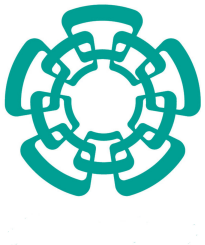


Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Laboratorio de Tecnologías de Información,
CINVESTAV-Tamaulipas

Algoritmos para el Problema de Minimización del Ancho de Banda Cíclico en Grafos Generales

Tesis que presenta:

Hillel Romero Monsivais

Para obtener el grado de:

Maestro en Ciencias en Computación

Director de la Tesis:
Dr. Eduardo Arturo Rodríguez Tello

Cd. Victoria, Tamaulipas, México

Diciembre, 2012

Nota aclaratoria:

Las reglas ortográficas que se siguieron en la escritura de esta tesis son las publicadas en el año 2010 por la Real Academia Española y la Asociación de Academias de la Lengua Española. De acuerdo a estas reglas para las palabras *este*, *esta*, *estos*, *estas*, *ese*, *esa*, *esos*, *esas*, *aquel*, *aquella*, *aquellos*, *aquellas* no se toma en consideración la posible ambigüedad entre determinantes demostrativos y pronombres demostrativos. Antiguamente se acentuaban cuando tenían función de pronombre. Las palabras *esto*, *eso*, *aquello*, que solo pueden ser pronombres, nunca se han acentuado. La palabra '*solo*' puede funcionar como adjetivo o como adverbio. Antiguamente se acentuaba cuando tenía función de adverbio, equivalente a solamente (Real Academia Española, 2010).

La tesis presentada por Hillel Romero Monsivais fue aprobada por:

Dr. Ricardo Landa Becerra

Dr. Gregorio Toscano Pulido

Dr. Eduardo Arturo Rodríguez Tello, Director

Cd. Victoria, Tamaulipas, México, 14 de Diciembre de 2012

Al único y soberano Dios

Agradecimientos

Gracias al Centro de Investigación y de Estudios Avanzados del IPN (CINVETAV-IPN), Unidad Tamaulipas por permitir desarrollarme académicamente en su plantel, por dejarme formar parte de su grupo y por todas las facilidades otorgadas para esta investigación.

Agradezco a mi asesor el Dr. *Eduardo A. Rodríguez Tello*, por introducirme en el mundo de la ciencia y demostrar que puedo ser mejor y que puedo dar más de lo que creo. Gracias por sus consejos, su ayuda y paciencia.

De igual manera les agradezco a todos los investigadores y al personal administrativo del Centro por todas sus enseñanzas y su amistad, pues forjaron en mi un nuevo carácter y una nueva persona.

A mis compañeros de generación muestro mi gratitud por estar conmigo en esta etapa de mi vida y por ayudarme a hacer más llevaderos mis momentos difíciles.

Le doy las gracias a Dios por sostenerme en los momentos de flaqueza, ayudarme a renovar mis fuerzas; y permitirme realizar este posgrado.

El agradecimiento más importante es a mi familia. A mi padre *Joaquín Romero Villegas*, por creer en mí y contagiarme en todo momento de su optimismo. A mi madre *Alicia Monsivais Gutierrez*, por darme ese espíritu de lucha. A mi hermano *Ivan Romero Monsivais*, por enseñarme que las personas puede cambiar si se lo proponen.

Esta tesis ha sido parcialmente financiada por la beca de estudios de maestría CONACyT No. 375305, así como los siguientes proyectos CONACyT: Algoritmos para la Canonización de Covering Arrays, No. 99276 y Fondo Mixto CONACyT y Gobierno del Estado de Tamaulipas, No. 51623.

Índice General

Índice de Figuras

Índice de Tablas

Índice de Algoritmos

Resumen

Algoritmos para el Problema de Minimización del Ancho de Banda Cíclico en Grafos Generales

por

Hillel Romero Monsivais

Maestro en Ciencias del Laboratorio de Tecnologías de Información, CINVESTAV-Tamaulipas
Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2012
Dr. Eduardo Arturo Rodríguez Tello, Director

El problema de *Minimización del Ancho de Banda Cíclico* (MABC) es un problema de grafos embebidos. Este fue establecido por Leung *et al.* (1984) en relación al diseño de una red tipo anillo. Su proposito era encontrar una disposición dentro del ciclo para un conjunto de computadoras V con un patron de comunicación conocido, dado por el grafo $G(V, E)$, de manera que cada mensaje enviado llegara a su destino en máximo k pasos. El problema de MABC surge también en otras áreas de aplicación como el diseño de circuitos VLSI, la representación de estructuras de datos, el diseño de código y las redes de interconexión para sistemas de cómputo paralelo. Como es el caso de muchos problemas de grafos embebidos, fue demostrado que encontrar el ancho de banda cíclico es NP-completo para grafos generales. Actualmente existen reportados estudios que abarcan sus propiedad básicas, límites teóricos y comparativas con otros problemas de etiquetado de grafos. Sin embargo, existe la necesidad de desarrollar algoritmos eficientes que permitan resolver este problema para el caso de grafos generales.

En este trabajo de tesis se presenta un algoritmo exacto de Ramificación y Poda y uno de tipo metaheurístico que implementa Búsqueda Tabú para el problema de MABC. Sus componentes principales fueron diseñados cuidadosamente después de un analisis profundo del problema planteado. La eficiencia de estos algoritmos fue evaluada mediante un proceso de experimentación con 78 grafos, provenientes de cuatro conjuntos distintos. Se llevó a cabo una etapa de sintonización de parámetros para el algoritmo metaheurístico mediante pruebas de interacción combinatoria utilizando arreglos de cobertura (*covering arrays*). Los resultados muestran que el algoritmo exacto propuesto llega al límite

inferior teórico de los grafos de orden $n \leq 35$ en un tiempo de cómputo razonable. Este enfoque pudo resolver de manera exacta el 53.84 % de todas las instancias empleando un límite máximo de 36 horas de CPU. El algoritmo metaheurístico fue capaz de resolver grafos aún más grandes y en un menor tiempo, mejorando la cota de 30 instancias con respecto a las encontradas por el algoritmo exacto propuesto.

Abstract

Algorithms for the Cyclic Bandwidth Problem for Graphs

by

Hillel Romero Monsivais
Master of Science from the Information Technology Laboratory, CINVESTAV-Tamaulipas
Research Center for Advanced Study from the National Polytechnic Institute, 2012
Dr. Eduardo Arturo Rodríguez Tello, Advisor

The *Cyclic Bandwidth* problem is a graph embedding problem. It was first stated by Leung *et al.* in 1984 in relation with the design of a ring interconnection network. Their aim was to find an arrangement on a cycle for a set V of computers with a known communication pattern, given by the graph $G(V, E)$, in such a way that every message sent can arrive at its destination in at most k steps. The Cyclic Bandwidth problem arises also in other important application areas like VLSI designs, data structure representations, code design and interconnection networks for parallel computer systems. As is the case with many graph embedding problems, it was demonstrated that finding the cyclic bandwidth is NP-complete for general graphs. At present there are studies about their properties, theoretical limits and comparatives with other labeling problems. However, it is necessary to create efective algorithms that solve this problem for general graphs.

In this thesis work an exact Branch and Bound and a metaheuristic Tabú Search algorithms for the CB problem are introduced. Its key components were carefully devised after an in-depth analysis of the given problem. The practical effectiveness of these algorithms are assessed through extensive experimentation over 78 graphs, of four different kinds. The parameter tuning process, for the metaheuristic algorithm was performed by employing a technique know as combinatorial intection testing that uses covering arrays. The results show that the proposed exact algorithm attains the lower bounds for graphs with order $n \leq 35$ expending a reasonable computational time. This approach was able to solve exactly the 53.84 % of all instances using a maximum of 36 CPU hours. The metaheuristic algorithm was able to solve even larger graphs and in less time, improving the bounds of 30 instances with respect to those produced by the proposed exact algorithm.

Nomenclatura

TG	Teoría de Grafos
PEG	Problemas de Etiquetado de Grafos
POC	Problemas de Optimización Combinatoria
MABC	Minimización de Ancho de Banda Cíclico
ABC	Ancho de Banda Cíclico
MAB	Minimización de Ancho de Banda
AB	Ancho de Banda
RP	Ramificación y Poda
BT	Búsqueda Tabú
MCA	Mixed Level Covering Array

1

Introducción

En los últimos años se han realizado estudios sobre la estrecha relación que existe entre la Teoría de Grafos (TG) y las ciencias computacionales. Dichos estudios han ayudado de manera significativa al desarrollo de ambas disciplinas, y se ha llegado a concluir que muchos de los problemas de la TG pueden ser resueltos usando técnicas computacionales. Los problemas de etiquetado de grafos son un claro ejemplo de ello.

En esta investigación se trabaja con un problema de etiquetado de grafos conocido como Minimización del Ancho de Banda Cíclico (MABC), el cual puede ser subclasificado como un problema de grafos embebidos. Aquí se presenta un algoritmo de tipo exacto y un algoritmo metaheurístico los cuales trabajan sobre grafos generales.

En este capítulo se mencionan de algunos aspectos que anteceden al problema de MABC, la motivación que impulsó a esta investigación y los objetivos que se plantearon para ella.

1.1 Antecedentes

En las ciencias computacionales, específicamente en el área de TG, el etiquetado de un grafo contempla a un conjunto de *unidades* representadas por nodos, a un conjunto de *etiquetas* que son los posibles nombres de los nodos y a un *modelo compatible*, conformado por los conjuntos ordenados de unidades y de pares unidad-etiqueta. Este modelo compatible es conocido como *modelo del mundo*. El problema reside en encontrar una etiqueta para cada unidad de manera que el conjunto resultante de pares unidad-etiqueta sea consistente con las restricciones del modelo del mundo (Haralick y Shapiro, 1979).

Los Problemas de Etiquetado de Grafos (PEG) involucran a la vez a subconjuntos de problemas específicos presentes en la TG, como lo son: grafos isomórficos (Ullmann, 1976), coloreo de grafos (Harary, 1969), empaquetamiento de grafos (Deutsch, 1966), grafos embebidos (Chinn *et al.*, 1982), entre otros.

El problema de un grafo embebido consiste en que: a partir de un grafo anfitrión H y un grafo huésped G, ambos de orden n, cada nodo de G debe ser mapeando biyectivamente a H para que tome la forma del anfitrión y quede embebido. De esta manera, el grafo anfitrión determinará las distancias entre los nodos del grafo huésped sin alterar su adyacencia. En su forma más simple se puede considerar a un grafo de tipo camino P_n como anfitrión, lo que da origen al problema de *Minimización del Ancho de Banda* (MAB) (Chung, 1988). También se puede considerar como grafo anfitrión a alguno de tipo ciclo C_n o malla $P_n \times P_n$. El problema de Minimización del Ancho de Banda Cíclico (MABC) consiste en embeber un grafo huésped dentro de un ciclo C_n, de manera que sea minimizada la distancia máxima entre las etiquetas de los nodos adyacentes (Leung *et al.*, 1984).

Es bien conocido que los PEG pueden ser formulados como Problemas de Optimización Combinatoria (POC), los cuales se enfocan en encontrar la mejor solución existente mediante la minimización o maximización de cierta función de costo (función objetivo). Como las variables que

intervienen en estos problemas son discretas, el dominio está restringido a un conjunto finito de soluciones (Nemhauser y Wolsey, 1988). Sin embargo, esto no significa que el conjunto sea pequeño, por lo que se han desarrollado técnicas especiales para tratar este tipo de problemas. En los últimos años ha habido un crecimiento considerable en el desarrollo de estas técnicas sobre todo después de que muchas situaciones prácticas de la vida cotidiana fueron definidas como grafos. Los primeros métodos reportados en la literatura para resolver estos problemas datan de los años 1960, y a partir de ese momento muchos algoritmos exactos y metaheurísticos se han desarrollado (Gallian, 1998).

De acuerdo con la definición de Woeginger (2003), los algoritmos exactos son aquellos que siempre encuentran una solución óptima gracias a que implementan técnicas que exploran de manera sistemática todo el espacio de búsqueda. Debido a esto, uno de sus objetivos primordiales es la reducción en lo posible del tiempo de ejecución. Por otro lado, existen métodos en donde se buscan soluciones de la misma calidad pero usando un tiempo aún menor, explorando solamente ciertas áreas potenciales del espacio de búsqueda. Estos son denominados algoritmos aproximados o metaheurísticos y son definidos por Blum y Roli (2003) como un proceso de generación iterativa que guía una heurística subordinada. Este combina de forma inteligente distintos conceptos para la exploración del espacio de búsqueda, y utilizan estrategias de aprendizaje para estructurar la información con el fin de encontrar de manera eficiente soluciones cercanas a la óptima.

En esta tesis se proponen 1) un algoritmo exacto basado en la técnica de Ramificación y Poda (RP) y 2) un algoritmo metaheurístico que implementa Búsqueda Tabú (BT) para dar solución al problema de MABC en grafos generales.

1.2 Motivación

El estudio del problema de MABC ha mostrado ser una área de oportunidad muy grande, pues a pesar de su importancia tanto teórica como práctica, se ha avanzado muy poco en torno a su solución.

Para el problema de minimización del ancho de banda (similar a nuestro problema pero utilizando un grafo tipo camino como anfitrión), se han desarrollado una amplia gama de algoritmos que incluyen algunos de tipo exacto como el de programación dinámica propuesto por Gurari y Sudborough (1984), o el planteado por Del Corso y Manzini (1999) quienes usan el método de ramificación y poda con una estrategia de búsqueda primero en profundidad. También, se han desarrollado algoritmos aproximados como el de búsqueda tabú (Martí *et al.*, 2001), algoritmos genéticos (Lim *et al.*, 2003) o recocido simulado (Rodriguez-Tello *et al.*, 2008). Estos algoritmos han permitido resolver grafos de tipo malla (Pintea *et al.*, 2010) y árboles (Gupta, 2000) por nombrar algunos. Sin embargo, para el problema de MABC solo se han aportado estudios teóricos que abarcan sus propiedades básicas y su complejidad computacional (Lin, 1994). Además, se han establecido ciertos límites teóricos (Lin, 1997) y se han realizado comparativas con otros PEG (Jinjiang y Sanming, 1995). Por lo tanto, existe la necesidad de desarrollar algoritmos que permitan resolver eficientemente este problema[1].

Otro aspecto importante que impulsa a este trabajo es que el conocimiento adquirido podría ser aplicado para resolver otros problemas de etiquetado de grafos, como el de *maximización del antibandwidth cíclico* (Leung *et al.*, 1984).

Por último, existe una serie de aplicaciones prácticas que podrían resolverse encontrando soluciones de buena calidad para el problema de MABC en grafos generales. Entre dichas aplicaciones está la de encontrar la configuración de un conjunto de computadoras sobre una red tipo anillo para que el envío de mensajes entre ellas se realize en máximo k pasos. También, la solución podría aplicarse en el diseño de tarjetas cilíndricas que incluyen circuitos integrados VLSI (Chung, 1988) o en el mejor manejo de estructuras de datos circulares (Lam *et al.*, 2002).

[1]En este caso un algoritmo eficiente es aquel que entrega soluciones de buena calidad (subóptimas) en un tiempo razonable.

1.3 Objetivos de la tesis

1.3.1 Objetivo general

Para formalizar esta investigación se estableció un objetivo general con el fin de enfocar los esfuerzos y concluir con satisfacción el trabajo de tesis. El objetivo fue resolver el problema de MABC para grafos generales de manera eficiente mediante la implementación de dos enfoques: un algoritmo exacto y un algoritmo aproximado de tipo metaheurístico.

1.3.2 Objetivos especificos

Los objetivos particulares para este trabajo de investigación son:

- Estudiar técnicas reportadas en la literatura que dan solución a problemas de etiquetado de grafos.
- Analizar las características del problema de MABC.
- Desarrollar un algoritmo exacto que encuentre la solución óptima para el problema de MABC en grafos menores a $n < 40$.
- Implementar un algoritmo metaheurístico para resolver, el caso general, del problema de MABC en grafos de orden $n > 40$.
- Llevar acabo una etapa de sintonización de parámetros para el algoritmo metaheurístico con la finalidad de mejorar su desempeño y obtener resultados de mejor calidad.

1.4 Organización de la tesis

El resto de la tesis está formada por cuatro capítulos más, los cuales abordan toda la investigación e implementaciones realizadas, así como los resulados obtenidos. A continuación se presenta una breve

descripción del contenido de cada uno de ellos.

- **Capítulo 2.** Presenta algunos conceptos básicos para introducir de manera formal el problema de MABC. También, detalla su complejidad computacional y muestra una reseña del trabajo previamente reportado en la literatura, la cual incluye límites teóricos establecidos para el problema bajo estudio.

- **Capítulo 3.** Describe los enfoques de solución propuestos para el problema de MABC. Para ambos casos se detallan sus componentes de manera específica y también se explica cómo trabajan de manera conjunta dentro del algoritmo.

- **Capítulo 4.** Expone la etapa de sintonización de parámetros del algoritmo metaheurístico, así como la etapa de experimentación final de esta investigación. Para esto se presentan las instancias de prueba utilizadas y los criterios de comparación considerados. Además, se muestra un análisis de los resultados obtenidos por ambos enfoques.

- **Capítulo 5.** Tiene como fin presentar las conclusiones a las que se llegó con este trabajo de investigación. También, se abordan algunas posibilidades para trabajo futuro

2

Estado del arte

En este capítulo se presentan algunos conceptos básicos los cuales ayudan a introducir de manera formal el problema de MABC. Se muestra mediante un ejemplo el problema de embeber un grafo en un ciclo. Se describe la complejidad computacional que tiene el encontrar el ancho de banda cíclico óptimo para un grafo general. Por último, se resume brevemente el trabajo reportado en la literatura incluyendo algunos límites teóricos.

2.1 Definición formal del problema de MABC

La Teoría de Grafos (TG) tuvo su origen en 1736 con el trabajo de Leonhard Euler, en donde se expuso el problema de los puentes de Königsberg. El problema plantea a una ciudad que está dividida en cuatro regiones (representadas por los nodos) a causa de un río, y dichos lugares están conectados por siete puentes (representados por los arcos) como se puede ver en la Figura 2.1. El objetivo es encontrar una ruta para recorrer a pie toda la ciudad, pasando solamente una vez por cada puente y terminando en el mismo lugar. Esto dio pie a la primera noción del grafo.

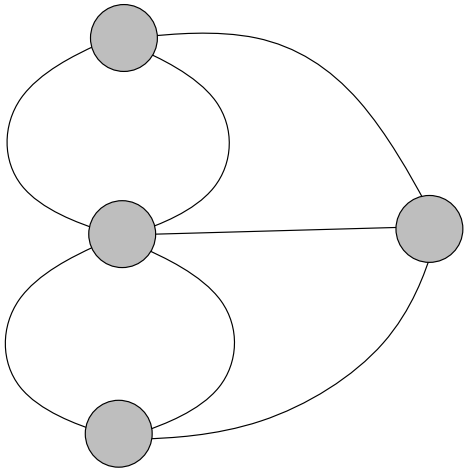


Figura 2.1: Grafo que presenta el problema de los puentes de Königsberg.

Haciendo un análisis detallado de la TG, es notorio que existe una gran diversidad de problemas específicos entre los que se encuentran los PEG. Un subconjunto particular de estos, es el de embeber un grafo dentro de otro. El problema de MABC estudiado en esta tesis, pertenece justamente a esta categoría.

Antes de definir formalmente el problema de MABC, se presentan algunos conceptos que nos ayudarán a entender claramente el problema bajo estudio.

Definición 2.1 (Etiquetado de un grafo.) *Un etiquetado de un grafo $G = (V, E)$ de orden n es una función biyectiva de la forma $\varphi = V \rightarrow \{1, 2, 3, .., n\}$, que a cada nodo del grafo asocia un número entero distinto entre 1 y n. En lo sucesivo se utilizará la notación $\varphi(u)$ para designar la etiqueta de un nodo u.*

Definición 2.2 (Problema de grafos embebidos.) *Sea $G = (V, E)$ un grafo huesped finito no dirigido, donde V ($|V| = n$) define el conjunto de nodos y $E \subseteq V \times V = \{\{u, v\} \mid u, v \in V\}$ el conjunto de arcos ($|E| = m$). Sea $H = (V', E')$ un grafo anfitrión con las mismas características que el grafo huesped. Embeber G en H consiste en una función biyectiva $\varphi = V \rightarrow V'$ (un etiquetado) que a cada nodo del grafo G asocia un nodo distinto del grafo H, y en donde $|\varphi(u) - \varphi(v)|$ denota la distancia entre los nodos $u, v \in V'$.*

De la forma más simple se puede considerar a un grafo camino P_n como anfitrión H, dando lugar al problema de Minimización del Ancho de Banda (MAB).

