PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

RESOLUCIÓN DEL SET COVERING PROBLEM UTILIZANDO EL ALGORITMO DE OPTIMIZACIÓN BASADO EN LA BIOGEOGRAFÍA

YARELYN MARCELA ALMARZA RODRÍGUEZ

INFORME FINAL DE PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO DE EJECUCIÓN EN INFORMÁTICA

JULIO 2014

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

RESOLUCIÓN DEL SET COVERING PROBLEM UTILIZANDO EL ALGORITMO DE OPTIMIZACIÓN BASADO EN LA BIOGEOGRAFÍA

YARELYN MARCELA ALMARZA RODRÍGUEZ

Profesor Guía: Ricardo Soto de Giorgis
Profesor Co-referente: Broderick Crawford

JULIO 2014

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Carrera: Ingeniería de Ejecución en Informática.

JULIO 2014

Resumen

Una metaheurística es un método del tipo heurístico para resolver problemas de optimización. La metaheurística Biogeography Based Optimization (BBO) se basa en los comportamientos observados durante el proceso de emigración e inmigración de las especies, entre distintos hábitats, para poder así encontrar una solución óptima. Los principales comportamientos representados en la metaheurística BBO, son Inmigración, Emigración y Mutación. El problema de optimización a resolver con esta metaheurística es el Set Covering Problem (SCP), cuyo propósito es cubrir una cierta cantidad de necesidades incurriendo en el menor costo posible.

Palabras Claves: SCP, BBO, Metaheurística, Algoritmo, Optimización.

Abstract

A metaheuristic is a heuristic method for solving optimization problems. The Biogeography Based Optimization (BBO) metaheuristic is based on the behaviors observed during the process of immigration and emigration of species between habitats in order to find an optimal solution. The main behaviors represented in the BBO metaheuristic are Immigration, Emigration and Mutation. The optimization problem to be solved via this metaheuristic corresponds to the Set Covering Problem (SCP), whose purpose is to cover a certain amount of needs at the lowest possible cost.

Keywords: SCP, BBO, Metaheuristics, Algorithm, Optimization.

Índice

Lista de Figuras

Lista de Tablas

1 Introducción

El diseño de algoritmos cada vez más eficientes para resolver problemas complejos ha sido tradicionalmente uno de los aspectos más importantes de la investigación en informática. El objetivo perseguido en este campo es, fundamentalmente, el desarrollo de nuevos métodos capaces de resolver los mencionados problemas con el menor esfuerzo computacional posible. En consecuencia, esto no solo permite afrontar problemas actuales de manera eficiente, sino también tareas vedadas en el pasado debido a su alto coste computacional.

La principal ventaja de la utilización de algoritmos exactos es que garantizan encontrar el óptimo global de cualquier problema, pero tienen el grave inconveniente de que en problemas reales su tiempo de ejecución crece de forma exponencial respecto al tamaño del problema. En cambio, los algoritmos heurísticos son normalmente bastante rápidos, pero las soluciones encontradas no garantizan el óptimo global. Otro inconveniente de los heurísticos es que no son fáciles de definir en determinados problemas. Las metaheurísticas ofrecen un equilibrio adecuado entre ambos extremos: son métodos genéricos que ofrecen soluciones de buena calidad (el óptimo global en muchos casos) en un tiempo acotado [10].

El Algoritmo de Optimización Basado en la Biogeografía (BBO), pertenece a las técnicas metaheurísticas de población, este permite encontrar una solución óptima en base al movimiento de emigración e inmigración en las distintas especies entre una u otra isla, dependiendo del coeficiente de idoneidad del hábitat.

En este proyecto, se propone utilizar BBO para optimizar el problema del conjunto de cobertura, más conocido como Set Covering Problem (SCP) el cuál es un problema clásico, que consiste en encontrar un conjunto de soluciones que permitan cubrir un conjunto de necesidades al menor costo posible. Existen muchas aplicaciones de este tipo de problemas, siendo las principales la localización de servicios, selección de archivos en un banco de datos, simplificación de expresiones booleanas, balanceo de líneas de producción, entre otros.

2 Definición de Objetivos

2.1 Objetivo General

- Implementar un algoritmo de resolución para el Set Covering Problem utilizando BBO.

2.2 Objetivos Específicos

- Comprender el SCP.
- Implementar un algoritmo BBO para SCP.
- Realizar experimentos con la implementación desarrollada.

3 Estado del Arte

A lo largo del tiempo, se han desarrollado muchas variaciones de las técnicas y métodos ya existentes, para resolver los distintos tipos de problemas relacionados a la optimización mediante la utilización de la programación con restricciones o de algún tipo de metaheurística (tales como las basadas en población o las basadas en trayectoria).

Se ha resuelto el SCP por Nehme Bilal y compañía, utilizando una nueva formulación de este, enfocando principalmente para ser utilizado con una metaheurística [18]. Este trabajo trata sobre eliminar las restricciones presentes en el problema SCP, eliminando así los problemas generados con las soluciones infactibles y problemas de redundancia.

Un trabajo reciente sobre la resolución del Set Covering Problem, es el de Ratnesh Rajan Saxena y compañía [19], el cual consiste en utilizar una técnica de enumeración para poder resolver el Set Covering Problem de característica difusa, linear y fraccional, utilizando una función de linealización para poder obtener una solución óptima al problema.

Jordi Pereira e Igor Averbakh [20], resolvieron el problema del set covering utilizando metaheurísticas genéticas y otros métodos, utilizando costos para el SCP de características inciertas, ya que estos valores son obtenidos desde un intervalo o conjuntos de valores finitos.

B. Crawford y C. Castro [21], han resuelto el problema del set covering utilizando una metaheurística ACO (Ant Colony Optimization) que utiliza procedimientos de lookahead para resolver tanto el Set Covering Problem, como el Set Partitioning Problem, ellos también hicieron un trabajo que consiste en integrar la metaheurística ACO con procedimientos de lookahead y procedimientos de post-procesos para llegar a una solución [22].

