27 Congreso Nacional de Estadística e Investigación Operativa
Lleida, 8-11 de abril de 2003

PROCEDIMIENTOS PARA LA LOCALIZACIÓN DE ÁREAS DE APORTACIÓN DE RESIDUOS URBANOS

J. Bautista[1], J. Pereira[2]

[1]Departamento de Organización de Empresas
Universitat Politècnica de Catalunya
E-mail: joaquin.bautista@upc.es

[2] Departamento de Organización de Empresas
Universitat Politècnica de Catalunya
E-mail: jorge.pereira@upc.es

RESUMEN

La localización de un conjunto mínimo de áreas de aportación de residuos urbanos puede verse, en algunos casos, como un problema de cubrimiento, Set Covering Problem, con función unicoste. El presente trabajo describe el diseño e implementación de dos metaheurísticas y un procedimiento exacto para resolver el problema. Se analizan los resultados ofrecidos por los procedimientos diseñados mediante una experiencia computacional constituida por ejemplares reales.

Palabras y frases clave: Localización, Set Covering Problem, Algoritmos Genéticos, GRASP.

Clasificación AMS: 68R10, 90C27, 90C59, 90B85

1. Introducción

El diseño de un sistema para la recogida selectiva de residuos urbanos plantea una serie de problemas para el planificador de carácter estratégico, como qué tipo de sistema implementar, ya sea recogida subterránea, mediante el clásico camión-contenedor u otros; táctico, como dónde localizar las áreas de aportación o bien las bocas de recogida pnuemática; y operativo, como el diseño de itinerarios para la recogida. Cada uno de estos problemas pueden tomar diferentes formulaciones dependiendo del objetivo del planificador, ya sea el reparto equitativo de aquellos recursos limitados disponibles, o bien el diseño de un sistema que cumpla unas condiciones de servicio previamente impuestas, siendo estas consideraciones principalmente de carácter estratégico.

Este trabajo se centra en la localización de las áreas de aportación, áreas a los que los usuarios deberán ir a depositar los residuos en contenedores para las diferentes fracciones residuo, en que el objetivo es la localización de un número mínimo de áreas de aportación dada una calidad de servicio, representada como una distancia máxima que el usuario más perjudicado deba recorrer. En el apartado dos del presente trabajo, se muestra que, una vez discretizado, el problema equivale al estudiado problema de cubrimiento de conjuntos, *Set Covering Problem*. Posteriormente, se analiza la literatura del problema y se detalla la implementación de un algoritmo genético y un algoritmo basado en la metaheurística GRASP para su resolución, comparando los resultados obtenidos con la cota ofrecida por el programa lineal relajado del problema en una pequeña experiencia computacional.

2. Discretización del problema de localización de áreas de aportación

Considerese una representación de de la ciudad, o área de la ciudad en que se pretende localizar, formada por un grafo G=(V,A). Este grafo representa las calles mediante un conjunto de vértices, cruces de las calles, y un conjunto de arcos correspondientes a los tramos de calles existentes entre estos vértices. Cada uno de los arcos tiene una distancia asociada igual a la longitud de la calle, una etiqueta identificando la posibilidad, o no, de localizar en este tramo, ya sea porque el vehículo que realiza la recogida no puede circular por este tramo, ya sea porque la calle no reúne las condiciones necesarias para establecer un área de aportación, y una población asociada a este tramo, población residente en el tramo, que se supondrá, sin falta de generalidad, distribuida uniformemente a través de él. Dada una longitud de cobertura *L*, longitud máxima que el usuario más perjudicado del sistema deba recorrer para llegar a una localización, el objetivo consiste en encontrar un conjunto mínimo de puntos en el grafo, ya sea en los vértices o puntos interiores de arcos, accesibles a la posterior recogida, tal que la distancia máxima entre cualquier punto del grafo con población asociada y el punto más lejano de la red sea igual o inferior a la longitud de cobertura. Este problema se conoce en la literatura de localización en redes como el *Problema de Cubrimiento Continuo Absoluto*, aunque este caso presenta adicionalmente restricciones adicionales, arcos que no deben cubrirse o no es factible localizar en ellos.

Para la resolución del problema pueden utilizarse procedimientos basados en propiedades geométricas, procedimientos heurísticos constructivos, o bien recurrir a la discretización del problema para su resolución mediante los más estudiados modelos discretos. En Tamir (1985) se muestra que, en caso que los arcos tengan una longitud entera, el problema es transformable en un problema de cubrimiento de conjuntos discreto mediante la definición de un conjunto finito dominante donde sólo deben considerarse como posibles localizaciones los vértices de la red y aquellos puntos interiores de los arcos situados a una distancia entera respecto a los vértices de su arco, sin pérdida de las propiedades de optimalidad de las soluciones. En caso que los arcos

tengan una distancia no entera, se puede redondear la distancia de cada arco al entero por exceso siendo una solución factible de este problema también una solución para el problema original, aunque no se pueda asegurar que el óptimo del problema discreto también lo sea del original. Esta última aseveración puede extenderse a una discretización del problema en que no se consideren todas las posibles localizaciones, por ejemplo sólo tener en cuenta aquellas localizaciones a distancia par del vértice inicio del arco, mediante el redondeo de la distancia de los arcos, en el ejemplo anterior al primer número par por exceso.