Definición 2.3 (Ancho de banda.) *Sea $G = (V, E)$ un grafo huésped y $P_n = (V', E')$ un grafo anfitrión tipo camino, ambos no dirigidos y de orden n; y sea φ una función biyectiva de la forma $\varphi = V \to V'$. El ancho de banda AB de G es la distancia más grande que existe en P_n para un par de nodos adyacentes. Formalmente puede definirse de la siguiente manera:*

$$B_P(G, \varphi) = \max_{\{u,v\} \in E} \{|\varphi(u) - \varphi(v)|\} \tag{2.1}$$

Definición 2.4 (Problema de MAB.) *Sea $G = (V, E)$ un grafo huésped y $P_n = (V', E')$ un grafo anfitrión tipo camino, ambos no dirigidos y de orden n. El problema consiste en encontrar un etiquetado φ^* para el cual el ancho de banda, denotado por $B_P(G, \varphi^*)$, sea mínimo:*

$$B_P(G, \varphi^*) = \min\{B_P(G, \varphi) \mid \varphi \in \mathscr{L}\}, \tag{2.2}$$

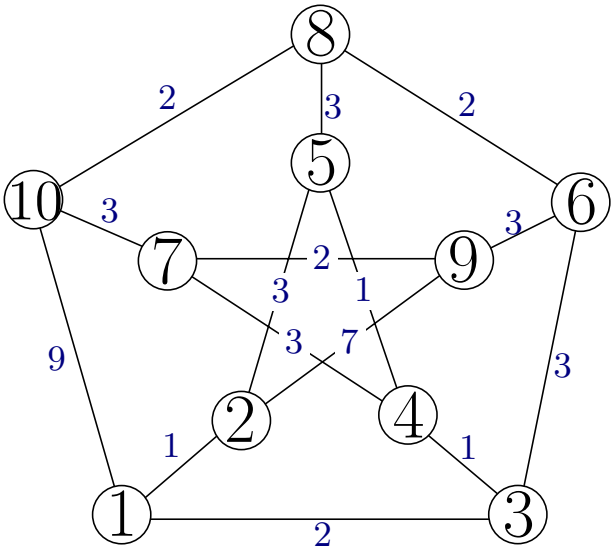
donde $\mathscr{L}$ es el conjunto de todos los etiquetados posibles.

Este problema es equivalente a encontrar un arreglo lineal de los nodos de un grafo, de manera que la separación máxima que exista entre dos nodos conectados sea mínima. La separación entre dos nodos está determinada por el número de nodos entre ellos.

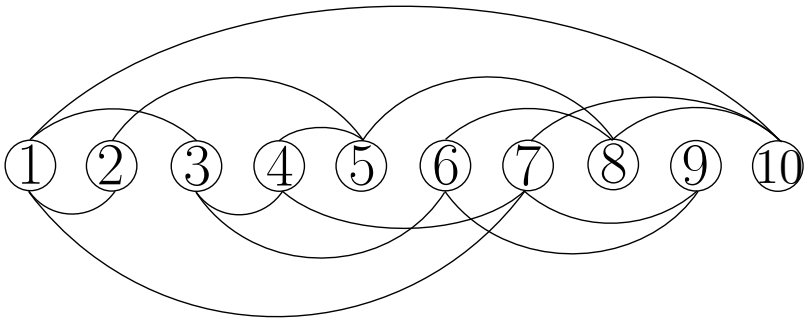
En la Figura 2.2(a) se muestra el grafo Petersen[1] el cual está formado por $n = 10$ nodos etiquetados con un número en su interior, y en donde se denota el valor del ancho de banda de cada arco[2]. La Figura 2.2(b) muestra como cada nodo de dicho grafo queda transpuesto en un grafo camino P_n, de manera que queda embebido generando con este etiquetado un ancho de banda $B_P(G, \varphi) = 9$.

[1] Grafo ampliamente utilizando en la literatura como instancia de prueba, introducido por Julius Peter Christian Petersen en 1898.

[2] En lo sucesivo se utilizará esta representación para indicar el etiquetado de un nodo y el valor de un arco.



(a) Grafo huésped. $B_P(G) = 9$



(b) Grafo anfitrión (H) tipo camino

Figura 2.2: Ejemplo de un grafo embebido dentro de un anfitrión tipo camino.

Otro tipo de grafo a considerar como anfitrión H podrían ser los ciclos C_n. Esto da origen al problema de Minimización del Ancho de Banda Cíclico (MABC), el cual es objeto de estudio en esta tesis.

Definición 2.5 (Ancho de banda cíclico.) *Sea $G = (V, E)$ un grafo huésped y $C_n = (V', E')$ un grafo anfitrión tipo ciclo, ambos no dirigidos y de orden n; y sea φ una función biyectiva de la forma $\varphi = V \to V'$. El ancho de banda cíclico ABC de G es la distancia más grande medida en C_n entre cada par de nodos adyacentes. Formalmente puede definirse de la siguiente manera:*

$$B_C(G,\varphi) = \max_{\{u,v\}\in E}\{|\varphi(u)-\varphi(v)|_n\}\,, \tag{2.3}$$

donde $|x|_n = \min\{|x|, n-|x|\}$ *para* $1 < |x| < n-1$*, y es nombrada distancia cíclica. Para un nodo* u *su ancho de banda cíclico estará determinado por el arco adyacente a la con mayor distancia cíclica.*

$$B_C(u,\varphi) = \max_{\{u,v\}\in E}\{|\varphi(u)-\varphi(v)|_n \;\big|\; v\in\mathcal{N}(u)\}\,, \tag{2.4}$$

donde $\mathcal{N}(u)$ *son los nodos adyacentes a* u *en el grafo* G*.*

Definición 2.6 (Problema de MABC.) *Sea* $G=(V,E)$ *un grafo huésped y* $C_n=(V',E')$ *un grafo anfitrión tipo ciclo, ambos no dirigidos y de orden* n*; y sea* φ *una función biyectiva de la forma* $\varphi = V \to V'$*. El problema de MABC consiste en encontrar un etiquetado* φ^* *para el cual el costo del* ABC*, sea mínimo:*

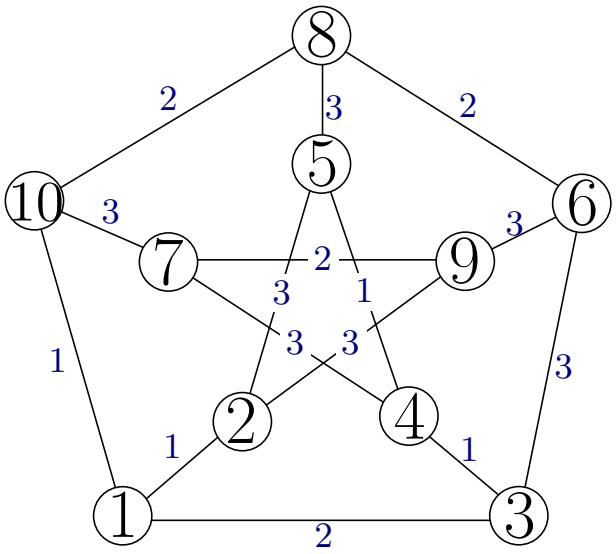
$$B_C(G,\varphi^*) = \min\{B_C(G,\varphi) \;\big|\; \varphi\in\mathscr{L}\}\,, \tag{2.5}$$

donde $\mathscr{L}$ *es el conjunto de todos los etiquetados posibles.*

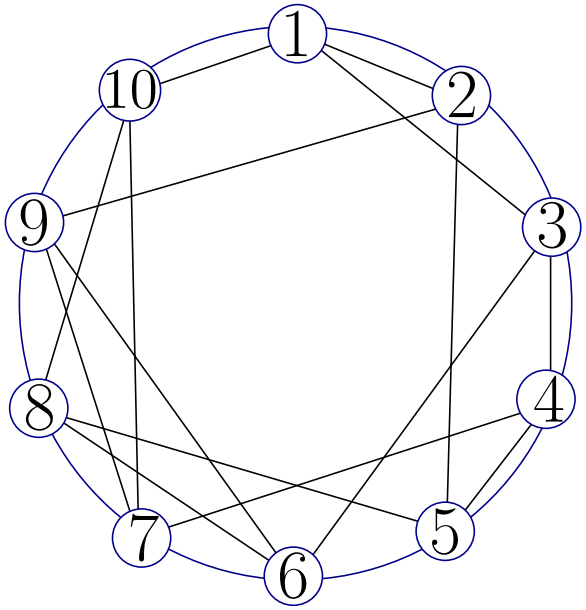
El problema de MABC fue introducido por Leung *et al.* (1984) como una variante del problema de MAB por las características que ambos presentan, demostrando que de igual forma pertenece a la clase NP-completo. Es importante mencionar que el tamaño de su espacio de búsqueda es $\frac{(n-1)!}{2}$ para un grafo de n nodos, y ya que el etiquetado de un grafo (solución φ) puede ser visto como una permutación de números, es evidente que el problema es altamente combinatorio.

En la Figura 2.3 se puede apreciar cómo queda embebido un grafo huesped dentro de un ciclo. La Figura 2.3(a) muestra nuevamente el grafo Petersen y la Figura 2.3(b) muestra como queda embebido dentro del ciclo, generando con ese etiquetado φ un ancho de banda cíclico $B_C(G,\varphi)=3$.

Una situación práctica en donde se encuentra presente este problema es en la interconexión de



(a) Grafo huésped. $B_C(G, \varphi) = 3$



(b) Grafo ciclo anfitrión (H)

Figura 2.3: Ejemplo de un grafo embebido dentro de un anfitrión tipo ciclo.

redes tipo anillo, en donde un conjunto de computadoras V muestra un patrón de comunicación establecido por el grafo $G(V, E)$, y en donde $\{u, v\} \in E$ establece el enlace que existe entre la computadora u y la v. El objetivo es encontrar un arreglo de estas computadoras, dentro del anillo, de manera que cada mensaje enviado llegue a su destino en máximo k pasos.

2.2 Complejidad computacional

Existen dos áreas importantes que han impulsado el desarrollo de las ciencias computacionales: la teoría de la computación y el diseño de algoritmos. En la teoría de la computación, la complejidad computacional estudia los recursos requeridos para resolver un problema respecto al tiempo[3] y al espacio[4]; de esta manera los problemas puede ser clasificados según su complejidad (Dasgupta *et al.*, 2008).

A la clase P la conforman aquellos problemas de decisión que pueden ser resueltos en tiempo polinomial por una máquina determinista, es decir, aquellos para los cuales existe a lo sumo una posible ejecución de cada par *estado-símbolo*. La clase NP contiene aquellos problemas cuya solución puede ser verificada en tiempo polinimial pero no pueden ser resueltos por una máquina determinista en ese tiempo. Dentro de esta clasificación se encuentra la clase NP-completo, que es el conjunto de problemas más difíciles de ese grupo (Cook, 2000).

Para demostrar que un problema pertenece a la clase NP-completo el teorema de Cook (1971) dice que es necesario que este pertenezca a la clase NP, y que todo problema de esta clase pueda ser reducido al problema en cuestión utilizando una transformación polinómica. Algunos problemas de este tipo ampliamente conocidos son *el problema del agente viajero* (Biggs *et al.*, 1976), *el problema de satisfactibilidad* (Cook, 1971) y *el problema del clique* (Luce y Perry, 1949); los cuales hasta el momento permanecen sin ser resueltos en tiempo polinomial y se sospecha fuertemente que tal solución no existe.

[3] Una aproximación al número y tipo de pasos de ejecución de un algoritmo para resolver un problema.
[4] Una aproximación a la cantidad de memoria utilizada para resolver el problema.

Leung *et al.* (1984) demostraron que el problema de MABC pertenece a la clase NP haciendo la reducción del problema de *la 3-partición* a este, para el caso donde la distancia cíclica del grafo es como máximo $k = 2$.

El problema de la 3-partición es un problema NP-completo (Garey y Johnson, 1989), y se define formalmente de la siguiente manera: Dado un entero positivo b y un conjunto de $3m$ enteros positivos $A = \{a_1, a_2, ..., a_{3m}\}$ tal que $\sum_{j=1}^{3m} a_j = mb$ y que $b/4 < a_j < b/2$ para cada $1 \leq j \leq 3m$, ¿Existe una partición de A en m conjuntos disjuntos $A_1, A_2, ..., A_m$ tal que $\sum_{a_j \in A_i} a_j = b$ para cada $1 \leq i \leq m$? (Note que las limitaciones en a_j implican que cada A_i debe contar con exactamente tres elementos de A). Ya que el problema de la 3-partición en un sentido estricto es NP-completo (Garey y Johnson, 1989), se puede asumir que el tamaño de los enteros en A está acotado por una función polinomial de la cardinalidad de A.

Dada una instancia particular $A = \{a_1, a_2, ..., a_{3m}\}$ y el entero positivo b, para el problema de la 3-partición, es posible construir un grafo $G = (V, E)$ de la siguiente manera: (1) un ciclo con $m(b+1)+1$ nodos, (2) subgrafos colocados en lugares apropiados del ciclo y (3) $3m$ cadenas tal que la j-ésima cadena tenga exactamente a_j nodos. Este grafo estará formado por $n = 2m(b + 1) + 2$ nodos, lo que permite su construcción en tiempo polinomial.

Leung *et al.* (1984) demostraron que A tiene una partición $A_1, A_2, ..., A_m$ si y solo si G tiene una disposición (*layout*) circular con ancho de banda cíclico no mayor a $k = 2$. Los autores finalizan indicando que es fácil modificar este procedimiento para demostrar que el problema de MABC es NP-completo para el caso donde la distancia cíclica del grafo es $k > 2$.

De igual forma Lin (1994) realizó una reducción del problema de MAB al de MABC para demostrar la NP-completez de este último, obviando que ya pertenecía a la clase NP.

2.3 Métodos de resolución del problema de MABC

La mayor parte del trabajo realizado sobre el problema de MABC está enfocado en estudios teóricos, los cuales han aportado información sobre las características del problema y su similitud con otros PEG. Además, se han establecido límites teóricos a partir de ciertas propiedades para grafos muy específicos.

Lin (1994) haciendo uso del trabajo previamente realizado sobre el problema de MAB (Bhatt y Leighton, 1984), propone una fórmula que calcula el límite inferior de árboles en general en términos de su diámetro[5]. En su trabajo plantea que los arcos que pertenecen al grafo anfitrión se pueden clasificar en *arcos excedidos* y *arcos normales* permitiendo la existencia de *huecos* en los grafos. Básicamente con esta propiedad demuestra que su ecuación da resultados correctos, permitiendo así mismo obtener el límite inferior de grafos oruga y estrellas de dos dimensiones. En su siguiente trabajo, Lin (1997) hizo un estudio sobre los grafos anfitriones caminos P_n y ciclos C_n, estableciendo los requisitos mínimos que debe cumplir un grafo G para que $B_C(G,\varphi) = B_P(G,\varphi)$ y $B_C(G,\varphi) = \frac{1}{2}B_P(G,\varphi)$. Esto significa que, si tenemos una aproximación μ para el ancho de banda cíclico, entonces contamos con una aproximación 2μ para el ancho de banda. Considerando este aspecto, las propiedades de los arcos y el diámetro del grafo nuevamente establece límites teóricos para los grafos planares, grafos circularmente simétricos y árboles en general.

Otro trabajo relevante fue el realizado por Hromkovič *et al.* (1992), en donde comprueban de manera matemática que todos los grafos que cumplan con ciertas condiciones pueden generar el mismo valor de ancho de banda como de ancho banda cíclico. Las condiciones que establecieron están presentes en los hipercubos, árboles de malla, árboles, mallas y pirámides. Ese mismo estudio les ayudó a establecer límites inferiores para árboles completos, hipercubos de n-dimensiones y mallas de 2 y 3 dimensiones.

[5]El diámetro d de un grafo es la distancia más grande que existe entre cualquier par de nodos. La distancia entre ellos está determinada por el recorrido con el menor número de arcos.

Tiempo después Zhou (2000) propuso un método sistemático basado en una idea de Chvátal (1970) que permitió calcular límites teóricos de un grafo para el problema de MABC y de MAB a partir de los grados y de la excentricidad[6] que este presenta. La idea consiste en que gracias a que estos parámetros poseen una propiedad de monotonía, es posible relajar la condición que embebe al grafo G en el grafo anfitrión permitiendo el cálculo de una cota inferior.

2.3.1 Límites teóricos para grafos especiales

Existe una cota superior obvia establecida por Lin (1994), la cual puede aplicarse a cualquier tipo de grafo, $B_C(G, \varphi) = \left\lfloor \frac{n}{2} \right\rfloor$. Sin embargo, en la literatura también se encuentran reportados límites inferiores para ciertos tipos particulares de grafos. A continuación, se detallan los más importantes.

2.3.1.1. Grafos planares

Un grafo planar es aquel que puede ser incrustado en un plano sin que sus arcos se intersecten, es decir, que puede ser dibujado sin que ninguno de sus arcos se cruce. El cálculo del límite inferior para este tipo de grafo particular es introducido por Lin (1997), estableciendo la siguiente ecuación:

$$B_C(G, \varphi) = \left\lceil \frac{m}{n} \right\rceil, \tag{2.6}$$

donde m es el número de arcos en el grafo.

[6] La excentricidad de un nodo es la distancia más grande que existe, considerando siempre el camino más corto, entre ese nodo y todos los demás presentes en el grafo.

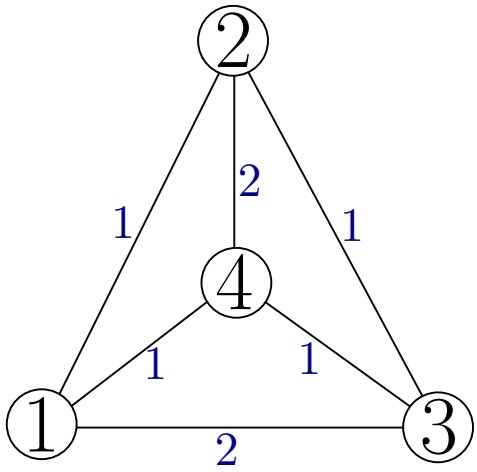


Figura 2.4: Grafo planar. $B_C(G, \varphi) = 2$.

Un ejemplo de esto se puede observar en la Figura 2.4 la cual muestra el etiquetado óptimo que cumple con la cota $B_C(G, \varphi) = \left\lceil \frac{6}{4} \right\rceil = 2$.

2.3.1.2. Grafos oruga

Un grafo oruga es aquel árbol que tiene todos sus nodos a distancia unitaria respecto a su columna central, de manera que si eliminamos todos sus nodos hoja tendremos un grafo camino. Esto se puede definir formalmente de la siguiente manera: Sea $G_1, G_2, G_3, ..., G_s$ una secuencia de grafos estrella con diámetro $d = 2$ (columna central del grafo), donde $|V_i| = n_i \geq 3$ denota el orden de G_i. Cuando $|i - j| = 1$, $G_i \cap G_j$ contiene un arco en común; cuando $|i - j| = 2$, $G_i \cap G_j$ contiene un nodo en común y cuando $|i - j| > 2$, $G_i \cap G_j = \varnothing$. Al árbol $G = G_1 \cup G_2 \cup G_3 \cup ... \cup G_s$ se le conoce como grafo *oruga*. La Ecuación 2.7 (Lin, 1994) permite calcular su $B_C(G, \varphi)$ mínimo.

$$B_C(G, \varphi) = \max_{1 \leq i \leq j \leq s} \left\lceil \frac{n_{ij} - 1}{j - i + 2} \right\rceil, \tag{2.7}$$

donde $G_{ij} = G_i \cup G_{i+1} \cup ... \cup G_{j-1} \cup G_j$ y $n_{ij} = |V_{ij}|$. En la Figura 2.5 se muestra un ejemplo de este de grafo con su etiquetado óptimo.

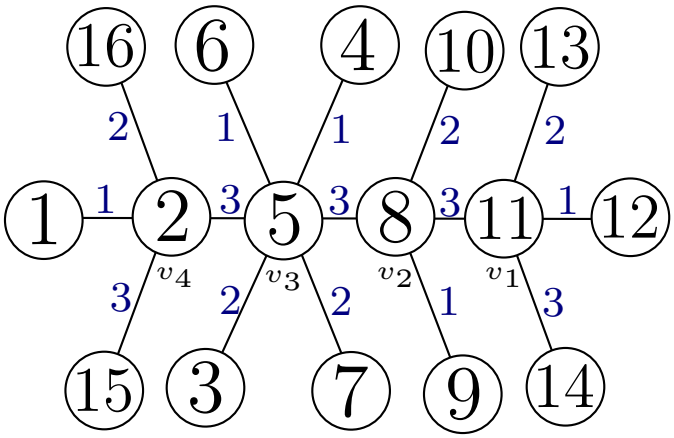


Figura 2.5: Grafo oruga. $B_C(G, \varphi) = 3$.

El etiquetado que se muestra en el grafo genera el valor máximo para la Ecuación 2.7 en $i = 1$ y $j = 2$. Por lo tanto, el límite teórico para ese grafo es de $B_C(G, \varphi) = \left\lceil \frac{8-1}{2-1+2} \right\rceil = 3$. Los cálculos necesarios para llegar a este resultado se pueden observar en la Tabla 2.1.

	j			
i	*1*	*2*	*3*	*4*
1	5	—	—	—
2	8	5	—	—
3	13	10	7	—
4	16	13	10	5

(a) Valor de n_{ij} para cada subgrafo G_{ij}

	j			
i	*1*	*2*	*3*	*4*
1	2	—	—	—
2	**3**	2	—	—
3	**3**	**3**	**3**	—
4	**3**	**3**	**3**	2

(b) Cálculo de la ecuación $\left\lceil \frac{n_{ij}-1}{j-i+2} \right\rceil$ para cada subgrafo G_{ij}

Tabla 2.1: Información necesaria para calcular el límite teórico del grafo oruga de la Figura 2.5.

