Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Estadística

Problemas de Cubrimiento Máximo. Modelización y métodos de resolución exactos y heurísticos.

Autora:
Isabel Muñoz Renilla

Tutores:
Jesús Sáez Aguado
Jesús Alberto Tapia García

Curso 2022 - 2023

Resumen

En este trabajo se tratan diferentes problemas de Cubrimiento Máximo, que entran dentro de problemas de localización más generales.

Se comparan, a partir de dos tipos de archivos de datos, distintos métodos de resolución heurística de problemas de Cubrimiento Máximo, para hallar la disposición de los servicios que atienda de la mejor forma posible las diferentes demandas dada una serie de restricciones.

Se empieza con un capítulo, en el que se abordan los principales problemas de localización discreta de los servicios, así como los distintos tipos de objetivo que se pueden lograr.

En los dos siguientes capítulos, se habla de las diferentes aplicaciones y adecuación del problema de Cubrimiento Máximo, y de las heurísticas y metaheurísticas para hallar su solución aproximada.
También se realiza un breve análisis de los resultados para esos métodos heurísticos, que concluye con la elección del método Greedy + Búsqueda Local como el que más rápidamente se aproxima a la solución óptima del problema de Cubrimiento Máximo y que en algunas ocasiones puede llegar a alcanzarlo.

A continuación se detallan algunos modelos de cubrimiento adicionales de interés, como son los modelos de varios equipos y los de cubrimiento reforzado, no estudiados durante el Grado.

Finalmente se muestran las conclusiones obtenidas de este trabajo.

Abstract

In this project, different Maximum Coverage problems are dealt with, which are part of more general location problems.

Using two types of data files, different methods for solving Maximum Coverage problems are compared to find the provision of services that best meets the different demands given a series of restrictions.

It begins with a chapter, in which the main problems of discrete location of services are addressed, as well as the different types of objectives that can be achieved.

In the next two chapters, we discuss the different applications and adequacy of the Maximum Coverage problem, and the heuristics and metaheuristics to find its approximate solution.
A brief analysis of the results for these heuristic methods is also carried out, which concludes with the choice of the Greedy + Local Search method as the one that most quickly approaches the optimal solution of the Maximum Coverage problem and that can sometimes reach it. .

Some additional coverage models of interest are detailed below, such as the multi-equipment models and the reinforced coverage models, not studied during the Degree.

Finally the conclusions obtained from this project are shown.

Índice general

Índice de figuras

Índice de tablas

Introducción

En los problemas de localización de servicios se busca localizar los puntos de servicio para que se pueda atender de la mejor manera posible a los distintos puntos de demanda.

El objetivo de este trabajo fin de grado es, dado un problema de Cubrimiento Máximo, comparar los distintos métodos de resolución para tratar de obtener la disposición de los servicios que atienda más adecuadamente las diferentes demandas, y estudiar la modelización de otros modelos de Programación Entera no vistos durante el Grado.

A veces se puede obtener el óptimo o solución exacta, pero requiere un Solver comercial de licencias caras, por lo que es necesaria la resolución por métodos heurísticos.

En este estudio se cuenta con dos tipos de archivos de datos sobre los que se saca las conclusiones oportunas.

La organización de este trabajo es la siguiente:

En el Capítulo 1 se habla de los principales problemas de localización discreta de los servicios, así como los distintos tipos de objetivo que se pueden lograr.

En el Capítulo 2 se introduce la modelización de la solución óptima de problemas de Cubrimiento Máximo y en el Capítulo 3 heurísticas y metaheurísticas para hallar la solución aproximada al problema de Cubrimiento Máximo, con las que se lleva a cabo un breve análisis de los resultados para los dos tipos de archivos de datos con los que se cuenta.

En el Capítulo 4 se habla de otros tipos de modelos de cubrimiento que pueden resultar interesantes, estos son los modelos de cubrimiento de varios equipos y los de cubrimiento reforzado

Se finaliza el trabajo fin de grado mostrando las conclusiones obtenidas.

Capítulo 1

1. Problemas de localización

En este capítulo se establece la notación que se utiliza a lo largo del trabajo, y se detallan los principales problemas de localización de los servicios y los distintos tipos de objetivo que se pueden lograr.

Se puede hacer distinción entre servicios públicos, (centros de salud, atención domiciliaria, emergencias, centros educativos, etc) y privados (instalación de centros comerciales, de farmacias, sucursales bancarias, etc).
Además, hay que diferenciar entre los servicios en los que el demandante acude al servicio, y en los que el servicio acude al demandante, o al punto de demanda, esto último es útil a la hora de comprobar la adecuación del problema de Cubrimiento Máximo.

El tipo de localización puede ser continua, en la que el servicio puede ser instalado en el espacio o en el plano de forma continua, y discreta, en la que el servicio puede ser instalado en un conjunto discreto o finito, como poblaciones, ciudades, etc.

En este Trabajo Fin de Grado sólo se trata el caso de localización discreta.

1.1. Notación

M = {1, …, m} puntos de demanda
Poblaciones, ciudades, etc.

N = {1, …, n} puntos de posible ubicación de los servicios
Centros de salud, emergencias, centros comerciales, etc.

d_{ij} = distancia (o tiempo) entre el punto de demanda i y el servicio j
$\forall i \in M \quad \forall j \in N$
La distancia o tiempo por carretera obtenida a través de Google maps, o bien, la distancia Euclídea calculada a partir de unas coordenadas.

dc $\in \mathbb{N}$ distancia de cubrimiento
Será la distancia límite a la que puede encontrarse un punto de demanda para poder ser atendida correctamente por un servicio.
No todos los puntos pueden quedar a esa distancia.

h_i = demanda en el punto de demanda i
$\forall i \in M$
Número de demandas o población a atender en un punto de demanda.

p = número de servicios disponibles
Solo en algunos modelos.

Abarcaremos dos clases de problemas:

- M = N
  En los que el número de puntos de demanda coincide con el número de servicios disponibles.

- N ⊂ M
  En los que hay más puntos de demanda que servicios disponibles.

1.2. Principales problemas de localización discreta

En DASKIN, M. S. (2013) (12) y en CHURCH, R. and C. REVELLE (1976) (7) se hace una descripción extensa de todos estos problemas.

Véase además EISELT, H. A. and V. MARIANOV (2011) (14).

- **Cubrimiento Total (Set Covering):**
  Consiste en encontrar la solución a la localización de los servicios que permita dejar cubiertas todas las demandas con el menor costo posible, dada una distancia de cubrimiento.

- **Cubrimiento Máximo o parcial (MCLP, Maximal Covering Location Problem):**
  Trata de encontrar la localización óptima de un número dado de servicios que maximice la demanda total cubierta dentro de la distancia de cubrimiento.

- **P-mediana:**
  Trata de localizar p servicios minimizando el desplazamiento total, ya sea el recorrido por los demandantes hasta los servicios o el que deben recorrer los servicios hasta los puntos de demanda.

- **P-centro:**
  Se localizan p servicios tratando de minimizar el desplazamiento máximo recorrido por los demandantes a los servicios, o bien, el que recorren los servicios para llegar hasta los puntos de demanda.

1.3. Eficiencia y equidad

Podemos modelar este tipo de problemas de localización con el objetivo de lograr eficiencia (p-mediana) o con el de lograr equidad (p-centro).

Véase CANÓS DARÓS, M. J., M. MOCHOLÍ ARCE and M. MARTÍNEZ ROMERO (2009) (4).

Eficiencia (p-mediana):

Se trata de minimizar la distancia total recorrida, objetivo que plantea el problema de la p-mediana.
La solución se alcanza situando los servicios en los puntos que presenten mayor demanda, de manera que la distancia a la mayoría de los puntos de demanda sea la más corta posible.
Un inconveniente del objetivo de eficiencia es que la localización final puede hacer que, para algunos puntos de demanda, haya que recorrer distancias demasiado largas, por lo que el problema de la p-mediana no cumple el objetivo de equidad.

Equidad (p-centro):

Se pretende minimizar la distancia máxima recorrida, por ello el problema de p-centro está directamente asociado al objetivo de lograr equidad.
Soluciona el problema de distancias excesivas para algunos puntos de demandas, pero la distancia total será mayor respecto de la del problema de la p-mediana, por tanto, no cumple el objetivo de eficiencia.

Para ilustrar los problemas de localización, se muestra el siguiente ejemplo:

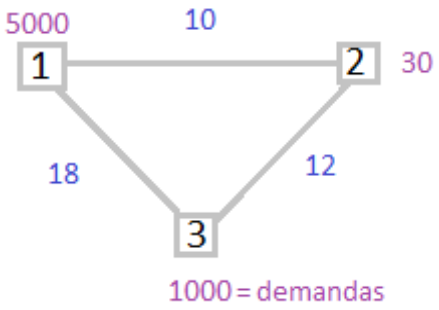

Figura 1: Ejemplo localización de servicios

p = número de servicios disponibles = 1

- **Problema de la p-mediana:**

  Si colocamos un servicio en el punto de demanda 1, es decir, el de mayor demanda:
  - Distancia máxima: 18
  - Distancia total: $30 \cdot 10 + 1000 \cdot 18 = 18300$

- **Problema de p-centro:**

  Si colocamos un servicio en el punto de demanda 2 para minimizar la distancia máxima:
  - Distancia máxima: 12
  - Distancia total: $5000 \cdot 10 + 1000 \cdot 12 = 62000$

- **Problema de p-centro con dc=15:**

  Si fijamos una distancia de cubrimiento, dc=15:
  - Distancia total: $5000 \cdot 10 + 1000 \cdot 12 = 62000$
  - Se cubre el 100% de las demandas

Observamos en este ejemplo que el problema de p-centro no cumple el objetivo de eficiencia ya que la distancia total recorrida frente a la del problema de la p-mediana es mucho mayor, pero tiene menor distancia máxima entre servicio y punto de demanda, luego la localización será mucho más equitativa.

El problema de p-centro con restricción de distancia de cubrimiento tiene la misma distancia total recorrida que el problema de p-centro sin esta restricción.

Para el caso en el que tratáramos con servicios de emergencias lo más conveniente sería tratar el problema como uno de Cubrimiento Máximo, buscando cumplir el objetivo de equidad propio del problema de p-centro.

Capítulo 2

2. Modelización de la solución óptima de problemas de Cubrimiento Máximo

Para tratar de dar solución a los problemas de Cubrimiento Máximo se pueden emplear los siguientes modelos de Programación Entera, introducidos por CHURCH, R. and C. REVELLE (1974) (6) y C-H. CHUNG, C-H. (1986) (5).

(En este capítulo sólo están los modelos que están muy relacionados con Cubrimiento Máximo, luego se contarán más modelos que no han sido contados durante el Grado).

