
**BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA**

**FACULTAD DE CIENCIAS DE LA COMPUTACIÓN**

**TESIS:**
# REDUCCIÓN DEL ANCHO DE BANDA DE MATRICES DISPERSAS UTILIZANDO METAHEURÍSTICAS

**QUE PARA OBTENER EL GRADO DE**

**MAESTRO EN CIENCIAS DE LA COMPUTACIÓN**

**PRESENTA:**

**ING. ALBERTO ROMÁN FLORES**

**DIRECTOR DE TESIS:**

**DRA. LOURDES SANDOVAL SOLÍS**

**PUEBLA, PUEBLA**

**2012**

# AGRADECIMIENTOS

Un profundo agradecimiento a la Dra. Lourdes Sandoval Solís, asesora de esta tesis por todo el apoyo y la paciencia que me tuvo durante la realización de este trabajo.

También quiero agradecer a mi familia en especial a mi mamá y a mi hermano por apoyarme en todo momento.

Un sincero agradecimiento a todos mis profesores y amigos ya que sin su ayuda y apoyo no hubiera sido posible realizar este trabajo de tesis.

Finalmente un especial agradecimiento al CONACYT ya que gracias al apoyo que me brindó me fue posible estudiar la Maestría en Ciencias de la Computación

# DEDICATORIA

Esta tesis se la dedico con mucho cariño a mi familia en especial a mi mamá quien siempre me ha enseñado a seguir adelante a no dejarme caer, además de que me ha inculcado buenos principios y valores.

También dedico esta tesis a todos mis profesores, en especial a la Dra. Lourdes Sandoval Solís por confiar y creer en mí, además por todo su apoyo durante la realización de esta tesis.

# INDÍCE

# INTRODUCCIÓN

El problema de minimizar el ancho de banda de matrices dispersas (Bandwidth Minimization Problem (BMP)) se originó en 1950 cuando los ingenieros estudiaron los marcos de acero haciendo uso de las computadoras [4] [5] [6]. El BMP se encuentra en un gran número de aplicaciones entre las que podemos mencionar:

- Solución de Sistemas de Ecuaciones Lineales Grandes.
- Ecuaciones Diferenciales.
- Diseño de Circuitos.
- Hipertextos.
- Sistemas de Transmisión de Alta Potencia, etc.

En 1976 Papadimitriu probó que el BMP se considera como un problema NP completo. El BMP consiste en encontrar una permutación de filas y columnas de tal forma que los elementos diferentes de cero de la matriz estén lo más cerca de la diagonal, por esto se considera un tema muy importante que se estudia día con día con el objetivo de implementar algoritmos que resuelvan este problema.

En este trabajo de tesis nosotros trabajaremos con matrices dispersas, este término se le atribuye al economista Harry Markowitz, quien al analizar modelos de actividad industrial en la Rand Corporation durante 1950, observó que al trabajar con matrices la mayoría de sus elementos eran ceros.

Por su trabajo Harry Markowitz fue premiado en 1989 con el Premio Von Neumann por la Sociedad de Investigación de Operaciones de América y en 1990 recibió el Premio Nobel en Ciencias Económicas.

William Orchand-Hayes fue el primero en implementar un código para matrices dispersas. La investigación y los trabajos realizados sobre este tipo de matrices son ahora de mucha importancia, ya que son estándares que se utilizan en códigos de programación lineal.

Por otra parte Ralph Arthur Willoughby fue otro de los pioneros de la utilización de las matrices dispersas, cuando intentó resolver sistemas de ecuaciones a gran escala, al trabajar en la simulación de circuitos. Junto con su compañero Werner Liniger, observaron que las ecuaciones con las que trabajaban eran muy grandes y dispersas. Los primeros códigos desarrollados fueron para resolver problemas específicos de este tipo, tiempo después Fred Gustavson un compañero de trabajo de ellos en IBM, creó un programa de Factorización Triangular Simbólica, el cual genera código en Fortran para cualquier problema.

Ralph Willoughby y su equipo de trabajo en IBM, desarrollaron técnicas para resolver la simulación de circuitos electrónicos. Las técnicas desarrolladas también se aplicaron en otros problemas como: programación lineal, análisis de sistemas de potencia, análisis estructural, entre otros. Él organizó el primer “Symposium on Sparse Matrices and Their Applications”, el encuentro fue financiado por la IBM T.J Watson Research Center in Yorktown Heights, New York, September 9-10, 1968. Las memorias de este Symposium aparecieron sólo como un reporte de la IBM

“Sparse Matrix Proceedings” (RAI 3-12-69), que fue solicitado por mucha gente en todo el mundo.

En 1971 en el tercer encuentro de este Symposium, se presentaron principalmente trabajos de métodos directos para resolver sistemas de ecuaciones lineales dispersos grandes, los cuales fueron publicados en 1972 por la IBM.

A través de los años se han desarrollado varios algoritmos para resolver el BMP, que también nos sirve para resolver sistemas de ecuaciones lineales grandes, la mayoría de ellos están basados en teoría de grafos, entre los más importantes podemos mencionar: el algoritmo de Cuthill-Mckee propuesto en 1969 [10], también tenemos el algoritmo de Grado Mínimo que fue propuesto por Tinney y Walker en 1967 que se utiliza para reducir el fill-in cuando se utiliza el método de eliminación Gaussiana, otro algoritmo es el inverso de Cuthill-Mckee desarrollado por George en 1971 que es una mejora de Cuthill-Mckee.

En 1976 Gibbs, Poole y Stockmeyer propusieron el algoritmo GPS el cual se basa también en teoría de grafos. En 1999 G. M. del Corso y Manzini proponen los algoritmos Minimum Bandwidth by iterative Deeping y el Minimum Bandwidth by Perimeter Search.

También se puede resolver el BMP, mediante metaheurísticas, las cuales podemos decir que son algoritmos que están inspirados en la naturaleza, que son comúnmente utilizadas en problemas de optimización global, y fueron desarrolladas después de 1960.

Además las metaheurísticas tienen algunas ventajas en cuanto a los algoritmos convencionales, dos de las principales características son intensificación y diversificación, la intensificación intenta buscar localmente, mientras que la diversificación asegura que el algoritmo explore el espacio de búsqueda global.

La eficiencia de las metaheurísticas radica en que imitan las mejores características de la naturaleza, especialmente la selección, dentro de estos algoritmos podemos mencionar:

- Recocido Simulado (Simulated Annealing (SA)) [12].
- Algoritmos Genéticos (AG) [12].
- Búsqueda Tabú (Tabú Search) [12].
- Colonia de Hormigas (Ant Colony Optimization (ACO)) [16].
- Cúmulo de Partículas (Particle Swarm Optimization (PSO)) [16].
- Búsqueda Harmónica (Harmony Search (HS)) [16].
- Colonia de Abejas (Bee Algorithms) [16].
- Luciérnagas (Firefly Algorithm) [16].

Haciendo una búsqueda en la página de Bibliotecas digitales de la BUAP sobre artículos de Bandwidth Sparse Matrix encontramos un total de 119 resultados, refinando nuestra búsqueda por el campo de textos completos obtuvimos 92 resultados y refinando más nuestra búsqueda agregando la palabra metaheurística obtuvimos sólo 3 artículos.

En cuanto a artículos publicados acerca del BMP podemos mencionar: En 1992 J. C. Lou publica el artículo denominado **Algorithms for Reducing the Bandwidth**

**and profile of a Sparse Matrix**. En 1995 I.L. Lim, I.W. Johnston, S.K. Choi publican su artículo **A Comparison of Algorithms for profile reduction of Sparse matrices.** Para 1999 G. M. Del Corso, G. Manzini publican el artículo **"Finding Exact Solutions to the Bandwidth Minimization Problem** [13]. En 2002 Andrew Lim propone un algoritmo híbrido usando algoritmos genéticos y Hill Climbing [4]. En 2004 Eduardo Rodríguez Tello y su equipo proponen una variante de la función delta y lo combinan con el algoritmo de Recocido Simulado [11]. También en este año Estefanía Piñana, Isaac Plana, Vicente Campos y Rafael Martí utilizan el algoritmo de GRASP para resolver el ancho de banda de matrices. Ya en 2005 Andrew Lim propone otro algoritmo híbrido, esta vez utiliza el algoritmo de ACO y el Hill Climbing, en 2006 publica el artículo **Heuristics for Matrix Bandwidth reduction**, y para 2007 propone un algoritmo que utiliza PSO y Hill Climbing [5] [6]. En ese mismo año A. Kaveh y P. Sharafi proponen un algoritmo para reducir el ancho de banda basado en ACO [3]. En el 2010 Behrooz Koohestani, Riccardo Poli proponen un algoritmo para resolver el BMP utilizando algoritmos genéticos [7], por su parte Camelia Pintea, Gloria Cerasela Crisan, Camelia Chira proponen su algoritmo basado en ACO en ese mismo año [8]. Para 2011 Vicente Campos, Estefanía Piñana, Rafael Martí proponen utilizar memorias adaptativas para resolver el BMP y lo mezclan con los algoritmos de Tabú Search y Scatter Search [15].

El objetivo de este trabajo de tesis es reducir el ancho de banda de matrices dispersas utilizando metaheurísticas. Los objetivos particulares son:

1. Implementar el Algoritmo de Cuthill-Mckee.

2. Implementar el Algoritmo de Recocido Simulado.
3. Implementar el Algoritmo de Algoritmos Genéticos.
4. Implementar el Algoritmo de Colonia de Hormigas.
5. Comparar los Algoritmos Implementados.

Así este trabajo de tesis está compuesto por los siguientes capítulos:

1. Conceptos Básicos.
2. Algoritmo de Cuthill-Mckee.
3. Metaheurísticas.
    - 3.1 Recocido Simulado.
    - 3.2 Algoritmos Genéticos.
    - 3.3 Colonia de Hormigas.
4. Implementación y Pruebas.
5. Conclusiones
6. Bibliografía.

En el capítulo 1 se dan los conceptos que manejaremos en este trabajo, en el capítulo 2 describiremos el algoritmo de Cuthill-Mckee, en el capítulo 3 trataremos la metaheurística de recocido simulado, en el capítulo 4 describiremos el algoritmo de Algoritmos Genéticos, ya en el capítulo 5 trabajamos con el algoritmo de Colonia de Hormigas, después presentamos las pruebas al aplicar los algoritmos a las matrices dispersas que generamos y presentamos una tabla en donde se comparan dichos algoritmos, por último presentamos las conclusiones del trabajo y la bibliografía utilizada durante el desarrollo del trabajo.