2.3.1.3. Estrellas de dos capas

En una de sus investigaciones Lin (1989) describe un grafo especial tipo árbol el cual nombra *estrella de dos capas*. Este se puede definir formalmente de la siguiente manera: Sea G un árbol con un nodo raíz v_0 (nivel cero), donde $v_1, v_2, v_3, ..., v_s$ son los nodos hijos de v_0 (nivel uno), y donde $v_{i1}, v_{i2}, v_{i3}, ..., v_{ik_i}$ son los hijos de v_i (nivel dos). Y sean $G_0, G_1, G_2, ..., G_s$ una secuencia de

subgrafos estrella donde $|V_i| = n_i \geq 3$ y $n_0 = s + 1$. Para $1 \leq i \leq s,\ V_0 \cap V_i = \{v_0, v_i\}$; para $1 \leq i < j \leq s$, $V_i \cap V_j = \{v_0\}$, donde v_i es el centro de G_i. Al árbol $G = G_0 \cup G_1 \cup G_2 \cup ... \cup G_s$ se le llama *estrella de dos capas* (ver Figura 2.6).

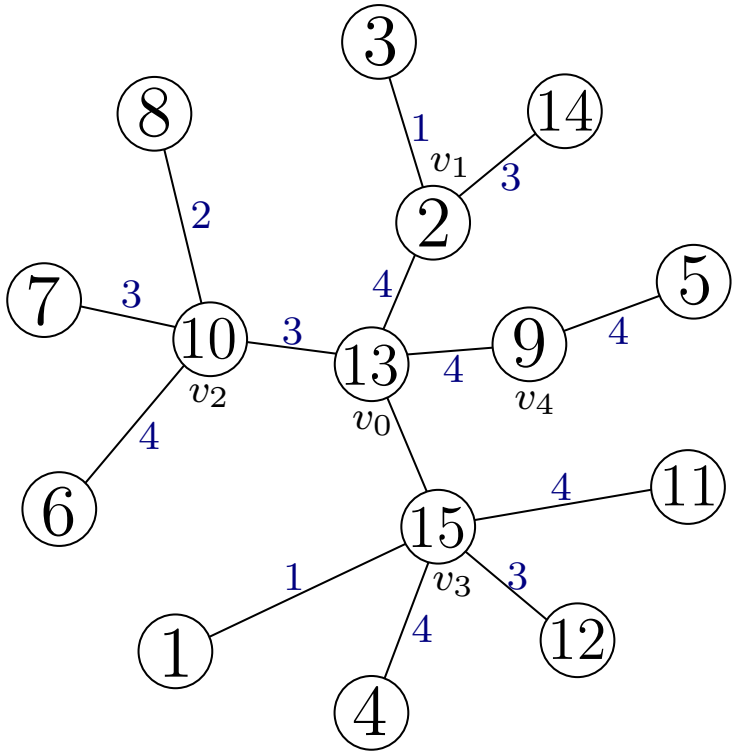


Figura 2.6: Grafo estrella de dos dimensiones. $B_C(G, \varphi) = 4$.

Años más tarde el mismo establece la Ecuación 2.8 para poder calcular su límite inferior (Lin, 1994).

$$B_C(G, \varphi) = \text{máx} \left\{ \max_{0 \leq i \leq s} \left\lceil \frac{n_i - 1}{2} \right\rceil, \max_{1 \leq i \leq s} \left\lceil \frac{n_{0i} - 1}{3} \right\rceil, \left\lceil \frac{n - 1}{4} \right\rceil \right\}, \tag{2.8}$$

donde $1 \leq i \leq s$ y $G_{0i} = G_0 \cup G_i, n_{0i} = |V_{0i}|$. En la Figura 2.6 se puede observar el etiquetado óptimo del grafo, el cual produce el valor máximo para la Ecuación 2.8. Para la primera parte ($\text{máx}_{0 \leq i \leq s} \left\lceil \frac{n_i - 1}{2} \right\rceil$) el valor máximo se obtiene en $i = 3$, y para la segunda parte ($\text{máx}_{1 \leq i \leq s} \left\lceil \frac{n_{0i} - 1}{3} \right\rceil$) en $i = 2$. Por lo tanto, la ecuación queda de la siguiente manera: $B_C(G, \varphi) = \text{máx} \left\{ \left\lceil \frac{6-1}{2} \right\rceil, \left\lceil \frac{8-1}{3} \right\rceil, \left\lceil \frac{15-1}{4} \right\rceil \right\} = \text{máx} \left\{ 3, 3, 4 \right\} = 4$. Los cálculos completos se pueden observar en la Tabla 2.2.

	i				
	0	*1*	*2*	*3*	*4*
n_i	5	4	5	6	3
$\left\lceil \frac{n_i-1}{2} \right\rceil$	2	2	2	**3**	1
n_{0i}		7	8	9	6
$\left\lceil \frac{n_{0i}-1}{3} \right\rceil$		2	**3**	**3**	2

Tabla 2.2: Información necesaria para calcular el límite teórico del grafo de la Figura 2.6.

2.3.1.4. Árboles completos

Un árbol s-ário completo de h-niveles, es un grafo tipo árbol que cuenta con h niveles, y en donde todo sus nodos tienen exactamente s hojas. La Fórmula 2.9 establecida por Hromkovič *et al.* (1992) ayuda a calcular el límite inferior de este tipo de grafos.

$$B_C(G,\varphi) = \left\lceil \frac{h(h^{s-1}-1)}{2(s-1)(h-1)} \right\rceil \tag{2.9}$$

Un árbol 2-ário (binario) completo de $h = 3$ niveles se muestra en la Figura 2.7. El etiquedo que presenta este grafo genera un ancho de banda cíclico similar al que se obtiene con la ecuación anterior, $B_C(G,\varphi) = \left\lceil \frac{3(3^{2-1}-1)}{2(2-1)(3-1)} \right\rceil = 2$.

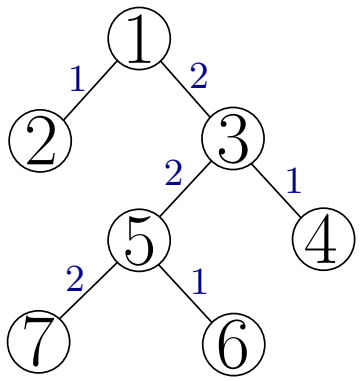


Figura 2.7: Árbol 2-ário (binario) completo de $h = 3$ niveles y $B_C(G,\varphi) = 2$.

2.3.1.5. Hipercubos

El límite inferior para los hipercubos fue establecido por Hromkovič *et al.* (1992). La Ecuación 2.10 ayuda a calcular esta cota considerando el número de dimensiones f del grafo.

$$\sum_{k=0}^{f-1} \binom{k}{\lfloor k/2 \rfloor} \tag{2.10}$$

En la Figura 2.8 se muestra un hipercubo de 2 dimensiones con su etiqueado óptimo, el cual cumple con la formula $B_C(G,\varphi) = \sum_{k=0}^{1} \binom{k}{\lfloor k/2 \rfloor} = 2$.

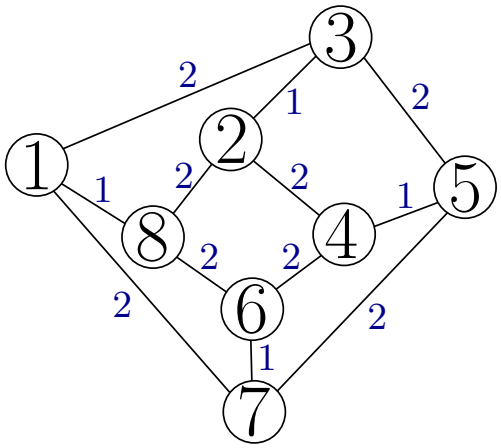


Figura 2.8: Hipercubo, con $f = 2$ y $B_C(G,\varphi) = 2$.

2.3.1.6. Mallas de 2 y 3 dimensiones

Las mallas de 2 y 3 dimensiones también cuentan con un límite inferior teórico, el cual fue establecido por Hromkovič *et al.* (1992). La Ecuación 2.11 es utilizada para las mallas de 2 dimensiones, donde o es el largo y p es el ancho del grafo. Para las mallas de 3 dimensiones se usa la Ecuación 2.12.

$$B_C(G,\varphi) = \text{mín}\{o,p\} \tag{2.11}$$

$$B_C(G,\varphi) = \left\lfloor \frac{3o^2}{4} + \frac{o}{2} \right\rfloor \tag{2.12}$$

Un ejemplo de como la fórmula para las mallas de 2 dimensiones establece el límite inferior se observa en la Figura 2.9. Aquí se muestra una malla de $o = 3$ y $p = 2$ con su etiquetado óptimo, el cual cumple con $B_C(G,\varphi) = \text{mín}\{3,2\} = 2$.

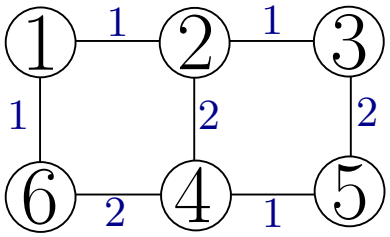


Figura 2.9: Malla de 2 dimensiones de $o = 3$, $p = 2$ y $B_C(G,\varphi) = 2$.

2.3.2 Límites teóricos para árboles respecto a sus características

De la misma manera es posible calcular el límite inferior de árboles en general a partir de las características que presenta. A continuación se muestran las ecuaciones reportadas en la literatura para obtener ese límite considerando su diámetro, los grados de sus nodos o su radio[7]. La Figura 2.10 muestra un árbol completo perfectamente balanceado con su etiquetado óptimo que genera un ancho de banda cíclico $B_C(G,\varphi) = 5$.

[7]El radio r de un grafo es el valor de excentricidad mínimo presente en alguno de sus nodos.

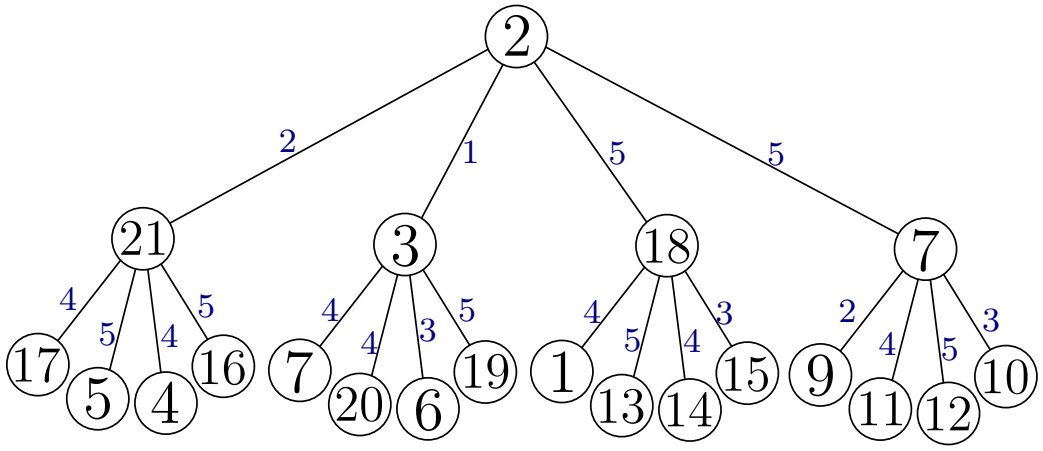


Figura 2.10: Grafo árbol, con $n = 21$, $m = 20$ y $B_C(G, \varphi) = 5$.

2.3.2.1. Diámetro

Lin (1994) estableció la Ecuación 2.13 para calcular el límite inferior de un grafo árbol considerando su diámetro d:

$$B_C(G, \varphi) \geq \left\lceil \frac{n-1}{d} \right\rceil, \tag{2.13}$$

en donde n es el orden del grafo. Para el árbol que se muestra en la Figura 2.10 el valor de su diámetro es $d = 4$, por lo tanto $B_C(G, \varphi) \geq \left\lceil \frac{21-1}{4} \right\rceil \geq 5$.

2.3.2.2. Radio

Zhou (2000) propuso la Ecuación 2.14, en donde a partir del radio de un grafo se puede calcular su límite inferior. Está ecuación solamente puede ser aplicada a árboles.

$$B_C(G, \varphi) \geq \left\lceil \frac{\lfloor n/2 \rfloor - 1}{r} \right\rceil, \tag{2.14}$$

donde n es el número de nodos y r es el radio del grafo. El árbol de la Figura 2.10 cuenta con un radio de $r = 2$, por lo que haciendo la sustitución en la ecuación tenemos que $B_C(G, \varphi) \geq \left\lceil \frac{\lfloor 21/2 \rfloor - 1}{2} \right\rceil \geq 5$.

2.3.2.3. Grado

Es posible obtener el límite inferior de un árbol en general a partir del grado $g(G)$ de sus nodos. En la investigación realizada por Zhou (2000) se estableció la Ecuación 2.15 que lo calcula.

$$B_C(G,\varphi) \geq \max_{1\leq\tau\leq n-1} \left\lfloor \frac{g_\tau(G)}{2\tau} \right\rfloor, \tag{2.15}$$

para cualquier entero $\tau \geq 1$, en donde $g_\tau(G)$ denota el grado τ máximo presente en el grafo G y n es el número de nodos.

Tipo			**Límite Inferior**	**Autor**
Planares			$B_C(G,\varphi) = \left\lceil \frac{m}{n} \right\rceil$	Lin (1997)
Orugas			$B_C(G,\varphi) = \text{máx}_{1\leq i\leq j\leq s} \left\lceil \frac{n_{ij}-1}{j-i+2} \right\rceil$	Lin (1994)
Estrellas			$B_C(G,\varphi) = \text{máx}\left\{ \text{máx}_{0\leq i\leq s} \left\lceil \frac{n_i-1}{2} \right\rceil, \text{máx}_{1\leq i\leq s} \left\lceil \frac{n_{0i}-1}{3} \right\rceil, \left\lceil \frac{n-1}{4} \right\rceil \right\}$	Lin (1994)
Árboles completos			$B_C(G,\varphi) = \left\lceil \frac{t(t^{s-1}-1)}{2(s-1)(t-1)} \right\rceil$	Hromkovič (1992)
Hipercubos			$B_C(G,\varphi) = \sum_{k=0}^{f-1} \left(\frac{k}{\lfloor k/2 \rfloor} \right)$	Hromkovič (1992)
Mallas de 2-dimensiones			$B_C(G,\varphi) = \text{mín}\{o,p\}$	Hromkovič (1992)
Mallas de 3-dimensiones			$B_C(G,\varphi) = \left\lfloor \frac{3o^2}{4} + \frac{o}{2} \right\rfloor$	Hromkovič (1992)
Árboles generales	**Característica**	Diámetro	$B_C(G,\varphi) \geq \left\lceil \frac{n-1}{d} \right\rceil$	Lin (1994 y 1997)
		Radio	$B_C(G,\varphi) \geq \left\lceil \frac{\lfloor n/2 \rfloor - 1}{r} \right\rceil$	Zhou (2000)
		Grados	$B_C(G,\varphi) \geq \text{máx}_{1\leq\tau\leq n-1} \left\lfloor \frac{g_\tau(G)}{2\tau} \right\rfloor$	Zhou (2000)

Tabla 2.3: Límites teóricos conocidos para el problema de MABC.

La Tabla 2.3 sintetiza las investigaciones y aportaciones realizadas a lo largo de los últimos años para el problema de MABC en relación a los límites teóricos, ya sea considerando el tipo de grafo o alguna de sus características.

Resumen del capítulo

Este capítulo presentó algunos términos que fueron base para definir formalmente el problema de MABC. También se habló de su complejidad computacional concluyendo que pertenece a la clase NP-completo. Por último, se analizaron las investigaciones previamente realizadas al problema, resumiendo en una tabla el trabajo teórico reportado. En el próximo capítulo se explicará detalladamente los enfoques de solución que se proponen para abordar este problema.

3

Enfoques de solución propuestos

Introducción

En la literatura especializada podemos encontrar una amplia varidad de algoritmos que dan solución a diversos PEG. El algoritmo de ramificación y poda ha sido utilizado para resolver problemas como el de ordenamiento lineal de corte mínimo (Palubeckis y Rubliauskas, 2011), etiquetado con máxima coincidencia (Carrabs *et al.*, 2009) y minimización del ancho de banda (Martí *et al.*, 2008) generando buenos resultados. También, se han creado algoritmos basados en la búsqueda tabú para medir la similitud entre grafos (Sorlin y Solnon, 2005), obtener el ancho de banda de una matriz dispersa (Martí *et al.*, 2001) y generar los ciclos hamiltonianos de mínimo costo (Cerulli *et al.*, 2006).

En este capítulo se describirán los algoritmos propuestos, los cuales dan solución al problema de MABC en grafos generales. Además, se especifican sus componentes generales así como las adaptaciones que fueron necesarias realizar.

3.1 Algoritmo de ramificación y poda

Los algoritmos de ramificación y poda son uno de los métodos exactos más utilizados para resolver problemas de optimización, ya que analizan de manera eficiente todo el conjunto de soluciones o espacio de búsqueda, descartando aquellas zonas donde no es factible encontrar la mejor solución.

El espacio de búsqueda es explorado mediante la generación dinámica de un árbol en donde el nodo raíz representa el caso a resolver, los nodos hojas son soluciones potenciales y los nodos internos son subproblemas de todo el conjunto de soluciones (Talbi, 2009). El recorrido de este árbol se puede realizar por medio de la técnica primero en anchura, primero en profundidad o utilizando el cálculo de funciones de costo. La implementación de alguna de estas técnicas dependerá de las características del problema a resolver. La estrategia de mínimo costo utiliza una función que evalúa cada nodo para decidir qué rama debe explorarse, con la esperanza de llegar rápido a una solución más económica que la mejor encontrada hasta el momento. Si alguna presenta un costo que empeore la solución actual, esta será podada (Guerequeta y Vallecillo, 2000).

Algunas funciones importantes a considerar cuando se realiza el recorrido de un árbol son: la función de selección, la función de ramificación y la función de poda. La función de *selección* se encarga de extraer un nodo del conjunto a explorar, que dependerá del tipo de recorrido que decidamos implementar. Dentro de la función de *ramificación* se construyen los nodos hijos del padre que se seleccionó en el paso anterior. Por último se realiza la *poda*, aquí se eliminan algunos de los nodos creados en la etapa anterior. Estas funciones se estarán ejecutando de manera iterativa mientras existan nodos sin explorar o mientras no se haya podado el nodo raíz, como se puede ver en la Figura 3.1. Mediante esta metodología se disminuye en lo posible el tamaño del espacio de búsqueda y se atenúa la complejidad de estos algoritmos que están basados en la exploración completa de un árbol.

Considerando las características antes mencionadas se ha diseñado un algoritmo de RP que resuelve el problema de MABC en instancias menores a 40 nodos, el cual es llamado RP-ABC en lo sucesivo. Este será descrito basándose en las tres estrategias que lo conforman.

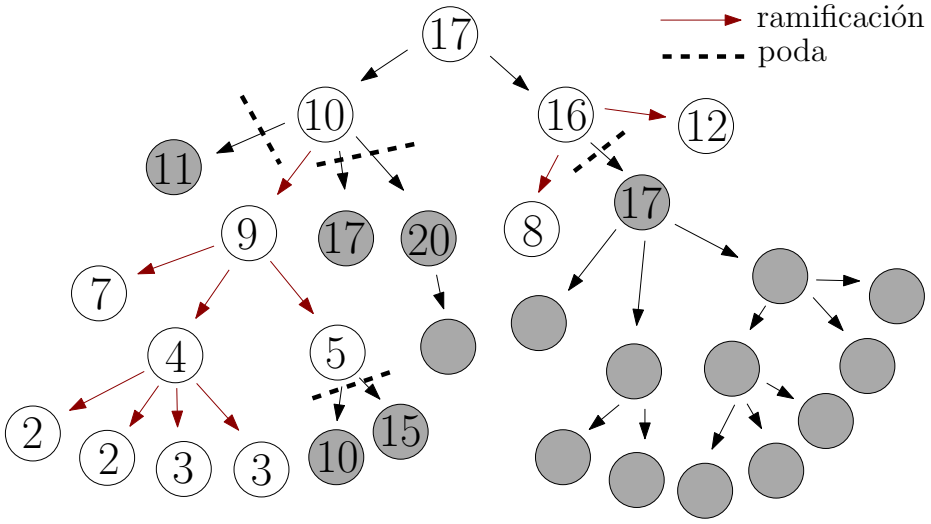


Figura 3.1: Ejemplo de un árbol de soluciones donde se implementa una función de costo y un recorrido primero en profundidad. El número dentro de cada nodo indica su costo y claramente se observan las ramas recorridas y las que son podadas.