Asi mismo se han realizado varios trabajos relacionados con el Set Covering Problem y la metaheuristica ACO como por ejemplo, por R. M. D. A. Silva y G. L. Ramalho [23], él hablaba sobre un sistema de hormigas (ant system); tambien se encuentra el trabajo realizado por Z.-G. Ren, Z.-R. Feng, L.-J. Ke y Z.-J. Zhang [24], el cual consiste en presentar nuevas ideas para aplicar la metaheurística ACO; L. Lessing, I. Dumitrescu, and T. Stutzle [25], realizaron una comparación entre dos algoritmos de la metaheurística ACO para resolver el Set Covering Problem, también se desarrolló un trabajo relacionado con el Set Covering Problem, por los autores M. Rahoual, R. Hadji, y V. Bachelet [26], el cual consistía en la realización de un sistema paralelo de la metaheurística ACO para resolver el Set Covering Problem y finalmente tenemos el trabajo realizado por H. Mulati y A. A Constantino [27], el cual consistía en un algoritmo ACO orientado linealmente, para resolver el Set Covering Problem.

4 Set Covering Problem

El Set Covering Problem (SCP) es un problema clásico, que pertenece a la clase NP-Completo, donde la entrada está dada por varios conjuntos de elementos o datos que tienen algún elemento en común. En general, estos problemas consisten en encontrar un conjunto de soluciones que permitan cubrir en forma total o parcial un conjunto de necesidades al menor costo posible.

En muchos casos, la distancia o tiempo de respuesta entre los clientes y los puntos que prestan los servicios, es decisiva para la satisfacción del cliente. Por ejemplo, si se incendia un edificio, el tiempo de respuesta de una estación de bomberos es vital, ya que si se demora en llegar es posible que el incendio haya consumido por completo el edificio.

Un conjunto de necesidades corresponde a las filas, y el conjunto soluciones a la selección de columnas que cubren en forma óptima al conjunto de filas.

Una definición más formal es la siguiente.

Sea:

A = (a_{ij}) una matriz binaria (0, 1) de dimensiones $m \times$n

C = (c_j) un vector *n* dimensional de enteros

M = {1, . . . ,*m*}

N = {1,. . . , *n*}

El valor $c_j (j \in N)$ representa el costo de la columna j, y podemos asumir que, $c_j > 0$ para $j \in N$. Así, decimos que una columna $j \in N$ cubre una fila $i \in M$ si $a_{ij} = 1$. El SCP busca un subconjunto de columnas de costo mínimo $S \subseteq N$, tal que cada fila $i \in M$está cubierta por al menos una columna $j \in S$.

Así, el problema de Set-Covering se puede plantear de la siguiente manera:

$$mín(z) = \sum_{j=1}^{n} c_j x_j$$

Sujeto a:

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \qquad \forall_i \in M$$

$$x_j = \begin{cases} 1 & si\ j \in S \\ 0 & sino \end{cases} \qquad \forall_j \in N$$

Para entender mejor el planteamiento (ver Figura 4.1). Se tiene una ciudad dividida en 20 sectores y 10 lugares en donde pueden ser instalados cuarteles de bomberos. Cada lugar puede dar servicio a las comunas adyacentes. Se pide determinar dónde ubicar las estaciones de tal forma de minimizar los costos, pero asegurar que se cubran todos los sectores.

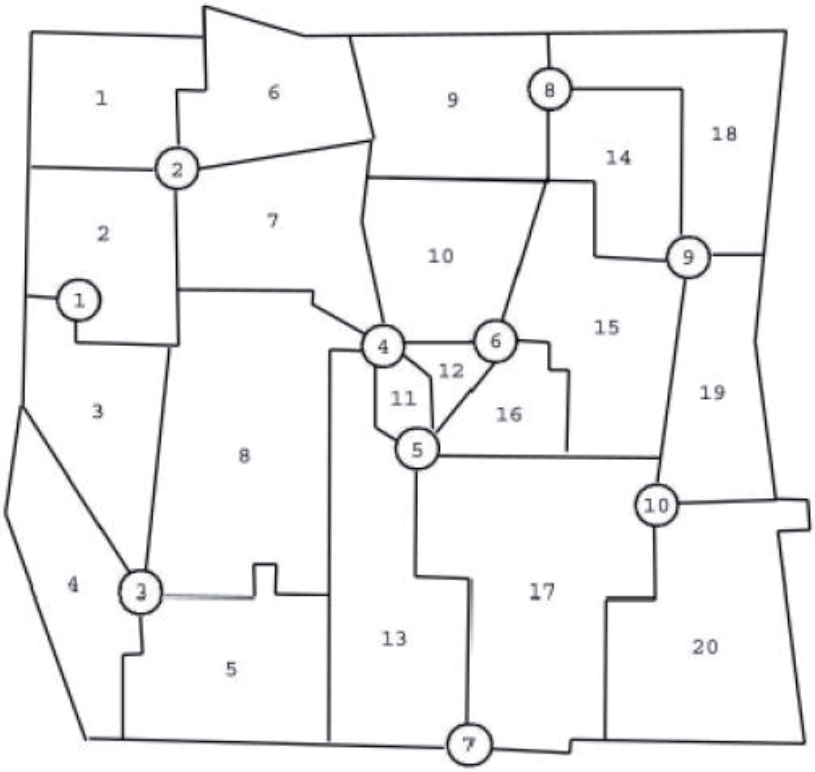

Figura 4. 1 Mapa de Ciudad

De la Figura 3.1 se pueden apreciar los sectores y posibles lugares de instalación de cuarteles de bomberos en base a esto podemos construir la Matriz de cobertura A. Así, un a_{ij} = 1 indica que el cuartel j cubre al sector i. Por ejemplo, el cuartel j = 2 cubre a los sectores i = 1, i = 2, i = 6 e i = 7 mientras que el cuartel j = 10 cubre a los sectores i = 17, i = 19 e i = 20.

Tenemos el vector de costos C = {c1, c2, c3, c4, c5, c6, c7, c8, c9, c10}, M = {1, . . . , 20} y N = {1, . . . , 10}.

Ahora, habría que encontrar el vector binario X = {x1, x2, x3, x4, x5, x6, x7, x8, x9, x10} tal que se cumpla lo siguiente:

$$mín(z) = \sum_{j=1}^{10} c_j x_j$$

Sujeto a:

$$\sum_{j=1}^{10} a_{ij} x_j \geq 1 \qquad \forall_i \in M$$

$$x_j = \begin{cases} 1 & si\ se\ instala\ el\ cuartel\ en\ la\ ubicacion\ j \\ 0 & sino \end{cases} \qquad \forall_j \in N$$

$$A = \begin{pmatrix} 0100000000 \\ 1100000000 \\ 1010000000 \\ 0010000000 \\ 0010000000 \\ 0100000000 \\ 0101000000 \\ 0011000000 \\ 0000000100 \\ 0001010000 \\ 0001100000 \\ 0001110000 \\ 0001101000 \\ 0000000110 \\ 0000010010 \\ 0000110000 \\ 0000101001 \\ 0000001100 \\ 0000000011 \\ 0000000001 \end{pmatrix}$$

La solución de este problema nos indicará donde debemos ubicar los cuarteles para que cubran todos los sectores a un costo mínimo [2].