El procedimiento de discretización debe obtener el número de localizaciones que conforma el conjunto finito dominante, *n*, el conjunto discretizado de usuarios, *m*, así como los coeficientes, a_{ij}, que indican el cubrimiento por parte de una un conjunto discretizado de usuarios por parte de un una localización a través del Algoritmo 1, propuesto por Tamir (1985). El procedimiento consiste en la asignación de una localización a cada vértice así como a cada punto interior del arco a distancia entera de los vértices, mientras que el conjunto discretizado de usuarios a cubrir se sitúa en los valores a distancia entera más *0,5* medidas de distancia de los vértices de los arcos. El cálculo de los coeficientes a_{ij}, se apoya en el cálculo del camino mínimo entre la posible localización y el conjunto discretizado de usuarios a través de la red, siendo igual a *1* si esta distancia es inferior a la distancia de cobertura *L*, y *0* en caso contrario.

El algoritmo para la obtención de estos valores se formaliza a continuación:

1. Calcular n y m según expresiones 2 y 3
2. Para todo i, $1 \leq i \leq m$
    a. Para todo j, $1 \leq j \leq n$
        i. Si la distancia entre i y j es inferior a L $a_{ij}=1$, si no $a_{ij}=0$
    b. Fin Para
3. Fin Para

Algoritmo 1. Construcción de la matriz de coeficientes para la discretización del problema de cubrimientos.

En general, la cardinalidad del conjunto localizaciones posibles para un arco, puede determinarse mediante la siguiente expresión:

$$|n_i| = \left\lceil \frac{l_i}{malla} \right\rceil - 1 \qquad (1)$$

dónde n_i representa el conjunto de posibles localizaciones del tramo *i*, l_i es la longitud del tramo *i* y la *malla* representa el nivel de precisión que se desea de la solución obtenida, aunque para el mantenimiento de las propiedades de optimalidad de la solución este valor debe ser igual a 1, propuesta original de Tamir. Aún así, este valor de malla resulta interesante en los procedimientos heurísticos al poder relacionarse con

el tamaño real del objeto a localizar al no tener sentido localizar con una precisión superior al tamaño real del elemento localizado cuando el algoritmo no asegura la optimalidad de las soluciones. Cabe destacar que la resta de la fórmula anterior se debe a que no se consideran los dos vértices dentro del arco, sino como posibles emplazamientos de por sí. En caso de no añadir la eliminación del último valor, el número de localizaciones asociadas al arco contaría también con la localización del vértice final del arco.

Para obtener el número de posibles emplazamientos, del problema bastará sumar el número de emplazamientos de cada arco más el conjunto de vértices del grafo:

$$n = \sum_{j=1}^{|A|} \left(\left\lceil \frac{l_j}{malla} \right\rceil - 1 \right) + |V| \qquad (2)$$

Dónde A representa el conjunto de arcos y V el conjunto de vértices del grafo asociado a la ciudad.

La determinación del número de tramos discretizados a cubrir, depende únicamente de cada uno de los arcos y su longitud tal como muestra la expresión 3:

$$m = \sum_{i=1}^{|A|} \lceil l_i \rceil \qquad (3)$$

Una vez discretizado, el problema a resolver tiene una formulación matemática equivalente al estudiado *Problema de Cubrimiento de Conjuntos* con una función de objetivo *unicoste*, ya que todas las posibles localizaciones tienen el mismo coste asociado. Las expresiones 4 muestran la formulación matemática de dicho modelo.

$$\begin{aligned} &MIN && \sum_{j=1}^{n} x_j && \\ &sujeto\,a: && \sum_{j=1}^{n} a_{ij} \cdot x_j \geq 1 && i = 1,...,m \\ & && x_j \in \{0,1\} && j = 1,...,n \end{aligned} \qquad (4)$$

dónde *n* es la cardinalidad del conjunto de posibles localizaciones, *m* es la cardinalidad del conjunto discreto de usuarios a cubrir, x_j (*j=1,...,n*) es el conjunto de posibles localizaciones y los coeficientes de la matriz *A* (formada por a_{ij} , con *i=1,..,m, j=1,...,n*) indican si la localización x_j cubre al conjunto discreto de usuarios *i*. Este programa matemático, mostrado en las expresiones 4, puede ser resuelto mediante programación lineal binaria. En la experiencia computacional se muestra el resultado obtenido por la relajación ofrecida por el paquete CPLEX.

Para asegurar el mantenimiento de las restricciones adicionales de localización, no localizar en aquellos tramos en que no hay posibilidad de recoger los contenedores de la área de aportación y no cubrir aquellos tramos sin usuarios, y no presentes en el procedimiento original, basta con aplicar un postproceso al algoritmo 1 que elimine, las filas asociadas, al cubrimiento de arcos sin usuarios, y columnas, que elimine aquellas localizaciones dónde no es posible la localización, no siendo tampoco necesarias las variables asociadas.

3. Resolución del problema de cubrimiento de conjuntos

El problema de cubrimiento de conjuntos forma parte de la categoría de problemas NP-Completos, véase Garey y Johnson (1979), para el que se han propuesto diversos procedimientos exactos y heurísticos de resolución.