# CAPÍTULO 1

# CONCEPTOS BÁSICOS

En este capítulo se presentan los conceptos que utilizamos durante el desarrollo de este trabajo, como el concepto de matriz dispersa, después damos la definición del Bandwidth Minimization Problem (BMP), luego se define el concepto de metaheurística.

**Definición 1.1 (Matriz Dispersa) [2] [9]**. Sea $A$ una matriz de orden $n$. Se dice que $A$ es dispersa si la mayoría de sus elementos son ceros.

Enseguida se muestran algunos ejemplos de matrices dispersas;

$$A = \begin{bmatrix} 1 & 0 & 0 & 2 & 0 \\ 0 & 3 & 0 & 0 & 4 \\ 0 & 0 & 5 & 0 & 0 \\ 6 & 0 & 0 & 7 & 8 \\ 0 & 9 & 0 & 10 & 11 \end{bmatrix}$$

*Matriz dispersa de orden* 5

Otra matriz dispersa podría ser:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

*Matriz Dispersa de orden* 10

**Definición 1.2 (Ancho de Banda) [4] [5] [6].** Sea $A$ una matriz de orden $n$. El ancho de Banda de $A$ se define como: la sumatoria del valor absoluto de la diferencia entre $i$ e $j$ tal que el elemento $a_{i,j}$ sea diferente de cero, así el Bandwidth Minimization Problem (BMP) se expresa matemáticamente como:

$$Min \left\{ \sum_{i,j}^{n} |i-j|, \;\; a_{i,j} \neq 0 \right\}$$

A continuación se muestra un pequeño ejemplo:

$$A = \begin{bmatrix} 1 & 0 & 0 & 2 & 0 \\ 0 & 3 & 4 & 0 & 5 \\ 0 & 6 & 7 & 0 & 8 \\ 9 & 0 & 0 & 10 & 11 \\ 0 & 12 & 13 & 14 & 15 \end{bmatrix}$$

*Matriz dispersa de orden* 5

El ancho de Banda de esta matriz está definido como:

$$Ancho\ de\ Banda$$

$$= |1-1| + |1-4| + |2-2| + |2-3| + |2-5| + |3-2| + |3-3|$$
$$+ |3-5| + |4-1| + |4-4| + |4-5| + |5-2| + |5-3| + |5-4|$$
$$+ |5-5|$$

Haciendo las operaciones correspondientes obtenemos:

$$Ancho\ de\ Banda = 20$$

Aquí tenemos otro ejemplo:

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

*Matriz Dispersa de orden* 10

Su ancho de bandas sería:

$$Ancho\ de\ Banda$$

$$= |1-1| + |1-4| + |1-6| + |2-2| + |2-3| + |2-7| + |2-10|$$
$$+ |3-2| + |3-3| + |3-4| + |3-5| + |4-1| + |4-3| + |4-4|$$
$$+ |4-6| + |4-9| + |5-3| + |5-5| + |5-7| + |6-1| + |6-4|$$
$$+ |6-6| + |6-7| + |6-8| + |7-2| + |7-5| + |7-6| + |7-7|$$
$$+ |7-8| + |8-6| + |8-7| + |8-8| + |9-4| + |9-9| + |10-2|$$
$$+ |10-10|$$

Efectuando las operaciones correspondientes obtenemos:

$$Ancho\ de\ Banda\ =\ 76$$

**Definición 1.3 (Metaheurística) [12].** Son algoritmos que dan soluciones aproximadas de propósito general, consistentes en procedimientos iterativos que guían una heurística, combinando de forma inteligente distintos conceptos para explorar y explotar adecuadamente el espacio de búsqueda.

### 1.4 Generación de Matrices Dispersas

Como nuestro objetivo es reducir el ancho de banda de matrices dispersas, enseguida se describe la manera en la que generamos dichas matrices. Primero se le pide al usuario que de la dimensión de la matriz, así como un valor de dispersión entre 0 y 1 que nos permitirá hacer un cierto porcentaje de ceros en nuestra matriz, luego utilizamos la función rand de MATLAB que nos permite crear una matriz aleatoria de tamaño $n\,x\,n$, una vez creada la hacemos simétrica y estrictamente diagonalmente dominante, después checamos si cada elemento de la matriz es menor o igual que el valor de dispersión entonces el elemento se

vuelve cero, si es mayor entonces el elemento se vuelve uno para conseguir el porcentaje de dispersión propuesto. A continuación se muestra el pseudocódigo de la función que implementamos para generar nuestras matrices dispersas:

$Función\ [A]\ =\ genera\_matriz\ (n, dispersión)$

$Crear\ una\ matriz\ Identidad\ E\ de\ tamaño\ n$

$Colocar\ en\ la\ diagonal\ de\ nuestra\ matriz\ E\ los\ valores\ de\ 1\ a\ n.$

$Generar\ una\ matriz\ aleatoria\ A\ de\ tamaño\ n\ con\ la\ función\ rand.$

$Almacenar\ en\ A\ la\ suma\ de\ las\ matrices\ A\ y\ E.$

$En\ una\ matriz\ S\ hacer\ la\ suma\ de\ la\ matriz\ A\ con\ su\ transpuesta\ y\ dividir$

$el\ resultado\ entre\ 2, esto\ para\ hacersimétrica\ nuestra\ matriz.$

$A\ =\ S$

$//\ Hacer\ que\ nuestra\ matriz\ sea\ estrictamente\ diagonalmente\ dominante$

$Para\ \ i = 1\ hasta\ n$

$A\ (i, i)\ =\ A\ (i, i)\ *\ 50\ *\ n$

$End\_Para$

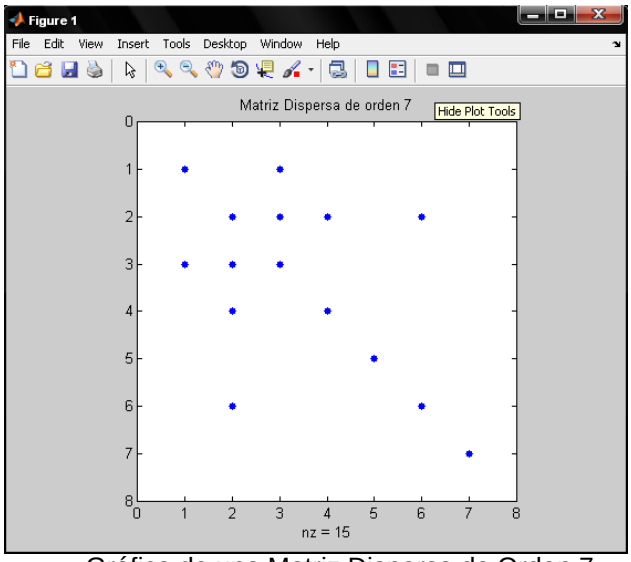
$A\ (find\ (A <=\ dispersión))\ =\ 0$

Enseguida se muestra un ejemplo de una matriz dispersa de $tamaño\ 7\ x\ 7$ con un valor de dispersión de 0.75, la cual fue generada con la función que implementamos:

```
Da la dimension de la matriz:  7
Da el porcentaje de dispersion de la matriz:   0.75

A =

  1.0e+003 *

    0.6350         0    0.0008         0         0         0         0
         0    0.8231    0.0008    0.0008         0    0.0008         0
    0.0008    0.0008    1.1832         0         0         0         0
         0    0.0008         0    1.5991         0         0         0
         0         0         0         0    1.8080         0         0
         0    0.0008         0         0         0    2.1801         0
         0         0         0         0         0         0    2.4873
```

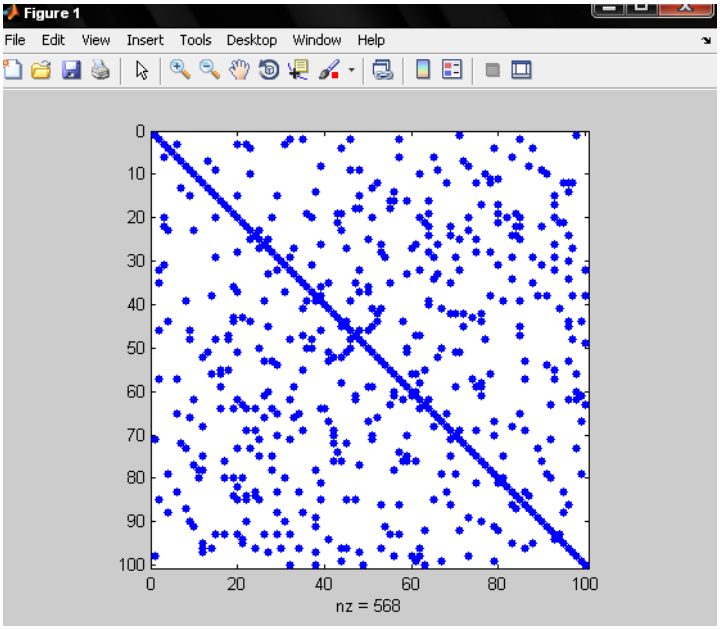
Matriz Dispersa de orden 7

Para poder visualizar nuestra matriz dispersa usamos la función spy que nos provee MATLAB, a continuación se muestra la gráfica de la matriz dispersa que generamos:



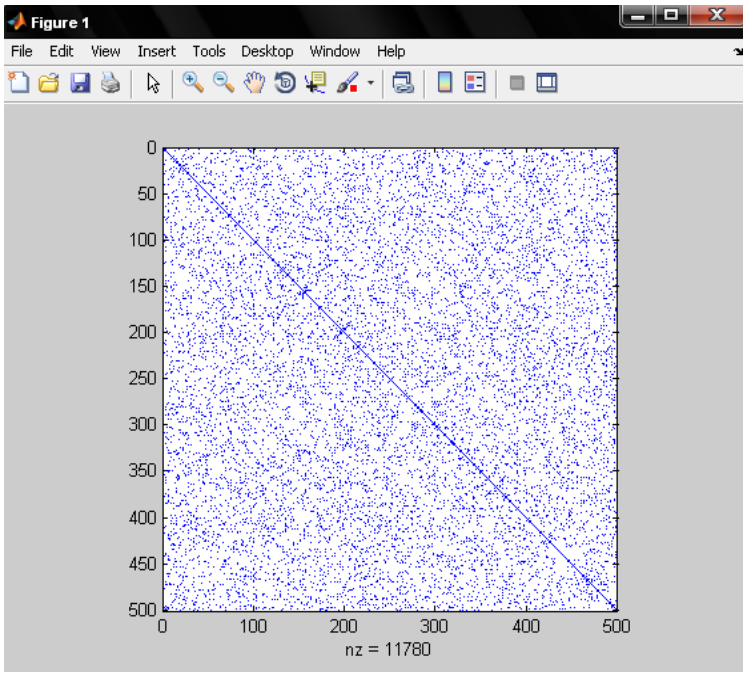

Gráfica de una Matriz Dispersa de Orden 7

Ahora se muestra la gráfica de una matriz dispersa de $tamaño\ 100\ x\ 100$ con un valor de dispersión de 0.85:




Gráfica de una Matriz Dispersa de orden 100

Aquí otra gráfica, esta vez de una matriz dispersa de $tamaño\ 500\ x\ 500$ con un valor de dispersión de 0.85:




Gráfica de una Matriz Dispersa de orden 500

# CAPÍTULO 2

# CUTHILL-MCKEE

En este capítulo se describirá el algoritmo de Cuthill-Mckee, que fue el primer algoritmo que se desarrolló para resolver el BMP.