- **Cubrimiento Máximo (MCLP):**

$$\max \sum_{i \in M} h_i \cdot y_i$$

Sujeto a

$$\sum_{j \in N_i} x_j \geq y_i \quad \forall i = 1 \dots m$$

$$\sum_{j \in N} x_j = p$$

$$x_j = \begin{cases} 1 & si\ un\ servicio\ está\ localizado\ en\ el\ punto\ j \\ 0 & si\ no \end{cases}$$

$$y_i = \begin{cases} 1 & si\ la\ demanda\ del\ punto\ de\ demanda\ i\ es\ cubierta \\ 0 & si\ no \end{cases}$$

$$N_i = \{j \mid d_{ij} \leq dc\}$$

N_i = los puntos de servicio que cubren adecuadamente los puntos de demanda i

- **Cubrimiento Máximo + restricción de distancia máxima:**
  Se pretende encontrar la localización de servicios que maximice la demanda cubierta total dada una distancia máxima de servicio.

$$\max \sum_{i \in M} h_i \cdot y_i$$

Sujeto a

$$\sum_{j \in N_i} x_j \geq y_i \quad \forall i = 1 \dots m$$

$$\sum_{j \in N} x_j = p$$

$$\sum_{j \in N_i^2} x_j \geq 1 \quad \forall i = 1 \dots m$$

$$x_j = \begin{cases} 1 & si\ un\ servicio\ está\ localizado\ en\ el\ punto\ j \\ 0 & si\ no \end{cases}$$

$$y_i = \begin{cases} 1 & si\ la\ demanda\ del\ punto\ de\ demanda\ i\ es\ cubierta \\ 0 & si\ no \end{cases}$$

$$N_i^2 = \{j \mid d_{ij} \leq dmax\}$$

N_i^2 = los puntos de servicio que cubren los puntos de demanda i

dmax = distancia máxima a la que deben quedar cubiertos todos

- **Cubrimiento Máximo Capacitado (CMCL):**
  Problema de Cubrimiento máximo en el que se establece un límite al número total de demandas que pueden ser cubiertas por cada uno de los servicios.

$$\max \sum_{i \in M} h_i \cdot y_i$$

Sujeto a

$$\sum_{j \in N_i} x_{ij} \geq y_i \quad \forall i = 1 \dots m$$

$$\sum_{i \in M} h_i \cdot x_{ij} \leq c_j \cdot x_j \quad \forall j = 1 \dots n$$

$$\sum_{j \in N} x_j = p$$

$$x_{ij} \leq x_j \quad \forall i = 1 \dots m, \qquad \forall j = 1 \dots n$$

Esta restricción nos indica que si x_j vale 0 no se abre un punto de servicio y no se puede atender a ningún punto de demanda i (Restricción lógica).

c_j = número de demandas que puede cubrir como máximo cada uno de los servicios

$$x_{ij} = \begin{cases} 1 & si\ el\ punto\ de\ demanda\ i\ está\ asignado\ al\ servicio\ j \\ 0 & si\ no \end{cases}$$

$$x_j = \begin{cases} 1 & si\ un\ servicio\ está\ localizado\ en\ el\ punto\ j \\ 0 & si\ no \end{cases}$$

$$y_i = \begin{cases} 1 & si\ la\ demanda\ del\ punto\ de\ demanda\ i\ es\ cubierta \\ 0 & si\ no \end{cases}$$

Este trabajo se centra en el objetivo que plantea el problema de Cubrimiento Máximo (MCLP) de maximizar la demanda cubierta total, y esto será lo que se intente lograr cuando se trate con casos reales.

2.1. Aplicaciones del MCLP

Servicios de emergencias

Cuando los servicios sean ambulancias.
La distancia de cubrimiento es la que fijan los estándares de cada servicio, por ejemplo, en ambulancias medicalizadas se fija un tiempo de 8 a 10 minutos en caso de un paciente que haya sufrido un infarto.

El problema de Cubrimiento Máximo es bastante adecuado cuando los servicios son ambulancias, pues es un caso en el que el servicio acude a la demanda.

Análisis Cluster

El modelo MCLP puede ser fácilmente aplicado al análisis cluster, este trata de agrupar entidades de datos de manera que se maximice la homogeneidad intra-grupos y la entre-grupos simultáneamente.

En este tipo de análisis, se cuenta con la llamada medida de similitud, que se suele representar con una distancia entre cada entidad para cada grupo y sirve para poder asignar las entidades de datos a un grupo u otro dependiendo del valor de esta medida. Se asigna una entidad al grupo con el que presente menor distancia.

Si se extrapola esto al modelo MCLP, cada entidad es considerada un punto de demanda y cada cluster un conjunto de posibles ubicaciones de los servicios.

Análisis discriminante

En este tipo de análisis se divide el conjunto de datos en k grupos, que se representan con la mediana de cada uno de ellos.
Para asignar una entidad a un grupo u otro, se toma como medida determinante la cercanía al valor de la mediana de cada uno de los grupos, siendo asignado al más cercano.

Al igual que lo dicho para el análisis cluster, en el modelo MCLP cada entidad es considerada un punto de demanda y, ahora para el análisis discriminante, cada uno de los k grupos es un conjunto de posibles ubicaciones de los servicios.

Proceso cognitivo:

Es interesante decir que el modelo MCLP proporciona una alternativa a lo ya existente en cuanto al estudio del proceso cognitivo humano, pues las relaciones entre información y conocimiento son parecidas a las que hay entre demanda y servicio.

Se trata de maximizar la información obtenida (demanda) de manera que la pérdida de información (el número de demandas que quedan sin atender) sea mínima.

2.2. Adecuación del modelo MCLP

La pregunta que surge ahora es cuándo es adecuado plantear el problema de Cubrimiento Máximo.

Cuando el problema trate de que los usuarios acudan a los puntos de servicio, no es adecuado este tipo de planteamiento pues sería muy ineficiente.
Un ejemplo es el caso de los servicios de atención primaria.
Podría aplicarse en este tipo de problemas el modelo de la p-mediana.

Usuarios → Servicio **MCLP no adecuado**

Sin embargo, cuando los servicios son los que acuden a los usuarios o puntos de demanda, entonces es adecuado plantear el problema de Cubrimiento Máximo.

Servicio → Usuario **MCLP adecuado**

Capítulo 3

3. Solución aproximada al MCLP

En este capítulo se explican los distintos métodos de resolución exacta y aproximada a los problemas de Cubrimiento Máximo.
Después se usan para dos tipos de archivos de datos y se anotan los resultados para luego poder realizar comparaciones en cuanto a porcentajes de demanda total cubierta y a tiempos de ejecución.

3.1. Resolución óptima o exacta al MCLP

Para resolver de forma exacta u óptima el problema de Cubrimiento Máximo se va a usar el algoritmo de Programación Entera Branch and Bound implementado en Xpress Mosel.

- **Resolución del modelo:**

  Utilizando el Solver de Programación Entera de Xpress se obtiene la solución exacta u óptima del modelo dado p (número de servicios disponibles) y dc (distancia de cubrimiento).

3.2. Heurísticas y metaheurísticas al MCLP

- **Resolución heurística:**

  - **Greedy Adding (GA)**

    Se describe en DASKIN, M. S. (2013) (12) y en CHURCH, R. and C. REVELLE (1976) (6).

    Se trata de encontrar al candidato que maximice la demanda cubierta, para incorporarlo a la solución final, siempre que cumpla con las restricciones del problema.
    Luego, el criterio Greedy es maximizar la demanda total cubierta.

    Hay tantas iteraciones como número de servicios disponibles, p.

    En cada iteración se escoge el máximo, es decir, una solución local, sin considerar como puede afectar a la solución global, hasta obtener al final la solución final.

Se define Z(j) como la demanda total que cubre el punto j a una distancia menor o igual que una distancia de cubrimiento dada.
Se puede calcular el máximo con **zmax:=max(j in pservicio)Z(j)**, pero para obtener el índice **jmax** con el lenguaje Mosel se tiene que recorrer el siguiente bucle:

```
zmax:= - 999
forall(j in pservicio | xsol(j)=0)do
        if(Z(j) > zmax)then
                zmax:=Z(j)
                jmax:=j
        end-if
end-do
```

Y luego se añade a la solución final:

```
xsol(jmax):=1
ind_ab(t):=jmax
```

Siendo ind_ab un array(1..p) con los índices de los puntos de servicio abiertos.

También se guarda en un array ind_noab(1..(n-p))los índices de los puntos de servicio cerrados o que no hayan sido abierto:

```
!Puntos no abiertos:
na:=0
forall(j in pservicio)do
        if(xsol(j)=0)then
                na:=na+1
                ind_noab(na):=j
        end-if
end-do
```

Las soluciones del método Greedy pueden mejorarse mediante heurísticas multi-arranque basadas en la aleatorización, como el método Greedy Aleatorizado, o aplicando Búsqueda Local, que parte de su solución factible y trata de mejorarla.

- **Greedy + Búsqueda Local**

  Véase DASKIN, M. S. (2013) (12).

  Se parte de una solución factible obtenida con Greedy, y se trata de mejorarla mediante un intercambio de índices, para escoger una nueva solución.

  Usaremos los p índices de puntos de servicio abiertos y los n-p índices de no abiertos, para mejorar la complejidad con respecto a si usamos xsol=1 y xsol=0, pues cambia de n^2 a p·(n-p), reduciendo considerablemente el tiempo de ejecución para los archivos de datos más grandes.
  Se observa si hay mejora en el valor objetivo, y mientras siga habiendo se realizan intercambios y comprobaciones de mejora hasta que se llegue a un mínimo local, es decir, una solución que es igual o mejor que las de su entorno, es decir, que no presenta mejora o es el de mejora máxima.
  Lamentablemente, la mayoría de las veces el óptimo local no será un óptimo global.

- **Greedy Aleatorizado**

  Se describe en RESENDE, M. G. C. and C. C. RIBEIRO (2016) (17).

  Para dotar de aleatoriedad al método Greedy, se utiliza una lista restringida de candidatos (RCL, Restricted Candidate List), formada por un número fijo de candidatos que maximizan la demanda (el criterio en este trabajo).
  La solución se elige aleatoriamente de entre los candidatos de la lista restringida.
  Esto se repite un número de iteraciones, niter.
  En cada iteración se comprueba si la demanda cubierta **zheur** es mayor que con la asignación actual **zmax**, y en ese caso pasa a ser la nueva solución.

  La solución final es la mejor solución que el método pueda encontrar fijados el tamaño de la lista restringida de candidatos y el número de iteraciones.

- **GRASP (Greedy Randomised Adaptive Search Procedure)**

  Es una metaheurística muy eficiente y está totalmente descrita en RESENDE, M. G. C. (1998) (16).

  Un algoritmo GRASP completo se obtiene aplicando búsqueda local a la solución Greedy obtenida en cada iteración del método Greedy Aleatorizado.

  En cada iteración se haya un mínimo local, como resultado de la búsqueda local, y gracias a la aleatoriedad del Greedy Aleatorizado, la mejor solución obtenida suele estar muy cerca del óptimo global.

3.3. Aplicación de heurísticas y metaheurísticas al MCLP

En este trabajo se utilizan de dos tipos de archivos de datos diferentes con los que se obtienen los resultados y las conclusiones oportunas.
Cada uno abarca una de las dos clases de problemas que se mencionan en el punto 1.1 ($M=N$ y $N \subset M$)

- **Archivos .dat (CYL):**
  Se cuenta con M puntos de demanda (poblaciones o consultorios) y N puntos de posible localización de los servicios (puntos de salud), $M > N$, por eso $N \subset M$

  En la fila 1 se encuentra el número de puntos de demanda.
  En la fila 2 se encuentra el número de puntos de servicio.
  A continuación se tiene la demanda, en este caso población, en cada punto de demanda.
  Y la matriz $M \cdot N$ de distancias de desplazamiento (o tiempos) para cada
  $i = 1,...,m$, $j = 1,...,n$

- **Archivos .pnl (plano):**
  Todos los puntos de demanda pueden ser puntos de servicio por eso $M=N$

  En la fila 1 se tiene el número de puntos de demanda = puntos de servicio.
  En la fila 2 la coordenada X (primera columna), la coordenada Y (segunda columna) y la demanda del punto de demanda 1.
  En la fila 3 la coordenada X (primera columna), la coordenada Y (segunda columna) y la demanda del punto de demanda 2.
  Y así sucesivamente.