Entre los procedimientos exactos cabe destacar los propuestos por Fisher y Kedia (1990) basado en una heurística dual y capaz de resolver instancias de hasta 200 filas y 2000 columnas y Beasley (1992), donde se combina una heurística Lagrangiana, cortes de Gomory y una estrategia de ramificación mejorada sobre un trabajo anterior, Beasley (1990), para resolver problemas de hasta 200 filas y 4000 columnas también de forma exacta. Entre los procedimientos heurísticos, las heurísticas propuestas por Beasley (1990a) basada en la relajación Lagrangiana del modelo matemático, el recocido simulado de Jacobs y Brusco (1993) y el algoritmo genético de Beasley y Chu (1996) para el problema con función no-unicoste, conforman el estado del arte del procedimientos de resolución del problema, siendo el algoritmo genético de Beasley y Chu el considerado como mejor algoritmo de resolución del problema en la actualidad.

Entre aquellos trabajos dedicados únicamente al problema con función objetivo unicoste puede verse en Grossman y Wool (1997) donde se comparan los resultados ofrecidos por una red neuronal y las heurísticas aparecidas entre 1974 y 1993 para el problema, tanto unicoste como no unicoste de Johnson (1974), Garey y Johnson (1979), Hochbaum (1982) y Peleg, Schechtman y Wool (1993).

Los algoritmos propuestos en el presente trabajo para la resolución del problema son una implementación de un algoritmo GRASP, acrónimo de *Greedy Randomized Search Procedure*, así como un algoritmo genético, GA, comparando la solución ofrecida por estos procedimientos con la solución relajada de los programas lineales
A continuación se muestra una descripción de las bases fundacionales de ambos procedimientos metaheurísticos.

Un algoritmo genético puede entenderse como un procedimiento de búsqueda probabilística aplicable a problemas de optimización combinatoria. Esta idea, originalmente desarrollada por Holland (1975), se basa en el paradigma biológico de la evolución, entendido como la mejora de la población de acuerdo con la selección natural y la “supervivencia del más fuerte”. En este enfoque, aquellos individuos que

están mejor “adaptados” al entorno tienen mayor probabilidad de sobrevivir y reproducirse, mientras que aquellos individuos peor adaptados tenderán a desaparecer. Esta idea se generaliza en las características de los individuos de la población, genes, bajo el supuesto que los genes más adecuados se esparcirán por un número cada vez mayor de individuos de la población en cada generación, y que la combinación de las características positivas de los ancestros más adaptados producirán descendientes aún mejor “adaptados” mejorando, en su conjunto, la especie ante el entorno.

Un algoritmo genético simula estos procesos mediante una población inicial de soluciones para el problema, la población de individuos, y aplicando, de forma iterativa, operadores genéticos en cada generación. Utilizando los términos propios de optimización combinatoria, cada solución, individuo, de la población codifica la información de la solución en un cromosoma, una ristra, que representa una solución al problema. La adaptación al entorno se mide mediante el valor de la función objetivo de la solución al problema, siendo aquellas soluciones con mejores soluciones, mejor adaptadas, las que más posibilidades tienen de reproducirse y generar nuevas soluciones intercambiando parte de su información mediante un procedimiento de cruce con otras soluciones de la población. Este procedimiento de genera nuevas soluciones hijo que comparten ciertas características de ambos padres. Posteriormente, se aplica un operador de mutación a los hijos para alterar parte de la información de los genes, añadir diversidad a la población y evitar el estancamiento prematuro de ésta durante el proceso. Los descendientes generados sustituyen toda, o una parte, de la población, repitiendo este ciclo de evaluación-selección-reproducción hasta que se encuentra una solución satisfactoria. Para una explicación más exhaustiva de los algoritmos genéticos véase Goldberg (1989). Los detalles de la implementación realizada de dicha metaheurística se muestra en el apartado cuatro.

Por otra parte, los algoritmos GRASP, véase Resende y Ribeiro (1995), son procedimientos constructivos multi-inicio que permiten la generación de múltiples soluciones para un problema. En cada iteración del algoritmo se procede a una fase constructiva en que se construye una solución mediante un procedimiento comilón con una componente aleatoria, y una fase de mejora local en que, partiendo de la solución dada por la fase anterior, se aplica un procedimiento de mejora iterativa hasta que se alcanza una solución óptima local. Este procedimiento ya ha sido utilizado para la resolución de problemas de satisfacción máxima de restricciones (MAX-SAT), véase Resende, Pitsoulis y Pardalos (2000), súper conjunto de problemas del que forma parte el problema de cubrimiento de conjuntos. Los detalles de la implementación realizada se muestra en el apartado cinco.

A lo largo del presente trabajo se realiza un uso indistinto del concepto de fila, restricción y conjunto discretizado de usuarios, y el de columna, variable y localización. Esta relación es unívoca y se debe al modelo matemático del problema, en que cada restricción del modelo corresponde a un conjunto discretizado de usuarios y a su vez se representa por una fila en el programa lineal binario, mientras que cada una de las variables del problema a resolver representa una columna del programa lineal binario

original. A su vez el término ristra y solución son equivalentes, ya que una ristra binaria de longitud igual al número de variables de la instancia representa una solución factible (ó infactible) para ésta. Durante el algoritmo genético se hace también uso indistinto del término valor de la función objetivo y *fitness*, término heredado para tal concepto de la literatura de algoritmos genéticos.

4. Un Algoritmo Genético para el problema de cubrimiento de conjuntos con función unicoste

A continuación se muestra el esquema de un algoritmo genético básico para posteriormente mostrar las modificaciones realizadas con el objetivo de dotar al algoritmo de un cierto conocimiento sobre las características específicas del problema a resolver.