Desde el origen del BMP, se han desarrollado varios algoritmos para tratar de resolver dicho problema entre los que podemos mencionar:

1. Algoritmo de Cuthill-Mckee.
2. Algoritmo Inverso de Cuthill-Mckee.
3. Algoritmo de Mínimo Grado.
4. Algoritmo de Conteo por Columna.

Como ya se mencionó el primer algoritmo que se implementó para resolver el BMP de matrices dispersas fue el de Cuthill-Mckee, dicho algoritmo fue publicado por E. Cuthill y J. Mckee en su artículo "**Reducing the Bandwidth of Sparse Symmetric matrices**" en 1969, cabe señalar que este algoritmo está basado en teoría de grafos.

Este algoritmo consta de los siguientes pasos:

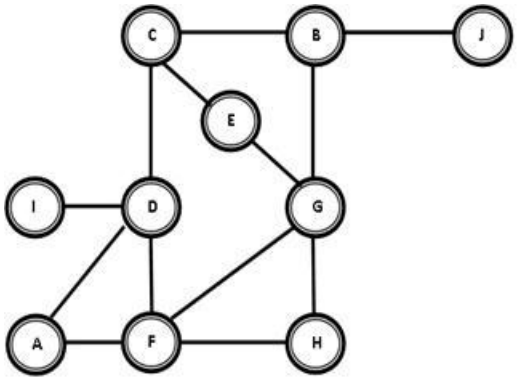
1. Construir una tabla donde se indique el grado de cada uno de los nodos.
2. Elegir un nodo inicial de menor grado.

3. Renumerar los nodos adyacentes al nodo inicial en orden ascendente de grado, los pasos 2 y 3 se repiten hasta que todos los nodos se hayan visitado.

A continuación se describe la manera en la que nosotros implementamos este algoritmo basado en los pasos mencionados:

1. Primero calculamos el *grado* de cada uno de los nodos, para ello contamos el número de nodos adyacentes a dicho nodo, esto lo almacenamos en un vector que denominamos *grado*, luego procedemos a reordenar dicho vector en forma ascendente.
2. Luego se toma el nodo de menor grado se marca como visitado y se coloca en un vector denominado *orden*, luego se escoge el nodo adyacente al nodo que visitamos y se checa si ya está visitado, en caso afirmativo se toma otro nodo adyacente en caso de que el nodo visitado lo tenga, en el caso en que el nodo adyacente no haya sido visitado este se guarda en un vector llamado cola para que una vez que se hayan visitado todos los nodos adyacentes al nodo inicial se visiten los nodos adyacentes al siguiente nodo, también se marca como visitado y se coloca en el vector *orden*, este proceso se continua hasta que todos los nodos hayan sido visitados.
3. Por último procedemos a reordenar nuestra matriz con ayuda del vector *orden*.

A continuación se muestra un pequeño ejemplo supongamos que tenemos el siguiente grafo:




***Grafo de 10 nodos***

Su matriz de adyacencia se muestra en la siguiente tabla:

| Nodo | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| A | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| B | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| C | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| D | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| E | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| F | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| G | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| H | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| I | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| J | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

***Tabla 1 Matriz de Adyacencia Original***

Si calculamos el ancho de banda original de nuestra matriz, nosotros obtenemos el siguiente valor:

$$Ancho\ de\ Banda$$

$$= |1-1| + |1-4| + |1-6| + |2-2| + |2-3| + |2-7| + |2-10|$$
$$+ |3-2| + |3-3| + |3-4| + |3-5| + |4-1| + |4-3| + |4-4|$$
$$+ |4-6| + |4-9| + |5-3| + |5-5| + |5-7| + |6-1| + |6-4|$$
$$+ |6-6| + |6-7| + |6-8| + |7-2| + |7-5| + |7-6| + |7-7|$$
$$+ |7-8| + |8-6| + |8-7| + |8-8| + |9-4| + |9-9| + |10-2|$$
$$+ |10-10|$$

$$Ancho\ de\ Banda\ Original = 76$$

Ahora se muestra el vector llamado ***grado*** donde se almacenan los grados de cada nodo, ya en orden:

| NODO | GRADO |
|---|---|
| (9) I | 1 |
| (10) J | 1 |
| (1) A | 2 |
| (5) E | 2 |
| (8) H | 2 |
| (2) B | 3 |
| (3) C | 3 |
| (4) D | 4 |
| (6) F | 4 |
| (7) H | 4 |

***Tabla 2 Lista de grado de cada nodo***

Al aplicar el algoritmo de Cuthill-Mckee a este ejemplo, obtenemos el siguiente vector ***orden***:

$$\boldsymbol{orden}: \mathbf{9\ 4\ 1\ 3\ 6\ 5\ 2\ 8\ 7\ 10}$$

Ahora se muestra la matriz de adyacencia reordenada para que se observe que los elementos diferentes de cero están más cerca de la diagonal.

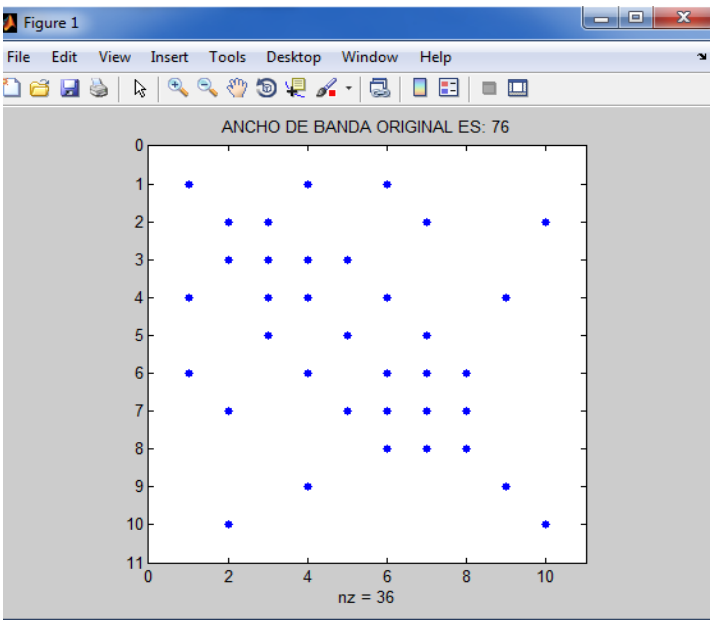
| Nodo | I | D | A | C | F | E | B | H | G | J |
|---|---|---|---|---|---|---|---|---|---|---|
| I | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| D | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| F | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| E | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| B | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| H | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| G | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| J | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

***Tabla 3 Matriz de Adyacencia Reordenada con CM***

Al calcular el ancho de Banda de nuestra matriz, después de aplicar el algoritmo de Cuthill-Mckee obtenemos:

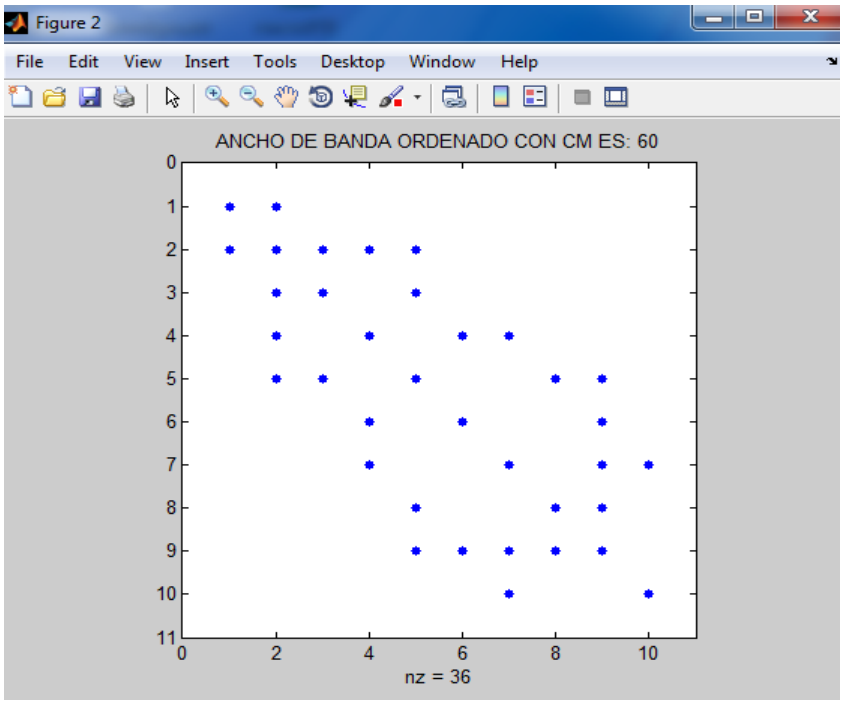
$$Ancho\ de\ Banda\ CM = 60$$

Gráficamente podemos visualizar este ejemplo como sigue:




Gráfica de una Matriz Dispersa de orden 10

Al aplicar el algoritmo de Cuthill-Mckee a nuestra matriz obtenemos la siguiente matriz la cual como se puede observar tiene un ancho de Banda menor:




Gráfica de una Matriz Dispersa de orden 10 después de aplicar CM

# CAPÍTULO 3

# METAHEURÍSTICAS

Este capítulo está integrado por tres secciones. En la primera sección se habla acerca de la metaheurística de Recocido Simulado (Simulated Annealing (SA)), primero daremos una pequeña introducción acerca de los orígenes de la metaheurística, después explicamos cómo funciona el algoritmo.

En la segunda sección se aborda la metaheurística de algoritmos genéticos, en primera instancia daremos una pequeña introducción acerca de sus orígenes, después explicaremos como es que trabajan los algoritmos genéticos. Por último en la tercera sección abordaremos la metaheurística de colonia de hormigas (Ant Colony Optimization (ACO)), con una introducción dicha metaheurística, y el funcionamiento de ella.

Es importante mencionar que estas metaheurísticas han sido implementadas para resolver el BMP (Bandwidth Minimization Problem).