Para cada uno de estos dos tipos de archivos de datos se elaboran dos tablas, una con los porcentajes de demanda total cubierta y otra con los tiempos de ejecución, para los diferentes métodos de resolución heurística de problemas de Cubrimiento Máximo.

En el algoritmo Greedy Aleatorizado se fija el tamaño de la lista de candidatos a 3 y el número de iteraciones a 10.

En las tablas se redondea a 3 decimales para los porcentajes y a 4 para los tiempos.

3.3.1. Resultados archivos .dat (CYL)

- **Porcentaje de demanda total cubierta**

Archivo datos	p, dc	Solver Xpress	Greedy	Greedy + Búsqueda Local	Greedy Aleatorizado	GRASP
aint1.dat	8, 20	53.674 %	53.674 %	53.674 %	53.674 %	53.674 %
aint2.dat	8, 20	73.396 %	73.396 %	73.396 %	73.396 %	73.396 %
aint3.dat	8, 20	56.629 %	56.629 %	56.629 %	55.186 %	56.629 %
aint4.dat	8, 20	44.600 %	44.600 %	44.600 %	44.600 %	44.600 %
aint5.dat	8, 20	66.970 %	66.970 %	66.970 %	66.970 %	66.970 %
aint6.dat	8, 20	67.650 %	67.650 %	67.650 %	67.650 %	67.650 %
aint7.dat	8, 20	43.100 %	43.100 %	43.100 %	43.100 %	43.100 %
aint8.dat	8, 20	57.959 %	57.959 %	57.959 %	57.959 %	57.959 %
aint9.dat	8, 20	68.427 %	68.427 %	68.427 %	68.427 %	68.427 %
aint10.dat	8, 20	66.453 %	66.453 %	66.453 %	66.394 %	66.453 %
aint11.dat	8, 20	76.335 %	76.335 %	76.335 %	76.335 %	76.335 %
aint12.dat	8, 20	52.398 %	52.398 %	52.398 %	52.398 %	52.398 %
aint13.dat	8, 20	54.108 %	54.108 %	54.108 %	54.108 %	54.108 %

Tabla 1: Porcentajes de demanda total cubierta para archivos .dat

Se observa que el porcentaje de demanda total cubierta para todas las heurísticas es el mismo en todos los archivos de datos , variando en alguno para el Greedy Aleatorizado, ya que este método no siempre devuelve el óptimo global.

- **Tiempo de ejecución**

Archivo datos	p, dc	Solver Xpress	Greedy	Greedy + Búsqueda Local	Greedy Aleatorizado	GRASP
aint1.dat	8, 20	0.0332 s	0.0062 s	0.0501 s	0.0489 s	0.4769 s
aint2.dat	8, 20	0.0464 s	0.0467 s	0.1299 s	0.1731 s	2.2351 s
aint3.dat	8, 20	0.0554 s	0.0094 s	0.0896 s	0.1048 s	1.7278 s
aint4.dat	8, 20	0.0692 s	0.0415 s	0.1364 s	0.2062 s	3.1114 s
aint5.dat	8, 20	0.0245 s	0.0193 s	0.0551 s	0.0457 s	0.3735 s
aint6.dat	8, 20	0.0210 s	0.0015 s	0.0301 s	0.0130 s	0.1682 s
aint7.dat	8, 20	0.0315 s	0.0164 s	0.0596 s	0.1627 s	1.5270 s
aint8.dat	8, 20	0.0491 s	0.0034 s	0.0602 s	0.0412 s	0.2687 s
aint9.dat	8, 20	0.0295 s	0.0069 s	0.0576 s	0.0327 s	0.2094 s
aint10.dat	8, 20	0.0184 s	0.0068 s	0.0464 s	0.0352 s	0.1470 s
aint11.dat	8, 20	0.0313 s	0.0056 s	0.0599 s	0.0491 s	0.3598 s
aint12.dat	8, 20	0.0356 s	0.0167 s	0.0721 s	0.0931 s	1.5379 s
aint13.dat	8, 20	0.0293 s	0.0099 s	0.0633 s	0.0800 s	0.9251 s

Tabla 2: Tiempos de ejecución para archivos .dat

En esta tabla se comprueba que el tiempo de ejecución para Greedy siempre es el menor, seguido generalmente por el del Solver de Xpress, después el de Greedy + Búsqueda Local, Greedy Aleatorizado y, por último, el algoritmo GRASP siendo esté el que mayor tiempo de ejecución requiere.

En algunos casos se observa que Greedy Aleatorizado tarda menos que el de Greedy + Búsqueda Local, esto se debe a la aleatoriedad propia de este método, y muchas veces se soluciona aumentando el tamaño de la lista de candidatos al menos en uno.

3.3.2. Resultados archivos .pnl (plano)

- **Porcentaje de demanda total cubierta**

Archivo datos	p, dc	Solver Xpress	Greedy	Greedy + Búsqueda Local	Greedy Aleatorizado	GRASP
s50_1	8, 20	98.206 %	94.020 %	96.688 %	96.688 %	97.332 %
s50_2	8, 20	100 %	95.363 %	98.271 %	98.507 %	100 %
s50_3	8, 20	100 %	99.664 %	99.664 %	99.664 %	99.664 %
s50_4	8, 20	100 %	95.708 %	96.862 %	97.554 %	97.739 %
s50_5	8, 20	100 %	96.973 %	99.550 %	98.528 %	100 %
s100_1	8, 20	100 %	95.512 %	98.566 %	97.377 %	98.566 %
s100_2	8, 20	100 %	98.670 %	99.841 %	99.365 %	98.789 %
s100_3	8, 20	99.833 %	97.099 %	99.269 %	97.328 %	99.353 %
s100_4	8, 20	98.709 %	98.402 %	98.709 %	97.603 %	98.320 %
s100_5	8, 20	99.515 %	98.099 %	98.362 %	97.149 %	98.362 %
s200_1	8, 20	98.931 %	95.169 %	96.923 %	96.745 %	97.186 %
s200_2	8, 20	99.971 %	96.499 %	98.133 %	96.653 %	97.340 %
s200_3	8, 20	99.362 %	94.183 %	96.963 %	95.511 %	99.362 %
s200_4	8, 20	99.711 %	99.631 %	99.631 %	99.631 %	99.123 %
s200_5	8, 20	99.253 %	95.974 %	98.419 %	96.236 %	97.157 %
s300_1	8, 20	99.621 %	95.198 %	97.593 %	95.027 %	98.082 %
s300_2	8, 20	98.549 %	94.474 %	97.797 %	94.744 %	97.718 %
s300_3	8, 20	99.167 %	95.319 %	97.220 %	96.165 %	97.034 %
s300_4	8, 20	99.486 %	96.694 %	98.750 %	96.745 %	97.805 %
s300_5	8, 20	99.477 %	97.145 %	99.314 %	96.616 %	97.773 %
s500_1	8, 20	98.097 %	96.441 %	96.999 %	96.614 %	97.246 %
s500_2	8, 20	99.064 %	95.232 %	99.049 %	97.080 %	99.064 %
s500_3	8, 20	99.407 %	96.377 %	98.798 %	96.241 %	97.628 %
s500_4	8, 20	99.017 %	94.694 %	95.591 %	95.419 %	96.566 %
s500_5	8, 20	99.236 %	94.119 %	97.295 %	95.733 %	95.911 %
s800_1	8, 20	98.850 %	97.137 %	97.958 %	96.101 %	98.162 %
s800_2	8, 20	98.788 %	94.348 %	97.526 %	94.539 %	98.213 %
s800_3	8, 20	98.498 %	95.280 %	97.127 %	95.280 %	97.987 %
s800_4	8, 20	99.021 %	94.227 %	96.452 %	94.836 %	98.012 %
s800_5	8, 20	98.909 %	93.916 %	98.114 %	93.926 %	98.564 %
s1000_1	8, 20	99.533 %	95.974 %	97.182 %	96.713 %	97.253 %
s1000_2	8, 20	98.929 %	94.181 %	94.181 %	95.735 %	98.115 %
s1000_3	8, 20	98.754 %	95.620 %	98.565 %	96.183 %	97.865 %
s1000_4	8, 20	98.801 %	96.055 %	97.974 %	96.862 %	98.233 %
s1000_5	8, 20	98.914 %	94.065 %	96.866 %	95.100 %	97.894 %

Tabla 3: Porcentajes de demanda total cubierta para archivos .pnl

El porcentaje de demanda total cubierta para Greedy + Búsqueda Local y para GRASP se aproxima bastante, y en contadas ocasiones alcanza el obtenido para el Solver de Xpress (solución que consideramos óptima global o exacta).

Greedy Aleatorizado, en algunas ocasiones, alcanza o supera por poco el porcentaje de demanda total cubierta de Greedy + Búsqueda Local debido a su aleatoriedad, aunque generalmente será menor, por lo que tampoco sería el mejor método.

- **Tiempo de ejecución**

Archivo datos	p, dc	Solver Xpress	Greedy	Greedy + Búsqueda Local	Greedy Aleatorizado	GRASP
s50_1	8, 20	0.0299 s	0.0064 s	0.0530 s	0.0515 s	0.3317 s
s50_2	8, 20	0.0234 s	0.0059 s	0.0432 s	0.0249 s	0.3773 s
s50_3	8, 20	0.0210 s	0.0037 s	0.0436 s	0.0234 s	0.2945 s
s50_4	8, 20	0.05083 s	0.0049 s	0.0609 s	0.0555 s	0.6032 s
s50_5	8, 20	0.0373 s	0.0036 s	0.0615 s	0.0425 s	0.4894 s
s100_1	8, 20	0.0288 s	0.0089 s	0.0516 s	0.0859 s	2.5493 s
s100_2	8, 20	0.0436 s	0.0066 s	0.0780 s	0.0993 s	3.0254 s
s100_3	8, 20	0.0672 s	0.0260 s	0.1188 s	0.0923 s	2.1304 s
s100_4	8, 20	0.1653 s	0.0238 s	0.2108 s	0.1418 s	1.8473 s
s100_5	8, 20	0.0715 s	0.0233 s	0.1201 s	0.0537 s	2.1078 s
s200_1	8, 20	0.0569 s	0.0541 s	0.1421 s	0.3201 s	17.2092 s
s200_2	8, 20	0.0894 s	0.0490 s	0.1973 s	0.2835 s	12.3382 s
s200_3	8, 20	0.1588 s	0.0375 s	0.2380 s	0.2993 s	18.0119 s
s200_4	8, 20	0.1126 s	0.0379 s	0.1942 s	0.1995 s	9.2002 s
s200_5	8, 20	0.0922 s	0.0531 s	0.1765 s	0.2526 s	14.8373 s
s300_1	8, 20	0.1614 s	0.0973 s	0.3198 s	0.5614 s	48.3102 s
s300_2	8, 20	1.9968 s	0.0778 s	2.1378 s	1.0351 s	43.1223 s
s300_3	8, 20	0.6953 s	0.0827 s	0.8453 s	0.5536 s	44.0793 s
s300_4	8, 20	0.1193 s	0.0681 s	0.2490 s	0.6462 s	42.2823 s
s300_5	8, 20	1.061 s	0.0733 s	1.1930 s	0.4982 s	44.4664 s
s500_1	8, 20	0.9674 s	0.168 s	1.285 s	2.689 s	156.3174 s
s500_2	8, 20	0.4072 s	0.3770 s	0.9675 s	1.8940 s	151.1923 s
s500_3	8, 20	0.2029 s	0.2546 s	0.7238 s	2.3343 s	177.9001 s
s500_4	8, 20	0.8490 s	0.4765 s	1.4622 s	1.9995 s	193.6371 s
s500_5	8, 20	0.8986 s	0.2841 s	1.3397 s	2.1182 s	166.1573 s
s800_1	8, 20	5.5836 s	1.0310 s	6.9754 s	4.7626 s	366.0746 s
s800_2	8, 20	5.0530 s	0.6458 s	6.1702 s	6.9079 s	351.2154 s
s800_3	8, 20	7.4791 s	0.9342 s	8.8772 s	12.5182 s	412.5361 s
s800_4	8, 20	0.5021 s	0.8945 s	1.8090 s	4.8681 s	312.3732 s
s800_5	8, 20	6.4488 s	0.9167 s	7.7705 s	5.9801 s	393.4124 s
s1000_1	8, 20	0.7490 s	0.9142 s	2.4199 s	8.3850 s	781.1773 s
s1000_2	8, 20	8.0755 s	1.1194 s	9.8055 s	11.7629 s	824.2235 s
s1000_3	8, 20	18.4879 s	1.1027 s	20.2579 s	12.7945 s	831.5672 s
s1000_4	8, 20	10.2407 s	1.4354 s	12.1756 s	10.6249 s	781.1773 s
s1000_5	8, 20	2.1893 s	0.9436 s	3.6894 s	9.4425 s	795.2645 s

Tabla 4: Tiempos de ejecución para archivos .pnl

Al igual que lo comprobado en los archivos de datos .dat, el tiempo de ejecución para Greedy siempre es el menor, seguido por el del Solver de Xpress, después se encuentra generalmente el de Greedy + Búsqueda Local, Greedy Aleatorizado y, por último, el algoritmo GRASP siendo esté el que mayor tiempo de ejecución requiere.