1. Generar una población inicial
2. Evaluar la función objetivo de cada individuo de la población
3. Repetir
    a. Seleccionar parejas de padres de la población
    b. Recombinar los padres para producir hijos
    c. Mutar los hijos generados
    d. Evaluar la función objetivo de los hijos
    e. Reemplazar toda o parte de la población por los hijos
4. Hasta condición de final

Algoritmo 2. Esquema de un algoritmo genético.

Cada una de las partes del esquema mostrado en el algoritmo 2 debe ser adaptado al problema a resolver, esto es qué método será utilizado para representar la población, las técnicas de selección de los padres para la reproducción, los operadores de cruce, la mutación empleada y, finalmente, el modelo para la inserción de los hijos en la población. A continuación se describen las propuestas realizadas para cada uno de estos elementos.

4.1 Representación y función objetivo

El primer paso para definir un algoritmo genético para un problema particular consiste en derivar una representación válida para las soluciones y una función de evaluación indicativa de la calidad de éstas. La representación binaria es la elección más natural para el problema en tratamiento, dado que las variables del problema son binarias. Un valor de *1* en el *i-ésimo* bit indicará que la columna *i* se encuentra en la solución, equivalente a decir que se localizará un área de aportación en el punto asociado al *i-ésimo* bit, en caso contrario, un valor de *0*, indicaría que esta columna no se encuentra en la solución. Con esta representación el valor de la función objetivo puede calcularse como indica la expresión 5.

$$f_i = \sum_{i=1}^{n} x_i + \sum_{j=1}^{m} restricciones\_violadas \quad (5)$$

ya que cada restricción violada puede solventarse mediante la instalación de una localización en alguna de las variables que cubren tal fila en caso que el problema tenga solución.

4.2 Generación de las soluciones iniciales

La generación de la población inicial de soluciones se realiza, normalmente, mediante un procedimiento aleatorio que debe asegurar la suficiente diversidad entre las opciones de la solución para que se explore todo el espacio de posibles soluciones. Se han desarrollado dos de ellos. En el primero se van seleccionando, de forma aleatoria, variables que cubren alguna fila no cubierta, mientras que en el segundo se asigna a cada variable x_i una probabilidad que su valor sea igual a *1* e igual a *0* en caso contrario, tal como se describe en la expresión 6.

$$p_i = \frac{2 \cdot L}{|X|} \quad (6)$$

Esta expresión ha sido obtenida empíricamente tras diversas pruebas y la observación del porcentaje de localizaciones seleccionadas respecto al tamaño del problema, obteniendo poblaciones iniciales que, en promedio, tienen más localizaciones que la solución óptima, permitiendo una distribución inicial de valores adecuada para el algoritmo.
Cabe notar que si bien el primer método genera una solución válida al problema, el segundo método no tiene por qué crear soluciones factibles, aunque el algoritmo se encargará también durante el procedimiento en convertir estas soluciones en válidas mediante los operadores del algoritmo.

4.3 Técnicas de selección de los padres

La selección de los padres es la tarea encargada de asignar probabilidades de reproducción a cada uno de los individuos de la población. Existen diversos métodos de selección usuales en la literatura; entre ellos la selección proporcional, el escalado de la función objetivo y la selección por torneo. A continuación se explica las versiones implementadas de cada uno de ellos, así como la propuesta de Beasley y Chu (1996).

La selección proporcional calcula las probabilidades de que un individuo sea seleccionado de forma proporcional a su valor de función objetivo. La probabilidad que el individuo *i* sea seleccionado es:

$$p_i = \frac{1/f_i}{\sum_1^N 1/f_i} \tag{7}$$

dónde f_i es el valor del *i-ésimo* individuo de la población y N es el tamaño de la población. Como el objetivo del problema es minimizar el valor de la función, la probabilidad es inversamente proporcional al valor de la función objetivo.

El escalado de la función objetivo usa la misma técnica excepto que mapea los valores de la función a una escala fija mediante la asignación de un valor escalado. El objetivo del escalado es mantener la diferenciación de calidad entre los diferentes valores de función objetivo una vez el algoritmo ha convergido. Cuando esto sucede los valores de fitness asociados a las soluciones son muy cercanos, y la posibilidad que todos los elementos tengan probabilidades cercanas para su elección es muy alta, concepto contrario a los principios de la metaheurística. Con el fin de asegurar una selección que tienda a aquellas soluciones de mayor calidad en la población, se calcula un *fitness* escalado mediante:

$$f_i^s = V_{max} - \frac{V_{max} - V_{min}}{f_{max} - f_{min}} \cdot (f_i - f_{min}) \tag{8}$$

Dónde f_i y f_i^s denotan el valor de la función objetivo y el valor de la función objetivo escalado, V_{max} y V_{min} son constantes que indican el rango de diferenciación que se quiere tener para los valores y f_{min}, f_{max} el valor mínimo y máximo de función objetivo entre los elementos de la población. Posteriormente se pasa a la selección mediante un método proporcional según la siguiente fórmula:

$$p_i = \frac{f_i^s}{\sum_1^N f_i^s} \tag{9}$$

Finalmente, la selección por torneo, Beasley y Chu (1996), se basa en la creación de dos grupos de individuos, cada uno de ellos consistente en *T* individuos, obtenidos al azar de la población. El individuo de cada grupo con mejor valor de función objetivo es el escogido para la reproducción, permitiendo determinar la presión selectiva mediante el parámetro *T*. Usando un valor mayor de *T* se incrementa la presión selectiva, ya aquellos elementos con peores valores de función objetivo tenderán a tener menos probabilidades de ser el mejor de su grupo, mientras que valores pequeños de *T* permitirán una mayor diversidad de la población.