5 Optimización Local SCP (Pre-selección)

Con el objetivo de obtener mejores resultados, de manera más rápida, se decidió hacer un pre procesamiento de los datos que va a utilizar la metaheurística [17]. El procedimiento de optimización local se basa en la siguiente idea:

Si una fila está cubierta por más de una columna, entonces se considera que esta fila posee redundancia de columnas, por lo cual se debe escoger una "columna superior", para esto se comparan todas las columnas en base al costo de cubrir la fila y se elige la más económica. De esta manera procuramos disminuir la cantidad de columnas que nuestra metaheurística tendrá que recorrer y además nos aseguramos de estar proporcionando las mejores columnas, se trabajó con un 20% de columnas optimizadas.

Pseudocódigo del procedimiento al cual llamaremos "EliminarRedundancia":

```
//S es una fila con columnas redundantes
EliminarRedundancia (var S)
      Begin
            Sup <-  selecionar_superior();
            While (sup != vacio) do
            //seleccionar mejor columna desde Sup
                  best <-   seleccionar_mejor();
                  sup <- sup - best;
            //Agregar superior y remover columnas redundantes de S
                  if(best superior)
                        S <-  S +  best ;
                        S <-  S - seleccionar_redundantes();
                  end if
            end while
            // S es la fila sin columnas redundantes
      return S
end
```

La función `selecionar_superior()` genera una lista *sup* la cual consiste en todas las columnas en orden decreciente según su costo. Luego, la lista *sup* se escanea en el bucle *while*. En cada iteración, se retira la cabeza de *sup* y se memoriza en la variable *best* usando la función `seleccionar_mejor()`. Si la columna seleccionada es todavía superior, es decir, si la condición *best superior* es satisfecha a continuación, se añade a *S*, y el conjunto de columnas redundantes se eliminan de la fila S, usando la función `seleccionar_redundantes()` [17].

6 Biogeography Based Optimization (BBO)

La biogeografía es el estudio de la distribución geográfica de los organismos biológicos. Las ecuaciones matemáticas que rigen la distribución de los organismos fueron descubiertas y desarrollada durante la década de 1960. La mentalidad del ingeniero es que podemos aprender de la naturaleza, esto motiva la aplicación de biogeografía para problemas de optimización.

En BBO, las soluciones de problemas se representan como islas, y el reparto de funciones entre las soluciones se representa como la migración entre las islas. Con los modelos matemáticos de la Biogeografía, se muestra como un proceso natural (de la migración, la especiación y extinción de especies de un isla a otra) puede ser un modelo para resolver problemas de optimización en general.

Los modelos matemáticos de la biogeografía describen cómo las especies migran de una isla a otra, cómo surgen nuevas especies, y cómo las especies se extinguen. El término "isla" aquí se utiliza descriptivamente. Es decir, una isla es cualquier hábitat que está geográficamente aislado de otros hábitats.

En islas o áreas geográficas que están bien adaptadas como residencias para las especies biológicas, se considera que poseen un alto índice de adecuación al habitad (HSI). Las características que se correlacionan con HSI incluyen factores tales como las precipitaciones, la diversidad de la vegetación, la diversidad de las características topográficas, la superficie y la temperatura, estas variables que caracterizan la habitabilidad de una isla son llamadas índices de idoneidad (SIV). Se pueden considerar las variables SIV como variables independientes del hábitat y HSI se puede considerar como una variable dependiente del habitad [6].

BBO se basa en la idea de compartir probabilísticamente características entre las soluciones basadas en las especies de las soluciones.

6.1 Migración

Suponiendo que tenemos un problema y una población de soluciones candidatas que se pueden representar como vectores de números binarios. Cada posición en él vector solución se considera que es un SIV. Además supongamos que tenemos una forma de evaluar la bondad de las soluciones. Esas soluciones que son buenas son consideradas como hábitats con una alta HSI, y los que son pobres son considerados como hábitats con una baja HSI.

Soluciones de alta HSI representan hábitats con muchas especies, y soluciones HSI bajos representan hábitats con poca especies. Suponemos que cada solución (hábitat) tiene una idéntica curva de especies (por simplicidad), pero el valor representado por la solución depende de su HSI. La figura 6.1.1 muestra dos soluciones en BBO. S1 representa una solución pobre, y S2 representa una solución más conveniente. La probabilidad de inmigración para S1 por lo tanto, será más alta que la probabilidad de inmigración para S2. El número de equilibrio de las especies es el punto en que las tasas de inmigración y la emigración son iguales. Sin embargo, puede haber excursiones ocasionales a causa de los efectos temporales. Una excursión positiva podría ser debido a un incremento repentino de la inmigración (causado quizá por inusuales trozos de restos flotantes procedentes de un hábitat vecino), o un repentino estallido de la especiación; por otra parte una excursión negativa podría ser debido a una enfermedad, la introducción de un depredador voraz, o alguna otra catástrofe natural. Se puede tomar un largo tiempo en la naturaleza para el recuento de especies para alcanzar el equilibrio después de una perturbación importante. [12]

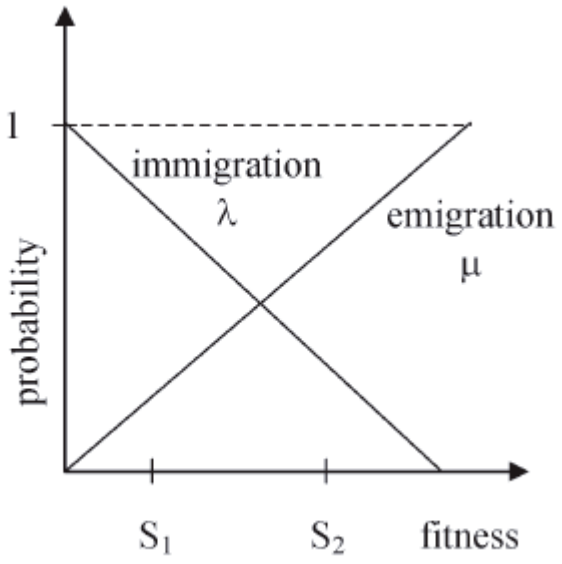

Figura 6. 1.1 Ilustración de dos soluciones

En BBO, si una copia de una especie *s* de la solución *x* sustituye a una de las especies de la solución *y*, nos dicen que *s* ha inmigrado de *x* y emigró a *y*. La probabilidad de que la solución *x* comparta sus especies con algún otro individuo de la población es proporcional a la cantidad de especies de *x*. La probabilidad de que la solución *y* reciba una especie de algún otro individuo de la población disminuye con la cantidad de especies de *y* [4].