Esto último comienza a observarse para los archivos con m=n=100, ya que para los archivos con m=n=50 es Greedy + Búsqueda Local el que tarda más que Greedy Aleatorizado, pero con los archivos de datos más grandes se deduce que generalmente Greedy Aleatorizado tarda más que Greedy + Búsqueda Local.

Se comprueba que Greedy, pese a que es el más rápido, es el que proporciona menor porcentaje de demanda total cubierta, por lo que no es un buen método si lo que se busca maximizar es la demanda total cubierta.

Por tanto, el método de Greedy + Búsqueda Local para archivos de datos con tamaño suficientemente grande es, dentro de los métodos que más se aproximan a la solución exacta u óptima, el que menos tiempo de ejecución requiere, y es por tanto, una de las mejores opciones a la hora de tratar de dar solución al problema de Cubrimiento Máximo (MCLP).

3.3.3. Representaciones gráficas de las aplicaciones al MCLP

Como en los archivos .pnl se proporcionan coordenadas, se pueden construir los siguientes gráficos, en ellos se representan los puntos de demanda en color negro, a los que se les asigna uno de los p=8 servicios disponibles, fijada una distancia de cubrimiento de 20.
Estas asignaciones se representan en color rojo y los puntos de demanda no cubiertos en color azul:

- **Para m=n=50:**

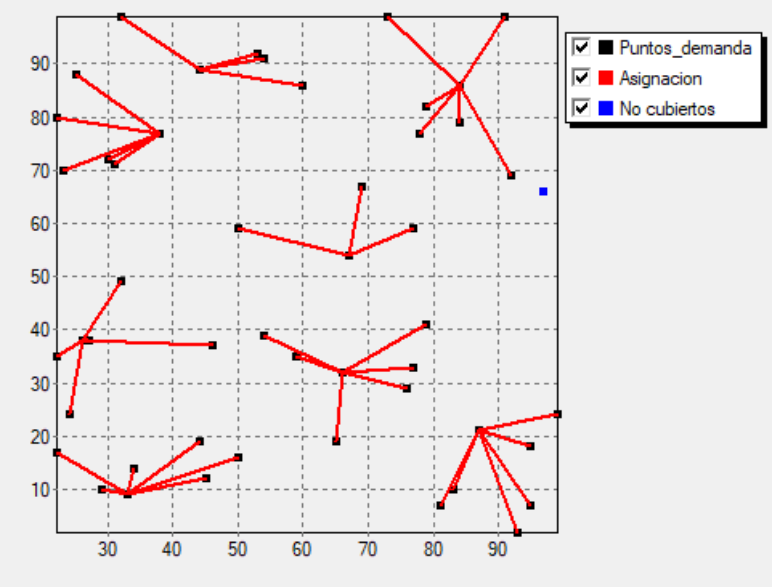

Figura 2: Gráfico archivo .pnl m=n=50

- **Para m=n=100:**

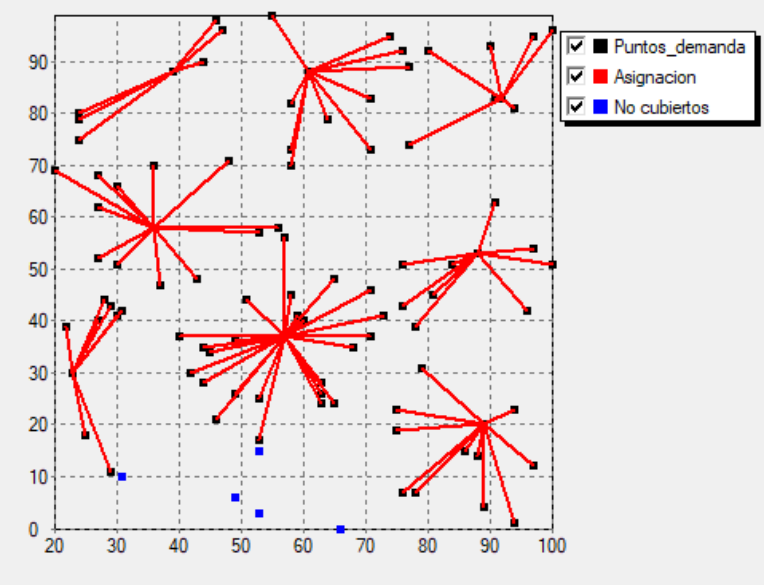

Figura 3: Gráfico archivo .pnl m=n=100

- **Para m=n=200:**

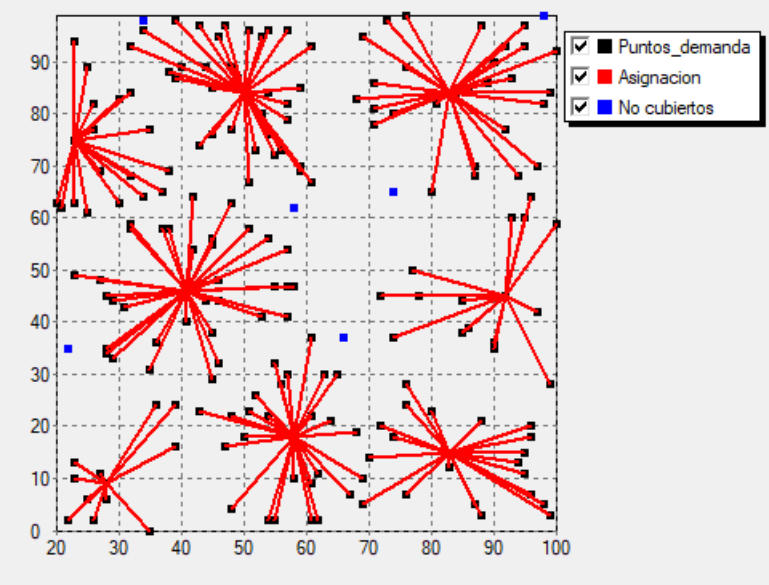

Figura 4: Gráfico archivo .pnl m=n=200

- **Para m=n=300:**

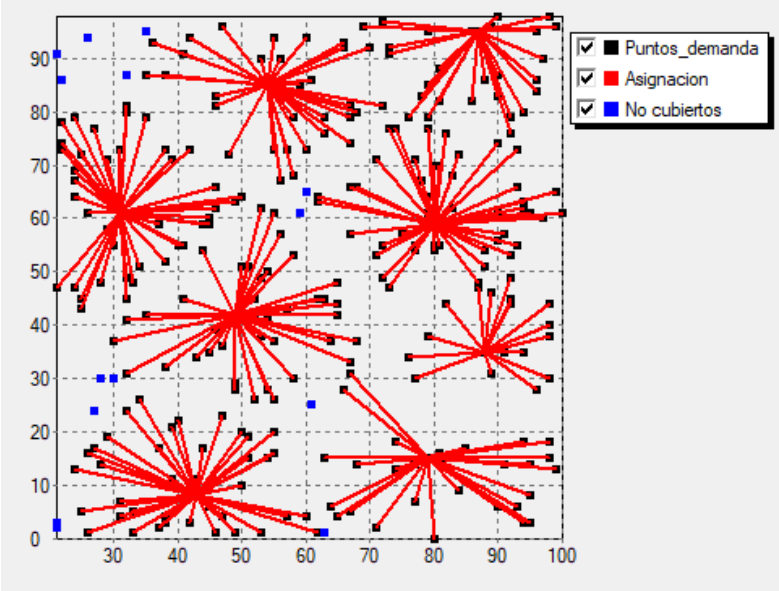

Figura 5: Gráfico archivo .pnl m=n=300

- **Para m=n=500:**

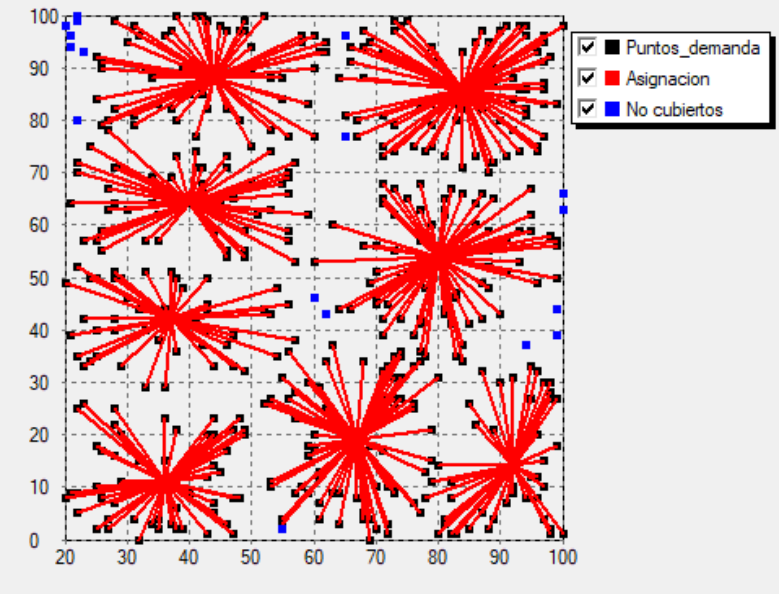

Figura 6: Gráfico archivo .pnl m=n=500

- **Para m=n=800:**

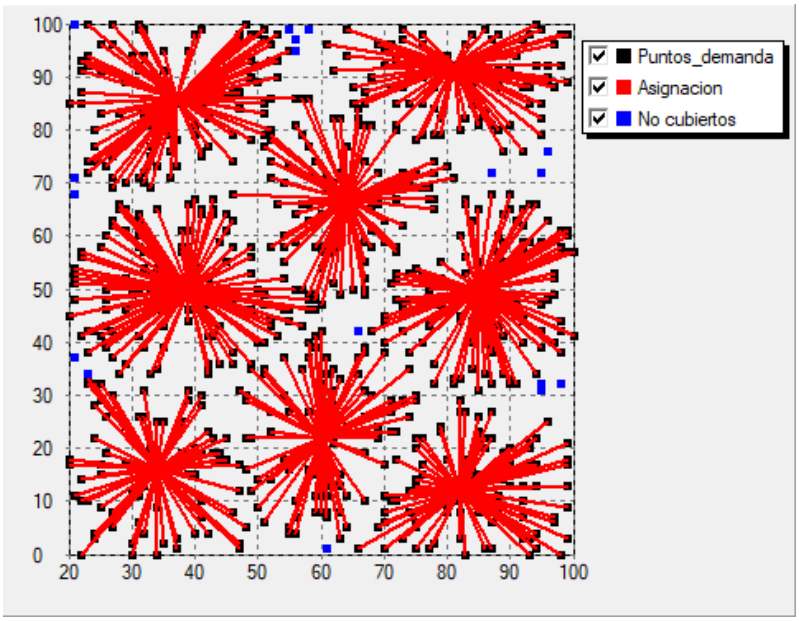

Figura 7: Gráfico archivo .pnl m=n=800

- **Para m=n=1000:**

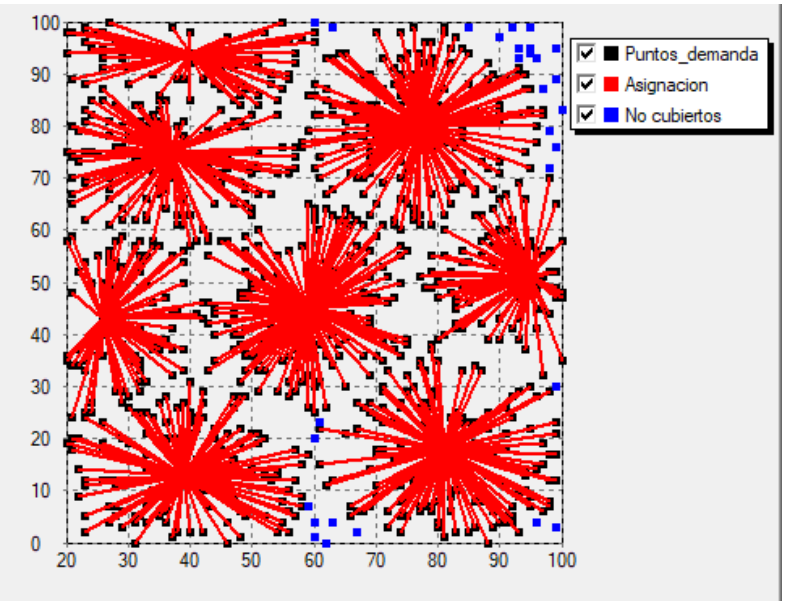

Figura 8: Gráfico archivo .pnl m=n=1000

A medida que el número de puntos de demanda (el mismo que de servicios) va aumentando, también lo hace el número de puntos de demanda no cubiertos, aunque estos lo hacen en una proporción mucho menor, evidenciando que para los archivos más grandes la demanda queda cubierta casi en su totalidad.

Capítulo 4

4. Modelos de cubrimiento adicionales

Existen además, otro tipo de modelos de cubrimiento, **no estudiados durante el Grado**, de los que podría resultar interesante hablar en este trabajo, estos son los modelos de cubrimiento de varios equipos y los de cubrimiento reforzado.
Por ello se dedica este capítulo a la explicación y formalización de algunos de los más destacables.

4.1. Modelos TEAM (Tandem Equipment Allocation Model)

Se pueden usar para manejar varios tipos de vehículos (ambulancias) o servicios.

Es una extensión del modelo de Cubrimiento Máximo.

Introducido por SCHILLING, D., D. ELZINGA, J. COHON, R. CHURCH, and C. REVELLE (1979) (19).

Está muy bien descrito en el contexto de ambulancias, en BROTCORNE, L., G. LAPORTE, F. SEMET (2003) (3).

4.1.1. Modelo TEAM Básico

Se tienen dos tipos de servicios o vehículos (ambulancias), A y B.

Sean p^A y p^B el número de servicios o vehículos de tipo A y B disponibles, y sean t^A y t^B los tiempos de cubrimiento estándar fijados para cada tipo de servicio, se define:

$N_i^A = \{j \in N \mid t_{ij} \leq t^A\}$
$N_i^B = \{j \in N \mid t_{ij} \leq t^B\}$

Sea $x_j^A \in \{0,1\}$, una variable binaria igual a 1 si un servicio tipo A está localizado en el punto $j \in N$, y lo mismo para x_j^B, $j \in N$.

Sea h_i la demanda del punto i, e $y_i \in \{0,1\}$, una variable binaria que vale 1 si el punto i está cubierto por los dos tipos de servicios, para $i \in M$.

Modelo TEAM:

$$\max \sum_{i \in M} h_i \cdot y_i$$

Sujeto a

$$\sum_{j \in N_i^A} x_j^A \geq y_i \quad \forall i = 1 \dots m$$

$$\sum_{j \in N_i^B} x_j^B \geq y_i \quad \forall i = 1 \dots m$$

$$\sum_{j \in N} x_j^A = p^A$$

$$\sum_{j \in N} x_j^B = p^B$$

$$x_j^A \leq x_j^B \quad \forall j = 1 \dots n$$

$$x_j^A = \begin{cases} 1 & \text{si un servicio tipo A está localizado en el punto } j \in N \\ 0 & si\ no \end{cases}$$