3.1.1 Estrategia de ramificación

Cada posible etiquetado φ_i, en el rango de $1 \leqslant i \leqslant \frac{(n-1)!}{2}$ debe ser probado individualmente. RP-ABC recorre este conjunto de etiquetados de manera lexicográfica, iniciando con la solución $\varphi_1 = \{1, 2, 3, ..., n\}$ y terminando con la $\varphi_{\frac{(n-1)!}{2}} = \{n, n-1, n-2, ..., 1\}$. Cada etiqueta particular ℓ de la solución φ_i, en el rango de $1 \leqslant \ell \leqslant n$, es asignada a un nodo del grafo G (*etiquetar*$(v_\ell, \varphi_i(\ell))$) , una a la vez, produciendo soluciones parciales compuestas de tres subconjuntos: el subconjunto V' de nodos etiquetados, el subconjunto V_i'' de nodos sin etiquetar (que en el grafo son adyacentes a algún elemento del conjunto V') y el conjunto φ_i'' de etiquetas disponibles (ordenado lexicográficamente).

Cada solución parcial es evaluada considerando V' en el grafo G, de manera que cuando todas las etiquetas de φ_i hayan sido asignadas, se contará con una solución completa φ' de un costo menor a la cota establecida b. RP-ABC guarda este etiquetado como la mejor solución φ^* encontrada hasta el momento. De acuerdo con Leung *et al.* (1984), el costo de un grafo para el problema de MABC nunca será mayor a $\left\lfloor \frac{n}{2} \right\rfloor$, por lo que al inicio $b = \left\lfloor \frac{n}{2} \right\rfloor$. Cuando una solución φ^* sea encontrada, la búsqueda se reiniciará con una nueva cota $b = B_c(G, \varphi^*) - 1$.

El enfoque propuesto implementa una búsqueda primero en profundidad, de manera que si al etiquetar un nodo su costo $B_c(G, \varphi') > b$, esta rama es podada. Cuando esto ocurra se hará un *retroceso* al nodo padre para después continuar avanzando al siguiente nodo hijo. La búsqueda se detendrá cuando todos los etiquetados φ_i hayan sido analizados.

En el Algoritmo 1 se puede observar el pseudocódigo de la propuesta desarrollada para resolver el problema de MABC utilizando la técnica de ramificación y poda.

3.1.2 Estrategia de asignación

Existen dos aspectos muy importantes a considerar cuando se etiqueta un nodo: el costo del $B_c(G, \varphi')$ que se obtendrá y el costo menor que se puede obtener al asignar las etiquetas disponibles φ_i'' al conjunto de nodos no etiquetados V_i''. Esta última tarea nos ayuda a saber si será posible etiquetar a los que son adyacentes a la solución parcial sin exceder la cota b en futuras iteraciones.

El método Húngaro es un algoritmo que ayuda a resolver problemas de asignación en tiempo polinomial (Kuhn, 1955). RP-ABC utiliza una adaptación de este algoritmo para calcular el costo de asignar cada etiqueta $e \in \varphi_i''$ a cada nodo $v \in V_i''$. Sin embargo, si la asignación sobrepasa la cota actual b entonces no es necesario continuar con la evaluación de φ_i.

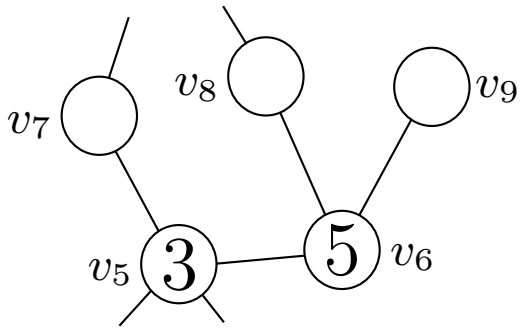


Figura 3.2: Ejemplo de una grafo parcial etiquetado.

A continuación se muestra un ejemplo de cómo se efectúa la asignación. Considere el grafo $G = (V, E)$ que se muestra parcialmente en la Figura 3.2, el cual está formado por cinco nodos etiquetados con el número que se muestra en su interior; también, considere $n = 10$, $\ell = 6$ y

Algoritmo 1 Algoritmo RP-ABC

$b = \frac{n}{2}$
$\varphi_i = \{1, 2, 3, ..., n\}$
$V' \leftarrow \varphi' \leftarrow \varphi^* \leftarrow \emptyset$
$\ell \leftarrow i \leftarrow 1$
mientras $i \leq (n-1)!/2$ **hacer**
  etiquetar$(v_\ell, \varphi_i(\ell))$
  $\varphi' \leftarrow \varphi' \cup \{\varphi_i(\ell)\}$
  $V' \leftarrow V' \cup \{v_\ell\}$
  si $B_c(G, \varphi') > b$ **entonces**
    $j \leftarrow$ retroceso$(\varphi', \varphi_i, \ell)$ // Sección 3.1.3
    si $j > 0$ **entonces**
      $\ell \leftarrow j$
      etiquetar$(v_\ell, \varphi_i(\ell))$
      $V' \leftarrow V' \cup \{v_\ell\}$
    sino
      devolver φ^*
    fin si
  fin si
  $V'' \leftarrow \{v \in (V \backslash V') | (v, u) \in E, u \in V'\}$
  $\varphi'' \leftarrow \{e \in (\varphi \backslash \varphi')\}$
  asignación(V'', φ'') // Sección 3.1.2
  si $B_c(G, \varphi') > b$ **entonces**
    $j \leftarrow$ retroceso$(\varphi', \varphi_i, \ell)$
    si $j > 0$ **entonces**
      $\ell \leftarrow j$
      etiquetar$(v_\ell, \varphi_i(\ell))$
      $V' \leftarrow V' \cup \{v_\ell\}$
    sino
      devolver φ^*
    fin si
  fin si
  si $\ell = n$ **entonces**
    $\varphi^* \leftarrow \varphi'$
    $b \leftarrow b - 1$
    $j \leftarrow$ retroceso$(\varphi', \varphi_i, \ell)$
    si $j > 0$ **entonces**
      $\ell \leftarrow j$
      etiquetar$(v_\ell, \varphi_i(\ell))$
      $V' \leftarrow V' \cup \{v_\ell\}$
    sino
      devolver φ^*
    fin si
  fin si
  $\ell \leftarrow \ell + 1$
fin mientras

	φ_i''			
V_i''	*1*	*4*	*9*	*10*
v_7	**2**	1	5	3
v_8	4	**1**	4	5
v_9	4	1	4	5

Tabla 3.1: Tabla de asignación nodo-etiqueta.

$b = 3$ para este ejemplo. Ya que los elementos del conjunto V son tomados de manera ordenada, la última acción de etiquetado que se realizó fue *etiquetar*$(v_6, 5)$, por lo tanto $V' = \{...v_{\ell-1}, v_\ell\}$, $\varphi' = \{..., 3, 5\}$ y $V_i'' = \{v_7, v_8, v_9\}$. Asumiendo que $\varphi_i'' = \{1, 4, 9, 10\}$ y que la solución parcial $B_c(G, \varphi') < b$, utilizamos el método de asignación para estimar el costo de aplicar las etiquetas φ_i'' a los nodos V_i'' como se presenta en la Tabla 3.1.

Se puede ver claramente en esta tabla que v_7 y v_8 podrán ser etiquetados sin exceder la cota b en futuras iteraciones. Sin embargo, la condición $B_c(G, \varphi') \leq b$ para v_9 no puede ser garantizada. En este caso la evaluación de φ_i es detenida, haciendo un retroceso al nodo padre y continuando con el siguiente nodo hijo lo cual generará el próximo etiquetado φ_{i+1} no evaluado.

3.1.3 Estrategia de retroceso

Un retroceso ocurre cuando se añade una etiqueta a la posición ℓ-ésima de la solución parcial φ' y el ancho de banda cíclico que genera excede la cota b. Los pasos que se llevan acabo en este proceso se describe en el Algoritmo 2.

Primero es necesario identificar al mejor elemento del conjunto φ_i''. El mejor elemento será la primera etiqueta que garantice que el costo del $B_c(G, \varphi')$ estará por debajo de la cota actual b cuando sea asignada al nodo v_ℓ. Si no existe tal etiqueta, entonces se hace un retroceso a la posición $\ell - 1$, se actualiza el conjunto φ_i'' y se reinicia la búsqueda del mejor elemento. La búsqueda de la mejor etiqueta y la acción de retroceso implica la generación de soluciones φ_i, por lo que la variable i se incrementa iterativamente.

Algoritmo 2 Estrategia de retroceso

```
retroceso( φ', φ_i, ℓ )
φ'' Etiquetas disponibles
mientras verdadero hacer
   φ'(ℓ) ← mejor(φ'', φ_i) // Cada evaluación es un incremento de i
   si φ'(ℓ) == ∅ entonces
      ℓ ← ℓ − 1
      si ℓ > 0 entonces
         φ'' ← φ'' ∪ {φ'(ℓ)}
         φ'(ℓ) ← ∅
      sino
         devolver ℓ
      fin si
   sino
      devolver ℓ
   fin si
fin mientras
```

Este proceso se repetirá hasta que dicha etiqueta sea encontrada o hasta que se analicen todas las posiciones del conjunto φ', es decir $\ell = 0$.

3.2 Algoritmo de búsqueda tabú

El enfoque de búsqueda tabú fue propuesto originalmente por Glover (1986) para resolver problemas de optimización, y es considerado como una opción viable pues haciendo un buen uso de sus componentes se pueden encontrar soluciones cercanas al óptimo. El objetivo con el que fue desarrollado fue el de continuar la exploración más allá de los puntos de optimalidad local, pues algunas técnicas se limitaban solo a cierta zonas del espacio de búsqueda (Gendreau y Potvin, 2010).

Para llevar acabo la exploración el algoritmo de búsqueda tabú se apoya en la creación de vecindarios haciendo pequeños cambios a un solución potencial. Este proceso se realiza iterativamente utilizando las técnicas reportadas en la literatura o desarrollando funciones a la medida del problema (Hertz *et al.*, 1995). Para controlar la cantidad de soluciones vecinas creadas se utiliza una *lista de nodos aspirantes*, la cuál permite obtener soluciones que no habían sido consideradas o de mejor calidad.

Un punto importante que contempla el algoritmo es la creación repetitiva de una solución, lo que origina entrar en ciclos durante la búsqueda. Para evitar este inconveniente, cada vez que una solución es evaluada se marca como prohibida (o *tabú*) y se almacena en una lista, de manera que no podrá ser contemplada por cierto tiempo hasta que pierda su estado de penalización. El tamaño de la lista tabú determina el tiempo o número de iteraciones que un elemento permanece en la lista, el cual puede ser fijo o variable (Talbi, 2009).

Utilizando el enfoque de BT, se desarrolló el Algoritmo 3 que proporciona soluciones para el problema de MABC en grafos generales, el cual es nombrado BT-ABC en lo sucesivo.

El algoritmo BT-ABC que se propone está conformado por un conjunto de componentes los cuales contribuyen al cumplimiento del objetivo de este enfoque heurístico que es reducir el tiempo de solución de grafos con $n < 40$ en relación con el empleado por el algoritmo RP-ABC. También, se pretende dar solución a grafos más grandes, con $n > 40$. Los componentes son los siguientes:

- Solución inicial
- Lista de nodos aspirantes
- Función de vecindad
- Función de diversificación
- Lista tabú
- Criterio de paro

Es importante mencionar que para algunos de estos componentes se desarrolló más de una propuesta con el fin de obtener el mejor desempeño posible del algoritmo. A continuación se explica de manera detallada el funcionamiento de cada uno de ellos.

Algoritmo 3 Algoritmo BT-ABC

BT-ABC(δ, ρ, $max\mathcal{S}$, $max\mathcal{F}$, $Diver$**)**
φ Solución inicial
$\varphi^* \leftarrow \varphi$
$it \leftarrow itSinMejora \leftarrow itDiver\mathcal{S} \leftarrow itDiver\mathcal{F} \leftarrow 0$
$ABCmin \leftarrow$ costoMínimo(G) // Sección 3.2.6
mientras $itDiver\mathcal{F} < max\mathcal{F}$ y $B_c(G, \varphi^*) > ABCmin$ **hacer**
  Actualizar y establecer tamaño de lista tabú // Sección 3.2.5
  $rnd \in [0, 1]$
  si $rnd > \delta$ **entonces**
    $\varphi' \leftarrow \mathcal{N}_1(\varphi, it)$ // Sección 3.2.3.1
  sino
    $\varphi' \leftarrow \mathcal{N}_2(\varphi, it)$ // Sección 3.2.3.2
  fin si
  $\varphi \leftarrow \varphi'$
  si $B_c(G, \varphi) < B_c(G, \varphi^*)$ **entonces**
    $\varphi^* \leftarrow \varphi$
    $itSinMejora \leftarrow 0$
    $itDiver\mathcal{S} \leftarrow 0$
  sino
    $itSinMejora \leftarrow itSinMejora + 1$
  fin si
  si $itSinMejora \geq maxitSinMejora$ **entonces**
    $itDiver\mathcal{S} \leftarrow itDiver\mathcal{S} + 1$
    si $itDiver\mathcal{S} \geq max\mathcal{S}$ **entonces**
      si $Diver == 0$ **entonces**
        $\varphi' \leftarrow \mathcal{D}_2(\varphi)$ // Sección 3.2.4.2
      sino
        $\varphi' \leftarrow \mathcal{D}_3(\varphi)$ // Sección 3.2.4.3
      fin si
      $itDiver\mathcal{F} \leftarrow itDiver\mathcal{F} + 1$
      $itDiver\mathcal{S} \leftarrow 0$
    sino
      $\varphi' \leftarrow \mathcal{D}_1(\varphi, \rho)$ // Sección 3.2.4.1
    fin si
  fin si
  $it \leftarrow it + 1$
fin mientras

3.2.1 Solución inicial

La primera solución que prueba BT-ABC es creada aleatoriamente, de manera que todos los nodos disponibles tienen la misma posibilidad de ser etiquetados con cualquiera de las etiquetas que no han sido asignadas.

3.2.2 Lista de nodos aspirantes

Para controlar la generación de grandes cantidades de soluciones por una función de vencidad se puede hacer uso de una lista de nodos aspirantes $\mathcal{A}(u)$, la cual ayuda a formar buenas soluciones y evitar las de mala calidad. Sin embargo, si este recurso no es abordado adecuadamente se puede afectar el desempeño del algoritmo, por lo que es recomendable conocer a fondo el problema que se va a abordar. Glover y Laguna (1997) realizan un estudio sobre diversas estrategias útiles para crear estas listas.

En esta propuesta de solución para el problema de MABC se utilizan dos enfoques para crear la lista de nodos aspirantes ($\mathcal{A}_1(u)$ y $\mathcal{A}_2(u)$). Es importante resaltar que en ambos casos la lista es creada para un nodo específico u. A continuación se detalla cómo se crea cada una de ellas.

- **Lista de nodos aspirantes** $\mathcal{A}_1(u)$. Esta lista está formada por nodos elegidos aleatoriamente y que no se encuentran en estado *tabú*. La cardinalidad de este conjunto es de $\mid \mathcal{A}_1(u) \mid = \gamma$.

- **Lista de nodos aspirantes** $\mathcal{A}_2(u)$. Para formar la lista de nodos aspirantes de un nodo u primero se requiere visitar todos sus nodos adyacentes $\mathcal{N}(u)$. De este conjunto se identifican los dos nodos que dentro del grafo cíclico se encuentran más alejados entre sí (ver Figura 2.3(b)), los cuales serán nombrados $\mathrm{máx}(u)$ y $\mathrm{mín}(u)$. En este caso la mejor opción es hacer el cambio con el nodo intermedio $mid(u) = \left\lfloor \frac{\mathrm{máx}(u)+\mathrm{mín}(u)}{2} \right\rfloor$; sin embargo, es necesario contemplar también otros factores. La lista $\mathcal{A}_2(u)$ la formarán los nodos que no sean *tabú* y que cumplan con la condición que se indica en la siguiente ecuación:

$$\mathcal{A}_2(u) = \{v \in \mathcal{N}(u) \big| \, |mid(u) - \varphi(v)|_n < |mid(u) - \varphi(u)|_n\} \tag{3.1}$$

Esta condición considera a todos los nodos v que cuentan con un etiquetado cercano a $mid(u)$, con el fin de retiquetar u y que este disminuya su ancho de banda cíclico.

3.2.3 Función de vecindad

Una de las características a resaltar del problema de MABC es que diversas soluciones pueden llegar a generar el mismo ancho de banda cíclico $B_C(G, \varphi)$. Además, como en muchos otros POC, se tiene un espacio de búsqueda que crece de manera factorial. Esto puede representar un problema muy grande en el desarrollo de algoritmos, pues es difícil encontrar soluciones de buena calidad lo que compromete a desarrollar funciones de vecidad que ayuden a salir de tantos puntos de optimalidad local.

Para hacer frente a esta problemática existen dos recursos, los cuales han ayudado a crear soluciones de mejor calidad: *lista de nodos críticos ($\mathcal{C}(\varphi)$)* y *lista de nodos aspirantes ($\mathcal{A}(u)$)*. La lista de nodos críticos está formada por aquellos que al ser reetiquetados podrían llevar a obtener una solución de mejor calidad. En este trabajo de tesis se experimentó con diferentes funciones de vecindad y se tomaron las dos que mejor se comportaron en los experimentos preliminares. Ambas funciones implementan un operador de intercambio de elementos nombrado *swap*, el cual puede definirse de la siguiente manera: Sea $u \in \mathcal{C}(\varphi) \subseteq V$ y sea $v \in \mathcal{A}(u) \subseteq V$ una opción de intercambio para u. El operador *swap*(φ, u, v) intercambia las etiquetas de los nodos u y v entre ellos ($\varphi'(u) = \varphi(v)$ y $\varphi'(v) = \varphi(u)$) generando una nueva solución φ'.

Las funciones de vecindad implementadas en BT-ABC son detalladas a continuación.

3.2.3.1. *Función de vecidad $\mathcal{N}_1$ (reparación simple)*

Para esta función de vecidad la lista de nodos críticos está formada por todos aquellos nodos que no se encuentran en estado *tabú* y que cumplen con el criterio que se indica en la siguiente ecuación:

$$\mathcal{C}_1(\varphi) = \{u \in V \mid B_C(u,\varphi) \geq \alpha \cdot B_C(G,\varphi)\}\,, \tag{3.2}$$

en donde $1 > \alpha > 0$. Estos son los nodos $u \in V$ que tienen un costo $B_C(u,\varphi)$ igual o cercano al ancho de banda cíclico del grafo $B_C(G,\varphi)$, de manera que al ser reparados podrían reducir el costo del grafo. En esta función en particular se utiliza la lista de nodos aspirantes $\mathcal{A}_1(u)$, descrita en la Sección 3.2.2.

La lista de nodos aspirantes se crea para cada $u \in \mathcal{C}_1(\varphi)$. Para seleccionar el nodo $v \in \mathcal{A}_1(u)$ que representa la mejor opción, se evalua el ancho de banda cíclico al intercambiar la etiqueta de u con cada una de las etiquetas asignadas a los nodos en $\mathcal{A}_1(u)$, eligiendo la de menor costo.

La construcción de los conjuntos $\mathcal{C}_1(\varphi)$ y $\mathcal{A}_1(u)$ para todos los nodos $u \in \mathcal{C}_1(\varphi)$, así como la selección del mejor nodo $v \in \mathcal{A}_1(u)$ es realizada de manera iterativa mientras el ancho de banda cíclico del grafo $B_C(G,\varphi)$ se reduzca o mientras este se mantenga pero disminuya $|\mathcal{C}_1(\varphi)|$. De manera formal $\mathcal{N}_1(\varphi)$ puede definirse como:

$$\mathcal{N}_1(\varphi) = \{\varphi' = \textit{swap}(\varphi,u,v) \mid \forall u \in \mathcal{C}_1(\varphi),\ v \in \mathcal{A}_1(u)\} \tag{3.3}$$

El Algoritmo 4 muestra el pseudocódigo del vecindario de reparación simple $\mathcal{N}_1$.

3.2.3.2. *Función de vecidad $\mathcal{N}_2$ (reparación especial)*

En esta función se hace uso de la lista de nodos aspirantes $\mathcal{A}_2(u)$ y la lista de nodos críticos $\mathcal{C}_1(\varphi)$ previamente definida en la Sección 3.2.3.1.