4.4 Operador de cruce

En un algoritmo genético tradicional, los operadores de cruce utilizados son el cruce por un punto y el cruce por dos puntos. Estos operadores se basan en la generación de uno o

más puntos de cruce e intercambiando la información de los dos padres según estos puntos de cruce para la generación de dos hijos. Formalmente el operador de cruce por un punto puede definirse como:

0. Sea P_1, P_2 dos soluciones padres formadas por $P_1[1]$, ...,$P_1[n]$ y $P_2[1]$,...,$P_2[n]$, respectivamente.
1. Generar un punto de cruce k, $1 \leq k \leq n$.
2. Las dos soluciones hijo C_1 y C_2 serán:
   a. $C_1:=P_1[1],\ldots,P_1[k],P_2[k+1],...,P_2[n]$
   b. $C_2:=P_2[1],\ldots,P_2[k],P_1[k+1],...,P_1[n]$

Algoritmo 3 Operador de cruce por un punto

El operador de cruce dos puntos es similar al un punto excepto que genera dos puntos de corte en vez de uno. Formalmente el operador de cruce por dos puntos puede describirse mediante el siguiente algoritmo:

0. Sea P_1, P_2 dos soluciones padre formadas por $P_1[1],...,P_1[n]$ y $P_2[1],...,P_2[n]$, respectivamente.
1. Generar dos puntos de cruce k_1, k_2, $1 \leq k_1 \leq n$, $1 \leq k_2 \leq n$ y $k_1 < k_2$.
2. Las dos soluciones hijo C_1 y C_2 serán:
   a. $C_1:=P_1[1],...,P_1[k_1],P_2[k_1+1],...,P_2[k_2],P_1[k_2+1],...,P_1[n]$
   b. $C_2:=P_2[1],...,P_2[k_1],P_1[k_1+1],...,P_1[k_2],P_2[k_2+1],...,P_2[n]$

Algoritmo 4 Operador de cruce por dos puntos

Además de estos operadores básicos utilizables en cualquier algoritmo genético cuya codificación sea binaria, Beasley y Chu (1996) presentan un procedimiento de cruce modificado en su implementación para resolver el problema que denominan cruce generalizado basado en la función objetivo, *generalised fitness-based crossover operator*, que tiene en cuenta tanto la estructura de la solución como el valor relativo de la solución en la función padre. El operador de cruce genera un único hijo, al contrario que los operadores de cruce un punto y dos puntos, tal como se muestra en el algoritmo 5.

0. Sea f_{p1} y f_{p2} el valor de la función objetivo del padre P_1 y el padre P_2 respectivamente.
1. Para todo *i=1,...,n*:
   a. Si $P_1[i]=P_2[i]$, entonces $C[i]:=P_1[i]=P_2[i]$
   b. Si $P1[i] \neq P2[i]$, entonces
      i. $C[i]:=P_1[i]$ con probabilidad $p=f_{p2}/(f_{p1}+f_{p2})$
      ii. $C[i]:=P_2[i]$ con probabilidad 1-p.

Algoritmo 5. Operador de cruce generalizado basado en el objetivo

Otra propuesta es un operador de cruce comilón, greedy. La propuesta del actual trabajo para el problema tratado es la generación de dos hijos diferentes escogiendo alternativamente de cada padre aquellas columnas, localizaciones, que más filas, tramos con población asignada, cubren respecto a la parte de la solución ya construida.

0. Inicialización: Sea PC_1:=P_1 y PC_2:=P_2
1. Mientras aún queden filas no cubiertas
    a. Sea k_1 la columna de PC_1 que cubre más filas no cubiertas de C_1 y k_2 la columna de PC_2 que cubre más filas no cubiertas de C_2
    b. $C_1[k_1]$:=1; $C_2[k_2]$:=1
    c. PC_1:=PC_2 y PC_2:=PC_1
2. Fin

Algoritmo 6. Operador de cruce comilón para el problema de cubrimiento de conjuntos con función unicoste.

4.5 Procedimientos de mutación, operadores de factibilidad y mejora local

La tarea principal del operador de mutación es permitir una cierta cantidad de búsqueda aleatoria por el espacio de soluciones que, a su vez, permita mantener una cierta protección contra la pérdida de información genética debida a la convergencia prematura mediante la reintroducción de ésta. Otro efecto deseado proporcionado por la mutación es la ampliación del espacio de búsqueda mediante una dosis de azar.

Generalmente, la mutación se aplica a cada hijo después del cruce mediante la inversión de alguno de los bits de la solución con una cierta probabilidad, denominada tasa. Esta tasa de mutación puede estar asociada a los hijos o a los bits que conforman una solución y puede ser variable, generalmente aumentando la tasa según el algoritmo converge hasta que llega a un nivel máximo de mutación, o bien fija durante todo el algoritmo.