$$x_j^B = \begin{cases} 1 & \text{si un servicio tipo B está localizado en el punto } j \in N \\ 0 & si\ no \end{cases}$$

$$y_i = \begin{cases} 1 & \text{si el punto i está cubierto por los dos tipos de servicios} \\ 0 & si\ no \end{cases}$$

La última restricción, $x_j^A \leq x_j^B$, impone una jerarquía entre ambos tipos de servicios, de forma que sólo puede haber un servicio o vehículo tipo A si hay uno B.

Esta restricción puede no estar presente en algunos casos.

4.1.2. Modelo MOTEAM (MultiObjective TEAM)

Cuando lo que interesa es diferenciar el cubrimiento de cada tipo de servicio, entonces se tiene un modelo bi-objetivo.

El modelo MOTEAM fue introducido por SCHILLING, D., D. ELZINGA, J. COHON, R. CHURCH, and C. REVELLE (1979) (19).
Se describe también en COHON, J. (1978) (8).

Sea h_i^A y h_i^B, para i ∈ M, la demanda de cada tipo de servicio o vehículo, se diferencia entre $y_i^A \in \{0,1\}$ e $y_i^B \in \{0,1\}$ para indicar que se tiene o no el cubrimiento por cada tipo de servicio.

Modelo MOTEAM:

$$\max \quad Z_1 = \sum_{i \in M} h_i^A \cdot y_i^A, \qquad Z_2 = \sum_{i \in M} h_i^B \cdot y_i^B$$

Sujeto a

$$\sum_{j \in N_i^A} x_j^A \geq y_i^A \quad \forall i = 1 \dots m$$

$$\sum_{j \in N_i^B} x_j^B \geq y_i^B \quad \forall i = 1 \dots m$$

$$\sum_{j \in N} x_j^A = p^A$$

$$\sum_{j \in N} x_j^B = p^B$$

$$x_j^A \leq x_j^B \quad \forall j = 1 \dots n$$

$x_j^A = \begin{cases} 1 & \text{si un servicio tipo A está localizado en el punto } j \in N \\ 0 & si\ no \end{cases}$

$x_j^B = \begin{cases} 1 & \text{si un servicio tipo B está localizado en el punto } j \in N \\ 0 & si\ no \end{cases}$

$y_i^A = \begin{cases} 1 & \text{si el punto i está cubierto por el tipo de servicio A} \\ 0 & si\ no \end{cases}$

$y_i^B = \begin{cases} 1 & \text{si el punto i está cubierto por el tipo de servicio B} \\ 0 & si\ no \end{cases}$

Para resolver el problema bi-objetivo, se usa el método de combinación convexa, resolviendo el problema con objetivo:

$$Z = \sum_{i \in M} h_i^A \cdot y_i^A + \lambda \cdot \sum_{i \in M} h_i^B \cdot y_i^B$$

Para un rango de valores de λ.

Posteriormente, se aplica el modelo al ejemplo 30x30 de TOREGAS, C., R. SWAIN, C. REVELLE, and L. BERGMAN (1971) (21) obteniendo una serie de puntos no dominados que dan una aproximación de la frontera eficiente.

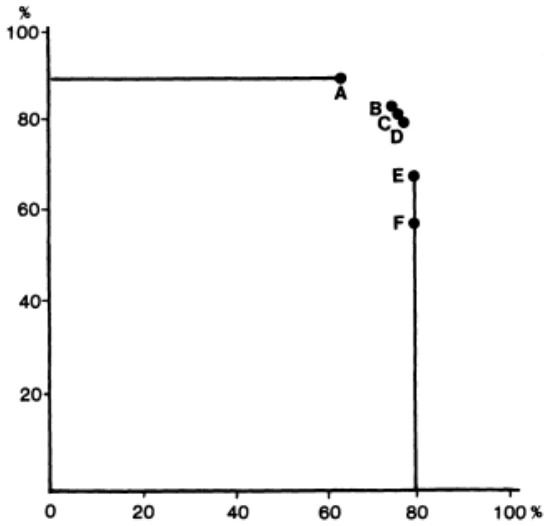

Figura 9: Frontera Eficiente

4.2. Modelos FLEET

Se tiene un modelo FLEET (Facility-Location, Equipment-Emplacement Technique) si se permite que los dos tipos de servicios o vehículos tengan su localización independiente, es decir, pueden estar en tándem o solos.De esta manera, se puede tener un mejor cubrimiento.

El modelo FLEET de SCHILLING, D., D. ELZINGA, J. COHON, R. CHURCH, and C. REVELLE (1979) (19) define:

Sea $Z_j \in \{0,1\}$ una variable binaria que vale 1 si se localiza un servicio en el punto j, para $j \in N$.

Modelo FLEET:

$$\max \sum_{i \in M} h_i \cdot y_i$$

Sujeto a

$$\sum_{j \in N_i^A} x_j^A \geq y_i \quad \forall i = 1 \dots m$$

$$\sum_{j \in N_i^B} x_j^B \geq y_i \quad \forall i = 1 \dots m$$

$$\sum_{j \in N} x_j^A = p^A$$

$$\sum_{j \in N} x_j^B = p^B$$

$$\sum_{j \in N} Z_j = p$$

$$x_j^A \leq Z_j \quad \forall \mathrm{j} \in N^*$$

$$x_j^B \leq Z_j \quad \forall \mathrm{j} \in N^*$$

$$\mathrm{x_j^A} = \begin{cases} 1 & \text{si un servicio tipo A está localizado en el punto j} \in \mathrm{N} \\ 0 & \textit{si no} \end{cases}$$

$$\mathrm{x_j^B} = \begin{cases} 1 & \text{si un servicio tipo B está localizado en el punto j} \in \mathrm{N} \\ 0 & \textit{si no} \end{cases}$$

$$\mathrm{y_i} = \begin{cases} 1 & \text{si el punto i está cubierto por los dos tipos de servicios} \\ 0 & \textit{si no} \end{cases}$$

Donde p es el número de servicios que van a ser abiertos, y N* ⊂ N es el conjunto de puntos donde podrían situarse nuevos puntos de servicios.

En la formulación se supone que los puntos de N y N* ya tienen puntos de servicio instalados, y allí pueden ubicarse vehículos o equipos.

En REVELLE, C. (1989) (18) se dan otras restricciones sobre el número total de vehículos conjunto.