Islas con un alto HSI tienen una también una alta tasa de emigración, pero por otra parte tienen una baja tasa de inmigración debido a que ya soportan muchas especies. Las especies emigran a las otras islas para no morir, ya que a pesar de el gran nivel de HSI que pueda tener la isla, hay demasiada competencia con las otras especies por los recursos.

Islas con una HSI bajo tienen una alta tasa de inmigración debido a sus bajas poblaciones. Una vez más, esto no se debe a que las especies quieren emigrar a esas islas, después de todo, se encuentran en islas que son lugares deseables para vivir. La razón por la que la inmigración se produce a estas islas es porque hay un montón de espacio disponible para otras especies. Independientemente de que si las especies que emigran pueden sobrevivir en su nuevo hogar, y por cuánto tiempo, eso es otra cuestión. Sin embargo, la diversidad de especies se correlaciona con el nivel de HSI, por lo que mientras más especies llegan a una isla con bajo HSI, este tenderá a aumentar.

La tasa de inmigración λ y la tasa de emigración μ son funciones del número de especies en la isla. La tasa máxima de la inmigración *I* ocurre cuando hay cero especies en la isla. A medida que el número de especies aumenta, la isla se vuelve más llena de especies y un menor número de especies son capaces de sobrevivir a la inmigración, y la tasa de inmigración disminuye. El mayor número posible de especies que el hábitat puede soportar es S_{max} y es el momento en el que la tasa de inmigración es cero. Si no hay especies en la isla, entonces la tasa de emigración es cero. A medida que el número de especies en las islas aumenta, se hace más apretado, más representantes de las especies son capaces de salir de la isla, y los aumentan tasas de emigración. Cuando la isla contiene el mayor número de especies posibles S_{max} la tasa de emigración alcanza su valor máximo posible E.

En BBO se utiliza las tasas de migración de cada solución para compartir probabilísticamente características entre las soluciones (islas). El operador de la migración en el BBO funciona con la curva que se utiliza para que cada función decida probabilísticamente si emigrar o no. Se selecciona la especie a inmigrar, y luego, la isla de la que emigrar se selecciona probabilísticamente [16].

El siguiente pseudocódigo ejemplifica la estructura del módulo:

```
Let ISIi denote the ith population member and contains n features
 For each island ISIi (where i=1,2,3,…,k)
     For each SIV s (where s=1,2,3,…,n)
       Use λi to probabilistically select the immigrating island ISIi
       If rand < λi
        For j=1 to k
         Use μj to probabilistically decide whether to emigrate to ISIi
           If ISIj is selected
             Randomly select an SIV σ from ISIj
             Replace a random SIV s in ISIi with SIV σ
            end if
          end for
         end if
       next SIV
   next island
```

6.2 Mutación

Eventos catastróficos pueden cambiar drásticamente el HSI natural de un hábitat. También pueden causar un recuento de las especies que perturba su valor de equilibrio, la HSI del hábitat puede, por lo tanto, cambiar repentinamente debido eventos aleatorios.
En BBO, el proceso de mutación se modela como mutación SIV, y a través del número de especies de probabilidades Ps, la tasa de mutación *m* puede ser determinado como:

$$m(S) = m_{max}\left(\frac{1-P_s}{P_{max}}\right) \quad (1)$$

Donde $m_{\max}$ es un parámetro definido por el usuario y corresponde a la tasa de mutación máxima que m puede alcanzar, y $P_{max} = \max(P_s)$.

En la ecuación de mutación, *m* llega a su mínimo "cero" en el valor máximo de P_s, y viceversa. Por lo tanto, *m* es inversamente proporcional a P*s*. Este proceso puede describirse gráficamente como en la figura. 6.2.1 donde la especie S comienza de cero a hasta llegar a S_{max}. A medida que aumenta m_{max}, la oportunidad para que las soluciones muten aumenta también.