Para seleccionar el nodo $v \in \mathcal{A}_2(u)$ que representa la mejor opción, se evalua el impacto de

Algoritmo 4 Función de vecidad $\mathcal{N}_1$ (reparación simple)

función $\mathcal{N}_1$**(** φ, it **)**
$\varphi^* \leftarrow \varphi$
$it \leftarrow it + 1$
mientras verdadero **hacer**
  construir $\mathcal{C}_1(\varphi)$
  $c \leftarrow \mid \mathcal{C}_1(\varphi) \mid$
  $ABC \leftarrow B_C(G, \varphi)$
  mientras $\mid \mathcal{C}_1(\varphi) \mid \neq 0$ **hacer**
    tomar el primero nodo $u \in \mathcal{C}_1(\varphi)$
    construir $\mathcal{A}_1(u)$
    buscar mejor opción $v \in \mathcal{A}_1(u)$
    $\varphi' \leftarrow$ swap(φ, u, v)
    tabú(it, u) // Considerando el tamaño de la lista tabú
    $\mathcal{C}_1(\varphi) \leftarrow \mathcal{C}_1(\varphi) - \{u\}$
  fin mientras
  Construir $\mathcal{C}_1(\varphi')$
  $c' \leftarrow \mid \mathcal{C}_1(\varphi') \mid$
  si $B_C(G, \varphi') > ABC \mid\mid (B_C(G, \varphi') == ABC$ && $c' \geq c)$ **entonces**
    devolver φ^*
  sino
    $\varphi^* \leftarrow \varphi'$
  fin si
fin mientras

intercambiar la etiqueta de u con cada una de las etiquetas asignadas a los nodos de la lista de nodos aspirantes, considerando en todo momento a los nodos adyacentes a u, $\mathcal{N}(u)$ (ver Algoritmo 5). Para cada $w \in \mathcal{N}(u)$ que cumpla con la condición: $|\varphi'(u) - \varphi(w)|_n > B_C(w, \varphi)$ y $|\varphi'(u) - \varphi(w)|_n > \beta \cdot B_C(G, \varphi)$ se le acomulará un punto negativo. Esta misma tarea también se aplica para los vecinos de v, $\mathcal{N}(v)$. Al final la mejor opción v será aquella que menos puntos negativos haya acumulado.

Al considerar los conjuntos de nodos $\mathcal{N}(v)$ y $\mathcal{N}(u)$ en el intercambio, se asegura que la solución φ' resultante mejore los costos en u y v, y además que no afecte a sus nodos adyacentes. Esta función se estará ejecutando iterativamente bajo las mismas condiciones que se mencionaron en la descripción de $\mathcal{N}_1$ (ver Algoritmo 4). Su definición formal se puede observar en la Ecuación 3.4.

$$\mathcal{N}_2(\varphi) = \{\varphi' = \textit{swap}(\varphi, u, v) \mid \forall u \in \mathcal{C}_1(\varphi),\ v \in \mathcal{A}_2(u)\} \tag{3.4}$$

Algoritmo 5 Búsqueda de la mejor opción

mejorOpción(φ, u, β **)**
construir $\mathcal{A}_2(u)$
$v^* \leftarrow \emptyset$
$costo^* \leftarrow n^2$
mientras $|\mathcal{A}_2(u)| \neq 0$ **hacer**
  tomar el primer nodo $v \in \mathcal{A}_2(u)$
  simular *swap*(φ, u, v)
  $costo \leftarrow 0$
  Para $\forall w \in \mathcal{N}(u)$
  si $|\varphi'(u) - \varphi(w)|_n > B_C(w, \varphi)$ && $|\varphi'(u) - \varphi(w)|_n > \beta \cdot B_C(G, \varphi)$ **entonces**
    $costo \leftarrow costo + 1$
  fin si
  Para $\forall w \in \mathcal{N}(v)$
  si $|\varphi'(v) - \varphi(w)|_n > B_C(w, \varphi)$ && $|\varphi'(v) - \varphi(w)|_n > \beta \cdot B_C(G, \varphi)$ **entonces**
    $costo \leftarrow costo + 1$
  fin si
  si $costo < costo^*$ **entonces**
    $costo^* \leftarrow costo$
    $v^* \leftarrow v$
  fin si
  $\mathcal{A}_2(u) \leftarrow \mathcal{A}_2(u) - \{v\}$
fin mientras
devolver v^*

Una vez definidas estas funciones de vecindad se puede obtener una tercera función $\mathcal{N}_3$ con la combinación de las dos anteriores. Para esta combinación de vecindarios $\mathcal{N}_3(\varphi, rnd)$ se aplica con una probabilidad δ que establece la intervención de cada una de ellas. De manera formal se puede definir esta nueva función como se muestra en la siguiente ecuación:

$$\mathcal{N}_3(\varphi, rnd) = \begin{cases} \mathcal{N}_1(\varphi) & \text{si } rnd > \delta \\ \mathcal{N}_2(\varphi) & \text{si } rnd \leq \delta \end{cases} \tag{3.5}$$

donde rnd es un número aleatorio con distribución uniforme en el intervalo $[0, 1]$.

3.2.4 Diversificación

Las estrategias de diversificación tienen como fin promover que el algoritmo visite zonas del espacio de búsqueda que aún no han sido exploradas. En las siguientes subsecciones se describirán

las tres funciones de diversificación que fueron implementadas en BT-MABC para dar solución al problema de MABC en grafos generales.

3.2.4.1. Función de diversificación $\mathcal{D}_1$ (nodos aleatorios)

Esta función de diversificación utiliza una lista de nodos aspirantes $\mathcal{A}_2(u)$ y una lista de nodos críticos $\mathcal{C}_2(\varphi)$ formada por nodos elegidos aleatoriamente que no se encuentran en estado *tabú*. Este conjunto tendrá una cardinalidad $|\mathcal{C}_2(\varphi)| = \gamma$. La estrategia para seleccionar el mejor nodo $v \in \mathcal{A}_2(u)$ es la misma que se aplicó en la Sección 3.2.3.2 correspondiente a la función de vecindad $\mathcal{N}_2$. La definición formal de esta función se puede apreciar en la Ecuación 3.6.

$$\mathcal{D}_1(\varphi, \rho) = \{\varphi' = \textit{swap}(\varphi, u, v) \mid \forall u \in \mathcal{C}_2(\varphi), v \in \mathcal{A}_2(u)\} \tag{3.6}$$

Esta función de diversificación cuenta con una variable externa ρ que indica con que fuerza se modificará la solución. En otras palabras, su valor establacerá el número de veces que intervendrá esta función sobre la misma solución.

3.2.4.2. Función de diversificación $\mathcal{D}_2$ (permutación de los mejores nodos)

Para esta función los nodos del grafo son ordenados de manera ascendente respecto a su ancho de banda cíclico. Después se permutan las etiquetas de los primeros γ nodos dando origen a una nueva solución φ'. Esto puede definirse formalmente de la siguiente manera: sea $\mathcal{R}$ un conjunto totalmente ordenado de V bajo la relación $B_C(u, \varphi) \leq B_C(v, \varphi)$, para todo $u, v \in V$.

$$\mathcal{R}(V, \leq) = \{u \mid \forall u, v \in V \ , \ B_C(u, \varphi) \leq B_C(v, \varphi)\} \tag{3.7}$$

Entonces es posible definir un conjunto $\mathcal{R}'(\gamma) \subseteq \mathcal{R}(V, \leq)$ de la siguiente manera:

$$\mathcal{R}'(\gamma) = \{u_i \mid 1 \leq i \leq \gamma \ , \ u_i \in \mathcal{R}\} \tag{3.8}$$

Por lo tanto la función de diversificación $\mathcal{D}_2$ puede definirse de manera formal como se expresa en la Ecuación 3.9.

$$\mathcal{D}_2(\gamma, \varphi) = \{\varphi' = \textit{swap}(\varphi, u, v) \mid u, v \in \mathcal{R}'(\gamma)\} \tag{3.9}$$

3.2.4.3. Función de diversificación $\mathcal{D}_3$ (etiquetas menos frecuentes)

En esta función los nodos son ordenados de manera descendente respecto a su ancho de banda cíclico ($\mathcal{R}$) y las etiquetas en orden lexicográfico ($\mathcal{L}$). Es importante mencionar que en las funciones de vencidad ($\mathcal{N}_1$, $\mathcal{N}_2$ y $\mathcal{N}_3$) cada vez que interviene el operador *swap*(φ, u, v) se registra la relación etiqueta-nodo. El objetivo es saber con que frecuencia una etiqueta específica se establece sobre cada nodo.

Para etiquetar un nodo $u \in \mathcal{R}$ se elige un número $i \in [1, n]$. Después se realizará la búsqueda entre las etiquetas $\gamma + i \leq \ell \leq i$, del conjunto $\mathcal{L}(\ell)$. La mejor etiqueta $m(u, \mathcal{L})$ para el nodo u es aquella que presenta menos frecuencia sobre dicho nodo y que además está disponible. Este proceso se repetirá para todo el conjunto $\mathcal{R}$. Formalmente puede definirse de la siguiente manera:

$$\mathcal{R}(V, \geq) = \{u \mid \forall u, v \in V,\ B_C(u, \varphi) \geq B_C(v, \varphi)\} \tag{3.10}$$

Para un conjunto ordenado $\mathcal{R}$ y una función $m(u, \mathcal{L})$ que retorna la mejor opción de intercambio del nodo $u \in \mathcal{R}$, la función de diversificación $\mathcal{D}_3$ puede definirse como:

$$\mathcal{D}_3(\varphi) = \{\varphi' = \textit{swap}(\varphi, u, v) \mid u \in \mathcal{R}, v = m(u, \mathcal{L})\} \tag{3.11}$$

3.2.5 Lista tabú

La lista tabú lleva un registro de los movimientos realizados. Esto tiene como objetivo no intensificar la búsqueda en un sector específico sino obligar al algoritmo a trasladarse más allá del

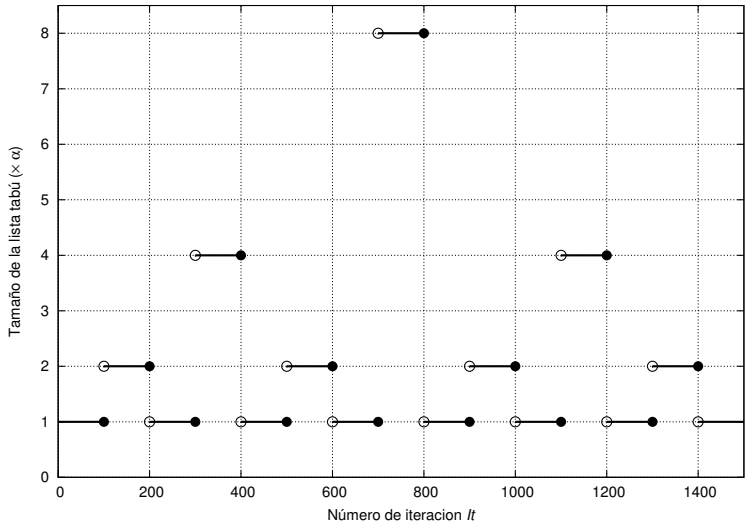


Figura 3.3: Representación gráfica de la función de paso periódico utilizada para definir el tamaño de la lista tabú.

punto actual. Por lo tanto, entre más grande sea el tamaño de la lista tabú más exploración se hará.

El tamaño de la lista tabú puede ser *fijo* o *variable*. Una lista de tamaño fijo siempre penaliza por la misma cantidad de tiempo (iteraciones), por lo que se puede decir que implementa una disciplina tipo FIFO (primero en entrar, primero en salir). Una lista de tamaño variable considerará diversos factores para determinar la cantidad de tiempo que una solución quedará penalizada.

Galinier *et al.* (2011) propusieron un método para crear una lista tabú de tamaño variable, que fue implementado posteriormente por Wu y Hao (2013). Este enfoque está basado en el uso de una función de paso periódico PP, que recibe como argumento el número de iteración actual it (ver Figura 3.3). A cada periodo lo conforman 1500 iteraciones divididas en 15 intervalos $[x_i, x_{i+1} - 1]_{i=1,2,3,\ldots,15}$, donde $x_1 = 1$ y $x_{i+1} = x_i + 100$. El resultado obtenido por PP para una iteración particular $it \in [x_i, x_{i+1} - 1]$ está dado por $(a)_{i=1,2,3,\ldots,15} = (1, 2, 1, 4, 1, 2, 1, 8, 1, 2, 1, 4, 1, 2, 1) \cdot \theta$, donde θ es una parámetro que fija el valor mínimo del tamaño de la lista tabú. Por lo tanto, el tamaño de la lista tabú es igual a θ entre la iteración 1 y la 100, $2 \cdot \theta$ entre la iteración 101 y 200, de nuevo θ para las iteraciones $[201, 300]$, $4 \cdot \theta$ para las iteraciones $[301, 400]$, etc. Este proceso se repite periódicamente después de cada 1500 iteraciones.

3.2.6 Criterio de paro

El algoritmo BT-ABC considera dos críterios para detener la búsqueda de la mejor solución:

- **Intervención de las funciones de diversificación**.

  Se finaliza la búsqueda cuando hayan intervenido $max\mathcal{F}$ veces cualesquiera de las funciones de diversificación $\mathcal{D}_2$ o $\mathcal{D}_3$. Este par de funciones a diferencia de $\mathcal{D}_1$ realizan cambios más drásticos sobre la solución actual, por tal motivo son tomadas en cuenta de manera diferente y son clasificadas como funciones de diversificación fuertes. El inconveniente de este criterio es que si ya no existe una mejor solución que alguna que se haya encontrado en las primeras iteraciones, es necesario esperar a que se ejecuten las funciones de diversificación $maxF$ veces.

- **Costo mínimo**.

  Una cota inferior que puede presentarse en un grafo para el problema de MABC está determinada por el nodo de menor grado. De manera que si se cuentra una solución $B_C(G, \varphi)$ igual a ese valor la búsqueda ha concluido. Un ejemplo claro de este caso se presenta en los grafos camino donde el nodo de menor grado es 1, y el valor de ancho de banda cíclico óptimo es $B_C(G, \varphi^*) = 1$.

Resumen del capítulo

En este capítulo se presentarón dos algoritmos para solucionar el problema de MABC. El algoritmo de RP, que como se mencionó es utilizado para resolver problemas de etiquetado de grafos, fue adaptado para resolver este problema. Su descripción fue dividida en tres etapas *ramificación, asignación y retroceso* y se justificó el porque cada una es importante. También, se describió la propuesta de un algoritmo de BT que utiliza funciones de vecidad avanzadas, con la finalidad de resolver de manera eficiente el problema de estudio de esta tesis.

En el siguiente capítulo se presenta la etapa de experimentación, donde se detallan las instancias que se utilizaron así como las condiciones experimentales bajo las cuales se ejecutaron los algoritmos. Además, se muestran los resultados obtenidos y algunas conclusiones a las que se llegaron.

4

Experimentación y Resultados

Introducción

En este capítulo se presentan los experimentos realizados para medir el desempeño de los algoritmos propuestos en capítulo 3 de esta tesis. Los resultados obtenidos por el algoritmo de ramificación y poda RP-ABC y el algoritmo BT-ABC que implementa una búsqueda tabú fueron contrastados, considerando factores como la calidad de la solución y el tiempo de cómputo para medir el desempeño. Ambos algoritmos se ejecutaron sobre la misma arquitectura de cómputo.

Es importante mencionar que el algoritmo BT-ABC fue sometido a un proceso de sintonización de parámetros utilizando un técnica inspirada en las pruebas de interacción combinatoria. Esto ayudó a mejorar el desempeño del algoritmo y la calidad de las soluciones. Al final del capítulo se presentan los resultados obtenidos en tablas y de manera gráfica, estableciendo cotas para los conjuntos de prueba previamente mencionados.

4.1 Criterios de comparación

Para medir el desempeño que tienen los algoritmos propuestos se han considerado tres criterios, los cuales se mencionan a continuación:

- Calidad de solución. El mejor valor de ancho de banda cíclico $B_c(G, \varphi)$ encontrado.
- Tiempo de ejecución. La cantidad de segundos que le toma al algoritmo concluir la búsqueda.
- Número de soluciones parciales probadas. Para el algoritmo RP-ABC se considera una solución parcial cada vez que un nodo es etiquetado y evaluado durante el proceso de búsqueda. En el algoritmo BT-ABC una solución parcial es aquella en la que todos sus nodos críticos (lista $\mathcal{C}$) han sido reparados.

Estos criterios se tomaron en cuenta al probar los algoritmos utilizando diversos conjuntos de prueba, los cuales se describen a continuación.

4.2 Instancias de prueba

En la literatura estudiada no se encontró reportado ningún conjunto de prueba que pudiera ser utilizado para comparar algoritmos. Por tal motivo, se propone una colección de instancias cuyo límite inferior es conocido, lo que permite comparar algoritmos con respecto a la calidad de la solución. También, se adoptó un conjunto que ha sido utilizado en diversos PEG ya que fue creado a partir de situaciones prácticas. Por último, se formó una colección de grafos aleatorios creados mediante un algoritmo reportado en la literatura. En las siguientes subsecciones se describe cada conjunto con más detalle.

4.2.1 Grafos ordinarios

Se propone un conjunto de prueba conformado por cuatro tipos de grafos: ciclos, mallas, caminos y árboles binarios. En esta colección de 20 instancias cada una tiene un tamaño que varía entre 20 y 40 nodos, y sus límites inferiores coinciden con los mencionados por Klerk *et al.* (2011) para el problema de MAB.

4.2.2 Grafos Harwell Boeing

La colección de matrices dispersas *Harwell-Boeing*[1] propuesta por (Duff *et al.*, 1992) ha sido utilizada como casos de prueba en diversos PEG. Las matrices fueron creadas a partir de problemas de sistemas lineales, mínimos cuadrados y cálculo de auto-valores de una amplia variedad de disciplinas científicas.

De este amplio repertorio se tomaron dos subconjuntos con 12 instancias cada uno. Anteriormente estos han sido utilizados para resolver otros PEG como el de MAB (Mladenovic *et al.*, 2010) o el de maximización del antibandwidth (Lozano *et al.*, 2011). El primer subconjunto contiene grafos con menos de 120 nodos (*HBp*) y el segundo grafos de entre 400 y 750 nodos (*HBg*).

4.2.3 Grafos aleatorios

Este grupo está formado por 20 grafos aleatorios creados con el algoritmo propuesto por Viger y Latapy (2005). Para ello se seleccionan n número aleatorios (cantidad de nodos) en un rango entre 0 y y, para formar una secuencia binomial utilizando una media z. Estos valores son los parámetros de entrada del algoritmo. El tamaño de las instancias formadas oscilan entre 20 y 40 nodos, y se utilizaron los siguientes valores para y:100 y 1000; y para z: 5, 10 y 15, mismos que fueron sugeridos por los autores. El nombre de las instancias esta formado por sus valores de n, y y z (rnd$n$-y-z).

[1]Este conjunto se encuentran disponibles en la siguiente dirección electrónica: `http://math.nist.gov/MatrixMarket/data/Harwell-Boeing`.

4.2.4 Instancias de árboles

Se construyó un conjunto de 14 árboles s-ários balanceados de h-niveles, mediante la librería *NetworkX*[2] desarrollada en python. Estos árboles cuentan con un nivel h de 1 a 4, y un factor de ramificación[3] s que oscila entre 3 y 8. El nombre de las instancias está formado por su valor de s y h (árbol$s$-h).

4.3 Condiciones experimentales

Los algoritmos propuestos fueron codificados en lenguaje C versión 4.6.3 y compilados con *gcc* usando la bandera de optimización *-O3*. Fueron ejecutados de manera secuencial en un cluster equipado con 4 procesadores Xeon X5650 de 6 núcleos cada uno a 2.66 GHz, 32 GB de RAM y sistema operativo Linux.

4.4 Sintonización de parámetros

La optimización de los parámetros de entrada es una tarea importante en el área de diseño de algoritmos. En la literatura especializada se encuentran reportados diversos métodos que ayudan en la búsqueda de la mejor combinación de estos valores (Adenso-Diaz y Laguna, 2006; Landgraaf *et al.*, 2007; Gunawan *et al.*, 2011). En este trabajo se emplea una técnica basada en pruebas de interacción combinatoria, la cual ha sido ampliamente utilizada en la literatura (Cohen *et al.*, 1996).

Las pruebas de interacción combinatoria reducen de manera significativa el número de pruebas (experimentos) necesarias para determinar cuál es la mejor combinación de parámetros de un algoritmo. En vez de ejecutar de manera exhaustiva todas las combinaciones de los parámetros de entrada en el algoritmo, solo se consideran las interacciones de t (o menos) parámetros de entrada

[2]La librería es gratuita y se encuentra disponible en la siguiente dirección: `http://networkx.lanl.gov/`.
[3]El factor de ramificación de un árbol es el nodo con mayor número de sucesores.

mediante la creación de pruebas de interacción incluyendo al menos una vez todas las t-combinaciones entre estos parámetros y sus valores.

Los Arreglos de Cobertura (*Covering Arrays*) son diseños combinatorios utilizados para representar conjuntos de pruebas de interacción. Un arreglo de cobertura, CA($N; t, k, v$), de tamaño N, fuerza t, grado k y orden v es una matriz de $N \times k$ con v símbolos de manera que para cada submatriz de $N \times t$ se tiene, al menos una vez, todas las t-tuplas de valores.

Los Arreglos de Cobertura son utilizados para representar conjuntos de pruebas de interacción de la siguiente manera. Un conjunto de pruebas para un algoritmo que cuenta con k parámetros de entrada de manera que cada uno de estos puede tomar v valores, es una matriz de $N \times k$ donde cada fila representa un caso de prueba. Cada columna representa un parámetro de entrada y el valor en la columna representa una configuración en particular.

