-538.

Laguna, M. (2002.: "*Scatter Search*," in *Handbook of Applied Optimization*, *P. M. Pardalos and M. G. C. Resende (Eds.), Oxford University Press, New York*, pp. 183-193.

Laguna, M. and Martí, R. (1999): "*GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization*" *in INFORMS Journal on Computing*, vol. 11, no. 1, pp. 44-52.

Laguna, M. Y Martí, R (2003): "*Scatter Search. Methodology and Implementations" in C. Kluwer Academic Publishers.*

Love, R.F., Morris, J.G. and Wesolowsky, G.O. (1988): *Facilities Location: Models and Methods*. North Holland.

Mladenović, N., Labbe, M. and Hansen, P. (2001): "*Solving the p-Center Problem with Tabu Search and Variable Neigborhood Search*". *Les Cahiers du GERAD*, G-2000-35.

Mladenović, N., Moreno, J.P. and Moreno-Vega, J. (1995): "*Tabu search in solving the p-facility location-allocation problems*" *in Les Cahiers du GERAD*, G-95-38.

Mladenović, N., Moreno, J.P. and Moreno-Vega, J. (1996): "*A Chain-Interchange Heuristic Method". Yugoslav. Journal Operations Research, 6 (1), pp.41-54.*

Mulvey J.M. y Beck M.P. (1984): "*Solving Capacitated Clustering Problems*" *in European Journal Operations Research*, 18 (3), pp. 339- 348.

Pitsoulis, L.S. and Resende, M.G.C., (2002): "*Greedy Randomized Adaptive Search Procedures" in Handbook of Applied Optimizaton*, *P. M. Pardalos and M. G. C. Resende (Eds.), Oxford University Press,* pp. 168-182.

Resende, M. G. C. (1998): "Computing Approximate Solutions of the Maximum Covering Problem using GRASP" *in J. of Heuristics*, vol. 4, pp. 161-171.

Resende, M. G. C. and Ribeiro, C.C. (2003). "*A GRASP with path-relinking for private virtual circuit routing*" *in Networks*, vol. 41, pp. 104-114.

Ribeiro C.C., Uchoa E. and Wernek R.F. (2002): *"A Hybrid GRASP with Perturbations for the Steiner Problem in Graphs" in INFORMS Journal on Computing*, vol. 14(3), pp.228-246.

Whitaker, R. (1983): "*A fast algorithm for the greedy interchange for large-scale clustering and median location problems" in INFOR* 21, pp. 95-108.