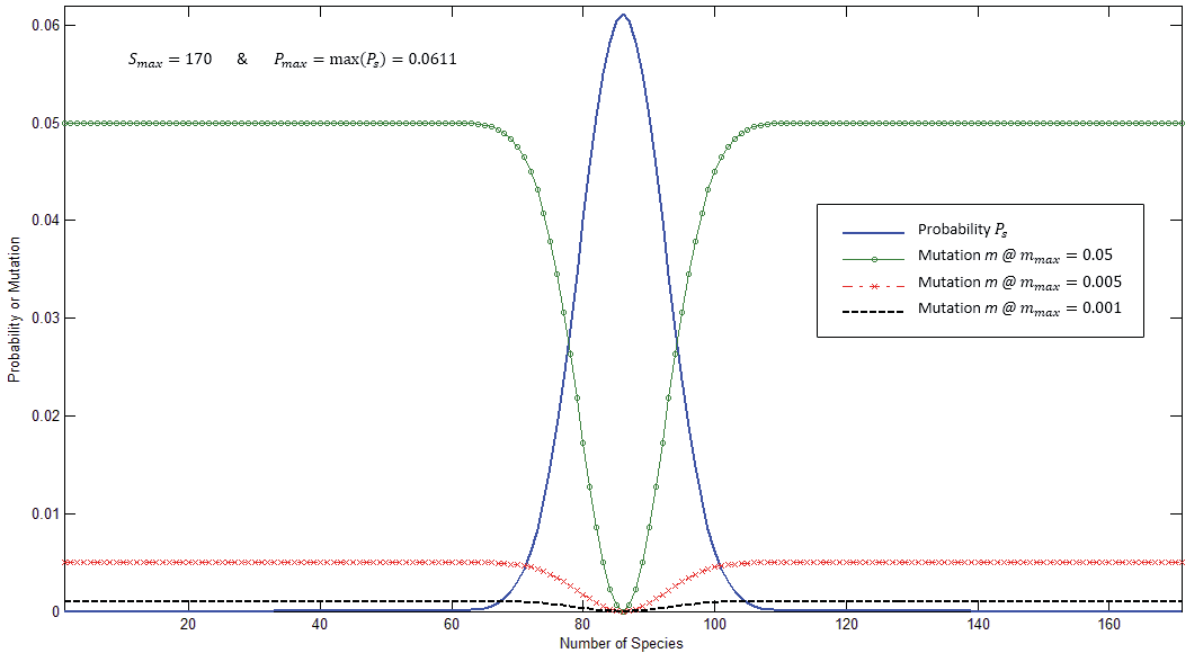

Figura 6. 2.1 modelo de equilibrio de una isla

Durante la etapa de mutación, las soluciones de bajo y alto HSI son propensas a mutar, y luego se podría mejorar más de lo que ya tienen, las soluciones que están en el punto de equilibrio no mutan. Incluso si las soluciones mutadas empeoran, la etapa opcional, llamado elitismo, almacenará las mejores soluciones de una generación a la siguiente [8].

El proceso de mutación se puede describir de la siguiente manera:

```
For i=1 a k (donde k es el número de islas)
      Calcular la probabilidad P_s basándose en λ_s y en μ_s
      Calcular la tasa de mutación m (usando ecuación 1)
      Select HSI_i con probabilidad proporcional a P_s
            If HSI_i es seleccionado
                  Replace SIV de HSI_i con una función que  tome un SIV
            aleatoriamente.
            end if
end for
```

7 Estructura e Implementación de la Solución

Luego de haberse explicado el problema del set Covering y la técnica que se utilizó para llevar acabo su solución, se procede a dar a conocer el planteamiento y la estructura de la solución. Una de las características del algoritmo de optimización basado en la biogeografía es que se puede hacer utilizando variables binarias, lo cual lo hace perfectamente compatible con la formulación del problema del set Covering, el cual también utiliza este tipo de variables.

Las soluciones factibles (vectores binarios) del set Covering, serán representadas como hábitats o islas para efectos de la metaheurística.

Para representar el problema de conjunto de cobertura, utilizando la metaheurística BBO, se utilizó el lenguaje de programación C, y todas las instancias necesarias para resolver el problema fueron representadas mediante vectores y matrices.

Para esto se han definido los siguientes parámetros necesarios:

- **Mapa del Set Covering:** corresponde a una matriz de nxm, donde n corresponde al número de restricciones presentes en el problema y m la cantidad de regiones presentes en este. Estos datos y los valores de las casillas para rellenar el mapa son leídos de un archivo txt.
- **Vector de Costos:** corresponde a un vector de largo o tamaño m, que contendrá el costo que implica cada región presente en el mapa. Los valores de este vector son leídos desde un archivo txt.
- **Archivo de texto del problema:** corresponde a un archivo de texto plano que posee cierto formato para ser leído por el programa, de manera que permita crear las instancias necesarias para representar el Set Covering Problem. El formato de este archivo se muestra en la figura 7.1.

```
Nº_filas  Nº_columnas

//Valores asociados al Vector de
Costos

Valor_col_1 ValorCol _2…….

Valorcol_n

Numero_regiones_con_valor _1

Posiciones de las regiones que
poseen el  valor 1 en la región del
mapa.
```

Figura 7. 1 Formato del archivo txt del SCP.

- **E:** Tasa más alta de emigración.
- **I:** Tasa más alta de inmigración.
- **MAXGEN:** Cantidad de veces que se ejecutara el algoritmo.
- **TAMPOB:** Cantidad de hábitats por generación.
- **Mmax:** Máxima tasa de mutación.
- **Smax:** Máxima cantidad de especies.

7.1 Representación

En nuestro caso se ha utilizado la representación binaria ya que es la representación más natural para el problema de set Covering. Así, tendremos un vector de igual largo que la cantidad de filas a ser cubiertas.

7.2 Proceso previo

Una vez definida la representación y dada la estructura de los archivos txt, se debió programar una rutina que tomara los datos y los cargara en la aplicación. La función `inivar`() es la encargada de tomar los datos desde un archivo tipo txt (el cual debe estar estandarizado bajo los criterios de OR-LIBRARY) y crear las matrices a_i, b_j y el vector de costos c, lo que permite que el algoritmo trabaje con esos datos.

7.3 Tratamiento de los individuos no factibles

Las soluciones modificadas por los operadores de mutación no siempre se van a mantener factibles, por lo que hay que aplicar algún método o función que trabaje con las soluciones no factibles. Para considerar lo anterior, se probara una estrategia la cual consiste en aplicar un método `reparar`() que arregla una solución no factible y la convierte en una factible.

Los pasos requeridos para hacer factible una solución pasan por determinar cuáles filas aún no han sido cubiertas y escoger las columnas necesarias para su cobertura. La búsqueda de dichas columnas se basa en la proporción:

$$\frac{\text{Costo de una columna}}{\text{Cantidad de filas no cubiertas que cubre}}$$

7.4 Inicialización

El tamaño de la población, así como la población inicial son elegidos de manera tal que el dominio de las soluciones asociadas con la población este cubierto adecuadamente. El tamaño de la población depende de cómo se genere la población inicial. En nuestro caso, la población inicial se generará aleatoriamente.

7.5 Condición de término

La condición de término estará dada por un número máximo de generaciones así el algoritmo itera un número determinado de veces, luego se detiene y entrega el mejor valor que tenga hasta ese momento.

7.6 Operadores BBO

Los operadores BBO utilizados en el desarrollo del algoritmo serán la migración y elitismo (opcional).

En el caso de la migración (que viene siendo el modulo más importante de la técnica y de hecho es que la diferencia de otros algoritmos de tipo genéticos), como se indicó anteriormente primero se calculan las variables correspondientes (tasas de inmigración y emigración por poblaciones) para ser utilizadas, la cual basándose en las tasas mencionadas anteriormente, busca probabilísticamente los individuos más propensos a intercambiar información (SIV). Lo destacable de este operador es que considera los valores de HSI y SIV para determinar qué elementos serán inmigrados o emigrados. Se posee un valor máximo definido por un coeficiente. Esto le permite al algoritmo efectuar más exploración a medida que avanza en generaciones.