En la práctica, los parámetros de entrada de un algoritmo no tienen exactamente la misma cantidad de valores. Para tratar esta limitante se hace uso de los Arreglos de Cobertura con Niveles Mezclados (*Mixed Level Covering Arrays*, MCA). Un MCA($N; t, k, (v_1, v_2, v_3, ..., v_k)$) es una matriz de $N \times k$ con v símbolos ($v = \sum_{i=1}^{k} v_i$), donde cada columna i $(1 \leq i \leq k)$ de la matriz contiene solo los elementos del conjunto S_i, donde $|S_i| = v_i$. Esta matriz tiene la propiedad de que las filas de cada $N \times t$ submatriz contiene todas las t-tuplas de valores para las t columnas al menos una vez. El conjunto v del MCA también puede ser representado en notación exponencial.

Símbolo	δ	**Tamaño de la lista tabú**	$maxS$	$maxF$	ρ	α	β	γ	***itSinMejora***	***Diver***
0	0	3	50	40	1	0.8	0.4	0.2	5	$\mathcal{D}_2(\varphi)$
1	0.2	5	60	50	2	0.9	0.5	0.3	10	$\mathcal{D}_3(\varphi)$
2	0.5	dinámica	70	60	3	—	—	—	—	—
3	0.8	—	—	—	—	—	—	—	—	—

Tabla 4.1: Valores de cada parámetro de entrada para el algoritmo BT-ABC.

El algoritmo BT-ABC cuenta con $k = 10$ parámetros de entrada: probabilidad de intervención de las funciones de vecindad δ, tamaño de lista tabú, número máximo de intervenciones de la función de diversificación suave $max\mathcal{S}$, número máximo de intervenciones de las funciones de diversificación fuerte $max\mathcal{F}$, fuerza ρ, probabilidad de aceptación α para los nodos de la lista de críticos $\mathcal{C}_1(\varphi)$, probabilidad de aceptación β para seleccionar el mejor nodo de la lista de aspirantes $\mathcal{A}(u)$, porcentaje γ que representa cierta cantidad de nodos del grafo, iteraciones sin mejora *itSinMejora* y tipo de diversificación *Diver*. Los valor posibles para cada parámetro se aprecian en la Tabla 4.1. Es importante mencionar que tanto en la etapa de sintonización como en la de experimentación se utiliza un valor de $\theta = 3$, que es el tamaño mínimo de la lista tabú dinámica.

Algunos de los valores de parámetros que se muestran en la Tabla 4.1 se obtuvieron al sintonizarse las funciones de vecindad y diversificación ($\mathcal{N}$ y $\mathcal{D}$) mediante la herramienta *BONESA* (Haasdijk *et al.*, 2012) en una etapa previa. Esta herramienta implementa un modelo iterativo basado en un procedimiento de búsqueda similar a la optimización secuencial de parámetros. Mediante un algoritmo evolutivo y desarrollando una búsqueda guiada, busca soluciones (combinación de parámetros) de buena calidad que minimicen (en nuestro caso) la función objetivo de un algoritmo externo. No se continuó utilizando este enfoque debido al largo tiempo de ejecución que presentaba.

El arreglo de cobertura con niveles mezclados MCA($120; 4, 10, 4^1, 3^4, 2^5$) que se utilizó en esta etapa de sintonización (ver Tabla 4.2), fue construido usando el Algoritmo Memético reportado por Rodriguez-Tello y Torres-Jimenez (2010). Esta matriz establece 120 casos de prueba, a diferencia de un enfoque exhaustivo en que se tendrían que hacer $4 \cdot 3^4 \cdot 2^5 = 10368$ pruebas.

La sustitución de los valores del MCA por los valores de los parámetros de entrada de BT-ABC, se puede hacer de manera sencilla. Simplemente es necesario reemplazar los símbolo de cada fila por el valor que le corresponde según el parámetro. Por ejemplo, para la fila uno (que corresponde al parámetro δ de la Tabla 4.1) el 0 mantiene su valor, pero el 1 es sustituido por 0.2, el 2 por 0.5 y el 3 por 0.8. Esta acción es aplicada a todas las filas de la Tabla 4.2 que representan los k parámetros de entrada del algoritmo BT-ABC, mencionados anteriormente.

1 3 1 2 0 0 3 0 0 3 1 0 3 0 0 1 3 3 0 1 2 3 1 1 2 0 1 2 0 2 3 2 3 0 3 2 2 0 1 3 1 3 3 1 3 0 2 0 3 0 2 2 3 0 1 0 1 3 0 2
0 0 0 2 2 2 2 1 1 2 0 0 0 1 1 2 2 2 1 2 2 0 0 0 1 0 2 0 0 2 0 1 1 0 1 0 0 0 2 1 1 1 0 2 0 0 1 0 2 1 1 0 1 1 0 2 1 2 0 1
1 1 1 2 0 0 1 2 2 0 2 2 0 0 1 0 1 0 2 2 0 0 2 0 1 0 2 1 2 0 2 0 2 2 0 2 0 1 2 0 0 2 2 1 0 1 2 0 2 0 0 1 0 1 0 1 2 0 0 2
0 2 1 2 1 2 1 0 1 2 0 0 1 0 1 0 2 1 1 0 0 2 2 2 1 1 1 2 0 1 2 2 1 2 2 1 2 0 2 1 1 2 2 1 0 2 0 1 0 1 0 0 1 2 1 2 0 2 2 1
2 0 0 1 2 1 2 0 2 0 2 2 0 1 1 1 1 1 2 0 1 2 0 1 2 1 2 2 1 0 0 0 2 2 1 1 1 0 1 0 0 0 1 1 1 1 1 1 1 0 2 2 2 0 2 2 1 2 0 0
0 1 0 0 0 1 1 1 0 1 1 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 1 0 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 0 1 0
0 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 1 1 0 1 0 1 1 0 1 0 0 1 0 1 0 1 1 1 0 0 0 0
1 1 0 1 0 0 0 1 0 0 0 0 0 1 1 0 0 1 1 1 1 1 0 1 0 0 0 1 0 1 0 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0 0 1 0 0 1 0 0 1 0 1 1 0 1 1
0 1 1 0 0 1 1 1 1 0 1 0 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 0 1 0 0 0 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 1 1 0
1 0 1 1 1 1 0 0 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 0 0 1 0 0 1 0 0 1

(a) Casos de prueba del 1 al 60

1 0 2 1 3 2 3 1 1 3 2 2 1 1 0 1 0 2 2 3 2 1 3 0 3 1 2 0 3 2 1 1 3 0 2 2 3 0 0 3 1 2 2 0 1 0 2 3 1 3 2 1 0 3 1 2 3 2 0 0
1 2 0 2 0 1 1 1 2 1 0 1 2 1 0 0 0 0 1 0 2 1 2 2 1 1 1 1 2 2 2 1 1 2 0 0 1 0 2 1 1 2 2 2 1 1 2 0 0 2 0 0 1 0 0 2 2 1 1 2
2 0 0 1 1 0 1 0 0 2 1 1 0 1 2 2 1 2 1 2 0 1 2 2 1 2 0 0 2 1 1 0 2 1 1 0 1 0 1 0 0 1 1 2 1 1 2 1 1 0 2 0 2 2 1 2 1 2 0 2
2 0 0 0 0 1 0 0 2 1 1 0 1 2 1 1 1 0 2 1 2 0 1 1 1 1 1 0 2 0 2 1 0 0 0 1 2 0 1 0 2 1 2 2 1 0 0 1 0 0 2 0 2 0 2 1 0 2 2 0
2 0 0 2 2 1 1 2 2 1 0 0 0 1 0 1 2 2 1 2 2 0 0 1 0 0 0 1 2 2 0 1 2 1 1 2 2 2 0 0 0 1 0 0 2 2 0 1 1 2 0 0 1 0 2 2 0 2 2 2
0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 0 1 1 1 0 0 1 1 0 1 1 1 1 0 1
1 1 1 1 0 0 1 1 0 1 0 0 1 0 1 0 0 0 1 1 1 1 1 1 0 1 1 0 1 0 0 1 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 0 1
1 1 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 1 1 1 0 0 1 1 0 0 0 0 1 1 1 0 0 1 1 1 1 1 0 0 0 0 0 0 1 0 0 1 0 1 1 1 1 1 0 1 1 0 0 0
0 0 0 1 0 0 1 1 1 0 0 1 1 1 0 1 1 0 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1 1 0 1 0 1
0 0 1 0 0 1 0 1 1 1 0 1 0 1 0 0 1 1 0 1 0 0 1 0 1 0 0 1 1 1 0 0 1 1 1 1 1 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 1 0 0 1 1 0 1

(b) Casos de prueba del 61 al 120

Tabla 4.2: Matriz transpuesta del MCA($120; 4, 10, 4^1, 3^4, 2^5$).

Grafo	n	m	den	B_C^*
ciclo35	35	35	0.059	1
ciclo40	40	40	0.051	1
malla5x7	35	58	0.097	5
malla5x8	40	67	0.086	5
camino35	35	34	0.057	1
camino40	40	39	0.050	1
árbolB33	33	34	0.064	4
árbolB35	35	34	0.057	4
rnd35-1000-15	35	260	0.437	—
rnd35-1000-5	35	95	0.160	—
rnd40-1000-15	40	300	0.385	—
rnd40-1000-5	40	99	0.127	—
ibm32	32	90	0.181	—
bcspwr01	39	46	0.062	—

Tabla 4.3: Características del conjunto de instancias usadas en la etapa de experimentación. Consiste en 14 instancias de diversos tipos de grafos.

El algoritmo BT-ABC fue ejecutado de manera independiente para los 120 casos de prueba, resolviendo un subconjunto de 14 instancias representativas de todos los tipos de grafos descritos en la Sección 4.2. Estos grafos son los de mayor grado y número de nodos, por lo que se pudieran considerar como los más difíciles de resolver. En la Tabla 4.3 se muestran estas instancias, en donde la primera columna indica el nombre del grafo, las siguientes dos columnas presentan el número de nodos n y de arcos m, después se muestra su densidad *den* y por último el valor B_C^* óptimo. Para las instancias donde no se conoce su valor óptimo se utiliza el símbolo —.

En la Figura 4.1 se observan los resultado obtenidos de todo el conjunto de casos de prueba. El eje de las abscisas presenta los 120 casos de prueba, mientras que en el eje de las ordenadas se indica el tiempo promedio de ejecución del algoritmo y el valor del $B_C(G, \varphi)$ encontrado promedio de todas las instancias.

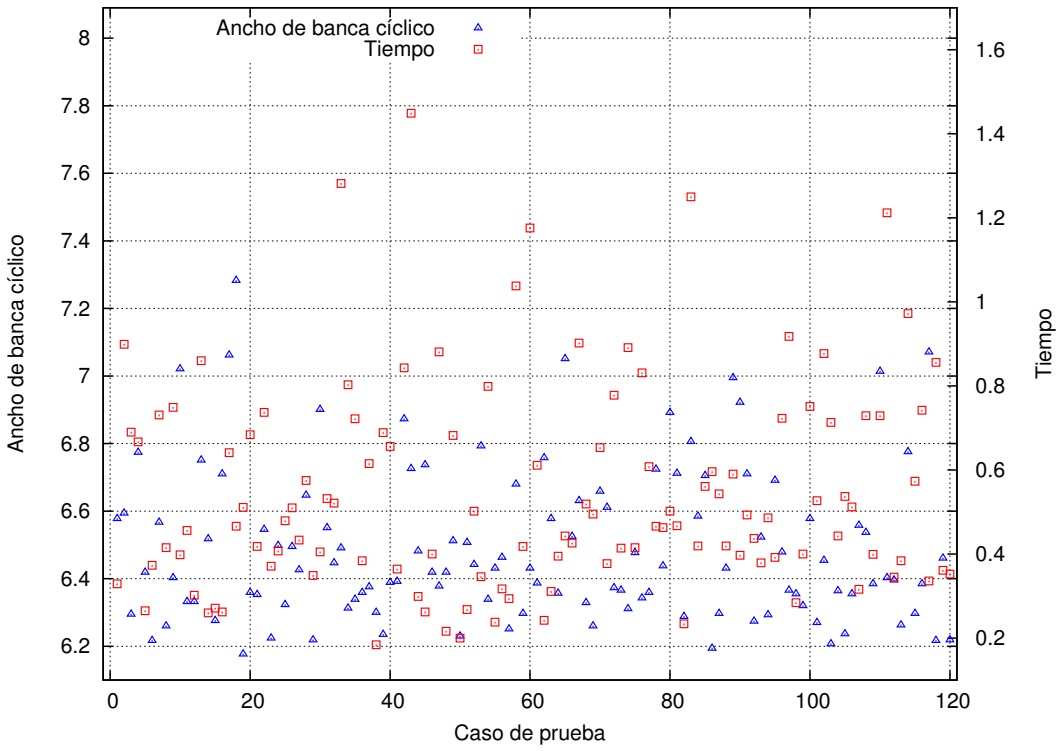


Figura 4.1: Resultados obtenidos en la etapa de sintonización para los 120 casos de prueba resolviendo un conjunto de 14 grafos.

La Tabla 4.4 muestra los resultados obtenidos de los mejores 10 casos de prueba. En la primera y segunda columna se indica el número de caso y la configuración ganadora, las siguientes dos columnas muestran el tiempo y el B_C promedio obtenido con las intancias mencionadas. Los resultados fueron ordenados contemplando el valor del $B_C(G, \varphi)$ promedio primeramente y después el tiempo de ejecución promedio. El caso ganador indica que para tener un mejor desempeño del algoritmo BT-ABC se deben utilizar los siguientes valores: $\delta = 0$, tamaño de lista tabú *dinámica*, $max\mathcal{S} = 70$, $max\mathcal{F} = 50$, $\rho = 3$, $\alpha = 0.9$, $\beta = 0.4$, $\gamma = 0.3$, $itSinMejora = 10$ y $Diver = \mathcal{D}_2(\varphi)$. Esta configuración fue utilizada para realizar la etapa de experimentación final, la cual se detalla a continuación.

Núm.	Caso de prueba	$\mathcal{T}$-prom	B_C-prom
19	**0121210111**	**0.511**	**6.177**
86	1121011010	0.596	6.194
103	2212011011	0.713	6.207
6	0202111011	0.373	6.217
118	2122210011	0.856	6.217
29	0020110011	0.349	6.219
120	0220211011	0.352	6.219
23	1022010001	0.371	6.224
50	0101011001	0.200	6.230
39	1222111010	0.689	6.235

Tabla 4.4: Resultados de los 10 mejores casos de prueba del proceso de sintonización.

4.5 Experimentación final

El algoritmo RP-ABC descrito en la Sección 3.1 y el algoritmo BT-ABC planteado en la Sección 3.2 se pusieron a prueba utilizando las instancias mencionadas en la Sección 4.2. Se contrasta el resultado promedio de 31 corridas del algoritmo metaheurístico contra una ejecución del algoritmo exacto.

Los resultados de la experimentación final son mostrados en las Tablas 4.5, 4.6, 4.7 y 4.8 en donde las primeras cuatro columnas indican las características de los grafos: nombre, número de nodos (n), número arcos (m) y densidad (den = $2m/n(n-1)$). Si se conoce el valor del ancho de banda cíclico óptimo (B_C^*), la quinta columna lo menciona. En las siguientes columnas se presentan los resultados alcanzados por ambos algoritmos. Para el algoritmo RP-ABC se indica el número de soluciones parciales evaluadas ($\#\varphi'$), el tiempo de ejecución del algoritmo en segundos ($\mathcal{T}$) y el ancho de banda cíclico (B_C) alcanzado. Para el algoritmo BT-ABC se reporta el número promedio de soluciones parciales evaluadas ($\#\varphi'$-prom), el tiempo de ejecución promedio ($\mathcal{T}$-prom) y el mejor ancho de banda cíclico (B_C) encontrado. También, se reporta el valor promedio del ancho de banda cíclico (B_C-prom), así como su desviación estándar (desv). En la última columna (dif) se encuentra la diferencia de los valores de ancho de banda cíclico B_C encontrados por los algoritmos comparados. Un valor positivo indica que el algoritmo BT-ABC encontró una mejor solución, de manera contraria si el algoritmo RP-ABC encontró una mejor solución el valor será negativo.

Para el algotimo exacto se estableció como máximo un tiempo de ejecución de 36 horas. Este parámetro se consideró como suficiente para que el algoritmo resuelva los grafos que están a su alcance.

De manera gráfica también se muestran los resultados obtenidos en esta etapa de experimentación final en las Figuras 4.2, 4.3, 4.4 y 4.5. Para cada instancia (eje de las abscisas) se registra el tiempo de cómputo que le tomó al algoritmo resolverla o la calidad de la solución encontrada (eje de las ordenadas).

4.5.1 Resultados para grafos ordinarios

Para este conjunto de prueba se realizó una comparativa de los límites reportados por Klerk *et al.* (2011) con la calidad de las soluciones encontradas por los enfoques propuestos. La Tabla 4.5 muestra los resultados obtenidos con esta colección de grafo ordinarios. Las instancias que cuentan con el símbolo $\star$ en la columna B_C, indican que RP-ABC no pudo explorar por completo todo el

espacio de búsqueda en el tiempo establecido por lo que se reporta la mejor solución encontrada.

Los resultados muestran que RP-ABC solo pudo resolver el 80 % del conjunto, igualando en esos casos el límite teórico reportado. A excepción de las instancias tipo ciclo, este enfoque no pudo resolver los grafos de orden $n > 35$.

					RP-ABC			BT-ABC					
Grafo	n	m	den	B_C^*	$\#\varphi'$	$\mathcal{T}$	B_C	$\#\varphi'$-prom	$\mathcal{T}$-prom	B_C	B_C-prom	desv	dif
ciclo20	20	20	0.11	1	2331	0.01	1	515.97	0.01	1	1.00	0.00	0
ciclo25	25	25	0.08	1	12900	0.12	1	1872.97	0.01	1	1.00	0.00	0
ciclo30	30	30	0.07	1	88516701	1007.90	1	6371.81	0.01	1	1.00	0.00	0
ciclo35	35	35	0.06	1	465326343	4184.05	1	5139.90	0.01	1	1.00	0.00	0
ciclo40	40	40	0.05	1	397409592	4996.08	1	18442.45	0.07	1	1.00	0.00	0
malla5x4	20	31	0.16	4	123230	0.82	4	46557.65	0.01	4	4.00	0.00	0
malla5x5	25	40	0.13	5	2674635	29.32	5	45268.65	0.11	5	5.13	0.34	0
malla5x6	30	49	0.11	5	144312826	3016.46	5	46326.13	0.16	5	5.00	0.00	0
malla5x7	35	58	0.10	5	1641	—	$\star$ 9	49265.06	0.23	5	5.00	0.00	4
malla5x8	40	67	0.09	5	175446	—	$\star$ 10	51567.00	0.31	5	5.00	0.00	5
camino20	20	19	0.10	1	695	0.01	1	1052.26	0.01	1	1.00	0.00	0
camino25	25	24	0.08	1	19997	0.07	1	1364.10	0.01	1	1.00	0.00	0
camino30	30	29	0.07	1	101821	0.60	1	9642.77	0.02	1	1.00	0.00	0
camino35	35	34	0.06	1	74072	0.83	1	2956.03	0.01	1	1.00	0.00	0
camino40	40	39	0.05	1	349320	—	$\star$ 9	10926.45	0.04	1	1.00	0.00	8
árbolB21	21	20	0.10	3	147232	0.47	3	47666.77	0.10	3	3.00	0.00	0
árbolB25	25	24	0.08	4	223355577	763.27	4	48466.29	0.12	4	4.00	0.00	0
árbolB31	31	30	0.06	4	707968183	3552.96	4	49733.32	0.15	4	4.00	0.00	0
árbolB33	33	34	0.06	4	2248823425	10965.57	4	49939.61	0.16	4	4.00	0.00	0
árbolB35	35	34	0.06	4	427025884	—	$\star$ 6	50463.65	0.17	4	4.00	0.00	2
Promedio					235321092.55		3.60	27176.94	0.08	2.65	2.65	0.01	

Tabla 4.5: Resultados obtenidos por los algoritmos RP-ABC y BT-ABC sobre el conjunto de grafos ordinarios.

En terminos generales, BT-ABC pudo resolver con facilidad estos grafos ya que en todos los casos encontró la solución óptima. Respecto a la desviación estándar, este algoritmo muestra una solidez en su búsqueda pues en la mayoría de los casos obtuvo un valor desv $= 0.0$, a excepción de la

instancia *malla5x5*. Considerando el tiempo de ejecución, resolver una instancia le llevó a lo más 0.31 segundos (tiempo promedio), mientras que al enfoque exacto explorar todo el espacio de búsqueda del *árbolB33* le tomó más de tres horas. Esto es evidente ya que los paradigmas implementados son distintos. Por una parte, el algoritmo exacto nos garantiza la exploración completa del espacio de búsqueda, mientras que un algoritmo metaheurístico pudiera reducir su tiempo de ejecución visitando solo ciertas áreas.