## 3.1 RECOCIDO SIMULADO

La metaheurística de Recocido Simulado (Simulated Annealing (SA)) está inspirada como su nombre lo indica en el proceso de recocido del acero, el cual consiste en someter dicho material a una temperatura muy alta para luego aplicar la calendarización de enfriamiento y así poder variar sus propiedades físicas.

Este algoritmo está basado en el famoso algoritmo de Metrópolis 1953 que es un método Montecarlo el cual sirve para generar muestras de estados en sistemas termodinámicos. El SA se ha utilizado con éxito en problemas de optimización global, el objetivo del SA es buscar un valor óptimo global de una función en un espacio de búsqueda grande.

El algoritmo de Recocido Simulado fue desarrollado por Scott Kirkpatrick, quien en 1983 aplicó el algoritmo de metrópolis para resolver problemas de optimización relacionados con el diseño de circuitos, Cerny fue otro de los autores que colaboró en el desarrollo del SA, él aplicó también el algoritmo de Metrópolis para resolver el TSP (Travel Salesman Problem) en 1985.

Al aplicar el algoritmo de Metrópolis a sus respectivos problemas, ambos autores se dieron cuenta de que era posible establecer una analogía entre los parámetros que intervienen en un problema de termodinámica, y los parámetros que se utilizan en los problemas de optimización local, de esta forma ellos obtuvieron la siguiente analogía:

| TERMODINÁMICA | OPTIMIZACIÓN |
|---|---|
| **Configuración** | Solución Factible |
| **Configuración Fundamental** | Solución Óptima Global |
| **Energía de la Configuración** | Coste de la Solución |
| **Calendarización de Enfriamiento** | Camino hacia el óptimo |

Esta metaheurística trabaja con los siguientes parámetros:

- Temperatura Inicial.

- Un valor denominado calendarización de la temperatura.
- Número máximo de iteraciones.
- Una solución Inicial al problema.
- Temperatura Final.

De esta manera el algoritmo de Recocido Simulado comienza con un valor alto de Temperatura denominado T, también se define una Temperatura Final que nos sirve como criterio de parada, se genera una solución inicial a nuestro problema, al inicio del algoritmo la solución óptima es la misma que la inicial, luego calculamos el costo de nuestra solución óptima actual $(sol_{act})$, después se itera un cierto número de ciclos, dentro de este ciclo, el algoritmo genera una nueva solución candidata $(sol_{cand})$en el espacio N(s) denominado vecindario de Solución que es nuestro espacio de búsqueda, una vez que tenemos nuestra nueva solución evaluamos el costo de dicha solución candidata y comparamos si este costo es menor que el que teníamos entonces sustituimos el costo anterior por el nuevo y la solución óptima pasa a ser la solución que hemos generado, en caso de que el costo no sea menor entonces generamos un número aleatorio entre 0 y 1, si este número es menor o igual que $e^{(-((CostoSol_{cand})-(CostoSol_{act}))/T)}$ conocido como el factor de Boltzmann, entonces reemplazamos el costo anterior por este nuevo costo, y la solución óptima pasa a ser la solución que hemos generado.

Este factor de Boltzmann es la clave del algoritmo de Recocido Simulado, ya que como mencionamos es una estrategia de búsqueda local, en la cual la elección de la nueva solución en el vecindario de solución N(s) se realiza de forma aleatoria y se puede dar el caso de que durante la búsqueda el algoritmo caiga en un óptimo

local y no pueda salir, es por eso que el algoritmo permite con cierta probabilidad, cada vez menor conforme nos acercamos a la solución óptima, el paso de soluciones peores.

De esta manera como mencionamos el algoritmo empieza con un valor alto de temperatura lo cual nos permitirá pasar a soluciones peores en los primeros pasos del algoritmo eso debido a que estamos alejados del óptimo global, pero conforme vayamos disminuyendo la temperatura también iremos reduciendo la posibilidad de pasar a soluciones peores lo cual nos permitirá estar más cerca del óptimo buscado.

A continuación se muestra el pseudocódigo del algoritmo de Recocido Simulado:

$$Algoritmo\ General\ de\ Recocido\ Simulado$$

$Ti$ % $Temperatura\ Inicial$

$Tf$ % $Temperatura\ Final$

$\alpha$ % $alpha$

$No\_rep$ % $Número\ de\ iteraciones$

$T = Ti.$

$Sact = Genera\ Solución\ Inicial$

$Mientras\ T > Tf\ hacer$

$K = 1$

$Mientras\ K <= No\_rep\ hacer$

$Scand = Seleccionar\ solución\ del\ N\ (Sact)$

$Delta = Coste\ (Scand) - Coste\ (Sact)$

$Si\ Coste\ (Scand) < Coste\ (Sact)\ entonces$

$Sact = Scand$

$Si\ no\ entonces$

$U\ =\ rand\ (0{,}1)$

$Si\ U\ <\ exp\ (-Delta/T)\ entonces$

$Sact\ =\ Scand$

$Fin\_Si$

$K\ =\ K\ +\ 1$

$Fin\ Mientras$

$T\ =\ \alpha\ (T)$

$Fin\_Mientras$

Ahora que sabemos cómo trabaja el algoritmo de SA, vamos a explicar la manera en la que nosotros implementamos dicho algoritmo para resolver el problema del BMP.

Nosotros tomamos una temperatura inicial $(T_i)$ de 1, la temperatura final $(T_f)$ que manejamos es de 0.000001, esto porque en la literatura el parámetro T que se recomienda manejar es entre 1 y 0. En cuanto al No_rep que representa el número de ciclo internos del algoritmo lo calculamos multiplicando $n$ que es la dimensión de nuestra matriz por 20, el valor de alpha que tomamos es de 0.95, y en lo que respecta a la solución inicial nosotros tomamos como permutación inicial el orden normal de la matriz es decir $Sact\ =\ 1{:}\ n$.

En cuanto al vecindario de Solución $N(s)$ del algoritmo de Recocido Simulado, cabe mencionar que hay diferentes formas para obtener nuestra nueva solución:

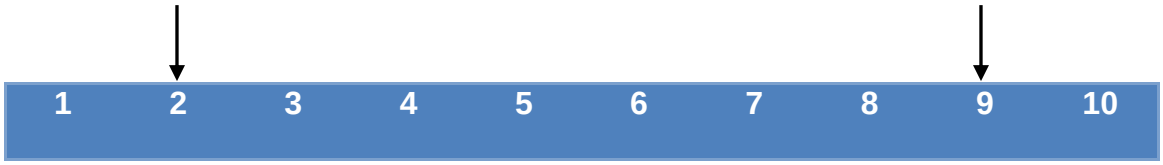
- Vecindario por Intercambio [14].
- Vecindario por traslado [14].

Nosotros utilizamos el tipo de vecindario por intercambio el cual se explica a continuación:

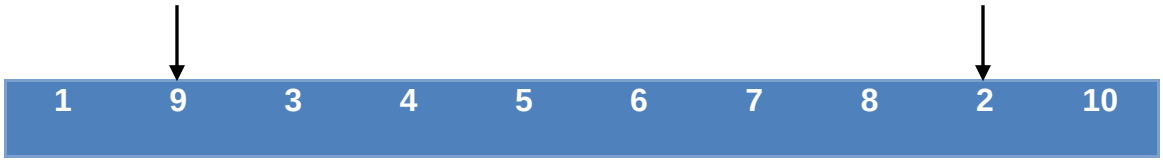
Supongamos que tenemos la siguiente permutación:


Ahora seleccionamos dos puntos cualesquiera de nuestra permutación por ejemplo la posición tres de la permutación y la posición ocho de la misma, lo único que tenemos que hacer es intercambiar dichos valores para generar una nueva permutación que será nuestra nueva solución al problema de esta forma obtenemos:

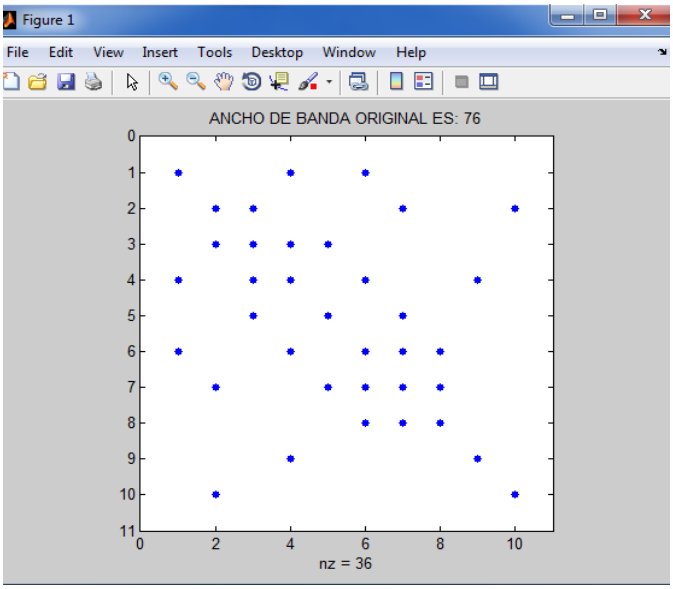

A continuación se muestra un ejemplo específico, supongamos que tenemos la siguiente permutación:



Seleccionamos las posiciones 2 y 9 de nuestra permutación para así obtener nuestra nueva permutación que sería:

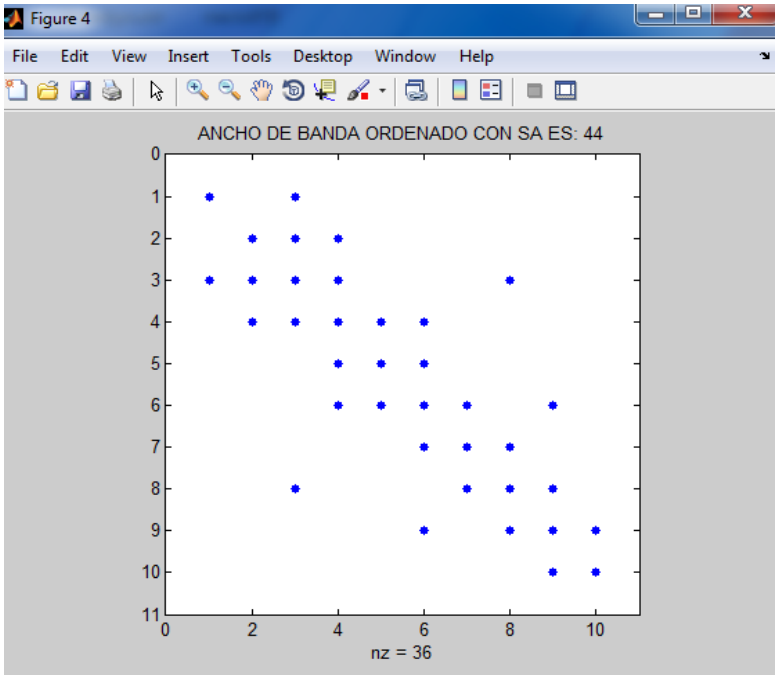



Una vez que generamos nuestra nueva permutación reordenamos la matriz original con dicha permutación, luego calculamos el ancho de banda de dicha matriz y la comparamos con el ancho de banda original si es menor nos quedamos con esta solución y el ancho de banda óptimo lo sustituimos por este nuevo y continuamos con el algoritmo hasta que termina, una vez que el algoritmo terminó reordenamos la matriz original con la mejor permutación que hayamos encontrado. Retomando el ejemplo utilizado en el algoritmo de Cuthill-Mckee, tenemos la siguiente figura que nos muestra la matriz con un ancho de banda original igual a 76:




Gráfica de una Matriz Dispersa de orden 10

Al aplicar el algoritmo de Recocido Simulado nosotros obtenemos la siguiente matriz con un ancho de banda igual a 44 la cual es menor que la original:




Gráfica de una Matriz Dispersa de orden 10 después de aplicar SA

## 3.2 ALGORITMOS GENÉTICOS

La metaheurística de Algoritmos Genéticos (Algorithms Genetic (AG)) está inspirada en el proceso evolutivo de los seres humanos y en la selección natural. Este algoritmo trabaja con una población de individuos de los cuales se seleccionan a los más aptos para posteriormente cruzarlos y mutarlos para así reemplazar a algunos individuos de la población este proceso se repite un cierto número de generaciones. Los algoritmos genéticos se han aplicado con éxito en problemas de optimización global y de combinatoria.

Los principios básicos de los Algoritmos Genéticos fueron establecidos y desarrollados por John Holland 1975, y se encuentran bien descritos en varios textos como: Goldberg 1989, Davis 1991, Michalewicz 1992 y Reeves 1993.

Los parámetros que se manejan en los algoritmos genéticos son:

- Población Inicial.
- Porcentaje de Selección.
- Porcentaje de Mutación.
- Operador de Selección.
- Operador de Cruza.
- Número máximo de Generaciones.

En cuanto al operador de selección nosotros tenemos varios tipos entre los que podemos mencionar:

- Selección por Ruleta.
- Selección por Torneo.
- Selección Elitista.
- Selección por Rango.
- Selección por Estado Estacionario.
- Selección Escalada.

Respecto al operador de cruza, existen varios tipos entre los que destacan:

- Operador de cruza en un punto.
- Operador de Cruza en dos puntos.
- Operador de cruza uniforme.
- Operador de cruza Parcial (PMX) [1].

Como se mencionó anteriormente los AG trabajan con una población de individuos en donde cada individuo es llamado cromosoma y es aquí donde se codifica la solución a nuestro problema, de esta forma nosotros construimos una población de individuos la cual tomaremos como población inicial para iniciar el AG.

Una vez generada nuestra población inicial, se procede a calcular el valor de aptitud de cada individuo, en algunos textos a la aptitud le llaman fitness, posteriormente mediante algún método de selección ya sea por el método de ruleta, torneo, etc., se seleccionan dos o más individuos para poder aplicar el

operador de cruza y así obtener nuevas soluciones al problema, cabe mencionar que en el proceso de selección los individuos mejores adaptados al problema tienen una mayor probabilidad de ser escogidos para cruzarse, una vez que tenemos a los nuevos individuos entonces aplicamos el operador de mutación para alterar a los nuevos individuos, cabe señalar que el umbral de mutación es pequeño .

Después se reemplazan a algunos individuos de nuestra población inicial por los nuevos individuos que se han creado para así poder generar una nueva población de soluciones, estas soluciones estarán más cerca de la solución óptima global buscada, esto debido a que cruzamos a los individuos mejores adaptados al problema, todo lo anterior se repite un cierto número de iteraciones a las que denominamos generaciones.

Así es como trabajan los AG, de esta manera tenemos que un AG general se puede expresar en pseudocódigo de la siguiente forma:

*Algoritmo Genético General*

*Paso* 1. *Generar una Población Inicial.*

*Paso* 2. *Calcular la aptitud de cada individuo de nuestra población.*

*Paso* 3. *Aplicar el Operador de Selección.*

*Paso* 4. *Aplicar el Operador de Cruza para obtener nuevos individuos.*

*Paso* 5. *Aplicar el Operador de Mutación a los nuevos individuos.*

*Paso* 6. *Actualizar nuestra población con los nuevos individuos.*

*Paso* 7. *Repetir del paso* 2 *al paso* 6 *un cierto número de generaciones.*

Para resolver el problema del BMP aplicando la metaheurística de AG, nosotros codificamos nuestro cromosoma de la siguiente manera:

| 3 | 5 | 10 | 2 | 9 | 6 | 8 | 4 | 7 | 1 |
|---|---|---|---|---|---|---|---|---|---|

**Cromosoma**

Como nuestro objetivo es buscar una permutación que nos permita tener los elementos lo más cerca de la diagonal entonces nuestro cromosoma no es más que una simple permutación donde cada casilla representan las filas de nuestra matriz, de esta forma creamos nuestra población inicial, el tamaño de nuestra población inicial es de N/2 donde N es el número de filas de nuestra matriz.

Una vez que generamos nuestra población inicial, calculamos la aptitud de cada uno de los individuos de la población, esto lo hacemos gracias a la función que implementamos para calcular el ancho de banda (concepto que se definió en el primer capítulo), dicho valor lo guardamos en un arreglo para luego reordenarlo en nuestro caso de menor a mayor, ya que con este arreglo reordenamos nuestra población inicial para así tener a los individuos más aptos al principio y los peores al final.

Posteriormente seleccionamos a los individuos que vamos a cruzar, para esto utilizamos el método de selección por torneo, el cual consiste en generar un número aleatorio, si este número es mayor que el valor del umbral que nosotros manejamos, entonces seleccionamos al individuo más apto, si no escogemos al menos apto, de igual forma seleccionamos al otro individuo, en nuestra implementación sólo escogemos a dos individuos para cruzarlos.

Ya que seleccionamos a los individuos que vamos a cruzar utilizamos el operador de cruza que hayamos escogido, en este trabajo utilizamos el operador de cruza basado en una correspondencia parcial (PMX), para evitar la repetición de valores en nuestra solución, la forma de trabajar de este operador es la siguiente:

Supongamos que tenemos a los siguientes individuos:

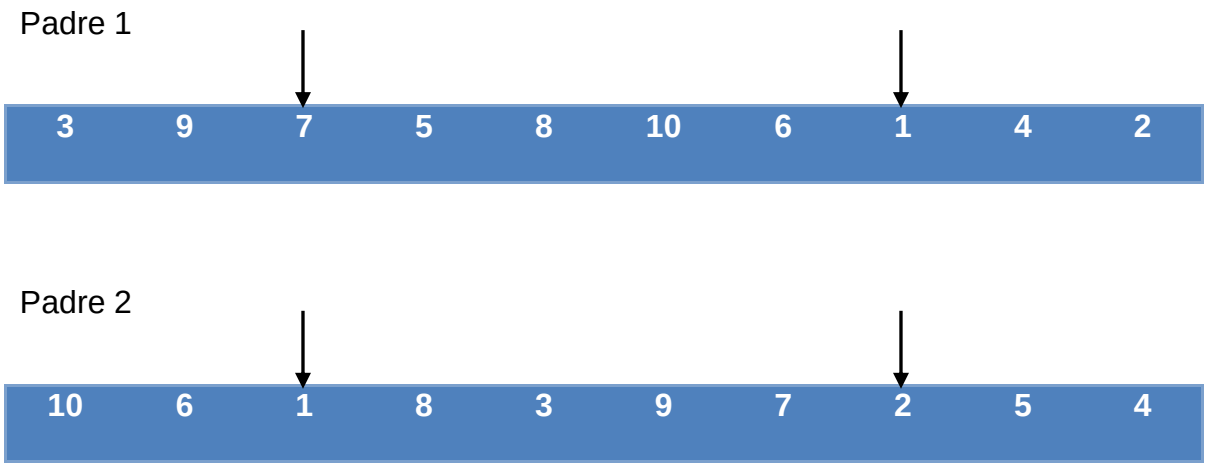



**Figura 1**

Seleccionamos dos puntos cualesquiera de nuestros cromosomas, para poder generar un segmento de cruza en nuestro ejemplo escogimos los puntos 3 y 8, luego procedemos a intercambiar los elementos que estén en el segmento de cruza, es decir el hijo 1 tendrá los primeros genes del padre 1, mientras que en el segmento de cruza tendrá los genes del padre 2 y los genes restantes del padre 1, por su parte el hijo 2 tendrá los primeros genes del padre 2, en su segmento de cruza irán los genes del padre 1 y los genes que queden serán del padre 2. De esta manera obtenemos los siguientes dos hijos:

Hijo 1


Hijo 2


**Figura 2**

Vamos a cambiar los valores 3, 9 y 2 que están fuera del segmento de cruza, para ello tenemos que tomar en cuenta que el elemento 3 del hijo 1 tiene una correspondencia con el elemento 8 del segundo hijo, de esta forma obtenemos:

Hijo 1


**Figura 3**

Como el 8 también es un elemento repetido, necesitamos cambiar este valor para ello se utiliza la correspondencia que hay entre el 8 y el 5 (ver figura 2), entonces reemplazamos el 8 por el 5 para obtener:


**Figura 4**

Hasta el momento sólo hemos quitado un elemento repetido, para el 9 nosotros utilizamos la correspondencia que hay entre el 9 y el 10 (ver figura 2), al efectuar este cambio obtenemos:


**Figura 5**

Finalmente para quitar el 2, primero utilizamos la correspondencia que hay entre 2 y 1 (ver figura 2), al efectuar el intercambio de 2 por 1 nos daremos cuenta de que el 1 se repite, ahora empleamos la relación que hay entre 1 y 7(ver figura 2) para reemplazar al hacer esto de nueva cuenta el 7 se repite así que utilizamos la relación entre 7 y 6, al efectuar este cambio obtenemos a nuestro nuevo hijo el cual ya no tiene elementos repetidos:

Hijo 1


**Figura 6**

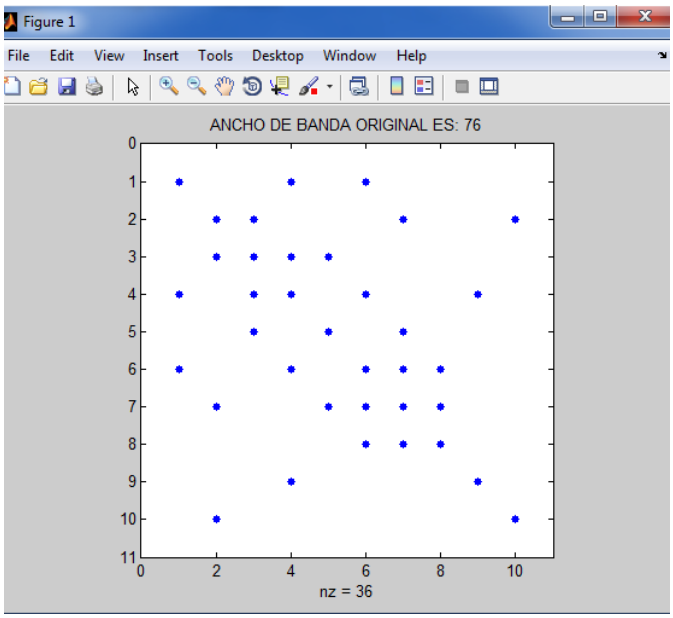
Para el hijo 2 se reemplazan los elementos 10, 6 y 5 de forma similar a como se hizo para el hijo 1.