En resumen la función de mutación toma un individuo, y de acuerdo con la probabilidad de mutación cambia sus componentes: si hay un 0 lo cambia por un 1 y si hay un 1, lo cambia por un 0.

Por otro lado, se implementara un operador de elitismo el cual representara un determinado número de las mejores soluciones que se obtengan en una generación para guardarlas y así luego traspasarlas a la siguiente solución, este módulo de elitismo es opcional por lo tanto no influye en la estructura principal de la técnica.

7.8 Selección

Los operadores de migración necesitan que se les entregue uno o dos hábitats para procesarlos. Esto hace necesario que se defina un criterio de selección de los individuos. En nuestro caso, la selección del SIV a emigrar se efectuara de acuerdo al método de la ruleta, en donde los individuos son seleccionados de acuerdo a una probabilidad determinada a partir de

la función de fitness. Así, un individuo con mayor valor de fitness tendrá más posibilidades de ser seleccionado.

Ya que el problema de Set Covering es de minimización, la probabilidad que un individuo sea seleccionado está determinado por:

$$pi = \frac{\frac{1}{fi}}{\sum_{j=1}^{m} \frac{1}{fj}}$$

Donde *fi* es el valor obtenido por la solución i en la función de evaluación.

7.9 Rutina

El operador de mutación BBO en binario, utiliza la fórmula (1) para decidir la probabilidad de cambio en cada número de bits en la población. [4]
La base del algoritmo esta explicada en el siguiente diagrama, el cual representa una rutina generica de minimizacion con BBO:

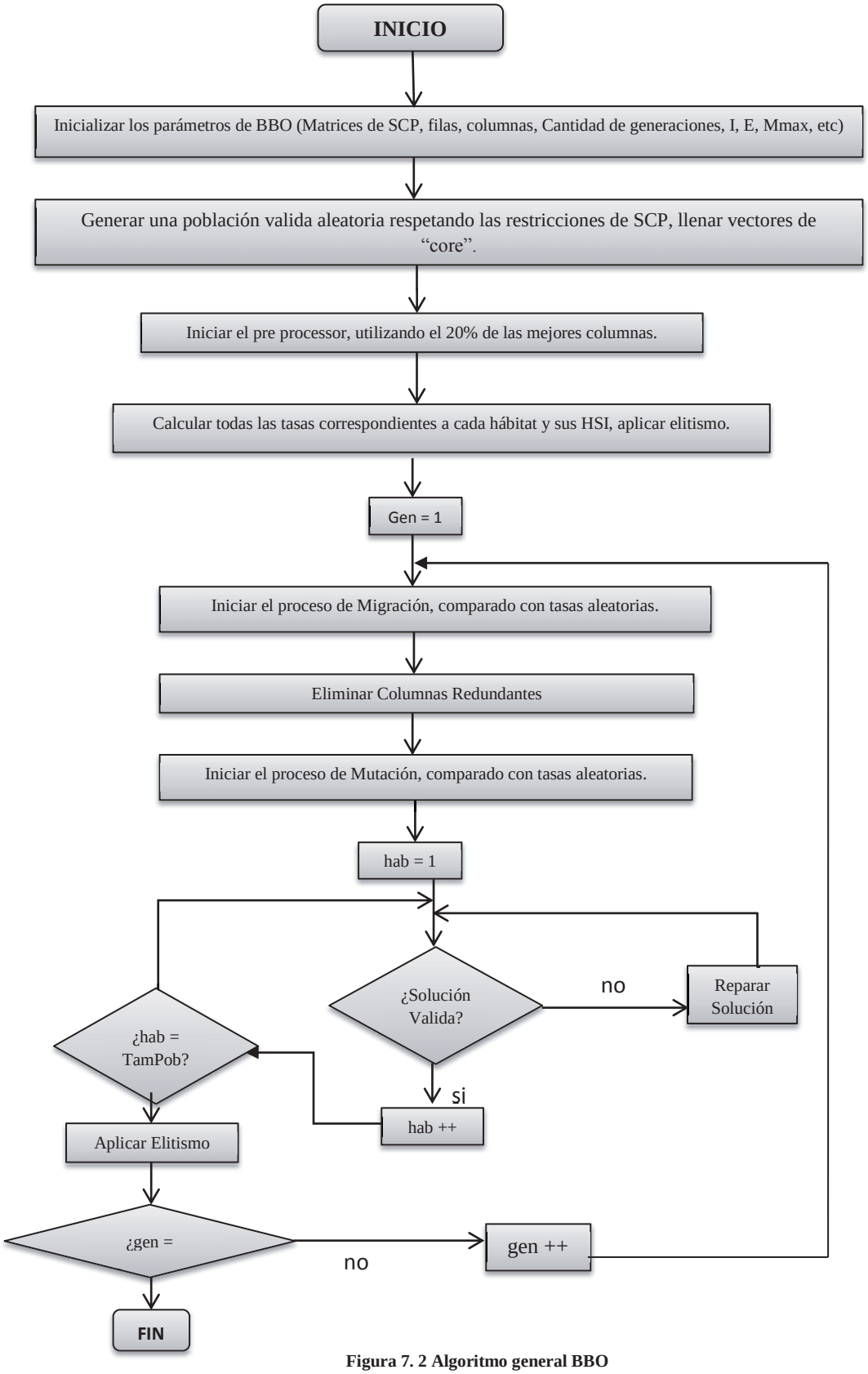

Figura 7. 2 Algoritmo general BBO

8 Experimentos y Resultados

Para realizar los experimentos se utilizó la consola de *DEVC++* y el lenguaje de programación es C. Con el objetivo de poder realizar un seguimiento a los resultados obtenidos por el programa creado, estos fueron almacenados en un archivo txt con un formato apropiado para realizar las conclusiones pertinentes. Se utilizaron los siguientes parámetros de entrada para realizar las pruebas: *TAMPOB* = 15, *Smax* = 100, PMIGRA = 0.75, PMUTA = 0.2, durante los cálculos se utilizó un notebook con procesador Intel Pentium CPU P6200 de 2,13 GHz, con 3 GB de memoria RAM. El sistema operativo en el cual se realizaron todos estos procedimientos corresponde a Windows 7 SP1 Ultimate de 32 bits. Con motivos de evitar resultados basados en la suerte, se corrió el algoritmo un 30 de veces, con una cantidad de iteraciones determinada (18000). En las tablas 1.1 y 1.2 se pueden apreciar los resultados parciales de las pruebas realizadas.