La Figura 4.2 muestra el tiempo de ejecución de ambos algoritmos. La gráfica indica para cada instancia del conjunto (abscisas), el tiempo de cómputo en segundos que le tomó para concluir la búsqueda usando una escala $\log_{10}$ (ordenadas). En esta gráfica se observa como la diferencia del tiempo es menor para los grafos con menos nodos que para los grafos de mayor orden. Este efecto se puede observar en las instancias *malla5x8* y *camino20*.

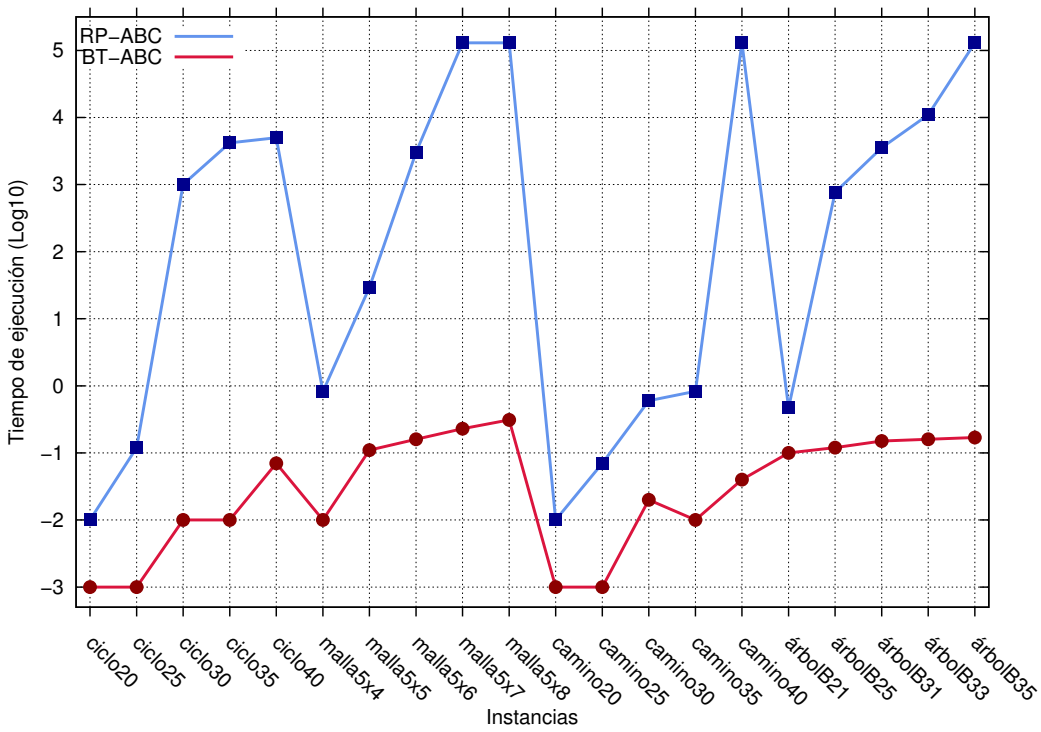


Figura 4.2: Comparativa entre el algoritmo RP-ABC y BT-ABC respecto al tiempo de ejecución sobre el conjunto de grafos ordinarios.

4.5.2 Resultados para grafos Harwell-Boeing

En la literatura especializada no se encuenta reportada ninguna cota sobre el conjunto Harwell-Boeing para el problema de MABC. Tampoco los estudios teóricos existentes pueden determinar el valor de la solución óptima.

				RP-ABC			BT-ABC					
Grafo	n	m	den	$\#\varphi'$	$\mathcal{T}$	B_C	$\#\varphi'$-prom	$\mathcal{T}$-prom	B_C	B_C-prom	desv	dif
pores_1	30	103	0.23	719525	10.35	7	45705.48	0.33	7	7.00	0.00	0
ibm32	32	90	0.18	1026552013	—	$\star$ 9	47288.03	0.49	9	9.00	0.00	0
bcspwr01	39	46	0.06	2188645742	—	$\star$ 9	51572.16	0.22	5	5.00	0.00	4
bcsstk01	48	176	0.15	494368750	—	$\star$ 16	51254.00	0.82	12	12.00	0.00	4
bcspwr02	49	59	0.05	3062428	—	$\star$ 14	50624.68	0.26	7	7.00	0.00	7
curtis54	54	124	0.08	2168	—	$\star$ 18	34369.48	0.36	8	8.74	0.86	10
will57	57	127	0.08	20170264	—	$\star$ 18	52192.03	0.47	6	6.00	0.00	12
impcol_b	59	281	0.16	75056537	—	$\star$ 23	49217.00	1.41	17	17.00	0.00	6
ash85	85	219	0.06	153	—	$\star$ 34	54126.35	1.27	9	9.71	0.46	25
nos4	100	247	0.05	95766189	—	$\star$ 39	56940.19	1.59	10	10.00	0.00	29
dwt_234	117	162	0.02	40132	—	$\star$ 37	61831.65	1.34	12	16.10	1.66	25
bcspwr03	118	179	0.02	2726	—	$\star$ 41	58391.16	1.29	11	15.10	2.62	30
bcsstk06	420	3720	0.04	264054	—	$\star$ 192	49420.26	37.13	108	109.00	0.68	84
bcsstk07	420	3720	0.04	264054	—	$\star$ 192	49933.29	40.28	108	109.58	0.81	84
impcol_d	425	1267	0.01	37637	—	$\star$ 181	72294.19	19.21	39	56.23	14.97	142
can_445	445	1682	0.01	5576	—	$\star$ 192	48620.68	19.67	48	129.45	40.56	144
494_bus	494	586	0.00	93047	—	$\star$ 197	85572.39	16.40	56	62.13	4.43	141
dwt_503	503	2762	0.03	17762	—	$\star$ 224	52487.87	32.50	46	114.10	54.35	178
sherman4	546	1341	0.01	201675	—	$\star$ 217	64709.00	26.25	29	57.77	47.66	188
dwt_592	592	2256	0.02	33114	—	$\star$ 252	55072.26	33.09	33	91.97	79.94	219
662_bus	662	906	0.01	477171	—	$\star$ 246	86893.35	29.53	83	90.03	5.37	163
nos6	675	1290	0.01	483807	—	$\star$ 241	58089.42	25.38	23	93.00	70.25	218
685_bus	685	1282	0.01	164991	—	$\star$ 282	74216.32	30.50	77	101.77	10.48	205
can_715	715	2975	0.03	100034	—	$\star$ 312	59603.00	51.96	60	70.23	19.53	252
Promedio				162772064.54		124.70	57101.01	15.49	34.29	50.33	14.77	

Tabla 4.6: Resultados obtenidos por los algoritmos RP-ABC y BT-ABC sobre el conjunto Harwell Boeing.

Los resultados obtenidos con estos grafos se aprecian en la Tabla 4.6. RP-ABC solo pudo resolver la instancía *pores_1*, la cual es de orden $n = 30$. Para el resto del conjunto que podría estar a su alcance ($n \leq 40$), se quedó a máximo cuatro unidades de la mejor solución encontrada por el

algoritmo metaheurístico.

En el caso de BT-ABC la calidad de las soluciones encontradas es mayor. Sin embargo, para los grafos del subconjunto *HBg* y algunos del *HBp* se observa una desviación estándar diferente de cero. Aquí se presentó el tiempo promedio de ejecución del algoritmo más largo de toda la experimentación, que fue de $\mathcal{T}$-prom $= 51.96$ en la instancia *can_715*.

En la Figura 4.2 se contrasta la calidad de las soluciones encontradas por los algoritmos RP-ABC y BT-ABC. La gráfica muestra en el eje de las ordenadas el mejor valor del B_C encontrado, mientras que en el eje de la abscisas se indica a que instancia corresponde. Claramente se observa que la exploración que realizó RP-ABC no fue suficiente para encontrar soluciones de igual o mejor calidad que las encontradas por BT-ABC. Conforme crece el valor de n y m en los grafos la diferencia entre los valores del B_C es más grande.

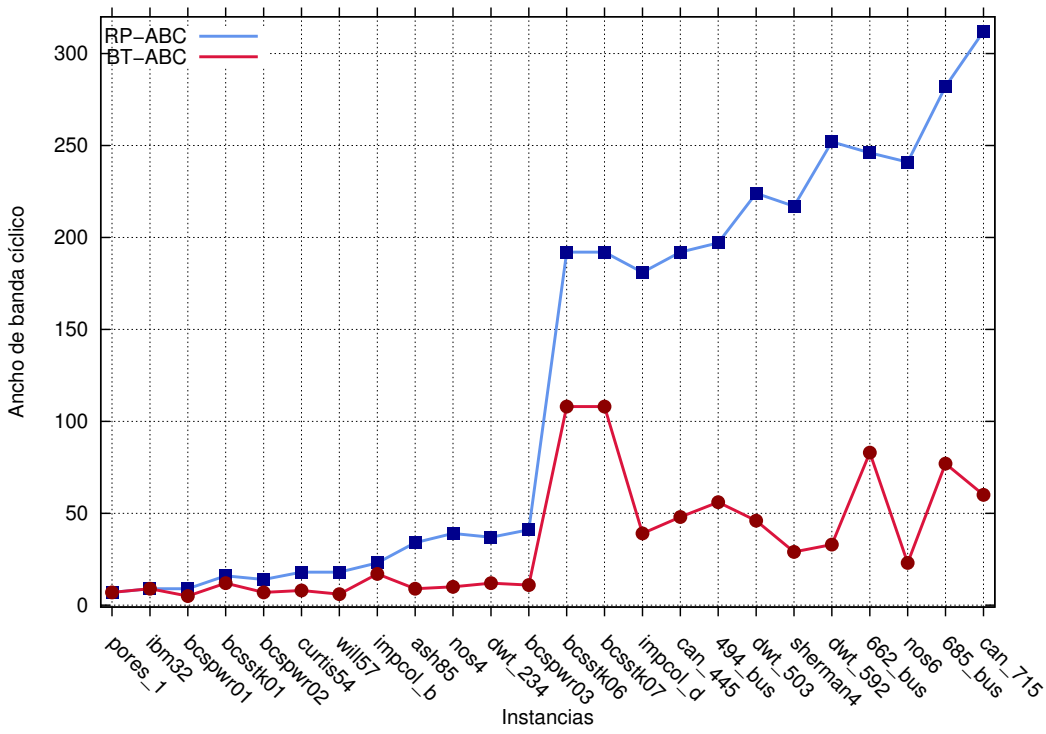


Figura 4.3: Comparativa entre el algoritmo RP-ABC y BT-ABC respecto a la calidad de solución encontrada sobre el conjunto Harwell Boeing.

4.5.3 Resultados para grafos aleatorios

Con los estudios teóricos analizados hasta el momento sobre el problema de MABC, no se ha encontrado ninguna fórmula que ayude a calcular el límite inferior de estos grafos. Sin embargo, en los resultados que se muestran en la Tabla 4.7 se indican algunas soluciones óptimas encontradas.

				RP-ABC			BT-ABC					
Grafo	n	m	den	$\#\varphi'$	$\mathcal{T}$	B_C	$\#\varphi'$-prom	$\mathcal{T}$-prom	B_C	B_C-prom	desv	dif
rnd20-100-5	20	52	0.27	1345278	7.72	6	44699.65	0.25	6	6.00	0.00	0
rnd20-1000-10	20	96	0.50	2407146	23.81	8	14196.77	0.22	8	8.00	0.00	0
rnd20-1000-15	20	146	0.76	26	0	9	4181.10	0.13	9	9.00	0.00	0
rnd20-1000-5	20	62	0.32	1930927	13.77	7	43790.87	0.32	7	7.00	0.00	0
rnd25-100-5	25	58	0.19	13463789	113.70	6	46520.10	0.24	6	6.00	0.00	0
rnd25-1000-10	25	124	0.41	93511158	1836.24	10	45132.10	0.93	10	10.00	0.00	0
rnd25-1000-15	25	185	0.61	3433711	79.80	11	42374.00	1.87	11	11.00	0.00	0
rnd25-1000-5	25	53	0.17	2276856	14.36	6	46795.35	0.21	6	6.00	0.00	0
rnd30-100-5	30	72	0.16	2934189571	39582.96	8	47085.68	0.38	8	8.00	0.00	0
rnd30-1000-10	30	140	0.32	136178	—	$\star$ 11	45434.52	0.97	11	11.00	0.00	0
rnd30-1000-15	30	216	0.49	19074053	601.79	12	45194.39	2.06	12	12.84	0.37	0
rnd30-1000-5	30	71	0.16	9221187860	117186.81	8	47280.00	0.34	8	8.00	0.00	0
rnd35-100-5	35	86	0.14	1350829697	—	$\star$ 9	48241.42	0.46	9	9.00	0.00	0
rnd35-1000-10	35	171	0.28	275642461	—	$\star$ 12	46429.39	0.99	12	12.00	0.00	0
rnd35-1000-15	35	260	0.43	156971419	11322.92	14	46346.00	2.06	14	14.17	0.46	0
rnd35-1000-5	35	95	0.16	204724692	—	$\star$ 10	48242.06	0.52	9	9.00	0.00	1
rnd40-100-5	40	109	0.14	690061231	—	$\star$ 12	48920.55	0.60	11	11.00	0.00	1
rnd40-1000-10	40	204	0.26	40874091	—	$\star$ 15	45577.13	1.36	14	14.52	0.51	1
rnd40-1000-15	40	300	0.38	6274213	—	$\star$ 16	46102.52	2.23	16	16.29	0.46	0
rnd40-1000-5	40	99	0.12	223905339	—	$\star$ 13	48264.48	0.56	11	11.23	0.43	2
Promedio				762111984.80		8.75	42540.40	0.84	9.90	10.00	0.09	

Tabla 4.7: Resultados obtenidos por los algoritmos RP-ABC y BT-ABC sobre el conjunto de grafos aleatorios.

En la experimentación con este conjunto RP-ABC resolvió más del 60 % de estas instancias en un tiempo menor a 11 horas, a excepción de *rnd30-1000-5*. Estos grafos resueltos son de orden $n \leq 30$ en su mayoría. Por otra parte, para aquellos en donde no se pudo explorar todo el espacio de búsqueda la calidad de la solución es cercana a la encontrada por BT-ABC, con una diferencia

máxima de dos unidades. Esto pudiera indicar que las soluciones encontradas por BT-ABC están muy cercanas al óptimo.

BT-ABC pudo igualar las cotas encontradas por el algoritmo exacto, y mejorar 4 de las instancias donde RP-ABC no pudo explorar todo el espacio de búsqueda. Para este conjunto se vuelve a presentar una consistencia en el proceso de búsqueda, ya que en la mayoría de los casos se cuenta con una desviación estándar desv $= 0.0$.

Se presenta de manera gráfica una comparativa de los enfoques propuestos en la Figura 4.4, indicando la calidad de la soluciones encontradas. En esta imagen se puede ver que para el 80 % de los casos ambos algoritmos encuentran una solución con el mismo valor de B_C, incluso aunque en el enfoque exacto no se haya explorado todo el espacio de búsqueda. Para el otro 20 %, la calidad de las soluciones son muy cercanas, a una diferencia máxima de dos unidades.

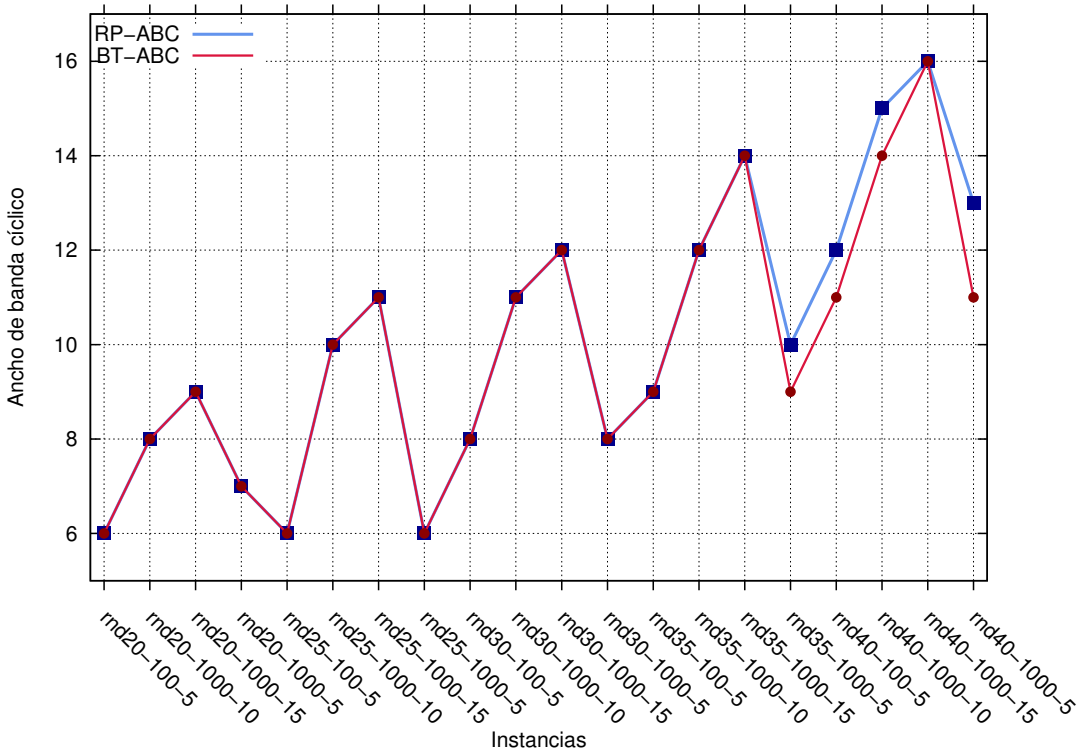


Figura 4.4: Comparativa entre el algoritmo RP-ABC y BT-ABC respecto a la calidad de solución encontrada sobre el conjunto de grafos aleatorios.

4.5.4 Resultados para árboles

Este conjunto de prueba fue resuelto únicamente con BT-ABC debido al tamaño de las instancias. En los conjuntos anteriores se observó que el enfoque exacto por lo general solo completa la búsqueda cuando el grafo es de orden $n \leq 35$. El límite inferior teórico de estas instancias se calculó mediante las ecuaciones descritas en la Sección 2.3.2. Cabe mencionar que en la Tabla 4.8 de resultados, la columna d representa el diámetro del grafo y la columna r su radio.

Los resultados muestran que en más de un 64 % de las instancias, el algoritmo encuentra la solución óptima, y en la mayoria de esos casos con una desviación estándar cercana a cero. En las instancias donde no se llegó al valor mínimo, se tuvo una diferencia no mayor a cinco unidades. Lo que indica que la búsqueda se encuentra muy cerca de la solución óptima.

							BT-ABC					
Grafo	n	m	den	d	r	B_C*	$\#\varphi'$-prom	$\mathcal{T}$-prom	B_C	B_C-prom	desv	dif
árbol3-1	13	12	0.15	4	2	3	42354.16	0.06	3	3.00	0.00	0
árbol3-2	40	39	0.05	6	3	7	48569.81	0.18	7	7.00	0.00	0
árbol3-3	121	120	0.01	8	4	15	54253.52	0.86	16	16.00	0.00	-1
árbol3-4	364	363	0.00	10	5	37	65603.55	5.98	39	39.48	0.51	-2
árbol4-1	21	20	0.09	4	2	5	44639.48	0.09	5	5.00	0.00	0
árbol4-2	85	84	0.02	6	3	14	51506.52	0.53	14	14.48	0.51	0
árbol4-3	341	340	0.00	8	4	43	64991.55	5.44	45	45.00	0.00	-2
árbol5-1	31	30	0.06	4	2	8	44798.35	0.13	8	8.00	0.00	0
árbol5-2	156	155	0.01	6	3	26	54991.52	1.44	26	26.84	0.37	0
árbol5-3	781	780	0.00	8	4	98	73364.03	24.15	103	103.45	0.51	-5
árbol6-1	43	42	0.04	4	2	11	46057.55	0.27	11	11.00	0.00	0
árbol6-2	259	258	0.00	6	3	43	60973.19	3.61	44	44.00	0.00	-1
árbol7-1	57	56	0.03	4	2	14	47146.74	0.35	14	14.00	0.00	0
árbol8-1	73	72	0.02	4	2	18	47578.52	0.50	18	18.00	0.00	0
Promedio							53344.89	3.11	25.21	25.38	0.14	

Tabla 4.8: Resultados obtenidos por el algoritmo BT-ABC sobre el conjunto de árboles.

Otro punto a resaltar es el poco tiempo de ejecución que necesita el algoritmo para encontrar una buena solución, pues en el caso del grafo de mayor orden ($n = 781$) tardó un promedio de 24.15

segundos.

En la Figura 4.5 se muestra una comparativa del límite inferior teórico calculado y la mejor solución encontrada por BT-ABC. En ella podemos apreciar que en muchos casos el valor de B_C encontrado corresponde al cálculo realizado, y en los árboles donde no se cumple esto la diferencia es mínima.