Se han implementado dos propuestas. La primera, basada en una tasa de mutación constante asociada a los hijos, decide con una probabilidad fija, parámetro del algoritmo, si uno de los bits escogidos al azar debe mutar o no. La segunda propuesta, utilizada por Beasley y Chu (1996), asigna un número de bits a mutar en cada hijo, nueva solución, que se incrementa durante la convergencia del algoritmo. El número de bits a mutar en cada solución se puede determinar mediante:

$$Bits = \left\lceil \frac{m_f}{1+\exp\left(-4m_g\left(t-m_c\right)/m_f\right)} \right\rceil \qquad (10)$$

dónde t es el número de soluciones hijo que se han generado durante el algoritmo, m_f especifica la tasa de mutación estable final, m_c especifica el número de soluciones con

que se pretende llegar a la tasa de mutación media m_f /2 se alcance y m_g especifica el gradiente cuando $t=m_c$.

Adicionalmente a los operadores de mutación, se pueden aplicar operadores adicionales de mejora local de las soluciones. Si bien todas las soluciones generadas por los operadores de cruce y mutación no tienen por qué ser factibles, cada fila no cubierta en la solución puede ser cubierta mediante una columna de la solución, siempre y cuando el problema sea factible, por tanto no es necesario un operador de factibilidad como en el algoritmo de Beasley y Chu. Aún así, esta estimación del valor de la función objetivo es pesimista, siendo probable que se pueda mejorar la solución generada por el procedimiento mediante un operador de factibilidad semejante al propuesto por Beasley y Chu. Otro procedimiento posible es un operador de mejora local tradicional que partiendo de las soluciones generadas lleve a la solución hasta un óptimo local.

El operador de factibilidad heurístico pretende, por una parte, eliminar aquellas columnas de la solución no necesarias, todas las filas que cubre esta columna ya se hayan cubiertas por otras columnas, y añadir aquellas columnas necesarias para que todas las filas queden cubiertas por al menos una columna. Este procedimiento se realiza en dos fases; durante la primera se analizan aquellas columnas que deberían ser insertadas en la solución para que todas las filas estén cubiertas y, en una segunda fase, se pasa a analizar todas aquellas columnas que forman parte de la solución examinando si su eliminación dejara alguna fila no fuera cubierta. Ambos procedimientos se detallan en el algoritmo 7 y 8:

1. Para cada columna que no forma parte de la solución
    a. Si alguna de las filas que cubre la columna no está cubierta, incluir la columna en la solución
2. Fin Para

Algoritmo 7. Procedimiento de inserción

1. Para cada columna que forma parte de la solución
    a. Si todas las filas que cubre la columna están cubiertas por más de una columna, eliminar la columna de la solución
2. Fin Para

Algoritmo 8. Procedimiento de eliminación:

Como el procedimiento depende en gran medida del orden en que las columnas son examinadas por el procedimiento, se ha decidido que este orden no es siempre el mismo, sino que se realiza de dos formas diferentes con un componente de azar. La primera forma, que se realiza en las llamadas impares a este procedimiento, escoge un punto al azar k, $1 \leq k \leq n$, y comprueba las variables desde $k+1$ hasta n y de 1 hasta k. La segunda forma, que se realiza en las llamadas pares a este procedimiento, escoge

también un punto al azar *k* como el anterior, pero comprueba las variables desde *k* hasta *1* y de *n* hasta *k+1*.

El procedimiento de mejora local propuesto realiza un barrido de todas las variables tratando de sustituir dos columnas actualmente en la solución por otra que no se halle actualmente en la solución manteniendo la factibilidad, todas las filas cubiertas, de la solución. El procedimiento se describe formalmente a continuación.

1. Cambio:=1
2. Mientras Cambio=1
    a. Cambio:=0
    b. Para toda columna *i* perteneciente a la solución
        i. Para toda columna *j* diferente de *i* y perteneciente a la solución
            1. Para toda columna *k* no perteneciente a la solución
                a. Eliminar *i*,*j* de la solución y añadir por *k*
                b. Si la solución es factible, mantener esta solución; Cambio:=1, volver a 2.b
                c. Añadir *i*,*j* en la solución y eliminar *k*.
            2. Fin Para
        ii. Fin Para
    c. Fin Para
3. Fin mientras

Algoritmo 3.9. Procedimiento de mejora local:

4.6 Modelos de reemplazo de la población

Una vez se han construido una o más soluciones hijo, el hijo, o hijos, reemplazan uno o varios miembros de la población activa. Dos de los métodos de reemplazo más comunes son el reemplazo generacional y el reemplazo rápido, *steady-state*. En el primer modelo de reemplazo, se genera un número de soluciones igual al tamaño de la población inicial y se sustituye completamente la población por la población de nuevos hijos. En el segundo modelo de reemplazo, se genera un número de soluciones igual a un parámetro y se sustituyen estos hijos probabilísticamente, o bien reemplazando a aquellos elementos de la población original más malos, procedimiento elitista. En este último procedimiento en caso que el número de soluciones generadas sea igual al tamaño de la población, el procedimiento equivaldría al reemplazo generacional.