Hijo 2


**Figura 7**

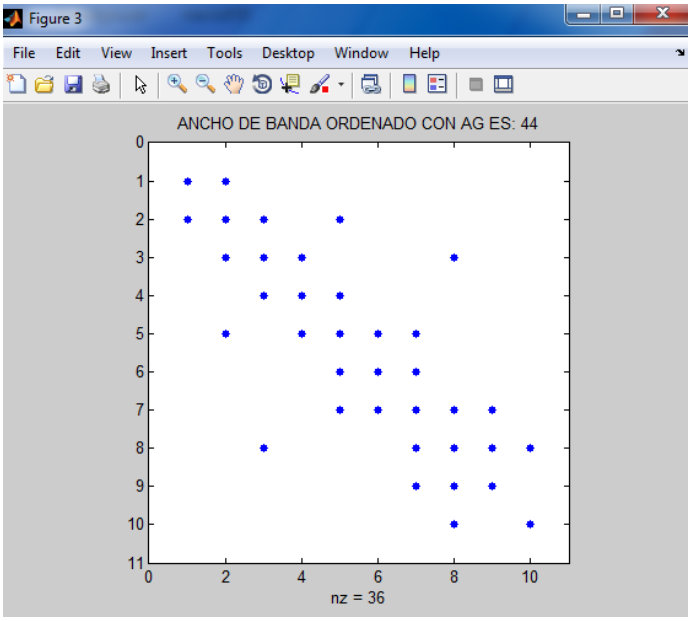
Después decidimos si mutamos o no a los nuevos individuos para ello generamos dos números aleatorios, uno por cada hijo, si estos números son menores que el umbral de mutación que utilizamos entonces mutamos al individuo para ello utilizamos el vecindario de intercambio, dicho método ya se explicó en el capítulo anterior de Recocido Simulado, luego reemplazamos a los individuos menos aptos de nuestra población por estos nuevos. Finalmente todo esto lo repetimos un cierto número de generaciones, en este trabajo es de n*100.

Retomando el ejemplo utilizado en el algoritmo de Cuthill Mckee, la siguiente figura muestra nuestra matriz original, la cual tiene un ancho de banda de 76:




Gráfica de una matriz dispersa de orden 10

Aplicando algoritmos genéticos, obtenemos la siguiente matriz que tiene un ancho de banda de 44 el cual es menor que el original:




Gráfica de una matriz dispersa de orden 10 después de aplicar AG

## 3.3 COLONIA DE HORMIGAS

La metaheurística de Colonia de Hormigas fue propuesta por Marco Dorigo en 1992, el ACO está inspirado en el comportamiento que muestran las hormigas dentro de su colonia para buscar su alimento. Este algoritmo se ha aplicado a varios problemas como el del agente viajero (TSP), el enrutamiento de redes, etc. La hormiga sale del nido y a través de su recorrido va dejando un rastro de una sustancia denominada feromona, para que las demás hormigas la sigan, si el rastro de feromona es alto entonces las hormigas siguientes pasarán por ese camino, en caso de que no sea muy alto el rastro entonces la hormiga elegirá otro camino de acuerdo a una probabilidad.

Esta metaheurística trabaja con los siguientes parámetros

- Número de Hormigas.
- Matriz de Distancias.
- Matriz de Visibilidad.
- Un parámetro de alpha entre 0 y 1.
- Un parámetro de beta entre 0 y 1.
- Número de iteraciones.

El algoritmo ACO trabaja con una matriz de distancias $D$, con ayuda de ésta se construye otra matriz $H$ que denominamos de visibilidad, por último tenemos la matriz de feromonas $tao$ que al principio tiene un valor constante. Una vez que tenemos dichas matrices y hemos seleccionado el número de hormigas, se procede a colocar a dicha hormiga en una posición aleatoria, que representa el

lugar de donde va a partir la hormiga, después cada hormiga va a construir la solución a nuestro problema, para decidir a qué lugar se va a mover cuando llega a una bifurcación, la hormiga utiliza la siguiente regla de transición para saber hacia dónde se va a ir:

$$P_{i,j} = \frac{tao_{i,j}^{\alpha} * h_{i,j}^{\beta}}{\sum_{i,j=1}^{n} tao_{i,j}^{\alpha} * h_{i,j}^{\beta}} \qquad regla\ de\ transición$$

Donde $\alpha$ y $\beta$ son mayores que 0, $tao_{i,j}$ es la concentración de feromona entre $i$ e $j$, $h_{i,j}$ es el valor de visibilidad, de esta forma cada hormiga construye la solución a nuestro problema

Una vez que hemos construido nuestra solución, el siguiente paso es evaluar el costo de nuestra solución, de esta manera nosotros calculamos el costo de cada una de las soluciones que generamos, y nos quedamos con la mejor, luego comparamos ese valor con el valor óptimo global que tenemos si este es mejor entonces sustituimos el valor anterior por el nuevo si no lo dejamos igual.

Por último nos falta actualizar la matriz de feromonas para ello utilizamos la siguiente fórmula:

$$tao_{i,j}^{t+1} = (1-\tau)tao_{i,j}^{t} + \delta tao_{i,j}^{t}$$

Donde $\tau \in [0,1]$ representa la evaporación de la feromona, por otra parte $\delta tao_{i,j}^{t}$ se define como:

$$\delta tao_{i,j}^{t} = {}^{10}/_{L}$$

Donde $L$ es el costo de nuestra solución. Todo lo anterior lo repetimos un cierto número de veces.

En general así es como trabaja la metaheurística de Colonia de Hormigas, a continuación se muestra el pseudocódigo de ACO en forma general:

$$Algoritmo\ de\ Colonia\ de\ Hormigas\ General$$

$Inicialización\ de\ parámetros\ (matriz\ de\ feromonas, distancia\ y\ visibilidad,$

$número\_hormigas, número\_iteraciones)$

$Para\ i = 1: número\_iteraciones\ hacer$

$Para\ k = 1: número\_hormigas$

$L\ (k, i)\ =\ nodo\ inicial;$

$End\_Para$

$Para\ i = 2\ hasta\ Número\_de\_nodos\ hacer$

$Para\ k = 1: hasta\ número\_hormigas\ hacer$

$L\ (k, i)\ =\ Regla\ de\ transición$

$End\_Para$

$End\_Para$

$Para\ k = 1\ hasta\ número\_hormigas\ hacer$

$Coste\ [k]\ =\ C\ (L\ (k)$

$Mejor\_Actual\ =\ Mejor\ (L\ (k))$

$End\_Para$

$Para\ i = 1\ hasta\ Número\_de\_nodos\ hacer$

$Para\ j = 1\ hasta\ Número\_de\_nodos\ hacer$

$Actualización\ de\ feromona$

$End\_Para$

$End\_Para$

$Si\ C\ (Mejor\_actual)\ <\ C\ (Mejor\_Global)\ entonces$

$Mejor\_Global\ =\ Mejor\_Actual;$

$End\_Si$

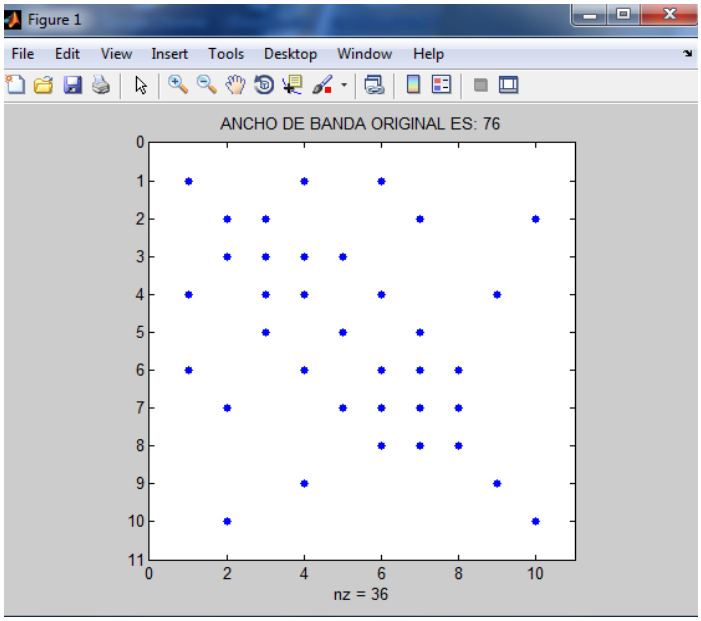
$End\_Para$

$Devolver\ Mejor\_Global$

Para resolver el BMP nosotros adaptamos el algoritmo de colonia de hormigas de la siguiente manera, nosotros manejamos un valor de alpha $\alpha = 2$, y beta $\beta = 5$, el coeficiente de evaporación $e = 0.6$, en cuanto al número de iteraciones nosotros manejamos el valor de 100 y el número de hormigas es de $n/2$ donde $n$ es la dimensión de la matriz con la que estamos trabajando.