Instancia SCP	Óptimo Conocido	Mejor Óptimo en 10 Ejecuciones.	% Diferencia entre resultado y optimo	Peor Óptimo en 10 Ejecuciones.	Promedio Óptimos en 10 Ejecuciones
4.1	429	470	8,72%	510	490
4.2	512	570	10,18%	600	585
4.3	516	580	11,03%	610	595
4.4	494	580	14,83%	590	585
4.5	512	596	14,09%	600	598
4.6	560	610	8,20%	689	649,5
4.7	430	500	14,00%	510	505
4.8	492	560	12,14%	590	575
4.9	641	720	10,97%	790	755
4.10	514	516	0,39%	588	532
5.1	253	350	27,71%	370	360
5.2	302	410	26,34%	450	430
5.3	226	376	39,89%	390	383
5.4	242	390	37,95%	410	400
5.5	211	328	35,67%	330	329
5.6	213	300	29,00%	310	305
5.7	293	350	16,29%	370	360
5.8	288	310	7,10%	350	330
5.9	279	300	7,00%	320	310
5.10	265	360	26,39%	392	376
6.1	138	170	18,82%	175	172,5
6.2	146	198	26,26%	198	198
6.3	145	170	14,71%	192	181
6.4	131	160	18,13%	185	172,5
6.5	161	190	15,26%	200	195
A.1	253	474	46,62%	480	477
A.2	252	373	8,72%	412	390
A.3	232	381	10,18%	464	457
A.4	234	342	11,03%	456	364
A.5	236	442	14,83%	469	466
B.1	69	181	14,09%	183	185
B.2	76	136	8,20%	136	136
B.3	80	110	14,00%	110	110

Tabla 1. 1 Resultados de pruebas BBO+SCP.

Instancia SCP	Óptimo Conocido	Mejor Óptimo en 10 Ejecuciones.	% Diferencia entre resultado y optimo	Peor Óptimo en 10 Ejecuciones.	Promedio Óptimos en 10 Ejecuciones
B.4	79	135	12,14%	135	135
B.5	72	130	10,97%	131	130,5
C.1	227	329	0,39%	329	329
C.2	219	320	27,71%	340	330
C.3	243	310	26,34%	320	315
C.4	219	305	39,89%	310	307,5
C.5	215	320	32,81%	340	330
D.1	60	120	50,00%	120	120
D.2	66	130	49,23%	130	130
D.3	72	155	53,55%	155	155
D.4	62	122	49,18%	122	122
D.5	61	148	58,78%	148	148
NRE.1	29	79	63,29%	79	79
NRE.2	30	86	65,12%	86	86
NRE.3	27	78	65,38%	78	78
NRE.4	28	80	65,00%	80	80
NRE.5	28	80	65,00%	80	80
NRF.1	14	42	66,67%	42	42
NRF.2	15	49	69,39%	49	49
NRF.3	14	45	68,89%	45	45
NRF.4	14	44	68,18%	44	44
NRF.5	13	46	71,74%	46	46
NRG.1	176	250	29,60%	250	250
NRG.2	154	260	40,77%	260	260
NRG.3	166	198	16,16%	198	198
NRG.4	168	201	16,42%	201	201
NRG.5	168	197	14,72%	197	197
NRH.1	63	110	42,73%	110	110
NRH.2	63	121	47,93%	121	121
NRH.3	59	111	46,85%	111	111
NRH.4	58	109	46,79%	115	112
NRH.5	55	105	47,62%	107	106

Tabla 1.2 Resultados de pruebas BBO+SCP

Las pruebas de las tablas 1.1 y 1.2 se ejecutaron con 20 mil iteraciones y en ninguna de las pruebas se logró alcanzar el óptimo conocido.

9 Conclusiones

El investigar sobre el problema del conjunto de cobertura, ayudó a conocer la complejidad de este, los factores a tomar en cuenta para su resolución y la importancia que este problema posee, ya que se presenta a menudo en la vida diaria, por ejemplo en la selección de archivos en un banco de datos, localización de servicios, balanceo en líneas de producción, entre otros, y de cómo un programa puede facilitar a la resolución de este tipo de problemas.

El uso de metaheurísticas o algoritmos aproximados es una práctica muy útil, basada en el estudio y aprendizaje de formas de optimización naturales, desarrollando y mejorando algoritmos que imiten estos procesos. Se utilizan principalmente cuando se trata de problemas con espacios de soluciones muy grandes, en los cuales utilizar algoritmos de búsqueda de soluciones exactos implica un costo muy alto. Las metaheurísticas si bien no garantizan el óptimo global, generan buenos resultados en tiempos razonables, lo cual disminuye los costos de la búsqueda de soluciones.

Para la resolución del problema en este caso, se estudió, analizó y aplico la metaheurística Biogeographic Based Optimization, la cual se basa en el comportamiento de los especies al moverse de un habitad a otro, para lo cual este algoritmo recrea los principales comportamientos observados que son el emigrar, el inmigrar y el mutar. La idea principal es encontrar un óptimo (ya sea global o local) como solución al problema.

El planteamiento y resultados presentados en el presente informe, permiten dar el primer paso hacia la inferencia según la observación del comportamiento de las distintas variables dentro del algoritmo para la resolución del Set Covering Problem teniendo en cuenta diversos parámetros que deben ser considerados para manejar la metaheurística y sus respectivos operadores.

A lo largo de este informe, se pueden apreciar las pruebas realizadas con la metaheurística BBO al problema de conjunto cobertura SCP, en la cual se ven reflejados los resultados obtenidos para las distintas instancias utilizadas como referencia. Si bien los resultados de estas instancias no alcanzaron el óptimo global con 20 mil iteraciones pero, si se acercaron bastante. Aun así, Intentar aumentar la cantidad de iteraciones, o probar una combinación diferente de parámetros, no implicaría un gran cambio a la hora de analizar los resultados, ya que estos están configurados casi bordeando los límites permitidos y definidos en la teoría.