Así, en vez de

$$\sum_{j \in N} x_j^A = p^A$$

$$\sum_{j \in N} x_j^B = p^B$$

se pone

$$\sum_{j \in N} x_j^A + \sum_{j \in N} x_j^B = p^{A+B}$$

Es claro que este tipo de restricciones, así como las

$$x_j^A \leq x_j^B \quad \forall j = 1 \dots n$$

pueden depender de situaciones concretas que habrá que averiguar.

4.3. Modelos de Cubrimiento Reforzado

Dirigidos a situaciones en las que se demanda un servicio y no es posible realizar dicha demanda en ese instante, por ejemplo, al realizar una llamada para solicitar una ambulancia resulta que la línea está ocupada.

Se pueden asignar entonces varios servicios a cada punto de demanda, de manera que si el punto de servicio situado a la distancia mínima está ocupado, que la demanda pueda ser atendida por otro servicio.

4.3.1. Modelo de Berlin (1972) (2) y Daskin-Stern (1981) (11)

En este modelo se supone un número p de servicios disponibles mayor igual que el mínimo del problema de Cubrimiento Total (Set Covering), de forma que se puede garantizar que todo punto de demanda queda cubierto (a una distancia o tiempo menor o igual que el establecido como límite).

Se maximiza el número total de cubrimientos adicionales o de refuerzo.

Modelo de Berlin, Daskin-Stern:

$$\max \sum_{i \in M} r_i$$

Sujeto a

$$\sum_{j \in N_i} x_j = 1 + r_i \quad \forall i = 1 \dots m$$

$$\sum_{j \in N} x_j = p$$

$$x_j = \begin{cases} 1 & \text{si un servicio está localizado en el punto } j \in N \\ 0 & si\ no \end{cases}$$

$$r_i \geq 0 \quad \in \mathbb{Z} \quad \forall i \in M$$

Donde r_i es el número de servicios de refuerzo en el punto de demanda $i \in M$.

Cuando p es el mínimo número de servicios (Set Covering Problem), la solución de este problema es el óptimo alternativo del problema de Cubrimiento Total, que maximiza el número total de servicios de refuerzo o adicionales.

Con este objetivo no se tiene en cuenta que los puntos que tienen mayor demanda deberían ser los que tengan más cubrimientos reforzados.

BENEDICT, J. M. (1983) (1) y EATON, D., M. DASKIN, D. SIMMONS, B. BULLOCH, and G. JANSMA (1985) (13) usan el siguiente objetivo para subsanar ese problema:

$$\max \sum_{i \in M} h_i \cdot r_i$$

No queda totalmente solucionado ya que, con este objetivo, puede haber puntos de demanda con muchos servicios de refuerzo asignados y otros que tengan únicamente el inicial.

Para solucionar esto, HOGAN, K. and C. REVELLE (1986) (15) desarrollaron los modelos BACOP.

4.3.2. Modelo BACOP1

Se trata de maximizar la demanda doblemente cubierta o más, es decir, el cubrimiento reforzado.
Con este modelo todo punto de demanda queda cubierto al menos una vez, para relajar esta condición HOGAN, K. and C. REVELLE (1986) (15) desarrollaron posteriormente el modelo BACOP2.

Sea $u_i \in \{0,1\}$ una variable binaria que vale 1 si y sólo si el punto de demanda i es cubierto dos veces o más con la distancia de cubrimiento, dc, establecida, $i \in M$.

Modelo BACOP1:

$$\max \sum_{i \in M} h_i \cdot u_i$$

Sujeto a

$$\sum_{j \in N_i} x_j \geq 1 + u_i \quad \forall i = 1 \dots m$$

$$\sum_{j \in N} x_j = p$$

$$x_j = \begin{cases} 1 & \text{si un servicio está localizado en el punto } j \in N \\ 0 & si\ no \end{cases}$$

$$u_i = \begin{cases} 1 & \text{si el punto i es cubierto dos veces o más con distancia de cubrimiento} \\ 0 & si\ no \end{cases}$$

4.3.3. Modelo BACOP2

Además de la variable binaria u_i indicando que el punto $i \in M$ está cubierto por lo menos dos veces se tiene $y_i \in \{0,1\}$, una variable binaria que vale 1 si el punto i es cubierto una vez.

Modelo BACOP2 (bi-objetivo):

$$\max \quad Z_1 = \sum_{i \in M} h_i \cdot y_i, \qquad Z_2 = \sum_{i \in M} h_i \cdot u_i$$

Sujeto a

$$\sum_{j \in N_i} x_j \geq u_i + y_i \quad \forall i = 1 \dots m$$

$$\sum_{j \in N} x_j = p$$

$$u_i \leq y_i \quad \forall i = 1 \dots m$$

$x_j = \begin{cases} 1 & \text{si un servicio está localizado en el punto } j \in N \\ 0 & si\ no \end{cases}$

$u_i = \begin{cases} 1 & \text{si el punto i es cubierto dos veces o más con distancia de cubrimiento} \\ 0 & si\ no \end{cases}$

$y_i = \begin{cases} 1 & \text{si el punto i es cubierto una vez} \\ 0 & si\ no \end{cases}$

HOGAN, K. and C. REVELLE (1986) (15) indican que, a veces, una pequeña reducción en el primer cubrimiento implica una gran mejora en el cubrimiento reforzado.

4.3.4. Modelo de Storbeck

En STORBECK, J. E. and R. V. VOHRA (1988) (20), Storbeck plantea las desviaciones por exceso (y_i^-) y por defecto (y_i^+) respecto de 1, del mismo número de veces que el punto i queda cubierto, es decir:

$$\sum_{j \in N_i} x_j = 1 + y_i^+ - y_i^- \quad \forall i = 1 \dots m$$

$$y_i^+ \geq 0 \quad \in \mathbb{Z} \quad \forall i \in M$$

Para que y_i^+ nunca sea negativo, y_i^- tiene que ser una variable binaria.
Entonces si y_i^- es 0 el punto de demanda i queda cubierto, y si yi-=1 el punto de demanda i no queda cubierto.
y_i^+ indica el cubrimiento reforzado, o el número de instalaciones adicionales (además de la primera).

Modelo de Storbeck (biobjetivo):

$$\max \quad Z_1 = \sum_{i \in M} h_i \cdot y_i^+$$

$$\min \quad Z_2 = \sum_{i \in M} h_i \cdot y_i^-$$

Sujeto a

$$\sum_{j \in N_i} x_j = 1 + y_i^+ - y_i^- \quad \forall i = 1 \dots m$$

$$\sum_{j \in N} x_j = p$$

$$x_j = \begin{cases} 1 & \text{si un servicio está localizado en el punto } j \in N \\ 0 & \textit{si no} \end{cases}$$

$$y_i^- = \begin{cases} 1 & \text{si el punto de demanda i no es cubierto} \\ 0 & \textit{si el punto de demanda i es cubierto} \end{cases}$$

$$y_i^+ \geq 0 \quad \in \mathbb{Z} \quad \forall i \in M$$

El objetivo Z_1 expresa la maximización de la población que recibe cubrimiento reforzado, mientras que Z_2 refleja la minimización de la población que se queda sin cubrir, lo que equivale a la maximización de la demanda cubierta por al menos una instalación.

Storbeck añade otras restricciones muy interesantes y flexibles.
Sea k_i, para cada $i \in M$, el número máximo de cubrimientos de refuerzo que queremos que ocurran, entonces se añaden las restricciones:

$$y_i^+ + k_i \cdot y_i^- \leq k_i \quad \forall i \in M \qquad (*)$$

Puestas de la forma:

$$y_i^+ \leq k_i - k_i \cdot y_i^- \quad \forall i \in M$$

Implican lo siguiente:

a) Si el punto de demanda i resulta no ser cubierto, y por lo tanto $y_i^-=1$, queda $y_i^+ \leq 0$, luego yi+=0.

b) Si el punto de demanda i resulta cubierto, $y_i^-=0$ y queda $y_i^+ \leq k_i$, la cota superior sobre el número de cubrimientos de refuerzo.
Por lo tanto, la demanda del punto i podría ser cubierta por ki+1 instalaciones.

La posibilidad de definir cotas superiores sobre el cubrimiento de los puntos de demanda en función de las demandas da a este modelo una gran flexibilidad, que no puede obtenerse en otros modelos de cubrimiento reforzado como el BACOP2 de HOGAN, K. and C. REVELLE (1986) (15).

La restricción (*) también implica que las variables y_i^+ e y_i^- son "complementarias", es decir, nunca pueden ser las dos positivas.

Si $y_i^+ > 0$ es que el punto i queda sobre-cubierto, luego $y_i^- = 0$.
Y si $y_i^- = 1$, de (*) queda que $y_i^+ \leq 0$, luego $y_i^+ = 0$.

En STORBECK, J. E. and R. V. VOHRA (1988) (20) se escribe el objetivo Z2 en forma de maximización de la población cubierta:

$$\max \quad Z_2' = \sum_{i \in M} h_i - Z_2$$

De esta forma, se pueden combinar los dos objetivos:

$$\max \ Z = \ w_1 \cdot Z_1 + w_2 \cdot Z_2'$$

donde w_1 y w_2 son pesos relativos.

Por lo tanto, usan el método de pesos únicamente. Y lo ilustran con el ejemplo de 21 nodos de CURRENT, J. and J. STORBECK (1987) (9).

Los pesos w_1 y w_2 son seleccionados con el método NISE de COHON, J. (1978) (8), y por ejemplo para p=4 y k=3 (suponiendo k_i=3 $\forall i \in M$), obtienen la tabla de valores de Z_1 (cubrimiento múltiple) y Z_2' (cubrimiento máximo):

Solution Values Generated by the NISE Method ($p = 4, k = 3$)

Noninferior Point	Weights W_1	Weights W_2	Multiple Coverage Z_1	Maximal Coverage Z_2
A	.35	1	1,300	6,300
B	.50	1	2,500	5,750
C	.71	1	3,600	5,150
D	.83	1	3,950	4,900
E	.90	1	5,650	3,450
F	1	0	6,050	2,950

Figura 10: NISE Cohon (1978)

4.3.5. Backup Covering Location Models

En CURTIN, K. M. and K. HAYSLETT-MCCALL and FANG QIU (2007) (10) por un lado se usa el **modelo de Cubrimiento Máximo (MCLP)**, que ellos denominan **PPAC** (Police Patrol Area Covering):

$$\max \sum_{i \in M} h_i \cdot y_i$$

Sujeto a

$$\sum_{j \in N_i} x_j \geq y_i \quad \forall i = 1 \dots m$$

$$\sum_{j \in N} x_j = p$$

$$x_j = \begin{cases} 1 & \text{si un servicio está localizado en el punto } j \in N \\ 0 & si\ no \end{cases}$$

$$y_i = \begin{cases} 1 & \text{si el punto i es cubierto} \\ 0 & si\ no \end{cases}$$

Para favorecer el cubrimiento reforzado simplemente se toma:

$$y_i \in \{0,1, \dots, p\} \qquad \forall i \in M$$

Esta formulación de cubrimiento reforzado difiere mucho de otras conocidas, pues el cubrimiento reforzado no es un objetivo secundario.

Desafortunadamente, la solución de este modelo suele dar soluciones agrupadas en torno a los puntos de mayor demanda, de forma que el resto de puntos quedan sin cubrir.

Para solucionar esta cuestión se plantea el modelo bi-objetivo para efectuar el trade-off (situación que implica la pérdida de una cosa y la ganancia de otra) entre el Cubrimiento Máximo ($y_i \in \{0,1\}$) y el cubrimiento backup ($y_i \in \{0,1, \dots, p\}$).