En caso de realizar una sustitución probabilística en reemplazo rápido, la probabilidad que una solución sea sustituida por uno de los hijos, depende del valor de la función objetivo escalado según la siguiente expresión:

$$f_i^s = V_{min} + \frac{V_{max} - V_{min}}{f_{max} - f_{min}} \cdot (f_{max} - f_i) \qquad (11)$$

Dónde f_i y f_i^s denotan el valor de la función objetivo y el valor de la función objetivo escalado, V_{max} y V_{min} son constantes que indican el rango de diferenciación que se quiere tener para los valores y f_{min}, f_{max} el valor mínimo y máximo de función objetivo entre los elementos de la población. Posteriormente se pasa a la selección del elemento a eliminar mediante la elección de la solución a eliminar con una probabilidad según la siguiente fórmula:

$$p_i = \frac{f_i^s}{\sum_1^N f_i^s} \tag{12}$$

4.7 Esquema general del algoritmo

Resumiendo, el algoritmo genético implementado se describe en el algoritmo 10

1. Generar una población inicial de tamaño *N* mediante uno de los dos procedimientos mostrados en la generación de soluciones iniciales.
2. Seleccionar un conjunto de parejas de padres *NC* de la población mediante una de las tres técnicas de selección de padres
3. Combinar cada pareja de padres para generar nuevas soluciones hijo usando uno de los cuatro procedimientos de cruce.
4. Aplicar uno de los dos operadores de mutación a cada una de las nuevas soluciones hijo generadas
5. Aplicar, o no, los operadores de factibilidad de la solución y de mejora local a cada solución generada
6. Reemplazar cada solución hijo por una solución original mediante el operador de reemplazo
7. Repetir los pasos 2-6 hasta condición de final, ya sea que se ha generado un número de soluciones *M* o bien el algoritmo haya convergido completamente (toda la población tiene el mismo valor de función objetivo).

Algoritmo 10. Implementación final del algoritmo genético

5. Una implementación de la metaheurística GRASP para el problema de cubrimiento de conjuntos con función unicoste

La metaheurística GRASP, *greedy randomized adaptive search procedure*, es un procedimiento para la resolución de problemas de optimización combinatoria. En general, un algoritmo GRASP se compone por dos fases, en una primera fase constructiva, se genera una solución mediante una heurística *greedy* aleatoriezada, mientras que en una segunda fase se realiza una mejora local iterativa partiendo de la

solución generada en la primera fase, hasta que se encuentra una solución localmente óptima. A continuación se describe la implementación realizada de ambas fases.

5.1 Fase constructiva

Un algoritmo constructivo comilón construye una solución elemento por elemento. En cada paso de la construcción, un conjunto *C* de elementos candidatos es identificado y se le asigna un valor de idoneidad que se utiliza para seleccionar uno de ellos para su introducción en la solución, siendo este valor de idoneidad el número de restricciones que la variable cubriría teniendo en cuenta la parte de la solución construida. Una vez se ha realizado esto, se actualiza el conjunto de candidatos y se recalcula el valor de idoneidad para volver a seleccionar otro elemento hasta que la solución construida sea una solución válida para el problema.

El procedimiento constructivo se aleatoriza no escogiendo siempre aquel elemento cuyo valor de idoneidad sea el más alto sino que se escoge entre un subconjunto reducido de elementos de "élite". Las dos propuestas habituales para construir este conjunto reducido de candidatos son la inclusión en el conjunto reducido de aquellos elementos a un porcentaje de desviación del mejor valor de idoneidad o bien, la construcción del conjunto reducido como un número fijo de los mejores valores de la función de idoneidad.

El procedimiento constructivo para el presente problema, corresponde al algoritmo 11.

1. Hasta que todas las filas estén cubiertas
    a. Determinar el conjunto *C* de elementos candidatos
    b. Asignar un valor de idoneidad a cada elemento candidato
    c. Escoger un subconjunto de los elementos candidatos para contruir el conjunto restringido de elementos candidatos *RC*
    d. Escoger al azar un elemento del conjunto *RC* e incluirlo en la solución
    e. Actualizar la lista de filas cubiertas
2. Fin

Algoritmo 11: Descripción del procedimiento constructivo

5.2 Fase de mejora local

Tanto el operador de factibilidad como la mejora local mostradas para el algoritmo genético, algoritmos 7 y 8, así como el algoritmo 9, pueden ser utilizados como fase de mejora local del algoritmo GRASP.

6. Elección de las variantes y parámetros de los algoritmos propuestos

Debido al gran número de combinaciones diferentes de procedimientos disponibles para los algoritmos genéticos, así como de parámetros, se ha realizado una pequeña experiencia computacional para seleccionar aquellos parámetros y combinaciones de procedimientos, en principio, más prometedoras.

En total se han escogido 24 combinaciones para los algoritmos genéticos resultantes de utilizar como procedimiento de generación aleatorio o bien mediante heurísticas, tres tipos de operadores de cruce, cruce por dos puntos, el cruce propuesto por Beasley y Chu (1996) o bien el procedimiento de cruce constructivo, dos tipos de procedimientos de sustitución, elitista o probabilística y dos valores para el número de cruces por iteración, un cruce o cinco. En todos los algoritmos se ha aplicado el procedimiento de mejora local resultante de la aplicación de los algoritmos 7 y 8 y la mutación propuesta por Beasley y Chu (1996), con los valores de m_f=10, m_c=200 y m_g=2, propuestos por los autores. Se ha utilizado como procedimiento de selección para el cruce, el procedimiento de selección con escalado de los valores de la función objetivo con valores V_{min}=1000 y V_{max}=10000. El tamaño de la población se ha fijado en 150 soluciones.