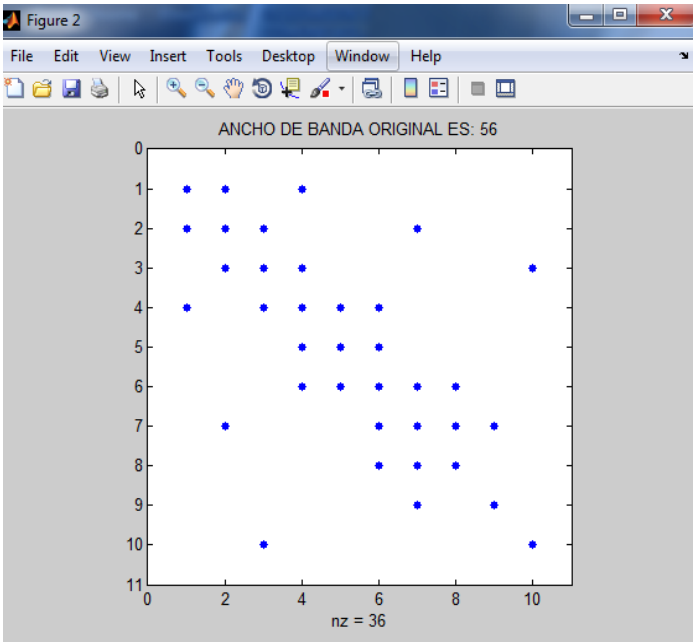
En cuanto a la matriz de distancia que necesitamos para el algoritmo, como nosotros manejamos matrices dispersas, entonces donde haya un elemento diferente de cero en nuestra matriz de distancia tendremos el valor absoluto de la diferencia entre $i$ e $j$, para calcular las entradas de nuestra matriz de visibilidad nosotros hacemos $1/d_{i,j}$, la matriz de feromonas nosotros la inicializamos con el valor de 1 * (0.001).

Retomando el ejemplo utilizado en el capítulo de Cuthill-Mckee, la siguiente figura nos muestra la gráfica de la matriz original la cual tiene un ancho de banda de 76:




Gráfica de la matriz original

Al aplicar el algoritmo de colonia de hormigas obtenemos la siguiente figura en la cual se observa que el ancho de banda es 56 el cual es menor que el original.




Gráfica de la matriz después de aplicar Colonia de Hormigas

# CAPÍTULO 4

# IMPLEMENTACIÓN Y PRUEBAS

Para resolver el problema del BMP, tuvimos que diseñar un programa que consta de las siguientes funciones:

1. Función Generamatriz.
2. Función Mckee.
3. Función SA.
4. Función AG.
5. Función ACO.

La primera función nos permite generar nuestra matriz dispersa de forma aleatoria, la función Mckee sirve para aplicar el algoritmo de Cuthill-Mckee [2] [10], por otra parte la función SA se utiliza para aplicar la metaheurística de Recocido Simulado, así mismo la función AG nos permite utilizar la metaheurística de Algoritmos Genéticos, finalmente la función ACO nos sirve para trabajar con la metaheurística de Colonia de hormigas.

Tanto el programa como las funciones se implementaron en una máquina con procesador Intel Quad-Core a 2.8 GHz, con memoria RAM de 4GB, sistema operativo Windows XP, el programa que se utilizó para programar es MATLAB 2010.

Las funciones de Mckee, SA, AG, y ACO, se implementaron con los parámetros y las especificaciones descritas en los capítulos respectivos.

Enseguida se muestra un pequeño ejemplo en donde se generó una matriz de dimensión 50 con un porcentaje de dispersión de 0.90, en la figura 1 se muestra la matriz original con su ancho de banda, luego en la figura 2 mostramos la matriz reordenada con Mckee, después en la figura 3 se muestra la matriz reordenada con Algoritmos Genéticos, posteriormente en la figura 4 con Recocido Simulado y finalmente en la figura 5 con Colonia de Hormigas:

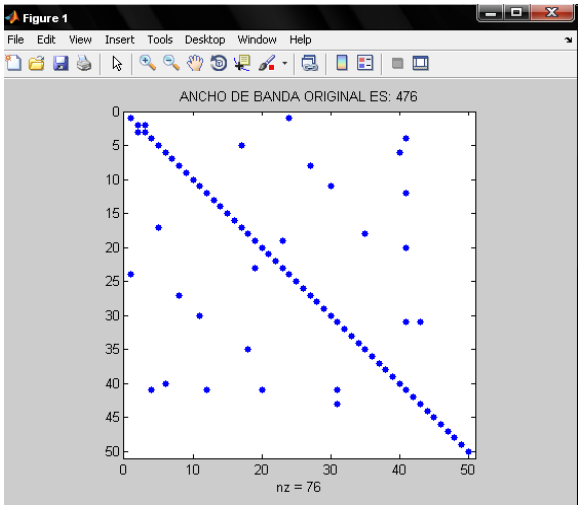



**Figura 1. Gráfica de la Matriz Original**

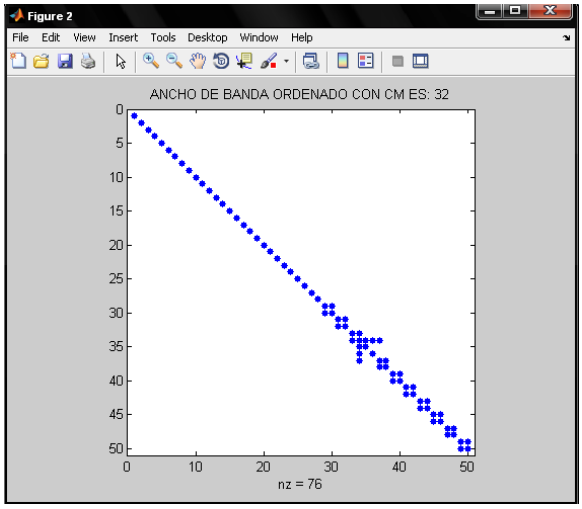



**Figura 2. Gráfica de la Matriz Reordenada con Cuthill-Mckee**

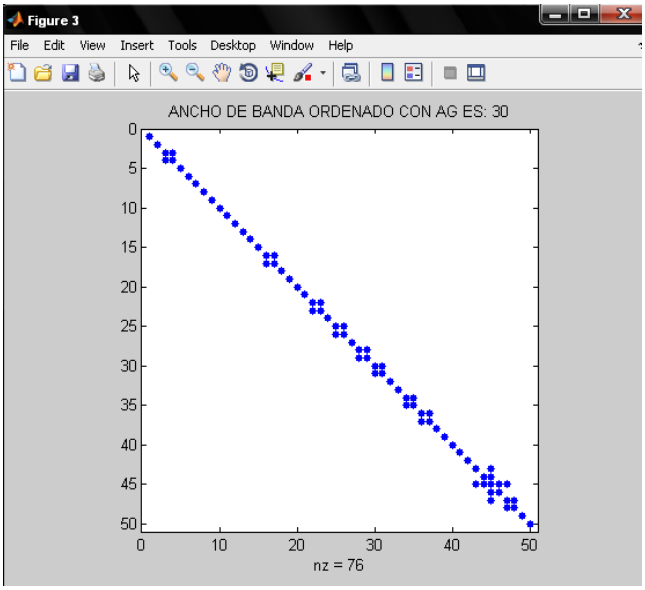



**Figura 3. Gráfica de la Matriz Reordenada con Algoritmos Genéticos**

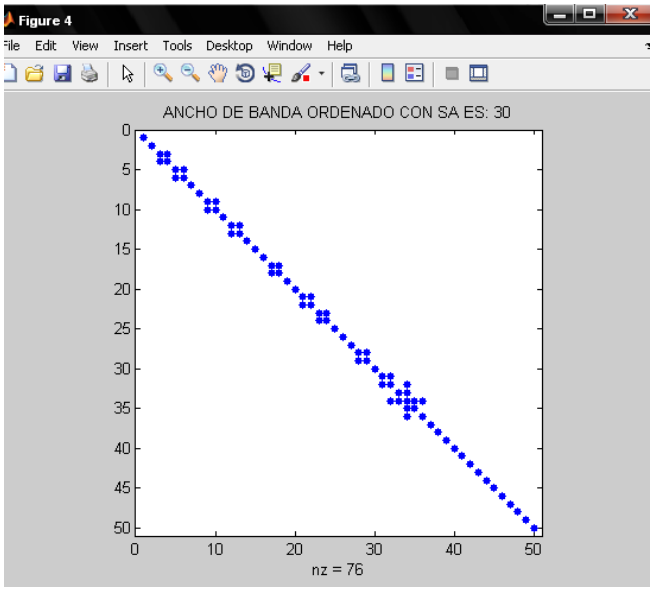



**Figura 4. Gráfica de la Matriz Reordenada con Recocido Simulado**

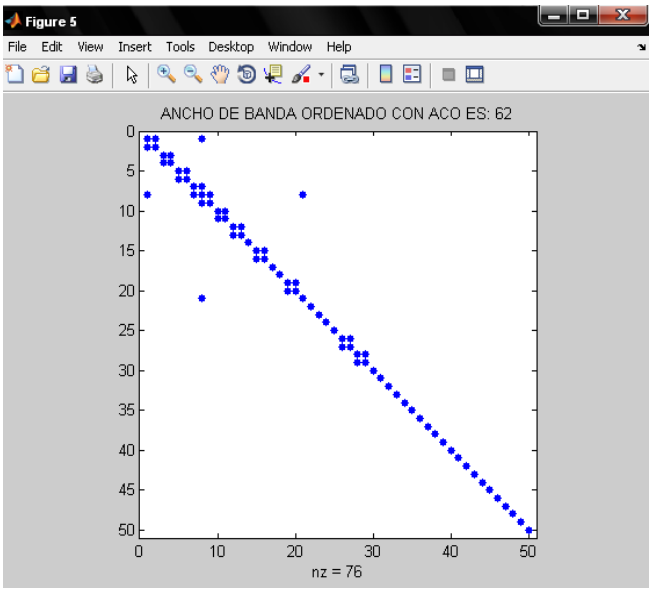



**Figura 5. Gráfica de la Matriz Reordenada con Colonia de Hormigas**

A continuación se muestra la primera tabla que obtuvimos al aplicar los algoritmos de Cuthill-Mckee, Algoritmos Genéticos y Recocido Simulado, para obtener dicha tabla primero se generaron las matrices con las dimensiones dadas, y luego se aplicaron los tres algoritmos a la misma matriz, sólo se corrieron una vez.

| DIMENSION | NNZ | ANCHO BANDA ORIGINAL | ANCHO BANDA CM | ANCHO BANDA AG | ANCHO BANDA SA |
|---|---|---|---|---|---|
| 50 | 498 | 7786 | 6018 | 4912 | 4734 |
| 100 | 2012 | 64010 | 59470 | 47894 | 46240 |
| 150 | 4078 | 195746 | 185068 | 153608 | 148216 |
| 200 | 7098 | 454308 | 440178 | 370646 | 363546 |
| 250 | 11330 | 930826 | 882854 | 770222 | 761320 |
| 300 | 16468 | 1625500 | 1567898 | 1374158 | 1342818 |

| | | | | | |
|---|---|---|---|---|---|
| 350 | 22442 | 2558006 | 2519066 | 2212534 | 2166648 |
| 400 | 29208 | 3842822 | 3763598 | 3341058 | 3279980 |
| 450 | 36472 | 5393368 | 5253126 | 4724636 | 4678946 |
| 500 | 45694 | 7555418 | 7395600 | 6631944 | 6533290 |

**Tabla 1 Comparación de Algoritmos**

Como se observa las metaheurísticas de Algoritmos Genéticos y Recocido Simulado arrojaron un menor ancho de banda con respecto al algoritmo de Cuthill-Mckee.