Para resolver el problema bi-objetivo se usa el método de restricción, y se obtienen gráficas como esta:

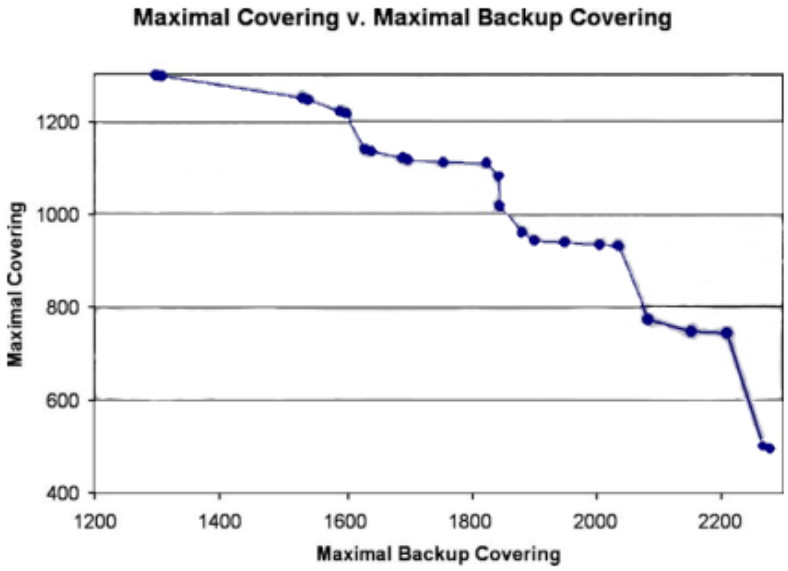

Figura 11: Maximal Backup Covering

4.3.6. Double Standard Models (DSM)

De BROTCORNE, L., G. LAPORTE, F. SEMET (2003) (3), en este modelo, hay dos distancias o tiempos estándar r_1 y r_2, con $r_1 < r_2$.
Toda la demanda va a ser cubierta a la distancia máxima de r_2, mientras que una proporción prefijada α de toda la demanda va a quedar cubierta a la distancia máxima de r_1.

El modelo busca maximizar la demanda cubierta dos veces a la distancia r_1, con un número total de p ambulancias, y además con un máximo de p_j ambulancias situadas en $j \in N$.

Llamando $N_i^1 = \{j \in N \mid t_{ij} \leq r^1\}$ y $N_i^2 = \{j \in N \mid t_{ij} \leq r^2\}$ se introducen las variables:

$$x_j \geq 0 \quad \in \mathbb{Z} \qquad \forall j \in N$$

$$y_i^1 = \begin{cases} 1 & \text{si el punto i es cubierto una vez} \\ 0 & \textit{si no} \end{cases}$$

$$y_i^2 = \begin{cases} 1 & \text{si el punto i es cubierto dos veces} \\ 0 & \textit{si no} \end{cases}$$

En ambos casos dentro de r_1.

Modelo DSM:

$$\max \sum_{i \in M} h_i \cdot y_i^2$$

Sujeto a

$$\sum_{j \in N_i^2} x_j \geq 1 \quad \forall i = 1 \dots m$$

$$\sum_{i \in M} h_i \cdot y_i^1 \geq \alpha \cdot \sum_{i \in M} h_i$$

$$\sum_{j \in N_i^1} x_j \geq y_i^1 + y_i^2 \quad \forall i = 1 \dots m$$

$$y_i^2 \leq y_i^1 \qquad \forall i = 1 \dots m$$

$$\sum_{j \in N} x_j = p$$

$$x_j \leq p_j \quad \forall j \in N$$

$$x_j \geq 0 \in \mathbb{Z} \quad \forall j \in N$$

$$y_i^1 = \begin{cases} 1 & \text{si el punto i es cubierto una vez} \\ 0 & \textit{si no} \end{cases}$$

$$y_i^2 = \begin{cases} 1 & \text{si el punto i es cubierto dos veces} \\ 0 & \textit{si no} \end{cases}$$

Conclusiones

El objetivo de este trabajo fin de grado, según se indicó en la introducción es, dado un problema de Cubrimiento Máximo, llevar a cabo una comparación de los distintos métodos de resolución para tratar de obtener la disposición de los servicios que mejor atiende las diferentes demandas de ese problema.

Este análisis se realizó para cada uno de los dos tipos de archivos de datos que se ofrecían, concluyendo finalmente que Greedy + Búsqueda Local y GRASP son los métodos que más se aproximan en la mayoría de los casos al óptimo global, es decir, al porcentaje de demanda total cubierta para el Solver de Xpress.

Analizando los tiempos obtenidos en cada método, para cada uno de los tipos de archivos de datos, se comprueba que dentro de los que más se aproximan a la solución exacta u óptima, Greedy + Búsqueda Local es el que menos tiempo de ejecución requiere y, es por tanto, una de las mejores opciones a la hora de tratar de dar solución al problema de Cubrimiento Máximo (MCLP).

También se planteó estudiar la modelización de otros modelos de Programación Entera no vistos durante el Grado, y es por eso por lo que se ha dedicado el Capítulo 4 a la explicación y formalización de algunos de los modelos más destacables de cubrimiento de varios equipos y de cubrimiento reforzado.

Referencias

1. BENEDICT, J. M. (1983) *Three Hierarchical Objective Models Which Incorporate the Concept of Excess Coverage to Locate EMS Vehicles or Hospitals*, M.S. Thesis, Department of Civil Engineering, Northwestern University, Evanston, Ill

2. BERLIN, G. N. (1972) *Facility Location and Vehicle Allocation for Provision of An Emergency Service* , Ph.D. Dissertation, The Johns Hopkins University, Baltimore

3. BROTCORNE, L., G. LAPORTE, F. SEMET (2003) *Ambulance location and relocation models*, EJOR 47, pp. 451-463

4. CANÓS DARÓS, M. J., M. MOCHOLÍ ARCE and M. MARTÍNEZ ROMERO (2009) *Eficiencia versus equidad en localización: aplicación al diseño de infraestructuras*, Rect@ **Vol 10**, pp. 59-76

5. CHUNG, C-H. (1986) *Recent Applications of the M.C.L.P. model*, J. Opl Res. Soc. **Vol. 37**, No. 8, pp. 735-746

6. CHURCH, R. and C. REVELLE (1974) *The maximal covering location problem*, Pap. region. Sci. Ass. 32, pp. 101-118

7. CHURCH, R. and C. REVELLE (1976) *Theoretical and computational links between the p-median, location set-covering and the maximal covering location problems*, Geogr. Analysis **Vol. VIII**, pp. 406-415

8. COHON, J. (1978) *Multiobiective Programming and Planning*, New York: Academic Press

9. CURRENT, J. and J. STORBECK (1987) *Satisficing Solutions to Infeasible Set Partitions*, Enoironment and Planning B 14, pp. 183-92

10. CURTIN, K. M. and K. HAYSLETT-MCCALL and FANG QIU (2007) *Determining Optimal Police Patrol Areas with Maximal Covering and Backup Covering Location Models*, Network and Spatial Economics 10, pp. 125-145

11. DASKIN, M. S. and E. N. STERN (1981) *A Hierarchical Objective Set Covering Model for Emergency Medical Service Deployment* , Transportation Sci., 15, 2, pp. 137-152

12. DASKIN, M. S. (2013) *Network and Discrete Location Models, Algorithms and Applications*, Second Edition, Ed. Wiley

13. EATON, D., M. DASKIN, D. SIMMONS, B. BULLOCH, and G. JANSMA (1985) *Determining Emergency Medical Service Vehicle Deployment in Austin, Texas*, Interfaces 15, 1, pp. 96-108

14. EISELT, H. A. and V. MARIANOV (2011) *Foundations of Location Analysis*, Ed. Springer

15. HOGAN, K. and C. REVELLE (1986) *Concepts and Applications of Backup Coverage,* Management Science **Vol. 32**, No. 11, pp. 1434-1444

16. RESENDE, M. G. C. (1998) *Computing Approximate Solutions of the Maximum Covering Problem with GRASP*, Journal of Heuristics, 4, pp. 161-177

17. RESENDE, M. G. C. and C. C. RIBEIRO (2016) *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures*, Handbook of metaheuristics

18. REVELLE, C. (1989) *Review, extension and prediction in emergency service sitting models,* EJOR 40, pp. 58-69

19. SCHILLING, D., D. ELZINGA, J. COHON, R. CHURCH, and C. REVELLE (1979) *The Team/Fleet Models for Simultaneous Facility and Equipment Siting*, Transportation Science , **Vol. 13**, No. 2, pp. 163-175

20. STORBECK, J. E. and R. V. VOHRA (1988) *A Simple Trade-off Model for Maximal and Multiple Coverage,* Geographical Analysis 20, pp. 220-230

21. TOREGAS, C., R. SWAIN, C. REVELLE, and L. BERGMAN (1971) *The Location of Emergency Service Facilities,* Opns. Res. 19, pp. 1363-1373