Para finalizar, con las modificaciones realizadas posteriormente al código, se pudo apreciar un gran cambio en cuanto a los resultados obtenidos, y esto se debe en parte a la función de reparación de soluciones, ya que permite reutilizar estados que poseían características que los convertían en estados infactibles. Por otro lado también se debe a el proceso de eliminación de columnas redundantes.

10 Referencias

[1] Ali R. Alroomi, "**Essentials Modifications on Biogeography-Based Optimixation Algorithm**", Computer Science & Information Technology (CS & IT) pp.1-7, 2013.

[2] Ananias,"**Resolucion del Problema de Set-Covering utilizando un Algoritmo Genetico**",pp. 1-13, (20 de Junio del 2005).

[3] Andrés Felipe Aponte Penagos, "**Propuesta de Solución al Problema de Localización de Centros de Distribución**", Pontificia Universidad Javierana, Procesos Productivos Bogota D.C. pp. 1-10, (Mayo, 2009).

[4] BiBingyan Zhao,Changshou Deng, Yanling Yang, Hu Peng "**Novel Binary Biogeography-Based Optimization Algorithm for the Knapsack Problem**", School of Business - School of Information Science and Tecnology, Jiujiang University, China. pp. 1-5, (2010).

[5] Dan Simon, "**Markov Models for Biogeography-Based Optimization. IEEE Transactions on Systems, Man, And Cybernetics—Part B: Cybernetics**", Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on (Volume:41 , Issue: 1) pp. 1-4, (Febrero,2011).

[6] Dawei Du, "**Biogeography-Based Optimization Combined with Evolutionary Strategy and Immigration Refusal**". Cleveland, Ohio, USA: Cleveland State University,pp. 1-13, (2009).

[7] Haiping Ma."**Biogeografía optimización basada en Noisy funciones de fitness.**", Laboratorio de Investigacion en en Sistemas Inteligentes (LISI), La Pampa. pp. 1-11, (2012).

[8] Haiping Ma, "**Biogeography-Based Optimization with Ensemble of Migration Models for Global Numerical Optimization**", N. Brisbane, Australi. Evoluionary Computation CEC IEEE Congress on ,pp. 1-9, (June, 2012)

[9] Jeffrey Abell, "**A Framework for Multi-objective, Biogeography Based Optimization of Complex System Families**" . Cleveland, Ohio: Cleveland State University, 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference 13-15 Septiembre 2010 , Texas. pp. 1-12

[10] José Francisco Chicano García "**Metaheurísticas e ingenieria del software**". Universidad de Malaga,pp. 5-12, (Julio de 2007)

[11] Nikumbh, S. "**`bbo' Package**", pp. 1-14, (April 19, 2013)

[12] Simon, D. "**Biogeography-Based Optimization**". IEEE Transactions On Evolutionary Computation, Vol. 12, No. 6. pp 1-15, (2008).

[13] Simon, D. "**Los algoritmos de optimización evolutiva**". Computación Evolutiva, principio y aplicaciones,pp 1-13,(s.f.).

[14] Sons, J. W." **Los algoritmos de optimización evolutiva** ". Inspiración biológica y métodos poblacionales a la Inteligencia Computer, pp.1-15, (2013)

[15] Ali R. Alroomi, Fadhel A. Albasri y Jawad H. Talaq." **Solving The Associated Weakness Of Biogeography-Based Optimization Algorithm**". Academy & Industry Research Collaboration Center (AIRCC) , International Journal of Soft Computing (Noviembre 2013), pp.1-6.

[16] Rick Rarick, Dan Simon, F. Eugenio Villaseca, Bharat Vyakaranam."**Biogeography-Based Optimization and the Solution of the Power Flow Problem.**". Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on. pp 1-4, (2009)

[17] Elena Marchiori, Adri Steenbeek. "**An Evolutionary Algorithm for Large Scale Set Covering Problems whit Application to Airline Crew Scheduling**". Real world applications of evolutionary computing. Lecture Notes in Computer Science Volume 1803. pp 1-7,(Febrero 2003).

[18] Nehme Bilal, Philippe Galinier, and Francois Guibault. "**A New Formulation of the Set Covering Problem for Metaheuristic Approaches,**" ISRN Operations Research, vol. 2013, Article ID 203032, 10 pages, 2013. doi:10.1155/2013/203032.

[19] Ratnesh Rajan Saxena, Rashmi Gupta. **Enumeration Technique for Solving Linear Fractional Fuzzy Set Covering Problem**. International Journal of Pure and Applied Mathematics**,** vol. 84, issue 5, 2013.

[20] Jordi Pereira, Igor Averbakh. **The Robust Set Covering Problem with interval data**. Annals of Operations Research, vol. 207, issue 1, pp 217-235. 2013.

[21] B. Crawford and C. Castro, "**Aco with lookahead procedures for solving set partitioning and covering problems**," in Proceedings of Workshop on Combination of Metaheuristic and Local Search with Constraint Programming Techniques, Nantes, France, November 2005.

[22] B. Crawford and C. Castro, "**Integrating lookahead and post processing procedures with aco for solving set partitioning and covering problem**," in Proceedings of the 8th International Conference on Artificial Intelligence and Soft Computing (ICAISC '06), L. Rutkowski, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, Eds., vol. 4029 of Lecture Notes in Computer Science, pp. 1082–1090, Springer, 2006.

[23] R. M. D. A. Silva and G. L. Ramalho, "**Ant system for the set covering problem**," in Proceedings of IEEE International Conference on Systems, Man and Cybernetics, vol. 5, pp. 3129–3133, October 2001.

[24] Z.-G. Ren, Z.-R. Feng, L.-J. Ke, and Z.-J. Zhang, "**New ideas for applying ant colony optimization to the set covering problem**," Computers and Industrial Engineering, vol. 58, no. 4, pp. 774–784, 2010.

[25] L. Lessing, I. Dumitrescu, and T. Stutzle, "**A comparison between aco algorithms for the set covering problem**," in Proceedings of the 4th International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS '04), pp. 1–12, 2004.

[26] M. Rahoual, R. Hadji, and V. Bachelet, "**Parallel ant system for the set covering problem**" in Ant Algorithms, Lecture Notes in Computer Science Volume 2463, pp 262-267.2002.

[27] M. H. Mulati and A. A. Constantino, "**Ant-line: a lineoriented aco algorithm for the set covering problem**," in Proceedings of the 30th International Conference of the Chilean Computer Science Society (SCCC '11), pp. 265–274, Curicó, Chile, November 2011