Después de haber obtenido estos resultados provisionales, se procedió a implementar otra metaheurística, en primer lugar se había decidió implementar PSO, se realizó una investigación profunda acerca de dicha heurística pero al adaptarla al problema se tuvieron algunos inconvenientes ya que no se pudo hallar la forma de actualizar los vectores de velocidad con los que trabaja dicha metaheurística, debido a eso no fue posible implementarla.

Así que se decidió implementar la metaheurística de Colonia de Hormigas, cabe mencionar que cuando se implementó dicho algoritmo también tuvimos algunos inconvenientes, la primera implementación que hicimos parecía que estaba bien, pero al compararla con los demás algoritmos nos dimos cuenta de que los resultados que obteníamos no concordaban con los mostrados en la tabla anterior.

Por este motivo realizamos otra implementación del algoritmo en donde se le modificó la forma en la que se recordaba los nodos ya visitados y dicha implementación arrojó mejores resultados que la primera de esta forma se obtuvo la siguiente tabla de resultados:

| DIMENSION | MCKEE | | ALGORITMOS GENETICOS | | | | RECOCIDO SIMULADO | | | | COLONIA DE HORMIGAS | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **REDUCCION (%)** | TIEMPO (Seg) | **REDUCCION (%)** | VMIN (%)) | VMAX (%) | TIEMPO (Seg) | **REDUCCION (%)** | VMIN (%) | VMAX (%) | TIEMPO (Seg) | **REDUCCION (%)** | VMIN (%) | VMAX (%) | TIEMPO (Seg) |
| 50 | **85.9327** | 0.1811 | **85.3624** | 75.8410 | 90.6728 | 8.6914 | **90.2018** | 85.6269 | 92.6606 | 23.1247 | **74.9113** | 74.3119 | 75.2294 | 2.6352 |
| 100 | **67.8471** | 0.0211 | **72.7690** | 65.0893 | 78.9721 | 146.1328 | **78.8743** | 71.5136 | 85.0517 | 156.4938 | **48.0865** | 46.8819 | 48.8248 | 14.4200 |
| 200 | **26..4632** | 0.1794 | **55.4830** | 51.1833 | 59.5927 | 1.4815e+003 | **60.9257** | 57.3500 | 64.4294 | 0.8014e+003 | **30.2835** | 29.4990 | 33.1254 | 0.0778e+003 |
| 500 | 17.7449 | 1.1388 | 39.98.80 | 39.3138 | 40.8598 | 1.11046e+005 | 43.86876 | 42.9883 | 44.8445 | 0.24065e+005 | 18.6673 | 18.2803 | 19.1529 | 0.0109e+005 |

**Tabla 2 (Comparación de los diferentes metaheurísticas)**

Para las matrices de dimensión 50, 100 y 200 el programa se corrió 100 veces y se calculó el promedio de esas 100 iteraciones.

Como se puede observar en la tabla 2 reportamos el promedio del porcentaje de reducción de Mckee y las tres metaheurísticas implementadas, el tiempo que tardó cada algoritmo y el valor mínimo y máximo del porcentaje que redujo el algoritmo. Para el caso de la matriz de dimensión 500 el programa sólo se corrió 10 veces, se calculó el promedio y es lo que se reportó, para la matriz de dimensión 1000 el algoritmo no se reportó por que lleva demasiado tiempo para terminar.

# CONCLUSIONES

Se estudió a fondo los algoritmos de Cuthill-Mckee, así como las metaheurísticas de Recocido Simulado, Algoritmos Genéticos y Colonia de Hormigas, para comprender la forma de trabajar de cada uno de ellos.

Se adaptaron las metaheurísticas mencionadas anteriormente para poder resolver el problema del BMP, para ello se definió una función que nos permite calcular el ancho de banda de nuestras matrices, dicha función esta explicada en la definición 1.2 de este trabajo.

Una vez que adaptamos el problema del BMP a cada uno de las metaheurísticas, se implementó cada uno de estos algoritmos.

Posteriormente se implementó cada uno de las metaheurísticas, es importante recalcar que para la implementación se realizó una calibración de los parámetros de dichos algoritmos,

- Para el recocido simulado se determinó el valor de la temperatura inicial y final, así como el número de ciclos del algoritmo y el valor de calendarización de la temperatura,
- En el caso de algoritmos genéticos se calculó el tamaño de la población, el número de generaciones, también se definió que operador de selección, cruza y mutación se iba a aplicar,

- Para Colonia de Hormigas el valor de alpha, de beta, la matriz de distancias, la matriz de visibilidad, el número de hormigas fueron los parámetros que se calibraron.

Después de haber implementado los algoritmos, se hizó una comparación del algoritmo de Cuthill-Mckee con respecto a las metaheurísticas, para ello generamos matrices dispersas aleatorias de dimensión 50, 100, 200 y 500 respectivamente, una vez generada la matriz se aplicaban los 4 algoritmos y se calculó el ancho de banda de las matrices después de aplicar dichos algoritmos (ver tabla 2 capitulo 4). Como se puede observar en la tabla 2 cuando tenemos matrices de dimensión pequeña el tiempo que tarda el programa en aplicar todos los algoritmos es poco, sin embargo conforme aumenta la dimensión de las matrices este tiempo aumenta.

Es importante aclarar que el resolver el BMP utilizando metaheurísticas, dió mejores resultados que el algoritmo de Cuthill-Mckee, esto es porque el algoritmo de Mckee siempre llega a un óptimo local, mientras que con las metaheurísticas se aproxima al óptimo global, ya que se puede explorar más el espacio de solución.

De las tres metaheurísticas implementadas, el algoritmo de Recocido Simulado fue la que mostró mejores resultados en comparación con las otras dos que son Algoritmos Genéticos y Colonia de Hormigas, es decir redujo un mayor porcentaje del ancho de banda de nuestras matrices respecto a los otros algoritmos.

Si bien al inicio del desarrollo de esta tesis, el dominio que se tenía acerca del tema era muy poco, conforme se fue avanzado, este dominio fue aumentando, hasta quedar familiarizado con el tema, por lo que este trabajo se ha concluido de manera satisfactoria, obteniendo los resultados que esperábamos tener cuando iniciamos esta tesis, además de que me llena de orgullo haber terminado este trabajo, ya que me ha servido para crecer profesionalmente.

En lo personal el desarrollo de este trabajo me ha servido de mucho, ya que gracias a eso he logrado manejar de una manera más profunda este tema, también me ha servido para desarrollar nuevas técnicas de programación, además me ha ayudado a reforzar todos los conocimientos y habilidades que tengo.

Para terminar cabe mencionar que este trabajo tiene todavía cosas que se pueden mejorar. Por ejemplo, en cuanto al tiempo que tardan los algoritmos en correr sobre todo con matrices grandes se puede buscar una manera de optimizarlo, también se podrían implementar otras metaheurísticas como colonia de abejas, búsqueda harmónica, luciérnagas y compararlas con las que ya tenemos.

# BIBLIOGRAFÍA


1. Abdelmalik Moujahid, Iñaki Inza y Pedro Larrañaga, “Tema 2. Algoritmos Geneticos”, Departamento de Ciencia de la Computación e Inteligencia Artificial Universidad del País Vasco-Euskal Herriko Unibertsitatea, pp.1-33.
2. Alberto Román Flores, 2012, “Sistema de Ecuaciones Lineales Para Matrices Dispersas”, Tesis de Licenciatura BUAP.
3. A. Kaveh, P. Shafari, 2007, "A simple Ant Algorithm for profile optimization of Sparse matrices", Asian Journal of civil Engineering, vol. 9, No 1, pp. 35-46.
4. Andrew Lim, Brian Rodrigues, Fei Xiao, 2002, "A genetic Algorithm with Hill Climbing for the Bandwidth Minimization Problem", available from Citeseer-X (2002).
5. Andrew Lim, Jing Lin, Brian Rodrigues, 2005, "Ant colony optimization with Hill Climbing for the Bandwidth Minimization Problem", Applied Soft Computing, vol. 6, pp. 180-188.
6. Andrew Lim, Jing Lin, Fei Xiao, 2007, "Particle Swarm Optimization and Hill Climbing for the Bandwidth Minimization Problem", Computer Science Applied Intelligence, vol. 6, pp. 175-182.
7. Behrooz Koohestani, Riccardo Poli, 2010, "A genetic Programming Approach to the Matrix Bandwidth Minimization Problem", LNCS 6239, pp. 482-491.
8. Camelia Pintea, Gloria Cerasela Crisan, Camelia Chira, 2010, "A hybrid ACO Approach to the Matrix Bandwidth Minimization Problem", Real World HAIS Applications and Data Uncertaint, pp. 405-412.

9. Damián Ginestar Peiró, 2012, “Matrices Dispersas”, Departamento de Matemática Aplicada, Universidad Politécnica de Valencia, pp. 13-14.
10. E. Cuthill, J. Mckee, 1969, “Reducing the Bandwidth of Sparse Symmetric matrices”, ACM, pp. 157-172.
11. Eduardo Rodríguez-Tello, Jin-Kao Hao, José Torres-Jiménez, 2004, "An improved Evaluation Function for the Bandwidth Minimization Problem", LNCS 3242, pp. 652-661.
12. Fred Glover, Hassan M. Ghaziri, J. L. González, Manuel Laguna, Pablo Moscato, Fan T. Tseng, 1996, “Optimización Heurística y Redes Neuronales”, Editorial Paraninfo.
13. G. M. Del Corso, G. Manzini, 1999, "Finding Exact Solutions to the Bandwidth Minimization Problem.”, Computing, 62, pp. 189-203.
14. Mario César Vélez, José Alejandro Montoya, 2007, “Metaheurísticas: Una alternativa para la solución de problemas combinatorios en administración de operaciones”, Revista EIA, ISSN 1794-1237, Número 8, p. 99-115.
15. Vicente Campos, Estefanía Piñana, Rafael Martí, 2011,"Adaptive Memory Programming for Matrix Bandwidth Minimization", Ann Oper Res, 183, pp. 7-23.
16. Xin-She Yang, 2008, “Nature-Inspired Metaheuristic Algorithms”, Luniver Press.