Anexo Código Xpress Mosel

```
(!      Problema de Cubrimiento Máximo

        Se leen datos en dos formatos diferentes dependiendo de la variable
formato_datos.

        Si vale 1 se leen datos de problemas con coordenadas y se calculan las distancias
euclídeas.

        Si vale 2, se leen datos de CyL con demandas.

        En ambos caso se necesita una distancia de cubrimiento dc.

        En los dos casos, se define la matriz A = (a(i,j)).


 !)

model ModelName
uses "mmxprs";
uses "mmsystem"
uses "mmive"


forward procedure solucion_xpress
forward procedure greedy
forward procedure greedy_aleatorizado
forward procedure busqueda_local
forward function demanda_que_cubre(xp:array(range)of integer):integer
forward procedure grasp
forward procedure grafico_solucion(ind_ab:array(range)of integer)




declarations
  m,n:integer
  archivo_datos = "s50_1.pln" !Ejemplo Formato 1 coordenadas pnl (plano) que
explicamos:
  !Todos los puntos de demanda pueden ser puntos de servicio por eso m=n
  !fila 1 puntos de demanda = puntos de servicio
  !fila 2 Coordenada X (primera columna), Coordenada Y (segunda columna), la
demanda del punto de demanda 1
  !fila 3 Coordenada X (primera columna), Coordenada Y (segunda columna), la
demanda del punto de demanda 2
  !.....
```

```
  formato_datos = 1 !Seleccionamos tipo de fichero de datos Formato 1


  !archivo_datos = "aint13.dat" !Ejemplo Formato 2 .cyl que explicamos:
  !fila 1 puntos de demanda
  !fila 2 puntos de servicio
  !la demanda del punto i-esimo, en este caso población del punto i
  !matriz mxn de distancias de desplazamiento (o tiempos) para cada i = 1,...,m, j =
1,...,n

  !formato_datos = 2 !Seleccionamos tipo de fichero de datos Formato 2

end-declarations



!Leemos datos
fopen(archivo_datos,F_INPUT)
if(formato_datos = 1)then
        read(n)
        m:=n
elif(formato_datos =2)then
        read(m)
        read(n)
end-if




declarations
        pdemanda= 1..m
        pservicio = 1..n
        cx, cy:array(pservicio)of integer
        dist:array(pdemanda,pservicio)of integer
        dem:array(pdemanda)of integer
        dem_total:integer
        a:array(pdemanda,pservicio)of integer
        x:array(pservicio)of mpvar
        z:array(pdemanda)of mpvar
        xsol:array(pservicio)of integer
        p = 8
        dc = 20


        !parámetros búsqueda local
        zheur:integer
        tiempo_busqueda_local: real
        abiertos=1..p
        no_abiertos=1..(n-p)
```

```
        ind_ab,ind_abp:array(abiertos)of integer
        ind_noab,ind_noabp:array(no_abiertos)of integer

        ! parámetros algoritmo greedy aleatorizado
        nrcl = 3 ! tamaño de la lista de candidatos RCL
        niter = 10 ! número de iteraciones



end-declarations


writeln("Archivo de datos ", archivo_datos)


if(formato_datos = 1)then
        forall(j in pservicio)read(cx(j),cy(j),dem(j))
        forall(i in pdemanda,j in pservicio)dist(i,j):=round(sqrt((cx(i)-cx(j))^2 + (cy(i)-
cy(j))^2))
        forall(i in pdemanda,j in pservicio)do
                if(dist(i,j)<=dc)then
                        a(i,j):=1
                else
                        a(i,j):=0
                end-if
        end-do
elif(formato_datos=2)then
        forall(i in pdemanda)read(dem(i))
        forall(i in pdemanda,j in pservicio)read(dist(i,j))
        forall(i in pdemanda,j in pservicio)do
                if(dist(i,j)<=dc)then
                        a(i,j):=1
                else
                        a(i,j):=0
                end-if
        end-do
end-if

dem_total:=sum(i in pdemanda)dem(i)
writeln("\nDemanda total: ",dem_total)




solucion_xpress
greedy

busqueda_local
demandacubierta:=demanda_que_cubre(ind_ab)
```

```
writeln("\n\n\n(3) Solucion greedy + busqueda local\n\n Demanda total cubierta = ",
demandacubierta)
writeln("\nPorcentaje de demanda total cubierta = ", (demandacubierta/dem_total)*100,
" %")
writeln("\n\tTiempo: ",gettime-tiempo_busqueda_local, " s")

greedy_aleatorizado
grasp
grafico_solucion(ind_ab)




!Modelo de cubrimiento maximo
procedure solucion_xpress

        tiempo_xpress:=gettime

        demmax:=sum(i in pdemanda)dem(i)*z(i)

                forall(i in pdemanda)rescub(i):=
                sum(j in pservicio)a(i,j)*x(j)>=z(i)
        rt:=sum(j in pservicio)x(j)=p
                forall(j in pservicio)x(j)is_binary
                        forall(i in pdemanda)z(i)is_binary
                                maximize(demmax)
        writeln("\nSolucion con p = ", p, ", dc = ", dc)
        writeln("\n\n\n(1) Solucion Xpress\n\n Demanda total cubierta = ", getobjval)
        writeln("\nPorcentaje de demanda total cubierta = ", (getobjval/dem_total)*100,
" %")

        writeln("\n\tTiempo: ",gettime-tiempo_xpress, " s")

end-procedure




!Método greedy
procedure greedy

        tiempo_greedy:=gettime

        declarations
        Z:array(pservicio)of integer
```

```
cubierto:array(pdemanda)of integer
end-declarations

demandacubierta:=0

forall(t in 1..p)do !hacemos p iteraciones
!calculo Z(j) demanda aun no cubierta que cubre cada  punto de servicio j
        forall(j in pservicio|xsol(j)=0)do
        Z(j):=0
                forall(i in pdemanda)do
                        if (cubierto(i)=0 and a(i,j)=1) then
                        Z(j):=Z(j)+dem(i)
                end-if
                end-do
        end-do
!Encuentro el maximo de Z(j)
        zmax:=-999 !empiezo en numero negativo porque voy a maximizar
        forall(j in pservicio|xsol(j)=0)do
                if(Z(j)>zmax)then
                        zmax:=Z(j)
                        jmax:=j
                end-if
        end-do

        xsol(jmax):=1
        ind_ab(t):=jmax

        forall(i in pdemanda)do
                if(cubierto(i)=0 and a(i, jmax)=1)then
                        cubierto(i):=1
                        demandacubierta:=demandacubierta+dem(i)
                end-if
        end-do
end-do


!Puntos no abiertos:
na:=0
forall(j in pservicio)do
        if(xsol(j)=0)then
        na:=na+1
        ind_noab(na):=j
        end-if
end-do



        writeln("\n\n\n(2) Solucion greedy\n\n Demanda total cubierta = ",
demandacubierta)
```

```
        writeln("\nPorcentaje de demanda total cubierta = ",
(demandacubierta/dem_total)*100, " %")

        writeln("\n\tTiempo: ",gettime-tiempo_greedy, " s")

end-procedure




!Búsqueda local


!Función para evaluar la demanda cubierta
function demanda_que_cubre(xp:array(range)of integer):integer

        demcub:=0
        forall(i in pdemanda)do
                np:=0
                forall(j in abiertos)if(dist(i,xp(j))<=dc)then np:=np+1;end-if
                if(np >=1)then
                        demcub:=demcub+dem(i)
                end-if
        end-do

        returned:=demcub
end-function




procedure busqueda_local

        tiempo_busqueda:=gettime

        final:=0

        while(final = 0)do
                zheur:=demanda_que_cubre(ind_ab)

                mejoramax:=-999
                !Dos indices k y s que intercambiamos
                forall(k in abiertos)do
```

```
                    forall(s in no_abiertos)do
                        ind_abp:=ind_ab                !copia
                        ind_noabp:=ind_noab!copia

                        delcell(ind_abp(k))
                        ind_abp(k):=ind_noabp(s)
                        delcell(ind_noabp(s))
                        ind_noabp(s):=ind_ab(k)


                        zheurp:=demanda_que_cubre(ind_abp)
                        mejora:=zheurp-zheur

                        if(mejora > mejoramax)then
                            mejoramax:=mejora
                            kmax:=k
                            smax:=s
                        end-if
                    end-do
                end-do

                if(mejoramax <=0)then
                    final:=1

                else
                    delcell(ind_ab(kmax))
                    ind_ab(kmax):=ind_noabp(smax)
                    delcell(ind_noab(smax))
                    ind_noab(smax):=ind_abp(kmax)
                    zheur:=zheur+mejoramax
                end-if

        end-do

        tiempo_busqueda_local:=gettime-tiempo_busqueda

end-procedure
```

```
!Método greedy aleatorizado
procedure greedy_aleatorizado
```

```
tiempo_greedy_aleatorizado:=gettime
zmax:=-1000
forall(iter in 1..niter)do

        forall(j in pservicio) xsol(j):=0
        forall(i in pdemanda)visitado(i):=0
        zheur:=0

        forall(f in 1..p)do
                ! primero calculo la demanda aún no cubierta
                ! que cubre cada punto de servicio no abierto
                forall (j in pservicio | xsol(j)=0) do
                        g(j):=0
                forall(i in pdemanda|visitado(i)=0)do
                if(dist(i,j)<=dc)then g(j):=g(j)+dem(i)
                end-if
                end-do
                        marca(j):=0
                end-do
                !calculo los nrcl  puntos de servicio no abiertos que cubren más
demanda
                forall (k in 1..nrcl) do
                        aux:=-9999
                        forall(j in pservicio|xsol(j)=0 and marca(j)=0)do
                                if (g(j)>aux) then
                                        aux:=g(j)
                                        jaux:=j
                                end-if
                        end-do
                        ind(k):=jaux
                        marca(jaux):=1
                end-do
                jaux:=ind(ceil(random*nrcl))
                xsol(jaux):=1
                zheur:=zheur+g(jaux)
                forall(i in pdemanda| dist(i,jaux)<=dc and visitado(i)=0) do
                        visitado(i):=1
                end-do
        end-do ! final de forall(f in 1..p)

        ! write("\niter ",iter,", z = ",zheur)

        if(zheur > zmax)then
        zmax:=zheur
        end-if

end-do !Cerramos el número de iteraciones niter
```

```
        writeln("\n\n\n(4) Solucion greedy aleatorizado\n\n Demanda total cubierta
(mejor solucion) = ", zmax)
        writeln("\nPorcentaje de demanda total cubierta = ", (zmax/dem_total)*100, "
%")

        writeln("\n\tTiempo: ",gettime-tiempo_greedy_aleatorizado, " s")

end-procedure




!ALGORITMO GRASP
procedure grasp

        tiempo_grasp:=gettime
        zmax:=-1000

        forall(iter in 1..niter)do
                !Inicializaciones:
                forall(j in pservicio)xsol(j):=0
                forall(i in pdemanda)visitado(i):=0
                zheur:=0
                ! Método de construcción greedy aleatorizado
                forall(k in 1..p)do
                        !Etapa 1: se define el criterio para los j que aún no estan en la
solución:
                        forall(j in pservicio|xsol(j)=0)do
                        g(j):=0
                        forall(i in pdemanda|visitado(i)=0)do
                                if(dist(i,j)<=dc)then g(j):=g(j)+dem(i)
                        end-if
                        end-do
                        marca(j):=0
                        end-do
                        ! Etapa 2: se encuentran los nlist indices con el mejor criterio
                        ! Los puntos de servicio no abiertos que cubren más demanda
                        forall(f in 1..nrcl)do
                                aux:=-9999
                                forall(j in pservicio|xsol(j)=0 and marca(j)=0)do
                                        if (g(j)>aux) then
                                                aux:=g(j)
                                                jaux:=j
                                        end-if
                                end-do
                                ind(f):=jaux
                                marca(jaux):=1
```

```
                    end-do
                    ! elijo al azar un elemento de la lista:
                    jaux:=ind(ceil(random*nrcl))

                    ! Etapa 3: se introduce en la solución
                    xsol(jaux):=1
                    ind_ab(k):=jaux
                    zheur:=zheur+g(jaux)

                    ! Etapa 4: se actualizan los puntos de servicios ya visitados o
abiertos
                    forall(i in pdemanda| dist(i,jaux)<=dc and visitado(i)=0) do
                        visitado(i):=1
                    end-do
                end-do ! final de forall(k in 1..p)


                !Puntos no abiertos:
                na:=0
                forall(j in pservicio)do
                    if(xsol(j)=0)then
                    na:=na+1
                    ind_noab(na):=j
                    end-if
                end-do


                busqueda_local


                !Hay que quedarse con la iteración que proporciona la mejor solución

                if(zheur > zmax)then
                zmax:=zheur
                end-if

        end-do !cierra forall(iter in 1..niter)do


        writeln("\n\n\n(5) Solucion algoritmo GRASP\n\n Demanda total cubierta = ",
zheur)
        writeln("\nPorcentaje de demanda total cubierta = ", (zheur/dem_total)*100, "
%")

        writeln("\n\tTiempo: ",gettime-tiempo_grasp, " s")

end-procedure
```

```
procedure grafico_solucion(ind_ab:array(range)of integer)


        forall(i in pdemanda)do
          daux:=999999999;
          forall(j in abiertos)do
            if(dist(i,ind_ab(j))<daux)then
              daux:=dist(i,ind_ab(j))
              jaux:=ind_ab(j);
            end-if
          end-do
          fi1(i):=jaux
        end-do

        IVEerase

        id1:=IVEaddplot("Puntos_demanda", IVE_BLACK)
        id2:=IVEaddplot("Asignacion", IVE_RED)
        id3:=IVEaddplot("No cubiertos", IVE_BLUE)

        forall(i in pdemanda)do

                if(dist(i,fi1(i))<=dc)then
                        IVEdrawpoint(id1,cx(i),cy(i))
                        IVEdrawline(id2,cx(i),cy(i),cx(fi1(i)),cy(fi1(i)))
                else
                IVEdrawpoint(id3,cx(i),cy(i))
                end-if
        end-do

end-procedure




end-model
```