Para el algoritmo GRASP se han probado dos procedimientos para la creación de la lista de candidatos, uno basado en número y otro basado en porcentaje, y para cada uno de ellos dos parámetros de corte, igual a 10 y 25 en el caso de limitar el número de candidatos, y 90% y 95% sobre el mejor valor de idoneidad en el caso de selección por porcentaje.

7. Experiencia computacional

Para validar los algoritmos presentados y sus parámetros, se ha realizado una experiencias computacionales con un juego de pruebas compuesto por diez instancias del problema que se pretende resolver obtenidas del sistema geográfico de información del ayuntamiento de Sant Boi del Llobregat. Estos problemas constan de un conjunto de arcos comprendidos entre 37 y 62 que se han resuelto con valor de malla 1 y dos distancias de cobertura máxima (60 y 100 metros). Las instancias se han discretizado utilizando el algoritmo 1.

Cabe destacar que, a diferencia de las instancias habituales para los problemas de cubrimientos de conjuntos como las presentes en la OR-Library, las instancias de los problemas de localización de áreas de aportación cuentan con un número de restricciones y de variables semejante, mientras que habitualmente el número de variables es superior al de restricciones.

A continuación se muestra el número de filas, columnas, porcentaje de esparsidad del grafo, así como el valor de la cota ofrecida por el programa lineal para cada instancia redondeada al entero por exceso y la mejor solución obtenida por las diferentes configuraciones de algoritmos genéticos, y de los algoritmos GRASP.

Inst.	Restr.	Var.	%esparsidad	Cota (c=60)	GA	GRASP	%esparsidad	Cota (c=100)	GA	GRASP
1	2792	2849	5.64	20	20	21	11.68	10	11	11
2	3018	3054	5.60	21	21	23	11.19	12	12	13
3	3111	3123	5.45	19	20	22	11.02	10	11	11
4	4015	4066	4.33	15	17	18	9.04	13	13	14
5	3033	2995	5.66	20	22	22	11.46	11	12	12
6	3600	3228	4.81	21	21	23	10.21	10	10	12
7	4139	3400	3.75	27	29	29	7.7	10	11	11
8	3663	3549	4.67	26	28	28	9.72	13	13	15
9	3288	3295	5.96	22	23	25	13.45	12	13	14
10	3041	2474	5.41	18	19	20	10.8	9	10	11

Tabla 1: Mejores soluciones y características de las instancias de la colección de problemas

La diferencia máxima entre la solución reportada y la peor solución obtenida por una de las configuraciones de los algoritmos genéticos es de dos unidades, aunque el número de iteraciones y tiempo de ejecución varía, siendo más rápidos aquellos con soluciones iniciales generadas de forma heurística, procedimientos de cruce constructivo y un mayor número de soluciones generadas por iteración del algoritmo los que convergen en soluciones de alta calidad de forma más rápida.

Entre los algoritmos GRASP, las soluciones más favorables provienen de las configuraciones con mayor número de candidatos, aunque no hay diferencias significativas entre ambos procedimientos para definir la lista de candidatas.

Referencias

J.E. Beasley (1990a) “A Lagrangean heuristic for set-covering problems”. *Naval Research Logistics Quaterly*, 37:151-164.

J.E. Beasley (1990b) “OR-Library: distributing test problems by electronic mail”. *Journal of the Operational Research Society*, 41:1069-1072.

J.E. Beasley y K.Jornsten.(1992) “Enhancing an algorithm for set covering problems”. *European Journal of Operational Research*, 58:293-300

J.E. Beasley y P.C.Chu (1996). “A genetic algorithm for the set covering problem”. *European Journal of Operational Research*, 94:392-404

D.E. Goldberg (1989) Genetic Algorithms in Search, Optimization and Machine Learning. *Addison-Wesley Pub. Co.*

T. Grossman y A. Wool (1997). “Computational experience with approximation algorithms for the set covering problem”. *European Journal of Operational Research*, 101(1):81-92.
M.L. Fisher y P.Kedia (1990).“Optimal solutions of set covering/partitioning problems using dual heuristics”. *Management Science*, 36:674-688
M.R.Garey y D.S.Johnson (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. W.*H. Freeman and Co., San Francisco*.
J.H. Holland (1975) Adaptation in Natural and Artifical Systems. MIT press.
D.S.Johnson (1974) “Approximation algorithms for combinatorial problems.” *Journal of Computer System Science*, 9:256-278
L.W.Jacobs y M.J.Brusco (1993) “A simulated annealing-based heuristic for the set-covering problem”. *Working paper, Operations Management and Information Systems Department, Northern Illionis University*, Dekalb, IL 60115, USA.
D.Peleg, G.Schechtman y A.Wool (1993). “Approximating bounded 0-1 integer linear programs”. *En Proceeding of the 2nd Israel Symposium in Theory of Computing Systems*. 69-77, Netanya, Israel.
M.G.C. Resende, L.S. Pitsoulis y P.M. Pardalos (2000) “Fortran subroutines for computing aproxímate solutions of MAX-SAT problems using GRASP”. *Discrete Applied Mathematics*, 100:95-113.
M.G.C.Resende y C.C.Ribeiro (1995). “Greedy Randomized Adaptive Search Procedures” *Journal of Global Optimization* 6:109-133.
Tamir A. (1985) “A finite algorithm for the continuous p-center location problem on a graph”, *Mathematical Programming* 31, pp. 298-306