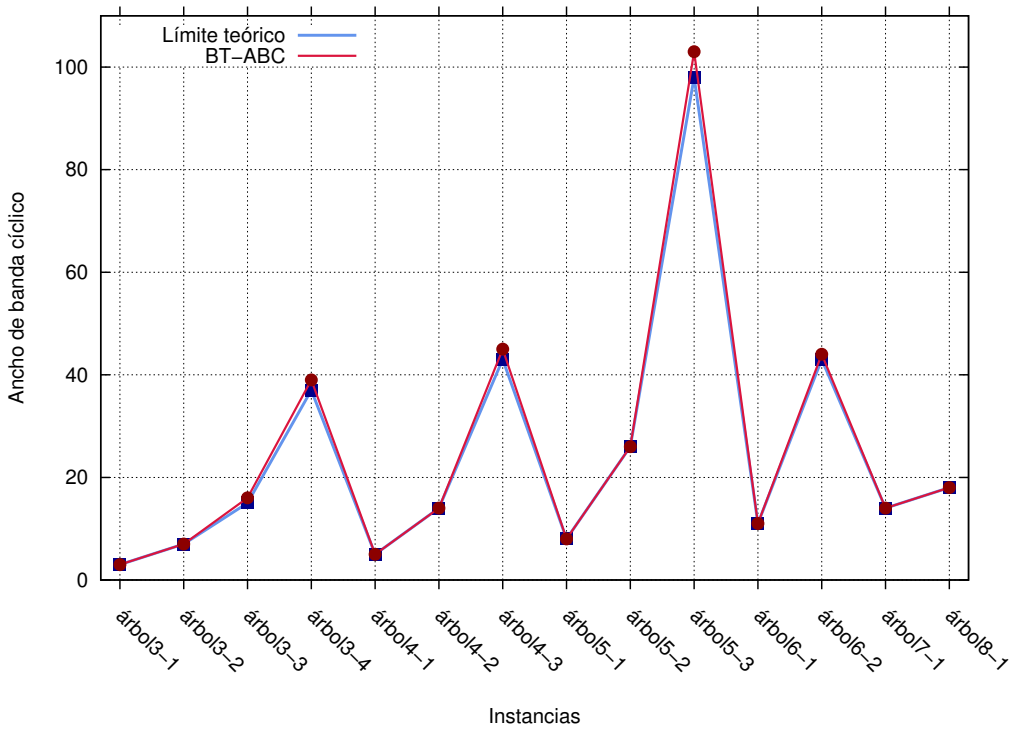


Figura 4.5: Comparativa entre la calidad de solución encontrada por el algoritmo BT-ABC y el límite inferior teórico sobre el conjunto de árboles.

4.6 Discusión de resultados

Con la información obtenida en la etapa de experimentación final, se presenta a continuación una discusión de resultados, la cual esta organizada y detallada por conjuntos de prueba.

4.6.1 Conjunto de grafos ordinarios

Para este conjunto, RP-ABC no fue capaz de resolver los grafos de orden $n > 35$, debido a que el espacio de búsqueda crece de manera factorial respecto al número de nodos lo que impacta de manera directa a este enfoque. Otro aspecto que influye es el grado de los nodos, ya que en este algoritmo la *estrategia de asignación* construye una matríz de costos con los adyacentes a la solución parcial. Por lo tanto, si se cuenta con nodos de grado alto se formarán matrices grandes, lo que se traduce en más tiempo de ejecución.

En el caso de BT-ABC, se pudo resolver con facilidad este conjunto de pruebas gracias a las funciones de vecindad que implementa, las cuales buscan reducir el $B_C(v, \varphi)$ de los nodos críticos evitando en lo posible las soluciones de igual o peor calidad. De esta manera se redujo la cantidad de soluciones a probar y se disminuyó el tiempo de ejecución del algoritmo para todas las instancias de este conjunto, contrastado con el enfoque exacto.

4.6.2 Conjunto de grafos Harwell-Boeing

El algoritmo RP-ABC solo pudo resolver una instancia de este conjunto. Esto fue debido al tamaño y al grado de los nodos presentes en estos grafos, tal como se explicó para el conjunto de grafos ordinarios.

Para todo el subconjunto HBg, se obtuvo una desviación estándar lejos de cero por parte del algoritmo BT-ABC. Lo anterior indica una variabilidad en la calidad de las soluciones entregadas por dicho algoritmo. Uno de los aspectos que influyeron fue el tamaño de los grafos, ya que en este subconjunto las instancias son de orden $n \geq 420$ y probablemente el número de iteraciones en general no fue suficiente para realizar una buena exploración en el conjunto de soluciones.

4.6.3 Conjunto de grafos aleatorios

En los resultados sobre este conjunto se observa que RP-ABC fue capaz de resolver de manera exacta el 60 % de las instancias. Y respecto al resto del conjunto (el 40 %) la calidad de las soluciones es similar a la encontrada por el algoritmo BT-ABC. Tal similitud indica que se encuentran soluciones de buena calidad desde las primeras permutaciones probadas en el espacio de búsqueda. Además, esto se presenta gracias a la variedad de soluciones que cuentan con el mismo ancho de banda cíclico.

Ya que este conjunto está formado por grafos de orden $n \leq 40$, nuevamente en los resultados de BT-ABC se muestra una desviación estándar que tiende a cero.

4.6.4 Conjunto de árboles

Al resolver este conjunto de prueba utilizando el algoritmo BT-ABC, se observó que en un 64 % de las instancias este pudo encontrar una solución con un ancho de banda cíclico $B_C(G, \varphi)$ igual al límite inferior teórico calculado. A los árboles que contaban con un número de nodos $n \geq 121$ no se les pudo encontrar una solución similiar a la cota inferior. Por lo tanto, se puede concluir que la sintonización que se realizó del algoritmo BT-ABC permite encontrar la solución óptima para árboles de orden $n \leq 73$.

Resumen del capítulo

En este capítulo se presentaron los diferentes conjuntos de instancias utilizados en la etapa experimentación, describiendo cómo se formaron o de dónde se obtuvieron. También, se establecieron las métricas utilizadas para contrastar los resultados obtenidos.

Una vez descritos los algoritmos en el capítulo anterior y aclarados las condiciones experimentales, se efectuó una etapa de sintonización de parámetros para obtener la mejor combinación de valores y lograr un mejor desempeño del algoritmo BT-ABC. Lo anterior se llevó acabo mediante el uso de

un arreglo de cobertura, el cual establece las pruebas suficientes para lograr este fin.

Por último se presentaron los resultados de la experimentación final con los cuatro conjuntos de prueba descritos. Para el analisis de los resultados se utilizaron tablas y gráficas con el objetivo de dejar en claro el comportamiento de ambos enfoques sobre las instancias.

En el siguiente capítulo se presentan las conclusiones de este trabajo de tesis, analizando los resultados obtenidos por las dos propuestas de solución. Además, se abordarán algunas posibilidades de trabajo futuro.

5

Conclusiones y trabajo futuro

El objetivo principal planteado para este trabajo de tesis (establecido en la Sección 1.3) fue cumplido satisfactoriamente. Se presentaron dos algoritmos que resuelven el problema de MABC para grafos generales. El primero fue un algoritmo de tipo exacto basado en la técnica de ramificación y poda, llamado RP-ABC. Su descripción fue dividida contemplando tres de sus componentes: la estrategia de ramificación, asignación y retroceso. La segunda propuesta fue un algoritmo de búsqueda tabú nombrado BT-ABC, que se encuentra en el grupo de las metaheurísticas. De este enfoque también se presentaron sus componentes principales que son: solución inicial, función de vecindad, lista de nodos aspirantes, función de diversificación, lista tabú y criterio de paro.

Para poder desarrollar estas propuestas fue necesario hacer un estudio del problema en cuestión, así como de otros problemas de etiquetado de grafos. De esta manera se pudieron crear componentes a la medida para realizar una búsqueda que encontrara soluciones de buena calidad. Además, se utilizó una técnica basada en pruebas de interacción combinatoria que mediante el uso de arreglos de cobertura ayudó a sintonizar los parámetros del algoritmo BT-ABC. Por último se llevó a cabo una

etapa de experimentación donde se pusieron a prueba ambos algoritmos utilizando cuatro conjuntos de instancias.

Enseguida se mencionarán las conclusiones a las que se llegaron analizando los componentes de los algoritmos, los conjuntos de instancias y los resultados obtenidos en la experimentación.

5.1 Conclusiones

Con los resultados obtenidos se concluye que los algoritmos de Ramificación y Poda (RP-ABC) y Búsqueda Tabú (BT-ABC) implementados en esta tesis son propuestas factibles para solucionar el problema de MABC en grafos generales. Por lo anterior, esta investigación ha aportado dos nuevos algoritmos al estado del arte de este problema combinatorio.

Para un conjunto total de 78 grafos: 20 grafos ordinarios, 24 instancias del conjunto Harwell-Boeing, 20 grafos aleatorios y 14 árboles, se han establecido nuevas cotas para este problema. Para un total de 42 grafos (53.84 %) se ha encontrado una solución óptima mientras que para 36 grafos (46.15 %) se encontraron soluciones cercanas a las óptimas.

El algoritmo RP-ABC en general pudo resolver instancias con $n \leq 35$ nodos para tres conjuntos de prueba, utilizando como máximo 3 horas de ejecución. La estrategia de asignación ayudó a realizar podas prematuras en el proceso de búsqueda, lo que redujo la cantidad de soluciones a probar y por lo tanto solucionar instancias de mayor orden. Gracias a esto, solo se dejó inconclusa la búsqueda en 14 instancias de las que se concideraban que estaban a su alcance resolver (de $n < 35$).

Los resultados obtenidos por el algoritmo BT-ABC muestran su rapidez de búsqueda, tardando como máximo un promedio de 51.96 segundos. La función de vecidad $\mathcal{N}_2$ que implementa un mecanismo especial de reparación de nodos críticos, y la función de diversificación $\mathcal{D}_2$ que permuta los nodos mejor etiquetados mostraron un buen desempeño en la etapa de sintonización. Por este motivo formaron parte de la experimentación final. BT-ABC pudo igualar los resultados obtenidos por el enfoque exacto, y en un 46.87 % de las instancias logró mejorarlos.

5.2 Trabajo futuro

A pesar de que en un inicio los objetivos pariculares fueron pensados para el desarrollo de un par de algoritmos que tuvieran un buen rendimiento, en el proceso se identificaron ciertos puntos que pueden ser mejorados. Esto brinda la oportunidad de seguir trabajando con la investigación realizada.

Para el algoritmo RP-ABC se propone lo siguiente. En la estrategia de asignación se mencionó que el método Húngaro ayuda a resolver el problema de asignar las etiquetas disponibles en los nodos sin etiquetas. Sin embargo, la propuesta desarrollada no contempla si existe alguna relación (arco) entre los nodos de ese mismo conjunto. Hacer este cálculo podría cambiar los valores de la matriz de costos, permitiendo hacer cortes tempranos en el proceso de búsqueda y de esta manera reducir el tiempo de ejecución. Otro aspecto que podría modificarse en esta propuesta es el incorporar un enfoque estocástico en la cálculo de la matriz de costos, lo que reduciría el tiempo de ejecución permitiendo resolver instancias con más nodos.

Ya que BT-ABC mostró una gran rapidez en la etapa de experimentación, tardando como máximo 51.96 segundos en el caso de prueba más grande. Este enfoque de solución propuesto puede ser utilizado como un operador de búsqueda local dentro de un algoritmo más sofisticado, como por ejemplo un algoritmo memético.

Para los dos algoritmos presentados en esta tesis existe también la posibilidad de hacer ciertas adaptaciones y cambiar de un paradigma secuencial a paralelo. Con este cambio se podrían resolver grafos con un número mayor de nodos o reducir los tiempos de cómputo de manera significativa.

Referencias

Adenso-Diaz, B., y Laguna, M. (2006). Fine-tuning of algorithms using fractional experimental designs and local search. *Oper. Res.*, *54*(1), 99–114. doi: 10.1287/opre.1050.0243

Bhatt, S. N., y Leighton, F. T. (1984). A framework for solving VLSI graph layout problems. *J. Comput. System Sci.*, *28*(2), 300–343.

Biggs, N. L., Lloyd, E. K., y Wilson, R. J. (1976). The solution of a problem relating to the geometry of position. En *Graph Theory: 1736-1936* (pp. 3–8). transl. Clarendon Press.

Blum, C., y Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, *35*(3), 268–308.

Carrabs, F., Cerulli, R., y Gentili, M. (2009). The labeled maximum matching problem. *Computers & OR*, *36*(6), 1859–1871.

Cerulli, R., Dell'Olmo, P., Gentili, M., y Raiconi, A. (2006). Heuristic approaches for the minimum labelling hamiltonian cycle problem. *Electronic notes in discrete mathematics*, *25*, 131–138.

Chinn, P. Z., Chvátalová, J., Dewdney, A. K., y Gibbs, N. E. (1982). The bandwidth problem for graphs and matrices–a survey. *J. Graph Theory*, *6*, 223–254.

Chung, F. R. K. (1988). Labelling of graphs. *Selected Topics in Graph Theory*, *3*, 151–168.

Chvátal, V. (1970). A remark on a problem of harary. *Czechoslovak Mathematical Journal*, *20*(1), 109–111.

Cohen, D. M., Dalal, S. R., Parelius, J., y Patton, G. C. (1996). The combinatorial design approach to automatic test generation. *IEEE Softw.*, *13*(5), 83–88. doi: 10.1109/52.536462

Cook, S. A. (1971). The complexity of theorem-proving procedures. En *Proceedings of the third annual acm symposium on theory of computing* (pp. 151–158). ACM. doi: 10.1145/800157.805047

Cook, S. A. (2000). The P versus NP problem. En *Clay Mathematical Institute; The Millennium Prize Problem.*

Dasgupta, S., Papadimitriou, C. H., y Vazirani, U. V. (2008). *Algorithms*. McGraw-Hill.

Del Corso, G. M., y Manzini, G. (1999). Finding exact solutions to the bandwidth Minimization Problem. *Computing*, *62*(3), 189–203.

Deutsch, J. P. A. (1966). A short cut for certain combinational problems. *In British Joint Comput*.

Duff, I. S., Grimes, R. G., y Lewis, J. G. (1992). *Users' Guide for the Harwell-Boeing Sparse Matrix Collection (Release I).*

Galinier, P., Boujbel, Z., y Coutinho Fernandes, M. (2011). An efficient memetic algorithm for the graph partitioning problem. *Annals of Operations Research*, *191*(1), 1–22.

Gallian, J. A. (1998). A dynamic survey of graph labeling. *The Electronic Journal of Combinatorics*, *6*(1), 1–219.

Garey, M. R., y Johnson, D. S. (1989). Tutorial: hard real-time systems. En J. A. Stankovic y K. Ramamritham (Eds.), (pp. 205–219). IEEE Computer Society Press.

Gendreau, M., y Potvin, J. (2010). *Handbook of Metaheuristics*. Springer Publishing Company, Incorporated.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, *13*(5), 533–549.

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

Guerequeta, R., y Vallecillo, A. (2000). *Técnicas de diseño de algoritmos* (2.ª ed.). Universidad de Málaga.

Gunawan, A., Lau, H. C., y Lindawati. (2011). Fine-tuning algorithm parameters using the design of experiments approach. En *Proceedings of the 5th international conference on learning and intelligent optimization* (pp. 278–292). Springer-Verlag. doi: 10.1007/978-3-642-25566-3-21

Gupta, A. (2000). Improved bandwidth approximation for trees. En *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms* (pp. 788–793). Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Gurari, E., y Sudborough, I. (1984). Improved dynamic programming algorithms for bandwidth

minimization and the MinCut Linear Arrangement problem. *Journal of Algorithms*, *5*(4), 531–546.

Haasdijk, E., Smit, S. K., y Eiben, A. E. (2012). Exploratory analysis of an on-line evolutionary algorithm in simulated robots. *Evolutionary Intelligence*, *5*(4), 213–230.

Haralick, R. M., y Shapiro, L. G. (1979). The consistent labeling problem: Part 1. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *1*(2), 173–184.

Harary, F. (1969). *Graph theory*. Reading, MA: Addison-Wesley.

Hertz, A., Taillard, E., y Werra, D. D. (1995). *A tutorial on tabu search* (Inf. Téc.). EPFL, Départment de Mathétiques, MA-Ecublens, CH–1015.

Hromkovič, J., Müller, V., Sýkora, O., y Vrťo, I. (1992). On embedding interconnection networks into rings of processors. En *Parle '92 parallel architectures and languages europe* (Vol. 605, pp. 51–62). Springer Berlin / Heidelberg.

Jinjiang, Y., y Sanming, Z. (1995). Optimal labelling of unit interval graphs. *Applied Mathematics - A Journal of Chinese Universities*, *10*, 337–344.

Klerk, E. d., Eisenberg-Nagy, M., y Sotirov, R. (2011). *On Semidefinite Programming Bounds For Graph Bandwidth* (CWI Technical Report).

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, *2*(1), 83–97.

Lam, P. C. B., Shiu, W. C., y Chan, W. H. (2002). Characterization of graphs with equal bandwidth and cyclic bandwidth. *Discrete Math*, *242*(1-3), 283–289.

Landgraaf, W. A. d., Eiben, A. E., y Nannen, V. (2007). Parameter Calibration using Meta-Algorithms. En *Ieee congress on evolutionary computation* (pp. 71–78). IEEE.

Leung, J. Y. T., Vornberger, O., y Witthoff, J. D. (1984). On some variants of the bandwidth minimization problem. *SIAM Journal on Computing*, *13*(3), 650–667.

Lim, A., Rodrigues, B., y Xiao, F. (2003). Integrated genetic algorithm with hill climbing for bandwidth minimization problem. En *Proceedings of the 2003 International Conference on*

Genetic and Evolutionary Computation: Part II (pp. 1594–1595). Berlin, Heidelberg: Springer-Verlag.

Lin, Y. (1989). A level structure approach on the bandwidth problem for special graphs. *Graph Theory and Its Applications: East and West*, *576*, 334–357.

Lin, Y. (1994). The cyclic bandwidth problem. *Journal of Systems Science and Complexity*, *7*(3), 282.

Lin, Y. (1997). Minimum bandwidth problem for embedding graphs in cycles. *Networks*, *29*(3), 135–140.

Lozano, M., Duarte, A., Gortázar, F., y Martí, R. (2011). Variable neighborhood search with ejection chains for the antibandwidth problem. *Journal of Heuristics*, 1–20.

Luce, R., y Perry, A. (1949). A method of matrix analysis of group structure. *Psychometrika*, *14*(2), 95–116.

Martí, R., Campos, V., y Piñana, E. (2008). A branch and bound algorithm for the matrix bandwidth minimization. *European Journal of Operational Research*, *186*(2), 513–528.

Martí, R., Laguna, M., Glover, F., y Campos, V. (2001). Reducing the bandwidth of a sparse matrix with tabu search. *European Journal of Operational Research*, *135*(2), 450–459.

Mladenovic, N., Urosevic, D., Pérez-Brito, D., y García-González, C. G. (2010). Variable neighbourhood search for bandwidth reduction. *European Journal of Operational Research*, *200*(1), 14–27.

Nemhauser, G. L., y Wolsey, L. A. (1988). *Integer and combinatorial optimization*. New York, NY, USA: Wiley-Interscience.

Palubeckis, G., y Rubliauskas, D. (2011). A branch-and-bound algorithm for the minimum cut linear arrangement problem. *Journal of Combinatorial Optimization*, 1–24. doi: 10.1007/s10878-011-9406-2

Pintea, C., Crisan, G. C., y Chira, C. (2010). A hybrid ACO approach to the matrix bandwidth minimization problem. En *Hybrid Artificial Intelligence Systems* (pp. 405–412). Springer.

Rodriguez-Tello, E., Hao, J. K., y Torres-Jimenez, J. (2008, March). An Effective Two-Stage Simulated Annealing Algorithm for the Minimum Linear Arrangement Problem. *European Journal of Operational Research*, *185*(3), 1319–1335.

Rodriguez-Tello, E., y Torres-Jimenez, J. (2010). Memetic algorithms for constructing binary covering arrays of strength three. *Lecture Notes in Computer Science*, *5975*, 86-97.

Sorlin, S., y Solnon, C. (2005). Reactive tabu search for measuring graph similarity. En *Proceedings of the 5th IAPR international conference on Graph-Based Representations in Pattern Recognition* (pp. 172–182). Berlin, Heidelberg: Springer-Verlag.

Talbi, E. (2009). *Metaheuristics : From design to implementation* (1.ª ed.). European: John Wiley & Sons.

Ullmann, J. R. (1976). An Algorithm for Subgraph Isomorphism. *J. ACM*, *23*(1), 31–42.

Viger, F., y Latapy, M. (2005). Efficient and simple generation of random simple connected graphs with prescribed degree sequence. En *in The Eleventh International Computing and Combinatorics Conference, Aug. 2005, kumming* (pp. 440–449). Springer.

Woeginger, G. J. (2003). Exact algorithms for NP-hard problems: A survey. *Combinatorial Optimization*(23), 185–207.

Zhou, S. (2000). Bounding the bandwidths for graphs. *Theoretical Computer Science*, *249*(2), 357–368.